Complete Problems for a Variant of P Systems with Active Membranes

Antonio E. Porreca, Alberto Leporati, Giancarlo Mauri, Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
`{porreca,leporati,mauri,zandron}@disco.unimib.it`

Summary. We identify a family of decision problems that are hard for some complexity classes defined in terms of P systems with active membranes working in polynomial time. Furthermore, we prove the *completeness* of these problems in the case where the systems are equipped with a form of priority that linearly orders their rules. Finally, we highlight some possible connections with open problems related to the computational complexity of P systems with active membranes.

1 Introduction

Membrane systems, usually called *P systems*, are computing devices inspired by the internal working of biological cells [6]. The main feature of P systems is a structure of membranes dividing the space into regions, inside which multisets of objects describe the molecular environment. A set of rules describe how the molecules (and often the membranes themselves) evolve during the computation; usually, the rules are applied in a maximally parallel way, i.e., each component of the P system *must* be subject to a rule during each computation step, if a suitable rule exists. When multiple rules may be applied to an object or membrane, one of them is nondeterministically chosen. The computation, starting from an initial configuration, proceeds until no further rule can be applied. For an introduction on membrane computing we refer the reader to [8, 9], and for the latest information to the P Systems Webpage [16], where an extensive bibliography on the topic can be found.

Families of P systems can be used as language recognizers, by associating with each input string (or to each input length) a P system; this association is subject to a uniformity condition (i.e., it must be computed by a Turing machine operating in polynomial time). The constructed P systems can then accept or reject, thus deciding the membership of strings to the language. The computational complexity of recognizer P systems with *active membranes* [7], where the membranes themselves play an important role during the computation, has been subject to extensive investigation, due to their ability of solving **NP**-complete and even **PSPACE**-complete problems [11] in polynomial time: this efficiency is due to the possibility of creating in polynomial time an exponential number of membranes, which then evolve in parallel using *membrane division* (a process found in nature, e.g., in cell mitosis).

In this paper we investigate the existence of complete problems for complexity classes defined in terms of P systems with active membranes. Some classes are known to possess them, simply because they coincide with classes defined in terms of Turing machines and inherit their complete problems: for instance, polynomial-time P systems with active membranes without membrane division characterize **P** [15], while they caracterize **PSPACE** if all kinds of rule are available [12]. Our approach is, however, more general; we exhibit problems (inspired by analogous ones for Turing machines) that are hard for *every* polynomial-time complexity class defined in terms of P systems with active membranes, independently of what rules are available, and complete for several of them if a restricted form of priority among rules is used.

The remainder of this paper is organized as follows. In Section 2 we recall the notion of recognizer P systems with active membranes and how to measure their time complexity. Section 3 describes the bounded acceptance problem; in particular, the variant for nondeterministic Turing machines is considered and its **NP**-completeness is proved. In Section 4 we introduce the bounded acceptance problem for P systems and prove its hardness for all polynomial-time complexity classes; we also prove that the problem is complete if we add a linear ordering on the rules of the P systems.

2 Definitions

We begin by recalling the definition of P systems with active membranes.

Definition 1. *A* P system with active membranes *of the initial degree $m \geq 1$ is a tuple*

$$\Pi = (\Gamma, \Lambda, \mu, w_1, \ldots, w_m, R)$$

where:

- *Γ is a finite alphabet of symbols, also called* objects*;*
- *Λ is a finite set of labels for the membranes;*
- *μ is a membrane structure (i.e., a rooted unordered tree) consisting of m membranes enumerated by $1, \ldots, m$; furthermore, each membrane is labeled by an element of Λ, not necessarily in a one-to-one way;*
- *$w_1, \ldots, w_m$ are strings over Γ, describing the multisets of objects placed in the m initial regions of μ;*
- *R is a finite set of rules.*

Each membrane possesses a further attribute, named *polarization* or *electrical charge*, which is either neutral (represented by 0), positive (+) or negative (−) and it is assumed to be initially neutral.

