Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 3, pp. 301-310

Introducing a Space Complexity Measure for P Systems

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

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

Received: April 5, 2009
Accepted: May 30, 2009

Abstract: We define space complexity classes in the framework of membrane computing, giving some initial results about their mutual relations and their connection with time complexity classes, and identifying some potentially interesting problems which require further research.
Keywords: membrane computing, complexity theory.

1 Introduction

Until now, research on the complexity theoretic aspects of membrane computing has mainly focused on the time resource. In particular, since the introduction of P systems with active membranes [5], various results concerning time complexity classes defined in terms of P systems with active membranes were given, comparing different classes obtained using various ingredients (such as, e.g., polarizations, dissolution, uniformity, etc.). Other works considered the comparisons between them and the usual complexity classes defined in terms of Turing machines, either from the point of view of time complexity [8, 3, 11], or space complexity classes [10, 1, 9].

Despite the vivid interest on this subject, up to now no investigations concerning space complexity classes defined in terms of P systems have been carried out in formal terms. Of course, the evident relation between time and space in P systems with active membranes is informally acknowledged: all results concerning solutions to ***NP***-complete problems are solved using an exponential workspace obtained in polynomial time. Nonetheless, there is no formal definition of space complexity classes for P systems and, as a consequence, no formal results concerning the relations between space and time.

In this paper, we make the first steps in this direction, first by defining the space requirements for a given P system on a specific computation, and then by formally defining space complexity classes for P systems. We will then give a first set of results concerning relations among complexity classes for P systems, some of them directly following from the definitions, and others which can be derived by considering space requirements of various solutions proposed in the literature which make use of P systems with active membranes.

In what follows we assume the reader is already familiar with the basic notions and the terminology underlying P systems. For a systematic introduction, we refer the reader to [6]. A survey and an up-to-date bibliography concerning P systems can be found at the web address `http://ppage.psystems.eu`.

The rest of the paper is organized as follows. In section 2 we give basic definitions for membrane systems which will be used throughout the rest of the paper. In section 3 we give formal definitions of space complexity classes in terms of P systems. In section 4 we present some results concerning such complexity classes, which follow immediately from the definitions, while in section 5 we present some results which can be obtained by considering known results for time complexity classes in the framework of P systems with active membranes. Section 6 concludes the paper by presenting some conjectures and open problems concerning space complexity.

2 Definitions

We begin by recalling the formal definition of P systems with active membranes and the usual process by which they are used to solve decision problems. Moreover, we recall the main definitions related to time complexity classes in this framework.

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

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

where

- Γ is a finite alphabet of symbols or objects;
- Λ is a finite set of labels;
- μ is a membrane structure (i.e. a rooted, unordered tree) of m membranes, labeled with elements of Λ; different membranes may be given the same label;
- $w_1, \ldots, w_m$ are multisets over Γ describing the initial contents of the m membranes in μ;
- R is a finite set of developmental rules.

The *polarization* of a membrane can be $+$ (positive), $-$ (negative) or 0 (neutral); each membrane is assumed to be initially neutral.

Developmental rules are of the following six kinds:

(a) *Object evolution rule* of the form $[a \to w]_h^\alpha$

It 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 the multiset w).

(b) *Communication rule* of the form $a\,[\,]_h^\alpha \to [b]_h^\beta$

It can be applied to a membrane labeled by h, having polarization α and such that the surrounding 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.

(c) *Communication rule* of the form $[a]_h^\alpha \to [\,]_h^\beta\, b$

It 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 surrounding region becoming b and, simultaneously, the polarization of h is changed to β.

(d) *Dissolution rule* of the form $[a]_h^\alpha \to b$

It 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 content is left in the surrounding region unaltered, except that an occurrence of a becomes b.

(e) *Elementary division rule* of the form $[a]_h^\alpha \to [b]_h^\beta\, [c]_h^\gamma$

It can be applied to an elementary membrane labeled by h, having polarization α and containing an occurrence of the object a; 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.

