SCIENCE CHINA
Information Sciences

• Supplementary File •

Monodirectional evolutional symport tissue P systems with channel states and cell division

Bosheng SONG[1], Kenli LI[1*], Xiangxiang ZENG[1],
Mario J. PÉREZ-JIMÉNEZ[2] & Claudio ZANDRON[3]

[1]*College of Computer Science and Electronic Engineering, Hunan University, Changsha* 410082*, Hunan, China;*
[2]*Department of Computer Science and Artificial Intelligence,*
Universidad de Sevilla, Avda. Reina Mercedes s/n, Sevilla 41012*, Spain;*
[3]*Dipartimento di Informatica, Sistemistica e Comunicazione,*
Università degli Studi di Milano-Bicocca, Viale Sarca 336/14, Milan 20126*, Italy*

Appendix A Preliminaries

Definition 1. A recognizer MESTCD P system of degree $q \geqslant 1$ is a tuple

$$\Pi = (\Gamma, K, \Sigma, \mathcal{E}, \mathcal{M}_1, \ldots, \mathcal{M}_q, ch, (s_{(i,j)})_{(i,j)\in ch}, \mathcal{R}, i_{in}, i_{out}),$$

where

- Γ is a finite alphabet, which includes two special objects `yes` and `no`;
- K is a set of states;
- Σ is an input alphabet, and $\Sigma \subseteq \Gamma$;
- $\mathcal{E}$ is a set of objects initially placed in the environment;
- $\mathcal{M}_1, \ldots, \mathcal{M}_q$ are finite multisets over $\Gamma \setminus \Sigma$;
- ch is a channels set among regions, ch includes at most one of channels $(i, j), (j, i)$ (0 represents environment);
- $s_{i,j}$ is an state initially located on channel $(i, j) \in ch$;
- $\mathcal{R}$ is a set of rules;
- $i_{in} \in \{1, \ldots, q\}$ is the input cell, and $i_{out} = 0$;
- all computations halt;
- if $\mathcal{C}$ is a computation of the system, then either object `yes` or object `no` (but not both) must been released into the output region only at the last step of the computation.

Next we present the notion of a uniform solution to a decision problem, one is referred to [1–4] for further information.

Definition 2. A family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer MESTCD P systems can solve a decision problem $X = (I_X, \theta_X)$ in a uniform way in polynomial time if the following prerequisites hold:

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

Appendix B The Proof of Theorem 1

Proof. Let $M = (m, H, l_0, l_h, I)$ be a register machine. We design the MESTC P system Π (see Figure B1) to simulate M.

$\Pi = (\Gamma, K, \mathcal{E}, \mathcal{M}_1, \mathcal{M}_2, \{(1, 0), (0, 2), (2, 1)\}, s, s, s, \mathcal{R}_{(1,0)}, \mathcal{R}_{(0,2)}, \mathcal{R}_{(2,1)}, 1)$, where

- $\Gamma = \{a_i \mid 1 \leqslant i \leqslant m\} \cup \{l, l', l'', l''', l^{iv}, l^{v}, l^{vi}, l^{vii} \mid l \in H\}$,
- $K = \{s, s'\}$,
- $\mathcal{E} = \emptyset$,
- $\mathcal{M}_1 = \{l_0\}$, $\mathcal{M}_2 = \emptyset$,

* Corresponding author (email: lkl@hnu.edu.cn)

the set of evolutional symport rules is designed as follows.

Every ADD (resp., SUB) instruction in M can be simulated in 3 steps (resp., 10 steps (when the value of register r is larger than 0) or 9 steps (when the value of register r is 0)) in Π. At the initial moment, cell 1 includes an object l_0, cell 2 and environment do not have object, at initial configuration, the state of channels $(1,0),(0,2),(2,1)$ is s. When the system receives object l_h in cell 1, the program machine M halts, and no rule is valid in the system. If the P system Π arrives halting configuration, the number of a_1 located in cell 1 is considered as the result for M.

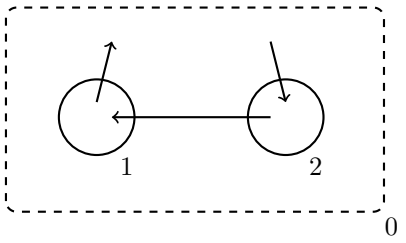

Figure B1 Membrane structure for designed MESTC P system, where all cells are situated in environment (indicated as 0), circles indicate as cells, numbers indicate as labels of cells, arrows indicate directions in which objects move.

- For every ADD instruction l_i, we devise the following rules to simulate it:
- $\mathcal{R}_{(1,0)}$ includes the following rule:
  $r_1 : (s, l_i \to l_i'/\lambda, s)$;
- $\mathcal{R}_{(0,2)}$ includes the following rule:
  $r_2 : (s, l_i'/\lambda, s)$;
- $\mathcal{R}_{(2,1)}$ includes the following rules:
  $r_3 : (s, l_i' \to a_r l_j/\lambda, s)$,
  $r_4 : (s, l_i' \to a_r l_k/\lambda, s)$.

An ADD instruction can be easy simulated by the system. Object l_i is revised to l_i', and it will enter cell 2. When object l_i' arises in cell 2, rule r_3 or r_4 is applied non-deterministically, object l_i' is revised to a_r, l_j or a_r, l_k, so the system will simulate l_j or l_k at the next step.

