Theoretical Computer Science 908 (2022) 56–63

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

P systems with evolutional symport and membrane creation rules solving QSAT

David Orellana-Martín[*], Luis Valencia-Cabrera, Mario J. Pérez-Jiménez

Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Avda. Reina Mercedes s/n, 41012, Universidad de Sevilla, Sevilla, Spain

A R T I C L E I N F O

Article history:
Received 31 May 2021
Received in revised form 22 November 2021
Accepted 24 November 2021
Available online 6 December 2021

Keywords:
Membrane computing
Membrane creation
QBF-SAT
Computational complexity theory

A B S T R A C T

P systems are computing devices based on sets of rules that dictate how they work. While some of these rules can change the objects within the system, other rules can even change the own structure, like creation rules. They have been used in cell-like membrane systems with active membranes to efficiently solve **NP**-complete problems. In this work, we improve a previous result where a uniform family of P systems with evolutional communication rules whose left-hand side (respectively, right-hand side) have most 2 objects (resp., 2 objects) and membrane creation solved SAT efficiently, and we obtain an efficient solution to solve QBF-SAT or QSAT (a **PSPACE**-complete problem) having at most 1 object (respectively, 1 object) in their left-hand side (resp., right-hand side) and not making use of the environment.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

In 1998, Membrane Computing was introduced as a model of computation inspired by the structure and behaviour of living cells [1], abstracting the inner organelles as internal regions and chemical elements as objects of the system. The devices generated in this framework are called membrane systems or P systems. The type of region depends on the exact living model selected: On the one hand, a single cell with its inner compartments is abstracted as a cell-like membrane system (or P system) [1], where these membranes are semi-permeable walls that have control over the objects that can pass through it. The structure of these systems can be represented as a graph arranged as a rooted tree, where the root node is the outermost membrane, called *skin* membrane. On the other hand, several cells communicating among them can be found in nature forming a tissue. We can find two main abstractions of this phenomenon: tissue-like membrane systems, or tissue P systems [2,3], where cells can interchange objects with other cells or even with the environment, and spiking neural P systems [4], where the flow of communication through electrical impulses between neurons is abstracted through a directed graph.

Cell-like membrane systems use a membrane structure that can be either static or dynamic. In the first case, objects can move through the different membranes in each step of computation. P systems with symport/antiport rules [5] make an abstraction of the symport/antiport mechanism of the organelles of a cell that allow only some objects to pass, or can interchange chemical elements from two adjoining compartments. In transition P systems and P systems with active membranes [1,6], the membrane structure could be changed, by dissolving the membranes or, in the latter, by duplicating

[*] Corresponding author.
E-mail addresses: dorellana@us.es (D. Orellana-Martín), lvalencia@us.es (L. Valencia-Cabrera), marper@us.es (M.J. Pérez-Jiménez).

https://doi.org/10.1016/j.tcs.2021.11.012
0304-3975/© 2022 Elsevier B.V. All rights reserved.

a membrane and its contents by using division rules. Another method for duplicating a membrane is by using membrane separation rules [7], where the contents of the original membrane are distributed among the two new membranes. Another method to increase the number of membranes of the system is by means of membrane creation rules [8], where a single object creates a new membrane. It could be seen as the chemical element taking some of the substrate of the membrane where it is and using it to create a new membrane. This kind of structure-changing mechanisms makes P systems capable of efficiently solving computationally hard problems. More precisely, **NP**-complete or even **PSPACE**-complete problems have been solved by means of families of P systems that use one of these kinds of rules [9–17].

In [18], a new variant of tissue P systems was introduced, where symport/antiport rules allow objects to evolve in the process of the communication from a region to another region. Some relevant results have been obtained in [19–22], where some frontiers of efficiency have been obtained. In the framework of cell-like membrane systems, creation rules have been used lately besides evolutional communication rules in [23] to solve the SAT problem by means of a family of recognizer P systems with evolutional communication rules and creation rules. In [24], an efficient solution to QSAT is given by means of a family of recognizer polarizationless P systems with active membranes and membrane creation. In this work, we replicate the results of the latter paper; that is, we provide an efficient solution to the QSAT problem, in this case by means of a uniform family of recognizer P systems with evolutional symport rules with a minimal amount of objects per communication rule. This is an important improvement in terms of the transport of objects, since, in this case, no cooperation between objects is needed in the left-hand side to obtain presumed efficiency. For an overview of the state of the art in the area, we refer the reader to [25].

The rest of the work is organized as follows: Section 2 is devoted to introduce some concepts used later through the paper. In Section 3, the definition of recognizer P systems with evolutional communication rules and membrane creation is given. In the following section, a polynomial-time and uniform solution to QSAT is given by means of a family of recognizer P systems with evolutional symport/antiport rules of length at most $(1, 1)$ and membrane creation, and an overview of the computations and the formal verification of the design are specified. Finally, some remarks and open research lines are indicated.

2. Preliminaries

Some basic notions of formal languages, set theory and other terms used throughout the paper are recalled in this section. For a deeper explanation on formal languages and membrane computing, we refer the reader to [26,27].

An alphabet Γ is a non-empty and finite set, and its elements are called *symbols*. A string u over Γ is a finite sequence of symbols from Γ. The number of occurrences of a symbol a in u is denoted by $|u|_a$. The *length* of u, denoted $|u|$ is $\sum_{a\in u} |u|_a$.

A *multiset* m over Γ is a pair (Γ, f_m) where $f_m : \Gamma \to \mathbb{N}$ is a mapping from Γ to the set of natural numbers $\mathbb{N}$. Let m_1, m_2 two multisets over Γ. The union of m_1 and m_2, denoted by $m_1 + m_2$ or $m_1 \cup m_2$ is defined as $(m_1 + m_2)(x) = f_{m_1}(x) + f_{m_2}(x), \forall x \in \Gamma$. The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is the defined as

$$(m_1 \setminus m_2)(x) = \begin{cases} f_{m_1}(x) - f_{m_2}(x) & \text{if } f_{m_1}(x) \geq f_{m_2}(x) \\ 0 & \text{if } f_{m_1}(x) < f_{m_2}(x) \end{cases}$$

The empty multiset (the multiset with no elements) is denoted by $\emptyset$, and the set of all finite multisets over Γ is denoted by $M_f(\Gamma)$.

