Limits on Efficient Computation in P Systems with Symport/Antiport Rules

Luis F. Macías-Ramos[1], Bosheng Song[2],
Tao Song[2], Linqiang Pan[2], Mario J. Pérez-Jiménez[1]

[1] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: lfmaciasr,marper@us.es

[2] Key Laboratory of Image Information Processing and Intelligent Control,
School of Automation, Huazhong University of Science and Technology,
Wuhan 430074, Hubei, China
E-mail: boshengsong@163.com, songtao0608@hotmail.com,
lqpan@mail.hust.edu.cn

Summary. Classical membrane systems with symport/antiport rules observe the *conservation law*, in the sense that they compute by changing the places of objects with respect to the membranes, and not by changing the objects themselves. In these systems the environment plays an active role because the systems not only send objects to the environment, but also bring objects from the environment. In the initial configuration of a system, there is a special alphabet whose elements appear in an arbitrary large number of copies. The ability of these computing devices with infinite copies of some objects has been widely exploited in the design of efficient solutions to computationally hard problems. This paper deals with computational aspects of P systems with symport/antiport rules and membrane division rules or membrane separation rules. Specifically, we study the limitations of such P systems when the only communication rules allowed have length 1.

Key words: Membrane Computing, P System with Symport/Antiport rules, Membrane Division, Membrane Separation, Computational Complexity.

1 Introduction

In Chapter 3, the computation efficiency of membrane systems has been studied and new techniques and tools have been provided to tackle the **P** versus **NP** problem. For that, two framework has been considered: cell-like P systems with active membranes (with or without using electrical charges) and tissue-like P systems with cell division or cell separation. In both cases, the communication rules are different. In the case of cell-like P systems, evolution rules, send-in and send-out rules and dissolution rules are considered. In the case of tissue-like P systems, communication rules have been implemented by using symport/antiport rules.

Membrane computing is a flexible and versatile branch of natural computing, which arises as an abstraction of the compartmentalized structure of living cells, and the way biochemical substances are processed in (or moved between) membrane bounded regions [10]. Inspired by the structure of living cells, two main classes of membrane systems have been investigated: a hierarchical (cell-like) arrangement of membranes, inspired from the structure of the cell [10] and a net of membranes (placed in the nodes of a directed graph), inspired from the cell-interconnection in tissues [5] or inspired from the the way that neurons communicate with each other by means of short electrical impulses (spikes), emitted at precise moments of time [4]. All classes of computing systems considered in the field of membrane computing are generally called *P systems*, which are parallel and distributed computational models. A comprehensive information in membrane computing can be found in [13] and [2], and for the most up-to-date source of this area, please refer to the P systems website `http://ppage.psystems.eu`.

On the one hand, cell-like P systems with symport/antiport rules were introduced in [9] aiming to abstract the biological phenomenon of trans-membrane transport of couples of chemical substances, in the same or in opposite directions. On the other hand, tissue P systems with symport/antiport rules were introduced in [8] by abstracting networks of elementary membranes such that some of them are linked by "communication channels".

In eukaryotic cells there are two relevant processes: *mitosis* and *membrane fission*. The first one is a process of nuclear division in eukaryotic cells during which one cell gives place to two genetically identical children cells. *Membrane fission* occurs when a membrane gives place to two separated membranes, that is, whenever a vesicle is produced or a larger subcellular compartment is divided into smaller discrete units. These processes have been a source of inspiration to incorporate new ingredients in membrane computing in order to be able to produce exponential workspace in polynomial time. With respect to the mitosis process, *P systems with membrane division* were introduced in [11], and with respect to the membrane fission process, *P systems with membrane separation* were introduced in [6]. These concepts were also introduced in the framework of tissue P systems: tissue P systems with cell division [12] and tissue P systems with cell separation [7].

Taking inspiration from living cells, we add abstractions of the mitosis and the membrane fission processes as ingredients in P systems with symport/antiport rules. Specifically, we allow new types of rules (membrane division and membrane separation) in that framework leading to P systems with symport/antiport rules and membrane division or membrane separation. The limitations of these systems from the efficiency point of view are studied.

The paper is structured as follows. First, some basic concepts and notations are introduced in order to provide a self-contained paper. Section 3 is devoted to define the framework of cell-like P systems with symport/antiport rules and membrane division or membrane separation. Next, recognizer tissue P systems are briefly described and computational complexity classes in these system are introduced. In Section 4, the limitations on the efficiency of cell-like P systems with membrane division or membrane separation which use communication rules of length one, that is, membrane systems without cooperation, are studied. Finally, some conclusions and open problems are presented.

