
A solution to the only one object problem with dissolution rules

Julien Caselmann¹, David Orellana-Martín^{2,3}

¹Humboldt-Universität zu Berlin

E-mail: caselmaj@informatik.hu-berlin.de

²Research Group on Natural Computing,

Department of Computer Science and Artificial Intelligence,

Universidad de Sevilla

Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

E-mail: dorellana@us.es

³SCORE Laboratory, I3US, Universidad de Sevilla

Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Summary. In membrane computing, it is usual to obtain solutions to decision problems by means of (non-)uniform families of membrane systems, where each P system of the family can solve one or more than one instance of the problem. In this work, a new solution to the **ONLY-ONE-OBJECT** problem is provided by means of a single membrane system, that is capable of solving each instance of the problem.

Key words: Membrane Computing, computational complexity theory, only one object problem.

1 Introduction

Membrane Computing is a relatively young compared to other areas of computer science research and therefore still presents us with some open questions. Some computational models inspired by biological cells have already been very successful. The ideas of this computational framework are materialized into non-deterministic, parallel and distributed models of computation called membrane systems (or P systems). Certain membrane systems have achieved surprising results, ranging from a simple computation of a square number [1] to an *ad-hoc* solution for the **SAT** problem [2]. It is still not entirely clear what exactly these membrane systems are capable of and what is beyond their reach. In this sense, several works concerning lower and upper bounds of different classes of membrane systems have been published [3, 4]. . Numerous questions arise from that, including the famous 26 problems of Păun [5]. **PMC \mathcal{R}** is the class of problems solvable in polynomial time by membrane systems of the class \mathcal{R} . In this area, a new way to tackle the **P**

versus NP problem appears in the form of efficiency frontiers. As **P versus NP** is one of the most exciting problems in complexity theory in computer science, it is of utmost interest to find out what the connections between the classes **PMC** and **P**, **NP** are.

In [6, 7, 8, 9], a new methodology for solving decision problems or demonstrating the non-solvability of a problem by means of a single membrane system is presented. On the one hand, a solution to the **PARITY** problem is given by means of a single membrane system using transition P systems using dissolution rules and only two objects in the left-hand side of evolution rules. On the other hand, it is demonstrated that the **ONLY-ONE-OBJECT** problem cannot be solved by means of a single P system with active membranes without polarizations and without dissolution rules.

The rest of the work is structured as follows. In the following sections, we present some preliminaries about languages and set theory and we recall the model of recognizer polarizationless P systems with active membranes with dissolution rules. In Section 4, we briefly recall the concept of complexity classes with single recognizer membrane systems. Section 5 is devoted to cite some of the main uses of the dependency graph technique. Next, we propose a solution to the **ONLY-ONE-OBJECT** by means of a single recognizer membrane system from $\mathcal{NAM}^0(+d, -ne)$. We finish the paper with some conclusions and interesting open research lines.

2 Preliminaries

In this section, we introduce some basic concepts and recognizer polarizationless P systems with active membranes with dissolution rules. For deeper questions concerning formal languages and membrane systems, we refer the reader to [10, 11].

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

A *multiset* over an alphabet Γ is an ordered pair (Γ, f) where f is a mapping from Γ onto the set of natural numbers \mathbb{N} . The *support* of a multiset $m = (\Gamma, f)$ is defined as $\text{supp}(m) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite (respectively, empty) if its support is a finite (respectively, empty) set. We denote by \emptyset the empty multiset. Let $m_1 = (\Gamma, f_1)$, $m_2 = (\Gamma, f_2)$ be multisets over Γ , then the union of m_1 and m_2 , denoted by $m_1 + m_2$, is the multiset (Γ, g) , where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$. We denote by $M(\Gamma)$ the set of all multisets over Γ .

3 Polarizationless P systems with active membranes

First presented in [1], a P system is a computational model inspired by a biological cell.

