Remarks on the Computational Power of Some Restricted Variants of P Systems with Active Membranes

Zsolt Gazdag, Gábor Kolonits

Department of Algorithms and their Applications
Faculty of Informatics
Eötvös Loránd University, Budapest, Hungary
{gazdagzs,kolomax}@inf.elte.hu

Summary. In this paper we consider three restricted variants of P systems with active membranes: (1) P systems using out communication rules only, (2) P systems using elementary membrane division and dissolution rules only, and (3) polarizationless P systems using dissolution and restricted evolution rules only. We show that every problem in **P** can be solved with uniform families of any of these variants. This, using known results on the upper bound of the computational power of variants (1) and (3) yields new characterizations of the class **P**. In the case of variant (2) we provide a further characterization of **P** by giving a semantic restriction on the computations of P systems of this variant.

1 Introduction

P systems with active membranes were introduced in [19]. These P systems have the possibility of dividing elementary (or even non-elementary) membranes. It was soon discovered that this feature (combined with maximal parallelism) makes this variant a rather powerful computational device, and efficient solutions of problems that are complete in **NP** [10, 19, 24, 30] (or even in **PSPACE** [1, 28]) were given. In order to establish the connection between classical complexity classes and P system families, recognizer P systems were introduced in [23, 25]. Since then recognizer P systems are considered as the natural framework to study the computational power of various classes of P system families. Among the many research lines in Membrane Computing, one is to find efficient solutions of computationally hard problems by various types of recognizer P systems with active membranes (see e.g. [2, 3, 4, 17, 18, 22]).

It is not too surprising that membrane division is necessary in these systems to solve computationally hard problems efficiently [30]. However, in [20] Păun conjectured that polarization is also necessary. More precisely, Păun conjectured that polarizationless P systems working in polynomial time can solve only problems

in **P**. Although this conjecture has not been proven yet, there are some partial results. In [8] it was shown that without dissolution rules these systems can solve exactly the problems in **P**. The conjecture was also confirmed in the following cases: when dissolution rules are allowed, but the P systems can employ only restricted, so-called symmetric, division rules [12], and when the initial membrane structure is a nested sequence of membranes, and the system can employ only dissolution and elementary membrane division rules [29].

It was observed in [13] that the **P** lower bound in the characterization of **P** in [8] comes from the polynomial uniformity of the examined P systems. In fact, according to [11] the used uniformity condition dominates the computational power of uniform families of polarizationless P systems with no dissolution rules. This initiated a sequence of papers where P systems with active membranes under reasonably tight uniformity conditions were examined [15, 16]. Moreover, several solutions of problems in **P** with restricted classes of P systems under tight uniformity conditions were given [5, 9, 14, 15].

In this paper we continue the work in this research line. First we show that uniform families of P systems with active membranes using out communication rules only can solve every problem in **P**. Then we show a similar result when the applicable rules are the elementary membrane division and the dissolution rules. The proofs are given by solving a restricted, but still **P**-complete variant of the well know HornSat problem, the satisfiability problem of Horn formulas.

Finally, we show that uniform families of polarizationless P systems with active membranes using dissolution and restricted evolution rules can simulate polynomial time Turing machines efficiently. The restriction made on the evolution rules is that each rule can introduce at most one object during a computation step. This result is stronger than the one appearing in [6] since there communication and not restricted evolution rules were used too. In [15] a solution of a **P**-complete problem was given using dissolution and restricted evolution rules only, however the presented family of P systems was semi-uniform.

Using the **P** upper bound given in [30], our first and third result give new characterizations of **P** in terms of Membrane Computing techniques. In our second result we use such P systems where the initial membrane structure is a nested sequence of membranes, and during the computation the number of membranes on the deepest level is at most two. It can be seen that the set of those problems that can be solved by those P systems with active membranes which have this semantic restriction during their computations are in **P**. This yields another characterization of the complexity class **P**.

The paper is organized as follows. In the next section the necessary notations and notions are recalled. In Section 3 we give the main result of the paper. Finally, some conclusions are given in the last section.

2 Preliminaries

Here we recall the necessary notions used later. Nevertheless, 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. [7, 21, 26], respectively). $\mathbb{N}$ denotes the set of natural numbers. For $n, m \in \mathbb{N}$, $n < m$, $[n, m]$ denotes the set $\{n, n+1, \ldots, m\}$. If $n = 1$, then $[n, m]$ is denoted by $[m]$.

Propositional formulas and the HornSat *problem.*

A *propositional variable* is a variable whose value can be either *true* or *false*. If it is not confusing, we will often call propositional variables simply *variables*. We fix an infinite set $Var = \{x_1, x_2, x_3, \ldots\}$ of variables (for the better readability of the paper we will often denote some of these variables by $x, y, z, \ldots$). For a number $n \in \mathbb{N}$, Var_n is the set $\{x_1, \ldots, x_n\}$. An *interpretation* of the variables in Var_n is a function $\mathcal{I} : Var_n \to \{true,\ false\}$.

The propositional variables and their *negations* are called *literals*. l is a *positive* (resp. *negative*) literal, if $l = x$ (resp. $l = \neg x$), for some $x \in Var$, where $\neg$ denotes the operation of *negation*. A *clause* $\mathcal{C}$ is a *disjunction* of finitely many pairwise different literals satisfying that there is no $x \in Var$ such that both x and $\neg x$ occur in $\mathcal{C}$. A clause $\mathcal{C}$ is a *positive unit clause* if $\mathcal{C}$ consists of one positive literal. A formula in *conjunctive normal form* (CNF) is a conjunction of finitely many clauses. Let φ be a formula in CNF with variables in Var_n ($n \in \mathbb{N}$). We will sometimes consider φ as a finite set of clauses, where the clauses are finite sets of literals. φ is satisfiable, if there is an interpretation under which φ evaluates *true*. Moreover, φ is a *Horn formula* if every clause in φ contains at most one positive literal.

The HornSat problem sounds as follows: *given a Horn formula φ, decide if φ is satisfiable*. It is known that HornSat is **P**-complete. Let Horn3Sat be that restriction of HornSat where every clause of the input formula can contain at most three literals. Moreover, let Horn3SatNorm be that restriction of Horn3Sat where the input formula is in the following normal form: every clause of the formula is a positive unit clause or it contains exactly two negative literals. For example, $x \wedge (\neg x \vee y) \wedge (\neg y \vee \neg z \vee u)$ is an instance of Horn3Sat, but not of Horn3SatNorm, since $(\neg x \vee y)$ neither is a positive unite clause nor contains exactly two negative literals.

Next we show that Horn3SatNorm is **P**-complete. The proof resembles to that of the **NP**-completeness of the 3Sat problem (the 3Sat problem is the satisfiability problem of those formulas in CNF which can have only clauses with three literals, see e.g. [27]).

Proposition 1. Horn3SatNorm *is* **P***-complete.*

Proof. Since this problem is a restriction of HornSat, it is in **P**. Thus, it is enough to show that HornSat can be reduced using logarithmic space to

HORN3SATNORM. First we show that HORNSAT reduces to HORN3SAT. Let φ be a Horn formula over the variables in Var_n ($n \in \mathbb{N}$). We construct an instance φ' of HORN3SAT such that φ' is satisfiable if and only if φ is satisfiable. Let $\mathcal{C}$ be a clause in φ. If $\mathcal{C}$ has at most three literals, then let $\mathcal{C}$ be a clause of φ'. Otherwise, assume that $\mathcal{C} = x_1 \vee \neg x_2 \vee \cdots \vee \neg x_k$ for some $k \in [4, n]$. It can be easily seen that $\mathcal{C}$ is satisfiable if and only if $(x_1 \vee \neg x_2 \vee \neg y) \wedge (y \vee \neg x_3 \vee \cdots \vee \neg x_k)$ is satisfiable, where y is a new variable, not included in Var_n. In this way we can construct the formula $(x_1 \vee \neg x_2 \vee \neg y_1) \wedge (y_1 \vee \neg x_3 \vee \neg y_2) \wedge \cdots \wedge (y_{k-3} \vee \neg x_{k-1} \vee \neg x_k)$, which is satisfiable (over $Var_n \cup \{y_1, \ldots, y_{k-3}\}$) if and only if $\mathcal{C}$ is satisfiable (over Var_n). To a clause with no positive literal one can give a very similar construction. Then we add these new clauses to φ'. Clearly, φ' is satisfiable if and only if φ is satisfiable, and the mapping $\varphi \mapsto \varphi'$ can be carried out by a deterministic Turing machine using logarithmic space in the size of φ.

