# A Framework for Complexity Classes in Membrane Computing

Agustín Riscos-Núñez

*Dpt. Computer Science and Artificial Intelligence*
*University of Sevilla*
*Sevilla, Spain*

**Abstract**

The purpose of the present work is to give a general idea about the existing results and open problems concerning the study of complexity classes within the membrane computing framework. To this aim, membrane systems (seen as computing devices) are briefly introduced, providing the basic definition and summarizing the key ideas, trying to cover the various approaches that are under investigation in this area – of course, special attention is paid to the study of complexity classes. The paper concludes with some final remarks that hint the reasons why this field (as well as other unconventional models of computation) is attracting the attention of a growing community.

*Keywords:* Membrane Computing, P systems, Complexity Classes

## 1 Introduction

Natural Computing is an attempt to infer from Nature ideas and/or intuitions that might yield new paradigms in Computation Theory (or, more generally speaking, in Information Processing). The earlier examples of this field are Genetic Algorithms and Neural Networks, and later on also DNA Computing and Membrane Computing appeared.

Membrane Computing arises as a new model of computation, inspired by the way that cells are structured into vesicles, and abstracting the chemical reactions taking place inside them. Since 1998, when the seminal paper [13] was first circulated through the Internet, the area has experienced a huge development, giving rise to an extensive literature and drawing the attention of mathematicians, computer scientists, and biologists, among others.

Work supported by project TIN2005-09345-C04-01 of Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and by the Project of Excelence TIC-581 of the Junta de Andalucía.

At the moment there are multiple approaches that are being subject to investigation, studying e.g. the universality (computational power) of membrane computing models, their efficiency (from a computational complexity point of view), or their potential usefulness for modeling biological phenomena.

Interested readers are referred to the monograph [14] for a complete "handbook" on the area. Another basic reference is the P systems web page (visit [21]), where a comprehensive and up-to-date bibliography can be found, as well as information about related conferences and events, etc.

## 2 Membrane Computing Preliminaries

Membrane Computing field was originated in the paper [13], where Gheorghe Păun presents a computational device called *super-cell system* (from now on referred to as membrane system or P system), consisting basically of a hierarchical arrangement of regions or membranes (called a membrane structure) within which are located multisets of objects that *evolve* according to some rules that are applied in a maximal parallel way. More precisely, let us recall the generic *syntax* of the model:

**Definition 2.1** A P system of degree $p \geq 1$ is a tuple

$$\Pi = (\Gamma, \mu, w_1, \ldots, w_p, R, i_0),$$

where:

- $\Gamma$ is a finite alphabet (known as *working alphabet*).
- $\mu$ is a *membrane structure*, composed by $p$ membranes labelled in a one-to-one manner using the natural numbers from 1 to $p$ (in some cases, one can include explicitly an alphabet of labels).
- $w_l$ (for every $1 \leq l \leq p$) is a finite *multiset* over $\Gamma$, eventually empty, associated with membrane $l$.
- $R$ is a finite set of *rules* associated with the membranes of the system. The existing types of rules will be discussed below.
- $i_0$ is a natural number between 1 and $p$ which indicates the label of a distinguished membrane in the system (that will be called *output membrane*).

**Note 1** *Recall that a multiset of objects is a set where the elements are allowed to be repeated. Every string over an alphabet determines, in a natural way, a multiset and, reciprocally, every multiset can be represented by a string. For the sake of simplicity, multisets are usually represented as strings over the working alphabet of the system,* $\Gamma$.

P systems have a very flexible initial definition, and therefore it has been possible to investigate up to now a huge number of different models. There are two key syntactical features where the adjustments are usually performed, yielding the corresponding classes of P systems: membranes and rules.

