Antimatter as a Frontier of Tractability in Membrane Computing

Daniel Díaz-Pernil[1], Francisco Peña-Cantillana[2],
Miguel A. Gutiérrez-Naranjo[2]

[1]Research Group on Computational Topology and Applied Mathematics
Department of Applied Mathematics - University of Sevilla, 41012, Spain
sbdani@us.es

[2]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, 41012, Spain
frapencan@gmail.com, magutier@us.es

Summary. It is well known that the polynomial complexity class of recognizer polarizationless P systems with active membranes, without dissolution and with division for elementary and non-elementary membranes is exactly the complexity class **P** (see [6], Th. 2). In this paper, we prove that if such P system model is endowed with antimatter and annihilation rules, then **NP** problems can be solved. In this way, antimatter is a frontier of tractability in Membrane Computing.

1 Introduction

Antimatter is material composed of antiparticles, which have the same mass as particles of ordinary matter but have opposite charge. Encounters between particles and antiparticles lead to the annihilation of the objects, giving energy proportional to the total matter and antimatter mass, in accord with the mass-energy equivalence equation, $E = mc^2$.

The term antimatter was first used by Arthur Schuster in 1898, (see [15]). He hypothesized antiatoms, as well as whole antimatter solar systems, and discussed the possibility of matter and antimatter annihilating each other. The modern theory of antimatter began in 1928, with the papers [4, 5] by Paul Dirac. Dirac realised that the relativistic version of the Schrödinger wave equation for electrons predicted the possibility of antielectrons. These were discovered by Carl D. Anderson in 1932 [1] and named positrons (a contraction of "positive electrons").

In Membrane Computing, the notion of antimatter has been previously associated to anti-spikes in the framework of Spiking Neural P Systems (see, e.g., [10, 11, 16, 18]). In this context, when a spike and anti-spike appear in the same neuron, the annihilation occurs and both, spike and anti-spike, disappear.

In this paper, we prove that antimatter is a frontier of tractability in Membrane Computing. As detailed below, it is well known that the polynomial complexity class of recognizer P systems with active membranes without polarizations, without dissolution and with division of elementary and non-elementary division is exactly the complexity class **P** (see [6], Th. 2). In such paper, it is proved that if the described P system model is endowed with dissolution rules, then **NP**-complete problems can be solved. In this way, dissolution is a frontier of tractability.

In this paper, we consider the polynomial complexity class of recognizer P systems with active membranes without polarizations, without dissolution and with division of elementary and non-elementary division (i.e., the class which is equal to **P**) and we add antimatter and the corresponding annihilation rules. In this new model, we show a semi-uniform family of P systems which solves the Subset Sum problem. Since the Subset Sum Problem is **NP**-complete, this P system family shows that antimatter is a new frontier of the tractability in Membrane Computing

The paper is organized as follows: In the next section, we recall some basics on recognizer P systems, complexity classes and a previous result about dissolution as a frontier of tractability. Section 3 is devoted to the concept of antimatter in Membrane Computing. In Section 4, a solution to the Subset Sum problem by using antimatter and annihilation rules is shown. The paper finishes with some conclusions.

2 Recognizer P Systems

First of all, we recall the main notions related to recognizer P systems and complexity in Membrane Computing. For a detailed description, see, e.g., [12, 14].

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 (the *skin*). Each membrane identifies a region inside the system. When a membrane has no membrane inside, it is called *elementary*. The objects can be described by symbols or by strings of symbols, in such a way that *multisets of objects* are placed in the regions of the membrane structure. The objects can evolve according to given *evolution rules*, associated with the regions.

The *semantics* of the cell–like membrane systems is defined through a non–deterministic and synchronous model. A *configuration* of a cell–like membrane system consists of a membrane structure and a family of multisets of objects associated with each region of the structure. At the beginning, there is a configuration called the *initial configuration* of the system. In each time unit we can transform a given configuration in another configuration by applying the evolution rules to the objects placed inside the regions of the configurations, in a non–deterministic, maximally parallel manner (the rules are chosen in a non–deterministic way, and in each region all objects that can evolve must do it). 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 through the multiset associated with a specific output membrane (or the environment) in the final configuration.

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 (the elements are called *instances*) and θ_X is a predicate (a total Boolean function) over I_X. Let $X = (I_X, \theta_X)$ be a decision problem. A *polynomial encoding* of X is a pair (cod, s) of polynomial time computable functions over I_X such that for each instance $w \in I_X$, $s(w)$ is a natural number representing the *size* of the instance and $cod(w)$ is an multiset representing an encoding of the instance. Polynomial encodings are stable under polynomial time reductions

2.1 The P systems Model

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

(a) $[\, a \to u \,]_h$ for $h \in H$, $a \in \Gamma$, $u \in \Gamma^*$. This is an object evolution rule, associated with a membrane labelled with h: an object $a \in \Gamma$ belonging to that membrane evolves to a string $u \in \Gamma^*$.