(f) *Non-elementary division rule* of the form

$$\left[\left[\,\right]^{+}_{h_1}\cdots\left[\,\right]^{+}_{h_k}\left[\,\right]^{-}_{h_{k+1}}\cdots\left[\,\right]^{-}_{h_n}\right]^{\alpha}_{h}\rightarrow\left[\left[\,\right]^{\delta}_{h_1}\cdots\left[\,\right]^{\delta}_{h_k}\right]^{\beta}_{h}\left[\left[\,\right]^{\varepsilon}_{h_{k+1}}\cdots\left[\,\right]^{\varepsilon}_{h_n}\right]^{\gamma}_{h}$$

It can be applied to a non-elementary membrane labeled by h, having polarization α, containing the positively charged membranes $h_1,\ldots,h_k$ and the negatively charged membranes $h_{k+1},\ldots,h_n$; no other non-neutral membrane may be contained in h. The membrane h is divided into two copies with polarization β and γ; 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.

Note that here we have used the original definition of division rules introduced in [5]. Afterwards, other papers have proposed several alternatives for non-elementary division rules; nonetheless, in this paper these variants are not considered.

A *configuration* of a P system with active membranes Π is given by a membrane structure and the multisets contained in its regions. In particular, the *initial configuration* is given by the membrane structure μ and the initial contents of its membranes $w_1,\ldots,w_m$. A computation step leads from a configuration to the next one according to the following principles:

- The developmental rules are applied in a *maximally parallel* way: when one or more rules can be applied to an object and/or membrane, then one of them *must* be applied. Notice that an object or membrane may remain inactive, even if it can trigger a rule, only when its use is inhibited by the application of another rule.
- Each object can be subject to only one rule during each step. Also membranes can be subject to only one rule, except that *any* number of object evolution rules can be applied inside them.
- When more than one rule can be applied to an object or membrane, then the one actually applied is chosen nondeterministically. Thus multiple, distinct configurations may be reachable by means of a computation step from a single configuration.
- When a dissolution or division rule is applied to a membrane, the multiset of objects to be released outside or copied is the one *after* any application of object evolution rules inside such membrane.
- The skin membrane cannot be divided or dissolved, nor any object can be sent in from the environment surrounding it (i.e. an object which leaves the skin membrane cannot be brought in again).

A sequence of configurations, each one reachable from the previous one by means of developmental rules applied according to the above principles, is called a *computation*. Due to nondeterminism, there may be multiple computations starting from the initial configuration, thus producing a computation tree. A computation halts when no further configuration can be reached, i.e. when no rule can be applied in a given configuration.

Families of *recogniser* P systems can be used to solve decision problems as follows.

Definition 2. Let Π be a P system whose alphabet contains two distinct objects *yes* and *no*, such that every computation of Π is halting and during each computation exactly one of the objects *yes*, *no* is sent out from the skin to signal acceptance or rejection. If all the computations of Π agree on the result, then Π is said to be *confluent*; if this is not necessarily the case, then it is said to be *non-confluent* and the global result is acceptance iff there exists an accepting computation.

Definition 3. Let $L\subseteq\Sigma^{\star}$ be a language, $\mathcal{D}$ a class of P systems and let $\boldsymbol{\Pi}=\{\Pi_x \mid x\in\Sigma^{\star}\}\subseteq\mathcal{D}$ be a family of P systems, either confluent or non-confluent. We say that $\boldsymbol{\Pi}$ *decides* L when, for each $x\in\Sigma^{\star}$, $x\in L$ iff Π_x accepts.

Complexity classes for P systems are defined by imposing a uniformity condition on $\boldsymbol{\Pi}$ and restricting the amount of time available for deciding a language.