The Cantor pairing function $\langle \cdot, \cdot \rangle$ over $\mathbb{N}^2$ is a bijective function defined as $\langle a, b \rangle = \frac{(a+b+1)(a+b)}{2} + b$.

A decision problem X over the alphabet Σ is a pair (I_X, θ_X) such that $I_X \in \Sigma^*$ is the set of instances of X and θ_X is a Boolean function over I_X. Let $\mathcal{R}$ a class of recognizer P systems.

3. Recognizer P systems with evolutional communication rules and creation rules

In this section, a definition of recognizer P systems with evolutional communication rules and creation rules is given, and both the syntax and semantics are recalled. It must be taken into account that, in recognizer P systems, the output region is the environment, marked as env.

Definition 1. A recognizer P system with evolutional communication rules and membrane creation of degree $q \geq 1$ is a tuple

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

where:

1. Γ is a finite alphabet of objects (the *working alphabet*), and $\Sigma, \mathcal{E} \subseteq \Gamma, \Sigma \cap \mathcal{E} = \emptyset$ (being Σ the *input alphabet* and $\mathcal{E}$ the *environment alphabet*);
2. H is a finite set of labels;
3. μ is a membrane structure whose elements are bijectively labelled by elements of H;
4. $\mathcal{M}_i, 1 \leq i \leq q$ are finite multisets over $\Gamma \setminus (\Sigma \cup \mathcal{E})$;
5. $\mathcal{R}$ is a set of rules of the following forms:

- Evolutional symport rules:
  - Between membranes:
    - $[\,u\,[\;\;]_i\,]_j \to [\,[\,u'\,]_i\,]_j$, where $i, j \in H, i \neq j$, $u, u' \in M_f(\Gamma), |u| > 0$;
    - $[\,[\,u\,]_i\,]_j \to [\,u'\,[\;\;]_i\,]_j$, where $i, j \in H, i \neq j$, $u, u' \in M_f(\Gamma), |u| > 0$;
  - Between the environment and the skin membrane:
    - $[\,u\,[\;\;]_i\,]_{env} \to [\,[\,u'\,]_i\,]_{env}$, where $i \in H$, being i the label of the skin membrane, $u, u' \in M_f(\Gamma), |u| > 0$, there exists at least one $a \in u$ such that $a \in \Gamma \setminus \mathcal{E}$ (otherwise, since there exists an arbitrary number of objects from $\mathcal{E}$ in the environment, that rule could be applied an infinite number of times);
    - $[\,[\,u\,]_i\,]_{env} \to [\,u'[\;\;]_i\,]_{env}$, where $i \in H$, being i the label of the skin membrane, $u, u' \in M_f(\Gamma), |u| > 0$;
- Evolutional antiport rules: $[\,u\,[\,v\,]_i\,]_j \to [\,v'\,[\,u'\,]_i\,]_j$, where $i \in H, j \in H \cup \{env\}, u, v, u', v' \in M_f(\Gamma), |u|, |v| > 0$;
- Creation rules: $[\,a \to [\,u\,]_i\,]_j$, where $i, j \in H, i \notin \{skin\}$, where $skin$ is the label of the skin membrane, $a \in \Gamma, u \in M_f(\Gamma)$;

6. $i_{in} \in H$;
7. $i_{out} = env$.

A *configuration* C_t of a P system with evolutional communication rules and creation rules is described by the membrane structure at the moment t and the multisets of objects over Γ of each membrane, and the multiset of objects over $\Gamma \setminus \mathcal{E}$ of the environment. Objects from the environment do not appear, in the initial configuration, within the P system. Therefore, $\mathcal{E} \cap \mathcal{M}_i = \emptyset, 1 \leq i \leq q$. Besides, objects from Σ are exclusively introduced to the corresponding input membrane encoding the instance of the problem. If $\mathcal{E} = \emptyset$, it is usually omitted in the tuple of the system.

The *initial configuration* of such a P system is defined as $(\mathcal{M}_1, \dots, \mathcal{M}_{i_{in}} + m, \dots, \mathcal{M}_q; \emptyset)$, begin m the input multiset and the last element being the multiset of objects from $\Gamma \setminus \mathcal{E}$ in the environment. We use the term region i to refer to a membrane if $i \in H$ and to the environment if $i = env$. μ is a tree-like structure representing the initial structure of the P system, where parent and children regions/membranes are defined in a natural way. The parent node of the *skin* membrane is the environment (with label env). In this sense, env is an artificial label to the environment, since it usually does not have any label. In this type of systems, due to the definition of the rules, this artificial label must be given to the environment.

An evolutional symport rule $[\,u\,[\;\;]_i\,]_j \to [\,[\,u'\,]_i\,]_j$, called evolutional send-in rule, can be applied to a configuration C_t if there exists a membrane labelled by i whose parent region is labelled by j, and this region j contains a multiset of objects u. When applying such a rule, the multiset of objects u is consumed from the region j and a multiset of objects u' is produced in the membrane i in the next configuration.

An evolutional symport rule $[\,[\,u\,]_i\,]_j \to [\,u'\,[\;\;]_i\,]_j$, called evolutional send-out rule, can be applied to a configuration C_t if there exists a membrane labelled by i that contains a multiset of objects u, and whose parent region is labelled by j. When applying such a rule, the multiset of objects u is consumed from the membrane i and a multiset of objects u' is produced in the region j in the next configuration.

