On The Semantics of Annihilation Rules in Membrane Computing

Daniel Díaz-Pernil[1], Rudolf Freund[2],
Miguel A. Gutiérrez-Naranjo[3], Alberto Leporati[4]

[1]Research Group on Computational Topology and Applied Mathematics
Department of Applied Mathematics - University of Sevilla, Spain
sbdani@us.es

[2]Faculty of Informatics
Vienna University of Technology, Austria
rudi@emcc.at

[3]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, Spain
magutier@us.es

[4]Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca, Italy
alberto.leporati@unimib.it

Summary. It is well known that polarizationless recognizer P systems with active membranes, without dissolution, with division of elementary and non-elementary membranes, with antimatter and matter/antimatter annihilation rules can solve all problems in **NP** when the annihilation rules have (weak) priority over all the other rules. Until now, it was an open problem whether these systems can still solve all **NP** problems if the priority of the matter/antimatter annihilation rules is removed.

In this paper we provide a negative answer to this question: we prove that the class of problems solvable by this model of P systems without priority of the matter/antimatter annihilation rules is exactly **P**. To the best of our knowledge, this is the first paper in the literature of P systems where the semantics of applying the rules constitutes a frontier of tractability.

1 Introduction

The concept of antimatter was first introduced in the framework of membrane computing as a control tool for the flow of spikes in spiking neural P systems [6, 5, 10, 11]). In this context, when one spike and one anti-spike appear in the same neuron, the annihilation occurs and both, spike and anti-spike, disappear. The concept of antimatter and matter/antimatter annihilation rules later was

adapted to other contexts in membrane computing, and currently it is an active research area [1, 2, 3].

In [3], the authors show that antimatter and matter/antimatter annihilation rules are a frontier of tractability. The starting point is a well-known result in the complexity theory of membrane computing: the decision problems which can be solved by polarizationless recognizer P systems with active membranes, without dissolution and with division of elementary and non-elementary membranes (denoted by $\mathcal{AM}^0_{-d,+ne}$) are exactly those in the complexity class **P** (see [4], Th. 2). The main result in [3] is that systems from $\mathcal{AM}^0_{-d,+ne}$ endowed with antimatter and matter/antimatter annihilation rules (denoted by $\mathcal{AM}^0_{-d,+ne,+ant}$) can solve all problems in **NP** and, hence, annihilation rules constitute a frontier of tractability.

In this paper, we revisit the question of determining the computational complexity of the problems which can be solved by P systems with the matter/antimatter annihilation rules not having priority over all the other rules. As previously pointed out (see [3, 9]), the solution presented in [3] to an **NP**-complete problem, namely Subset Sum, uses this weak priority of the annihilation rules, and until now it has been an open problem if the model $\mathcal{AM}^0_{-d,+ne,+ant}$ is still capable to solve **NP**-complete problems *without this priority*. In this paper we show that the answer to this open question is negative. We prove that the complexity class of decision problems solvable by $\mathcal{AM}^0_{-d,+ne,+ant}$ systems is exactly equal to **P** if the priority relation is removed from the semantics for the annihilation rules.

In this way, we propose a new kind of frontier of tractability. Up to now, these frontiers were based on *syntactic ingredients* of the P systems, that is, the type of rules and not the *way* in which such rules are applied. In this paper, the frontier of tractability is based on the *semantics* of the P system, i.e., on the way the rules are applied.

The paper is organised as follows. First, we recall some concepts about recognizer P systems, antimatter, matter/antimatter annihilation rules and the model $\mathcal{AM}^0_{-d,+ne,+ant}$. Next, we prove our main result of computational complexity. The paper ends with some final considerations.

2 Recognizer P Systems

First of all, we recall the main notions related to recognizer P systems and computational complexity in membrane computing. For a detailed description see, for example, [7, 8].

