Membrane creation and symport/antiport rules solving QSAT

David Orellana-Martín[1] · Luis Valencia-Cabrera[1,2] · Mario J. Pérez-Jiménez[1,2]

Abstract

In Membrane Computing, different variants of devices can be found by changing both syntactical and semantic ingredients. These devices are usually called membrane systems or P systems, and they recall the structure and behavior of living cells in the nature. In this sense, rules are introduced as a way for objects to interact with membranes, giving P systems the ability to solve computational problems. Some of these rules, as division, separation and creation rules are inspired by the membrane division through the mitosis process or new membranes are created through gemmation. These rules seem to be crucial in the path to solve computationally hard problems. In this work, creation rules are used in classical P systems with symport/antiport rules, where objects travel through membranes without changing to achieve enough computational power to efficiently solve **PSPACE**-complete problems. More precisely, a solution to the QSAT problem is given by means of a uniform family of these systems. This paper was originally submitted to the International Conference on Membrane Computing 2021.

Keywords Membrane computing · Membrane creation · QSAT · Computational complexity theory

1 Introduction

Membrane Computing, since its beginnings [1], has covered a wide spectrum of applications, from computability theory [2] and computational complexity theory [3] to pandemics [4] and engineering [5]. The model is inspired in the structure and behavior of living cells and the chemical reactions occurring within them. From the beginning, solving computationally hard problems has been a hot topic in this area [6], being one of the most studied fields with membrane systems.

Cell-like P systems can be seen as a hierarchical structure of membranes that let chemical compounds pass by them. Making an abstraction of integral membrane proteins and their role in the transport of molecules, symport/antiport rules let objects move to one region to other one or interchange them between two regions [7]. Usually, P systems are found to be Turing complete [8], but from the point of view of the computational complexity theory, they can only solve problems from the class **P** [9]. To increase the efficiency of these systems, different types of rules have been introduced: division rules [6], separation rules [10] and creation rules [11] are three of the protocols that have been implemented in membrane computing with this purpose. The latter uses a single object and transforms it into a new membrane.

In the framework of cell-like membrane systems, creation rules have been used lately besides evolutional communication rules in [12] to solve the SAT problem by means of a family of recognizer P systems with evolutional communication rules and creation rules. In [13], an efficient solution to QSAT is given by means of a family of recognizer polarizationless P systems with active membranes and membrane creation, and in [14], an efficient solution to QSAT is given by means of a family of P systems with evolutional symport/antiport rules of length (1, 1) and creation rules. Creation rules have not been used in P systems with classical symport/antiport rules. In this work, we try to replicate the results of the latter paper using recognizer P systems with classical symport/antiport rules with a minimal amount of objects per communication rule. This is a really important

✉ David Orellana-Martín
dorellana@us.es

Luis Valencia-Cabrera
lvalencia@us.es

Mario J. Pérez-Jiménez
marper@us.es

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

2 SCORE Laboratory, I3US, Universidad de Sevilla, Avda. Reina Mercedes, Sevilla 41012, Sevilla, Spain

result, since evolutional communication rules are intrinsically changing the nature of the objects, and thus both in a software simulation and in real-life implementations, it would need to implement these changes in some sense, leading to a more complex implementation. That is why the use of classical symport/antiport rules is really interesting in order to reduce the "number of ingredients" (we could think that the evolution part of evolutional symport/antiport rules is an ingredient) of the P systems using these rules.

The rest of the work is organized as follows: Section 2 is devoted to introduce some formal language and set theory concepts used later through the paper. In Sect. 3, the definition of recognizer P systems with symport/antiport 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 symport/antiport rules of length at most 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 [15, 16].

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

A *multiset* over Γ is a pair (Γ, f) where $f : \Gamma \to \mathbb{N}$ is a mapping from Γ to the set of natural numbers $\mathbb{N}$. Let $m_1 = (\Gamma_1, f_1), m_2 = (\Gamma_2, f_2)$ 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 $(f_1 + f_2)(x) = f_1(x) + f_2(x)$. The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is the defined as

$$(m_1 \setminus m_2)(x) = \begin{cases} f_1(x) - f_2(x) & \text{if } f_1(x) \geq f_2(x) \\ 0 & \text{if } f_1(x) < f_2(x). \end{cases}$$

The empty multiset is denoted by $\emptyset$, and the set of all finite multisets over Γ is denoted by $M_f(\Gamma)$.