2 Preliminaries

An *alphabet* $\varSigma$ is a finite non-empty set and their elements are called *symbols*. An ordered finite sequence of symbols over $\varSigma$ forms a *string* or *word*. The set of symbols occurring in a string u over $\varSigma$ is denoted by $alph(u)$. The *length* of a string u, denoted by $|u|$, is the number of occurrences of symbols it contains. For an alphabet $\varSigma$, we denote by $\varSigma^*$ the set of all strings of symbols from $\varSigma$. The empty string (with length 0) is denoted by λ. A *language* over $\varSigma$ is a subset of $\varSigma^*$.

A *multiset* over an alphabet $\varSigma$, is an ordered pair $(\varSigma, f)$ where $f : \varSigma \to \mathbb{N}$ is a mapping from $\varSigma$ onto the set of non-negative numbers $\mathbb{N}$. If $m = (\varSigma, f)$ is a multiset then its *support* is defined as $supp(m) = \{x \in \varSigma \mid f(x) > 0\}$. A multiset is finite if its support is a finite set. We denote by $\emptyset$ the empty multiset and we denote by $\mathcal{M}_f(\varSigma)$ the set of all finite multisets over $\varSigma$.

Let $m_1 = (\varSigma, f_1)$, $m_2 = (\varSigma, f_2)$ are multisets over $\varSigma$, then the union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset $(\varSigma, g)$, where $g(x) = f_1(x) + f_2(x)$ for each $x \in \varSigma$. The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is the multiset $(\varSigma, g)$, where $g(x) = f_1(x) - f_2(x)$ if $f_1(x) \geq f_2(x)$, and $g(x) = 0$ otherwise.

Let us recall that a *free tree* (*tree*, for short) is a connected, acyclic, undirected graph. A *rooted tree* is a tree in which one of the vertices (called *the root of the tree*) is distinguished from the others. In a rooted tree the concepts of ascendants and descendants are defined in a usual way. Given a node x (different from the root), if the last edge on the (unique) path from the root of the tree to the node x is $\{x, y\}$ (in this case, $x \neq y$), then y is **the** *parent* of node x and x is **a** *child* of node y. The root is the only node in the tree with no parent (see [1] for details).

Let us recall that the `Reachability Problem` is the following: *given a (directed or undirected) graph G and two nodes a, b, determine whether or not the node b is reachable from a, that is, whether or not there exists a path in the graph from a to b.* We denote by `Reachability`(G, a, b) the answer (`yes` or `no`) to the Reachability problem with instance (G, a, b). It is easy to design an algorithm running in polynomial time solving this problem. For example, given a (directed or undirected) graph G and two nodes a, b, we consider a depth–first–search with source a, and we check if b is in the tree of the computation forest whose root is

a. The total running time of this algorithm is $O(|V| + |E|)$, that is, in the worst case is quadratic in the number of nodes. Moreover, this algorithm needs to store a linear number of items (it can be proved that there exists another polynomial time algorithm which uses $O(\log^2(|V|))$ space).

3 P Systems with Symport/Antiport Rules

In this section we introduce a kind of cell-like P systems that use communication rules capturing the biological phenomenon of trans-membrane transports of several chemical substances. Specifically, two processes have been considered. The first one allows a multiset of chemical substances to pass through a membrane in the same direction. In the second one, two multisets of chemical substances (located in different biological membranes) only pass with the help of each other (an *exchange* of objects between both membranes).

Next, we introduce an abstraction of these operation in the framework of P systems with symport/antiport rules following [9]. In these models, the membranes are not polarized.

Definition 1. *A P system with symport/antiport rules of degree* $q \geq 1$ *is a tuple* $\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$*, where:*

1. Γ *is a finite* alphabet*;*
2. $\mathcal{E} \subsetneq \Gamma$*;*
3. μ *is a membrane structure (a rooted tree) whose nodes are injectively labelled with* $1, 2 \ldots, q$ *(the root of the tree is labelled by* 1*);*
4. $\mathcal{M}_1, \ldots, \mathcal{M}_q$ *are finite multises over* Γ*.*
5. $\mathcal{R}_1, \cdots, \mathcal{R}_q$ *are finite set of communication rules of the following forms:*
   - $\star$ Symport rules*:* (u, out) *or* (u, in)*, where* u *is a finite multiset over* Γ *such that* $|u| > 0$*;*
   - $\star$ Antiport rules*:* $(u, out; v, in)$*, where* u, v *are finite multisets over* Γ *such that* $|u| > 0$ *and* $|v| > 0$*;*
6. $i_{out} \in \{0, 1, \ldots, q\}$*.*

A P system with symport/antiport rules of degree q

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

can be viewed as a set of q membranes, labelled by $1, \ldots, q$, arranged in a hierarchical structure μ given by a rooted tree whose root is called the *skin membrane*, such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the finite multisets of objects initially placed in the q membranes of the system; (b) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; (c) $\mathcal{R}_1, \cdots, \mathcal{R}_q$ are finite sets of communication rules over Γ ($\mathcal{R}_i$ is associated with the membrane i of μ); and (d) i_{out} represents a distinguished *region* which will encode the output of the system. We use the term *region* i ($0 \leq i \leq q$) to refer to membrane i in the case $1 \leq i \leq q$ and to refer to the environment in the case $i = 0$. The length of rule (u, out) or (u, in) (resp. $(u, out; v, in)$) is defined as $|u|$ (resp. $|u| + |v|$).