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

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

(d) $[\, a \,]_h \to [\, b \,]_h\ [\, c \,]_h$ for $h \in H$, $a, b, c \in \Gamma$. An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects.

(e) $[\,[\,]_{h_1}[\,]_{h_2}\,]_{h_0} \to [\,[\,]_{h_1}\,]_{h_0}\,[\,[\,]_{h_2}\,]_{h_0}$, where h_0, h_1, h_2 are labels. They are 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 only one rule (chosen in a non–deterministic way), but any object which can evolve by one rule of any form, must evolve.

- If at the same time a membrane labelled with h is divided by a rule of type (d) or (e) and there are objects in this membrane which evolve by means of rules of type (a), then we suppose that first the evolution rules of type (a) 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. At one step, a membrane can be the subject of *only one* rule of types (b)-(e).

We denote by $\mathcal{AM}^0_{-d,+ne}$ the class of all recognizer P systems with active membranes without polarizations, without dissolution and with division of elementary and non-elementary division. We keep the subscript $-d$ in order to stress that no dissolution rules are used in this model.

2.2 Polynomial complexity classes in recognizer P systems

Let $\mathbf{\Pi} = (\Pi(w))_{w\in I_X}$ be a family of recognizer membrane systems and let $\mathcal{R}$ be a class of recognizer P systems without input membrane. A decision problem $X = (I_X, \theta_X)$ is solvable in a semi-uniform way and in polynomial time by the family $\mathbf{\Pi} = (\Pi(w))_{w\in I_X}$ of P systems of type $\mathcal{R}$, and we denote this by $X \in \mathbf{PMC}^*_{\mathcal{R}}$, 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 $\theta_X(w) = 1$ and $\mathbf{\Pi}$ is complete with regard to X if for each instance of the problem $w \in I_X$, if $\theta_X(w) = 1$, then every computation of $\Pi(w)$ is an accepting computation.

Let $\mathcal{R}$ be a class of recognizer P systems with input membrane. A decision problem $X = (I_X, \theta_X)$ is solvable in a uniform way and polynomial time by a family $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$, of P systems from $\mathcal{R}$, and we denote this by $X \in \mathbf{PMC}_{\mathcal{R}}$, if the family $\mathbf{\Pi}$ is polynomially uniform by Turing machines, i.e., there exists a polynomial encoding[1] (cod, s) from I_X to $\mathbf{\Pi}$ such that the family $\mathbf{\Pi}$ is polynomially bounded with regard to (X, cod, s); that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ 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 $\mathbf{PMC}^*_{\mathcal{R}}$ and $\mathbf{PMC}_{\mathcal{R}}$ are closed under polynomial–time reduction and complement.

According to these formal definitions, in [6] it is proved that the polynomial complexity class of recognizer P systems with active membranes without polarizations, without dissolution and with division of elementary and non-elementary division is exactly the complexity class $\mathbf{P}$. With the standard notation, $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne}} = \mathbf{PMC}^*_{\mathcal{AM}^0_{-d,+ne}}$.

[1] See [12, 14] for the details.

3 Antimatter

In this paper, we will use the physical inspiration of antimatter in the framework of cell-like P systems. In such way, given two object a and b from the alphabet Γ, an annihilation rule of a and b is written as $[ab \rightarrow \lambda]_h$. The *meaning* of the rule follows the idea of annihilation: If a and b occur simultaneously in the same membrane 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 objects from Γ and they can trigger any other rule of type (a) - (d) described above, not only annihilation rules. Nonetheless, in order to make the readability easier, if b annihilates the object a then b will be called the *antiparticle* of a and we will write $\overline{a}$ instead of b.

With respect to the semantics, let us notice that this rule must be applied as many times as possible in each membrane, 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 $m \geq n$ and the rule $[a\overline{a} \rightarrow \lambda]_h$ is defined in the P system, then the rule is applied n times, n copies of a and $\overline{a}$ are consumed and $m - n$ copies of a are not affected by this rule.

Finally, a last consideration about the application of the annihilation rule. According to the non-determinism, if an object a can trigger more than one rule of types (a) - (d), then one rule of the applicable ones is non-deterministically chosen. Nonetheless, annihilation rules introduce a novelty. According to the physical intuition, if a and $\overline{a}$ occur simultaneously in the same membrane h and the annihilation rule $[a\overline{a} \rightarrow \lambda]_h$ is defined, then it is applied, regardless other options. In this sense, any annihilation rule has priority on the other types of rules.

