mathematics

MDPI

Article

P Systems with Proteins on Active Membranes

Chuanlong Hu, Yanyan Li and Bosheng Song *

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
* Correspondence: boshengsong@hnu.edu.cn

Abstract: P systems with active membranes, as a sort of basic P system, include *in* communication rules and *out* communication rules, where communication rules are controlled by polarizations. However, the communication of objects among living cells may be controlled by several factors, such as proteins, polarizations, etc. Based on this biological fact, in this article, a new class of P systems, named P systems with proteins on active membranes (known as PAM P systems) is considered, where the movement of objects is controlled by both proteins and polarizations. The computational theory of PAM P systems is discussed. More specifically, we show that PAM P systems achieve Turing universality when the systems use two membranes, one protein and one polarization. Moreover, the PAM P systems, with the help of membrane division rules, make the SAT problem solvable. These results indicate that PAM P systems are also a sort of powerful system.

Keywords: SAT problem; Turing universality; P systems; active membranes; membrane computing

MSC: 68Q07; 68Q10; 68Q17

Citation: Hu, C.; Li, Y.; Song, B. P Systems with Proteins on Active Membranes. *Mathematics* **2022**, *10*, 4076. https://doi.org/10.3390/math10214076

Academic Editor: Konstantin Kozlov

Received: 9 October 2022
Accepted: 29 October 2022
Published: 2 November 2022

1. Introduction

Membrane computing, which aims to discuss computational models abstracted from the functioning and structure of biological cells, is one sort of natural computing, where the models are known as *P systems* [1]. All computing models have in common a set of compartments separated by membranes and organized by a certain structure (tree, graph). In general, three main sorts of P systems are discussed: *cell-like* [1–5], *tissue-like* [6–11] and *neural-like* [12–20] P systems. Since the first article about membrane computing was published, a large bibliography has been accumulated [5,21–25]. More information on membrane computing is available on the website http://imcs.org.cn/ or http://ppage.psystems.eu (accessed on 8 October 2022), where readers can view and download papers in this field.

P systems with active membranes (known as AM P systems) were first considered in [26], and the structure of these membranes is represented by a tree. It is an obvious characteristic of AM P systems that the movement and evolution of objects are controlled by polarization associated with each membrane, where polarization changes between positive, negative, or neutral. Some further works on AM P systems have been published, such as AM P systems and separation rules [27]: separation rules are imported into AM P systems, and this proves that such P systems are able to make the SAT problem solvable in polynomial time; AM P systems without using polarizations [28]: in this system, polarization is avoided but modification of the labels of the membranes is allowed. It has been proved that the computational power of AM P systems and their variants is powerful; most of them were proved to achieve Turing universality, and they can give the result of **NP**-complete problems or even **PSPACE**-complete problems [5,29–33].

P systems with proteins on membranes (known as PM P systems) were first considered in [34]. Such systems include two types of objects: usual objects and proteins. Usual objects are placed in membrane or in the environment; proteins are placed on membranes (note that a multiset of proteins is placed on a membrane). According to whether proteins

can be changed or not in systems [34], there are two groups of rules: "*res*" (representing "restricted") and "*cp*" (representing "change protein"). In [35], a special class of PM P system, named P systems with flip-flop proteins on membranes, was considered, where at most two states are allowed for each protein. The SAT problem is solvable with the help of PM P systems and membrane division rules [36]. Readers who want to learn more information about PM P systems can review the references [34,37].

As mentioned above, AM P systems contain communication rules, which are controlled by polarizations. PM P systems contain several types of rules, which are controlled by proteins. Obviously, rules in AM P systems and in PM P systems are controlled by only one factor. However, chemical reactions happening among living cells may be controlled by several factors, such as proteins, polarizations, etc. Based on this biological fact, in this article, a new sort of P system, named P systems with proteins on active membranes (known as PAM P systems), is considered, where the movement of objects is controlled by both protein and polarization.

The computational theory of PAM P systems is studied. More specifically, we indicate that PAM P systems achieve Turing universality when the systems use two membranes, one protein and one polarization. Moreover, the PAM P systems, with the help of membrane division rules, make the SAT problem solvable. The results among the P systems with active membranes, P systems with proteins on membranes and this paper are in Table 1.

Table 1. Results of the P systems with active membranes, P systems with proteins on membranes and this paper, where $NOP_m(pro_r; act_c; list - of - rules)$ represents the family of all sets of numbers produced by P systems with m membranes, r proteins in each membrane, and which use c polarizations and types of rules $list - of - rules$; $*$ represents the case where parameter m or r are boundless; the types of rules $2cpp$ or $3ffp$ combined with the polarizations of membranes become the types of rules (3) and (4); the types of rules (b) and (c) combined with proteins on membranes become the types of rules (3) and (4); the types of rules (3) and (4) are defined in this paper; and NRE is the family of all recursively enumerable sets of natural numbers.

P Systems with Active Membranes	P systems with Proteins on Membranes	This Paper
$NOP_2(act_2; (a), (c)) = NRE$ [38]	$NOP_1(pro_2; 2cpp) = NRE$ [34]	$NOP_2(pro_1; act_1; (3), (4)) = NRE$
$NOP_2(act_1; (a), (b'), (c)) = NRE$ [38]	$NOP_1(pro_*; 3ffp) = NRE$ [34]	

2. Preliminaries and Model Description

2.1. Preliminaries

In this subsection, the concepts of formal languages and automata theory are introduced, which will be employed in this article. Readers who want to obtain more details can refer to [39].

