Cell-like and Tissue-like Membrane Systems as Recognizer Devices

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

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

Abstract. *Most of the variants of membrane systems found in the literature are generally thought as generating devices. In this paper recognizer computational devices (cell–like and tissue–like) are presented in the framework of Membrane Computing, using the biological membranes arranged hierarchically, inspired from the structure of the cell, and using the biological membranes placed in the nodes of a graph, inspired from the cell inter–communication in tissues. In this context, polynomial complexity classes of recognizer membrane systems are introduced. The paper also addresses the* **P** *versus* **NP** *problem, and the (efficient) solvability of computationally hard problems, in the framework of these new complexity classes.*

1 Introduction

One of the main goals of a computing model is to solve problems. In order to design computational devices capable of attacking decision problems, we must decide how to represent by strings the instances of the problem. In that context, to solve a decision problem consists of recognizing the language associated with it.

Membrane Computing is a young branch of Natural Computing providing distributed parallel computing models whose computational devices are called *membrane systems*, which are inspired by some basic biological features, by the structure and functioning of the living cells, as well as from the cooperation of cells in tissues, organs, and organisms.

In this area there are basically two ways to consider computational devices: cell–like membrane systems and tissue–like membrane systems. The first one, using the biological membranes arranged hierarchically, inspired from the structure of the cell, and the second one using the biological membranes placed in the nodes of a graph, inspired from the cell inter–communication in tissues.

In this paper we present recognizer membrane systems (both cell–like and tissue–like variants) as a framework to address ways to efficiently solving computationally hard problems, capturing the true concept of algorithm in spite of providing a non–deterministic computing model.

2 Preliminaries

The computational devices of a model of computation are designed to handle inputs and outputs that are strings over a finite alphabet.

Usually, **NP**-completeness has been studied in the framework of *decision problems*, but it is not an important restriction because one can easily transform any optimization problem into a

roughly equivalent decision problem by supplying a target value for the quantity to be optimized, and asking the question whether this value can be attained.

Definition 1. A decision problem is a pair (I, θ) such that I is a language over a finite alphabet (whose elements are called *instances*) and θ is a total boolean function (that is, a predicate) over I.

There exists a natural correspondence between languages and decision problems in the following way. Each language L, over an alphabet Σ, has a decision problem, X_L, associated with it as follows: $I_{X_L} = \Sigma^*$, and $\theta_{X_L} = \{(u, 1) \mid u \in L\} \cup \{(u, 0) \mid u \in \Sigma^* - L\}$; reciprocally, given a decision problem $X = (I_X, \theta_X)$, the language L_X over the alphabet of I_X corresponding to it is defined as follows: $L_X = \{u \in I_X \mid \theta_X(u) = 1\}$.

The **P** versus **NP** problem is the problem of determining whether every problem solvable by some non-deterministic Turing machine in polynomial time can also be solved by some deterministic Turing machine in polynomial time. It is one of the outstanding open problems in theoretical computer science. A negative answer to this question would confirm that the majority of current cryptographic systems are secure from a practical point of view. A positive answer would not only show the uncertainty about the security of these systems, but also this kind of answer is expected to come together with a general procedure that provides a deterministic algorithm solving most of the **NP**–complete problems in polynomial time.

In the last years several computing models using powerful tools from nature have been developed (because of this, they are known as *bio-inspired* models) and several solutions in polynomial time to problems from the class **NP** have been presented, making use of non-determinism and/or of an exponential amount of space. This is the reason why a practical implementation of such models (in biological, electronic, or other media) could provide a significant advance in the resolution of computationally hard problems.

3 Cell–like recognizer membrane 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 *syntactic* ingredients of a cell–like membrane system are the *membrane structure*, the *multisets*, and the *evolution rules*.

- A *membrane structure* consists of several membranes arranged in a hierarchical structure 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 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 (in the sense that 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 one is obtained from the previous one by a transition, and shows how the system is evolving.
- 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.

In the basic version, cell–like membrane systems can be seen as generating devices, working in a non–deterministic and maximally parallel manner, with output membrane, and without input membrane.

But we are very interested in to use cell-like membrane systems in order to solve decision problems. What are the necessary ingredients to solve problems with a computational device?

We will see that we must work with non–deterministic (but confluent) systems, using maximal parallelism, without output membrane (the output will be in the enviroment), and with input membrane.

Definition 2. 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 $\Gamma - \Sigma$; and (c) i_Π is the label of a distinguished (input) membrane.

If m is a multiset over Σ, then the *initial configuration of* (Π, Σ, i_Π) *with input* m is $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots \mathcal{M}_p)$.

Definition 3. A *cell–like recognizer membrane system* is a P system with input, (Π, Σ, i_Π), and with external output such that: (1) The working alphabet contains two distinguished elements YES, NO; (2) All computations halt; and (3) In every computation of Π, either some object YES or some object NO (but not both) must have been released into the environment, and only in the last step of the computation.

