Research Article
From NP-Completeness to DP-Completeness: A Membrane Computing Perspective

Luis Valencia-Cabrera, David Orellana-Martín, Miguel Á. Martinez-del-Amor, Ignacio Pérez-Hurtado, and Mario J. Pérez-Jiménez

Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Correspondence should be addressed to David Orellana-Martín; dorellana@us.es

Received 23 January 2020; Revised 21 July 2020; Accepted 28 July 2020; Published 30 August 2020

Academic Editor: Jose C. Valverde

Copyright © 2020 Luis Valencia-Cabrera et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Presumably efficient computing models are characterized by their capability to provide polynomial-time solutions for NP-complete problems. Given a class $\mathcal{R}$ of recognizer membrane systems, $\mathcal{R}$ denotes the set of decision problems solvable by families from $\mathcal{R}$ in polynomial time and in a uniform way. $\text{PMC}_\mathcal{R}$ is closed under complement and under polynomial-time reduction. Therefore, if $\mathcal{R}$ is a presumably efficient computing model of recognizer membrane systems, then $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_\mathcal{R}$. In this paper, the lower bound $\text{NP} \cup \text{co-NP}$ for the time complexity class $\text{PMC}_\mathcal{R}$ is improved for any presumably efficient computing model $\mathcal{R}$ of recognizer membrane systems verifying some simple requirements. Specifically, it is shown that $\text{DP} \cup \text{co-DP}$ is a lower bound for such $\text{PMC}_\mathcal{R}$, where $\text{DP}$ is the class of differences of any two languages in $\text{NP}$. Since $\text{NP} \cup \text{co-NP} \subseteq \text{DP} \cap \text{co-DP}$, this lower bound for $\text{PMC}_\mathcal{R}$ delimits a thinner frontier than that with $\text{NP} \cup \text{co-NP}$.

1. Introduction

Membrane Computing is a computing paradigm inspired by some basic biological features. Specifically, it is inspired by the structure and functioning of the living cells, as well as from the cooperation of cells in tissues, organs, and organisms. This paradigm provides distributed, parallel non-deterministic computing models whose computational devices are generically called membrane systems or P systems and has processing units called compartments. In this framework, there basically exist two ways to consider computational devices: cell-like membrane systems and tissue-like membrane systems. The former uses the biological membranes arranged hierarchically as compartments, inspired by the structure of the cell. The latter takes as compartments the cells placed in the nodes of a directed graph, which is inspired by the cell intercommunication in tissues.

Recognizer membrane systems were introduced in [1], in order to provide a natural framework in Membrane Computing, aiming to solve decision problems by means of language recognition. A recognizer membrane system has the following additional syntactic and semantic peculiarities: (a) the working alphabet has two distinguished objects (yes and no); (b) there exist an input compartment, and there is also an input alphabet strictly contained in the working alphabet; (c) the initial content of each compartment is a multiset of objects from the working alphabet not belonging to the input alphabet; (d) all computations halt; and (e) for each computation and only at its last step, either object yes or object no (but not both) must have been released to the environment. Recognizer membrane systems can only have accepting or rejecting computations, depending on whether they return yes or no to the environment, respectively.

On this basis, given a recognizer membrane system $\Pi$, for each multiset $m$ over the input alphabet, we denote by $\Pi + m$ the membrane system obtained from $\Pi$ by adding the multiset $m$ to the content of the input membrane at the initial configuration. Unlike a Turing machine, all the
elements that make up a recognizer membrane system have a finite description. Therefore, while a decision problem (with infinite instances) can be solved by a single Turing machine, an infinite family of recognizer membrane systems is necessary to solve it. We say that a family $\Pi = \{\Pi(n) | n \in \mathbb{N}\}$ of recognizer membrane systems solves a decision problem $X$ in polynomial time in a uniform way if such family $\Pi$ can be generated by a deterministic Turing machine working in polynomial time, and there exists a pair $(\text{cod}, s)$ of polynomial-time computable functions over the set of instances of $X$ verifying the following: (a) for each instance $u \in I_X$, $s(u)$ is a natural number and $\text{cod}(u)$ is an input multiset of the system $\Pi(s(u))$; (b) for each $n \in \mathbb{N}$, the set $s^{-1}(\{n\})$ is a finite set; and (c) the family $\Pi$ is polynomially bounded, sound, and complete with regard to $(X, \text{cod}, s)$. The pair $(\text{cod}, s)$ is called a polynomial encoding from the decision problem $X$ to the family $\Pi$. For more details, see [2]. Given a computing model $\mathcal{R}$ of recognizer membrane systems, $\text{PMC}_{\mathcal{R}}$ denotes the set of decision problems solvable by families from $\mathcal{R}$ in polynomial time in a uniform way. The class $\text{PMC}_{\mathcal{R}}$ is closed under complement and under polynomial-time reduction [2].

A computing model is said to be efficient (respectively, presumably efficient) if it has the ability to provide polynomial-time solutions for intractable problems (resp., $\text{NP}$-complete problems) [3]. The term presumably efficient refers to the fact that, as generally believed, if $\text{P} \neq \text{NP}$ then each $\text{NP}$-complete problem is intractable and, consequently, any presumably efficient model would be efficient. The computing model $\text{M}_{\mathcal{R}}$ of recognizer P systems with active membranes and electrical charges is presumably efficient [4], as well as the computing model $\text{M}_{\mathcal{R}}^0(+d, +ve)$ of recognizer polarizationless P systems with active membranes which makes use of dissolution rules and division for elementary and nonelementary membranes [5]. However, the computing model $\text{M}_{\mathcal{R}}^0(-d)$ of recognizer polarizationless P systems with active membranes not making use of dissolution rules is not efficient, even when allowing division rules for elementary and nonelementary membranes [5]. Thus, if $\mathcal{R}$ is a presumably efficient computing model of recognizer membrane systems, then $\text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\mathcal{R}}$.

Several variants of P systems have been demonstrated to be presumably efficient. A solution to QSAT has been given in the framework of P systems with active membranes and division rules for elementary and nonelementary membranes but only evolution rules of the RHS limited to length 1 [6]. When creation rules are used instead of division rules as a method of generating an exponential workspace in polynomial time, it is possible to get rid of polarizations to obtain a solution to the QSAT problem [7]. A new research line taking tissue P systems with synport/antiport rules where objects can evolve during the application of the rules has been developed recently. Several results have been obtained, and new solutions to $\text{NP}$-complete problems have been given in this framework [8].

