

Internship report

Characterization of Symport/Antiport P Systems Complexity Classes

Vivien DUCROS

M1 MPRI, DER Informatique
ENS Paris-Saclay, Gif-sur-Yvette, France

supervised by

Claudio ZANDRON

Dipartimento di Informatica, Sistemistica e Comunicazione,
Università degli Studi di Milano-Bicocca, Milano, Italia

March – July 2023

Abstract

This report presents the characterization results obtained during my internship in the Natural Computing Lab of the Computer Science Department of the University of Milano-Bicocca. These results apply to Membrane systems complexity classes. Membrane systems are distributed and parallel computational models inspired by biological cells. We studied some variants of these models, called *Symport/Antiport P systems*. These systems are efficient (that is, they can solve computationally hard problems in a polynomial time) because they can perform some actions at each single step. We were able to characterize **PSPACE** in term of symport/antiport P systems with membrane division (process inspired by the mitosis of cells) and to characterize $\mathbf{P}^{\#\mathbf{P}}$ in term of symport/antiport P systems with membrane separation (process inspired by the meiosis of cells).

Keywords Membrane Computing, P system with symport/antiport, PSPACE, Computational complexity

Contents

1	Introduction	2
1.1	About the internship	2
1.2	Scientific context	3
2	Definitions	4
2.1	Basic definitions	4
2.2	Symport/antiport P systems	5
2.3	Decision problems	7
3	State of the art	8

4	Characterization of PSPACE by symport/antiport P systems with division rules	9
4.1	PSPACE, a tight upper bound for symport/antiport P systems with membrane division	9
4.2	The case of symport/antiport P systems with membrane separation	12
5	Characterization of $P^{\#P}$ by symport/antiport P systems with separation rules	13
6	Conclusion	16
	References	17
	Appendices	18
A	Simulation algorithms for symport/antiport P systems	18
A.1	P systems with membrane division	18
A.2	P systems with membrane separation	20

1 Introduction

This report is organized as follow: this Section 1 presents the practical information about my internship and introduces to the reader the scientific context of the Membrane Computing. Section 2 contains the definitions of membrane systems and their related complexity classes. In Section 3, we give an overview of the hierarchical structure of the relations between complexity classes known at the beginning of the internship. In Section 4 we characterize **PSPACE** in term of symport/antiport P systems with membrane division, and in Section 5 we characterize **$P^{\#P}$** in term of symport/antiport P systems with membrane separation. Finally, we conclude in Section 6 about the achieved work and we propose some line of research to continue the work in this field.

1.1 About the internship

From March to July 2023, I did my Master 1 internship in the [Natural Computing Lab](#), under the supervision of Claudio Zandron. This team of the Department of Informatics, Systems and Communication of the University of Milano-Bicocca, works on theoretical computational models inspired by biological systems. The objective of my internship was to study computational aspect of some variant of membrane systems with Symport/Antiport communication rules.

Here is the initial subject of the internship:

Computational Complexity Aspects of Membrane systems

In the research field of membrane computing, various works considered the possibility to attack computationally hard problems, solving them in polynomial time (but exponential space) by exploiting the possibility to duplicate cells.

Some variants are known to be able to solve all problems in the class NP, while other variants are even more powerful, being able to solve problems in the class PSPACE. Classes of systems with some limitations that can only solve problems in the class P, despite the possibility to duplicate cells, are also known.

In particular, in the stage we will concentrate on tissue P systems (that is, systems where cells are not organized in a hierarchical structure, but in a neighborhood graph, like a sort of generalized tissue) using symport/antiport rules. Some works have already considered such systems, but an interesting research topics concern the power of such systems using only symport rules or only antiport rules. The goal of the stage is the characterization of complexity classes related to such systems.

Ref [20]: Sosík, P systems attacking hard problems beyond NP: a survey.

The proposed work was about tissue P systems, and the effects of some restrictions on the complexity of problems solved by these systems. Finally we focused on the characterizations of classes of cell-like P systems, an other way to organize the membranes of the systems. These results complete the complexity structure of cell-like P systems as the one already known on tissue P systems.

After a deep bibliographical work, I found quickly some ideas leading to a first characterization. I formalized the definitions and the proof, so we decided with Claudio to publish the result. I wrote the whole manuscript which was corrected by my supervisor. We completed together the introduction part and discussed about remaining open problems and future work. The submission of our article to two Elsevier Journals was unsuccessful, the manuscript had been rejected for a high similarity overlap with other articles. The rate given by the similarity check tool was above the editor's threshold. As our work is original, we are searching for another journal. The likeness can be explained by our definition section, which is similar to all other articles in our field. Simultaneously with the submissions, I began to work on the characterization for P systems with membrane separation. I finally found the last result just before the end of the internship.

This report is based on the manuscript I worked on. The introduction and definition sections (Sections 1 and 2) have been adapted to be more accessible to non-specialists. Section 4 on the characterization of **PSPACE** is the center piece of the article. For layout reasons, the algorithms have been moved to the Appendix A.

We sum up here the activities carried out during the internship chronologically:

- Bibliographical work
- First ideas to simulate a symport/antiport P system with membrane division in **PSPACE**
- Writing the simulation algorithm
- Programming and testing the algorithm
- Modifications and proof of the algorithm
- Redaction of the article
- Article submission #1
- Additional bibliographic research
- Work on the simulation of cell-like P system with membrane separation by tissue P system (unsuccessful)
- Article submission #2
- Solving **LEXICAL_MIDDLE_3SAT** with a P system with membrane separation

1.2 Scientific context

Membrane Computing refers to a research area dealing with the investigation of a computation model inspired by the functioning of the biological cell, introduced by Georghe Păun at the early beginning of the 21st century [10]. Such a model, called Membrane system or P system, is constituted by a hierarchy of embedded membranes delimiting different regions. A single external membrane, called the *skin*, delimits the whole system, separating it from the environment. Regions contain objects that evolve according to some evolution rules, which transform and communicate them through membranes among different regions.

In [11] a highly dynamic branch of study was started, centered around the notion of *active membranes*: membranes can be *divided* through evolutionary rules that duplicate their contents. A different possibility is to consider *separation* of membranes, in which membranes can be duplicated but the content of the original membrane is distributed between the two newly obtained membranes (instead of being duplicated). These approaches enable the construction of an exponential amount of resources within a polynomial time frame. Subsequently, numerous works have appeared regarding the exploration of complexity classes defined by P systems with active membranes, which make use of different features and incorporate various limitations. These investigations aim to comprehend how specific characteristics impact the development of time-efficient systems, capable of solving computationally challenging problems in polynomial time while utilizing exponential space, as opposed to non-efficient systems.

In this paper we follow this line, by considering a variant of membrane systems called symport/antiport P systems, introduced in [9]. In such a model, the communication of objects between two regions can be done either by moving two objects simultaneously, from the same starting region to an adjacent one (symport), or by exchanging two objects from two adjacent regions (antiport).

In particular, we investigate the computational efficiency of such systems augmented either with division or separation rules. It has been shown in [18] that the class of symport/antiport P systems with membrane division and rules of length not greater than 3 can solve, in a uniform way and in polynomial time (and exponential space), the problem **QSAT**, a well-known **PSPACE**-complete problem.

PSPACE is a complexity class that plays a crucial role in complexity theory and it is particularly important for parallel models of computation like, e.g., alternating Turing machines. While a characterization of **PSPACE** in terms of membrane systems has already been obtained for P systems with active membranes in [22] (and then in related works like [3] for non-confluent and shallow systems, and in [21] using proteins on membranes), there are other variants of membrane systems for which only partial answers have been obtained so far. Thus, such variants are, at the moment, on the “candidate list” of models that could give another characterization of this class. Among them, one of the most prominent classes conjectured to characterize **PSPACE** are P systems with symport/antiport, as pointed out in [20].

Some partial known results for other variants concern, for instance, symport/antiport P systems with the use of promoters, for which it has been proved that **PSPACE** is a lower bound [19]; symport/antiport P systems with membrane separation are able to solve the problem **SAT** ([24]), and thus for them **NP** is a lower bound. The effect of environment has been studied in [7]: it has no influence on the computational power of symport/antiport P systems with membrane division, but in the case of membrane separation, removing the environment makes the class of problem solved equate **P**. When the length of the rules is limited to 1, the symport/antiport P systems with membrane division or separation are inefficient and can only solve the problems in **P** ([4]). In all these examples, the membranes are organized in a nested structure, but another variant called *tissue P systems* organizes the membranes as a directed graph. When fission (division or separation) rules are allowed, they characterize the class $\mathbf{P}^{\#\mathbf{P}}$ ([2]). This result will lead us to a characterization of $\mathbf{P}^{\#\mathbf{P}}$. For tissue P systems with evolutionary symport/antiport rules [8] it is known that all problems in the class **PP** can be solved, but it seems that to reach the power of **PSPACE** non-deterministic models must be considered.

In Section 4, we show that the conjecture stated in [20] about the characterization of **PSPACE** in terms of symport/antiport P systems is true: in fact, the class **PSPACE** coincides with the set of problems solved in polynomial time by symport/antiport P systems with membrane division. Furthermore, we show that **PSPACE** is also an upper bound for symport/antiport P systems using membrane separation instead of membrane division. Then, in Section 5, we refine this upper bound and prove that the set of problems solved in polynomial time by symport/antiport P systems with membrane separation is exactly $\mathbf{P}^{\#\mathbf{P}}$.