Definition 4. Consider a language $L \subseteq \Sigma^\star$, a class of recogniser P systems $\mathcal{D}$, and let $f\colon \mathbb{N} \to \mathbb{N}$ be a proper complexity function. We say that L belongs to the complexity class $\boldsymbol{MC}^\star_{\mathcal{D}}(f)$ if and only if there exists a family of confluent P systems $\boldsymbol{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L such that

- $\boldsymbol{\Pi}$ is *semi-uniform*, i.e. there exists a deterministic Turing machine which, for each input $x \in \Sigma^\star$, constructs the P system Π_x in polynomial time;
- $\boldsymbol{\Pi}$ operates in time f, i.e. for each $x \in \Sigma^\star$, every computation of Π_x halts within $f(|x|)$ steps.

In particular, a language $L \subseteq \Sigma^\star$ belongs to the complexity class $\boldsymbol{PMC}^\star_{\mathcal{D}}$ iff there exists a semi-uniform family of confluent P systems $\boldsymbol{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L in polynomial time.

The analogous complexity classes for *non*-confluent P systems are denoted by $\boldsymbol{NMC}^\star_{\mathcal{D}}(f)$ and $\boldsymbol{NPMC}^\star_{\mathcal{D}}$.

Another set of complexity classes is defined in terms of *uniform* families of recogniser P systems:

Definition 5. Consider a language $L \subseteq \Sigma^\star$, a class of recogniser P systems $\mathcal{D}$, and let $f\colon \mathbb{N} \to \mathbb{N}$ be a proper complexity function. We say that L belongs to the complexity class $\boldsymbol{MC}_{\mathcal{D}}(f)$ if and only if there exists a family of confluent P systems $\boldsymbol{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L such that

- $\boldsymbol{\Pi}$ is *uniform*, i.e. for each $x \in \Sigma^\star$ deciding whether $x \in L$ is performed as follows: first, a polynomial-time deterministic Turing machine, given the length $n = |x|$ as a unary integer, constructs a P system Π_n with a distinguished input membrane; then, another polynomial-time DTM computes a coding of the string x as a multiset w_x, which is finally added to the input membrane of Π_n, thus obtaining a P system Π_x accepting iff $x \in L$.
- $\boldsymbol{\Pi}$ operates in time f, i.e. for each $x \in \Sigma^\star$, every computation of Π_x halts within $f(|x|)$ steps.

In particular, a language $L \subseteq \Sigma^\star$ belongs to the complexity class $\boldsymbol{PMC}_{\mathcal{D}}$ iff there exists a uniform family of confluent P systems $\boldsymbol{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding L in polynomial time.

The analogous complexity classes for *non*-confluent P systems are denoted by $\boldsymbol{NMC}_{\mathcal{D}}(f)$ and $\boldsymbol{NPMC}_{\mathcal{D}}$.

3 A measure of space complexity for P systems

In order to define the space complexity of P systems, we first need to establish a measure of the size of their configurations. The first definition we propose is based on an hypothetical implementation of P systems by means of real biochemical materials (cellular membranes and molecules). Under this assumption, every single object takes some constant physical space: this is equivalent to using a unary coding to represent multiplicities.

Definition 6. Let $\mathcal{C}$ be a configuration of a P system Π, that is, a rooted, unordered tree μ representing the membrane structure of Π, whose vertices are labeled with the multisets describing the contents of each region. The *size* $|\mathcal{C}|$ *of* $\mathcal{C}$ is then defined as the sum of the number of membranes in μ and the total number of objects they contain.

An alternative definition focuses on the simulative point of view, i.e. on the implementation of P systems *in silico*, where it is not necessary to actually store every single object (using a unary representation), but we can just store their multiplicity as a binary number, thus requiring exponentially less space for each kind of symbol.

Definition 7 (Alternative)**.** Let $\mathcal{C}$ be a configuration of a P system Π, that is, a rooted, unordered tree μ representing the membrane structure of Π, whose vertices are labeled with the multisets describing the contents of each region. The *size* $|\mathcal{C}|$ *of* $\mathcal{C}$ is then defined as the sum of the number of membranes in μ and the total number of bits required to store the objects they contain.