Next we show that HORN3SAT reduces to HORN3SATNORM. To this end let φ be an instance of HORN3SAT over the variables in Var_n. We construct an instance φ' of HORN3SATNORM such that φ' is satisfiable if and only if φ is satisfiable. For every clause $\mathcal{C}$ of φ, if $\mathcal{C}$ corresponds to the restrictions made on the instances of HORN3SATNORM, then let $\mathcal{C}$ be a clause of φ'. Otherwise we replace $\mathcal{C}$ with the set $\mathcal{C}'$ of clauses defined as follows:

- if $\mathcal{C} = \neg x$, then let $\mathcal{C}' = \{\neg x \vee \neg y, y\}$,
- if $\mathcal{C} = x_1 \vee \neg x_2$, then let $\mathcal{C}' = \{x_1 \vee \neg x_2 \vee \neg y, y\}$, and
- if $\mathcal{C} = \neg x_1 \vee \neg x_2 \vee \neg x_3$, then let $\mathcal{C}' = \{\neg x_1 \vee \neg x_2 \vee y, \neg y \vee \neg x_3\}$,

where y is always a new variable not used yet during the construction. Clearly the clauses in $\mathcal{C}'$ always have the desired forms, and φ' is satisfiable if and only if φ is satisfiable. Moreover, the described construction can be carried out by a logarithmic space Turing machine. Thus, since logarithmic space reductions are closed under composition, we have that HORNSAT can be efficiently reduced to HORN3SATNORM, which finishes the proof of the statement.

Turing machines.

In this paper we will use that variant of Turing machines which appears, e.g., in [27]. A *(deterministic) Turing machine* is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ where

- Q is the finite set of *states*,
- Σ is the *input alphabet*,
- Γ is the tape alphabet including Σ and a distinguished symbol $\sqcup \notin \Sigma$, called the *blank symbol*,
- $\delta : (Q - \{q_a, q_r\}) \times \Gamma \to Q \times \Gamma \times \{-1, 1\}$ is the *transition function*; the ith component of $\delta(q, X)$ ($i \in [1, 3], q \in Q - \{q_a, q_r\}, X \in \Gamma$) is denoted by $\mathrm{proj}_i(\delta(q, X))$,
- q_0, q_a, and q_r are the *initial*, *accepting*, and *rejecting states*, respectively.

M works on a single infinite tape that is closed on the left-hand side. During the computation of M, the tape contains only finitely many non-blank symbols, and it is blank elsewhere. Let $w \in \Sigma^*$. The initial configuration of M on w is the configuration where w is placed at the beginning of the tape, the head points to the first letter of w, and the current state of M is q_0. A computation step performed by M can be described as follows. If M is in state p and the head of M reads the symbol X, then M changes its state to q and writes X' onto X if and only if $\delta(p, X) = (q, X', d)$, for some $d \in \{-1, 1\}$. Moreover, if $d = 1$ (resp. $d = -1$), then M moves its head one position to the right (resp. to the left) (by definition, M can never move the head off the left-hand end of the tape even if the head points to the first cell and $d = -1$). We say that M accepts (resp. rejects) w, if M can reach from the initial configuration on w the accepting state q_a (resp. the rejecting state q_r). We note here that M can stop only in these states. The language accepted by M is the set $L(M)$ consisting of those words in Σ^* that are accepted by M.

P systems with active membranes.

In this paper we consider several restricted variants of P systems with active membranes. In general, a *P system* with active membranes [19] is a construct of the form $\Pi = (\Gamma, H, \mu, w_1, \ldots, w_m, R)$, where m is the initial *degree* of the system, Γ is the alphabet of *objects*, H is a finite set of *labels* of the membranes; μ is a *membrane structure* consisting of m membranes and labelled with elements of H; $w_1, \ldots, w_m \subseteq \Gamma^*$ are the *initial multisets of objects* placed in the m regions of μ; and R is a finite set of *rules* defined as follows:

(a) $[a \to v]_h^e$, for $e \in \{+, -, 0\}$, $h \in H, a \in \Gamma, v \in \Gamma^*$
(object *evolution* rules, associated with membranes and depending on the label and the charge of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);

(b) $a[\ ]_h^{e_1} \to [b]_h^{e_2}$, for $e_1, e_2 \in \{+, -, 0\}$, $h \in H$, $a, b \in \Gamma$
(*in communication* rules, sending an object into a membrane, maybe modified during this process; also the polarization of the membrane can be modified, but not its label);

(c) $[a]_h^{e_1} \to [\ ]_h^{e_2} b$, for $e_1, e_2 \in \{+, -, 0\}$, $h \in H$, $a, b \in \Gamma$
(*out communication* rules; an object is sent out of the membrane, maybe modified during this process; also the polarization of the membrane can be modified, but not its label);

(d) $[a]_h^e \to b$, for $e \in \{+, -, 0\}$, $h \in H$, $a, b \in \Gamma$
(*membrane dissolving* rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

(e) $[a]_h^{e_1} \to [b]_h^{e_2} [c]_h^{e_3}$, for $e_1, e_2, e_3 \in \{+, -, 0\}$, $h \in H$, $a, b, c \in \Gamma$
(*division* rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with possibly different polarizations; the object a specified in the rule is replaced in the two new membranes by (possibly new) objects b and c respectively, and the remaining objects are duplicated).

As it is usual in membrane computing, P systems with active membranes work in a *maximally parallel* manner:

- In one step, any object of a membrane that can be used by a rule must be used, but one object can be used by only one rule in (a)-(e).
- If an object can be used by two or more different rules, then one of these rules is non-deterministically chosen.
- A membrane can be the subject of only one rule in (b)-(d) during each step.

We say that an evolution rule $[a \to v]_h^e$ is 1-*restricted* if $|v| \leq 1$ (i, the number of objects in v is at most one). A *layer* is a nested membrane structure, that is a layer has the form $[\dots[\ ]_{h_1}\dots]_{h_n}$ $(n \geq 1, h_1, \dots, h_n \in H)$. For two layers $\mu_1 = [\dots[\ ]_{h_1}\dots]_{h_j}$ and $\mu_2 = [\dots[\ ]_{g_1}\dots]_{g_k}$ $(j, k \geq 1, h_1, \dots, h_j, g_1, \dots, g_k \in H)$, the *composition* $\mu_1[\mu_2]$ of μ_1 and μ_2 is the layer $[\dots[[\dots[\ ]_{g_1}\dots]_{g_k}]_{h_1}\dots]_{h_j}$. A *region* is a composition of finitely many layers.

Recognizer P systems.

A *recognizer P system* [23, 25] is a P system Π with a designated *input* membrane and having the following properties. The alphabet Γ of objects has two designated elements *yes* and *no*. Every computation of Π halts and sends to the environment the same object which is either *yes* or *no*, and these objects are sent out in the last step of the computation (if the examined P system model does not have out communication rules, then the output of the systems appears in the skin membrane). The *input* of Π is a multiset over Γ, which is added to the input membrane of the system in the initial configuration.

Uniform families of P systems.

A family $\mathbf{\Pi} = \{\Pi(i)\}_{i\in\mathbb{N}}$ of recognizer P systems *decides a problem* L if, for every instance x of L with length n, starting $\Pi(n)$ with an appropriate encoding of x in its input membrane, $\Pi(n)$ sends to the environment *yes* if and only if $x \in L$.

We will use uniform families of recognizer P systems to solve problems in $\mathbf{P}$. Clearly, we should use such a uniformity condition that is reasonably weak to work with in class $\mathbf{P}$. According to the widely believed fact that Turing machines using logarithmic space are strictly weaker than Turing machines working in polynomial time, we will use logarithmic space uniform families of P systems. We denote by $\mathbf{L}$ the family of functions that can be computed by Turing machines using logarithmic amount of space.

Assume that a family $\mathbf{\Pi} = \{\Pi(i)\}_{i\in\mathbb{N}}$ of recognizer P systems decides a problem L. $\mathbf{\Pi}$ is called **(L,L)**-*uniform* if and only if (i) there are functions $f, cod \in \mathbf{L}$ such that, for every $n \in \mathbb{N}$, $\Pi(n) = f(1^n)$ (i.e., the P system $\Pi(n)$ can be constructed by a logarithmic space Turing machine from the unary representation of n); (ii) for every instance x of L with size n, $cod(x)$ is a multiset encoding x over the alphabet of objects in $\Pi(n)$.