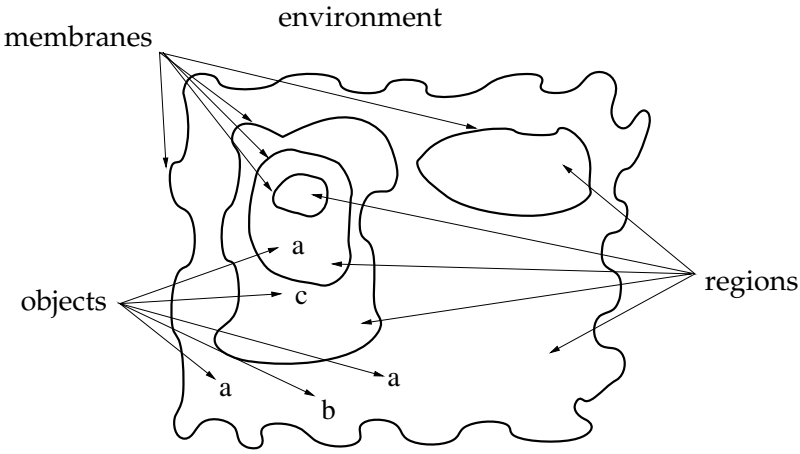


Fig. 1. A "picture" of a membrane system

**Membranes** may have associated with them not only labels, but also electrical charges, which affect their behavior (e.g. P systems with active membranes, addressed in Section 3). On the other hand, objects in $\Gamma$ may play the role of labels, if we allow them to attach or de-attach to/from membranes (see e.g. [4]). There are many other possibilities, like for instance considering several degrees of permeability for each membrane, indicating whether objects are allowed to cross it or not. Interested readers are referred to [14] for a detailed overview on most of the existing models.

**Rules** are doubtless one of the basic ingredients characterizing a model. In the original paper rules were of a rewriting nature, that is, a multiset $u$ occurring in a region is replaced by a multiset $v$, including *target indicators* for the objects in $v$ indicating whether such objects remain in the same region or they move to an adjacent one. One can also restrict to purely communication rules (symport/antiport model, see e.g. [12]), and still the obtained model is very interesting and powerful.

Finally, let us mention that in the first model it was possible to dissolve a membrane after the application of some rules. Later on models allowing the addition of new membranes (by dividing existing ones or creating membranes from objects) have also been investigated.

In order to describe the *semantics* of the model, we make use of the concept of *configuration* of a P system, which can be informally defined simply as an instantaneous description of the system, expressing the current membrane structure including labels, charges, or any other ingredients associated with them, and the contents of the regions of the system.

We assume the existence of a global clock, in such a way that the application of the rules takes from a configuration to the next one, performing a *transition step*. A sequence of such steps, starting from the initial configuration of the system (initial membrane structure and initial contents) is called a *computation* of the P system.

Rules are applied in a maximal parallel way, that is, rules are applied in parallel (moreover, a rule can be applied more than once provided that there are enough objects available) and it is forbidden that a multiset of objects remains inactive during a step if it can trigger a rule. In case that there are several combinations of rules that can be applied, then the system chooses nondetermistically. In addition, priorities among rules were considered in the original definition, imposing that a rule cannot be applied if there is another rule of a higher priority that is also applicable. Nowadays, most of the models do not include priorities, but there are cases where the parallelism is bounded, either at the level of membranes or for the whole system (see e.g. [3], [5] or [9]).

It is important to recall that, apart from stating the syntax and semantics of a P system model, it is also necessary to fix how the evolution of the system will be interpreted. In general, there exist four approaches:

- P systems can be seen as *generative* devices, if we have a fixed initial configuration and we collect the outputs of all the nondeterministic computations.
- The evolution of a P system can be seen as a *computing* process (i.e. the system "computes" a function), if we consider that a certain number, $n$, is encoded somehow in the initial configuration, and we take the cardinality of the output multiset as the result of the computation.
- P systems can be seen as *decision* devices, provided that we have some special objects *yes* and *no* such that we can check their presence in the output in order to decide whether the given input was accepted by the P system or not.
- One of the most promising ways to use P systems is for *simulating* purposes, in the sense that we do not consider the standard *input* $\rightarrow$ *computation* $\rightarrow$ *output* scheme, but we focus instead on the evolution process itself.