The main contribution of this paper is to provide a lower bound for $\text{PMC}_{\mathcal{R}}$ delimiting a frontier with such class that is thinner to the one found with respect to $\text{NP} \cup \text{co-NP}$, with $\mathcal{R}$ being a presumably efficient computing model of recognizer membrane systems verifying some simple requirements (e.g., in the case of cell-like membrane systems, allowing object evolution rules, dissolution rules, and communication rules). It is also worth pointing out that the methodology used to obtain this result is constructive in the following sense: given an $\text{NP}$-complete problem $X$, a $\text{DP}$-complete problem $Y_X$ is associated, in such a manner that if $X \in \text{PMC}_{\mathcal{R}}$ then $Y_X \in \text{PMC}_{\mathcal{R}}$, for such kind of presumably efficient computing model $\mathcal{R}$ of recognizer membrane systems.

The remainder of this paper is structured as follows. Section 2 will be devoted to introduce some concepts that can be useful to understand the rest of the work. The complexity class $\text{DP}$ is introduced in Section 3, where the main theorem presented in this paper is established. In Section 4, the methodology will be applied with an example. Finally, in Section 5, the most relevant conclusions are drawn, and some possible further developments are outlined.

## 2. Preliminaries

In order for the paper to be self-contained, several basic concepts and results will be introduced to facilitate a better understanding of the text.

A language is a nonempty set whose elements are called symbols. A string $u$ over an alphabet is an ordered finite sequence of symbols. The length of a string $u$, denoted by $|u|$, is the number of occurrences of symbols that it contains. The empty string (with length 0) contains no symbols, and it is denoted by $\lambda$. The set of all strings over an alphabet $\Gamma$ is denoted by $\Gamma^*$. A language over $\Gamma$ is a subset of $\Gamma^*$. If $L_1, L_2$ are languages over $\Gamma$, the set-theoretic difference of $L_1$ and $L_2$ (in this order), denoted by $L_1 \setminus L_2$, is defined as follows:

$$(L_1 \setminus L_2) = \{u \in \Gamma^* | u \in L_1 \land u \notin L_2\}.$$

A multiset $m$ over an alphabet $\Gamma$ is a mapping from $\Gamma$ onto the set $\mathbb{N}$ of natural numbers. A multiset $m$ over $\Gamma$ is finite (respectively, empty) if the set $\{x \in \Gamma \mid m(x) > 0\}$ is a finite (respectively, empty) set. If $m, n'$ are multisets over $\Gamma$, then the union of $m$ and $n'$, denoted by $m + n'$, is the multiset over $\Gamma$ defined as follows: $(m + n')(x) = m(x) + n'(x)$, for each $x \in \Gamma$.

### 2.1. The Cantor Pairing Function

The Cantor pairing function is the mapping from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$ defined as follows: the natural number $[(t_1 + t_2) \cdot (t_1 + t_2 + 1)/2] + t_1$ is associated with every ordered pair $(t_1, t_2)$ of natural numbers. We denote it by $\langle t_1, t_2 \rangle$, that is, $\langle t_1, t_2 \rangle = [(t_1 + t_2) \cdot (t_1 + t_2 + 1)/2] + t_1$. The Cantor pairing function is a primitive recursive function and bijective from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$. Consequently, for each natural number $t$, there exists a unique pair of natural numbers $t_1$ and $t_2$ such that $t = \langle t_1, t_2 \rangle$.

### 2.2. Decision Problems and Languages

A decision problem $X$ is an ordered pair $(I_X, \theta_X)$, where $I_X$ is a language over a finite alphabet $\Sigma_X$ and $\theta_X$ is a total Boolean function over $I_X$. The elements of $I_X$ are called instances of the problem $X$. 
The satisfiability problem, denoted by SAT, is the problem of determining whether or not a Boolean expression in conjunctive normal form is unsatisfiable. We say that a Boolean expression is unsatisfiable if it is in conjunctive normal form, literals are not repeated in each clause, and a literal and its negation do not appear in the same clause. We say that a Boolean expression makes the Boolean expression true.

According to the classical definition of truth tables for each \( \sigma \), \( \Theta(\sigma) \) is a mapping from \( \{0, 1\} \) onto the set \( \{0, 1, \text{false}, \text{true}\} \) according to the classical definition of truth tables for the function \( \Theta(\sigma) \). A truth assignment \( \sigma \) for a Boolean expression \( \varphi \) provides a unique Boolean value to it. We say that \( \varphi(\sigma) \) is true if \( \sigma \) makes the Boolean expression \( \varphi \) true.

A literal is a Boolean variable or the negation of a Boolean variable. A clause is a finite disjunction of literals. A propositional Boolean variable is in \( \mathsf{P} \); \( \mathsf{P}^{2k+1} \subseteq \mathsf{P} \), \( \mathsf{P}^{2k+1} \subseteq \mathsf{P}^{2k+1} \subseteq \mathsf{P} \), and \( \mathsf{P}^{2k+1} \subseteq \mathsf{P}^{2k+1} \subseteq \mathsf{P} \) for each \( k \in \mathbb{N} \). Sometimes, \( \mathsf{P} \) sets are denoted by \( \mathsf{BH}_1 \). The first level of this hierarchy is \( \mathsf{NP} \) and the second level is the set of languages which are intersections of a language in \( \mathsf{NP} \) and another one in \( \mathsf{co-NP} \).

### 3. The Complexity Class \( \mathsf{DP} \)

The complexity class \( \mathsf{DP} \) was introduced by Papadimitriou and Yannakis in 1982 [14] as follows.

**Definition 1.** A language \( L \) is in the class \( \mathsf{DP} \) if and only if there are two languages \( L_1 \) and \( L_2 \) such that \( L_1, L_2 \in \mathsf{NP} \) and \( L = L_1 \cap L_2 \) is \( \mathsf{NP} \)-complete.

The complexity class \( \mathsf{DP} \) can be easily characterized by the following property: a language \( L \) is in \( \mathsf{DP} \) if and only if there exist two languages \( L_1 \) and \( L_2 \) in \( \mathsf{NP} \) such that \( L = L_1 \cap L_2 \). Class \( \mathsf{DP} \) is the second level in the Boolean hierarchy. Besides, classes in the Boolean hierarchy can be described by finite unions of \( \mathsf{DP} \) sets (or by finite intersections of \( \mathsf{co-DP} \) sets). For more details, see [13, 15].

The complexity class \( \mathsf{DP} \) can also be expressed in terms of decision problems as follows: a decision problem \( X \) is in class \( \mathsf{DP} \) if and only if the language \( L_X \) associated with it belongs to class \( \mathsf{DP} \), that is, if there are two decision problems \( X_1 = (I_{X_1}, \theta_{X_1}) \) and \( X_2 = (I_{X_2}, \theta_{X_2}) \) such that \( L_{X_1} \) and \( L_{X_2} \) are languages in \( \mathsf{NP} \) and \( L_X = L_{X_1} \cap L_{X_2} \). It is easy to prove that \( \mathsf{NP} \cup \mathsf{co-NP} \subseteq \mathsf{DP} \cup \mathsf{co-DP} \).

