P systems with symport/antiport rules: When do the surroundings matter?

David Orellana-Martín[a], Miguel Á. Martínez-del-Amor[a], Luis Valencia-Cabrera[a], Bosheng Song[b], Linqiang Pan[b,c], Mario J. Pérez-Jiménez[a]

[a]*Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain*
[b]*Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, School of Automation, Huazhong University of Science and Technology, Wuhan, Hubei, 430074, China*
[c]*School of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China*

Abstract

Cell-like P systems where communication between the regions are carried out by rules of type symport/antiport are considered. These systems compute by changing the places of objects with respect to the membranes, and not by changing the objects themselves. The environment plays an *active* role in the sense that it not only can receive objects from the system, but also send objects into it. There is an alphabet associated with the environment whose elements appear in an arbitrary large number of copies at the initial configuration. This property seems too strong from a complexity view, but it has been widely exploited in the design of efficient solutions to computationally hard problems when some mechanisms (inspired by mitosis and membrane fission) allowing to construct an exponential workspace in linear time, are considered. In this paper, complexity aspects of P systems with symport/antiport rules and membrane division are considered when the set associated with the environment is the emptyset. It is shown that the role of

Email addresses: `dorellana@us.es` (David Orellana-Martín), `mdelamor@us.es` (Miguel Á. Martínez-del-Amor), `lvalencia@us.es` (Luis Valencia-Cabrera), `boshengsong@163.com` (Bosheng Song), `lqpan@mail.hust.edu.cn` (Linqiang Pan), `marper@us.es` (Mario J. Pérez-Jiménez)

the environment is irrelevant for such kind of P systems, in contrast with the well known results concerning to its relevance when membrane separation is used instead of membrane division.

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

1. Introduction

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 [15]. Two main classes of systems have been investigated: a hierarchical (cell-like) arrangement of membranes, inspired from the structure of the cell [15] and a network of cells (placed in the nodes of a directed graph), inspired from the cell-interconnection in tissues [9] or inspired from the way that neurons communicate with each other by means of short electrical impulses (spikes), emitted at precise moments of time [4]. In this computing paradigm, parallel and distributed computational models, called *P systems*, are considered. Basic concepts and a comprehensive information can be found in [17] and [19].

Networks of (elementary) membranes (linked by "communication channels") which compute by communication only, using symport/antiport rules, were first considered in [13]. These networks are formally structured by means of directed graphs whose nodes are abstractions of membranes (or cells). They are inspired by the biological phenomenon of trans-membrane transport of couples of chemical substances, in the same or in opposite directions. The cell-like version is structured by means of rooted trees and they were introduced in [14]. It is worth noting that in membrane systems with symport/antiport rules, the environment plays an active role: it not only receives objects from the system, but can also send objects inside it.

These basic systems are non-efficient in the sense that only problems in class **P** can be solved in polynomial time by means of families of those systems. In order to provide efficient solutions to computationally hard problems, some mechanisms able to produce an exponential workspace, expressed in terms of the number of membranes/cells and objects, are needed.

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 daughter cells. Membrane fission occurs when a membrane gives place to two separated membranes where the initial chemical substances are distributed. 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. In this context, *P systems with membrane division* [16] and *P systems with membrane separation* [10] were introduced in the framework of cell-like systems. These concepts were also introduced in the framework of tissue P systems in [18] and [11].

The paper is structured as follows. First, some preliminaries needed for the complete understanding of the work are given. 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

In order to make self-contained this paper, some basic concepts and notations are introduced in this Section.

2.1. Alphabet, multisets

An *alphabet* Γ is a non–empty set whose elements are called *symbols*. A *string* over Γ is an ordered finite sequence of symbols, that is, a mapping from a natural number n onto Γ. The number n is called the *length* of the string and it is denoted by $|u|$. The empty string (with length 0) will be denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^* and subsets of Γ^* are referred to as *languages* over Γ. Given an (ordered) alphabet $\Sigma = \{a_1, \dots, a_r\} \subseteq \Gamma$, the *Parikh mapping* Ψ_Σ is a mapping whose domain is Γ^* defined as follows: $\Psi_\Sigma(u) = (|u|_{a_1}, \dots, |u|_{a_r})$, where $|u|_{a_i}$ denotes the number of ocurrences of symbol a_i in string u.

A *multiset* m over a set A is a pair (A, f) where $f : A \to \mathbb{N}$ is a mapping. The set $supp(m) = \{x \in A \mid f(x) > 0\}$ is called the *support* of multiset $m = (A, f)$. A multiset is finite if its support is a finite set. If $m = (A, f)$ is a finite multiset over A and $supp(m) = \{a_1, \ldots, a_k\}$ then the *cardinal* of m, denoted by $|m|$, is $\sum_{i=1}^{k} f(a_i)$. If $m_1 = (A, f_1)$, $m_2 = (A, f_2)$ are multisets over A then the *union* of m_1 and m_2, denoted by $m_1 + m_2$, is the multiset (A, g) defined as follows: $g(a) = f_1(a) + f_2(a)$, for each $a \in A$.

For any sets A and B the *relative complement* $A \setminus B$ of B in A is defined as follows: $A \setminus B = \{x \in A \mid \ x \notin B\}$. For any set A we denote by $|A|$ the cardinality (number of elements) of A, as usual.

2.2. Graphs and trees

Let us recall some notions concerning to graph theory which are used throughout the paper (see [1] for more details). An *undirected graph* is an ordered pair (V, E) where V is a set whose elements are called *nodes* or vertices and $E \subseteq \{\{x, y\} \mid x \in V, y \in V, x \neq y\}$ whose elements are called *edges*. A *directed graph* is an ordered pair (V, E) where V is a set whose elements are called nodes or vertices and $E \subseteq \{(x, y) \mid x \in V, y \in V\}$ whose elements are called *arcs*. A *path* of length $k \geq 1$ from a node u to a node v in a graph (V, E) is a finite sequence $(x_0, x_1, \ldots, x_k)$ of nodes such that $x_0 = u$, $x_k = v$ and $(x_i, x_{i+1}) \in E$ (in the case of a directed graph) or $\{x_i, x_{i+1}\} \in E$ (in the case of an undirected graph), for $0 \leq i \leq k-1$. If $k \geq 2$ and $x_0 = x_k$ then we say that the path is a *cycle* of the graph. A graph with no cycle is said to be *acyclic*. An undirected graph is *connected* if there exist paths between every pair of nodes.

A *rooted tree* is a connected, acyclic, undirected graph with a distinguished node called the *root* of the tree. Given a node x in a rooted tree with root r, being $x \neq r$, any node y on the unique path from x to r is called an *ancestor* of x. If y is an ancestor of x, then x is a *descendant* of y. If the last edge on the (unique) path from r to x is $\{y, x\}$, then we say that 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*.

2.3. Decision problems

A *decision problem* X 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 X. Each decision problem X has an associated language L_X defined as follows: $L_X = \{u \in I_X \mid \theta_X(u) = 1\}$. Conversely, each language L over an alphabet Γ has an associated decision problem $X_L = (I_{X_L}, \theta_{X_L})$ defined as follows: $I_{X_L} = \Gamma^*$ and $\theta_{X_L}(u) = 1$, for each $u \in L$, and $\theta_{X_L}(u) = 0$, for each $u \notin L$. According with these definitions, for each decision problem X we have $X_{L_X} = X$ and for each language L we have $L_{X_L} = L$.

Let us recall that given a language L over an alphabet Γ, a deterministic Turing machine M *recognizes* the language L if and only if for each string $u \in \Gamma^*$ the following holds: $u \in L$ if and only if $M(u)$ answers *yes*. Then, we say that a deterministic Turing machine M *solves* a decision problem X if and only if M recognizes the language L_X associated with X.