A non-empty finite set constitutes an *alphabet* denoted by Γ, where each element in the set is known as a symbol. All strings created by joining any symbols from Γ form a set which is symbolized by Γ^*. We use λ to represent the case that no symbol appears in a string, which is called *empty string*. The non-empty set of Γ^* is symbolized as $\Gamma^+ = \Gamma^* \setminus \{\lambda\}$. $|u|$ is known as the *length* of u, which symbolizes the total number of symbols presented in string u.

For an alphabet Γ, we use the ordered pair (Γ, f) to represent a *multiset* over Γ symbolized by $\mathcal{M}$, where f is the mapping of Γ to $\mathbb{N}$ (natural numbers set). In addition, $\mathcal{M}(\Gamma)$ and $\mathcal{M}^+(\Gamma)$ are known as the set of all multisets and of all non-empty multisets, respectively. Given a $\Gamma = \{a_1, \ldots, a_k\}$, $\{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$ expresses multiset $\mathcal{M}$.

A tuple $M = (m, H, l_0, l_h, I)$ symbolizes a *register machine* including m registers, a label set H, an initial instruction $l_0 \in H$, the halting instruction $l_h \in H$, and a labeled program-instructions set I whose instructions are of two types: (1) $l_i : (\mathtt{ADD}(r), l_j, l_k)$ (register r increases by one, after that the instructions l_j or l_k are executed in a non-deterministical

way); and (2) $l_i : (\mathtt{SUB}(r), l_j, l_k)$ (subtract 1 from register r if it is non-zero, after that, execute the instruction l_j; otherwise, execute the instruction l_k).

$N(M)$ denotes a set including all numbers that are the result of a register machine M executing instructions. The M works as follows: being empty in all registers at the initial moment, l_0 is applied to start the machine and subsequent instructions continue to be applied according to the labels; the register 1 contains a number n corresponding to the result generated by M only when the instruction labeled l_h appears in the system. It is known that all sets of numbers generated by register machines are Turing computable; therefore, both of them recognize the same family of sets of numbers NRE (the family of all recursively enumerable sets of natural numbers) [40].

2.2. PAM P Systems

In the following, the definition of PAM P systems will be given. Readers who want to learn more about AM P systems and PM P systems can review [26] and [34], respectively.

Definition 1. *A P system with proteins on active membranes (PAM P system) of degree $m \geq 1$ is a tuple*

$$\Pi = (O, P, \mu, H, w_1/z_1, \ldots, w_m/z_m, \mathcal{E}, R, i_{out}),$$

where:

- *O symbolizes an alphabet including all symbols of objects;*
- *P symbolizes an alphabet including all the symbols of proteins where $O \cap P = \emptyset$;*
- *μ symbolizes the initial structure of a membrane including a set of membranes with labels in $1, \ldots, m$;*
- *H symbolizes a size-limited set of labels of membranes;*
- *$\mathcal{E} \subseteq O$ symbolizes a set including all objects that are initially placed in the environment;*
- *$w_1 \ldots w_m$, symbolize all multisets of objects from O;*
- *$z_1 \ldots z_m$, symbolize all multisets of proteins from P;*
- *R symbolize a set including a finite number of rules which have the following types:*
  - *(1)* $a[p|\,]_i^{e_1} \to b[p'|\,]_i^{e_2}, i \in H, p, p' \in P, a, b \in O, e_1, e_2 \in \{+, -, 0\}$
    (this rule can be used only when an object a appears in the parent of membrane i, the protein p is placed on membrane i and its polarization is e_1. When such a rule is applied, an object a may be evolved to b, the protein p may be evolved to p', and the polarization of the membrane may also be modified);
  - *(2)* $[p|a]_i^{e_1} \to [p'|b]_i^{e_2}, i \in H, p, p' \in P, a, b \in O, e_1, e_2 \in \{+, -, 0\}$
    (this rule can be used only when an object a appears in membrane i, the protein p is placed on membrane i and its polarization is e_1. When such a rule is applied, an object a may be evolved to b, the protein p may be evolved to p', and the polarization of the membrane may also be modified);
  - *(3)* $a[p|\,]_i^{e_1} \to [p'|b]_i^{e_2}, i \in H, p, p' \in P, a, b \in O, e_1, e_2 \in \{+, -, 0\}$
    (this rule can be used only when an object a appears in the parent of membrane i, the protein p is placed on membrane i and its polarization is e_1. When such a rule is applied, an object a is moved into membrane i, and possibly evolved to b during this process, the protein p may be evolved to p', and the polarization of the membrane may also be modified);
  - *(4)* $[p|a]_i^{e_1} \to b[p'|\,]_i^{e_2}, i \in H, p, p' \in P, a, b \in O, e_1, e_2 \in \{+, -, 0\}$
    (this rule can be used only when an object a appears in the membrane i, the protein p is placed on membrane i and its polarization is e_1. When such a rule is applied, an object a is sent out of membrane i, and possibly evolved to b during this process; the protein p may be evolved to p'; and the polarization of the membrane may also be modified);
  - *(5)* $a[p|b]_i^{e_1} \to c[p'|d]_i^{e_2}, i \in H, p, p' \in P, a, b, c, d \in O, e_1, e_2 \in \{+, -, 0\}$
    (this rule can be used only when an object a is contained in the parent of membrane i, an object b is contained in membrane i, a protein p is placed on membrane i and its polarization is e_1. When such a rule is applied, an object a will be sent into membrane

i, and possibly evolved to d during this process; simultaneously, an object b will be sent out of membrane i, and possibly evolved to c during this process; the protein p may be evolved to p'; and the polarization e_1 may also be modified to e_2);