One of the earlier topics addressed in the literature is the quest for universality, mainly from a formal language theory point of view. That is, in order to show the "power" of the models, the most used reference was initially the Chomsky hierarchy of languages, and also proofs within the framework of automata theory (keep in mind that P systems are a machine-oriented computing model). Later on also simulations of recursive functions by means of P systems have been performed [16]. An interesting result in this line was obtained by O.H. Ibarra in [8], where an infinite hierarchy (on the computational power) is obtained.

Concerning the simulating approach, we can say that it is the closer one to Biology, as in a biological system where the concept of starting a computation and wait for a halting configuration does not make sense, one can try instead to investigate the behavior of the system. Some work has been done already in this line (see e.g. [2,11]), which is actually attracting an increasing attention from the membrane computing community.

## 3 Complexity Theory in Membrane Computing

We shall focus here in the approach considering membrane systems as devices able to *solve* decision problems, paying special attention to the resources needed to perform these solutions.

Formally, we deal with P systems that are able to receive an input, encoding the instance of the problem that is to be solved, and they produce a boolean output (of type Yes/No). Moreover, we impose a number of restrictions in the definition of a "membrane solution", requiring for instance that the designed systems are *confluent*, in the sense that despite the nondeterminism of the system, all computations yield the same answer. All these considerations lead to the definition of *recognizer P systems* (see e.g. [15] or [18]).

**Definition 3.1** A *recognizer P system* is a P system with input membrane and external output such that:

- $Yes, No \in \Gamma$ (working alphabet).
- all the computations halt.
- for every computation, one symbol $Yes$ or one symbol $No$ (but not both) is sent out (and in the last step of the computation).

Before going on, it is worth mentioning that in the literature there is a distinction between the solutions where a specific P system is given for each instance of the problem (*semi-uniform* solutions) and those designs where one system is able to deal with several instances of the problem (*uniform* solutions). Some examples of the earlier semi-uniform designs can be found in [10,20], and a complete discussion about the different complexity classes obtained in both approaches can be found in [15]. We recall here the definition of the uniform computational complexity framework:

**Definition 3.2** Let $\mathcal{R}$ be a class of recognizer P systems. A decision problem $X = (I_X, \theta_X)$ is *solvable in polynomial time* by a family $F_X = (\Pi(n))_{n\in\mathbb{N}^+}$, of $\mathcal{R}$, (we denote this by $X \in \mathbf{PMC}_{\mathcal{R}}$), if

- $F_X$ is polynomially uniform by Turing machines, i.e. there exists a deterministic Turing machine such that, given $n \in \mathbb{N}$, constructs the system $\Pi(n)$ in a polynomial time.
- There exists a pair $(cod, s)$ of polynomial time computable functions $cod : I_X \to \bigcup_{n\in\mathbb{N}^+} I_{\Pi(n)}$ and $s : I_X \to \mathbb{N}^+$ such that for every $u \in I_X$ we have $cod(u) \in I_{\Pi(s(u))}$, and

(i) $F_X$ 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 the system $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps.

(ii) $F_X$ is sound, with regard to $(X, cod, s)$; that is, for each $u \in I_X$ it is verified that if there exists a computation of the system $\Pi(s(u))$ with input $cod(u)$ that is accepting, then $\theta_X(u) = 1$.

(iii) $F_X$ 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 the system $\Pi(s(u))$ with input $cod(u)$ is accepting.