For example, let us consider the rules $R_1 \equiv [a \rightarrow cd]_h$, $R_2 \equiv [a]_h \rightarrow b\,[\,]_h$ and $R_3 \equiv [a\overline{a} \rightarrow \lambda]_h$ defined on a membrane with label e. If the multiset inside the membrane is ab, then R_1 and R_2 are applicable and one of then must be non-deterministically chosen. Nevertheless, if the multiset is $ab\overline{a}$, then R_3 must be applied since it is an annihilation rule. If the multiset is $a^3b\overline{a}^2$, then the annihilation rule is applied twice, by consuming two copies of a and $\overline{a}$ and the third a is consumed by one rule of R_1 or R_2 non-deterministically chosen.

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

1. $m \geq 1$ is the initial degree of the system;
2. O 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 with elements of H;
5. $w_1, \ldots, w_m$ are strings over O, describing the *multisets of objects* placed in the m regions of μ;
6. R is a finite set of *rules* of the types (a) - (e) describe in the Section 2.1 and the following type of rules:

(f) $[\, a\overline{a} \rightarrow \lambda\,]_h$ for $h \in H$, $a\,\overline{a} \in \Gamma$. This is an annihilation rule, associated with a membrane labelled with h: the pair of objects $a\,\overline{a} \in \Gamma$ belonging simultaneously to that membrane disappear.

By following with the standard notation, we denote by $\mathcal{AM}^0_{-d,+ne,+ant}$ the class of these P systems, where $-d$ denotes that dissolution rules are not used, $+ne$ denotes the use of elementary and non elementary division and we add $+ant$ to denote the use of antimatter and annihilation rules.

4 Solving the Subset Sum Problem

In this paper we show that the class of decision problems solvable in polynomial time in a semi–uniform way by families of recognizer P systems in $\mathcal{AM}^0_{-d,+ne,+ant}$ contains the standard complexity class **NP**. Formally, we will prove the following theorem

Theorem 1. $\mathbf{NP} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^0_{-d,+ne,+ant}}$

We will prove it by the construction of a semi-uniform family of such P systems that solves the Subset Sum Problem in a linear time. It is well known that the Subset Sum problem is the following one: *Given a finite set A, a weight function, $w : A \rightarrow \mathbb{N}$, and a constant $k \in \mathbb{N}$, determine whether or not there exists a subset $B \subseteq A$ such that $w(B) = k$.* This problem has been widely studied in Membrane Computing (see, e.g., [2, 3, 7, 8, 9, 13, 17]).

Let us start by considering a tuple $u = (n, (w_1, \ldots, w_n), k)$ to represent an instance of the problem, where n stands for the size of $A = \{a_1, \ldots, a_n\}$, $w_i = w(a_i)$, and k is the constant given as input for the problem.

As usual, the idea of the design is better understood if we divide the solution to the problem into several stages:

- *Generation stage*: for every subset of A, a membrane is generated via membrane division.
- *Weight calculation stage*: in each membrane the weight of the associated subset is calculated.
- *Checking stage*: for each membrane it is checked whether or not the weight of its associated subset is exactly k. This stage cannot start before the previous ones are over.
- *Output stage*: when the previous stage has been completed in all membranes, the system sends out the answer to the environment.

For each instance $u = (n, (w_1, \ldots, w_n), k)$ of the Subset Sum problem we consider the P system $\Pi(u)$ defined as follows:

• Working alphabet:

$$\begin{aligned}\Gamma(u) = &\{a, \overline{a}, b, c_1, c_2, \overline{c}_2, d_0, \ldots, d_{2n+3}, e_1, \ldots, e_n\} \\ &\cup \{p_1, \ldots, p_n, k_{2n+4}, k_{2n+5}, r_{2n+2}, \ldots, r_{2n+6}\} \\ &\cup \{yes_{2n+8}, \ldots, yes_{2n+k+12}, \overline{yes}_{2n+8}, \overline{yes}_{2n+k+12}\} \\ &\cup \{z_{2n+4}, \ldots, z_{2n+7}, no_1, \ldots, no_{2n+k+10}, \overline{no}_{2n+k+10}, \overline{no}_{2n+k+11}\} \\ &\cup \{yes, no\}\end{aligned}$$

- Initial membrane structure: $\mu = [\,[\,[\,[\,]_0\,]_1\,]_2\,]_3$.
- Initial Multisets: $w_0 = d_0$, $w_1 = \emptyset$, $w_2 = \emptyset$ and $w_3 = no_0$.
- The set of evolution rules, $R(u)$, consists of the following rules.

Generation stage

(a) $[d_{2i} \to p_{i+1} d_{2i+1}]_0$ for $i \in \{0, \ldots, n-1\}$
$[d_{2i+1} \to d_{2i+2}]_0$ for $i \in \{0, \ldots, n-1\}$
$[d_{2n} \to d_{2n+1}]_0$
$[d_{2n+1} \to d_{2n+2} r_{2n+2}]_0$

The goal of the counter d_i is to control the apparition of the object p_i only in the odd steps. These p_i will produce the division of elementary membranes.

