P Systems with Active Membranes, Without Polarizations and Without Dissolution: A Characterization of P

Miguel A. Gutiérrez-Naranjo, Mario J. Pérez-Jiménez, Agustín Riscos-Núñez, and Francisco J. Romero-Campero

Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence,
University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
{magutier, marper, ariscosn, fran}@us.es

Abstract. We study the computational efficiency of recognizer P systems with active membranes without polarizations and without dissolution. The main result of the paper is the following: the polynomial computational complexity class associated with the class of recognizer P systems is equal to the standard complexity class **P**.

1 Introduction

The *theory of computation* deals with the *mechanical solvability* of problems, that is, searching solutions that can be described by a finite sequence of elementary processes or instructions. The first goal of this theory is general problem solving; that is, develop principles and methods that are able to solve any problem from a certain class of questions.

A *computational model* tries to capture those aspects of mechanical solutions of problems that are relevant to these solutions, including their inherent limitations. In some sense, we can think that computational models design machines according to certain necessity.

If we have a mechanically solvable problem and we have a specific algorithm solving it that can be implemented in a real machine, then it is very important to know how much computational resources (time or memory) are required for a given instance, in order to recognize the limitations of the real device.

Thus, one of the main goals of the *theory of computational complexity* is the study of the efficiency of algorithms and their data structures through the analysis of the resources required for solving problems (that is, according to their intrinsic computational difficulty). This theory provides a classification of the *abstract problems* that allows us to detect their inherent complexity from the computational solutions point of view.

Many interesting problems of the real world are presumably intractable and hence it is not possible to execute algorithmic solutions in an electronic computer when we deal with instances of those problems whose size is large. The theoretical

C.S. Calude et al. (Eds.): UC 2005, LNCS 3699, pp. 105–116, 2005.

limitations of the Turing machines in terms of computational power are also practical limitations to the digital computers.

Natural Computing is a new computing area inspired by nature, using concepts, principles and mechanisms underlying natural systems. *Evolutionary Computation* uses computational models of evolutionary processes as key elements in the design and implementation of computer–based problem solving systems [18]. *Neural Networks* are inspired in the structures of the brain and nervous system. *DNA Computing* is based on the computational potential of DNA molecules and on the capacity to handle them. *Membrane Computing* is inspired by the structure and functioning of living cells, and it is a cross-disciplinary field with contributions by computer scientists, biologists, formal linguists and complexity theoreticians, enriching each others with results, open problems and promising new research lines.

This emergent branch of Natural Computing was introduced by Gh. Păun in [8]. Since then it has received important attention from the scientific community. In fact, Membrane Computing has been selected by the Institute for Scientific Information, USA, as a fast *Emerging Research Front* in Computer Science, and [6] was mentioned in [19] as a highly cited paper in October 2003.

This new non-deterministic model of computation starts from the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. The devices of this model are called *P systems*.

Roughly speaking, a P system consists of a cell-like membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules in a synchronous non-deterministic maximally parallel manner[1].

Inspired in living cells, P systems abstract the way of obtaining new membranes. These processes are basically two: *mitosis* (membrane division) and *autopoiesis* (membrane creation). Both ways of generating new membranes have given rise to different variants of P systems: *P systems with active membranes*, where the new workspace is generated by membrane division, and *P systems with membrane creation*, where the new membranes are created from objects.

Both models are universal from a computational point of view, but technically, they are pretty different. In fact, nowadays there does not exist any theoretical result which proves that these models can simulate each other in polynomial time.

P systems with active membranes have been successfully used to design solutions to well-known **NP**-complete problems, as SAT [16], Subset Sum [13], Knapsack [14], Bin Packing [15] and Partition [3], but as Gh. Păun pointed in [10] *"membrane division was much more carefully investigated than membrane creation as a way to obtain tractable solutions to hard problems"*. Recently, the first results related to the power and design of algorithms to solve NP problems in these model have arisen (see [4,5]).

P systems with active membranes were introduced in [7] with the membranes having polarizations, one of the "electrical charges" $0, -, +$, and several times the problem was formulated whether or not these polarizations are necessary in

[1] A layman-oriented introduction can be found in [9] and further bibliography at [20].

order to obtain polynomial solutions to **NP**–complete problems. The last current result is that from [1], where one proves that two polarizations suffice.

The present paper is both a contribution to this problem, and a contribution to another interesting problem in membrane computing, namely, of characterizing classic complexity classes, such as **P** and **NP**, by means of membrane computing complexity classes.