The main *syntactic* ingredients of a cell–like P system are the *membrane structure*, the *multisets*, and the *evolution rules*. A *membrane structure* consists of several membranes arranged hierarchically inside a main membrane, called the *skin*. Each membrane identifies a region inside the system. When a membrane has no membrane inside, it is called *elementary*. The objects are instances of symbols from a finite alphabet, and *multisets of objects* are placed in the regions determined by

the membrane structure. The objects can evolve according to given *evolution rules*, associated with the regions.

The *semantics* of cell–like P systems is defined through a non-deterministic and synchronous model. A *configuration* of a cell-like P system consists of a membrane structure and a sequence of multisets of objects, each associated with one region of the structure. At the beginning of the computation, the system is in the *initial configuration*, which possibly comprises an *input multiset*. In each time step the system transforms its current configuration into another configuration by applying the evolution rules to the objects placed inside the regions of the system, in a non-deterministic and maximally parallel manner (the precise semantics will be described later). In this way, we get *transitions* from one configuration of the system to the next one. A *computation* of the system is a (finite or infinite) sequence of configurations such that each configuration –except the initial one– is obtained from the previous one by a transition. A computation which reaches a configuration where no more rules can be applied to the existing objects and membranes, is called a *halting computation*. The result of a halting computation is usually defined by the multiset associated with a specific output membrane (or the environment) in the final configuration.

In this paper we deal with *recognizer* P systems, where all computations halt and exactly one of the distinguished objects *yes* and *no* is sent to the environment, and only in the last step of any computation, in order to signal acceptance or rejection, respectively. All recognizer P systems considered in this paper are *confluent*, meaning that if computations start from the same initial configuration then either all are accepting or all are rejecting.

Recognizer P systems can thus be used to recognize formal languages (equivalently, solve decision problems). Let us recall that a decision problem X is a pair (I_X, θ_X) where I_X is a language over a finite alphabet and θ_X is a predicate (a total Boolean function) over I_X. The elements of I_X are called *instances* of the problem, and those for which predicate θ_X is true (respectively false) are called *positive* (respectively *negative*) instances. A *polynomial encoding* of a decision problem X is a pair (cod, s) of functions over I_X, computable in polynomial time by a deterministic Turing machine, such that for each instance $u \in I_X$, $s(u)$ is a natural number representing the *size* of the instance and $cod(u)$ is a multiset representing an encoding of the instance. Polynomial encodings are stable under polynomial time reductions.

2.1 The Class $\mathcal{AM}^{\mathbf{0}}_{-d,+ne}$

A P system with active membranes without polarizations, without dissolution and with division of elementary and non-elementary membranes is a P system with Γ as the alphabet of symbols, with H as the finite set of labels for membranes, and where the rules are of the following forms:

(a_0) $[\, a \to u \,]_h$ for $h \in H$, $a \in \Gamma$, $u \in \Gamma^*$. This is an *object evolution rule*, associated with the membrane labelled with h. When the rule is applied, an object $a \in \Gamma$

inside that membrane is rewritten into the multiset $u \in \Gamma^*$. (Note that here and in the rest of the paper, we write $u \in \Gamma^*$ to indicate both the multiset u of objects from the alphabet Γ and one of the possible strings which represent it.)

(b_0) $a\,[\ ]_h \to [\,b\,]_h$ for $h \in H$, $a, b \in \Gamma$ *(send-in rules)*. An object from the region immediately outside a membrane labelled with h is sent into this membrane, possibly transformed into another object.

(c_0) $[\,a\,]_h \to b\,[\ ]_h$ for $h \in H$, $a, b \in \Gamma$ *(send-out rules)*. An object is sent out from the membrane labelled with h to the region immediately outside, possibly transformed into another object.

(d_0) $[\,a\,]_h \to [\,b\,]_h\,[\,c\,]_h$ for $h \in H$, $a, b, c \in \Gamma$ *(division rules for elementary membranes)*. An elementary membrane can be divided into two membranes with the same label; object a in the original membrane is rewritten to b (respectively to c) in the first (respectively second) generated membrane.

