On accepting conditions in P systems with active membranes

(extended abstract)

Zsolt Gazdag, Károly Hajagos

Institute of Informatics, University of Szeged,
Árpád tér 2, H-6720 Szeged, Hungary
{gazdag,hajagos}inf.u-szeged.hu

Abstract. We consider a variant of P systems called elimination P systems. These are polarizationless P systems with active membranes with no dissolution and non-elementary membrane division rules, extended with rules of the form $[ab \to \varepsilon]$, where a, b are objects and ε represents the empty word. We investigate the computational power of two types of elimination P systems designed to solve decision problems. In the first type, as usual, an accepting computation must output a single *yes* object, and a rejecting computation must output one *no* object, both occurring precisely in the final step of the computation. In the second type, an accepting computation should produce one or more *yes* objects, whereas a rejecting computation should not produce any *yes* objects. We show that while the first model can solve in polynomial time only problems in **NL**, the second model is able to solve **NP**-complete problems.

Introduction. In the field of complexity theory, a decider for a language L is typically a Turing machine M that halts on each input and produces an output that is either *accepted* or *rejected*. The output for a word w is *accepted* if and only if $w \in L$. M may determine the output in several equivalent ways, such as halting in a designated *accept* or *reject* state, halting in either a *final* or *non-final* state, or by writing the output on the output tape. In membrane computing, the approach to solving decision problems mirrors that of complexity theory: the P system eventually halts, accepting or rejecting the input multiset. Typically, P systems indicate their decision by sending out a specific symbol, *yes* or *no*, to the environment. The initial formalization of these conditions appeared in [11], where the concept of *accepting P systems* was introduced. A P system Π is an accepting P system if it has only halting computations and its object alphabet includes symbols *yes* and *no*. Furthermore, for any computation of Π, it is required that the object *yes* or the object *no* is sent out to the environment, but not both. This definition was further refined in the concept of *recognizer* P systems, which stipulates that exactly one *yes* or *no* must be sent out to the environment (or to a designated *output membrane*) exclusively in the last step of the computation (see, e.g.,[10]).

P systems with active membranes were introduced in [8]. These P systems allow the membranes to exhibit polarizations and enable the division of both

elementary and non-elementary membranes. It quickly became apparent that, when combined with maximal parallelism, these properties enable this variant to address problems that are **NP**-complete or even **PSPACE**-complete. In fact, the class of problems that can be solved in polynomial time by P systems with active membranes is **PSPACE** [13]. Moreover, without non-elementary membrane division, these P systems can solve in polynomial-time exactly the problems in $\mathbf{P}^{\#\mathbf{P}}$, which is the class of problems solvable by deterministic Turing machines in polynomial time with polynomial-time counting oracles [4]. For a recent survey on the complexity of various classes of P systems with active membranes, see [12].

It is an interesting open question whether P systems with active membranes can efficiently solve **NP**-complete problems without utilizing membrane polarization. In [9], Păun conjectured that the answer is negative. While this conjecture remains unproven, there are some partial results (see, e.g., [5] and the references therein). For instance, in [3], it was demonstrated, using the concept of a dependency graph, that polarizationless P systems with no dissolution rules can solve exactly the problems in **P**. For such a P system, a dependency graph is a directed graph where edges represent dependencies between objects on the left and right sides of the P system's rules. Then one can verify whether *yes* appears in the output membrane by simply checking if *yes* can be reached from the relevant nodes in the dependency graph.

As mentioned in [6], the polynomial-time construction of the analyzed P systems is the reason for the **P** lower bound in the characterization of **P** discussed in [3]. In fact, [7] discovered that FAC^0-uniform families of recognizer polarizationless P systems with no dissolution rules cannot even solve the Parity problem, which entails determining whether a binary string has an odd number of 1s (FAC^0 is the class of functions computed by uniform constant depth polynomial size Boolean circuits). To better understand the effect of the definition of accepting conditions in recognizer P systems, a more general variant, called *acknowledger P systems*, was defined and examined in [7]. These are P systems in which all computations terminate and one or more copies of the distinguished object *yes* may or may not appear in the output membrane of the system. Moreover, these P systems have no rules applicable to *yes*. Recall that in recognizer P systems exactly one *no* or *yes* must be produced in the output membrane, but not both, and only in the last step of the computation. In [7] it was found that acknowledger polarizationless P systems without dissolution rules decide exactly those languages that are FAC^0 disjunctive truth-table reducible to the unary languages in **NL**. However, this result has not been established for recognizer polarizationless P systems without dissolution rules in [7]. Using reasonably tight uniformity conditions has become standard in membrane computing when P systems solve problems in **P** (see, for example, [2] and the references therein). However, the power of acknowledger P systems has received less attention when considering P systems solving problems beyond **P**.

