An Efficient Simulation of Polynomial-Space Turing Machines by P Systems with Active Membranes

Andrea Valsecchi, 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
{valsecchi,porreca,leporati,mauri,zandron}@disco.unimib.it

Summary. We show that a deterministic single-tape Turing machine, operating in polynomial space with respect to the input length, can be efficiently simulated (both in terms of time and space) by a semi-uniform family of P systems with active membranes and three polarizations, using only communication rules. Then, basing upon this simulation, we prove that a result similar to the *space hierarchy theorem* can be obtained for P systems with active membranes: the larger the amount of space we can use during the computations, the harder the problems we are able to solve.

1 Introduction

Membrane systems (also known as *P systems*) have been introduced in [11] as a parallel, nondeterministic, synchronous and distributed model of computation inspired by the structure and functioning of living cells. The basic model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into *regions*, that contain multisets of *objects* (represented by symbols of an alphabet) and *evolution rules*. Using these rules, the objects may evolve and/or move from a region to a neighboring one. Usually, the rules are applied in a nondeterministic and maximally parallel way. A *computation* starts from an initial configuration of the system and terminates when no evolution rule can be applied. The result of a computation is the multiset of objects contained into an *output membrane*, or emitted from the skin of the system. An interesting subclass of membrane systems is constituted by *recognizer* P systems, in which: (1) all computations halt, (2) only two possible outputs exist (usually named *yes* and *no*), and (3) the result produced by the system only depends upon its input, and is not influenced by the particular sequence of computation steps taken to produce it. For a systematic introduction

to P systems we refer the reader to [13], whereas the latest information can be found in [22].

Since the introduction of membrane systems, many investigations have been performed on their computational properties: in particular, many variants have been proposed in order to study the contribution of various ingredients (associated with the membranes and/or with the rules of the system) to the achievement of the computational power of these systems. In this respect, it is known [14, 20, 6] that the class of all decision problems which can be solved in polynomial time by a family of recognizer P systems that use only basic rules, that is, evolution, communication and membrane dissolution, coincides with the complexity class **P**. Hence, in order to efficiently solve computationally difficult (for example, **NP**-complete) problems by means of P systems it seems necessary to be able to exponentially increase (in polynomial time) the number of membranes, that can be regarded as the size of the workspace. In particular, two features have proven to be of paramount importance in establishing whether a membrane system is able to solve computationally difficult decision problems in polynomial time: membrane dissolution and division. Dissolution rules simply dissolve the surrounding membrane when a specified symbol occurs. Division rules are inspired from the biological process called *mitosis*: they allow to duplicate a given membrane that contains a specified symbol, possibly rewriting this symbol in a different way in each of the membranes produced by the process. All the other symbols, as well as the rules, which are contained in the original membrane are copied unaltered into each of the resulting regions. As for the membranes possibly contained in the original region (if any), we can consider the following situations. If no membrane occurs, then we say that the division is *elementary*; if one or more membranes occur, then we have to specify how they are affected by the division operation. If all the membranes are copied to each of the resulting regions, then we have a *weak* (non-elementary) division; if, instead, we can choose what membranes are copied into each of the resulting regions, then we have a *strong* (non-elementary) division.

Recognizer P systems with active membranes (using division rules and, possibly, polarizations associated to membranes) have been successfully used to efficiently solve **NP**-complete problems. The first solutions were given in the so called *semi-uniform* setting [12, 20, 9, 10], which means that we assume the existence of a deterministic Turing machine that, for every instance of the problem, produces in polynomial time a description of the P system that solves such an instance. The solution is computed in a *confluent* manner, meaning that the instance given in input is positive (resp., negative) if and only if every computation of the P system associated with it is an accepting (resp., rejecting) computation. Another way to solve **NP**-complete problems by means of P systems is by considering the *uniform* setting, in which any instance of the problem of a given length can be fed as input – encoded in an appropriate way – to a specific P system and then solved by it. Sometimes, a uniform solution to a decision problem Q is provided by defining a family $\{\Pi_Q(n)\}_{n \in \mathbb{N}}$ of P systems such that for every $n \in \mathbb{N}$ the system $\Pi_Q(n)$ reads in input an encoding of any possible instance of size n, and solves it. P sys-

tems with active membranes have thus been successfully used to design uniform polynomial-time solutions to some well-known **NP**-complete problems, such as SAT [15].

