On the power of P systems with active membranes using weak non-elementary membrane division

Zsolt Gazdag¹ · Károly Hajagos¹ · Szabolcs Iván¹

Received: 5 May 2021 / Accepted: 3 September 2021 / Published online: 5 October 2021
© The Author(s) 2021

Abstract

It is known that polarizationless P systems with active membranes can solve PSPACE-complete problems in polynomial time without using in-communication rules but using the classical (also called strong) non-elementary membrane division rules. In this paper, we show that this holds also when in-communication rules are allowed but strong non-elementary division rules are replaced with weak non-elementary division rules, a type of rule which is an extension of elementary membrane divisions to non-elementary membranes. Since it is known that without in-communication rules, these P systems can solve in polynomial time only problems in P^{NP} , our result proves that these rules serve as a borderline between P^{NP} and PSPACE concerning the computational power of these P systems.

Keywords Membrane computing · P systems with active membranes · Non-elementary membrane division · PSPACE-completeness

1 Introduction

The investigation of the computational power of variants of P systems with active membranes [26] is a widely studied research area in membrane computing. It is well known that these P systems can solve PSPACE-complete problems in polynomial time if they can use non-elementary membrane division rules of the form $[[]_{h_1}^+ []_{h_2}^-]_{h_0}^0 \rightarrow [[]_{h_1}^+]_{h_0}^0 []_{h_2}^-]_{h_0}^0$, where h_0 , h_1 , and h_2 are labels of the membranes [3, 31]. Following the literature, we call these rules *strong* non-elementary membrane division rules in this paper. It is also known that P systems with active membranes can solve NP-complete problems in polynomial time without employing any non-elementary membrane division rules [12, 26, 35]. Solving NP-complete problems by P systems with active membranes has a huge literature in membrane computing

(from now on, by a solution we mean a polynomial-time solution). The NP-complete SAT problem, i.e., the satisfiability problem of Boolean formulas was solved, for example, without polarizations but using membrane label changing [4] or using separation rules instead of division rules [23, 24]. In [7], the authors solved SAT using a method different from the one commonly used in the literature (see also [5, 6]). In fact, it is known that P systems with active membrane employing no non-elementary membrane division rules characterise the complexity class $P^{#P}$, that is, the class of all problems solved by polynomial-time deterministic Turing machines with polynomial-time counting oracles [13]. We note here that in [17], the authors presented a characterization of PSPACE by P systems employing no non-elementary membrane division rules, but the P systems used there were non-confluent (i.e., strongly nondeterministic). In this paper, we consider only confluent P systems.

It is also known that without any membrane division rules, P systems with active membranes can solve only problems in P [35]. It is natural to ask what components other than membrane division rules are necessary for these P systems to solve computationally hard problems. Păun conjectured already in 2005 that P systems without non-elementary membrane division rules can solve only problems in P if we remove the polarizations of the membranes [25]. So far, this conjecture was confirmed only in some special cases,

✉ Zsolt Gazdag
gazdag@inf.u-szeged.hu

Károly Hajagos
hajagos@inf.u-szeged.hu

Szabolcs Iván
szabivan@inf.u-szeged.hu

¹ Institute of Informatics, University of Szeged, Árpád tér 2,
Szeged 6720, Hungary

for example, when dissolution rules are not used [11], when the division rules are symmetric [18], or when dissolution and elementary membrane division rules are allowed but both the use of other types of rules and the initial membrane structure are restricted [9, 15, 34]. The aim to prove Păun's conjecture initiated a research line in membrane computing where the computational power of restricted variants of polarizationless P systems is investigated (see, e.g., [19–21], where these P systems with no dissolution rules were studied and [8], where it is shown that these systems can simulate Turing machines efficiently using only evolution and dissolution rules). For a recent survey on exploring the boundary between P and NP in terms of membrane computing, see, e.g., [22].

Polarization, on the other hand, is known to be not necessary to solve computationally hard problems when the use of non-elementary membrane division rules is allowed. For example, the PSPACE-complete QSAT problem (QSAT is the problem of deciding if a fully quantified Boolean formula is true or not) was solved in [1] without using polarizations. In fact, the construction presented in [1] does not employ in-communication rules either, and out-communication rules are used only in the last step of the computation to send the answer of the system out to the environment. Moreover, the employed strong non-elementary membrane division rules are also restricted as they have the form $[[]_{h_1} []_{h_1}]_{h_0} \rightarrow [[]_{h_1}]_{h_0} [[]_{h_1}]_{h_0}$, that is, the inner membranes in the left-hand side of these rules have the same labels.

It is known [11] that dissolution rules are necessary to solve QSAT in the P systems considered in [1]. To the best of our knowledge, it is still open whether evolution rules are necessary, too. Notice that in [1], the evolution rules had an essential role in selecting those clauses of the input formula that are satisfied by a truth assignment of the variables. Although dissolution and elementary membrane division rules can evolve objects, it is not clear how they could be used to take over the roles of evolution rules in [1].

In [16], the Q3SAT problem, a PSPACE-complete restriction of QSAT was solved with polarizationless P systems using no evolution and communication rules. The presented P systems used however not only strong non-elementary membrane division rules, but so-called weak division rules for non-elementary membranes. These rules are in fact generalisations of the well-known elementary membrane division rules to non-elementary membranes and thus have the form $[a]_h^{e_0} \rightarrow [b]_h^{e_1} [c]_h^{e_2}$, where e_0, e_1, e_2 are polarizations (see also [2, 36], where the space complexity of P systems employing weak non-elementary membrane division rules is discussed). Let us note here that, to the best of our knowledge, it is not known whether weak non-elementary membrane division rules are indeed weaker than strong non-elementary membrane division rules. More precisely, it is not known whether we can find a set C of types of active

membrane rules containing weak non-elementary membrane divisions but excluding strong divisions such that polynomial-time P systems using rules of types in C can decide presumably less problems than those P systems that can use strong division rules in addition to rules of types in C .

Recently, monodirectional P systems were introduced and investigated [14]. These are P systems with active membranes that have no in-communication rules and have weak non-elementary membrane division rules instead of the strong ones. It is easy to see that in these P systems the information can flow only towards the skin membrane. It turned out in [14] that this syntactical restriction is enough to restrict the computational power of these P systems. More precisely, it was shown in [14] that monodirectional P systems working in polynomial time characterize the class P^{NP} , that is, the class of problems solved by polynomial-time Turing machines with NP oracles. Interestingly, it turned out also that removing weak non-elementary membrane division rules or dissolution rules does not affect the computational power of these P systems. On the other hand, if both of these rules are removed, then the P systems characterise only $P_{\parallel}^{NP}$. For a recent survey on the relation of classical complexity classes beyond NP and classes of problems solvable by various families of P systems with active membranes, see [32].

