Proof techniques in Membrane Computing

David Orellana-Martín *, Luis Valencia-Cabrera, Mario J. Pérez-Jiménez

Research Group on Natural Computing, Dept. of Computer Science and Artificial Intelligence, Universidad de Sevilla, E.T.S.I. Informática, Avda. Reina Mercedes s/n, 41012, Sevilla, Spain

Keywords:
Membrane Computing
Computability theory
Computational complexity theory

A B S T R A C T

From the creation of the field of Membrane Computing in 1998, several research lines have been opened. On the one hand, theoretical questions like the computational power and the computational efficiency of P systems have been studied. In this sense, several techniques to demonstrate the ability of these systems to provide solutions to computational problems have been explored. The study of *efficient* (polynomial-time) solutions to presumably hard problems for finding thin frontiers of efficiency is a very active area. On the other hand, several applications in biology, ecology, economy, robotics and fault diagnosis, among others, have been investigated. Real systems with some characteristics seem to be easy to model with membrane systems due to their behaviour. In this work, a survey of the theoretical part will be given, explaining techniques both in the field of computability theory and in the field of computational complexity theory.

1. Introduction

Membrane Computing was created by Gh. Păun [17] inspired from the fact that processes which take place in the complex structure of a living cell can be considered, in some sense, as singular computing procedures. At the beginning [19], the founder asks about the roots of the Membrane Computing discipline: formal multiset theory, formal language theory? He is convinced that *the whole program of formal language theory can be repeated here, irrespective of the fact whether or not we deal with languages or with multisets* [19]. Nevertheless, it seems that within Membrane Computing there is a lot of world beyond formal multisets and formal languages.

In this paper, different proof techniques that have been and continue to be used in Membrane Computing, in order to obtain interesting results, will be described and illustrated. The work is focuses on aspects concerning to the computability theory (*computational completeness*) and to the computational complexity theory (*computational efficiency*) of membrane systems. With respect to the computational completeness, the equivalence in power to deterministic Turing machines of membrane systems, is studied. With respect to the computational efficiency, the capability of membranes systems to provide *efficient* (polynomial-time) solutions of computationally hard problems, is analized. For an introduction to membrane systems and notation used through this paper we refer the reader to [16]. Apart from the techniques appearing in this paper, we refer the reader to [12].

The paper is organized as follows. In Section 2 some techniques used to demonstrate the computational completeness of different variants of membrane systems are introduced. Section 3 is devoted to introducing some basic definitions of

* Corresponding author.
E-mail addresses: dorellana@us.es (D. Orellana-Martín), lvalencia@us.es (L. Valencia-Cabrera), marper@us.es (M.J. Pérez-Jiménez).

computational complexity terms, as well as some techniques used for this topic among others. The paper concludes with some comments and suggestions for future research.

2. Computational completeness

The computational completeness of computing models of membrane systems was initially showed by making use of techniques of formal languages and multisets [4] (i.e. grammars, matrix grammars) as well as simulating the behaviour of well known universal models, as register machines [8,27]. The universality of basic transition P systems with external output, as devices able to compute functions, was obtained by showing that these systems have the capability of computing any partial recursive function on natural numbers [24]. For that, operations of composition, iteration and unbounded minimization (μ-recursion), among such kinds of P systems were introduced and their closure was established.

The universality of computing models of membrane systems, also can be reached by simulating universal devices as *deterministic Turing machines* or by generating *diophantine sets*. Specifically, given a deterministic Turing machine, a P system with external output associated with it was constructed, in such manner that it generates the same language as the one recognized by the Turing machine [22]. In this context, it is worth noting that there is an important difference between Turing machines and P systems of the type considered above: the Turing machines work with strings (instantaneous descriptions of the tape), while P systems work with symbol/objects and multisets of symbols. Hence, in principle, P systems can compute nothing else than numbers (of objects). On the other hand, by using only operations of composition and iteration with computing P systems, every *diophantine set* can be partially generated by such kind of computing models [23]. Thus, by using the MRDP (Matijasevich, Robinson, Davis, Putnam) theorem, the universality of such computing P systems is followed.

3. Computational efficiency

The *computational efficiency* of a computing model in a computing paradigm, refers to its ability to provide polynomial time solutions for computationally hard problems by making use of an exponential workspace constructed in a "natural way". Following [11], a computing model is said to be *efficient* (respectively, *presumably efficient*) if it has the ability to provide polynomial-time solutions for intractable problems (resp., **NP**-complete problems). The term *presumably efficient* refers to the fact that in the case, generally believed, that $\mathbf{P} \neq \mathbf{NP}$, each **NP**-complete problem is an intractable one; consequently, under this hypothesis, any presumably efficient computing model would be efficient.

Aspects related to the computational efficiency were first analyzed in 1999 with the introduction of a new computing model, called *P system with active membranes* [18]. These systems are non-cooperative (the left hand side of any rule consists of only one object) and their membranes play a relevant role in computations to the extent that new membranes can be created by division rules. The membranes of these systems are supposed to have one of the three possible electrical polarizations: positive, negative or neutral. In this context, an *ad-hoc* solution to the Boolean satisfiability problem (SAT) by means of such kind of P systems, was given, understanding that it is an *ad-hoc* solution since for every instance of the SAT problem, a specific P system is required. Specifically, a P system with active membranes which makes use of *simple* object evolution rules (only one object is produced for this kind of rules), dissolution rules and division rules for elementary and non-elementary membranes, is associated with every instance φ of SAT. Thus, the syntactic structure of the formula is "captured" by the description of the system and, furthermore, in this context a P system can only process one instance of the problem. The provided solution runs in linear time with respect to the input length of φ, that is, the maximum of number of variables and number of clauses of the formula φ.

The concept of solvability of decision problems in polynomial time and uniform way by means of a family of basic *recognizer transition P systems* (initially called *decision P systems*) were defined in [21].

Definition 1. Let $X = (I_X, \theta_X)$ be a decision problem and let $\mathcal{R}$ be a class of recognizer membrane systems with input membrane. Let $\Pi + w$ be a P system Π with the multiset w located in the input membrane in the initial configuration. We say that X is solvable in polynomial time and uniform way by a family $\{\Pi(n) \mid n \in \mathbb{N}\}$ of systems from $\mathcal{R}$, denoted by $X \in \mathbf{PMC}_{\mathcal{R}}$, if the following holds:

- The family is *polynomially uniform by Turing machines*, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from the number $n \in \mathbb{N}$, expressed in unary.
- There exists a pair (cod, s) of polynomial-time computable functions over I_X such that for each $n \in \mathbb{N}$, the set $s^{-1}(n)$ is finite, and for each $u \in I_X$, $s(u) \in \mathbb{N}$ and $cod(u)$ is an input multiset of the system $\Pi(s(u))$.
- The family is *polynomially bounded with respect to* (X, cod, s); that is, there exists $k \in \mathbb{N}$ such that for each $u \in I_X$, every computation of the system $\Pi(s(u)) + cod(u)$ performs at most $|u|^k$ steps.
- The family is *sound with respect to* (X, cod, s), that is, for each $u \in I_X$, if *there exists* an accepting computation of $\Pi(s(u)) + cod(u)$ then $\theta_X(u) = 1$.
- The family is *complete with respect to* (X, cod, s), that is, for each $u \in I_X$ such that $\theta_X(u) = 1$ then *every* computation of $\Pi(s(u)) + cod(u)$ is an accepting one.