2 Definitions

In this section we recall some definitions that will be used in the rest of the paper. For further information, we refer the reader to the *Introduction to Membrane Computing* [12], the *Oxford Handbook of Membrane Computing* [13], and the *Handbook of Formal Languages* [17]. Concerning the notions on computational complexity in the framework of P systems, we refer the reader to [14].

2.1 Basic definitions

A *multiset* m over a set E is a pair (E, f) where $f : E \rightarrow \mathbb{N}$ and for $x \in E$, $f(x)$ is the number of occurrences of x in m . The *support* of a multiset $m = (E, f)$ is $\text{Supp}(m) = \{x \in E \mid f(x) > 0\}$. Its *cardinal* is $|m| = \sum_{x \in E} f(x)$. Let $m_1 = (E, f_1)$ and $m_2 = (E, f_2)$ two multiset. Their *sum*, denoted $m_1 + m_2$ is the multiset (E, g) where $g(x) = f_1(x) + f_2(x)$ for all $x \in E$. Their *difference*, denoted $m_1 - m_2$ is the multiset (E, g) where for each $x \in E$ $g(x) = f_1(x) - f_2(x)$ if $f_1(x) \geq f_2(x)$ and $g(x) = 0$ else. m_1 is *included* in (or is a *submultiset* of) m_2 , denoted $m_1 \subseteq m_2$, if $f_1(x) \leq f_2(x)$ for each $x \in E$.

In our examples, we describe the multisets by words. With $a, b, c \in E$, the multiset $m = (E, f)$ with $f(a) = 2$, $f(b) = 0$ and $f(c) = 1$ is represented by aac , a^2c or ca^2 . The order of the elements doesn't matter, and the multiplicity is often indicated in exponent.

$\mathcal{M}_{k,n}(E)$ is the set of *matrices* of $k \in \mathbb{N}$ rows and $n \in \mathbb{N}$ columns with elements in E . If $k = 0$ or $n = 0$ the matrix is empty. Let $A \in \mathcal{M}_{k,n}(E)$, $i \in \llbracket 1, k \rrbracket$ and $j \in \llbracket 1, n \rrbracket$. $A_{i,j} \in E$ represents the element at row i and column j , $A|_{i,j} \in \mathcal{M}_{i,j}(E)$ is the *submatrix* of A with only the first i rows and the first j columns of A .

2.2 Symport/antiport P systems

Definition 1 (Membrane structure). A membrane structure is a rooted tree in which nodes are labeled by integers from 1 to q , where $q > 0$ is the size of the membrane. The nodes are called membranes. The depth $d(h)$ of a membrane h is its distance from the root. The root is called the *skin*. A leaf is called an *elementary* membrane. The parent of a membrane h is denoted by $p(h)$.

A membrane structure can be represented in different ways, for instance as a rooted tree or as a Venn diagram (see Figure 1).

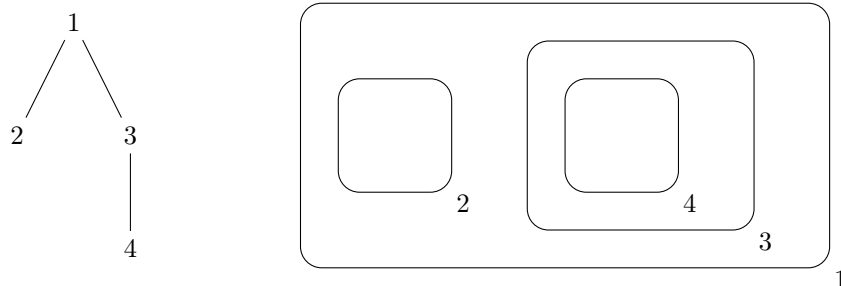


Figure 1: Membrane $\mu = [[]_2 [[]_4]_3]_1$ represented as a rooted tree and as a Venn diagram

Definition 2 (Symport/antiport P system). A symport/antiport P system with membrane division is a tuple

$$\Pi = (\Gamma, \mathcal{E}, \mu, m_1, \dots, m_q, \mathcal{R}_1, \dots, \mathcal{R}_q, i_{out})$$

where:

- Γ is a finite alphabet of objects
- $\mathcal{E} \subseteq \Gamma$ is a finite alphabet of objects in the environment
- μ is a membrane structure of size q
- m_i , $1 \leq i \leq q$ is the finite multiset of objects of Γ initially located in membrane i
- \mathcal{R}_i , $1 \leq i \leq q$ is the finite set of evolution rules of the membrane i of type (u, out) , (u, in) the symport rules; $(u, out; v, in)$ the antiport rules; and $[a]_i \rightarrow [b]_i[c]_i$ the division rules with u with v non-empty multisets of Γ and $a, b, c \in \Gamma$
- i_{out} is the index of the output membrane.

A symport/antiport system with membrane separation has a partition $\{\Gamma_1, \Gamma_2\}$ of Γ ($\Gamma_1, \Gamma_2 \neq \emptyset$, $\Gamma_1 \cup \Gamma_2 = \Gamma$ and $\Gamma_1 \cap \Gamma_2 = \emptyset$). Instead of division rules, this system has separation rules: $[a]_i \rightarrow [\Gamma_1]_i[\Gamma_2]_i \in \mathcal{R}_i$.

Definition 3 (Symport/antiport P system with input). A symport/antiport P system with input is a tuple

$$\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, m_1, \dots, m_q, \mathcal{R}_1, \dots, \mathcal{R}_q, i_{in}, i_{out})$$

where $(\Gamma, \mathcal{E}, \mu, m_1, \dots, m_q, \mathcal{R}_1, \dots, \mathcal{R}_q, i_{out})$ is a symport/antiport P system and where:

- Σ is the input alphabet such that $\mathcal{E} \subseteq \Gamma \setminus \Sigma$ and $m_i \subseteq \Gamma \setminus \Sigma$, $1 \leq i \leq q$
- i_{in} is the index of the input membrane.

The *skin* is the membrane containing all the others, its index is the root of μ . A *configuration* $\mathcal{C} = (\mu', (m'(h))_{h \in \mu'})$ of a P system is a tuple with a membrane structure and a family of multisets of objects in the membranes at a given step. The symport/antiport P system evolves according to the maximal parallel mode, that is: all rules that can be applied are applied in parallel as long as they can. Objects can be used by only one rule at a given step, and the objects created or communicated can't be used again. If an object can be used by more than one rule, the applied rule is chosen non-deterministically. If a division rule or a separation rule is applied, it is the only one applied to this membrane. Furthermore, all membranes except the *skin* and the output membrane can divide, and **only the elementary membranes** except the *skin* and the output membrane can separate.

The symport rule $(u, out) \in \mathcal{R}_h$ can be used if the multiset u is included in the membrane h . It sends u outside the membrane h (to the parent membrane $p(h)$). The symport rule $(u, in) \in \mathcal{R}_h$ can be used if the multiset u is included in the parent membrane $p(h)$. It sends u inside the membrane h . The antiport rule $(u, out; v, in) \in \mathcal{R}_h$ can be used if u is included in h and v is included in $p(h)$. Then u is sent inside $p(h)$ whereas v is sent inside h . The objects of \mathcal{E} initially present in the environment are available in arbitrary number of copies.

The division rule $[a]_h \rightarrow [b]_h[c]_h \in \mathcal{R}_h$ can be used if the object a is in h . Then two identical copies of h are created, replicating the objects and the membranes contained in h . In the first copy, the object a is replaced by b , and by c in the second one. The initial membrane h is deleted. The new membranes keep the index h so the rules of \mathcal{R}_h can be applied to both at the next step. The membrane separation rule $[a]_h \rightarrow [\Gamma_1]_h[\Gamma_2]_h \in \mathcal{R}_h$ is used when the object a is in h . The object a is consumed, and the membrane h is split into two membranes indexed by h , the first one containing the objects of Γ_1 in h and the second the objects of Γ_2 in h .

Starting with the initial configuration of a P system and applying the rules in maximal parallel order, we obtain a sequence of configurations called computation, representing the state of the system at each step. When there are no rules to apply to a configuration, the computation halts. The result is the multiset of objects in the membrane i_{out} of the final configuration.