(6) $[p|\ ]_i^e \rightarrow [p'|\ ]_i^{e_1}[p''|\ ]_i^{e_2}$, $i \in H$, $p, p', p'' \in P$, $e, e_1, e_2 \in \{+, -, 0\}$, $1 \leq i \leq m$, $i \neq i_{out}$, *and i is unable to be the root of the tree*
(when such a rule is applied, membrane i is divided into two membranes without changing label, with protein p evolved to p' and p'', with polarization e modified to e_1 and e_2, respectively; all the objects in the parent membrane are duplicated in the two new membranes).

- $i_{out} \in \{0, 1, \ldots, m\}$ *is the output area (0 is the label of environment).*

The maximally parallel way is a common strategy of using rules in a PAM P system, where the applicable rules of a PAM P system are added into a maximal multiset to be employed (no more rules are able to be added) at each step. A PAM P system working in this way has the following limitations: any object can be used in only one rule of any type, and any membrane can be used in only one rule of types (1)–(5); when a rule of type (6) (membrane division rules) is applied, the system cannot employ other rules for that membrane in that step.

For a PAM P system as defined above, a *configuration* $\mathcal{C}_t$ contains the following factors: the membrane structure and polarization of membranes at moment t; all objects presented in each area of this membrane structure; all proteins presented on membranes; and all objects presented in the environment at moment t. We can obtain the *initial configuration* from $(\mu, w_1 \ldots, w_m, z_1 \ldots z_m, \mathcal{E})$, and the polarization of each membrane in the initial state is neutral by default.

A system which starts from initial configuration and works in a maximally parallel way with the restrictions mentioned above can achieve a series of consecutive configurations. If all the rules of a system are unavailable in a configuration, this configuration is known as a *halting configuration*. A *transition* denoted by $\mathcal{C}_t \Rightarrow \mathcal{C}_{t+1}$ represents that one configuration $\mathcal{C}_t$ transfers to the next configuration $\mathcal{C}_{t+1}$. A (finite or infinite) sequence of transitions between configurations indicates a *computation* for a PAM P system. Only *halting computations* whose last configuration is a halting configuration give a result which appears in the output area i_{out} as a multiset of objects.

$N(\Pi)$ symbolizes a set including all numbers produced by a PAM P system Π. $NOP_m(pro_r; act_c; list - of - rules)$ symbolizes the family of sets of numbers $N(\Pi)$ produced by the system Π with the following explanations: (1) the number of membranes are at most m; (2) the total proteins placed on one membrane are no more than r; (3) c represents the number of polarizations which will be used in the system; and (4) $list - of - rules$ includes the specifical types of rules used in the system. When parameters r or m are boundless, we use $*$ instead.

To study the computational efficiency of the PAM P systems, the definition of such systems will be given below [29].

Definition 2. *A recognizer PAM P system of degree $m \geq 1$ is defined as a tuple*

$$\Pi = (O, P, \Sigma, \mu, H, w_1/z_1, \ldots, w_m/z_m, \mathcal{E}, R, i_{in}, i_{out}),$$

where:

- *O symbolizes an alphabet including all the symbols of usual objects and two special objects* `yes` *and* `no`*;*
- *P symbolizes an alphabet including all the symbols of proteins;*
- $\Sigma \subseteq O$ *symbolizes an input alphabet;*
- μ *symbolizes the initial structure of the membranes including a set of membranes with labels in* $1, \ldots, m$*;*
- *H symbolizes a size-limited set of labels of membranes;*

- $\mathcal{E} \subseteq O$ *symbolizes a set including all the objects initially placed in the environment;*
- $w_1 \dots w_m$, *symbolize multisets of objects from O;*
- $z_1 \dots z_m$, *symbolize multisets of proteins from P;*
- *R symbolizes a set including the limited number of rules described above in each membrane i;*
- $i_{in} \in \{0, 1, \dots, m\}$ *is the input area and* $i_{out} = 0$;
- *all computations halt;*
- *the halting condition of* Π *is that the system must send either object* `yes` *or object* `no` *(but not both) into the environment only at the last step of the computation.*

In the following, we will give the definition for which the problem is solvable in polynomial time (in a uniform way) by a family of PAM P systems [41].