For each membrane $i \in \{2, \dots, q\}$ (different from the skin membrane) we denote by $p(i)$ the parent of membrane i in the rooted tree μ. We define $p(1) = 0$, that is, by convention the "parent" of the skin membrane is the environment.

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

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

A symport rule $(u, in) \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t if membrane i is in $\mathcal{C}_t$ and multiset u is contained in the parent of i. When applying a rule $(u, in) \in \mathcal{R}_i$, the multiset of objects u goes out from the parent membrane of i and enters into the region defined by the membrane i.

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

The rules of a P system with symport/antiport rules are applied in a non-deterministic maximally parallel manner: at each step we apply a multiset of rules which is maximal, no further applicable rule can be added.

Let us fix a P system with symport/antiport rules Π. We say that configuration $\mathcal{C}^1$ yields configuration $\mathcal{C}^2$ in one *transition step*, denoted by $\mathcal{C}^1 \Rightarrow_{\Pi} \mathcal{C}^2$, if we can pass from $\mathcal{C}^1$ to $\mathcal{C}^2$ by applying the rules from $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_q$ following the previous remarks. A *computation* of Π is a (finite or infinite) sequence of configurations such that: (a) the first term of the sequence is the initial configuration of the system; (b) each non-initial configuration of the sequence is obtained from the previous configuration by applying rules of the system in a maximally parallel manner with the restrictions previously mentioned; and (c) if the sequence is finite (called *halting computation*) then the last term of the sequence is a *halting configuration* (a configuration where no rule of the system is applicable to it).

All computations start from an initial configuration and proceed as stated above; only halting computations give a result, which is encoded by the objects present in the output region i_{out} in the halting configuration. If $\mathcal{C} = \{\mathcal{C}_t\}_{t \leq r}$ of Π ($r \in \mathbb{N}$) is a halting computation, then the *length of* $\mathcal{C}$, denoted by $|\mathcal{C}|$, is r, that is,

$|\mathcal{C}|$ is the number of non-initial configurations which appear in the finite sequence $\mathcal{C}$. We denote by $\mathcal{C}_t(i), 1 \leq i \leq q$, the multiset of objects over Γ contained in the membrane labelled by i at configuration $\mathcal{C}_t$. We also denote by $\mathcal{C}_t(0)$ the multiset of objects over $\Gamma \setminus \mathcal{E}$ contained in the environment at configuration $\mathcal{C}_t$.

3.1 Recognizer P systems with symport/antiport rules

Recognizer P systems were introduced in [16] and they provide a natural framework to solve decision problems. Next, we introduce the concept of recognizer associated with the systems defined in the previous section.

Definition 2. *A recognizer P system with symport/antiport rules of degree* $q \geq 1$ *is a tuple* $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$*, where:*

- $\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ *is a P system with symport/antiport rules of degree* $q \geq 1$*;*
- *the working alphabet* Γ *has two distinguished objects* `yes` *and* `no`*, with at least one copy of them presents in some initial multisets* $\mathcal{M}_1, \dots, \mathcal{M}_q$*, but none of them present in* $\mathcal{E}$*;*
- Σ *is an (input) alphabet strictly contained in* Γ *such that* $\mathcal{E} \subseteq \Gamma \setminus \Sigma$*;*
- $\mathcal{M}_1, \dots, \mathcal{M}_q$ *are finite multisets over* $\Gamma \setminus \Sigma$*;*
- $i_{in} \in \{1, \dots, q\}$ *is the input membrane;*
- *the output region* i_{out} *is the environment;*
- *all computations halt;*
- *if* $\mathcal{C}$ *is a computation of* Π*, then either object* `yes` *or object* `no` *(but not both) must have been released into the environment, and only at the last step of the computation.*

Let us notice that if a recognizer P system

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

has a symport rule of the type (u, out) or (u, in) then $alph(u) \cap (\Gamma \setminus \mathcal{E}) \neq \emptyset$, that is, the multiset u must contains some object from $\Gamma \setminus \mathcal{E}$ because on the contrary, all computations of Π would be non halting.

