Membrane fission versus cell division: When membrane proliferation is not enough

Luis F. Macías-Ramos, Mario J. Pérez-Jiménez, Agustín Riscos-Núñez, Luis Valencia-Cabrera

Research Group of Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Sevilla, 41012, Spain

Keywords:
Cell division
Membrane fission
Membrane computing
Symport/antiport rules
Tractability frontier

A B S T R A C T

Cell division is a process that produces two or more cells from one cell by *replicating* the original chromosomes so that each daughter cell gets a copy of them. Membrane fission is a process by which a biological membrane is split into two new ones in such a manner that the contents of the initial membrane get *distributed* or *separated* among the new membranes. Inspired by these biological phenomena, new kinds of models were considered in the discipline of *Membrane Computing*, in the context of P systems with active membranes, and tissue P systems that use symport/antiport rules, respectively.

This paper combines the two approaches: cell-like P systems with symport/antiport rules and membrane separation are studied, from a computational complexity perspective. Specifically, the role of the environment in the context of cell-like P systems with membrane separation is established, and additional borderlines between tractability and **NP**-hardness are summarized.

1. Introduction

Cell division is one of the basic processes in the cell life cycle and it allows producing two or more cells from one cell. Basically, there are three processes associated with cell division: *binary fission* (typical of prokaryotic cells), *mitosis* and *meiosis* (these two taking place in eukaryotic cells). In eukaryotic cells, mitosis generally involves forming two identical daughter cells by replicating and dividing the original chromosomes. Meiosis allows genetic variation through a process of DNA shuffling while the cells are dividing.

Several cell division inspired mechanisms were introduced in *Membrane Computing*, a distributed parallel computing paradigm inspired by the way the living cells process chemical substances, energy and information. In this discipline these mechanisms are called *cell division* rules and *membrane division* rules that are triggered by an object which is replaced in the two new cells by possibly new objects and the remaining objects are *duplicated* in both new cells/membranes. These two ways have given rise to cell-like P systems based models (*P systems with active membranes* [21]) and tissue-like P systems based models (*tissue P systems with cell division* [20]), respectively.

Lipid membranes separate the interior of a cell from its environment or surround membrane compartments (mitochondria, endosomes, Golgi complex, etc.) allowing the cells and compartments to have an identity. They serve as concentrations

barriers allowing the incorporation of material from its environment, in the case of a cell, or exchange material between compartments (from a donor membrane to an acceptor membrane), implemented by membrane carriers. The formation of such carriers in cells follows a simple three-step process whose last step is *membrane fission* consisting in the splitting of the membranes in two new ones [15]. The biological phenomenon of membrane fission was incorporated in *Membrane Computing* through a new kind of rules, called *membrane separation rules*, in the framework of polarizationless P systems with active membranes [3]. Originally, each of these separation rules could have their own partition of the working alphabet. Nevertheless, in [16] a new definition of separation rules in the framework of P systems with active membranes was introduced. In the new definition, all separation rules are associated with the same partition of the working alphabet in two subsets, which is given in advance. By applying these kinds of rules, the object triggering them is consumed and the remaining objects are *distributed* both in the created membranes.

Networks of membranes which compute by communication only, using symport/antiport rules were considered in [18]. By means of this kind of rules, a change of the places of objects with respect to the membranes of a system takes place along computations but not a change/evolution of the objects themselves. Such rules are used both for communication with the environment and for direct communication between different membranes. It is worth noting that, in these systems, the environment plays an active role because we cannot only send objects outside the system, but we can also bring in objects from the environment.

With respect to tissue-like approaches, from the seminal definitions of tissue P systems [13,14], one of the most interesting variants of tissue P systems was presented in [20]. In that paper, the definition of tissue P systems with symport/antiport rules is combined with the one of P systems with active membranes, yielding *tissue P systems with cell division*. One of the latest studies on their computational power can be found in [22]. In tissue-like systems, the membrane fission phenomenon has been considered together with symport/antiport rules [17] by means of cell separation rules. These models are called *tissue P systems with cell separation* and its computational efficiency was investigated. Besides, a tractability border in terms of upper bound of the length of communication rules was obtained: passing from 1 to 8 amounts to passing from non-efficiency to efficiency, assuming that $\mathbf{P} \neq \mathbf{NP}$ [17]. Nevertheless, in [24] that frontier was refined in an optimal sense.

Cell-like P systems with symport/antiport rules were introduced in [19]. This kind of P systems was shown to be computationally complete.[1] In this paper we consider membrane separation in the framework of cell-like P systems with symport/antiport rules, and the computational efficiency of these systems is investigated.

The paper is organized as follows. Next section briefly describes some preliminaries in order to make the work self-contained. In Section 3, complexity classes of recognizer P systems with symport/antiport rules are introduced. In Section 4, the main result is presented: only tractability problems can be solved by families of P systems with symport/antiport rules, membrane separation and without environment. Section 5 summarizes different boundaries between tractability and **NP**-hardness in the framework of recognizer P systems with symport/antiport rules. Finally, conclusions and some open problems are drawn.

