Theses of the PhD thesis

Results on the computational power of polarizationless P systems with active membranes

Károly Hajagos

Supervisor:

Dr. Zsolt Gazdag, PhD

Doctoral School of Computer Science
University of Szeged

Department of Foundations of Computer Science

2025

1 Introduction

Membrane computing is a branch of natural computing inspired by the architecture and functioning of living cells. The aim of membrane computing is to discover and formulate computational models to simulate the behavior of biological cells. These models are called *membrane systems*, which are also known as *P systems*.

Membrane structure. A P system operates on a membrane structure that is built by membranes nested in each other. Let O be an alphabet of *objects* and H be a finite set of *labels*. Formally, a *membrane structure* is a triple (V, E, L) where (V, E) is a nonempty, finite, rooted, directed tree (with its edges directed towards the root) and $L : V \to H$ assigns labels to each node, such that only the root is labeled with the symbol skin, and it has to be labeled with skin. The nodes are called *membranes* of the structure. For any edge $(x, y) \in E$, x is a *child membrane* of y and y is the *parent membrane* of x. A membrane with label h is called an *h-membrane*. We can assume that initially, each membrane has its unique label. A membrane that has only an outgoing edge is called an *elementary membrane*.

Membrane configuration. Let $\boldsymbol{\mathcal{M}}(O)$ be the set of all *multisets* over O. A *membrane configuration* is a tuple (V, E, L, ω) where (V, E, L) is a membrane structure and $\omega : V \to \boldsymbol{\mathcal{M}}(O)$ is a function that assigns a finite multiset of objects to each membrane. Each multiset is represented by some words over O. An instance of an object a is called an a-copy. If x is a membrane and $a \in \omega(x)$ then we say that x *contains an a-copy*. Let $C = (V, E, L, \omega)$ be a membrane configuration. The membrane configuration $C' = (V', E', L', \omega')$ is a *sub-configuration* of C if (V', E') is a subtree of (V, E), and L' and ω' are restrictions of L and ω to V', respectively.

Polarizationless P systems with active membranes. A *polarizationless P system with active membranes* is a construct of the form $\Pi = (O, H, \mu, R)$, where $\mu = (V, E, L, \omega)$ is the initial membrane configuration and R is a finite set of *rules* of the following types:

(a) $[a \to v]_h$, for some $h \in H, a \in O, v \in O^*$ (*object evolution* rules);

(b) $a[\ ]_h \to [b]_h$, for some $h \in H$, $a, b \in O$ (*in-communication* rules);

(c) $[a]_h \to [\ ]_h b$, for some $h \in H$, $a, b \in O$ (*out-communication* rules);

(d) $[a]_h \to b$, for some $h \in H$, $a, b \in O$ (*membrane dissolution* rules);

(e) $[a]_h \to [b]_h[c]_h$, for some $h \in H$, $a, b, c \in O$ (*division* rules for elementary membranes).

The left-hand side of a rule is the string that lies to the left of the symbol $\to$. In each time unit, some of the rules are *applied*. In this way, the P system can perform *transitions* from one configuration to another. A sequence of these transitions is called *computation*, and such a transition is called *computation step*.

During a computation step, the rules are applied nondeterministically according to the concepts of *maximal parallelism*. We note that *division rules for non-elementary membranes* are often considered, but we do not view them as part of the basic model. These rules provide the possibility of duplicating subtrees of the membrane structure.

We often refer to Păun's conjecture [9] which is one of the key elements in the thesis and roughly sounds as follows. Polarizationless P systems with active membranes cannot solve computationally hard problems in polynomial time without non-elementary membrane division rules.

Recognizer P systems. *Recognizer P systems* are common tools for deciding problems. A P system is a recognizer P system if each of its computations *halts*, it has two designated objects yes and no, it has designated *input* and *output membranes*, and each of its computations must produce exactly one yes or no (but not both) in the output membrane, exactly in the last step of the computation. A P system is *confluent* if all halting computations on the same input must produce the same output. In the thesis, we consider only confluent P systems.

Uniformity for P systems. A family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of P systems is $\mathbf{P}$-*uniform* (respectively, $\mathbf{L}$-*uniform*) if there is a deterministic Turing machine working in polynomial time (resp., using logarithmic space) that computes a reasonable representation of $\Pi(n)$ whenever it is started with 1^n on its input tape and computes the encoding of the input.

Let D be a decision problem and $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ be a $\mathbf{P}$-uniform (resp., $\mathbf{L}$-uniform) family of recognizer P systems. We say that $\mathbf{\Pi}$ *solves D in polynomial time* (resp., *solves D using logarithmic space*) if there is a polynomial-time (resp., log-space) computable encoding cod that transforms instances of D into multisets of objects and there exists an integer $k \in \mathbb{N}$ such that the following holds. For every instance x of D with size n, each computation of $\Pi(n)$ starting with $cod(x)$ in its input membrane halts in at most n^k steps and produces yes in the output membrane if and only if x is a positive instance of D.

Denote $\mathbb{N}$ the set of natural numbers including zero. For numbers $i \leq j$ in $\mathbb{N}$, $[i, j]$ denotes the set $\{i, i+1, \ldots, j\}$ and, for $j \geq 1$, $[j]$ denotes the set $[1, j]$.

Sections 2, 3, 4 and 5 summarize Chapter 3, 4, 5 and 6 of the thesis, respectively. The numbers of the theorems and propositions are the same as those in the thesis.

2 On the power of membrane dissolution

We consider polarizationless P systems with active membranes using only dissolution rules and investigate their computational power. We call these P systems dissolution P systems. We show that the $\mathbf{NL}$-complete REACHABILITY problem can be solved in polynomial time by $\mathbf{L}$-uniform families of these P systems.

The REACHABILITY **problem.** We consider the following variant of the well-known $\mathbf{NL}$-complete REACHABILITY problem. Consider a digraph $G = (\mathcal{V}, \mathcal{E})$. We call G *topologically sorted*, if $\mathcal{V} = \{1, 2, \ldots, n\}$, $n \geq 2$, and for every $(i, j) \in \mathcal{E}$, $i < j$. We will use the notation $i \rightsquigarrow_G j$ to denote that there exists a path from i to j in G, that is j *is reachable from* i *in* G. Then REACHABILITY is defined as follows:

Input: a topologically sorted digraph $G = (\mathcal{V}, \mathcal{E})$.
Output: yes if $1 \rightsquigarrow_G n$ and no otherwise.

REACHABILITY defined in this restricted form is still $\mathbf{NL}$-complete. To solve REACHABILITY, our P systems will implement a function $reachability$ given in Algorithm 1. This algorithm is based on the following observation. Let $G = (\mathcal{V}, \mathcal{E})$ be a topologically sorted digraph with n nodes. Moreover, let $i \in [2, n]$ and $j \in [i+1, n]$. Clearly, if a node i is reachable from 1 in G and $(i, j) \in \mathcal{E}$, then node j is reachable from 1 in G as well. Thus, the algorithm, roughly, checks whether $1 \rightsquigarrow_G i$ and $(i, j) \in \mathcal{E}$. If so, then it extends $\mathcal{E}$ with the edge $(1, j)$. This extension is executed by the corresponding cycle of the innermost loop (Lines 6–9) which we call the (i, j)*th edge operation* of Algorithm 1.