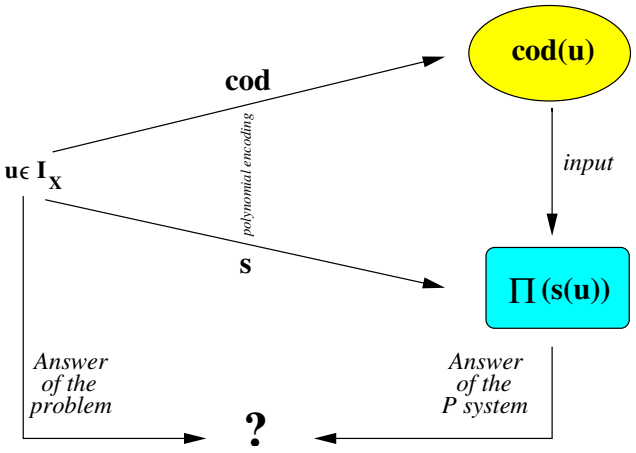


Fig. 2. For uniform families of P systems the diagram commutes

The intrinsic maximal parallelism of P systems can be exploited to produce a speed-up for solutions to **NP**–complete problems. In order to achieve this, the model needs several ingredients, among them the ability to generate an exponential workspace in polynomial time. One of the models that fulfills these requirements is the so-called *active membranes* model, which works with membranes having electrical charges associated with them, and allows membrane division rules. More precisely,

**Definition 3.3** A *P system with active membranes* is a tuple

$$\Pi = (\Gamma, H, \mu, w_1, \ldots, w_p, R),$$

where:

- $\Gamma$ is a finite alphabet (the working alphabet).
- $H$ is a finite set of labels.
- $\mu$ is a membrane structure of degree $p$, with labels from $H$. Membranes have electrical charges, that can be either neutral (0), positive (+) or negative (−).
- $w_1, \ldots, w_p$ are multisets over $\Gamma$, which express the initial contents of membranes from $\mu$.
- $R$ is a finite set of *developmental rules* of the following types:

(a) $[a \to v]_l^\alpha$, where $a \in \Gamma$, $v \in \Gamma^*$, $\alpha \in \{0, +, -\}$ (object evolution rules). It is an internal rule, that does not affect the outside of membrane $l$, nor its polarity. Its execution produces the substitution of an object $a$ by a multiset $v$, inside a membrane labelled by $l$ and with electrical charge $\alpha$.

(*b*) $[a]_l^\alpha \to b[\,]_l^\beta$, where $a, b \in \Gamma$, $\alpha, \beta \in \{0, +, -\}$ (send-out communication rules). An element $a$ can get out of a membrane labelled by $l$ and with electrical charge $\alpha$, possibly transformed in a new object $b$ and, simultaneously, modify the polarity of this membrane (from $\alpha$ to $\beta$), but keeping the same label.

(*c*) $a[\,]_l^\alpha \to [b]_l^\beta$, where $a, b \in \Gamma$, $\alpha, \beta \in \{0, +, -\}$ (send-in communication rules). Analogous to the previous one, but moving the object in region $l$, instead of getting it out.

(*d*) $[a]_l^\alpha \to b$, where $a, b \in \Gamma$, $\alpha \in \{0, +, -\}$, $l \neq skin$ (dissolution rules). Its execution produces the transformation of an object $a$ inside a membrane labelled by $l$ and with electrical charge $\alpha$ into a new object $b$. Also, simultaneously, the membrane is dissolved.

(*e*) $[a]_l^\alpha \to [b]_l^\beta [c]_l^\gamma$, where $a, b, c \in \Gamma$, $\alpha, \beta, \gamma \in \{0, +, -\}$, $l \neq skin$ (2-division rules for elementary membranes). An object $a$ inside a membrane labelled by $l$ and with electrical charge $\alpha$ can cause the membrane to divide into two new ones, with the same label but, eventually, with different electrical charges, in such a way that the object $a$ is transformed independently in each membrane into new objects $b$ and $c$ (apart from these objects, the contents of the resulting membranes are identical).