Example 1. Let $\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, m_1, m_2, \mathcal{R}_1, \mathcal{R}_2, i_{in}, i_{out})$ with:

- $\Gamma = \{a, b, c\}$, $\mathcal{E} = \{b, c\}$, $\Sigma = \{a\}$
- $\mu = [[]_2]_1$
- $m_1 = m_2 = \emptyset$
- $\mathcal{R}_1 = \{(a, out; bc, in), (b^2, out; c, in)\}$
- $\mathcal{R}_2 = \{(c, in)\}$
- $i_{in} = 1$ and $i_{out} = 2$

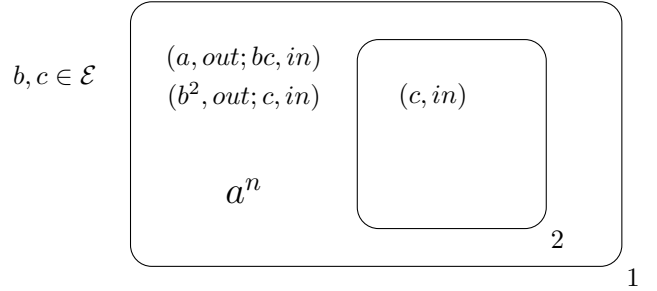


Figure 2: The P system Π of example 1

This symport/antiport P system with input computes the multiset $c^{n+\lceil n/2 \rceil}$ when the multiset a^n is given as an input in membrane $1 = i_{in}$. The computation needs only three steps:

- At the first step, the rule $(a, out; bc, in) \in \mathcal{R}_1$ is used n times. At the end of this step, membrane 1 contains the multiset $b^n c^n$ (the objects b and c are present in the environment in infinitely many copies).
- Then, at step 2, the rule $(b^2, out; c, in) \in \mathcal{R}_1$ is used $\lceil n/2 \rceil$ times and the rule $(c, in) \in \mathcal{R}_2$ is used n times. Thus, after the parallel application of the rules, membrane 1 contains $c^{\lceil n/2 \rceil}$ and membrane 2 contains c^n .
- Finally, the rule $(c, in) \in \mathcal{R}_2$ is applied $\lceil n/2 \rceil$ times. No other rules can be used, so we are in the halting configuration. The multiset $c^{n+\lceil n/2 \rceil}$ in membrane $2 = i_{out}$ is the result of the computation.

2.3 Decision problems

Definition 4 (Decision problem). Let Σ be an alphabet. A decision problem X on Σ is a pair (I_X, θ_X) where I_X is the set of instances and $\theta_X : I_X \rightarrow \{0, 1\}$. An instance $x \in I_X$ is said to be accepted by the problem if $\theta_X(x) = 1$.

Definition 5 (Recognizer symport/antiport P system). A recognizer symport/antiport P system is a tuple:

$$\Pi = (\Gamma, \mathcal{E}, \Sigma, \mu, m_1, \dots, m_q, \mathcal{R}_1, \dots, \mathcal{R}_q, i_{in}, i_{out})$$

where:

- **yes**, **no** $\in \Gamma \setminus \Sigma$ are the answer objects, $\exists i, j \in \llbracket 1, q \rrbracket$, **yes** $\in m_i$, **no** $\in m_j$
- $i_{out} = 0$

and such that:

- Every computation ends
- For all computations of Π on the same input, either **yes** (the computation is said to be accepting) or **no** (the computation is said to be rejecting) have been released in the environment **only at the last step**.

Definition 6 ($\mathbf{CD}_e\mathbf{C}$, $\mathbf{CD}_{ne}\mathbf{C}$, and \mathbf{CSC} recognizer P systems classes). For $k \in \mathbb{N}$, by $\mathbf{CD}_e\mathbf{C}(k)$ (resp. $\mathbf{CSC}(k)$) we denote the class of all recognizer symport/antiport P systems with division (resp. separation) rules only for elementary membranes and having rules of length at most k . Furthermore, $\mathbf{CD}_{ne}\mathbf{C}(k)$ is the class where elementary and non-elementary membranes (except the *skin* and the output membrane) can divide and where rules have a length of at most k .

According to this definition, the following results hold: $\mathbf{CD}_e\mathbf{C} = \bigcup_{k \in \mathbb{N}} \mathbf{CD}_e\mathbf{C}(k)$, $\mathbf{CD}_{ne}\mathbf{C} = \bigcup_{k \in \mathbb{N}} \mathbf{CD}_{ne}\mathbf{C}(k)$ and $\mathbf{CSC} = \bigcup_{k \in \mathbb{N}} \mathbf{CSC}(k)$.

Definition 7 (\mathcal{D} -consistent family, polynomially uniform family). Let $\mathbf{\Pi} = (\Pi(i))_{i \in I}$ be a family of P systems. It is said to be \mathcal{D} -consistent if $\Pi(i) \in \mathcal{D}$ for all $i \in I$. It is said to be polynomially uniform if there exists a deterministic Turing Machine that computes the P system $\Pi(i)$ for all $i \in I$ in polynomial time of $|i|$.

Definition 8 (Recognition in a semi-uniform way). A decision problem $X = (I_X, \theta_X)$ is said to be recognizable in a semi-uniform way by a polynomially uniform family $\mathbf{\Pi} = (\Pi(x))_{x \in I_X}$ of P systems (without input) such that for all $x \in I_X$, if $\theta_X(x) = 1$ then all computations of $\Pi(x)$ are accepting (soundness), and for all $x \in I_X$, if there exists one accepting computation of $\Pi(x)$ then $\theta_X(x) = 1$ (completeness).

Remark 1. The properties of soundness and completeness imply the *confluence*, which is that the computations of $\Pi(x)$, $x \in I_X$ are all accepting or all rejecting. Such a P system is said to be confluent.

Definition 9 (Recognition in a uniform way). A decision problem $X = (I_X, \theta_X)$ is recognizable in a uniform way by a polynomially uniform family $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbb{N}}$ of P systems with input if there exist two polynomially computable functions $cod : I_X \rightarrow \bigcup_{n \in \mathbb{N}} \Sigma(n)$ and $s : I_X \rightarrow \mathbb{N}$, such that for all $x \in I_X$, if $\theta_X(x) = 1$ then all computations of $\Pi(s(x))$ on the input $cod(x)$ are accepting (soundness), and for all $x \in I_X$, if there exists one accepting computation of $\Pi(s(x))$ on $cod(x)$, then $\theta_X(x) = 1$ (completeness).

Definition 10 ($\mathbf{MC}_{\mathcal{D}}(g)$ and $\mathbf{MC}_{\mathcal{D}}^*(g)$ complexity classes). Let \mathcal{D} be a class of recognizer P systems and g be a constructible function. A decision problem X is in $\mathbf{MC}_{\mathcal{D}}(g)$ (resp. $\mathbf{MC}_{\mathcal{D}}^*(g)$) if there exists a \mathcal{D} -consistent family $\mathbf{\Pi}$ recognizing X in a uniform way (resp. semi-uniform way) and such that for all $x \in I_X$, each computation of $\Pi(s(x))$ on $cod(x)$ (resp. of $\Pi(x)$) ends in less than $g(|x|)$ steps.

Definition 11 ($\mathbf{PMC}_{\mathcal{D}}$ and $\mathbf{PMC}_{\mathcal{D}}^*$ complexity classes). Let \mathcal{D} be a class of recognizer P systems. The polynomial time complexity classes of P systems are defined as follows:

$$\mathbf{PMC}_{\mathcal{D}} = \bigcup_{k \in \mathbb{N}} \mathbf{MC}_{\mathcal{D}}(O(x^k)) \quad \mathbf{PMC}_{\mathcal{D}}^* = \bigcup_{k \in \mathbb{N}} \mathbf{MC}_{\mathcal{D}}^*(O(x^k))$$

Proposition 1. *Let $\mathcal{D}, \mathcal{D}'$ be two classes of recognizer P systems such that $\mathcal{D} \subseteq \mathcal{D}'$, and g be a constructible function. The following statements hold by definition:*

- $\text{MC}_{\mathcal{D}}(g) \subseteq \text{MC}_{\mathcal{D}}^*(g)$, and in particular $\text{PMC}_{\mathcal{D}} \subseteq \text{PMC}_{\mathcal{D}}^*$
- $\text{MC}_{\mathcal{D}}(g) \subseteq \text{MC}_{\mathcal{D}'}(g)$, $\text{MC}_{\mathcal{D}}^*(g) \subseteq \text{MC}_{\mathcal{D}'}^*(g)$, and in particular $\text{PMC}_{\mathcal{D}} \subseteq \text{PMC}_{\mathcal{D}'}$, $\text{PMC}_{\mathcal{D}}^* \subseteq \text{PMC}_{\mathcal{D}'}^*$

3 State of the art

In this section, I give an overview of the results on complexity classes of symport/antiport P systems. In order to present a larger context I added the results on the symport/antiport tissue P systems.

The symport/antiport tissue P systems work similarly to the symport/antiport P systems defined in Section 2, but they are structured as graphs and not trees. The symport/antiport rules are adapted to graphs: with h, k two nodes of the graph, a rule between membranes h and k is written $[u]_h \leftrightarrow [v]_k$ where u, v are multisets such that at most one of them is empty. The rules implying the environment are written $[u]_h \leftrightarrow v$, where u is not empty. Because the membranes are no longer nested, the division rules create a copy of a membrane, copying its elements and not the linked membrane. Each new copy can interact with the same rules with the other membranes. The separation rules work the same, but objects are shared between the copies, according to the partition $\Gamma = \Gamma_1 \cup \Gamma_2$.

To avoid any confusion, the symport/antiport P systems initially defined in this report are called *cell-like* symport/antiport P systems, in opposition with the symport/antiport *tissue* P systems. The recognizer class are written **TDC**, **TDC**(k) for the classes of symport/antiport tissue P systems with membrane division, and **TSC**, **TSC**(k) for the symport/antiport tissue P systems with membrane separation. The complexity classes on the tissue P systems are defined similarly as the cell-like ones. The notion of elementary membrane has no longer sense for tissue P system so there is no need to add a notation.