- For every SUB instruction l_i, we devise the following rules to simulate it:
- $\mathcal{R}_{(1,0)}$ includes the following rules:
  $r_5 : (s, l_i \to l_i'/\lambda, s)$,
  $r_6 : (s, l_i''/\lambda, s')$,
  $r_7 : (s', l_i''' a_r \to l_i^{iv}/\lambda, s)$,
  $r_8 : (s, l_i^{iv} l_i^v \to l_i^{vi}/\lambda, s)$,
  $r_9 : (s', l_i''' l_i^v \to l_i^{vii}/\lambda, s)$;
- $\mathcal{R}_{(0,2)}$ includes the following rules:
  $r_{10} : (s, l_i'/\lambda, s)$,
  $r_{11} : (s, l_i''/\lambda, s)$,
  $r_{12} : (s, l_i^{iv}/\lambda, s)$,
  $r_{13} : (s, l_i^{vi}/\lambda, s)$,
  $r_{14} : (s, l_i^{vii}/\lambda, s)$;
- $\mathcal{R}_{(2,1)}$ includes the following rules:
  $r_{15} : (s, l_i' \to l_i'' l_i'''/\lambda, s)$,
  $r_{16} : (s, l_i'' \to l_i^v/\lambda, s)$,
  $r_{17} : (s, l_i^{iv}/\lambda, s)$,
  $r_{18} : (s, l_i^{vi} \to l_j/\lambda, s)$,
  $r_{19} : (s, l_i^{vii} \to l_k/\lambda, s)$.

When a SUB instruction l_i is present in system, the system starts to simulate l_i. Object l_i is revised to l_i', which will enter cell 2. When cell 2 gets object l_i', and cell 1 gets object l_i', such object is modified to l_i'', l_i'''. At step 4, object l_i'' is transferred to environment by employing rule r_6, channel $(1,0)$ changes its state to s'. Next the following cases are occurred in the system.

- Object a_r arises in cell 1. When channel $(1,0)$ has state s', rule r_7 is employed, objects a_r, l_i''' are transferred to environment, which are modified to l_i^{iv}; in addition, cell 2 gets object l_i''. Next rule r_{12} is employed, cell 2 receives object l_i^{iv}; if object l_i'' is present in cell 2, rule r_{16} is employed, object l_i'' is arisen in cell 1, and it is modified to l_i^v. At step 7, rule r_{17} is employed, cell 1 obtains object l_i^{iv}. With object l_i^{iv} presents in cell 1, rule r_8 is employed, objects l_i^{iv}, l_i^v are transferred to environment, which are revised to l_i^{vi}. At step 9, cell 2 obtains an object l_i^{vi} by employing rule r_{13}. When object l_i^{vi} presents in cell 2, object l_i^{vi} is transferred to cell 1, meanwhile object l_i^{vi} is modified to l_j. So in this process of simulation, one a_r is reduced (the value of register r is reduced by 1), consequently the system begins to simulate next instruction l_j (see Table B1).
- Object a_r does not appear in cell 1. So at step 5, rule r_{11} is employed, object l_i'' will be transferred to cell 2 and to cell 1, then object l_i'' is modified to l_i^v at next step. Rule r_9 is employed at step 7, objects l_i''', l_i^v are transferred to environment, which will be modified to l_i^{vii}. Next cell 2 gets object l_i^{vii} by using rule r_{14}. When object l_i^{vii} occurs in cell 2, rule r_{19} is employed, object l_i^{vii} is transferred to cell 1, which is modified to l_k. Consequently instruction l_k will be started to simulate (see Table B2).

Therefore the system simulates a SUB instruction accurately for both cases.

At some moment, when object l_h is present in cell 1, there is no available rule in Π, and the system halts. The computing result of the system is the number of a_1 located in cell 1. Consequently $N(M) = N(\Pi)$.

Table B1 The application of rules in $\mathcal{R}_{(1,0)}, \mathcal{R}_{(0,2)}$ and $\mathcal{R}_{(2,1)}$, the evolution of channel states $s_{(1,0)}, s_{(0,2)}$ and $s_{(2,1)}$, and the rewriting of multisets $\mathcal{M}_1$ and $\mathcal{M}_2$ in cells 1 and 2, respectively, during the simulation of a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ with register r not empty, where z, z' are multisets of objects from the set $\{a_1, \ldots, a_m\}$, $z = z' \cup \{a_r\}$.

Step	$\mathcal{R}_{(1,0)}$	$\mathcal{R}_{(0,2)}$	$\mathcal{R}_{(2,1)}$	$s_{(1,0)}$	$s_{(0,2)}$	$s_{(2,1)}$	$\mathcal{M}_1$	$\mathcal{M}_2$	Env
0	–	–	–	s	s	s	$z \cup \{l_i\}$	$\emptyset$	$\emptyset$
1	r_5	–	–	s	s	s	z	$\emptyset$	$\{l_i'\}$
2	–	r_{10}	–	s	s	s	z	$\{l_i'\}$	$\emptyset$
3	–	–	r_{15}	s	s	s	$z \cup \{l_i'', l_i'''\}$	$\emptyset$	$\emptyset$
4	r_6	–	–	s'	s	s	$z \cup \{l_i'''\}$	$\emptyset$	$\{l_i''\}$
5	r_7	r_{11}	–	s	s	s	z'	$\{l_i''\}$	$\{l_i^{iv}\}$
6	–	r_{12}	r_{16}	s	s	s	$z' \cup \{l_i^v\}$	$\{l_i^{iv}\}$	$\emptyset$
7	–	–	r_{17}	s	s	s	$z' \cup \{l_i^{iv}, l_i^v\}$	$\emptyset$	$\emptyset$
8	r_8	–	–	s	s	s	z'	$\emptyset$	$\{l_i^{vi}\}$
9	–	r_{13}	–	s	s	s	z'	$\{l_i^{vi}\}$	$\emptyset$
10	–	–	r_{18}	s	s	s	$z' \cup \{l_j\}$	$\emptyset$	$\emptyset$

Table B2 The application of rules in $\mathcal{R}_{(1,0)}, \mathcal{R}_{(0,2)}$ and $\mathcal{R}_{(2,1)}$, the evolution of channel states $s_{(1,0)}, s_{(0,2)}$ and $s_{(2,1)}$, and the rewriting of multisets $\mathcal{M}_1$ and $\mathcal{M}_2$ in cells 1 and 2, respectively, during the simulation of a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ with register r empty, where z is a multiset of objects from the set $\{a_1, \ldots, a_m\}$.