Algorithm 1 Decision of REACHABILITY

```
function $reachability(G)$
    $\mathcal{E}_1 = \mathcal{E}$
    for $i = 2$ to $n-1$ do
        $\hat{\mathcal{E}}_i = \mathcal{E}_{i-1}$
        for $j = i+1$ to $n$ do
            if $(1,i) \in \hat{\mathcal{E}}_i$ and $(i,j) \in \hat{\mathcal{E}}_i$ then
                Remove($\hat{\mathcal{E}}_i$,$(i,j)$)
                Add($\hat{\mathcal{E}}_i$,$(1,j)$)
            end if
        end for
        $\mathcal{E}_i = \hat{\mathcal{E}}_i$
    end for
    if $(1,n) \in \mathcal{E}_{n-1}$ then
        return $true$
    else
        return $false$
    end if
end function
```

According to Proposition 1, Algorithm 1 is correct and complete.

Proposition 1. *Let $G = (\mathcal{V}, \mathcal{E})$ be a topologically sorted digraph with n nodes. Then, $1 \rightsquigarrow_G n$ if and only if $(1,n) \in \mathcal{E}_{n-1}$ in Algorithm 1.*

Now, we present the main result of this section which is the following.

Theorem 1. *The* REACHABILITY *problem can be solved in polynomial time by an* **L**-*uniform family of dissolution P systems.*

We roughly describe the construction and the behavior of the **L**-uniform family $\mathbf{\Pi} = \{\Pi(n) \mid n \geq 2\}$ of deterministic dissolution P systems where $\Pi(n)$ is devoted to decide REACHABILITY for topologically sorted graphs of n nodes by simulating Algorithm 1.

Let us fix a topologically sorted graph $G = (\mathcal{V}, \mathcal{E})$ with n nodes and m edges. First, we define the encoding of G, denoted by $cod(G)$, as a subset of $\Sigma(n) = \{e_{i,j} \mid i,j \in [n]\}$ in the following way: $cod(G) = \{e_{i,j} \mid (i,j) \in \mathcal{E}\}$. That is, an edge leading from i to j is represented by the object $e_{i,j}$.

Figure 1 shows the initial configuration of $\Pi(n)$. A subconfiguration $op_{i,j}$ in the working subtree, $i \in [2, n-1]$, $j \in [i+1, n]$, is called an *operational subconfiguration* and corresponds to the (i,j)th edge operation of Algorithm 1. This subconfiguration contains two membranes with labels $h_{i,j}$ and $h'_{i,j}$. In both of these membranes, there is a membrane structure with $k_{i,j}$ l-membranes linearly nested in each other, where the number $k_{i,j}$ is defined recursively as follows:

$$k_{i,j} = \begin{cases} 2, \text{ if } i = 2, j = 3, \\ 5 + k_{i,j-1}, \text{ if } i \in [2, n-1], j \in [i+2, n], \\ 5 + k_{i-1,n}, \text{ if } i \in [3, n-1], j = i+1. \end{cases} \tag{1}$$

Initially, the leaves of these subtrees contain an object c.

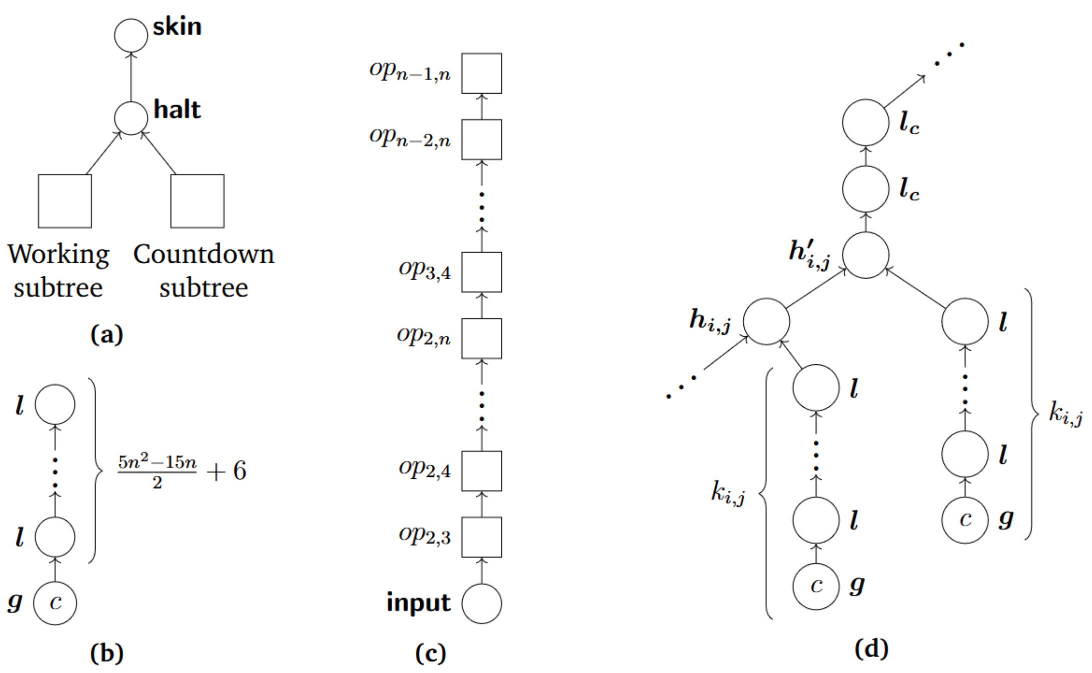

Figure 1: *The initial membrane configuration (a), the countdown subtree (b), the working subtree (c), and the operational subconfiguration $op_{i,j}$ (d). The rectangles denote subconfigurations, the circles denote membranes.*

The behavior of $\Pi(n)$. During the computation, the input objects in $cod(G)$ may evolve to objects in $\Sigma(n)$ by means of dissolution rules. Nevertheless, the number of these objects remains m during the whole computation. Moreover, in each configuration of the computation, these m objects always occur in the same membrane. We call the multiset that contains the objects derived from $cod(G)$ the *edge multiset*.

In the course of the computation, when the edge multiset appears in the $h_{i,j}$-membrane of the operational subconfiguration $op_{i,j}$, the following happens. If $e_{1,i}$ occurs in the $h_{i,j}$-membrane then it triggers its dissolution without evolution, resulting in the appearance of the edge multiset in the $h'_{i,j}$-membrane. Next, if $e_{i,j}$ occurs in the edge multiset, i.e. appears in the $h'_{i,j}$-membrane, then it dissolves this membrane and becomes $e_{1,j}$ during the process, indicating that there is a path between nodes 1 and j. Then, after a few steps, the edge multiset gets to the next operational subconfiguration.