We say that $\mathcal{C}$ is an accepting (respectively, rejecting) computation if the object YES (respectively, NO) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$.

We denote by $\mathcal{R}$ the class of all cell–like recognizer membrane systems.

We propose to solve a decision problem through a family of P systems (constructed in a uniform way) such that each element of the family processes all the instances of *equivalent size*, in some sense (we say that these solutions are *uniform* solutions).

Next, we define what means to solve a decision problem in the framework of cell–like membrane systems, and in an uniform way.

Definition 4. We say that 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 $\mathcal{R}$, 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 constructing $\Pi(n)$ from $n \in \mathbf{N}$ in polynomial time.
- There exists a pair (cod, s) of polynomial-time computable functions whose domain is L, such that:
  - For each $u \in L$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of $\Pi(s(u))$.

- 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 $\Pi(h(u))$ with input $g(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps.
- The family $\mathbf{\Pi}$ is sound, with regard to (X, cod, s); that is, for each $u \in I_X$ it is verified that if there exists an accepting computation of $\Pi(h(u))$ with input $g(u)$, then $\theta_X(u) = 1$.
- The family $\mathbf{\Pi}$ is complete with regard to (X, cod, s); that is, for each $u \in I_X$ it is verified that if $\theta_X(u) = 1$, then every computation of $\Pi(h(u))$ with input $g(u)$ is an accepting one.

In the above definition we have imposed every P system $\Pi(n)$ to be *confluent*, in the following sense: every computation with the *same* input produces the *same* output.

We have the class $\mathbf{PMC}_{\mathcal{R}}$ is closed under polynomial–time reduction and complement.

If systems without input membrane are used, constructing *one* specific system for each instance, but keeping the 'polynomially uniform by Turing machines' condition, then we say that the obtained solutions are *semi–uniform*. We shall denote by $\mathbf{PMC}^*_{\mathcal{R}}$ the class of problems solvable in polynomial time by a semi–uniform family of systems in $\mathcal{R}$.

4 The P versus NP problem in the context of cell–like recognizer membrane systems

We consider deterministic Turing machines as language recognizer devices. Then, we can associate with each deterministic Turing machine a decision problem, which will permit us to define when such a machine is simulated by a family of P systems (this issue was also addressed e.g. in [12, 20]).

Definition 5. Let M be a Turing machine with input alphabet Σ_M. The *decision problem associated with* M is the problem $X_M = (I, \theta)$, where $I = \Sigma^*_M$, and for every $w \in \Sigma^*_M$, $\theta(w) = 1$ if and only if M accepts w.

Obviously, the decision problem X_M is solvable by the Turing machine M.

Definition 6. We say that a Turing machine, M, is simulated in polynomial time by a family of systems of the class $\mathcal{R}$, if $X_M \in \mathbf{PMC}_{\mathcal{R}}$.

In cell–like membrane systems, evolution rules, communication rules and rules involving dissolution are called *basic rules*. That is, by applying this kind of rules the size of the membrane structure does not increase. Hence, it is not possible to construct an exponential working space in polynomial time using only basic rules in a cell–like membrane system.

We recall here a result from Chapter 9 of [22].

Proposition 1. *Let M be a deterministic Turing machine working in polynomial time. Then M can be simulated in polynomial time by a family of cell–like recognizer membrane systems using only basic rules.*

Reciprocally, in [20] the following result was proved:

Proposition 2. *For every decision problem solvable in polynomial time by a family of cell–like recognizer membrane systems using only basic rules, there exists a Turing machine solving it in polynomial time.*

Under the hypothesis $\mathbf{P}{\neq}\mathbf{NP}$, Zandron et al. [23] established the limitations of cell-like membrane systems which use only basic rules concerning the efficient solution of **NP**-complete problems. This result was generalized by Pérez–Jiménez et al. [20] obtaining the following two characterizations of the $\mathbf{P} \neq \mathbf{NP}$ relation by means of unsolvability results in polynomial time for **NP**–complete problems by families of cell–like recognizer membrane systems using only basic rules.

Theorem 1. *The following propositions are equivalent:*

1. $\mathbf{P} \neq \mathbf{NP}$.
2. *There exists an* $\mathbf{NP}$*–complete decision problem unsolvable in polynomial time by a family cell–like recognizer membrane systems using only basic rules.*
3. *Each* $\mathbf{NP}$*–complete decision problem is unsolvable in polynomial time by a family of cell–like recognizer membrane systems using only basic rules.*

Let us denote by $\mathcal{RB}$ the class of cell–like recognizer membrane systems using only basic rules. From the constructive proof given in [22], we deduce the following result characterizing the standard complexity class $\mathbf{P}$.

Theorem 2. $\mathbf{P} = \mathbf{PMC}_{\mathcal{RB}}$.

5 Recognizer cell–like membrane systems with active membranes

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.

