Computational efficiency and universality of timed P systems with membrane creation

Bosheng Song[1] · **Mario J. Pérez-Jiménez**[2] · **Linqiang Pan**[1]

Abstract P systems are a class of distributed parallel computing models inspired by the structure and the functioning of a living cell, where the execution of each rule is completed in exactly one time unit (a global clock is assumed). How-ever, in living cells, the execution time of different biological processes is difficult to know precisely and very sensi-tive to environmental factors that might be hard to control. Inspired from this biological motivation, in this work, timed polarization P systems with membrane creation are intro-duced and their computational efficiency and universality are investigated. Specifically, we give a time-free semi-uniform solution to the SAT problem by a family of P systems with membrane creation in the sense that the correctness of the solution is irrelevant to the times associated with the involved rules. We also prove that time-free P systems with membrane creation are computationally universal.

Keywords P system · Membrane creation · Time-free solution · SAT problem · Universality

✉ Linqiang Pan
lqpan@mail.hust.edu.cn

[1] Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China

[2] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

1 Introduction

Membrane computing is a vivid branch of natural computing, which was initiated by Păun (2000). The aim of membrane computing is focused to abstract computing concepts from the structure and the functioning of living cells and the ways in which cells are organized in tissues and higher order biological structures. All classes of computing devices considered in the framework of membrane computing are usually called *P systems*, which are distributed and parallel models. In recent years, many variants of P systems have been proposed (Pan and Păun 2010; Pan et al. 2011; Zhang et al. 2012). For general information in membrane computing, one may consult Păun (2000) and Păun et al. (2010), and for the current developments in the area of membrane computing, please refer to the P systems website http://ppage.psystems.eu.

The present work deals with a class of cell-like P systems, called *P systems with membrane creation*, introduced in Mutyam and Krithivasan (2001), inspired by the fact that when a compartment becomes too large, it often happens that new membranes appear inside it, more or less spontaneously or during biological evolution (Păun 2000). P systems with membrane creation have similar types of rules as P systems with active membranes (Păun 2001) except that membrane division rules are substituted with membrane creation rules. A membrane creation rule has no replication of objects into the new membrane, which becomes a daughter of the original membrane (the depth of the membrane structure increases). Usually, the rules of P systems with membrane creation are applied in a non-deterministic maximally parallel way, that is, in one step, one object of a membrane is used by at most one rule (chosen in a non-deterministic way), but any object which can evolve by one rule of any form must evolve.

P systems with membrane creation are proved to be universal (Mutyam and Krithivasan 2001) and have been used to solve computationally hard problems (Gut-iérrez-Naranjo et al. 2005, 2006, 2007). All these above mentioned P systems with membrane creation work in a parallel and synchronized way (a global clock is assumed to mark the time for the system); in each tick of the global clock, all the applicable rules are applied simultaneously, and the execution of rules takes exactly one time unit. However, in biological reality, the execution time of a chemical reaction might be difficult to know precisely and usually such a parameter is very sensitive to environmental factors that might be hard to control. With this biological motivation, *timed P systems* were introduced in Cavaliere and Sburlan (2005), where a natural number representing the execution time is associated with each rule. In Cavaliere and Sburlan (2005), a class of systems, called *time-free P systems*, was also introduced and proved computationally universal, where such P systems always produce the same result, independent from the execution times of the rules.

Time-free solutions to computationally hard problems were proposed as open problems in Gheorghe et al. (2013). In Song et al. (2014), a family of P systems with active membranes was designed for a time-free solution to the SAT problem in the sense that the correctness of the solution does not depend on the precise timing of the involved rules, but the newly generated membranes can have the different labels with their parent membrane. An improved result was presented in Song et al. (2015b), where the newly generated membranes have the same label with their parent membrane. A time-free uniform solution to the SAT problem using P systems with active membranes was given in Song et al. (2015a). Several variants of P systems were also considered to time-freely solve *NP*-complete problems: tissue P systems were used to solve the subset sum problem (Song et al. 2015c) and in Song et al. (2013), a family of P systems with d-division was designed for solving the Hamiltonian path problem.

In this work, we focus on timed P systems with membrane creation, where the computational efficiency and universality of such P systems are investigated. Specifically, we present a time-free solution to the SAT problem using P systems with membrane creation in the sense that the correctness of the solution is irrelevant to the times associated with the involved rules, but this result is obtained on the condition that the membranes are polarized. We also prove that time-free P systems with membrane creation with electrical charges on membranes are computationally universal.

2 Formal language theory preliminaries

The reader is assumed to have some familiarity with (basic elements of) language theory, e.g., from Rozenberg and Salomaa (1997). Here, we directly introduce some notations and basic definitions used in this work.

For an alphabet V, V^* denotes the set of all strings over V (ordered finite sequences of elements in V), with the empty string denoted by λ, and the set of all nonempty strings over V is denoted by V^+.

By $\mathbb{N}$ we denote the set of non-negative integers. A multiset over a finite alphabet $V = \{a_1, a_2, \ldots, a_n\}$ is a mapping $m : V \to \mathbb{N}$. The support of a multiset m over V is the set of elements $x \in V$ such that $m(x) > 0$. A multiset is finite if its support is a finite set. We denote by $M_f(V)$ the set of all finite multisets over V.

For a family FL of languages (over a given alphabet V), we denote by PsFL the family of Parikh images of languages in FL. By PsRE we denote the family of recursively enumerable sets of vectors of natural numbers; this is equal to the family of Parikh sets of recursively enumerable languages.

A *matrix grammar with appearance checking* is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N$, $x_i \in (N \cup T)^*$, in all cases), and F is a set of occurrences of rules in M (we say that N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w, z \in (N \cup T)^*$, we write $w \Rightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*$, $1 \leq i \leq n+1$, such that $w = w_1$, $z = w_{n+1}$, and for all $1 \leq i \leq n$, either $w_i = w'_i A_i w''_i$, $w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied; we say that these rules are applied in the appearance checking mode.)

We denote by $\Rightarrow^*$ the reflexive and transitive closure of the relation $\Rightarrow$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. It is known that $\text{MAT}_{ac} = \text{RE}$.

A matrix grammar $G = (N, T, S, M, F)$ is said to be in the *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, and the matrices in M are of one of the following forms:

1. $(S \to XA)$, with $X \in N_1$, $A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$,
3. $(X \to Y, A \to \#)$, with $X, Y \in N_1$, $A \in N_2$,
4. $(X \to \lambda, A \to x)$, with $X \in N_1$, $A \in N_2$, $x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; # is a trap symbol, once introduced, it is never removed.

A matrix of type 4 is used only once, at the last step of a derivation.

It is known that for each matrix grammar there is an equivalent matrix grammar in the binary normal form; hence each language in RE can be generated by a grammar in the binary normal form.

3 Timed P systems with membrane creation

3.1 Timed and time-free polarization P systems with membrane creation

In this subsection, we directly give the definitions of timed and time-free P systems with membrane creation. For the notion of P systems with membrane creation, please refer to Păun (2000) and Păun et al. (2010).