On the other hand, if the edge multiset does not contain $e_{1,i}$ when it appears in the $h_{i,j}$-membrane, then it indicates that at this point it is not known whether i is reachable from 1. Moreover, if the edge multiset contains $e_{1,i}$ but does not contain $e_{i,j}$ when it appears in the $h'_{i,j}$-membrane then it implies that i is reachable from 1 but there is no edge leading from i to j. That is, the $h_{i,j}$-membrane got dissolved by the edge multiset, but the $h'_{i,j}$-membrane does not. In these cases, the following happens. The two c-copies are getting closer to the $h_{i,j}$-membrane and the $h'_{i,j}$-membrane in each step by dissolving the l-membranes (see, Figure 1). After they dissolve all l-membranes, they appear in the $h_{i,j}$-membrane and the $h'_{i,j}$-membrane, respectively (resp., appear together in the $h'_{i,j}$-membrane if the $h_{i,j}$-membrane is dissolved). They dissolve both membranes (resp., one of them dissolves the $h'_{i,j}$-membrane), resulting in the appearance of the edge multiset in the next operational subconfiguration after a few steps without any change. The key factors here are the appropriate embedding of membranes and the timing: the objects need to appear in the right membrane at the right time.

Now, according to Proposition 3, the dissolution of all membranes in the operational subconfiguration $op_{i,j}$ implements the (i,j)th edge operation of Algorithm 1.

Proposition 3. *For every $i \in [2, n-1]$ and $j \in [i+1, n]$, $\mathcal{E}_{i,j} \approx \Omega_{i,j}$ implies $\hat{\mathcal{E}}_{i,j} \approx \hat{\Omega}_{i,j}$.*

Next, Proposition 4 determines the exact number of computational steps necessary for the edge multiset to reach the $halt$-membrane, that is, to dissolve all the membranes in the working subtree.

Proposition 4. *The edge multiset $\hat{\Omega}_{n-1,n}$ appears in the* halt*-membrane in at most $\kappa = \frac{5n^2-15n}{2} + 6$ steps.*

The following two propositions are needed to show the correctness of $\Pi(n)$.

Proposition 5. *During the computation of $\Pi(n)$, the object $e_{1,n}$ appears in the* halt*-membrane if and only if $1 \rightsquigarrow_G n$.*

Proposition 6. *$\Pi(n)$ releases object yes to the* skin*-membrane, if objects $e_{1,n}$ appears in the* halt*-membrane. Otherwise, $\Pi(n)$ releases object no to the* skin*-membrane.*

Correctness, L-uniformity, and running time of $\Pi(n)$. The correctness of $\Pi(n)$ follows from Propositions 5 and 6. Using Proposition 4, one can see that the computation of $\Pi(n)$ halts in $O(n^2)$ steps. Moreover, both the encoding of G and the description of $\Pi(n)$ can be computed by a deterministic Turing machine using $O(\log n)$ space. Therefore, $\mathbf{\Pi}$ is an $\mathbf{L}$-uniform family of dissolution P systems capable of solving the problem REACHABILITY problem in polynomial time.

Conclusions. We have seen that an $\mathbf{L}$-uniform family of dissolution P systems can efficiently solve an $\mathbf{NL}$-complete variant of the well-known REACHABILITY problem. However, the exact computational power of polynomial-time dissolution P systems is still unclear.

3 The characterization of P by dissolution P systems

We continue to investigate dissolution P systems, as we show that these P systems working in polynomial time characterize $\mathbf{P}$ even under a very tight uniformity condition, namely $\mathbf{DLOGTIME}$-uniformity. The $\mathbf{P}$ upper bound in this result is known, and we show the $\mathbf{P}$ lower bound by solving a $\mathbf{P}$-complete variant of the satisfiability problem for Horn formulas.

The HORN3SATNORM problem. We consider a $\mathbf{P}$-complete variant of the HORNSAT problem, the HORN3SATNORM problem (H3SN in short). An instance of HORN3SATNORM is either a positive unit clause or it is either of the forms $\neg x \vee \neg y$ or $\neg x \vee \neg y \vee z$, where

$x, y, z \in Var_n = \{x_1, \ldots, x_n\}$. By basic equivalences of propositional logic, a clause of the form $\neg x \vee \neg y \vee z$ (resp. $\neg x \vee \neg y$) is equivalent to $x \wedge y \rightarrow z$ (resp. $x \wedge y \rightarrow \downarrow$), where $\downarrow$ denotes the constant $false$ value. For convenience, by a non-unit clause of φ, we will mean a formula of the form $x \wedge y \rightarrow z$ or $x \wedge y \rightarrow \downarrow$. Moreover, for the sake of brevity, we will denote $x \wedge y$ by xy in the above formulas.

Let φ be a formula, and let $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ be an enumeration of all the non-unit clauses over Var_n. We denote the set $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$ by C_n. Clearly, $m = |C_n| = O(n^3)$.

Let $Unit^+_\varphi$ be the smallest set in $(Var_n \cup \{\downarrow\})$ satisfying the following properties: (i) all positive unit clauses of φ are in $Unit^+_\varphi$ and (ii) if $x, y \in Unit^+_\varphi$ and $xy \rightarrow z \in \varphi$, then $z \in Unit^+_\varphi$. It is not difficult to see that a formula φ is unsatisfiable if and only if $\downarrow \in Unit^+_\varphi$.

Algorithm 2 Decision of H3SN

```
1: function $H3SN(\varphi)$                              ▷ $\varphi$ is an instance of H3SN with $n$ variables
2:     $\Delta_0 = Unit^0_\varphi$
3:     for $i = 1 \ldots n-1$ do
4:         $\Theta_i = \Delta_{i-1}$
5:         for $j = 1 \ldots m$ do                                       ▷ $m = |C_n|$
6:             if $\mathcal{C}_j = xy \rightarrow z \in \varphi$ and $x, y \in \Theta_i$ then   ▷ $x, y \in Var_n, z \in Var_n \cup \{\downarrow\}$
7:                 Add($\Theta_i$,$z$)                                ▷ extending $\Theta_i$ by $z$
8:             end if
9:         end for
10:        $\Delta_i = \Theta_i$
11:    end for
12:    if $\downarrow \in \Delta_{n-1}$ then
13:        return $false$
14:    else
15:        return $true$
16:    end if
17: end function
```

Denote $Unit^0_\varphi$ the set of positive unit clauses occurring in φ. Algorithm 2 is designed to solve H3SN and to be implemented by the P systems we present in this section. We will call the execution of Lines 6 and 7 in Algorithm 2 for a clause $xy \rightarrow z \in C_n$ in the ith iteration of the outer cycle the *ith conditional extension by* $xy \rightarrow z$.

The main result of the section is the following.

Theorem 2. *The* HORN3SATNORM *problem can be solved in polynomial time by a* **DLOGTIME***-uniform family of dissolution P systems.*

We discuss roughly how the dissolution P systems of the **L**-uniform family $\mathbf{\Pi} = \{\Pi(n) \mid n \geq 2\}$ work, where $\Pi(n)$ is devoted to decide whether an instance of H3SN over Var_n is satisfiable or not.

Let us fix a formula φ over Var_n. First, we define an alphabet to encode φ: $\Sigma(n) = \{v_x, \overline{v}_x \mid x \in Var_n \cup \{\downarrow\}\} \cup \{c_{xy\rightarrow z} \mid xy \rightarrow z \in C_n\}$. The encoding of φ, denoted by $cod(\varphi)$, is the following subset of $\Sigma(n)$:

$$cod(\varphi) = \{v_x \mid x \in \varphi\} \cup \{\overline{v}_x \mid x \notin \varphi\} \cup \{c_{xy\rightarrow z} \mid xy \rightarrow z \in \varphi\} \cup \{\overline{v}_\downarrow\}.$$

For a variable x, v_x will represent that x must be true in any satisfying truth assignment, that is, $x \in Unit^+_\varphi$.