All the papers mentioned above deal with P systems having three polarizations, that use only division rules for elementary membranes (in [19] also division for non-elementary membranes is permitted, and in this way a semi-uniform solution to the **PSPACE**-complete problem QSAT is provided), and working in the *maximally parallel* way. As shown in [2], the number of polarizations can be decreased to two without loss of efficiency. On the other hand, in [5] the computational power of recognizer P systems with active membranes but *without* electrical charges and dissolution rules was investigated, establishing that they characterize the complexity class **P**. Finally, in [21] it was shown that polarizationless P systems with active membranes that use strong division for non-elementary membranes and dissolution rules, working in the maximally parallel way, are able to solve in polynomial time the **NP**-complete problem 3-SAT. This result establishes that neither evolution nor communication rules, and no electrical charges are needed to solve **NP**-complete problems, provided that we can use strong division rules for non-elementary membranes (as well as dissolution rules, otherwise we would fall in the case considered in [5]).

By looking at the literature one can see that, until now, the research on the complexity theoretic aspects of P systems with active membranes has mainly focused on the *time* resource. In particular, we can find several results that compare time complexity classes obtained by using various ingredients (such as, e.g., polarizations, dissolution, uniformity, etc.). Other works make a comparison between these classes and the usual complexity classes defined in terms of Turing machines, either from the point of view of time complexity [14, 4, 20], or space complexity [19, 1, 17]. A first definition of space complexity for P systems was given in [7], where the measure of space is given by the maximum number of objects occurring during the computation. The definition was then generalized to P systems with mutable membrane structure [16], in particular P systems with active membranes, thus formalizing the usual notion of exponential workspace generated through membrane division.

In this paper, basing upon the formal definitions given in [16], we present some results concerning the relations among space complexity classes defined in terms of P systems, under some specified constraints. In particular, we first show how to simulate a deterministic single-tape Turing machine by a semi-uniform family of P systems with active membranes and three polarizations.

Then, by focusing our attention on computations occurring in polynomial space, we define a pseudo-hierarchy of space complexity classes. Such classes are inspired by the *space hierarchy theorem*, that we restate and prove (albeit in a slightly different form) for P systems with active membranes. Let us note that a different hierarchy for catalytic P systems with a fixed membrane structure has been introduced in [7].

The paper is organized as follows. In section 2 we recall the definition of recogniser P systems with active membranes, thus establishing our model of computation, and we recall some basic notions that will be used in the rest of the paper. In section 3 we show how to simulate a deterministic single-tape Turing machine by means of a semi-uniform family of P systems with active membranes. In section 4 we recall the space hierarchy theorem and, inspired by it, we define a pseudo-hierarchy of space complexity classes determined by P systems with active membranes. Finally, section 5 contains the conclusions and some directions for further research.

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 $[a \rightarrow w]_h^\alpha$*
  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).
- Communication rules*, of the form $a\, [\,]_h^\alpha \rightarrow [b]_h^\beta$*
  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.

- 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*

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

  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 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 $\mathcal{C}$ *of a P system* Π *is a 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).*

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

Definition 2. *A* recogniser 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.*

If all computations starting from the initial configuration 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 an accepting computation exists).*

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

Usually, a condition of uniformity, inspired by those of families of Boolean circuits, is imposed on families of P systems.

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

Time complexity classes for P systems are defined as usual, by restricting the amount of time available for deciding a language. By $\mathbf{MC}_{\mathcal{D}}^{\star}(f(n))$ we denote the class of languages which can be decided by a semi-uniform class of confluent P systems $\mathbf{\Pi}$ of class $\mathcal{D}$ (e.g., $\mathcal{AM}$ denotes the class of P systems with active membranes) where each computation of $\Pi_x \in \mathbf{\Pi}$ halts within $f(|x|)$ steps. The class of languages decidable in polynomial time is denoted by $\mathbf{PMC}_{\mathcal{D}}^{\star}$.

Recently, a space complexity measure for P systems has been introduced [16]. We recall here the relevant definitions.

Definition 5. *Let* $\mathcal{C}$ *be a configuration of a P system* Π. *The* size $|\mathcal{C}|$ *of* $\mathcal{C}$ *is defined as the sum of the number of membranes in* μ *and the total number of objects they*

contain[1]. If $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ is a halting computation of Π, then the space required by $\mathcal{C}$ is defined as

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

The space required by Π itself is then

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

Finally, let $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ be a family of recogniser P systems; also let $f \colon \mathbb{N} \to \mathbb{N}$. We say that $\mathbf{\Pi}$ operates within space bound f iff $|\Pi_x| \leq f(|x|)$ for each $x \in \Sigma^\star$.