For a type $\mathcal{F}$ of recognizer P systems, we denote by $(\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{F}}$ the class of those problems that can be decided by $(\mathbf{L}, \mathbf{L})$-uniform families of P systems

of type $\mathcal{F}$ working in polynomial time. $\mathcal{AM}_{+out}$ (resp. $\mathcal{AM}_{+e,+d}$) denotes the family of P systems with active membranes having out communication (resp. division and dissolution) rules only. Similarly, $\mathcal{AM}^0_{+evo(1),+d}$ denotes the family of polarizationless P systems having 1-restricted evolution and dissolution rules only.

3 Results

Here we show that recognizer P systems of type $\mathcal{AM}_{+out}$, $\mathcal{AM}_{+e,+d}$, or $\mathcal{AM}^0_{+evo(1),+d}$ and working in polynomial time are capable to solve every problem in **P**. First we consider two solutions of Horn3SatNorm, then we give an efficient simulation of Turing machines.

3.1 The solution of Horn3SatNorm

By definition, if φ is an instance of Horn3SatNorm, then every clause of φ is either a positive unit clause or it has exactly two negative literals. In the rest of this section by a clause we mean a clause having this property. Using the well known equivalences of propositional logic, a clause having exactly two negative literals $\neg x$ and $\neg y$ can be written in the form $x \wedge y \rightarrow\downarrow$ or $x \wedge y \rightarrow z$, where z is a variable, $\rightarrow$ denotes the operation of implication and $\downarrow$ denotes a formula with constant *false* truth value. We will often use these expressions to denote the corresponding clauses of the input formula (in fact, we will often call these expressions clauses, although strictly speaking they are not clauses). Moreover, for the sake of simplicity, we will not indicate the sign $\wedge$ of conjunction in the left-hand side of these expressions.

Let φ be an instance of Horn3SatNorm. Clearly, if φ is *true* in an interpretation I, then $I(x) = true$ must hold for every positive unite clause $\{x\}$ in φ. Assume now that $\mathcal{C} = xy \rightarrow z$ is a clause of φ, where x, y are variables and z is either a variable or $\downarrow$. We observe that if $I(x) = I(y) = true$, then $\mathcal{C}$ is *true* in I if and only if z is *true* too. That is, if $z =\downarrow$, then x, y, z cannot be all *true* in I. We will use these observations in the following algorithm H3SN, which decides if an instance φ of Horn3SatNorm over variables in Var_n $(n \geq 1)$ is satisfiable or not. Let $\mathcal{N}(n)$ denote the set of those clauses over variables in Var_n which contain exactly two negative literals, and let $m = |\mathcal{N}(n)|$. In the rest of this section we assume a fixed enumeration $c_1, \ldots, c_m$ of clauses in $\mathcal{N}(n)$.

Algorithm H3SN

1. **input**: φ
2. $X := \{x \in Var_n \mid x \in \varphi\}$ // x is a positive unit clause in φ
3. **For** $i = 1 \ldots n$ do
4. **For** $j = 1 \ldots m$ **do**
5. **If** $c_j = xy \rightarrow u \in \varphi$ **and** $x, y \in X$ **then** $X := X \cup \{u\}$
6. **If** $\downarrow$ is in X **then return** *no*
7. **else return** *yes*

To demonstrate the work of H3SN consider the following example. Let $\varphi = x \wedge y \wedge (xy \to z) \wedge (xz \to\downarrow)$. Then, initially, $X = \{x, y\}$. Since $x, y \in X$ and $xy \to z \in \varphi$, X becomes $\{x, y, z\}$. Then, since $x, z \in X$ and $xz \to\downarrow\in \varphi$, X becomes $\{x, y, z, \downarrow\}$. After this the value of X remains the same until H3SN halts. Thus, since $\downarrow\in X$, H3SN outputs *no*. This is correct as φ is unsatisfiable.

In this section we give two families of P systems with rather restricted sets of applicable rules to solve the Horn3SatNorm problem in polynomial time. Both solutions are based on Algorithm $HS3N$. In these solutions the P systems cannot employ evolution and in communication rules. In addition, in the first solution dissolution and membrane division rules, while in the second solution out communication rules are also not allowed.

In both solutions the P systems, roughly, work as follows. Let φ be an instance of Horn3SatNorm over the variables in Var_n. The initial membrane structure consists of n regions, and the innermost membrane contains $cod(\varphi)$ (that is, the encoding of φ). A region r_i corresponds to the ith round of the main loop in Algorithm H3SN.

For an arbitrary clause $\mathcal{C}$ with variables in Var_n, $cod(\varphi)$ contains an object $O^{\exists}_{\mathcal{C}}$ or $O^{\nexists}_{\mathcal{C}}$ (but not both) according to that $\mathcal{C}$ occurs in φ or not. Moreover, for every clause of the form $xy \to u$ ($x, y \in Var_n, u \in Var_n \cup \{\downarrow\}$), r_i has a layer l whose membranes are indexed by this clause. The objects in the inner membrane of l go through l (either by out communication or by dissolution rules, according to the used model), and during this the system performs the following task. It first checks whether all the objects $O^{\exists}_{xy\to u}$, $O^{\exists}_{x}$, $O^{\exists}_{y}$, and $O^{\nexists}_{u}$ were present in the innermost membrane of l. If yes, then the system rewrites $O^{\nexists}_{u}$ to $O^{\exists}_{u}$. In this way the system can determine which variables of φ must be *true* in order to make φ *true* in an interpretation. After performing the above task in all layers of region r_n, the skin contains either $O^{\exists}_{\downarrow}$ or $O^{\nexists}_{\downarrow}$. If $O^{\exists}_{\downarrow}$ occurs in the skin, then φ cannot be satisfied and the system introduces object *no*, otherwise it introduces *yes*.

Formally, we encode an instance φ of Horn3SatNorm with variables in Var_n as follows. First, let

$$\Sigma(n) = \{O^e \mid O \in V(n) \cup C(n), e \in \{\exists, \nexists\}\},$$

where $V(n) = \{V_u \mid u \in Var_n \cup \{\downarrow\}\}$ and $C(n) = \{C_{xy\to u} \mid x, y \in Var_n, u \in Var_n \cup \{\downarrow\}\}$. Then the encoding of φ is $cod(\varphi) = \{O^{\exists}_c \in \Sigma(n) \mid c \in \varphi\} \cup \{O^{\nexists}_c \in \Sigma(n) \mid c \notin \varphi\} \cup \{V^{\nexists}_{\downarrow}\}$. We note here that technically there is no need to distinguish in the notation between positive unite clauses and clauses having two negative literals. Nevertheless, we decided to do so to improve the readability of the constructions. Since the size of φ is clearly polynomial in n, it can be seen that *cod* is a function in **L**.

A solution using out communication rules only.

Here we solve Horn3SatNorm with a family $\mathbf{\Pi} = \{\Pi(n)\}_{n\in\mathbb{N}}$ of recognizer P systems of type $\mathcal{AM}_{+out}$, where $\Pi(n) = (\Gamma(n), H(n), \mu(n), W(n), R(n))$ is defined as follows:

- $\Gamma(n) = \Sigma(n) \cup \{V_x^{\exists +} \mid x \in Var_n \cup \{\downarrow\}\} \cup \{yes, no\}$.
- $H = \{(xy \to u, \alpha) \mid x, y \in Var_n, u \in Var_n \cup \{\downarrow\}, \alpha \in \{a, b, c\}\} \cup \{s_k \mid k \in [m+n]\} \cup \{skin\}$.
- $\mu(n) = S[r_n[r_{n-1}[\ldots r_2[r_1]\ldots]]]$, where $S = [[[\ ]_{s_1} \ldots]_{s_{m+n}}]_{skin}$ and, for every $i \in [n]$, r_i is a region defined as follows. $r_i = l_{c_m}[\ldots l_{c_2}[l_{c_1}]\ldots]$, where, for every $j \in [m]$, the layer l_{c_j} has the form $[[[\ ]_{(c_j,a)}]_{(c_j,b)}]_{(c_j,c)}$.
- The input membrane is the innermost membrane in the initial membrane structure.
- $W(n)$ is the sequence of empty initial multisets.
- R consists of the following subsets of rules, where $x, y \in Var_n$ and $u \in Var_n \cup \{\downarrow\}$:

  (1) $[C^{\exists}_{xy\to u}]^0_{(xy\to u,a)} \to [\ ]^+_{(xy\to u,a)} C^{\exists}_{xy\to u}$,
  $[C^{\exists}_{xy\to u}]^0_{(xy\to u,\beta)} \to [\ ]^0_{(xy\to u,\beta)} C^{\exists}_{xy\to u}$,
  $[C^{\nexists}_{xy\to u}]^0_{(xy\to u,\alpha)} \to [\ ]^-_{(xy\to u,\alpha)} C^{\nexists}_{xy\to u}$ $\quad (\alpha \in \{a, b, c\}, \beta \in \{b, c\})$.
  These rules are used to initialize the layers in the following sense: the first membranes of those layers that are indexed by a clause in φ get positive charges, the second and third membranes keep their neutral charges, while all the membranes of the remaining layers get negative charges.