Next, we define the product of two decision problems, and some properties of the new concept are analyzed.

**Definition 2.** Let \( X_1 = (I_{X_1}, \theta_{X_1}) \) and \( X_2 = (I_{X_2}, \theta_{X_2}) \) be two decision problems. The product problem of \( X_1 \) and \( X_2 \), denoted by \( X_1 \times X_2 \), is the decision problem \((I_{X_1} \times I_{X_2}, \theta_{X_1} \times \theta_{X_2})\) defined as follows: \( I_{X_1 \times X_2} = I_{X_1} \times I_{X_2} \) and for each \( (u_1, u_2) \in I_{X_1 \times X_2} \), we have \( \theta_{X_1 \times X_2}(u_1, u_2) = 1 \) if and only if \( \theta_{X_1}(u_1) = 1 \) and \( \theta_{X_2}(u_2) = 1 \).

A product problem \( Y_X \) can be naturally associated to a problem \( X \). This problem \( Y_X \) is the product problem, defined as follows: \( Y_X = (I_{Y_X}, \theta_{Y_X}) \) such that \( I_{Y_X} = \{(u_X, \)}
Proposition 1. Let \( X_1 \) and \( X_2 \) be decision problems:

(i) If \( X_1 \in \text{NP} \) and \( X_2 \in \text{co-NP} \), then \( X_1 \otimes X_2 \in \text{DP} \)

(ii) If \( X_1 \) is an \( \text{NP} \) complete problem and \( X_2 \) is a \( \text{co-NP} \) complete problem, then \( X_1 \otimes X_2 \) is a \( \text{DP} \) complete problem

(iii) If \( X_1 \) is an \( \text{NP} \) complete problem, then the product problem \( X_1 \otimes X_1 \) is a \( \text{DP} \) complete problem

Since the SAT problem is an \( \text{NP} \) complete problem, we deduce that the product problem \( \text{SAT} \otimes \text{SAT} \) is a \( \text{DP} \) complete problem. This problem is referred as the SAT-UNSAT problem [10] and can be expressed as follows: given two Boolean expressions \( \varphi_1 \) and \( \varphi_2 \) in simplified conjunctive normal forms, determine whether or not \( \varphi_1 \) is satisfiable and \( \varphi_2 \) is unsatisfiable.

4. The Methodology

The main result of this paper refers to the robustness of the complexity class \( \text{PMC}_{\mathbb{R}} \) associated with some computing models of recognizer membrane systems, with respect to the operation product of two decision problems.

Main Theorem 1. Let \( \mathcal{R} \) be a computing model of recognizer noncooperative cell-like \( P \) systems allowing dissolution rules, object evolution rules, and communication rules (send-in and send-out). If \( X_1 \) and \( X_2 \) are decision problems belonging to the time complexity class \( \text{PMC}_{\mathbb{R}} \), then \( X_1 \otimes X_2 \in \text{PMC}_{\mathbb{R}} \).

Proof. First, without loss of generality, we can assume that the semantics associated with the application of the rules of systems from \( \mathcal{R} \) is similar to \( P \) systems with active membranes, that is, at one transition step, a membrane can be a subject of only one rule of type dissolution, communication, or division, but object evolution rules are applied simultaneously with one of these rules and, moreover, in a maximal way. By using a different semantics, only few parameters that will appear in the proof should be modified.

For \( i = 1, 2 \), let \( \Pi^{(i)} = \{ \Pi^{(i)}(t) \mid t \in \mathbb{N} \} \) be a family of membrane systems from \( \mathcal{R} \) solving \( X_i \) in polynomial-time in a uniform way. Let \((\text{cod}, s_i)\) be a polynomial encoding from \( X_i \) into \( \Pi^{(i)} \). Let \( p_i(n) \) be a polynomial function such that, for each instance \( u_i \) from \( X_i \), any computation of \( \Pi^{(i)}(s_i(u_i)) + \text{cod}_i(u_i) \) performs at most \( p_i(|u_i|) \) steps. In this situation, a family \( \Pi = \{ \Pi(t) \mid t \in \mathbb{N} \} \) of membrane systems from \( \mathcal{R} \) will be defined from \( \Pi^{(1)} \) and \( \Pi^{(2)} \), in such a manner that \( \Pi \) provides a uniform and polynomial-time solution to the product problem \( X_1 \otimes X_2 \). The family \( \Pi \) will be called the product family of families \( \Pi^{(1)} \) and \( \Pi^{(2)} \), and it will be denoted by \( \Pi^{(1)} \otimes \Pi^{(2)} \).

First, a pair of polynomial-time functions \((\text{cod}, s)\) over the set of instances of \( X_1 \otimes X_2 \) is considered as follows:

(a) \( \text{cod}(u_1, u_2) = \text{cod}_1(u_1) + \text{cod}_2(u_2) \)

(b) \( s(u_1, u_2) = \langle s_1(u_1), s_2(u_2) \rangle \), for each \( (u_1, u_2) \in I_{X_1} \times I_{X_2}. \)

Hence, we want the instance \((u_1, u_2)\) to be processed by the system \( \Pi(s(u_1, u_2)) \) with input multiset \( \text{cod}(u_1, u_2) \). In the following, the membrane system \( \Pi(s_1(u_1), s_2(u_2)) \) will be explicitly defined from \( \Pi^{(1)}(s_1(u_1)) \) and \( \Pi^{(2)}(s_2(u_2)) \), for each \( (u_1, u_2) \in I_{X_1} \times I_{X_2} \), so that (a) if there exist accepting computations of the systems \( \Pi^{(1)}(s_1(u_1)) + \text{cod}_1(u_1) \) and \( \Pi^{(2)}(s_2(u_2)) + \text{cod}_2(u_2) \), then \( \theta_{\Pi};_{X_1}X_2 = 1 \) and (b) if \( \theta_{\Pi};_{X_1}X_2 = 1 \), then any computation of \( \Pi^{(1)}(s_1(u_1)) + \text{cod}_1(u_1) \) and any computation of \( \Pi^{(2)}(s_2(u_2)) + \text{cod}_2(u_2) \) are accepting ones. According to this, \((\text{cod}, s)\) will be a polynomial encoding from the product problem \( X_1 \otimes X_2 \) into the family \( \Pi(t) \). It is worth pointing out that, in fact, only membrane systems \( \Pi(t) \) such that \( t \in \text{range}(s) \) are defined, and this is enough to achieve our goal.