2.4. On efficiency of computing models

Let us recall that each computing model provides a mathematical definition of the informal idea of solving abstract problems by means of a mechanical procedure (*algorithm*). A computing model which is equivalent in power to Turing machines is called *universal*.

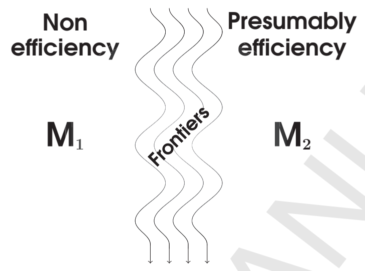
An abstract problem is said to be *tractable* if it can be solved by a deterministic Turing machine working in polynomial time (the upper bound of computational resources is polynomial). The complexity class of decision tractable problems is denoted by **P**. An abstract problem is said to be *intractable* if it cannot be solved by a deterministic Turing machine working in polynomial time (the lower bound of the computational resources is exponential).

A computing model with the ability to provide polynomial-time solutions to intractable problems is called an *efficient* computing model. In a *non efficient* computing model, only problems in class **P** can be solved in polynomial time. It is widely believed that $\mathbf{P} \neq \mathbf{NP}$ and so **NP**-complete problems (the hardest problems in class **NP**) are considered as *presumably intractable* problems. A computing model with the ability to provide polynomial-time solutions to **NP**-complete problems is called a *presumably efficient* computing model.

Given two computing models M_1 and M_2 we say that that M_1 is a *submodel* of M_2, denoted by $M_1 \subseteq M_2$, if each solution of a problem in M_1 is also a solution in M_2, that is, M_2 is an *extension* of M_1 in the sense that M_2 can be obtained from M_1 by adding some syntactic or semantic ingredients. If M_1 is a non-efficient computing model and M_2 is a presumably efficient one such that $M_1 \subseteq M_2$, then we can say that passing from M_1 to M_2 amounts to passing from tractability to presumably intractability. In some sense, it gives us a novel tool to tackle the **P** versus **NP** problem:

– In order to show that $\mathbf{P} = \mathbf{NP}$, it is enough to find a polynomial-time solution to one **NP**-complete problem in M_2 and translate it to a polynomial-time solution in M_1, that is, the ingredients added to obtain M_2 from M_1 do not play a relevant role in that solution.

– In order to show that $\mathbf{P} \neq \mathbf{NP}$, it is enough to find one **NP**-complete problem that cannot be solved efficiently in M_1, that is, that the ingredients added to obtain M_2 from M_1 are crucial to obtain the presumably efficiency.



In what follows, we assume the reader is already familiar with the basic notions and terminology of P systems. For more details, see [17].

3. P systems with symport/antiport rules

Networks of membranes which compute by communication only, using symport/antiport rules, were considered in [13]. These networks aim to abstract the biological phenomenon of trans-membrane transport of couples of chemical substances, in the same or in opposite directions. Such rules are used both for communication with the environment and for direct communication between different membranes.

Definition 3.1. *A basic P system with symport/antiport rules of degree* $q \geq 1$ *is a tuple* $(\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$*, where:*

– Γ *is a finite alphabet and* $\mathcal{E} \subseteq \Gamma$.
– μ *is a rooted tree with* q *nodes which are bijectively labelled with* $1, \dots, q$.
– $\mathcal{M}_i$, $1 \leq i \leq q$, *are multisets over* Γ.
– $\mathcal{R}_i$, $1 \leq i \leq q$, *are finite sets of rules over* Γ *of the following form:*

  ⋆ (u, out) *or* (u, in)*, being* u *a non-empty multiset over* Γ (symport rules).

  ⋆ $(u, out; v, in)$*, being* u, v *non-empty multisets over* Γ (antiport rules).

– $i_{out} \in \{0, 1, \ldots, q\}$.

A basic P system with symport/antiport rules of degree q

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

can be viewed as a set of q membranes, labelled by $1, \ldots, q$, arranged in a hierarchical structure given by a rooted tree μ whose root (called the *skin membrane*) is labelled by 1, such that: (a) for each i, $1 \leq i \leq q$, $\mathcal{M}_i$ represents the multiset of objects initially placed in membrane labelled by i, and $\mathcal{R}_i$ is a finite set of symport/antiport rules associated with membrane i; (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; and (c) i_{out} represents a distinguished *zone* which will encode the output of the system. We use the term *zone* 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 $i = 0$. In the rooted tree μ, the skin membrane is the only membrane in μ with no parent. However, by convention we consider the environment (labelled by 0) as the parent of the skin membrane The length of a symport rule (u, out) or (u, in) is $|u|$. The length of an antiport rule $(u, out; v, in)$ is $|u| + |v|$.

In the framework of P systems with symport/antiport rules, some mechanisms allowing to construct an exponential workspace (expressed in terms of the number of objects and the number of membranes) in polynomial time. Next, two mechanisms (*membrane division* and *membrane separation*) inspired by mitosis and membrane fission, respectively, are introduced. Mitosis is a process in which an eukaryotic *cell division* produces two daughter cells from a single parent cell that are genetically identical to one another and to the original parent cell. Before a cell enters mitosis, it undergoes a period of growth where its genetical components are *replicated* in the new created cells. *Membrane fission* is a process in which “one membrane-enclosed compartment splits into two separate membrane-enclosed compartments”. The contents of the initial membrane is *distributed* between the two new membranes.

Next, an abstraction of these operations is introduced in the framework of P systems with symport/antiport rules.

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

- $(\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ *is a basic P system with symport/antiport rules.*
- *There exists* $i \in \{2, \ldots, q\}$ *such that* $i \neq i_{out}$*, i is the label of a leaf of* μ*, and* $\mathcal{R}_i$ *contains at least one rule of the form* $[a]_i \to [b]_i[c]_i$*, being* $a, b, c \in \Gamma$ (division rule).

Definition 3.3. *A P system with symport/antiport rules and membrane separation of degree* $q \geq 1$ *is a tuple* $(\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$*, where:*

- $(\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ *is a basic P system with symport/antiport rules.*
- $\{\Gamma_0, \Gamma_1\}$ *is a partition of* Γ*, that is,* $\Gamma_0, \Gamma_1 \neq \emptyset$*,* $\Gamma_0 \cap \Gamma_1 = \emptyset$*,* $\Gamma = \Gamma_0 \cup \Gamma_1$*.*
- *There exists* $i \in \{2, \ldots, q\}$ *such that* $i \neq i_{out}$*, i is the label of a leaf of* μ*, and* $\mathcal{R}_i$ *contains at least one rule of the form* $[a]_i \to [\Gamma_0]_i[\Gamma_1]_i$*, being* $a \in \Gamma$ (separation rule)*.*

Definition 3.4. *A P system with symport/antiport rules and membrane division or membrane separation such that the set* $\mathcal{E}$ *associated with the environment is the empty set, is called a P system* without environment*.*

Usually, the set $\mathcal{E}$ associated with the environment will be omited in the tuple describing such P system without environment.

Next, the semantics of P systems with symport/antiport rules and membrane division or membrane separation, is introduced. A *configuration* at an instant t of such a P system is described by all multisets of objects over Γ associated with all the membranes present in the system at instant t, and the multiset of objects over $\Gamma \backslash \mathcal{E}$ associated with the environment at that moment. Recall that there are infinitely many copies of objects from $\mathcal{E}$ in the environment, and hence this set is not properly changed along the computation. The *initial configuration* of a P system with symport/antiport rules of degree q whose initial multisets are $\mathcal{M}_1, \cdots, \mathcal{M}_q$ is the tuple $(\mathcal{M}_1, \cdots, \mathcal{M}_q; \emptyset)$.

