RESEARCH ARTICLE

Membrane Fission: A Computational Complexity Perspective

LUIS F. MACÍAS-RAMOS,[1] BOSHENG SONG,[2] LUIS VALENCIA-CABRERA,[1] LINQIANG PAN,[2] AND MARIO J. PÉREZ-JIMÉNEZ[1]

[1]*Research Group of Natural Computing, Department of Computer Science and Artificial Intelligence, University of Seville, Sevilla, 41012, Spain; and* [2]*Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China*

Received 12 February 2015; revised 6 April 2015; accepted 7 April 2015

Membrane fission is a process by which a biological membrane is split into two new ones in the manner that the content of the initial membrane is separated and distributed between the new membranes. Inspired by this biological phenomenon, membrane separation rules were considered in membrane computing. In this work, we investigate cell-like P systems with symport/antiport rules and membrane separation rules from a computational complexity perspective. Specifically, we establish a limit on the efficiency of such P systems which use communication rules of length at most two, and we prove the computational efficiency of this kind of models when using communication rules of length at most three. Hence, a sharp borderline between tractability and ***NP****–hardness is provided in terms of the length of communication rules.* © *2015 Wiley Periodicals, Inc. Complexity 21: 321–334, 2016*

Key Words: bioinspired computing; membrane computing; membrane fission; tractability border

1. INTRODUCTION

Let us recall that a lipid membrane, called a cell membrane or plasma membrane, separates the inside of a cell from its environment. Furthermore, membrane compartments (e.g., mitochondria, endosomes, endoplasmic reticulum, and Golgi complex) inside eukaryotic cells are surrounded by lipid membranes allowing the compartments to have an identity. In both cases, the lipid membranes serve as concentration barriers allowing to incorporate material from its environment in the case of a cell, or exchange material between compartments (from a donor membrane to an acceptor membrane). This is done by means of a simple three-step process whose last step is *membrane fission* consisting in splitting it into two new membranes [1].

The biological phenomenon of membrane fission process was incorporated in *membrane computing* [2] as a new kind of computational rules, called *membrane separation rules*, in the framework of polarizationless P systems with active membranes [3]. These rules were associated

Correspondence to: Linqiang Pan, E-mail: lqpan@mail.hust.edu.cn

 Vol. 21 No. 6
DOI 10.1002/cplx.21691
Published online 27 April 2015 in Wiley Online Library (wileyonlinelibrary.com)

with different subsets of the working alphabet. In [4], a new definition of separation rules in the framework of P systems with active membranes was introduced, where there exists a distinguished partition of the working alphabet into two subsets such that each separation rule is associated with that predefined partition. By applying such a rule, two new membranes are created, the object triggering it is consumed and the remaining objects are *distributed* among the newly created membranes. A uniform and polynomial time solution to SAT problem by a family of P systems with active membranes and membrane separation rules was given in [3].

Networks of membranes, which compute by communication only in the form of symport/antiport rules, were considered in [5]. These networks aim to abstract the biological phenomenon of transmembrane transport of couples of chemical substances, in the same or opposite directions. Such rules are used both for communication with the environment and for direct communication between different membranes. It is worth noting that in such a system the environment plays an active role because not only objects can be sent outside the system but also objects can be brought into the system from the environment.

With respect to the tissue-like computation models, from the seminal definitions of tissue P systems [6,7], one of the most interesting variants of tissue P systems was presented in [8]. In that paper, the definition of tissue P systems with symport/antiport rules is combined with the one of P systems with active membranes, yielding *tissue P systems with cell division*. Membrane fission was introduced into tissue P systems with symport/antiport rules through *cell separation* rules yielding *tissue P systems with cell separation* [9]. The computational efficiency of these systems was investigated and a tractability border in terms of the length of communication rules was obtained: passing from 1 to 8 amounts to passing from tractability to **NP**–hardness [9]. Furthermore, in [10], that frontier was refined in an optimal sense with respect to communication rules length (passing from 2 to 3).

Cell-like P systems with symport/antiport rules were introduced in [11] and their computational completeness (five membranes are enough if at most two objects are used in the rules) was shown. In this work, we investigate membrane separation rules in this kind of cell-like P systems from a computational complexity point of view.

The article is organized as follows. Section 2 briefly describes some preliminaries to make the article self-contained. In section 3, limits on computational efficiency of P systems with symport/antiport rules of length at most two and membrane separation are established. Section 4 provides a uniform and polynomial time solution to the SAT problem by a family of P systems with symport/antiport rules of length at most three and membrane separation. Finally, conclusions are drawn.

2. PRELIMINARIES

2.1. Multisets Over an Alphabet

An *alphabet* Γ is a nonempty set and their elements are called *symbols*. A *string* u over Γ is an ordered finite sequence of symbols. The *length* of a string u, denoted by $|u|$, is the number of occurrences of symbols that it contains. The empty string (with length 0) is denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*. A *language* over Γ is a subset of Γ^*.

A *multiset* over an alphabet Γ is an ordered pair (Γ, f), where f is a mapping from Γ onto the set of natural numbers $\mathbb{N}$. The *support* of a multiset $m=(\Gamma, f)$ is defined as $\mathrm{supp}(m)=\{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite (resp., empty) if its support is a finite (resp., empty) set. We denote by $\varnothing$ the empty multiset. Let $m_1=(\Gamma, f_1)$, $m_2=(\Gamma, f_2)$ be multisets over Γ, then the union of m_1 and m_2, denoted by m_1+m_2, is the multiset (Γ, g), where $g(x)=f_1(x)+f_2(x)$ for each $x \in \Gamma$. We say that m_1 is contained in m_2, and we denote it by $m_1 \subseteq m_2$, if $f_1(x) \leq f_2(x)$ for each $x \in \Gamma$. The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is the multiset (Γ, g), where $g(x)=f_1(x)-f_2(x)$ if $f_1(x) \geq f_2(x)$, and $g(x) = 0$ otherwise.

A *rooted tree* is a connected, acyclic, undirected graph in which one of the vertices (called *the root of the tree*) is distinguished from the others.

Given a node x different from the root in a rooted tree, if the last edge on the (unique) path from the root to the node x is $\{x, y\}$ (so $x \neq y$), then y is the *parent* of node x and x is a *child* of node y, which is denoted by $y=p(x)$ and $x \in ch(y)$. The root is the only node in the tree with no parent. A node with no children is called a *leaf* (see [12] for details).

Let us recall that the *pair function* $\langle n, m \rangle=((n+m)(n+m+1)/2)+n$ is a polynomial-time computable function, which is also a primitive recursive and bijective function from IN $\times$ IN to IN.

2.2. P systems with Symport/Antiport Rules and Membrane Separation Rules

Next, we introduce an abstraction of membrane fission process in the framework of cell-like P systems with communication rules of the kind symport/antiport rules, through *membrane separation* rules.

Definition 2.1

A P system with symport/antiport rules and membrane separation rules (SAS P system, for short), of degree $q \geq 1$, *is a tuple*

$$\Pi=(\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{\mathrm{in}}, i_{\mathrm{out}})$$

where:

1. Γ *is a finite alphabet;*
2. $\{\Gamma_0, \Gamma_1\}$ is a partition of Γ, that is, $\Gamma=\Gamma_0 \cup \Gamma_1$, $\Gamma_0, \Gamma_1 \neq \varnothing$, $\Gamma_0 \cap \Gamma_1=\varnothing$;

DOI 10.1002/cplx

3. $\mathcal{E} \subsetneq \Gamma$;
4. Σ is an (input) alphabet strictly contained in Γ such that $\mathcal{E} \subseteq \Gamma \setminus \Sigma$;
5. μ is a membrane structure (a rooted tree) whose nodes are injectively labelled with $1, \ldots, q$ (the root of the tree is labelled with 1);
6. $\mathcal{M}_1, \ldots, \mathcal{M}_q$ are finite multisets over $\Gamma \setminus \Sigma$;
7. $\mathcal{R}_i$, $1 \leq i \leq q$, are finite sets of rules over Γ of the following forms:
   (*a*) Communication rules:
      (*a*.1) *Symport rules*: (u, out) or (u, in), where u is a finite multiset over Γ such that $|u| > 0$;
      (*a*.2) *Antiport rules*: $(u, \text{out}; v, \text{in})$, where u, v are finite multisets over Γ such that $|u| > 0$ and $|v| > 0$;
   (b) Separation rules: $[a]_i \to [\Gamma_0]_i [\Gamma_1]_i$, where $a \in \Gamma$, $i \in \{2, \ldots, q\}$, and i is the label of a leaf of the tree;
8. $i_{\text{in}} \in \{1, \ldots, q\}$ and $i_{\text{out}} = 0$;

An SAS P system $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{\text{in}}, i_{\text{out}})$ of degree q can be viewed as a set of q membranes, labelled with $1, \ldots, q$, arranged in a hierarchical structure μ given by a rooted tree whose root is called the *skin membrane*, labelled by 1, such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the finite multisets of *objects* (symbols of Γ) initially placed in the q membranes of the system; (b) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them are available in an arbitrary number of copies; (c) $\mathcal{R}_1, \ldots, \mathcal{R}_q$ are finite sets of rules over Γ associated with the membranes labelled with $1, \ldots, q$, respectively; (d) $i_{\text{in}} \in \{1, \ldots, q\}$ is a label that represents a distinguished *membrane* called the *input membrane*, and $i_{\text{out}} = 0$ is a label that represents the environment of the system where the system outputs the computation results into. We use the term *region* i ($0 \leq i \leq q$) to refer to membrane i in the case $1 \leq i \leq q$ and to refer to the environment in the case $i = 0$. The length of rule (u, out) or (u, in) (resp., $(u, \text{out}; v, \text{in})$) is defined as $|u|$ (resp., $|u| + |v|$). For each membrane $i \in \{2, \ldots, q\}$ (different from the skin membrane), we denote by $p(i)$, the parent of membrane i in the rooted tree μ. We define $p(1) = 0$, that is, by convention the "parent" of the skin membrane is the environment. The leaves of the rooted tree are called *elementary membranes*.

An *instantaneous description* or a *configuration* $\mathcal{C}_t$ at an instant t of an SAS P system is described by the following: (a) the membrane structure at instant t; (b) all multisets of objects over Γ associated with all the membranes present in the system; and (c) the multiset of objects over $\Gamma - \mathcal{E}$ associated with the environment at that moment. Recall that initially there are infinite copies of objects from $\mathcal{E}$ in the environment, and hence this set is not properly changed along the computation. The *initial configuration* of the system is $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_q; \varnothing)$.

A symport rule $(u, \text{out}) \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t if membrane i is in $\mathcal{C}_t$ and multiset u is contained in the multiset associated with such membrane. When applying a rule $(u, \text{out}) \in \mathcal{R}_i$, the objects specified by u are sent out of membrane i into the region immediately outside (i.e., the parent $p(i)$ of i), which can be the environment in the case of the skin membrane (i.e., $i = 1$). A symport rule $(u, \text{in}) \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t if membrane i is in $\mathcal{C}_t$ and multiset u is contained in the multiset associated with the parent of i. When applying a rule $(u, \text{in}) \in \mathcal{R}_i$, the multiset of objects u goes out from the parent membrane of i and enters into the region defined by membrane i.