Definition 7. A recognizer cell–like membrane system with active membranes is a recognizer cell–like membrane system (Π, Σ, i_Π) where the rules of the associated P system are of the following forms (H being the set of labels of Π):

1. $[\, a \to \omega\,]_h^\alpha$ for $h \in H$, $\alpha \in \{+, -, 0\}$, $a \in \Sigma$, $\omega \in \Sigma^*$: An object a within a membrane labelled with h and polarity α, evolves to a multiset ω.
2. $a\,[\ \,]_h^{\alpha_1} \to [\, b\,]_h^{\alpha_2}$ for $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Sigma$: An object from the region immediately outside a membrane labelled with h is introduced in this membrane, possibly transformed into another object, and simultaneously, the polarity of the membrane can be changed.
3. $[\, a\,]_h^{\alpha_1} \to b\,[\ \,]_h^{\alpha_2}$ for $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Sigma$: An object is sent out from membrane labelled with h to the region immediately outside, possibly transformed into another object, and simultaneously, the polarity of the membrane can be changed.
4. $[\, a\,]_h^\alpha \to b$ for $h \in H$, $\alpha \in \{+, -, 0\}$, $a, b \in \Sigma$: A membrane labelled with h is dissolved in reaction with an object. The skin is never dissolved.
5. $[\, a\,]_h^{\alpha_1} \to [\, b\,]_h^{\alpha_2}\ [\, c\,]_h^{\alpha_3}$ for $h \in H$, $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $a, b, c \in \Sigma$: An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects and their polarities.
6. $[\ [\ ]_{h_1}^{\alpha_1} \ldots [\ ]_{h_k}^{\alpha_1}\ \ [\ ]_{h_{k+1}}^{\alpha_2} \ldots [\ ]_{h_m}^{\alpha_2}\ ]_{h_0}^{\alpha_0} \quad \to \quad [\ [\ ]_{h_1}^{\alpha_3} \ldots [\ ]_{h_k}^{\alpha_3}\ ]_{h_0}^{\alpha_5}\ \ [\ [\ ]_{h_{k+1}}^{\alpha_4} \ldots [\ ]_{h_m}^{\alpha_4}\ ]_{h_0}^{\alpha_6}$, for $k \geq 1$, $m > k$, $h_i \in H$ for $0 \leq i \leq m$, and $\alpha_1, \ldots, \alpha_6 \in \{+, -, 0\}$, with $\{\alpha_1, \alpha_2\} = \{+, -\}$. These are division rules for non–elementary membranes. If the membrane with label h_0 contains other membranes than those with labels $h_1, \ldots, h_m$, then they must have neutral charge in order to make this rule applicable; these membranes and their contents are duplicated and placed in both new copies of the membrane h_0. Besides, every object in region h_0, as well as all membranes and objects placed inside membranes $h_1, \ldots, h_m$, 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 a membrane is dissolved, its content (multiset and internal membranes) is left free in the surrounding region.
- If at the same time a membrane labelled by h is divided by a rule of type (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 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)-(e).

Let us denote by $\mathcal{AM}$ the class of recognizer P systems with active membranes using 2-division. Different polynomial time solutions for **NP**–complete problems have been obtained using this class of cell–like recognizer membrane systems: *Knapsack* ([14]), *Subset Sum* ([13]), *Partition* ([4]), *SAT* ([19]), *Clique* ([1]), *Bin Packing* ([15]), and *CAP* ([16]).

Having in mind that the complexity class $\mathbf{PMC}_{\mathcal{AM}}$ is closed under complement and polynomial time reductions we have the following result.

Proposition 3. $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$, *and* $\mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$.

The complexity class $\mathbf{PMC}_{\mathcal{AM}}$ does not seem precise enough to describe classical complexity classes below **NP**. Therefore, it is challenging to investigate weaker variants of cell–like membrane systems able to characterize classical complexity classes.

In [2] universality has been achieved by removing the polarization of membranes from P systems with active membranes but allowing the change of membrane labels.

Several efficient solutions to **NP**–complete problems have been obtained within the following variant of membrane systems with active membranes:

- P systems using 2–division for elementary membranes, without cooperation, without priorities, without label changing, but using only two electrical charges (A. Alhazov [2], A. Riscos [21]).
- P systems using 2–division for elementary membranes, without cooperation, without priorities, without label changing, without polarizations, but using bi–stable catalysts (M.J. Pérez and F.J Romero [17]).
- P systems without polarizations, without cooperation, without priorities, without label changing, without division, but using three types of membrane rules: separation, merging and release (L. Pan et al. [7]).
- P systems with separation rules instead of division rules, in two different cases: (a) using polarizations and separation rules; and (b) without polarizations, but using separation rules with change of membrane labels (L. Pan and T.O. Ishdorj [8]).

It is possible to obtain polynomial time solutions to **NP**–complete problems through cell–like recognizer membrane systems with active membranes using 2-division for elementary membranes. But, what happens if we remove polarizations? We denote by $\mathcal{AM}^0$ the class of this kind of recognizer P systems.