Rules are applied according to the following principles (informal semantics of P systems with active membranes):

- The rules are used as usual in the framework of membrane computing; that is, in a maximal parallel way, with the restrictions indicated below.
- The rules associated with a label are used for all membranes with this label. At one step, different rules can be applied to different membranes with the same label, but one membrane can only be the subject of *at most* one rule of types (*b*)-(*e*).
- If a membrane is dissolved, its content (multiset and interior membranes) becomes part of the immediately external membrane (more precisely, of the closest predecessor which is not dissolved).
- All elements which are not specified in any of the operations to apply remain unchanged.
- A division rule can be applied to a membrane and, at the same time, some evolution rules can be applied to some objects inside that membrane. In this case, we can suppose that "first" the evolution rules are used, changing the objects, and "after that" the division takes place, introducing copies of the results of the evolutions in the two newly generated membranes (but keeping in mind that all these processes take place in the same step of computation).
- The skin membrane can never divide nor be dissolved, although, as the rest of membranes, it can be electrically charged.

We denote by $\mathcal{AM}$ the class of recognizer P systems with active membranes using only binary division for elementary membranes. One can then consider the class of problems solvable (uniformly) in polynomial time by P systems with active

membranes, $\mathbf{PMC}_{\mathcal{AM}}$ (see e.g. [18]). This class is closed under polynomial-time reduction and under complement, so one deduces[3] the following inclusion

**Theorem 3.4** $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$.

However, this model proved to be too powerful when Petr Sosík found a polynomial-time semi-uniform solution for a **PSPACE**–complete problem in [19]. There is a series of papers investigating which are the features that are actually responsible for the efficiency (in time) of the model, trying to bring light on the boundaries between classical complexity classes such as **P**, **NP** and **PSPACE**. Which are the "ingredients" that play a relevant role in the power of the model, concerning the efficiency of problem solving?

We can mention here for instance one of the open problems in this direction, known as the **P**-conjecture: *the complexity class associated with P systems with active membranes without electrical charges is exactly* **P**. A partial affirmative answer to this conjecture was found in [6], where it is proved that the class of problems solvable by polarizationless P systems with active membranes **without** dissolution rules and with division rules for non-elementary membranes is exactly **P**. Nevertheless, also a partial negative answer was found in [1], where it is shown (by constructing a uniform solution to QSAT) that the complexity class associated with polarizationless P systems with active membranes **with** dissolution rules and with division rules for non-elementary membranes contains **PSPACE**.

Thus, it turns out that dissolution rules play a key role in determining the power to solve problems of the class of polarizationless P systems with division rules for non-elementary membranes. It remains open to investigate the case when only 2-division rules for elementary membranes are allowed, or one can also redefine different division rules for non-elementary membranes, but we will not get into details here. A good summary of the results obtained up to now in this line can be found at [7].

## 4 Final Remarks

We have given here a very brief overview on complexity theory in membrane computing models. It is important to mention that it is often hard to carry out formal studies within this unconventional machine-oriented model. Indeed, there does not exist up to now any standard procedure to address, given a design of a family of P systems and a problem, the formal verification proving that this family *solves* the problem, in the sense of Definition 3.2. The main difficulty usually comes from the fact that there may exist many possible computation paths, and many objects and membranes evolving in a maximally parallel way.

Besides, another interesting remark is that working with multisets leads to unary encoding, and one has to keep in mind this when we speak about linear or polynomial time (number of parallel steps) on the "size" of the input.

[3] The reader is referred again to [18] for details.

We believe that the current investigations on the borderline of tractability in membrane computing can bring light on the **P** versus **NP** conjecture. Actually, Pérez–Jiménez et al. [17] have characterized the $\mathbf{P} \neq \mathbf{NP}$ relation by means of unsolvability results in polynomial time for **NP**–complete problems by families of recognizer membrane systems using only basic rules. This research might be useful also when looking for implementations (either biological or via computer simulations), as it points out which are the more interesting features to be captured in the simulated model.