According to the previous definition:

- We say that the family $\{\Pi(n) \mid n \in \mathbb{N}\}$ provides a *polynomial time and uniform solution* to the problem X, and the ordered pair (cod, s) is a *polynomial encoding* from the problem X to the family $\{\Pi(n) : n \in \mathbb{N}\}$.
- For each instance $u \in I_X$, the system $\Pi(s(u))$ processes u when the input multiset $cod(u)$ is supplied to the corresponding input membrane. Besides, the recognizer membrane system $\Pi(s(u)) + cod(u)$ is *confluent*, in the sense that all computations must give the same answer (either all computations are accepting ones or all computations are rejecting ones).

As a direct consequence of working with recognizer membrane systems, these complexity classes are closed under complement. Moreover, it is easy to prove that they are closed under polynomial-time reductions [25]. These families of recognizer membrane systems provide an efficient solution to a decision problem. Their formal verification is crucial to understand the behaviour and to prove that the answer of the system is always correct. Some examples of efficient solutions for presumably intractable problems can be found in the bibliography [2,9,13].

A related concept to maintain compatibility with previous *ad-hoc* solutions was created, in the sense that these *non-uniform* solutions associate a membrane system with a single input; that is, every $\Pi(u), u \in I_X$, will solve a single input, in this case, u. The class of problems solvable in polynomial time and non-uniform way by a family of systems from $\mathcal{R}$ ($\mathcal{R}$ being a class of recognizer membrane systems without input membrane) is denoted by $\mathbf{PMC}^{*}_{\mathcal{R}}$. In order to establish the presumed efficiency of some computing models of membrane systems it suffices to design polynomial time and uniform solutions to **NP**-complete problems by means of families of these kinds of systems. Thus, it is interesting to develop some design techniques but recalling that this development can not be "mechanized". However, some patterns can be observed during the process of creating a new efficient solution to **NP**-complete problems by means of a uniform family of membrane systems [26].

It seems interesting to analyse what kind of membrane systems are capable of solving decision problems through only one unique system. In this context, it is fundamental how the instances of the problem are introduced into the system. Next, we consider the case in which the instances are directly introduced inside the system (*free* of resources) by means of a representation of the problem to be solved; that is, the instance is introduced without any previous encoding, e.g. $X = (I_X, \theta_X)$, and $I_X = a^n$ for $n \geq 0$, then the input multiset will be the multiset $\{a^n\}$.

Definition 2. Let $X = (I_X, \theta_X)$ be a decision problem where I_X is a language over a finite alphabet Σ_X. Let $\mathcal{R}$ be a class of recognizer membrane systems with input membrane. We say that problem X is solvable in polynomial time by a single membrane system Π from $\mathcal{R}$, free of resources, denoted by, $X \in \mathbf{PMC}^{1f}_{\mathcal{R}}$, if the following hold:

- The input alphabet of Π is Σ_X.
- The system Π is *polynomially bounded* with regard to X; that is, there exists $k \in \mathbb{N}$ such that for each instance $u \in I_X$, every computation of the system Π with input multiset u performs at most $|u|^k$ steps.
- The system Π is *sound* with regard to X; that is, for each instance $u \in I_X$, if there exists an accepting computation of the system $\Pi + u$ then $\theta_X(u) = 1$.
- The system Π is *complete* with regard to X; that is, for each instance $u \in I_X$ such that $\theta_X(u) = 1$, every computation of the system $\Pi + u$ is an accepting one.

From the previous definition it is easy to prove that $\mathbf{PMC}^{1f}_{\mathcal{R}} \subseteq \mathbf{PMC}_{\mathcal{R}}$, for every class $\mathcal{R}$ of recognizer membrane systems with input membrane, since every single recognizer membrane system free of resources can be seen as a family of recognizer membrane systems with a single system $\mathbf{\Pi} = \{\Pi(0)\}$ such that $s(u) = 0$ and $cod(u) = u$, for $u \in I_X$. A related concept is created concerning systems that count with a polynomial-time encoding function cod such that $\Pi + cod(u), u \in I_X$, is capable of solving the problem X. This class of problems is denoted by $\mathbf{PMC}^{1p}_{\mathcal{R}}$. It can be easily proven that $\mathbf{PMC}^{1f}_{\mathcal{R}} \subseteq \mathbf{PMC}^{1p}_{\mathcal{R}} \subseteq \mathbf{PMC}_{\mathcal{R}}$, since a single recognizer membrane system can be seen as a family of recognizer membrane systems as seen above, and every single recognizer membrane system *free of resources* can be seen as a single membrane system whose cod function is $cod(u) = u$ for every $u \in I_X$.

3.1. Dependency graph technique

The dynamic of a membrane system provides, in a natural way, a tree of computations. Specifically, the *computation tree* of a membrane system Π, denoted $\mathbf{Comp}(\Pi)$, is a rooted labelled maximal tree defined as follows:

- Nodes are labelled by configurations of Π.
- Edges are labelled by applicability matrices (maximal multiset of rules applicable to a configuration).
- The root of the tree is the initial configuration of Π.
- The children of a node labelled by $\mathcal{C}$ are the configurations $\mathcal{C}'$ which can be obtained from $\mathcal{C}$ in one transition step.

The maximal branches of **Comp**(Π) will be called *computations* of Π. A computation of Π is a halting computation if and only if it is a finite branch. The labels of the leaves of **Comp**(Π) are called *halting configurations*.

Given a semi-uniform or uniform solution (in polynomial time) for a decision problem by means of a family of recognizer membrane systems, every instance of the problem is processed by a system of the family, which is confluent. Thus, in order to know the answer of the system for any instance it is enough to consider only one computation of the previous system. In this context, an exciting challenge would be looking for a computation with minimum length. For that, it would be interesting to analyze the degree of closeness between two configurations. The problem is specially hard if we want to quantify that proximity in order to make useful comparisons. Some *weak metrics* on configurations of a membrane system with a fixed structure of membranes has been studied in [3]. In this context, in order to search the shortest paths in that graph providing a *sound* computation of the system, the *dependency graph* associated with the set of rules of a recognizer membrane system, was introduced. This concept is based on the dependence among elements of the alphabet with respect to the set of rules of the P system. Several weak metrics over the set of configurations of the system based on the concept of dependency graph, were considered, starting from the notion of *distance* between two nodes of that graph (the length of the shortest path that connect v_1 and v_2, or infinite if there is no path from v_1 to v_2).

Dependency graph as a proof technique for the non-efficiency of membrane systems

With some kind of recognizer membrane systems, it is possible to consider a directed graph (also called *dependency graph*) verifying the following properties: (a) it can be constructed from the working alphabet and the set of rules of the system, in polynomial time; that is, in a time bounded by a polynomial function depending on the size of the working alphabet and the total number of rules and the maximum length of them; and (b) accepting computations of such systems can be characterized by means of a "reachability" property in the dependency graph associated with it (the existence of a path in the graph between two specific nodes). Therefore, dependency graphs provide a technique to demonstrate the non-efficiency of membrane systems ([6], [7]).

As an illustration example, recognizer polarizationless P systems with active membranes which do not make use of dissolution rules, are considered. The rules of such systems can be considered as a *dependency* relation between the object triggering the rule and the objects produced by its application. For instance,

- Object evolution rules $[\,a \to u\,]_h$ can be described as follows: the pair (a, h) produces the pair (b, h), for each b being a symbol belonging to u.
- Send-in communication rules $a[\ ]_h \to [\,b\,]_h$ can be described as follows: the pair $(a, p(h))$ produces the pair (b, h).
- Send-out communication rules $[\,a\,]_h \to [\ ]_h\, b$ can be described as follows: the pair (a, h) produces the pair $(b, p(h))$.
- Division rules for elementary membranes $[\,a\,]_h \to [\,b\,]_h\,[\,c\,]_h$ can be described as follows: the pair (a, h) produces the pairs (b, h) and (c, h).