Question: What is exactly the class of decision problems solvable in polynomial time by families of systems belonging to $\mathcal{AM}^0$?

We denote by $\mathcal{AM}^0(\alpha,\beta)$, where $\alpha \in \{-d,+d\}$ and $\beta \in \{-ne,+ne\}$, the class of all cell–like recognizer P systems with polarizationless active membranes such that: (a) if $\alpha = +d$ (resp. $\alpha = -d$) then dissolution rules are permitted (resp. forbidden); and (b) if $\beta = +ne$ (resp. $\beta = -ne$) then division rules for elementary and non–elementary (resp. only division rules for elementary) membranes are permitted.

Proposition 4. *For each* $\alpha \in \{-d,+d\}$ *and* $\beta \in \{-ne,+ne\}$ *we have:*

(1) $\mathbf{PMC}_{\mathcal{AM}^0(\alpha,\beta)} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^0(\alpha,\beta)}$.

(2) $\mathbf{PMC}_{\mathcal{AM}^0(\alpha,-ne)} \subseteq \mathbf{PMC}_{\mathcal{AM}^0(\alpha,+ne)}$.
(3) $\mathbf{PMC}^*_{\mathcal{AM}^0(\alpha,-ne)} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^0(\alpha,+ne)}$.
(4) $\mathbf{PMC}_{\mathcal{AM}^0(-d,\beta)} \subseteq \mathbf{PMC}_{\mathcal{AM}^0(+d,\beta)}$.
(5) $\mathbf{PMC}^*_{\mathcal{AM}^0(-d,\beta)} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^0(+d,\beta)}$.

In the framework of recognizer P systems with membrane division but without using polarizations it has been shown a surprising role of the dissolution rules, as it makes the difference between efficiency and non–efficiency for P systems with membrane division and without polarization ([3, 5]).

Theorem 3. *We have the following:*

(1) $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(-d,\beta)} = \mathbf{PMC}^*_{\mathcal{AM}^0(-d,\beta)}$, *for each* $\beta \in \{-ne, +ne\}$.
(2) $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^0(+d,+ne)}$.
(3) $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\mathcal{AM}^0(+d,+ne)}$.

6 Tissue–like recognizer membrane systems with active membranes

In this section we consider computational devices inspired from the cell inter–communication in tissues, and adding the ingredient of cell division rules of the same form as in cell–like membrane systems with active membranes, but without using polarizations.

In these systems, the rules are used in the non-deterministic maximally parallel way, but we suppose that when a cell is divided, its interaction with other cells or with the environment is blocked; that is, if a division rule is used for dividing a cell, then this cell does not participate in any other rule, for division or communication. The set of communication rules implicitely provides the graph associated with the system through the labels of the membranes. The cells obtained by division have the same labels as the mother cell, hence the rules to be used for evolving them or their objects are inherited.

Definition 8. A *tissue–like membrane system with active membranes* is a tuple

$$\Pi = (\Gamma, \Sigma, \mathcal{M}_1, \ldots, \mathcal{M}_p, E, R, i_{in}),$$

where:

1. $p \geq 1$ (the initial degree of the system; the system contains p cells, labelled with $1, 2, \ldots, p$);
2. Γ is the working alphabet containing two distinguished objects *YES* and *NO*;
3. Σ is an (input) alphabet strictly contained in Γ.
4. $\mathcal{M}_1, \ldots, \mathcal{M}_p$ are multisets over $\Gamma - \Sigma$, describing the objects placed in the cells of the system (we suppose that at least one copy of *YES* and *NO* is in some of these multisets);
5. $E \subseteq \Gamma$ is the set of objects present in the environment in arbitrary many copies each (the objects *YES* and *NO* are not present in E);
6. R is a finite set of *developmental rules*, of the following forms:
   (a) $(i, u/v, j)$, for $i, j \in \{0, 1, 2, \ldots, p\}, i \neq j$, and $u, v \in \Gamma^*$; $1, 2, \ldots, p$ identify the cells of the system, 0 is the environment: When applying a rule $(i, x/y, j)$, the objects of the multiset represented by u are sent from region i to region j and simultaneously the objects of the multiset v are sent from region j to region i;
   (b) $[\,a\,]_i \rightarrow [\,b\,]_i[\,c\,]_i$, where $i \in \{1, 2, \ldots, p\}$ and $a, b, c \in \Gamma$: Under the influence of object a, the cell with label i is divided in two cells with the same label; in the first copy the object a is replaced by b, in the second copy the object a is replaced by c; all other objects are replicated and copies of them are placed in the two new cells.

7. $i_{in} \in \{1, \ldots, n\}$ is the label of the input membrane.

Let m be a multiset over Σ. The *initial configuration of Π with input m* is the tuple $(\mathcal{M}_1, \ldots, \mathcal{M}_{i_{in}} \cup m, \ldots, \mathcal{M}_p)$.