An antiport rule $(u, \text{out}; v, \text{in}) \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t if membrane i is in $\mathcal{C}_t$ and multiset u is contained in membrane i, and multiset v is contained in the parent of i. When applying a rule $(u, \text{out}; v, \text{in}) \in \mathcal{R}_i$, the objects specified by u are sent out of membrane i into the parent of i and, at the same time, bringing the objects specified by v into membrane i.

A separation rule $[a]_i \to [\Gamma_0]_i [\Gamma_1]_i \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists an elementary membrane labelled by i in $\mathcal{C}_t$, different from the skin membrane, such that it contains object a. When applying a separation rule $[a]_i \to [\Gamma_0]_i [\Gamma_1]_i \in \mathcal{R}_i$ to a membrane labelled by i in a configuration $\mathcal{C}_t$, in reaction with object a, membrane i is separated into two membranes with the same label; at the same time, object a is consumed; the objects from Γ_0 already existing in membrane i of $\mathcal{C}_t$ are placed in the first membrane, while those from Γ_1 are placed in the second membrane. In this way, several membranes with the same label i can be present in the new membrane structure μ' of the system: an arc $(p(i), i)$ is established for each of the two new membranes with label $i \neq 1$.

With respect to the semantics of such P systems, the rules are applied in a nondeterministic maximally parallel manner, with the following important remark: when a membrane i is separated, the membrane separation rule is the only one from $\mathcal{R}_i$ which is applied for that membrane at that step. The new membranes resulting from separation could participate in the interaction with other membranes or the environment by means of communication rules at the next step—providing that they are not separated once again. The label of a membrane precisely identifies the rules which can be applied to it.

Let us consider an SAS P system Π. We say that configuration $\mathcal{C}_t$ yields configuration $\mathcal{C}_{t+1}$ in one *transition step*, denoted by $\mathcal{C}_t \Rightarrow_\Pi \mathcal{C}_{t+1}$, if the system can pass from $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$ by applying the rules following the previous remarks. A *computation* of Π is a (finite or infinite) sequence of configurations such that: (a) the first term is the initial

DOI 10.1002/cplx

configuration of the system; (b) for each $n \geq 2$, the nth configuration of the sequence is obtained from the previous configuration in one transition step; and (c) if the sequence is finite (called *halting computation*) then the last term is a *halting configuration* (a configuration where no rule of the system is applicable to it).

For each finite multiset w over the input alphabet Σ, a *computation* of an SAS P system $\Pi=(\Gamma,\Gamma_0,\Gamma_1,\mathcal{E},\Sigma,\mu,\mathcal{M}_1,\ldots,\mathcal{M}_q,\mathcal{R}_1,\ldots,\mathcal{R}_q,i_{\text{in}},i_{\text{out}})$ with input multiset w starts from the configuration of the form $(\mu,\mathcal{M}_1,\ldots,\mathcal{M}_{i_{\text{in}}}+w,\ldots,\mathcal{M}_q,\varnothing)$, where w is added to the content of the input membrane i_{in}. That is, in Π, we have an initial configuration associated with such a w. We denote by $\Pi+w$, the P system Π with input multiset w.

All the computations start from an initial configuration and proceed as stated above; only a halting computation gives a result, which is encoded by the objects present in the output region i_{out} associated with the halting configuration. If $\mathcal{C}=\{\mathcal{C}_t\}_{t<r+1}$ of Π ($r \in \mathbb{N}$) is a halting computation, then the *length* of $\mathcal{C}$, denoted by $|\mathcal{C}|$, is r, that is, $|\mathcal{C}|$ is the number of noninitial configurations which appear in the finite sequence $\mathcal{C}$. For each i ($1 \leq i \leq q$), we denote by $\mathcal{C}_t(i)$, the finite multiset of objects over Γ contained in all membranes labelled by i (by applying separation rules, different membranes with the same label can be created) at configuration $\mathcal{C}_t$. We also denote by $\mathcal{C}_t(0)$, the finite multiset of objects over $\Gamma \setminus \mathcal{E}$ contained in the environment at configuration $\mathcal{C}_t$. Finally, we denote by $\mathcal{C}_t^*$, the finite multiset $\mathcal{C}_t(0)+\mathcal{C}_t(1)+\ldots+\mathcal{C}_t(q)$.

2.3. Recognizer P Systems with SAS P system

Recognizer P systems were introduced in [13], and they provide a natural framework to solve decision problems by means of computational devices in membrane computing (i.e., P systems).

Definition 2.2

A recognizer P system with SAS P system, of degree $q \geq 1$, is a tuple

$$\Pi=(\Gamma,\Gamma_0,\Gamma_1,\mathcal{E},\Sigma,\mu,\mathcal{M}_1,\ldots,\mathcal{M}_q,\mathcal{R}_1,\ldots,\mathcal{R}_q,i_{\text{in}},i_{\text{out}})$$

where:

1. $\Pi=(\Gamma,\Gamma_0,\Gamma_1,\mathcal{E},\Sigma,\mu,\mathcal{M}_1,\ldots,\mathcal{M}_q,\mathcal{R}_1,\ldots,\mathcal{R}_q,i_{\text{in}},i_{\text{out}})$ *is an SAS P system*;
2. *alphabet Γ has two distinguished symbols yes and no;*
3. *$\mathcal{M}_1,\ldots,\mathcal{M}_q$ are finite multisets over $\Gamma \setminus \Sigma$ such that at least one copy of yes or no is present in some of them*;
4. *all computations halt;*
5. *if $\mathcal{C}$ is a computation of Π, then either symbol yes or symbol no (but not both) must have been released into the environment, and only at the last step of the computation.*

Let us notice that if a recognizer P system has a symport rule of the type $(u,\text{in}) \in \mathcal{R}_1$ then the multiset u must contain some object from $\Gamma \setminus \mathcal{E}$ because on the contrary, it might exist nonhalting computations of Π.

We say that a computation $\mathcal{C}$ of a recognizer P system is an *accepting computation* (respectively, *rejecting computation*) if object yes (respectively, object no) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$, and neither object yes nor no appears in the environment associated with any nonhalting configuration of $\mathcal{C}$.

For each natural number $k \geq 1$, we denote by $\mathbf{CSC}(k)$, the class of recognizer SAS P systems such that the communication rules allowed have length at most k. Next, we define the concept of efficient solvability by means of a family of such P systems (see [14] for more details).

Definition 2.3

A decision problem $X=(I_X,\theta_X)$ is solvable in polynomial time by a family $\Pi=\{\Pi(n)|n \in \mathbb{N}\}$ of recognizer P systems with SAS P system if the following holds:

- *the family Π is polynomially uniform by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$;*
- *there exists a pair (cod, s) of polynomial-time computable functions over I_X such that*:
  - for each instance $u \in I_X$, $s(u)$ is a natural number and $\text{cod}(u)$ is an input multiset of the system $\Pi(s(u))$;
  - for each $n \in \mathbb{N}$, $s^{-1}(n)$ is a finite set;
  - the family Π is polynomially bounded with regard to (X, cod, s), that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))+\text{cod}(u)$ is halting and it performs at most $p(|u|)$ steps;
  - the family Π is sound with regard to (X, cod, s), that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))+\text{cod}(u)$, then $\theta_X(u)=1$;
  - the family Π is complete with regard to (X, cod, s), that is, for each $u \in I_X$, if $\theta_X(u)=1$, then every computation of $\Pi(s(u))+\text{cod}(u)$ is an accepting one.

According to Definition 2.3, we say that the family Π provides a *uniform solution* to the decision problem X. We also say that ordered pair (cod, s) is a polynomial encoding from X in Π and s is the size mapping associated with that solution. It is worth pointing out that for each instance $u \in I_X$, the P system $\Pi(s(u))+\text{cod}(u)$ is *confluent*, in the sense that all

DOI 10.1002/cplx

possible computations of the system must give the same answer.

If **R** is a class of recognizer SAS P systems, then we denote by $\mathbf{PMC_R}$, the set of all decision problems which can be solved in polynomial time (and in a uniform way) by means of recognizer SAS P systems from **R**. The class $\mathbf{PMC_R}$ is closed under complement and polynomial-time reductions (see [14] for details). Besides, we have $\mathbf{P} \subseteq \mathbf{PMC_R}$. Indeed, if $X \in \mathbf{P}$ then we consider the family $\Pi=\{\Pi(n)|n \in \mathbb{N}\}$, where $\Pi(n)=\Pi(0)$, for each $n \in$ IN, and $\Pi(0)$ is a P system from **R** of degree 1 containing only two rules (yes, out) and (no, out). Let us consider the polynomial encoding from X in Π defined as follows: (a) $s(u) = 0$, for each $u \in I_X$; and (b) $\text{cod}(u)=yes$ if $\theta_X(u)=1$ and $\text{cod}(u)=no$ if $\theta_X(u)=0$. Then, the family Π solves X according to Definition 2.3.

3. THE LIMITATION ON THE EFFICIENCY OF CSC(2)

3.1. Representation of P Systems from CSC(2)

Let $\Pi=(\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{\text{in}}, i_{\text{out}})$ be a recognizer SAS P system of degree $q \geq 1$ from **CSC**(2). To identify the membranes created by the application of a separation rule, we modify the labels of the new membranes in the following recursive manner:

- The label of a membrane will be a pair (i, σ), where $1 \leq i \leq q$ and σ is a string over {0, 1}. At the initial configuration, the labels of the membranes are $(1, \lambda)$, $\ldots, (q, \lambda)$.
- If a separation rule is applied to a membrane labelled by (i, σ), then the newly created membranes will be labelled by $(i, \sigma 0)$ and $(i, \sigma 1)$, respectively. Membrane $(i, \sigma 0)$ will only contain the objects of membrane (i, σ) which belong to Γ_0, and membrane $(i, \sigma 1)$ will only contain the objects of membrane (i, σ) which belong to Γ_1. The skin membrane cannot be separated, so the label of the skin membrane is not changed along any computation, remaining $(1, \lambda)$. Note that we can consider a lexicographical order over the set of labels of cells in the system along any computation.

If a membrane labelled by (i, σ) is engaged by a communication rule, then, after the application of the rule, the membrane keeps its label.

A configuration at an instant t of a P system from **CSC**(2) is described by the current membrane structure, the multisets of objects over Γ contained in each membrane and the multiset of objects over $\Gamma \setminus \mathcal{E}$ from the environment. Hence, a configuration of Π can be described by a multiset of labelled objects:

$$\{(a, i, \sigma)|a \in \Gamma \cup \{\lambda\}, 1 \leq i \leq q, \sigma \in \{0,1\}^*\} \cup \{(a, 0)|a \in \Gamma \setminus \mathcal{E}\}.$$

Let us notice that the number of labels we need to identify all membranes appearing along any computation of a P system from **CSC**(2) is quadratic in the size of the initial configuration of the system and the length of the computation.