Definition 1. *A polarizationless P system with active membranes without division rules of degree $q \geq 1$ is a tuple*

$$\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}, i_{out})$$

where:

- Γ is a finite (working) alphabet whose elements are called objects.
- H is a finite alphabet such that $H \cap \Gamma = \emptyset$ whose elements are called labels.
- $q \geq 1$ is the degree of the system.
- μ is a labelled rooted tree (called membrane structure) consisting of q nodes injectively labelled by elements of H (the root of μ is labelled by r_μ).
- $\mathcal{M}_1, \dots, \mathcal{M}_q$ are multisets over $\Gamma \setminus \Sigma$.
- \mathcal{R} is a finite set of rules, of the following forms:
 - (a) $[a \rightarrow u]_h$, where $h \in H$, $a \in \Gamma$, $u \in M(\Gamma)$, (object evolution rules).
 - (b) $a[]_h \rightarrow [c]_h$, where $h \in H \setminus \{r_\mu\}$, $a, b, c \in \Gamma$ (send-in communication rules).
 - (c) $[a]_h \rightarrow b[]_h$, where $h \in H$, $a, b \in \Gamma$ (send-out communication rules).
 - (d) $[a]_h \rightarrow b$, where $h \in H \setminus \{i_{out}, skin\}$, $a, b \in \Gamma$ (dissolution rules).
- $i_{out} \in H \cup \{env\}$.

A polarizationless P system with active membranes

$$\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}, i_{out})$$

can be viewed as a set of q membranes, labelled by elements of H , arranged in a hierarchical structure μ given by a rooted tree (called membrane structure) whose root is called the *skin membrane*, such that: (a) $\mathcal{M}_1, \dots, \mathcal{M}_q$ represent the finite multisets of *objects* initially placed in the q membranes of the system; (b) \mathcal{R} is a finite set of rules over Γ associated with the labels; and (c) $i_{out} \in H \cup \{env\}$ indicates the output region. We use the term *region i* to refer to membrane i in the case $i \in H$ and to refer to the “environment” of the system in the case $i = env$.

A membrane that does not have internal membranes (i.e. it is a leaf of the tree structure μ) is called an elementary membrane. Otherwise, it is considered a non-elementary membrane. The membrane that surrounds the whole system is called *skin* membrane.

An *instantaneous description* or a *configuration* \mathcal{C}_t at an instant t of a polarizationless P system with active membranes is described by the following elements: (a) the membrane structure at instant t , and (b) all multisets of objects over Γ associated with all the membranes present in the system at that moment. The

initial configuration of Π is described as $\mathcal{C}_0 = (\mu, \mathcal{M}_1, \dots, \mathcal{M}_q; \emptyset)$. We denote the contents of the region h in the moment t as $\mathcal{C}_t(h)$.

An object evolution rule $[a \rightarrow u]_h$ for $h \in H$, $a \in \Gamma$, $u \in M(\Gamma)$ is *applicable* to a configuration \mathcal{C}_t at an instant t , if there exists a membrane labelled by h in \mathcal{C}_t which contains object a . When applying such a rule, object a is consumed and objects from multiset u are produced in that membrane.

A send-in communication rule $[a]_\mu \rightarrow [b]_h$ for $h \in H$, $a, b \in \Gamma$ is *applicable* to a configuration \mathcal{C}_t at an instant t , if there exists a membrane labelled by h in \mathcal{C}_t such that h is not the label of the root of μ and its parent membrane contains object a . When applying such a rule, object a is consumed from the parent membrane and object b is produced in the corresponding membrane h .

A send-out communication rule $[a]_h \rightarrow [b]_\mu$ for $h \in H$, $a, b \in \Gamma$ is *applicable* to a configuration \mathcal{C}_t at an instant t , if there exists a membrane labelled by h in \mathcal{C}_t such that it contains object a . When applying such a rule, object a is consumed from such membrane h and object b is produced in the parent of such membrane.

A dissolution rule $[a]_h \rightarrow b$ for $h \in H \setminus \{i_{out}\}$, $a, b \in \Gamma$ is *applicable* to a configuration \mathcal{C}_t at an instant t , if there exists a membrane labelled by h in \mathcal{C}_t , different from the skin membrane and the output region, such that it contains object a . When applying such a rule, object a is consumed, membrane h is dissolved and its objects are sent to the parent (or the first ancestor that has not been dissolved).

A computational step is made by the application of the aforementioned rules in the following way:

- One object can fire only one rule;
- Object evolution rules can fire in a maximal parallel way; that is, all the evolution rules that can be fired will be fired;
- In each membrane, only one rule of the types (b), (c) or (d) can be fired in each computational step;
- A computational step is divided in two microsteps: First, all the transformations of objects are made, and second, membranes that must be dissolved will be dissolved.