Step	$\mathcal{R}_{(1,0)}$	$\mathcal{R}_{(0,2)}$	$\mathcal{R}_{(2,1)}$	$s_{(1,0)}$	$s_{(0,2)}$	$s_{(2,1)}$	$\mathcal{M}_1$	$\mathcal{M}_2$	Env
0	–	–	–	s	s	s	$z \cup \{l_i\}$	$\emptyset$	$\emptyset$
1	r_5	–	–	s	s	s	z	$\emptyset$	$\{l_i'\}$
2	–	r_{10}	–	s	s	s	z	$\{l_i'\}$	$\emptyset$
3	–	–	r_{15}	s	s	s	$z \cup \{l_i'', l_i'''\}$	$\emptyset$	$\emptyset$
4	r_6	–	–	s'	s	s	$z \cup \{l_i'''\}$	$\emptyset$	$\{l_i''\}$
5	–	r_{11}	–	s'	s	s	$z \cup \{l_i'''\}$	$\{l_i''\}$	$\emptyset$
6	–	–	r_{16}	s'	s	s	$z \cup \{l_i''', l_i^v\}$	$\emptyset$	$\emptyset$
7	r_9	–	–	s	s	s	z	$\emptyset$	$\{l_i^{vii}\}$
8	–	r_{14}	–	s	s	s	z	$\{l_i^{vii}\}$	$\emptyset$
9	–	–	r_{19}	s	s	s	$z \cup \{l_k\}$	$\emptyset$	$\emptyset$

Appendix C The Proof of Theorem 2

Proof. Let $M = (m, H, l_0, l_h, I)$ be a register machine. We design the MESTC P system Π (Figure C1) to simulate M.

$\Pi = (\Gamma, K, \mathcal{E}, \mathcal{M}_1, \ldots, \{(1, add_1), \ldots, (sub_i', 1)\}, s, \ldots, s, \mathcal{R}_{(0,1)}, \ldots, 1)$, where

- $\Gamma = \{a_i \mid 1 \leqslant i \leqslant m\} \cup \{l, l', l'', l''' \mid l \in H\}$,
- $K = \{s, s'\}$,
- $\mathcal{E} = \{a_i \mid 1 \leqslant i \leqslant m\}$,
- $\mathcal{M}_1 = \{l_0\}$, $\mathcal{M}_{add_i} = \emptyset$, $\mathcal{M}_{add_i'} = \emptyset$, $\mathcal{M}_{sub_i} = \emptyset$, $\mathcal{M}_{sub_i'} = \emptyset$,

and the set of rules is given below.

Every ADD (resp., SUB) instruction in M can be simulated in 8 steps (resp., 6 steps) in Π. At the initial moment, cell 1 has object l_0, all the other cells do not have objects, environment includes multiset of objects $\{a_i \mid 1 \leqslant i \leqslant m\}$ (note that every kind of objects in $\mathcal{E}$ is valid in arbitrary number of copies). Initially, every existing channel has state s. When the system appears object l_h in cell 1, the program machine M halts, and no rule is applicable in the system. If the P system Π reaches the halting configuration, at this moment, the number of a_1 located in cell 1 is considered the result for M.

- For every ADD instruction l_i, we devise the following rules to simulate it:
- $\mathcal{R}_{(1,add_i)}$ includes the following rule:

  $r_1 : (s, l_i/\lambda, s)$;
- $\mathcal{R}_{(add_i, add_i')}$ includes the following rules:

  $r_2 : (s, l_i/\lambda, s)$,

  $r_3 : (s, a_r/\lambda, s')$,

  $r_4 : (s', l_i'/\lambda, s)$;
- $\mathcal{R}_{(add_i', 0)}$ includes the following rule:

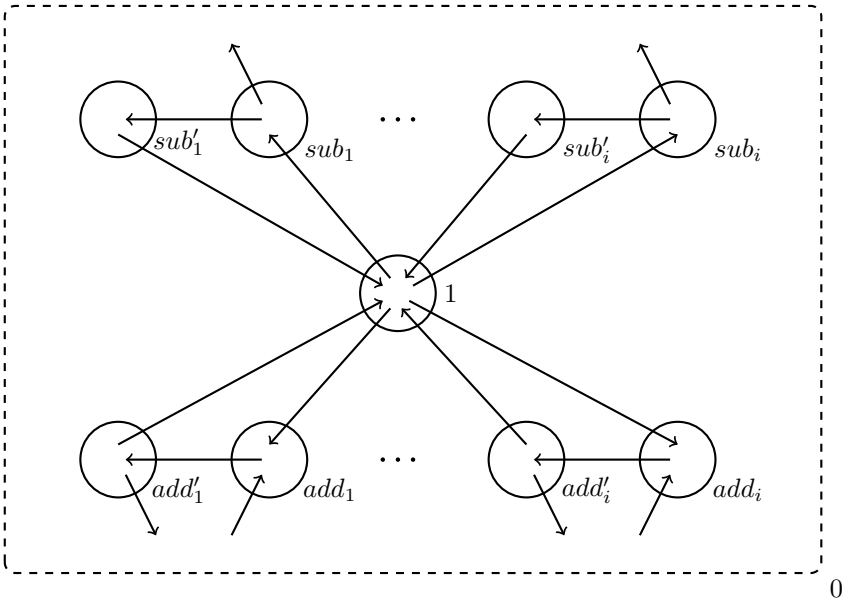

Figure C1 The membrane structure for designed MESTC P system, where all cells are suited in environment (indicated as 0), circles indicate as cells, numbers beside circles indicate as labels of cells, and arrows represent directions in which objects move.