The rules of a tissue–like membrane system as above are used in the non-deterministic maximally parallel manner as customary in membrane computing. In each step, we apply a set of rules which is maximal (no further rule can be added), with the following important restriction: if a cell is divided, then the division rule is the only one which is applied for that cell in that step, its objects do not participate in any communication rule.

The computation starts from the initial configuration and proceeds as defined above; only halting computations give a result, and the result is given by the presence of a distinguished object in the environment.

Definition 9. A tissue–like membrane system with active membranes Π is a recognizer system if: (a) all computations halt; and (b) in every computation of Π, either a copy of the object *YES* or a copy of the object *NO* (but not both) is sent into the environment.

We say that $\mathcal{C}$ is an accepting (rejecting) computation if the object *YES* (respectively, the object *NO*) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$.

We denote by $\mathcal{TR}$ the class of tissue–like recognizer membrane systems with active membranes.

In order to present the concept of uniform solvability in the framework of membrane systems as above, we define the concept of polynomial encoding from a language in a family of tissue–like recognizer membrane systems with active membranes.

Definition 10. Let L be a language, and $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ a family of tissue–like recognizer membrane systems of $\mathcal{TR}$. A polynomial encoding of L in $\mathbf{\Pi}$ is a pair (cod, s) of polynomial-time computable functions whose domain is L, and for each $u \in L$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$.

In the present paper we are interested in the computing efficiency. That is why we have introduced a variant of tissue–like systems with membrane division.

Next we define the concept of solvability in this new framework and in a similar way to Definition 4.

Definition 11. We say that 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 tissue–like recognizer membrane systems with active membranes, and we denote this by $X \in \mathbf{PMC}_{\mathcal{TR}}$, if the following is true:

- The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines.
- There exists a polynomial encoding (cod, s) from I_X to $\mathbf{\Pi}$ such that the family $\mathbf{\Pi}$ is polynomially bounded, sound and complete with regard to (X, cod, s).

We also have the class $\mathbf{PMC}_{\mathcal{TR}}$ is closed under polynomial–time reduction and complement.

We have said nothing about the way the computations proceed; in particular, they can be non-deterministic, as standard in membrane computing. It is important however to remark that the systems always stop and they always send out an object which is the correct answer to the instance of the problem that they are processing.

This natural extension of tissue P systems provides the possibility of solving `SAT` in polynomial time, in a confluent way: at precise times, one of the objects YES, NO is sent to the environment, giving the answer to the question whether the input propositional formula is satisfiable. We refer to [11] for a more detailed presentation of the following design.

Let us consider a propositional formula $\varphi = C_1 \wedge \cdots \wedge C_m$, consisting of m clauses $C_j = y_{j,1} \vee \cdots \vee y_{j,k_j}$, where $y_{j,i} \in \{x_l, \neg x_l \mid 1 \leq l \leq n\}$ (there are used n variables). Without loss of generality, we may assume that no clause contains two occurrences of some x_i or two occurrences

of some $\neg x_i$ (the formula is not redundant at the level of clauses), or both x_i and $\neg x_i$ (otherwise such a clause is trivially satisfiable, hence can be removed).

We consider the family $\mathbf{\Pi} = \{\Pi(\langle n, m\rangle) : n, m \in \mathbf{N}\}$ of tissue–like recognizer membrane systems, being $\langle n, m\rangle = \frac{(n+m)\cdot(n+m+1)}{2} + n$.

The tissue–like recognizer membrane system

$$\Pi(\langle n, m\rangle) = (\Gamma(\langle n, m\rangle), \Sigma(\langle n, m\rangle), \mathcal{M}_1, \mathcal{M}_2, E(\langle n, m\rangle), R(\langle n, m\rangle), i_{in})$$

will process all Boolean formulae in conjunctive normal form with n variables and m clauses, and is defined as follows:

$$\begin{aligned}
\Gamma(\langle n, m\rangle) &= \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{r_i \mid 1 \leq i \leq m\} \\
&\cup \{T_{i,1}, F_{i,1} \mid 1 \leq i \leq n\} \cup \{T'_{i,j}, F'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m+1\} \\
&\cup \{s_{i,j}, s'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\} \\
&\cup \{b_i \mid 1 \leq i \leq 3n+m+1\} \cup \{c_i \mid 1 \leq i \leq n+1\} \\
&\cup \{d_i \mid 1 \leq i \leq 3n+nm+m+2\} \cup \{e_i \mid 1 \leq i \leq 3n+nm+m+4\} \\
&\cup \{f, g, YES, NO\}, \\
\Sigma(\langle n, m\rangle) &= \{s_{i,j}, s'_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}, \\
\mathcal{M}_1 &= YES\ NO\ b_1 c_1 d_1 e_1, \\
\mathcal{M}_2 &= f g a_1 a_2 \ldots a_n, \\
E(\langle n, m\rangle) &= \Gamma(\langle n, m\rangle) - \{YES, NO\}, \\
i_{in} &= 2,
\end{aligned}$$

and the following rules.