Specifically, we prove that **P** is equal to the family of problems which can be solved in a polynomial time by P systems with membrane division, without polarizations and *without dissolution*. At this moment, we do not know whether this last condition can be avoided, but either result would be of a great interest: if our result would remain true also when using membrane dissolution, then we would have the possitive answer to the problem of removing polarization; the other possibility would indicate a surprising role of the –apparently "innocent"– operation of membrane dissolution, as it will make the difference between efficiency and non–efficiency for P systems with membrane division and without polarization.

2 Preliminaries

2.1 The Reachability Problem

The Reachability Problem is the following: *given a (directed or undirected) graph, G, and two nodes a, b, determine whether or not the node b is reachable from a, that is, whether or not there exists a path in the graph from a to b.*

This problem belongs to the complexity class **P**. Indeed, it is very easy to design an algorithm running in polynomial time solving it. For example, given a (directed or undirected) graph, G, and two nodes a, b, we consider a depth–first–search with source a, and we check if b is in the tree of the computation forest whose root is a. The total running time of this algorithm is $O(|V| + |E|)$, that is, in the worst case is quadratic in the number of nodes. Morover, this algorithm needs to store a linear number of items (it can be proved that there exists another polynomial time algorithm which uses $O(\log^2(|V|))$ space).

2.2 Recognizer P Systems

In the structure and functioning of a cell, biological *membranes* play an essential role. The cell is separated from its environment by means of a *skin membrane*, and it is internally compartmentalized by means of *internal membranes*.

The main *syntatic* ingredients of a cell–like membrane system (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*), and delimiting *regions* (the space in–between a membrane and the immediately inner membranes, if any). Each membrane identifies a region inside the system. A membrane structure can be considered as a rooted tree.

- Regions defined by a membrane structure contain objects corresponding to chemical substances present in the compartments of a cell. The objects can be described by symbols or by strings of symbols, in such a way that *multiset of objects* are placed in the regions of the membrane structure.
- The objects can evolve according to given *evolution rules*, associated with the regions (hence, with the membranes).

The *semantics* of the cell–like membrane systems is defined through a non–deterministic and synchronous model (a global clock is assumed) as follows:

- 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, and 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, 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.

That is, a computation in a P system is structured as follows: it starts with the initial configuration of the system, then the computation proceeds, and when it stops the result is to be found in the output membrane.

In this paper we use membrane computing as a framework to attack the resolution of decision problems. In order to solve this kind of problems and having in mind the relationship between the solvability of a problem and the recognition of the language associated with it, we consider P systems as *recognizer language* devices.

Definition 1. *A* P system with input *is a tuple* (Π, Σ, i_Π)*, where: (a)* Π *is a P system, with working alphabet* Γ*, with* p *membranes labelled by* $1, \ldots, p$*, and initial multisets* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *associated with them; (b)* Σ *is an (input) alphabet strictly contained in* Γ *and the initial multisets are over* $\Gamma - \Sigma$*; (c)* i_Π *is the label of a distinguished (input) membrane.*

The computations of a P system with input in the form of a multiset over Σ are defined in a natural way, but the initial configuration of (Π, Σ, i_Π) must be the initial configuration of the system Π to which we add the input multiset.

Definition 2. *Let (Π, Σ, i_Π) be a P system with input. Let Γ be the working alphabet of Π, μ the membrane structure, and $\mathcal{M}_1, \ldots, \mathcal{M}_p$ the initial multisets of Π. Let m be a multiset over Σ. The* initial configuration of (Π, Σ, i_Π) with input m *is* $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots, \mathcal{M}_p)$.

Let (Π, Σ, i_Π) be a P system with input. Let Γ be the working alphabet of Π, μ the membrane structure and $\mathcal{M}_1, \ldots, \mathcal{M}_p$ the initial multisets of Π. Let m be a multiset over Σ. Then we denote $\mathcal{M}_j^* = \{(a, j) : \ a \in \mathcal{M}_j\}$, for $1 \leq j \leq p$, and $m^* = \{(a, i_\Pi) : \ a \in m\}$.

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 (whose elements are called *instances*) and θ_X is a predicate (a total boolean function) over I_X.

Definition 3. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer P systems with input. A* polynomial encoding *from X to $\mathbf{\Pi}$ 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 and $cod(w)$ is an input multiset for the system $\Pi(s(w))$.*

It is easy to prove that polynomial encodings are stable under polynomial time reductions.