Definition 1 A timed polarization P system with membrane creation of degree $m \geq 1$ is a construct

$$\Pi(e) = (O, H, \mu, w_1, \ldots, w_m, R, i_{\text{out}}, e),$$

where O is a finite alphabet of *objects*; H is a finite set of *labels* for membranes; μ is an initial *membrane structure*, consisting of m membranes; membranes are labelled (not necessarily in an injective way) with elements of H; $w_1, \ldots, w_m$ are finite multisets over O, describing the *initial multisets of objects* placed in the m regions of μ; $i_{\text{out}} \in \{0, 1, \ldots, m\}$ is the output region (0 is the label representing the environment); e is a mapping from the finite set of rules R into the set of natural numbers $\mathbb{N}$, which represent the execution time of the rules; R is a finite set of rules of the forms:

(a) $[a \to v]_h^{\alpha}$, $h \in H, \alpha \in \{+, -, 0\}, a \in O, v \in M_f(O)$ (object evolution rules), associated with membranes and depending on the label and the charge of the membranes.

(b) $a[\,]_h^{\alpha_1} \to [b]_h^{\alpha_2}$, $h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a \in O, b \in O \cup \{\lambda\}$ (send-in communication rules). An object is sent into the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label.

(c) $[a]_h^{\alpha_1} \to [\,]_h^{\alpha_2} b$, $h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a \in O, b \in O \cup \{\lambda\}$ (send-out communication rules). An object is sent out of the membrane, possibly modified during this process; also the polarization of the membrane can be modified, but not its label.

(d) $[a]_h^{\alpha} \to b$, $h \in H, \alpha \in \{+, -, 0\}, a \in O, b \in O \cup \{\lambda\}$ (dissolving rules). In reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified.

(e) $[a \to [\ v\ ]_{h_2}^{\alpha_2}]_{h_1}^{\alpha_1}$, $h_1, h_2 \in H, \alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $a \in O$, $v \in M_f(O)$ (creation rules). In reaction with an object, a new membrane is created, maybe of different polarization. This new membrane is placed inside of the membrane of the object which triggers the rule and has associated an initial multiset, a label and a polarization.

A timed P system with membrane creation $\Pi(e)$ works in the following way: An external clock is assumed, which marks time-units of equal length, starting from instant 0. According to this clock, the step t of computation is defined by the period of time that goes from instant $t - 1$ to instant t. If a membrane i contains a rule r from types (a)–(e) selected to be executed, then the execution of such rule takes $e(r)$ time units to complete. Therefore, if the execution is started at instant j, the rule is completed at instant $j + e(r)$ and the resulting objects become available only at the beginning of step $j + e(r) + 1$. When a rule r from types (a)–(e) is started, then the occurrences of symbol-objects subject to this rule cannot be subject to other rules until the implementation of the rule completes. In timed P systems with membrane creation, only halting computations give a result, which is encoded by the objects present in the output region.

Note that the rules of a timed polarization P system with membrane creation are applied according to usual principles of timed P system with membrane creation as mentioned above. We only emphasize that in timed polarization P system with membrane creation, several rules can be applied to different objects in the same membrane simultaneously if the polarization of the membrane is not modified during the applications of these rules.

Definition 2 A P system with membrane creation Π is time-free if and only if every timed P system with membrane creation in the set $\{\Pi(e) \mid e\colon R \to \mathbb{N} - \{0\}\}$ produces the same set of vectors of natural numbers.

The set of vectors generated by a time-free P system with membrane creation Π is denoted by $\text{Ps}(\Pi)$. By $\text{PsOP}^{\text{free}}$(list-of-rules) we denote the families of the sets of vectors generated by time-free P systems with membrane creation, using rules as specified in the *list-of-rules*; *free* stands for *time-free*.

Definition 3 A recognizer timed (polarization) P system with membrane creation of degree $m \geq 1$ without input is a tuple $\Pi = (O, H, \mu, w_1, \ldots, w_m, R, i_{\text{out}}, e)$, such that

- the tuple $(O, H, \mu, w_1, \ldots, w_m, R, i_{\text{out}}, e)$ is a timed (polarization) P system with membrane creation;
- the working alphabet contains two distinguished elements `yes` and `no`;

- all the computations halt;
- if $\mathcal{C}$ is a computation of the system, then either object yes or object no (but not both) must appear in the environment when the system halts.

3.2 Time-free solutions to decision problems by P systems with membrane creation

In timed (polarization) P systems with membrane creation, we use rule starting step (RS-step, for short) to define the computation step (Song et al. 2014).

Definition 4 In timed (polarization) P systems with membrane creation, a computation step is called an RS-step if at this step at least one rule starts its execution, that is, steps in which some objects "start" to evolve or some membranes "start" to change.

A *decision problem*, X, is a pair (I_X, Θ_X) such that I_X is a language over a finite alphabet (whose elements are called *instances*) and Θ_X is a total Boolean function (that is, predicate) over I_X.

Definition 5 Let $X = (I_X, \Theta_X)$ be a decision problem. We say that X is solvable in polynomial RS-steps by a family of recognizer (polarization) P systems with membrane creation $\mathbf{\Pi} = \{\Pi_u, u \in I_X\}$, in a time-free manner, if the following items are true:

- the family $\mathbf{\Pi}$ is polynomially uniform by a Turing machine; that is, there exists a deterministic Turing machine working in polynomial time which constructs the system Π_u from the instance $u \in I_X$;
- the family $\mathbf{\Pi}$ is time-free polynomially bounded; that is, there exists a polynomial function $p(n)$ such that for any time-mapping e and for each $u \in I_X$, all computations in $\Pi_u(e)$ halt in, at most, $p(|u|)$ RS-steps;
- the family $\mathbf{\Pi}$ is time-free sound (with respect to X); that is, for any time-mapping e, the following property holds: for each instance of the problem $u \in I_X$ such that there exists an accepting computation of $\Pi_u(e)$, we have $\Theta_X(u) = 1$;
- the family $\mathbf{\Pi}$ is time-free complete (with respect to X); that is, for any time-mapping e, the following property holds: for each instance of the problem $u \in I_X$ such that $\Theta_X(u) = 1$, every computation of $\Pi_u(e)$ is an accepting computation.

We also say that the family $\mathbf{\Pi}$ provides a time-free semi-uniform efficient solution to the decision problem X.

4 A time-free solution to the SAT problem by P systems with membrane creation

The SAT problem is the well-known *NP*-complete problem (Garey and Johnson 1979). It asks whether for a given formula in the conjunctive normal form there is a truth-assignment of variables or not such that makes the formula evaluate to be *true*.

The following theorem shows that the SAT problem can be solved in polynomial RS-steps by a family of polarization P systems with membrane creation in a time-free manner.

Theorem 1 *The* SAT *problem can be solved in polynomial RS-steps by a family of polarization P systems with membrane creation with rules of types* (*a*)–(*e*) *in a time-free manner.*

Proof Let us consider a propositional formula $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, with $C_i = y_{i,1} \vee \cdots \vee y_{i,p_i}$, for some $m \geq 1$, $p_i \geq 1$, and $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\}$, for each $1 \leq i \leq m, 1 \leq j \leq p_i$, where $\neg x_k$ is the negation of a propositional variable x_k, and the two connections $\vee$, $\wedge$ are *or*, *and*, respectively.