1. $[\, a_i \,]_2 \rightarrow [\, T_{i,1} \,]_2\, [\, F_{i,1} \,]_2$, for all $i = 1, 2, \ldots, n$.
2. $(1, b_i / b_{i+1}^2, 0)$, for all $i = 1, 2, \ldots, n+1$.
3. $(1, c_i / c_{i+1}^2, 0)$, for all $i = 1, 2, \ldots, n+1$.
4. $(1, d_i / d_{i+1}^2, 0)$, for all $i = 1, 2, \ldots, n+1$.
5. $(1, e_i / e_{i+1}, 0)$, for all $i = 1, 2, \ldots, 3n + nm + m + 3$.
6. $(1, b_{n+1} c_{n+1} / f, 2)$.
7. $(1, d_{n+1} / g, 2)$.
8. $(2, c_{n+1} T_{i,1} / c_{n+1} T'_{i,1}, 0)$.
9. $(2, c_{n+1} F_{i,1} / c_{n+1} F'_{i,1}, 0)$, for each $i = 1, 2, \ldots, n$.
10. $(2, T'_{i,j} / t_i T'_{i,j+1}, 0)$.
11. $(2, F'_{i,j} / f_i F'_{i,j+1}, 0)$, for each $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.
12. $(2, b_i / b_{i+1}, 0)$.
13. $(2, d_i / d_{i+1}, 0)$, for all $i = n+1, \ldots, (n+1) + (2n+m) - 1$.
14. $(2, b_{3n+m+1} t_i s_{i,j} / b_{3n+m+1} r_j, 0)$.
15. $(2, b_{3n+m+1} f_i s'_{i,j} / b_{3n+m+1} r_j, 0)$, for all $1 \leq i \leq n$ and $1 \leq j \leq m$.
16. $(2, d_i / d_{i+1}, 0)$, for all $i = 3n+m+1, \ldots, (3n+m+1) + nm - 1$.
17. $(2, d_{3n+nm+m+i} r_i / d_{3n+nm+m+i+1}, 0)$, for all $i = 1, 2, \ldots, m$.
18. $(2, d_{3n+nm+2m+1} / f\ YES, 1)$.
19. $(2, YES / \lambda, 0)$.
20. $(1, e_{3n+nm+2m+2} f\ NO / \lambda, 2)$.
21. $(2, NO / \lambda, 0)$.

6.1 An overview of the computation

Membrane 2 is repeatedly divided, each time expanding one object a_i, corresponding to a variable x_i, into $T_{i,1}$ and $F_{i,1}$, corresponding to the values *true* and *false* which this variable may assume. In this way, in n steps, we get 2^n cells with label 2, each one containing one of the 2^n possible truth assignments for the n variables. The objects f, g are duplicated, hence a copy of each of them will appear in each cell.

In parallel with the operation of dividing cell 2, the counters b_i, c_i, d_i, e_i from cell 1 grow their subscripts. In each step, the number of copies of objects of the first three types is doubled, hence after n steps we get 2^n copies of b_{n+1}, c_{n+1} and d_{n+1}. Objects b_i will check which clauses are satisfied by a given truth assignment, objects c_i are used in order to multiply the number of copies of t_i, f_i as we will see immediately, d_i are used to check whether there is at least one truth assignment which satisfies all clauses, and if such an assignment does not exists, then e_i will be used in order to produce the object NO at the end of the computation.

In step $n+1$, the counters $b_{n+1}, c_{n+1}, d_{n+1}$ are brought in cells with label 2, in exchange of f and g. Because we have 2^n copies of each object of these types and 2^n cells 2, each one containing exactly one copy of f and one of g, due to the maximality of the parallelism of using the rules, each cell 2 gets precisely one copy of each of $b_{n+1}, c_{n+1}, d_{n+1}$. Note that cells 2 cannot divide anymore, because the objects a_i were exhausted.

In the presence of c_{n+1}, the objects $T_{i,1}, F_{i,1}$ get primed, which initiates the possibility of introducing m copies of each t_i and f_i in each cell 2. As we have m clauses, then in order to check their values for a given truth assignment, we need for each clause one set of objects encoding the values of all variables. Note that this phase needs $2n$ steps for priming the objects $T_{i,1}, F_{i,1}$ – for each object we need one step, because we have only one copy of c_{n+1} available – then m further steps for each $T'_{i,1}, F'_{i,1}$; all these steps are done in parallel, but for the last primed $T_{i,1}, F_{i,1}$ we have to continue m steps after the $2n$ necessary for priming. Thus, the total number of steps performed in this process is $2n+m$.

In parallel with the previous operations, the counters b_i and d_i increase their subscripts, until reaching the value $3n+m+1$. This is done in all cells 2 at the same time. Simultaneosly, e_i increases its subscript in cell 1.

In the presence of b_{3n+m+1} – and not before – we check the values assumed by clauses for the truth assignments from each cell 2. We have only one copy of b_{3n+m+1} in each cell, hence we need at most nm steps for this: each clause contains at most n literals, and we have m clauses. In parallel, d increases the subscript, until reaching the value $3n+nm+m+1$.