$r_5 : (s, l_i \to l'_i/\lambda, s)$;

- $\mathcal{R}_{(0,add_i)}$ includes the following rules:
  $r_6 : (s, l'_i/\lambda, s')$,
  $r_7 : (s', a_r/\lambda, s)$;
- $\mathcal{R}_{(add'_i,1)}$ includes the following rules:
  $r_8 : (s, a_r/\lambda, s)$,
  $r_9 : (s, l'_i \to l_j/\lambda, s)$,
  $r_{10} : (s, l'_i \to l_k/\lambda, s)$.

At first step, cell add_i obtains object l_i, and this object is moved to cell add'_i later. When cell add'_i presents object l_i, object l_i is transferred to environment, which is modified to l'_i. Next, cell add_i obtains object l'_i, and state for channel $(0, add_i)$ is revised to s'. If channel $(0, add_i)$ has state s', an object a_r is transferred to cell add_i, and state s is achieved on channel $(0, add_i)$. Next, object a_r is introduced into cell add'_i, and state s' is achieved on channel (add_i, add'_i). If channel (add_i, add'_i) has state s', then rule r_4 is employed, object l'_i is introduced into cell add'_i, and state s is achieved on channel (add_i, add'_i) from s'; in addition, rule r_8 is employed, cell 1 obtains an object a_r. At step 8, rule r_9 or r_{10} is applied non-deterministically, object l'_i is revised to l_j or l_k, and the system begins to simulate l_j or l_k (see Table C1).

Table C1 The application of rules in $\mathcal{R}_{(1,add_i)}, \mathcal{R}_{(add_i,add'_i)}, \mathcal{R}_{(add'_i,0)}, \mathcal{R}_{(0,add_i)}$ and $\mathcal{R}_{(add'_i,1)}$, the evolution of channel states $s_{(1,add_i)}, s_{(add_i,add'_i)}, s_{(add'_i,0)}, s_{(0,add_i)}$ and $s_{(add'_i,1)}$, and the rewriting of multisets $\mathcal{M}_1, \mathcal{M}_{add_i}$ and $\mathcal{M}_{add'_i}$ in cells 1, add_i and add'_i, respectively, during the simulation of an ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$, where z, z', z'', z''' are multisets of objects from the set $\{a_1, \ldots, a_m\}$, where $z''' = z \cup \{a_r\}$, $z' = z'' \cup \{a_r\}$.

Step	$\mathcal{R}_{(1,add_i)}$	$\mathcal{R}_{(add_i,add'_i)}$	$\mathcal{R}_{(add'_i,0)}$	$\mathcal{R}_{(0,add_i)}$	$\mathcal{R}_{(add'_i,1)}$	$s_{(1,add_i)}$	$s_{(add_i,add'_i)}$	$s_{(add'_i,0)}$	$s_{(0,add_i)}$	$s_{(add'_i,1)}$	$\mathcal{M}_1$	$\mathcal{M}_{add_i}$	$\mathcal{M}_{add'_i}$	Env
0	–	–	–	–	–	s	s	s	s	s	$z \cup \{l_i\}$	$\emptyset$	$\emptyset$	z'
1	r_1	–	–	–	–	s	s	s	s	s	z	$\{l_i\}$	$\emptyset$	z'
2	–	r_2	–	–	–	s	s	s	s	s	z	$\emptyset$	$\{l_i\}$	z'
3	–	–	r_5	–	–	s	s	s	s	s	z	$\emptyset$	$\emptyset$	$z' \cup \{l'_i\}$
4	–	–	–	r_6	–	s	s	s	s'	s	z	$\{l'_i\}$	$\emptyset$	z'
5	–	–	–	r_7	–	s	s	s	s	s	z	$\{l'_i, a_r\}$	$\emptyset$	z''
6	–	r_3	–	–	–	s	s'	s	s	s	z	$\{l'_i\}$	$\{a_r\}$	z''
7	–	r_4	–	–	r_8	s	s	s	s	s	z'''	$\emptyset$	$\{l'_i\}$	z''
8	–	–	–	–	r_9 or r_{10}	s	s	s	s	s	$z''' \cup \{l_j\}$ or $z''' \cup \{l_k\}$	$\emptyset$	$\emptyset$	z''

- For every SUB instruction l_i, we devise the following rules to simulate it:
- $\mathcal{R}_{(1,sub_i)}$ includes the following rules:
  $r_{11} : (s, l_i/\lambda, s')$,
  $r_{12} : (s', a_r/\lambda, s)$,
  $r_{13} : (s, l'_i \to l''_i/\lambda, s)$,
  $r_{14} : (s', l'_i \to l'''_i/\lambda, s)$;
- $\mathcal{R}_{(sub_i,sub'_i)}$ includes the following rules:
  $r_{15} : (s, l_i/\lambda, s)$,

$r_{16} : (s, l_i''/\lambda, s)$,
$r_{17} : (s, l_i'''/\lambda, s)$;

- $\mathcal{R}_{(sub_i,0)}$ includes the following rule:
  $r_{18} : (s, a_r/\lambda, s)$;
- $\mathcal{R}_{(sub_i',1)}$ includes the following rules:
  $r_{19} : (s, l_i \to l_i'/\lambda, s)$,
  $r_{20} : (s, l_i'' \to l_j/\lambda, s)$,
  $r_{21} : (s, l_i''' \to l_k/\lambda, s)$.

When a SUB instruction l_i presents in Π, the system starts to simulate l_i. At first step, object l_i is transferred to cell sub_i, state s' for channel $(1, sub_i)$ is achieved. Next the following cases are occurred in the system.

- Object a_r arises in cell 1. Rule r_{12} is employed, an object a_r is transferred to sub_i, state s on channel $(1, sub_i)$ is achieved; in addition, cell sub_i' receives object l_i. Next, rule r_{18} is employed, and environment receives an object a_r; in addition, object l_i is shifted to cell 1 by employing rule r_{19}, and object l_i is modified to l_i' during this process. Next, object l_i' is transferred to cell sub_i by employing rule r_{13}, which is revised to l_i'' during this process. Next l_i'' is introduced into cell sub_i' by employing rule r_{16}. If cell sub_i' comprises object l_i'', rule r_{20} is employed, cell 1 gets object l_i'', and it is modified to l_j. So in this process of simulation, the number of a_r is reduced by one (the value of register r is reduced by 1), consequently Π begins to simulate l_j (see Table C2).