An evolutional antiport rule $[\,u\,[\,v\,]_i\,]_j \to [\,v'\,[\,u'\,]_i\,]_j$ can be applied to a configuration C_t if there exists a membrane labelled by i that contains a multiset of objects v, and whose parent region is labelled by j and contains a multiset of objects u. When applying such a rule, the multisets of objects v and u are consumed from the membrane i and the region j, respectively, and multisets u' and v' are produced in the membrane i and the region j, respectively.

A creation rule $[\,a \to [\,u\,]_i\,]_j$ can be applied to a configuration C_t if there exists a membrane labelled by j that contains an object a. When applying such a rule, object a is consumed from membrane j and a new membrane labelled by i and containing the multiset of objects u appears as a child membrane of j.

A recognizer P system with evolutional communication rules and creation rules that does not send objects from the environment to the system is said to be a P system with evolutional communication rules and creation rules without environment (or with passive environment). In this case, the set of objects of the environment $\mathcal{E}$ is usually not defined in the tuple.

A *transition* of a P system Π is defined as a computational step of Π, passing from one configuration to the next one, and denoted by $C_t \Rightarrow_\Pi C_{t+1}$. A *computation* of a P system is a sequence of configurations such that a configuration C_{t+1} is always obtained from C_t by applying a computation step. C_0 is the initial configuration of Π.

A family of recognizer P systems with evolutional communication rules and creation rules solving a decision problem $X = (I_X, \theta_X)$ is a sequence $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ such that $\Pi(n)$ solves every instance of length n. For this purpose, a pair of two polynomial-time computable functions (cod, s) must be provided, such that $cod(u)$, for $u \in I_X$ is a multiset of objects from Σ representing the particular instance u of the problem X, and $s(u)$ is the *length* of the instance. As recognizer membrane systems, $\{\texttt{yes}, \texttt{no}\} \subseteq \Gamma$, representing the solution for each instance, and either an object $\texttt{yes}$ or an object $\texttt{no}$ (but not both) must appear in the environment only in the last step of computation.

In [28,29], the semantics applied are *maximalist* in the following sense: In each membrane, an arbitrary number of creation rules can be applied, and they do not interfere with the application of other types of rules. In [24,23], more restrictive semantics were introduced. When a creation rule is applied in a membrane h, no other rules can be applied in the same computational step. In the case of P systems with evolutional symport/antiport rules, either evolutional communication rules in a maximal parallel way or a single creation rule can be applied in a membrane in a transition, but not both at the same time. As a recognizer membrane system, all the computation of a recognizer P system with evolutional communication rules and creation rules halt and either an object `yes` or an object `no` (but not both) is sent to the environment at the last step of the computation.

The class of all recognizer P systems with evolutional symport/antiport rules (respectively, symport rules) and membrane creation is denoted by $\mathcal{CCEC}(k_1, k_2)$ (resp., $\mathcal{CCES}(k_1, k_2)$), where k_1 represents the maximal number of objects in the LHS of an evolutional communication rule (resp., evolutional symport rule) and k_2 represents the maximal number of objects in the RHS of an evolutional communication rule (resp., evolutional symport rule). The class of recognizer membrane systems of this type when environment plays a passive role is denoted by $\widehat{\mathcal{CCEC}}(k_1, k_2)$ (resp., $\widehat{\mathcal{CCES}}(k_1, k_2)$). Besides, $\mathcal{CCEC}(k)$ (respectively, $\widehat{\mathcal{CCEC}}(k)$) represents the class of recognizer P systems with evolutional communication rules with rules of length at most k and membrane creation (resp., without environment), where the length of evolutional communication rules in this case is the sum of the number of objects of both the LHS and the RHS of the rule. In the same way, $\mathcal{CCES}(k)$ (respectively, $\widehat{\mathcal{CCES}}(k)$) represent the evolutional symport-only counterparts of the previously described classes of P systems.

Then $\mathbf{PMC}_{\mathcal{R}}$ is the set of decision problems solvable by a uniform family of recognizer P systems from $\mathcal{R}$. The term uniform is used as in circuit complexity; instead of using one P system for each instance (non-uniform solutions), a P system is able to solve all the instances of the same length.

A more comprehensive introduction to the concepts of decision problems and the class of decision problems that can be solved by means of a family of membrane systems from $\mathcal{CCES}(k_1, k_2)$, as well as from the other classes mentioned above, can be extracted from [23,26,30]. The class of problems that can be solved in polynomial time by means of a uniform family of recognizer P systems with evolutional communication rules of length at most (k_1, k_2) (i.e., k_1 objects in the LHS of the rules, k_2 objects in the RHS of the rules) (respectively, (k)) and membrane creation is denoted by $\mathbf{PMC}_{\mathcal{CCES}(k_1,k_2)}$ (resp., $\mathbf{PMC}_{\mathcal{CCES}(k)}$), and the corresponding complexity class for membrane systems without environment is $\mathbf{PMC}_{\widehat{\mathcal{CCES}}(k_1,k_2)}$ (resp., $\mathbf{PMC}_{\widehat{\mathcal{CCES}}(k)}$).

4. An efficient solution to QSAT in $\widehat{\mathcal{CCES}}(1, 1)$

In this section, an efficient solution to QSAT problem by means of a uniform family of P systems $\mathbf{\Pi}$ from $\widehat{\mathcal{CCES}}(1, 1)$ with evolutional communication rules and creation rules. Each P system $\Pi(\langle n, p\rangle)$ from $\mathbf{\Pi}$ solves all instances from QSAT with n variables and p clauses.

Let $\varphi^* = \exists x_1 \forall x_2 \dots Q_n x_n \varphi(x_1, \dots, x_n)$ be an existential fully quantified formula associated with a Boolean formula $\varphi(x_1, \dots, x_n) \equiv C_1 \wedge \dots \wedge C_p$ in CNF, where each clause $C_j = l_{j,1} \vee \dots \vee l_{j,r_j}$, $Var(\varphi) = \{x_1, \dots, x_n\}$ and $l_{j,k} \in \{x_i, \neg x_i \mid 1 \le i \le n\}$ for all j and all k. Without loss of generality, we assume that existential and universal quantifiers are alternated. The problem QSAT can be described as follows: given a existential fully quantified formula φ^*, determine if it is true or false.