For the given propositional formula C, we construct the polarization P systems with membrane creation

$$\Pi_C = (O, H, \mu, w_1, w_2, R, i_{\text{out}}),$$

where $O = \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{r_i, r'_i, r_{i,t}, r_{i,f}, b_i^{(1)}, b_i^{(2)}, b_i^{(3)}, b_i^{(4)}, c'_i \mid 1 \leq i \leq m\} \cup \{a_i^{(1)}, a_i^{(2)}, a_i^{(3)} \mid 2 \leq i \leq n+1\} \cup \{c_i \mid 2 \leq i \leq m+1\} \cup \{a_i^{(4)} \mid 3 \leq i \leq n+1\} \cup \{\text{yes}, \text{no}, a_{n+1}, r_{m+1,t}, d, e_1, e_2, e_3\}$ is the alphabet, $H = \{-1, 0, 1, 2, \ldots, n+m+1\}$ is the set of labels of the membranes, $\mu = [\,[\,]_2^0\,]_1^0$ is initial membrane structure, $w_1 = \text{no}$ is the initial multiset contained in membrane 1, $w_2 = a_1$ is the initial multiset contained in membrane 2, $i_{\text{out}} = 0$, and the set R contains the following rules:

G_1: $[a_1 \to t_1 f_1 d]_2^0$.
G_2: $[d]_2^0 \to \lambda$.
G_3: $[t_1 \to [r_{h_{1,1,t}} r_{h_{1,1,f}} \ldots r_{h_{1,j_1,t}} r_{h_{1,j_1,f}} a_2^{(1)}]_2^+]_1^+$, and the clauses $C_{h_{1,1}}, \ldots, C_{h_{1,j_1}}$ contain the literal x_1.
G_4: $[f_1 \to [r_{h_{1,1,t}} r_{h_{1,1,f}} \ldots r_{h_{1,j_1,t}} r_{h_{1,j_1,f}} a_2^{(2)}]_2^+]_1^+$, and the clauses $C_{h_{1,1}}, \ldots, C_{h_{1,j_1}}$ contain the literal $\neg x_1$.
G_5: $[a_2^{(1)}]_2^+ \to [\,]_2^+ a_2^{(1)}$.
G_6: $[a_2^{(2)}]_2^+ \to [\,]_2^+ a_2^{(2)}$.
G_7: $[a_2^{(1)} \to [\,]_0^0]_1^+$.
G_8: $a_2^{(2)}[\,]_0^0 \to [a_2^{(2)}]_0^0$.
G_9: $[a_2^{(2)}]_0^0 \to a_2^{(3)}$.
G_{10}: $[a_2^{(3)} \to a_2^2]_1^+$.
G_{11}: $a_2[\,]_2^+ \to [a_2]_2^0$.
$G_{12,i}$: $[a_i \to t_i f_i]_i^0$, $2 \leq i \leq n$.

$G_{13,i}$: $[t_i \to [r_{h_{i,1,t}} r_{h_{i,1,f}} \dots r_{h_{i,j_i,t}} r_{h_{i,j_i,f}} a_{i+1}^{(1)}]_{i+1}^{+}]_i^0$, $2 \le i \le n-1$, and the clauses $C_{h_{i,1}}, \dots, C_{h_{i,j_i}}$ contain the literal x_i.

$G_{14,i}$: $[f_i \to [r_{h_{i,1,t}} r_{h_{i,1,f}} \dots r_{h_{i,j_i,t}} r_{h_{i,j_i,f}} a_{i+1}^{(2)}]_{i+1}^{-}]_i^0$, $2 \le i \le n-1$, and the clauses $C_{h_{i,1}}, \dots, C_{h_{i,j_i}}$ contain the literal $\neg x_i$.

G_{15}: $[t_n \to [r_{h_{n,1}} \dots r_{h_{n,j_n}} a_{n+1}^{(1)}]_{n+1}^{+}]_n^0$, and the clauses $C_{h_{n,1}}, \dots, C_{h_{n,j_n}}$ contain the literal x_n.

G_{16}: $[f_n \to [r_{h_{n,1}} \dots r_{h_{n,j_n}} a_{n+1}^{(2)}]_{n+1}^{-}]_n^0$, and the clauses $C_{h_{n,1}}, \dots, C_{h_{n,j_n}}$ contain the literal $\neg x_n$.

$G_{17,i}$: $[a_i^{(1)}]_i^+ \to [\ ]_i^+ a_i^{(1)}$, $3 \le i \le n+1$.

$G_{18,i}$: $[a_i^{(2)}]_i^- \to [\ ]_i^- a_i^{(2)}$, $3 \le i \le n+1$.

$G_{19,i}$: $[a_{i+1}^{(1)} \to [\ ]_0^0]_i^0$, $2 \le i \le n$.

$G_{20,i}$: $a_i^{(2)}[\ ]_0^0 \to [a_i^{(2)}]_0^0$, $3 \le i \le n+1$.

$G_{21,i}$: $[a_i^{(2)}]_0^0 \to a_i^{(3)}$, $3 \le i \le n+1$.

$G_{22,i}$: $[a_{i+1}^{(3)} \to [r_{1,t} a_{i+1}^{(4)}]_{-1}^0]_i^0$, $2 \le i \le n$.

$G_{23,i}$: $[a_i^{(4)} \to []_i^+]_{-1}^0$, $3 \le i \le n+1$.

$G_{24,i,j}$: $r_{j,t}[\ ]_i^+ \to [r_j]_i^+$, $3 \le i \le n$, $1 \le j \le m$.

$G_{25,i,j}$: $[r_j \to r_{j,t} r_{j,f} b_j^{(1)}]_i^+$, $3 \le i \le n$, $1 \le j \le m$.

$G_{26,i,j}$: $[b_j^{(1)}]_i^+ \to [\ ]_i^+ b_j^{(1)}$, $3 \le i \le n+1$, $1 \le j \le m$.

$G_{27,j}$: $[b_j^{(1)} \to b_j^{(2)}]_{-1}^0$, $1 \le j \le m$.

$G_{28,i,j}$: $b_j^{(2)}[\ ]_i^+ \to [b_j^{(2)}]_i^-$, $3 \le i \le n+1$, $1 \le j \le m$.

$G_{29,i,j}$: $[b_j^{(2)}]_i^- \to [\ ]_i^- r_{j,f}$, $3 \le i \le n+1$, $1 \le j \le m$.

$G_{30,i,j}$: $r_{j,f}[\ ]_i^- \to [r_j]_i^-$, $3 \le i \le n$, $1 \le j \le m$.

$G_{31,i,j}$: $[r_j \to r_{j,t} r_{j,f} b_j^{(3)}]_i^-$, $3 \le i \le n$, $1 \le j \le m$.

$G_{32,i,j}$: $[b_j^{(3)}]_i^- \to [\ ]_i^- b_j^{(3)}$, $3 \le i \le n+1$, $1 \le j \le m$.

$G_{33,j}$: $[b_j^{(3)} \to b_j^{(4)}]_{-1}^0$, $1 \le j \le m$.