(e_0) $[\,[\,]_{h_1}[\,]_{h_2}\,]_{h_0} \to [\,[\,]_{h_1}\,]_{h_0}\ [\,[\,]_{h_2}\,]_{h_0}$, for $h_0, h_1, h_2 \in H$ *(division rules for non-elementary membranes)*. If the membrane with label h_0 contains other membranes than those with labels h_1, h_2, then such membranes and their contents are duplicated and placed in both new copies of the membrane h_0; all membranes and objects placed inside membranes h_1, h_2, as well as the objects from membrane h_0 placed outside membranes h_1 and h_2, are reproduced in the new copies of membrane h_0.

These rules are applied according to the following principles:

- All the rules are applied in parallel and in a maximal manner. In one step, one object of a membrane can be used by at most one rule (chosen in a non–deterministic way), and each membrane can be the subject of *at most one* rule of types (b_0), (c_0), (d_0), and (e_0).
- If at the same time a membrane labelled with h is divided by a rule of type (d_0) or (e_0) and there are objects in this membrane which evolve by means of rules of type (a_0), then we suppose that first the evolution rules of type (a_0) are used, and then the division is produced. Of course, this process takes only one step.
- The rules associated with membranes labelled with h are used for all copies of this membrane with label h.

The class of all polarizationless recognizer P systems with active membranes, without dissolution and with division of elementary and non-elementary membranes is denoted by $\mathcal{AM}^0_{-d,+ne}$.

2.2 Polynomial Complexity Classes in Recognizer P Systems

Let $\mathcal{R}$ be a class of recognizer P systems. A decision problem $X = (I_X, \theta_X)$ is solvable in a *semi-uniform* way and in polynomial time by a family of recognizer P systems $\mathbf{\Pi} = \{\Pi(w)\}_{w \in I_X}$ of type $\mathcal{R}$, denoted by $X \in \mathbf{PMC}^{\star}_{\mathcal{R}}$, if $\mathbf{\Pi}$ is *polynomially uniform* by Turing machines, that is, there exists a deterministic Turing machine

working in polynomial time which constructs the system $\Pi(w)$ from the instance $w \in I_X$, and $\mathbf{\Pi}$ is *polynomially bounded*, that is, there exists a polynomial function $p(n)$ such that for each $w \in I_X$, all computations of $\Pi(w)$ halt in at most $p(|w|)$ steps. It is said that $\mathbf{\Pi}$ is *sound* with regard to X if for each instance of the problem $w \in I_X$, if there exists an accepting computation of $\Pi(w)$ then $\theta_X(w)$ is true, and $\mathbf{\Pi}$ is *complete* with regard to X if for each instance of the problem $w \in I_X$, if $\theta_X(w)$ is true then every computation of $\Pi(w)$ is an accepting computation.

Let $\mathcal{R}$ be a class of recognizer P systems with a distinguished input membrane, and let $\mathbf{\Pi} = \{\Pi(n)\}_{n\in\mathbf{N}}$ be a family of recognizer P systems of type $\mathcal{R}$. A decision problem $X = (I_X, \theta_X)$ is solvable in a *uniform way* and polynomial time by $\mathbf{\Pi}$, denoted by $X \in \mathbf{PMC}_{\mathcal{R}}$, if $\mathbf{\Pi}$ is *polynomially uniform* by Turing machines, i.e., there exists a polynomial encoding[1] (cod, s) such that the family $\mathbf{\Pi}$ is *polynomially bounded* with regard to (X, cod, s); that is, there exists a polynomial function $p(n)$ such that for each $u \in I_X$, every computation of $\Pi(s(u))$ with input $cod(u)$ – denoted by $\Pi(s(u)) + cod(u)$, for short– is halting and, moreover, it performs at most $p(|u|)$ steps, and the family $\mathbf{\Pi}$ is *sound* and *complete* with regard to (X, cod, s). It is easy to see that the classes $\mathbf{PMC}^{\star}_{\mathcal{R}}$ and $\mathbf{PMC}_{\mathcal{R}}$ are closed under polynomial-time reduction and complement. Moreover, since uniformity can be considered to be a special case of semi-uniformity, the inclusion $\mathbf{PMC}_{\mathcal{R}} \subseteq \mathbf{PMC}^{\star}_{\mathcal{R}}$ holds.