For each finite multiset w over the input alphabet Σ, a *computation* of $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ with input multiset w starts from the configuration of the form $(\mu, \mathcal{M}_1, \dots, \mathcal{M}_{i_{in}} + w, \dots, \mathcal{M}_q, \emptyset)$, where the input multiset w is added to the content of the input membrane i_{in}. That is, we have an initial configuration associated with each input multiset w over Σ in recognizer P systems with symport/antiport rules. We denote by $\Pi + w$ the P system Π with input multiset w.

3.2 Polynomial complexity classes of recognizer P systems with symport/antiport rules

Let us recall that a decision problem X is one whose solution is either "yes" or "no". This can be formally defined by an ordered pair (I_X, θ_X), where I_X is a language over a finite alphabet and θ_X is a total boolean function over I_X. The elements of I_X are called *instances* of the problem X. Next, according to [15], we define what solving a decision problem by a family of recognizer P systems with symport/antiport rules, *in a uniform way*, means.

Definition 3. *A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer P systems with symport/antiport rules (in a uniform way) if the following conditions 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 $\Pi(s(u))$;*
  - *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 $\Pi(s(u)) + cod(u)$ 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 $\Pi(s(u)) + cod(u)$, 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 $\Pi(s(u)) + cod(u)$ is an accepting one.*

According to Definition 3, we say that for each $u \in I_X$, the P system $\Pi(s(u)) + cod(u)$ is *confluent*, in the sense that all possible computations of the system must give the same answer.

If $\mathbf{R}$ is a class of recognizer P systems, then we denote by $\mathbf{PMC_R}$ the set of all decision problems which can be solved in polynomial time (and in a uniform way) by means of recognizer P systems from $\mathbf{R}$. The class $\mathbf{PMC_R}$ is closed under complement and polynomial–time reductions (see [15] for details).

3.3 P systems with symport/antiport rules and membrane division or membrane separation

In this section, we introduce new types of rules (membrane division and membrane separation) inspired by the mitosis and the membrane fission processes, in the framework of P systems with symport/antiport rules. These rules provide a mechanism to construct an exponential workspace in polynomial time.

Definition 4. *A P system with symport/antiport rules and membrane division of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$, where:*

1. *$\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ is a P system with symport/antiport rules.*
2. *$\mathcal{R}_1, \cdots, \mathcal{R}_q$ are finite set of symport/antiport rules which can also contain rules of the following form: $[a]_i \to [b]_i[c]_i$, where $i \notin \{1, i_{out}\}$ and $a, b, c \in \Gamma$* (division rules).

A division rule $[a]_i \to [b]_i[c]_i \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t if the following holds: (a) membrane i is in $\mathcal{C}_t$; (b) object a is contained in such membrane; and (c) membrane i is neither the skin membrane nor the output membrane (if $i_{out} \in \{1, \dots, q\}$). When applying a division rule $[a]_i \to [b]_i[c]_i$, under the influence of object a, the membrane with label i is divided into two membranes with the same label; in the first copy, object a is replaced by object b, in the second one, object a is replaced by object c; all the other objects residing in membrane i are replicated and copies of them are placed in the two new membranes.

Definition 5. *A P system with symport/antiport rules and membrane separation of degree $q \geq 1$ is a tuple*

$$\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$$

where:

1. *$\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ is a P system with symport/antiport rules.*
2. *$\{\Gamma_0, \Gamma_1\}$ is a partition of Γ, that is, $\Gamma = \Gamma_0 \cup \Gamma_1$, $\Gamma_0, \Gamma_1 \neq \emptyset$, $\Gamma_0 \cap \Gamma_1 = \emptyset$;*
3. *$\mathcal{R}_1, \cdots, \mathcal{R}_q$ are finite set of rules symport/antiport rules which can also contain rules of the following form: $[a]_i \to [\Gamma_0]_i[\Gamma_1]_i$, where $i \notin \{1, i_{out}\}$ and $a \in \Gamma$* (separation rules).

A separation rule $[a]_i \to [\Gamma_0]_i[\Gamma_1]_i \in \mathcal{R}_i$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t if the following holds: (a) membrane i is in $\mathcal{C}_t$; (b) object a is contained in such membrane; and (c) membrane i is neither the skin membrane nor the output membrane (if $i_{out} \in \{1, \dots, q\}$). When applying a separation rule $[a]_i \to [\Gamma_0]_i[\Gamma_1]_i \in \mathcal{R}_i$, in reaction with an object a, the membrane i is separated into two membranes with the same label; at the same time, object a is consumed; the objects from Γ_0 are placed in the first membrane, those from Γ_1 are placed in the second membrane.