Next, we formally define the variant of Turing machine we use in the following sections.

Definition 6. *A single-tape deterministic Turing machine is a tuple*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, A, R)$$

where:

- *Q is a finite and nonempty set of states;*
- *Σ is the finite input alphabet;*
- *Γ is the tape alphabet, a finite superset of Σ;*
- *the partial function $\delta \colon \Gamma \times Q \to \Gamma \times Q \times \{\leftarrow, -, \rightarrow\}$ is the transition function; we assume that δ is undefined on both accepting and rejecting states;*
- *$q_0 \in Q$ is the initial state;*
- *$A \subseteq Q$ is the set of accepting states;*
- *$R \subseteq Q$ is the set of rejecting states, disjoint from A.*

Finally, we recall the definition of constructible function (for further information on this topic see, for instance, [8, 3, 18]).

Definition 7. *A function $f \colon \mathbb{N} \to \mathbb{N}$ is said to be* time-constructible *iff the mapping $1^n \mapsto 1^{f(n)}$, i.e., from the unary representation of n to the unary representation of $f(n)$, can be computed by a deterministic Turing machine in $O(f(n))$ time.*

The function f is space-constructible *iff the mapping $1^n \mapsto 1^{f(n)}$ can be computed by a deterministic Turing machine in $O(f(n))$ space.*

---

[1] An alternative definition, where the size of a configuration is given by the sum of the number of membranes and *the number of bits required to store the objects they contain*, has been considered in [16]. However, the choice between the two definitions is irrelevant as far as the results of this paper are concerned.

3 Simulating Turing machines

In this section we show that a single-tape Turing machine M having $\Sigma = \{0, 1\}$ as input alphabet and operating in polynomial space $f(n)$ and time $g(n)$ can be simulated efficiently (i.e., by using $O(f(n))$ space and $O(g(n))$ time) by a semi-uniform family $\mathbf{\Pi}_M = \{\Pi_{M,x} : x \in \{0,1\}^\star\}$ of P systems with active membranes and three polarizations, where each $\Pi_{M,x}$ simulates the computation of M on input x. We also stress the fact that these P systems can be defined in such a way that communication is the only required kind of rule.

Turing machines operate by reading and writing symbols on a tape divided into cells: the main idea of our simulation is representing each cell by a membrane. In a Turing machine the tape cells are linearly ordered (we assume they are numbered by nonnegative integers); one way to organise the membranes without losing this information is in a nested way, i.e., one inside the other. Either the innermost or the outermost membrane can be put into correspondence with the leftmost tape cell; without loss of generality, we choose the outermost one.

Each cell of the Turing machine contains a symbol taken from the tape alphabet, which we assume to be $\Gamma = \{0, 1, \textvisiblespace\}$, where $\textvisiblespace$ denotes the blank symbol. In the P system, the symbol written in a cell is stored *in the polarization* of the corresponding membrane. The default neutral polarization represents a blank cell, while the negative and positive polarizations represent 0 and 1, respectively.

A single object in the P system represents the state of the Turing machine (an element q of the finite set Q), and its location inside the membrane structure represents the position of the tape head: the object is located immediately inside the i-th membrane iff the tape head of the simulated machine is located on the i-th leftmost tape cell. The object is changed (via communication rules) both in form and location in order to reflect the change of state and position of the tape head of the Turing machine.

Finally, the transition function $\delta \colon \Gamma \times Q \to \Gamma \times Q \times \{\leftarrow, -, \rightarrow\}$ of the Turing machine is implemented by using a set of communication rules. The object representing the head position and state of the Turing machine is moved to the new position, while simultaneously changing the polarization of the current membrane in order to update the contents of the tape; it is also rewritten into a (possibly different) symbol, representing the new state of the machine. In order to execute these operations, the P system requires a constant number of steps for each computation step of the simulated Turing machine.