In this extended abstract, we demonstrate that a variant of polarizationless P systems can efficiently solve **NP**-complete problems when considering uni-

form families of acknowledger P systems, whereas they are limited to solving problems in **NL** when considering recognizer P systems. In this study, we permit acknowledger P systems to include rules that can be applied to the output object *yes*.

Our contribution. In [1], the study focused on P systems with antimatter. These are polarizationless P systems with active membranes, where each object a has an antiparticle counterpart represented as $\bar{a}$. Additionally, the system can use annihilation rules like $[a\bar{a} \to \varepsilon]$. When a and $\bar{a}$ appear together in the same membrane, they are both disappear without producing anything. These rules are applied alongside other rules based on the principle of maximal parallelism, except that annihilation rules have priority over all other types of rules. According to [1], recognizer P systems with antimatter that do not use non-elementary membrane division and dissolution rules can efficiently solve the SAT problem. However, it was also found that these systems can only solve problems in **P** if annihilation rules do not have priority over other types of rules.

In this study, we explore a minor extension of these P systems, termed *elimination P systems*. An elimination P system is a variant of polarizationless P system with active membranes that include only evolution rules, division rules for elementary membranes, in- and out-communication rules, and elimination rules of the form $[ab \to \varepsilon]$, where a and b can be any objects (unlike in P systems with antimatter, where b must be the antiparticle of a). The effect of applying elimination rules is identical to that of applying annihilation rules. Moreover, elimination rules have no priority over the other types of rules.

To investigate the computation power of elimination P systems, we will use logarithmic-space uniformity (see, e.g., [2]). Consider a *recognizer* P system Π. A computation of Π that halts with *yes* (resp. *no*) in the output membrane is called *accepting* (resp. *rejecting*). Likewise, consider an *acknowledger* P system Π. A computation of Π that halts with at least one copy of *yes* in the output membrane is called *accepting*, whereas a computation that halts with no *yes* in the output membrane is called *rejecting*. Notice that both recognizer and acknowledger P systems exclusively have halting computations, and each computation is classified as either accepting or rejecting.

Let $\mathbf{\Pi} = \{\Pi(n) \mid n \in \mathbb{N}\}$ be an **L**-uniform family of recognizer or acknowledger P systems. We say that $\mathbf{\Pi}$ *solves* a decision problem R in polynomial time if (i) there is a deterministic Turing machine using logarithmic space computing the encoding function *cod* that transforms instances of R into multisets of objects, (ii) there exists an integer $k \in \mathbb{N}$ such that for every instance x of R with size n, each computation $\mathfrak{C}$ of $\Pi(n)$ starting with $cod(x)$ in its input membrane halts in at most n^k steps, and (iii) $\mathfrak{C}$ is accepting if and only if x is a positive instance of R.

Applying a reasoning similar to that used in the proof of Theorem 2 from [1], we can derive the following result.

Proposition 1. *The class of problems that can be solved in polynomial time by* ***L****-uniform families of recognizer elimination P systems is contained within*

NL*, the set of problems solvable by nondeterministic Turing machines using logarithmic space.*

Next, we discuss that acknowledger elimination P systems can solve **NP**-complete problems efficiently. It is well known that **NP** is the class of problems verifiable in polynomial time. Specifically, a problem L is verifiable in polynomial time if for each positive instance I of L, there exists a polynomial-size proof (or certificate) that I is indeed a positive instance. Consider the SAT problem, which involves determining whether a Boolean formula in conjunctive normal form is satisfiable. A certificate which proves that a formula φ is satisfiable is a truth assignment that satisfies φ. Clearly, it is decidable in polynomial time whether a truth assignment satisfies φ or not. According to this, deciding if a formula φ is satisfiable with a recognizer P system with active membranes Π usually involves the following main steps. Given an input multiset $cod(\varphi)$ that encodes φ, Π (i) first generates all possible truth assignments for the variables of φ, placing each assignment in a separate membrane, then (ii) checks within each membrane if it contains a satisfying truth assignment, and finally (iii) produces *yes* in the output membrane at the final step of the computation if there is at least one satisfying truth assignment, and produces *no* otherwise. We will describe that elimination P systems, whether recognizer or acknowledger, are capable of efficiently executing Steps (i) and (ii) above. According to Proposition 1, recognizer elimination P systems are unable to perform Step (iii). Conversely, acknowledger P systems, as we will demonstrate, are capable of doing so, which brings us to the following conclusion.