  (2) $[V^e_v]^-_{(xy\to u,\alpha)} \to [\ ]^-_{(xy\to u,\alpha)} V^e_v$,
  $[C^e_{rs\to v}]^-_{(xy\to u,\alpha)} \to [\ ]^-_{(xy\to u,\alpha)} C^e_{rs\to v}$
  $(e \in \{\exists, \nexists\}, r, s \in Var_n, v \in Var_n \cup \{\downarrow\}, \alpha \in \{a, b, c\})$.
  Every membrane with negative charge lets all of the objects to pass through itself.

  (3) $[V^{\exists}_x]^+_{(xy\to u,a)} \to [\ ]^-_{(xy\to u,a)} V^{\exists +}_x$,
  $[V^{\exists +}_x]^0_{(xy\to u,b)} \to [\ ]^+_{(xy\to u,b)} V^{\exists}_x$.
  If φ has a clause $xy \to u$, that is, the membrane with label $(xy \to u, a)$ has positive charge, and $V^{\exists}_x$ exists in this membrane, then these rules are used to store this information in the positive charge of the membrane with label $(xy \to u, b)$.

  (4) $[V^{\exists}_y]^+_{(xy\to u,b)} \to [\ ]^-_{(xy\to u,b)} V^{\exists +}_y$,
  $[V^{\exists +}_y]^0_{(xy\to u,c)} \to [\ ]^+_{(xy\to u,c)} V^{\exists}_y$.
  If the membrane with label $(xy \to u, b)$ has positive charge and $V^{\exists}_y$ exists in this membrane, then these rules are used to store this information in the positive charge of the membrane with label $(xy \to u, c)$.

  (5) $[V^{\nexists}_u]^+_{(xy\to u,c)} \to [\ ]^-_{(xy\to u,c)} V^{\exists}_u$.
  The positive charge of the membrane with label $(xy \to u, c)$ indicates that $xy \to u$ is a clause of the system and that both variables x and y has to be *true* in an interpretation in order to make φ *true*. Thus, with this rule the system rewrites $V^{\nexists}_u$ to $V^{\exists}_u$ indicating that u must be also *true* to make φ *true*.

(6) $[V_u^{\exists}]^p_{(xy\to u,\alpha)} \to [\ ]^-_{(xy\to u,\alpha)} V_u^{\exists}$ $(p \in \{+,0\}, \alpha \in \{a,b,c\})$.
If the system already knows that u must be *true* to make φ *true*, then the charges of the corresponding membranes are set to negative.

(7) $[V_x^{\nexists}]^p_{(xy\to u,\alpha)} \to [\ ]^-_{(xy\to u,\alpha)} V_x^{\nexists}$,
$[V_y^{\nexists}]^p_{(xy\to u,\alpha)} \to [\ ]^-_{(xy\to u,\alpha)} V_y^{\nexists}$ $(p \in \{+,0\}, \alpha \in \{a,b,c\})$.
If any of the variables on the left-hand side of a clause $xy \to u$ is not considered to be *true* yet, then the charges of membranes of the corresponding layer are set to negative by these rules, and $V_u^{\exists}$ cannot be introduced by this layer.

(8) $[V_{\downarrow}^{e}]^0_{s_k} \to [\ ]^0_{s_k} V_{\downarrow}^{e}$, $[V_{\downarrow}^{\exists}]^0_{skin} \to [\ ]^0_{skin} no$, $[V_{\downarrow}^{\nexists}]^0_{skin} \to [\ ]^0_{skin} yes$
$(k \in [m+n], e \in \{\exists, \nexists\})$.
The first rule is used to move object $V_{\downarrow}^{\exists}$ or $V_{\downarrow}^{\nexists}$ towards the skin membrane. When they arrive at the skin, the system sends to the environment the correct answer.

Correctness, running time, and $(\mathbf{L}, \mathbf{L})$*-uniformity.*

First we observe that during the computation of $\Pi(n)$ the following holds.

1. If all the membranes in a layer l have negative charge, then l does not contribute to the computation, i.e. all objects pass through the membranes of l without any change.
2. For every $\mathcal{C} \in C(n)$, either $C_{\mathcal{C}}^{\exists}$ or $C_{\mathcal{C}}^{\nexists}$ (but not both) occurs in the system (the same object during the whole computation).
3. For every $x \in Var_n \cup \{\downarrow\}$, either $V_x^{\exists}$ or $V_x^{\nexists}$ (but not both) occurs in the system. Indeed, the rules that can change an object of this form are rules in (3)-(5) (not counting the rules that introduce *yes* or *no* at the last step of the computation). Rule in (5) removes $V_u^{\nexists}$ and introduces $V_u^{\exists}$, thus the observation remains true after applying it. Concerning rules in (3)-(4), it is enough to observe that if the first rule can be applied, then the second rule can be applied too in the next step.

Now consider a layer $l_{xy\to u}$ $(x, y \in Var_n, u \in Var_n \cup \{\downarrow\})$. At the beginning of the computation every membrane in $l_{xy\to u}$ has neutral charge. According to the objects that pass through this layer we can distinguish the following cases.

1. All of the objects $C_{xy\to u}^{\exists}$, $V_x^{\exists}$, $V_y^{\exists}$, and $V_u^{\nexists}$ pass through the membranes of $l_{xy\to u}$. Then the system rewrites the object $V_u^{\nexists}$ to $V_u^{\exists}$.
2. Any of the objects $C_{xy\to u}^{\nexists}$, $V_x^{\nexists}$, $V_y^{\nexists}$, or $V_u^{\exists}$ passes through the membranes of $l_{xy\to u}$. Then the charge of every membrane in $l_{xy\to u}$ is set to negative, and thus this layer cannot contribute to the computation. (Notice that in this case the computation is not deterministic but confluent, i.e., all the possible computations in the layer yield the same result.)

It follows that the objects passing trough the layer $l_{xy\to u}$ simulate step 5 of Algorithm H3SN. Thus, sending objects through a region corresponds to performing

steps $4-5$ of this algorithm. Since the algorithm performs steps $4-5$ n times, the work of the P system in the n regions corresponds to the work of the algorithm. Thus, $V_{\downarrow}^{\exists}$ or $V_{\downarrow}^{\nexists}$ eventually appears in membrane s_1. In the next $m+n$ steps this object gets to the skin by rules in (8). There the system computes *yes* or *no* accordingly, which is then sent to the environment. It can bee seen that during this computation all the other objects occurring in the systems arrive to membrane s_1, and the computation halts.

This justifies the correctness of $\Pi(n)$. Since $\Pi(n)$ has polynomial number of objects in the initial configuration and no evolution rules are performed during its work, sending all the objects through a region takes polynomial steps. Thus the running time of $\Pi(n)$ is also polynomial.

It can be seen that all the ingredients of $\Pi(n)$ can be enumerated and written onto the output tape by a logarithmic space Turing machine. Thus, using that HORN3SATNORM is **P**-complete, we get the following result.

Theorem 1. $\mathbf{P} \subseteq (\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{AM}_{+out}}$.

A solution using elementary membrane division and dissolution rules only.