In this paper, we show that weak non-elementary membrane division is enough to solve PSPACE-complete problems in polarizationless P systems if dissolution and in-communication rules are allowed. More precisely, we show that the QSAT problem can be solved with polarizationless P systems with active membranes where the applicable types of rules are weak non-elementary membrane division, dissolution, and in-communication. As we will not use out-communication rules, the answer of the system will appear in the skin membrane in the last step of the computation. All types of rules used in our P systems are presumably necessary to solve the QSAT problem. Indeed, as we have already discussed, without in-communication, dissolution, or weak non-elementary division these systems can solve presumably less problems due to [11], [14], and [35], respectively (notice that although in [11] the P systems employ strong non-elementary membrane divisions, their result can easily be adopted to those P systems which have weak non-elementary divisions instead of the strong ones).

Our proof resembles the one given in [1]. That solution of QSAT first builds a binary tree of membranes starting from a linearly nested membrane structure. Each elementary membrane in this tree corresponds to a truth assignment of the variables of the input. Our solution builds a similar tree of membranes, but a truth assignment is associated with a set of elementary membranes instead of one. However, in [1], this binary tree is built in a bottom-up manner, that is, the strong non-elementary division rules divide the innermost membrane first and the division is propagated towards the

skin membrane. Meanwhile, the mentioned truth assignments are created using elementary membrane division rules, that is, they appear at the leaves of the created binary tree. On the other hand, the weak non-elementary division rules used in our paper create a binary tree of the membranes in a top-down manner: first, one of the upper membranes of the tree is divided and the division continues towards the elementary membranes. The weak non-elementary divisions are used to create the objects representing the truth assignments as well. These objects then are sent to the elementary membranes using in-communication rules. This means that in-communication rules play an important role in our result. In fact, as we have discussed above, QSAT cannot be solved without in-communication rules with our P systems unless $PSPACE = P^{NP}$.

The rest of the paper is organised as follows. In the next section, we present the necessary notions and notations. Then, in Sect. 3, we give the main result of the paper. In Sect. 4, we give some concluding remarks.

2 Preliminaries

Here, we recall the necessary notions used in the paper. However, we assume that the reader is familiar with the basic concepts of formal language theory, propositional logic, and membrane computing techniques (for a comprehensive guide to these topics, see e.g. [10, 27, 30], respectively). We denote by $\mathbb{N}$ the set of natural numbers including zero. For $n, i \in \mathbb{N}$ with $0 \leq i \leq n$, $[i, n]$ denotes the set $\{i, i+1, \dots, n\}$, and we use the notation $[n]$ for the set $[1, n]$.

P systems and membrane configurations. In this paper, we examine *polarizationless* P systems with active membranes, that is, such P systems where no electrical charge is associated with any membrane. Before giving a formal definition of these P systems, we define first *membrane configurations* based on the notion of membrane structures. Intuitively, a *membrane structure* consists of membranes hierarchically embedded in the outermost membrane, called the skin membrane, and membranes delimit *regions*. Let O be an alphabet of objects and H be a finite set of membrane labels. We assume that H always contains the special label skin. 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 $L : V \rightarrow H$ assigns labels to each node, such that only the root is labeled by the symbol skin, and it has to be labeled by skin. Nodes are called *membranes* of the structure. A membrane $x \in V$ with label ℓ is called an ℓ -*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*.

A *membrane configuration* is a tuple (V, E, L, ω) , where (V, E, L) is a membrane structure and $\omega : V \rightarrow O^*$ is a

function which assigns a finite word of objects to each membrane. We view these words as *multisets* of O . The empty word is denoted by ε . If x is a membrane and $a \in \omega(x)$, then we say that the *region of x contains a* , or just that x *contains a* . Let $C = (V, E, L, \omega)$ be a membrane configuration. The membrane configuration $C' = (V', E', L', \omega')$ is a subconfiguration of C if (V', E') is a subtree of (V, E) , and L' and ω' are restrictions of L and ω to V' , respectively.

A *polarizationless P system with active membranes* is a construct of the form $\Pi = (O, H, \mu, R)$, where O is the alphabet of *objects*, H is a finite set of *membrane labels*, $\mu = (V, E, L, \omega)$ is the initial membrane configuration with L and ω mapping nodes of μ to H and O^* , respectively, and R is a finite set of *rules*. We allow the following types of rules:

- (a) $[a \rightarrow v]_h$, for some $h \in H, a \in O, v \in O^*$ (*object evolution rules*);
- (b) $a[]_h \rightarrow [b]_h$, for some $h \in H, a, b \in O$ (*in-communication rules*);
- (c) $[a]_h \rightarrow []_h b$, for some $h \in H, a, b \in O$ (*out-communication rules*);
- (d) $[a]_h \rightarrow b$, for some $h \in H, a, b \in O$ (*membrane dissolving rules*);
- (e) $[a]_h \rightarrow [b]_h[c]_h$, for some $h \in H, a, b, c \in O$ (*weak division rules* for elementary or non-elementary membranes).

Rules of type (e) are generalisations of the classical elementary membrane division rules to non-elementary membranes with the following semantics. Consider a membrane configuration $C = (V, E, L, \omega)$, a rule r of type (e), and an edge $(x, y) \in E$ such that x is a subject of r (that is, $L(x) = h$) and $a \in \omega(x)$. Let (V', E') be that subtree of (V, E) which has x as the root. The application of r to x duplicates the subtree (V', E') into two disjoint trees (V'_1, E'_1) and (V'_2, E'_2) with roots x_1 and x_2 , respectively, and replaces the edge (x, y) with edges (x_1, y) and (x_2, y) . Each node n of V'_1 (resp. of V'_2) is associated with the same label as that of the corresponding node in V' . Moreover, x_1 (resp. x_2) is associated with $(\omega(x) - \{a\}) \cup \{b\}$ (resp. with $(\omega(x) - \{a\}) \cup \{c\}$), and the rest of the nodes in V'_1 (resp. in V'_2) are associated with the same object multiset as that of the corresponding node in V' .