According to these formal definitions, in [4] it is proved that the complexity class of decision problems solved by uniform or semi-uniform families of polarizationless recognizer P systems with active membranes, without dissolution and with division of elementary and non-elementary membranes, is exactly $\mathbf{P}$. With the standard notation, $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne}} = \mathbf{PMC}^{\star}_{\mathcal{AM}^0_{-d,+ne}}$.

2.3 Antimatter

Antimatter and matter/antimatter annihilation rules have been introduced in the framework of cell-like P systems in [2]. Given two objects a and b from the alphabet Γ in a membrane labeled by h, an annihilation rule of a and b is written as $[ab \to \lambda]_h$. The *meaning* of the rule follows the physical idea of annihilation: If a and b occur simultaneously in the same region with label h, then both are consumed (disappear) and nothing is produced (denoted by the empty string λ). Let us remark that both objects a and b are ordinary elements from Γ and they can trigger any other rule of type (a_0) to (d_0) described above, not only annihilation rules. Nonetheless, in order to improve the readability, if b annihilates the object a then b will be called the *antiparticle* of a and we will write $\overline{a}$ instead of b.

[1] See [7, 8] for the details. Informally, given an instance $u \in I_X$, $s(u)$ is a natural number which identifies a P system $\Pi(s(u))$ in the family. When fed with the multiset $cod(u)$ as input, this P system computes the value of predicate $\theta_X(u)$. In uniform families of P systems, the structure and definition of $\Pi(s(u))$ is the same for all instances $u \in I_X$ having the same size $s(u)$.

With respect to the *semantics*, let us recall that the rule $[a\overline{a} \to \lambda]_h$, provided that annihilation rules have priority over all other rules, must be applied as many times as possible in every membrane labeled by h, according to the maximal parallelism, i.e., if m copies of a and n copies of $\overline{a}$ occur simultaneously in a membrane of label h, with $m \geq n$ (respectively $m \leq n$), then the rule is applied n times (respectively m times), n (respectively m) copies of a and $\overline{a}$ are consumed and $m - n$ copies of a (respectively $n - m$ copies of $\overline{a}$) are not affected by this rule.

The key point in the use of the semantics of the annihilation rules in this paper is related to the priority of this type of rules with respect to the other types. In [3], according to the non-determinism, if an object a can trigger more than one rule of types (a_0) to (d_0), then one rule among the applicable ones is non-deterministically chosen. Nonetheless, if a and $\overline{a}$ occur simultaneously in the same membrane h and the annihilation rule $[a\overline{a} \to \lambda]_h$ is defined, then it is applied, regardless of other options. In this sense, any annihilation rule had priority over the other types of rules.

In this paper, we consider the case that the annihilation does not have priority over the other rules. If an object a can trigger more than one rule, then one rule among the applicable ones is non-deterministically chosen regardless of its type (obviously, for annihilation rules object $\overline{a}$ has also to occur in the same region).

Formally, a polarizationless P system with active membranes, without dissolution, with division of elementary and non-elementary membranes and with annihilation rules is a construct of the form $\Pi = (\Gamma, H, \mu, w_1, \ldots, w_m, R)$, where:

1. $m \geq 1$ is the initial degree of the system;
2. Γ is the alphabet of *objects*;
3. H is a finite set of *labels* for membranes;
4. μ is a *membrane structure* consisting of m membranes labelled in a one-to-one way with elements of H;
5. $w_1, \ldots, w_m$ are strings over Γ, describing the *multisets of objects* placed in the m regions of μ;
6. R is a finite set of *rules* of the types (a_0) to (e_0) described in Section 2.1, and the following type of rules:

   (f_0) $[\, a\overline{a} \to \lambda \,]_h$ for $h \in H$, $a, \overline{a} \in \Gamma$ *(annihilation rules)*. The pair of objects $a, \overline{a} \in \Gamma$ occurring simultaneously inside membrane h disappears.