2. Preliminaries

An *alphabet* Γ is a non-empty set and their elements are called *symbols*. A *string* u over Γ is an ordered finite sequence of symbols, that is, a mapping from a natural number $n \in \mathbb{N}$ onto Γ. Number n is called the *length* of string u and it is denoted by $|u|$, that is, the length of a string is the number of occurrences of symbols it contains. The empty string (with length 0) is denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*. A *language* over Γ is a subset of Γ^*.

A *multiset* over an alphabet Γ is an ordered pair (Γ, f) where f is a mapping from Γ onto the set of natural numbers $\mathbb{N}$. The *support* of a multiset $m = (\Gamma, f)$ is defined as $supp(m) = \{x \in \Gamma : f(x) > 0\}$. A multiset is finite (resp. empty) if its support is a finite (resp. empty) set. We denote by $\emptyset$ the empty multiset.

Let $m_1 = (\Gamma, f_1)$, $m_2 = (\Gamma, f_2)$ be multisets over Γ, then the union of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset (Γ, g), where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$. We say that m_1 is contained in m_2 and we denote it by $m_1 \subseteq m_2$, if $f_1(x) \leq f_2(x)$ for each $x \in \Gamma$. The relative complement of m_2 in m_1, denoted by $m_1 \setminus m_2$, is multiset (Γ, g), where $g(x) = f_1(x) - f_2(x)$ if $f_1(x) \geq f_2(x)$, and $g(x) = 0$ otherwise.

A *symport rule* (respectively, *antiport rule*) over an alphabet Γ is an expression (u, out) or (u, in), where u is a finite multiset over Γ such that $|u| > 0$ (resp. an expression $(u, out; v, in)$, where u, v are finite multisets over Γ such that $|u| > 0$ and $|v| > 0$). The length of rule (u, out) or (u, in) (resp. $(u, out; v, in)$) is defined as $|u|$ (resp. $|u| + |v|$). A *division rule* over Γ is an expression $[\,a\,]_i \rightarrow [\,b\,]_i\,[\,c\,]_i$, where $a, b, c \in \Gamma$. A *separation rule* over Γ is an expression $[a]_i \rightarrow [\Gamma_0]_i[\Gamma_1]_i$ where $a \in \Gamma$ and $\{\Gamma_0, \Gamma_1\}$ is a partition of Γ.

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 as follows. Given a node x (different from the root), if the last edge on the (unique) path from the root 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. A node with no children is called a *leaf* (see [4] for details).

[1] In [2], the authors show that systems with $s \geq 2$ symbols and $m \geq 1$ membranes (such that $m + s \geq 6$) are universal.

3. P systems with symport/antiport rules and membrane division/separation

In this section, we define specific cell-like models of Membrane Computing capturing the biological phenomena of transmembrane transport of couples of chemical substances, cell division, and membrane fission.

Definition 3.1. 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* and $\mathcal{E} \subsetneq \Gamma$;
2. μ is a rooted tree whose nodes are injectively labeled with $1, \ldots, q$;
3. $\mathcal{M}_1, \ldots, \mathcal{M}_q$ are finite multisets over Γ;
4. $\mathcal{R}_i$, $1 \leq i \leq q$, are finite sets of symport/antiport rules over Γ;
5. $i_{out} \in \{0, 1, \ldots, q\}$.

A P system with symport/antiport rules of degree q, can be viewed as a set of q membranes, labeled by $1, \ldots, q$, arranged in a hierarchical (membrane) structure μ given by a rooted tree whose root is called the *skin membrane*, labeled by 1, such that: (a) $\mathcal{M}_i$, $1 \leq i \leq q$, represent the finite multisets of *objects* (symbols of Γ) 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) $\mathcal{R}_i$, $1 \leq i \leq q$, are finite sets of rules over Γ associated with the membranes labeled by i; (d) i_{out} is a label that represents the output *region* of the system. We use the term *region* i to refer to membrane i, in case $1 \leq i \leq q$, and to refer to the environment, in case $i = 0$. For each membrane $i \in \{2, \ldots, q\}$ (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. The leaves of the rooted tree are called *elementary membranes*. If the alphabet of the environment is an empty set then we say that the P system is *without environment*. This term means that there does not exist any object initially located in the environment of the system available in an arbitrary number of copies, that is, in P systems without environment there is an environment (labeled by 0 as usual) but in any moment each object in it has a *finite multiplicity*.

In P systems with symport/antiport rules and *membrane division* (resp. *membrane separation*) also membrane division rules (resp. membrane separation rules associated with the same partition of the working alphabet, which is given in advance) are allowed.

A *configuration* $\mathcal{C}_t$ at an instant t of a P system with symport/antiport rules is described by the following elements: (a) the membrane structure at instant t; (b) all multisets of objects over Γ associated with all the membranes present in the system; and (c) the multiset of objects over $\Gamma - \mathcal{E}$ associated with the environment at that moment. 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 the multiset associated with 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 multiset associated with 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 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.

A membrane division rule $[a]_i \to [b]_i[c]_i \in \mathcal{R}_i$ is *applicable* to a configuration at an instant t, if there is an elementary membrane i in that configuration and object a is contained in that membrane. When applying a division rule $[a]_i \to [b]_i[c]_i$, under the influence of object a, a 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. The skin membrane and the output membrane cannot be divided.