As it is usual in membrane computing, P systems with active membranes work in a *maximally parallel* manner: at each step, the system first non-deterministically assigns appropriate rules to the occurrences of objects of the current membrane configuration such that the assigned multiset S of rules satisfies the following properties: (i) at most one rule from S is assigned to any occurrence of an object in the membrane configuration, (ii) a membrane can be the subject of at most one rule of types (b)–(e) in S , and (iii) S is maximal among the multisets of rules satisfying (i) and (ii). Then, the rules in S are applied in the usual bottom-up

manner during the computation step: when a weak division rule is applied to a membrane x , then all the rules assigned to the objects in the subtree with root x are assumed to be applied already.

Uniform families of recognizer P systems. Recognizer P systems [28, 29] were introduced to provide a framework to be able to consider P systems as *deciders* of decision problems.

Definition 1 A P system Π is a recognizer P system if it satisfies the following properties:

1. Π has a designated *input membrane*,
2. the alphabet of objects has two designated elements *yes* and *no*,
3. all computations of Π halt, and
4. each computation of Π must send out to the environment the output of the system which is either *yes* or *no* (but not both), and these objects are produced exactly in the last step of the computation.

Recognizer P systems, being non-deterministic computation models, can have different computations with different output on the same input. However, to use them as deciders, *confluent* recognizer P systems are usually considered. In these P systems all halting computations on the same input must produce the same output.

Decision problems in membrane computing are usually solved by *polynomially uniform* families of recognizer P systems. A family $\Pi = \{\Pi(n) \mid n \in \mathbb{N}\}$ of P systems is *polynomially uniform* if there is a polynomial-time deterministic Turing machine that computes $\Pi(n)$ whenever it is started with 1^n , that is, with the unary representation of $n \in \mathbb{N}$ on its input tape.

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

In this paper, we will consider P systems with no out-communication rules. Therefore, we alter Condition 4 in Definition 1 as follows:

4 each computation of Π must produce in the skin-membrane the output of the system which is either *yes* or *no* (but not both), and these objects are produced exactly in the last step of the computation.

The QSAT problem. In this paper, we are going to construct P systems that can decide if a *Quantified Boolean formula* (QBF) is true or not. QBFs extend propositional logic by allowing universal and existential quantifiers over propositional variables. It is assumed that each variable is under the bound of at most one quantifier. We consider fully quantified QBFs in *prenex normal form*, that is, in the form $F = Q_1x_1 \dots Q_nx_n\Phi(x_1, \dots, x_n)$, where $n \geq 1$, Q_i ($i \in [n]$) is either $\exists$ or $\forall$, and $\Phi(x_1, \dots, x_n)$ is the quantifier-free part of F indicating that the variables of Φ are in $\{x_1, \dots, x_n\}$.

Let $\Phi(x_1, \dots, x_n)$ be a quantifier-free formula, $i \in [n]$, and, for every $j \in [i]$, $v_j \in \{\text{true}, \text{false}\}$. Then $\Phi(v_1, \dots, v_i, x_{i+1}, \dots, x_n)$ is that formula $\Phi'(x_{i+1}, \dots, x_n)$ which we get from Φ by replacing, for every $j \in [i]$, the variable x_j by v_j . Clearly, the truth values *true* and *false* serve in Φ' as logical constants with the corresponding truth values.

In this paper, we consider the following PSPACE-complete variant of the QSAT problem: the input is a fully quantified QBF

$$F = \exists x_1 \forall x_2 \dots \exists x_{n-1} \forall x_n \Phi(x_1, \dots, x_n),$$

in prenex normal form, where $n \geq 2$ and Φ is in conjunctive normal form. The task is to decide whether F is *true* or *false*. Clearly, F is *true* if and only if the following holds:

(**exists** $v_1 \in \{\text{true}, \text{false}\}$)(**for all** $v_2 \in \{\text{true}, \text{false}\}$) ...

(**exists** $v_{n-1} \in \{\text{true}, \text{false}\}$)(**for all** $v_n \in \{\text{true}, \text{false}\}$)

($\Phi(v_1, \dots, v_n) \equiv \text{true}$).

(1)

We will show that QSAT can be solved by polarizationless P systems with active membranes using only weak non-elementary division, in-communication and dissolution rules. Roughly, our P systems will implement a function EVAL given in Algorithm 1 which is in fact a variant of the well-known basic method of recursively evaluating Expression (1).

Algorithm 1 the decision of QSAT

```

1: function EVAL( $\Phi(x_1, \dots, x_n), i$ ) ▷ initially,  $i = 1$ 
2:   if ( $i \leq n$ ) then
3:     if ( $i \bmod 2 == 0$ ) then
4:       return EVAL( $\Phi(\text{true}, x_{i+1}, \dots, x_n), i+1$ )  $\wedge$  EVAL( $\Phi(\text{false}, x_{i+1}, \dots, x_n), i+1$ )
5:     else
6:       return EVAL( $\Phi(\text{true}, x_{i+1}, \dots, x_n), i+1$ )  $\vee$  EVAL( $\Phi(\text{false}, x_{i+1}, \dots, x_n), i+1$ )
7:     end if
8:   else
9:     for all  $C \in \Phi$  do ▷ no free variables occur in  $\Phi$ 
10:      if  $\text{true} \notin C$  then
11:        return false
12:      end if
13:    end for
14:    return true
15:  end if
16: end function

```

3 Results

This section is devoted to the proof of the main result of the paper which can be stated as follows:

Theorem 1 *The QSAT problem can be solved in polynomial time by a polynomially uniform family $\Pi = \{\Pi(n, m) \mid n \geq 2, m \geq 1\}$ of polarizationless recognizer P systems with active membranes such that the members of Π employ only rules of type (b), (d), and (e).*

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 $\{x_1, x_2, \dots, x_n\}$. We will use $\Pi(n, m)$ to decide whether the instances of QSAT(n, m) are true or not.