Let us recall that if h is the label of a membrane, then $p(h)$ denotes the label of the parent of such membrane labelled by h. We adopt the convention that the father of the skin membrane (the root of the tree) is the environment and its label is denoted by env.

It is worth pointing out that dissolution rules cannot be expressed in a similar way due to the fact that such kind of rules affects objects that do not appear in the rule, and therefore depends on the current configuration.

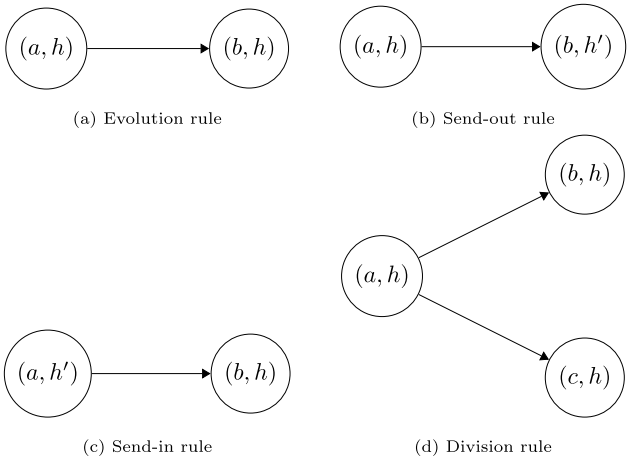
In this context, division rules for non-elementary membranes do not provide any information. These ideas can be formalized as follows:

Definition 3. Let $\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{in}, i_{out})$ be a recognizer polarizationless P system with active membranes of degree $q \geq 1$ which do not make use of dissolution rules. Let us assume that the label "env" of the environment of Π is in H. Let $supp(u)$ be the support set of u; that is, its underlying set. The *dependency graph* associated with Π is the directed graph $G_\Pi = (V_\Pi, E_\Pi)$ defined as follows:

- The set of vertices is $V_\Pi = \{s_\Pi\} \cup VL_\Pi \cup VR_\Pi$, where $s_\Pi \notin \Gamma \times H$ and:

$$\begin{aligned} VL_\Pi = \{(a,h) \in \Gamma \times H \mid\ & \exists u \in M(\Gamma)\ ([a \to u]_h \in \mathcal{R}) \vee \\ & \exists b \in \Gamma\ ([a]_h \to [\ ]_h b \in \mathcal{R}) \vee \\ & \exists b \in \Gamma\ \exists h' \in ch(h)\ (a[\ ]_{h'} \to [b]_{h'} \in \mathcal{R}) \vee \\ & \exists b, c \in \Gamma\ ([a]_h \to [b]_h[c]_h \in \mathcal{R}))\}. \end{aligned}$$

$$\begin{aligned} VR_\Pi = \{(b,h) \in \Gamma \times H \mid\ & \exists a \in \Gamma\ \exists u \in M(\Gamma)\ ([a \to u]_h \in \mathcal{R} \wedge b \in \mathrm{supp}(u)) \vee \\ & \exists a \in \Gamma\ \exists h' \in ch(h)\ ([a]_{h'} \to [\ ]_{h'} b \in \mathcal{R}) \vee \\ & \exists a \in \Gamma\ (a[\ ]_h \to [b]_h \in \mathcal{R}) \vee \\ & \exists a, c \in \Gamma\ ([a]_h \to [b]_h[c]_h \in \mathcal{R} \vee [a]_h \to [c]_h[b]_h \in \mathcal{R})\}. \end{aligned}$$



(a) Evolution rule (b) Send-out rule (c) Send-in rule (d) Division rule

Fig. 1. Graphical representation of the creation of the dependency graph depending on the rule.

– The set of arcs is $E_\Pi = E^1_\Pi \cup E^2_\Pi$, where:

$$E^1_\Pi = \{(s_\Pi, (a,h)) \mid (h \in H \setminus \{env\} \wedge a \in \mathcal{M}_h) \ \vee \ (h = i_{in} \wedge a \in cod(u))\}$$

$$\begin{aligned} E^2_\Pi = \{((a,h),(b,h')) \mid \exists u \in M(\Gamma) \ & ([a \to u]_h \in \mathcal{R} \wedge b \in \mathrm{supp}(u) \wedge h = h') \vee \\ & ([a]_h \to [\ ]_h b \in \mathcal{R} \ \wedge \ h' = p(h)) \vee \\ & (a[\ ]_{h'} \to [b]_{h'} \in \mathcal{R} \ \wedge h \ = p(h')) \vee \\ & \exists c \in \Gamma \ ([a]_h \to [b]_h[c]_h \in \mathcal{R} \wedge h = h') \vee \\ & \exists c \in \Gamma \ ([a]_h \to [c]_h[b]_h \in \mathcal{R} \wedge h = h')\}. \end{aligned}$$

Let us recall that if h is the label of a membrane, then $ch(h)$ denotes the set of labels of the children membranes of such membrane labelled by h.

The node s_Π is called the source node of G_Π. The node $(\mathtt{yes}, env)$ is called the accepting node, and the node $(\mathtt{no}, env)$ is called the rejecting node of G_Π.

A graphical interpretation of the process to create the dependency graph is the following:

- For an evolution rule $[\,a \to u\,]_h$, a node (a,h) and nodes (b,h), for $b \in supp(u)$ are defined. Edges from (a,h) to each node (b,h) are defined. This can be seen graphically in Fig. 1a.
- For a communication send-out rule $[\,a\,]_h \to b[\ \ ]_h$, a node (a,h) and a node (b,h'), for $h' = p(h)$ are defined. An edge from (a,h) to node (b,h') is defined. This can be seen graphically in Fig. 1b.
- For a communication send-in rule $a[\ \ ]_h \to [\,b\,]_h$, a node (a,h') and a node (b,h), for $h' = p(h)$ are defined. An edge from (a,h') to node (b,h) is defined. This can be seen graphically in Fig. 1c.
- For a division rule for elementary membranes $[\,a\,]_h \to [\,b\,]_h[\,c\,]_h$, nodes (a,h), (b,h) and (c,h) are defined. Edges from (a,h) to (b,h) and (c,h) are defined. This can be seen graphically in Fig. 1d.
- For a division rule for non-elementary membranes, neither new nodes nor new edges are defined.

Bearing in mind that all computations of a recognizer P system halt, any path in the dependency graph associated with it must be simple; that is, all vertices in a path are distinct.

A partial answer to the Păun's conjecture

At the beginning of 2005, Gh. Păun (problem **F** from [20]) wrote: *My favourite question (related to complexity aspects in P systems with active membranes and with electrical charges) is that about the number of polarizations. Can the polarizations be completely avoided? The feeling is that this is not possible – and such a result would be rather sound: passing from no polarization to two polarizations amounts to passing from non-efficiency to efficiency.*

Let us denote $\mathcal{AM}^0(\alpha,\beta)$ the class of all recognizer polarizationless P systems with active membranes such that: (a) if $\alpha = +d$ (resp., $\alpha = -d$) then dissolution rules are permitted (resp., forbidden); and (b) if $\beta = +ne$ (resp., $\beta = -ne$) then division rules for elementary and non-elementary membranes (resp., only for elementary membranes) are permitted. Then, the so-called *Păun's conjecture* can be formally formulated in terms of membrane computing complexity classes as follows: $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(+d,-ne)}$.

An *affirmative answer* to this conjecture would indicate that the ability to create an exponential amount of workspace (expressed in terms of the number of membranes and objects) in polynomial time, is not enough in order to solve computationally hard problems efficiently. On the other hand, assuming that $\mathbf{P} \neq \mathbf{NP}$, a *negative answer* to the conjecture would show that division rules for elementary membranes provide a borderline between the *non-efficiency* and the *presumed efficiency* of polarizationless P systems with active membranes.