A membrane 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 there exists an elementary membrane labeled by i in $\mathcal{C}_t$, different from the skin membrane, such that it contains object a. When applying a separation rule $[a]_i \to [\Gamma_0]_i\,[\Gamma_1]_i \in \mathcal{R}_i$ to a membrane labeled by i in a configuration $\mathcal{C}_t$, in reaction with object a, that membrane is separated into two membranes with the same label; at the same time, object a is consumed; all the other objects residing in membrane i are *distributed*: the objects from Γ_0 are placed in the first membrane, while those from Γ_1 are placed in the second membrane. The skin membrane and the output membrane cannot be separated.

Note that, due to membrane division or membrane separation rules, several elementary membranes with the same label i ($i \neq 1$) can be present at a given configuration.

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/separated, the membrane division/separation rule is the

only one from $\mathcal{R}_i$ which is applied for that membrane at that step. The new membranes resulting from division/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 separated once again. The label of a membrane identifies the rules which can be applied to it precisely.

Let us consider a recognizer P system with symport/antiport rules Π. We say that configuration $\mathcal{C}_t$ yields configuration $\mathcal{C}_{t+1}$ in one *transition step*, denoted by $\mathcal{C}_t \Rightarrow_\Pi \mathcal{C}_{t+1}$, if we can pass from $\mathcal{C}_t$ to $\mathcal{C}_{t+1}$ by applying the rules from the system following the previous remarks. A *computation* of Π is a (finite or infinite) sequence of configurations such that: (a) the first term is the initial configuration of the system; (b) for each $n \geq 2$, the n-th configuration of the sequence is obtained from the previous one in one transition step; and (c) if the sequence is finite (called *halting computation*) then the last term is a *halting configuration* (a configuration where no rule of the system is applicable to it). All the 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} associated with the halting configuration. If $\mathcal{C} = \{\mathcal{C}_t\}_{t<r+1}$ of Π ($r \in \mathbb{N}$) is a halting computation, then the *length* of $\mathcal{C}$, denoted by $|\mathcal{C}|$, is r.

Definition 3.2. A recognizer P system with symport/antiport rules of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ where:

1. $\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ is a P system with symport/antiport rules of degree $q \geq 1$.
2. The working alphabet Γ has two distinguished objects `yes` and `no`, at least one copy of them present in some initial multisets $\mathcal{M}_1, \ldots, \mathcal{M}_q$, but none of them is present in $\mathcal{E}$.
3. Σ is an (input) alphabet strictly contained in Γ such that $\mathcal{E} \cap \Sigma = \emptyset$.
4. $\mathcal{M}_1, \ldots, \mathcal{M}_q$ are finite multisets over $\Gamma \setminus \Sigma$.
5. $i_{in} \in \{1, \ldots, q\}$ represents the input membrane.
6. The output region i_{out} is the environment.
7. All computations halt.
8. 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 has a symport rule of the type (u, in) associated with the skin membrane, then the multiset u must contain some object from $\Gamma \setminus \mathcal{E}$ because on the contrary, this would lead to non-halting computations of Π.

For each finite multiset w over the input alphabet Σ, a *computation* of $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ with input multiset w starts from the configuration of the form $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} + w, \ldots, \mathcal{M}_q, \emptyset)$, 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.

We say that a computation $\mathcal{C}$ is an *accepting computation* (resp. *rejecting computation*) if object `yes` (resp. object `no`) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$, and neither object `yes` nor `no` appears in the environment associated with any non-halting configuration of $\mathcal{C}$.

For each natural number $k \geq 1$, we denote by $\mathbf{CDC}(k)$ (resp. $\mathbf{CSC}(k)$) the class of recognizer P systems with symport/antiport rules and membrane division (resp. membrane separation) such that the length of the communication rules allowed is at most k. In the case of P systems without environment we denote $\widehat{\mathbf{CDC}}(k)$ or $\widehat{\mathbf{CSC}}(k)$, respectively.

3.1. Complexity classes of recognizer P systems with symport/antiport rules

Let us recall that a decision problem X is one that has a `yes`/`no` answer, that is, is 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, we define what solving a decision problem by a family of recognizer P systems with symport/antiport rules, *in a uniform way*, means (see [23] for details).

Definition 3.3. A decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ of recognizer P systems with symport/antiport rules if the following conditions hold:

- the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines;
- 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, sound and complete with regard to (X, cod, s).

We say that $\mathbf{\Pi}$ provides a *uniform solution* to the problem X, and pair (cod, s) is a *polynomial encoding* from X in $\mathbf{\Pi}$. It is worth pointing out that for each instance u, the 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 by means of recognizer P systems from $\mathbf{R}$, according to Definition 3.3.

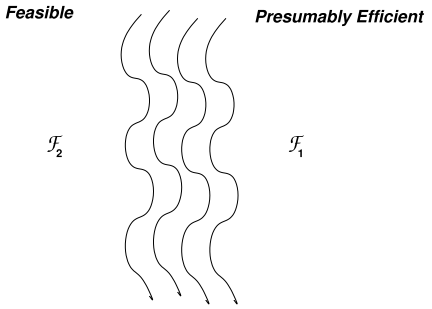
4. Tractability frontiers in P systems with symport/antiport rules

We say that a class of recognizer P systems with symport/antiport rules $\mathcal{F}$ is *presumably efficient* if there exists an **NP**-complete problem that can be solved in polynomial time by a family of systems from $\mathcal{F}$. From the properties of the **NP**-completeness, we deduce that *any* **NP**-complete problem can be solved in polynomial time by families of a presumably efficient class of recognizer membrane systems. Because class $\mathbf{PMC}_{\mathcal{F}}$ is closed under complement and polynomial-time reductions (see [23] for details), if the class $\mathcal{F}$ is presumably efficient then $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathcal{F}}$.