In what follows, we will denote \( t = \langle s_1(u_1), s_2(u_2) \rangle \), \( t_1 = s_1(u_1) \), and \( t_2 = s_2(u_2) \). Let us recall that given \( t \in \mathbb{N} \), there exists a unique pair of natural numbers \( t_1 \) and \( t_2 \) such that \( t = \langle t_1, t_2 \rangle \). The underlying idea of the construction of \( \Pi(t) \) is the following: when \( \text{cod}(u_1, u_2) \) is considered as the input multiset of this system; then, computations of \( \Pi^{(1)}(t_1) + \text{cod}_1(u_1) \) and \( \Pi^{(2)}(t_2) + \text{cod}_2(u_2) \) are simulated. The answers of these systems are placed in a membrane where the final decision is taken; accepting computations of \( \Pi(t) + \text{cod}(u_1, u_2) \) only come from accepting computations of \( \Pi^{(1)}(t_1) + \text{cod}_1(u_1) \) and \( \Pi^{(2)}(t_2) + \text{cod}_2(u_2) \).

Without loss of generality we can assume that \( \Gamma^{(i)}(t_i)[\text{yes, no}] \), for \( i = 1, 2 \), are mutually disjoint, being \( \Gamma^{(i)}(t_i) \) the working alphabet of \( \Pi^{(i)}(t_i) \). Likewise, without loss of generality, we can assume the same with the corresponding sets of labels \( \Pi^{(1)}(t_i) \) and \( \Pi^{(2)}(t_i) \). For \( 1 \leq i \leq 2 \), let \( q_i \) be the maximum value of \( |\text{cod}(u_i)| \) such that \( s_i(u_i) = t_i \), and let \( d_i \) be the depth of the input membrane \( h_{\Pi^{(i)}(t_i)} \) in the corresponding membrane structure. Let us denote \( m_i = q_i + d_i + 2 \), for \( i = 1, 2 \), and \( n = 2 + \max\{d_i + q_i + p_i(|u_i|) \mid 1 \leq i \leq 2 \} \). Systems \( \Pi^{(1)}(t_i) + \text{cod}_i(u_i) \) will be “prepared” after \( m_i + 1 \) steps, and computations of such systems will start at the next step. It will last \( p_i(|u_i|) \) transition steps. Besides, it will take \( n \) steps to have both responses prepared to let \( \Pi(t) + \text{cod}(u_1, u_2) \) give the answer.

The syntactical ingredients of \( \Pi(t) \) are the following:

(i) Input alphabet. The input alphabet is \( \Sigma(t) = \Sigma^{(1)*}(t_1) \cup \Sigma^{(2)*}(t_2) \), being \( \Sigma^{(i)*}(t_i) \) the input alphabet of \( \Pi^{(i)}(t_i) \), by replacing objects yes and no by objects yes, and \( \text{no}_i \), for \( i = 1, 2 \), respectively.

(ii) Working alphabet. The working alphabet is

\[
\Gamma^{(1)*}(t) \cup \Gamma^{(2)*}(t) \cup \Gamma^{(3)*}(t) \cup \Gamma^{(4)*}(t) \cup \Gamma^{(5)*}(t) \cup \Gamma^{(6)*}(t),
\]
where
\[
\Gamma^{(1)}(t) = \left( \Gamma^{(1)}(t_1) \cup \{\text{yes, no}\} \right) \cup \{\text{yes}_1, \text{no}_1\},
\]
\[
\Gamma^{(2)}(t) = \left( \Gamma^{(2)}(t_2) \cup \{\text{yes, no}\} \right) \cup \{\text{yes}_2, \text{no}_2\},
\]
\[
\Gamma^{(3)}(t) = \#: \text{YES, NO} \cup \{a' | a \in \Sigma(t)\},
\]
\[
\Gamma^{(4)}(t) = \left\{ a \big| 0 \leq j \leq m_1 \cap a \in \Gamma^{(1)}(t) \right\}
\cup \left\{ a \big| 0 \leq j \leq m_2 \cap a \in \Gamma^{(2)}(t) \right\},
\]
\[
\Gamma^{(5)}(t) = \left\{ a_0, a_1 \right\} \cup \left\{ \beta_j | 0 \leq j \leq n \right\},
\]
\[
\Gamma^{(6)}(t) = \left\{ y | 0 \leq j \leq \max\{m_1, m_2\} \right\},
\]

that is, objects \( a' \in \Gamma^{(1)}(t) \) and \( a_j \in \Gamma^{(4)}(t) \) are “copies” of objects \( a \in \Gamma^{(1)}(t_1) \cup \Gamma^{(2)}(t_2) \), and objects in \( \Gamma^{(5)}(t) \) and \( \Gamma^{(6)}(t) \) are counters for implementing a synchronization process.

(iii) Set of labels. The set of labels \( H(t) \) is \( H^{(1)}(t_1) \cup H^{(2)}(t_2) \) \( \cup \left\{ r_t', r_t'' \right\} \), \( H^{(1)}(t_1) \cup H^{(2)}(t_2) \), where \( r_t' \) and \( r_t'' \) are different from each other, and \( r_t' \) and \( r_t'' \) are different from \( r_t \).

(iv) Compartment structure. The membrane structure of \( \Pi(t) \) is obtained from the membrane structures \( \mu^{(1)}(t_1) \), for \( i = 1, 2 \), by adding four new nodes \( r_t', r_t'', h^{(1)}(t_1), h^{(2)}(t_2) \) such that (a) \( r_t' \) is the skin membrane of \( \Pi(t_1) \); (b) \( r_t '' \) is the father of \( r_t '' \); (c) \( r_t '' \) is the father of \( r_t '' \); (d) \( h^{(1)}(t_1) \) is a new son of the input membrane \( \mu^{(1)}(t_1) \) of \( \Pi(t_1) \), for \( i = 1, 2 \); and (e) \( h^{(2)}(t_2) \) is a new son of the input membrane \( \mu^{(2)}(t_2) \) of \( \Pi(t_2) \).

(v) Initial multisets. The initial multisets of \( \Pi(t) \) are the following: if \( h \) is a label of a membrane from \( \Pi^{(1)}(t_1) \) whose initial multiset \( \mathcal{M}_h^{(1)}(t_1) \) associated with \( h \) is nonempty, then the initial multiset \( \mathcal{M}_h(t) \) of \( \Pi(t) \) is a “copy” of \( \mathcal{M}_h^{(1)}(t_1) \) by replacing each symbol \( a \) by \( a_0 \). Moreover, \( \mathcal{M}_t(t) = \mathcal{M}_t(t_1) = \mathcal{M}_h(t) = \mathcal{M}_h^{(1)}(t_1) \) \( \cup \{a_0, a_1\} \) and \( \mathcal{M}_h^{(1)}(t_1) \) \( \cup \{y_0, y_1\} \), for \( i = 1, 2 \).

(vi) Set of rules. The set of rules is \( \mathcal{R}^{(1)}(t_1) \cup \mathcal{R}^{(2)}(t_2) \cup \mathcal{R}_t \), where \( \mathcal{R}^{(1)}(t_1) \) is the set of rules of \( \Pi^{(1)}(t_1) \), for \( i = 1, 2 \), obtained through the replacement of objects yes and no by objects yes and no, respectively, in each rule. \( \mathcal{R}_t \) is the following set of rules.