The Figure 3 represents the inclusions between the complexity classes. An arrow between two classes indicates that the starting class is included in the ending class. The arrows are labelled by the reference to the result's article. Let's give some elements about this classification:

- In the center of the classification we find the common complexity classes. We prefer use the **NP** \cup **coNP** class because our systems always halt, so we could exchange the roles of **yes** and **no** to recognize the complement problem.
- The dotted lines indicate the inclusions given by the definitions of our systems. Vertical ones are trivial. The important thing is that symport/antiport P systems with membrane separation extends the cell-like ones because, a tree is also a graph. This is not true for P systems with membrane division because the cell-like ones have the power to replicate the entire descendants whereas tissue P systems don't.
- At the bottom, there are a lot of separation results between the efficient (containing **NP**) and non-efficient variants. We note that the restriction used is often the length of the rules.
- The common class **P^{#P}** characterizes the problems solved by polynomial time symport/antiport tissue P systems with membrane division or separation. These results comes from [2].
- The result of [18] about the power of membrane division on tree structures states that cell-like symport/antiport P systems with membrane division can solve in polynomial time problems of **PSPACE**.
- Finally, the colored arrows are the inclusions I worked on during my internship. They are proven by Theorems 1 and 2 in Section 4 for the red arrows. The blue arrow is obtained from Theorem 3 of Section 5. With the other inclusions, we see the characterizations appear with the cycles in the classification.

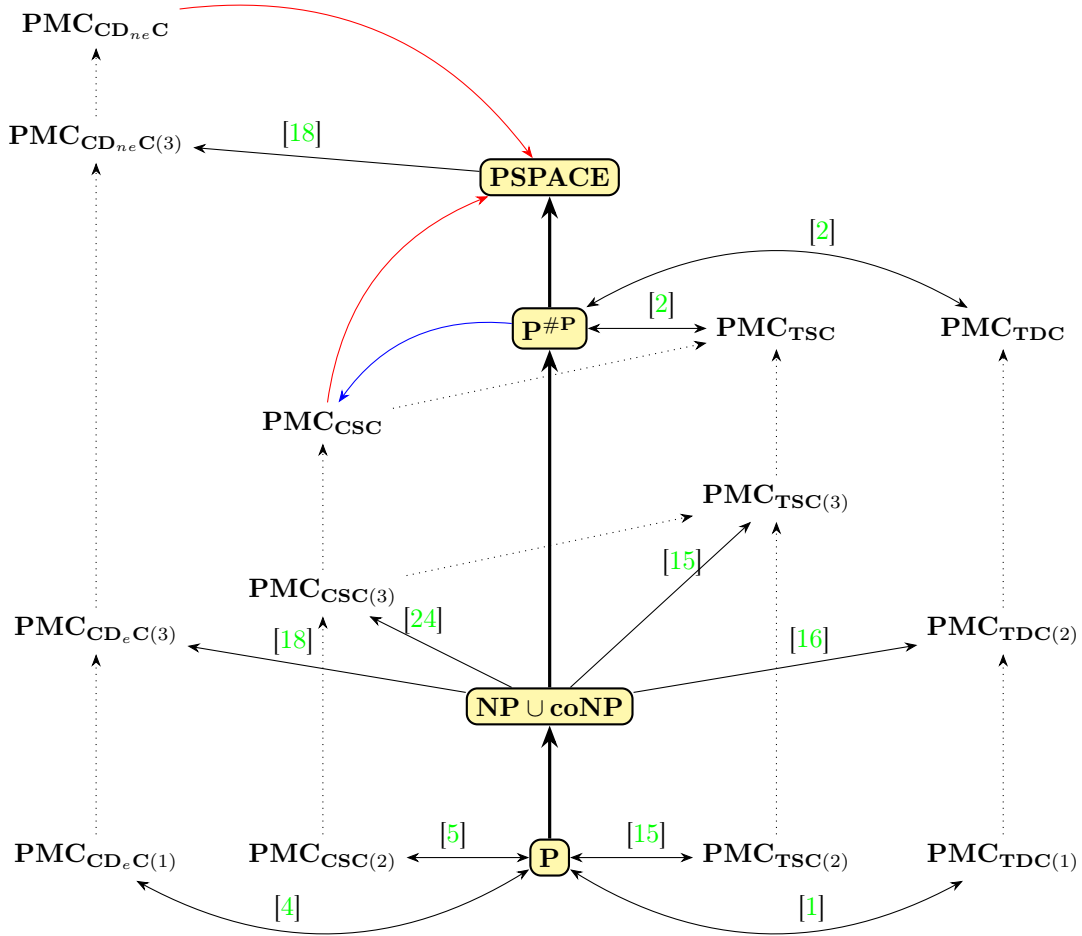


Figure 3: Classification of complexity classes of symport/antiport P systems

4 Characterization of PSPACE by symport/antiport P systems with division rules

This section is taken from the manuscript written during my internship. We prove that **PSPACE** is an upper bound for the polynomial classes of cell-like symport/antiport P systems. This leads us to a characterization of **PSPACE**.

4.1 PSPACE, a tight upper bound for symport/antiport P systems with membrane division

In this section, we prove that the complexity class **PSPACE** represents an upper bound for the $\mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}}^*$ class. This demonstration is inspired by the proof of the simulation in **PSPACE** of a P system with active membranes presented in [22]. Thanks to the result in [18], showing that the opposite inclusion is also true, we obtain a characterization of **PSPACE**.

Lemma 1. *Given $X \in \mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}}^*$, $\Pi = (\Pi(x))_{x \in I_X}$ a family of $\mathbf{CD}_{ne}\mathbf{C}$ P systems solving X in a semi-uniform way, and $x \in I_X$, there exists a Turing Machine $\mathcal{M}_{\Pi(x)}$ that computes in polynomial space of $|x|$ the result of $\Pi(x)$. This machine can be constructed in polynomial time of $|x|$.*

Proof. A problem $X \in \mathbf{PMC}_{\mathbf{CD}_{ne}\mathbf{C}}^*$, a family $\mathbf{\Pi} = (\Pi(x))_{x \in I_X}$ of $\mathbf{CD}_{ne}\mathbf{C}$ P systems solving X in a semi-uniform way, and $x \in I_X$ are given. The objective is to simulate in polynomial space of $|x|$ the P system $\Pi(x) = (\Gamma, \mathcal{E}, \mu, m_1, \dots, m_q, \mathcal{R}_1, \dots, \mathcal{R}_q, i_{out} = 0)$.

Because of membrane division, it isn't possible to simulate $\Pi(x)$ in **PSPACE** from the initial configuration: the number of membranes to store can be exponential. The solution is to compute recursively the objects contained in a membrane at any step. The result is then obtained by checking if the skin membrane released the objects **yes** or **no** in the environment at each step. Because the P systems to simulate are confluent, a priority order on rules has been added to keep the computation consistent. This unique computation will have the same result as all the other possible computations.

To identify a membrane during the computation, its index $h \in \mu$ must be known. It also must be specified when it has been divided and which part of the divided membrane is considered, and the same thing for all the parents of that membrane. Thus, to identify a membrane h of depth k and at step n , a matrix of indices $I \in \mathcal{M}_{k,n}(\{1,2\})$ is given. The last row represents the evolution of the membrane at each step. If the element $I_{k,m} = 2$, it indicates that h_I represents the second membrane issued by the division of membrane h at step m . If $I_{k,m} = 1$, then either the membrane divided at step m and the first membrane of the division is considered, or the membrane hasn't divided.

Example 2. In Figure 4 one can find the matrix representations of the membranes of a P system. At step $n = 1$, membrane 3 has been divided. Then at step $n = 2$, membrane 4 situated in the first membrane created by the division of membrane 3 has been divided.

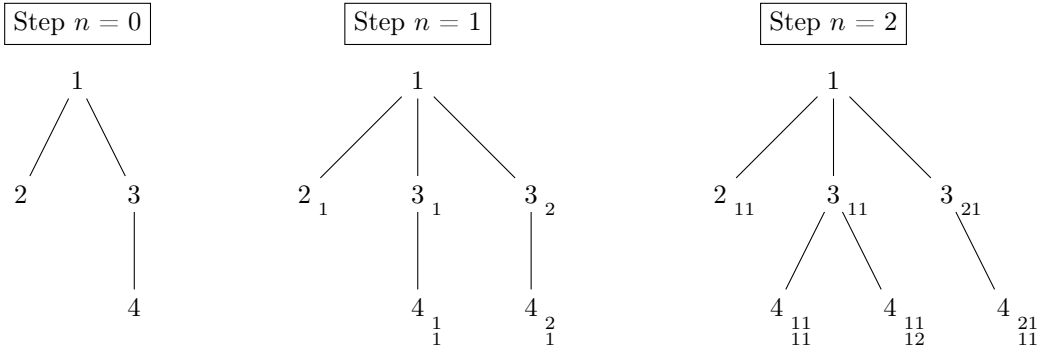


Figure 4: Evolution of a P system with membrane division

The main function of the algorithm is the function **State**. With the parameters h, I, k, n it computes the multiset of objects in the membrane h_I at step n . The parameters verify: $h \in \mu, k$ is the depth of $h, I \in \mathcal{M}_{k,n}(\{1,2\})$.