For each pair $n, p \in \mathbb{N}$, we consider a recognizer P system with evolutional communication rules of length $(1, 1)$ and creation rules

$$\Pi(\langle n, p\rangle) = (\Gamma, \Sigma, H, \mu, \mathcal{M}_{skin}, \mathcal{M}_{<0,\#>}, \mathcal{R}, i_{in}, i_{out})$$

that will solve all instances with n variables and p clauses, where the input multiset is defined as

$$cod(\varphi) = \{x_{i,j} | x_i \in C_j\} \cup \{\overline{x}_{i,j} | \neg x_i \in C_j\}$$

$$s(\varphi) = < n, p >$$

1. The working alphabet is defined as follows:
$\Gamma = \Sigma \cup \{\mathtt{yes}, \mathtt{no}, d', d'_t, d'_f, d''\} \cup$
$\{\alpha_i \,|\, 0 \le i \le n^2 \cdot p + 5n + 2p + 3\} \cup \{\alpha'_i \,|\, 0 \le i \le n^2 \cdot p + 5n + 2p + 4\} \cup$
$\{c_{i,j,r} \,|\, 1 \le i \le n, 1 \le j \le p, r \in \{t, f\}\} \cup$.
$\{d_{i,r} \,|\, 0 \le i \le n, r \in \{t, f\}\} \cup \{z_i, z_{i,t}, z_{i,f} \,|\, 1 \le i \le n\} \cup$
$\{x_{i',i,j,t}, \overline{x}_{i',i,j,f} \,|\, 1 \le i, i' \le n, 1 \le j \le p\}$
2. The input alphabet $\Sigma = \{x_{i,j}, \overline{x}_{i,j} \,|\, 1 \le i \le n, 1 \le j \le p\}$.
3. $H = \{skin, 1, \dots, p, yes, no, \#\} \cup \{< i, r > \,|\, 0 \le i \le n, r \in \{t, f\}\} \cup$.
$\{< i, Q, r, r' > \,|\, 0 \le i \le n, Q \in \{\exists, \forall\}, r, r' \in \{t, f\}\} \cup$
$\{< i, \# > \,|\, 0 \le i \le n^2 \cdot p + 5n + 2p + 4\}$
4. $\mu = [\,[\;\;]_{<0,\#>}\,]_{skin}$.
5. $\mathcal{M}_{skin} = \{z_{1,t}, z_{1,f}\}$, $\mathcal{M}_{<0,\#>} = \{\alpha_0, \alpha'_0\}$.
6. The set of rules $\mathcal{R}$:

6.1 Rules for the counter of the elements of $< 0, \# >$

$[\alpha_i \to [\alpha_{i+1}\,]_{<i+1,\#>}\,]_{<i,\#>}$ for $0 \le i \le n^2 \cdot p + 5n + 2p + 2$

$[\alpha'_i \to [\alpha'_{i+1}\,]_{<i+1,\#>}\,]_{<i,\#>}$ for $0 \le i \le n^2 \cdot p + 5n + 2p + 3$

$[\,[\alpha_{n^2\cdot p+5n+2p+3}\,]_{<i,\#>}\,]_{<i-1,\#>} \to [\alpha_{n^2+5n+2p+3}\,[\ \ ]_{<i,\#>}\,]_{<i-1,\#>}$
for $0 \le i \le n^2 \cdot p + 5n + 2p + 3$

$[\,[\alpha'_{n^2\cdot p+5n+2p+4}\,]_{<i,\#>}\,]_{<i-1,\#>} \to [\alpha'_{n^2+5n+2p+4}\,[\ \ ]_{<i,\#>}\,]_{<i-1,\#>}$
for $1 \le i \le n^2 \cdot p + 5n + 2p + 4$

$[\,[\alpha_{n^2+5n+2p+3}\,]_{<0,\#>}\,]_{skin} \to [\alpha_{n^2+5n+2p+3}\,[\ \ ]_{<0,\#>}\,]_{skin}$

$[\,[\alpha'_{n^2+5n+2p+4}\,]_{<0,\#>}\,]_{skin} \to [\alpha'_{n^2+5n+2p+4}\,[\ \ ]_{<0,\#>}\,]_{skin}$

6.2 Rules to return a positive answer

$[\alpha_{n^2+5n+2p+3}\,[\ \ ]_{yes}]_{skin} \to [\,[\alpha_{n^2+5n+2p+3}\,]_{yes}\,]_{skin}$

$[\alpha_{n^2+5n+2p+3} \to [\,\mathtt{yes}\,]_{\#}\,]_{yes}$

$[\,[\mathtt{yes}\,]_{\#}\,]_{yes} \to [\mathtt{yes}\,[\ \ ]_{\#}\,]_{yes}$

$[\,[\mathtt{yes}\,]_{yes}\,]_{skin} \to [\mathtt{yes}\,[\ \ ]_{yes}\,]_{skin}$

$[\,[\mathtt{yes}]_{skin}\,]_{env} \to [\mathtt{yes}\,[\ \ ]_{skin}\,]_{env}$

6.3 Rules to return a negative answer

$[\alpha'_{n^2+5n+2p+4} \to [\ \ ]_{no}\,]_{skin}$

$[\alpha_{n^2+5n+2p+3}\,[\ \ ]_{no}\,]_{skin} \to [\,[\alpha_{n^2+5n+2p+3}\,]_{no}\,]_{skin}$