The rules are of the following kinds:

- *Object evolution rules*, of the form $[a \to w]_h^\alpha$
  They can be applied inside a membrane labeled by h, having polarization α and containing an occurrence of the object a; the object a is rewritten into the multiset w (i.e., a is removed from the multiset in h and replaced by every object in w).
- *Send-in communication rules*, of the form $a\,[\;]_h^\alpha \to [b]_h^\beta$
  They can be applied to a membrane labeled by h, having polarization α and such that the external region contains an occurrence of the object a; the object a is sent into h becoming b and, simultaneously, the polarization of h is changed to β.
- *Send-out communication rules*, of the form $[a]_h^\alpha \to [\;]_h^\beta\, b$
  They can be applied to a membrane labeled by h, having polarization α and containing an occurrence of the object a; the object a is sent out from h to the outside region becoming b and, simultaneously, the polarization of h is changed to β.
- *Dissolution rules*, of the form $[a]_h^\alpha \to b$
  They can be applied to a membrane labeled by h, having polarization α and containing an occurrence of the object a; the membrane h is dissolved and its contents are left in the surrounding region unaltered, except that an occurrence of a becomes b.
- *Elementary division rules*, of the form $[a]_h^\alpha \to [b]_h^\beta\,[c]_h^\gamma$
  They can be applied to a membrane labeled by h, having polarization α, containing an occurrence of the object a but having no other membrane inside; the membrane is divided into two membranes having label h and polarizations β and γ; the object a is replaced, respectively, by b and c while the other objects in the initial multiset are copied to both membranes.
- *Non-elementary division rules*, of the form

  $$\big[[\;]_{h_1}^+ \cdots [\;]_{h_k}^+ [\;]_{h_{k+1}}^- \cdots [\;]_{h_n}^-\big]_h^\alpha \to \big[[\;]_{h_1}^\delta \cdots [\;]_{h_k}^\delta\big]_h^\beta \big[[\;]_{h_{k+1}}^\varepsilon \cdots [\;]_{h_n}^\varepsilon\big]_h^\gamma$$

  They can be applied to a membrane labeled by h, having polarization α, containing the positively charged membranes $h_1, \ldots, h_k$, the negatively charged membranes $h_{k+1}, \ldots, h_n$, and possibly some neutral membranes. The membrane h is divided into two copies having polarization β and γ, respectively; the positive children are placed inside the former, their polarizations changed to δ, while the negative ones are placed inside the latter, their polarizations changed to ε. Any neutral membrane inside h is duplicated and placed inside both copies.

A *configuration* in a P system with active membranes is described by its current membrane structure, together with its polarizations and the multisets of objects contained in its regions. The *initial configuration* is given by μ, all membranes having polarization 0 and the initial contents of the membranes being $w_1, \ldots, w_m$. A *computation step* changes the current configuration according to the following principles:

- Each object and each membrane can be subject to only one rule during a computation step.
- The rules are applied in a *maximally parallel way*: each object which appears on the left-hand side of applicable evolution, communication, dissolution or elementary division rules must be subject to exactly one of them; the same holds for each membrane which can be involved in a communication, dissolution or division rule. The only objects and membranes which remain unchanged are those associated with no rule, or with unapplicable rules.
- When more than one rule can be applied to an object or membrane, the actual rule to be applied is chosen nondeterministically; hence, in general, multiple configurations can be reached from the current one.
- When dissolution or division rules are applied to a membrane, the multiset of objects to be released outside or copied is the one resulting *after* all evolution rules have been applied.
- The skin membrane cannot be divided, nor it can be dissolved. Furthermore, every object which is sent out from the skin membrane cannot be brought in again.

A *halting computation* $\boldsymbol{\mathcal{C}}$ of a P system Π is a finite sequence of configurations $(\mathcal{C}_0, \ldots, \mathcal{C}_k)$, where $\mathcal{C}_0$ is the initial configuration of Π, every $\mathcal{C}_{i+1}$ can be reached from $\mathcal{C}_i$ according to the principles just described, and no further configuration can be reached from $\mathcal{C}_k$ (i.e., no rule can be applied). P systems might also perform *non-halting* computations; in this case, we have infinite sequences $\boldsymbol{\mathcal{C}} = (\mathcal{C}_i : i \in \mathbb{N})$ of successive configurations.