Definition 3. *A family of PAM P systems* $\mathbf{\Pi} = \{\Pi(n) | n \in \mathbb{N}\}$ *can give the polynomial time result of a decision problem* $X = (I_X, \Theta_X)$ *if the following conditions are satisfied:*

- *the family* $\mathbf{\Pi}$ *is polynomially uniform by Turing machines;*
- *there exists a pair* (cod, s) *of polynomial-time computable functions over* I_X *such that:*
  - *for each instance* $u \in I_X$, $s(u)$ *is a natural number and* $cod(u)$ *is an input multiset of system* $\Pi(s(u))$;
  - *for each* $n \in \mathbb{N}$, $s^{-1}(n)$ *is a finite set;*
  - *the family* $\mathbf{\Pi}$ *is polynomially bounded with regard to* (X, cod, s);
  - *the family* $\mathbf{\Pi}$ *is sound with regard to* (X, cod, s);
  - *the family* $\mathbf{\Pi}$ *is complete with regard to* (X, cod, s).

$\mathbf{PMC}_{\mathbf{MRC}(list-of-rules)}$ symbolizes a set including all decision problems for which the recognizer PAM P systems work in a maximally parallel way to give a result with polynomial time cost, where *list-of-rules* represents the types of rules used in the system.

3. Computational Power of PAM P Systems

In what follows, the computational power of PAM P systems will be considered. As we all know, the register machine has Turing universality, so we can prove that the PAM P systems are equivalent to the register machine to prove its Turing universality.

Theorem 1. $NOP_2(pro_1; act_1; (3), (4)) = NRE$.

Proof. We design the PAM P system Π to imitate the register machine M.

$$\Pi = (O, P, [[\,]_2^0]_1^0, H, \lambda/l_0, \lambda/p, \mathcal{E}, R, 1),$$

where

- $O = \{a_r | 1 \leq r \leq m\} \cup \{c, c', c''\}$,
- $P = \{l, l', l'' | l \in H\} \cup \{p\}$,
- $H = \{1, 2\}$,
- $\mathcal{E} = \{a_r | 1 \leq r \leq m\} \cup \{c\}$.

We assume that the numbers of copies of object a_r are infinite.

To simulate the ADD instruction $l_i : (\mathtt{ADD}(r), l_j, l_k)$ of M, the R is designed as follows:

$r_1 : a_r[l_i|\,]_1^0 \to [l_j|a_r]_1^0$,
$r_2 : a_r[l_i|\,]_1^0 \to [l_k|a_r]_1^0$.

This system takes one step to simulate an ADD instruction l_i. Rules r_1 and r_2 are applied non-deterministically. By using r_1 or r_2, membrane 1 receives one object a_r, and protein l_i on membrane 1 is modified to l_j or l_k. Consequently, the system increases one copy of object a_r and turns to simulate the instruction l_j or l_k.

To simulate the SUB instruction $l_i : (\mathtt{SUB}(r), l_j, l_k)$ of M, the R is designed as follows:

$r_3 : c[l_i|\,]_1^0 \to [l_i'|c]_1^0$;

$r_4 : [l_i'|a_r]_1^0 \to a_r[l_i''|\,]_1^0$;
$r_5 : [l_i''|c'']_1^0 \to c[l_j|\,]_1^0$;
$r_6 : [l_i'|c'']_1^0 \to c[l_k|\,]_1^0$;
$r_7 : c[p|\,]_2^0 \to [p|c']_2^0$;
$r_8 : [p|c']_2^0 \to c''[p|\,]_2^0$.

The system takes four steps to simulate a SUB instruction l_i. At step one, with the influence of protein l_i, rule r_3 is employed, object c in the environment is sent to membrane 1, and protein l_i is modified to l_i' on membrane 1. In the next steps, the running of the system is divided into two cases, according to whether membrane 1 contains objects a_r or not.

- At least one copy of object a_r exists in membrane 1. There are two rules r_4 and r_7 applied in a parallel way in step two. The system sends a copy of object a_r out of membrane 1 and protein l_i' is modified to l_i''. In addition, membrane 2 receives an object c which is modified to c' in this process. In step three, the system employs rule r_8 to send object c' back to membrane 1 and modifies object c' to c''. In step four, rule r_5 is applied, and object c'' is sent to the environment, which is modified to c in this process; simultaneously, protein l_i'' is revised to l_j on membrane 1. Consequently, the system decreases one copy of object a_r and turns to simulating instruction l_j.
- No object a_r exists in membrane 1. Only one rule r_7 is employed in step two. The system sends object c into membrane 2 and modifies object c to c' in this process. In step three, object c' is sent out of membrane 2, which is modified to c'' in this process. In step four, the system employs rule r_6 to send object c'' into the environment and modifies c'' to c; simultaneously, protein l_i' is modified to l_k on membrane 1. Consequently, the system turns to simulating instruction l_k.

A system only halts when protein l_h appears on membrane 1. Membrane 1 contains the number of copies of object a_r corresponding to the result generated by Π. Therefore, Π is equivalent to M. □

4. Solving the SAT Problem using PAM P Systems

The SAT problem is a classic **NP**-complete problem [42] with the following description: *judging whether a Boolean formula in conjunctive normal form (CNF) is satisfiable, that is, whether there is a truth-value assignments that means the value of CNF is true.*

In the following, we will prove that working in a maximally parallel way, a family of PAM P systems is able to give the result of the SAT problem in polynomial time.