Let M be a single-tape deterministic Turing machine operating in space $f(n)$. Let $x = x_1 x_2 \cdots x_n \in \{0,1\}^\star$ be an input for M. The membrane structure $\mu_{M,x}$ is made of $f(n)$ membranes labelled by h and placed one inside the other; this structure is surrounded by a further membrane h_0, which also contains a membrane labelled by w. The initial configuration of $\mu_{M,x}$ is as follows:

$$[[\hat{q}_0]_w^0 \overbrace{[x_1[x_2 \cdots [x_n[\cdots [}^{f(n) \text{ membranes}} \overbrace{]_h^0 \cdots]_h^0}^{f(n) \text{ membranes}} \underbrace{]_h^0 \cdots]_h^0]_h^0}_{n \text{ membranes}}]_{h_0}^0$$

Each of the outermost n membranes labelled by h contains an object $x_i \in \{0, 1\}$, representing the i-th input symbol of M; these objects are used to set up the initial contents of the tape (recall that, by definition, all membranes are initially required to be neutral). The following communication rules serve the purpose of changing the polarization of a membrane h according to the symbol contained in the corresponding tape cell:

$$[0]_h^0 \rightarrow [\,]_h^- \,\# \tag{1}$$

$$[1]_h^0 \rightarrow [\,]_h^+ \,\# \tag{2}$$

where $\#$ is a "junk" object, i.e., an object which does not appear on the left-hand side of any rule.

While the initial configuration of the simulated machine M is being set up (only one step is required to do so) the head/state object $\hat{q}_0$, where q_0 is the initial state of M, is sent out from w by means of the following rule:

$$[\hat{q}_0]_w^0 \rightarrow [\,]_w^0 \,\hat{q}_0 \tag{3}$$

After that, $\hat{q}_0$ enters the membrane corresponding to the leftmost tape cell, while simultaneously losing the "hat", by using one of the following communication rules:

$$\hat{q}_0 \, [\,]_h^\alpha \rightarrow [q_0]_h^\alpha \qquad\qquad \forall \alpha \in \{-, 0, +\} \tag{4}$$

Object $\hat{q}_0$ is initially located inside w so that it requires two steps in order to reach the membrane corresponding to the initial cell of M, thus avoiding conflicts with the rules setting up the initial tape contents.

Now the real simulation begins. To each quintuple (a, q_1, b, q_2, d) describing a transition of M (i.e., denoting the fact that $\delta(a, q_1) = (b, q_2, d)$) corresponds a constant number of communication rules. If $\delta(a, q_1) = (b, q_2, \leftarrow)$ then there is a single rule

$$[q_1]_h^\alpha \rightarrow [\,]^\beta \, q_2 \tag{5}$$

where α and β are $-$ or $+$ when a and b are 0 or 1 respectively. The rule moves the head/state object outwards (which corresponds to moving the tape head of M one position to the left) while changing it as the state of M does.

If the tape head does not move, as in $\delta(a, q_1) = (b, q_2, -)$, then two rules are needed:

$$[q_1]_h^\alpha \rightarrow [\,]_h^\beta \, q_2' \tag{6}$$

$$q_2' \, [\,]_h^\beta \rightarrow [q_2]_h^\beta \tag{7}$$

The first rule changes the symbol in the current cell, while the second one moves the (updated) head/state symbol back to that cell.

When the tape head moves right, i.e., $\delta(a, q_1) = (b, q_2, \rightarrow)$, five rules are needed:

$$[q_1]_h^\alpha \rightarrow [\,]^\beta \, q_2'' \tag{8}$$

$$q_2'' \, [\,]_h^\beta \rightarrow [q_2']_h^\beta \tag{9}$$

$$q_2' \, [\,]_h^\gamma \rightarrow [q_2]_h^\gamma \qquad\qquad \forall \gamma \in \{-, 0, +\} \tag{10}$$

The three rules in (10) are used to move the head/state symbol one membrane deeper, thus completing the simulated movement of the tape head to the right.

Finally, the result of the computation of M is sent out of the membrane structure. If M enters an accepting state q, the head/state symbol is changed to *yes* and expelled:

$$[q]_h^\alpha \to [\,]_h^\alpha \; yes \qquad\qquad \forall \alpha \in \{-, 0, +\} \qquad\qquad (11)$$

$$[yes]_h^\alpha \to [\,]_h^\alpha \; yes \qquad\qquad \forall \alpha \in \{-, 0, +\} \qquad\qquad (12)$$

$$[yes]_{h_0}^0 \to [\,]_{h_0}^0 \; yes \qquad\qquad\qquad\qquad\qquad (13)$$

An analogous situation occurs when q is a rejecting state:

$$[q]_h^\alpha \to [\,]_h^\alpha \; no \qquad\qquad \forall \alpha \in \{-, 0, +\} \qquad\qquad (14)$$

$$[no]_h^\alpha \to [\,]_h^\alpha \; no \qquad\qquad \forall \alpha \in \{-, 0, +\} \qquad\qquad (15)$$

$$[no]_{h_0}^0 \to [\,]_{h_0}^0 \; no \qquad\qquad\qquad\qquad\qquad (16)$$

Definition 8. *With a slight abuse of notation, we denote by $\mu_{M,x}$ the whole P system "module" consisting of both the membrane structure described above and the set of rules (1)–(16). We also denote by $\mathbf{\Pi}_M$ the family of P systems with active membranes $\{\Pi_{M,x} : x \in \{0,1\}^\star\}$, where $\Pi_{M,x}$ consists of the module $\mu_{M,x}$ only.*

Theorem 1. *Let M be a single-tape deterministic Turing machine halting on every input and operating in space $f(n)$, where $f(n) = \Omega(n)$, $f(n) = O(n^k)$ for some fixed k and $f(n)$ is time-constructible. Also assume that M operates in time $g(n)$. Then $\mathbf{\Pi}_M$ is semi-uniform and decides the same language as M in $O(f(n))$ space and $O(g(n))$ time; furthermore, $\mathbf{\Pi}_M$ can be constructed in $O(f(n))$ time.*

Proof. Each P system $\Pi_{M,x}$ consists of $f(n) + 2$ membranes and contains $n + 1$ objects, where $n = |x|$; hence $\mathbf{\Pi}_M$ clearly uses $O(f(n))$ space.

Each transition of M on input x is simulated by $\Pi_{M,x}$ in at most three steps; another step is required to set up the initial contents of the tape. When the result object *yes/no* is produced, it is expelled from the system after a number of steps which equals the number of the tape cell where M enters the final state, plus a further step to exit the outermost membrane h_0. Hence, the total time is $O(g(n))$.

The mapping $x \mapsto \Pi_{M,x}$ can be computed in time $O(f(n))$, as

- the membrane structure consists of $f(|x|)$ identical membranes (and two further membranes w and h_0) and can be constructed in $O(f(n))$ time steps, as f is time-constructible by hypothesis;
- the initial configuration of the P system can be constructed in linear time from x, as exactly n symbols are to be placed inside the outermost membranes;
- the set of communication rules only depends on M, and not on x.

Since $f(n)$ is bounded by a polynomial, the construction of $\mathbf{\Pi}_M$ is semi-uniform.

4 A space pseudo-hierarchy

The space hierarchy theorem, a fundamental result in complexity theory, states that Turing machines are able to solve harder problems when given a larger amount of space to exploit. The proof [18] is constructive, as for every space bound $f(n)$ an explicit language is described which cannot be decided by using less space.

Definition 9. *Let $f\colon \mathbb{N} \to \mathbb{N}$. We denote by $L(f)$ the language of strings $x \in \{0,1\}^\star$ of the form $\langle M\rangle 10^\star$, where $\langle M\rangle$ is the binary description of a single-tape deterministic Turing machine that rejects x without using more than $f(|x|)$ space.*

Theorem 2 (Space hierarchy theorem). *Let f be a space-constructible function such that $f(n) = \Omega(n)$. Then $L(f)$ is decidable in space $O(f(n))$ but not in space $o(f(n))$.*

Proof (Proof sketch). The language $L(f)$ can be decided by a deterministic Turing machine D which simulates M on x within a $f(|x|)$ space limit, flipping the result whenever the simulation completes successfully and rejecting if the non-blank portion of the tape of M becomes longer than the space limit. Such a simulation can be carried out in space $O(f(n))$.

If $L(f)$ could be decided in space $g(n) = o(f(n))$ by some deterministic Turing machine M, then D on input $\langle M\rangle 10^k$ (for large enough values of k) could complete the simulation within the space limit and give a different result from M, thus contradicting the hypothesis that $L(D) = L(M) = L(f)$.

The trailing k zeros in the description of $L(f)$ are a technical requirement: since $g(n)$ may be larger than $f(n)$ for small n even when $g(n) = o(f(n))$, for some inputs the simulation might not complete successfully; but D certainly answers the opposite of M on all strings $\langle M\rangle 10^k$ for large enough values of k, thus ensuring they decide different languages.