In this subsection we solve HORN3SATNORM with a family $\mathbf{\Pi} = \{\Pi(n)\}_{n\in\mathbb{N}}$ of recognizer P systems of type $\mathcal{AM}_{+e,+d}$. The solution is similar to the one given in the previous subsection, however, there is a substantial difference: here the presence of the necessary objects to simulate step 5 of Algorithm H3SN are checked by the application of membrane division rules. Consequently, those objects that do not take part in the simulation are duplicated several times. In particular, at certain points of the computation the P system has multiple copies of objects of the form $V_x^{\nexists}$. However, the correctness of the computation requires that at the beginning of the work in a layer there is at most one copy of objects of this form. Therefore we will apply special layers, that will remove those objects that could cause the system to give incorrect results. The following is the formal definition of $\Pi(n) = (\Gamma(n), H(n), \mu(n), W(n), R(n))$:

- $\Gamma(n) = \Sigma(n) \cup \{w, \overline{w}_1, \overline{w}_2, \#, \$\} \cup \{yes, no\}$.
- $H(n) = \{skin, s\} \cup \{(xy \to u, \alpha) \mid x, y \in Var_n, u \in Var_n \cup \{\downarrow\}, \alpha \in \{a, b, c, d\}\} \cup \{d_O \mid O \in V(n) \cup C(n) \cup \{w\}\}$.
- $\mu(n)$ is defined as follows. Let $\mathcal{C} = xy \to u$ be a clause ($x, y \in Var_n, u \in Var_n \cup \{\downarrow\}$) and $l_{\mathcal{C}}$ be the layer $D_{\mathcal{C}}[M_{\mathcal{C}}]$, where $D_{\mathcal{C}}$ and $M_{\mathcal{C}}$ are defined as follows:

  $$M_{\mathcal{C}} = [\,[\,[\,[\,[\,[\,[\,]_{(xy\to u,a)}]_{(xy\to u,b)}]_{d_w}]_{(xy\to u,c)}]_{d_w}]_{(xy\to u,d)}$$

  and $D_{\mathcal{C}}$ is a layer containing, for every $O \in V(n) \cup C(n)$, the membrane $[\,]_{d_O}$ fifteen times if $O \neq V_u$, and once, otherwise. Intuitively, $M_{\mathcal{C}}$ is that part of the layer which is responsible to simulate step 5 in Algorithm H3SN, and layer $D_{\mathcal{C}}$ is used (together with membranes with label d_w in $M_{\mathcal{C}}$) to remove those

objects that are produced by the used division rules, but should be removed in order to keep the behaviour of the system correct.
To finish the construction, let $\mu(n) = S[r_n[r_{n-1}[\dots r_2[r_1]\dots]]$, where $S = [[\ ]_s]_{skin}$ and, for every $i \in [n]$, r_i is the region $l_{c_m}[\dots l_{c_2}[l_{c_1}]\dots]$.

- The input membrane is the innermost membrane in the initial membrane structure.
- $W(n)$ is a sequence of empty initial multisets.
- R consists of the following subsets of rules, where $x, y \in Var_n$ and $u \in Var_n \cup \{\downarrow\}$:

(1) $[V_u^{\nexists}]^0_{(xy\to u,a)} \to [w]^-_{(xy\to u,a)}[\#]^-_{(xy\to u,a)}$,
$[V_u^{\exists}]^0_{(xy\to u,a)} \to [\overline{w}_1]^-_{(xy\to u,a)}[\#]^-_{(xy\to u,a)}$.
These rules are used to decide if $V_u^{\nexists}$ or $V_u^{\exists}$ is present in a membrane with label $(xy \to u, a)$. If $V_u^{\nexists}$ is present, then the system introduces w which indicates that the system should work further to decide if $V_u^{\exists}$ should be introduced or not. Object $\overline{w}_1$ indicates that $V_u^{\exists}$ is present in the system and thus it should not be introduced later. # indicates that the membrane containing it is not used effectively in the computation.

(2) $[w]^-_{(xy\to u,a)} \to w, [\overline{w}_1]^-_{(xy\to u,a)} \to \overline{w}_1$,
$[\#]^-_{(xy\to u,a)} \to \$$.
These rules pass the information computed by rules in (1) to the membrane labelled with $(xy \to u, b)$. \$ is a dummy object not used later.

(3) $[C^{\exists}_{xy\to u}]^0_{(xy\to u,b)} \to [C^{\exists}_{xy\to u}]^+_{(xy\to u,b)}[C^{\exists}_{xy\to u}]^+_{(xy\to u,b)}$,
$[C^{\nexists}_{xy\to u}]^0_{(xy\to u,b)} \to [C^{\nexists}_{xy\to u}]^-_{(xy\to u,b)}[C^{\nexists}_{xy\to u}]^-_{(xy\to u,b)}$.
These rules decide if object $C^{\exists}_{xy\to u}$ or $C^{\nexists}_{xy\to u}$ exists in the system. The result is stored in the polarizations of the new membranes.

(4) $[w]^+_{(xy\to u,b)} \to w, [w]^-_{(xy\to u,b)} \to \overline{w}_2$,
$[\overline{w}_1]^+_{(xy\to u,b)} \to \overline{w}_1, [\overline{w}_1]^-_{(xy\to u,b)} \to \overline{w}_1$.
These rules introduce objects that will control the computation according to the information computed by the previous subsets of rules. For example, if w and $C^{\nexists}_{xy\to u}$ is present in the inner membrane, then $\overline{w}_2$ is introduced. In this case $V_u^{\exists}$ will not be introduced at the end of the computation in this layer (see rules in (8)).

(5) $[V_y^{\exists}]^0_{(xy\to u,c)} \to [V_y^{\exists}]^+_{(xy\to u,c)}[V_y^{\exists}]^+_{(xy\to u,c)}$,
$[V_y^{\nexists}]^0_{(xy\to u,c)} \to [V_y^{\nexists}]^-_{(xy\to u,c)}[V_y^{\nexists}]^-_{(xy\to u,c)}$.
These rules decide if object $V_y^{\exists}$ or $V_y^{\nexists}$ exists in the system. The result is stored in the polarizations of the new membranes.

(6) $[w]^+_{(xy\to u,c)} \to w, [w]^-_{(xy\to u,c)} \to \overline{w}_2$,
$[\overline{w}_1]^+_{(xy\to u,c)} \to \overline{w}_1, [\overline{w}_1]^-_{(xy\to u,c)} \to \overline{w}_1$,
$[\overline{w}_2]^+_{(xy\to u,c)} \to \overline{w}_2, [\overline{w}_2]^-_{(xy\to u,c)} \to \overline{w}_2$.

These rules introduce objects that will control the computation according to the information computed by the previous subset of rules.

(7) $[V_x^{\exists}]^0_{(xy\to u,d)} \to [V_x^{\exists}]^+_{(xy\to u,d)}[V_x^{\exists}]^+_{(xy\to u,d)}$,
$[V_x^{\not\exists}]^0_{(xy\to u,d)} \to [V_x^{\not\exists}]^-_{(xy\to u,d)}[V_x^{\not\exists}]^-_{(xy\to u,d)}$.
These rules decide if object $V_x^{\exists}$ or $V_x^{\not\exists}$ exists in the system. The result is stored in the polarizations of the new membranes.

(8) $[w]^+_{(xy\to u,d)} \to V_u^{\exists}, [w]^-_{(xy\to u,d)} \to V_u^{\not\exists}$,
$[\overline{w}_1]^+_{(xy\to u,d)} \to V_u^{\exists}, [\overline{w}_1]^-_{(xy\to u,d)} \to V_u^{\exists}$
$[\overline{w}_2]^+_{(xy\to u,d)} \to V_u^{\not\exists}, [\overline{w}_2]^-_{(xy\to u,d)} \to V_u^{\not\exists}$.
These rules are used to handle the different cases of possible computations in a layer. For example, w indicates that at the beginning of the computation in a layer the system contained objects $V_u^{\not\exists}$, $C^{\exists}_{xy\to u}$, and $V_y^{\exists}$.

(9) $[O^e]^0_{d_O} \to \$, [w]^0_{d_w} \to \$, [\overline{w}_i]^0_{d_w} \to \$$
$(O \in V(n) \cup C(n), e \in \{\exists, \not\exists\}, i \in [2])$.
These rules are used to remove certain objects from the system.

(10) $[V_{\downarrow}^{\not\exists}]^0_s \to [no]^-_s[\$]^-_s$, $[V_{\downarrow}^{\exists}]^0_s \to [yes]^-_s[\$]^-_s$, $[\kappa]^-_s \to \kappa \quad (\kappa \in \{yes, no\})$.
These rules are used to send out the computed answer to the environment.

Correctness, running time, and $(\mathbf{L}, \mathbf{L})$*-uniformity.*

First we observe that during the computation of $\Pi(n)$ the following holds:

1. The membrane structure has the form $[\ldots[M]_{h_1}\ldots]_{h_k}$ $(h_1, \ldots, h_k \in H(n))$, where M is either a membrane or it is of the form $[\ ]_{g_1}[\ ]_{g_2}$ $(g_1, g_2 \in H(n))$, and
2. objects occur only in the innermost membranes.

The correctness of the system follows from the following lemma.