$G_{34,i,j}$: $b_j^{(4)}[\ ]_i^- \to [b_j^{(4)}]_i^+$, $3 \le i \le n+1$, $1 \le j \le m$.

$G_{35,i,j}$: $[b_j^{(4)}]_i^+ \to [\ ]_i^+ r_{j+1,t}$, $3 \le i \le n+1$, $1 \le j \le m$.

$G_{36,j}$: $r_{j,t}[\ ]_{n+1}^+ \to [r'_j]_{n+1}^+$, $1 \le j \le m$.

$G_{37,j}$: $[r'_j \to r_j b_j^{(1)}]_{n+1}^+$, $1 \le j \le m$.

$G_{38,j}$: $r_{j,f}[\ ]_{n+1}^- \to [r'_j]_{n+1}^-$, $1 \le j \le m$.

$G_{39,j}$: $[r'_j \to r_j b_j^{(1)}]_{n+1}^-$, $1 \le j \le m$.

$G_{40,i}$: $r_{m+1,t}[\ ]_i^+ \to [e_1]_i^+$, $3 \le i \le n+1$.

$G_{41,i}$: $[e_1]_i^+ \to e_1$, $3 \le i \le n+1$.

G_{42}: $[e_1]_{-1}^0 \to e_1$.

$G_{43,i}$: $e_1[\ ]_i^- \to [e_1]_i^-$, $3 \le i \le n+1$.

$G_{44,i}$: $[e_1]_i^- \to [\ ]_i^+ e_2$, $3 \le i \le n+1$.

$G_{45,i}$: $[e_2 \to a_{i+1}^2 e_3]_i^0$, $2 \le i \le n$.

$G_{46,i}$: $a_i[\ ]_i^+ \to [a_i]_i^0$, $3 \le i \le n+1$.

$G_{47,i}$: $[e_3]_i^0 \to \lambda$, $2 \le i \le n$.

C_1: $[a_{n+1} \to [c'_1]_{n+2}^0]_{n+1}^0$.

$C_{2,j}$: $[c_j \to [c'_j]_{n+1+j}^0]_{n+1}^0$, $2 \le j \le m$.

$C_{3,j}$: $r_j[\ ]_{n+1+j}^0 \to [r'_j]_{n+1+j}^+$, $1 \le j \le m$.

$C_{4,j}$: $[c'_j]_{n+1+j}^+ \to c_{j+1}$, $1 \le j \le m$.

O_1: $[\mathtt{no}]_1^0 \to [\ ]_1^+ \mathtt{no}$.

O_2: $\mathtt{no}[\ ]_1^- \to [\mathtt{no}]_1^-$.

O_3: $[c_{m+1}]_{n+1}^0 \to [\ ]_{n+1}^0 \mathtt{yes}$.

O_4: $[\mathtt{yes}]_1^+ \to [\ ]_1^- \mathtt{yes}$.

In what follows, we show how the above constructed system Π_C gives a solution to the propositional formula C. Generally, the computation process can be divided into three phases: generation phase, checking phase, and output phase.

Generation phase At the beginning of the computation, we have object no in membrane 1, object a_1 in membrane 2, and the object a_1 corresponds to variable x_1. At step 1, rule G_1 is applied, the object a_1 evolves to the objects t_1, f_1, d in membrane 2. For any given time-mapping e, the execution of rule G_1 completes in $e(G_1)$ steps. As we will see below, at step 1, exception for the application of rule G_1, the application of rule O_1 also starts, object no exits the membrane 1, changing the polarization from neutral to positive, and from step 2 to step $e(G_1)$, there is no rule starting. Thus, during the execution of rule G_1 (i.e., from step 1 to step $e(G_1)$), there is one RS-step. Note that the number of RS-steps during the execution of rule G_1 is independent of the time-mapping e.

After the execution of rule G_1 completes, rule G_2 is enabled and applied, membrane with label 2 is dissolved, and the objects t_1, f_1 will present in membrane 1 at the same step. At this moment, if the execution of rule O_1 is not yet completed, then no rule can be started in the system. When the polarization of membrane 1 is changed to positive, the applications of rules G_3 and G_4 start at the same step, of course, the executions of rules G_3 and G_4 may complete at different steps due to the fact that the execution time associated with rules G_3 and G_4 can be different. The execution of rule G_3 (resp. G_4) corresponds to the process of looking for the clauses satisfied by the truth-assignment *true* (resp. *false*) of variable x_1. Note that in each of the new generated membrane 2, if the clause $C_{h_{1,j_1}}$ is satisfied, the objects $r_{h_{1,j_1,t}}$, $r_{h_{1,j_1,f}}$ are produced, which will evolve to $r_{h_{1,j_1}}$, and enter the membrane 3 having polarization positive and negative, respectively.

When the execution of rule G_3 (resp. G_4) completes, the application of rule G_5 (resp. G_6) starts, object $a_2^{(1)}$ (resp. $a_2^{(2)}$) exits the membrane with label 2. Rules G_5 and G_6 may start at different steps, since the executions of rules G_3 and G_4 may finish at different steps.

With the appearance of object $a_2^{(1)}$ in membrane 1, rule G_7 is enabled and used, a new membrane with label 0 is generated in membrane 1. Since rule G_8 can be used only when the object $a_2^{(2)}$ presents in membrane 1 and there is a membrane 0 having polarization neutral, that is, rule G_8 can be used only when the applications of both rules G_6 and G_7 have completed. By using rule G_8, object $a_2^{(2)}$ enters the membrane 0. So, the rule G_8 has a synchronization function because $e(G_3)+e(G_5)+e(G_7)$ may not be equal to $e(G_4)+e(G_6)$.

When the execution of rule G_8 finishes, the application of rule G_9 starts, object $a_2^{(2)}$ evolves to $a_2^{(3)}$, and membrane 0 is dissolved. With the appearance of object $a_2^{(3)}$ in membrane 1, rule G_{10} is enabled and used, one copy of object $a_2^{(3)}$ evolves to two copies of object a_2. When the execution of rule G_{10} completes, rule G_{11} is enabled and applied, each object a_2 enters a membrane 2 changing its polarization from positive to neutral (this is due to the fact that there are two copies of object a_2 and two membranes 2 with positive polarization in membrane 1, and the system works in a maximally parallel manner). In summary, when the execution of rule G_{11} completes, the computation takes at most 10 RS-steps, which is independent of any time-mapping e, and the system finishes assigning truth-assignment of variable x_1, and looking for the clauses satisfied by the truth-assignment of variable x_1.