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

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

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 a membrane labelled by i is in $\mathcal{C}_t$ and object a is contained in such membrane. When applying a division rule $[a]_i \to [b]_i[c]_i$ to such membrane i, under the influence of object a, the membrane is divided into two membranes with label i; 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. Let us recall that the skin membrane and the output membrane (if $i_{out} \in \{1, \dots, q\}$) cannot be divided.

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 a membrane labelled by i is in $\mathcal{C}_t$ and object a is contained in such membrane. When applying a separation rule $[a]_i \to [\Gamma_0]_i\,[\Gamma_1]_i \in \mathcal{R}_i$ to such membrane i, under the influence of object a, the membrane 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, while those from Γ_1 are placed in the second membrane.

In this way, by means of the application of division or separation rules, several membranes with the same label i can be present in the new membrane structure μ' of the system: for each new membrane labelled by i we have an arc $(p(i), i)$ in μ' as result from the application of a division rule $[a]_i \to [b]_i[c]_i$ or a membrane separation rule $[a]_i \to [\Gamma_0]_i[\Gamma_1]_i$.

The rules of a P system with symport/antiport rules and membrane division or membrane separation, are applied in a non-deterministic maximally parallel manner (at each step a multiset of rules which is maximal, that is, no further applicable rule can be added, is applied), with the following important remark: if a membrane divides/separates, then the division/separation rule is the only one which is applied for that membrane at that step, that is, the objects inside that membrane do not evolve by means of communication rules. In other words, before division/separation a membrane interrupts all its communication channels with the other membranes and with the environment. The new membranes resulting from division/separation can interact with other membranes or with the environment only from the next step.

Let Π be a P system with symport/antiport rules and membrane division or membrane separation. 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 $\mathcal{R}_1 \cup \cdots \cup \mathcal{R}_q$ following the previous remarks. A *computation* of Π is a (finite or infinite) sequence of configurations such that: the first term of the sequence is the initial configuration of the system; for each $n \geq 2$, the n-th term of the sequence is obtained from the $(n-1)$-th term in one transition step; and if the sequence is finite (called *halting computation*) then the last term of the sequence is a *halting configuration*, that is, a configuration where no rule of the system is applicable to it. If $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots \mathcal{C}_r)$, with $r \in \mathbb{N}$, is a halting computation of Π, then r is called the *length of* $\mathcal{C}$, denoted by $|\mathcal{C}|$. For each i, $1 \leq i \leq q$, and t, $0 \leq t \leq r$, $\mathcal{C}_t(i)$ denotes the multiset of objects over Γ contained in all membranes labelled by i at configuration $\mathcal{C}_t$. We denote by $\mathcal{C}_t(0)$ the multiset of objects over $\Gamma \setminus \mathcal{E}$ contained in the environment at configuration $\mathcal{C}_t$. Finally, we denote by $\mathcal{C}_t^*$ the multiset $\mathcal{C}_t(0) + \mathcal{C}_t(1) + \ldots + \mathcal{C}_t(q)$.

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 zone i_{out} associated with the halting configuration.

3.1. Recognizer membrane systems

Throughout this paper, the term *membrane system* is used to refer to a generic P system with symport/antiport rules and membrane division or membrane separation (with or without environment). These systems can be described by means a tuple $(\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ where $\Gamma_0 = \Gamma_1 = \emptyset$ in the case of P systems with membrane division and $\mathcal{E} = \emptyset$ in the case of P systems without environment.

Solving a decision problem is equivalent to recognize the language associated with it. In this Section, recognizer membrane systems are introduced following [21]. In a previous paper [20] these systems were called *accepting P systems.*

Definition 3.5. *A* recognizer membrane system *of degree* $q \geq 1$ *is a tuple* $(\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$*, where:*

- $(\Gamma, \Gamma_0, \Gamma_1, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$ *is a membrane system of degree* q*, as considered above.*

- *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* $\mathcal{E}$*.*
- Σ *is an (input) alphabet strictly contained in* Γ *such that* $\mathcal{E} \cap \Sigma = \emptyset$*.*
- $\mathcal{M}_i$*,* $1 \leq i \leq q$*, are multises over* $\Gamma \setminus \Sigma$*.*
- $i_{in} \in \{1, \ldots, q\}$ *is the input membrane.*
- $i_{out} = 0$*, that is, the output zone* 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 output zone (the environment), and only at the last step of the computation.*

Given a multiset m over the input alphabet Σ, the *computation* of the recognizer membrane system Π *with input multiset* m starts from the configuration of the form $(\mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} + m, \ldots, \mathcal{M}_q; \emptyset)$, that is, the input multiset m has been added to the initial contents of the input membrane i_{in}. Thus, in this kind of P systems there exists an initial configuration associated with each multiset over the input alphabet Σ. We denote it by $\Pi + m$.

Given a recognizer membrane system Π and a halting computation $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_r)$ of, we define the result of $\mathcal{C}$ as follows:

$$Output(\mathcal{C}) = \begin{cases} \texttt{yes}, & \text{if } \Psi_{\{\texttt{yes},\texttt{no}\}}(M_{r,i_{out}}) = (1,0) \wedge \\ & \quad \Psi_{\{\texttt{yes},\texttt{no}\}}(M_{t,i_{out}}) = (0,0) \text{ for } t = 0, \ldots, r-1 \\ \texttt{no}, & \text{if } \Psi_{\{\texttt{yes},\texttt{no}\}}(M_{r,i_{out}}) = (0,1) \wedge \\ & \quad \Psi_{\{\texttt{yes},\texttt{no}\}}(M_{t,i_{out}}) = (0,0) \text{ for } t = 0, \ldots, r-1 \end{cases}$$

where Ψ is the Parikh mapping, and $M_{t,i_{out}}$ is the multiset over $\Gamma \setminus \mathcal{E}$ associated with the environment at configuration $\mathcal{C}_t$. In particular, $M_{r,i_{out}}$ is the multiset over $\Gamma \setminus \mathcal{E}$ associated with the environment at the halting configuration $\mathcal{C}_r$. We say that a computation $\mathcal{C}$ is an *accepting computation* (resp., *rejecting computation*) if $Output(\mathcal{C}) = \texttt{yes}$ (resp., $Output(\mathcal{C}) = \texttt{no}$), that is, if object `yes` (resp., object `no`) appears in the environment associated with the halting configuration of $\mathcal{C}$, and neither object `yes` nor `no` appears in the environment associated with any non–halting configuration of $\mathcal{C}$.

If a recognizer P system has a symport rule of the type $(i, \lambda/u, 0)$ then $supp(u) \cap (\Gamma \setminus \mathcal{E}) \neq \emptyset$, that is, multiset u must contains some object from $\Gamma \setminus \mathcal{E}$ because on the contrary, all computations of Π would be not halting.

For each natural number $k \geq 1$, $\mathbf{CDC}(k)$ (resp., $\mathbf{CSC}(k)$) denotes the class of recognizer P systems with membrane division (resp., membrane separation) and with communication rules of length at most k. The corresponding classes associated with P systems without environment will be denoted by $\widehat{\mathbf{CDC}}(k)$ and $\widehat{\mathbf{CSC}}(k)$, respectively.

Next, we define what solving a decision problem in a uniform and efficient way by recognizer membrane systems, means (see [20] for more details). These systems have a finite description, so in order to solve a decision problem, a numerable family of membrane systems will be needed.