The dependency graph associated with a P system from $\mathcal{AM}^0(-d,+ne)$, can be constructed by a deterministic Turing machine working in polynomial time (see [7] for details). Moreover, dependency graphs can be used to characterize the behaviour of the system through the analysis of simple paths. Specifically, given a recognizer P system Π from $\mathcal{AM}^0(-d,+ne)$, there exists an accepting computation of Π if and only if there exists a simple path in the dependency graph G_Π from the source node s_Π to the accepting node $(\mathtt{yes}, env)$ with length greater than or equal to 2 (see [7] for details).

Let $\{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of P systems from $\mathcal{AM}^0(-d,+ne)$ solving a decision problem $X = (I_X, \theta_X)$ in polynomial time. Let (cod, s) be a polynomial encoding associated with that solution. Then, for each instance $u \in I_X$, the answer of the problem is yes; that is, $\theta_X(u) = \mathtt{yes}$, if and only if there exists a simple path in the dependency graph associated with $\Pi' = \Pi(s(u)) + cod(u)$, from the source node $s_{\Pi'}$ to the accepting node $(\mathtt{yes}, env)$.

Theorem 1. $\mathbf{PMC}_{\mathcal{AM}^0(-d,+ne)} = \mathbf{P}$.

Proof. On the one hand, it is well known that if $\mathcal{R}$ is a class of recognizer membrane systems then $\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{R}}$. On the other hand, in order to show that $\mathbf{PMC}_{\mathcal{AM}^0(-d,+ne)} \subseteq \mathbf{P}$, let $X \in \mathbf{PMC}_{\mathcal{AM}^0(-d,+ne)}$ and let $\{\Pi(n) \mid n \in \mathbb{N}\}$ be a family of P systems from $\mathcal{AM}^0(-d,+ne)$ solving X in polynomial time and uniform way. Let (cod, s) be a polynomial encoding associated with that solution. Let us see that there exists a polynomial time reduction from X to the REACHABILITY problem.[1] For that, let us consider the mapping F from I_X to the set of instances $I_{\mathtt{REACHABILITY}}$ defined as follows: $F(u) = (G_{\Pi(s(u))+cod(u)}, s_{\Pi(s(u))+cod(u)}, (yes, env))$, for each instance u of X. Then, F is a polynomial time computable function such that

$$u \in L_X \Longleftrightarrow F(u) \in L_{\mathtt{REACHABILITY}}.$$

Finally, we deduce that $X \in \mathbf{P}$ because the class $\mathbf{P}$ is closed under polynomial-time reductions, $X \leq^p$ REACHABILITY and REACHABILITY $\in \mathbf{P}$. □

Hence, in the framework of polarizationless P systems with active membranes which do not make use of dissolution rules we have a *partial affirmative* answer to Păun's conjecture, that is, $\mathbf{PMC}_{\mathcal{AM}^0(-d,+ne)} = \mathbf{P}$. The answer is partial because dissolution rules have been forbidden.

In [1], a uniform and linear time solution to the QBF-SAT problem, a well known **PSPACE**-complete problem [15], by means of a family of recognizer polarizationless P systems from $\mathcal{AM}^0(+d,+ne)$, was given. Thus, $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\mathcal{AM}^0(+d,+ne)}$. Hence, assuming that $\mathbf{P} \neq \mathbf{NP}$, we have a *partial negative* answer to Păun's conjecture in the framework $\mathcal{AM}^0(-d,+ne)$: computationally hard problems can be efficiently solved avoiding polarizations. The answer is partial because division rules for non-elementary membranes have been required.

Dependency graph as a proof technique for negative results in Membrane Computing

Let $\mathcal{R}$ be a class of recognizer membrane systems such that every system from $\mathcal{R}$ is associated with a dependency graph verifying the following property: a computation of a system from $\mathcal{R}$ is an accepting computation if and only if there exists a path between two distinguished nodes, in the dependency graph associated with the system. In this situation, it is possible to show that some decision problem $X = (I_X, \theta_X)$ *cannot be solved* in polynomial time and in a uniform way by means of a single membrane system, free of resources, from $\mathcal{R}$. This remark is illustrated by an example.

The ONLY-ONE-OBJECT problem is the decision problem $X = (I_X, \theta_X)$ defined as follows: $I_X = \{a^n \mid n \in \mathbb{N}, n \geq 1\}$ and $\theta_X(a^n) = 1$ if and only if $n = 1$. It is easy to design a deterministic Turing machine which takes two computation steps, solving the ONLY-ONE-OBJECT problem. Let us see that ONLY-ONE-OBJECT $\notin \mathbf{PMC}^{1f}_{\mathcal{AM}^0(-d,+ne)}$.

Theorem 2. *There is not a recognizer membrane system Π from the class $\mathcal{AM}^0(-d,+ne)$ solving the ONLY-ONE-OBJECT problem in polynomial time by a single membrane system and free of resources.*

[1] The REACHABILITY or STCON problem is the following decision problem: *given a directed graph $G = (V, E)$ with two specified vertices s and t, determine whether or not there is a path from s to t*. There are algorithms solving this problem, for instance, search algorithms based on breadth-first search or depth-first search. These algorithms determine whether two vertices are connected in $O(\max(|V|, |E|))$ time. Moreover, they basically need to store at most $|V|$ items, so these algorithms use $O(|V|)$ space. But this quantity of space can be reduced to $O(\log^2|V|)$ by using an algorithm that could be called middle-first search (see [15] for details, pp. 149–150). In particular, REACHABILITY $\in \mathbf{P}$.

Proof. Let us assume that there exists a recognizer membrane system Π from $\mathcal{AM}^0(-d,+ne)$ verifying the following: (a) the input alphabet of Π is the singleton $\{a\}$; (b) every computation of Π with input multiset $\{a\}$ is an accepting computation; and (c) every computation of Π with input multiset $\{a^n\}$, for each $n>1$, is a rejecting computation.

Let us denote by $G_{\Pi+\{a\}}$ (respectively, $G_{\Pi+\{a^n\}}$, for each $n>1$) the dependency graph associated with the system $\Pi+\{a\}$ (resp. $\Pi+\{a^n\}$). Then, for each $n>1$, we have $G_{\Pi+\{a\}}=G_{\Pi+\{a^n\}}$. Besides, every computation of $\Pi+\{a\}$ is an accepting computation if and only if every computation of $\Pi+\{a^n\}$, for each $n>1$, is an accepting computation. This is a contradiction. □

3.2. Simulation technique

Let us define the meaning of *efficient simulation* in the framework of recognizer membrane systems.

Definition 4. Let Π and Π' be two recognizer membrane systems. We say that system Π' *simulates* system 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 $\mathbf{Comp}(\Pi)$ of computations of Π onto the set $\mathbf{Comp}(\Pi')$ of computations of Π' such that:

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

We say that a configuration $\mathcal{C}_t$ of a membrane system is the current state of the system; that is, the structure and the contents of the cells at the moment t. A computation $\mathcal{C}=(\mathcal{C}_0,\mathcal{C}_1,\ldots,\mathcal{C}_r)$ is a sequence of configurations such that: (1) $\mathcal{C}_0$ is the initial configuration; and (2) a configuration $\mathcal{C}_t$ yields $\mathcal{C}_{t+1}$ if the second can be obtained from the first by applying a set of rules according to the semantics of the system, and it is denoted by $\mathcal{C}_t\Rightarrow_\Pi\mathcal{C}_{t+1}$. We denote by $\mathcal{C}_t(i)$ the contents of the cells labelled by i in the configuration $\mathcal{C}_t$.