We can use families of P systems with active membranes as language recognizers, thus allowing us to solve decision problems.

Definition 2. *A* recognizer P system with active membranes Π *has an alphabet containing two distinguished objects yes and no, used to signal acceptance and rejection respectively; every computation of Π is halting and exactly one object among yes, no is sent out from the skin membrane during each computation.*

In what follows we will only consider *confluent* recognizer P systems with active membranes, in which all computations starting from the initial configuration agree on the result.

Definition 3. *Let $L \subseteq \Sigma^\star$ be a language and let $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ be a family of recognizer P systems. We say that $\boldsymbol{\Pi}$* decides *L, in symbols $L(\boldsymbol{\Pi}) = L$, when for each $x \in \Sigma^\star$, the result of Π_x is acceptance iff $x \in L$.*

Usually some uniformity condition, inspired by those applied to families of Boolean circuits, is imposed on families of P systems. Two different notions of uniformity have been considered in the literature; they are defined as follows.

Definition 4. *A family of P systems* $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ *is said to be* semi-uniform *when the mapping* $x \mapsto \Pi_x$ *can be computed in polynomial time, with respect to* $|x|$*, by a deterministic Turing machine.*

Definition 5. *A family of P systems* $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ *is said to be* uniform *when there exist two polynomial-time Turing machines* M_1 *and* M_2 *such that, for each* $n \in \mathbb{N}$ *and each* $x \in \Sigma^n$

- M_1*, on input* 1^n *(the unary representation of the length of* x*), outputs the description of a P system* Π_n *with a distinguished input membrane;*
- M_2*, on input* x*, outputs a multiset* w_x *(an encoding of* x*);*
- Π_x *is* Π_n *with* w_x *added to the multiset located inside its input membrane.*

In other words, the P system Π_x *associated with string* x *consists of two parts; one of them,* Π_n*, is common for all strings of length* $|x| = n$ *(in particular, the membrane structure and the set of rules fall into this category), and the other (the input multiset* w_x *for* Π_n*) is specific to* x*. The two parts are constructed independently and, only as the last step,* w_x *is inserted in* Π_n*.*

Time complexity classes for P systems [10] are defined as usual, by restricting the amount of time available for deciding a language. By $\mathbf{PMC}_{\mathcal{D}}$ (resp., $\mathbf{PMC}^\star_{\mathcal{D}}$) we denote the class of languages which can be decided by uniform (resp., semi-uniform) families $\boldsymbol{\Pi}$ of confluent P systems of class $\mathcal{D}$ (e.g., $\mathcal{AM}$ denotes the class of P systems with active membranes) where each computation of $\Pi_x \in \boldsymbol{\Pi}$ halts in polynomial time with respect to $|x|$.

3 The bounded acceptance problem for Turing machines

A classic decision problem related to each class of automata is the *acceptance* or *prediction problem*: given an automaton A and a string x over its input alphabet, is it the case that A accepts x? The difficulty of this problem varies according to the class of automata: for instance, if we consider finite automata it is in **P**, while it is **PSPACE**-complete for linear bounded automata [3, p. 265]; one of the first and most important results of computability theory asserts that the problem is undecidable for Turing machines [13].

Even when the problem is not solvable, one can often devise an easier version by limiting the amount of resources allocated. Consider the variant defined as follows.

Definition 6. *The* bounded acceptance problem $\mathrm{BAP}_{\mathrm{NTM}}$ *for nondeterministic Turing machines is the set of triples* $(N, x, 1^t)$ *where*

- N *is a "reasonable" encoding [2] of a nondeterministic Turing machine;*
- x *is a string over the input alphabet of* N*;*
- 1^t *is the unary encoding of a natural number* t*;*

such that N *accepts the string* x *within* t *computation steps.*

$\mathrm{BAP}_{\mathrm{NTM}}$ is an interesting problem because it is very easy to prove its **NP**-completeness.