The size of the set u is given by the total number of objects in u, and it is denoted by $\mid u \mid$.

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

3 Recognizer P systems with symport/antiport rules and creation rules

In this section, a definition of recognizer P systems with symport/antiport rules and creation rules is given, and both the syntax and semantics are recalled.

Definition 1 A recognizer P system with symport/antiport rules and membrane creation of degree $q \geq 1$ is a tuple

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

where

1. Γ, Σ and $\mathcal{E}$ are finite alphabets of objects, where $\Sigma, \mathcal{E} \subseteq \Gamma, \Sigma \cap \mathcal{E} = \emptyset$;
2. H is a finite set of labels;
3. μ is a membrane structure whose elements are injectively labeled 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:
   - Symport rules:
     - $(u, in) \in \mathcal{R}_i$, where $u \in M_f(\Gamma)$, except if i is the skin membrane, where $u \in M_f(\Gamma) \wedge u \notin M_f(\mathcal{E})$ (send-in rules);
     - $(u, out) \in \mathcal{R}_i$, where $u \in M_f(\Gamma)$ (send-out rules);
   - Antiport rules:
     - $(u, out; v, in) \in \mathcal{R}_i$, where $u, v \in M_f(\Gamma)$;
   - Creation rules: $[a \to [u]_i]_j$, where $i, j \in H, i \notin \{skin, i_{out}\}$, where *skin* is the label of the skin membrane, $a \in \Gamma, u \in M_f(\Gamma)$;
6. $i_{in} \in H$ is the label of the *input membrane*;
7. $i_{out} = env$ is the label of the *output zone*, in this case, the environment.

A *configuration* C_t of a P system with symport/antiport 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. We use the term region i to refer to a membrane if $i \in H$ and to the environment if $i = env$. We can suppose that in each moment, there is an arbitrary number of objects from $\mathcal{E}$ in the environment. Let m be the input multiset encoding the corresponding instance of a problem. The initial configuration is of such a P system Π is $\mathcal{C}_0 = (\mu, \mathcal{M}_1, \dots, \mathcal{M}_{i_{in}} + m, \dots, \mathcal{M}_q; \emptyset)$.

A symport rule $(u, in) \in \mathcal{R}_i$, called send-in rule, can be applied to a configuration C_t if there exists a membrane labeled by i, and the parent region contains a multiset of objects u. When applying such a rule, the multiset of objects u is consumed from the parent region and a multiset of objects u is produced in the membrane i in the next configuration.

A symport rule $(u, out) \in \mathcal{R}_i$, called send-out rule, can be applied to a configuration C_t if there exists a membrane labeled by i that contains a multiset of objects u. 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 parent region.

An antiport rule $(u, out; v, in) \in \mathcal{R}_i$ can be applied to a configuration C_t if there exists a membrane labeled by i that contains a multiset of objects u, and whose parent region contains a multiset of objects v. When applying such a rule, the multisets of objects u and v are consumed from the membrane i and its parent region, respectively, and multisets v and u are produced in the membrane i and its parent region, respectively.

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

A recognizer P system with symport/antiport rules and creation rules that does not send objects from the environment to the system is said to be a P system with symport/antiport rules and creation rules without 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.

In [17, 18], 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 [12, 13], 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 this case, dealing with P systems with symport/antiport rules and membrane creation, either 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. That is, if a creation rule $[\, a \to [\, b \,]\,]_h$ is applied, then neither other creation rules in that membrane h nor other symport/antiport rules from $\mathcal{R}_h$. In this paper, the latter semantics, called *minimalist* semantics are going to be used. As a recognizer membrane system, all the computations of a recognizer P system with 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 length of a symport rule $r \equiv (u, in)$ or $r \equiv (u, out)$ is given by the number of objects in multiset u; that is, it is equal to $|\, u \,|$. The length of an antiport rule $r \equiv (u, out; v, in)$ is given by the total number of objects in the rule; that is, it is equal to $|\, u \,| + |\, v \,|$. Let us denote the length of a rule r by $l(r)$.