In the following discussion we will assume the first definition; however notice that the actual results might or might not depend on the precise choice between Definitions 6 and 7 (a thorough analysis of the differences involves a clarification of the relative importance of the number of membranes and the number of objects in various classes of P systems, and it is left as an open problem).

Once a notion of configuration size is established, we need to take account of all possible computation paths which can develop even in confluent recogniser P systems; the following definitions are given in the spirit of those concerning time complexity for P systems [7].

Definition 8. Let Π be a (confluent or non-confluent) recogniser P system, and let $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_m)$ be a halting computation of Π, that is, a sequence of configurations starting from the initial one and such that every subsequent one is reachable in one step by applying the rules in a maximally parallel way. The *space required by* $\vec{\mathcal{C}}$ is defined as

$$|\vec{\mathcal{C}}| = \max\{|\mathcal{C}_0|, \ldots, |\mathcal{C}_m|\}.$$

The *space required by* Π itself is then

$$|\Pi| = \max\{|\vec{\mathcal{C}}| \text{ such that } \vec{\mathcal{C}} \text{ is a halting computation of } \Pi\}.$$

Definition 9. Let $\boldsymbol{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\}$ be a uniform or semi-uniform family of recogniser P systems, each Π_x deciding the membership of the string x in a language $L \subseteq \Sigma^\star$; also let $f\colon \mathbb{N} \to \mathbb{N}$. We say that $\boldsymbol{\Pi}$ *operates within space bound* f iff $|\Pi_x| \leq f(|x|)$ for each $x \in \Sigma^\star$.

We are now ready to define space complexity classes for P systems.

Definition 10. Let $\mathcal{D}$ be a class of confluent recogniser P systems; let $f\colon \mathbb{N} \to \mathbb{N}$ and $L \subseteq \Sigma^\star$. Then $L \in \boldsymbol{MCSPACE}^\star_{\mathcal{D}}(f)$ iff L is decided by a semi-uniform family $\boldsymbol{\Pi} \subseteq \mathcal{D}$ of P systems operating within space bound f.

The corresponding complexity class for uniform families of confluent P systems is denoted by $\boldsymbol{MCSPACE}_{\mathcal{D}}(f)$, while in the non-confluent case we have the classes $\boldsymbol{NMCSPACE}^\star_{\mathcal{D}}(f)$ and $\boldsymbol{NMCSPACE}_{\mathcal{D}}(f)$ respectively.

As usual, we provide a number of abbreviations for important space classes.

Definition 11. The classes corresponding to polynomial and exponential space, in the semi-uniform and confluent case, are

$$\boldsymbol{PMCSPACE}^\star_{\mathcal{D}} = \bigcup_{k\in\mathbb{N}} \boldsymbol{MCSPACE}^\star_{\mathcal{D}}(O(n^k))$$
$$\boldsymbol{EXPMCSPACE}^\star_{\mathcal{D}} = \bigcup_{k\in\mathbb{N}} \boldsymbol{MCSPACE}^\star_{\mathcal{D}}\big(2^{O(n^k)}\big).$$

The definitions are analogous in the uniform and non-confluent cases.

4 Basic results

From the above definitions, some results concerning space complexity classes and their relations with time complexity classes follow immediately. We state them only for semi-uniform families, but they also hold in the uniform case.

The first two propositions can be immediately derived from the definitions.

Proposition 12. *The following inclusions hold:*

$$\boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}} \subseteq \boldsymbol{EXPMCSPACE}^{\star}_{\mathcal{D}}$$
$$\boldsymbol{NPMCSPACE}^{\star}_{\mathcal{D}} \subseteq \boldsymbol{NEXPMCSPACE}^{\star}_{\mathcal{D}}.$$

Proposition 13. $\boldsymbol{MCSPACE}^{\star}_{\mathcal{D}}(f) \subseteq \boldsymbol{NMCSPACE}^{\star}_{\mathcal{D}}(f)$ *for each* $f\colon \mathbb{N} \to \mathbb{N}$*, and in particular*

$$\boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}} \subseteq \boldsymbol{NPMCSPACE}^{\star}_{\mathcal{D}}$$
$$\boldsymbol{EXPMCSPACE}^{\star}_{\mathcal{D}} \subseteq \boldsymbol{NEXPMCSPACE}^{\star}_{\mathcal{D}}.$$

The following results mirror those which hold for Turing machines, and they describe closure properties and provide an upper bound for time requirements of P systems operating in bounded space.