The priority order on rules chosen is as follows: at step n , the P system evolves by applying first the rules of shallow membranes (the *skin*, then its children and their children...), and for membranes at the same depth, priority is given to the membrane with the lowest membrane index h in μ . For a given membrane h , the first rules applied are division rules, in the order of appearance in \mathcal{R}_h , followed by the symport-out rules, the symport-in rules, and the antiport rules.

A call to the function **State** first checks if the membrane h_I exists: does the membrane $h_{I|_{k,n-1}}$ exist at step $n-1$? Do its parents exist and have they been divided if there is a 2 in the last column? Does the membrane divide if the element $I_{k,n} = 2$? In parallel of that, it computes the multisets of objects present in the parent of $h_{I|_{k,n-1}}$ at step $n-1$ and the evolution at the beginning of step n by applying rules to the parents. All these things are done in the function **ParentSimulation**. It also computes the multiset of membrane $h_{I|_{k,n-1}}$ at step $n-1$ (recursive call to **State**). Then, it applies the division and communication rules to h (functions **ApplyDivisionRule** and **ApplyCommunicationRules**). Finally, it computes the state of all the children of membrane h at step $n-1$ to apply the communication rules between h and its children by calling the function **ContributionFromChildren**. If $k = 0$, then the function **State** returns the multiset of objects contained in the environment. If $n = 0$, then

the function **State** returns the initial multiset of objects contained in h .

Space complexity Let r the maximal length of the rules of $\mathcal{R}_1, \dots, \mathcal{R}_q$, d the depth of μ and o_n the number of objects in the system at step n . By definition, $\Pi(x)$ is constructed in polynomial time of $|x|$ so $p = |\Gamma|$, q , r , d and $o_0 = |m_1| + \dots + |m_q|$ are polynomial in $|x|$.

At step $n > 0$, the number of membranes in $\Pi(x)$ is bounded by $q2^{nd}$. In each membrane there are at most $c_n \leq ro_{n-1}$ objects, and there are $o_n \leq q2^{nd}c_n \leq qr2^{nd}o_{n-1} \leq (qr2^{nd})^n o_0$ objects in the system.

To store the objects contained in a membrane, the number of bits required is

$$b_n \leq p \lceil \log(o_n) \rceil \leq p \lceil \log(o_0) \rceil + np \lceil \log(qr) \rceil + n^2 pd$$

Thus, b_n is a polynomial in n with coefficients that are polynomial in $|x|$.

Let's denote $S(k, n)$ the space used by the function **State** when called with the parameters k and n . Then, with c polynomial in $|x|$:

$$\begin{aligned} S(k, 0) &= c \\ S(k, n) &\leq kn + c + 4b_n + \max_{0 \leq i \leq k+1} (S(i, n-1)) \\ &\leq \mathcal{O}(kn + c + b_n) + \max_{0 \leq i \leq k+1} (S(i, n-1)) \\ &\leq \mathcal{O}(kn^2 + nc + nb_n) \end{aligned}$$

The simulation calls the function **State** for n polynomial in $|x|$ because the P system $\Pi(x)$ computes in polynomial time by definition. Then b_n and $S(k, n)$ are polynomial in $|x|$.

Correctness The algorithm must simulate a computation of a $\mathbf{CD}_{ne}\mathbf{C}$ regonizer P system.

Let $\mathcal{C}_0 \Rightarrow \mathcal{C}_1 \Rightarrow \dots \Rightarrow \mathcal{C}_N$ the computation of $\Pi(x)$ following the priority order on rules chosen earlier. The configurations are denoted $\mathcal{C}_n = (\mu_n, (m(h_I))_{h_I \in \mu_n})$ for $0 \leq n \leq N$ with $\mathcal{C}_0 = (\mu, (m_h)_{1 \leq h \leq q})$.

To prove that the algorithm is correct, one must show:

$$\forall n \in \llbracket 0, N \rrbracket, \forall h \in \mu, \forall I \in \mathcal{M}_{d(h), n}(\{1, 2\}), \underbrace{\mathbf{State}(h, I, d(h), n)}_{P(n)} = \begin{cases} m(h_I) & \text{if } h_I \in \mu_n \\ \mathbf{null} & \text{else} \end{cases}$$

By induction on $n \in \llbracket 0, N \rrbracket$:

- If $n = 0$, then for all $h \in \mu$, $\mathbf{State}(h, (), d(h), 0) = m_h = m(h_{()})$ and $h_{()} \in \mu_0$.
- Supposing that $\exists n \in \llbracket 0, N-1 \rrbracket$ such as $P(n)$, let's show $P(n+1)$. Let $h \in \mu$, $k = p(h)$, and $I \in \mathcal{M}_{k, n+1}(\{1, 2\})$.

The following equivalence holds:

$$\begin{aligned} h_I \in \mu_{n+1} \\ \iff \\ \forall i \in \llbracket 0, k \rrbracket, p^{k-i}(h)_{I|_{i, n}} \in \mu_n \wedge (I_{i, n+1} = 2 \implies p^{k-i}(h)_{I|_{i, n}} \text{ must divide at step } n+1) \end{aligned}$$

This is verified by the algorithm. If it is false, then **null** is returned by $\mathbf{State}(h, I, k, n+1)$. Else, the P system is executed following the priority order on rules up to all possible children membranes of $h_{I|_{k, n}}$. The multiset $m(h_I)$ is computed by $\mathbf{State}(h, I, k, n+1)$ with the multisets $m(p^{k-i}(h)_{I|_{i, n}})$ for $i \in \llbracket 0, k \rrbracket$ and the multisets $m(g_{I'})$ for g such as $p(g) = h$, $I' \in \mathcal{M}_{k+1, n}(\{1, 2\})$ such as $I'|_{k, n} = I|_{k, n}$ and $g_{I'} \in \mu_n$. These multisets are obtained by a call to $\mathbf{State}(-, -, -, n)$.

Thus, $P(n+1)$ holds.

The algorithm is correct, the function **State** returns the multiset of objects contained in the membrane if it exists.

Then one can construct in polynomial time of $|x|$ a Turing Machine $\mathcal{M}_{\Pi(x)}$ that computes the result of $\Pi(x)$ calling the function **State** as follows: execute **State**(*skin*, (), 0, n) by incrementing n from 0 until one object **yes** or **no** is released in the environment (it will happen because every computation of a recognizer P system halts releasing **yes** or **no** in the environment). If this object is **yes**, then the machine accepts, otherwise it is **no** and the machine rejects.

This machine computes in polynomial space of $|x|$, the lemma 1 is correct. \square

From the previous lemma, we deduce the following theorem:

Theorem 1. $\text{PMC}_{\text{CD}_{ne}\text{C}}^* \subseteq \text{PSPACE}$

Proof of Theorem 1. Let $X \in \text{PMC}_{\text{CD}_{ne}\text{C}}^*$. Let Π be a family of CD_{ne}C P systems solving X in a semi-uniform way.

Because Π is polynomially uniform, there exists a Turing Machine \mathcal{M}_{Π} that constructs $\Pi(x)$ from the input $x \in I_X$ in polynomial time of $|x|$.

The Turing Machine that runs (in polynomial time of $|x|$) \mathcal{M}_{Π} on its input $x \in I_X$ to obtain the system $\Pi(x)$ and then constructs (in polynomial time of $|x|$) and runs (in polynomial space of $|x|$) the machine $\mathcal{M}_{\Pi(x)}$ given by the lemma 1; solves the problem X in polynomial space of $|x|$.

So $X \in \text{PSPACE}$ and the theorem 1. \square

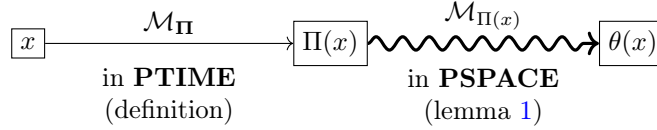


Figure 5: Sketch of the proof of Theorem 1

Corollary 1 (Characterization of **PSPACE**).

$$\text{PSPACE} = \text{PMC}_{\text{CD}_{ne}\text{C}(3)} = \text{PMC}_{\text{CD}_{ne}\text{C}} = \text{PMC}_{\text{CD}_{ne}\text{C}}^*$$

Proof. By the result in [18], $\text{PSPACE} \subseteq \text{PMC}_{\text{CD}_{ne}\text{C}(3)}$.

Then $\text{PMC}_{\text{CD}_{ne}\text{C}(3)} \subseteq \text{PMC}_{\text{CD}_{ne}\text{C}} \subseteq \text{PMC}_{\text{CD}_{ne}\text{C}}^*$ by proposition 1.

Finally, Theorem 1 gives $\text{PMC}_{\text{CD}_{ne}\text{C}}^* \subseteq \text{PSPACE}$. \square

4.2 The case of symport/antiport P systems with membrane separation

We now consider P systems with symport/antiport that use membrane separation and not membrane division. As previously said, in this case, the content of the membrane being divided is distributed between the two new membranes obtained instead of being duplicated. Moreover, only elementary membranes can be separated. We show that, even in this case, the class **PSPACE** represents an upper bound.