Proposition 1. $\mathrm{BAP}_{\mathrm{NTM}} \in \mathbf{NP}$.

Proof (a variant of the proof of Theorem 2.9 in [1]). A nondeterministic Turing machine N', given $(N, x, 1^t)$ as input, can easily perform a step-by-step simulation of a computation of N on x with a polynomial slowdown, making a nondeterministic choice every time N does; simultaneously, on another tape, N' counts the number of simulated computation steps. If N accepts before the counter reaches $t + 1$, then N' halts and accepts; if, on the contrary, the counter reaches $t + 1$ without N having accepted, then N' halts and reject. Since the input t is given in unary notation, the whole simulation runs in polynomial time, and N' has an accepting computation on $(N, x, 1^t)$ iff N has an accepting computation on x. $\square$

Proposition 2. $\mathrm{BAP}_{\mathrm{NTM}}$ *is* **NP***-hard.*

Proof. Let $L \in \mathbf{NP}$ be a language. Then, there exists a nondeterministic Turing machine N deciding L in polynomial time $p(n)$.

Let $R\colon \Sigma^\star \to \Sigma^\star$ be defined by $R(x) = (N, x, 1^{p(|x|)})$; the function R can be computed by a *deterministic* Turing machine operating in polynomial time, since N does not depend on x (hence it can be output in constant time) and $1^{p(|x|)}$ can be easily computed in polynomial time given a unary encoding of $|x|$ as input (which can be easily obtained from x itself).

Now let $x \in \Sigma^\star$. Clearly $x \in L$ iff N accepts x; since N runs in $p(n)$ steps, this is equivalent to saying that $R(x) = (N, x, 1^{p(|x|)}) \in \mathrm{BAP}_{\mathrm{NTM}}$: thus, R is a polynomial-time reduction of L to $\mathrm{BAP}_{\mathrm{NTM}}$. $\square$

4 The bounded acceptance problem for P systems

In this section we discuss a variant of the bounded acceptance problem in the setting of P systems with active membranes.

Definition 7. *Let $\mathcal{D}$ be any subclass of P systems with active membranes; the* bounded acceptance problems $\mathrm{BAP}_{\mathcal{D}}$ *and* $\mathrm{BAP}^\star_{\mathcal{D}}$ *for uniform and semi-uniform families of P systems of class $\mathcal{D}$ are, respectively, the set of triples $(\Pi, w, 1^t)$ and the set of pairs $(\Pi, 1^t)$ where*

- *Π is a P system in the class $\mathcal{D}$;*
- *w is a multiset over the alphabet of Π;*
- 1^t *is the unary encoding of a natural number t;*

such that Π accepts within t steps when the multiset w is placed inside its input membrane (resp., Π accepts within t steps).

It is easy to show that every problem in $\mathbf{PMC}_{\mathcal{D}}$ can be reduced to $\mathrm{BAP}_{\mathcal{D}}$ (and every problem in $\mathbf{PMC}^\star_{\mathcal{D}}$ to $\mathrm{BAP}^\star_{\mathcal{D}}$) by modifying the proof of Proposition 2.

Lemma 1. $\mathrm{BAP}_{\mathcal{D}}$ *is* $\mathbf{PMC}_{\mathcal{D}}$*-hard, and* $\mathrm{BAP}^{\star}_{\mathcal{D}}$ *is* $\mathbf{PMC}^{\star}_{\mathcal{D}}$*-hard.*

Proof. Let $L \in \mathbf{PMC}_{\mathcal{D}}$ be a language. Then, there exists a uniform family of P systems $\boldsymbol{\Pi} = \{\Pi_x : x \in \Sigma^{\star}\}$ deciding L in polynomial time $p(n)$; the family $\boldsymbol{\Pi}$ is constructed by polynomial-time Turing machines M_1 (for the portion that only depends on the length of the input) and M_2 (for the input multiset, which depends on the whole string).