With respect to the semantics of these variants, the rules of such P systems are applied in a non-deterministic maximally parallel manner (at each step we apply a multiset of rules which is maximal, no further applicable rule can be added), with the following important remark: when a membrane i is divided (resp. separated), the division rule (resp. separation rule) is the only one from $\mathcal{R}_i$ which is applied for that membrane at that step (however, some rules can be applied in a daughter membrane). The new membranes resulting from division (resp. separation) could participate in the interaction with other membranes or the environment by means of communication rules at the next step – providing that they are not divided (resp. separated) once again. The label of a membrane precisely identify the rules which can be applied to it.

The concept of recognizer is extended to P systems with symport/antiport rules and membrane division or membrane separation, in a natural way. We denote by **CDC**(k) (resp. **CSC**(k)) the class of recognizer P systems with symport/antiport rules and membrane division (resp. membrane separation) such that the communication rules of the system have length at most k.

4 Non Efficiency of P Systems from CDC(1)

In this section, we study the limitations of efficient computations in systems from **CDC**(1). Specifically, we show that $\mathbf{P} = \mathbf{PMC}_{\mathbf{CDC(1)}}$, that is, the polynomial complexity class associated with the class of recognizer P systems **CDC**(1) is equal to the class **P**.

Let $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ be a recognizer P system from **CDC**(1). We denote by $\mathcal{M}_j^*$ the multiset over $\Gamma \times \{j\}$ obtained from $\mathcal{M}_j$ by replacing $a \in \Gamma$ by (a, j), and for each finite multiset w over Σ, we denote w^* the multiset over $\Sigma \times \{i_{in}\}$ obtained from $\mathcal{M}_j$ by replacing $a \in \Sigma$ by (a, i_{in})

The rules from $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_q$ are of the following form: (a, out), (b, in) and $[a]_i \to [b]_i\ [c]_i$. These rules can be considered, in a certain sense, as a *dependency* between the object triggering the rule and the object produced by its application.

- The rules in $\mathcal{R}_i$ of type (a, out) can be described as the pair (a, i) produces the pair $(a, p(i))$.
- The rules in $\mathcal{R}_i$ of type (b, in) can be described as the pair $(b, p(i))$ produces the pair (b, i).
- The rules in $\mathcal{R}_i$ of type $[a]_i \to [b]_i\ [c]_i$ can be described as the pair (a, i) produces the pairs (b, i) and (c, i).

We formalize these ideas in the following definition.

Definition 6. *Let $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ be a recognizer P system from* **CDC**(1). *The dependency graph associated with Π is the directed graph $G_\Pi = (V_\Pi, E_\Pi)$ defined as follows:*

- *The set of vertices is $V_\Pi = \{s\} \cup VL_\Pi \cup VR_\Pi$, where:*
  $VL_\Pi = \{(a,i) \in \Gamma \times \{0, \dots, q\} \mid [(a, out) \in \mathcal{R}_i] \vee [\exists j \in ch(i)((a, in) \in \mathcal{R}_j)] \vee$
  $[\exists b, c \in \Gamma\ ([a]_i \to [b]_i [c]_i \in \mathcal{R}_i])\}$
  $VR_\Pi = \{(a,i) \in \Gamma \times \{0, \dots, q\} \mid [(a, in) \in \mathcal{R}_i] \vee [\exists j \in ch(i)((a, out) \in \mathcal{R}_j)] \vee$
  $[\exists b, c \in \Gamma([b]_i \to [a]_i [c]_i \in \mathcal{R}_i)]\}.$
- *The set of edges is:*
  $E_\Pi = \{(s, (a, j)) \mid 1 \le j \le q \wedge (a, j) \in \mathcal{M}_j^*\} \cup$
  $\{((a, i), (b, j)) \in V_\Pi \times V_\Pi \mid [a = b] \wedge [j = p(i) \wedge (a, out) \in \mathcal{R}_i] \vee$

$$[a = b] \wedge [i = p(j) \wedge (a, in) \in \mathcal{R}_j] \vee$$
$$[i = j] \wedge [\exists c \in \Gamma\ ([a]_i \to [b]_i[c]_i \in \mathcal{R}_i)]\}.$$

In what follows, we show that the dependency graph associated with a P system from **CDC**(1), can be constructed by a single deterministic Turing machine working in polynomial time.

Proposition 1. *Let $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ be a recognizer P systems from* **CDC**(1)*. There exists a Turing machine that constructs the dependency graph, G_Π, associated with Π, in polynomial time (that is, in a time bounded by a polynomial function depending on the total number of rules and the maximum length of the rules).*