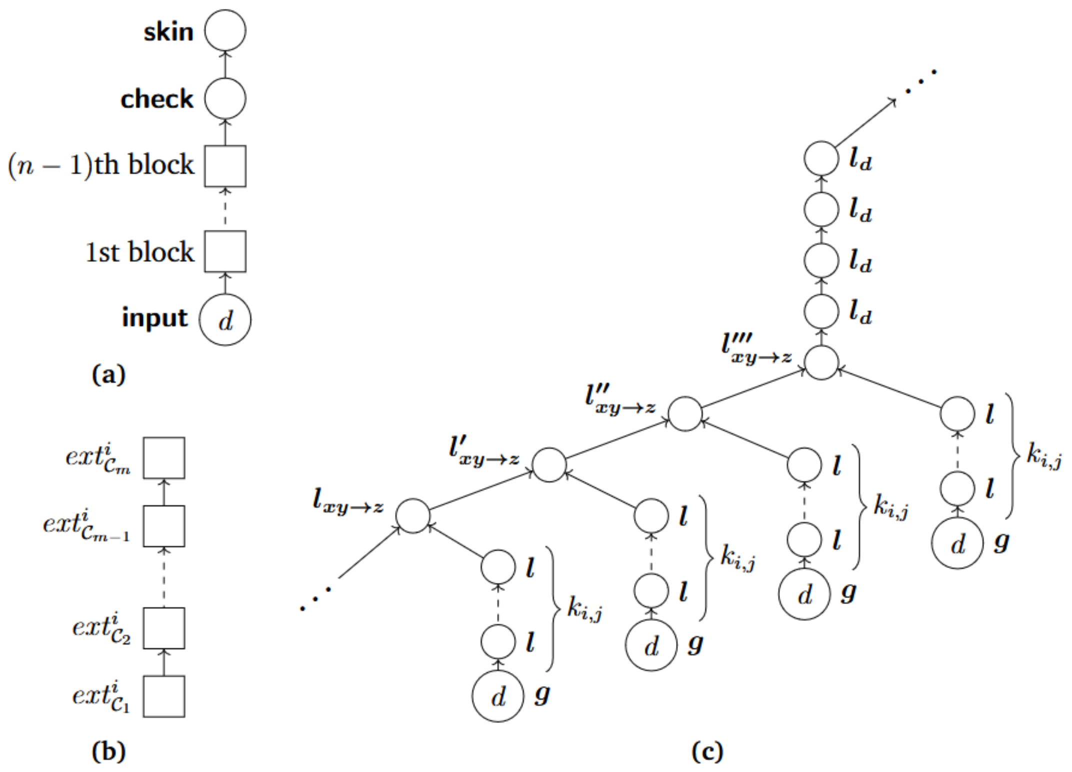

Figure 2: *The initial membrane configuration (a), the* i*th block with* $i \in [n-1]$ *(b) and the subconfiguration* $ext^i_{xy\to z}$ *of* μ *where* $xy \to z = \mathcal{C}_j$ *for some* $j \in [m]$ *(c). The rectangles denote subconfigurations, the circles denote membranes.*

The initial membrane structure of $\Pi(n)$ can be seen in Figure 2. A subconfiguration $ext^i_{\mathcal{C}_j}$, $i \in [n-1], j \in [m]$, corresponds to the ith conditional extension by $\mathcal{C}_j$ of Algorithm 2. Thus, we call these subconfigurations *extensional subconfigurations*. Moreover, we call the subconfiguration consisting of $ext^i_{\mathcal{C}_1}, \ldots, ext^i_{\mathcal{C}_m}$ the *ith block*. If $\mathcal{C}_j = xy \to z$ then we use both the notations $ext^i_{\mathcal{C}_j}$ and $ext^i_{xy\to z}$ for the same extensional subconfiguration. The innermost l_d-membrane in $ext^i_{xy\to z}$ contains four nested membranes labeled with $l'''_{xy\to z}$, $l''_{xy\to z}$, $l'_{xy\to z}$, and $l_{xy\to z}$, respectively, which we will call *clause-membranes*. Furthermore, for every $i \in [n-1], j \in [m]$, each clause-membrane of $ext^i_{\mathcal{C}_j}$ contains a membrane structure of linearly nested l-membranes. We call this structure an *l-structure* and by the *depth* of an l-structure we mean the number of l-membranes in this l-structure. The depth $k_{i,j}$ of each l-structure is defined recursively as follows:

$$k_{i,j} = \begin{cases} 4, \text{ if } i = 1, j = 1, \\ 9 + k_{i,j-1}, \text{ if } i \in [n-1], j \in [2,m], \\ 9 + k_{i-1,m}, \text{ if } i \in [2, n-1], j = 1. \end{cases} \tag{2}$$

The behavior of $\Pi(n)$. Notice that, the number of objects derived from the objects in $cod(\varphi)$ is $|cod(\varphi)|$ during the whole computation. Moreover, these objects occur in the same region in each configuration of the computation. In what follows, we call a multiset that contains the objects derived from $cod(\varphi)$ a *clause-multiset*. The clause-multisets contain $c_{xy\to z}$ if and only if $c_{xy\to z} \in cod(\varphi)$.

$\Pi(n)$ decides in each extensional subconfiguration $ext^i_{xy\to z}$, whether z should be added to $Unit^+$ or not. If the clause multiset reaches the $l_{xy\to z}$-membrane, the following happens. The $l_{xy\to z}$-membrane gets dissolved if and only if the edge multiset contains $c_{xy\to z}$. If so then the edge multiset gets to the $l'_{xy\to z}$-membrane. Now, the $l'_{xy\to z}$-membrane gets dissolved if and only if the edge multiset contains v_x. Again, if so, the membrane dissolves and the edge multiset gets to the $l''_{xy\to z}$-membrane. Lastly, the $l''_{xy\to z}$-membrane gets dissolved if and only if the edge multiset contains v_y. If so, the membrane dissolves and the edge multiset gets to the $l'''_{xy\to z}$-membrane. During these dissolutions, none of the objects evolves. Using these three steps, $\Pi(n)$ checks whether $xy \to z \in \varphi$ and $x, y \in Unit^+$. If these conditions hold, the edge multiset reaches the $l'''_{xy\to z}$-membrane. Next, if the edge multiset contains v_z then it triggers the dissolution of the $l'''_{xy\to z}$-membrane without evolving. That is, z was already in $Unit^+$. Otherwise, if the edge multiset contains $\overline{v_z}$ then it dissolves the $l'''_{xy\to z}$-membrane while evolving to v_z representing that z is added to $Unit^+$.

If the edge multiset cannot reach the $l'''_{xy\to z}$-membrane, then one of the conditions above does not hold. In this case, some of the c-copies coming up from the l-structures will reach the undissolved membranes and dissolve them, resulting in the appearance of the edge multiset in the next extensional subconfiguration without any change.