Let $R\colon \Sigma^{\star} \to \Sigma^{\star}$ be defined by $R(x) = \big(M_1(|x|), M_2(x), 1^{p(|x|)}\big)$; clearly, the function R can be computed in polynomial time by a deterministic Turing machine. For each $x \in \Sigma^{\star}$ we have, by hypothesis, $x \in L$ iff the P system Π_x, constructed by combining $M_1(|x|)$ with the multiset $M_2(x)$, accepts within $p(|x|)$ steps; in other words, $x \in L$ iff $R(x) \in \mathrm{BAP}_{\mathcal{D}}$. But then R is a polynomial-time reduction from L to $\mathrm{BAP}_{\mathcal{D}}$.

The proof is completely analogous in the semi-uniform case: the only difference is that we have no input multiset. $\square$

Unfortunately, proving that the BAP for standard P systems with active membranes belongs to $\mathbf{PMC}_{\mathcal{D}}$ or $\mathbf{PMC}^{\star}_{\mathcal{D}}$ (if this is indeed the case) seems to be much more difficult than proving the same result for Turing machines. The reason is that it is very easy to equip a Turing machine with a clock that halts the machine after a certain number of steps: there are few "moving parts" to stop, namely the tape heads, and the transition function controls them all simultaneously. On the other hand, in a P system the global state is distributed, and each rule only affects a small part of it; the computation is also, in general, nondeterministic. It is then very difficult to stop every region of the P system, at least in an efficient way (i.e., with a polynomial slowdown): the intuitive idea of halting the system by dissolving every membrane when a certain period of time has passed also fails, since the dissolution rule may enter into a conflict with the original rules, and hence might never be applied.

4.1 Completeness of BAP in a special case

By adding a limited form of priority among rules to P systems with active membranes (both with and without polarizations), we are able to find a class $\mathcal{D}$ such that a variant of $\mathrm{BAP}_{\mathcal{D}}$ (resp., $\mathrm{BAP}^{\star}_{\mathcal{D}}$) is complete for $\mathbf{PMC}_{\mathcal{D}}$ (resp., $\mathbf{PMC}^{\star}_{\mathcal{D}}$).

Definition 8. *A* P system with active membranes *(with or without polarizations)* and linear priority

$$\Pi = (\Gamma, \Lambda, \mu, w_1, \ldots, w_m, R, \leq)$$

is a P system $(\Gamma, \Lambda, \mu, w_1, \ldots, w_m, R)$ *with active membranes equipped with a linear (i.e., total) order* $\leq$ *over* R*, which is used as follows: whenever a nondeterministic choice between two rules* $r_1 < r_2$ *has to be made during the computation, the rule which is actually applied is* r_2*.*

Confluent families of P systems with active membranes and polarizations using only elementary membrane division are powerful enough to solve **NP**-complete problems in polynomial time [15]. Due to confluence, we know that any possible computation (i.e., any sequence of nondeterministic, maximally paralles choices of rules) yields the correct result: in particular, the computation obtained by linearly ordering the set of rules in an arbitrary way. This proves that introducing priorities does not affect the ability of solving **NP**-complete problems in polynomial time. On the other hand, one can simulate polynomial-time P systems with active membranes in polynomial space, and this simulation can be easily extended to include priorities without increasing the space requirements[12]. In other words, the class of problems decided in polynomial time by P systems with active membranes (with polarizations) and linear priority is located between **NP** and **PSPACE**.

Proposition 3. $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{D}} \subseteq \mathbf{PMC}^{\star}_{\mathcal{D}} \subseteq \mathbf{PSPACE}$. □

The existence of complete problems for $\mathbf{PMC}_{\mathcal{D}}$ and $\mathbf{PMC}^{\star}_{\mathcal{D}}$ not only provides us with a possible way of proving $\mathbf{PMC}_{\mathcal{D}} \subseteq \mathbf{NP}$ and $\mathbf{PMC}^{\star}_{\mathcal{D}} \subseteq \mathbf{NP}$ by solving one of these problems via a polynomial-time nondeterministic Turing machines, but it also gives us another hint on the nature of this complexity class. It is a common assumption [5] that the the cumulative polynomial hierarchy **PH** (i.e., the union of all levels of the hierarchy), which is also located between **NP** and **PSPACE**, consists of infinitely many distinct levels. As a consequence, **PH** is conjectured not to have complete problems, since their existence would imply the *collapse* of the hierarchy, i.e., only finitely many levels would exist: we can then conjecture that $\mathbf{PMC}_{\mathcal{D}}$ and $\mathbf{PMC}^{\star}_{\mathcal{D}}$ differ from **PH** (or, for that matter, from any other class lacking complete problems).