For the rest of this section, we fix an instance

$$F = \exists x_1 \forall x_2 \dots \exists x_{n-1} \forall x_n \Phi(x_1, \dots, x_n),$$

of QSAT(n, m). Denote C_j ($j \in [m]$) the j th clause of Φ . First, we define the encoding of F , denoted by $\text{cod}(F)$, as a subset of $\Sigma(n, m) = \{v_{ij}, v'_{ij} \mid i \in [n], j \in [m]\}$ in the following way:

$$\text{cod}(F) = \{v_{ij} \mid x_i \text{ occurs in } C_j\} \cup \{v'_{ij} \mid \neg x_i \text{ occurs in } C_j\}.$$

We now construct the P system $\Pi(n, m) = (O, H, \mu, R)$ as follows:

- $O = \{t_i, f_i, t'_i, f'_i, \hat{t}_i, \hat{f}_i \mid i \in [n]\} \cup \Sigma(n, m) \cup \{a, T, s, \text{yes}, \text{no}\},$
- $H = \{x_i, x'_i, \alpha_i, \beta_i \mid i \in [n]\} \cup \{C_j \mid j \in [m]\} \cup \{p, l, h, x_{n+1}, \text{skin}, \text{halt}, 0\},$ and
- the initial membrane configuration (μ) and the set of rules (R) are defined below.

The initial membrane configuration $\mu = (V, E, L, \omega)$ can be seen in Fig. 1. The labels of the membranes are placed beside the nodes, and the multisets associated with the nodes are written in the nodes. Notice that some nodes denote subconfigurations that are explained separately. The root of the membrane structure is the skin which has an only child with label halt. Below halt there are two subconfigurations called the *working subtree* and the *countdown subtree*, respectively (see Fig. 1a). The working subtree (Fig. 1b) will be used to decide if the input formula F is true. More precisely, this

Fig. 1 The initial membrane configuration (a), the working subtree (b) and the countdown subtree (c)

part will produce at least one object T in the halt-membrane if and only if F is *true*. Then T triggers the dissolution of the halt-membrane resulting *yes* in the skin. The countdown subtree (Fig. 1c) will be used to produce an object s in the halt-membrane if and only if F is *false*, that is, no object T was produced in the halt-membrane by the working subtree. Then, s triggers the dissolution of the halt-membrane resulting *no* in the skin. The countdown subtree has $7n + 2m + 5$ linearly nested l -membranes and an elementary h -membrane in the innermost l -membrane. Initially, all the l -membranes are empty, and for the only h -membrane x , $\omega(x) = s$.

The working subtree is divided into three parts: the *quantification*, the *clause evaluation*, and the *assignment* parts. We now describe the structure of these parts and give a brief, general description of their role in the whole computation.

A more detailed explanation about their operation will be given later.

The quantification part will be used to create exponentially many subtrees of the membrane structure, each of them containing objects that encode a possible valuation of the variables of F . In this part, we have membranes with unique labels in $\{x_i, x'_i \mid i \in [n]\} \cup \{x_{n+1}\}$ and several further membranes with label p . The membrane structure of the quantification part can be seen in Fig. 2a. It is a single path with a root that is labeled by x_1 , and a leaf that is the innermost p -membrane. Figure 2a shows the labels of the other nodes in this part. Notice that there is a regularity in the labelling below the x_2 -membrane: for nodes x, y, z with $(z, y), (y, x) \in E$ and $i \in [3, n]$, if $L(x) = x_i$, then $L(y) = p$ and $L(z) = x'_i$. Labels in $\{x_i, x'_i \mid i \in [n]\}$ refer to

Fig. 2 The quantification (a), the clause evaluation (b), and the assignment part (c) of the working subtree of the initial membrane configuration

the corresponding variables of F , while the label p refers to the word “prison”, because the object a which is initially placed in the p -membranes cannot trigger any rule — so it is imprisoned — until another object sets it free by using a dissolution rule. The membranes of the quantification part are associated with initial multisets of objects as follows. Let x be a node of this part. If x is the x_2 -membrane or it is a p -membrane, then $\omega(x) = a$, and $\omega(x) = \varepsilon$ otherwise.

The clause evaluation part will be used to check if a valuation of the variables satisfies all the clauses. In this part, we have membranes with unique labels in $\{C_1, \dots, C_m\}$. These labels refer to the corresponding clauses of F . The membrane structure of this part can be seen in (Fig. 2b). It is a single path with a root labeled by C_1 and descendants of this root labeled by $C_2, \dots, C_m$, respectively. Initially, these membranes are all empty, that is, for each node x in this part $\omega(x) = \varepsilon$.

The assignment part (Fig. 2c) will be used to select those objects from the input which represent literals that can make a clause true according to the corresponding valuation. This part of the working subtree is a tree with height one. Its root is labeled by 0, and below the root there are $2n$ membranes labelled by $\alpha_1, \dots, \alpha_n$ and $\beta_1, \dots, \beta_n$, respectively. Notice that these membranes are the only elementary membranes of the working subtree. Initially, these elementary membranes are empty, while for the root x of the assignment part, $\omega(x) = \{T\}$.

With this, we have finished the description of the initial membrane configuration of $\Pi(n, m)$. Now, we set the root of the assignment part, that is the node x with $L(x) = 0$, to be the input membrane of $\Pi(n, m)$.

Next, we present the rules of R . We group them so that we can explain how the whole computation of $\Pi(n, m)$ goes. In particular, the correctness of $\Pi(n, m)$ will also be justified. Therefore, we assume that the input membrane contains not only its initial multiset, but also $\text{cod}(F)$, the encoding of the input formula F . We distinguish the following two main stages of the computation, the *generation stage* and the *verification stage*.

First, we give the rules applied in the generation stage. Here, the P system generates all the possible valuations of the variables of F and sends the objects encoding them down in the membrane structure. This is carried out in the working subtree of the initial membrane structure using several groups of rules presented below.

The following rules divide the x_{i+1} -membranes ($i \in [n]$), that is, duplicate those subtrees of the quantification part which have an x_{i+1} -membrane as the root.

$$\begin{aligned} -[a]_{x_{i+1}} &\rightarrow [\hat{t}_i]_{x_{i+1}} [\hat{f}_i]_{x_{i+1}}, & (i \in [n-1]) \\ -[a]_{x_{n+1}} &\rightarrow [t'_n]_{x_{n+1}} [f'_n]_{x_{n+1}}. \end{aligned} \quad (\dagger)$$

After the division, the root of the first new subtree contains either $\hat{t}_i$ with $i \in [n-1]$ or t'_n , while the root of the other one contains either $\hat{f}_i$ with $i \in [n-1]$ or f'_n , representing that x_i is set to *true* and to *false*, respectively. In fact, the duplication of these subtrees corresponds to the recursive calls in lines 4 and 6 in Algorithm 1 where the first and second calls use Φ with x_i set to *true* and to *false*, respectively. The reason why we handle the case $i = n$ differently by introducing t'_n and f'_n instead of $\hat{t}_n$ and $\hat{f}_n$ will be discussed later.