If an object can fire more than one rule, then it will select one of them non-deterministically. \mathcal{C}_t leads to \mathcal{C}_{t+1} if the latter can be obtained from the former by applying the rules in the previously explained way, and it is denoted by $\mathcal{C}_t \Rightarrow_{\Pi} \mathcal{C}_{t+1}$. A computation of the system is a sequence of configurations $\mathcal{C} = (\mathcal{C}_0, \dots, \mathcal{C}_n)$, where \mathcal{C}_0 is the initial configuration of the membrane system, and for each \mathcal{C}_t , $t \geq 1$, $\mathcal{C}_{t-1} \Rightarrow_{\Pi} \mathcal{C}_t$. We say that the computation is finite if $n \in \mathbb{N}$.

In [12, 13], a special type of membrane systems is introduced in order to solve decision problems, the well-known recognizer membrane systems. A recognizer membrane system is a membrane system from any class of P systems (e.g., cell P systems, tissue P systems and so on) that has special requirements.

Definition 2. *A recognizer polarizationless P system with active membranes without division rules of degree $q \geq 1$ is a tuple*

$$(\Pi, \Sigma, i_{in})$$

where:

- $\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \dots, \mathcal{M}_q, \mathcal{R}, i_{out})$ is a polarizationless P system with active membranes without division rules of degree $q \geq 1$ where:
 - Γ has two special symbols, **yes** and **no**.
 - $\mathcal{M}_1, \dots, \mathcal{M}_q$ are multisets over $\Gamma \setminus \Sigma$.
 - $i_{out} = env$ is the environment of the system.
- $\Sigma \subsetneq \Gamma$.
- $i_{in} \in H$ is the input membrane.

Let $m \in M(\Sigma)$. We denote by $\Pi + m$ the membrane system Π with input m ; that is, the membrane system Π where the multiset m is introduced in the initial configuration in the input membrane i_{in} . Then, the initial configuration of $\Pi + m$ is $\mathcal{C}_0 = (\mu, \mathcal{M}_1, \dots, \mathcal{M}_{i_{in}} + m, \dots, \mathcal{M}_q; \emptyset)$. We recall the concept of solvability of decision problems by means of recognizer membrane systems:

Definition 3. Let $X = (I_X, \theta_X)$ be a decision problem, and let $cod : I_X \rightarrow M(\Sigma)$ be a function that transforms an instance of X to a multiset over Σ , that will be the input of Π . We say that Π solves an instance $u \in I_X$ of the decision problem X if:

- The system $\Pi + cod(u)$ sends only one object **yes** or one object **no**, but not both, to the environment, and only in the last step of the computation; and
- The system $\Pi + cod(u)$ is **confluent**; that is, all the computations halt and return the same result.

4 The complexity classes $\text{PMC}_{\mathcal{R}}$ and $\text{PMC}_{\mathcal{R}}^{1f}$

We recall the definition of various computational complexity classes in the framework of membrane computing.

Definition 4. Let \mathcal{R} be a class of recognizer membrane systems. We say that a family of recognizer membrane systems $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ from \mathcal{R} solves a decision problem $X = (I_X, \theta_X)$ in a uniform way if the following hold:

- There exists a pair of functions (cod, s) computable in polynomial time over I_X such that $cod(u) \in M(\Sigma)$ (input multiset) and $s(u) \in \mathbb{N}$ (size of the instance);
- For each $n \in \mathbb{N}$, $s^{-1}(n) \subseteq I_X$.
- $\mathbf{\Pi}$ is polynomially bounded, sound and complete with regard to (X, cod, s) .

In this way, for solving a certain instance $u \in I_X$, we need to know the answer of the membrane system $\Pi(s(u)) + cod(u)$. For more detail, we refer the reader to [12].

In [9, 6], authors introduce the complexity class $\text{PMC}_{\mathcal{R}}^{1f}$ as a manner to deal with decision problems with a single recognizer membrane system.

Definition 5. Let \mathcal{R} be a class of recognizer membrane systems. Let $X = (I_X, \theta_X)$ be a decision problem such that I_X is a language over a finite alphabet Σ_X . We say that problem X is solvable in polynomial time by a single membrane system Π from \mathcal{R} free of external resources, denoted by $X \in \mathbf{PMC}_{\mathcal{R}}^{1f}$, if the following hold:

- The input alphabet of Π is Σ_X .
- The system Π is polynomially bounded, sound and complete with regard to X

The term *free of resources* means that the input is directly introduced as a multiset from the instance, without “being encoded”.