(1) Rules for simultaneously transporting \( \text{cod}_t(u) \) from \( r_t'' \) to \( h^{(1)}(t_1) \), for \( i = 1, 2 \) and for each \( a \in \Sigma^{(1)}(t_1) \):

\[
\Gamma^{(1)}(t_1) \cup \{a' | a \in \Sigma^{(1)}(t_1)\} \rightarrow [a'] \Gamma^{(1)}(u), \text{for each } a \in \Sigma(t_1) \text{ and each membrane } h \text{ being either } h \rightarrow h^{(1)}(t_1) \text{ or being } h \text{ an ancestor of it.}
\]

(2) Rules for simultaneously transporting \( \text{cod}_t(u) \) from \( h^{(1)}(t_1) \) to the input membrane \( \text{in}^{(1)}(t_1) \) by transforming it into \( \text{cod}_t(u) \), for \( i = 1, 2 \):

\[
\Gamma^{(1)}(t_1) \cup \{a', a'' | a \in \Sigma^{(1)}(t_1)\} \rightarrow \#.
\]

(3) Rules for synchronizing the simulation of \( \Pi^{(1)}(t_1) + \text{cod}_t(u) \) in order to start at the \( (m_i + 2)th \) transition step, for \( i = 1, 2 \):

\[
\Gamma^{(1)}(t_1) \cup \{a_0, a_1 | a \in \Sigma^{(1)}(t_1)\} \rightarrow \#.
\]

(4) Rules for synchronizing the instant \( n = 2 + \max\{d_1 + q_1 + p_1 \} \) \( \in \{1 \leq \leq 2\} \) in which the answers of \( \Pi^{(1)}(t_1) + \text{cod}_t(u) \) and \( \Pi^{(2)}(t_2) + \text{cod}_t(u) \) reach the membrane \( r_t'' \):

\[
\Gamma^{(1)}(t_1) \cup \{a_0, a_1 | a \in \Sigma^{(1)}(t_1)\} \rightarrow \#.
\]

(5) Rules for the output of the system \( \Pi(t) + \text{cod}_t(u) \):

(5.1) \( \Gamma^{(1)}(t_1) \cup \{a_0, a_1 | a \in \Sigma^{(1)}(t_1)\} \rightarrow \#.
\]

(5.2) \( \{\text{NO}, \text{NO} | a \in \Sigma^{(1)}(t_1)\} \rightarrow \text{NO}, \text{for } i = 1, 2.
\]

(5.3) \( \{\text{NO}, \text{NO} | a \in \Sigma^{(1)}(t_1)\} \rightarrow \text{NO}.
\]

(5.4) \( \{\text{YES}, \text{YES} | a \in \Sigma^{(1)}(t_1)\} \rightarrow \text{YES}.
\]

(5.5) \( \{\text{YES}, \text{YES} | a \in \Sigma^{(1)}(t_1)\} \rightarrow \text{YES}.
\]

(5.6) \( \{\text{YES}, \text{YES} | a \in \Sigma^{(1)}(t_1)\} \rightarrow \text{YES}.
\]

4.1. Input Membrane. The input membrane is the membrane labelled by \( r_t'' \).

An overview of the computations of \( \Pi(t) + \text{cod}_t(u) \) is as follows.

The proposed solution can be structured in the following stages.

(i) Stage 1

It consists of several phases: for \( i = 1, 2 \), in a simultaneous way, (a) the input multiset \( \text{cod}_t(u) \) is transported from the input membrane \( r_t'' \) of \( \Pi(t) \) to the skin membrane \( r^{(2)}_t \) of \( \Pi^{(2)}(t_2) \) by transforming into the multiset \( \text{cod}_t(u) \) obtained from \( \text{cod}_t(u) \) by replacing each symbol \( a \) in \( \text{cod}_t(u) \) by \( a' \); (b) then, \( \text{cod}_t(u) \) is transported from the skin membrane \( r^{(2)}_t \) to membrane \( h^{(2)}(t_2) \); finally, (c) \( \text{cod}_t(u) \) moves from \( h^{(2)}(t_2) \) to the input membrane of \( \Pi^{(2)}(t_2) \), being transformed into \( \text{cod}_t(u) \).

(ii) Simulation stage

Computations of the system \( \Pi^{(1)}(t_1) \) with input multiset \( \text{cod}_t(u) \) are simulated, for \( i = 1, 2 \).

(iii) Output stage

The system \( \Pi(t) \) with input multiset \( \text{cod}_t(u_1, u_2) \) sends the right answer to the environment according to the results obtained in the previous stage.
4.1.1. Transport Stage. Once the input multiset \( \text{cod}(u_1, u_2) = \text{cod}_1(u_1) + \text{cod}_2(u_2) \) is supplied to the system \( \Pi(t) \), by applying the rules from 1.1, multiset \( \text{cod}_i(u_i) \), for \( i = 1, 2 \), enters into membrane \( \Pi^{(i)}(t_i) \) transformed in \( \text{cod}_i(u_i) \). Next, by applying the rules from 1.2, multiset \( u_i \) is transported to membrane \( h_i^{(t_i)} \). Then, multiset \( \text{cod}_i(u_i) \) is transported from membrane \( h_i^{(t_i)} \) to the input membrane of \( \Pi^{(i)}(t_i) \) being transformed in \( \text{cod}_i(u_i) \), by dissolving the membrane \( h_i^{(t_i)} \) (rule 2.2) and evolving objects \( a_i \) to objects \( a_i \) by applying the rules from 2.3. This process can be observed in a graphical way in Figure 1.

It is worth pointing out that, in this process, we must prevent the rules of \( \Pi^{(i)}(t_i) \) from being applied. In order to do that rules from 3.1 handle copies \( a_i \) of objects initially placed in membranes during \( m_i \) transition steps, and rules from 3.2 transform those objects in objects \( a_i \) in order to be used in the simulation of \( \Pi^{(i)}(t_i) + \text{cod}_i(u_i) \). Consequently, at the configuration \( \mathcal{C}_{m_i+1} \), the simulation of \( \Pi^{(i)}(t_i) + \text{cod}_i(u_i) \), for \( i = 1, 2 \), will start.

4.1.2. Simulation Stage. The simulation of \( \Pi^{(i)}(t_i) + \text{cod}_i(u_i) \) starts at the \((m_i + 2)\)th transition step. Then, after at most \( p_i(u_i) \) transition steps, an answer will be sent to the membrane \( r_i' \). Consequently, after at most \( n = 2 + \max\{d_i + q_i + p_i, (u_i) : 1 \leq i \leq 2 \} \) transition steps, systems \( \Pi^{(1)}(t_j) + \text{cod}_1(u_1) \) and \( \Pi^{(2)}(t_j) + \text{cod}_2(u_2) \) will send their responses to membrane \( r_i' \), where counter \( \beta_i \) has been evolving from \( \beta_0 \) to \( \beta_n \) by applying rules from 4.1.