Definition 3.6. *A decision problem $X = (I_X, \theta_X)$ is solvable in a uniform way and polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer membrane if the following holds:*

– *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 *with regard to (X, cod, s);*

  ⋆ *the family $\mathbf{\Pi}$ is* sound *with regard to (X, cod, s);*

  ⋆ *the family $\mathbf{\Pi}$ is* complete *with regard to (X, cod, s).*

From the soundness and completeness conditions above we deduce that for each instance $u \in I_X$, the system $\Pi(s(u)) + cod(u)$ is *confluent*, in the following sense: every computation always gives the *same* answer.

Let $\mathbf{R}$ be a class of recognizer P systems with symport/antiport rules. We denote by $\mathbf{PMC_R}$ the set of all decision problems which can be solved in a uniform way and polynomial time by means of families of systems from $\mathbf{R}$. The class $\mathbf{PMC_R}$ is closed under complement and polynomial–time reductions (see [20] for more details).

4. Recognizer membrane systems from CDC($\boldsymbol{k}$): Technical results

In this Section, an upper bound of the number of objects handled along any computation of each system from CDC(k), $k \geq 2$, corresponding to a family solving a decision problem in a polynomial and uniform way, is provided.

Lemma 1. *Let* $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{in}, i_{out})$ *be a recognizer P systems with symport/antiport rules with length at most* $k \geq 2$, *and without membrane division/separation. Let* $M = |\mathcal{M}_1 + \ldots + \mathcal{M}_q|$ *and let* $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_r)$ *be a computation of* Π. *Then,* $|\mathcal{C}_0^*| = M$, *and for each* t, $0 \leq t < r$, *we have* $|\mathcal{C}_{t+1}^*| \leq M \cdot k^{t+1}$

Proof. First, let us note that $|\mathcal{C}_0^*| = |\mathcal{C}_0(0) + \mathcal{C}_0(1) + \ldots + \mathcal{C}_0(q)| = M$. In order to compute $\mathcal{C}_{t+1}^* = \mathcal{C}_{t+1}(0) + \mathcal{C}_{t+1}(1) + \ldots + \mathcal{C}_{t+1}(q)$, for $0 \leq t < r$, let us note that only the skin membrane can send and receive objects from the environment. Then, on one hand

$$\mathcal{C}_{t+1}(0) + \mathcal{C}_{t+1}(2) + \mathcal{C}_{t+1}(3) + \ldots + \mathcal{C}_{t+1}(q) \subseteq \mathcal{C}_t(0) + \mathcal{C}_t(1) + \ldots + \mathcal{C}_t(q)$$

On the other hand, at step $t+1$, membrane 1 can receive objects from $\mathcal{C}_t(0)$ and objects from $\mathcal{E}$ by means of rules in the skin membrane of the types:

– $(a\ e_{i_1} \ldots\ e_{i_s}, in)$ with $a \in \mathcal{C}_t(0)$ and $e_{i_1}, \ldots, e_{i_s} \in \mathcal{E}$, $s \leq k-1$.

– $(a, out; e_{i_1} \ldots\ e_{i_s}, in)$ with $a \in \mathcal{C}_t(1)$ and $e_{i_1}, \ldots, e_{i_s} \in \mathcal{E}$, $s \leq k-1$.

Thus, $|\mathcal{C}_{t+1}(1)| \leq |\mathcal{C}_t(0) + \mathcal{C}_t(1)| \cdot (k-1)$. So, for each t, $0 \leq t < r$, we have

$$\begin{array}{rcl} |\mathcal{C}_{t+1}^*| & = & |\mathcal{C}_{t+1}(0) + \mathcal{C}_{t+1}(2) + \mathcal{C}_{t+1}(3) + \ldots + \mathcal{C}_{t+1}(q)| + |\mathcal{C}_{t+1}(1)| \\ & \leq & |\mathcal{C}_t(0) + \mathcal{C}_t(1) + \ldots + \mathcal{C}_t(q)| + |\mathcal{C}_t(0) + \mathcal{C}_t(1)| \cdot (k-1) \\ & \leq & |\mathcal{C}_t^*| + |\mathcal{C}_t^*| \cdot (k-1) \leq |\mathcal{C}_t^*| \cdot k \end{array}$$

Finally, it is easy to show that $|\mathcal{C}_{t+1}^*| \leq M \cdot k^{t+1}$, for $0 \leq t < r$, by induction on t.

□

Proposition 1. *Let* $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ *a family of systems from* $\mathbf{CDC}(k)$, $k \geq 2$, *solving a decision problem* $X = (I_X, \theta_X)$ *in a uniform way and polynomial time. Let* (cod, s) *be a polynomial encoding associated with that solution. Then, there exists a polynomial function* $f(n)$ *such that for each instance* $u \in I_X$ *we have:*

(*a*) $2^{f(|u|)}$ *is an upper bound of the number of objects in all membranes of the system* $\Pi(s(u)) + cod(u)$ *along any computation.*

(*b*) $2^{f(|u|)}$ *is an upper bound of the number of objects from* $\mathcal{E}$ *which are moved from the environment to all membranes of the system* $\Pi(s(u)) + cod(u)$ *along any computation.*

Proof. Let $p(n)$ be a polynomial function such that for each instance $u \in I_X$ every computation of $\Pi(s(u)) + cod(u)$ is a halting computation and it performs at most $p(|u|)$ computation steps. Let us denote

$$\Pi(s(u)) + cod(u) = (\Gamma, \mathcal{E}, \Sigma, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out})$$

Let $M = |\mathcal{M}_1 + \ldots + \mathcal{M}_q|$ and $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_r)$, $0 \leq r \leq p(|u|)$, be a computation of Π.
(a) If only communication rules at m consecutive transition steps are applied, then from Lemma 1 we deduce that $|\mathcal{C}_0^*| = M$ and $|\mathcal{C}_{t+1}^*| \leq M \cdot k^{t+1}$, for each $t,\ 0 \leq t < r$. Thus, if we apply in a consecutive way the maximum possible number of communication rules (without applying any division rules) to the system $\Pi(s(u)) + cod(u)$, in any instant of any computation of the system, $M \cdot k^{p(|u|)}$ is an upper bound of the number of objects in the whole system.

Now, let us consider the effect of applying in a consecutive way the maximum possible number of division rules (without applying any communication rules) to the system $\Pi(s(u)) + cod(u)$ when the initial configuration has $M \cdot k^{p(|u|)}$ objects. After that, an upper bound of the number of objects in the whole system by any computation is $M \cdot k^{p(|u|)} \cdot 2^{p(|u|)} \cdot p(|u|)$. Then, we consider a polynomial function $f(n)$ such that $f(|u|) \geq \log(M) + p(|u|) \cdot (1 + \log\, k) + \log(p(|u|))$, for each instance $u \in I_X$. The polynomial function $f(n)$ fulfills the property required.

Finally, (b) follows from (a).

□

5. Simulating the environment in systems from $\widehat{\mathbf{CDC}}(\boldsymbol{k})$

In this Section, the meaning of efficient simulations in the framework of recognizer P systems with symport/antiport rules is defined and it is shown that any system $\Pi(n)$ from $\mathbf{CDC}(k)$, $k \geq 1$, belonging to a family solving a decision problem in a unifor way and polynomial time, can be efficiently simulated by a system $\Pi'(n)$ from $\widehat{\mathbf{CDC}}(k)$.