The class $\mathbf{PMC}_{\mathcal{R}}^{1p}$ is also defined in the aforementioned paper, being defined in a similar way to $\mathbf{PMC}_{\mathcal{R}}^{1f}$, but in this case the encoding of the input instance is allowed.

From these asserts, it is trivial to see that $\mathbf{PMC}_{\mathcal{R}}^{1f} \subseteq \mathbf{PMC}_{\mathcal{R}}^{1p} \subseteq \mathbf{PMC}_{\mathcal{R}}$.

5 The Origins of the Dependency Graph

The non-deterministic nature of membrane systems let a membrane system have, instead of a single computation, a tree of computations, usually denoted by $Comp(\Pi)$. The dynamics of the system are captured by $Comp(\Pi)$, being the initial configuration the root node of the tree and there exists an edge between \mathcal{C} and \mathcal{C}' if and only if $\mathcal{C} \Rightarrow_{\Pi} \mathcal{C}'$. The configuration paths of maximum length up to a leaf are the computations of Π and a computation terminates if and only if the path is finite. From the definition of $\mathbf{PMC}_{\mathcal{R}}$ we know that the obtained membrane systems must be confluent. Therefore, it is sufficient to consider only one computation per problem case. An interesting question would consist on finding the computational path of minimum length, but for this one has to measure the degree of similarity between two configurations (e.g., by the distance metric [14]). In this context, the dependency graph was introduced, which represents the dependencies between the membrane states (configurations) and the set of rules of the P system.

6 The Dependency Graph as a Proof Technique

From that very first application, different applications that are not related to the first one appeared.

6.1 Non-Efficiency of specific Membrane Systems

Let us now consider polarizationless recognizer P systems with active membranes that do not use dissolution rules. The dependency graph G is a directed graph whose vertex set consists of the initial configuration and all membrane configurations (a, h) (= alphabet object and membrane label on which it appears), for which

a appears on the right or left side of a rule. If there is a rule leading from configuration 1 to configuration 2, then nodes 1 and 2 in G are connected by an edge. Here, they use the concept of “accessibility” in the graph. The objects initially placed in the system can be considered as the initial nodes (taking into account the corresponding membranes). Instead of simulating the whole system, the graph can be constructed from the definition of the recognizer membrane system and the question is transformed to the following one:

Is there a path from the initial nodes to the node (yes, env) ?

In [15], the well-known Păun’s conjecture was stated. Is $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(+d, -ne)}$. In [16], they use the concept to give a partial affirmative answer for the case where dissolution is forbidden and division rules both for elementary and non-elementary membranes. In that work, it is shown that $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(-d, +ne)}$.

As stated above, we need to find a path from the initial nodes to the node (yes, env) . But for this purpose, we reduce the entire problem to the **REACHABILITY** problem, that is stated as follows:

Given a directed graph $G = (V, E)$ and two vertices $s, t \in V$. Is there a path from s to t ?

It is known that **REACHABILITY** $\in \mathbf{P}$, thus completing the proof.

6.2 Negative Results in Membrane Computing

The dependency graph can be used to show negative results as well. For instance, one can show:

$$\text{ONLY-ONE-OBJECT} \notin \mathbf{PMC}_{\mathcal{AM}^0(-d, +ne)}^{1f} \tag{1}$$

In this context, a computation is accepting if and only if there is a path in G from s to t , with s being the initial and t the final vertex of the computation. We will show (1) by contradiction. Let us assume there were such a \mathbf{P} system $\Pi \in \mathcal{AM}^0(-d, +ne)$. Then, a path from the initial vertex to the vertex (yes, env) that passes the vertex (a, i_{in}) would exist, to correctly resolve the case $\{a\}$. When we analyze the case $\{a^n\}, n > 1$, the following holds:

$$\forall n > 1 : G_{\Pi+\{a\}} = G_{\Pi+\{a^n\}}$$

This holds, because the vertex set of the dependency graph is a **set** and not a multiset. And because of that, a different multiplicity of the same element (n elements a instead of a single a) does not change the graph at all. Thus, every computation would be accepting \Rightarrow this contradicts our assumption.