Proposition 1. *Let X_1, X_2 be decision problems. Let r be a polynomial time reduction from X_1 to X_2. Let (cod, s) be a polynomial encoding from X_2 to $\mathbf{\Pi}$. Then $(cod \circ r, s \circ r)$ is a polynomial encoding from X_1 to $\mathbf{\Pi}$.*

Definition 4. *A* recognizer P system *is a P system with input and external output such that:*

1. *The working alphabet contains two distinguished elements* yes *and* no.
2. *All computations halt.*
3. *If $\mathcal{C}$ is a computation of the system, then either object* yes *or object* no *(but not both) must have been released into the environment, and only in the last step of the computation.*

In recognizer P systems, we say that a computation $\mathcal{C}$ is an *accepting computation* (respectively, *rejecting computation*) if the object *yes* (respectively, *no*) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$. Hence, these devices send to the environment an accepting or rejecting answer, in the end of their computations.

2.3 A Polynomial Complexity Class in Recognizer P Systems

Definition 5. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer P systems* with input. *Let (cod, s) be a polynomial encoding from X to $\mathbf{\Pi}$.*

- *We say that the family* $\mathbf{\Pi}$ *is sound with regard to* (X, cod, s) *if the following is true: for each instance of the problem* $w \in I_X$*, if there exists an accepting computation of* $\Pi(s(w))$ *with input* $cod(w)$*, then* $\theta_X(w) = 1$*.*
- *We say that the family* $\mathbf{\Pi}$ *is complete with regard to* (X, cod, s) *if the following is true: for each instance of the problem* $u \in I_X$*, if* $\theta_X(u) = 1$ *then every computation of* $\Pi(s(u))$ *with input* $cod(u)$ *is an accepting computation.*

Next, we propose to solve a decision problem through a family of P systems constructed in polynomial time by a Turing machine, and verifying that each element of the family processes, in a specified sense, all the instances of *equivalent size*. We say that these solutions are *uniform solutions*.

Definition 6. *Let* $\mathcal{R}$ *be a class of recognizer P systems with input membrane. A decision problem* $X = (I_X, \theta_X)$ *is solvable in 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 following is true:*

- *The family* $\mathbf{\Pi}$ *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* $w \in I_X$*.*
- *There exists a polynomial encoding* (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.*
  - *The family* $\mathbf{\Pi}$ *is sound and complete with regard to* (X, cod, s)*.*

It is easy to see that the class $\mathbf{PMC}_{\mathcal{R}}$ is closed under polynomial–time reduction and complement (see [11] for details).

3 Recognizer P Systems with Active Membranes Without Polarizations and Without Dissolution

A particularly interesting class of cell–like membrane systems are the systems with active membranes, where the membrane division can be used in order to solve computationally hard problems, e.g., **NP**-complete problems, in polynomial or even linear time, by a space–time trade-off.

In this paper we work with a variant of P systems with active membranes that does not use electrical charges or dissolution rules.

Definition 7. *A recognizer P system with active membranes without polarizations and without dissolution is a recognizer P system* (Π, Γ, i_Π)*, where the rules of the associated P system 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.*

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 by h is divided by a rule of type (d) 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 by h are used for all copies of this membrane. At one step, a membrane can be the subject of *only one* rule of types $(b), (c), (d)$.

In this framework we work without cooperation, without priorities, with cell division rules for elementary membranes, and without changing the labels of membranes.

We denote by $\mathcal{AM}^0_{-d}$ the class of all recognizer P systems with active membranes without polarizations and without dissolution.

4 Dependency Graph of a Recognizer P System with Active Membranes

Let Π be a recognizer P systems with active membranes without polarizations and without dissolution. Let R be the set of rules associated with Π.

Each rule can be considered, in a certain sense, as a *dependency* between the object triggering the rule and the object or objects produced by its application.

We can consider a general format of all kinds of rules of such systems as follows: $(a, h) \to (a_1, h')(a_2, h') \dots (a_s, h')$, according to the following criterion:

- The rules of type (a) correspond to the case $h = h'$ and $s \geq 1$.
- The rules of type (b) correspond to the case $h = f(h')$ and $s = 1$.
- The rules of type (c) correspond to the case $h' = f(h)$ and $s = 1$.
- The rules of type (d) correspond to the case $h = h'$ and $s = 2$.

If h is the label of a membrane, then $f(h)$ (respectively, $ch(h)$) denotes the label of the father (resp. a child) of the membrane labelled by h. We adopt the convention that the father of the skin membrane is the environment.