The class of all recognizer P systems with symport/antiport rules and membrane creation of degree q is denoted by $\mathcal{CCC}(k)$ with *minimalistic* semantics, where k represents the maximal number of objects in a communication rule; that is, $k = max(l(r) \mid r \in \mathcal{R}_i, 1 \le i \le q)$. The class of recognizer membrane systems of this type when environment plays a passive role; that is, when no objects can be sent from the environment to the P system itself, is denoted by $\widehat{\mathcal{CCC}}(k)$.

All the concepts of a decision problem and the class of decision problems that can be solved by means of a uniform family of membrane systems from $\mathcal{CCC}(k)$ can be extracted from [12, 15, 19]. The class of problems that can be solved efficiently (i.e., in polynomial time with respect to the input) by means of a uniform family of recognizer P systems with symport/antiport rules of length at most k and membrane creation with environment (respectively, without environment) is denoted by $\mathbf{PMC}_{\mathcal{CCC}(k)}$ (resp., $\mathbf{PMC}_{\widehat{\mathcal{CCC}}(k)}$).

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

In this section, we give an efficient solution to the QSAT problem by means of a uniform family $\mathbf{\Pi}$ of P systems from $\widehat{\mathcal{CCC}}(1)$. Let $t = \langle n, p \rangle$. Each P system $\Pi(t), t \in \mathbb{N}$, from $\mathbf{\Pi}$ solves all instances from QSAT with n variables and p clauses.

For each pair $n, p \in \mathbb{N}$, we consider a recognizer P system with symport/antiport rules of length 1 and creation rules

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

that will solve all instances with n variables and p clauses, where $\varphi^* = \exists x_1 \forall x_2 \dots Q_n x_n \varphi(x_1, \dots, x_n)$ is 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 $l_{j,k} \in \{x_i, \neg x_i \mid 1 \le i \le n\}$. 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 with an existential fully quantified formula φ^* with n variables and p clauses, $s(\varphi) = \langle n,p\rangle$ and $cod(\varphi) = \{x_{i,j} \mid x_i \in C_j\} \cup \{\overline{x}_{i,j} \mid \neg x_i \in C_j\}$ and $s(\varphi) = \langle n,p\rangle$

1. The working alphabet is defined as follows:

$$\Gamma = \Sigma \cup \{\texttt{yes}, \texttt{no}, d', d'_t, d'_f, d''\} \cup$$
$$\{\alpha_i \mid 0 \le i \le n^2 \cdot p + 5n + 2p + 3\} \cup$$
$$\{\alpha'_i \mid 0 \le i \le n^2 \cdot p + 5n + 2p + 4\} \cup$$
$$\{c_{i,j,r} \mid 1 \le i \le n, 1 \le j \le p, r \in \{t,f\}\} \cup$$
$$\{d_{i,r} \mid 0 \le i \le n, r \in \{t,f\}\} \cup$$
$$\{z_i, z_{i,t}, z_{i,f} \mid 1 \le i \le n\} \cup$$
$$\{x_{i',i,j,t}, \overline{x}_{i',i,j,f} \mid 1 \le i, i' \le n, 1 \le j \le p\}$$

.