(b) $[p_i]_0 \to [e_i]_0\,[b]_0$ for $i \in \{1, \ldots, n\}$

The object p_i triggers the rule for division of elementary membranes: in one membrane is placed the object e_i and in the other the object b.

(c) $[\,[\,]_i\,[\,]_i\,]_{i+1} \to [\,[\,]_i\,]_{i+1}\,[\,[\,]_i\,]_{i+1}$ for $i \in \{0, 1, 2\}$.

This is the set of rules for the division of non-elementary membranes.

These three first set of rules produce the membrane structure needed for computing the solution. Notice that the objects p_i produce the division of the elementary membranes for $i \in \{1, \ldots, n\}$ and therefore, in the configuration C_{2i} there are 2^i elementary membranes. Let us also remark that the division of the elementary membranes is propagated by the rules of division on non elementary membranes, so in the configuration C_{2n+2}, the skin, labelled by 3, contains 2^n membranes labelled by 2. Each of them contains one membrane labelled by 1, and each of these membranes labelled by 1 contains one elementary membrane labelled by 0.

Weight calculation stage

(d) $[e_i \to a^{w_i}]_0$ for $i \in \{1, \ldots, n\}$

After the application of the membrane division rule by the object p_i, in one membrane is placed the object e_i and in the other the object b. Since the division is produced by p_i with $i \in \{1, \ldots, n\}$ this means that each of the 2^n elementary membranes receive a possible subset of $\{e_1, \ldots, e_n\}$. Object b remain inactive whereas the objects e_i evolve in the next step to as many objects s as the weight w_i. In such way, each elementary membrane contains as many objects s as the weight of the associated subset.

(e) $[d_{2n+2} \to d_{2n+3}]_0$
$[d_{2n+3} \to \overline{a}^{k+1}]_0$

When the generation stage has finished, the object d_{2n+3} produces $k+1$ copies of the object $\overline{a}$. These objects will interact with the object a by producing annihilation according to the following rule.

(f) $[a\overline{a} \to \lambda]_0$
$[\overline{a}]_0 \to b\,[\,]_0$

These are the key rules this stage. The rules $[e_i \to s^{w_i}]_0$ from the set b have generated as many copies of objects a as the weight of the subset encoded in the membrane. On the other hand, the rule $[d_{2n+3} \to \overline{a}^{k+1}]_0$ has generated $k+1$ copies of the object $\overline{a}$.

- If the weight of the subset encoded in the membrane (number of objects a) is greater than or equal to $k+1$ (number of objects $\overline{a}$), then all the objects $\overline{a}$ are consumed by the annihilation rules.
- If the weight of the subset encoded in the membrane (number of objects a) is equal to k, then the annihilation rule is applied k times and k copies of a and $\overline{a}$ are consumed. The remaining copy of $\overline{a}$ triggers the rule $[\overline{a}]_0 \to b\,[\,]_0$ and one object b appears in the corresponding membrane 1 in the configuration C_{2n+5}.
- Finally, if the weight of the subset encoded in the membrane (number of objects a) is lower than k, then all the copies of a are consumed by the annihilation rule, but p objects $\overline{a}$ are not affected by this rule, where $p \in \{2, \ldots, k+1\}$. These objects will trigger the rule $[\overline{a}]_0 \to b\,[\,]_0$, by due to the semantics of the P systems with active membranes, only one object can cross the membrane in each step and therefore, one object b will appear in the membrane 1 at the configuration C_{2n+5} whereas $p-1$ copies of $\overline{a}$ remain in the membrane 0.

(g) $[r_{2n+2}]_0 \to r_{2n+3}\,[\,]_0$
$[r_{2n+3} \to r_{2n+4} k_{2n+4} z_{2n+4}]_1$
$[r_{2n+4}]_1 \to r_{2n+5}\,[\,]_1$
$[k_{2n+4} \to k_{2n+5}]_1$
$[z_{2n+4} \to z_{2n+5}]_1$

The object r_{2n+2} produced by the last rule of the set (a) is the starting point for this set of rules. Its purpose is to place the counters r_i, z_i and k_i in the right membranes before starting the checking stage. Notice that the starting indices have been chosen for improving the readability. In this configuration C_{2n+5}, an object z_{2n+5} and an object k_{2n+5} are placed in each membrane with label 1 and one object r_{2n+5} is placed on each membrane with label 2.

Checking stage

This stage is quite technical. The aim is that each elementary membrane produces an object yes_{2n+k+9} in a membrane labelled by 2 in the configuration

C_{2n+k+9} if and only if in the configuration C_{2n+5} one and only one object $\overline{a}$ is placed in the elementary membrane.