Theorem 2. $\text{PMC}_{\text{CSC}}^* \subseteq \text{PSPACE}$

Proof. The proof follows the same line as the one of Theorem 1, but there are a few differences we need to consider, as only elementary membranes can separate. We designed a new algorithm, where the function **State** is replaced by **State_CSC** and is updated to call the new function **ApplySeparationRule_CSC**. This function is

similar to [ApplyDivisionRule](#) but we only need one multiset S since, as we said only elementary membranes can separate, so there is no inner membrane that could use the objects. For the same reason, it is not necessary to keep a matrix to identify a membrane. A n -tuple $(i_1, \dots, i_n) \in \{1, 2\}^n$ is associated with membrane h at step n , where $i_1 = \dots = i_n = 1$ if h is not elementary, otherwise an index i_m indicates that h is issued from the i_m -th membrane produced by the separation of $h_{(i_1, \dots, i_{(m-1)})}$ at step m .

The maximal number of membranes in the P system at step n is q^{2^n} , with q being the number of initial membranes, that is less than in the case of membrane division ($q^{2^{nd}}$), so the maximal number of objects o'_n in the P system with membrane separation is bounded by o_n . The number b'_n of bits required to store the objects in a membrane is then (as b_n) a polynomial in n whose coefficients are polynomial in $|x|$.

In the function [ParentSimulation_CSC](#), since only elementary membranes can separate, we removed the use of [ApplySeparationRule_CSC](#) for the parent membranes of h . So the case $i = k$ had been pulled out of the loop **foreach**. Moreover, we now check if a membrane is elementary before executing [ApplySeparationRule_CSC](#) on it (for h or its children). These modifications allow to follow the symport/antiport P system with membrane separation computation process.

Denoting $S'(n)$ the space used by the new function **State** when called with the parameter n . Then, with c polynomial in $|x|$:

$$\begin{aligned} S'(0) &= c \\ S'(n) &\leq n + c + 4b'_n + S'(n-1) \\ &\leq \mathcal{O}(n + c + b'_n) + S'(n-1) \\ &\leq \mathcal{O}(n^2 + nc + nb'_n) \end{aligned}$$

Then, by the same reasoning as in the proof of Theorem 1, $\text{PMC}_{\text{CSC}}^* \subseteq \text{PSPACE}$.

□

5 Characterization of $\mathbf{P}^{\#\mathbf{P}}$ by symport/antiport P systems with separation rules

In this section we obtain a characterization of $\mathbf{P}^{\#\mathbf{P}}$ showing that symport/antiport P systems with membrane separation can solve in polynomial time the **LEXICAL_MIDDLE_3SAT** $\mathbf{P}^{\#\mathbf{P}}$ -complete problem. I found this result one week before the end of my internship, so I didn't had enough time to describe and prove formally the P system solving **LEXICAL_MIDDLE_3SAT**.

Definition 12. The **LEXICAL_MIDDLE_3SAT** problem is defined as follows in [23]:

Input: A boolean formula $\phi(x_1, \dots, x_n)$ in 3CNF (conjunction of clauses of three literals)

Question: Is $x_n = 1$ in the lexicographically middle satisfying assignment $x_1 \dots x_n \in \{0, 1\}^n$ of ϕ ?

The question is to determine the truth value of x_n in the satisfying assignment of ϕ which is the median among all the satisfying assignments of ϕ ordered in the lexicographic order. This problem is $\mathbf{P}^{\#\mathbf{P}}$ -complete ([23]).

Example 3. With the boolean formula $\phi(x_1, x_2, x_3) = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$, the satisfying assignments are $(0, 0, 0); (0, 1, 1); (1, 0, 1); (1, 1, 0); (1, 1, 1)$, ordered in the lexicographic order. Then the median is $(1, 0, 1)$ and the answer to the problem is **yes**.

When an even number of assignments satisfy the formula, the lowest of the two median assignments is chosen as the median.

Theorem 3. **LEXICAL_MIDDLE_3SAT** $\in \text{PMC}_{\text{CSC}}$

Sketch of a proof for Theorem 3. The objective of this proof is to construct a polynomially uniform family of symport/antiport P systems with membrane separation which solves the **LEXICAL_MIDDLE_3SAT** decision

problem in a uniform way. We won't describe entirely the rules of the P system and we don't prove formally its functioning.

The P system we create is an evolution of the one used to solve the **SAT** problem in [24]. This one runs in three phases: the generation phase, where 2^n membranes are created with the separation rule, each one representing an assignment of the n variables of the formula ; the checking phase consists in the evaluation of the formula in each membrane ; finally, the output phase answers **yes** or **no** depending on whether a membrane satisfies the formula.

The system of [24] has initially three membranes: the skin is the membrane 1, the membrane 2 that will separate to obtain 2^n membranes 2 and membrane 3, used during the output phase. The rules have a maximum length of 3 objects, but for our result we accept rules of polynomial length.

We re-use this system until the end of the checking phase. We delete the membrane 3, and the rules associated with. We replace the rules $(e_{i,m}E_0, out), (\bar{e}_{i,m}E_0, out) \in \mathcal{R}_2$ by the rules $(e_{i,m}, out; \sigma^n, in), (\bar{e}_{i,m}, out; \sigma^n, in) \in \mathcal{R}_2$ of length $n + 1$. With these changes, in the configuration obtained after the step $3n + 2m + 1$, only the membranes 2 having an assignment satisfying the formula contain the object σ with multiplicity n .

Then, we add three phases so the computation solve **LEXICAL_MIDDLE_3SAT**:

- The *activation phase* makes the membranes 2 satisfying the formula able to interact during the next phase.
- The *search phase* searches for the lexicographically middle satisfying assignment, determining the values of the variables, one by one. The determination of a variable needs three sub-phases:
 - Preparation sub-phase: the preparation cannot be done before in parallel because it needs the result of the previous determination phase.
 - Comparison sub-phase: all the satisfying assignments are compared to a well-chosen assignment.
 - Determination sub-phase: depending on the number of assignments lower or greater than the one they have been compared with, the variable of the median assignment is determined.
- The *output phase* returns **yes** or **no** whether $x_n = 1$ or $x_n = 0$ in the assignment found previously.

Let's develop what the phases do.

Activation phase This phase lasts for only one step and uses the rules $(\sigma T_i, out; T'_i, in), (\sigma F_i, out; F'_i, in) \in \mathcal{R}_2$, with $1 \leq i \leq n$ to bring in the membranes satisfying ϕ the objects T'_i and F'_i that will interact in the next phases.

Our reader could remark that the objects σ, T'_i and F'_i haven't ever been sent from the environment to membrane 1. This can be easily done by adding rules using a global counter of steps to insure the objects will be present in enough quantities at the exact step in which there are needed.

Search phase This phase determines the values of the variables x_1, \dots, x_n of the lexicographically middle satisfying assignment. The P system will repeat n times the three sub-phases, finding the value of one variable at each loop. The variables are determined in order, from x_1 to x_n .

This phase does the same thing as a research of the median of a set of values between 0 and $2^n - 1$. The idea is to determine the bits of the median, from the most to the least significant bit, by counting the number of values greater and lower than a current well-chosen number. This process is very close to a binary search: we first consider $10 \dots 0 = 2^{n-1}$, if there are strictly more numbers, greater or equal than $10 \dots 0$, then the most significant bit is 1, else 0. To find the i -th most significant bit x_i , we must consider $x_1 \dots x_{i-1} 10 \dots 0$. After n iteration, we obtain the median.

Let's describe the i -th iteration of the sub-phases, in which x_i is determined ($1 \leq i \leq n$). Preparation and comparison sub-phases last $n + 1$ steps each, determination sub-phase lasts 3 step. Thus, the search phase lasts $n(n + 1 + n + 1 + 3) = 2n^2 + 5n$ steps.

Preparation sub-phase i At the beginning of this step, the values of x_1, \dots, x_{i-1} are coded by the presence of μ_j if $x_j = 1$ and $\bar{\mu}_j$ if $x_j = 0$ in membrane 1 ($1 \leq j \leq i-1$). The objective of this phase is to bring 2^n occurrences of the objects $\check{\chi}_{i,1} \cdots \check{\chi}_{i,i-1} \chi_{i,i} \bar{\chi}_{i,i+1} \cdots \bar{\chi}_{i,n}$ representing $x_1 \cdots x_{i-1} 10 \cdots 0$, where $\check{\chi}_{i,j} = \chi_{i,j}$ or $\bar{\chi}_{i,j}$ whether $x_j = 1$ or 0 ($1 \leq j \leq i-1$). These objects encode the reference assignment we want to compare to all satisfying assignments.

This sub-phase must also introduce 2^n occurrences of the objects $\{l_{i,j}^v, q_{i,j}^v, g_{i,j}^v\}_{1 \leq j \leq n, v \in \{t,f\}}$ that will be used in the next sub-phase.