2. The input alphabet $\Sigma = \{x_{i,j}, \overline{x}_{i,j} \mid 1 \le i \le n, 1 \le j \le p\}$.

$$H = \{skin, 1, \dots, p, yes, no, \#\} \cup$$
$$\{\langle i,r\rangle \mid 0 \le i \le n, r \in \{t,f\}\} \cup$$
$$\{\langle i,Q,r,r'\rangle \mid 0 \le i \le n, Q \in \{\exists,\forall\}, r, r' \in \{t,f\}\} \cup$$
$$\{\langle i,\#\rangle \mid 0 \le i \le n^2 \cdot p + 5n + 2m + 4\}$$

3. .
4. $\mu = [\,[\ \ ]_\alpha\,]_{skin}$.
5. $\mathcal{M}_{skin} = \{z_{1,t}, z_{1,f}\}, \mathcal{M}_\alpha = \{\alpha_0, \alpha'\}$.
6. The set of rules $\mathcal{R}$:

6.1 Rules for the counter of the elements of membrane α; let $k = n^2p + 13n + 5p + 2$

$[\,\alpha_i \to [\,\alpha_{i+1}\,]_\#\,]_\alpha$ for $0 \le i \le \lfloor k/2 \rfloor - 1$
$(\alpha_i, out) \in \mathcal{R}_\#$ for $0 \le i \le \lfloor k/2 \rfloor$
$[\,\alpha_{\lfloor k/2 \rfloor} \to [\,\alpha\,]_{\alpha'}\,]_\alpha$
$(\alpha, out) \in \mathcal{R}_{\alpha'}$
$(\alpha, out) \in \mathcal{R}_\alpha$
$(\alpha', in) \in \mathcal{R}_{\alpha'}$
$[\,\alpha \to [\,\alpha''\,]_\#\,]_{\alpha'}$
$(\alpha'', out) \in \mathcal{R}_\#$
$(\alpha'', out) \in \mathcal{R}_{\alpha'}$
$(\alpha'', out) \in \mathcal{R}_\alpha$

6.2 Rules to return a positive answer

$[\,D_{1,r} \to [\,\#\,]_{yes}\,]_{skin}$ for $r \in \{t,f\}$
$(\alpha, in) \in \mathcal{R}_{yes}$
$[\,\alpha \to [\,\texttt{yes}\,]_\#\,]_{yes}$
$(\texttt{yes}, out) \in \mathcal{R}_\#$
$(\texttt{yes}, out) \in \mathcal{R}_{yes}$
$(\texttt{yes}, out) \in \mathcal{R}_{skin}$

6.3 Rules to return a negative answer

$[\,\alpha'' \to [\,\#\,]_{no}\,]_{skin}$
$(\alpha, in) \in \mathcal{R}_{no}$
$[\,\alpha \to [\,\texttt{no}\,]_\#\,]_{no}$
$(\texttt{no}, out) \in \mathcal{R}_\#$
$(\texttt{no}, out) \in \mathcal{R}_{no}$
$(\texttt{no}, out) \in \mathcal{R}_{skin}$

6.4 Rules to generate the membrane structure

$[\,z_{1,r} \to [\,z_1\, d_\forall\,]_{\langle 1,r\rangle}\,]_{skin}$ for $r \in \{t,f\}$

$[\,z_{i,r} \to [\,z_1\, d_\exists\,]_{\langle i,r\rangle}\,]_{\langle i-1,r'\rangle}$ for $3 \le i \le n-1$, i odd , $r, r' \in \{t,f\}$