Let $r=(ab, \text{out}) \in \mathcal{R}_i$, $2 \leq i \leq q$, be a symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (i, \sigma), (p(i), \tau))$, the multiset of objects $(a, i, \sigma)^n (b, i, \sigma)^n$, and we denote by $n \cdot \text{RHS}(r, (i, \sigma), (p(i), \tau))$, the multiset $(a, p(i), \tau)^n (b, p(i), \tau)^n$. In a similar way, $n \cdot \text{LHS}(r, (i, \sigma), (p(i), \tau))$ and $n \cdot \text{RHS}(r, (i, \sigma), (p(i), \tau))$ are defined when r is of the form $(a, \text{out}) \in \mathcal{R}_i$.

Let $r=(ab, \text{out}) \in \mathcal{R}_1$ be a symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (1, \lambda), 0)$, the multiset of objects $(a, 1, \lambda)^n (b, 1, \lambda)^n$. We denote by $n \cdot \text{RHS}(r, (1, \lambda), 0)$, the multiset of objects

$$\begin{cases} (a,0)^n (b,0)^n, & \text{if } a, b \in \Gamma \setminus \mathcal{E}; \\ (a,0)^n, & \text{if } a \in \Gamma \setminus \mathcal{E} \text{ and } b \in \mathcal{E}; \\ (b,0)^n, & \text{if } b \in \Gamma \setminus \mathcal{E} \text{ and } a \in \mathcal{E}; \\ \varnothing, & \text{if } a, b \in \mathcal{E}. \end{cases}$$

In a similar way, $n \cdot \text{LHS}(r, (1, \lambda), 0)$ and $n \cdot \text{RHS}(r, (1, \lambda), 0)$ are defined when r is of the form $(a, \text{out}) \in \mathcal{R}_1$.

Let $r=(ab, \text{in}) \in \mathcal{R}_i$, $2 \leq i \leq q$, be a symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (i, \sigma), (p(i), \tau))$ and $n \cdot \text{RHS}(r, (i, \sigma), (p(i), \tau))$, the multiset of objects $(a, p(i), \tau)^n (b, p(i), \tau)^n$ and $(a, i, \sigma)^n (b, i, \sigma)^n$, respectively. In a similar way, $n \cdot \text{LHS}(r, (i, \sigma), (p(i), \tau))$ and $n \cdot \text{RHS}(r, (i, \sigma), (p(i), \tau))$ are defined when r is of the form $(a, \text{in}) \in \mathcal{R}_i$.

Let $r=(ab, \text{in}) \in \mathcal{R}_1$ be a symport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (1, \lambda), 0)$, the multiset of objects

$$\begin{cases} (a,0)^n (b,0)^n, & \text{if } a, b \in \Gamma \setminus \mathcal{E}; \\ (a,0)^n, & \text{if } a \in \Gamma \setminus \mathcal{E} \text{ and } b \in \mathcal{E}; \\ (b,0)^n, & \text{if } b \in \Gamma \setminus \mathcal{E} \text{ and } a \in \mathcal{E}; \\ \varnothing, & \text{if } a, b \in \mathcal{E}. \end{cases}$$

We denote by $n \cdot \text{RHS}(r, (1, \lambda), 0)$, the multiset of objects $(a, 1, \lambda)^n (b, 1, \lambda)^n$. In a similar way, $n \cdot \text{LHS}(r, (1, \lambda), 0)$ and $n \cdot \text{RHS}(r, (1, \lambda), 0)$ are defined when r is of the form $(a, \text{in}) \in \mathcal{R}_1$.

Let $r=(a, \text{out}; b, \text{in}) \in \mathcal{R}_i$, $2 \leq i \leq q$, be an antiport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (i, \sigma), (p(i), \tau))$, the multiset of objects $(a, i, \sigma)^n (b, p(i), \tau)^n$. Similarly, we denote by $n \cdot \text{RHS}(r, (i, \sigma), (p(i), \tau))$, the multiset of objects $(a, p(i), \tau)^n (b, i, \sigma)^n$.

Let $r=(a, \text{out}; b, \text{in}) \in \mathcal{R}_1$ be an antiport rule of Π and $n \in \mathbb{N}$. We denote by $n \cdot \text{LHS}(r, (1, \lambda), 0)$, the multiset of objects

$$\begin{cases} (a,1,\lambda)^n (b,0)^n, & \text{if } b \in \Gamma \setminus \mathcal{E}; \\ (a,1,\lambda)^n, & \text{if } b \in \mathcal{E}. \end{cases}$$

Similarly, we denote by $n \cdot \text{RHS}(r, (1, \lambda), 0)$, the multiset of objects

DOI 10.1002/cplx

$$\begin{cases} (a,0)^n(b,1,\lambda)^n, & \text{if } a \in \Gamma \setminus \mathcal{E}; \\ (b,1,\lambda)^n, & \text{if } a \in \mathcal{E}. \end{cases}$$

If $\mathcal{C}_t$ is a configuration of Π, then we denote by $\mathcal{C}_t + \{(x,i,\sigma)/\sigma'\}$, the multiset obtained by replacing in $\mathcal{C}_t$ every occurrence of (x,i,σ) by (x,i,σ'). Besides, $\mathcal{C}_t + m$ (resp., $\mathcal{C}_t \setminus m$) is used to denote that a multiset m of labelled objects is added to (resp., removed from) the configuration.

3.2. P systems from CSC(2) Characterize Classical Complexity Class P

To show that only tractable problems can be solved efficiently using families of SAS P systems from **CSC**(2), we first prove a technical result concerning recognizer P systems from **CSC**(2).

Lemma 3.1

Let $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{\text{in}}, i_{\text{out}})$ *be a recognizer P system of degree* $q \geq 1$ *from* **CSC**(2). *Let* $M = |\mathcal{M}_1 + \ldots + \mathcal{M}_q|$ *and let* $\mathcal{C} = \{\mathcal{C}_0, \ldots, \mathcal{C}_r\}$ *be a computation of* Π. *Then, we have*

1. $|\mathcal{C}_0^*| = M$, and for each t, $0 \leq t < r$, $\mathcal{C}_{t+1}^* \cap (\Gamma \setminus \mathcal{E}) \subseteq \mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})$;
2. for each t, $0 \leq t \leq r$, $\mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E}) \subseteq (\mathcal{M}_1 + \ldots + \mathcal{M}_q) \cap (\Gamma \setminus \mathcal{E})$, and $|\mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})| \leq M$;
3. for each t, $0 \leq t < r$, $|\mathcal{C}_{t+1}^*| \leq |\mathcal{C}_t^*| + M$;
4. for each t, $0 \leq t \leq r$, $|\mathcal{C}_t^*| \leq M \cdot (1+t)$;
5. the number of created membranes along the computation $\mathcal{C}$ by the application of membrane separation rules is bounded by $2M \cdot (1+r)$.

Proof

(1) First, let us notice that

$$|\mathcal{C}_0^*| = |\mathcal{C}_0(0) + \mathcal{C}_0(1) + \ldots + \mathcal{C}_0(q)| = |\mathcal{M}_1 + \ldots + \mathcal{M}_q| = M.$$

Let x be an arbitrary object of the multiset

$$\mathcal{C}_{t+1}^* \cap (\Gamma \setminus \mathcal{E}) = (\mathcal{C}_{t+1}(0) + \mathcal{C}_{t+1}(1) + \ldots + \mathcal{C}_{t+1}(q)) \cap (\Gamma \setminus \mathcal{E}).$$

Bear in mind that membrane separation rules neither replicate objects nor produce new objects. Then, at the $(t+1)$ th step, we have the following two cases:

- x has not been produced by the application of any rule, and then $x \in \mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})$;
- x has been obtained by applying a communication rule.
  - If x appears in a membrane with labels j, $2 \leq j \leq q$, then the result holds obviously.
  - If x appears in the skin membrane, we have the following four types of rules: (i) $(x, \text{in}) \in \mathcal{R}_1$, then $x \in \mathcal{C}_t(0) \cap (\Gamma \setminus \mathcal{E})$; (ii) $(x, \text{out}) \in \mathcal{R}_{\text{ch}(1)}$, then $x \in \mathcal{C}_t(\text{ch}(1)) \cap (\Gamma \setminus \mathcal{E})$; (iii) $(x', \text{out}; x, \text{in}) \in \mathcal{R}_1$, then $x \in \mathcal{C}_t(0) \cap (\Gamma \setminus \mathcal{E})$; and (iv) $(x, \text{out}; x', \text{in}) \in \mathcal{R}_{\text{ch}(1)}$, then $x \in \mathcal{C}_t(\text{ch}(1)) \cap (\Gamma \setminus \mathcal{E})$. Hence, in all of the above four cases, $x \in \mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})$.
  - If x appears in the environment, we have the following two types of rules: (i) $(x, \text{out}) \in \mathcal{R}_1$, then $x \in \mathcal{C}_t(1) \cap (\Gamma \setminus \mathcal{E})$ and (ii) $(x, \text{out}; x', \text{in}) \in \mathcal{R}_1$, then $x \in \mathcal{C}_t(1) \cap (\Gamma \setminus \mathcal{E})$. Hence, in both cases, $x \in \mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})$.

Hence, $\mathcal{C}_{t+1}^* \cap (\Gamma \setminus \mathcal{E}) \subseteq \mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})$.

(2) By induction on t. Let us start analyzing the basic case $t = 0$. The result is trivial because

$$\begin{aligned} \mathcal{C}_0^* \cap (\Gamma \setminus \mathcal{E}) &= (\mathcal{C}_0(0) + \mathcal{C}_0(1) + \ldots + \mathcal{C}_0(q)) \cap (\Gamma \setminus \mathcal{E}) \\ &= (\mathcal{M}_1 + \ldots + \mathcal{M}_q) \cap (\Gamma \setminus \mathcal{E}). \end{aligned}$$

By induction hypothesis, let us suppose the result holds for t, $0 \leq t < r$. Let us see that the result also holds for $t+1$. We have

$$\mathcal{C}_{t+1}^* \cap (\Gamma \setminus \mathcal{E}) \overset{(1)}{\subseteq} \mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E}) \overset{\text{h.i.}}{\subseteq} (\mathcal{M}_1 + \ldots + \mathcal{M}_q) \cap (\Gamma \setminus \mathcal{E}).$$

Hence, $|\mathcal{C}_t^* \cap (\Gamma \setminus \mathcal{E})| \leq |(\mathcal{M}_1 + \ldots + \mathcal{M}_q) \cap (\Gamma \setminus \mathcal{E})| \leq M$.

(3) For $0 \leq t < r$, from configuration $\mathcal{C}_t$ to configuration $\mathcal{C}_{t+1}$, only the skin membrane (labelled by 1) can communicate objects with the environment. Thus, all the objects in membranes $2, \ldots, q$ at configuration $\mathcal{C}_t$ are still placed in membranes $1, 2, \ldots, q$ at configuration $\mathcal{C}_{t+1}$. Let us see what are the objects evolved in membrane 1 from configuration $\mathcal{C}_t$ to configuration $\mathcal{C}_{t+1}$. In what follows, we consider additional objects introduced in $\mathcal{C}_{t+1}$ from $\mathcal{C}_t$ in membrane 1.