We say that a class of recognizer membrane systems $\mathcal{F}$ is *feasible* if only tractable problems can be solved in polynomial time by a family of systems from $\mathcal{F}$, that is, if $\mathbf{PMC}_{\mathcal{F}} = \mathbf{P}$. According with these definitions, if $\mathbf{P} = \mathbf{NP}$ then a class $\mathcal{F}$ is feasible if and only if it is presumably efficient. Besides, if $\mathbf{P} \neq \mathbf{NP}$ then each feasible class is not presumably efficient. Nevertheless, under that hypothesis a non-feasible class could be non-presumably efficient (as a consequence of the Ladner theorem by which if $\mathbf{P} \neq \mathbf{NP}$ then there exist **NP**-intermediate problems, that is, problems which are neither in the class **P** nor in the class of **NP**-complete problems, see [7] for details).

A good strategy to prove that $\mathbf{P} = \mathbf{NP}$ is the following: let us suppose we have two classes $\mathcal{F}_1$ and $\mathcal{F}_2$ of recognizer P systems such that:

(a) $\mathcal{F}_1$ is feasible and $\mathcal{F}_2$ is presumably efficient.
(b) Each solution S of a decision problem X in $\mathcal{F}_1$ is also a solution in $\mathcal{F}_2$;



The syntactical ingredients which are required to be added to membrane systems in $\mathcal{F}_1$ in order to obtain membrane systems in $\mathcal{F}_2$, provide a frontier between tractability and **NP**-hardness. Therefore, *translating* an efficient solution of an **NP**-complete problem by a family of systems in $\mathcal{F}_2$, into an efficient solution by a family of systems in $\mathcal{F}_1$ amounts to proving $\mathbf{P} = \mathbf{NP}$.

4.1. Feasibility of recognizer P systems from $\widehat{\mathbf{CSC}}$

In this section we will show that $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathbf{CSC}}}$. The proof is inspired on a similar result, obtained in the framework of tissue P systems with symport/antiport rules and cell separation [12].

Let $\Pi = (\Gamma, \Gamma_0, \Gamma_1, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out})$ be a recognizer P system from $\widehat{\mathbf{CSC}}$ of degree $q \geq 1$.

- We denote by $p(i)$ (resp., $ch(i)$) the label of the parent (resp., a child) of the membrane labeled by i, the parent of the skin membrane is the environment (we write $p(1) = 0$). We denote by $\mathcal{R}_C$ (resp., $\mathcal{R}_S$) the set of communication rules (resp., separation rules) of Π. We will fix total orders in $\mathcal{R}_C$ and $\mathcal{R}_S$.
- Let $\mathcal{C}$ be a computation of Π, and $\mathcal{C}_t$ a configuration of $\mathcal{C}$. The application of a communication rule keeps the multiset of objects of the whole system unchanged because only movement of objects between the regions of the system is produced. On the other hand, the application of a separation rule causes that an object is removed from the system, and since there is no objects replication, the rest remains unchanged. Thus, the multiset of objects of the system in any configuration $\mathcal{C}_t$ is contained in $\mathcal{M}_0 + \ldots + \mathcal{M}_q$. Moreover, if $M = |\mathcal{M}_0 + \ldots + \mathcal{M}_q|$ then the total number of membranes having label i, $1 \leq i \leq q$, at configuration $\mathcal{C}$ is, at most, M because the copies can only be produced by the application of a separation rule, and each application of this kind of rule consumes one object. Consequently, $M \cdot q$ is an upper bound of the number of membranes at any configuration of the system.

- In order to identify the membranes created by the application of a separation rule, we modify the labels of the new membranes in the following recursive manner:
  - The label of a membrane will be a pair (i,σ) where $0 \leq i \leq q$ and $\sigma \in \{0,1\}^*$. At the initial configuration, the labels of the membranes are $(1,\lambda),\ldots,(q,\lambda)$. The label of the environment is denoted by $(0,\lambda)$.
  - If a separation rule is applied to a membrane labeled by (i,σ), then the new created membranes will be labeled by $(i,\sigma 0)$ and $(i,\sigma 1)$, respectively. Membrane $(i,\sigma 0)$ will only contain the objects of membrane (i,σ) which belong to Γ_0, and membrane $(i,\sigma 1)$ will only contain the objects of membrane (i,σ) which belong to Γ_1. Only elementary membranes can be separated, so if a membrane i is non-elementary then we denote it by the label (i,λ).
- If a membrane labeled by (i,σ) is engaged by a communication rule, then, after the application of the rule, the membrane keeps its label.
- A configuration $\mathcal{C}_t$ of a P system from $\widehat{\mathbf{CSC}}$ is described by the current membrane structure and the multisets of labeled objects of the type