The following proposition states that $\Pi(n)$ in $ext^i_{\mathcal{C}_j}$ implements the ith conditional extension by $\mathcal{C}_j$ of Algorithm 2.

Proposition 10. *For each $i \in [n-1]$ and $j \in [m]$, $\Delta_{i,j} \approx \Omega_{i,j}$ implies $\hat{\Delta}_{i,j} \approx \hat{\Omega}_{i,j}$.*

Correctness and running time of $\Pi(n)$. Using Proposition 10, the fact that Algorithm 2 solves HORN3SATNORM and the rules of $\Pi(n)$, one can see that $\Pi(n)$ is correct and complete.

When $ext^{n-1}_{\mathcal{C}_m}$ gets dissolved, all the extensional subconfigurations below $ext^{n-1}_{\mathcal{C}_m}$ are already dissolved. Clearly, $ext^{n-1}_{\mathcal{C}_m}$ gets dissolved in at most $k_{n-1,m} + 6$ steps. By Equation (2), $k_{n-1,m} = O(nm)$. Since $m = O(n^3)$, we get that $\Pi(n)$ halts in $O(n^4)$ steps.

The DLOGTIME-uniformity of $\mathbf{\Pi}$. Since deterministic log-time Turing machines are not able to compute the representation and the encoding of the input, in what follows, we define **DLOGTIME**-uniformity in a different way. Instead of constructing $\Pi(n)$ and the encoding $cod(\varphi)$, we give languages $L^n_{\mathbf{\Pi}}$ and L_{H3SN} of such words that describe various features of $\Pi(n)$ and $cod(\varphi)$, respectively. These features are the following: the form of the membrane structure, the multisets in the initial configurations, and the rules of $\Pi(n)$. Consider the language $L_{\mathbf{\Pi}} = \bigcup_{n=1}^{\infty} L^n_{\mathbf{\Pi}}$. $\mathbf{\Pi}$ is **DLOGTIME**-uniform if $L_{\mathbf{\Pi}} \cup L_{\text{H3SN}}$ is recognizable by a deterministic log-time Turing machine. To achieve this, we need to modify the membrane labeling in $\Pi(n)$ to be unique. Then, we need to encode the input, the membrane labels, and the objects as it is discussed in Section 4.5 of the thesis.

Therefore, we give a characterization of **P** in this way. As a consequence, if Păun's conjecture is true, then the computational power of dissolution P systems running in polynomial time remains the same even if we allow these P systems to use evolution, communication, and elementary membrane division rules, too.

4 On the power of weak non-elementary membrane division rules

We show that polarizationless P systems with active membranes are able to solve all problems in $\mathbf{PSPACE}$ even if they use only in-communication rules, dissolution rules, and *weak division rules for non-elementary membranes* of the form $[a]_h \to [b]_h[c]_h$ for some $h \in H$ and $a, b, c \in O$. Consequently, we give a characterization of $\mathbf{PSPACE}$. Our result is interesting for the following reason. It was shown in [7] that polarizationless P systems with active membranes can solve $\mathbf{PSPACE}$-complete problems in polynomial time if they can use non-elementary membrane division rules, even if in-communication is not allowed. However, in [8], a $\mathbf{P}^{\mathbf{NP}}$ upper bound was given to the computational power of polynomial-time P systems with active membranes without using in-communication rules but using weak non-elementary division rules instead of the classical ones, and using even polarizations. With our result, we show that in-communication rules are crucial to solve $\mathbf{PSPACE}$-complete problems efficiently with polarizationless P systems with active membranes that divide membranes only with the use of weak non-elementary membrane division rules.

We consider the following $\mathbf{PSPACE}$-complete variant of the QSAT problem: the input is a fully quantified QBF (*Quantified Boolean formula*) $\varphi = \exists x_1 \forall x_2 \ldots \exists x_{n-1} \forall x_n \Phi(x_1, \ldots, x_n)$ in prenex normal form, where $n \geq 2$ and Φ is in CNF. The task is to decide whether φ is *true* or *false*. Clearly, F is *true* if and only if the following holds:

$$(\textbf{exists } v_1 \in \{true, false\})(\textbf{for all } v_2 \in \{true, false\}) \ldots$$
$$\ldots (\textbf{exists } v_{n-1} \in \{true, false\})(\textbf{for all } v_n \in \{true, false\})$$
$$(\Phi(v_1, \ldots, v_n) \equiv \mathit{true}) \quad (3)$$

Algorithm 3 the decision of QSAT

```
1:  function EVAL($\Phi(x_i, \ldots, x_n), i$)                              ▷ initially, $i = 1$
2:      if ($i \leq n$) then
3:          if ($i \bmod 2 == 0$) then
4:              return EVAL($\Phi(true, x_{i+1}, \ldots, x_n), i+1$) $\wedge$ EVAL($\Phi(false, x_{i+1}, \ldots, x_n), i+1$)
5:          else
6:              return EVAL($\Phi(true, x_{i+1}, \ldots, x_n), i+1$) $\vee$EVAL($\Phi(false, x_{i+1}, \ldots, x_n), i+1$)
7:          end if
8:      else
9:          for all $C \in \Phi$ do                                   ▷ no free variables occur in $\Phi$
10:             if $true \notin C$ then
11:                 return $false$
12:             end if
13:         end for
14:         return $true$
15:     end if
16: end function
```

The main result of the section is the following.

Theorem 3. *The* QSAT *problem can be solved in polynomial time by a polynomially uniform family* $\mathbf{\Pi} = \{\Pi(n, m) \mid n \geq 2, m \geq 1\}$ *of polarizationless recognizer P systems with active membranes such that the members of* $\mathbf{\Pi}$ *employ only in-communication rules, dissolution rules and weak division rules for non-elementary membranes.*

Roughly, our P systems will implement function Eval given in Algorithm 3 which is, in fact, a variant of the well-known basic method of recursively evaluating Expression (3).

For $n \geq 2$ and $m \geq 1$, QSAT(n, m) denotes the set of those instances of QSAT which have m clauses and variables in Var_n. We use $\Pi(n, m)$ to decide whether the instances of QSAT(n, m) are true or not. The initial membrane structure of $\Pi(n, m)$ can be seen in Figures 3 and 4.