Definition 5.1. *Let Π and Π' be recognizer membrane systems. We say that* Π' simulates Π in an efficient way *if the following holds:*

- *Π' can be constructed from Π by a deterministic Turing machine working in polynomial time.*
- *There exists an injective function, $\mathbf{F}$, from the set $\mathbf{Comp}(\Pi)$ of computations of Π onto the set $\mathbf{Comp}(\Pi')$ of computations of Π' such that:*
  - $\star$ *There exists a deterministic Turing machine that constructs computation $\mathbf{F}(\mathcal{C})$ from computation $\mathcal{C}$ in polynomial time.*
  - $\star$ *A computation $\mathcal{C} \in \mathbf{Comp}(\Pi)$ is an accepting computation if and only if $\mathbf{F}(\mathcal{C}) \in \mathbf{Comp}(\Pi')$ is an accepting one.*
  - $\star$ *There exists a polynomial function $p(n)$ such that for each computation $\mathcal{C} \in \mathbf{Comp}(\Pi)$ we have $|\mathbf{F}(\mathcal{C})| \leq p(|\mathcal{C}|)$.*

In what follows throghout this section, let $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ a family of recognizer P systems with symport/antiport rules and membrane division solving a decision problem X in a uniform way and polynomial time. From Proposition 1 we deduce that there exists a polynomial function $f(n)$ such that for each instance $u \in I_X$, $2^{f(|u|)}$ is an upper bound of the number of objects from $\mathcal{E}$ which are moved from the environment to all membranes of the system by any computation of $\Pi(s(u)) + cod(u)$.

For each $n \in \mathbb{N}$, let $\Pi(n) = (\Gamma, \mathcal{E}, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \ldots, \mathcal{R}_q, i_{in}, i_{out})$. For the sake of simplicity we denote f instead of $f(n)$. Let us consider the recognizer P system with symport/antiport rules of degree $q_1 = 1 + q \cdot (f + 2) + |\mathcal{E}|$, with membrane division but without environment

$$\Pi'(n) = (\Gamma', \Sigma', \mu', \mathcal{M}'_0, \mathcal{M}'_1, \ldots, \mathcal{M}'_{q_1}, \mathcal{R}'_0, \mathcal{R}'_1, \ldots, \mathcal{R}'_{q_1}, i'_{in}, i'_{out})$$

defined as follows:

- $\Gamma' = \Gamma \ \cup \ \{\alpha_i : \ 0 \leq i \leq f - 1\}$ and $\Sigma' = \Sigma$.
- μ' is a rooted tree with $q_1 = 1 + q \cdot (f + 2) + |\mathcal{E}|$ nodes (membranes):
  - $\star$ A distinguished membrane labelled by 0 that is the root of μ'.
  - $\star$ For each i, $1 \leq i \leq q$, membranes labelled by $i, (i, 0), (i, 1), \ldots, (i, f)$,
  - $\star$ A membrane labelled by l_b, for each $b \in \mathcal{E}$.

  - ⋆ The root of μ', membrane 0, is the parent of the root of μ and the parent of membrane l_b, for each $b \in \mathcal{E}$.
  - ⋆ For each i, $1 \leq i \leq q$, membrane i is the parent of membrane (i, f) and membrane (i, j) is the parent of membrane $(i, j-1)$, for $1 \leq j \leq f$.

– Initial multisets: $\mathcal{M}'_0 = \emptyset$, $\mathcal{M}'_{l_b} = \{\alpha_0\}$, for each $b \in \mathcal{E}$, and for each i, $1 \leq i \leq q$, $\mathcal{M}'_{(i,0)} = \mathcal{M}_i$ and $\mathcal{M}'_i = \mathcal{M}'_{(i,j)} = \emptyset$, for $1 \leq j \leq f$.

– Set of rules: $\mathcal{R}'_0 = \emptyset$, $\mathcal{R}'_i = \mathcal{R}_i$ for $1 \leq i \leq q$, and

$$\begin{array}{lcl} \mathcal{R}'_{(i,j)} & = & \{(a, out; \lambda, in) : \ a \in \Gamma\}, \text{ for } 1 \leq i \leq q \ \wedge \ 0 \leq j \leq f \\ \mathcal{R}'_{l_b} & = & \{[\alpha_j]_{l_b} \to [\alpha_{j+1}]_{l_b} \ [\alpha_{j+1}]_{l_b} : \ 0 \leq j \leq f-2\} \cup \\ & & \{[\alpha_{f-1}]_{l_b} \to [b]_{l_b} \ [b]_{l_b}, (l_b, out; \lambda, in)\}, \text{ for each } b \in \mathcal{E} \end{array}$$

– $i'_{in} = (i_{in}, 0)$, and $i'_{out} = 0$.

Let us notice that $\Pi'(n)$ can be considered as an *extension* of $\Pi(n)$ without environment, in the following sense:

- ⋆ $\Gamma \subseteq \Gamma'$, $\Sigma \subseteq \Sigma'$, $\mathcal{E} = \emptyset$ and μ is a subtree of μ'.
- ⋆ Each membrane in $\Pi(n)$ is also a membrane in $\Pi'(n)$ and there is a distinguished membrane in $\Pi'(n)$ labelled by 0 which plays the role of environment of $\Pi(n)$.
- ⋆ Each rule in $\Pi(n)$ is also a rule in $\Pi'(n)$.

Next, we analyze the structure of the computations of system $\Pi'(n)$ and we compare them with the computations of $\Pi(n)$.

Lemma 2. *Let $\mathcal{C}' = (\mathcal{C}'_0, \mathcal{C}'_1, \ldots)$ be a computation of $\Pi'(n)$. Then,*

(a) *For each t, $0 \leq t \leq f$:*

  – *$\mathcal{C}'_t(i) = \emptyset$, for $0 \leq i \leq q$.*

  – *For each i, j, $1 \leq i \leq q$ and $0 \leq j \leq f$, we have:*

$$\mathcal{C}'_t(i, j) = \begin{cases} \mathcal{M}_i, & \text{if} \quad j = t \\ \emptyset, & \text{if} \quad j \neq t \end{cases}$$

– *For each $b \in \mathcal{E}$ there exist 2^t membranes labelled by l_b whose parent is membrane 0 and their content is:*

$$\mathcal{C}'_t(l_b) = \begin{cases} \{\alpha_t\}, & \text{if} \quad 0 \leq t \leq f-1 \\ \{b\}, & \text{if} \quad t = f \end{cases}$$

(*b*) *For $1 \leq i \leq q$, $0 \leq j \leq f$, we have $\mathcal{C}'_{f+1}(i) = \mathcal{M}_i = \mathcal{C}_0(i)$ and $\mathcal{C}'_{f+1}(i,j) = \emptyset$. Moreover, $\mathcal{C}'_{f+1}(0) = b_1^{2^f} \dots b_\alpha^{2^f}$, where $\mathcal{E} = \{b_1, \dots, b_\alpha\}$, and for each $b \in \mathcal{E}$ there exist 2^f membranes labelled by l_b whose parent is membrane 0 and their content is empty.*

Proof. Let us show (a) by induction on t. The result holds for $t = 0$ according to the initial configuration $\mathcal{C}'_0$ of system $\Pi'(n)$. Only the following rules are applicable to configuration $\mathcal{C}'_0$:

(1) $[\alpha_0]_{l_b} \to [\alpha_1]_{l_b}\ [\alpha_1]_{l_b}$, for each $b \in \mathcal{E}$.

(2) $(a, out; \lambda, in) \in \mathcal{R}_{(i,0)}$, for each $a \in supp(\mathcal{M}_i)$.

Thus, $\mathcal{C}'_1(0) = \emptyset$ and for each i, $1 \leq i \leq q$, we have $\mathcal{C}'_1(i,1) = \mathcal{M}_i$ and $\mathcal{C}'_1(i) = \mathcal{C}'_1(i,0) = \mathcal{C}'_1(i,2) = \mathcal{C}'_1(i,3) = \ldots = \mathcal{C}'_1(i,f) = \emptyset$. Moreover, for each $b \in \mathcal{E}$, there are two membranes labelled by l_b whose parent is membrane 0 and their content is $\{\alpha_1\}$. Hence, the result holds for $t = 1$.