# References


[1] Alhazov, A., and M.J. Pérez–Jiménez. *Uniform solution to QSAT using polarizationless active membranes*, Proc. Fourth Brainstorming Week on Membrane Computing, Volume I, Fénix Editora, Sevilla, 2006, 29–40. URL: http://www.gcn.us.es/4BWMC/4BWMC_proceedings.htm.

[2] Ardelean, I., and D. Besozzi, *New proposals for the formalization of membrane proteins*, Proc. Second Brainstorming Week on Membrane Computing, Sevilla, 2004, 53–59. URL: http://www.gcn.us.es/Brain/Volumen.htm.

[3] Bernardini, F. et al., *On P systems with bounded parallelism*, Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005), IEEE Computer Society, 2005, 399-406.

[4] Cavaliere, M., A. Riscos–Núñez, R. Brijder and G. Rozenberg, *Membrane systems with proteins embedded in membranes*, Theoretical Computer Science, in press.

[5] Csuhaj-Varjú, E., R. Freund and D. Sburlan, *Modeling dynamical parallelism in bio-systems*, Lecture Notes in Computer Science **4361** (2006), 330–351.

[6] Gutiérrez–Naranjo, M.A., 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** (2006), 224–240.

[7] Gutiérrez–Naranjo, M.A., M.J. Pérez–Jiménez, A. Riscos–Núñez and F.J. Romero–Campero, *Computational efficiency of dissolution rules in membrane systems*, Int. Journal of Computer Mathematics, **83** 7 (2006), 593–611.

[8] Ibarra, O.H. *The number of membranes matters*, Lecture Notes in Computer Science **2933** (2004), 218–231.

[9] Ishdorj, T.-O. *Minimal Parallelism for Polarizationless P Systems*, Lecture Notes in Computer Science **4287** (2006), 17–32.

[10] Krishna S.N. "Languages of P Systems: Computability and Complexity". Ph.D. Thesis, Indian Institute of Technology Madras, 2001.

[11] Manca, V., L. Bianco and F. Fontana *Evolution and oscillation in P systems: applications to biological phenomena*, Lecture Notes in Computer Science **3365** (2005), 63–84.

[12] Păun, A., and Gh. Păun, *The Power of Communication: P Systems with Symport/Antiport*. New Generation Computing, **20** 3 (2002), 295–305.

[13] Păun, Gh. *Computing with membranes*, Journal of Computer and System Sciences **61** 1 (2000), 108–143, previously circulated as a Turku Center for Computer Science-TUCS Report No **208** (1998), URL: http://www.tucs.fi.

[14] Păun, Gh. "Membrane Computing. An introduction". Springer-Verlag, Berlin, 2002.

[15] Pérez–Jiménez, M.J. *An approach to computational complexity in Membrane Computing*, Lecture Notes in Computer Science **3365** (2005), 85–109.

[16] Pérez–Jiménez, M.J., and A. Romero–Jiménez, *Simulating Turing machines by P systems with external output*, Fundamenta Informaticae **49** 1-3 (2002), 273–287.

[17] Pérez–Jiménez, M.J., A. Romero–Jiménez and F. Sancho–Caparrini, *The **P** versus **NP** problem through cellular computing with membranes*, Lecture Notes in Computer Science **2950** (2004), 338–352.

[18] Riscos–Núñez, A. "Cellular programming: efficient resolution of numerical **NP**-complete problems". Ph.D. thesis, University of Seville, Spain, 2004.

[19] Sosík, P. *Solving a **PSPACE**-Complete Problem by P Systems with Active Membranes*, Proc. Brainstorming Week on Membrane Computing, Tarragona, 2003, 305–312.

[20] Zandron, C., G. Mauri and C. Ferretti, *Solving **NP**-complete problems using P systems with active membranes*, in I. Antoniou, C.S. Calude y M.J. Dinneen (eds), "Unconventional Models of Computation. UMC'2K". Springer-Verlag, 2001, 289–301.

[21] The P systems web page. URL: http://psystems.disco.unimib.it.