$[\,z_{1,r} \to [\,z_1\, d''\,]_{\langle 1,r\rangle}\,]_{skin}$ for $r \in \{t,f\}$
$[\,z_{i,r} \to [\,z_i\, d'\,]_{\langle i,r\rangle}\,]_{\langle i-1,r'\rangle}$ for $2 \le i \le n$, i odd , $r, r' \in \{t,f\}$
$[\,z_{i,r} \to [\,z_i\, d''\,]_{\langle i,r\rangle}\,]_{\langle i-1,r'\rangle}$ for $2 \le i \le n$, i even , $r, r' \in \{t,f\}$
$[\,z_i \to [\,z_{i+1,t}\, z_{i+1,f}\,]_\#\,]_{\langle i,r\rangle}$ for $1 \le i \langle n, r \in \{t,f\}$
$(z_{i,r}, out) \in \mathcal{R}_\#$ 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 $1 \le i \le n, 1 \le j \le p$
$[\,\overline{x}_{i,j} \to [\,\overline{x}_{1,i,j,t}\, \overline{x}_{1,i,j,f}\,]_\#\,]_{skin}$ for $1 \le i \le n, 1 \le j \le p$
$[\,x_{i',i,j,r} \to [\,x_{i'+1,i,j,t}\, x_{i'+1,i,j,f}\,]_\#\,]_{\langle i'+1,r\rangle}$
for $1 \le i' < i \le n, 1 \le j \le p, r \in \{t,f\}$
$[\,\overline{x}_{i',i,j,r} \to [\,\overline{x}_{i'+1,i,j,t}\, \overline{x}_{i'+1,i,j,f}\,]_\#\,]_{\langle i'+1,r\rangle}$
for $1 \le i' < i \le n, 1 \le j \le p, r \in \{t,f\}$
$(x_{i',i,j,r}, out) \in \mathcal{R}_\#$
for $1 \le i' < i \le n, 1 \le j \le p, r \in \{t,f\}$
$(\overline{x}_{i',i,j,r}, out) \in \mathcal{R}_\#$
for $1 \le i' < i \le n, 1 \le j \le p, r \in \{t,f\}$
$(x_{i'+1,i,j,r}, in) \in \mathcal{R}_{\langle i'+1,r\rangle}$
for $1 \le i < i' \le n, 1 \le j \le p, r \in \{t,f\}$
$(\overline{x}_{i'+1,i,j,r}, in) \in \mathcal{R}_{\langle i'+1,r\rangle}$
for $1 \le i < i' \le n, 1 \le j \le p, r \in \{t,f\}$
$[\,x_{i,i,j,t} \to [\,c_{i,j,t}\, c_{i,j,f}\,]_\#\,]_{\langle i,t\rangle}$ for $1 \le i \langle n, 1 \le j \le p$
$[\,\overline{x}_{i,i,j,f} \to [\,c_{i,j,t}\, c_{i,j,f}\,]_\#\,]_{\langle i,f\rangle}$ for $1 \le i \langle n, 1 \le j \le p$
$[\,x_{n,n,j,t} \to [\,c_{n,j,t}\,]_\#\,]_{\langle n,t\rangle}$ for $1 \le j \le p$
$[\,\overline{x}_{n,n,j,f} \to [\,c_{n,j,f}\,]_\#\,]_{\langle n,f\rangle}$ for $1 \le j \le p$

$(c_{i,j,r}, out) \in \mathcal{R}_\#$ for $1 \le i \le n, 1 \le j \le p, r \in \{t,f\}$
$(c_{i,j,r}, in) \in \mathcal{R}_{\langle i+1,r\rangle}$ for $1 \le i \langle n, 1 \le j \le p, r \in \{t,f\}$
$[\,c_{i,j,r} \to [\,c_{i+1,j,t}\, c_{i+1,j,f}\,]_\#]_{\langle i',r\rangle}$
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

$[\,z_n \to [\,d_0]_\#\,]_{\langle n,r\rangle}$ for $r \in \{t,f\}$
$[\,c_{n,j,r} \to [\ \ ]_j\,]_{\langle n,r'\rangle}$ for $1 \le j \le p, r, r' \in \{t,f\}$
$(d_j, in) \in \mathcal{R}_{j+1}$ for $0 \le j < p, r \in \{t,f\}$
$[\,c_{n,j,r} \to [\ \ ]_j\,]_{\langle n,r\rangle}$ for $1 \le j \le p, r \in \{t,f\}$
$[\,d_j \to [\,d_{j+1}\,]_\#\,]_{j+1}$ for $0 \le j < p$
$(d_j, out) \in \mathcal{R}_\#$ for $0 \le j \le p, r \in \{t,f\}$
$(d_j, out) \in \mathcal{R}_j$ for $0 \le j \le p, r \in \{t,f\}$
$[\,d_p \to [\,d_{n,r}\,]_\#\,]_{\langle n,r\rangle}$ for $r \in \{t,f\}$
$(d_{n,r}, out) \in \mathcal{R}_\#$ for $r \in \{t,f\}$