Lemma 1. *Let* $\mathcal{C} = xy \to u$ $(x, y \in Var_n, u \in Var_n \cup \{\downarrow\})$ *and consider the layer* $l_{\mathcal{C}} = D_{\mathcal{C}}[M_{\mathcal{C}}]$. *Assume that, for every* $O \in C(n) \cup V(n)$, *either one copy of* $O^{\exists}$ *or one copy of* $O^{\not\exists}$ *occurs in* $l_{\mathcal{C}}$. *Let* O *be an object in* $l_{\mathcal{C}}$. *Depending on* O *the following holds:*

1. *If* $O \in \Sigma(n) - \{V_u^{\not\exists}\}$, *then after dissolving all the membranes in* $l_{\mathcal{C}}$, $\Pi(n)$ *contains exactly one copy of* O.
2. *If* $O = V_u^{\not\exists}$ *and* $l_{\mathcal{C}}$ *contains all of the objects* $C_{\mathcal{C}}^{\exists}$, $V_x^{\exists}$, *and* $V_y^{\exists}$, *then after dissolving all the membranes in* $l_{\mathcal{C}}$, $\Pi(n)$ *contains no* $V_u^{\not\exists}$ *and exactly one copy of* $V_u^{\exists}$.
3. *If* $O = V_u^{\not\exists}$ *and* l_c *contains* $C_{\mathcal{C}}^{\not\exists}$, $V_x^{\not\exists}$, *or* $V_y^{\not\exists}$, *then after the work in* $l_{\mathcal{C}}$ $\Pi(n)$ *contains exactly one copy of* $V_u^{\not\exists}$.

Proof. By assumption, $l_{\mathcal{C}}$ contains exactly one copy of O. Then Statement 1 can be seen by distinguishing the following two sub-cases:

Case 1. $O \neq V_u^{\exists}$. Then during the work in $M_{\mathcal{C}}$, O is duplicated by the corresponding rules rules in (1), (3), (5), and (7), and the other rules are not applied to O in $M_{\mathcal{C}}$. This yields sixteen copies of O in $D_{\mathcal{C}}$. Out of these copies fifteen ones are removed during the computation in $D_{\mathcal{C}}$.
Case 2. $O = V_u^{\exists}$. Then the second rule in (1) removes first $V_u^{\exists}$ and introduces one copy of $\overline{w}_1$. After this, membrane $(\mathcal{C}, a)$ is dissolved using rules in (2). In the next two steps, $\overline{w}_1$ is duplicated first due the division of membrane $(\mathcal{C}, b)$ by rules in (3), then the yielded membranes are dissolved by rules in (4). Thus, at this point of the computation two copies of $\overline{w}_1$ are in membrane d_w. However, in the next step one copy is removed due to the corresponding rule in (9). After this, membrane $(\mathcal{C}, c)$ is divided (rules in (5)) and the new membranes are dissolved (rules in (6)). At this point, two copies of $\overline{w}_1$ are in membrane d_w, and one copy is removed by the corresponding rule in (9). Finally, $\overline{w}_1$ is duplicated by rules in (7), and then the two copies of $\overline{w}_1$ introduce two copies of $V_u^{\exists}$. During the dissolution of membranes in $D_{\mathcal{C}}$ one copy of $V_u^{\exists}$ is removed which proves the statement.

Statement 2 can be seen as follows. The computation starts with removing the object $V_u^{\nexists}$ and introducing one w (first rule in (1)). Then the new membranes with label $(\mathcal{C}, a)$ are dissolved by the corresponding rules in (2). In membrane $(\mathcal{C}, b)$ the first rule of (3) is applied and thus w is duplicated. At this point membranes with label $(\mathcal{C}, b)$ have positive charges, thus only the first rule in (4) can be applied. After this the corresponding rule in (9) removes one copy of w. During the next step the first rule in (5) is applied, and then only the first rule in (6) can be used. Again, one copy of w is removed by the corresponding rule in (9). Then the first rule in (7) divides membrane $(\mathcal{C}, d)$, w is again duplicated, and by the first rule in (8) each w introduces one copy of $V_u^{\exists}$. During the work in $D_{\mathcal{C}}$, one copy of $V_u^{\exists}$ is removed.

The system has several different computations in the case of Statement 3. We discuss here only one of them, the remaining ones can be treated similarly. Assume for example that $l_{\mathcal{C}}$ contains $C_{\mathcal{C}}^{\exists}$ and $V_y^{\nexists}$. Then the computation goes in the same way as in the case of Statement 2 until the application of the corresponding dissolution rules in (4). But now the second rule in (5) is applied, and thus, in the next step, only the second rule in (6) can be applied. Therefore here two copies of $\overline{w}_2$ are introduced. Then the computation continues similarly as in Case 2 in the proof of Statement 1. However here, when rules from (8) are applied the system has two copies of $\overline{w}_2$, and thus two copies of $V_u^{\nexists}$ are introduced by the fifth and sixth rules in (8). One of these copies is eliminated during the work in $D_{\mathcal{C}}$.

Clearly, the initial configuration of $\Pi(n)$ satisfies the conditions of Lemma 1. Let $\mathcal{C}$ be a clause having exactly two negative literals. Let moreover $x \in Var_n \cup \{\downarrow\}$. Then at the end of the computation in layer $l_{\mathcal{C}}$ either $V_x^{\exists}$ or $V_x^{\nexists}$ occurs in the system. Therefore the computation of the system in a region corresponds to performing steps $4-5$ of Algorithm H3SN. Since this algorithm performs steps $4-5$ n times, the work of $\Pi(n)$ in the n regions corresponds to the work of the algorithm. This justifies the correctness of $\Pi(n)$.

Since $\Pi(n)$ has polynomial number of membranes in layer $l_{\mathcal{C}}$, and in $l_{\mathcal{C}}$ the number of the applied division rules is constant, we have that dissolving all the membranes in $l_{\mathcal{C}}$ takes polynomial time. As in the initial configuration there are n regions and each region has polynomial number of layers, it follows that the running time of $\Pi(n)$ is also polynomial.

Finally, it can be seen that all the ingredients of $\Pi(n)$ can be enumerated and written onto the output tape by a logarithmic space Turing machine. Using the **P**-completeness of Horn3SatNorm we obtain the following theorem.

Theorem 2. $\mathbf{P} \subseteq (\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{AM}_{+e,+d}}$.

3.2 Simulating Turing Machines

Here we show that, for every polynomial time Turing machine M, an $(\mathbf{L}, \mathbf{L})$-uniform family $\mathbf{\Pi}$ of polarizationless recognizer P systems can be constructed such that the members of $\mathbf{\Pi}$ can simulate the work of M efficiently using only dissolution and 1-restricted evolution rules.

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be an $f(n)$-time Turing machine, for some polynomial $f(n)$. Notice that M can use at most $f(n)$ cells of its tape during its computations. Let $k = |Q|$ and $m = |\Gamma|$. Assume that $Q = \{s_1, \ldots, s_k\}$, where $s_1 = q_0, s_{k-1} = q_a$ and $s_k = q_r$. Likewise, assume that $\Gamma = \{X_1, \ldots, X_m\}$, where $X_m = \sqcup$. The idea of the simulation is the following. The initial membrane structure μ is a composition of $f(n)$ regions. The input membrane is the innermost membrane. During the simulation of the tth step of M, the objects in the innermost membrane will dissolve all the membranes in the tth region as follows. Assume that after $t-1$ steps M is in state s_i ($i \in [k-2]$), the position of the head is p, and the head scans X_j. Then the innermost membrane of the tth region contains an object O that represents s_i and p, and another object O' representing X_j on the pth position of the tape. The regions are composed from $k \cdot m \cdot f(n)$ membranes (that is, in every region, for every state–tape symbol–position triple there is a corresponding membrane). During the simulation of the tth step of M, O dissolves all the membranes that correspond to a state $s_{i'}$ with $i' < i$ or a position $p' < p$. Meanwhile O' evolves using a counter and at the appropriate time step it starts to dissolve all the membranes corresponding to s_i, p, and tape symbol $X_{j'}$ with $j' < j$. After this the simulation of one step of M is performed using the value $\delta(s_i, X_j)$. Then the remaining membranes in the tth region are dissolved, and the system continues with the simulation of the next step of M.

Construction of the P system.