7 Proposed Solution for the ONLY-ONE-OBJECT Problem Using Dissolution

As we have just seen, the ONLY-ONE-OBJECT problem cannot be solved with a P system $\Pi \in \mathcal{AM}^0(-d, +ne)$. However, if we allow the use of dissolution rules, the problem is solvable. In Figure 1, we propose a P system that is in $\mathcal{AM}^0(+d, -ne)$ and solves ONLY-ONE-OBJECT. The system takes a multiset $\{a^n\}, n \in \mathbb{N}^+$ as input in membrane 3 and outputs the answer to the problem (*yes* or *no*) into the system's surrounding. The system uses three dissolution rules: one in membrane 3 and two in membrane 2, two of these only activate themselves if there is an a present in the membrane. In the case $n = 1$, only membrane 3 will be diluted by those rules, in the other case ($n > 1$), the membranes 3 and 2 are diluted. Making use of the auxiliary element β , we can now “count“ the number of necessary computation steps to reach membrane 1. If there is more than one a , membranes 3 and 2 will be diluted in two computation steps, so the auxiliary element β'' will be in membrane 1 and the system will output *no*. On the other hand, if there is only one a , membrane 2 won't be diluted using the dissolution rule $[a]_2 \rightarrow \delta$, but using this one instead: $[\beta''']_2 \rightarrow \delta$. So, if there is only one a , we can increment the auxiliary element β until it reaches β''' . In the final step, if there is a β''' in membrane 1, we output *yes*. It is impossible by design that there are a β'' and a β''' at the same time in membrane 1, which means that the system will always output the same and the correct answer.

8 Conclusions

Multiple use cases exist for the dependency graph in the analysis of computational models. With that tool, we were able to show the non-efficiency of a membrane system and the non-existence of a solution for a given problem when using a specific membrane system (ONLY-ONE-OBJECT $\notin \mathbf{PMC}_{\mathcal{AM}^0(-d, +ne)}^{1f}$). Furthermore, we presented a P system in $\mathcal{AM}^0(+d, -ne)$ that solves the ONLY-ONE-OBJECT problem.

If we want to solve **P** versus **NP**, we should keep making efforts in exploring new techniques to analyze the possibility and feasibility in the context of membrane systems.

Acknowledgements

D. Orellana-Martín acknowledges Contratación de Personal Investigador Doctor. (Convocatoria 2019) 43 Contratos Capital Humano Línea 2. Paidi 2020, supported by the European Social Fund and Junta de Andalucía.