(i) If rules of type (u, out), $u \in \Gamma^+$ and $1 \leq |u| \leq 2$, are used, then $|\mathcal{C}_{t+1}^*| \leq |\mathcal{C}_t^*|$.

(ii) If rules of type (u, in), $u \in \Gamma^+$ and $1 \leq |u| \leq 2$, are used, then an additional object is added in $\mathcal{C}_{t+1}$ from $\mathcal{C}_t$ only using a rule of type (ab, in), with $a \in \Gamma \setminus \mathcal{E}, b \in \mathcal{E}, a \in \mathcal{C}_t(0)$ or $b \in \Gamma \setminus \mathcal{E}, a \in \mathcal{E}, b \in \mathcal{C}_t(0)$.

(iii) If rules of type $(a, \text{out}; b, \text{in})$, $a \in \Gamma$ and $b \in \Gamma$, are used, then an additional object is added in $\mathcal{C}_{t+1}$ from $\mathcal{C}_t$ only using a rule of type $(a, \text{out}; b, \text{in})$, with $a \in \mathcal{C}_t(1) \cap (\Gamma \setminus \mathcal{E})$, $b \in \mathcal{E}$.

Thus, by combining the cases (i), (ii), and (iii) with the result (2), it is easy to deduce that the total number of additional objects added in $\mathcal{C}_{t+1}$ from $\mathcal{C}_t$ is bounded by M. Hence, for each t, $0 \leq t < r$, $|\mathcal{C}_{t+1}^*| \leq |\mathcal{C}_t^*| + M$.

(4) By induction on t. Let us start analyzing the basic case $t = 0$. The result is trivial because of $|\mathcal{C}_0^*| = M$. By induction hypothesis, let us suppose the result holds for t, $0 \leq t < r$. Then,

DOI 10.1002/cplx

$$\mathcal{C}^*_{t+1} \overset{(3)}{\leq} |\mathcal{C}^*_t| + M \overset{\text{h.i.}}{\leq} M \cdot (1+t) + M = M \cdot (1+(t+1)).$$

Hence, the result is also true for $t+1$.

(5) According to the fact that the application of a membrane separation rule consumes an object and produces two new membranes, result (5) can be obtained from (4) easily. ■

In what follows, we present a deterministic algorithm $\mathcal{A}$ working in polynomial time that receives as input a P system Π from **CSC**(2) and an input multiset m of Π, and reproduces the behavior of a computation of $\Pi+m$. In particular, if Π is confluent, then the algorithm $\mathcal{A}$ will provide the same answer of the system Π. The pseudocode of the algorithm $\mathcal{A}$ is described as follows:

Input: A P system Π from **CSC**(2) and an input multiset m
  Initialization phase: $\mathcal{C}_0$ is the initial configuration of $\Pi + m$
  $t \leftarrow 0$
  while $\mathcal{C}_t$ is a nonhalting configuration **do**
    Selection phase: Input $\mathcal{C}_t$, Output $(\mathcal{C}_t', A)$
    Execution phase: Input $(\mathcal{C}_t', A)$, Output $\mathcal{C}_{t+1}$
    $t \leftarrow t+1$
  end while
Output: *Yes* if $\Pi+m$ has an accepting computation, *No* otherwise

The algorithm $\mathcal{A}$ receives a recognizer P system

$$\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{\text{in}}, i_{\text{out}})$$

from **CSC**(2) and an input multiset m. Let $M = |\mathcal{M}_1 + \ldots + \mathcal{M}_q|$, p be a natural number such that any computation of Π performs at most p transition steps. Hence, from Lemma 3.1, we know that the number of membranes in the system along any computation is bounded by $2M(1+p)+q$.

A transition step of a recognizer P system $\Pi+m$ is performed by the selection phase and the execution phase. Specifically, the selection phase receives as input a configuration $\mathcal{C}_t$ of $\Pi+m$ at an instant t. The output of this phase is a pair $(\mathcal{C}_t', A)$, where A encodes a multiset of rules selected to be applied to $\mathcal{C}_t$, and $\mathcal{C}_t'$ is the configuration obtained from $\mathcal{C}_t$ once the labelled objects corresponding to the left-hand side of the rules from A have been consumed. The execution phase receives as input the pair $(\mathcal{C}_t', A)$ and the output of this phase is the next configuration $\mathcal{C}_{t+1}$ of $\mathcal{C}_t$, where the configuration $\mathcal{C}_{t+1}$ is obtained from $\mathcal{C}_t'$ by adding the labelled objects produced by the application of rules from A, which is the labelled objects corresponding to the right-hand side of the rules from A.

Selection Phase

Input: A configuration $\mathcal{C}_t$ of $\Pi+m$ at instant t
  $\mathcal{C}_t' \leftarrow \mathcal{C}_t$; $A \leftarrow \varnothing$; $B \leftarrow \varnothing$
  for $r=(u, \text{out}; v, \text{in}) \in \mathscr{R}_i, 2 \leq i \leq q$ according to the order chosen **do**
    for each membrane (i, σ) of $\mathcal{C}_t'$ according to the lexicographical order **do**
      $n_r \leftarrow$ maximum number of times that r is applicable to (i, σ)
      if $n_r > 0$ then
        $\mathcal{C}_t' \leftarrow \mathcal{C}_t' - n_r \cdot \text{LHS}(r, (i, \sigma), (p(i), \tau))$
        $A \leftarrow A \cup \{(r, n_r, (i, \sigma), (p(i), \tau))\}$
        $B \leftarrow B \cup \{(i, \sigma), (p(i), \tau)\}$
      end if
    end for
  end for
  for $r=(u, \text{out}; v, \text{in}) \in \mathscr{R}_1$ according to the order chosen **do**
    for membrane $(1, \lambda)$ of $\mathcal{C}_t'$ according to the lexicographical order **do**
      $n_r \leftarrow$ maximum number of times that r is applicable to $(1, \lambda)$
      if $n_r > 0$ **then**
        $\mathcal{C}_t' \leftarrow \mathcal{C}_t' - n_r \cdot \text{LHS}(r, (1, \lambda), 0)$
        $A \leftarrow A \cup \{(r, n_r, (1, \lambda), 0)\}$
      end if
    end for
  end for
  for $r=[a\,]_i \rightarrow [\Gamma_0\,]_i\,[\Gamma_1]_i \in \mathscr{R}_i \ \ (i \neq 1)$ according to the order chosen **do**
    for each $(a, i, \sigma) \in \mathcal{C}_t'$ according to the lexicographical order, and such that $(i, \sigma) \in B$ **do**
      $\mathcal{C}_t' \leftarrow \mathcal{C}_t' \setminus \{(a, i, \sigma)\}$
      $A \leftarrow A \cup \{(r, 1, (i, \sigma))\}$
    end for
  end for

This algorithm is deterministic and works in a polynomial time. Indeed, the running time of this algorithm is polynomial in the size of Π because the number of cycles of the first main loop **for** is of order $O(|\mathcal{R}| \cdot M \cdot p \cdot q)$; the number of cycles of the second main loop **for** is of order $O(|\mathcal{R}|)$; and the number of cycles of the third main loop **for** is of order $O(|\mathcal{R}| \cdot M \cdot p \cdot q \cdot |\Gamma|)$.

Execution Phase

Input: The output $(\mathcal{C}_t', A)$ of the selection phase

DOI 10.1002/cplx

for each $(r, n_r, (i,\sigma), (p(i),\tau)) \in A$ **do**
$\mathcal{C}_t' \leftarrow \mathcal{C}_t' + n_r \cdot \mathrm{RHS}(r, (i,\sigma), (p(i),\tau))$
end for
for each $(r, n_r, (1,\lambda), 0) \in A$ **do**
$\mathcal{C}_t' \leftarrow \mathcal{C}_t' + n_r \cdot \mathrm{RHS}(r, (1,\lambda), 0)$
end for
for each $(r, 1, (i,\sigma)) \in A$ **do**
$\mathcal{C}_t' \leftarrow \mathcal{C}_t' + \{(\lambda, i, \sigma)/\sigma 0\}$
$\mathcal{C}_t' \leftarrow \mathcal{C}_t' + \{(\lambda, i, \sigma 1)\}$
for each $(x, i, \sigma) \in C_t'$ according to the lexicographical order **do**
if $x \in \Gamma_0$ **then**
$\mathcal{C}_t' \leftarrow \mathcal{C}_t' + \{(x, i, \sigma)/\sigma 0\}$
Else
$\mathcal{C}_t' \leftarrow \mathcal{C}_t' + \{(x, i, \sigma)/\sigma 1\}$
end if
end for
end for
$\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t'$

This algorithm is deterministic and works in a polynomial time. Indeed, the running time of this algorithm is polynomial in the size of Π because the number of cycles of the first main loop **for** is of order $O(|\mathcal{R}| \cdot M \cdot p \cdot q)$; the number of cycles of the second main loop **for** is of order $O(|\mathcal{R}|)$; and the number of cycles of the third main loop **for** is of order $O(|\mathcal{R}| \cdot M \cdot p \cdot q \cdot |\Gamma|)$.

Theorem 3.1
$\mathbf{P} = \mathbf{PMC}_{\mathbf{CSC}(2)}$.

Proof
It suffices to show that $\mathbf{PMC}_{\mathbf{CSC}(2)} \subseteq \mathbf{P}$. Let $X \in \mathbf{PMC}_{\mathbf{CSC}(2)}$ and let $\Pi = \{\Pi(n) | n \in \mathbb{N}\}$ be a family of recognizer P systems from **CSC**(2) solving X, according to Definition 2.3. Let (cod, s) be a polynomial encoding associated with that solution. If $u \in I_X$ is an instance of the problem X, then u will be processed by the system $\Pi(s(u)) + \mathrm{cod}(u)$. Let us consider the following deterministic algorithm $\mathcal{A}'$:

Input: an instance u of the problem X
Construct the system $\Pi(s(u)) + \mathrm{cod}(u)$
Run algorithm $\mathcal{A}$ with input $\Pi(s(u)) + \mathrm{cod}(u)$
Output: Yes if $\Pi(s(u)) + \mathrm{cod}(u)$ has an accepting computation
No otherwise

The algorithm $\mathcal{A}'$ receives as input an instance u of the decision problem $X = (I_X, \theta_X)$ and works in a polynomial time with respect to the size of the input. The following assertions are equivalent:

- $\theta_X(u) = 1$, that is, the answer of problem X to instance u is affirmative.
- Every computation of $\Pi(s(u)) + \mathrm{cod}(u)$ is an accepting computation.
- The output of the algorithm $\mathcal{A}'$ with input u is Yes.