After the execution of rule G_{11} completes, rule $G_{12,2}$ is enabled and used in each membrane 2 at the same step, since the execution of rule G_{11} in all membranes 2 takes the same time $e(G_{11})$ for a given time-mapping e. When the execution of rule $G_{12,2}$ completes, rules $G_{13,2}$ and $G_{14,2}$ are used at the same step, but they may complete at different steps due to the fact that the execution time associated with rules $G_{13,2}$ and $G_{14,2}$ can be different. The execution of rule $G_{13,2}$ (resp. $G_{14,2}$) corresponds to the process of looking for the clauses satisfied by the truth-assignment *true* (resp. *false*) of variable x_2. When the execution of rule $G_{13,2}$ (resp. $G_{14,2}$) completes, the applications of the rule $G_{17,3}$ (resp. $G_{18,3}$) and the rules $G_{24,3,j}$ (resp. $G_{30,3,j}$) [all the object $r_{j,t}$ (resp. $r_{j,f}$) presented in membrane 2 will enter the membrane 3 having polarization positive (resp. negative)] start at the same step, of course, they may complete at different steps because the execution time associated with these rules can be different. As we will see below, we use the rules from $G_{24,3,j}$ to $G_{35,3,j}$ to ensure that all the possible objects $r_{j,t}, r_{j,f}$ presented in membrane 2 have entered the membrane 3. With the appearance of object $a_3^{(1)}$ in membrane 2, the application of rule $G_{19,2}$ starts, a new membrane with label 0 is generated. However, the rule $G_{20,3}$ is enabled only when the rules $G_{18,3}$, $G_{19,2}$ have started and completed their executions. When the object $a_3^{(2)}$ appears in membrane 0, rule $G_{21,3}$ is enabled and used, object $a_3^{(2)}$ evolves to $a_3^{(3)}$, and membrane 0 is dissolved. With the appearance of object $a_3^{(3)}$ in membrane 2, rule $G_{22,2}$ is enabled and applied, a new membrane with label -1 is generated, which contains the objects $r_{1,t}, a_3^{(4)}$. When the execution of rule $G_{22,2}$ completes, rule $G_{23,3}$ is enabled and used, the object $a_3^{(4)}$ creates a new membrane with label 3. Note that the executions of rules $G_{12,2}$, $G_{13,2}$, $G_{14,2}$, $G_{17,3}$, $G_{18,3}$, $G_{19,2}$, $G_{20,3}$, $G_{21,3}$, $G_{22,2}$, $G_{23,3}$ in all membranes 2 start and complete at the same step (there are at most 9 RS-steps), since the executions of the corresponding rules in all membranes 2 take the same time for a given time-mapping e.

With the appearance of membrane 3 in membrane -1, the application of rule $G_{24,3,1}$ starts, object $r_{1,t}$ evolves to r_1, and object r_1 enters the membrane 3. When the execution of rule $G_{24,3,1}$ finishes, rule $G_{25,3,1}$ is used, object r_1 evolves to objects $r_{1,t}, r_{1,f}, b_1^{(1)}$. If the object $b_1^{(1)}$ presents in membrane 3, the application of rule $G_{26,3,1}$ starts, and object $b_1^{(1)}$ exits the membrane 3. When the execution of rule $G_{26,3,1}$ completes, rule $G_{27,1}$ is used, and object $b_1^{(1)}$ evolves to $b_1^{(2)}$. If the object $b_1^{(2)}$ appears in membrane -1, the application of rule $G_{28,3,1}$ starts, and object $b_1^{(2)}$ enters the membrane 3, changing the polarization from positive to negative. After the execution of rule $G_{28,3,1}$ finishes, rule $G_{29,3,1}$ is enabled and applied, object $b_1^{(2)}$ evolves to $r_{1,f}$, and object $r_{1,f}$ exits the membrane. When the execution of rule $G_{29,3,1}$ completes, the application of rule $G_{30,3,1}$ starts, object $r_{1,f}$ evolves to r_1, and object r_1 enters the membrane 3. With the appearance of object r_1 in membrane 3 having polarization negative, rule $G_{31,3,1}$ is used, and object r_1 evolves to $r_{1,t}, r_{1,f}, b_1^{(3)}$. If the object $b_1^{(3)}$ presents in membrane 3, the application of rule $G_{32,3,1}$ starts, and object $b_1^{(3)}$ exits the membrane. After the execution of rule $G_{32,3,1}$ finishes, rule $G_{33,1}$ is enabled and used, and object $b_1^{(3)}$ evolves to $b_1^{(4)}$. When the execution of rule $G_{33,1}$ completes, rule $G_{34,3,1}$ is used, and object $b_1^{(4)}$ enters the membrane, changing the polarization from negative to positive. When the execution of rule $G_{34,3,1}$ completes, the application of rule $G_{35,3,1}$ starts, object $b_1^{(4)}$ evolves to $r_{2,t}$, and object $r_{2,t}$ exits the membrane. With the appearance of object $r_{2,t}$ in membrane -1, rule $G_{24,3,2}$ is enabled and used. Using the rules from $G_{24,3,2}$ to $G_{35,3,2}$ one by one, two copies of objects $r_{2,t}, r_{2,f}$ will be generated in membrane 3, which is contained in membrane -1. Note that the applications of rules $G_{k,3,1}$ ($24 \leq k \leq 26$, $28 \leq k \leq 32$, $k = 34, 35$), $G_{27,1}$, $G_{33,1}$ in all membranes -1 start and complete at the same step, since the executions of these rules in all membranes -1 take the same time for a given time-mapping e. We also note that the rules $G_{k,3,1}$ ($24 \leq k \leq 26$, $28 \leq k \leq 32$, $k = 34, 35$), $G_{27,1}$, $G_{33,1}$ in membrane -1 are used to ensure that all the objects $r_{j,t}$ (resp. $r_{j,f}$) (if they exist) contained in membrane 2 have been entered the membrane 3 having polarization positive (resp. negative).

The system continues to generate the objects $r_{3,t}r_{3,f}, \ldots, r_{m,t}r_{m,f}$ in each membrane 3, which is contained in membrane -1. When the execution of rule $G_{35,3,m}$ completes, the computation takes at most $16m + 19$ RS-steps: the processes of assigning truth-assignment of variable x_1 as well as looking for the clauses satisfied by the truth-assignment of variable x_1 take at most 10 RS-steps; the applications of rules $G_{25,3,j}, G_{26,3,j}, G_{31,3,j}, G_{32,3,j}$ in membrane -1 and membrane 2 may start at different steps; hence, there are at most $4m$ RS-steps during the applications of rules $G_{25,3,j}, G_{26,3,j}$,

$G_{31,3,j}$, $G_{32,3,j}$ in membrane 2; and in membrane -1, the process of applications of rules from $G_{24,3,j}$ to $G_{35,3,j}$ takes at most $12m$ RS-steps.