6.7 Rules to check if quantifiers are satisfied

$(d_{i,r}, out) \in \mathcal{R}_{\langle i,r\rangle}$ for $1 \le i \le n, r, r' \in \{t,f\}$
$[\, d_{i+1,r} \to [\quad]_{\langle i,\exists,r,r'\rangle}\,]_{\langle i,r'\rangle}$ for $1 \le i < n, i$ odd $, r, r' \in \{t,f\}$
$[\, d_{i+1,r} \to [\quad]_{\langle i,\forall,r,r'\rangle}\,]_{\langle i,r'\rangle}$ for $2 \le i < n, i$ even $, r, r' \in \{t,f\}$

$(d', in) \in \mathcal{R}_{\langle i,\exists,r,r'\rangle}$ for $1 \le i \le n, i$ odd $, r, r' \in \{t,f\}$
$[\, d' \to [\, d_{i,r'}\,]_{\#}]_{\langle i,\exists,r,r'\rangle}$ for $1 \le i \le n, i$ odd $, r, r' \in \{t,f\}$
$(d_{\langle i,r'\rangle}, out) \in \mathcal{R}_{\#}$ for $1 \le i \le n, i$ odd $, r, r' \in \{t,f\}$
$(d_{\langle i,r'\rangle}, out) \in \mathcal{R}_{\langle i,\exists,r,r'\rangle}$ for $1 \le i \le n, i$ odd $, r, r' \in \{t,f\}$

$(d'', in) \in \mathcal{R}_{\langle i,\forall,r,r'\rangle}$ for $1 \le i \le n, i$ even $, r, r' \in \{t,f\}$
$[\, d'' \to [\, d'_r\,]_{\#}]_{\langle i,\forall,r,r'\rangle}$ for $1 \le i \le n, i$ even $, r, r' \in \{t,f\}$
$(d'_r, out) \in \mathcal{R}_{\#}$ for $1 \le i \le n, r, r' \in \{t,f\}$
$(d'_r, out) \in \mathcal{R}_{\langle i,\forall,r,r'\rangle}$ for $1 \le i \le n-1, r, r' \in \{t,f\}$
$(d'_{r''}, in) \in \mathcal{R}_{\langle i,\forall,r,r'\rangle}$ for $1 \le i \le n, i$ even $, r, r', r'' \in \{t,f\}, r \ne r''$
$[\, d'_{r''} \to [\, d_{i,r'}\,]_{\#}]_{\langle i,\forall,r,r'\rangle}$ for $1 \le i \le n, i$ even $, r, r', r'' \in \{t,f\}, r \ne r''$
$(d_{i,r'}, out) \in \mathcal{R}_{\#}$ for $1 \le i \le n, r, r' \in \{t,f\}$

$(d_{i,r'}, out) \in \mathcal{R}_{\langle i,Q,r,r'\rangle}$ for $1 \le i \le n-1, r, r' \in \{t,f\}, Q \in \{\exists,\forall\}$
$[\, d_{1,r} \to [\quad]_{yes}\,]_{skin}$ for $r \in \{t,f\}$

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

4.1 An overview of the computation

The proposed solution follows a brute force scheme of recognizer P systems with symport/antiport 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 "garbage" membranes, labeled by #, used to generate the objects $z_{i+1,t}$ and $z_{i+1,f}$. 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 labeled by $\langle n, r\rangle$. 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 labeled by $\langle n, r\rangle$, it means that the corresponding truth assignment makes true the clause j. Therefore, a membrane labeled by j is created within such a membrane $\langle n, r\rangle$. Object d_0 will go through all membranes, creating a "garbage" membrane within them and passing to the next one, possibly arriving to membrane p. In that case, object d_p creates a new garbage 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 should be 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 $\langle i, \exists, r, r'\rangle$. 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 labeled by $\langle i, \forall, r, r'\rangle$. 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 to know if an object $d_{1,r}$ has reached the skin membrane. In such a case, a membrane labeled by *yes* will be created, and when object $\alpha_{n^2+5n+2p+3}$ reaches the skin membrane, it will go into membrane *yes* and will change into an object `yes` that will be sent to the environment. In the case that object $d_{1,r}$ does not appear in the skin membrane, object $\alpha'_{n^2+5n+2p+4}$ will generate a membrane labeled by *no*, that will make the counter $\alpha_{n^2+5n+2p+3}$ change into an object `no`, and will be sent to the environment. Rules from 6.2 and 6.3 are the responsible in this stage. It takes $p + 8$ steps if n is even and $p + 9$ steps if n is odd.