Theorem 2. $\mathtt{SAT} \in \mathbf{PMC}_{\mathbf{MRC}((2),(3),(4),(6))}$.

Proof. Let $\varphi = C_1 \wedge \cdots \wedge C_m$ be a Boolean formula, where $C_i = y_{i,1} \vee \cdots \vee y_{i,p_i}$, with $y_{i,j} \in \{x_k, \neg x_k \mid 1 \leq k \leq n\}, 1 \leq i \leq n, 1 \leq j \leq p_i$.

We use a family of PAM P systems $\mathbf{\Pi} = \{\Pi(t) | t \in \mathbb{N}\}$ to work out the result of the SAT problem in polynomial time, where all propositional formulas φ have m clauses and n variables.

We define $s(\varphi) = \langle n, m \rangle$ and

$$cod(\varphi) = \alpha_{1,1} \ldots \alpha_{n,1} \alpha_{1,2} \ldots \alpha_{n,2} \ldots \alpha_{1,m} \ldots \alpha_{n,m},$$

where, for each $1 \leq i \leq n, 1 \leq j \leq m$, we have:

$$\alpha_{i,j} = \begin{cases} d_{i,j} & \text{if } x_i \text{ appears in } C_j; \\ d_{i,j}' & \text{if } \neg x_i \text{ appears in } C_j; \\ d_{i,j}'' & \text{if } x_i \text{ and } \neg x_i \text{ do not appears in } C_j. \end{cases}$$

For each $m, n \in \mathbb{N}$, we design the recognizer PAM P system

$$\Pi = (O, P, \Sigma, [\,]_1^0, H, w_1/z_1, \mathcal{E}, R, 1, 0),$$

where

- $O = \Sigma \cup \{m_i | 1 \le i \le nm + m + 3\} \cup \{c_j | 1 \le j \le m\} \cup \{e, m, \texttt{yes}, \texttt{no}\}$;
- $P = \{\beta_i | 1 \le i \le nm + m + 3\} \cup \{p_{i,j}, p_{i,j}^+, p_{i,j}^- | 1 \le i \le n+1, 1 \le j \le m+1\}$;
- $\Sigma = \{d_{i,j}, d'_{i,j}, d''_{i,j} | 1 \le i \le n, 1 \le j \le m\}$;
- $H = \{1\}$;
- $w_1 = \{m_1, cod(\varphi)\}$;
- $z_1 = \{\beta_1, p_{1,1}\}$;
- $\mathcal{E} = \{m\}$;
- the rules of R are designed as follows:

$$\begin{aligned}
r_{1,i} &\equiv [p_{i,1}|\,]_1^0 \to [p_{i,1}^+|\,]_1^0[p_{i,1}^-|\,]_1^0, 1 \le i \le n.\\
r_{2,i,j} &\equiv \{[p_{i,j}^+|d_{i,j}]_1^0 \to [p_{i,j+1}^+|c_j]_2 1^0,\\
&\qquad [p_{i,j}^+|d'_{i,j}]_1^0 \to [p_{i,j+1}^+|d'_{i,j}]_1^0,\\
&\qquad [p_{i,j}^+|d''_{i,j}]_1^0 \to [p_{i,j+1}^+|d''_{i,j}]_1^0 | 1 \le i \le n, 1 \le j \le m-1\}.\\
r_{3,i,j} &\equiv \{[p_{i,j}^-|d_{i,j}]_1^0 \to [p_{i,j+1}^-|d_{i,j}]_1^0,\\
&\qquad [p_{i,j}^-|d'_{i,j}]_1^0 \to [p_{i,j+1}^-|c_j]_1^0,\\
&\qquad [p_{i,j}^-|d''_{i,j}]_1^0 \to [p_{i,j+1}^-|d''_{i,j}]_1^0 | 1 \le i \le n, 1 \le j \le m-1\}.\\
r_{4,i} &\equiv \{[p_{i,m}^+|d_{i,m}]_1^0 \to [p_{i+1,1}|c_m]_1^0,\\
&\qquad [p_{i,m}^+|d'_{i,m}]_1^0 \to [p_{i+1,1}|d'_{i,m}]_1^0,\\
&\qquad [p_{i,m}^+|d''_{i,m}]_1^0 \to [p_{i+1,1}|d''_{i,m}]_1^0 | 1 \le i \le n\}.\\
r_{5,i} &\equiv \{[p_{i,m}^-|d_{i,m}]_1^0 \to [p_{i+1,1}|d_{i,m}]_1^0,\\
&\qquad [p_{i,m}^-|d'_{i,m}]_1^0 \to [p_{i+1,1}|c_m]_1^0,\\
&\qquad [p_{i,m}^-|d''_{i,m}]_1^0 \to [p_{i+1,1}|d''_{i,m}]_1^0 | 1 \le i \le n\}.\\
r_{6,j} &\equiv [p_{n+1,j}|c_j]_1^0 \to [p_{n+1,j+1}|c_j]_1^0, 1 \le j \le m.\\
r_7 &\equiv m[p_{n+1,m+1}|\,]_1^0 \to [p_{n+1,m+1}|e]_1^0.\\
r_8 &\equiv [p_{n+1,m+1}|e]_1^0 \to e[p_{n+1,m+1}|\,]_1^0.\\
r_{9,i} &\equiv [\beta_i|m_i]_1^0 \to [\beta_{i+1}|m_{i+1}]_1^0, 1 \le i \le nm + m + 2.\\
r_{10} &\equiv e[\beta_{nm+m+3}|\,]_1^0 \to [\beta_{nm+m+3}|yes]_1^0.\\
r_{11} &\equiv m[\beta_{nm+m+3}|\,]_1^0 \to [\beta_{nm+m+3}|no]_1^0.\\
r_{12} &\equiv [\beta_{nm+m+3}|yes]_1^0 \to yes[\beta_{nm+m+3}|\,]_1^0.\\
r_{13} &\equiv [\beta_{nm+m+3}|no]_1^0 \to no[\beta_{nm+m+3}|\,]_1^0.
\end{aligned}$$