(h) $[b \to c_1\overline{c}_2]_1$ $\quad$ $[k_{2n+5} \to c_2]_1$
$[c_1 \to c_2]_1$ $\quad$ $[z_{2n+5} \to z_{2n+6}]_1$
$[c_2\overline{c}_2 \to \lambda]_1$ $\quad$ $[z_{2n+6} \to z_{2n+7}]_1$
$[c_2]_1 \to yes_{2n+8}\,[\,]_1$ $\quad$ $[z_{2n+7} \to \overline{c}_2]_1$

Each object $\overline{a}$ in the elementary membrane is sent out, step by step, transformed into a b object. Since there exists one elementary membrane where the empty subset is encoded with zero objects a, then such elementary membrane contains $k+1$ objects $\overline{a}$ and the process of sending out these objects will take $k+1$ steps. Notice also that a new annihilation process (of the objects c_2 and $\overline{c}_2$) is also considered.

(i) $[r_{2n+5} \to r_{2n+6}]_2$ $\quad$ $[\overline{yes}_{2n+8} \to b]_2$
$[r_{2n+6} \to \overline{yes}_{2n+8}]_2$ $\quad$ $[yes_{2n+8}\,\overline{yes}_{2n+8} \to \lambda]_2$

The counter r_i produces the object $\overline{yes}_{2n+8}$ in the membrane 2 exactly in the configuration C_{2n+7}. If in this time there is an object yes_{2n+8} in the membrane, both are annihilated. If not, $\overline{yes}_{2n+8}$ evolves to b. The purpose is this set of rules is to control the evolution of the object yes_{2n+8}. If it appears in a membrane labelled by 2 in the configuration C_{2n+7}, it will be annihilated. If it appears later, it will survive.

(j) $[yes_{2n+8+i} \to yes_{2n+8+(i+1)}]_2$ for $i \in \{0, \ldots, k-1\}$

The objects yes_i evolves in the membrane 2 waiting for the end of this stage. As pointed above, the $k+1$ objects $\overline{a}$ from the membrane encoding the empty subset take $k+1$ steps to cross out the membrane.

The result of the checking stage can be summarized in the following lemma.

Lemma 1. *In each of the 2^n membrane structures $[\,[\,[\,]_0\,]_1\,]_2$ at configuration C_{2n+k+8}:*

- *Membranes 0 and 1 are inactive. No rules can be applied inside them.*
- *Membrane 2 contains one object yes_{2n+k+8} if and only if the number of objects a in the membrane 0 at the configuration C_{2n+4} is exactly k. In other words, if it corresponds to a subset of weight k.*

This lemma will be proved in the Appendix.

Output stage

(k) $[no_i \to no_{i+1}]_3$ for $i \in \{0, \ldots, 2n+k+9\}$
$[no_{2n+k+10}]_3 \to no\,[\,]_3$

From the initial configuration, the counter no_i is evolving[2]. If the evolution is not interrupted, an object no is sent out as answer of the computation. In

[2] Of course, this counter also evolves in the previous stages, but it has not been mentioned for the sake of simplicity.

this design, if an object yes_{2n+k+8} occurs in a membrane labelled by 2, then the counter no_i is stopped and yes will be sent out as the answer. Nonetheless, more than one of such objects yes_{2n+k+8} can be produced in different membranes. Dealing with this possibility needs to add some technical rules.

(l)
$[yes_{2n+k+8}]_2 \to yes_{2n+k+9}\,[\,]_2$
$[yes_{2n+k+9} \to \overline{no}_{2n+k+10}\,yes_{2n+k+10}]_3$
$[yes_{2n+k+10} \to yes_{2n+k+11}]_3$
$[yes_{2n+k+11} \to yes_{2n+k+12}]_3$
$[yes_{2n+k+12}\,\overline{yes}_{2n+k+12} \to \lambda]_3$
$[yes_{2n+k+12}]_3 \to yes\,[\,]_3$

This set of rules, together with the next one, controls the output of the system. The key ideas are that the object yes_{2n+k+9} produces an object $\overline{no}_{2n+k+10}$, which stops the counter no_i and $yes_{2n+k+12}]_3$ sends out the answer yes in the last step of computation.

(m)
$[no_{2n+k+9} \to no_{2n+k+10}]_3$
$[no_{2n+k+10}]_3 \to no\,[\,]_3$
$[no_{2n+k+10}\overline{no}_{2n+k+10} \to \lambda]_3$
$[\overline{no}_{2n+k+10} \to \overline{no}_{2n+k+11}]_3$
$[\overline{no}_{2n+k+11} \to \overline{yes}_{2n+k+12}]_3$

This is the last set of rules in our design. Let us remark that the object $no_{2n+k+11}$ sends to the environment the object no as an answer if no object $\overline{no}_{2n+k+11}$ is produced. If this object is produced, then the annihilation occurs and the object no is never sent out. The result of this checking stage is summed up in the following lemma.