With the appearance of object $r_{m+1,t}$ in membrane -1, the application of rule $G_{40,3}$ starts, object $r_{m+1,t}$ evolves to e_1, and object e_1 enters the membrane 3. If object e_1 presents in membrane 3, rule $G_{41,3}$ is enabled and used, membrane 3 is dissolved, and object e_1 will appear in membrane -1 (note that the membranes 3, which are contained in membrane 2, can not be dissolved, because the object $r_{m+1,t}$ cannot be generated in membrane 2). If the object e_1 appears in membrane -1, the application of rule G_{42} starts, membrane -1 is dissolved, and object e_1 will present in membrane 2. When the execution of rule G_{42} completes, rule $G_{43,3}$ is used, and object e_1 enters the membrane 3 having polarization negative. With the appearance of object e_1 in membrane 3, rule $G_{44,3}$ is enabled and applied, object e_1 evolves to e_2, and object e_2 exits the membrane, changing the polarization from negative to positive. After the execution of rule $G_{44,3}$ completes, rule $G_{45,2}$ is used, and object e_2 evolves to one copy of e_3 and two copies of a_3. When the execution of rule $G_{45,2}$ finishes, the applications of rules $G_{46,3}$ and $G_{47,2}$ start at the same step. Using the rule $G_{46,3}$, all the objects a_3 enter membrane 3 changing the polarization from positive to neutral at the same step (there are two copies of object a_3 and two copies of membrane with label 3 in each membrane 2; each object a_3 enters a membrane 3 using the rule $G_{46,3}$ in a maximally parallel manner). Using the rule $G_{47,2}$, object e_3 dissolves the membrane with label 2. In this process, rules $G_{40,3}$, $G_{41,3}$, G_{42}, $G_{43,3}$, $G_{44,3}$, $G_{45,2}$, $G_{46,3}$ are used one by one, there are seven RS-steps. Hence, when the execution of rule $G_{46,3}$ completes, the computation takes at most $16m + 26$ RS-steps, and the system finishes assigning truth-assignment of variables x_1, x_2, and looking for the clauses satisfied by the truth-assignment of variables x_1, x_2. Thus, the processes of assigning truth-assignment of variable x_2 as well as looking for the clauses satisfied by the truth-assignment of variable x_2 take at most $16m + 16$ RS-steps.

When the object a_3 presents in membrane 3, rule $G_{12,3}$ is enabled and applied. Note that the application of rule $G_{12,3}$ in all membranes 3 starts and completes at the same step, since the execution of rule $G_{12,3}$ in all membranes 3 takes the same time $e(G_{12,3})$ for a given time-mapping e. The applications of rules $G_{13,3}$ and $G_{14,3}$ start at the same step, but may complete at different steps. Furthermore, the applications of rules from $G_{17,4}$ to $G_{35,4,j}$ and rules from $G_{40,4}$ to $G_{46,4}$ may start and complete at different steps. Similar with the case of variable x_2, the processes of assigning truth-assignment of variable x_3 as well as looking for the clauses satisfied by the truth-assignment of variable x_3 take at most $16m + 16$ RS-steps.

The system continues to assign the truth-assignment of variables $x_4, x_5, \ldots, x_n$ and look for the clauses satisfied by the truth-assignment of variables. Note that when the system assigns truth-assignment of variable x_n, and looks for the clauses satisfied by the truth-assignment of variable x_n, one copy of object r_j generated in membrane $n + 1$ is enough if the clause C_j is satisfied; hence, in this case, when the system looks for the clauses satisfied by the truth-assignment of variable x_n, rules G_{15}, G_{16} are used. It is important to note that when all of the objects $r_{j,t}, r_{j,f}$ enter the membrane $n + 1$, they will finally evolve to r_j using the rules $r_{36,j}, r_{37,j}, r_{38,j}, r_{39,j}$.

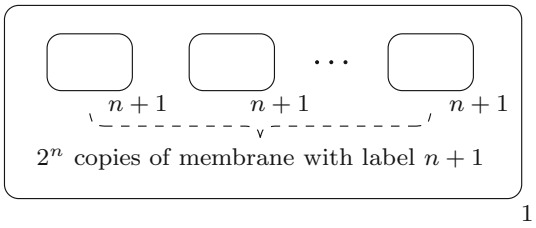

Fig. 1 The membrane structure at the moment when generation phase completes

In general, when the system completes the processes of assigning truth-assignment of variable x_n and checking for the clauses satisfied by the truth-assignment of variable x_n, 2^n copies of membrane with label $n+1$ are generated, which are placed in the membrane 1 (see Fig. 1 for the membrane structure at the moment when the generation phase completes). Note that at the output phase, we will check whether the applications of all rules $G_{47,i} (2 \leq i \leq n)$ complete. The generation phase takes at most $10 + (16m + 16)(n - 1) = 16mn + 16n - 16m - 6$ RS-steps.

Checking phase After the generation phase completes, each membrane with label $n + 1$ contains some of objects from the set $\{r_1, r_2, \ldots, r_m\}$ whose elements denote the corresponding clauses satisfied by the truth assignments of the variables. If there is at least one membrane with label $n + 1$ that contains all objects $r_1, r_2, \ldots, r_m$, this means that the truth assignment from that membrane satisfies all clauses; hence it satisfies formula C. Otherwise, if no membrane with label $n + 1$ contains all objects $r_1, r_2, \ldots, r_m$, the formula C is not satisfiable.

When the execution of rule $G_{45,n}$ completes, all the object a_{n+1} enter membrane $n + 1$, changing the polarization from positive to neutral at the same step (in each membrane n, there are two copies of object a_{n+1} and two copies of membrane with label $n + 1$, each object a_{n+1} enters a membrane $n + 1$ using the rule $G_{46,n+1}$ in a maximally parallel manner). Hence, the object a_{n+1} presents in each membrane $n + 1$ at the same step. After the generation phase completes, the system starts to check whether object r_1 presents in each membrane with label $n + 1$.

Using the rule C_1, the object a_{n+1} creates a new neutral membrane $n + 2$, which contains the object c'_1. If the object r_1 exists, rule $C_{3,1}$ is enabled and used, object r_1 evolves

to r_1', and object r_1' enters the membrane $n+2$, changing its polarization from neutral to positive. When the execution of rule $C_{3,1}$ completes, the application of rule $C_{4,1}$ starts, object c_1' evolves to c_2, and the membrane with label $n+2$ is dissolved. The appearance of object c_2 means that the clause C_1 is satisfied, and the system starts to check whether the object r_2 appears in a membrane $n+1$ (i.e., checking whether the corresponding true assignment satisfies the clause C_2).

After the execution of rule $C_{4,1}$ completes, the application of rule $C_{2,2}$ starts, and object c_2 creates a new neutral membrane $n+3$, which contains the object c_2'. The object c_3 appears only if the corresponding membrane $n+1$ contains object r_2.

The system continues to check whether the objects $r_3, r_4, \ldots, r_m$ appear in membranes $n+1$. If the membrane $n+1$ does not contain the object r_j, $j = 1, 2, \ldots, m$, then the computation in this membrane stops at the time when $C_{3,j}$ is supposed to be applied. In general, the checking phase takes at most $3m$ RS-steps (if there is a membrane $n+1$ containing all objects $r_1, r_2, \ldots, r_m$, the process takes $3m$ RS-steps).

Output phase At step 1, the rule O_1 is used, and object no exits the skin membrane 1, changing its polarization from neutral to positive. When the checking phase completes, we have the following two cases:

- If no object c_{m+1} presents in any membrane with label $n+1$, then the rule O_3 and rule O_4 cannot be applied, and thus, rule O_2 cannot be applied. In this case, when the computation halts, object no remains in the environment, telling us that the formula is not satisfiable.
- If there exists at least one membrane with label $n+1$ that contains object c_{m+1}, then the rule O_3 will be applied, object c_{m+1} evolves to yes, and object yes exits the membrane. At this moment, if the executions of all rules $G_{47,i}$ $(2 \leq i \leq n)$ have not yet completed, then no rule can be started in the system before the executions of all rules $G_{47,i}$ complete. Only when the executions of all rules $G_{47,i}$ complete, the rule O_4 is enabled. By applying the rule O_4, object yes exits the membrane with label 1, changing its polarization from positive to negative. Therefore, the other objects yes remaining in membrane 1 are not able to continue exiting into the environment. After the execution of rule O_4 completes, the rule O_2 is enabled and applied, object no enters membrane 1. In this case, when the computation halts, one copy of yes appears in the environment, telling us that the formula is satisfiable. This process takes three RS-steps.