Hence, $X \in \mathbf{P}$. ■

4. AN EFFICIENT SOLUTION TO SAT IN CSC(3)

In this section, a polynomial time solution to the SAT problem by a family of recognizer P systems $\Pi = \{\Pi(t) | t \in \mathrm{IN}\}$ from **CSC**(**3**) is provided. Each system $\Pi(t)$ will process all Boolean formula φ in conjunctive normal form with n variables and m clauses, where $t = \langle n, m\rangle = ((n+m)(n+m+1)/2) + n$, provided that the appropriate input multiset $\mathrm{cod}(\varphi)$ is supplied to the system.For each $n, m \in \mathrm{IN}$, we consider the recognizer P system

$$\Pi(\langle n, m\rangle) = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{\mathrm{in}}, i_{\mathrm{out}})$$

from **CSC**(3), defined as follows:

1. Working alphabet:

$$\begin{aligned}\Gamma = \Sigma \cup \mathcal{E} \cup \{&\alpha_{i,0,k}, \alpha'_{i,0,k} | 1 \le i \le n-1, 0 \le k \le 1\} \cup \\ &\{A_1, B_1, b_1, b'_1, c_1, c'_1, v_1, q_{1,1}, \beta_0, \beta'_0, \beta''_0, \gamma_0, \gamma'_0, \gamma''_0, \gamma'''_0, f_0, \\ &yes, no\} \cup \{f'_i | 0 \le i \le 3n+2m+1\} \cup \\ &\{\rho_{i,0}, \tau_{i,0} | 1 \le i \le n\} \cup \{\delta_{j,0} | 0 \le j \le m\},\end{aligned}$$

where the input alphabet is $\Sigma = \{x_{i,j}, \bar{x}_{i,j} | 1 \le i \le n, 1 \le j \le m\}$. The alphabet of the environment is

$$\begin{aligned}\mathcal{E} = &\{\alpha_{i,j,k}, \alpha'_{i,j,k} | 1 \le i \le n-1, 1 \le j \le 3(n-1), 0 \le k \le 1\} \cup \\ &\{\beta_j, \beta'_j, \beta''_j, \gamma_j, \gamma'_j, \gamma''_j, \gamma'''_j | 1 \le j \le 3(n-1)\} \cup \\ &\{\rho_{i,j}, \tau_{i,j} | 1 \le i \le n, 1 \le j \le 3n-1\} \cup \\ &\{T_{i,j}, T'_{i,j}, F_{i,j}, F'_{i,j} | 1 \le i < j, 1 \le j \le n\} \cup \\ &\{T_{i,i}, F'_{i,i}, T_i, F_i | 1 \le i \le n\} \cup \{A_i, A'_i, B_i, B'_i | 2 \le i \le n+1\} \cup \\ &\{b_i, b'_i, c_i, c'_i | 2 \le i \le n\} \cup \{v_i | 2 \le i \le n-1\} \cup \\ &\{y_i, a_i, w_i | 1 \le i \le n-1\} \cup \{z_i | 1 \le i \le n-2\} \cup \\ &\{q_{i,j} | 1 \le i \le j, 2 \le j \le n-1\} \cup \\ &\{u_{i,j} | 1 \le i \le j, 1 \le j \le n-2\} \cup \\ &\{t_{i,j}, f_{i,j}, r_{i,j}, s_{i,j} | 1 \le i \le j, 1 \le j \le n-1\} \cup \\ &\{d_{i,j,k}, \bar{d}_{i,j,k} | 1 \le i \le n, 1 \le j \le m, 1 \le k \le n-1\} \cup \\ &\{f_r | 1 \le r \le 3n+2m\} \cup \{e_{i,j}, \bar{e}_{i,j} | 1 \le i \le n, 1 \le j \le m\} \cup \\ &\{\delta_{j,r} | 0 \le j \le m, 1 \le r \le 3n\} \cup \{E_j | 0 \le j \le m\} \cup \{S\}.\end{aligned}$$

2. The partition is $\{\Gamma_0, \Gamma_1\}$, where $\Gamma_0 = \Gamma \setminus \Gamma_1$ and

$$\begin{aligned}\Gamma_1 = &\{T'_{i,j} F'_{i,j} | 1 \le i < j, 1 \le j \le n\} \cup \{F'_{i,i} | 1 \le i \le n\} \cup \\ &\{A'_i, B'_i | 2 \le i \le n+1\}.\end{aligned}$$

3. Membrane structure: $\mu = [[\ ]_2 [\ ]_3]_1$. The input membrane is the membrane labelled by 1.

DOI 10.1002/cplx

4. Initial multisets:

$$\begin{aligned}
\mathcal{M}_1 &= \{\alpha_{i,0,k}, \alpha'_{i,0,k} | 1 \le i \le n-1, 0 \le k \le 1\} \cup \\
&\quad \{\rho_{i,0}, \tau_{i,0} | 1 \le i \le n\} \cup \\
&\quad \{\beta_0, \beta'_0, \beta''_0, \gamma_0, \gamma'_0, \gamma''_0, \gamma'''_0, c_1, c'_1, b_1, b'_1, \nu_1, q_{1,1}, f_0, yes\} \cup \\
&\quad \{\delta_{j,0} | 0 \le j \le m\} \cup \{f'_p | 1 \le p \le 3n+2m+1\}, \\
\mathcal{M}_2 &= \{A_1, B_1\}, \\
\mathcal{M}_3 &= \{f'_0, no\}.
\end{aligned}$$

5. Rules in $\mathcal{R}_1$:

- Rules to generate in membrane 1 of configuration $\mathcal{C}_{3p+1}$ (p=0, 1, ..., n−1), the objects $T^{2^{p-1}}_{i,p+1}, T'_{i,p+1}2^{p-1}$, $F^{2^{p-1}}_{i,p+1}, F'_{i,p+1}2^{p-1}$:

$(\alpha_{i,0,k}, \text{out}; \alpha_{i,1,k}, \text{in}), 1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha'_{i,0,k}, \text{out}; \alpha'_{i,1,k}, \text{in}), 1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha_{i,1,k}, \text{out}; \alpha_{i,2,k}, \text{in}), 1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha'_{i,1,k}, \text{out}; \alpha'_{i,2,k}, \text{in}), 1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha_{i,2,k}, \text{out}; \alpha_{i,3,k}, \text{in}), 1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha'_{i,2,k}, \text{out}; \alpha'_{i,3,k}, \text{in}), 1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha_{i,3p,k}, \text{out}; \alpha_{i,3p+1,k}\Delta^k_{i,p+1}, \text{in}),$
$1 \le i \le p, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha'_{i,3p,k}, \text{out}; \alpha'_{i,3p+1,k}\Delta_{i,p+1}{}'k, \text{in}),$
$1 \le i \le p, 1 \le p \le n-2, 0 \le k \le 1$

$(\alpha_{i,3p,k}, \text{out}; \alpha_{i,3p+1,k}, \text{in}),$
$p+1 \le i \le n-1, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha'_{i,3p,k}, \text{out}; \alpha'_{i,3p+1,k}, \text{in}),$
$p+1 \le i \le n-1, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha_{i,3p+1,k}, \text{out}; \alpha_{i,3p+2,k}, \text{in}),$

$1 \le i \le n-1, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha'_{i,3p+1,k}, \text{out}; \alpha'_{i,3p+2,k}, \text{in}),$

$1 \le i \le n-1, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha_{i,3p+2,k}, \text{out}; \alpha^2_{i,3p+3,k}, \text{in}),$

$1 \le i \le n-1, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha'_{i,3p+2,k}, \text{out}; \alpha'_{i,3p+3,k}\alpha'_{i,3p+3,k}, \text{in}),$

$1 \le i \le n-1, 1 \le p \le n-2, 0 \le k \le 1;$

$(\alpha_{i,3(n-1),k}, \text{out}; \Delta^k_{i,n}, \text{in}),$

$1 \le i \le n-1, 0 \le k \le 1;$

$(\alpha'_{i,3(n-1),k}, \text{out}; \Delta_{i,n}{}'k, \text{in}),$

$1 \le i \le n-1, 0 \le k \le 1;$

where $\Delta^0_{i,j}=F_{i,j}$, $\Delta_{i,j}{}'0=F'_{i,j}$, $\Delta^1_{i,j}=T_{i,j}$, $\Delta_{i,j}{}'1=T'_{i,j}$.

- Rules to generate in membrane 1 of configuration $\mathcal{C}_{3p+1}$ (p=0, 1, ..., n−1), the objects $B^{2^p}_{p+2}, B'_{p+2}2^p, S^{2^p}$:

$(\beta_{3p}, \text{out}; \beta_{3p+1}B_{p+2}, \text{in}), 0 \le p \le n-3;$

$(\beta'_{3p}, \text{out}; \beta'_{3p+1}B'_{p+2}, \text{in}), 0 \le p \le n-3;$

$(\beta''_{3p}, \text{out}; \beta''_{3p+1}S, \text{in}), 0 \le p \le n-3;$

$(\beta_{3p+1}, \text{out}; \beta_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\beta'_{3p+1}, \text{out}; \beta'_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\beta''_{3p+1}, \text{out}; \beta''_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\beta_{3p+2}, \text{out}; \beta^2_{3p+3}, \text{in}), 0 \le p \le n-3;$

$(\beta'_{3p+2}, \text{out}; \beta'_{3p+3}\beta'_{3p+3}, \text{in}), 0 \le p \le n-3;$

$(\beta''_{3p+2}, \text{out}; \beta''_{3p+3}\beta''_{3p+3}, \text{in}), 0 \le p \le n-3;$

$(\beta_{3(n-2)}, \text{out}; \beta_{3(n-2)+1}B_n, \text{in});$

$(\beta'_{3(n-2)}, \text{out}; \beta'_{3(n-2)+1}B'_n, \text{in});$

$(\beta''_{3(n-2)}, \text{out}; \beta''_{3(n-2)+1}S, \text{in});$

$(\beta_{3(n-2)+1}, \text{out}; \beta_{3(n-2)+2}, \text{in});$

$(\beta'_{3(n-2)+1}, \text{out}; \beta'_{3(n-2)+2}, \text{in});$

$(\beta''_{3(n-2)+1}, \text{out}; \beta''_{3(n-2)+2}, \text{in});$

$(\beta_{3(n-2)+2}, \text{out}; \beta^2_{3(n-2)+3}, \text{in});$

$(\beta'_{3(n-2)+2}, \text{out}; \beta'_{3(n-2)+3}\beta'_{3(n-2)+3}, \text{in});$

$(\beta''_{3(n-2)+2}, \text{out}; \beta''_{3(n-2)+3}\beta''_{3(n-2)+3}, \text{in});$

$(\beta_{3(n-1)}, \text{out}; B_{n+1}, \text{in});$

$(\beta'_{3(n-1)}, \text{out}; B'_{n+1}, \text{in});$

$(\beta''_{3(n-1)}, \text{out}; S, \text{in}).$

- Rules to generate in membrane 1 of configuration $\mathcal{C}_{3p+1}$ (p=0, 1, ..., n−1), the objects $T^{2^p}_{p+1,p+1}$, $T'_{p+1,p+1}2^p, A^{2^p}_{p+2}, A'_{p+2}2^p$:

$(\gamma_{3p}, \text{out}; \gamma_{3p+1}T_{p+1,p+1}, \text{in}), 0 \le p \le n-3;$

$(\gamma'_{3p}, \text{out}; \gamma'_{3p+1}F'_{p+1,p+1}, \text{in}), 0 \le p \le n-3;$

$(\gamma''_{3p}, \text{out}; \gamma''_{3p+1}A_{p+2}, \text{in}), 0 \le p \le n-3;$

$(\gamma'''_{3p}, \text{out}; \gamma'''_{3p+1}A'_{p+2}, \text{in}), 0 \le p \le n-3;$