As stated above, in this paper rules of type (f_0) have no priority over the other types of rules. If at the same time a membrane labelled with h is divided by a rule of type (d_0) or (e_0) and there are objects in this membrane which are chosen to be annihilated by means of rules of type (f_0), then we assume that first the annihilation is performed and then the division is produced. Of course, this process takes only one step.

By following the standard notation, in [3] the authors denote the class of polarizationless recognizer P systems with active membranes without dissolution, with division of elementary and non-elementary membranes, and with antimatter and matter/antimatter annihilation rules by $\mathcal{AM}^0_{-d,+ne,+ant}$. They do not include

any symbol in the name to specify the priority, because they assume it as being part of the model definition. In this paper, we will consider a class of P systems which uses the same model of P systems $\mathcal{AM}^0_{-d,+ne,+ant}$, but without priority for the application of the annihilation rules; in order to stress this difference, we will denote this class of P systems by $\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}$.

3 Removing Priority for Annihilation Rules

The main contribution of this paper is the proof of the following claim.

Theorem 1. $\mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}} = \mathbf{P}$

Proof. It is well known (e.g., see [4]) that $\mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne}} = \mathbf{PMC}^{\star}_{\mathcal{AM}^0_{-d,+ne}} = \mathbf{P}$. On the other hand, the following inclusion obviously holds:

$$\mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne}} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}},$$

therefore $\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}}$. Thus it only remains to prove that also the converse inclusion holds:

$$\mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}} \subseteq \mathbf{P}. \tag{1}$$

Since $\mathbf{PMC}^{\star}_{\mathcal{AM}^0_{-d,+ne}} = \mathbf{P}$, in order to prove (1) it suffices to prove that $\mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}} \subseteq \mathbf{PMC}^{\star}_{\mathcal{AM}^0_{-d,+ne}}$.

Hence, let $X \in \mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}}$ be a decision problem. By definition, there exist a polynomial encoding (cod, s) and a family of P systems $\{\Pi(i)\}_{i\in\mathbb{N}}$ in $\mathcal{AM}^0_{-d,+ne,+ant\_NoPri}$ such that for each instance u of the problem X:

- all computations of $\Pi(s(u)) + cod(u)$ halt;
- in all computations, the system sends out either one copy of the object *yes* or one copy of the object *no* (but not both), and only in the last step of computation.

Let us first provide an informal idea of the proof. Given an instance $u \in I_X$, we know that all computations of $\Pi(s(u)) + cod(u)$ halt, and that they all answer *yes* or all answer *no*. Let $\mathcal{C} = \{C_0, \ldots, C_n\}$ be one of these halting computations, and let us assume that the answer is *yes* (the other case is analogous). Then there exists an object a_1 and a rule $r_1 \equiv [\, a_1 \,]_{skin} \to yes\, [\,]_{skin}$ which has been applied in the last step of the computation. There are two possibilities: either object a_1 is in the skin membrane since the beginning of the computation, or there exists a rule r_2 which must have produced it inside or moved it into the skin membrane. Rule r_2 is triggered by the occurrence of an object a_2 in a membrane with label h_2. Obviously, r_2 cannot be an annihilation rule, since no object is produced by such rules, then rule r_2 must belong to types (a_0) to (d_0). Going back with the

reasoning, either a_2 appears in the membrane with label h_2 since the beginning of the computation, or it is produced or moved there by the application of a rule r_3, and so on.

Finally we have a chain

$$(yes, env) \xleftarrow{r_1} (a_1, skin) \xleftarrow{r_2} (a_2, h_2) \xleftarrow{r_3} \cdots \xleftarrow{r_k} (a_k, h_k)$$

where $k \leq n$ and a_k appears in a membrane with label h_k in the initial configuration (possibly as part of the input multiset). The key idea here is two-folded. On the one hand, annihilation rules do not produce any object; the objects that trigger an annihilation rule disappear and nothing is produced. On the other hand, for any halting configuration there *must* exist a finite sequence of rules $(r_k, r_{k-1}, \ldots, r_2, r_1)$ where r_k is triggered by an object from the initial configuration, r_1 produces yes and each r_i produces an object that triggers r_{i-1}. Therefore, none of rules $r_1, \ldots, r_k$ is an annihilation rule.