  $$\{(a,i,\sigma) : a \in \Gamma, 0 \leq i \leq q, \sigma \in \{0,1\}^*\}$$

  The expression $(a,i,\sigma) \in \mathcal{C}_t$ means that object a belongs to membrane labeled by (i,σ). Let us notice that the number of labels we need to identify all membranes appearing along any computation of a P system from $\mathbf{CSC}(2)$ is (in the worst case) quadratic in the size of the initial configuration of the system and the length of the computation.
- Let $r = (a_1,\ldots,a_s, out; b_1,\ldots,b_{s'}, in) \in \mathcal{R}_i$ be an antiport rule of Π. We denote by $n \cdot LHS(r,(i,\sigma))$, $n \in \mathbb{N}$, the multiset of labeled objects $(a_1,i,\sigma)^n \cdots (a_s,i,\sigma)^n (b_1,p(i),\tau)^n \cdots (b_{s'},p(i),\tau)^n$, where $(p(i),\tau)$ is the parent of membrane (i,σ). Similarly, $n \cdot RHS(r,(i,\sigma))$ denotes the multiset $(a_1,p(i),\tau)^n \cdots (a_s,p(i),\tau)^n (b_1,i,\sigma)^n \cdots (b_{s'},i,\sigma)^n$, produced by applying n times rule r over membrane (i,σ).
- Let $r = (a_1,\ldots,a_s, out) \in \mathcal{R}_i$ be a symport rule of Π. We denote by $n \cdot LHS(r,(i,\sigma))$, $n \in \mathbb{N}$, the multiset $(a_1,i,\sigma)^n \cdots (a_s,i,\sigma)^n$. Similarly, $n \cdot RHS(r,(i,\sigma))$ denotes the multiset $(a_1,p(i),\tau)^n \cdots (a_s,p(i),\tau)^n$, produced by applying n times rule r over membrane (i,σ), where $(p(i),\tau)$ is the parent of membrane (i,σ).
- Let $r = (a_1,\ldots,a_s, in) \in \mathcal{R}_i$ be a symport rule of Π. We denote by $n \cdot LHS(r,(i,\sigma))$, $n \in \mathbb{N}$, the multiset $(a_1,p(i),\tau)^n \cdots (a_s,p(i),\tau)^n$, where $(p(i),\tau)$ is the parent of membrane (i,σ). Similarly, $n \cdot RHS(r,(i,\sigma))$ denotes the multiset $(a_1,i,\sigma)^n \cdots (a_s,i,\sigma)^n$, produced by applying n times rule r over membrane (i,σ).
- Let $\mathcal{C}_t$ is a configuration of Π, we denote by $\mathcal{C}_t + \{(x,i,\sigma)/\sigma'\}$ the multiset obtained by replacing in $\mathcal{C}_t$ every occurrence of (x,i,σ) by (x,i,σ'). Besides, $\mathcal{C}_t + m$ (resp., $\mathcal{C}_t \setminus m$) is used to denote that a multiset m of labeled objects is added (resp., removed) to the configuration.

Next, we provide a deterministic algorithm $\mathcal{A}$ working in polynomial time that receives as input a recognizer P system Π from $\widehat{\mathbf{CSC}}$ together with an input multiset m of Π. Then algorithm $\mathcal{A}$ reproduces the behavior of a single computation of such system.

The pseudocode of the algorithm $\mathcal{A}$ is described as follows:

```
Input: A recognizer P system Π from CSC^ and an input multiset m of Π
     Initialization stage: the initial configuration C_0 of Π + m
     t ← 0
     while C_t is a non-halting configuration do
        Selection stage: Input C_t, Output (C'_t, A)
        Execution stage: Input (C'_t, A), Output C_{t+1}
        t ← t + 1
     end while
Output: Yes if C_t is an accepting configuration, No otherwise
```

The selection stage and the execution stage implement a transition step of a recognizer P system Π. Specifically, the selection stage receives as input a configuration $\mathcal{C}_t$ of Π at an instant t. The output of this stage is a pair $(\mathcal{C}'_t, A)$, where A encodes a multiset of rules selected to be applied to $\mathcal{C}_t$, and $\mathcal{C}'_t$ is the configuration obtained from $\mathcal{C}_t$ once the labeled objects corresponding to the application of rules from A have been consumed. The execution stage receives as input the output $(\mathcal{C}'_t, A)$ of the selection stage, and the output is the next configuration $\mathcal{C}_{t+1}$ of $\mathcal{C}_t$. Specifically, at this stage, the configuration $\mathcal{C}_{t+1}$ is obtained from $\mathcal{C}'_t$ by adding the labeled objects produced by the application of rules from A.

Next, selection stage and execution stage are described in detail.