For example, let us consider a general rule $(a, h) \rightarrow (a_1, h')(a_2, h') \ldots (a_s, h')$. Then we can interpret that from the object a in membrane labelled by h we can *reach* the objects $a_1, \ldots, a_s$ in membrane labelled by h'.

Next, we formalize these ideas in the following definition.

Definition 8. *Let Π be a recognizer P systems with active membranes without polarizations and without dissolution. Let R be the set of rules associated with Π. The dependency graph associated with Π is the directed graph $G_\Pi = (V_\Pi, E_\Pi)$ defined as follows:*

$V_\Pi = VL_\Pi \cup VR_\Pi$,

$$VL_\Pi = \{(a,h) \in \Gamma \times H : \exists u \in \Gamma^* \ ([a \rightarrow u]_h \in R) \vee$$
$$\exists b \in \Gamma \ ([a]_h \rightarrow [\ ]_h b \in R) \vee$$
$$\exists b \in \Gamma \ \exists h' = ch(h) \ (a[\ ]_{h'} \rightarrow [b]_{h'} \in R) \vee$$
$$\exists b, c \in \Gamma \ ([a]_h \rightarrow [b]_h [c]_h \in R)\},$$

$$VR_\Pi = \{(b,h) \in \Gamma \times H : \exists a \in \Gamma \ \exists u \in \Gamma^* \ ([a \rightarrow u]_h \in R \wedge b \in alph(u)) \vee$$
$$\exists a \in \Gamma \ \exists h' = ch(h) \ ([a]_{h'} \rightarrow [\ ]_{h'} b \in R) \vee$$
$$\exists a \in \Gamma \ (a[\ ]_h \rightarrow [b]_h \in R) \vee$$
$$\exists a, c \in \Gamma \ ([a]_h \rightarrow [b]_h [c]_h \in R)\},$$

$$E_\Pi = \{((a,h),(b,h')) : \exists u \in \Gamma^* \ ([a \rightarrow u]_h \in R \wedge b \in alph(u) \wedge h = h') \vee$$
$$([a]_h \rightarrow [\ ]_h b \in R \ \wedge \ h' = f(h)) \vee$$
$$(a[\ ]_{h'} \rightarrow [b]_{h'} \in R \ \wedge h \ = f(h')) \vee$$
$$\exists c \in \Gamma \ ([a]_h \rightarrow [b]_h [c]_h \in R \wedge h = h')\}.$$

Proposition 2. *Let Π be a recognizer P systems with active membranes without polarizations and without dissolution. There exists a Turing machine that constructs the dependency graph, G_Π, associated with Π, in polynomial time (that is, in a time bounded by a polynomial function depending on the total number of rules and the maximum length of the rules).*

Proof. A deterministic algorithm that, given a P system Π with the set R of rules, constructs the corresponding dependency graph, is the following:

Input: (Π, R)
$V_\Pi \leftarrow \emptyset$; $E_\Pi \leftarrow \emptyset$
for each rule $r \in R$ of Π do
  if $r = [a \rightarrow u]_h \wedge \mathtt{alph}(u) = \{a_1, \ldots, a_s\}$ then
    $V_\Pi \leftarrow V_\Pi \cup \bigcup_{j=1}^{s} \{(a,h),(a_j,h)\}$; $E_\Pi \leftarrow E_\Pi \cup \bigcup_{j=1}^{s} \{((a,h),(a_j,h))\}$
  if $r = [a]_h \rightarrow [\ ]_h b$ then
    $V_\Pi \leftarrow V_\Pi \cup \{(a,h),(b,f(h))\}$;
    $E_\Pi \leftarrow E_\Pi \cup \{((a,h),(b,f(h)))\}$
  if $r = a[\ ]_h \rightarrow [b]_h$ then
    $V_\Pi \leftarrow V_\Pi \cup \{(a,f(h)),(b,h)\}$;
    $E_\Pi \leftarrow E_\Pi \cup \{((a,f(h)),(b,h))\}$
  if $r = [a]_h \rightarrow [b]_h[c]_h$ then
    $V_\Pi \leftarrow V_\Pi \cup \{(a,h)),(b,h),(c,h)\}$;
    $E_\Pi \leftarrow E_\Pi \cup \{((a,h)),(b,h)),((a,h),(c,h))\}$

The running time of this algorithm is bounded by $O(|R| \cdot q)$, where q is the value $\max\{length(r) : r \in R\}$. □

Proposition 3. *Let $\Pi = (\Gamma, \Sigma, H, \mathcal{M}_1, \ldots, \mathcal{M}_p, R_1, \ldots, R_p, i_\Pi)$ be a recognizer P systems with active membranes without polarizations and without dissolution. Let Δ_Π be defined as follows:*

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

Then, there exists a Turing machine that constructs the set Δ_Π in polynomial time (that is, through a polynomial function depending on the total number of rules and the maximum length of the rules).