Table C2 The application of rules in $\mathcal{R}_{(1,sub_i)}, \mathcal{R}_{(sub_i,sub_i')}, \mathcal{R}_{(sub_i,0)}$ and $\mathcal{R}_{(sub_i',1)}$, the evolution of channel states $s_{(1,sub_i)}, s_{(sub_i,sub_i')}, s_{(sub_i,0)}$ and $s_{(sub_i',1)}$, and the rewriting of multisets $\mathcal{M}_1, \mathcal{M}_{sub_i}$ and $\mathcal{M}_{sub_i'}$ in cells 1, sub_i and sub_i', respectively, during the simulation of a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ with register r not empty, where z, z', z'' are multisets of objects from the set $\{a_1, \ldots, a_m\}$, where $z = z'' \cup \{a_r\}$.

Step	$\mathcal{R}_{(1,sub_i)}$	$\mathcal{R}_{(sub_i,sub_i')}$	$\mathcal{R}_{(sub_i,0)}$	$\mathcal{R}_{(sub_i',1)}$	$s_{(1,sub_i)}$	$s_{(sub_i,sub_i')}$	$s_{(sub_i,0)}$	$s_{(sub_i',1)}$	$\mathcal{M}_1$	$\mathcal{M}_{sub_i}$	$\mathcal{M}_{sub_i'}$	Env
0	–	–	–	–	s	s	s	s	$z \cup \{l_i\}$	$\emptyset$	$\emptyset$	z'
1	r_{11}	–	–	–	s'	s	s	s	z	$\{l_i\}$	$\emptyset$	z'
2	r_{12}	r_{15}	–	–	s	s	s	s	z''	$\{a_r\}$	$\{l_i\}$	z'
3	–	–	r_{18}	r_{19}	s	s	s	s	$z'' \cup \{l_i'\}$	$\emptyset$	$\emptyset$	$z' \cup \{a_r\}$
4	r_{13}	–	–	–	s	s	s	s	z''	$\{l_i''\}$	$\emptyset$	$z' \cup \{a_r\}$
5	–	r_{16}	–	–	s	s	s	s	z''	$\emptyset$	$\{l_i''\}$	$z' \cup \{a_r\}$
6	–	–	–	r_{20}	s	s	s	s	$z'' \cup \{l_j\}$	$\emptyset$	$\emptyset$	$z' \cup \{a_r\}$

- Object a_r does not appear in cell 1. The system begins to apply rule r_{15}, cell sub_i' acquires object l_i. Next in cell sub_i', cell 1 gets object l_i, which is modified to l_i'. Next, rule r_{14} is employed, cell sub_i gets object l_i', it is modified to l_i''', state s on channel $(1, sub_i)$ is achieved. Next cell sub_i' gets object l_i'''. If cell sub_i' comprises object l_i''', l_i''' is transferred to cell 1 by applying rule r_{21}, so such object is modified to l_k. Consequently instruction l_k will be started to simulate (see Table C3).

Table C3 The application of rules in $\mathcal{R}_{(1,sub_i)}, \mathcal{R}_{(sub_i,sub_i')}, \mathcal{R}_{(sub_i,0)}$ and $\mathcal{R}_{(sub_i',1)}$, the evolution of channel states $s_{(1,sub_i)}, s_{(sub_i,sub_i')}, s_{(sub_i,0)}$ and $s_{(sub_i',1)}$, and the rewriting of multisets $\mathcal{M}_1, \mathcal{M}_{sub_i}$ and $\mathcal{M}_{sub_i'}$ in cells 1, sub_i and sub_i', respectively, during the simulation of a SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ with register r empty, where z, z' are multisets of objects from the set $\{a_1, \ldots, a_m\}$.

Step	$\mathcal{R}_{(1,sub_i)}$	$\mathcal{R}_{(sub_i,sub_i')}$	$\mathcal{R}_{(sub_i,0)}$	$\mathcal{R}_{(sub_i',1)}$	$s_{(1,sub_i)}$	$s_{(sub_i,sub_i')}$	$s_{(sub_i,0)}$	$s_{(sub_i',1)}$	$\mathcal{M}_1$	$\mathcal{M}_{sub_i}$	$\mathcal{M}_{sub_i'}$	Env
0	–	–	–	–	s	s	s	s	$z \cup \{l_i\}$	$\emptyset$	$\emptyset$	z'
1	r_{11}	–	–	–	s'	s	s	s	z	$\{l_i\}$	$\emptyset$	z'
2	–	r_{15}	–	–	s'	s	s	s	z	$\emptyset$	$\{l_i\}$	z'
3	–	–	–	r_{19}	s'	s	s	s	$z \cup \{l_i'\}$	$\emptyset$	$\emptyset$	z'
4	r_{14}	–	–	–	s	s	s	s	z	$\{l_i'''\}$	$\emptyset$	z'
5	–	r_{17}	–	–	s	s	s	s	z	$\emptyset$	$\{l_i'''\}$	z'
6	–	–	–	r_{21}	s	s	s	s	$z \cup \{l_k\}$	$\emptyset$	$\emptyset$	z'

Therefore the system simulates a SUB instruction accurately for both cases.

At some moment, if cell 1 gets object l_h, there is no available rule in Π, and the system halts. The computing result of the system is the number of a_1 located in cell 1. Consequently $N(M) = N(\Pi)$.

Appendix D The Proof of Theorem 3

Proof. We construct a family of recognizer MESTCD P systems $\mathbf{\Pi} = \{\Pi(t) \mid t \in \mathbb{N}\}$, which is employed to solve the Boolean satisfiability problem.