4.2 Results

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

Theorem 1 `QSAT` $\in \mathbf{PMC}_{\widehat{\mathcal{CCC}}(1)}$.

The family of P systems $\mathbf{\Pi}$ is polynomially uniform by Turing machines, polynomially bounded, sound and complete with regard to (`QSAT`, cod, s) and both cod and s are polynomial-time computable functions.

Corollary 2 $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\widehat{\mathcal{CCC}}(1)}$.

Proof It suffices to know that QSAT is a **PSPACE**-complete problem, QSAT $\in \mathbf{PMC}_{\widehat{\mathcal{CCC}}(1)}$ and the class $\mathbf{PMC}_{\widehat{\mathcal{CCC}}(1)}$ is closed under polynomial-time reduction and under complementary. □

In [20], a Python simulator developed for this framework can be found. This framework can simulate any P system with symport/antiport rules and creation rules. However, some specific functions have been implemented to make it easier to obtain solution to QSAT formulas. In the `main.py` file, you can find the line

`formula = examples.generate_random_formula(n, p).`

This line indicates which formula is going to be solved, in this case, a randomly generated formula with at most n variables and p clauses. In the case that an specific formula is to be solved, this line can be changed by, for instance,

`formula = examples.example1()).`

In the file `examples.py`, it can be found how these formulas are generated, and they can be taken as a reference for creating new formulas. To create a new formula, the same protocol can be followed: we can create a tuple of tuples, where each of these tuples symbolize a clause, and each element of these tuples represent a literal, either in positive form with `x[i]` or in negative form with `xb[i]`, where `i` is the variable being used. Therefore, following the format of the examples, the formula

$$\varphi \equiv \exists x_1 \forall x_2 \exists x_3 (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

could be represented by

$$\begin{aligned}\mathtt{f}ormula = &(('x[1]','xb[2]'),('xb[1]','xb[2]',\\ &'x[3]'),('x[2]','xb[3]')).\end{aligned}$$

It can be directly simulated by `python main.py`, and the result will be directly shown in the screen since the `compute` function will show the contents of the environment.

5 Conclusions and future work

In this work, a result concerning evolutional symport/antiport rules has been improved, in the sense that no evolution is needed in these kinds of rules while using creation rules. While in the previous work, a solution based on a family of P systems from $\widehat{\mathcal{CCEC}}(1,1)$ was detailed, in this work, we restrict the number of objects used in a symport/antiport rule to one; that is, to P systems from $\widehat{\mathcal{CCC}}(1)$. This is a demonstration of the power of creation rules, showing that rules with a minimal number of objects in symport/antiport rules is enough to reach presumed efficiency. The use of division rules and separation rules with this length of symport/antiport rules gives P systems the power to efficiently solve only problems from **P**.

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. In this case, the implementation and use of a simulator has helped to correct some errors in the design, showing the advantages of using them as research assistants. The non-evolutive nature of symport/antiport rules are an improvement with respect to their evolutional counterparts, since in the simulator, no changes of these objects must be taken into account, then the same objects will be moved throughout the whole system, except for the ones created when using creation rules.