By induction hypothesis, let t be such that $1 \leq t < f$ and let us suppose the result holds for t. Then, at configuration $\mathcal{C}'_t$ only the following rules are applicable:

(1) If $t \leq f-2$, the rules $[\alpha_t]_{l_b} \to [\alpha_{t+1}]_{l_b}\ [\alpha_{t+1}]_{l_b}$, for each $b \in \mathcal{E}$.

(2) If $t = f-1$, the rules $[\alpha_t]_{l_b} \to [b]_{l_b}\ [b]_{l_b}$, for each $b \in \mathcal{E}$.

(3) $(a, out; \lambda, in) \in \mathcal{R}'_{(i,t)}$, for each $a \in supp(\mathcal{M}_i)$.

By applying the rules of types (1) or (2) at configuration $\mathcal{C}'_t$, we deduce that there are 2^{t+1} membranes labelled by l_b in $\mathcal{C}'_{t+1}$, for each $b \in \mathcal{E}$, whose parent is membrane 0 and their content is $\{\alpha_{t+1}\}$, if $t \leq f-2$, or $\{b\}$, if $t = f-1$. By applying the rules of type (3) at configuration $\mathcal{C}'_t$, we deduce that

$$\mathcal{C}'_{t+1}(i,j) = \begin{cases} \mathcal{M}_i, & \text{if} \quad j = t+1 \\ \emptyset, & \text{if} \quad 0 \leq j \leq f \ \wedge\ j \neq f+1 \end{cases}$$

Bearing in mind that no other rule of system $\Pi'(n)$ is applicable, we deduce that $\mathcal{C}'_{t+1}(i) = \emptyset$, for $0 \le i \le q$ and the result holds for $t+1$.

In order to prove (b), let us note that from (a), the configuration $\mathcal{C}'_f$ is the following: $\mathcal{C}'_f(0) = \mathcal{C}'_f(i) = \mathcal{C}'_f(i,0) = \ldots = \mathcal{C}'_f(i, f-1) = \emptyset$ and $\mathcal{C}'_f(i,f) = \mathcal{M}_i$, for $1 \le i \le q$. Besides, for each $b \in \mathcal{E}$, there exist 2^f membranes labelled by l_b whose parent is membrane 0 and their content is $\{b\}$.

At configuration $\mathcal{C}'_f$ only the following rules are applicables:

(1) $(a, out; \lambda, in) \in \mathcal{R}'_{(i,f)}$, for each $a \in \Gamma \cap supp(\mathcal{M}_i)$.

(2) $(b, out; \lambda, in) \in \mathcal{R}'_{l_b}$, for each $b \in \mathcal{E}$.

Thus,

– $\mathcal{C}'_{f+1}(0) = b_1^{2^f} \ldots b_\alpha^{2^f}$, where $\mathcal{E} = \{b_1, \ldots, b_\alpha\}$.

– $\mathcal{C}'_{f+1}(i) = \mathcal{M}_i = \mathcal{C}_0(i)$, for $1 \le i \le q$.

– $\mathcal{C}'_{f+1}(i,j) = \emptyset$, for $1 \le i \le q$ and $0 \le j \le f$.

– For each $b \in \mathcal{E}$, there exist 2^f membranes labelled by l_b whose parent is membrane 0 and their content is empty.

□

In order to show that system $\Pi'(n)$ simulates system $\Pi(n)$ in an efficient way, a mapping $\mathbf{F}$ from $\mathbf{Comp}(\Pi(n))$ onto $\mathbf{Comp}(\Pi'(n))$ is defined in such manner that if verifies the conditions described in Definition 5.1. For that, let $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_m)$ be a halting computation of $\Pi(n)$. Then we define the computation $\mathbf{F}(\mathcal{C}) = (\mathcal{C}'_0, \mathcal{C}'_1, \ldots, \mathcal{C}'_f, \mathcal{C}'_{f+1}, \ldots, \mathcal{C}'_{f+1+m})$ of $\Pi'(n)$ as follows:

(1) The initial configuration is:

$$\begin{cases} \mathcal{C}'_0(i) & = & \emptyset, \text{ for } 0 \le i \le q \\ \mathcal{C}'_0(i,0) & = & \mathcal{C}_0(i), \text{ for } 1 \le i \le q \\ \mathcal{C}'_0(i,j) & = & \emptyset, \text{ for } 1 \le i \le q \text{ and } 1 \le j \le f \\ \mathcal{C}'_0(l_b) & = & \{\alpha_0\}, \text{ for each } b \in \mathcal{E} \end{cases}$$

(2) The configuration $\mathcal{C}'_t$, for $1 \le t \le f+1$, is described by Lemma 2.

(3) The configuration $\mathcal{C}'_{f+1+s}$, for $0 \leq s \leq m$, coincides with the configuration $\mathcal{C}_s$ of Π, that is, $\mathcal{C}_s(i) = \mathcal{C}'_{f+1+s}(i)$, for $1 \leq i \leq q$. The content of the remaining membranes (excluding membrane 0) at configuration $\mathcal{C}'_{f+1+s}$ is equal to the content of that membrane at configuration $\mathcal{C}'_{f+1}$, that is, these membranes do not evolve after step $f+1$.

That is, every computation $\mathcal{C}$ of $\Pi(n)$ can be "reproduced" by a computation $\mathbf{F}(\mathcal{C})$ of $\Pi'(n)$ with a delay: from step $f+1$ to step $f+1+m$, the computation $\mathbf{F}(\mathcal{C})$ restricted to membranes $1,\ldots,q$ provides the computation $\mathcal{C}$ of $\Pi(n)$. From Lemma 2 we deduce that $\mathbf{F}(\mathcal{C})$ is a computation of $\Pi'(n)$ and $\mathbf{F}$ is an injective function from $\mathbf{Comp}(\Pi(n))$ onto $\mathbf{Comp}(\Pi'(n))$.

Proposition 2. *The system $\Pi'(n)$ previously defined simulates the system $\Pi(n)$ in an efficient way, according with Definition 5.1.*

Proof. In order to show that $\Pi'(n)$ can be constructed from $\Pi(n)$ by a deterministic Turing machine working in polynomial time, it is enough to note that the amount of resources needed to generate $\Pi'(n)$ from $\Pi(n)$ is polynomial in the size of the initial resources of $\Pi(n)$. Indeed,

- The size of the alphabet of $\Pi'(n)$ is $|\Gamma'| = |\Gamma| + f$.
- The initial number of membranes of $\Pi'(n)$ is $1 + q \cdot (f+2) + |\mathcal{E}|$.
- The initial number of objects of $\Pi'(n)$ is the initial number of objects of $\Pi(n)$ plus $|\mathcal{E}|$.
- The number of rules of $\Pi'(n)$ is $|\mathcal{R}'| = |\mathcal{R}| + (r+1) \cdot |\mathcal{E}| + |\Gamma| \cdot q \cdot (f+1)$.
- The maximal length of a communication rule of $\Pi'(n)$ is equal to the maximal length of a communication rule of $\Pi(n)$.

From Lemma 2 we deduce that:

(a) Every computation $\mathcal{C}'$ of $\Pi'(n)$ has associated a computation $\mathcal{C}$ of $\Pi(n)$ such that $\mathbf{F}(\mathcal{C}) = \mathcal{C}'$ in a natural way.

(b) The function $\mathbf{F}$ is injective.

(c) A computation $\mathcal{C}$ of $\Pi(n)$ is an accepting computation if and only if $\mathbf{F}(\mathcal{C})$ is an accepting computation of $\Pi'(n)$.