Selection stage.

```
Input: A configuration C_t of Π at instant t
  C'_t ← C_t;  A ← ∅;   B ← ∅
  for each membrane (i,σ) of C'_t according to the lexicographical order do
     for each r ∈ R_C ∩ R_i according to the order chosen do
        n_r ← maximum number of times that r is applicable to (i,σ)
        if n_r > 0 then
           C'_t ← C'_t \ n_r · LHS(r,(i,σ))
           A ← A ∪ {(r, n_r, (i,σ))}
```

```
            B ← B ∪ {(i, σ)}
        end if
    end for
    for r ≡ [a]_i → [Γ_0]_i[Γ_1]_i ∈ R_i according to the order chosen do
        if (i, σ) ∉ B then
            for each (a, i, σ) ∈ C'_t according to the lexicographical order do
                C'_t ← C'_t \ {(a, i, σ)}
                A ← A ∪ {(r, 1, (i, σ))}
                B ← B ∪ {(i, σ)}
            end for
        end if
    end for
end for
```

This algorithm is deterministic and works in polynomial time. Indeed, the cost in time is polynomial in the size of Π because the number of cycles of the external main **for** loop is of order $O(M \cdot q)$, and the number of cycles of the two internal main **for** loops are of order $O(|R|)$ and $O(|R| \cdot |\Gamma|)$, respectively. Besides, the last loop includes a membership test of order $O(M \cdot q)$.

In order to complete the simulation of a computation step of the system Π, the execution stage takes care of the effects of applying the rules selected in the previous stage: updating the objects according to the RHS of the rules.

Execution stage.

```
Input: The output C'_t and A of the selection stage
    for each (r, n_r, (i, σ)) ∈ A do
        if r ∈ R_C ∩ R_i then
            C'_t ← C'_t + n_r · RHS(r, (i, σ))
        else if r ∈ R_S ∩ R_i then
            C'_t ← C'_t + {(λ, i, σ)/σ0}
            C'_t ← C'_t + {(λ, i, σ1)}
            for each (x, i, σ) ∈ C'_t according to the lexicographical order do
                if x ∈ Γ_0 then
                    C'_t ← C'_t + {(x, i, σ)/σ0}
                else
                    C'_t ← C'_t + {(x, i, σ)/σ1}
                end if
            end for
        end if
    end for
    C_{t+1} ← C'_t
```

This algorithm is deterministic and works in polynomial time. Indeed, the cost in time is polynomial in the size of Π because the number of cycles of the main **for** loop is of order $O(|R| \cdot |\Gamma| \cdot M \cdot q)$. Besides, inside the body of the last loop there is a membership test of order $O(|\Gamma|)$.

Theorem 4.1. $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathbf{CSC}}}$.

Proof. It suffices to prove that $\mathbf{PMC}_{\widehat{\mathbf{CSC}}} \subseteq \mathbf{P}$. Let $k \in \mathbb{N}$ such that $X \in \mathbf{PMC}_{\widehat{\mathbf{CSC}}(k)}$ and let $\{\Pi(n) : n \in \mathbb{N}\}$ be a family of P systems from $\widehat{\mathbf{CSC}}(k)$ solving X according to Definition 3.3. Let (cod, s) be a polynomial encoding associated with that solution. Let us recall that instance $u \in I_X$ of the problem X is processed by the system $\Pi(s(u)) + cod(u)$.

Let us consider the following algorithm $\mathcal{A}'$:

```
Input: an instance u of the decision problem X
    Construct the system Π(s(u)) + cod(u)
    Run algorithm A with input Π(s(u)) + cod(u)
Output: Yes if Π(s(u)) + cod(u) has an accepting computation, No otherwise
```

Given an instance u of the decision problem $X = (I_X, \theta_X)$, the following assertions are equivalent:

1. $\theta_X(u) = 1$, that is, the answer of problem X to instance u is affirmative.
2. Every computation of $\Pi(s(u)) + cod(u)$ is an accepting computation.
3. The output of the algorithm with input u is *Yes*.

Therefore, algorithm $\mathcal{A}'$ provide a solution of the decision problem X. Bearing in mind that $\mathcal{A}'$ works in polynomial time, we finally deduce that $X \in \mathbf{P}$. □

5. On efficiency of P systems with symport/antiport rules

Let us recall three important techniques that have been used to study the efficiency of classes of membrane systems in the literature: dependency graph technique, simulation technique, and algorithmic technique.

The *dependency graph technique* consists on the construction of a directed graph (*dependency graph*) G_Π associated with a P system Π verifying the following: there exists an accepting computation of Π if and only if there exists a path between two distinguished nodes in the dependency graph associated with it (see [5] and [6] for more details). This property is verified by recognizer P systems with symport/antiport rules where all its communication rules are of length 1. In this case, each rule of Π can be activated by a single object and then we can interpret that there exists a *dependency* between the object triggering the rule and the objects produced by its application. By using this technique, it has been shown (see [9] for details) that only tractable problems can be efficiently solved by using families of recognizer P systems with membrane division or membrane separation which use communication rules of length 1, that is, $\mathbf{P} = \mathbf{PMC}_{\mathbf{CDC}(1)} = \mathbf{PMC}_{\mathbf{CSC}(1)}$.