Generation stage.

The system produces 2^n truth assignments for n variables in the generation stage, $\Pi(\langle n, m\rangle)$ checks each truth assignment to decide whether all clauses are true; the generation stage takes $nm + n$ steps, which contain n iterations (each iteration has $m + 1$ steps). Next, we analyze each iteration as follows.

In the first step of the i-th iteration, the system employs rule $r_{1,i}$ for each membrane 1 divided into two membranes without changing its label, and each protein $p_{i,1}$ is replaced by $p_{i,1}^+$ and $p_{i,1}^-$, respectively.

In the next $m - 1$ steps of the i-th iteration, in membrane 1 containing protein $p_{i,j}^+$ (resp., protein $p_{i,j}^-$), one of the rules in $r_{2,i,j}$ (resp., $r_{3,i,j}$) can be employed. If membrane 1 contains protein $p_{i,j}^+$ and object $d_{i,j}$ (resp., $d'_{i,j}$ or $d''_{i,j}$), with the influence of protein $p_{i,1}^+$, object $d_{i,j}$ is modified to c_j, and the protein is modified to $p_{i,j+1}^+$. If membrane 1 contains protein

$p^{-}_{i,j}$ and object $d'_{i,j}$ (resp., $d_{i,j}$ or $d''_{i,j}$), with the influence of protein $p^{-}_{i,j}$, object $d'_{i,j}$ is modified to c_j, and the protein is modified to $p^{-}_{i,j+1}$.

In step $m+1$ of the i-th iteration, if membrane 1 contains protein $p^{+}_{i,m}$ and object $d_{i,m}$ (resp., $d'_{i,m}$ or $d''_{i,m}$), with the influence of protein $p^{+}_{i,m}$, one of the rules in $r_{4,i,j}$ is used, object $d_{i,m}$ is modified to c_m, and the protein is modified to $p_{i+1,1}$. If membrane 1 contains protein $p^{-}_{i,m}$ and object $d'_{i,m}$ (resp., $d_{i,m}$ or $d''_{i,m}$), with the influence of protein $p^{-}_{i,m}$, one of the rules in $r_{5,i,j}$ is used, object $d'_{i,m}$ is modified to c_m, and the protein is modified to $p_{i+1,1}$.

Checking stage.

The checking stage takes m steps. In the checking stage, each assignment in each membrane 1 is checked for whether all clauses are satisfied. More specifically, when at least one membrane 1 includes all objects $c_1, \ldots, c_m$, then it indicates that all clauses in φ are satisfied for that assignment in that membrane; consequently, the computation result of φ is TRUE; when no membrane 1 includes all objects $c_1, \ldots, c_m$, then it indicates that in each membrane 1, the Boolean formula φ cannot be satisfied; consequently, the computation result of φ is FALSE.

At the j-th ($1 \leq j \leq m$) step of the checking stage, rule $r_{6,j}$ is employed, and protein $p_{n+1,j}$ is modified to $p_{n+1,j+1}$. Note that when object c_j ($1 \leq j \leq m$) does not appear in a membrane 1, rule $r_{6,j}$ cannot be employed in this step.

Output stage.

The system sends the correct result to the environment in this stage. Rules $r_{9,i}$ are used for counting the computation steps except when rules $r_{1,i}$ are used.

If there exists a membrane 1 that has protein $p_{n+1,m+1}$ (φ is satisfied), at step $nm+n+m+1$, rule r_7 is employed, object m is sent into membrane 1, and it is modified to e. At step $nm+n+m+2$, rule r_8 is employed, and object e is sent to the environment. At step $nm+n+m+3$, the system employs rule r_{10}, and object e is sent to membrane 1, which is modified to `yes` in this process. At step $nm+n+m+4$, rule r_{12} is applied, object `yes` is sent to the environment, and the system stops. Consequently, the computation result of φ is positive.

If protein $p_{n+1,m+1}$ does not appear in any membrane 1 (φ is not satisfied), then at step $nm+n+m+1$ and step $nm+n+m+2$, only rules $r_{9,i}$ are applied. At step $nm+n+m+3$, rule r_{11} is applied, object m is sent to membrane 1, and it is modified to `no`. At step $nm+n+m+4$, rule r_{13} is applied, object `no` is sent to the environment, and the system stops. Consequently, the computation result of φ is negative.