Proof. We can construct the set Δ_Π from Π as follows:

- We construct the dependency graph G_Π associated with Π.
- Then we consider the following algorithm:

Input: $G_\Pi = (V_\Pi, E_\Pi)$
  $\Delta_\Pi \leftarrow \emptyset$
  for each $(a,h) \in V_\Pi$ do
    if **reachability** $(G_\Pi, (a,h), (\mathtt{yes}, environment)) = \mathtt{yes}$ then
      $\Delta_\Pi \leftarrow \Delta_\Pi \cup \{(a,h)\}$

The running time of this algorithm is of the order $O(|V_\Pi| \cdot |V_\Pi|^2)$, hence it is of the order $O(|\Gamma|^3 \cdot |H|^3)$. □

Proposition 4. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ be a family of recognizer P systems with input membrane solving X, according to Definition 6. Let (cod, s) be the polynomial encoding associated with that solution. Then, for each instance w of the problem X the following assertions are equivalent:*

(a) $\theta_X(w) = 1$ *(that is, the answer to the problem is* `yes` *for* w*).*

(b) $\Delta_{\Pi(s(w))} \cap ((cod(w))^* \cup \bigcup_{j=1}^{p} \mathcal{M}_j^*) \neq \emptyset$*, where* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *are the initial multisets of the system* $\Pi(s(w))$*.*

Proof. Let $w \in I_X$. Then $w \in L_X$ if and only if there exists an accepting computation of the system $\Pi(s(w))$ with input multiset $cod(w)$. But this condition is equivalent to the following: in the initial configuration of $\Pi(s(w))$ with input multiset $cod(w)$ there exists an object $a \in \Gamma$ in a membrane labelled by h such that in the dependency graph the node $(\texttt{yes}, environment)$ is reachable from (a, h).

Hence, $\theta_X(w) = 1$ if and only if $\Delta_{\Pi(s(w))} \cap \mathcal{M}_1^* \neq \emptyset$, or $\ldots$, or $\Delta_{\Pi(s(w))} \cap \mathcal{M}_p^* \neq \emptyset$, or $\Delta_{\Pi(s(w))} \cap (cod(w))^* \neq \emptyset$. □

Theorem 1. $\mathbf{PMC}_{\mathcal{AM}^0_{-d}} = \mathbf{P}$.

Proof. We have $\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d}}$ because the class $\mathbf{PMC}_{\mathcal{AM}^0_{-d}}$ is closed under polynomial time reduction. Next, we show that $\mathbf{PMC}_{\mathcal{AM}^0_{-d}} \subseteq \mathbf{P}$. For that, let $X \in \mathbf{PMC}_{\mathcal{AM}^0_{-d}}$. Let $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ a family of recognizer P systems with input membrane solving X, according to Definition 6. Let (cod, s) be the polynomial encoding associated with that solution.

We consider the following deterministic algorithm:

```
Input: An instance w of X
  - Construct the system Π(s(w)) with input multiset cod(w).
  - Construct the dependency graph G_Π(s(w)) associated with Π(s(w)).
  - Construct the set Δ_Π(s(w)) according to Proposition 3
    answer ← No; j ← 1
    while j ≤ p ∧ answer = No do
        if Δ ∩ M_j^* ≠ ∅ then
            answer ← yes
        j ← j + 1
    endwhile
    if Δ ∩ (cod(w))^* ≠ ∅ then
        answer ← yes
```

On one hand, the answer of this algorithm is `yes` if and only if there exists a pair (a, h) belonging to $\Delta_{\Pi(s(w))}$ such that in the membrane labelled by h in the initial configuration (with input the multiset $cod(w)$) appears the symbol a.

On the other hand, a pair (a, h) belongs to $\Delta_{\Pi(s(w))}$ if and only if there exists a path from (a, h) to $(\texttt{yes}, environment)$; that this, if and only if we can obtain an accepting computation of $\Pi(s(w))$ with input $cod(w)$. Hence, the algorithm above described solves the problem X.