Let $\varphi = C_1 \wedge \cdots \wedge C_m$ be a Boolean formula, where $C_i = y_{i,1} \vee \cdots \vee y_{i,p_i}$, with $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leqslant k \leqslant n\}$, $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant p_i$, we define $s(\varphi) = \langle n, m \rangle$ and $cod(\varphi) = \alpha_{1,1}^2 \ldots \alpha_{n,1}^2 \alpha_{1,2}^2 \ldots \alpha_{n,2}^2 \cdots\cdots \alpha_{1,m}^2 \cdots \alpha_{n,m}^2$, where

$$\alpha_{i,j} = \begin{cases} g_{i,j} & \text{if clause } C_j \text{ has variable } x_i; \\ g_{i,j}' & \text{if clause } C_j \text{ has variable } \neg x_i; \\ g_{i,j}'' & \text{if clause } C_j \text{ has neither } x_j \text{ nor } \neg x_j. \end{cases}$$

for $1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m$.

The system $\Pi(s(\varphi)+cod(\varphi))$ will handle for all propositional formulas φ with m clauses and n variables.

Specifically, for each pair (n,m) of natural numbers the recognizer MESTCD P systems $\Pi(\langle n,m\rangle)$ is constructed as follows: $\Pi(\langle n,m\rangle)=(\Gamma,K,\Sigma,\mathcal{E},\mathcal{M}_1,\mathcal{M}_2,\mathcal{M}_3,\mathcal{M}_4,\{(0,2),(0,4),(1,0),(2,3),(3,0),(3,1),(4,3)\},s,s_1,s,s,s,s,s,\mathcal{R},i_{in},i_{out})$, where

- $\Gamma=\Sigma\cup\mathcal{E}\cup\{a_i,b_i,t_i,f_i\mid 1\leqslant i\leqslant n\}\cup\{c_{i,j}\mid 1\leqslant i\leqslant n,1\leqslant j\leqslant m\}\cup\{b_{n+1},d,d',\mathtt{yes},\mathtt{no}\}$,
- $K=\{s_{t,i,j},s_{f,i,j},s'_{t,i,j},s'_{f,i,j},s''_{t,i,j},s''_{f,i,j}\mid 1\leqslant i\leqslant n,1\leqslant j\leqslant m\}\cup\{s_{t,i,m+1},s_{f,i,m+1}\mid 1\leqslant i\leqslant n\}\cup\{s_{m+j}\mid 1\leqslant j\leqslant m+1\}\cup\{s_i\mid 1\leqslant i\leqslant 2nm+2n+m+8\}\cup\{s,s',s''\}$
- $\Sigma=\{g_{i,j},g'_{i,j},g''_{i,j}\mid 1\leqslant i\leqslant n,1\leqslant j\leqslant m\}$,
- $\mathcal{E}=\{\alpha_i\mid 1\leqslant i\leqslant 2nm+2n+m+7\}$,
- $\mathcal{M}_1=\{a_i\mid 1\leqslant i\leqslant n\}\cup\{c_{i,j}\mid 1\leqslant i\leqslant n,1\leqslant j\leqslant m\}\cup\{d,d'\}$, $\mathcal{M}_2=\{b_1\}$, $\mathcal{M}_3=\{\mathtt{yes},\mathtt{no}\}$, $\mathcal{M}_4=\emptyset$,
- $i_{in}=1$ and $i_{out}=0$.

The rules in $\mathcal{R}$ are designed as follows for each stage.

Generation stage.

- Division rules:

$$r_{1,i}:[\ a_i\ ]_1\to[\ t_i\ ]_1[\ f_i\ ]_1,1\leqslant i\leqslant n,$$
$$r_{2,i}:[\ b_i\ ]_2\to[\ b_{i+1}\ ]_2[\ b_{i+1}\ ]_2,1\leqslant i\leqslant n.$$

- Rules in $\mathcal{R}_{(2,3)}$:

$$r_3:(s,b_{n+1}/\lambda,s).$$

- Rule in $\mathcal{R}_{(3,1)}$:

$$r_4:(s,b_{n+1}/\lambda,s).$$

- Rules in $\mathcal{R}_{(1,0)}$:

$$\begin{aligned}
r_5&:(s,b_{n+1}\to\lambda/\lambda,s_1),\\
r_6&:(s_1,t_1\to\lambda/\lambda,s_{t,1,1}),\\
r_7&:(s_1,f_1\to\lambda/\lambda,s_{f,1,1}),\\
r_{8,i,j}&:(s_{t,i,j},g_{i,j}\to\lambda/\lambda,s''_{t,i,j}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{9,i,j}&:(s''_{t,i,j},g_{i,j}\to\lambda/\lambda,s_{t,i,j+1}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{10,i,j}&:(s_{t,i,j},g'_{i,j}\to\lambda/\lambda,s'_{t,i,j}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{11,i,j}&:(s'_{t,i,j},c_{i,j}\to\lambda/\lambda,s_{t,i,j+1}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{12,i,j}&:(s_{t,i,j},g''_{i,j}\to\lambda/\lambda,s'_{t,i,j}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{13,i,j}&:(s'_{t,i,j},c_{i,j}\to\lambda/\lambda,s_{t,i,j+1}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{14,i,j}&:(s_{f,i,j},g_{i,j}\to\lambda/\lambda,s'_{f,i,j}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{15,i,j}&:(s'_{f,i,j},c_{i,j}\to\lambda/\lambda,s_{f,i,j+1}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{16,i,j}&:(s_{f,i,j},g'_{i,j}\to\lambda/\lambda,s''_{f,i,j}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{17,i,j}&:(s''_{f,i,j},g'_{i,j}\to\lambda/\lambda,s_{f,i,j+1}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{18,i,j}&:(s_{f,i,j},g''_{i,j}\to\lambda/\lambda,s'_{f,i,j}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{19,i,j}&:(s'_{f,i,j},c_{i,j}\to\lambda/\lambda,s_{f,i,j+1}),1\leqslant i\leqslant n,1\leqslant j\leqslant m,\\
r_{20,i}&:(s_{t,i,m+1},t_{i+1}\to\lambda/\lambda,s_{t,i+1,1}),1\leqslant i\leqslant n-1,\\
r_{21,i}&:(s_{t,i,m+1},f_{i+1}\to\lambda/\lambda,s_{f,i+1,1}),1\leqslant i\leqslant n-1,\\
r_{22,i}&:(s_{f,i,m+1},t_{i+1}\to\lambda/\lambda,s_{t,i+1,1}),1\leqslant i\leqslant n-1,\\
r_{23,i}&:(s_{f,i,m+1},f_{i+1}\to\lambda/\lambda,s_{f,i+1,1}),1\leqslant i\leqslant n-1.
\end{aligned}$$