On the other hand, if we consider *polarizationless* P systems with active membranes and elementary division only, we run into the so-called *P conjecture* [14, 4]: these systems are conjectured to solve only **P** problems in polynomial time, but a proof of this statement is still missing. We are able to exhibit a complete problem when linear priority among rules is assumed: solving it, or solving any new complete problem which could emerge in the future, via a polynomial-time deterministic Turing machine (if this is possible at all) might provide some insight for the P conjecture.

The bounded acceptance problem for P systems with active membranes and linear priority is hard for $\mathbf{PMC}_{\mathcal{D}}$ and $\mathbf{PMC}^{\star}_{\mathcal{D}}$ as described above, and its completeness can be proved as follows.

Theorem 1. *Let $\mathcal{D}$ be any subclass of P systems with active membranes (with or without polarizations) and linear priority, using at least evolution, communication and dissolution rules. Then* $\mathrm{BAP}_{\mathcal{D}}$ *and* $\mathrm{BAP}^{\star}_{\mathcal{D}}$ *are complete for* $\mathbf{PMC}_{\mathcal{D}}$ *and* $\mathbf{PMC}^{\star}_{\mathcal{D}}$ *respectively.*

Proof. We only have to prove membership in $\mathbf{PMC}_{\mathcal{D}}$ (we focus on the uniform version here, as the argument for the semi-uniform one is just a simplification of it). Given $(\Pi, w, 1^t)$, a polynomial-time Turing machine M_1 can construct a P system Π' which is identical to Π except for the following differences:

- The membrane structure is enclosed by two further membranes with new labels h_1 and h_0 (h_0 becomes the new skin membrane).
- The objects *yes* and *no* of Π are renamed as the two new symbols *yes*$'$ and *no*$'$ (they are also renamed in all rules involving them).
- Each of the original membranes of Π contains a timer object z_0, that evolves to the new objects $z_1, \ldots, z_t$ according to the rules

$$[z_0 \to z_1]_h^\alpha \qquad [z_1 \to z_2]_h^\alpha \qquad \cdots \qquad [z_{t-1} \to z_t]_h^\alpha \qquad [z_t]_h^\alpha \to \#$$

  for all $\alpha \in \{+, 0, -\}$ and for each original label h. These objects count up to t and then dissolve the membrane producing a new "junk" object. In the polarizationless case, the charges are simply omitted.
- The new membrane h_1 also contains a timer object z_0' with the rules

$$[z_0' \to z_1']_{h_1}^\alpha \qquad [z_1' \to z_2']_{h_1}^\alpha \qquad \cdots \qquad [z_t' \to z_{t+1}']_{h_1}^\alpha \qquad [z_{t+1}']_{h_1}^\alpha \to no$$

  The goal of these objects is to count up to $t+1$ and then dissolve h_1, producing the object *no*. The objects z_i' are also assumed to be different from any object in Π.
- The new membranes h_1 and h_0 are also involved in the following rules:

$$\begin{aligned} &[yes']_{h_1} \to yes & \qquad &[no']_{h_1} \to no \\ &[yes]_{h_0} \to [\,]_{h_0}\, yes & \qquad &[no]_{h_0} \to [\,]_{h_0}\, no \end{aligned}$$

- The priority among the original rules is left untouched, but all the new rules have a higher priority than them; the precise ordering among the new rules is arbitrary.

The Turing machine M_2 constructing the input multiset simply outputs the multiset w. Thus, we obtain a uniform family $\boldsymbol{\Pi}$ of P systems.