Theorem 1. SAT *can be solved in polynomial time by* ***L****-uniform families of acknowledger elimination P systems.*

We briefly outline a proof of Theorem 1. Let φ be a formula with n variables and m clauses. We encode φ as a multiset using a method commonly employed in membrane computing:

$$cod(\varphi) = \{v_{i,j} \mid x_i \in \mathcal{C}_j\} \cup \{\overline{v_{i,j}} \mid \neg x_i \in \mathcal{C}_j\}.$$

We present an elimination P system Π designed to decide the satisfiability of φ. The initial membrane configuration of Π consists of an outer skin membrane labeled by s and an inner membrane labeled by 1. The membrane with label 1 serves as the input membrane, while the skin membrane acts as the output membrane. Beginning with $cod(\varphi)$ in the input membrane, Π initially generates 2^n membranes with label 1, each corresponding to a truth assignment of the variables in φ. Meanwhile, Π eliminates objects from $cod(\varphi)$ that represent *false* literals under the corresponding truth assignment of φ by applying specific elimination rules. In the subsequent step, objects representing the clauses of φ are introduced into each membrane with label 1. Then, using elimination rules acting on objects representing the corresponding literals and clauses, Π removes objects that represent *true* clauses under the truth assignment represented by the corresponding membrane with label 1. At this stage of the computation,

a membrane M with label 1 contains an object representing a clause of φ if and only if φ is *false* under the truth assignment represented by M. Through carefully designed elimination rules, it is ensured in the following steps that a membrane M with label 1 contains one object d if φ is not satisfied and no object d if φ is satisfied by the truth assignment corresponding to M. In the next step, membranes with label 1 that contain an object d release this object to the skin membrane. Simultaneously, in the skin membrane, 2^n *yes* objects are generated, and in the final step of Π, an elimination rule $[d\, yes \to \varepsilon]_s$ is applied to each d that occurs in the skin. Consequently, at the end of the computation, the skin membrane contains at least one *yes* if and only if φ is satisfiable. Therefore, the described computation is accepting if and only if φ is satisfiable. □

It is not hard to see that when Π halts, the number of *yes* objects in the skin membrane is equal to the number of truth assignments satisfying φ. With elimination rules, it would be easy to remove 2^{n-1} of these objects. Thus, it would be easy to construct a P system whose skin membrane has at least one *yes* after the final step if and only if more than half of the truth assignments of φ are satisfying. Consequently, acknowledger elimination P systems are able to solve the **PP**-complete Majority-Sat problem, too.

Conclusions. In this extended abstract, we introduced a variant of P systems that can efficiently solve problems within **NL** when using recognizer P systems to decide problems. Conversely, they can address **PP**-complete problems when employing the more general acknowledger P systems. This highlights the importance of defining the accepting conditions in specific classes of P systems.

References

1. Díaz-Pernil, D., Alhazov, A., Freund, R, Gutiérrez-Naranjo, M. A., Leporati, A.: Recognizer P Systems with Antimatter. Romanian Journal of Information Science and Technology 18(3), 201–217, 2015.
2. Gazdag, Z., Kolonits, G.: Remarks on the computational power of some restricted variants of P systems with active membranes. In: Leporati, A., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing, 17th International Conference, LNCS vol. 10105, 209-–232, 2017.
3. Gutierrez-Naranjo, M.A., Perez-Jimenez, 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, 224–240, 2006.
4. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Simulating elementary active membranes, with an application to the P conjecture. In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron (Eds.) Membrane Computing – 15th International Conference, CMC15, LNCS vol. 8961, 284–299, 2014.
5. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Solving a special case of the P conjecture using dependency graphs with dissolution. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing: 18th International Conference, LNCS vol. 10725, 196-213, 2017.

6. Murphy, N., Woods, D.: The computational power of membrane systems under tight uniformity conditions. Natural Computing **10**(1) 613–632, 2011.
7. Murphy, N., Woods, D.: Uniformity is weaker than semi-uniformity for some membrane systems. Fundam. Inf. **134**(1-2) 129–152, 2014.
8. Păun, Gh.: P Systems with Active Membranes: Attacking NP-Complete Problems. Journal of Automata, Languages and Combinatorics **6**(1) 75–90, 2001.
9. Păun, Gh.: Further twenty six open problems in membrane computing. In: Third Brainstorming Week on Membrane Computing. Fénix Editora, Sevilla 249–262, 2005.
10. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England, 2010.
11. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. Natural Computing **2**(3) 265–285, 2003.
12. Sosík, P.: P systems attacking hard problems beyond NP: a survey. J. Membr. Comput. 1, 198–208, 2019.
13. Sosík, P., Rodríguez-Patón, A.: Membrane computing and complexity theory: A characterization of PSPACE. Journal of Computer and System Sciences **73**(1), 137–152, 2007.