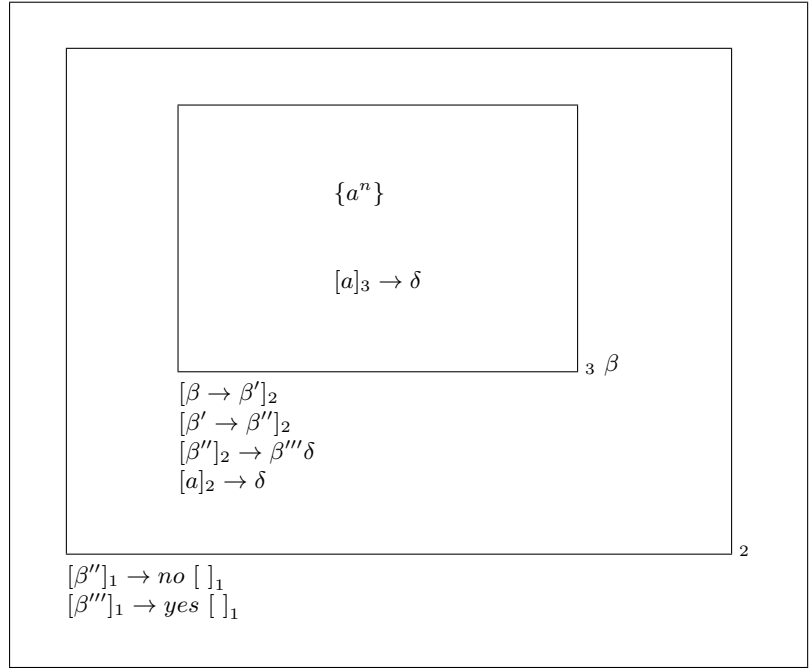


Fig. 1. P system $\Pi \in \mathcal{AM}^0(+d, -ne)$ solving ONLY-ONE-OBJECT

References

- [1] G. Păun. “Computing with Membranes”. In: *Journal of Computer and System Sciences* 61.1 (2000), pp. 108–143. ISSN: 0022-0000. DOI: <https://doi.org/10.1006/jcss.1999.1693>.
- [2] G. Păun. “P Systems with Active Membranes: Attacking NP-Complete Problems”. In: *J. Autom. Lang. Comb.* 6.1 (Jan. 2001), pp. 75–90. ISSN: 1430-189X.
- [3] B. Song, K. Li, D. Orellana-Martín, M. J. Pérez-Jiménez, and I. Pérez-Hurtado. “A Survey of Nature-Inspired Computing: Membrane Computing”. In: *ACM Comput. Surv.* 54.1 (Feb. 2021). ISSN: 0360-0300. DOI: [10.1145/3431234](https://doi.org/10.1145/3431234).
- [4] G. Mauri, A. Leporati, L. Manzoni, A. E. Porreca, and C. Zandron. “Complexity Classes for Membrane Systems: A Survey”. In: *Language and Automata Theory and Applications*. Ed. by A.-H. Dediu, E. Formenti, C. Martín-Vide, and B. Truthe. Cham: Springer International Publishing, 2015, pp. 56–69. ISBN: 978-3-319-15579-1.
- [5] G. Păun. “Introduction to Membrane Computing”. In: *The Journal of Logic and Algebraic Programming* 79 (Jan. 2006), pp. 1–42. DOI: [10.1007/3-540-29937-8_1](https://doi.org/10.1007/3-540-29937-8_1).

- [6] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, and M. J. Pérez-Jiménez. “A new perspective on computational complexity theory in Membrane Computing”. In: *BWMC2019: 17th Brainstorming Week on Membrane Computing*. 2019, pp. 117–126.
- [7] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, and M. J. Pérez-Jiménez. “An apparently innocent problem in Membrane Computing”. In: *BWMC2019: 17th Brainstorming Week on Membrane Computing*. IMCS: International Membrane Computing Society. 2019, pp. 127–138.
- [8] L. Valencia-Cabrera, D. Orellana-Martín, I. Pérez-Hurtado, and M. J. Pérez-Jiménez. “New applications for an old tool”. In: *BWMC2019: 17th Brainstorming Week on Membrane Computing*. 2019, pp. 165–170.
- [9] L. Valencia-Cabrera, D. Orellana-Martín, I. Pérez-Hurtado, and M. J. Pérez-Jiménez. “Dependency Graph Technique Revisited”. In: *CMC20: 20th International Conference on Membrane Computing (2019)*. IMCS: International Membrane Computing Society. 2019, pp. 513–522.
- [10] G. Rozenberg and A. Salomaa, eds. *Handbook of Formal Languages. 3 vols.* Berlin, Heidelberg: Springer-Verlag, 1997.
- [11] G. Păun, G. Rozenberg, and A. Salomaa. *The Oxford Handbook of Membrane Computing*. USA: Oxford University Press, Inc., 2010. ISBN: 0199556679.
- [12] M. Pérez-Jiménez, Á. Jiménez, and F. Caparrini. “Complexity classes in cellular computing with membranes”. In: *Natural Computing 2* (Sept. 2003), pp. 265–285. DOI: 10.1023/A:1025449224520.
- [13] M. J. Pérez-Jiménez, A. R. Jiménez, and F. S. Caparrini. “Decision P Systems and the $P \neq NP$ Conjecture”. In: *Membrane Computing*. Ed. by G. Păun, G. Rozenberg, A. Salomaa, and C. Zandron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 388–399. ISBN: 978-3-540-36490-0.
- [14] A. Cordon-Franco, M. Ángel-Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez. “Weak Metrics on Configurations of a P System”. In: *Proceedings of the Second Brainstorming Week on Membrane Computing (2004)*. 2004, pp. 139–151.
- [15] G. Păun. “Further Twenty Six Open Problems in Membrane Computing”. In: *Proceedings of the Third Brainstorming Week on Membrane Computing*. Ed. by M. Gutiérrez-Naranjo, A. Riscos-Núñez, F. Romero-Campero, and D. Sburlan. Sevilla: Fénix Editora, 2005, pp. 249–262.
- [16] I. Pérez-Hurtado, M. J. Pérez-Jiménez, A. Riscos-Núñez, M. Á. Gutiérrez-Naranjo, and M. Rius-Font. “On a partial affirmative answer for a Păun’s conjecture”. In: *International Journal of Foundations of Computer Science* 22.01 (2011), pp. 55–64. DOI: 10.1142/S0129054111007824. eprint: <https://doi.org/10.1142/S0129054111007824>.