To formally prove the result we have to check that the amount of resources for finding the sequence of rules is polynomially bounded. With this aim, we will start by considering the dependency graph associated with $\Pi(s(u))$, but considering only evolution, communication and division rules[2] (i.e., only rules which can produce new occurrences of objects). Namely, if R is the set of rules associated with $\Pi(s(u))$, we will consider the corresponding directed graph $G = (V, E)$ defined as follows, where the function $f : H \to H$ returns the label of the parent membrane:

$V = VL \cup VR$,

$$\begin{aligned} VL = \{(a,h) \in \Gamma \times H :\ & \exists u \in \Gamma^* \ ([a \to u]_h \in R) \vee \\ & \exists b \in \Gamma \ ([a]_h \to [\ ]_h b \in R) \vee \\ & \exists b \in \Gamma \ \exists h' \in H \ (h = f(h') \wedge a[\ ]_{h'} \to [b]_{h'} \in R) \vee \\ & \exists b, c \in \Gamma \ ([a]_h \to [b]_h [c]_h \in R)\}, \end{aligned}$$

$$\begin{aligned} VR = \{(b,h) \in \Gamma \times H :\ & \exists a \in \Gamma \ \exists u \in \Gamma^* \ ([a \to u]_h \in R \wedge b \in u) \vee \\ & \exists a \in \Gamma \ \exists h' \in H \ (h = f(h') \wedge [a]_{h'} \to [\ ]_{h'} b \in R) \vee \\ & \exists a \in \Gamma \ (a[\ ]_h \to [b]_h \in R) \vee \\ & \exists a, c \in \Gamma \ ([a]_h \to [b]_h [c]_h \in R)\}, \end{aligned}$$

$$\begin{aligned} E = \{((a,h),(b,h')) : \ & \exists u \in \Gamma^* \ ([a \to u]_h \in R \wedge b \in u \wedge h = h') \vee \\ & ([a]_h \to [\ ]_h b \in R \ \wedge \ h' = f(h)) \vee \\ & (a[\ ]_{h'} \to [b]_{h'} \in R \ \wedge h \ = f(h')) \vee \\ & \exists c \in \Gamma \ ([a]_h \to [b]_h [c]_h \in R \wedge h = h')\}. \end{aligned}$$

Such a dependency graph can be constructed by a Turing machine working in polynomial time with respect to the instance size. Finally, let us consider the set

[2] See [4] for the details about polynomial resources.

$$\Delta_\Pi = \{(a,h) \in \Gamma \times H : \text{there exists a path (within the dependency graph) from } (a,h) \text{ to } (yes, env)\}.$$

It has also been proved that there exists a Turing machine that constructs Δ_Π in polynomial time; the proof uses the *Reachability Problem* in order to prove the polynomially bounded construction.

From this construction we directly obtain that the set of rules used in the chain

$$(yes, env) \xleftarrow{r_1} (a_1, skin) \xleftarrow{r_2} (a_2, h_2) \xleftarrow{r_3} \cdots \xleftarrow{r_k} (a_k, h_k)$$

described above can be found in polynomial time.

Finally, for the instance $u \in I_X$, let us consider the P system $\Pi(u')$ with only one membrane with label s and only one object (a_k, h_k) in the initial configuration. The set of rules is

- $[(a_i, h_i) \to (a_{i-1}, h_{i-1})]_s$ for each $i \in \{3, \ldots, k-1\}$
- $[(a_2, h_2) \to (a_1, skin)]_s$
- $[(a_1, skin)]_s \to yes\,[\,]_s$

The system $\Pi(u')$ can be built in polynomial time by a deterministic Turing machine. A direct inspection of the rules shows that $\Pi(u') \in \mathcal{AM}^0_{-d,+ne}$. The behavior of the system is deterministic, and it computes the correct answer for the instance $u \in I_X$, sending out the object *yes* to the environment in the last step of computation.