The cost to determine whether or not $\Delta \cap M_j^* \neq \emptyset$ (or $\Delta \cap (cod(w))^* \neq \emptyset$) is of the order $O(|\Gamma|^2 \cdot |H|^2)$.

Hence, the running in time of this algorithm can be bounded as $f(|w|) + O(|R| \cdot q) + O(p \cdot |\Gamma|^2 \cdot |H|^2)$, where f is the (total) cost of a polynomial encoding from X to $\mathbf{\Pi}$, R the set of rules of Π, and $q = \max\{length(r) : r \in R\}$. But from Definition 6 we have that all involved parameters are polynomials in $|w|$. That is, the algorithm is polynomial in the size $|w|$ of the input. □

5 Conclusions

Dependency graphs associated with a variant of recognizer P systems with active membranes are introduced. This concept allows us to characterize the accepting computations of these systems through the reachability of a distinguished node of the graph from other nodes associated with the initial configuration.

In this paper, we have showed that in the framework of P systems with active membranes if we remove electrical charges and dissolution, then it is possible to solve in polynomial time only problems which are tractable in the standard sense.

But what happens if in this framework we consider dissolution rules? Will be possible to solve **NP**–complete problems? If the answer is yes, then this result will provide a negative answer to the P–conjecture ($\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0}$, where $\mathcal{AM}^0$ is the class of all recognizer P systems with active membranes, without polarization, using dissolution rules and cell division rules only for elementary membranes).

Acknowledgement

The authors wish to acknowledge the support of the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds.

References

1. A. Alhazov, R. Freund, Gh. Păun: P systems with active membranes and two polarizations. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, 2004, 20–35.
2. M.R. Garey, D.S. Johnson: *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
3. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: A fast P system for finding a balanced 2-partition. *Soft Computing*, in press.

4. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero: Solving SAT with membrane creation. *CiE 2005: New Computational Paradigms, Amsterdan* (S. Barry Cooper, B. Lowe, L. Torenvliet, eds.), Report ILLC X-2005-01, University of Amsterdam, 82–91.
5. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, F.J. Romero-Campero: A linear solution for QSAT with Membrane Creation. Submitted, 2005.
6. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–305
7. Gh. Păun, Computing with membranes: Attacking **NP**–complete problems. In *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), 2000, 94–115.
8. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
9. Gh. Păun, M.J. Pérez-Jiménez: Recent computing models inspired from biology: DNA and membrane computing. *Theoria*, 18, 46 (2003), 72–84.
10. Gh. Păun: Further open problems in membrane computing. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero, F. Sancho, eds.), Report RGNC 01/04, University of Seville, 2004, 354–365.
11. M.J. Pérez–Jiménez: An approach to computational complexity in Membrane Computing. En G. Mauri, Gh. Păun, M. J. Pérez-Jiménez, Gr. Rozenberg, A. Salomaa (eds.) *Membrane Computing, 5th International Workshop, WMC5, Revised Selected and Invited Papers*. LNCS 3365 (2005), 85-109.
12. M.J. Pérez–Jiménez, A. Riscos–Núñez: A linear time solution to the Knapsack problem using active membranes. *Membrane Computing* (C. Martín-Vide, Gh. Păun, G. Rozenberg, A. Salomaa, eds.). LNCS 2933 (2004), 250–268.
13. M.J. Pérez-Jiménez, A. Riscos-Núñez: Solving the Subset-Sum problem by active membranes. *New Generation Computing*, in press.
14. M.J. Pérez-Jiménez, A. Riscos-Núñez: A linear solution for the Knapsack problem using active membranes. In *Membrane Computing* (C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa, eds.). LNCS 2933 (2004), 250–268.
15. M.J. Pérez-Jiménez, F.J. Romero-Campero: Solving the BIN PACKING problem by recognizer P systems with active membranes. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos, A. Romero and F. Sancho, eds.), Report RGNC 01/04, University of Seville, 2004, 414–430.
16. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: A polynomial complexity class in P systems using membrane division. *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003* (E. Csuhaj-Varjú, C. Kintala, D. Wotschke, G. Vaszil, eds.), 2003, 284-294.
17. P. Sosik: The computational power of cell division. *Natural Computing*, 2, 3 (2003), 287–298.
18. W.M. Spears: *Evolutionary Algorithms. The role of mutation and recombination.* Natural Computing Series, Springer, 2000.
19. ISI web page `http://esi-topics.com/erf/october2003.html`
20. P systems web page `http://psystems.disco.unimib.it/`