The uniform family of P systems that will perform the above described simulation is defined as follows. Let $w = a_1 \ldots a_n$ be an input of M ($a_1, \ldots, a_n \in \Sigma$) and $N = f(n) \cdot k \cdot m$. Let $cod(w)$ be a multiset over the alphabet

$$\Sigma(n) = \{(X_j, p, t)^{(c)}, (s_i, p, t)^{(c')} \mid j \in [m], i \in [k], p \in [f(n)], t \in [0, f(n)], c \in [0, N], c' \in [0, N+m]\}$$

defined as follows: $cod(w) = \{(a_1, 1, 0)^{(0)}, \ldots, (a_n, n, 0)^{(0)}\} \cup \{(\sqcup, n+1, 0)^{(0)}, \ldots, (\sqcup, f(n), 0)^{(0)}\} \cup (s_1, 1, 0)^{(0)}$. Intuitively, an object $(X_j, p, t)^{(c)}$ in $\Sigma(n)$ represents the fact that after t steps M has X_j on the pth position of its tape. We call these objects *position objects*. Similarly, an object $(s_i, p, t)^{(c')}$ represents the fact that after t steps M is in state s_i and the head points to the pth position of the tape. We call these objects *state objects*. The indexes c, c' are counters used for technical reasons. It can be seen that $cod \in \mathbf{L}$.

Let $\mathbf{\Pi} = \{\Pi(n)\}_{n\in\mathbb{N}}$ be a uniform family of P systems, where $\Pi(n) = (\Gamma(n), H(n), \mu(n), W(n), R(n))$ is defined as follows:

- $\Gamma(n) = \Sigma(n) \cup \{yes, no\}$.
- $H(n) = \{(s_i, p, X_j, t) \mid i \in [k], p, t \in [f(n)], j \in [m]\}$.
  Intuitively, a label (s_i, p, X_j, t) corresponds to the following configuration of M after t steps on w: the current state is s_i, the position of the head is p, and the scanned symbol is X_j. We will often call s_i, p, and t the *state*, *position*, and *time* labels of the corresponding membrane, respectively.
- $\mu(n)$ is a composition $S[r_{f(n)}[\ldots[r_1]]]$ of regions, where $S = [\ ]_{skin}$, and a region r_t ($t \in [f(n)]$) is a composition of layers defined as follows. For every $i \in [k]$ and $p \in [f(n)]$, let $l_{s_i,p,t} = [\ldots[\ ]_{(s_i,p,X_1,t)}\ldots]_{(s_i,p,X_m,t)}$, and let $r_t = l_{s_k,f(n),t}[\ldots[l_{s_k,1,t}[\ldots[l_{s_1,f(n),t}[\ldots[l_{s_1,1,t}]\ldots]]\ldots]]\ldots]$.
- The input membrane is the innermost membrane in $\mu(n)$.
- $W(n)$ is a sequence of empty initial multisets.
- R consists of the following sets of rules:

  (1) $[(s_i, p, t)^{(0)}]_{(s_{i'}, p', X_j, t+1)} \to (s_i, p, t)^{(0)}$
  ($j \in [m]$, $i \in [k-2], i' \in [k]$, $p, p' \in [f(n)], t \in [0, f(n)-1]$, and $i' < i$ or $p' < p$).
  These rules are used to find the first such membrane whose state and position labels correspond to the state and position stored in the state object.

  (2) $[(X_j, p, t)^{(c)} \to (X_j, p, t)^{(c+1)}]_{(s_i, p', X_{j'}, t+1)}$
  ($j, j' \in [m], i \in [k], p, p' \in [f(n)], t \in [0, f(n)-1], c \in [0, N-1]$).
  These rules are used to increment the counter c in the position objects. When this counter equals to N, the system can start to use rules in (3).

  (3) $[(X_j, p, t)^{(N)}]_{(s_i, p, X_l, t+1)} \to (X_j, p, t)^{(N)}$,
  $[(X_j, p, t)^{(N)} \to (X_{j'}, p, t+1)^{(0)}]_{(s_i, p, X_j, t+1)}$,
  $[(X_j, p', t)^{(N)} \to (X_j, p', t+1)^{(0)}]_{(s_i, p, X_1, t+1)}$
  ($j, l \in [m]$, $l < j$, $i \in [k-2]$, $p, p' \in [f(n)]$, $p \neq p'$, $t \in [0, f(n)-1]$, and $X_{j'} = \mathrm{proj}_2(\delta(s_i, X_j))$).
  If the position stored in an object $(X_j, p, t)^{(N)}$ corresponds to the position label of the current membrane, then this object starts to dissolve the membranes until a membrane whose label stores X_j is found. When this membrane is found, $(X_j, p, t)^{(N)}$ evolves according to the value of $\delta(s_i, X_j)$, its counter is reset, and its component t is incremented. Those position objects that store a different position than the position label of the current

membrane evolve immediately such that their counter is reset and their component t is incremented. Notice that after performing the computations by these rules, the position objects have no impact on the computation in region r_{t+1}.

(4) $[(s_i, p, t)^{(c)} \to (s_i, p, t)^{(c+1)}]_{(s_i,p,X_l,t+1)}$,
$[(s_i, p, t)^{(N+m)} \to (s_{i'}, p', t+1)^{(0)}]_{(s_i,p,X_l,t+1)}$
$(i \in [k-2], i' \in [k]$, $p \in [f(n)]$, $t \in [0, f(n)-1]$, $c \in [N+m-1]$, $l \in [m]$, $s_{i'} = \text{proj}_1(\delta(s_i, X_l))$, $p' = \max\{p + \text{proj}_3(\delta(s_i, X_l)), 1\})$.
The counter of the state object is incremented using the first rule. Until the counter reaches $N+m$, the appropriate position object can find the corresponding membrane using rules in (3). Then the state object evolves according to the value of the transition function of M. Moreover, its counter is reset and its component t is incremented.

(5) $[(s_i, p, t+1)^{(0)}]_{(s_{i'},p',X_j,t+1)} \to (s_i, p, t+1)^{(0)}$
$(i \in [k-2], i' \in [k], p, p' \in [f(n)], j \in [m], t \in [0, f(n)-1])$.
After simulating a step of M using rules in (1)-(4), the remaining membranes in region r_{t+1} are dissolved by these rules.

(6) $[(s_{k-1}, p, t)^{(0)} \to yes]_h$, $[(s_k, p, t)^{(0)} \to no]_h$,
$[yes]_{h'} \to yes$, and $[no]_{h'} \to no$
$(p, t \in [f(n)], h \in H(n), h' \in H(n) - \{skin\})$.
These rules are used to produce the answer of $\Pi(n)$ according to which halting state is reached by M on the input.

Correctness, running time, and $(\mathbf{L}, \mathbf{L})$*-uniformity.*

Let $w = a_1 \dots a_n$ be an input of M ($a_1, \dots, a_n \in \Sigma$). We show that $\Pi(n)$ produces *yes* started with $cod(w)$ in its input membrane if and only if $w \in L(M)$. The work of $\Pi(n)$ can be described as follows. Initially, the object $(s_1, 1, 0)^{(0)}$ (representing that M starts its work in its initial state and the head is positioned to the first letter of the input) is in the innermost membrane of region r_1. Now assume that $\Pi(n)$ has already simulated t steps of M, that is, the innermost membrane of $\Pi(n)$ is the most deeply nested membrane of region r_{t+1}, and this membrane contains an object $(s_i, p, t)^{(0)}$, for some $i \in [k]$ and $p \in [f(n)]$. If $i \in [k-1, k]$, i.e., M has reached one of its halting states, then $\Pi(n)$, using rules in (6) computes the answer of the system *yes* or *no* accordingly. Otherwise, rules from (1) are applied until a membrane with label $(s_i, p, X_1, t+1)$ is reached. Meanwhile, the counter c in position objects is incremented using rules in (2). By the time this counter becomes N, the corresponding membrane is reached by the rules in (1).

Now those position objects that store different positions than p evolve by the third rule in (3) to such objects that will be used next time only in the next region r_{t+2} (i.e., in the simulation of the next step of M). Concerning the position object storing p, assume that this object is $(X_j, p, t)^{(N)}$. Then $(X_j, p, t)^{(N)}$ is used to find that membrane in layer $l_{s_i,p,t+1}$ whose label contains X_j. When this membrane is

found, $(X_j, p, t)^{(N)}$ evolves according to the transition function of M. Moreover, its counter is reset and its time component is incremented. Thus this object is not used any more in this region.

Meanwhile, rules in (4) are used to increment the counter c of $(s_i, p, t)^{(c)}$. By the time this counter becomes $N+m$, the position object $(X_j, p, t)^{(N)}$ has reached the membrane it searched for. Now the second rule in (4) is used to produce object $(s_{i'}, p', t+1)^{(0)}$ where $s_{i'}$ and p' are calculated according to the transition function of M. Finally, $(s_{i'}, p', t+1)^{(0)}$ is used to dissolve the remaining membranes of r_{t+1}. If this is done, the system is ready to simulate the next step of M. With this we have seen that $\Pi(n)$ simulates correctly the computation of M on w.