We finally observe that a similar construction can be carried out for the answer *no*. Hence, we conclude that $X \in \mathbf{PMC}^\star_{\mathcal{AM}^0_{-d,+ne}} = \mathbf{P}$. □

Remark 1. Let us finally explain the idea how to even get a uniform family of recognizer P systems from the family constructed in the preceding proof by making some preprocessing: For any input of length n, we include all possible input symbols in the dependency graph. If there is a path from some symbol to *yes* and from another symbol to *no*, then by the definition of confluence, an input containing both of these symbols simultaneously cannot be a valid input. So, once we get an input of length n, we first check if it has symbols deriving *yes* and symbols deriving *no*. This certainly is possible within polynomial time.

4 Conclusions

We have proved that by removing priority in polarizationless recognizer P systems with antimatter and annihilation rules, without dissolution, and with division of elementary and non-elementary membranes, we obtain a new characterization of the standard complexity class **P**. Since it was previously known that the same model of P systems can solve the **NP**-complete problem Subset Sum when the priority of annihilation rules is used [3], we have shown that this priority plays an important role in the computational power of these P systems.

Indeed, the most interesting aspect of our result is the fact that if the rules of these P systems are applied in different ways, a different computational power is obtained. We have thus proved that the semantics of a model can be a useful tool for studying problems of tractability. To the best of our knowledge, this is the first time where it is proved that two models of P systems syntactically identical correspond to two (presumably) different complexity classes simply because they use different semantics.

This opens a new research area in the study of tractability in membrane computing. Not only new ingredients or new models must be studied in order to find new frontiers: classical results can also be revisited in order to explore the consequences of considering alternative semantics.

Acknowledgements

Miguel A. Gutiérrez-Naranjo acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain. The authors are very grateful to Artiom Alhazov for carefully reading the paper and for also pointing out Remark 1.

References

1. Alhazov, A., Aman, B., Freund, R.: P systems with anti-matter. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Sosík, P., Zandron, C. (eds.) Membrane Computing - 15th International Conference, CMC 2014, Prague, Czech Republic, August 20-22, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8961, pp. 66–85. Springer (2014)
2. Alhazov, A., Aman, B., Freund, R., Păun, Gh.: Matter and anti-matter in membrane systems. In: DCFS 2014. Lecture Notes in Computer Science, vol. 8614, pp. 65–76. Springer (2014)
3. Díaz-Pernil, D., Peña-Cantillana, F., Alhazov, A., Freund, R., Gutiérrez-Naranjo, M.A.: Antimatter as a frontier of tractability in membrane computing. Fundamenta Informaticae 134, 83–96 (2014)
4. 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.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 3850, pp. 224–240. Springer, Berlin Heidelberg (2005)
5. Metta, V.P., Krithivasan, K., Garg, D.: Computability of spiking neural P systems with anti-spikes. New Mathematics and Natural Computation (NMNC) 08(03), 283–295 (2012)
6. Pan, L., Păun, Gh.: Spiking neural P systems with anti-spikes. International Journal of Computers, Communications & Control IV(3), 273–282 (2009)
7. Pérez-Jiménez, M.J.: An approach to computational complexity in membrane computing. In: Mauri, G., Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A.

(eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 3365, pp. 85–109. Springer (2004)
8. Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Jiménez, A., Woods, D.: Complexity - membrane division, membrane creation. In: Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) The Oxford Handbook of Membrane Computing, pp. 302 – 336. Oxford University Press, Oxford, England (2010)
9. Păun, Gh.: Some quick research topics., In these proceedings.
10. Song, T., Jiang, Y., Shi, X., Zeng, X.: Small universal spiking neural P systems with anti-spikes. Journal of Computational and Theoretical Nanoscience 10(4), 999–1006 (2013)
11. Tan, G., Song, T., Chen, Z., Zeng, X.: Spiking neural P systems with anti-spikes and without annihilating priority working in a 'flip-flop' way. International Journal of Computing Science and Mathematics 4(2), 152–162 (2013)