$(\gamma_{3p+1}, \text{out}; \gamma_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\gamma'_{3p+1}, \text{out}; \gamma'_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\gamma''_{3p+1}, \text{out}; \gamma''_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\gamma'''_{3p+1}, \text{out}; \gamma'''_{3p+2}, \text{in}), 0 \le p \le n-3;$

$(\gamma_{3p+2}, \text{out}; \gamma^2_{3p+3}, \text{in}), 0 \le p \le n-3;$

$(\gamma'_{3p+2}, \text{out}; \gamma'_{3p+3}\gamma'_{3p+3}, \text{in}), 0 \le p \le n-3;$

$(\gamma''_{3p+2}, \text{out}; \gamma''_{3p+3}\gamma''_{3p+3}, \text{in}), 0 \le p \le n-3;$

$(\gamma'''_{3p+2}, \text{out}; \gamma'''_{3p+3}\gamma'''_{3p+3}, \text{in}), 0 \le p \le n-3;$

DOI 10.1002/cplx

$(\gamma_{3(n-2)}, \text{out}; \gamma_{3(n-2)+1} T_{n-1,n-1}, \text{in});$

$(\gamma'_{3(n-2)}, \text{out}; \gamma'_{3(n-2)+1} F'_{n-1,n-1}, \text{in});$

$(\gamma''_{3(n-2)}, \text{out}; \gamma''_{3(n-2)+1} A_n, \text{in});$

$(\gamma'''_{3(n-2)}, \text{out}; \gamma''_{3(n-2)+1} A'_n, \text{in});$

$(\gamma_{3(n-2)+1}, \text{out}; \gamma_{3(n-2)+2}, \text{in});$

$(\gamma'_{3(n-2)+1}, \text{out}; \gamma'_{3(n-2)+2}, \text{in});$

$(\gamma''_{3(n-2)+1}, \text{out}; \gamma''_{3(n-2)+2}, \text{in});$

$(\gamma'''_{3(n-2)+1}, \text{out}; \gamma'''_{3(n-2)+2}, \text{in});$

$(\gamma_{3(n-2)+2}, \text{out}; \gamma^2_{3(n-2)+3}, \text{in});$

$(\gamma'_{3(n-2)+2}, \text{out}; \gamma'_{3(n-2)+3}\, \gamma'_{3(n-2)+3}, \text{in});$

$(\gamma''_{3(n-2)+2}, \text{out}; \gamma''_{3(n-2)+3}\, \gamma''_{3(n-2)+3}, \text{in});$

$(\gamma'''_{3(n-2)+2}, \text{out}; \gamma'''_{3(n-2)+3}\, \gamma'''_{3(n-2)+3}, \text{in});$

$(\gamma_{3(n-1)}, \text{out}; T_{n,n}, \text{in});$

$(\gamma'_{3(n-1)}, \text{out}; F'_{n,n}, \text{in});$

$(\gamma''_{3(n-1)}, \text{out}; A_{n+1}, \text{in});$

$(\gamma'''_{3(n-1)}, \text{out}; A'_{n+1}, \text{in}).$

- Rules to generate in membrane 1 of configuration $\mathcal{C}_{3n}$, the objects $T_i^{2^{n-1}}, F_i^{2^{n-1}}$ $(1 \le i \le n)$:

$(\rho_{i,0}, \text{out}; \rho_{i,1}, \text{in}), 1 \le i \le n;$

$(\tau_{i,0}, \text{out}; \tau_{i,1}, \text{in}), 1 \le i \le n;$

$(\rho_{i,1}, \text{out}; \rho_{i,2}, \text{in}), 1 \le i \le n;$

$(\tau_{i,1}, \text{out}; \tau_{i,2}, \text{in}), 1 \le i \le n;$

$(\rho_{i,2}, \text{out}; \rho_{i,3}, \text{in}), 1 \le i \le n;$

$(\tau_{i,2}, \text{out}; \tau_{i,3}, \text{in}), 1 \le i \le n;$

$(\rho_{i,3p}, \text{out}; \rho_{i,3p+1}, \text{in}), 1 \le i \le n, 1 \le p \le n-2;$

$(\tau_{i,3p}, \text{out}; \tau_{i,3p+1}, \text{in}), 1 \le i \le n, 1 \le p \le n-2;$

$(\rho_{i,3p+1}, \text{out}; \rho^2_{i,3p+2}, \text{in}), 1 \le i \le n, 1 \le p \le n-2;$

$(\tau_{i,3p+1}, \text{out}; \tau^2_{i,3p+2}, \text{in}), 1 \le i \le n, 1 \le p \le n-2;$

$(\rho_{i,3p+2}, \text{out}; \rho_{i,3p+3}, \text{in}), 1 \le i \le n, 1 \le p \le n-2;$

$(\tau_{i,3p+2}, \text{out}; \tau_{i,3p+3}, \text{in}), 1 \le i \le n, 1 \le p \le n-2;$

$(\rho_{i,3(n-1)}, \text{out}; \rho_{i,3(n-1)+1}, \text{in}), 1 \le i \le n;$

$(\tau_{i,3(n-1)}, \text{out}; \tau_{i,3(n-1)+1}, \text{in}), 1 \le i \le n;$

$(\rho_{i,3(n-1)+1}, \text{out}; \rho^2_{i,3(n-1)+2}, \text{in}), 1 \le i \le n;$

$(\tau_{i,3(n-1)+1}, \text{out}; \tau^2_{i,3(n-1)+2}, \text{in}), 1 \le i \le n;$

$(\rho_{i,3(n-1)+2}, \text{out}; T_i, \text{in}), 1 \le i \le n;$

$(\tau_{i,3(n-1)+2}, \text{out}; F_i, \text{in}), 1 \le i \le n;$

$(A_i, \text{out}; a_i, \text{in}), 1 \le i \le n-1;$

$(A'_i, \text{out}; a_i, \text{in}), 1 \le i \le n-1;$

$(B_i, \text{out}; a_i, \text{in}), 1 \le i \le n-1;$

$(B'_i, \text{out}; a_i, \text{in}), 1 \le i \le n-1;$

$(y_i, \text{out}; z_i\, w_i, \text{in}), 1 \le i \le n-2;$

$(y_{n-1}, \text{out}; w_{n-1}, \text{in});$

$(w_i, \text{out}; c_{i+1}\, c'_{i+1}, \text{in}), 1 \le i \le n-1;$

$(z_i, \text{out}; v_{i+1}, \text{in}), 1 \le i \le n-2;$

$(v_i, \text{out}; y_i^2, \text{in}), 1 \le i \le n-1;$

$(a_i, \text{out}; b_{i+1}\, b'_{i+1}, \text{in}), 1 \le i \le n-1;$

$(q_{1,1}, \text{out}; r_{1,1}, \text{in});$

$(q_{ij}, \text{out}; r^2_{ij}, \text{in}), 1 \le i \le n-1, i \le j \le n-1;$

$(r_{ij}, \text{out}; s_{ij}\, u_{ij}, \text{in}), 1 \le i \le n-2, i \le j \le n-2;$

$(r_{i,n-1}, \text{out}; s_{i,n-1}, \text{in}), 1 \le i \le n-1;$

$(u_{1,j}, \text{out}; q_{1,j+1}\, q_{2,j+1}, \text{in}), 1 \le j \le n-2;$

$(u_{ij}, \text{out}; q_{i+1,j+1}, \text{in}), 2 \le i \le j, 2 \le j \le n-2;$

$(T_{ij}\, t_{ij}, \text{out}), 1 \le i \le j, 1 \le j \le n;$

$(T'_{ij}\, t_{ij}, \text{out}), 1 \le i \le j, 1 \le j \le n;$

$(F_{ij} f_{ij}, \text{out}), 1 \le i \le j, 1 \le j \le n;$

$(F'_{ij} f_{ij}, \text{out}), 1 \le i \le j, 1 \le j \le n.$

- Rules to have multiplicity 2^{n-1} of objects x_{ij} and $\bar{x}_{ij}$ such that $x_i \in C_j$ and $\neg x_i \in C_j$ in membrane 1 of configuration $\mathcal{C}_{n+1}$, and change the name x by e:

$(x_{ij}, \text{out}; d^2_{ij,1}; \text{in}), 1 \le i \le n, 1 \le j \le m;$

$(\bar{x}_{ij}, \text{out}; \bar{d}^2_{ij,1}; \text{in}), 1 \le i \le n, 1 \le j \le m;$

$(d_{ij,k}, \text{out}; d^2_{ij,k+1}, \text{in}), 1 \le i \le n, 1 \le j \le m, 1 \le k \le n-2;$

$(\bar{d}_{ij,k}, \text{out}; \bar{d}^2_{ij,k+1}, \text{in}), 1 \le i \le n, 1 \le j \le m, 1 \le k \le n-2;$

$(d_{ij,n-1}, \text{out}; e_{ij}, \text{in}), 1 \le i \le n, 1 \le j \le m;$

$(\bar{d}_{ij,n-1}, \text{out}; \bar{e}_{ij}, \text{in}), 1 \le i \le n, 1 \le j \le m.$

- Output rule with **affirmative** answer: $(E_0 f_{3n+2m}\, yes; \text{out})$.
- Output rule with **negative** answer: $(f_{3n+2m}\, no; \text{out})$.
- Rules to generate in membrane 1 of configuration $\mathcal{C}_{3n}$, the objects $E_1^{2^n}$, and in membrane 1 of configuration $\mathcal{C}_{3n+1}$, the objects $E_0^{2^n}, E_2^{2^n}, \ldots, E_m^{2^n}$:

$(\delta_{j,3p}, \text{out}; \delta_{j,3p+1}, \text{in}), 0 \le j \le m, 0 \le p \le n-1;$

$(\delta_{j,3p+1}, \text{out}; \delta^2_{j,3p+2}, \text{in}), 0 \le j \le m, 0 \le p \le n-1;$

$(\delta_{j,3p+2}, \text{out}; \delta_{j,3p+3}, \text{in}), 0 \le j \le m, 0 \le p \le n-2;$

$(\delta_{1,3(n-1)+2}, \text{out}; E_1, \text{in});$

$(\delta_{j,3(n-1)+2}, \text{out}; \delta_{j,3(n-1)+3}, \text{in}), 0 \le j \le m, j \ne 1;$

$(\delta_{j,3n}, \text{out}; E_j, \text{in}), 0 \le j \le m, j \ne 1;$

$(f_p, \text{out}; f_{p+1}; \text{in}), 0 \le p \le 3n+2m-1.$

DOI 10.1002/cplx

- Rules to remove a part of the garbage objects:

$$(t_{i,k}\, T_{i,k}\,, \text{out}), 1 \le i < k, 2 \le k \le n;$$
$$(t_{i,k}\, T'_{i,k}\,, \text{out}), 1 \le i < k, 2 \le k \le n;$$
$$(f_{i,k}\, F_{i,k}\,, \text{out}), 1 \le i < k, 2 \le k \le n;$$
$$(f_{i,k}\, F'_{i,k}\,, \text{out}), 1 \le i < k, 2 \le k \le n;$$
$$(t_{i,i}\, T_{i,i}\,, \text{out}), 1 \le i \le n;$$
$$(f_{i,i}\, F'_{i,i}\,, \text{out}), 1 \le i \le n;$$

$$(b_k\, B_{k+1}\,, \text{out}), n-1 \le k \le n;$$
$$(b'_k\, B'_{k+1}\,, \text{out}), n-1 \le k \le n;$$
$$(c_k\, A_{k+1}\,, \text{out}), n-1 \le k \le n;$$
$$(c'_k\, A'_{k+1}\,, \text{out}), n-1 \le k \le n.$$