Next, the simulation technique by considering the presumed efficiency of the computing model of recognizer membrane systems $\mathcal{TDC}(2)$ is presented as an illustration of it.

In what follows, throughout this section, let $\mathbf{\Pi}=\{\Pi(n)\mid n\in\mathbb{N}\}$ be a family of recognizer tissue P systems with cell division and with environment solving a decision problem $X=(I_X,\theta_X)$ in polynomial time and uniform way. Let $r(n)$ be a polynomial function such that for each instance $u\in I_X$, $2^{r(|u|)}$ is an upper bound of the number of objects from $\mathcal{E}$ which are moved from the environment to all cells of the system by any computation of $\Pi(s(u))+cod(u)$.

Definition 5. For each $n\in\mathbb{N}$, let $\Pi(n)=(\Gamma,\mathcal{E},\Sigma,\mathcal{M}_1,\ldots,\mathcal{M}_q,\mathcal{R},i_{in},i_{out})$ be an element of the family $\mathbf{\Pi}$ of degree q, and for the sake of simplicity we denote r instead of $r(n)$. Let us consider the recognizer tissue P system of degree $q_1=1+q\cdot(r+2)+|\mathcal{E}|$ with cell division and without environment

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

defined as follows:

- $\Gamma'=\Gamma\ \cup\ \{\alpha_i:\ 0\leq i\leq r-1\}$.
- $\Sigma'=\Sigma$.
- Each cell $i\in\{1,\ldots,q\}$ of Π provides a cell of $\mathbf{S}(\Pi(n))$ with the same label. In addition, $\mathbf{S}(\Pi(n))$ has:
  - $r+1$ new cells, labelled by $(i,0),(i,1),\ldots,(i,r)$, respectively, for each $i\in\{1,\ldots,q\}$.
  - A distinguished cell labelled by 0.
  - A new cell, labelled by l_b, for each $b\in\mathcal{E}$.
- Initial multisets: $\mathcal{M}'_{l_b}=\{\alpha_0\}$, for each $b\in\mathcal{E}$, and

$$\left.\begin{array}{rcl}\mathcal{M}'_{(i,0)} &=& \mathcal{M}_i\\ \mathcal{M}'_{(i,1)} &=& \emptyset\\ \cdots\cdots & & \cdots\\ \mathcal{M}'_{(i,r)} &=& \emptyset\\ \mathcal{M}'_i &=& \emptyset\end{array}\right\}(1\leq i\leq q)$$

- Set of rules:

$$\begin{aligned}\mathcal{R}' = \mathcal{R} \cup\ & \{[\alpha_j]_{l_b} \to [\alpha_{j+1}]_{l_b}\ [\alpha_{j+1}]_{l_b} : b \in \mathcal{E} \wedge 0 \le j \le r-2\} \\ \cup\ & \{[\alpha_{p-1}]_{l_b} \to [b]_{l_b}\ [b]_{l_b} : b \in \mathcal{E}\} \\ \cup\ & \{(l_b,\ b/\lambda\ , 0) : b \in \mathcal{E}\} \\ \cup\ & \{\big((i,j),\ a/\lambda\ , (i,j+1)\big) : a \in \Gamma \wedge 1 \le i \le q \wedge 0 \le j \le p-1\} \\ \cup\ & \{\big((i,r),\ a/\lambda\ , i\big) : a \in \Gamma \wedge 1 \le i \le q\}\end{aligned}$$

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

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

⋆ $\Gamma \subseteq \Gamma'$, $\Sigma \subseteq \Sigma'$ and $\mathcal{E} = \emptyset$.
⋆ Each cell in Π is also a cell in $\mathbf{S}(\Pi(n))$.
⋆ There is a distinguished cell in $\mathbf{S}(\Pi(n))$ labelled by 0 which plays the role of environment of $\Pi(n)$.
⋆ $\mathcal{R} \subseteq \mathcal{R}'$, and now 0 is the label of a "normal cell" in $\mathbf{S}(\Pi(n))$.

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

Lemma 1. *Let $\mathcal{C}' = (\mathcal{C}'_0, \mathcal{C}'_1, \dots)$ be a computation of $\mathbf{S}(\Pi(n))$. For each t $(1 \le t \le r)$ the following holds:*

- *$\mathcal{C}'_t(i) = \emptyset$, for $0 \le i \le q$.*
- *For each $1 \le i \le q$, and $0 \le j \le r$ 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 cells labelled by l_b whose content is:*

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

Proof. By induction on t.

Let us start with the basic case $t = 1$. The initial configuration of system $\mathbf{S}(\Pi(n))$ is the following:

- $\mathcal{C}'_0(i) = \emptyset$, for $0 \le i \le q$.
- For each $1 \le i \le q$ we have $\mathcal{C}'_0(i,0) = \mathcal{M}_i$, and $\mathcal{C}'_0(i,j) = \emptyset$, for $1 \le j \le r$.
- For each $b \in \mathcal{E}$, there exists only one cell labelled by l_b whose contents is $\{\alpha_0\}$.

At configuration $\mathcal{C}'_0$, only the following rules are applicable:

- $[\alpha_0]_{l_b} \to [\alpha_1]_{l_b}\ [\alpha_1]_{l_b}$, for each $b \in \mathcal{E}$.
- $\big((i,0),\ a/\lambda\ , (i,1)\big)$, for each $a \in supp(\mathcal{M}_i)$.

Thus,

- For each i $(1 \le i \le q)$ we have:

$$\begin{cases} \mathcal{C}'_1(i) & = \emptyset \\ \mathcal{C}'_1(0) & = \emptyset \\ \mathcal{C}'_1(i,0) & = \emptyset \\ \mathcal{C}'_1(i,1) & = \mathcal{M}_i \\ \mathcal{C}'_1(i,j) & = \emptyset, \text{ for } 2 \le j \le r \end{cases}$$

- For each $b \in \mathcal{E}$, there are 2 cells labelled by l_b whose content is $\{\alpha_1\}$.

Hence, the result holds for $t = 1$.

By induction hypothesis, let t be such that $1 \le t < r$, and let us suppose the result holds for t, that is,

- $\mathcal{C}'_t(i) = \emptyset$, for $0 \leq i \leq q$.
- For each $1 \leq i \leq q$, and $0 \leq j \leq r$ 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 cells labelled by l_b whose contents is $\mathcal{C}'_t(l_b) = \{\alpha_t\}$ (because $t \leq r-1$).

Then, at configuration $\mathcal{C}'_t$ only the following rules are applicable:

(1) If $t \leq r-2$, 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 $t = r-1$, the rules $[\alpha_t]_{l_b} \to [b]_{l_b}\ [b]_{l_b}$, for each $b \in \mathcal{E}$.
(3) $\big((i,t),\ a/\lambda\ ,(i,t+1)\big)$, for each $a \in \Gamma$.

From the application of rules of type (1) or (2) at configuration $\mathcal{C}'_t$, we deduce that there are 2^{t+1} cells labelled by l_b in $\mathcal{C}'_{t+1}$, for each $b \in \mathcal{E}$, whose content is $\{\alpha_{t+1}\}$, if $t \leq r-2$, or $\{b\}$, if $t = r-1$.

From the application of rules of type (3) at configuration $\mathcal{C}'_t$, we deduce that

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

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

This completes the proof of this Lemma. □

Lemma 2. *Let $\mathcal{C}' = (\mathcal{C}'_0, \mathcal{C}'_1, \dots)$ be a computation of the tissue P system $\mathbf{S}(\Pi(n))$. Configuration $\mathcal{C}'_{r+1}$ is the following:*

(1) $\mathcal{C}'_{r+1}(0) = b_1^{2^r} \dots b_\alpha^{2^r}$, *where* $\mathcal{E} = \{b_1, \dots, b_\alpha\}$.
(2) $\mathcal{C}'_{r+1}(i) = \mathcal{M}_i = C_0(i)$, *for* $1 \leq i \leq q$.
(3) $\mathcal{C}'_{r+1}(i, j) = \emptyset$, *for* $1 \leq i \leq q$, $0 \leq j \leq r$.
(4) *There exist* 2^r *cells labelled by* l_b *whose content is empty, for* $b \in \mathcal{E}$.