4.1.3. Output Stage. At the \((n + 1)\)th transition step, the membrane \( r_i'' \) is dissolved by applying the rule 5.1. Then, membrane \( r_i' \) will contain objects \( a_{i, #} \) along with the two answers (yes, and/or no) provided by the simulations of \( \Pi^{(i)}(t_j) + \text{cod}_i(u_i) \), for \( i = 1, 2 \).

Case 1. Membrane \( r_i' \) from configuration \( \mathcal{C}_{n+1} \) contains at least an object no. In this case, the answer must be negative. In fact, at the \((n + 2)\)th step, by applying rule 5.2 one object no dissolves the membrane \( r_i' \) and evolves to object NO. Simultaneously, by applying the rule 5.4, object \( a_1 \) evolves to object \( a_i \). Then, at configuration \( \mathcal{C}_{n+2} \) the skin membrane \( r_1' \) contains objects \( a_i, \# \), NO and one object yes, or no. Finally, at the next step, by applying the rule 5.3, object no is sent out of system. A graphical interpretation of this case is present in Figure 2.

Case 2. Membrane \( r_i' \) from configuration \( \mathcal{C}_{n+1} \) does not contain any object no. In this case, the answer must be affirmative. In fact, at the \((n + 2)\)th step, by applying rule 5.4, object \( a_0 \) evolves to \( a_i \) inside membrane \( r_i' \). Then, membrane \( r'_j \) from configuration \( \mathcal{C}_{m_2+1} \) will contain objects \( a_i, \#, \text{yes}_1 \), and \( \text{yes}_2 \). Next, by applying rule 5.5, object YES reaches membrane \( r_i' \). Finally, an object yes is sent out to the environment by applying rule 5.6. A graphical interpretation of this case is present in Figure 3.

Remark 1. Concerning to the proof of Theorem 1, we would like to draw attention to two aspects: (a) recognizer membrane systems from class \( \mathcal{R} \) are required to allow the use of dissolution rules, object evolution rules, and communication rules (in both directions: send-in and send-out) and (b) an explicit solution of the product problem product is provided from two solutions of the problems involved in that operation.

Corollary 1. Let \( \mathcal{R} \) be a presumably efficient computing model of recognizer cell-like P systems allowing dissolution rules, object evolution rules, and communication rules (send-in and send-out). Then, \( \mathcal{DP} \cup \text{co-DP} \subseteq \mathcal{PMC}_{\mathcal{R}} \).

Proof. It suffices to note that if \( X \) is an \( \text{NP} \) complete problem such that \( X \in \mathcal{PMC}_{\mathcal{R}} \), then the product problem \( X \circ \overline{X} \) is a \( \text{DP} \) complete problem such that \( X \circ \overline{X} \in \mathcal{PMC}_{\mathcal{R}} \). Since the complexity class \( \mathcal{PMC}_{\mathcal{R}} \) is closed under complement and under polynomial-time reduction, \( \mathcal{DP} \cup \text{co-DP} \subseteq \mathcal{PMC}_{\mathcal{R}} \) follows.

5. Applying the Methodology to MAJORITY

In this section, we detail the application of the methodology through a simple example. For the sake of simplicity, a simple problem \( X \) will be selected so the focus can be on the application of the problem. The application of the methodology to an \( \text{NP} \)-complete problem will be applied in [16].

Let \( M_f(A) \) be the set of all finite multisets over \( A \), and let \( f_u(x) \) be the number of appearances of symbol \( x \) in the multiset \( u \). Let size \((u)\) be the number of elements of \( u \).

Let \( X = (I_X, \theta_X) \) be the MAJORITY problem. Let \( I_X = \{x \mid x \in M_f((0, 1)) \} \) and \( \theta_X(u) = 1 \) if and only if \( f_u(1) > f_u(0), \) that is, if \( f_u(1) \geq 1 + \lfloor n/2 \rfloor \).

A uniform family of polarizationless \( P \) systems with dissolution rules can be designed for this purpose (the MAJORITY problem falls under the class \( \mathcal{P} \); therefore, it could be enough to use the cod function to encode the input as yes in the affirmative case and no in the negative case, but for the application of the methodology, we will make that \( \text{cod}(u) \) will encode a 1 as an object \( t \) and a 0 as an object \( f \). Let \( n > 0 \) be the number of elements of the input instance. We define the recognizer polarizationless \( P \) system with dissolution rules:

\[
\Pi(n) = (\Gamma, \Sigma, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_3, \mathcal{R}_1, i_{\text{in}}, i_{\text{out}}),
\]

where

\begin{enumerate}
\item \( \Gamma = \Sigma \cup \{\text{yes}, \text{no}, y, \#\} \cup \{d_k \mid 0 \leq k \leq 3 + \lfloor n/2 \rfloor\} \)
\item \( \Sigma = \{f, t\} \)
\item \( \mu = \{[0 \ldots 2 + \lfloor n/2 \rfloor] \}_{0} \}
\item \( \mathcal{M}_{3+(n/2)} = \{y\} \), \( \mathcal{M}_2 = \{d_0\} \), \( \mathcal{M}_1 = \emptyset, i \notin \{2, 3 + \lfloor n/2 \rfloor\} \)
\item \( \text{The set } \mathcal{R}_i \text{ of rules is formed by the following rules:} \)
\begin{enumerate}
\item These rules will carry the object \( y \) to the membrane labelled by \( f \) if and only if there are at least \( 1 + \lfloor n/2 \rfloor \) objects \( 1 \):}
\end{enumerate}
\end{enumerate}
Figure 1: Transport of objects from $r''_t$ to the corresponding in $\Pi^{i}(t_i)$. For this purpose, rules from 1 and 2 are applied. Besides these rules, rules from 3 are applied to make the objects from $M^h$ be available in their corresponding membranes only when objects from $\Sigma^h$ are in their corresponding input membranes.

Figure 2: Negative answer. With rules from 4 and 5, an object no is sent to the environment in the last step of the computation.

(c) Rules for returning yes if object $y$ has arrived the membrane labelled by 1 in the configuration $[n/2] + 1$:

\[
|y_2| \rightarrow |y_2|_2, \\
|y|_2 \rightarrow \text{yes}, \\
|\text{yes}|_1 \rightarrow \text{yes}_1.
\]
(d) Rules for returning an object no in case the object y is not in membrane 1 in configuration \([n/2]_2\):

\[
d_{[n/2]_2} \rightarrow \text{no}_2, \\
\text{no}_1 \rightarrow \text{no}_1.
\] (7)

(6) \(i_{\text{in}} = 2 + [n/2]\) and \(i_{\text{out}} = \text{env}\).