$[\alpha_{n^2+5n+2p+3} \to [\,\mathtt{no}\,]_{\#}\,]_{no}$

$[\,[\mathtt{no}\,]_{\#}\,]_{no} \to [\mathtt{no}\,[\ \ ]_{\#}\,]_{no}$

$[\,[\mathtt{no}\,]_{no}\,]_{skin} \to [\mathtt{no}\,[\ \ ]_{no}\,]_{skin}$

$[\,[\mathtt{no}\,]_{skin}\,]_{env} \to [\mathtt{no}\,[\ \ ]_{skin}]_{env}$

6.4 Rules to generate the membrane structure

$[z_{1,r} \to [z_1\,d''\,]_{<1,r>}\,]_{skin}$ for $r \in \{t, f\}$

$[z_{i,r} \to [z_i\,d'\,]_{<i,r>}\,]_{<i-1,r'>}$ for $1 \le i \le n$, i odd, $r, r' \in \{t, f\}$

$[z_{i,r} \to [z_i\,d''\,]_{<i,r>}\,]_{<i-1,r'>}$ for $2 \le i \le n$, i even, $r, r' \in \{t, f\}$

$[z_i \to [z_{i+1,t}\,z_{i+1,f}\,]_{\#}\,]_{<i,r>}$ for $1 \le i < n$, $r \in \{t, f\}$

$[\,[z_{i,r}\,]_{\#}\,]_{<i,r>} \to [z_{i,r}[\ \ ]_{\#}\,]_{<i,r>}$ for $1 \le i \le n$, $r \in \{t, f\}$

6.5 Rules to check which clauses are satisfied

$[x_{i,j} \to [x_{1,i,j,t}\,x_{1,i,j,f}\,]_{\#}\,]_{skin}$ for $1 \le i \le n$, $1 \le j \le p$

$[x_{i',i,j,r} \to [x_{i'+1,i,j,t}\,x_{i'+1,i,j,f}\,]_{\#}\,]_{<i'+1,r>}$
for $1 \le i < i' \le n$, $1 \le j \le m$, $r \in \{t, f\}$

$[x_{i'+1,i,j,r}[\ \ ]_{<i'+1,r>}\,]_{<i',r'>} \to [\,[x_{i'+1,i,j,r}\,]_{<i'+1,r>}\,]_{<i',r'>}$
for $1 \le i < i' \le n$, $1 \le j \le p$, $r, r' \in \{t, f\}$

$[x_{i,i,j,t} \to [c_{i,j,t}\,c_{i,j,f}\,]_{\#}\,]_{<i,t>}$ for $1 \le i < n$, $1 \le j \le p$

$[\overline{x}_{i,i,j,f} \to [c_{i,j,t}\,c_{i,j,f}\,]_{\#}\,]_{<i,f>}$ for $1 \le i < n$, $1 \le j \le p$

$[x_{n,n,j,t} \to [c_{n,j,t}\,]_{\#}\,]_{<n,t>}$ for $1 \le j \le p$

$[\overline{x}_{n,n,j,f} \to [c_{n,j,f}\,]_{\#}\,]_{<n,f>}$ for $1 \le j \le p$

$[\,[c_{i,j,r}\,]_{\#}\,]_{<i,r'>} \to [c_{i,j,r}\,[\ \ ]_{\#}\,]_{<i,r'>}$
for $1 \le i \le n$, $1 \le j \le p$, $r, r' \in \{t, f\}$

$[c_{i,j,r}\,[\ \ ]_{<i+1,r>}\,]_{<i,r>} \to [\,[c_{i,j,r}\,]_{<i+1,r>}\,]_{<i,r>}$
for $1 \le i < n$, $1 \le j \le p$, $r \in \{t, f\}$

$[c_{i,j,r} \to [c_{i+1,j,t}\,c_{i+1,j,f}\,]\#]_{<i',r>}$
for $1 \le i' \le i < n$, $1 \le j \le p$, $r \in \{t, f\}$

6.6 Rules to check if **all** clauses are satisfied

$[z_n \to [d_0]_{\#}\,]_{<n,r>}$ for $r \in \{t, f\}$

$[c_{n,j,r} \to [\ \ ]_j\,]_{<n,r'>}$ for $1 \le j \le p$, $r, r' \in \{t, f\}$

$[d_j\,[\ \ ]_{j+1}\,]_{<n,r>} \to [\,[d_j]_{j+1}\,]_{<n,r>}$ for $0 \le j < p$, $r \in \{t, f\}$

$[c_{n,j,r} \to [\ \ ]_j\,]_{<n,r>}$ for $1 \le j \le p$, $r \in \{t, f\}$

$[d_j \to [d_{j+1}\,]_{\#}\,]_{i+1}$ for $0 \le j < p$

$[\,[d_j\,]_{\#}\,]_j \to [d_j\,[\ \ ]_{\#}\,]_j$ for $0 \le j \le p$, $r \in \{t, f\}$

$[\,[d_j\,]_j\,]_{<n,r>} \to [d_j\,[\ \ ]_j\,]_{<n,r>}$ for $0 \le j \le p$, $r \in \{t, f\}$

$[d_p \to [d_{n,r}\,]_{\#}\,]_{<n,r>}$ for $r \in \{t, f\}$

$[\,[d_{n,r}\,]_{\#}\,]_{<n,r>} \to [d_{n,r}\,[\ \ ]_{\#}\,]_{<n,r>}$ for $r \in \{t, f\}$

6.7 Rules to check if quantifiers are satisfied

$[\,[d_{i,r}\,]_{<i,r>}\,]_{<i-1,r'>} \to [d_{i,r}\,[\ \ ]_{<i,r>}\,]_{<i-1,r'>}$
for $1 \le i \le n$, $r, r' \in \{t, f\}$

$[d_{i+1,r} \to [\ \ ]_{<i,\exists,r,r'>}\,]_{<i,r'>}$ for $1 \le i \le n$, i odd, $r, r' \in \{t, f\}$

$[d_{i+1,r} \to [\ \ ]_{<i,\forall,r,r'>}\,]_{<i,r'>}$ for $2 \le i \le n$, i even, $r, r' \in \{t, f\}$