Acknowledgements This work was supported by the following research project: FEDER/Junta de Andalucía - Paidi 2020/ _Proyecto (P20_00486). D. Orellana-Martín 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.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Păun, Gh. (1998). *Computing with Membranes*. Turku Centre for Computer Science: Technical Report.
2. Alhazov, A., Freund, R., Ivanov, S., & Oswald, M. (2022). Variants of derivation modes for which purely catalytic P systems are computationally complete. *Theoretical Computer Science, 920*, 95–112. https://doi.org/10.1016/j.tcs.2022.03.007

3. Leporati, A., Manzoni, L., Mauri, G., & Zandron, C. (2022). Depth-two P systems can simulate Turing machines with NP oracles. *Theoretical Computer Science, 908*, 43–55. https://doi.org/10.1016/j.tcs.2021.11.010
4. Baquero, F., Campos, M., Llorens, C., & Sempere, J. M. (2021). P systems in the time of COVID-19. *Journal of Membrane Computing, 3*(4), 246–257. https://doi.org/10.1007/s41965-021-00083-1
5. Liu, Y., Chen, Y., Paul, P., Fan, S., Ma, X., & Zhang, G. (2021). A review of power system fault diagnosis with spiking neural P systems. *Applied Sciences*. https://doi.org/10.3390/app11104376.
6. Păun, Gh. (2001). P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages, and Combinatorics, 6*(1), 75–90.
7. Păun, A., & Păun, Gh. (2002). The power of communication: P systems with symport/antiport. *New Generation Computing, 20*(3), 295–305.
8. Alhazov, A., Freund, R., & Ivanov, S. (2021). When catalytic P systems with one catalyst can be computationally complete. *Journal of Membrane Computing, 3*(3), 170–181. https://doi.org/10.1007/s41965-021-00079-x
9. Macías-Ramos, L. F., Song, B., Song, T., Pan, L., & Pérez-Jiménez, M. J. (2017). Limits on efficient computation in p systems with symport/antiport. Fifteenth Brainstorming Week on Membrane Computing (BWMC2017), 147–160.
10. Pan, L., & Ishdorj, T.-O. (2004). P systems with active membranes and separation rules. *Journal of Universal Computer Science, 10*(5), 630–649.
11. Mutyam, M., & Krithivasan, K. (2001). P systems with membrane creation: Universality and efficiency. In: Proceedings of the Third International Conference on Machines, Computations, and Universality. MCU '01, (pp. 276–287). Springer, Berlin, Heidelberg.
12. Song, B., Li, K., Orellana-Martín, D., Valencia-Cabrera, L., & Pérez-Jiménez, M. J. (2020). Cell-like P systems with evolutional symport/antiport rules and membrane creation. *Information and Computation, 275*, 104542.
13. Orellana-Martín, D., Valencia-Cabrera, L., Riscos-Núñez, A., & Pérez-Jiménez, M. J. (2020). Membrane creation in polarizationless P systems with active membranes. *Fundamenta Informaticae, 171*, 297–311.
14. Orellana-Martín, D., Valencia-Cabrera, L., & Pérez-Jiménez, M. J. (2022). P systems with evolutional symport and membrane creation rules solving QSAT. *Theoretical Computer Science, 908*, 56–63. https://doi.org/10.1016/j.tcs.2021.11.012
15. Paun, G., Rozenberg, G., & Salomaa, A. (2010). *The Oxford Handbook of Membrane Computing*. Oxford University Press Inc.
16. Rozenberg, G., & Salomaa, A. (eds.). (1997). *Handbook of Formal Languages*. 3 Vols. Springer.
17. Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J., & Romero-Campero, F. J. (2005). A linear solution of subset sum problem by using membrane creation. In J. Mira & J. R. Álvarez (Eds.), *Mechanisms, Symbols, and Models Underlying Cognition* (pp. 258–267). Berlin, Heidelberg: Springer.
18. Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J., & Romero-Campero, F. J. (2006). A linear solution for qsat with membrane creation. In R. Freund, G. Păun, G. Rozenberg, & A. Salomaa (Eds.), *Membrane Computing* (pp. 241–252). Berlin, Heidelberg.
19. Pérez-Jiménez, M.J., Álvaro Romero-Jiménez, & Sancho-Caparrini, F. (2003). Complexity classes in models of cellular computing with membranes. *Natural Computing: An International Journal*, *2*(3), 265–285.
20. Orellana-Martín, D. (2022) creation-ccc. GitHub.