The system works are as follows: the instance is introduced in the input membrane as a multiset of objects \(t\) and \(f\) (\(\text{cod}(u)\) will encode a 1 as an object \(t\) and a 0 as an object \(f\)). Objects \(t\) will start dissolving membranes starting from the membrane labelled by \([n/2]\) to the membrane labelled by 3. If there are enough objects \(t\) (that is, if there is a majority of 1 in the instance), then object \(y\) will be in the membrane labelled by 1 in configuration \(C_{[n/2]+1}\). Therefore, object \(y\) will go into membrane 2 and will dissolve itself, avoiding object \(d_{[n/2]_2}\) to evolve into an object no. While dissolving, object \(y\) will be transformed into an object yes, that will go to the environment in the last step of the computation. If object \(y\) is not present in the membrane 1, then object \(d_{[n/2]_2}\) will be transported to membrane 1 as an object no in the membrane 1 and will be sent to the environment in the last step of the computation. The system will take \(5 + [n/2]\) steps in the affirmative case, and \(6 + [n/2]\) steps in the negative case. With this solution to the MAJORITY problem, we can proceed to apply the methodology.

Let \(\text{MAJORITY}\) be the complementary problem of \(\text{MAJORITY}\), that is, \(I_{\text{MAJORITY}} = I_{\text{MAJORITY}}(u) = 1 - \theta_{\text{MAJORITY}}(u)\). A solution to this problem could be given simply changing the roles of objects yes and no.

Let

\[
\Pi^{(1)}(n) = (\Gamma^{(1)}, \Sigma^{(1)}, \mu^{(1)}, M_1^{(1)}, \ldots, M_{[n/2]}^{(1)}, \mathcal{R}^{(1)}, i_{\text{in}}, i_{\text{out}}),
\] (8)

be recognizer polarizationless P systems with dissolution rules that solves the instances of \(\text{MAJORITY}\) of size \(n\) as described above. We can define a recognizer polarizationless P system with dissolution rules:

\[
\Pi^{(2)}(n) = (\Gamma^{(2)}, \Sigma^{(2)}, \mu^{(2)}, M_{[n/2]+1}^{(2)}, \ldots, M_{6+[n/2]_2}^{(2)}, \mathcal{R}^{(2)}, i_{\text{in}}, i_{\text{out}}),
\] (9)

which solves the instances of \(\text{MAJORITY}\) of size \(n\) as follows. For doing \(\Gamma^{(1)}(u)\) disjoint, \(\Gamma^{(2)}(t)\) will consist on the symbols from \(\Gamma^{(1)}(t)\) in capital letters. All the rules are changed accordingly to this. To make \(H^{(1)}\) and \(H^{(2)}\) disjoint, given \(n_1, n_2 > 0\), we change the label \(i\) of \(H^{(2)}\) to \(3 + [n_1/2]_2 + i\). In \(\Pi^{(1)}\), we change objects yes and no by yes, and no\(_0\), respectively.

From here, we are going to define P system that will solve the problem \(\text{MAJORITY} \otimes \text{MAJORITY}\), which is defined as follows. Let \(u_1, u_2 \in M_f(\{0, 1\})\).

\[
\theta_{\text{MAJORITY} \otimes \text{MAJORITY}}(u_1, u_2) = 1 \text{ if and only if } \theta_{\text{MAJORITY}}(u_1) = 1 \text{ and } \theta_{\text{MAJORITY}}(u_2) = 1.\]

Let \(\text{cod}(u_1) = \text{cod}(u_1) + \text{cod}(u_2)\). Let \(t_i = \text{size}(u_i)\), and \(t = \langle t_1, t_2 \rangle\). Let \(q_i = t_i\) and \(d_i = 3 + [t_i/2]\). Let \(p_i(|u_i|) = 6 + [t_i/2]\). Let \(m_i = q_i + d_i + 2 = t_i + 4 + [t_i/2]\) and \(n = \max\{d_i + q_i + p_i(|u_i|)\} = \max\{t_i + 8 + 2 \cdot [t_i/2], t_2 + 8 + 2 \cdot [t_2/2]\}\). We define a recognizer polarizationless P system with dissolution rules:

\[
\Pi(t) = (\Gamma, \Sigma, \mu, M, \ldots, M_{6+[t/2]+[t/2]}, \mathcal{R}, i_{\text{in}}, i_{\text{out}}),
\]

\[
M_{\#}^{(1)}(t_1), m_{\#}^{(2)}(t_2), \mathcal{R}, i_{\text{in}}, i_{\text{out}}),
\]

(10)

where

\[
\begin{align*}
\Gamma^{(1)}(t) & = (\# \#) \cup \{\text{YES}, \text{NO} \} \cup \{t', f', T', F'\}, \\
\Gamma^{(2)}(t) & = \Sigma(2) (t_2) \cup \{y_1, no_1, y, \#\} \cup \{D_k | 0 \leq k \leq 3 + \lfloor t/2 \rfloor\}, \\
\Gamma^{(3)}(t) & = \{\# \# \}, \\
\Gamma^{(4)}(t) & = \left\{\text{yes}_{t}, no_{t}, y | 0 \leq j \leq m_1\right\} \cup \{D_k | 0 \leq k \leq 3 + \lfloor t_1/2 \rfloor, 0 \leq j \leq m_1\} \cup \left\{\text{yes}_{t}, no_{t}, Y | 0 \leq j \leq m_2\right\} \cup \{D_k | 0 \leq k \leq 3 + \lfloor t_2/2 \rfloor, 0 \leq j \leq m_2\}, \\
\Gamma^{(5)}(t) & = \{\text{yes}_{t}, no_{t}, y | 0 \leq j \leq n\}, \\
\Gamma^{(6)}(t) & = \{y_1 | 0 \leq j \leq \max\{m_1, m_2\}\}.
\end{align*}
\] (11)
(ii) $\Sigma(t) = \Sigma^{(1)}(t_1) \cup \Sigma^{(2)}(t_2)$, being $\Sigma^{(1)}(t_1) = \{t, f\}$ and $\Sigma^{(2)}(t_2) = \{T, F\}$.

(iii) $H(t) = H^{(1)}(t) \cup H^{(2)}(t) \cup \{r, r', r'', h^{(1)}(t_1), h^{(2)}(t_2)\}$, where $r, r', r'', h^{(1)}(t_1), h^{(2)}(t_2)$ are different from each other, and $r, r', r'', h^{(1)}(t_1), h^{(2)}(t_2) \notin H^{(1)}(t) \cup H^{(2)}$.