Proof. A deterministic algorithm that, given a recognizer P system Π from **CDC**(1), whose set of rules is $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_q$, constructs the corresponding dependency graph, is the following:

```
Input: $(\Pi, \mathcal{R})$
$V_\Pi \leftarrow \{s\}$; $E_\Pi \leftarrow \emptyset$
for $j = 1$ to $q$ do
  for each pair $(a, j) \in \mathcal{M}_j^*$ do
    $E_\Pi \leftarrow E_\Pi \cup \{(s, (a, j))\}$
  end for
end for
for each rule $r \in \mathcal{R}$ of $\Pi$ do
  if $r = (a, in) \in \mathcal{R}_i$ then
    $V_\Pi \leftarrow V_\Pi \cup \{(a, p(i)), (a, i)\}$; $E_\Pi \leftarrow E_\Pi \cup \{((a, p(i)), (a, i))\}$
  end if
  if $r = (a, out) \in \mathcal{R}_i$ then
    $V_\Pi \leftarrow V_\Pi \cup \{(a, i), (a, p(i))\}$; $E_\Pi \leftarrow E_\Pi \cup \{((a, i), (a, p(i)))\}$
  end if
  if $r = [a]_i \to [b]_i[c]_i \in \mathcal{R}_i$ then
    $V_\Pi \leftarrow V_\Pi \cup \{(a, i), (b, i), (c, i)\}$;
    $E_\Pi \leftarrow E_\Pi \cup \{((a, i)), (b, i))\} \cup \{((a, i), (c, i))\}$
  end if
end for
```

The running time of this algorithm is bounded by $O(|\mathcal{R}|) \subset O(q \cdot |\Gamma|^3)$. □

Proposition 2. *Let $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ be a recognizer confluent P system from* **CDC**(1)*. The following assertions are equivalent:*

(1) There exists an accepting computation of Π.
(2) There exists a path (with length greater or equal than 2) from s to (yes, 0) *in the dependency graph associated with Π.*

Proof. (1) $\Rightarrow$ (2). First, we show that for each accepting computation $\mathcal{C}$ of Π there exists a path from s to $(\mathtt{yes}, 0)$ in the dependency graph associated with Π. By induction on the length n of $\mathcal{C}$.

Let $n = 1$ and $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1)$ be an accepting computation of Π with length 1. Then, a rule of the form $(\mathtt{yes}, out) \in \mathcal{R}_1$, with $a \in \Gamma$, has been applied at initial configuration $\mathcal{C}_0$. Then, $\mathtt{yes} \in \mathcal{C}_0(1)$, so $(\mathtt{yes}, 1) \in \mathcal{M}_1^*$. Hence, $(s, (\mathtt{yes}, 1), (\mathtt{yes}, 0))$ is a path from s to $(\mathtt{yes}, 0)$ in the dependency graph associated with Π.

Let us suppose that the result holds for n. Let $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n, \mathcal{C}_{n+1})$ be an accepting computation of Π with length $n + 1$. In this situation, $\mathcal{C}' = (C_1, \ldots, C_n, C_{n+1})$ is an accepting computation of the system $\Pi' = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}'_1, \ldots, \mathcal{M}'_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$, being $\mathcal{M}'_j = \{(a, i) \in \Gamma \times \{0, \ldots, q\} \mid C_1(j) = a\}$ the "content" of membrane j in configuration $\mathcal{C}_1$, for $1 \leq j \leq q$. By induction hypothesis there exists a path $\gamma_{\mathcal{C}'} = (s, (b_1, i_1), \ldots, (\mathtt{yes}, 0))$ from s to $(\mathtt{yes}, 0)$ in the dependency graph associated with Π' (with length greater or equal than 2). We distinguish two cases. If $b_1 \in \mathcal{C}_0(i_1)$ (that means that in the first step of computation $\mathcal{C}$, a division rule has been applied to membrane i_1 such that object b_1 does not appear in the rule), then $\gamma_{\mathcal{C}} = (s, (b_1, i_1), \ldots, (\mathtt{yes}, 0))$ is a path from s to $(\mathtt{yes}, 0)$ in the dependency graph associated with Π, and the result holds. Otherwise, there is an element $b_0 \in \mathcal{C}_0(i_0)$ producing (b_1, i_1) at the first step of computation $\mathcal{C}$. Hence, $\gamma_{\mathcal{C}} = (s, (b_0, i_0), (b_1, i_1), \ldots, (\mathtt{yes}, 0))$ is a path from s to $(\mathtt{yes}, 0)$ in the dependency graph associated with Π.

(2) $\Rightarrow$ (1). Let us see that for each path from s to $(\mathtt{yes}, 0)$ in the dependency graph associated with Π, with length $k \geq 2$, there exists an accepting computation of Π. By induction on the length k of the path.

Let $k = 2$ and $(s, (a_0, i_0), (\mathtt{yes}, 0))$. Then, $i_0 = 1$ is the label of the skin membrane, $(a_0, out) \in \mathcal{R}_1$, $a_0 = \mathtt{yes}$, and the computation $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1)$ where the rule $(a_0, out) \in \mathcal{R}_1$ belongs to the multiset of rules that yields configuration C_1 from C_0, is an accepting computation of Π.