Comparison sub-phase i During these phase, each membrane 2 satisfying the formula ϕ will compare their assignment (stored by the objects T'_j or F'_j , $1 \leq j \leq n$) to the reference assignment $x_1 \cdots x_{i-1} 10 \cdots 0$ coded in membrane 1. The comparison is in the lexicographic order, we first compare T'_1/F'_1 with $\chi_{i,1}/\bar{\chi}_{i,1}$. An object $l/q/g_{i,1}^{t/f}$ is sent in the membrane 2 to indicate if after the comparison with the first bit, the membrane assignment is lower (l), equal (q) or greater (g) than the reference. The exponent t/f is used to know which among T'_1 and F'_1 must be re-sent to the membrane 2. We don't write here the whole set of rules needed to do the comparison.

The comparison continues with the other bits. In the last step, the object $l/q/g_{i,n}^{t/f}$ is sent in membrane 1 while the bit T'_n/F'_n is sent to the membrane 2 it comes from. All objects $l/q/g_{i,n}^{t/f}$ present before in the membrane 1 must have been sent to the environment before, thanks to a counter.

Determination sub-phase i This phase will count the number of satisfying assignments greater or equal than the reference. If they are strictly more, than the ones strictly lower than the reference, then the system must bring the object μ_i from the environment to the membrane 1.

To count, we first exchange the objects $q, g_{i,n}^{t/f}$ (from membrane 1 to the environment) with objects G_i (from the environment to membrane 1), and the objects $l_{i,n}^{t/f}$ with objects L_i . At the next step, the objects G_i and L_i annihilate with the rule $(L_i G_i, out) \in \mathcal{R}_1$. Finally, if there are objects G_i remaining, and with the help of a counter, the object μ_i enters membrane 1. If not, it's an object $\bar{\mu}_i$ that enters membrane 1.

The preparation sub-phase $i+1$ (or the output phase if $i=n$) can now begin.

Output phase At the end of the previous phase, membrane 1 contains μ_n or $\bar{\mu}_n$ depending on whether $x_n = 1$ or 0 in the lexicographically middle satisfying assignment. This phase lasts only one step, were one of these two rules (**yes** $\mu_n \xi, out$), (**no** $\bar{\mu}_n \xi, out$) $\in \mathcal{R}_1$ is applied. The object ξ must appear in membrane 1 during the previous step, if it was present previously, the rules could have been applied before. The objects **yes** and **no** are present in membrane 1 since the beginning of the computation. After this step, no rules can be applied yet, and the environment contains either **yes** or **no**, according to the value of x_n in the lexicographically middle satisfying assignment.

Finally, after $2n^2 + 8n + 2m + 3$ steps, the P system we have constructed solves **LEXICAL_MIDDLE_3SAT**. The system needs a polynomial number of rules of polynomial length, and a polynomial number of distinct objects in Γ so it can be constructed in polynomial time of the length of ϕ .

Thus, we have **LEXICAL_MIDDLE_3SAT** \in **PMC_{CSC}**.

□

It would be interesting to implement this P system in **MeCoSim**, the application of the P system simulator **P-Lingua** for symport/antiport P systems [6]. The authors of [24] provide the implementation of their solution of **SAT** on the [MeCoSim website](#).

Corollary 2. **P^{#P} = PMC_{CSC}**

Proof. From Theorem 3 and because **LEXICAL_MIDDLE_3SAT** is **P^{#P}**-complete (result obtained in [23]), the following inclusion holds: **P^{#P} \subseteq PMC_{CSC}**.

Then by definition of the cell-like and tissue symport/antiport P system with membrane separation, we have $\mathbf{PMC}_{\text{CSC}} \subseteq \mathbf{PMC}_{\text{TSC}}$.

Finally, the article [2] shows the characterization $\mathbf{PMC}_{\text{TSC}} = \mathbf{P}^{\#\mathbf{P}}$.

The corollary comes out from these inclusions. □

6 Conclusion

During my internship, I discovered the scientific field of Membrane Computing. I was able to work on the characterizations of some classes of symport/antiport P systems.

We first proved a conjecture stated in [20], about the characterization of the complexity class **PSPACE** by means of cell-like symport/antiport P systems with membrane division. In particular, we provided and analyzed an algorithm allowing us to state that **PSPACE** is an upper bound for the class of problems solved by such systems in a polynomial number of steps. This result, together with the opposite inclusion proved in [18], provides the characterization.

We then adapt the solution of **SAT** in [24] to get a cell-like symport/antiport P systems with membrane separation solving the **LEXICAL_MIDDLE_3SAT** problem. Because this problem is $\mathbf{P}^{\#\mathbf{P}}$ -complete, we obtained the inclusion $\mathbf{P}^{\#\mathbf{P}} \subseteq \mathbf{PMC}_{\text{CSC}}$. The results of the State of the Art bring a characterization of $\mathbf{P}^{\#\mathbf{P}}$ as the set of all problems solved in polynomial time by a polynomially uniform family of cell-like symport/antiport P systems with membrane separation.

This result implies that the computational power of cell-like and tissue symport/antiport P system with membrane separation running in polynomial time are the same. Therefore, there must be a way to simulate such a tissue system by a cell-like one. We have unsuccessfully tried this approach to get the characterization. Some inclusion results have been tackled by constructing a system from another, by example in [2] they simulate the division of a deterministic tissue P system with a tissue P system with separation rules.

As the results on (non-)efficiency of the variants of P systems bring some ideas to tackle the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ unsolved problem, there is a hope to approach the $\mathbf{PSPACE} \stackrel{?}{=} \mathbf{P}^{\#\mathbf{P}}$ question by characterizing **PSPACE** and $\mathbf{P}^{\#\mathbf{P}}$ with some new models of computation.

References

- [1] Rosa Gutiérrez-Escudero, Mario J. Pérez-Jiménez, and Miquel Rius-Font. Characterizing tractability by tissue-like P systems. In Gheorghe Păun, Mario J. Pérez-Jiménez, Agustín Riscos-Núñez, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, pages 289–300, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [2] Alberto Leporati, Luca Manzoni, Giancarlo Mauri, Antonio E. Porreca, and Claudio Zandron. Characterising the complexity of tissue P systems with fission rules. *Journal of Computer and System Sciences*, 90:115–128, 2017.
- [3] Alberto Leporati, Luca Manzoni, Giancarlo Mauri, Antonio E. Porreca, and Claudio Zandron. Characterizing PSPACE with shallow non-confluent P systems. *Journal of Membrane Computing*, 1:75–84, 2019.
- [4] Luis Felipe Macías Ramos, Bosheng Song, Tao Song, Linqiang Pan, and Mario de Jesús Pérez Jiménez. Limits on efficient computation in P systems with symport/antiport rules. *BWMC 2017: 15th Brainstorming Week on Membrane Computing (2017)*, p 147-160, 2017.
- [5] Luis Macías-Ramos, Bosheng Song, Luis Valencia-Cabrera, Linqiang Pan, and Mario Pérez-Jiménez. Membrane fission: A computational complexity perspective. *Complexity*, 21, 04 2015.

- [6] Luis Macías-Ramos, Luis Valencia-Cabrera, Bosheng Song, Tao Song, Linqiang Pan, and Mario Pérez-Jiménez. A P-Lingua based simulator for P systems with symport/antiport rules. *Fundamenta Informaticae*, 139:211–227, 07 2015.
- [7] David Orellana-Martín, Miguel À. Martínez del Amor, Luis Valencia-Cabrera, Bosheng Song, Linqiang Pan, and Mario J. Pérez-Jiménez. P systems with symport/antiport rules: When do the surroundings matter? *Theoretical Computer Science*, 805:206–217, 2020.
- [8] Linqiang Pan, Bosheng Song, and Claudio Zandron. On the computational efficiency of tissue P systems with evolutionary symport/antiport rules. *Knowledge-Based Systems*, 262:110266, 2023.
- [9] Andrei Păun and Gheorghe Păun. The power of communication: P systems with symport/antiport. *New Generation Comput.*, 20:295–306, 09 2002.
- [10] Gheorghe Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
- [11] Gheorghe Păun. P systems with active membranes: Attacking NP complete problems. *Journal of Automata, Languages and Combinatorics*, 6:75–90, 01 2001.
- [12] Gheorghe Păun. *Introduction to Membrane Computing*, chapter 1, pages 1–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [13] Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., USA, 2010.
- [14] Mario J. Pérez-Jiménez. A computational complexity theory in membrane computing. In Gheorghe Păun, Mario J. Pérez-Jiménez, Agustín Riscos-Núñez, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, pages 125–148, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [15] Mario J. Pérez-Jiménez and Petr Sosík. An optimal frontier of the efficiency of tissue p systems with cell separation. *Fundam. Informaticae*, 138:45–60, 2015.
- [16] Antonio E Porreca, Niall Murphy, and Mario de Jesús Pérez Jiménez. An optimal frontier of the efficiency of tissue P systems with cell division. *Proceedings of the Tenth Brainstorming Week on Membrane Computing, (2) 141-166. Sevilla, ETS de Ingeniería Informática, 30 de Enero-3 de Febrero, 2012.,* 2012.
- [17] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*. Springer, 1997.
- [18] Bosheng Song, Mario J. Pérez-Jiménez, and Linqiang Pan. Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division. *Biosystems*, 130:51–58, 2015.
- [19] Bosheng Song and Xiangxiang Zeng. Solving a PSPACE-complete problem by symport/antiport P systems with promoters and membrane division. *Journal of Membrane Computing*, 3, 12 2021.
- [20] Petr Sosík. P systems attacking hard problems beyond NP: a survey. *Journal of Membrane Computing*, 1, 09 2019.
- [21] Petr Sosík, Andrei Păun, and Alfonso Rodríguez-Patón. P systems with proteins on membranes characterize PSPACE. *Theoretical Computer Science*, 488(1):78–95, 2013.
- [22] Petr Sosík and Alfonso Rodríguez-Patón. Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73(1):137–152, 2007.
- [23] Seinosuke Toda. Simple characterizations of $P(\#P)$ and complete problems. *Journal of Computer and System Sciences*, 49(1):1–17, 1994.
- [24] Luis Valencia-Cabrera, Bosheng Song, Luis F. Macías-Ramos, Linqiang Pan, Agustín Riscos-Núñez, and Mario J. Pérez-Jiménez. Computational efficiency of P systems with symport/antiport rules and membrane separation. *Thirteenth Brainstorming Week on Membrane Computing*, 1:325–370, 02/2015 2015.