Proposition 14. *The classes* $\boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}}$*,* $\boldsymbol{NPMCSPACE}^{\star}_{\mathcal{D}}$*,* $\boldsymbol{EXPMCSPACE}^{\star}_{\mathcal{D}}$*, and* $\boldsymbol{NEXPMCSPACE}^{\star}_{\mathcal{D}}$ *are all closed under polynomial-time reductions.*

Proof. Let $L \in \boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}}$ and let M be the Turing machine constructing the family $\boldsymbol{\Pi}$ that decides L. Let L' be reducible to L via the polynomial-time computable function f.

Now let M' be the following Turing machine: on input x of length n compute $f(x)$; then behave like M on input $f(x)$, thus constructing $\Pi_{f(x)}$. Since $|f(x)|$ is bounded by a polynomial, M' operates in polynomial time and $\Pi_{f(x)}$ in polynomial space; but then $\boldsymbol{\Pi}' = \{\Pi_{f(x)} \mid x \in \Sigma^{\star}\}$ is a polynomially semi-uniform family of P systems deciding L' in polynomial space. Thus $L' \in \boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}}$.

The proof for the three other classes is analogous. □

Proposition 15. $\boldsymbol{MCSPACE}^{\star}_{\mathcal{D}}(f)$ *is closed under complement for each function* $f\colon \mathbb{N} \to \mathbb{N}$*.*

Proof. Simply reverse the roles of objects *yes* and *no* in order to decide the complement of a language. □

Proposition 16. *For each function* $f\colon \mathbb{N} \to \mathbb{N}$

$$\boldsymbol{MCSPACE}^{\star}_{\mathcal{D}}(f(n)) \subseteq \boldsymbol{MC}^{\star}_{\mathcal{D}}\left(2^{O(f(n)\log f(n))}\right)$$
$$\boldsymbol{NMCSPACE}^{\star}_{\mathcal{D}}(f(n)) \subseteq \boldsymbol{NMC}^{\star}_{\mathcal{D}}\left(2^{O(f(n)\log f(n))}\right).$$

Proof. Let $L \in \boldsymbol{MCSPACE}^{\star}_{\mathcal{D}}(f(n))$ be decided by the semi-uniform family $\boldsymbol{\Pi}$ of recogniser P systems in space f; let $\Pi_x \in \boldsymbol{\Pi}$ with $|x| = n$ and let $\mathcal{C}$ be a configuration of Π_x. Then $\mathcal{C}$ can be described with a string of length at most $kf(n)\log f(n)$ over a finite alphabet, say with $b \geq 2$ symbols, for some constant k:

- We need $kf(n)$ symbols in order to represent the membrane structure and the objects it contains, as well as the polarizations.
- We also need to encode the labels of the membranes: even if they do not contribute to the space required by the P system, nonetheless each assignment of labels gives rise to a different configuration. Since all labels might be distinct, and there are at most $f(n)$ of them, $kf(n)\log f(n)$ symbols are needed.

Notice that there are less than $b^{kf(n)\log f(n)+1}$ such strings. Since Π_x is a recogniser P system, by definition every computation halts: then it must halt within $b^{kf(n)\log f(n)+1}$ steps in order to avoid repeating a previous configuration (thus entering an infinite loop). This number of steps is $2^{O(f(n)\log f(n))}$.

The same proof also works in the non-confluent case (only the acceptance criterion is different). □

5 Space complexity of P systems with active membranes

In this section we provide a brief review of part of the ample literature on complexity results about P systems with active membranes; our aim is to analyse existing polynomial-time solutions to hard computational problems in order to obtain space complexity results.