6. Rules in $\mathcal{R}_2$:

- Separation rule to produce all truth assignments for the variables $\{x_1, \ldots, x_n\}$ associated with the input formula: $[\,S\,]_2 \rightarrow [\,\Gamma_0\,]_2\, [\,\Gamma_1\,]_2$.
- Rules to produce objects $T_{i,i}, A_{i+1}, F'_{i,i}, A'_{i+1}$ in membrane 2:

$$(A_i, \text{out}; c_i\, c'_i, \text{in}), 1 \le i \le n;$$
$$(A'_i, \text{out}; c_i\, c'_i, \text{in}), 1 \le i \le n;$$
$$(B_i, \text{out}; b_i\, b'_i, \text{in}), 1 \le i \le n;$$
$$(B'_i, \text{out}; b_i\, b'_i, \text{in}), 1 \le i \le n;$$
$$(b_i, \text{out}; B_{i+1}\, S, \text{in}), 1 \le i \le n;$$
$$(b'_i, \text{out}; B'_{i+1}, \text{in}), 1 \le i \le n;$$
$$(c_i, \text{out}; T_{i,i}\, A_{i+1}, \text{in}), 1 \le i \le n;$$
$$(c'_i, \text{out}; F'_{i,i} A'_{i+1}, \text{in}), 1 \le i \le n.$$

- Rules to produce an object E_1 in each membrane 2 of configuration $\mathcal{C}_{3n+1}$ and an object E_0 in each membrane 2 of configuration $\mathcal{C}_{3n+2}$:

$$(B_{n+1}, \text{out}; E_1, \text{in})\,;\ (B'_{n+1}, \text{out}; E_1, \text{in});$$
$$(A_{n+1}, \text{out}; E_0, \text{in})\,;\ (A'_{n+1}, \text{out}; E_0, \text{in}).$$

- Rules to produce a truth assignment in each membrane 2 of configuration $\mathcal{C}_{3n+1}$:

$$(T_{i,j}, \text{out}; t_{i,j}, \text{in}), 1 \le i \le j, 1 \le j \le n;$$
$$(T'_{i,j}, \text{out}; t_{i,j}, \text{in}), 1 \le i \le j, 1 \le j \le n;$$
$$(F_{i,j}, \text{out}; f_{i,j}, \text{in}), 1 \le i \le j, 1 \le j \le n;$$
$$(F'_{i,j}, \text{out}; f_{i,j}, \text{in}), 1 \le i \le j, 1 \le j \le n;$$

$$(t_{i,j}, \text{out}; T_{i,j+1}\, T'_{i,j+1}, \text{in}), 1 \le i \le j, 1 \le j \le n-1;$$
$$(f_{i,j}, \text{out}; F_{i,j+1}\, F'_{i,j+1}, \text{in}), 1 \le i \le j, 1 \le j \le n-1;$$
$$(T_{i,n}, \text{out}; T_i, \text{in}), 1 \le i \le n;$$
$$(T'_{i,n}, \text{out}; T_i, \text{in}), 1 \le i \le n;$$
$$(F_{i,n}, \text{out}; F_i, \text{in}), 1 \le i \le n;$$
$$(F'_{i,n}, \text{out}; F_i, \text{in}), 1 \le i \le n.$$

- Rules to check the clause j through the truth assignment encoded by a membrane 2:

$$(E_j\, T_i, \text{out}; e_{i,j}, \text{in}), 1 \le i \le n, 1 \le j \le m;$$
$$(E_j\, F_i, \text{out}; \bar{e}_{i,j}, \text{in}), 1 \le i \le n, 1 \le j \le m.$$

- Rules to restore the truth assignment encoded by a membrane 2 which make true, the clause j:

$$(e_{i,j}, \text{out}, E_{j+1}\, T_i, \text{in}), 1 \le i \le n, 1 \le j \le m-1;$$
$$(\bar{e}_{i,j}, \text{out}, E_{j+1}\, F_i, \text{in}), 1 \le i \le n, 1 \le j \le m-1.$$

- Rules to send to the membrane 1 of configuration $\mathcal{C}_{3n+2m+1}$, an object E_0, meaning that some truth assignment encoded by a membrane labelled by 2 makes true the input formula φ:

$$(e_{i,m}\, E_0\,; \text{out}), 1 \le i \le n;$$
$$(\bar{e}_{i,m}\, E_0\,; \text{out}), 1 \le i \le n.$$

7. Rules in $\mathcal{R}_3$:

- Rules to produce objects $f'_{3n+2m+1}$ and no in membrane 1 of configuration $\mathcal{C}_{3n+2m+2}$ to produce a proper answer in the negative case (not satisfiable formula).

$$(f'_p, \text{out}; f'_{p+1}, \text{in}), 0 \le p \le 3n+2m;$$

$$(f'_{3n+2m+1}\, no\,; \text{out}).$$

8. $i_{\text{in}} = 1$ and $i_{\text{out}} = 0$.

5. AN OVERVIEW OF THE COMPUTATIONS

Let $\varphi = C_1 \wedge \ldots \wedge C_m$ be an instance of the SAT problem consisting of m clauses $C_j = l_{j,1} \vee \ldots \vee l_{j,r_j}$, $1 \le j \le m$, where $\text{Var}(\varphi) = \{x_1, \ldots, x_n\}$ and $l_{j,k} \in \{x_i, \neg x_i | 1 \le i \le n\}$, $1 \le j \le m$, $1 \le k \le r_j$. Let us assume that the number of variables n, and the number of clauses m, of φ, are greater than or equal to 2.

We consider the polynomial encoding (cod, s) of instances from SAT in Π defined as follows: $s(\varphi) = \langle m, n \rangle$ and $\text{cod}(\varphi) = \{x_{i,j} | x_i \in C_j\} \cup \{\bar{x}_{i,j} | \neg x_i \in C_j\}$, for each $\varphi \in I_{SAT}$. The Boolean formula φ will be processed by the system $\Pi(s(\varphi)) + \text{cod}(\varphi)$. In what follows, we informally describe how that system works.

DOI 10.1002/cplx

The solution proposed follows a brute force algorithm in the framework of recognizer P systems with SAS P system, and it consists of the following phases:

- *Generation phase*: using separation rules, all truth assignments for the variables $\{x_1,\ldots,x_n\}$ associated with φ are produced. This phase takes $3n+1$ computation steps.
- *Checking phase*: checking whether or not the input formula φ is satisfied by some truth assignment generated in the previous phase. This phase takes $3m+1$ steps, where m is the number of clauses of φ.
- *Output phase*: sending the right answer into the environment according to the results of the previous phase. This phase takes 1 step if the answer is affirmative or 3 steps if the answer is negative.

5.1. Generation Phase

In this phase, all truth assignments for the variables associated with the Boolean formula $\varphi(x_1,\ldots,x_n)$ are going to be generated, by applying separation rules in membranes labelled by 2. In this way, after completing the phase, there exist 2^n membranes labelled by 2 such that each of them encodes a different truth assignment of the variables $\{x_1,\ldots,x_n\}$.

This phase consists of a loop with n iterations and one additional final step. Each iteration of the loop takes three steps. So, this phase takes $3n+1$ steps.

To do this, in the configurations $\mathcal{C}_{3p+2}$ $(0\leq p\leq n-1)$, there exist 2^p membranes labelled by 2 containing objects

$$A_{p+2},\ A'_{p+2},\ B_{p+2},\ B'_{p+2},\ T_{p+1,p+1},\ F'_{p+1,p+1},\ S,$$

along with $2p$-tuples of objects $(\pi_{1,p+1},\pi'_{1,p+1},\ldots,\pi_{p,p+1},\pi'_{p,p+1})$ with $\pi\in\{T,F\}$, in such a way that the corresponding tuples in the different membranes are different from each other.

In this way, a separation rule can be applied to each of the membranes labelled by 2, producing that in the configuration $\mathcal{C}_{3p+3}$ $(0\leq p\leq n-2)$,there exist 2^{p+1} membranes labelled by 2, with 2^p membranes of them containing objects A_{p+2} and B_{p+2}, as well as $(p+1)$-tuples $(\pi_{1,p+1},\ldots,\pi_{p+1,p+1})$ with $\pi\in\{T,F\}$, in such a way that $\pi_{p+1,p+1}=T_{p+1,p+1}$ and the corresponding tuples in the different membranes are different from each other. The other 2^p membranes labelled by 2, contain the objects A'_{p+2} and B'_{p+2}, as well as $(p+1)$-tuples $(\pi'_{1,p+1},\ldots,\pi'_{p+1,p+1})$ with $\pi\in\{T,F\}$, in such a way that $\pi'_{p+1,p+1}=F'_{p+1,p+1}$ and the corresponding tuples in the different membranes are different from each other.

Finally, in the configuration $\mathcal{C}_{3n}$, there exist 2^n membranes labelled by 2, where 2^{n-1} membranes of them contain the objects A_{n+1} and B_{n+1}, as well as n-tuples $(\pi_{1,n},\ldots,\pi_{n,n})$ with $\pi\in\{T,F\}$, in such a way that $\pi_{n,n}=T_{n,n}$ and the corresponding tuples in the different membranes are different from each other; the other 2^{n-1} membranes labelled by 2, contain the objects A'_{n+1} and B'_{n+1}, as well as n-tuples $(\pi'_{1,n},\ldots,\pi'_{n,n})$ with $\pi\in\{T,F\}$, in such a way that $\pi'_{n,n}=F'_{n,n}$ and the corresponding tuples in the different membranes are different from each other.

This phase ends in step $3n+1$, where the configuration $\mathcal{C}_{3n+1}$ contains 2^n membranes labelled by 2, each of these 2^n membranes contains the objects A_{n+1} and E_1, as well as n-tuples $(\pi_1,\ldots,\pi_n)$ with $\pi\in\{T,F\}$, and the corresponding tuples in the different membranes are different from each other.