It can be seen that dissolving a region in the membrane structure takes $O(N)$ steps and $N = O(f(n))$. Moreover, $\Pi(n)$ has $f(n)$ regions. Thus the running time of the system is $O(f^2(n))$, that is, polynomial in n. The $(\mathbf{L}, \mathbf{L})$-uniformity of $\mathbf{\Pi}$ follows from the observation that the size of $\Pi(n)$ is also polynomial in n. Thus we have the following result.

Theorem 3. $\mathbf{P} \subseteq (\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{AM}^0_{+evo(1),+d}}$.

As we have observed on page 197, our solution of Horn3SatNorm by P systems of type $\mathcal{AM}_{+e,+d}$ is such that every membrane has at most two child membranes in every configuration of each computation of the system. Let $k \geq 1$. We say that a P system Π is k-bounded, if every membrane has at most k child membranes in every configuration of each computation of Π. For a type $\mathcal{F}$ of P systems, denote $\mathbf{PMC}_{\mathcal{F}_{\leq k}}$ the set of those problems that can be decided by such polynomially uniform families of P systems of type $\mathcal{F}$ which have k-bounded members only. Denote $\mathcal{AM}_{-e}$ those P systems with active membranes that do not employ membrane division rules. It can be seen using the generalization of the proof of $\mathbf{PMC}_{\mathcal{AM}_{-e}} \subseteq \mathbf{P}$ in [30] that $\mathbf{PMC}_{\mathcal{AM}_{\leq 2}} \subseteq \mathbf{P}$ also holds. Using the results obtained in the paper we can give the following new characterizations of $\mathbf{P}$.

Corollary 1. $\mathbf{P} = (\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{AM}_{+out}} = (\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{AM}_{+e,+d,\leq 2}} = (\mathbf{L}, \mathbf{L}) - \mathbf{PMC}_{\mathcal{AM}^0_{+evo(1),+d}}$.

4 Conclusions

In this paper we have shown that uniform families of the following restricted variants of P systems with active membranes can solve all problems in $\mathbf{P}$: (1) P systems where only out communication rules are used, (2) P systems where only elementary membrane division and dissolution rules are used, and (3) polarizationless P systems where only dissolution and 1-restricted evolution rules are used. Using the obtained results concerning variants (1) and (3), and known results about the upper bound on the power of these variants we could give new characterizations of $\mathbf{P}$ in terms of Membrane Computing techniques.

It remained an open question if the variant (2) could solve problems outside of **P**. It is known that without polarizations of the membranes this is not possible [29]. It is also an open question if these systems can solve all problems in **P** when polarizations of the membranes are not allowed. Nevertheless, we could give another characterization of **P** using variant (2) when we made a simple semantic restriction on the computations of this variant.

References

1. Alhazov, A., Martín-Vide, C., Pan, L.: Solving a PSPACE-Complete Problem by P Systems with Restricted Active Membranes. Fundamenta Informaticae **58** (2003) 67–77
2. Alhazov, A., Pan, L., Păun, G.: Trading polarizations for labels in P systems with active membranes. Acta Inf. **41**(2-3) (2004) 111–144
3. Gazdag, Z,: Solving SAT by P Systems with Active Membranes in Linear Time in the Number of Variables. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Y., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 14th International Conference, LNCS vol. 8340 (2014) 189–205
4. Gazdag, Z., Kolonits, G.: A new approach for solving SAT by P systems with active membranes. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) Membrane Computing: 13th International Conference, LNCS vol. 7762 (2013) 195–207
5. Gazdag, Z., Gutiérrez-Naranjo, M.A.: Solving the ST-connectivity problem with pure membrane computing techniques. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Sosík, P., Zandron, C. (eds.) Membrane Computing: 15th International Conference, LNCS vol. 8961 (2014) 215–228
6. Gazdag, Z., Kolonits, G., Gutiérrez-Naranjo, M.A.: Simulating Turing machines with polarizationless P systems with active membranes. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Sosík, P., Zandron, C. (eds.) Membrane Computing: 15th International Conference, LNCS vol. 8961 (2014) 229–240
7. Gensler, H.J.: Introduction to Logic, Routledge, London (2002)
8. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.J.: On the Power of Dissolution in P Systems with Active Membranes. In: Freund, R., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 6th International Workshop, LNCS vol. 3850 (2006) 224–240
9. Kolonits, G.: A Solution of Horn-SAT with P Systems Using Antimatter. In: Membrane Computing: 16th International Conference, LNCS vol. 9504 (2015) 236–250
10. Krishna, S.N., Rama, R.: A variant of P systems with active membranes: Solving NP-complete problems. Romanian J. of Information Science and Technology, 2, 4 (1999) 357–367
11. Murphy, N.: Uniformity conditions for membrane systems: uncovering complexity below P. Ph.D. thesis, National University of Ireland, Maynooth (2010)
12. Murphy, N., Woods, D.: Active Membrane Systems Without Charges and Using Only Symmetric Elementary Division Characterise P. In: Eleftherakis, G., Kefalas, P., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 8th International Workshop, LNCS vol. 4860 (2007) 367–384

13. Murphy, N., Woods, D.: A Characterisation of NL Using Membrane Systems without Charges and Dissolution. In: Calude, C.S. et al (eds.) Unconventional Computing: 7th International Conference, LNCS vol. 5204 (2008) 164–176
14. Murphy, N., Woods, D.: On acceptance conditions for membrane systems: characterisations of **L** and **NL**. In Neary, T., Woods, D., Seda, T., Murphy, N., eds.: Proceedings International Workshop on The Complexity of Simple Programs, Cork, Ireland, Volume 1 of Electronic Proceedings in Theoretical Computer Science., Open Publishing Association (2009) 172–184
15. Murphy, N., Woods, D.: The computational power of membrane systems under tight uniformity conditions. Natural Computing **10**(1) (2011) 613–632
16. Murphy, N., Woods, D.: Uniformity is weaker than semi-uniformity for some membrane systems. Fundam. Inf. **134**(1-2) (2014) 129–152
17. Pan, L., Alhazov, A., Isdorj, T.-O.: Further remarks on P systems with active membranes, separation, merging, and release rules. Soft Computing **9**(9) (2004) 686–690
18. Pan, L., Ishdorj, T.-O.: P systems with active membranes and separation rules. Journal of Universal Computer Science 10(5) (2004) 630649
19. Păun, Gh.: P Systems with Active Membranes: Attacking NP-Complete Problems. Journal of Automata, Languages and Combinatorics **6**(1) (2001) 75–90
20. Păun, Gh.: Further twenty six open problems in membrane computing. In: Third Brainstorming Week on Membrane Computing. Fénix Editora, Sevilla (2005) 249–262
21. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)
22. Pérez-Jiménez, M.J., Romero-Campero, F.J.: Trading Polarization for Bi-stable Catalysts in P Systems with Active Membranes. In: Mauri, G., Păun, G., Prez-Jimnez, M.J., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing: 5th International Workshop, LNCS vol. 3365 (2005) 373–388
23. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: A polynomial complexity class in P systems using membrane division. In: Csuhaj-Varjú, E., Kintala, C., Wotschke, D., Vaszil, G. (eds.) Proceeding of the 5th Workshop on Descriptional Complexity of Formal Systems. DCFS 2003 (2003) 284–294
24. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Natural Computing **2**(3) (2003) 265–285
25. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: A polynomial complexity class in P systems using membrane division. Journal of Automata, Languages and Combinatorics **11**(4) (2006) 423–434
26. Salomaa, A.: Formal Languages. Academic Press, New York, London (1973)
27. Sipser, M.: Introduction to the Theory of Computation. 3rd edn. Cengage Learning (2012)
28. Sosík, P.: The computational power of cell division in P systems. Nat. Comput. **2**(3) (2003) 287–298
29. Woods, D., Murphy, N., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Membrane Dissolution and Division in P. In: Calude, C.S., da Costa, J.F.G., Dershowitz, N., Freire, E., Rozenberg, G. (eds.) Unconventional Computation: 8th International Conference, LNCS vol. 5715 (2009) 262–276
30. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-Complete Problems Using P Systems with Active Membranes. In: Unconventional Models of Computation, UMC2K: Proceedings of the Second International Conference on Unconventional Models of Computation. Springer London, London (2001) 289–301