$[d'[\ ]_{<i,\exists,r,r'>}]_{<i,r'>} \rightarrow [[d']_{<i,\exists,r,r'>}]_{<i,r'>}$
for $1 \le i \le n$, i odd, $r, r' \in \{t, f\}$
$[d' \rightarrow [d_{i,r'}]_{\#}]_{<i,\exists,r,r'>}$ for $1 \le i \le n$, i odd, $r, r' \in \{t, f\}$
$[[d_{<i,r'>}]_{\#}]_{<i,\exists,r,r'>} \rightarrow [d_{<i,r'>}[\ ]_{\#}]_{<i,\exists,r,r'>}$
for $1 \le i \le n$, i odd, $r, r' \in \{t, f\}$
$[[d_{<i,r'>}]_{<i,\exists,r,r'>}]_{<i,r'>} \rightarrow [d_{<i,r'>}[\ ]_{<i,\exists,r,r'>}]_{<i,r'>}$
for $1 \le i \le n$, i odd, $r, r' \in \{t, f\}$
$[d''[\ ]_{<i,\forall,r,r'>}]_{<i,r'>} \rightarrow [[d'']_{<i,\forall,r,r'>}]_{<i,r'>}$
for $2 \le i \le n$, i even, $r, r' \in \{t, f\}$
$[d'' \rightarrow [d'_r]_{\#}]_{<i,\forall,r,r'>}$ for $2 \le i \le n$, i even, $r, r' \in \{t, f\}$
$[[d'_r]_{\#}]_{<i,\forall,r,r'>} \rightarrow [d'_r[\ ]_{\#}]_{<i,\forall,r,r'>}$ for $1 \le i \le n$, $r, r' \in \{t, f\}$
$[[d'_r]_{<i,\forall,r,r'>}]_{<i,r'>} \rightarrow [d'_r[\ ]_{<i,\forall,r,r'>}]_{<i,r'>}$
for $1 \le i \le n-1$, $r, r' \in \{t, f\}$
$[d'_{r''}[\ ]_{<i,\forall,r,r'>}]_{<i,r'>} \rightarrow [[d']_{<i,\forall,r,r'>}]_{<i,r'>}$
for $2 \le i \le n$, i even, $r, r', r'' \in \{t, f\}, r \ne r''$
$[d'_{r''} \rightarrow [d_{i,r'}]_{\#}]_{<i,\forall,r,r'>}$ for $2 \le i \le n$, i even, $r, r', r'' \in \{t, f\}, r \ne r''$
$[[d_{i,r'}]_{\#}]_{<i,\forall,r,r'>} \rightarrow [d_{i,r'}[\ ]_{\#}]_{<i,\forall,r,r'>}$ for $1 \le i \le n$, $r, r' \in \{t, f\}$
$[[d_{i,r'}]_{<i,Q,r,r'>}]_{<i,r'>} \rightarrow [d_{i,r'}[\ ]_{<i,Q,r,r'>}]_{<i,r'>}$
for $1 \le i \le n-1$, $r, r' \in \{t, f\}$, $Q \in \{\exists, \forall\}$
$[d_{1,r} \rightarrow [\ ]_{yes}]_{skin}$ for $r \in \{t, f\}$

7. $i_{in} = skin$.
8. $i_{out} = env$.

4.1. An overview of the computation

Let φ^* with $n \ge 2$ variables and $p \ge 2$ clauses be an instance of QSAT as defined before. Let us suppose that the number of variables, n, and the number of clauses, p, is at least 2. We consider a polynomial encoding (cod, s) from QSAT in $\mathbf{\Pi}$ as follows: for each formula φ associated to an existential fully quantified formula φ^* with n variables and p clauses, $s(\varphi) = \langle n, p \rangle$ and $cod(\varphi) = \{x_{i,j} \mid x_i \in C_j\} \cup \{\overline{x}_{i,j} \mid \neg x_i \in C_j\}$.

The proposed solution follows a brute force scheme of recognizer P systems with evolutional communication rules and membrane creation without environment, and it consists of the following stages.

4.1.1. Generation and first checking stage

By applying rules from 6.4, a membrane structure is generated. In some sense, it reminds a binary tree, but having some "auxiliary" membranes, labelled by #, used to generate the objects $z_{i+1,t}$ and $z_{i+1,f}$. The truth assignment of each branch is determined by the labels of the membranes forming that branch. Besides, using rules from 6.5, objects from $cod(\varphi)$ will be passed throughout the membrane structure in such a way that in the level i, the i-th variable will be checked and, if the corresponding truth assignment makes true a literal in a clause j, then objects $c_{i,j,t}$ and $c_{i,j,f}$ will appear, that will be passed by the membranes up to a membrane labelled by $<n, r>$, for $r \in \{t, f\}$. In this stage, the *minimalist* creation rules semantics take special importance. Since two creation rules cannot be applied at the same time in the same membrane, the system has to make the objects wait the correct amount of time in order to go up to the different "levels" of the system at the same time. This protocol makes the system slower than it would if *maximalist* semantics are chosen. On the other hand, the selected semantics make it easier to control that, in each computational step $t+1$, at most $2 \cdot m$ new membranes can be created, where m is the number of membranes present in the configuration $\mathcal{C}_t$. This stage takes $2n^2 \cdot 2p$ steps.

4.1.2. Second checking stage

Rules from 6.6 are in charge of checking whether all the clauses are satisfied in a truth assignment. For that, if there exists an object $c_{n,j,r}$ in a membrane labelled by $<n, r>$, it means that the corresponding truth assignment makes true the clause j. Therefore, a membrane labelled by j is created within such a membrane $<n, r>$. Object d_0 will go through all membranes within the membrane $<n, r>$, creating a "auxiliary" membrane within them and passing to the next one, possibly arriving to membrane p. In that case, object d_p creates a new auxiliary membrane with an object $d_{n,r}$, that will be useful in the next stage. This stage takes $3p + 5$ steps.