The system produces 2^n truth assignments for n variables in generation stage, $\Pi(\langle n,m\rangle)$ checks each truth assignment that to decide whether all clauses are true; Generation stage takes $2nm+2n+3$ steps, which contains $n+3$ auxiliary steps and n iterations (each iteration has $2m+1$ steps). Next we analyze auxiliary steps as follows.

For the first n steps, division rules $r_{1,i},r_{2,i}$ are applied, 2^{n+1} cells (2^n cells with label 1 and 2^n cells with label 2) are produced. Note that all truth assignments are generated, and each cell 1 contains a truth assignment; the role of all cells with label 2 are auxiliary.

At step $n+1$ and step $n+2$, by employing rules r_3 and r_4, the object b_{n+1} in each cell 2 is transferred to cell 3 and then to cell 1 (each cell 1 obtains only an object b_{n+1}). At step $n+3$, rule r_5 is applied, object b_{n+1} is consumed during the movement, the state on each channel (1,0) is revised from s to s'.

Now the system enters n iteration, and the computation process is described as follows.

At the first step of i-th iteration, rules r_6,r_7 (resp., rules $r_{20,i},r_{21,i},r_{22,i},r_{23,i}$) are applied, objects t_1,f_1 (resp., objects t_{i+1},f_{i+1}) are consumed during the movement, the state on each channel (1,0) is revised from s_1 (resp., $s_{t,i,m+1},s_{f,i,m+1}$) to $s_{t,1,1},s_{f,1,1}$ (resp., $s_{t,i+1,1},s_{f,i+1,1}$), respectively.

For the next $2m$ steps of i-th iteration, the system checks whether the clauses are satisfied for truth assignment of variable x_i or $\bar{x}_i$. For each variable $g_{i,j}$ (resp., $g'_{i,j}$ or $g''_{i,j}$), it costs two steps to check whether the clauses are satisfied. More specifically, at the first step, if the state on channel (1,0) is $s_{t,i,j}$ (resp., $s_{f,i,j}$), then rule $r_{8,i,j}$ or $r_{10,i,j}$ or $r_{12,i,j}$ (resp., $r_{14,i,j}$ or $r_{16,i,j}$ or $r_{18,i,j}$) is employed, object $g_{i,j}$ or $g'_{i,j}$ or $g''_{i,j}$ is consumed during the movement, the state on each channel (1,0) is revised to $s''_{t,i,j}$ or $s'_{t,i,j}$. At the second step, if the state on channel (1,0) is $s''_{t,i,j}$ or $s'_{t,i,j}$ (resp., $s'_{f,i,j}$ or $s''_{f,i,j}$), then rule $r_{9,i,j}$ or $r_{11,i,j}$ or $r_{13,i,j}$

(resp., $r_{15,i,j}$ or $r_{17,i,j}$ or $r_{19,i,j}$) is applied, object $g_{i,j}$ (resp., $g'_{i,j}$) or $c_{i,j}$ is consumed during the movement, the state on each channel (1,0) is revised to $s_{t,i,j+1}$ (resp., $s_{f,i,j+1}$) (the membrane structure after the generation stage is shown in Figure D1).

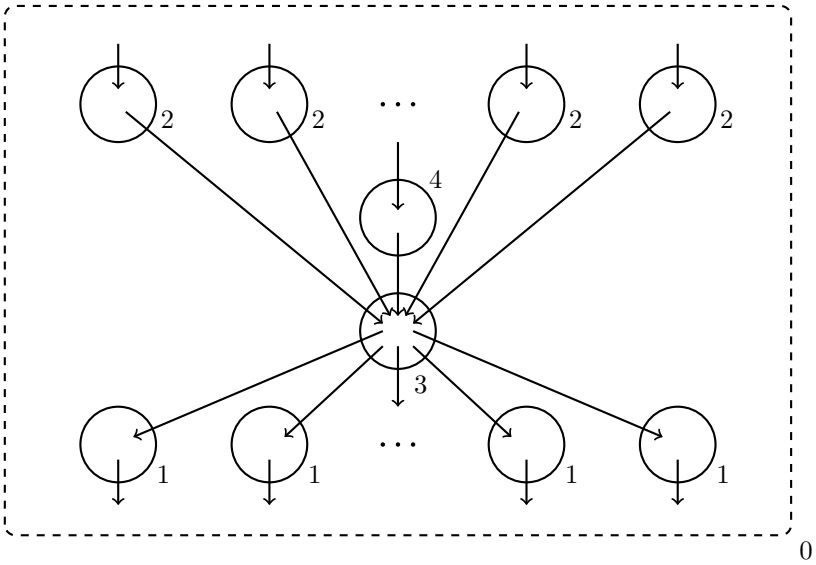

Figure D1 The membrane structure of the constructed MESTCD P system when the generation stage completes.

Checking stage.