In each cell with label 2 we check whether or not all clauses are satisfied by the corresponding truth assignment. For each clause which is satisfied, we increase by one the subscript of d, hence the subscript reaches the value $3n+nm+2m+1$ if and only if all clauses are satisfied.

If one of the truth assignments from a cell 2 has satisfied all clauses, then we reach $d_{3n+nm+2m+1}$, which is sent to cell 1 in exchange of the objects YES and f.

In the next step, the object YES leaves the system, signaling the fact that the formula is satisfiable. In cell 1, the counter e will increase one more step its subscript, but after that it will remain unchanged – it can leave cell 1 only in the presence of f, but this object was already moved to cell 2.

If the counter e reaches the subscript $3n+nm+2m+2$ and the object f is still in cell 1, then the object NO can be moved to a cell 2, randomly chosen, and from there it exits the system, signaling that the formula is not satisfiable.

6.2 Some formal details

We consider the polynomial encoding (cod, s) from $I_{\mathtt{SAT}}$ to $\mathbf{\Pi}$, defined as follows: if the formula φ is an instance of `SAT` with size parameters n (number of variables) and m (number of clauses), then

$s(\varphi) = \langle n, m \rangle$ and $cod(\varphi)$ is the set

$$\bigcup_{1 \leq i \leq n, 1 \leq j \leq m, 1 \leq r \leq k_j} \{s_{i,j} \mid y_{j,r} = x_i\} \cup \{s'_{i,j} \mid y_{j,r} = \neg x_i, \}$$

That is, in the multiset $cod(\varphi)$ we replace each variable x_i from each clause C_j with $s_{i,j}$ and each negated variable $\neg x_i$ from each clause C_j with $s'_{i,j}$, then we remove all parentheses and connectives. In this way we pass from φ to $cod(\varphi)$ in a number of steps which is linear with respect to $n \cdot m$.

The presented family of tissue–like recognizer membrane systems is polynomially uniform by Turing machines, because the definition of the family is done in a recursive manner from a given instance of SAT, in particular from the constants n (number of variables) and m (number of clauses). Furthermore the required resources to build the element $\Pi(\langle n, m \rangle)$ of the family are the following:

- Size of the working alphabet: $5nm + 17n + 4m + 12 \in O((max\{n, m\})^2)$.
- Number of membranes: $2 \in \Theta(1)$.
- $|\mathcal{M}_1| + |\mathcal{M}_2| = n + 8 \in \Theta(n)$.
- Maximum length of the rules: 3.

Finally, we can prove, using a formal description of the computations, that the family $\mathbf{\Pi}$ is sound and complete with regard to $(\mathtt{SAT}, cod, s)$. The number of steps of the computations is polynomial in terms of n and m: the answer YES is sent out in step $3n + nm + 2m + 2$, while the answer NO is sent out in step $3n + nm + 2m + 4$.

From the above we deduce the following results:

Theorem 4.

1. $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{TR}}$.
2. $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{TR}}$, *and* $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{TR}}$.

7 Conclusions

In this paper, we have presented cell–like (inspired from the structure of the cell) and tissue–like (inspired from the cell inter–communication in tissues) recognizer membrane systems as computational devices specially suitable to attack the efficient solvability of computationally hard problems.

In that new framework, two characterizations of the relation $\mathbf{P} = \mathbf{NP}$ have been described through the solvability of $\mathbf{NP}$–complete problems by a family of cell–like recognizer membrane systems using only basic rules.

The main contribution of this paper is to present, in the framework of tissue–like recognizer membrane systems, a formal definition of the cellular complexity class $\mathbf{PMC}_{\mathcal{TR}}$.

Recognizer membrane systems with active membranes have been studied in the variants cell–like and tissue–like, and an efficient and uniform solution to the satisfiability problem by tissue–like recognizer membrane systems with cell division has been presented.

Acknowledgement

The authors wish to acknowledge the support of the project TIN2005-09345-C04-01 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds.