4.1.3. Quantifier checking stage

If an object $d_{i,r}$ ($r \in \{t, f\}$) appears in a membrane, then the quantifier in this level is checked. Depending on the parity of the level, either a universal or an existential quantifier is checked. In the generation stage, objects d' and d'' were created for this purpose. On the one hand, when an existential quantifier is being checked, an object d' will exist in such a membrane, and will change into an object $d_{i-1,r}$ if and only if there is at least one object $d_{i,r}$ that has created a membrane $<i, \exists, r, r'>$. On the other hand, when a universal quantifier is to be checked, an object d'' will be present in such a membrane, and will change into an object $d_{i-1,r}$ if and only if there are two objects $d_{i,r}$ that have created two membranes labelled by

$< i, \forall, r, r' >$. These objects will reach the skin membrane giving way to the last stage. This whole stage is computed by the application of rules from 6.6. This stage takes $8n - 6$ steps if n is even and $8n$ steps if n is odd.

4.1.4. Output stage

Counters α and α' are used in this stage in order to know if an object $d_{1,r}$ has reached the skin membrane. In such a case, a membrane labelled by *yes* will be created, and when object $\alpha_{n^2 \cdot p+5n+2p+3}$ reaches the skin membrane, it will go into membrane *yes* and will change into an object $\texttt{yes}$ that will be sent to the environment. In the case that object $d_{1,r}$ does not appear in the skin membrane, object $\alpha'_{n^2 \cdot p+5n+2p+4}$ will generate a membrane labelled by *no*, that will make the counter $\alpha_{n^2 \cdot p+5n+2p+3}$ change into an object $\texttt{no}$, that will be sent to the environment. Rules from 6.2 and 6.3 are the responsible of this stage, that takes $p + 8$ steps if n is even and $p + 9$ steps if n is odd.

4.2. Formal verification

Next, we prove that $\mathbf{\Pi}$ provides a polynomial time and uniform solution to $\texttt{QSAT}$.

Theorem 1. $\texttt{QSAT} \in \mathbf{PMC}_{\mathcal{CCES}(1,1)}$

The family of P systems $\mathbf{\Pi}$ verifies the following:

- Every system of the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines because, for each $n, p \in \mathbb{N}$, the rules of $\Pi(\langle n, p\rangle)$ of the family are recursively defined from $n, p \in \mathbb{N}$ and the amount of resources needed to build an element of the family is of a polynomial order in n and p.
- The pair (cod, s) of polynomial-time computable functions defined fulfill the following: for each formula φ^* of $\texttt{QSAT}$, $s(\varphi^*)$ is a natural number, $cod(\varphi^*)$ is an input multiset of the system $\Pi(s(\varphi^*))$ and for each $\varphi \in I_{QSAT}$, $s^{-1}(\varphi^*)$ is a finite set.
- The family $\mathbf{\Pi}$ is polynomially bounded: for each input formula φ^* of $\texttt{QSAT}$, the P system $\Pi(\langle n, p\rangle)$ with input $cod(\varphi^*)$ takes k steps, with $k \le 2n^2 + 10n + 4p + 12$ steps, being n the number of variables and p the number of clauses of φ^*.
- The family $\mathbf{\Pi}$ is sound with regard to (X, cod, s): for each formula φ^*, if the computation of $\Pi(s(\varphi^*)) + cod(\varphi^*)$ is an accepting computation, then φ^* is true.
- The family $\mathbf{\Pi}$ is complete with regard to (X, cod, s): for each input formula φ^* such that it is satisfiable, the computation of $\Pi(s(\varphi^*)) + cod(\varphi^*)$ is an accepting computation.

Corollary 1. $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\widehat{\mathcal{CCES}}(1,1)}$

Proof. It suffices to know that $\texttt{QSAT}$ is a **PSPACE**-complete problem, $\texttt{QSAT} \in \mathbf{PMC}_{\widehat{\mathcal{CCES}}(1,1)}$ and the class $\mathbf{PMC}_{\widehat{\mathcal{CCES}}(1,1)}$ (in general, $\mathbf{PMC}_{\mathcal{R}}$) is closed under polynomial-time reduction and under complementation [30].

Corollary 2. $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\widehat{\mathcal{CCES}}(2)}$

Proof. It suffices to see that, in the family $\mathbf{\Pi}$ the maximum amount of objects in an evolutional communication rule is 2.

5. Conclusions and future work

In this work, a previous result has been improved, passing from being able to solve a **NP**-complete problem ($\texttt{SAT}$) by means of a family of membrane systems from $\mathcal{CCEC}(2,2)$ to solving a **PSPACE**-complete problem ($\texttt{QSAT}$) by means of a family of membrane systems from $\widehat{\mathcal{CCES}}(1,1)$. This is a demonstration of the power of creation rules, showing that rules with a minimal number of objects both in the LHS and in the RHS are enough to reach presumed efficiency; that is, to solve problems supposed not to be solvable in polynomial time by a Turing machine. The use of division rules and separation rules with this length of evolutional symport/antiport rules give tissue P systems the power to efficiently solve only problems from **P** [18,19]. An interesting research work would be to investigate the same result for tissue-like membrane systems.

From the beginning, creation rules have been only used in cell P systems, because of the biological inspiration of the use of parts of a membrane to create a new membrane within it, but using creation rules in tissue P systems, where new cells would be created in the environment, and not in the cell itself, would be an interesting research line.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by “FEDER/Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación/ _Proyecto (TIN2017-89842-P)” - MABICAP. D. Orellana-Martín also acknowledges Contratación de Personal Investigador Doctor. (Convocatoria 2019) 43 Contratos Capital Humano Línea 2. Paidi 2020, supported by the European Social Fund and Junta de Andalucía.