Proof. From Lemma 1, the configuration $\mathcal{C}'_r$ is the following:

- $\mathcal{C}'_r(i) = \emptyset$, for $0 \leq i \leq q$.
- For each i $(1 \leq i \leq q)$ we have

$$\mathcal{C}'_r(i, j) = \begin{cases} \mathcal{M}_i, & \text{if} \quad j = r \\ \emptyset, & \text{if} \quad j \neq r \end{cases}$$

- For each $b \in \mathcal{E}$, there exist 2^r cells labelled by l_b whose content is $\{b\}$.

At configuration $\mathcal{C}'_r$ only the following rules are applicable:

- $\big((i,r),\ a/\lambda\ ,i\big)$, for each $a \in \Gamma \cap supp(\mathcal{M}_i)$.
- $\big(l_b,\ b/\lambda\ ,0\big)$, for each $b \in \mathcal{E}$.

Thus,

- $\mathcal{C}'_{r+1}(0) = b_1^{2^r} \dots b_\alpha^{2^r}$, where $\mathcal{E} = \{b_1, \dots, b_\alpha\}$.
- $\mathcal{C}'_{r+1}(i) = \mathcal{M}_i = C_0(i)$, for $1 \leq i \leq q$.
- $\mathcal{C}'_{r+1}(i, j) = \emptyset$, for $1 \leq i \leq q$ and $0 \leq j \leq r$.
- There exist 2^r cells labelled by l_b whose content is empty, for each $b \in \mathcal{E}$. □

Definition 6. Let $\mathcal{C} = (\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_m)$ be a halting computation of $\Pi(n)$. Then we define the computation $\mathbf{S}(\mathcal{C}) = (\mathcal{C}'_0, \mathcal{C}'_1, \dots, \mathcal{C}'_r, \mathcal{C}'_{r+1}, \dots, \mathcal{C}'_{r+1+m})$ of $\mathbf{S}(\Pi(n))$ as follows:

(1) The initial configuration is:

$$\begin{cases} \mathcal{C}'_0(i) &= \emptyset, \text{ for } 0 \leq i \leq q \\ \mathcal{C}'_0(i,0) &= C_0(i), \text{ for } 1 \leq i \leq q \\ \mathcal{C}'_0(i,j) &= \emptyset, \text{ for } 1 \leq i \leq q \text{ and } 1 \leq j \leq r \\ \mathcal{C}'_0(l_b) &= \alpha_0, \text{ for each } b \in \mathcal{E} \end{cases}$$

(2) The configuration $\mathcal{C}'_t$, for $1 \leq t \leq r$, is described by Lemma 1.
(3) The configuration $\mathcal{C}'_{r+1}$ is described by Lemma 2.
(4) The configuration $\mathcal{C}'_{r+1+s}$, for $0 \leq s \leq m$, coincides with the configuration $\mathcal{C}_s$ of Π, that is, $\mathcal{C}_s(i) = \mathcal{C}'_{r+1+s}(i)$, for $1 \leq i \leq q$. The content of the remaining cells (excluding cell 0) at configuration $\mathcal{C}'_{r+1+s}$ is equal to the content of that cell at configuration $\mathcal{C}'_{r+1}$, that is, these cells do not evolve after step $r+1$.

That is, every computation $\mathcal{C}$ of $\Pi(n)$ can be "reproduced" by a computation $\mathbf{S}(\mathcal{C})$ of $\mathbf{S}(\Pi(n))$ with a delay: from step $r+1$ to step $r+1+m$, the computation $\mathbf{S}(\mathcal{C})$ restricted to cells $1,\ldots,q$ provides the computation $\mathcal{C}$ of $\Pi(n)$.

From Lemma 1 and Lemma 2 we deduce the following:

(a) $\mathbf{S}(\mathcal{C})$ is a computation of $\mathbf{S}(\Pi(n))$.
(b) S is an injective function from $\mathbf{Comp}(\Pi(n))$ onto $\mathbf{Comp}(S(\Pi(n)))$.

Moreover, if r is a polynomial function on the size of $\Pi(n)$, then we have the following result.

Proposition 1. *The tissue P system $\mathbf{S}(\Pi(n))$ from Definition 5 simulates $\Pi(n)$ in an efficient way.*

Proof. In order to show that $\mathbf{S}(\Pi(n))$ 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 construct $\mathbf{S}(\Pi(n))$ from $\Pi(n)$ is polynomial in the size of the initial resources of $\Pi(n)$. Indeed,

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

From Lemma 1 and Lemma 2 we deduce that:

(a) Every computation $\mathcal{C}'$ of $\mathbf{S}(\Pi(n))$ has an associated computation $\mathcal{C}$ of $\Pi(n)$ such that $\mathbf{S}(\mathcal{C}) = \mathcal{C}'$ in a natural way.
(b) The function $\mathbf{S}$ is injective.
(c) A computation $\mathcal{C}$ of Π is an accepting computation if and only if $\mathbf{S}(\mathcal{C})$ is an accepting computation of $\mathbf{S}(\Pi(n))$.

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

Taking into account these results, we analyze the role of the environment in the efficiency of tissue P systems with cell division. That is, we study the ability of these P systems with respect to the computational efficiency when the alphabet of the environment is an empty set.

Theorem 3. *For each $k \in \mathbb{N}$ we have $\mathbf{PMC}_{\mathbf{TDC}(k+1)} = \mathbf{PMC}_{\widehat{\mathbf{TDC}}(k+1)}$.*

Proof. Let us recall that $\mathbf{PMC}_{\mathbf{TDC}(1)} = \mathbf{P}$ (see [6] for details). Then,

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

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

Let $\{\Pi(n) : n \in N\}$ be a family of tissue P systems from $\mathbf{TDC}(k+1)$ a polynomial time and uniform solution to X. Let (cod, s) be a polynomial 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, let $r(n)$ be a polynomial function such that $2^{r(|u|)}$ is an upper bound of the number of objects from $\mathcal{E}$ which are moved from the environment to all cells of the system by any computation of $\Pi(s(u)) + cod(u) = (\Gamma, \mathcal{E}, \Sigma, \mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} + cod(u), \ldots, \mathcal{M}_{q_1}, \mathcal{R}, i_{in}, i_{out})$.

Then, we consider the tissue P system without environment

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

according to Definition 5, where $q_1 = 1 + q \cdot (r(|u|) + 2) + |\mathcal{E}|$.

Therefore, $\mathbf{S}(\Pi(s(u))) + cod(u)$ is a tissue P system from $\widehat{TDC}(k+1)$ such that verifies the following:

- A distinguished cell labelled by 0 has been considered, which will play the role of the environment in the system $\Pi(s(u)) + cod(u)$.
- At the initial configuration, it has enough objects in cell 0 in order to simulate the behaviour of the environment of the system $\Pi(s(u))) + cod(u)$.
- After $r(n) + 1$ steps, computations of $\Pi(s(u)) + cod(u)$ are reproduced by the computations of $\mathbf{S}(\Pi(s(u))) + cod(u)$ exactly.