References

1. A. Alhazov, C. Martín–Vide, L. Pan. Solving graph problems by P systems with restricted elementary active membranes. In N. Jonoska, Gh. Păun, and G. Rozenberg, editors, *Aspects of Molecular Computing: Essays Dedicated to Tom Head, on the Occasion of His 70th Birthday*, pages 1–22. Lecture Notes in Computer Science, 2950, 2004.
2. A. Alhazov, L. Pan, Gh. Păun. Trading polarizations for labels in P systems with active membranes. *Acta Informaticae*, 41(2-3):111–144, 2004.
3. A. Alhazov and M.J. Pérez–Jiménez. Uniform solution of `QSAT` using polarizationless active membranes. In M.A. Gutiérrez–Naranjo et al., editors, *Proceedings of the Fourth Brainstorming Week on Membrane Computing (vol. I)*, pages 29–40. Report RGNC 02/2006, 2006.
4. M.A. Gutiérrez–Naranjo, M.J. Pérez–Jiménez, and A. Riscos–Núñez. A fast P system for finding a balanced 2-partition. *Soft Computing*, 9(9):673–678, September 2005.
5. M.A. Gutiérrez–Naranjo, M.J. Pérez–Jiménez, A. Riscos–Núñez, and F.J. Romero–Campero. On the power of dissolution in P systems with active membranes. *Lecture Notes in Computer Science*, 3850:224–240, 2006.
6. T. Head, M. Yamamura, and S. Gal. Aqueous computing: writing on molecules. *Proceedings of the Congress on Evolutionary Computation 1999*, pages 1006–1010. IEEE Service Center, 1999. Piscataway, NJ.
7. L. Pan, A. Alhazov, and T.O. Ishdorj. Further remarks on P systems with active membranes, separation, merging, and release rules. In Gh. Păun et al., editors, *Proceedings of the Second Brainstorming Week on Membrane Computing*, pages 316–324. Report RGNC 01/04, 2004.
8. L. Pan and T.O. Ishdorj. P systems with active membranes and separation rules. *Journal of Universal Computer Science*, 10(5):630–649, 2004.
9. Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000, and Turku Center for Computer Science - TUCS Report 208, November 1998, `www.tucs.fi`
10. Gh. Păun. Membrane Computing. An introduction. Springer-Verlag, Berlin, 2002.
11. Gh. Păun, M.J. Pérez–Jiménez, and A. Riscos–Núñez. Tissue P systems with cell division. In Gh. Păun et al., editors, *Proceedings of the Second Brainstorming Week on Membrane Computing*, pages 380–386. Report RGNC 01/04, 2004.
12. M.J. Pérez–Jiménez. An approach to computational complexity in Membrane Computing. In G. Mauri, Gh. Păun, M.J. Pérez–Jiménez, Gr. Rozenberg, and A. Salomaa, editors, *Membrane Computing, 5th International Workshop, WMC5, Revised Selected and Invited Papers*, pages 85–109. Lecture Notes in Computer Science, 3365, 2005.
13. M.J. Pérez–Jiménez and A. Riscos–Núñez. Solving the Subset-Sum problem by P systems with active membranes. *New Generation Computing*, 23(4):367–384, 2005.
14. M.J. Pérez–Jiménez and A. Riscos–Núñez. A linear-time solution to the knapsack problem using P systems with active membranes. In C. Martín-Vide, Gh. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, pages 248–266. Lecture Notes in Computer Science, 2933, 2004.
15. M.J. Pérez–Jiménez and F.J. Romero–Campero. An efficient family of P systems for packing items into bins. *Journal of Universal Computer Science*, 10(5):650–670, 2004.
16. M.J. Pérez–Jiménez and F.J. Romero–Campero. Attacking the Common Algorithmic Problem by recognizer P systems. In M. Margenstern, editor, *Machines, Computations and Universality, MCU'2004*, pages 304–315. Lecture Notes in Computer Science, 3354, 2005.
17. M.J. Pérez–Jiménez and F.J. Romero–Campero. Trading polarizations for bi-stable catalysts in P systems with active membranes. In G. Mauri, Gh. Păun, M.J. Pérez–Jiménez, Gr. Rozenberg, and A. Salomaa, editors, *Membrane Computing, 5th International Workshop, WMC5, Revised Selected and Invited Papers*, pages 373–388. Lecture Notes in Computer Science, 3365, 2005.
18. M.J. Pérez–Jiménez, A. Romero–Jiménez, and F. Sancho–Caparrini. Teoría de la Complejidad en modelos de computación con membranas. Ed. Kronos, Sevilla, 2002.

19. M.J. Pérez–Jiménez, A. Romero–Jiménez, and F. Sancho–Caparrini. Complexity classes in cellular computing with membranes. *Natural Computing*, 2(3):265–285, 2003.
20. M.J. Pérez–Jiménez, A. Romero–Jiménez, and F. Sancho–Caparrini. The P versus NP problem through cellular computing with membranes. In N. Jonoska, Gh. Păun, and G. Rozenberg, editors, *Aspects of Molecular Computing: Essays Dedicated to Tom Head, on the Occasion of His 70th Birthday*, pages 338–352. Lecture Notes in Computer Science, 2950, 2004.
21. A. Riscos–Núñez. *Cellular Programming: efficient resolution of NP–complete numerical problems*. PhD. Thesis, University of Seville, Spain, 2004.
22. A. Romero–Jiménez. *Complexity and Universality in Cellular computing models*. PhD. Thesis, University of Seville, Spain, 2003.
23. C. Zandron, C. Ferreti, and G. Mauri. Solving NP-Complete Problems Using P Systems with Active Membranes. In I. Antoniou, C.S. Calude and M.J. Dinneen, editors, *Unconventional Models of Computation, UMC'2K*, pages 289–301. Springer-Verlag, 2000.