By the above checking of the computation process, we have the following facts:

- For any time-mapping e, the object yes appears in the environment when the computation halts if and only if the formula C is satisfiable; and the object no appears in the environment when the computation halts if and only if the formula C is not satisfiable. Thus, the system Π_C is time-free sound and time-free complete.
- If the formula C is satisfiable, the computation takes $16mn + 16n - 13m - 3$ RS-steps: it takes $16mn + 16n - 16m - 6$ RS-steps to generate 2^n membranes with label $n+1$; it takes $3m$ RS-steps to check whether all clauses are satisfied by an assignment; the output phase takes 3 RS-steps, and the system halts. If the formula C is not satisfiable, the computation takes at most $16mn+16n-13m-6$ RS-steps (it takes no RS-step at the output phase), and the system halts. Thus, the family of constructed P systems is time-free polynomially bounded.

The family $\boldsymbol{\Pi}$ is polynomially uniform because the construction of P systems described in the proof can be done in polynomial time by a Turing machine:

- size of the alphabet: $7n + 10m + 7$;
- initial number of membranes: 2;
- initial number of objects: 2;
- number of evolution rules: $10mn + 17n - 5m - 1$;
- the maximal length of a rule (the number of symbols necessary to write a rule, both its left and right sides, the membranes, and the polarizations of membranes involved in the rule): $2m + 6$.

Therefore, the SAT problem can be solved in a polynomial RS-steps by a family of recognizer polarization P systems with membrane creation in a time-free manner. □

5 Universality of time-free P systems with membrane creation

In this section, we prove that time-free P systems with membrane creation are universal by simulating a matrix grammar with appearance checking in the binary normal form.

Theorem 2 $\mathrm{PsOP}^{\mathrm{free}}((a), (b), (c), (d), (e)) = \mathrm{PsRE}$.

Proof Consider a matrix grammar $G = (N, T, S, M, F)$ with appearance checking in the binary normal form, with $N = N_1 \cup N_2 \cup \{S, \#\}$ and matrices of the four forms mentioned in Sect. 2. Each type 4 matrix of the form $(X \to \lambda, A \to x), X \in N_1, A \in N_2, x \in T^*$ can be replaced by $(X \to Z, A \to x)$, where Z is a new symbol. The obtained grammar is denoted by G'. Assume that M contains $n+1$ matrices, injectively labelled with $m_0, m_1, \ldots, m_n$. Let us also assume that we have n_1 matrices of the form $m_i = (X \to Y, A \to x)$, $1 \leq i \leq n_1$, with $X \in N_1, Y \in N_1 \cup \{Z\}, A \in N_2, x \in (N_2 \cup T)^*$, and n_2 matrices of the

form $m_i = (X \to Y, A \to \#)$, $n_1 + 1 \le i \le n$, with $X, Y \in N_1$, $A \in N_2$, such that $n_1 + n_2 = n$.

We construct the time-free P system with membrane creation

$$\Pi = (V, T, H, \mu, w_0, R, i_{\text{out}}),$$

where

$$V = N_1 \cup N_2 \cup T \cup \{Z, \dagger\} \cup \{X', X'' \mid X \in N_1\} \cup \{d_i, x_i \mid 1 \le i \le n_1\},$$
$$H = \{0, 1, 2, \ldots, 2n_1 + n_2\},$$
$$\mu = [\ ]_0^0,$$
$$w_0 = XA, \text{ for the initial matrix of} G,$$
$$i_{\text{out}} = 0,$$

and the set R of rules is as follows.

Let e be arbitrary time-mapping from R to $\mathbb{N}$, representing the execution times of the rules in R.

- For each matrix $m_i = (X \to Y, A \to x)$, $1 \le i \le n_1$, with $X \in N_1$, $Y \in N_1 \cup \{Z\}$, $A \in N_2$, $x \in (N_2 \cup T)^*$, we introduce the following rules:

  $r_1 : [X \to [X']_i^0]_0^0$,
  $r_2 : A[\ ]_i^0 \to [x_i]_i^+$,
  $r_3 : [x_i \to d_i x]_i^+$,
  $r_4 : [d_i \to [\ ]_{n+i}^0]_i^+$,
  $r_5 : X'[\ ]_{n+i}^0 \to [Y]_{n+i}^0$,
  $r_6 : [Y]_{n+i}^0 \to Y$,
  $r_7 : [Y]_i^+ \to Y$.

- For each matrix $m_i = (X \to Y, A \to \#)$, $n_1 + 1 \le i \le n$, with $X, Y \in N_1$, $A \in N_2$, we introduce the following rules:

  $r_8 : [X \to [X']_i^0]_0^0$,
  $r_9 : [X']_i^0 \to [\ ]_i^- X''$,
  $r_{10} : A[\ ]_i^- \to [\dagger]_i^0$,
  $r_{11} : [X'' \to Y]_0^0$.

- We also consider the following rules:

  $r_{12} : [\dagger \to \dagger]_i^0$, $n_1 + 1 \le i \le n$,
  $r_{13} : [a]_0^0 \to [\ ]_0^0 a$, for all $a \in T$,
  $r_{14} : [\alpha \to \alpha]_0^0$, for all $\alpha \in N_1 \cup N_2$.

In what follows, we show how the above constructed P system Π simulates the matrix grammar G.

The simulation of a matrix $m_i = (X \to Y, A \to x)$, $1 \le i \le n_1$, with $X \in N_1$, $Y \in N_1 \cup \{Z\}$, $A \in N_2$, $x \in (N_2 \cup T)^*$, is done as follows.

At some step, rule r_1 is used, the object X creates a new neutral membrane with label i, where the object X' is placed. For any time-mapping e, the membrane i containing object X' will appear in membrane 0, which is independent of the execution time of rule r_1.

When the execution of rule r_1 completes, the applications of rule r_2 starts, only one copy of object A evolves to x_i, and object x_i enters the membrane i, and changing the polarization from neutral to positive. For any time-mapping e, the execution time of the rule r_2 has no influence on the appearance of the object x_i in membrane i.

When the execution of rule r_2 completes, rule r_3 starts to use, object x_i evolves to d_i, x. For any time-mapping e, the appearance of objects d_i, x in membrane i is independent of the execution time of rule r_3.

With the appearance of object d_i in membrane i, rule r_4 is enabled and used, object d_i is consumed, and a new neutral membrane with label $n + i$ is created. For any time-mapping e, the generation of membrane $n + i$ is independent of the execution time of rule r_4.