We first consider the class of P systems with active membranes which do not make use of membrane division rules, usually denoted by $\mathcal{NAM}$. It is a well known fact that such P systems are able to solve only problems in $\boldsymbol{P}$ (the so-called Milano theorem [11]); on the other hand, they can be used to solve *all* problems in $\boldsymbol{P}$ with a minimal amount of space, when a semi-uniform construction is considered:

Proposition 17. $\boldsymbol{P} \subseteq \boldsymbol{MCSPACE}^{\star}_{\mathcal{NAM}}(O(1))$.

Proof. Let $L \in \boldsymbol{P}$. Then there exists a deterministic Turing machine M deciding L in polynomial time. Now consider the family of P systems $\boldsymbol{\Pi} = \{\Pi_{no}, \Pi_{yes}\}$, where Π_{no} (resp. Π_{yes}) is the following trivial P system with active membranes:

- the membrane structure consists of the skin only, labelled by h;
- in the initial configuration, exactly one object a is located inside the skin;
- the only rule is $[a]^{\mathrm{o}}_h \rightarrow [\,]^{\mathrm{o}}_h\, no$ (resp. $[a]^{\mathrm{o}}_h \rightarrow [\,]^{\mathrm{o}}_h\, yes$).

It is clear that such P systems halt in one step and that the space they require is independent of the size of the instance they decide.

The family of P systems $\boldsymbol{\Pi}$ can be constructed in a semi-uniform way in order to decide L by a deterministic Turing machine which first simulates M (it can do so, since M operates in polynomial time), then outputs one of Π_{yes}, Π_{no} according to the result (acceptance or rejection, respectively). □

One of the most powerful features of P systems with active membranes is the possibility of creating an exponential workspace in polynomial time by means of elementary membrane division rules; we denote the class of such P systems by $\mathcal{EAM}$. This feature was exploited for solving $\boldsymbol{NP}$-complete problems in polynomial (often even linear) time. In terms of space complexity, this can be stated as follows:

Proposition 18. $\boldsymbol{NP} \cup \boldsymbol{coNP} \subseteq \boldsymbol{EXPMCSPACE}^{\star}_{\mathcal{EAM}}$.

Proof. In [11] a polynomial-time semi-uniform solution to SAT is described; the number of membranes and objects required is exponential with respect to the length of the Boolean formula. The result then follows from closure under reductions and complement of $\boldsymbol{EXPMCSPACE}^{\star}_{\mathcal{EAM}}$. □

This result can be improved when the use of non-elementary membrane division rules is allowed; indeed, all problems in $\boldsymbol{PSPACE}$ can be solved by such class of P systems with active membranes, denoted by $\mathcal{AM}$.

Proposition 19. $\boldsymbol{PSPACE} \subseteq \boldsymbol{EXPMCSPACE}^{\star}_{\mathcal{AM}}$.

Proof. In [10] a polynomial-time uniform solution to QBF (also known as QSAT), the canonical $\boldsymbol{PSPACE}$-complete problem, is described; the space required by each P system is still exponential, and the result follows from the closure properties. □

In [1] a *uniform* solution for the same problem was achieved, with the same space requirements; this provides a tighter upper bound to $\boldsymbol{PSPACE}$:

Proposition 20. $\boldsymbol{PSPACE} \subseteq \boldsymbol{EXPMCSPACE}_{\mathcal{AM}}$.