By using the *algorithmic technique*, it has been shown that the class of recognizer P systems with membrane separation which use communication rules with length at most 2 is *feasible* from the computational complexity point of view [10]. Specifically, a deterministic algorithm $\mathcal{A}$ working in polynomial time receives a P system Π from $\mathbf{CSC}(\mathbf{2})$ and an input multiset m of Π, as input. Then, algorithm $\mathcal{A}$ reproduces the behavior of a computation of $\Pi + m$. In particular, if the given P system is confluent then the algorithm will provide the same answer than the system, that is, the answer of algorithm $\mathcal{A}$ is affirmative if and only if the system $\Pi + m$ has an accepting computation (and then, any computation is an accepting one). Nevertheless, in the context of recognizer P systems which use communication rules with length at most 2, the use of membrane division allows us to solve **NP**-complete problems in polynomial time (in [11] an efficient solution of HAM-CYCLE by means of a family from $\mathbf{CDC(2)}$ is provided). Therefore, it is observed that the behavior of the classes of systems highly depends on the type of rules used for create new membranes: division rules where original objects are *replicated* in the two new membranes, or separation rules where original objects are *distributed* in the membranes created.

In the definition of P systems with symport/antiport rules, the objects in the alphabet of the environment appear at the initial configuration of the system in an arbitrarily large number of copies. This property seems an unfair tool when designing efficient solutions to computationally hard problems in the framework of membrane computing, by performing a space-time trade-off. In the previous section we have shown that the environment is *relevant* in the framework of P systems with symport/antiport rules and membrane separation in the following sense: with environment we can solve efficiently **NP**-complete problems but without environment only tractable problems can be efficiently solved. Nevertheless, if we use division instead of separation, then the environment is *irrelevant* from a complexity point of view.[2] More precisely, by using the simulation technique, it has been shown that for all $k \geq 1$ we have $\mathbf{PMC}_{\mathbf{CDC}(k)} = \mathbf{PMC}_{\widehat{\mathbf{CDC}}(k)}$ (see [8] for details).

The *simulation technique* can be informally described as follows. Given two recognizer P systems, Π and Π', we say that Π' *simulates* Π *in an efficient way* if the following holds: (a) Π' can be constructed from Π by a deterministic Turing machine working in polynomial time; and (b) there exists an injective function, f, from the set $\mathbf{Comp}(\mathbf{\Pi})$ of computations of Π onto the set $\mathbf{Comp}(\mathbf{\Pi}')$ of computations of Π' such that:

- ⋆ There exists a deterministic Turing machine that constructs computation $f(\mathcal{C})$ from computation $\mathcal{C}$ in polynomial time.
- ⋆ A computation $\mathcal{C} \in \mathbf{Comp}(\mathbf{\Pi})$ is an accepting computation if and only if $f(\mathcal{C}) \in \mathbf{Comp}(\mathbf{\Pi}')$ is an accepting one.
- ⋆ There exists a polynomial function $p(n)$ such that for each $\mathcal{C} \in \mathbf{Comp}(\mathbf{\Pi})$ we have $|f(\mathcal{C})| \leq p(|\mathcal{C}|)$.

5.1. Tractability frontiers

Next, based on Theorem 4.1 and results from the previous section, we present different borderlines between tractability and **NP**-hardness in terms of syntactical ingredients of recognizer P systems with communication rules. (See Table 1.)

1. Classes $\mathbf{CDC}(1)$ and $\mathbf{CSC}(2)$ are feasible, while classes $\mathbf{CDC}(2)$ and $\mathbf{CSC}(3)$ are presumably efficient. So, in the framework of recognizer P systems with membrane division (resp. membrane separation), passing from 1 to 2 (resp. from 2 to 3) in the bound of the length of communication rules amounts to passing from tractability to **NP**-hardness.
2. Class $\mathbf{CSC}(2)$ is feasible, while class $\mathbf{CDC}(2)$ is presumably efficient. Hence, in the framework of recognizer P systems with the length of communication rules bounded by 2, allowing division rules instead of separation rules is a tractability border.
3. Class $\widehat{\mathbf{CSC}}(3)$ is feasible, while class $\mathbf{CSC}(3)$ is presumably efficient. Hence, in the framework of recognizer P systems with cell separation and communication rules with length at most 3, the use of objects with infinite multiplicity provides a tractability frontier.
4. For each $k \geq 2$, class $\widehat{\mathbf{CSC}}(k)$ is feasible and class $\widehat{\mathbf{CDC}}(k)$ is presumably efficient. Hence, in the framework of recognizer P systems without environment, using division rules instead of separation rules amounts to passing from tractability to **NP**-hardness.

[2] In [1], a solution to SAT was provided, by using symport/antiport P systems with membrane division and with empty environment.

Table 1
Frontiers of the efficiency.

Feasible	Presumably efficient	
$\mathbf{CDC}(1)$	$\mathbf{CDC}(2)$	(*length of rules*)
$\mathbf{CSC}(2)$	$\mathbf{CSC}(3)$	(*length of rules*)
$\mathbf{CSC}(2)$	$\mathbf{CDC}(2)$	(*kind of rules*)
$\widehat{\mathbf{CSC}}(3)$	$\mathbf{CSC}(3)$	(*environment*)
$\widehat{\mathbf{CSC}}(k), k \geq 2$	$\widehat{\mathbf{CDC}}(k), k \geq 2$	(*kind of rules*)

6. Conclusions

Abstractions of the biological phenomena of cell division and membrane fission have been studied in cell-like models of Membrane Computing, an unconventional bio-inspired computing framework. The different behaviors associated with the replication of objects (cell division) with respect to the distribution/separation of objects (membrane fission) from a computational complexity point of view, have been highlighted. Specifically, in the framework of P systems with symport/antiport rules the role of the environment has been shown to be relevant. Nevertheless, the environment is not relevant when we use membrane division instead of membrane separation.