During the generation phase, 2^{n-1} copies are generated from each object contained in the input multiset placed initially in the skin membrane. Due to technical reasons, we change variables $x_{i,j}$ and $\bar{x}_{i,j}$ to $e_{i,j}$ and $\bar{e}_{i,j}$, respectively. This is accomplished using the following rules from $\mathcal{R}_1$:

$$(x_{ij},\text{out};d^2_{i,j,1};\text{in}),\ 1\leq i\leq n,1\leq j\leq m;$$

$$(\bar{x}_{ij},\text{out};\bar{d}^2_{i,j,1};\text{in}),\ 1\leq i\leq n,1\leq j\leq m;$$

$$(d_{i,j,k},\text{out};d^2_{i,j,k+1},\text{in}),\ 1\leq i\leq n,1\leq j\leq m,1\leq k\leq n-2;$$

$$(\bar{d}_{i,j,k},\text{out};\bar{d}^2_{i,j,k+1},\text{in}),\ 1\leq i\leq n,1\leq j\leq m,1\leq k\leq n-2;$$

$$(d_{i,j,n-1},\text{out};e_{i,j},\text{in}),\ 1\leq i\leq n,1\leq j\leq m;$$

$$(\bar{d}_{i,j,n-1},\text{out};\bar{e}_{i,j},\text{in}),\ 1\leq i\leq n,1\leq j\leq m.$$

The multiset that codifies the input formula will be denoted by $(\text{cod}(\varphi))^{2^{n-1}}_e$.

5.2. Checking Phase

This phase begins at computation step $3n+2$ and consists of a loop with m iterations, where each iteration takes two steps. So, the checking phase takes $2m$ steps.

In the configuration $\mathcal{C}_{3n+1}$, the presence of an object E_1 in each membrane labelled by 2, along with the code of a truth assignment, marks the beginning of this phase. In the first iteration of the loop, the truth assignments making true the first clause of φ are found. To do this, the following rules of $\mathcal{R}_2$ are applied:

$$(E_1\,T_i,\text{out};e_{i,1},\text{in}),\ 1\leq i\leq n;$$
$$(E_1\,F_i,\text{out};\bar{e}_{i,1},\text{in}),\ 1\leq i\leq n.$$

In step $(3n+1)+2$, the object E_0 is incorporated to each of the membranes labelled by 2, by means of the application of the following rules of $\mathcal{R}_2$: $(A_{n+1},\text{out},E_0,\text{in})$ and $(A'_{n+1},\text{out},E_0,\text{in})$. The presence of an object $e_{i,1}$ or an object $\bar{e}_{i,1}$ in a membrane 2 of the configuration $\mathcal{C}_{(3n+1)+1}$ indicates that this membrane codifies a truth assignment making true the first clause. In the next computation step, those membranes containing object $e_{i,1}$ or $\bar{e}_{i,1}$ will incorporate an object E_2 from the skin membrane by applying the following rules from $\mathcal{R}_2$:

DOI 10.1002/cplx

$$(e_{i,1}, \text{out}, E_2\, T_i, \text{in}), 1 \leq i \leq n;$$
$$(\bar{e}_{i,1}, \text{out}, E_2\, F_i, \text{in}), 1 \leq i \leq n.$$

In this way, the presence of an object E_2 in a membrane 2 of the configuration $\mathcal{C}_{(3n+1)+2}$ indicates that this membrane codifies a truth assignment making true, the first clause and that the system is ready to check the second clause of the formula. That is, from this moment, the membranes labelled by 2 not making true, the first clause will not evolve.

In the jth iteration $(2 \leq j \leq m)$ of the aforementioned loop, the truth assignments making true, the clause j of the formula are checked, taking into account that only the truth assignments containing the object E_j will be checked, since only these membranes make true, the clauses $1, \ldots, j-1$ of φ. This is accomplished by applying the following rules from $\mathcal{R}_2$:

$$(E_j\, T_i, \text{out}; e_{i,j}, \text{in}), 1 \leq i \leq n, 1 \leq j \leq m;$$
$$(E_j\, F_i, \text{out}; \bar{e}_{i,j}, \text{in}), 1 \leq i \leq n, 1 \leq j \leq m.$$

The presence of an object $e_{i,j}$ or an object $\bar{e}_{i,j}$ in a membrane 2 of the configuration $\mathcal{C}_{(3n+1)+2\cdot(j-1)+1}$ indicates that this membrane codifies a truth assignment making true the clauses $1, \ldots, j$ of φ. Then, those membranes will incorporate an object E_{j+1} from the skin membrane by applying the following rules from $\mathcal{R}_2$:

$$(e_{i,j}, \text{out}, E_{j+1}\, T_i, \text{in}), 1 \leq i \leq n, 1 \leq j \leq m-1;$$
$$(\bar{e}_{i,j}, \text{out}, E_{j+1}\, F_i, \text{in}), 1 \leq i \leq n, 1 \leq j \leq m-1.$$

If the input formula φ is satisfiable, then in some membrane labelled by 2 of the configuration $\mathcal{C}_{(3n+1)+2(m-1)+1}$, there exists an object $e_{i,m}$ or an object $\bar{e}_{i,m}$, which indicates that the truth assignment codified by this membrane makes true, all the clauses from φ and, consequently, makes true, the input formula. In this case, by applying a rule from $\mathcal{R}_2$ of the form $(e_{i,m}\, E_0\, ; \text{out})$ or $(\bar{e}_{i,m}\, E_0\, ; \text{out})$, an object E_0 goes to the skin membrane of the configuration $\mathcal{C}_{(3n+1)+2(m-1)+2}$, and the checking phase ends, where the object f_{3n+2m} has also been produced.

If the input formula φ is not satisfiable, then no membrane labelled by 2 of the configuration $\mathcal{C}_{(3n+1)+2(m-1)+1}$ contains an object $e_{i,m}$ or an object $\bar{e}_{i,m}$. In this case, by applying the rule $(f'_{3n+2m}, \text{out}; f'_{3n+2m+1}, \text{in}) \in \mathcal{R}_3$ (in fact, this is the only rule applicable to the configuration $\mathcal{C}_{(3n+1)+2(m-1)+1}$), the object $f'_{3n+2m+1}$ appears in the skin membrane, and the checking phase ends.

In general, the checking phase ends at step $(3n+1)+2(m-1)+2=3n+2m+1$.

5.3. Output Phase

If the input formula φ is satisfiable, then objects E_0 and f_{3n+2m} will appear in the skin membrane of the configuration $\mathcal{C}_{3n+2m+1}$. By applying the rule $(E_0 f_{3n+2m}\, yes\, ; \text{out})$ in the skin membrane, the object yes is released into the environment, providing an affirmative answer at computation step $(3n+1)+2m+1=3n+2m+2$.

If the input formula φ is not satisfiable, then object f'_{3n+2m} is present in the skin membrane of the configuration $\mathcal{C}_{(3n+1)+2(m-1)+1}=\mathcal{C}_{3n+2m}$, but not the object E_0. In this case, only the rule $(f'_{3n+2m}, \text{out}; f'_{3n+2m+1}, \text{in}) \in \mathcal{R}_3$ is applicable and applied, and in the next step only the rule $(f'_{3n+2m+1}\, no\, ; \text{out}) \in \mathcal{R}_3$ is applicable and applied. Consequently, objects $f_{3n+2m} yes, f'_{3n+2m+1}$ no appear in the skin membrane of the configuration $\mathcal{C}_{3n+2m+2}$. Then, by applying the rule $(f_{3n+2m}\, no\, ; \text{out})$ in the skin membrane, an object no is released into the environment, providing a negative answer in step $3n+2m+4$.

In general, the output phase takes one computation step in the case of an affirmative answer, and three computation steps in the case of a negative answer.

As a result of the above overview of the computations, we have the following theorem.

Theorem 5.1

SAT $\in$ $\mathbf{PMC}_{\mathbf{CSC(3)}}$.

Noting that the SAT problem is a **NP**-complete problem and the complexity class $\mathbf{PMC}_{\mathbf{CSC(3)}}$ is closed under polynomial-time reduction and under complement, we have the following corollary.

Corollary 5.1

$$\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} \subseteq \mathbf{PMC}_{\mathbf{CSC(3)}}.$$

6. CONCLUSIONS

In this work, cell-like P systems with SAS P system have been investigated from a computational complexity point of view. Two main results have been obtained. On the one hand, only tractable problems can be efficiently solved by families of P systems with symport/antiport rules of length at most two and membrane separation rules, that is, $\mathbf{PMC}_{\mathbf{CSC(2)}}=\mathbf{P}$. On the other hand, a uniform polynomial time solution to the SAT problem by a family of such P systems which use symport/antiport rules of length at most three has been provided. Thus, a new optimal tractability border has been obtained. Specifically, we have shown that in the framework of SAS P systems, the length of symport/antiport rules passing from 2 to 3 amounts to the computational power passing from nonefficiency to efficiency, assuming that $\mathbf{P} \neq \mathbf{NP}$. So, in the biology inspired computational models, cell-like P systems with SAS P system, the length of symport/antiport rules is an essential parameter for the computational power.

7. ACKNOWLEDGMENTS

The work of L. Pan and B. Song was supported by National Natural Science Foundation of China (61033003,

DOI 10.1002/cplx

91130034 and 61320106005). The work of M.J. Pérez-Jiménez, L. Valencia-Cabrera and L.F. Macías-Ramos was supported by Project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

REFERENCES

1. Morlot, S.; Roux, A. Mechanics of dynamic-mediated membrane fission. Ann Rev Biophys 2013, 42, 629–649.
2. Păun, G.; Rozenberg, G.; Salomaa, A. The Oxford Handbook of Membrane Computing. Oxford University Press: New York, 2010.
3. Alhazov, A.; Ishdorj, T.-O. Membrane operations in P systems with active membranes. In: Proceedings of the Second Brainstorming Week on Membrane Computing, Sevilla: Spain, 2–7 February 2004, pp 37-44.
4. Pan, L.; Ishdorj, T.-O. P systems with active membranes and separation rules. J Univ Comput Sci 2004, 10, 630–649.
5. Păun, A.; Păun, G.; Rozenberg, G. Computing by communication in networks of membranes. Int J Found Comput Sci 2002, 13, 779–798.
6. Martín Vide, C.; Pazos, J.; Păun, G.; Rodríguez Patón, A. A new class of symbolic abstract neural nets: tissue P systems. In: Computing and Combinatorics, Lecture Notes in Computer Science, Vol. 2387; 2002, pp 290–299.
7. Martín Vide, C.; Pazos, J.; Păun, G.; Rodríguez Patón, A. Tissue P systems. Theor Comput Sci 2003, 296, 295–326.
8. Păun, G.; Pérez-Jiménez, M.J.; Riscos-Núñez, A. Tissue P systems with cell division. Int J Comput Commun 2008, 3, 295–303.
9. Pan, L.; Pérez-Jiménez, M.J. Computational complexity of tissue–like P systems. J Complex 2010, 26, 296–315.
10. Pérez-Jiménez, M.J.; Sosík, P. An optimal frontier of the efficiency of tissue P systems with cell separation, Fund inform 2015, 138: 45–60.
11. Păun, A.; Păun, G. The power of communication: P systems with symport/antiport. New Generat Comput 2002, 20, 295–305.
12. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L. An Introduction to Algorithms. The MIT Press: Cambridge, MA, 1994.
13. Pérez-Jiménez, M.J.; Romero-Jiménez, A.; Sancho-Caparrini, F. A polynomial complexity class in P systems using membrane division. J Autom Lang Combinat 2006, 11, 423–434.
14. Pérez-Jiménez, M.J.; Romero-Jiménez, A.; Sancho-Caparrini, F. Complexity classes in models of cellular computing with membranes. Nat Comput 2003, 2, 265–285.

DOI 10.1002/cplx