Appendices

A Simulation algorithms for symport/antiport P systems

One can find here the algorithms from Section 4.

A.1 P systems with membrane division

Algorithm 1: $\text{State}(h, I, k, n)$

```
Data:  $P, Y$ 
//  $P + Y$  is a multiset of objects in a membrane.
//  $P$  contains the objects that can be used by the division or communication rules,
//  $Y$  contains the ones that can't because they are already used.
if  $k = 0$  then return  $\mathcal{E}$ ;
if  $n = 0$  then return  $m_h$ ;
 $P \leftarrow \mathcal{E}$ ;
 $Y \leftarrow \emptyset$ ;
ParentSimulation( $h, I, k, n, P, Y$ );
ContributionFromChildren( $h, I, k, n, P, Y$ );
return  $P + Y$ ;
```

 $\text{ParentSimulation}(h, I, k, n, P, Y)$ // P and Y passed by reference

```
// Simulation of step  $n$  from the skin to the membrane  $h$ 
Data:  $S, X, \text{divided}$ 
foreach  $i \in \llbracket 0, k \rrbracket$  do
   $S \leftarrow \text{State}(p^{k-i}(h), I|_{i,n-1}, i, n-1)$ ;
   $X \leftarrow \emptyset$ ;
  if  $S = \text{null}$  then return null; // The membrane does not exist
  if  $p^{k-i}(h) \neq \text{skin}$  then
     $\text{divided} \leftarrow \text{ApplyDivisionRule}(p^{k-i}(h), S, X, I_{i,n})$ 
  else
     $\text{divided} \leftarrow \text{false}$ 
  if not divided then
    if  $I_{i,n} = 2$  then return null; // The membrane  $p^{k-i}(h)$  must have been divided
     $\text{ApplyCommunicationRules}(p^{k-i}(h), S, X, P, Y)$ 
   $P, Y \leftarrow S, X$ ;
```

ContributionFromChildren(h, I, k, n, P, Y)

// P and Y passed by reference

// Simulation of step n for every child membrane g of h

Data: $S, X, divided, I'$

foreach g such as $p(g) = h$ in μ **do**

foreach $(i_{k+1,1}, \dots, i_{k+1,n-1}) \in \{1,2\}^{n-1}$ **do**

$I' \leftarrow \begin{pmatrix} I|_{k,n-1} \\ i_{k+1,1} \cdots i_{k+1,n-1} \end{pmatrix};$

$S \leftarrow \text{State}(g, I', k+1, n-1);$

$X \leftarrow \emptyset;$

if not $(S = \text{null})$ **then**

$divided \leftarrow \text{ApplyDivisionRule}(g, S, X, 1);$

if not $divided$ **then**

$\text{ApplyCommunicationRules}(g, S, X, P, Y);$

ApplyDivisionRule(h, S, X, i)

// S and X passed by reference

foreach rule of the form $[a]_h \rightarrow [b]_h[c]_h \in \mathcal{R}_h$ **do**

if $a \in S$ **then**

$S \leftarrow S - \{a\};$

if $i = 1$ **then** $X \leftarrow X + \{b\};$

if $i = 2$ **then** $X \leftarrow X + \{c\};$

return $True;$

return $False;$

ApplyCommunicationRules(h, S, X, P, Y)

// S, X, P and Y passed by reference

foreach rule of the form $(u, out) \in \mathcal{R}_h$ **do**

while $u \subseteq S$ **do**

$S \leftarrow S - u;$

$Y \leftarrow Y + u;$

foreach rule of the form $(u, in) \in \mathcal{R}_h$ **do**

while $u \subseteq P$ **do**

$P \leftarrow P - u;$

$X \leftarrow X + u;$

foreach rule of the form $(u, out; v, in) \in \mathcal{R}_h$ **do**

while $u \subseteq S$ and $v \subseteq P$ **do**

$S \leftarrow S - u;$

$Y \leftarrow Y + u;$

$P \leftarrow P - v;$

$X \leftarrow X + v;$

A.2 P systems with membrane separation

Algorithm 2: $\text{State_CSC}(h, (i_1, \dots, i_n), k, n)$

Data: P, Y
 // $P + Y$ is a multiset of objects in a membrane.
 // P contains the objects that can be used by the division or communication rules,
 // Y contains the ones that can't because they are already used.
if $k = 0$ **then return** \mathcal{E} ;
if $n = 0$ **then return** m_h ;
 $P \leftarrow \mathcal{E}$;
 $Y \leftarrow \emptyset$;
 ParentSimulation_CSC($h, (i_1, \dots, i_n), k, n, P, Y$);
 ContributionFromChildren_CSC(h, k, n, P, Y);
return $P + Y$;

ParentSimulation_CSC($h, (i_1, \dots, i_n), k, n, P, Y$) // P and Y passed by reference

// Simulation of step n from the *skin* to the membrane $p(h)$
Data: S, X , *separated*
foreach $i \in \llbracket 0, k - 1 \rrbracket$ **do**
 | $S \leftarrow \text{State_CSC}(p^{k-i}(h), (1, \dots, 1), i, n - 1)$;
 | $X \leftarrow \emptyset$;
 | **if** $S = \text{null}$ **then return null**; // The membrane does not exist
 | ApplyCommunicationRule_CSC($p^{k-i}(h), S, X, P, Y$);
 | $P, Y \leftarrow S, X$;
 // Simulation of step n of the membrane $p(h)$
 $S \leftarrow \text{State_CSC}(h, (i_1, \dots, i_{n-1}), k, n - 1)$;
 $X \leftarrow \emptyset$;
if $S = \text{null}$ **then return null**; // The membrane does not exist
if $h \neq \text{skin}$ and h is elementary **then**
 | *separated* $\leftarrow \text{ApplySeparationRule_CSC}(h, S, I_{i,n})$
else
 | *separated* $\leftarrow \text{false}$
if *not separated* **then**
 | **if** $i_n = 2$ **then return null**; // The membrane h must have been separated
 | ApplyCommunicationRule_CSC($p^{k-i}(h), S, X, P, Y$)
 $P, Y \leftarrow S, X$;

ContributionFromChildren_CSC(h, k, n, P, Y) // P and Y passed by reference

// Simulation of step n for every child membrane g of h

Data: $S, X, separated, (j_1, \dots, j_{n-1}) \in \{1, 2\}^{n-1}$

foreach g such as $p(g) = h$ in μ **do**

foreach $(j_1, \dots, j_{n-1}) \in \{1, 2\}^{n-1}$ **do**

$S \leftarrow \text{State_CSC}(g, (j_1, \dots, j_{n-1}), k + 1, n - 1)$;

$X \leftarrow \emptyset$;

if not ($S = \text{null}$) **then**

if g is elementary **then**

$separated \leftarrow \text{ApplySeparationRule_CSC}(g, S, 1)$

else

$separated \leftarrow \text{false}$

if not separated then

$\text{ApplyCommunicationRule_CSC}(g, S, X, P, Y)$;

ApplySeparationRule_CSC(h, S, i) // S passed by reference

foreach rule of the form $[a]_h \rightarrow [\Gamma_1]_h[\Gamma_2]_h \in \mathcal{R}_h$ **do**

if $a \in S$ **then**

$S \leftarrow S - \{a\}$;

if $i = 1$ **then** $S \leftarrow \{x \in S \mid x \in \Gamma_1\}$;

if $i = 2$ **then** $S \leftarrow \{x \in S \mid x \in \Gamma_2\}$;

return *True*;

return *False*;

ApplyCommunicationRule_CSC(h, S, X, P, Y) // S, X, P and Y passed by reference

foreach rule of the form $(u, out) \in \mathcal{R}_h$ **do**

while $u \subseteq S$ **do**

$S \leftarrow S - u$;

$Y \leftarrow Y + u$;

foreach rule of the form $(u, in) \in \mathcal{R}_h$ **do**

while $u \subseteq P$ **do**

$P \leftarrow P - u$;

$X \leftarrow X + u$;

foreach rule of the form $(u, out; v, in) \in \mathcal{R}_h$ **do**

while $u \subseteq S$ and $v \subseteq P$ **do**

$S \leftarrow S - u$;

$Y \leftarrow Y + u$;

$P \leftarrow P - v$;

$X \leftarrow X + v$;