Initially, only the x_2 -membrane can be divided, since the objects that could divide the x_k -membranes with $k \in [3, n+1]$ are imprisoned in the p -membranes. The P system uses the following rules triggered by $\hat{t}_i$ and $\hat{f}_i$, respectively, to dissolve the p -membranes and, in turn, to set objects a free:

$$\begin{aligned} \left. \begin{aligned} -\hat{t}_i[]_{x'_{i+1}} &\rightarrow [\hat{t}_i]_{x'_{i+1}}, \\ -\hat{t}_i[]_{x_{i+2}} &\rightarrow [\hat{t}_i]_{x_{i+2}}, \\ -\hat{t}_i[]_p &\rightarrow [\hat{t}_i]_p, \\ -[\hat{t}_i]_p &\rightarrow t'_i, \end{aligned} \right\} i \in [n-1] \\ \left. \begin{aligned} -\hat{f}_i[]_{x'_{i+1}} &\rightarrow [\hat{f}_i]_{x'_{i+1}}, \\ -\hat{f}_i[]_{x_{i+2}} &\rightarrow [\hat{f}_i]_{x_{i+2}}, \\ -\hat{f}_i[]_p &\rightarrow [\hat{f}_i]_p, \\ -[\hat{f}_i]_p &\rightarrow f'_i, \end{aligned} \right\} i \in [n-1] \end{aligned}$$

Notice that an x_{i+2} -membrane can be divided only after that its ancestor x_{i+1} -membrane is already divided. Indeed, as we have discussed above, to divide an x_{i+2} -membrane, an object $\hat{t}_i$ or $\hat{f}_i$ is necessary and these objects are introduced during the division of an x_{i+1} -membrane.

The evolution of $\hat{t}_i$ to t'_i and $\hat{f}_i$ to f'_i in the dissolution rules above is necessary since we want to send the objects encoding the valuations down to the leaves. Without evolving these objects there would occur undesired nondeterminism: we would have in-communication and dissolution rules triggered by the same objects. Notice moreover that the rule $[a]_{x_{n+1}} \rightarrow [t'_n]_{x_{n+1}} [f'_n]_{x_{n+1}}$ given in rules $(\dagger)$ needs to introduce no objects marked with $\wedge$ since when it is applied there are no membranes left to divide. This is why it introduces t'_n and f'_n directly.

The following rules are used to send the objects that represent the valuations towards the leaves.

Fig. 3 The shape of the working subtree after that all the divisions are done

$$\left. \begin{array}{l} -t'_i[]_{x'_k} \rightarrow [t'_i]_{x'_k}, \quad (k \in [i+2, n]) \\ -t'_i[]_{x_k} \rightarrow [t'_i]_{x_k}, \quad (k \in [i+3, n+1]) \\ -t'_i[]_p \rightarrow [t'_i]_p, \\ \\ -f'_i[]_{x'_k} \rightarrow [f'_i]_{x'_k}, \quad (k \in [i+2, n]) \\ -f'_i[]_{x_k} \rightarrow [f'_i]_{x_k}, \quad (k \in [i+3, n+1]) \\ -f'_i[]_p \rightarrow [f'_i]_p, \end{array} \right\} i \in [n-2]$$

By the application of these rules, objects t'_i and f'_i will reach the innermost p -membranes which are in fact the innermost membranes of the quantification part of the working subtree (see Fig. 2a). The system is set up so that when the x_{i+1} -membranes are divided, the objects t'_j and f'_j with $j < i$ have been already passed through the x_{i+1} -membranes. Therefore, these objects are duplicated with the division of the x_{i+1} -membranes. This means that after that the x_{n+1} -membranes are divided, the working subtree of the P system has 2^n subtrees with x_{n+1} -membranes at the roots (see Fig. 3). Moreover, each of these subtrees contains a set of objects encoding a possible valuation of the variables. Furthermore, all of the possible valuations are represented in these subtrees.

In parallel to the computation carried out in the quantification part, objects in $\text{cod}(F)$ are sent from the input membrane to the α_i - and β_i -membranes, respectively, using the following rules.

$$\left. \begin{array}{l} -v_{ij}[]_{\alpha_i} \rightarrow [v_{ij}]_{\alpha_i}, \\ -v'_{ij}[]_{\beta_i} \rightarrow [v'_{ij}]_{\beta_i}, \end{array} \right\} i \in [n], j \in [m].$$

Recall that object v_{ij} (resp. v'_{ij}) encode that x_i (resp. $\neg x_i$) occurs in C_j . Therefore, there are at most m objects representing that the i th variable occurs in a clause (with or without negation). This means that in at most m steps, all the input objects get into the corresponding α_i - or β_i -membranes. The role of sending these objects into these membranes will be discussed later.

The following rules send the objects representing the valuations through the clause evaluation part of the working subtree. Notice that the actual evaluation of the clauses does not happen in this part of the computation but in a later stage of it.

$$\left. \begin{array}{l} -t'_i[]_{C_j} \rightarrow [t'_i]_{C_j}, \\ -t'_i[]_0 \rightarrow [t'_i]_0, \\ -t'_i[]_{\alpha_i} \rightarrow [t'_i]_{\alpha_i}, \\ \\ -f'_i[]_{C_j} \rightarrow [f'_i]_{C_j}, \\ -f'_i[]_0 \rightarrow [f'_i]_0, \\ -f'_i[]_{\beta_i} \rightarrow [f'_i]_{\beta_i}, \end{array} \right\} i \in [n], j \in [m].$$

When objects t'_i and f'_i reach the input membrane, they go inside the α_i - and β_i -membranes, respectively. Note that reaching the leaves for t'_1 and f'_1 takes more than m steps (hence for any of the t'_i, f'_i objects), thus by the time they do so, all the input objects in $\text{cod}(F)$ are already separated into the α_i - and β_i -membranes, respectively. The next rules are used to dissolve certain α_i - and β_i -membranes.

$$\left. \begin{array}{l} -[t'_i]_{\alpha_i} \rightarrow t_i, \\ -[t_n]_0 \rightarrow t_n, \\ -[f'_i]_{\beta_i} \rightarrow f_i, \\ -[f_n]_0 \rightarrow f_n, \end{array} \right\} i \in [n].$$