(iv) $\mu = \mu = \{[[\ldots [h_{m_1}(t_1)]_{m_1+1} [t_2/2] \ldots] [h_{m_2}(t_2)]_{m_2+1} t_1/2] + [t_2/2] \ldots [h_{m_2}(t_2)]_{m_2+1} t_1/2\}$.

(v) $\mathcal{M}_{2+1}(t) = \{y_0\}, \mathcal{M}_{2+1}[t_1/2] = \{y_0\}, \mathcal{M}_0 = \{d_0\}, \mathcal{M}_0(t) = \{a_0, \beta_0\}$, and

\[
\begin{align*}
A_i \rightarrow [d'_{1}], \\
A_{i+1} \rightarrow [d'_{1}], \\
a_i \rightarrow [a'_{1}], \\
h^{(1)}(t_1) \rightarrow [a'_{1}], \\
a_i \rightarrow [a'_{1}], \\
h^{(2)}(t_1) \rightarrow [a'_{1}], \\
A^{' i} \rightarrow A_{i}^{(1)}, \\
A^{' i} \rightarrow A_{i}^{(2)},
\end{align*}
\]

for $a \in \{t, f\}, A \in \{T, F\}$,

\[
\begin{align*}
\mathcal{M}_{9+1}(t_1) = \{y_0\}, \mathcal{M}_{9+1}[t_2/2] = \{y_0\}, \mathcal{M}_0 = \{d_0\}, \mathcal{M}_0(t) = \{a_0, \beta_0\},
\end{align*}
\]

and $\mathcal{M}_{9+1}(t_1) = \{y_0\}$, for $i = 1, 2$ and $\mathcal{M}_j = \varnothing$ for the rest of membranes.

(vi) Set of rules. The set of rules is $\mathcal{R}_{\Pi^{(1)}(t_1)} \cup \mathcal{R}_{\Pi^{(2)}(t_2)}$, where $\mathcal{R}_{\Pi^{(1)}(t_1)}$ is the set of rules of $\Pi^{(1)}(t_1)$, for $i = 1, 2$, obtained through the replacement of objects yes and no by objects yes, and no, respectively, in each rule. $\mathcal{R}_{\Pi^{(1)}(t_1)}$ is the following set of rules:

(1) Rules for simultaneously transporting $\cod_i(u_i)$ from $h^{(i)}(t_i)$, for $i = 1, 2$ and transforming it into $\cod_i(u_i)$:

\[
\begin{align*}
\beta_i \rightarrow \beta_{i+1}, \\
\beta_i \rightarrow \beta_{i+1}, \\
\alpha_i \rightarrow \alpha_{i+1}, \\
\alpha_i \rightarrow \alpha_{i+1}, \\
\end{align*}
\]

for $a \in \{t, f\}, A \in \{T, F\}$.

(2) Rules for simultaneously transporting $\cod_i(u_i)$ from $h^{(i)}(t_i)$ to the input membrane $\Pi^{(i)}(t_i)$ by transforming it into $\cod_i(u_i)$, for $i = 1, 2$:

\[
\begin{align*}
\begin{cases}
y_j \rightarrow y_{j+1}, & \text{for } i \in [1, 2], 0 \leq j \leq m_i - 1, \\
y_0 \rightarrow \#, & \text{for } i \in [1, 2], \\
a' \rightarrow a', & \text{for } a \in \{t, f\}, A \in \{T, F\}, \\
A' \rightarrow A', & \text{for } a \in \{t, f\}, A \in \{T, F\},
\end{cases}
\end{align*}
\]

(3) Rules for synchronizing the simulation of $\Pi^{(i)}(t_i) + \cod_i(u_i)$ in order to start at the $(m_i + 2)$th transition step, for $i = 1, 2$:

\[
\begin{align*}
\begin{cases}
d_0 \rightarrow d_{m_i}, & \text{for } 0 \leq j \leq m_i, \\
y_j \rightarrow y_{j+1}, & \text{for } 0 \leq h \leq m_2, \\
\end{cases}
\end{align*}
\]

(4) Rules for synchronizing the instant $n = 2 + \max\{d_i + q_i + p_i(u_i) : 1 \leq i \leq 2\}$. in which the answers of $\Pi^{(i)}(t_i) + \cod_i(u_i)$ and $\Pi^{(2)}(t_2) + \cod_2(u_2)$ reach the membranes $\mathcal{M}$:

\[
\begin{align*}
\begin{cases}
\beta_i \rightarrow \beta_{i+1}, & \text{for } 0 \leq j \leq n - 1, \\
\alpha_i \rightarrow \alpha_{i+1}, & \text{for } i \in [1, 2], \\
\end{cases}
\end{align*}
\]

(5) Rules for the output of the system $\Pi(t) + \cod(t_1, t_2)$:

\[
\begin{align*}
\begin{cases}
[\beta_i]_{t_i} \rightarrow \# & \text{for } i \in [1, 2], \\
[\alpha_i]_{t_i} \rightarrow \text{NO}, & \text{for } i \in [1, 2], \\
[\alpha_i]_{t_i} \rightarrow \text{YES}_{t_i}, & \text{for } i \in [1, 2], \\
[\text{YES}]_{t_i} \rightarrow \text{yes}_{t_i}, & \text{for } i \in [1, 2],
\end{cases}
\end{align*}
\]

\[
\begin{align*}
i_{\text{in}} = r_i, \\
i_{\text{out}} = \text{env}.
\end{align*}
\]

This system will return an object yes or an object no, but not both, only in the last step of the computation. The computation will take $n + 4$ computational steps in the affirmative case and $n + 3$ steps in the negative case.

6. Conclusions

Given a computing model $\mathcal{R}$ of recognizer membrane systems with the ability to provide polynomial-time solutions to $\text{NP}$ complete problems (presumably efficient computing model), the time complexity class $\text{PMC}_{\mathcal{R}}$ contains
the complexity classes NP and co-NP, since PMC_{GR} is closed under complement and under polynomial-time reductions [2]. In this paper, a lower bound thinner than NP ∪ co-NP is provided for the class PMC_{GR} associated with a presumably efficient computing model R of recognizer P systems allowing object evolution rules, dissolution rules, and communication rules. Specifically, for such kind of computing models, the following result has been obtained: DP ∪ co-DP ⊆ PMC_{GR}, where DP is the set of languages which can be expressed as the difference of two languages in NP or, equivalently, as the intersection of a language in NP and a language in co-NP.

In this paper, we have applied the methodology to the MAJORITY problem. We have chosen such a simple problem for the sake of simplicity. An efficient solution to the SAT-UNSAT problem, a DP-complete problem, is given in [16].

In this research line, it would be interesting to analyze the possibility of extending the main theorem to other presumably efficient computing models of recognizer membrane systems; for instance, cooperative cell-like membrane systems without dissolution rules, tissue-like P systems with symport/antiport rules or spiking neural P systems.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

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

References