To design the PAM P system, the necessary resources are counted as follows: (1) the total number of objects is $4nm+n+2m+7$; (2) the total number of proteins is $4nm+3n+4m+6$; (3) the initial number of membranes is 1; (3) the total number of objects at initial configuration is $nm+1$; (4) the total number of rules used in the system is $7nm+n+2m+8$; and (5) the length of a rule (not counting proteins) is no more than 2. Consequently, a PAM P system $\Pi(\langle n, m\rangle)$ is constructible in polynomial time by a Turing machine.

Object `yes` (resp., `no`) as the result of the PAM P system $\Pi(\langle n, m\rangle)$ is transferred to the environment in the last step $nm+n+m+4$. Consequently, the PAM P system $\Pi(\langle n, m\rangle)$ is polynomially bounded concerning the sizes of clauses and variants for φ.

Consequently, the family of PAM P systems $\mathbf{\Pi}$ offers a uniform result of a Boolean satisfiability problem. □

5. Conclusions

In this paper, a novel class of P systems whose name is P systems with proteins on active membranes is introduced. The computational power and computational efficiency of PAM P systems are discussed. We showed that PAM P systems achieve Turing universality when the systems use two membranes, one protein and one polarization. In addition, PAM P systems with the help of membrane division rules are able to make the `SAT` problem solvable.

We showed conclusions that PAM P systems (using rules of types (3) and (4)) have Turing universality and (using rules of types (2), (3), (4) and (6)) can give the result of the

SAT problem. A clear remaining problem is discussing whether the numbers of the types of rules of PAM P systems are optimal in order to achieve universality and to give the result of the SAT problem.

Many variants of P systems are able to make the QSAT problem solvable [4,5,32]. A clear remaining problem is how to make the QSAT problem solvable in PAM P systems.

Author Contributions: Investigation, Y.L. and B.S.; Writing—original draft, C.H. All authors have read and agreed to the published version of the manuscript.

Funding: The work was supported by the National Natural Science Foundation of China (62272151, 61972138, 62122025, 61872309, 62102140); Hunan Provincial Natural Science Foundation of China (2022JJ20016, 2022RC1099); the Key Research and Development Program of Changsha (kq2004016); and the Open Research Projects of Zhejiang Lab (2021RD0AB02).

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Not applicable

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Păun, G. Computing with membranes. *J. Comput. Syst. Sci.* **2000**, *61*, 108–143. [CrossRef]
2. Song, B.; Li, K.; Orellana-Martín, D.; Pérez-Jiménez, M.J.; PéRez-Hurtado, I. A survey of nature-inspired computing: Membrane computing. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–31. [CrossRef]
3. Song, B.; Li, K.; Orellana-Martín, D.; Valencia-Cabrera, L.; Pérez-Jiménez, M.J. Cell-like P systems with evolutional symport/antiport rules and membrane creation. *Inf. Comput.* **2020**, *275*, 104542. [CrossRef]
4. Pan, L.; Orellana-Martín, D.; Song, B.; Pérez-Jiménez, M.J. Cell-like P systems with polarizations and minimal rules. *Theor. Comput. Sci.* **2020**, *816*, 1–18. [CrossRef]
5. Gazdag, Z.; Hajagos, K.; Iván, S. On the power of P systems with active membranes using weak non-elementary membrane division. *J. Membr. Comput.* **2021**, *3*, 258–269. [CrossRef]
6. Bernardini, F.; Gheorghe, M. Cell communication in tissue P systems: Universality results. *Soft Comput.* **2005**, *9*, 640–649. [CrossRef]
7. Martín-Vide, C.; Păun, G.; Pazos, J.; Rodríguez-Patón, A. Tissue P systems. *Theor. Comput. Sci.* **2003**, *296*, 295–326. [CrossRef]
8. Freund, R.; Păun, G.; Pérez-Jiménez, M.J. Tissue P systems with channel states. *Theor. Comput. Sci.* **2005**, *330*, 101–116. [CrossRef]
9. Song, B.; Huang, S.; Zeng, X. The computational power of monodirectional tissue P systems with symport rules. *Inf. Comput.* **2021**, *281*, 104751. [CrossRef]
10. Song, B.; Li, K.; Zeng, X. Monodirectional evolutional symport tissue P systems with promoters and cell division. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 332–342. [CrossRef]
11. Song, B.; Pan, L. Rule synchronization for tissue P systems. *Inf. Comput.* **2021**, *281*, 104685. [CrossRef]
12. Păun, A.; Păun, G. Small universal spiking neural P systems. *BioSystems* **2007**, *90*, 48–60. [CrossRef] [PubMed]
13. Ionescu, M.; Păun, G.; Yokomori, T. Spiking neural P systems. *Fundam. Inform.* **2006**, *71*, 279–308.
14. Ibarra, O.H.; Woodworth, S. Characterizations of some restricted spiking neural P systems. In *International Workshop on Membrane Computing*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 424–442. [CrossRef]
15. Peng, H.; Wang, J.; Pérez-Jiménez, M.J.; Wang, H.; Shao, J.; Wang, T. Fuzzy reasoning spiking neural P system for fault diagnosis. *Inf. Sci.* **2013**, *235*, 106–116. [CrossRef]
16. Zhu, M.; Yang, Q.; Dong, J.; Zhang, G.; Gou, X.; Rong, H.; Paul, P.; Neri, F. An adaptive optimization spiking neural P system for binary problems. *Int. J. Neural Syst.* **2021**, *31*, 2050054. [CrossRef]
17. Jiang, Y.; Su, Y.; Luo, F. An improved universal spiking neural P system with generalized use of rules. *J. Membr. Comput.* **2019**, *1*, 270–278. [CrossRef]
18. Song, T.; Pan, L.; Păun, G. Asynchronous spiking neural P systems with local synchronization. *Inf. Sci.* **2013**, *219*, 197–207. [CrossRef]
19. Pan, L.; Păun, G.; Zhang, G.; Neri, F. Spiking neural P systems with communication on request. *Int. J. Neural Syst.* **2017**, *27*, 1750042. [CrossRef]
20. Wu, T.; Neri, F.; Pan, L. On the Tuning of the Computation Capability of Spiking Neural Membrane Systems with Communication on Request. *Int. J. Neural Syst.* **2022**, *32*, 2250037. [CrossRef]
21. Bäck, T.; Kok, J.N.; Rozenberg, G. *Handbook of Natural Computing*; Springer: Berlin/Heidelberg, Germany; New York, NY, USA, 2012. [CrossRef]
22. Alhazov, A.; Freund, R.; Verlan, S. P systems working in maximal variants of the set derivation mode. In *International Conference on Membrane Computing*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 83–102. [CrossRef]