Since standard P systems with active membranes are very powerful when division rules are allowed, but very weak otherwise, another line of research involves removing some other features, such as polarizations. Polarizationless P systems with active membranes have been proved able to solve QSAT uniformly in polynomial time by making use of both elementary and non-elementary division rules [2]. Since the space requirements are once again exponential, the following result is immediate:

Proposition 21. *$\boldsymbol{PSPACE} \subseteq \boldsymbol{EXPMCSPACE}_{\mathcal{AM}^0}$, where $\mathcal{AM}^0$ is the class of polarizationless P systems with active membranes and both kinds of division rules.*

6 Open problems

In P systems with active membranes, division rules are usually exploited by producing an exponential number of membranes in linear time, which then evolve in parallel; for instance, several solutions to **NP**-complete problems explore the full solution space (e.g. generating every possible truth assignment and then checking whether one of them satisfies a Boolean formula). It appears that membrane division may become much less useful when a polynomial upper bound on space is set; or, in other words,

Conjecture 22. *The three complexity classes $\boldsymbol{PMCSPACE}^{\star}_{\mathcal{NAM}}$, $\boldsymbol{PMCSPACE}^{\star}_{\mathcal{EAM}}$ and $\boldsymbol{PMCSPACE}^{\star}_{\mathcal{AM}}$ coincide.*

An idea which might be useful in proving this conjecture is precomputing the "final" membrane structure (which is obtained via division rules) during the construction phase. While this is straightforward when considering membrane divisions which always occur, the matter might be much more difficult in the case of "conditional" division (i.e. division rules are applied only when certain conditions are met) or when the P system exhibits a recurring behaviour (e.g. a membrane divides, then one of the two copies is dissolved, and the process is repeated continuously).

Another interesting problem involves the relations between time and space complexity classes for P systems with active membranes. We know that Turing machines, once a polynomial space bound is fixed, are able to solve more problems in exponential time than in polynomial time (at least when $\boldsymbol{P} \neq \boldsymbol{PSPACE}$ is assumed). This fact has not been investigated yet in the setting of membrane computing, as all solutions to decision problems presented until now (up to the knowledge of the authors) require only a polynomial amount of time. Formally, the question we pose is the following:

Problem 23. *Is $\boldsymbol{PMC}^{\star}_{\mathcal{D}} \neq \boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}}$ for any class of P systems $\mathcal{D}$ among $\mathcal{NAM}$, $\mathcal{EAM}$, $\mathcal{AM}$? That is, do problems which can be solved in polynomial space but not in polynomial time exist?*

Another important property of traditional computing devices is described by Savitch's theorem: nondeterministic space-bounded Turing machines can be simulated deterministically with just a polynomial increase in space requirements, and as a consequence $\boldsymbol{PSPACE} = \boldsymbol{NPSPACE}$ holds. The proof does not appear to be transferable to P systems in a straightforward way; nonetheless, an analogous result might hold even in this setting:

Problem 24. *Does $\boldsymbol{PMCSPACE}^{\star}_{\mathcal{D}} = \boldsymbol{NPMCSPACE}^{\star}_{\mathcal{D}}$ hold for any class of P systems $\mathcal{D}$ among $\mathcal{NAM}, \mathcal{EAM}, \mathcal{AM}$?*

The classes of P systems with active membranes we have considered in all the previous problems are only defined according to which kinds of membrane division rules are available (none, just elementary or both elementary and non-elementary). The same questions may be also worth posing about other restricted classes, such as P systems without object evolution or communication [12, 4], P systems with division but without dissolution, or even purely communicating P systems, with or without polarizations.

Finally, we feel that the differences between P systems and traditional computing devices deserve to be investigated for their own sake also from the point of view of space-bounded computations. We formulate this as an open-ended question:

Problem 25. *What are the relations between space complexity classes for P systems and traditional ones, such as* ***P****,* ***NP****,* ***PSPACE****,* ***EXP****,* ***NEXP****, and* ***EXPSPACE****?*

Bibliography

[1] A. Alhazov, C. Martín-Vide, L. Pan, Solving a PSPACE-Complete Problem by Recognizing P Systems with Restricted Active Membranes, *Fundamenta Informaticae*, vol. 58(2), pp. 67–77, 2003.

[2] A. Alhazov, M. J. Pérez-Jiménez, Uniform Solution of QSAT Using Polarizationless Active Membranes, in: J. Durand-Lose, M. Margenstern, eds., *Machines, Computations, and Universality*, 5th International Conference, MCU 2007, Orléans, France, Lecture Notes in Computer Science, vol. 4664, pp. 122–133, Springer, 2007.