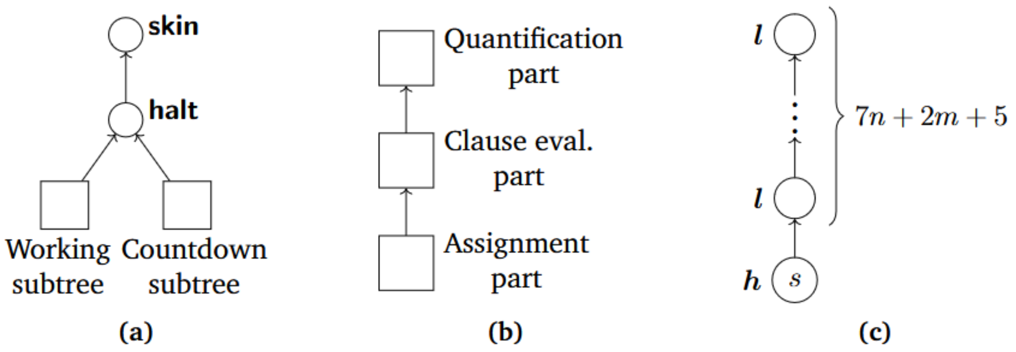

Figure 3: *The initial membrane configuration (a), the working subtree (b) and the countdown subtree (c). The rectangles denote subconfigurations and the circles denote membranes.*

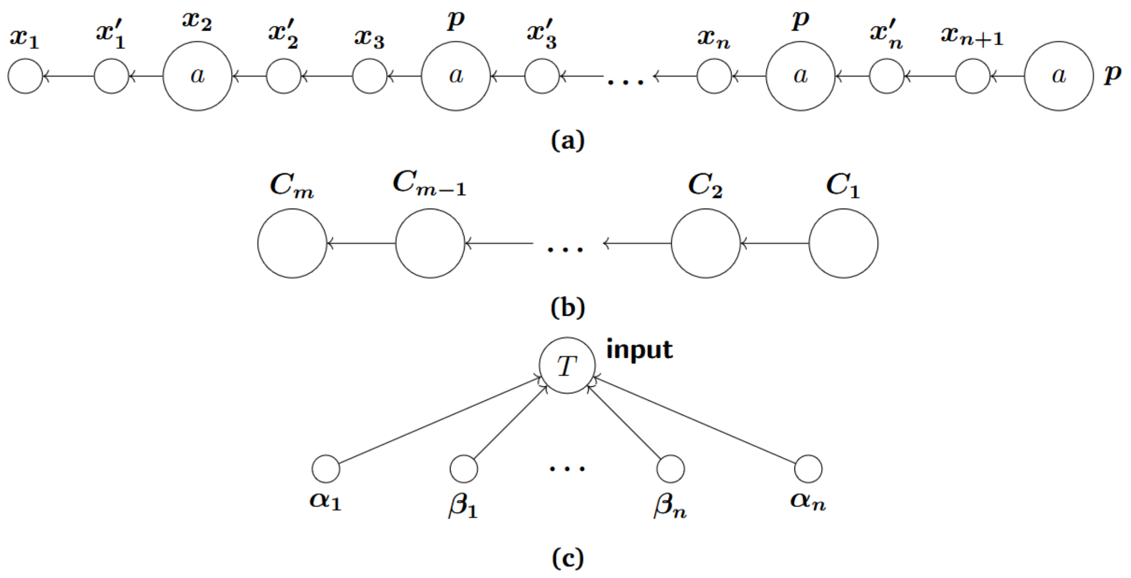

Figure 4: *The quantification (a), the clause evaluation (b), and the assignment part (c) of the working subtree of the initial membrane configuration.*

The behavior of $\Pi(n, m)$**.** First, $\Pi(n, m)$ uses weak division on n membranes in the Quantification part to create 2^n branches in the structure. Each branch is associated with a truth assignment of the variables, and contains a copy of the Clause evaluation part and a copy of the Assignment part (see Figure 5). These divisions are made one after another from top to bottom while producing objects t_i and f_i, with $i \in [n]$, separately in the two new branches created. During these divisions, $\Pi(n, m)$ uses in-communication rules to send these objects towards the elementary membranes where the objects of the input multiset are located. Then, using the objects of input multiset and the appropriate dissolution rules, $\Pi(n, m)$ determines in each branch what literals are true under the corresponding truth assignment. After that, in the Clause evaluation part of each branch, $\Pi(n, m)$ decides whether all clauses are true in the quantifier-free part of φ or not, according to the true literals. At this point, an object T appears in a leaf of the Quantification part (having 2^n leaves) if and only if the corresponding truth assignment satisfies the quantifier-free part of φ. In the Quantification part, each pair of the levels of the tree is associated with a variable x_i and its quantifier Q_i (again, see Figure 5). The inner membranes are labeled with x'_i and the outer membranes with x_i. Each odd pair of levels is associated with an existential quantifier, and each even pair of levels is associated with a universal quantifier. Thus, if i is an odd number, using dissolution rules, $\Pi(n, m)$ releases an object T from the x_i-membrane to the x'_{i-1}-membrane if and only if at least one object T came from the x_{i+1}-membranes. Similarly, if i is even, $\Pi(n, m)$ releases an object T to the x'_{i-1}-membrane if and only if exactly two objects T came from the two x_{i+1}-membranes, respectively. That is, the suffix of φ that starts with $Q_i x_i$ is true. At the end of the computation, some objects T reach the halt-membrane if and only if φ is true. If so, object yes appears in the skin-membrane, the output membrane.

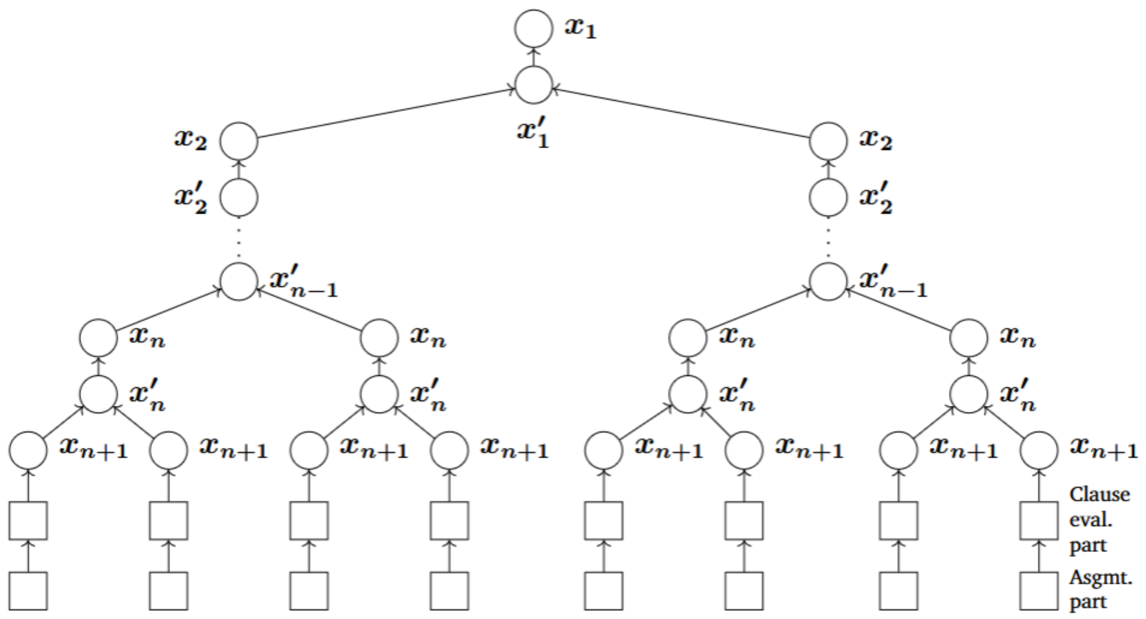

Figure 5: *The shape of the working subtree after that all the divisions are done. The rectangles denote subconfigurations and the circles denote membranes.*

Conclusions. We showed that polarizationless P systems with active membranes employing only in-communication, dissolution, and weak non-elementary membrane division rules can solve the **PSPACE**-complete QSAT problem efficiently. It is known that polarizationless P systems with active membranes can solve exactly the problems in **PSPACE**

in polynomial time. Using our result, we get that these P systems characterize $\mathbf{PSPACE}$.