After an object t'_i (resp. an f'_i) gets into an α_i -membrane (resp. a β_i -membrane), the P system dissolves such membrane using the above rules. Notice that, for each $i \in [n]$ and $x, y \in V$ with $L(x) = \alpha_i$ and $L(y) = \beta_i$, if x and y are siblings, then either x or y is dissolved but not both. Moreover, since t'_n and f'_n are the last objects that reach the leaves, the last membrane that is dissolved in this part of the computation is either an α_n - or a β_n -membrane. After the dissolution, 0-membranes contain exactly those objects from $\text{cod}(F)$ that correspond to literals that are true under the corresponding evaluation of the variables. Clearly, these literals satisfy the clauses they occur in under this evaluation. During the dissolution of the α_i - and β_i -membranes, objects t'_i and f'_i evolve to t_i and f_i , respectively, which is necessary to avoid undesired nondeterminism in the remaining part of the computation. After dissolving all of the α_i - and β_i -membranes, t_n or f_n dissolves the 0-membrane and all objects contained in this membrane, including an object T initially placed in it, get to the C_1 -membrane.

We now give the rules that are used in the verification stage of the computation. Here, the P system checks first whether a valuation satisfies all the clauses then it combines the information about the different valuations appropriately. To evaluate the clauses, it uses the following rules.

$$\left. \begin{array}{l} -[v_{i,j}]_{C_j} \rightarrow v_{i,j}, \\ -[v'_{i,j}]_{C_j} \rightarrow v'_{i,j}, \end{array} \right\} i \in [n], j \in [m].$$

With these rules, $\Pi(n, m)$ in fact implements lines 9–14 of Algorithm 1 by searching, for each clause C of Φ , a literal that occurs in C and is true under the corresponding valuation. Let $x \in V$ be an x_{n+1} -membrane and $\mathcal{A}$ denote a valuation represented by objects t_i and f_i occurring in the subtree with root x . The sub-computation here starts with an attempt to apply a rule of the form $[v_{i,1}]_{C_1} \rightarrow v_{i,1}$ or $[v'_{i,1}]_{C_1} \rightarrow v'_{i,1}$. If no object can trigger any of these rules, then the computation halts in this subtree. Assume now that $v_{i,1}$ occurs in a C_1 -membrane y . Then t_i occurs in y , too, and thus $\mathcal{A}$ assigns *true* to x_i . Moreover, by the definition of $\text{cod}(F)$, x_i occurs in the clause C_1 of F which yields that $\mathcal{A}$ satisfies C_1 . If, on the other hand, $v'_{i,1}$ occurs in y , then f_i should occur here, too. This means that $\mathcal{A}$ assigns *false* to x_i and thus, since in this case $\neg x_i$ should occur in C_1 , $\mathcal{A}$ satisfies C_1 in this case, too. That is, the C_1 -membrane y is dissolved if and only if $\mathcal{A}$ satisfies C_1 . Generalising this argumentation to the rest of

the C_j -membranes one can see that $\Pi(n, m)$ dissolves all the C_j -membranes ($j \in [m]$) if and only if $\mathcal{A}$ satisfies Φ . Thus, at the end of this part of the verification stage (corresponding to the clause evaluation part, see Fig. 3), the following statement holds.

- (1) An x_{n+1} -membrane either contains no objects, or contains objects t_i and f_i (for each $i \in [n]$, exactly one of t_i or f_i) encoding a satisfying valuation of Φ (with t_i and f_i encoding $x_i = \text{true}$ and $x_i = \text{false}$, respectively) along with an object T .

In the next part of the verification stage, the P system evaluates F using the following rules.

$$\left. \begin{array}{l} -[T]_{x_{n+1}} \rightarrow T, \\ -[t_i]_{x_i} \rightarrow t_i, \\ -[f_i]_{x'_i} \rightarrow f_i, \end{array} \right\} i \in [n] \text{ and } i \text{ is even,}$$

and

$$\left. \begin{array}{l} -[t_i]_{x_i} \rightarrow t_i, \\ -[t_i]_{x'_i} \rightarrow t_i, \\ -[f_i]_{x_i} \rightarrow f_i, \\ -[f_i]_{x'_i} \rightarrow f_i, \end{array} \right\} i \in [n] \text{ and } i \text{ is odd.}$$

We now describe the computation of $\Pi(n, m)$ using these rules and at the same time show that $\Pi(n, m)$ evaluates F correctly. First, all the x_{n+1} -membranes which contain an object T are dissolved. Let $i \in [n]$ and $\tau \in \{t_i, f_i\}$, and let us denote by $v(\tau)$ the truth value represented by τ . To see that $\Pi(n, m)$ evaluates F correctly, we first show the following statement.

- ($\ddagger$) If an x_i -membrane ($i \in [n+1]$) x is dissolved by $\Pi(n, m)$, then

$$Q_i x_i Q_{i+1} x_{i+1} \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-1}), x_i, x_{i+1}, \dots, x_n) = \text{true},$$

where $Q_i, Q_{i+1}, \dots, Q_n$ are the corresponding quantifiers of F and, for every $j \in [i-1]$, τ_j is either t_j or f_j according to the objects occurring in x .

Due to the following observation, τ_j in ($\ddagger$) is well defined.

- (2) For any x_i -membrane ($i \in [n+1]$) x and $j \in [i-1]$, x cannot contain both t_j and f_j during the whole computation of $\Pi(n, m)$.

We now show ($\ddagger$) by induction on i . If $i = n+1$, then ($\ddagger$) follows from Statement (1). Assume now that ($\ddagger$) holds for $1 < i \leq n+1$. We show it for $i-1$. Consider an x_{i-1} -membrane x . We distinguish the following two cases.

1. $i - 1$ is even. By looking at the rules above, one can see that to dissolve x , this membrane has to contain t_{i-1} . This object can occur in x only if the x'_{i-1} -membrane y below x contains both t_{i-1} and f_{i-1} . The only way to introduce these objects in y is to dissolve some of the x_i -membranes below y . Since by Statement (2) x_i -membranes cannot contain both t_{i-1} and f_{i-1} , to introduce these objects in y , both of the x_i -membranes below y have to be dissolved. Then, by the induction hypothesis,

$$Q_i x_i Q_{i+1} x_{i+1} \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-2}), v, x_i, \dots, x_n),$$

is *true*, for every truth value $v \in \{\text{true}, \text{false}\}$. Therefore,

$$\forall x_{i-1} Q_i x_i \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-2}), x_{i-1}, x_i, \dots, x_n),$$

is *true* as well.