Different boundaries between tractability and **NP**–hardness have been summarized. The borderlines have been expressed in terms of syntactical ingredients of the computing models: the length of communication rules, the kind of rules, and the use or not of the environment. Finally, it is worth pointing out that each such boundary provides a tool to attack the **P** versus **NP** problem.

Acknowledgements

This work has been supported by Project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain, cofinanced by FEDER funds.

References

[1] A. Alhazov, Solving SAT by symport/antiport P systems with membrane division, in: M.Á. Gutiérrez-Naranjo, et al. (Eds.), Cellular Computing (Complexity Aspects), ESF PESC Exploratory Workshop, Fénix Editora, Sevilla, 2005, pp. 1–6.
[2] A. Alhazov, R. Freund, M. Oswald, Symbol/membrane complexity of P systems with symport/antiport rules, Lecture Notes in Computer Science 3850 (2006) 96–113.
[3] A. Alhazov, T.O. Ishdorj, Membrane operations in P systems with active membranes, in: Gh. Păun, et al. (Eds.), Proceedings of the Second Brainstorming Week on Membrane Computing, RGNC, Technical report 01/2004, Fénix Editora, pp. 37–44.
[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, An Introduction to Algorithms, The MIT Press, Cambridge, Massachusetts, 1994.
[5] R. Gutiérrez-Escudero, M.J. Pérez-Jiménez, M. Rius-Font, Characterizing tractability by tissue-like P systems, Lecture Notes in Computer Science 5957 (2010) 289–300.
[6] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero, On the power of dissolution in P systems with active membranes, Lecture Notes in Computer Science 3850 (2006) 224–240.
[7] R.E. Ladner, On the structure of polynomial time reducibility, J. ACM 22 (1) (1975) 155–171.
[8] L.F. Macías-Ramos, B. Song, T. Song, L. Pan, M.J. Pérez-Jiménez, On efficiency of P systems with symport/antiport and membrane division, 2015, submitted for publication.
[9] L.F. Macías-Ramos, L. Valencia-Cabrera, B. Song, T. Song, L. Pan, M.J. Pérez-Jiménez, Limits on efficient computation in P systems with symport/antiport rules, 2015, submitted for publication.
[10] L.F. Macías-Ramos, L. Valencia-Cabrera, B. Song, L. Pan, M.J. Pérez-Jiménez, Complexity aspects of membrane fission, 2015, submitted for publication.
[11] L.F. Macías-Ramos, L. Valencia-Cabrera, B. Song, T. Song, L. Pan, M.J. Pérez-Jiménez, On efficiency of minimal cooperation in P Systems with symport/antiport rules, 2015, submitted for publication.
[12] L.F. Macías-Ramos, M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font, The efficiency of tissue P systems with cell separation relies on the environment, Lecture Notes in Computer Science 7762 (2013) 243–256.
[13] C. Martín Vide, J. Pazos, Gh. Păun, A. Rodríguez Patón, A New Class of Symbolic Abstract Neural Nets: Tissue P Systems, Lecture Notes in Computer Science, vol. 2387, 2002, pp. 290–299.
[14] C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodríguez-Patón, Tissue P systems, Theoret. Comput. Sci. 296 (2) (2003) 295–326.
[15] S. Morlot, A. Roux, Mechanics of dynamic-mediated membrane fission, Annu. Rev. Biophys. 42 (2013) 629–649.
[16] L. Pan, T.-O. Ishdorj, P systems with active membranes and separation rules, J. Univers. Comput. Sci. 10 (5) (2004) 630–649.
[17] L. Pan, M.J. Pérez-Jiménez, Computational complexity of tissue-like P systems, J. Complexity 26 (3) (2010) 296–315.
[18] A. Păun, Gh. Păun, G. Rozenberg, Computing by communication in networks of membranes, Internat. J. Found. Comput. Sci. 13 (6) (2002) 779–798.
[19] A. Păun, Gh. Păun, The power of communication: P systems with symport/antiport, New Gener. Comput. 20 (3) (2002) 295–305.
[20] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, Tissue P system with cell division, Int. J. Comput. Commun. Control III (3) (2008) 295–303.
[21] Gh. Păun, Attacking NP-complete problems, in: I. Antoniou, C. Calude, M.J. Dinneen (Eds.), Unconventional Models of Computation, UMC'2K, Springer-Verlag, 2000, pp. 94–115.
[22] M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font, F.J. Romero-Campero, A polynomial alternative to unbounded environment for tissue P systems with cell division, Int. J. Comput. Math. 90 (4) (2013) 760–775.
[23] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Complexity classes in models of cellular computing with membranes, Nat. Comput. 2 (3) (2003) 265–285.
[24] M.J. Pérez-Jiménez, P. Sosík, Improving the efficiency of tissue P systems with cell separation, in: M. García-Quismondo, et al. (Eds.), Proceedings of the Tenth Brainstorming Week on Membrane Computing, vol. II, Report RGNC 01/2012, Fénix Editora, pp. 105–140.