5 Accepting conditions in P systems with active membranes

We introduce *elimination P systems* that are polarizationless P systems with active membranes extended with rules of the form $[ab \to \varepsilon]_h$, where a and b are objects, and ε denotes the empty word. We call these rules *elimination* rules. Moreover, we present *extended acknowledger P systems* that are more general in terms of acceptance conditions than their recognizer counterpart. We will show that - along with some other important results - there are problems that recognizer elimination P systems cannot solve, but extended acknowledger elimination P systems can.

Extended acknowledger P systems. A P system Π is an *acknowledger* P system if (1) Π has a designated input membrane and a designated output membrane, (2) the alphabet of objects has a designated element yes, (3) Π has no rules that contain yes on the left-hand side, and (4) all computations of Π halt. Let $\mathfrak{C}$ be a computation of Π. The output of Π related to $\mathfrak{C}$ is the multiset yes^t, where $t \in \mathbb{N}$ is the number of instances of yes in the output membrane of the halting configuration of $\mathfrak{C}$. Then $\mathfrak{C}$ is accepting if $t \neq 0$, otherwise $\mathfrak{C}$ is rejecting. we consider a generalization of acknowledger P systems. We call a P system Π an *extended acknowledger* P system if Π satisfies only Conditions 1,2 and 4.

Counting P systems. A P system Π is a *counting* P system if Π has a designated input membrane and a designated output membrane, the alphabet of objects has a designated *output object*, and all computations of Π halt. Let $\mathfrak{C}$ be a computation of Π. The output of Π related to $\mathfrak{C}$ is the multiset σ^t, where σ is the output object and $t \in \mathbb{N}$ is the number of instances of σ in the output membrane of the halting configuration of $\mathfrak{C}$. An $\mathbf{L}$-uniform family $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ of counting P systems solves a counting problem R, if the basic conditions of $\mathbf{L}$-uniformity hold, moreover, for every instance x of R the output of each computation of $\Pi(n)$ starting with $cod(x)$ in its input membrane is σ^t, $t \in \mathbb{N}$, if and only if t is the solution of x.

The MAJORITY-SAT **and the** #SAT **problems.** The well-known $\mathbf{NP}$-complete problem SAT sounds as follows: given a propositional formula in CNF, decide whether it is satisfiable or not. We consider two extensions of this problem: MAJORITY-SAT and #SAT. In the case of MAJORITY-SAT, the task is to determine for a given formula φ whether the majority of all truth assignments satisfy φ or not. MAJORITY-SAT is a $\mathbf{PP}$-complete problem. Whereas MAJORITY-SAT is a decision problem, #SAT is a $\#\mathbf{P}$-complete counting problem, which sounds as follows: given a formula φ, count all the satisfying truth assignments of φ. Notice that the instances of SAT, MAJORITY-SAT and #SAT are the same.

To simplify, hereafter, when we refer to solving a problem using P systems, we imply that the problem can be solved in polynomial time by an $\mathbf{L}$-uniform family of these P systems. For a given class $\mathcal{P}$ of P systems, $\mathbf{L} - \mathbf{PMC}^{R}_{\mathcal{P}}$ (respectively, $\mathbf{L} - \mathbf{PMC}^{EA}_{\mathcal{P}}$ and $\mathbf{L} - \mathbf{PMC}^{C}_{\mathcal{P}}$) denotes the family of problems that can be solved by recognizer (respectively, extended acknowledger and counting) P systems in $\mathcal{P}$. Denote $\mathcal{E}_{-d}$ the class of elimination P systems with no dissolution rules. The following theorem states that recognizer elimination P systems without dissolution rules cannot solve any problem beyond $\mathbf{NL}$.

Theorem 4. $\mathbf{L}-\mathbf{PMC}^{R}_{\mathcal{E}_{-d}} \subseteq \mathbf{NL}$.

Now, denote $\mathcal{E}_{-c}$ the class of elimination P systems with no communication rules. We show that counting elimination P systems are able to solve #SAT, that is, all problems in $\#\mathbf{P}$ even without communication rules.

Theorem 5. $\#\mathbf{P} \subseteq \mathbf{L}-\mathbf{PMC}^{C}_{\mathcal{E}_{-c}}$.

Next, we show that the P systems defined in the proof of Theorem 5 can be generalized to solve $\mathbf{PP}$-complete problems.

Theorem 6. $\mathbf{PP} \subseteq \mathbf{L}-\mathbf{PMC}^{R}_{\mathcal{E}_{-c}}$.

In what follows, we show that elimination P systems without dissolution rules are capable of counting the satisfying truth assignments of a formula. Furthermore, we show that by employing the acceptance condition of extended acknowledger P systems, these P systems can also solve the MAJORITY-SAT problem.

Theorem 7. $\#\mathbf{P} \subseteq \mathbf{L}-\mathbf{PMC}^{C}_{\mathcal{E}_{-d}}$

Theorem 8. $\mathbf{PP} \subseteq \mathbf{L}-\mathbf{PMC}^{EA}_{\mathcal{E}_{-d}}$.

Looking at Theorem 4 and Theorem 8, we can conclude that, in certain cases, extended acknowledger P systems are more powerful in terms of computational power than recognizer P systems.

Conclusions. This section leaves several interesting questions unanswered. For example, are acknowledger P systems without dissolution rules capable of solving MAJORITY-SAT efficiently? Recall that in acknowledger P systems, no rule can be applied on the output symbol yes. Furthermore, are the $\mathbf{PP}$-lower bounds established in this section tight or not? Addressing these questions is part of our future research plan.

Bibliography

Journal publications of the author

[1] Gazdag, Zs., **Hajagos, K.**, Iván, Sz.: On the power of P systems with active membranes using weak non-elementary membrane division. *Journal of Membrane Computing*, **3**, 258–269 (2021)

[2] Gazdag, Zs., **Hajagos, K.**: On the power of membrane dissolution in polarizationless P systems with active membranes. *Natural Computing*, 22(1), 1-10 (2022).

[3] Gazdag, Zs., **Hajagos, K.**: A characterisation of P by DLOGTIME-uniform families of polarizationless P systems using only dissolution rules. *Theoretical Computer Science*, 965(3), (2023)

[4] Gazdag, Zs., **Hajagos, K.**: On accepting conditions in P systems with active membranes. *Journal of Membrane Computing*- Accepted, soon to be published - (2025)

Further related publications of the author

[5] Gazdag, Zs., **Hajagos, K.**, Iván, Sz.: On the number of useful objects in P systems with active membranes. https://arxiv.org/abs/2008.04993 (2020)

[6] Gazdag, Zs., **Hajagos, K.**: A characterisation of **P** by polarizationless P systems using only membrane dissolution rules. Presented at the 23rd Conference on Membrane Computing (CMC 2022), 6-9 September 2022, Trieste, Italy

Other references

[7] Alhazov, A., Pérez-Jiménez, M.J.: Uniform solution of QSAT using polarizationless active membranes. International Conference on Machines, Computations and Universality, 122-133 (2007)

[8] Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Monodirectional P systems. Natural Computing, 15, 551–564 (2016)

[9] Păun, Gh.: Further twenty six open problems in membrane computing. In: Third Brainstorming Week on Membrane Computing. Fénix Editora, Sevilla, 249–262 (2005)

Összefoglalás

A disszertáció a polarizációmentes aktív membrános P rendszerek számítási erejére vonatkozóan mutat be eredményeket. A dolgozat összesen hat fejezetből áll.