Lemma 2. *If any of the 2^n membranes with label 2 contains an object yes_{2n+k+8} in the configuration C_{2n+k+8}, then the P system halts at the configuration $C_{2n+k+13}$ and sends yes to the environment in the last step of computation. Otherwise, the P system halts at the configuration $C_{2n+k+11}$ and sends no to the environment in the last step of computation.*

This lemma will be proved in the Appendix and it finishes the proof of Th. 1.

5 Conclusions

In this paper, we present a new frontier of tractability in Membrane Computing by adding annihilation rules and the concept of antimatter to a P system model with active membranes with division and without dissolution. Let us remark that the presented design makes use of a singularity in the semantics of the annihilation rule. According to the physical intuition, in presence of the antiparticle $\overline{a}$, the particle a has no option and both are annihilated. The translation of this intuition is a priority relation with respect to the remaining applicable rules. An open problem is to know if removing this priority feature from the model, it is still possible to solve **NP** problems.

Acknowledgements

MAGN acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

References

1. Anderson, C.D.: The positive electron. Physical Review 43, 491–494 (1933)
2. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A logarithmic bound for solving subset sum with P systems. In: Eleftherakis, G., Kefalas, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 4860, pp. 257–270. Springer, Berlin Heidelberg (2007)
3. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving subset sum in linear time by using tissue P systems with cell division. In: Mira, J., Álvarez, J.R. (eds.) IWINAC (1). Lecture Notes in Computer Science, vol. 4527, pp. 170–179. Springer, Berlin Heidelberg (2007)
4. Dirac, P.A.M.: The quantum theory of the electron. I. Proceedings of the Royal Society A 117(778), 610–624 (1928)
5. Dirac, P.A.M.: The quantum theory of the electron. II. Proceedings of the Royal Society A 118(779), 351–361 (1928)
6. 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)
7. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Romero-Campero, F.J.: A linear solution of subset sum problem by using membrane creation. In: Mira, J., Álvarez, J.R. (eds.) IWINAC (1). Lecture Notes in Computer Science, vol. 3561, pp. 258–267. Springer, Berlin Heidelberg (2005)
8. Leporati, A., Gutiérrez-Naranjo, M.A.: Solving subset sum by spiking neural P systems with pre-computed resources. Fundamenta Informaticae 87(1), 61–77 (2008)
9. Leporati, A., Mauri, G., Zandron, C., Păun, Gh., Pérez-Jiménez, M.J.: Uniform solutions to SAT and subset sum by spiking neural P systems. Natural Computing 8(4), 681–702 (2009)
10. Metta, V.P., Krithivasan, K., Garg, D.: Computability of spiking nueral P systems with anti-spikes. New Mathematics and Natural Computation (NMNC) 08(03), 283–295 (2012)
11. Pan, L., Păun, Gh.: Spiking neural P systems with anti-spikes. International Journal of Computers, Communications & Control IV(3), 273–282 (September 2009)
12. 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)
13. Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving the subset-sum problem by P systems with active membranes. New Generation Computing 23(4), 339–356 (2005)

14. 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)
15. Schuster, A.: Potential Matter. A Holiday Dream. Nature 58, 367 (Aug 1898)
16. 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)
17. Song, T., Luo, L., He, J., Chen, Z., Zhang, K.: Solving subset sum problems by time-free spiking neural P systems. Applied Mathematics & Information Sciences 8(1), 327–332 (2014)
18. 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 (Jul 2013)

Appendix

Firstly, we will prove Lemma 1.

Proof. The proof is by inspection of the cases. In order to simplify the notation, we will omit the occurrences of objects a and b in the membrane 0, since they do not trigger any rule. The process is deterministic, so we will not remark the applied rules. They can be easily found in the corresponding set.

Case 1. Let us suppose that the weight of the encoded set is greater than k. In this case, the rule $[\overline{a}]_0 \to b\,[\,]_0$ is no applied and there are no objects $\overline{a}$ in membrane 0 and no object b in the membrane 1 at the configuration C_{2n+5}. With the considerations claimed above, the evolution of the membrane structure is

$$\begin{array}{ll}
C_{2n+5} & \equiv [\,[\,[\,]_0\, k_{2n+5}\, z_{2n+5}\,]_1\, r_{2n+5}\,]_2 \\
C_{2n+6} & \equiv [\,[\,[\,]_0\, c_2\, z_{2n+6}\,]_1\, r_{2n+6}\,]_2 \\
C_{2n+7} & \equiv [\,[\,[\,]_0\, z_{2n+7}\,]_1\, yes_{2n+8}\, \overline{yes}_{2n+8}\,]_2 \\
C_{2n+8} & \equiv [\,[\,[\,]_0\, \overline{c}_2\,]_1\,]_2 \\
\ldots & \ldots \\
C_{2n+k+8} & \equiv [\,[\,[\,]_0\, \overline{c}_2\,]_1\,]_2
\end{array}$$

In this case, the computation stops at the configuration C_{2n+8}. Since no more rules are applied, the result holds for C_{2n+k+8}.