2. $i - 1$ is odd. In this case, both t_{i-1} and f_{i-1} can dissolve the x_{i-1} -membranes as well as the x'_{i-1} -membranes. Using this and following the argumentation given in the previous case, we can conclude that in this case at least one of the x_i -membranes below x has to be dissolved. Then, by the induction hypothesis, there is v in $\{\text{true}, \text{false}\}$ such that

$$Q_i x_i \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-2}), v, x_i, \dots, x_n),$$

is *true*. Therefore,

$$\exists x_{i-1} Q_i x_i \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-2}), x_{i-1}, x_i, \dots, x_n),$$

is *true* as well.

This concludes the proof of ($\ddagger$). Reversing the argumentation used in this proof, it can be seen that if an x_i -membrane is not dissolved during the computation of $\Pi(n, m)$, then $Q_i x_i \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-1}), x_i, x_{i+1}, \dots, x_n)$ must be *false*. Consequently,

($\star$) an x_i -membrane ($i \in [n+1]$) x is dissolved by $\Pi(n, m)$ if and only if $Q_i x_i Q_{i+1} x_{i+1} \dots Q_n x_n \Phi(v(\tau_1), \dots, v(\tau_{i-1}), x_i, x_{i+1}, \dots, x_n)$ is *true*.

Using ($\star$), it can be seen that, for every $i \in [n]$, if i is even (resp. odd), then the return value in Line 4 (resp. in Line 6) of Algorithm 1 is *true* if and only if the corresponding x_i -membrane is dissolved. Consequently, F is *true* if and only if the x_1 -membrane is dissolved. Therefore, an object T can reach the halt-membrane if and only if F is *true*.

Using the following, final set of rules, $\Pi(n, m)$ introduces the answer of the system in the skin-membrane.

$$\begin{aligned} -[s]_h &\rightarrow s, \\ -[s]_l &\rightarrow s, \\ &\vdots \\ -[T]_{\text{halt}} &\rightarrow \text{yes}, \\ -[s]_{\text{halt}} &\rightarrow \text{no}. \end{aligned}$$

The first two rules are used in the countdown subtree (Fig. 1c). From the very beginning of the computation, an l -membrane containing an object s gets dissolved in each computational step. Since the countdown subtree contains $7n + 2m + 5$ l -membranes, it takes $7n + 2m + 6$ steps for s to dissolve the h -membrane and all the l -membranes. On the other hand, it can be seen that if F is *true*, then exactly $7n + 2m + 5$ steps are necessary for an object T to reach the halt-membrane.

If F is *true* and thus an object T reaches the halt-membrane, then the system uses $[T]_{\text{halt}} \rightarrow \text{yes}$ to introduce *yes* in the skin-membrane. In this case, s arrives from the countdown subtree into the skin-membrane and the computation halts.

If F is *false* and thus none of the objects T reaches the halt-membrane, then s arrives from the countdown subtree to the halt-membrane. Then s dissolves this membrane introducing *no* in the skin-membrane as the last step of the computation. With this, we have finished the description of $\Pi(n, m)$, and the correctness of the family Π has also been justified. To conclude the proof of Theorem 1, observe that

- the P systems in Π can be constructed by a logarithmic space Turing machine,
- the input encoding *cod* can be constructed by a logarithmic space Turing machine, and
- for each input formula F with n variables and m clauses, $\Pi(n, m)$ halts in linear time $O(n + m)$.

Hence, as logarithmic space Turing machines are guaranteed to halt within a polynomial number of steps, our construction is indeed a polynomially uniform, polynomial-time family of polarizationless recognizer P systems, employing only in-communication, membrane dissolving and weak division rules, solving the PSPACE-complete QSAT problem.

4 Conclusion

In this paper, we have investigated the computation power of polarizationless P systems with active membranes employing only in-communication, dissolution, and weak non-elementary membrane division rules. We have shown that these P systems can solve PSPACE-complete problems in polynomial time. It is well known that P systems with active membranes defined in [26] can solve exactly the problems in PSPACE in polynomial time [33]. Using our result and generalising the proof of the PSPACE upper bound given in [33], we get that the computational power of polynomial-time P systems investigated in this paper characterises PSPACE, too.

The term “weak non-elementary membrane division” suggests that this kind of membrane division is weaker than the classical strong non-elementary division introduced in [26]. According to our knowledge, it is open whether weak non-elementary division is indeed weaker than the strong one. In fact, we suspect that this is not the case and probably the computational power of P systems employing either weak or strong non-elementary membrane division is incomparable. The comparison of these non-elementary membrane division rules in terms of computational power is a topic of our further research plans. For more open problems in the vast landscape of membrane computing, the interested reader is referred to [32].

Acknowledgements This research was supported by Grant NKFIH-1279-2/2020 of the Ministry for Innovation and Technology, Hungary.