23. Liu, L.; Jiang, K. Universality of spiking neural P systems with polarizations working in sequential mode induced by maximum spike number. *J. Membr. Comput.* **2021**, *4*, 56–67. [CrossRef]
24. Aman, B. On the efficiency of synchronized P systems. *J. Membr. Comput.* **2022**, *4*, 1–10. [CrossRef]
25. Ceterchi, R.; Orellana-Martín, D.; Zhang, G. Division rules for tissue P systems inspired by space filling curves. *J. Membr. Comput.* **2021**, *3*, 105–115. [CrossRef]
26. Păun, G. *P Systems with Active Membranes: Attacking NP Complete Problems*; Technical Report; Department of Computer Science, The University of Auckland: Auckland, New Zealand, 1999.
27. Pan, L.; Ishdorj, T.O. P systems with active membranes and separation rules. In Proceedings of the Second Brainstorming Week on Membrane Computing, 325–341. Sevilla, ETS de Ingeniería Informática, Sevilla, Spain, 2–7 February 2004 .
28. Alhazov, A.; Pan, L.; Păun, G. Trading polarizations for labels in P systems with active membranes. *Acta Inform.* **2004**, *41*, 111–144. [CrossRef]
29. Song, T.; Macías-Ramos, L.F.; Pan, L.; Pérez-Jiménez, M.J. Time-free solution to SAT problem using P systems with active membranes. *Theor. Comput. Sci.* **2014**, *529*, 61–68. [CrossRef]
30. Zandron, C.; Ferretti, C.; Mauri, G. Solving NP-complete problems using P systems with active membranes. In *Unconventional Models of Computation, UMC'2K*; Springer: New York, NY, USA, 2001; pp. 289–301. [CrossRef]
31. Alhazov, A.; Martín-Vide, C.; Pan, L. Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes. *Fundam. Inform.* **2003**, *58*, 67–77.
32. Song, B.; Pérez-Jiménez, M.J.; Pan, L. An efficient time-free solution to QSAT problem using P systems with proteins on membranes. *Inf. Comput.* **2017**, *256*, 287–299. [CrossRef]
33. Wu, T.; Pan, L.; Yu, Q.; Tan, K.C. Numerical Spiking Neural P Systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2443–2457. [CrossRef]
34. Păun, A.; Popa, B. P systems with proteins on membranes. *Fundam. Inform.* **2006**, *72*, 467–483.
35. Păun, A.; Rodríguez-Patón, A. On flip-flop membrane systems with proteins. *Lect. Notes Comput. Sci.* **2007**, *4860*, 414–427. [CrossRef]
36. Păun, A.; Popa, B. P systems with proteins on membranes and membrane division. In *International Conference on Developments in Language Theory*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 292–303. [CrossRef]
37. Song, B.; Luo, X.; Valencia-Cabrera, L.; Zeng, X. The computational power of cell-like P systems with one protein on membrane. *J. Membr. Comput.* **2020**, *2*, 332–340. [CrossRef]
38. Păun, G. Membrane computing. *Scholarpedia* **2010**, *5*, 9259. [CrossRef]
39. Rozenberg, G.; Salomaa, A. (Eds.) *Handbook of Formal Languages: Volume 3 Beyond Words*; Springer Science & Business Media: New York, NY, USA, 2012.
40. Clarke, D. Computation: Finite and Infinite Machines. *SIAM Rev.* **1969**, *11*, 99–100. [CrossRef]
41. Pérez-Jiménez, M.J. An approach to computational complexity in membrane computing. In *International Workshop on Membrane Computing*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 85–109. [CrossRef]
42. Hartmanis, J. Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Rev.* **1982**, *24*, 90. [CrossRef]