Case 2. Let us suppose that the weight of the encoded set is exactly equal to k. In this case, in the configuration C_{2n+5} there are one object b in the corresponding membrane 1 and no objects in the membrane 0.

$$\begin{array}{ll}
C_{2n+5} & \equiv [\,[\,[\,\overline{a}\,]_0\, b\, k_{2n+5}\, z_{2n+5}\,]_1\, r_{2n+5}\,]_2 \\
C_{2n+6} & \equiv [\,[\,[\,]_0\, c_1\overline{c}_2\, c_2\, z_{2n+6}\,]_1\, r_{2n+6}\,]_2 \\
C_{2n+7} & \equiv [\,[\,[\,]_0\, c_2\, z_{2n+7}\,]_1\, \overline{yes}_{2n+8}\,]_2 \\
C_{2n+8} & \equiv [\,[\,[\,]_0\, \overline{c}_2\,]_1\, b\, yes_{2n+8}\,]_2 \\
\ldots & \ldots \\
C_{2n+k+8} & \equiv [\,[\,[\,]_0\, \overline{c}_2\,]_1\, b\, yes_{2n+k+8}\,]_2
\end{array}$$

Case 3. Let us suppose that the weight of the encoded set is lower than k. In this case, in the configuration C_{2n+5} there are $p-1$ objects $\overline{a}$ in the corresponding membrane 0 and one b in the membrane 1 with $p \in \{2, \ldots, k+1\}$. . We split this case into three subcases.

Case 3a: $p = 2$

$$\begin{array}{ll}
C_{2n+5} & \equiv [\,[\,[\,\overline{a}\,]_0\, b\, k_{2n+5}\, z_{2n+5}\,]_1\, r_{2n+5}\,]_2 \\
C_{2n+6} & \equiv [\,[\,[\,]_0\, b\, c_1 \overline{c}_2\, c_2\, z_{2n+6}\,]_1\, r_{2n+6}\,]_2 \\
C_{2n+7} & \equiv [\,[\,[\,]_0\, c_1\, \overline{c}_2\, c_2\, z_{2n+7}\,]_1\, \overline{yes}_{2n+8}\,]_2 \\
C_{2n+8} & \equiv [\,[\,[\,]_0\, c_2\, \overline{c}_2\,]_1\, b\,]_2 \\
C_{2n+9} & \equiv [\,[\,[\,]_0\,]_1\, b\,]_2 \\
\ldots & \ldots \\
C_{2n+k+8} & \equiv [\,[\,[\,]_0\,]_1\, b\,]_2
\end{array}$$

Case 3b: $p = 3$

$$\begin{array}{ll}
C_{2n+5} & \equiv [\,[\,[\,\overline{a}^2\,]_0\, b\, k_{2n+5}\, z_{2n+5}\,]_1\, r_{2n+5}\,]_2 \\
C_{2n+6} & \equiv [\,[\,[\,\overline{a}\,]_0\, b\, c_1 \overline{c}_2\, c_2\, z_{2n+6}\,]_1\, r_{2n+6}\,]_2 \\
C_{2n+7} & \equiv [\,[\,[\,]_0\, b\, c_1\, \overline{c}_2\, c_2\, z_{2n+7}\,]_1\, \overline{yes}_{2n+8}\,]_2 \\
C_{2n+8} & \equiv [\,[\,[\,]_0\, c_1\, c_2\, \overline{c}_2^2\,]_1\, b\,]_2 \\
C_{2n+9} & \equiv [\,[\,[\,]_0\, c_2\, \overline{c}_2\,]_1\, b\,]_2 \\
C_{2n+10} & \equiv [\,[\,[\,]_0\,]_1\, b\,]_2 \\
\ldots & \ldots \\
C_{2n+k+8} & \equiv [\,[\,[\,]_0\,]_1\, b\,]_2
\end{array}$$