Funding Open access funding provided by University of Szeged.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Alhazov, A., Pérez-Jiménez, M.J. (2007). Uniform solution of QSAT using polarizationless active membranes. *International Conference on Machines, Computations and Universality*, 122–133.
- Alhazov, A., Leporati, A., Manzoni, L., Mauri, G., & Zandron, C. (2021). Alternative space definitions for P systems with active membranes. *Journal of Membrane Computing*, 3, 87–96.
- Alhazov, A., Martín-Vide, C., & Pan, L. (2003). Solving a PSPACE-complete problem by P systems with restricted active membranes. *Fundamenta Informaticae*, 58, 67–77.
- Alhazov, A., Pan, L., & Păun, Gh. (2004). Trading polarizations for labels in P systems with active membranes. *Acta Informatica*, 41(2–3), 111–144.
- Buño, K., & Adorna, H. (2020). Distributed computation of a kP system with active membranes for SAT using clause completion. *Journal of Membrane Computing*, 2(2), 108–120.
- Gazdag, Zs. (2014). Solving SAT by P systems with active membranes in linear time in the number of variables. In: A. Alhazov, S. Cojocaru, M. Gheorghe, Y. Rogozhin, G. Rozenberg, A. Salomaa (Eds.), *Membrane Computing: 14th International Conference, LNCS* (vol. 8340, pp. 189–205).
- Gazdag, Zs., Kolonits, G. (2013). A new approach for solving SAT by P systems with active membranes. In: E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, G. Vaszil (Eds.), *Membrane Computing: 13th International Conference, LNCS* (vol. 7762, pp. 195–207).
- Gazdag, Zs., Kolonits, G. (2017). Remarks on the computational power of some restricted variants of P systems with active membranes. In: A. Leporati, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), *Membrane Computing, 17th International Conference, LNCS* (vol. 10105, pp. 209–232).
- Gazdag, Zs., & Kolonits, G. (2019). A new method to simulate restricted variants of polarizationless P systems with active membranes. *Journal of Membrane Computing*, 1(4), 251–261.
- Gensler, H. J. (2002). *Introduction to logic*. Routledge.
- Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.J. (2006). On the power of dissolution in P systems with active membranes. In: R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa (eds.), *Membrane computing: 6th international workshop, LNCS* (vol. 3850, pp. 224–240).
- Krishna, S. N., & Rama, R. (1999). A variant of P systems with active membranes: Solving NP-complete problems. *Romanian Journal of Information Science and Technology*, 2(4), 357–367.
- Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C. (2014). Simulating elementary active membranes, with an application to the P conjecture. In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron (Eds.), *Membrane computing – 15th international conference, CMC15, LNCS* (vol. 8961, pp. 284–299).
- Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C. (2016). Monodirectional P systems. *Natural Computing*, 15, 551–564.
- Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C. (2017). Solving a special case of the P conjecture using dependency graphs with dissolution. In: M. Gheorghe, G. Rozenberg, A. Salomaa, C. Zandron (eds.), *Membrane computing: 18th international conference, LNCS* (vol. 10725, pp. 196–213).
- Leporati, A., Ferretti, C., Mauri, G., Pérez-Jiménez, M. J., & Zandron, C. (2009). Complexity aspects of polarizationless membrane systems. *Natural Computing*, 8(4), 703–717.
- Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2019). Characterizing PSPACE with shallow non-confluent P systems. *Journal of Membrane Computing*, 1, 75–84.
- Murphy, N., Woods, D. (2007). Active membrane systems without charges and using only symmetric elementary division characterise P. In: G. Eleftherakis, P. Kefalas, Gh. Păun, G. Rozenberg, A. Salomaa (eds.), *Membrane computing: 8th international workshop, LNCS* (vol. 4860, pp. 367–384).
- Murphy, N., Woods, D. (2009). On acceptance conditions for membrane systems: Characterisations of **L** and **NL**. In T. Neary, D. Woods, T. Seda, N. Murphy (eds.), *Proceedings international workshop on the complexity of simple programs*, Cork, Ireland, Volume 1 of Electronic Proceedings in Theoretical Computer Science., Open Publishing Association (pp. 172–184).
- Murphy, N., & Woods, D. (2011). The computational power of membrane systems under tight uniformity conditions. *Natural Computing*, 10(1), 613–632.
- Murphy, N., & Woods, D. (2014). Uniformity is weaker than semi-uniformity for some membrane systems. *Fundamental Information*, 134(1–2), 129–152.

22. Orellana-Martín, D., & Riscos-Núñez, A. (2020). Seeking computational efficiency boundaries: The Păun's conjecture. *Journal of Membrane Computing*, 2, 323–331.
23. Pan, L., & Alhazov, A. (2006). Solving HPP and SAT by P systems with active membranes and separation rules. *Acta Informatica*, 43(2), 131–145.
24. Pan, L., Alhazov, A., & Ishdorj, T.-O. (2004). Further remarks on P systems with active membranes, separation, merging, and release rules. *Soft Computing*, 9(9), 686–690.
25. Păun, Gh. (2005). Further twenty six open problems in membrane computing. In: *Third Brainstorming Week on Membrane Computing*, (pp. 249–262). Fénix Editora, Sevilla
26. Păun, Gh. (2001). P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1), 75–90.
27. Păun, Gh., Rozenberg, G., & Salomaa, A. (Eds.). (2010). *The Oxford Handbook of Membrane Computing*. Oxford University Press.
28. Pérez-Jiménez, M. J., Romero-Jiménez, Á., & Sancho-Caparrini, F. (2003). Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2(3), 265–285.
29. Pérez-Jiménez, M. J., Romero-Jiménez, Á., & Sancho-Caparrini, F. (2006). A polynomial complexity class in P systems using membrane division. *Journal of Automata, Languages and Combinatorics*, 11(4), 423–434.
30. Salomaa, A. (1973). *Formal languages*. Academic Press.
31. Sosík, P. (2003). The computational power of cell division in P systems. *Natural Computing*, 2(3), 287–298.
32. Sosík, P. (2019). P systems attacking hard problems beyond NP: A survey. *Journal of Membrane Computing*, 1, 198–208.
33. Sosík, P., & Rodríguez-Patón, A. (2007). Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73(1), 137–152.
34. Woods, D., Murphy, N., Pérez-Jiménez, M.J., Riscos-Núñez, A. (2009). Membrane dissolution and division in P. In: C.S. Calude, J.F.G. da Costa, N. Dershowitz, E. Freire, G. Rozenberg (Eds.), *Unconventional computation: 8th international conference, LNCS* (vol. 5715, pp. 262–276)
35. Zandron, C., Ferretti, C., Mauri, G. (2001). Solving NP-complete problems using P systems with active membranes. In: *Unconventional models of computation, UMC'2K: Proceedings of the second international conference on unconventional models of computation* (pp. 289–301). Springer London, London
36. Zandron, C. (2020). Bounding the space in P systems with active membranes. *Journal of Membrane Computing*, 2, 137–145.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Zsolt Gazdag is an associate professor at the Institute of Informatics, University of Szeged, Hungary. He received his Ph.D. degree in Mathematics and Computer Science from the University of Szeged in 2007. His research interests include the computational complexity of P systems with active membranes, the expressive power of grammars with controlled derivations, and the theory of weighted tree automata.

Károly Hajagos is a Ph.D. student—with a research topic about the computational power of P systems with active membranes—at the Institute of Informatics, University of Szeged, Hungary. He finished his B.Sc. and M.Sc. studies of Computer Science at the University of Szeged in 2018 and 2020, respectively. His research interests include complexity theory and membrane computing techniques.

Szabolcs Iván is an associate professor at the Institute of Informatics, University of Szeged, Hungary. He received his Ph.D. in Mathematics and Computer Science from University of Szeged in 2009. His main research area is focusing on computability and complexity-related questions regarding various notions of formal languages and occasionally data structures, studying the lexicographic ordering of scattered context-free languages with graph-theoretic tools being his main project in the recent years.