Let us suppose that the result holds for $k \geq 2$. Let

$$(s, (a_0, i_0), (a_1, i_1), \ldots (a_{k-1}, i_{k-1}), (\mathtt{yes}, 0))$$

be a path from s to $(\mathtt{yes}, 0)$ in the dependency graph of length $k + 1$. If $(a_0, i_0) = (a_1, i_1)$, then the result holds by induction hypothesis. Otherwise, let $\mathcal{C}_1$ be a configuration of Π reached from $\mathcal{C}_0$ by the application of a multiset of rules containing a rule that yields (a_1, i_1) from (a_0, i_0). Then $(s, (a_1, i_1), \ldots (a_{k-1}, i_{k-1}), (\mathtt{yes}, 0))$ is a path from s to $(\mathtt{yes}, 0)$ of length k, in the dependency graph of associated with the system

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

where $\mathcal{M}'_j = \{(a, i) \mid \mathcal{C}_1(j)\}$ is the content of membrane j in configuration $\mathcal{C}_1$, for $1 \leq j \leq q$. By induction hypothesis, there exists an accepting computation $\mathcal{C}' = (\mathcal{C}_1, \ldots, \mathcal{C}_t)$ of Π'. Hence, $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_t)$ is an accepting computation of Π. $\square$

Corollary 1. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of recognizer P systems from* $\mathbf{CDC}(1)$ *solving X, according to Definition 3. Let (cod, s) be the polynomial encoding associated with that solution. Then, for each instance w of the problem X the following assertions are equivalent:*

(a) $\theta_X(w) = 1$ *(that is, the answer to the problem is* `yes` *for w).*
(b) There exists a path from s to $(\texttt{yes}, 0)$ *in the dependency graph associated with the system $\Pi(s(w))$ with input multiset $cod(w)$.*

Proof. Let $w \in I_X$. Then $\theta_X(w) = 1$ if and only if there exists an accepting computation of the system $\Pi(s(w)) + cod(w)$. Bearing in mind that $\Pi(s(w)) + cod(w)$ is a confluent system, from Proposition 4 we deduce that $\theta_X(w) = 1$ if and only if there exists a path from s to $(\texttt{yes}, 0)$ in the dependency graph associated with the system $\Pi(s(w)) + cod(w)$. □

Theorem 1. $\mathbf{P} = \mathbf{PMC_{CDC(1)}}$

Proof. We have $\mathbf{P} \subseteq \mathbf{PMC_{CDC(1)}}$ because $\mathbf{PMC_{CDC(1)}}$ is a nonempty class closed under polynomial–time reduction. Next, we show that $\mathbf{PMC_{CDC(1)}} \subseteq \mathbf{P}$. For that, let $X \in \mathbf{PMC_{CDC(1)}}$ and let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer P systems from $\mathbf{CDC}(1)$ solving X, according to Definition 3. Let (cod, s) be the polynomial encoding associated with that solution.

We consider the following deterministic algorithm:

```
Input: An instance $w$ of $X$
  - Construct the system $\Pi(s(w)) + cod(w)$.
  - Construct the dependency graph $G_{\Pi(s(w))+cod(w)}$ associated with
    $\Pi(s(w)) + cod(w)$.
  - Reachability $(G_{\Pi(s(w))+cod(w)}, s, (\texttt{yes}, 0))$
```

Obviously this algorithm is polynomial in the size $|w|$ of the input. □

5 Non efficiency of P systems from CSC(1)

In this section, we show that the polynomial complexity class associated with the class of recognizer P systems with symport/antiport rules and membrane separation is equal to the class $\mathbf{P}$, when we consider only communication rules with length 1.

In order to associate a dependency graph with each P system from $\mathbf{CSC}(1)$, let us notice that the application of a membrane separation rule $[a]_i \to [\Gamma_0]_i\ [\Gamma_1]_i$ consumes object a and the remaining objects in that membrane are separated in two membranes with the same label.

Definition 7. *Let Π be a recognizer P system from* $\mathbf{CSC}(1)$ *whose set of rules is* $\mathcal{R} = \mathcal{R}_1 \cup \cdots \cup \mathcal{R}_q$*. The dependency graph associated with Π is the directed graph* $G_\Pi = (V_\Pi, E_\Pi)$ *defined as follows:*