Az első fejezetben ismertetjük a membránszámítás alapfogalmait, majd az aktív membrános P rendszerek informális bemutatása után beszámolunk az ezen P rendszerekhez kapcsolódó néhány jól ismert eredményről, amik kijelölték a szerző számára a kutatási irányokat.

A második fejezetben azokat az alapfogalmakat vesszük át, melyek a dolgozat tartalmának a megértéséhez szükségesek. Ennek keretében formálisan definiáljuk a polarizációmentes aktív membrános P rendszereket, valamint részletesen bemutatjuk azoknak működését.

A harmadik fejezetben ismertetjük az első eredményünket, miszerint kizárólag membránfeloldó szabályokat használó polarizációmentes aktív membrános P rendszereknek **L**-uniform családjával polinomiális időben megoldható minden probléma az **NL** osztályon belül. Ennek megmutatásához az **NL**-teljes ELÉRHETŐSÉG problémát oldjuk meg ilyen P rendszerekkel.

A negyedik fejezetben a fent említett P rendszerek számítási erejére egy bővebb, **P** alsó korlátot adunk meg. Ráadásul ezt úgy tesszük, hogy sokkal korlátozottabb, **DLOGTIME**-uniformizálási feltételeket alkalmazunk. Ismert, hogy az elemi membránosztó szabályok nélkül működő polinomiális-idejű polarizációmentes aktív membrános P rendszerek számítási erejének a felső korlátja **P**. Tehát ezzel **P** egy jellemzését adjuk meg.

Az ötödik fejezetben a gyenge nemelemi membránosztó szabályok erejét vizsgáljuk. Megmutatjuk, hogy a kizárólag befelé kommunikáló, membránfeloldó és gyenge nemelemi membránosztó szabályokat használó, polinomiális-idejű polarizációmentes aktív membrános P rendszereknek a számítási erejére **PSPACE** egy felső korlát. Ismert, hogy a polinomiális-idejű polarizációmentes aktív membrános P rendszerek számítási erejének egy felső korlátja **PSPACE**, még akkor is, ha a klasszikus nemelemi membránosztási szabályok használata is megengedett. Ez ugyanúgy fennáll akkor is, ha ezen klasszikus szabályok helyett gyenge nemelemi membránosztó szabályokat használunk. Ebből következik, hogy a **PSPACE** osztálynak egy jellemzését definiáljuk.

A hatodik fejezetben a felismerő P rendszereknél általánosabb elfogadási feltételekkel bíró kiterjesztett nyugtázó P rendszerekkel, valamint számlálási problémák megoldására használt számláló P rendszerekkel foglalkozunk. Ezek mellett bevezetjük az elimináló P rendszereket, melyek olyan polarizációmentes aktív membrános P rendszerek, melyek az ugyancsak ebben a fejezetben bevezetett elimináló szabályokat is alkalmazhatják. Öt eredményt ismertetünk eldöntési és számlálási problémák megoldásáról egyaránt. A legfontosabb eredmény, hogy míg bizonyos felismerő P rendszerek számítási erejének egy felső korlátja **NL**, addig ugyanazon szabályokat alkalmazó kiterjesztett nyugtázó P rendszerek esetén **PP** egy felső korlát. Ebből látható, hogy a számítási erőt olykor az elfogadási feltételek is korlátozhatják.

Társszerzői nyilatkozat

Hajagos Károly „Results on the computational power of polarizationless P systems with active membranes" című PhD disszertációjában a következő eredményekben Hajagos Károly hozzájárulása volt a meghatározó:

A disszertáció **3. fejezetében** és a:

- Zs. Gazdag, K. Hajagos: On the power of membrane dissolution in polarizationless P systems with active membranes. *Natural Computing*, 22(1), 1-10, (2022)

cikkben szereplő eredményekre vonatkozóan:

- Az implementált algoritmus megalkotása és a központi tétel bizonyításában szereplő, a vizsgált problémát megoldó L-uniform P rendszer család megkonstruálása, valamint helyességének megmutatása közösen történt a témavezetővel.

A disszertáció **4. fejezetében** és a

- Zs. Gazdag, K. Hajagos: A characterisation of P by DLOGTIME-uniform families of polarizationless P systems using only dissolution rules. *Theoretical Computer Science*, 965(3), (2023)

cikkben szereplő eredményekre vonatkozóan:

- Az implementált algoritmus megalkotása és a központi tétel bizonyításában szereplő, a vizsgált problémát megoldó L-uniform P rendszer család megkonstruálása, valamint helyességének megmutatása közösen történt a témavezetővel.
- A fenti P rendszer család DLOGTIME-uniform P rendszer családdá történő átalakításában a disszertáció szerzőjének döntő szerepe volt.

A disszertáció **5. fejezetében** és a

- Zs. Gazdag, K. Hajagos and Sz. Ivan. On the power of P systems with active membranes using weak non-elementary membrane division. *Journal of Membrane Computing*, 3(2-3), 258-269, (2021)

cikkben szereplő eredményekre vonatkozóan:

- Az implementált algoritmus megalkotása és a központi tétel bizonyításában szereplő, a vizsgált problémát megoldó P-uniform P rendszer család megkonstruálása, valamint helyességének megmutatása közösen történt a szerzőtársakkal.

A disszertáció **6. fejezetében** és a

- Zs. Gazdag, K. Hajagos: On accepting conditions in P systems with active membranes. *Journal of Membrane Computing* - Accepted, soon to be published - (2025)

cikkben szereplő eredményekre vonatkozóan:

- A dolgozatban 4. és a cikkben 1. tétel bizonyításában a disszertáció szerzőjének csekély szerepe volt.
- A dolgozatban 5. és a cikkben 2. tétel bizonyítása közösen történt a témavezetővel.
- A dolgozatban 6. és a cikkben 3. tétel bizonyítása közösen történt a témavezetővel.
- A dolgozatban 7. és a cikkben 4. tétel bizonyításában a disszertáció szerzőjének döntő szerepe volt.
- A dolgozatban 8. és a cikkben 5. tétel bizonyítása közösen történt a témavezetővel.

Ezek az eredmények Hajagos Károly PhD disszertációján kívül más tudományos fokozat megszerzésére nem használhatók fel.

Szeged, 2025.01.20.

Hajagos Károly
Jelölt

[illegible]
Témavezető

Az Informatika Doktori Iskola vezetője kijelenti, hogy jelen nyilatkozatot minden társszerzőhöz eljuttatta, és azzal szemben egyetlen társszerző sem emelt kifogást.

Szeged, 2025. 01. 28.

SZEGEDI TUDOMÁNYEGYETEM INFORMATIKA DOKTORI ISKOLA SZEGED

[illegible]
Doktori Iskola vezető