Let us suppose that $\mathcal{E} = \{b_1, \ldots, b_\alpha\}$. In order to simulate $\Pi(s(u)) + cod(u)$ by a tissue P system without environment in an efficient way, we need to have enough objects in the cell of $\mathbf{S}(\Pi(s(u))) + cod(u)$ labelled by 0 available. That is, $2^{r(n)}$ objects in that cell 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 $2^{r(n)}$ copies of each object $b_j \in \mathcal{E}$ in the cell of $\mathbf{S}(\Pi(s(u))) + cod(u)$ labelled by 0. For this purpose

- For each $b \in \mathcal{E}$ we consider a cell in $\mathbf{S}(\Pi(s(u))) + cod(u)$ labelled by l_b 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 $0 \leq j \leq r(|u|) - 2$,
  - $[\alpha_{p(n)-1}]_{l_b} \rightarrow [b]_{l_b}\ [b]_{l_b}$,
  - $(l_b, b/\lambda, 0)$.
- By applying the previous rules, after $r(|u|)$ transition steps we get $2^{r(|u|)}$ cells 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 $2^{r(|u|)}$ copies of objects b in cell 0, for each $b \in \mathcal{E}$.

Therefore, after the execution of $r(|u|) + 1$ transition steps in each computation of $\mathbf{S}(\Pi(s(u))) + cod(u)$ in cell 0 of the corresponding configuration, we have $2^{r(|u|)}$ 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 $\mathbf{S}(\Pi(s(u)) + cod(u))$.

From Proposition 1 we deduce that the family $\{\mathbf{S}(\Pi(n))|\ n \in N\}$ solves X in polynomial time and uniform way. Hence, $X \in \mathbf{PMC}_{\widehat{TDC}(k+1)}$. □

3.3. Algorithmic technique

In order to simulate the behaviour of a recognizer membrane system, Π, when an input multiset, m, is supplied to the corresponding input membrane, the algorithmic technique can be used. This technique consists on the construction of a deterministic decision algorithm $\mathcal{A}$ working in polynomial time that receives as input a recognizer membrane system Π and an input multiset m of Π. Then, the algorithm $\mathcal{A}$ reproduces the behaviour of one computation of $\Pi + m$. Since the recognizer membrane system $\Pi + m$ is confluent, then the algorithm will provide the same answer of 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). In [14], another application of the algorithmic technique to prove the non-efficiency of certain kind of P systems is found.

The goal of this section is the use of the algorithmic technique to show that only tractable problems can be solved efficiently by using tissue P systems with communication rules of any length, separation rules and without environment. That is, we will prove that $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathbf{TSC}}}$.

For this purpose, given a family of recognizer tissue P systems, we provide a deterministic algorithm $\mathcal{A}$ working in polynomial time that receives as input a recognizer tissue P system from $\widehat{\mathbf{TSC}}$ together with an input multiset, and reproduces the behaviour of a computation of such system.

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

```
Input: A recognizer tissue P system Π from TSC^ and an input multiset m
    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 tissue 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 labelled 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. The output of this stage 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 labelled 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 r ≡ (i, u/v, j) ∈ R_C according to the order chosen do
        for each pair of cells (i, σ_i), (j, σ_j) of C'_t according to the
            lexicographical order do
            n_r ← maximum number of times that r is applicable to (i, σ_i), (j, σ_j)
            if n_r > 0 then
                C'_t ← C'_t \ n_r · LHS(r, (i, σ_i), (j, σ_j))
                A ← A ∪ {(r, n_r, (i, σ_i), (j, σ_j))}
                B ← B ∪ {(i, σ_i), (j, σ_j)}
            end if
        end for
    end for
    for r ≡ [a]_i → [Γ_0]_i[Γ_1]_i ∈ R_S according to the order chosen do
        for each (a, i, σ_i) ∈ C'_t, according to the lexicographical order, and
            such that (i, σ_i) ∉ B do
            C'_t ← C'_t \ {(a, i, σ_i)}
            A ← A ∪ {(r, 1, (i, σ_i))}
            B ← B ∪ {(i, σ_i)}
        end for
end for
```

This algorithm is deterministic and works in polynomial time. Indeed, the cost in time of the previous algorithm is polynomial in the size of Π because the number of cycles of the first main loop **for** is of order $O(|R| \cdot \frac{(2M+q)(2M+q-1)}{2})$, and the number of cycles of the second main loop **for** is of order $O(|R| \cdot |\Gamma| \cdot (2M+q))$. Besides, the last loop includes a membership test of order $O(2M+q)$.

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 right-hand side of the rules.

Execution stage.

```
Input: The output C'_t and A of the selection stage
    for each (r, n_r, (i, σ_i), (j, σ_j)) ∈ A do
        C'_t ← C'_t + n_r · RHS(r, (i, σ_i), (j, σ_j))
    end for
    for each (r, 1, (i, σ_i)) ∈ A do
        C'_t ← C'_t + {(λ, i, σ_i)/σ_i 0}
        C'_t ← C'_t + {(λ, i, σ_i 1)}
        for each (x, i, σ_i) ∈ C'_t according to the lexicographical order do
            if x ∈ Γ_0 then
                C'_t ← C'_t + {(x, i, σ_i)/σ_i 0}
            else
                C'_t ← C'_t + {(x, i, σ_i)/σ_i 1}
            end if
        end for
    end for
    C_{t+1} ← C'_t
```

This algorithm is deterministic and works in polynomial time. Indeed, the cost in time of the previous algorithm is polynomial in the size of Π because the number of cycles of the first main loop **for** is of order $O(|R|)$, and the number of cycles of the second main loop **for** is of order $O(|R| \cdot |\Gamma| \cdot (2M+q))$. Besides, inside the body of the last loop there is a membership test of order $O(|\Gamma|)$.

Throughout this algorithm we have deterministically simulated a computation of Π in such a manner that the answer of the algorithm is affirmative if and only if the simulated computation is accepting.

Theorem 4. $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathbf{TSC}}}$.

Proof. It suffices to prove that $\mathbf{PMC}_{\widehat{\mathbf{TSC}}} \subseteq \mathbf{P}$. Let $k \in \mathbb{N}$ such that $X \in \mathbf{PMC}_{\widehat{\mathbf{TSC}}(k)}$ and let $\{\Pi(n) : n \in \mathbb{N}\}$ be a family of tissue P systems from $\widehat{\mathbf{TSC}}(k)$ solving X according to Definition 1. Let (cod, s) be a polynomial encoding associated with that solution. If $u \in I_X$ is an instance of the problem X, then u will be processed by the system $\Pi(s(u)) + cod(u)$.

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

Input: an instance u of the problem X.

Construct the system $\Pi(s(u)) + cod(u)$.
Run algorithm $\mathcal{A}$ with input $\Pi(s(u)) + cod(u)$.

Output: *Yes* if $\Pi(s(u)) + cod(u)$ has an accepting computation, *No* otherwise

The algorithm $\mathcal{A}'$ receives as input an instance u of the decision problem $X = (I_X, \theta_X)$ and works in polynomial time. 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*.

Hence, $X \in \mathbf{P}$. □

4. Conclusions

A survey on proof techniques to demonstrate, on the one hand the completeness of classes of membrane systems, and on the other hand the presumed efficiency or non-efficiency of classes of membrane systems, is presented. The refinement of these techniques has supported the demonstration of several new frontiers of efficiency. Apart from these techniques, other techniques have been used both in demonstrating the computational power [5] and the computational efficiency [10,28] can be found in the literature. However, due to the restriction of length of the work it would be impossible to include them all.

Let M_1 be a non-efficient model of computation and M_2 a presumably efficient model of computation. Passing from M_1 to M_2 is equivalent to passing from the non-efficiency to presumed efficiency. If we have a solution to an **NP**-complete problem in M_2, passing that solution to M_1 would directly imply $\mathbf{P} = \mathbf{NP}$. Finding thinner frontiers provides easier ways to demonstrate that statement.