Case 3c: $p \geq 4$

$$\begin{array}{ll}
C_{2n+4+1} & \equiv [\,[\,[\,\overline{a}^{p-1}\,]_0\, k_{2n+5}\, z_{2n+5}\,]_1\, r_{2n+5}\,]_2 \\
C_{2n+4+2} & \equiv [\,[\,[\,\overline{a}^{p-2}\,]_0\, b\, c_1 \overline{c}_2\, c_2\, z_{2n+6}\,]_1\, r_{2n+6}\,]_2 \\
C_{2n+4+3} & \equiv [\,[\,[\,\overline{a}^{p-3}\,]_0\, b\, c_1\, \overline{c}_2\, c_2\, z_{2n+7}\,]_1\, \overline{yes}_{2n+8}\,]_2 \\
C_{2n+4+4} & \equiv [\,[\,[\,\overline{a}^{p-4}\,]_0\, b\, c_1\, c_2\, \overline{c}_2^2\,]_1\, b\,]_2 \\
\ldots & \ldots \\
C_{2n+4+i} & \equiv [\,[\,[\,\overline{a}^{p-i}\,]_0\, b\, c_1\, c_2\, \overline{c}_2^2\,]_1\, b\,]_2 \\
\ldots & \ldots \\
C_{2n+4+(p-1)} & \equiv [\,[\,[\,\overline{a}\,]_0\, b\, c_1\, c_2\, \overline{c}_2^2\,]_1\, b\,]_2 \\
C_{2n+4+p} & \equiv [\,[\,[\,]_0\, b\, c_1\, c_2\, \overline{c}_2^2\,]_1\, b\,]_2 \\
C_{2n+4+p+1} & \equiv [\,[\,[\,]_0\, c_1\, c_2\, \overline{c}_2^2\,]_1\, b\,]_2 \\
C_{2n+4+p+2} & \equiv [\,[\,[\,]_0\, c_2\, \overline{c}_2\,]_1\, b\,]_2 \\
C_{2n+4+p+3} & \equiv [\,[\,[\,]_0\,]_1\, b\,]_2
\end{array}$$

Notice that in one of the elementary membranes, the empty set is encoded. It means that in each computation, in one of the 2^n membranes, $p = k+1$ (the greater value). In this case, $C_{2n+4+(k+1)+3} = C_{2n+k+8}$.

Next, we provide the proof of the Lemma 2.

Proof. The proof is also by inspection of the cases. As in the previous proof, the computations are deterministic, and we do not remark the applied rules.

Case 1: Let us consider that there exists exactly one membrane with label 2 contains an object yes_{2n+k+8} in the configuration C_{2n+k+8}. By application of the rule $[yes_{2n+k+8}]_2 \rightarrow yes_{2n+k+9}\,[\,]_2$, an object yes_{2n+k+9} arrives to the membrane 2 in the configuration C_{2n+k+9}. Next, we show the evolution of this membrane.

$$\begin{aligned}
C_{2n+k+9} &\equiv [\, yes_{2n+k+9}\, no_{2n+k+9}\,]_3\\
C_{2n+k+10} &\equiv [\, yes_{2n+k+10}\, \overline{no}_{2n+k+10}\, no_{2n+k+10}\,]_3\\
C_{2n+k+11} &\equiv [\, yes_{2n+k+11}\,]_3\\
C_{2n+k+12} &\equiv [\, yes_{2n+k+12}\,]_3\\
C_{2n+k+13} &\equiv yes\,[\,]_3
\end{aligned}$$

Case 2: Let us consider that there exist t membranes ($t \geq 2$) with label 2 containing an object yes_{2n+8} in the configuration C_{2n+k+8}. By application of the rule $[yes_{2n+k+8}]_2 \rightarrow yes_{2n+k+9}\,[\,]_2$, t objects yes_{2n+k+9} arrive to the membrane 2 in the configuration C_{2n+k+9}. Next, we show the evolution of this membrane.

$$\begin{aligned}
C_{2n+k+9} &\equiv [\, yes^{t}_{2n+k+9}\, no_{2n+k+9}\,]_3\\
C_{2n+k+10} &\equiv [\, yes^{t}_{2n+k+10}\, \overline{no}^{p}_{2n+k+10}\, no_{2n+k+10}\,]_3\\
C_{2n+k+11} &\equiv [\, yes^{t}_{2n+k+11}\, \overline{no}^{t-1}_{2n+k+11}\,]_3\\
C_{2n+k+12} &\equiv [\, yes^{t}_{2n+k+12}\, \overline{yes}^{t-1}_{2n+k+13}]_3\\
C_{2n+k+13} &\equiv yes\,[\,]_3
\end{aligned}$$

Case 3: Finally, let us consider that there do not exist membranes with label 2 containing an object yes_{2n+8} in the configuration C_{2n+k+8}.

$$\begin{aligned}
C_{2n+k+9} &\equiv [\, no_{2n+k+9}\,]_3\\
C_{2n+k+10} &\equiv [\, no_{2n+k+10}\,]_3\\
C_{2n+k+11} &\equiv no\,[\,]_3
\end{aligned}$$