References

[1] Gh. Păun, Computing with membranes, Tech. Rep., Turku Centre for Computer Science, 1998.
[2] C. Martín-Vide, G. Păun, J. Pazos, A. Rodríguez-Patón, Tissue P systems, Theor. Comput. Sci. 296 (2) (2003) 295–326.
[3] G. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, Tissue P systems with cell division, Int. J. Comput. Commun. Control 3 (3) (2008) 295–303.
[4] M. Ionescu, Gh. Păun, T. Yokomori, Spiking neural P systems, Fundam. Inform. 71 (2, 3) (2006) 279–308.
[5] A. Păun, Gh. Păun, The power of communication: P systems with symport/antiport, New Gener. Comput. 20 (3) (2002) 295–305.
[6] Gh. Păun, P systems with active membranes: attacking NP-complete problems, J. Autom. Lang. Comb. 6 (1) (2001) 75–90.
[7] L. Pan, T.-O. Ishdorj, P systems with active membranes and separation rules, J. Univers. Comput. Sci. 10 (5) (2004) 630–649.
[8] M. Mutyam, K. Krithivasan, P systems with membrane creation: universality and efficiency, in: Proceedings of the Third International Conference on Machines, Computations, and Universality, MCU '01, Springer-Verlag, Berlin, Heidelberg, 2001, pp. 276–287.
[9] P. Sosík, P systems attacking hard problems beyond NP: a survey, J. Membr. Comput. 1 (09 2019).
[10] A. Alhazov, C. Martín-Vide, L. Pan, Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes, Fundam. Inform. 58 (2) (2003) 67–77.
[11] A. Alhazov, L. Pan, Gh. Păun, Trading polarizations for labels in P systems with active membranes, Acta Inform. 41 (2–3) (2004) 111–144.
[12] P. Sosík, A. Rodríguez-Patón, Membrane computing and complexity theory: a characterization of PSPACE, J. Comput. Syst. Sci. 73 (1) (2007) 137–152.
[13] B. Song, X. Zeng, M. Jiang, M.J. Pérez-Jiménez, Monodirectional tissue P systems with promoters, IEEE Trans. Cybern. 51 (2021) 438–450.
[14] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Solving QSAT in sublinear depth, CoRR, arXiv:1902.03879 [abs], 2019, arXiv:1902.03879, http://arxiv.org/abs/1902.03879.
[15] T. Ishdorj, A. Leporati, L. Pan, X. Zeng, X. Zhang, Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources, Theor. Comput. Sci. 411 (25) (2010) 2345–2358, https://doi.org/10.1016/j.tcs.2010.01.019.
[16] A.E. Porreca, A. Leporati, G. Mauri, C. Zandron, P systems with elementary active membranes: beyond NP and coNP, in: M. Gheorghe, T. Hinze, G. Paun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing - 11th International Conference, CMC 2010, Jena, Germany, August 24–27, 2010, in: Lecture Notes in Computer Science, vol. 6501, Springer, 2010, pp. 338–347. Revised Selected Papers.
[17] A. Alhazov, R. Freund, On the efficiency of P systems with active membranes and two polarizations, in: G. Mauri, G. Paun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, 5th International Workshop, WMC 2004, Milan, Italy, June 14–16, 2004, in: Lecture Notes in Computer Science, vol. 3365, Springer, 2004, pp. 146–160. Revised Selected and Invited Papers.
[18] B. Song, C. Zhang, L. Pan, Tissue-like P systems with evolutional symport/antiport rules, Inf. Sci. 378 (2017) 177–193.
[19] L. Pan, B. Song, L. Valencia-Cabrera, M.J. Pérez-Jiménez, S.F. Hafstein, The computational complexity of tissue P systems with evolutional symport/antiport rules, Complexity 2018 (Jan. 2018).
[20] D. Orellana-Martín, L. Valencia-Cabrera, B. Song, L. Pan, M.J. Pérez-Jiménez, Narrowing frontiers with evolutional communication rules and cell separation, in: Proceedings of the 16th Brainstorming Week on Membrane Computing, Sevilla, Spain, 2018, pp. 123–162.
[21] D. Orellana-Martín, L. Valencia-Cabrera, B. Song, L. Pan, M.J. Pérez-Jiménez, Tuning frontiers of efficiency in tissue P systems with evolutional communication rules, Complexity 2021 (Apr. 2021).
[22] B. Song, K. Li, X. Zeng, Monodirectional evolutional symport tissue P systems with promoters and cell division, IEEE Trans. Parallel Distrib. Syst. (2021).
[23] B. Song, K. Li, D. Orellana-Martín, L. Valencia-Cabrera, M.J. Pérez-Jiménez, Cell-like P systems with evolutional symport/antiport rules and membrane creation, Inf. Comput. 275 (2020) 104542.
[24] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, M.J. Pérez-Jiménez, Membrane creation in polarizationless P systems with active membranes, Fundam. Inform. 171 (2020) 297–311.
[25] B. Song, K. Li, D. Orellana-Martín, M.J. Pérez-Jiménez, I. Pérez-Hurtado, A survey of nature-inspired computing: membrane computing, ACM Comput. Surv. 54 (2021) 1–31.
[26] Gh. Păun, G. Rozenberg, A. Salomaa, The Oxford Handbook of Membrane Computing, Oxford University Press, Inc., USA, 2010.
[27] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages. 3 vols., Springer-Verlag, Berlin, Heidelberg, 1997.
[28] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero, A linear solution of subset sum problem by using membrane creation, in: J. Mira, J.R. Álvarez (Eds.), Mechanisms, Symbols, and Models Underlying Cognition, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 258–267.
[29] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero, A linear solution for QSAT with membrane creation, in: R. Freund, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 241–252.
[30] M.J. Péerez-Jiménez, Álvaro Romero-Jiménez, F. Sancho-Caparrini, Complexity classes in models of cellular computing with membranes, Nat. Comput. 2 (3) (2003) 265–285.