Finally, let us notice that if $\mathcal{C}$ is a computation of $\Pi(n)$ with length m, then $\mathbf{F}(\mathcal{C})$ is a computation of $\Pi'(n)$ with length $f+1+m$.

□

6. The irrelevance of the environment in CDC

In this section, the role of the environment in P systems with symport/antiport rules and membrane division is analized from a computational complexity point of view. That is, the ability of these P systems with respect to their computational efficiency when the alphabet of the environment is the empty set, is studied.

Theorem 1. *For each $k \in \mathbb{N}$ we have* $\mathbf{PMC}_{\mathbf{CDC}(k+1)} = \mathbf{PMC}_{\widehat{\mathbf{CDC}}(k+1)}$.

Proof. Let us recall that $\mathbf{PMC}_{\mathbf{CDC}(1)} = \mathbf{P}$ (see [8] for details). Then,

$$\mathbf{P} \subseteq \mathbf{PMC}_{\widehat{\mathbf{CDC}}(1)} \subseteq \mathbf{PMC}_{\mathbf{CDC}(1)} = \mathbf{P}$$

Thus, the result holds for $k = 0$. Next, let us show the result holds for $k \geq 1$. Since $\widehat{\mathbf{CDC}}(k+1) \subseteq \mathbf{CDC}(k+1)$ it suffices to prove that $\mathbf{PMC}_{\mathbf{CDC}(k+1)} \subseteq \mathbf{PMC}_{\widehat{\mathbf{CDC}}(k+1)}$. For that, let $X \in \mathbf{PMC}_{\mathbf{CDC(k+1)}}$.

Let $\{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of P systems from $\mathbf{CDC}(k+1)$ solving X according to Definition 3.6. Let (cod, s) be a polinomial encoding associated with that solution. Let $u \in I_X$ be an instance of the problem X that will be processed by the system $\Pi(s(u)) + cod(u)$. According to Proposition 1, there exists a polynomial function $f(n)$ that $2^{f(|u|)}$ is an upper bound of the number of objects from $\mathcal{E}$ which are moved from the environment to all membranes of the system by any computation of the system of degree q:

$$\Pi(s(u)) + cod(u) = (\Gamma, \mathcal{E}, \Sigma, \mathcal{M}_1, \dots, \mathcal{M}_{i_{in}} + cod(u), \dots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$$

Then, we consider the "corresponding" P system without environment of degree $q_1 = 1 + q \cdot (f(|u|) + 2) + |\mathcal{E}|$, according with the previous definition:

$$\Pi'(s(u)) + cod(u) = (\Gamma', \Sigma', \mathcal{M}'_0, \mathcal{M}'_1, \dots, \mathcal{M}'_{i_{in}} + cod(u), \dots, \mathcal{M}'_{q_1}, \mathcal{R}', i'_{in}, i'_{out})$$

Therefore, if $\Pi(s(u)) + cod(u)$ is a membrane system from $\mathbf{CDC}(k+1)$ then $\Pi'(s(u)) + cod(u)$ is a membrane system from $\widehat{\mathbf{CDC}}(k+1)$ such that verifies the following:

– A distinguished membrane labelled by 0 has been considered, which will play the role of the environment at the system $\Pi(s(u)) + cod(u)$.

– At the initial configuration, it has enough objects in membrane 0 in order to simulate the behaviour of the environment of the system $\Pi(s(u))) + cod(u)$.

– After $f(n) + 1$ step, computations of $\Pi(s(u)) + cod(u)$ are reproduced by the computations of $\Pi'(s(u)) + cod(u)$ exactly.

Let us suppose that $\mathcal{E} = \{b_1, \ldots, b_\alpha\}$. In order to simulate $\Pi(s(u))+cod(u)$ by a P system without environment $\Pi'(s(u))+cod(u)$ in an efficient way, we need to have enough objects in the membrane labelled by 0 of $\Pi'(s(u)) + cod(u)$ available. Specifically, $2^{f(|u|)}$ objects in that membrane are enough.

In order to start the simulation of any computation $\mathcal{C}$ of $\Pi(s(u))+cod(u)$, it would be enough to have $2^{f(|u|)}$ copies of each object $b_j \in \mathcal{E}$ in the membrane of $\Pi'(s(u)) + cod(u)$ labelled by 0. For this purpose,

– For each $b \in \mathcal{E}$ a membrane in $\Pi'(s(u)) + cod(u)$ labelled by l_b is considered which only contains object α_0 initially. We also consider the following rules:

  ⋆ $[\alpha_j]_{l_b} \rightarrow [\alpha_{j+1}]_{l_b}\ [\alpha_{j+1}]_{l_b}$, for $0 \leq j \leq f(|u|) - 2$,

  ⋆ $[\alpha_{f(|u|)-1}]_{l_b} \rightarrow [b]_{l_b}\ [b]_{l_b}$,

  ⋆ $(l_b, b/\lambda, 0)$.

– By applying the previous rules, after $f(|u|)$ transition steps we get $2^{f(|u|)}$ membranes labelled by l_b, for each $b \in \mathcal{E}$ in such a way that each of them contains only object b. Finally, by applying the third rule we get $2^{f(|u|)}$ copies of objects b in membrane 0, for each $b \in \mathcal{E}$.

Therefore, after the execution of $f(|u|) + 1$ transition steps in each computation of $\Pi'(s(u)) + cod(u)$ in membrane 0 of the corresponding configuration, we have $2^{f(|u|)}$ copies of each object $b \in \mathcal{E}$. This number of copies is enough to simulate any computation $\mathcal{C}$ of $\Pi(s(u)) + cod(u)$ through the system $\Pi'(s(u)) + cod(u))$.

From Proposition 2 we deduce that the family $\{\Pi'(n)|\ n \in \mathbb{N}\}$ of systems from $\widehat{\mathbf{CDC}}(k+1)$ solves X in a uniform way and polynomial time according to Definition 3.6. Hence, $X \in \mathbf{PMC}_{\widehat{\mathbf{CDC}}(k+1)}$.

□

From this result, a new frontier of the efficiency can be obtained in the framework of P systems with symport/antiport rules with membrane division and without environment, expressed in terms of the length of communication rules. Indeed, of the one hand it is well known [3] that by using families from $\mathbf{CDC}(1)$ only problems in class $\mathbf{P}$ can be efficiently solved, in particular $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathbf{CDC}}(1)}$, that is, computing models from $\widehat{\mathbf{CDC}}(1)$ are non-efficient. On the other hand, in [25] a uniform and polynomial time solution of the `HAM-CYCLE` problem by a family of P systems from $\mathbf{CDC}(2)$ is given. From Theorem 1 we deduce that `HAM-CYCLE` $\in \mathbf{PMC}_{\widehat{\mathbf{CDC}}(2)}$, that is, computing models from $\widehat{\mathbf{CDC}}(2)$ are presumably efficient. Hence, assuming that $\mathbf{P} \neq \mathbf{NP}$, in the framework of P systems with membrane division and without environment, the length of communication rules provides a new borderline of the tractability: passing from 1 to 2 amounts to passing from non-efficiency to the (presumably) efficiency.

7. The relevance of the environment in CSC

On the one hand, in [6] it has been shown that $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathbf{CSC}}(k+1)}$, for each $k \in \mathbb{N}$, that is, computing models from $\widehat{\mathbf{CSC}}(k+1)$ are non-efficient. On the other hand, in [5], a uniform and polynomial time solution to the `SAT` problem has been given by means of a family of systems from $\mathbf{CSC}(3)$, that is, computing models from $\mathbf{CSC}(3)$ are presumably efficient. Thus, assuming that $\mathbf{P} \neq \mathbf{NP}$, in the framework of P systems with membrane separation which use communication rules of length at most three, the environment provides a new borderline between the non-efficiency and the (presumably) efficiency.