The development of new techniques, or the refinement of existing ones, for demonstrating the non-efficiency of classes of membrane systems seems crucial to solve some open problems in the field of computational complexity theory in Membrane Computing, as the Păun's conjecture.

Another interesting research line is to keep finding efficient solutions to **NP**-complete (or even **PSPACE**-complete) problems with families of membrane systems that could be implemented in highly-parallel platforms that could compete with current problem-solvers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported in part by the research project TIN2017-89842-P, cofinanced by Ministerio de Economía, Industria y Competitividad (MINECO) of Spain, through the Agencia Estatal de Investigación (AEI), and by Fondo Europeo de Desarrollo Regional (FEDER) of the European Union.

References

[1] A. Alhazov, M.J. Pérez-Jiménez, Uniform solution to QSAT using polarizationless active membranes, in: J. Durand-Lose, M. Margenstern (Eds.), Proceedings of the 5th International Conference: Machines, Computations, and Universality, MCU 2007, Orléans, France, September 10–13, 2007, in: Lecture Notes in Computer Science, vol. 4664, 2007, pp. 122–133.
[2] K. Buño, H. Adorna, Distributed computation of a k P systems with active membranes for SAT using clause completion, J. Membr. Comput. 2 (2020) 108–120.
[3] A. Cordón-Franco, M.A. Gutiérrez, M.J. Pérez-Jiménez, A. Riscos-Núñez, Weak metrics on configurations of a P system, in: Gh. Păun, A. Riscos, Á. Romero, F. Sancho (Eds.), Proceedings of the Second Brainstorming Week on Membrane Computing, 2004, pp. 139–151, Report RGNC 01/2004.
[4] R. Freund, Asynchronous P systems and P systems working in the sequential mode, in: G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, WMC 2004, in: Lecture Notes in Computer Science, vol. 3365, Springer, Berlin, Heidelberg, 2005, pp. 36–62.
[5] R. Freund, How derivation modes and halting conditions may influence the computational power of P systems, J. Membr. Comput. 2 (2020) 14–25.
[6] R. Gutiérrez-Escudero, M.J. Pérez-Jiménez, M. Rius-Font, Characterizing tractability by tissue-like P systems, in: International Conference on Membrane Computing, in: Lecture Notes in Computer Science, vol. 5957, 2010, pp. 289–300.
[7] 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, in: R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, 6th International Workshop, WMC 2005, Vienna, Austria, July 18–21, in: Lecture Notes in Computer Science, vol. 3850, 2006, pp. 224–240, Revised Selected and Invited Papers.
[8] Y. Jiang, Y. Su, F. Luo, An improved universal spiking neural P system with generalized use of rules, J. Membr. Comput. 1 (2019) 270–278.
[9] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Characterizing PSPACE with shallow non-confluent P systems, J. Membr. Comput. 1 (2019) 75–84.
[10] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Shallow laconic P systems can count, J. Membr. Comput. 2 (2020) 49–58.
[11] L.F. Macías-Ramos, M.J. Pérez-Jiménez, A. Riscos-Núñez, L. Valencia-Cabrera, Membrane fission versus cell division: when membrane proliferation is not enough, Theor. Comput. Sci. 608 (2015) 57–65.

[12] G. Mauri, A. Leporati, A.E. Porreca, C. Zandron, Recent complexity-theoretic results on P systems with active membranes, J. Log. Comput. 25 (2015) 1047–1071.
[13] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, M.J. Pérez-Jiménez, Minimal cooperation as a way to achieve the efficiency in cell-like membrane systems, J. Membr. Comput. 1 (2019) 85–92.
[14] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, M.J. Pérez-Jiménez, P systems with proteins: a new frontier when membrane division disappears, J. Membr. Comput. 1 (2019).
[15] C.H. Papadimitriou, Computational Complexity, Addison–Wesley, Massachusetts, 1995.
[16] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), The Oxford Handbook of Membrane Computing, Oxford University Press, Oxford, 2010.
[17] Gh. Păun, Computing with membranes, J. Comput. Syst. Sci. 61 (1) (2000) 108–143.
[18] Gh. Păun, P systems with active membranes: attacking NP-complete problems, J. Autom. Lang. Comb. 6 (1) (2001) 75–90. A preliminary version in Centre for Discrete Mathematics and Theoretical Computer Science, CDMTCS Research Report Series-102, May 1999, 16 pages.
[19] Gh. Păun, Computing with Membranes (P Systems): Twenty Six Research Topics, Centre for Discrete Mathematics and Theoretical Computer Science, CDMTCS Research Report Series, vol. 119, February 2000, 16 pages.
[20] Gh. Păun, Further twenty six open problems in membrane computing, in: M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan (Eds.), Proceedings of the Third Brainstorming Week on Membrane Computing, Fénix Editora, Sevilla, 2005, pp. 249–262.
[21] M.J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini, Decision P systems and the P $\neq$ NP conjecture, in: Gh. Păun, Gr. Rozenberg, A. Salomaa, C. Zandron (Eds.), Membrane Computing 2002, in: Lecture Notes in Computer Science, vol. 2597, 2003, pp. 388–399. A preliminary version in Gh. Păun, C. Zandron (eds.) Pre-proceedings of Workshop on Membrane Computing 2002, MolCoNet project-IST-2001-32008, Publication No. 1, Curtea de Arges, Romanian, August 19–23, 2002, pp. 345–354.
[22] M.J. Pérez-Jiménez, Á. Romero-Jiménez, Simulating Turing machines by P systems with external output, Fundam. Inform. 49 (2002) 1–3, 273–287, Annales Societatis Mathematicae Polonae, Series IV, IOS Press, Amsterdam.
[23] M.J. Pérez-Jiménez, Á. Romero-Jiménez, Generation of diophantine sets by computing P systems with external output, in: C. Calude, M.J. Dinneen, F. Peper (Eds.), Unconventional Models of Computation, in: Lecture Notes in Computer Science, vol. 2509, 2002, pp. 176–190.
[24] M.J. Pérez-Jiménez, Á. Romero-Jiménez, Computing partial recursive functions by transition P systems, in: C. Martín-Vide, Gh. Păun, Gr. Rozenberg, A. Salomaa (Eds.), Membrane Computing, in: Lecture Notes in Computer Science, vol. 2933, 2004, pp. 320–340. A preliminary version in A. Alhazov, C. Martín Vide, Gh. Păun (eds.), Pre-proceedings of the Workshop on Membrane Computing, WMC-2003, Tarragona, Spain, July 17–22, 2003, pp. 428–444.
[25] M.J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini, A polynomial complexity class in P systems using membrane division, J. Autom. Lang. Comb. 11 (2006) 423–434. A preliminary version in E. Csuhaj-Varjú, C. Kintala, D. Wotschke, Gy. Vaszil (eds.), Proceedings of the Fifth International Workshop on Descriptional Complexity of Formal Systems, DCFS 2003, Budapest, Hungary, July 12–14, 2003, pp. 284–294.
[26] Á. Romero-Jiménez, D. Orellana-Martín, Design patterns for efficient solutions to NP-complete problems in membrane computing, in: C. Graciani, A. Riscos-Núñez, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Enjoying Natural Computing, in: Lecture Notes in Computer Science, vol. 11270, Springer, Cham, 2018, pp. 237–255.
[27] P. Sosík, M. Langer, Small (purely) catalytic P systems simulating register machines, Theor. Comput. Sci. 623 (2016) 65–74.
[28] C. Zandron, Bounding the space in P systems with active membranes, J. Membr. Comput. 2 (2020) 137–145.