[3] M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, F. J. Romero-Campero, P Systems with Active Membranes, without Polarizations and without Dissolution: A Characterization of P, in: C. Calude, M. J. Dinneen, G. Păun, M. J. Pérez-Jiménez, G. Rozenberg, eds., *Unconventional Computation*, 4th International Conference, UC 2005, Sevilla, Spain, Lecture Notes in Computer Science, vol. 3699, pp. 105–116, Springer, 2005.

[4] A. Leporati, C. Ferretti, G. Mauri, M. J. Pérez-Jiménez, C. Zandron, Complexity Aspects of Polarizationless Membrane Systems, *Natural Computing*, Special issue devoted to IWINAC 2007, to appear.

[5] G. Păun, P-Systems with Active Membranes: Attacking NP Complete Problems, in: I. Antoniou, C. Calude, M. J. Dinneen, eds., *Unconventional Models of Computation*, 2nd International Conference, UMC'2K, Brussels, Belgium, Springer, 2001.

[6] G. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, 2002.

[7] M. J. Pérez-Jiménez, Á. Romero Jiménez, F. Sancho-Caparrini, Complexity Classes in Models of Cellular Computing with Membranes, *Natural Computing*, vol. 2(3), pp. 265–285, 2003.

[8] M. J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini, The P versus NP Problem through Cellular Computing with Membranes, in: N. Jonoska, G. Păun, G. Rozenberg, eds., *Aspects of Molecular Computing*, Lecture Notes in Computer Science, vol. 2950, pp. 338–352, Springer, 2004.

[9] A. E. Porreca, G. Mauri, C. Zandron, Complexity Classes for Membrane Systems, *RAIRO Theoretical Informatics and Applications*, vol. 40(2), pp. 141–162, 2006.

[10] P. Sosík: The Computational Power of Cell Division in P Systems: Beating Down Parallel Computers?, *Natural Computing*, vol. 2(3), pp. 287–298, 2003.

[11] C. Zandron, C. Ferretti, G. Mauri, Solving NP-Complete Problems Using P Systems with Active Membranes, in: I. Antoniou, C. Calude, M. J. Dinneen, eds., *Unconventional Models of Computation*, 2nd International Conference, UMC'2K, Brussel, Belgium, Springer, 2001.

[12] C. Zandron, A. Leporati, C. Ferretti, G. Mauri, M. J. Pérez-Jiménez, On the Computational Efficiency of Polarizationless Recognizer P systems with Strong Division and Dissolution, *Fundamenta Informaticae*, vol. 87(1), pp. 79–91, 2008.

Antonio E. Porreca was born on September 18, 1982 in Monza, Italy. He got his B.Sc. and M.Sc. in Computer Science, in 2005 and 2008 respectively, from University of Milano-Bicocca, where he is now a Ph.D. student. His current research interests focus on complexity-theoretic aspects of membrane computing.

Alberto Leporati obtained a Ph.D. in Computer Science from the University of Milano (Italy) in 2002. Since 2004, he is assistant professor at the University of Milano-Bicocca. His research interests are in membrane computing, theoretical computer science and computational complexity.

Giancarlo Mauri is full professor of Computer Science at the University of Milano-Bicocca. His research interests are mainly in the area of theoretical computer science, and include: formal languages and automata, computational complexity, bioinformatics and unconventional computing models, in particular membrane systems. On these subjects, he published more than 200 scientific papers in international journals, contributed volumes and conference proceedings.

Claudio Zandron was born in 1972 in Milan, Italy. He received his M.Sc. in Computer Science from the University of Milano in 1996, and a Ph.D. degree in 2002 from the same university. Since 2006 he is associate professor at the University of Milano-Bicocca, Department of Informatics, Systems, and Communication. His research interests concern the areas of formal languages, molecular computing, and computational complexity.