- *The set of vertices is* $V_\Pi = \{s\} \cup VL_\Pi \cup VR_\Pi$*, where:*
$$VL_\Pi = \{(a,i) \in \Gamma \times \{0,\dots,q\} : [(a,out) \in \mathcal{R}_i] \vee [\exists j \in ch(i)\ ((a,in) \in \mathcal{R}_j)] \vee [[a]_i \to [\Gamma_0]_i[\Gamma_1]_i \in \mathcal{R}_i])\}$$
$$VR_\Pi = \{(a,i) \in \Gamma \times \{0,\dots,q\} : [(a,in) \in \mathcal{R}_i] \vee [\exists j \in ch(i)((a,out) \in \mathcal{R}_j)]\}.$$
- *The set of edges is*
$$E_\Pi = \{(s,(a,j)) \mid 1 \le j \le q \wedge (a,j) \in \mathcal{M}_j^*\} \cup \{((a,i),(a,j)) \in V_\Pi \times V_\Pi : [j = p(i) \wedge (a,out) \in \mathcal{R}_i] \vee [i = p(j) \wedge (a,in) \in \mathcal{R}_j]\}.$$

In a similar way as in the previous section, the following results are obtained.

Proposition 3. *Let Π be a recognizer P system from* $\mathbf{CSC}(1)$*. There exists a Turing machine that constructs the dependency graph G_Π associated with Π, in polynomial time.*

Proposition 4. *Let Π be a recognizer confluent P system from* $\mathbf{CSC}(1)$*. The following assertions are equivalent:*

(1) There exists an accepting computation of Π.
(2) There exists a path (with length greater or equal than 2) from s to $(\texttt{yes}, 0)$ *in the dependency graph associated with Π.*

Theorem 2. $\mathbf{P} = \mathbf{PMC}_{\mathbf{CSC(1)}}$

6 Conclusions and Further Works

In the framework of (cell-like) P systems with symport/antiport rules, two new kind of rules inspired by the processes of *mitosis* and *membrane fission* in eukaryotic cells, have been considered, called P systems with symport/antiport rules and membrane division or membrane separation.

By using the *dependency graph* technique, the computational efficiency of these P systems has been studied in the case of non-cooperative systems, that is, systems with communication rules of length 1.

For future work, we plan to establish the efficiency of these kind of P systems in order to obtain borderline of the efficiency of the problems in terms of syntactical ingredients of P systems with symport/antiport rules.

Acknowledgements

The work of L. Pan was supported by National Natural Science Foundation of China (61033003, 91130034 and 61320106005). The work of M.J. Pérez-Jiménez and L.F. Macías-Ramos was supported by Project TIN2012-37434 of the Ministerio de Ciencia e Innovación of Spain.

References

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest. *An Introduction to Algorithms.* The MIT Press, Cambridge, Massachussets, 1994.
2. P. Frisco. *Computing with Cells: Advances in Membrane Computing*, Oxford University Press, Oxford, 2009.
3. R. Gutiérrez-Escudero, M.J. Pérez-Jiménez, M. Rius-Font. Characterizing tractability by tissue-like P systems. *Lecture Notes in Computer Science* **5957**, 5957 (2010), 289-300.
4. M. Ionescu, Gh. Păun, T. Yokomori. Spiking neural P systems, *Fundamenta Informaticae*, **71**, 2-3 (2006), 279–308.
5. C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodriguez-Paton. Tissue P systems, *Theoretical Computer Science*, **296**, 2 (2003), 295–326.
6. L. Pan, T.-O. Ishdorj. P systems with active membranes and separation rules. *Journal of Universal Computer Science*, **10**, 5 (2004), 630–649.
7. L. Pan, M.J. Pérez-Jiménez. Computational complexity of tissue–like P systems. *Journal of Complexity*, **26**, 3 (2010), 296–315.
8. A. Păun, Gh. Păun, G. Rozenberg. Computing by communication in networks of membranes, *International Journal of Foundations of Computer Science*, **13**, 6 (2002), 779–798
9. A. Păun, Gh. Păun. The power of communication: P systems with symport/antiport, *New Generation Computing*, **20**, 3 (2002), 295–305.
10. Gh. Păun. Computing with membranes, *Journal of Computer and Systems Science*, **61**, 1 (2000), 108–143.
11. Gh. Păun. P systems with active membranes: attacking **NP**–complete problems, *Journal of Automata, Languages and Combinatorics*, **6** (2001), 75–90.
12. Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez. Tissue P system with cell division. *International Journal of Computers, Communications & Control*, **Vol. III**, 3 (2008), 295–303.
13. Gh. Păun, G. Rozenberg, A. Salomaa (eds.). *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, 2010.
14. Gh. Păun. Attacking **NP**-complete problems. In *Unconventional Models of Computation, UMC'2K* (I. Antoniou, C. Calude, M. J. Dinneen, eds.), Springer-Verlag, 2000, 94-115.
15. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, F. Complexity classes in models of cellular computing with membranes. *Natural Computing*, **2**, 3 (2003), 265–285.
16. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, A polynomial complexity class in P systems using membrane division, *Journal of Automata, Languages and Combinatorics*, **11**, 4 (2006) 423–434.