If the original P system Π sends out its output from its skin membrane within t steps, then either *yes*$'$ or *no*$'$ appears in membrane h_1 of Π'. In such a case, *yes* (resp., *no*) is first sent out to h_0 (by dissolution, hence interrupting the timer z') and then to the environment as the result of the computation, which in this case is clearly the same as Π. If, however, Π computes for more than t steps, then its membranes are all simultaneously dissolved in Π' in step $t+1$ (recall that the new rules have highest priority) and membrane h_1 produces a *no* object that becomes the result of the computation. Thus, the family $\boldsymbol{\Pi}$ solves $\mathrm{BAP}_{\mathcal{D}}$, and it does so in $O(t)$ steps, which is polynomial time with respect to the input size. □

5 Conclusions

We have shown that the bounded acceptance problem, that is, determining whether a given P systems accepts its input within a certain number of steps, is complete for the class of decision problems that can be solved in polynomial time by certain families of P systems. The P systems covered by the proof are recognizer P systems with active membranes, with or without polarizations, using at least evolution, communication and dissolution rules; additionally, it is required that a linear priority relation is defined on the rules of the system. The result holds both in the uniform and semi-uniform cases.

The classes of decision problems $\mathbf{PMC}_{\mathcal{D}}$ and $\mathbf{PMC}^{\star}_{\mathcal{D}}$ defined as above are easily shown to be located between **NP** and **PSPACE**: hence, the existence of decision problems for these classes might provide an useful tool in finding a tighter upper bound, as well as in differentiating them from classes, such as the cumulative polynomial hierarchy **PH**, that do not (apparently) possess complete problems.

An outstanding open question concerns the existence of complete problems for complexity classes defined in terms of *standard* (i.e., without priority) polynomial-time P systems with active membranes, particularly those without non-elementary division rules, as they still lack a characterization in terms of Turing machines; investigating this question in the polarizationless case might shed new light on the *P conjecture*.

Acknowledgements

This work was partially supported by the Italian project FIAR 2008 "Modelli di calcolo naturale e applicazioni alla systems biology".

References

1. S. Arora, B. Barak: *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.
2. J.L. Balcázar, J. Díaz, J. Gabarró: *Structural Complexity I.* Springer, 1995.
3. M.R. Garey, D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman & Co., 1979.
4. G. Mauri, M.J. Pérez-Jiménez, C. Zandron: On a Păun's conjecture in membrane systems. *Second International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2007* (J. Mira, J.R. Álvarez, eds.), volume 4527 of *Lecture Notes in Computer Science.* Springer, 2007, 180–192.
5. C.H. Papadimitriou: *Computational Complexity.* Addison-Wesley, 1993.
6. G. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 1, 61 (2000), 108–143.
7. G. Păun: P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, 6, 1 (2001), 75–90.
8. G. Păun, G. Rozenberg: A guide to membrane computing. *Theoretical Computer Science*, 287 (2002), 73–100.
9. G. Păun, G. Rozenberg, A. Salomaa (eds.): *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.
10. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2, 3 (2003), 265–285.

11. P. Sosík: The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing*, 2, 3 (2003), 287–298.
12. P. Sosík, A. Rodríguez-Patón: Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73, 1 (2007), 137–152.
13. A.M. Turing: On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42 (1936), 230–265.
14. D. Woods, N. Murphy, M.J. Pérez-Jiménez, A. Riscos-Núñez: Membrane dissolution and division in P. *Unconventional Computation, 8th International Conference, UC 2009* (C.S. Calude, J.F. Costa, N. Dershowitz, E. Freire, G. Rozenberg, eds.), volume 5715 of *Lecture Notes in Computer Science*. Springer, 2009, 262–276.
15. C. Zandron, C. Ferretti, G. Mauri: Solving NP-complete problems using P systems with active membranes. *Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Brussel, Belgium, 13–16 December 2000, Springer, 2001, 289–301.
16. *The P Systems Webpage*, `http://ppage.psystems.eu`.