- Rules in $\mathcal{R}_{(1,0)}$:

$$r_{24}:(s_{t,n,m+1}, d' \to \lambda/\lambda, s_{m+1}),$$
$$r_{25}:(s_{f,n,m+1}, d' \to \lambda/\lambda, s_{m+1}),$$
$$r_{26,i,j}:(s_{m+j}, c_{i,j} \to \lambda/\lambda, s_{m+j+1}), 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m.$$

In checking stage, each assignment in each cell 1 is checked whether all clauses are satisfied. More specifically, when a cell 1 includes all objects $c_{i,1}, \ldots, c_{i,m}$ $(1 \leqslant i \leqslant n)$, then it indicates all clauses in φ are satisfied for that assignment in that cell, consequently the computation result of φ is TRUE; when there is no one cell 1 that includes all objects $c_{i,1}, \ldots, c_{i,m}$ $(1 \leqslant i \leqslant n)$, then it indicates in each cell 1, Boolean formula φ cannot be satisfied, consequently the computation result of φ is FALSE.

Checking stage takes $m+1$ steps.

At first step of checking stage, rules r_{24} and r_{25} are employed, object d' is consumed during the movement, the state on each channel (1,0) is revised from $s_{t,n,m+1}$ or $s_{f,n,m+1}$ to s_{m+1}.

At the j-th $(1 \leqslant j \leqslant m)$ step of checking stage, rule $r_{26,i,j}$ is employed, object $c_{i,j}$ is consumed during the movement, the state on each channel (1,0) is revised from s_{m+j} to s_{m+j+1}. Note that when object $c_{i,j}$ $(1 \leqslant i \leqslant n)$ does not appear in a cell 1, rule $r_{26,i,j}$ cannot be applied at this step.

Output stage.

- Rule in $\mathcal{R}_{(0,4)}$:

$$r_{27,i}: (s_i, \alpha_i/\lambda, s_{i+1}), 1 \leqslant i \leqslant 2nm + 2n + m + 7.$$

- Rules in $\mathcal{R}_{(1,0)}$:

$$r_{28}: (s_{2m+1}, d/\lambda, s_{2m+1}).$$

- Rules in $\mathcal{R}_{(0,2)}$:

$$r_{29}: (s, d/\lambda, s).$$

- Rules in $\mathcal{R}_{(2,3)}$:

$$r_{30}: (s, d/\lambda, s).$$

- Rule in $\mathcal{R}_{(4,3)}$:

$$r_{31}: (s, \alpha_{2nm+2n+m+7}/\lambda, s).$$

- Rules in $\mathcal{R}_{(3,0)}$:

$$r_{32}: (s, d \to \lambda/\lambda, s'),$$
$$r_{33}: (s', \texttt{yes}/\lambda, s'),$$

r_{34} : $(s, \alpha_{2nm+2n+m+7} \to \lambda/\lambda, s'')$,
r_{35} : $(s'', \mathtt{no}/\lambda, s'')$.

The system sends the correct result to environment in this stage. Rules $r_{27,i}$ are used for counting the computation steps.

If there exist a cell 1 that has state s_{2m+1} (φ is satisfied). On the one hand, from step $2nm+2n+m+5$ to step $2nm+2n+m+7$, rules r_{28}, r_{29}, r_{30} are employed one by one, and finally object d will be delivered to cell 3; on the other hand, object $\alpha_{2nm+2n+m+7}$ is delivered into cell 4. At step $2nm + 2n + m + 8$, rule r_{32} is used, object d is consumed during the movement, the state on each channel (3,0) is revised from s to s'; at the same step, cell 3 receives object $\alpha_{2nm+2n+m+7}$. At step $2nm + 2n + m + 9$, the environment acquires object `yes`, the system stops. Consequently, computation result of φ positive.

If state s_{2m+1} does not appear in any cell 1 (Boolean formula φ is not satisfied), then from step $2nm + 2n + m + 5$ to step $2nm + 2n + m + 7$, rules $r_{27,i}$ are employed, object $\alpha_{2nm+2n+m+7}$ is delivered into cell 4. At step $2nm + 2n + m + 8$, cell 3 receives object $\alpha_{2nm+2n+m+7}$ by applying rule r_{31}. At step $2nm + 2n + m + 9$, rule r_{34} is used, object $\alpha_{2nm+2n+m+7}$ is consumed during the movement, the state on each channel (3,0) is revised from s to s''. At step $2nm + 2n + m + 10$, rule r_{35} is workable, environment acquires object `no`, the system stops. Consequently, computation result of φ is negative.

The resources for designing the MESTCD P systems are counted as follows: (1) total number of symbols is $6nm + 6n + m + 12$; (2) number of cells at initial configuration is 4; (3) total number of objects at initial configuration is $nm + n + 5$; (4) total number of rules used in system is $15nm + 8n + m + 18$; (5) maximum length of a symport rule (state is not counted) is (1,1). Consequently there exists a Turing machine that an MESTCD P system $\Pi(\langle n, m\rangle)$ is designed in polynomial time.

Object `yes` (resp, `no`) as the result of MESTCD P system $\Pi(\langle n, m\rangle)$ is transferred to environment at the last step $2nm+2n+m+9$ (resp., $2nm + 2n + m + 10$). Consequently the MESTCD P system $\Pi(\langle n, m\rangle)$ is polynomially bounded concerning the sizes of clauses and variants for φ.

Consequently, the family of MESTCD P systems $\mathbf{\Pi}$ offers a uniform solution to Boolean satisfiability problem.

References

1 Pérez-Jiménez M J, Sosík P. An optimal frontier of the efficiency of tissue P systems with cell separation. Fundam Inform, 2015, 138(1-2): 45–60
2 Aman B, Ciobanu G. Travelling salesman problem in tissue P systems with costs. J Membr Comput, 2021, 3(2): 97–104
3 Song B S, Li K L, Zeng X X. Monodirectional evolutional symport tissue P systems with promoters and cell division. IEEE T Parall Distr, 2022, 33(2): 332–342
4 Henderson A, Nicolescu R, Dinneen M J. Solving a PSPACE-complete problem with cP systems. J Membr Comput, 2020, 2(4): 311–322