Besides, computing models from $\widehat{\mathbf{CSC}}(3)$ are non-efficient and it has been show that `VERTEX-COVER` $\in \mathbf{PMC}_{\mathbf{CDC}(3)}$ [2]. Then, from Theorem 1 we deduce that computing models from $\widehat{\mathbf{CDC}}(3)$ are presumably efficient. Thus, assuming that $\mathbf{P} \neq \mathbf{NP}$, in the framework of P systems without environment and with communication rules with length at most 3, the kind of rules (separation versus division) provides a new borderline between the non-efficiency and the (presumably) efficiency.

8. When the structure does not matter

Networks of cells linked by *communication channels* which compute by communication only were first considered in [13]. These networks were formally structured by means of *directed graphs* leading to tissue P systems with symport/antiport rules. The cell-like version is structured by means of *rooted trees* and they were introduced in [14].

By using similar notations, the classes of recognizer tissue P systems $\mathbf{TDC}(k)$, $\mathbf{TSC}(k)$, $\widehat{\mathbf{TDC}}(k)$ and $\widehat{\mathbf{TSC}}(k)$ are defined in a natural way. The following results have been obtained:

– $\mathbf{PMC}_{\mathbf{TDC}(1)} = \mathbf{P}$ and $\mathbf{PMC}_{\mathbf{TSC}(2)} = \mathbf{P}$ ([3] and [12]).

– For each $k \in \mathbb{N}$ we have $\mathbf{PMC}_{\widehat{\mathbf{TSC}}(k+1)} = \mathbf{P}$ and $\mathbf{PMC}_{\mathbf{TDC}(k+1)} = \mathbf{PMC}_{\widehat{\mathbf{TDC}}(k+1)}$ ([7] and [23]).

– `HAM-CYCLE` $\in \mathbf{PMC}_{\mathbf{TDC}(2)}$ and `SAT` $\in \mathbf{PMC}_{\mathbf{TSC}(3)}$ ([22] and [24]).

Comparing these results with the ones commented in the previous section, it is easy to see that while changing the underlying directed graph structure of tissue P systems with symport/antiport rules to the explicit rooted tree-like structure of their cell-like counterpart, the frontiers from non-efficiency to presumably efficiency are the same while keeping the length of communication rules. In some sense, we can affirm that from a complexity view the structure of these systems does not matter.

9. Conclusions

In cell-like P systems with symport/antiport rules, $\mathcal{E}$ represents objects that appear in an arbitrary large amount at the environment. This condition is not too nice from the computational complexity point of view. In this paper, we show that in P systems with symport/antiport rules and membrane division the environment can be “removed” without a loss of efficiency, that is, its role is irrelevant from a complexity view. However, by using membrane separation instead of membrane division, the environment plays a crucial role on the (presumably) efficiency of these systems. Besides, in this context it has been justified that the structure of these types of P systems (directed graph versus rooted trees) does not matter.

Acknowledgements

This work was supported by Project TIN2017-89842-P of the Ministerio de Economía y Competitividad of Spain and by Grants No 61320106005 of the National Natural Science Foundation of China.

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest. *An Introduction to Algorithms*. The MIT Press, Cambridge, Massachussets, Third Edition, 2009.

[2] D. Díaz-Pernil, M.Á. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Á. Romero-Jiménez. Computational efficiency of cellular division in tissue-like P systems. *Romanian Journal of Information Science and Technology*, **11**, 3 (2008), 229-241.

[3] 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.

[4] M. Ionescu, Gh. Păun, T. Yokomori. Spiking neural P systems. *Fundamenta Informaticae*, **71**, 2-3 (2006), 279–308.

[5] L.F. Macías-Ramos, B. Song, L. Valencia-Cabrera, L. Pan, M.J. Pérez-Jiménez. Membrane fission: A computational complexity perspective. *Complexity*, **21**, 6 (2016), 321-334.

[6] L.F. Macías-Ramos, M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font, L. Valencia-Cabrera. Membrane fission versus cell division: when membrane proliferation is not enough. *Theoretical Computer Science*, **608** (2015), 57-65

[7] L.F. Macías-Ramos, M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font, L. Valencia-Cabrera. The efficiency of tissue P systems with cell separation relies on the environment. *Lecture Notes in Computer Science*, **7762** (2013), 243-256.

[8] L.F. Macías-Ramos, B. Song, T. Song, L. Pan, M.J. Pérez-Jiménez. Limits on efficient computation in P systems with symport/antiport. In C. Graciani, Gh. Păun, A. Riscos-Núñez, L. Valencia-Cabrera (eds.). *Proceedings of the Fifteenth Brainstorming Week on Membrane Computing*, Sevilla, Spain, January 31, February 3, 2017, Fénix Editora, pp. 147-160.

[9] C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodriguez-Paton. Tissue P systems. *Theoretical Computer Science*, **296**, 2 (2003), 295–326.

[10] L. Pan, T.-O. Ishdorj. P systems with active membranes and separation rules. *Journal of Universal Computer Science*, **10**, 5 (2004), 630–649.

[11] L. Pan, M.J. Pérez-Jiménez. Computational complexity of tissue–like P systems. *Journal of Complexity*, **26**, 3 (2010), 296–315.

[12] L. Pan, M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font. New frontiers of the efficiency in tissue P systems. L. Pan, Gh. Păun, T. Song (eds.) *Pre-proceedings of Asian Conference on Membrane Computing (ACMC 2012)*, Huazhong University of Science and Technology, Wuhan, China, October 15-18, 2012, pp. 61-73.

[13] 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.

[14] A. Păun, Gh. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, **20**, 3 (2002), 295–305.

[15] Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, **61**, 1 (2000), 108–143.

[16] Gh. Păun. P systems with active membranes: attacking **NP**–complete problems. *Journal of Automata, Languages and Combinatorics*, **6** (2001), 75–90.

[17] Gh. Păun. *Membrane Computing. An Introduction*. Springer–Verlag, Berlin, 2002.

[18] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez. Tissue P Systems with cell division. *International Journal of Computers, Communications and Control*, **3**, 3 (2008), 295-303.

[19] Gh. Păun, G. Rozenberg, A. Salomaa (eds.). *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, 2010.

[20] M.J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, **2**, 3 (2003), 265-285.

[21] M.J. Pérez-Jiménez, Á. 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.

[22] A.E. Porreca, N. Murphy, M.J. Pérez-Jiménez. An optimal frontier of the efficiency of tissue P systems with cell division. In M. García-Quismondo, L.F. Macías-Ramos, Gh. Păun, I. Pérez-Hurtado, L. Valencia-Cabrera (eds.) *Proceedings of the Tenth Brainstorming Week on Membrane Computing*, Volume II, Seville, Spain, January 30- February 3, 2012, Report RGNC 01/2012, Fénix Editora, 2012, pp. 141-166.

[23] 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. International Journal of Computer Mathematics, **90**, 4 (2013), 760-775

[24] M.J. Pérez-Jiménez, P. Sosík. An Optimal Frontier of the Efficiency of Tissue P Systems with Cell Separation. Fundamenta Informaticae, **138**, 1-2 (2015), 45-60

[25] L. Valencia-Cabrera, B. Song, L.F. Macías-Ramos, L. Pan, A. Riscos-Núñez, M.J. Pérez-Jiménez. Minimal cooperation in P systems with symport/antiport: A complexity approach. In L.F. Macías-Ramos, Gh. Păun, A. Riscos-Núñez, L. Valencia-Cabrera (eds.). *Proceedings of the Thirteenth Brainstorming Week on Membrane Computing*, Sevilla, Spain, February 2-6, 2015, Fénix Editora, pp. 301-323.