After finishing the execution of rule r_4, rule r_5 is enabled and applied, object X' evolves to Y, which enters the membrane $n + i$. Clearly, for any time-mapping e, the appearance of object Y in membrane $n + i$ is independent of the execution time of rule r_5.

When the execution of rule r_5 completes, the application of rule r_6 starts, the membrane with label $n + i$ is dissolved. With the appearance of object Y in membrane i, rule r_7 is enabled and used, membrane i is dissolved, and objects Y, x appear in membrane 0 at the same step. Note that in any moment, there is at most one and only one copy of object from the set N_1 in membrane 0, and object Y is available in membrane 0 only after completing the simulation of the matrix. For any time-mapping e, the execution times of the rules r_6, r_7 have no influence on the appearance of the objects Y, x in membrane 0.

We remark that in the simulation of a matrix $m_i = (X \to Y, A \to x)$, the rules may have different execution times for different time-mapping e; however, before the execution of the current rule completes (before the corresponding objects appear or membranes generate by the current rule), the follow-up rules cannot be used, and this guarantees the robustness of the system in the sense of time-independency, which is also used in the following simulations of matrix $m_i = (X \to Y, A \to \#)$.

The simulation of a matrix $m_i = (X \to Y, A \to \#)$, $n_1 + 1 \le i \le n$, with $X, Y \in N_1$, $A \in N_2$, is done as follows:

At some step, rule r_8 is used, the object X creates a new neutral membrane with label i, which contains the object X'. When the execution of rule r_8 completes, the application of rule r_9 starts, and object X' evolves to X'', which is sent out of the membrane i, changing the polarization from neutral to negative. When the execution of rule r_9 completes, we have the following two cases:

- if the object A presents in membrane 0, then after the execution of rule r_9 completes, the applications of rules r_{10} and r_{11} start at the same time, but they may complete at different steps due to the fact that the execution time associated with rules r_9 and r_{10} can be different. Using the rule r_{10}, object A evolves to the trap object †, and object † enters the membrane i. Once the object † appears in membrane i, rule r_{12} is enabled and used, the computation will never stop. Using the rule r_{11}, object X'' evolves to object Y. In this process, we can check that for any time-mapping e, only when the generation of corresponding objects and the membranes finish, the follow-up rules can be applied, and the appearance of object † in membrane i is independent of the execution times of rules r_8, r_9, r_{10}, r_{11}.
- If the object A does not appear in membrane 0. In this case, when the execution of rule r_9 finishes, only the rule r_{11} is enabled. Using rule r_{11}, object X' evolves to X''. In this process, we can check that for any time-mapping e, the rules r_8, r_9, r_{11} are used one by one, and the appearance of object Y in membrane 0 does not depend on the execution times of rules r_8, r_9, r_{11}.

Therefore, the simulation of the matrix $m_i = (X \rightarrow$

proved that time-free P systems with membrane creation are computationally universal.

The solution to the SAT problem in this work is semi-uniform in the sense that P systems with membrane creation are constructed from the instances of the problem. It remains open how we can construct a uniform time-free solution to the SAT problem in the sense that P systems with membrane creation are constructed from the size of instances of the problem.

The P systems constructed in Sect. 4 and in Sect. 5 have polarizations on membranes. It remains open whether P systems with membrane creation can still solve the SAT problem or the universality result still holds in the time-free context.

The P systems with membrane creation constructed in Theorem 1 and in Theorem 2 have rules of all types. It is of interest to investigate whether the types of rules in Theorem 1 and in Theorem 2 are optimal.

Acknowledgments This is an extended version of a manuscript presented at the conference BIC-TA 2014. The work of L. Pan was supported by National Natural Science Foundation of China (61033003, 91130034 and 61320106005), Ph.D. Programs Foundation of Ministry of Education of China (20100142110072 and 20120142130008), and Natural Science Foundation of Hubei Province (2011CDA027). The work of M.J. Pérez-Jiménez was supported by "Ministerio de Economía y Competitividad" of Spanish government (TIN2012-37434), cofunded by FEDER funds.

References

Cavaliere M, Sburlan D (2005) Time-independent P systems. In: Mauri G, Păun Gh, Pérez-Jiménez MJ, Rozenberg G, Salomaa A (eds), Membrane Computing. Lecture notes in computer science, vol 3365, pp 239–258

Garey MR, Johnson DJ (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H Freeman, New York

Gheorghe M, Păun G, Pérez-Jiménez MJ (2013) Research frontiers of membrane computing: open problems and research topics. Section 12 Cavaliere M, time-free solutions to hard computational problems. Int J Found Comput Sci 24(5):579–582

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Romero-Campero FJ (2005) A linear solution of subset sum problem by using membrane creation. In: Mira J, Álvarez JR (eds), Mechanisms, symbols, and models underlying cognition. Lecture notes in computer science, vol 3561, pp 258–267

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Romero-Campero FJ (2006) A linear time solution for QSAT with membrane creation. In: Freund R, Păun Gh, Rozenberg G, Salomaa A (eds), Membrane computing. Lecture notes in computer science, vol 3850, pp 241–252

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Romero-Campero FJ (2007) A uniform solution to SAT using membrane creation. Theor Comput Sci 371:54–61

Mutyam M, Krithivasan K (2001) P systems with membrane creation: universality and efficiency. In: Margenstern M, Rogozhin Y (eds), Machines, computations, and universality. Lecture notes in computer science, vol 2055, pp 276–287

Pan L, Păun G (2010) Spiking neural P systems: an improved normal form. Theor Comput Sci 411:906–918

Pan L, Zeng X, Zhang X (2011) Time-free spiking neural P systems. Neural Comput 23:1–23

Păun G (2000) Computing with membranes. J Comput Syst Sci 61(1):108–143 (Also in Turku Center for Computer Science-TUCS, Report 208, November 1998)

Păun G (2001) P systems with active membranes: attacking NP-complete problems. J Auto Langua Comb 6(1):75–90

Păun G (2002) Membrane computing: an introduction. Springer Science & Business Media, Berlin

Păun G, Rozenberg G, Salomaa A (eds) (2010) The Oxford handbook of membrane computing. Oxford University Press, New York

Rozenberg G, Salomaa A (eds) (1997) Handbook of formal languages. Springer, New York

Song T, Macías-Ramos LF, Pan L, Pérez-Jiménez MJ (2014) Time-free solution to SAT problem using P systems with active membranes. Theor Comput Sci 529:61–68

Song B, Pan L (2015a) Computational efficiency and universality of timed P systems with active membranes. Theor Comput Sci 567:74–86

Song B, Song T, Pan L (2015b) Time-free solution to SAT problem by P systems with active membranes and standard cell division rules. Nat Comput. doi:10.1007/s11047-014-9471-4

Song B, Song T, Pan L (2015c) A time-free uniform solution to subset sum problem by tissue P systems with cell division. Math Struct Comput Sci. doi:10.1017/S0960129515000018

Song T, Wang X, Zheng H (2013) Time-free solution to Hamilton path problems using P systems with d-division. J Appl Math Article ID 975798

Zhang X, Luo B, Fang X, Pan L (2012) Sequential spiking neural P systems with exhaustive use of rules. BioSystems 108:52–62