A related result can be proved in the setting of P systems with active membranes. The main idea is to modify the Turing machine D of the above proof, in such a way that, instead of directly simulating the machine M it receives as input, it constructs a P system $\Pi''_{M,x,f}$ which carries out this task. $\Pi''_{M,x,f}$ is a variant of the P system $\Pi_{M,x}$ described in the previous section; notice that $\Pi_{M,x}$ is not suitable for the present task, as it is designed to simulate only halting Turing machines operating in polynomial space. The Turing machine D, instead, receives arbitrary machines M as input, which on some input x could try to use more space than we took into account when constructing $\Pi_{M,x}$; alternatively, they could also run forever, whereas we need to always give an answer.

We begin by modifying the P system module $\mu_{M,x}$ such that, when M exceeds the allocated space (i.e., when the tape head moves to the right of the rightmost cell), the simulation ends by rejecting. Furthermore, when the simulation is completed correctly, we return the opposite result of M.

The P system module $\mu'_{M,x,f}$, simulating M on x with a $f(|x|)$ space bound, has the following membrane structure and initial configuration:

$$[[\hat{q}_0]_w^0 \overbrace{[x_1[x_2\cdots[x_n[\cdots[}^{f(n)\text{ membranes}}\ [\ ]_{h_1}^0 \overbrace{]_h^0\cdots]_h^0}^{} \underbrace{]_h^0\cdots]_h^0}_{n\text{ membranes}}]_h^0]_h^0]_{h_0}^0$$

that is, the same structure of $\mu_{M,x}$ except for an additional membrane h_1 in the innermost position. Such a membrane is used to detect a space "overflow" and halt the simulation if this event occurs, according to the following rules:

$$q\,[\,]_{h_1}^0 \rightarrow [yes]_{h_1}^0 \qquad\qquad \text{for all states } q \text{ of } M \qquad (17)$$

$$[yes]_{h_1}^0 \rightarrow [\,]_{h_1}^0\,yes \qquad\qquad\qquad\qquad\qquad\qquad (18)$$

Furthermore, the same rules (1)–(16) of definition 8 are used, except that rules (13) and (16), involving the outermost membrane, are changed in order to flip the result:

$$[yes]_{h_0}^0 \rightarrow [\,]_{h_0}^0\,no \qquad\qquad\qquad\qquad\qquad\qquad (13')$$

$$[no]_{h_0}^0 \rightarrow [\,]_{h_0}^0\,yes \qquad\qquad\qquad\qquad\qquad\qquad (16')$$

Definition 10. *The P system consisting only of module $\mu'_{M,x,f}$ is denoted by $\Pi'_{M,x,f}$; we also define the family $\mathbf{\Pi}'_{M,f} = \{\Pi'_{M,x,f} : x \in \{0,1\}^\star\}$.*

Lemma 1. *Let $f\colon \mathbb{N} \to \mathbb{N}$, with $f(n) = \Omega(n)$ and $f(n) = O(n^k)$ for some fixed k, be time-constructible; let M be a single-tape Turing machine which halts on every input. Then the family of P systems $\mathbf{\Pi}'_{M,f}$ is semi-uniform, in particular constructible in $O(f(n))$ time, and*

$$L(\mathbf{\Pi}'_{M,f}) = \{x \in \{0,1\}^\star : M \text{ rejects } x \text{ in } f(|x|) \text{ space}\}.$$

Proof. The family $\mathbf{\Pi}'_{M,f}$ is obviously constructible in $O(f(n))$ time (hence semi-uniform) as in the proof of theorem 1, since there is only one extra membrane and the new rules (17)–(18) do not depend on x.

The language decided by $\mathbf{\Pi}'_{M,f}$ is, by construction, the complement of that of M, except that strings x generating computations which require more that $f(|x|)$ cells are rejected.

Another stumbling block we need to overcome is the fact that some Turing machines might operate within the space bound we fixed, but without halting. Fortunately, we know that a single-tape Turing machine, having tape alphabet $\{0,1,\lrcorner\}$ and operating in $f(n)$ space, either halts within $f(n) \cdot |Q| \cdot 3^{f(n)}$ steps (Q being its set of states), or does not halt at all. We can solve the problem by counting the number of simulated steps, and halting the simulation when such time bound is exceeded. The usual solution, i.e., having an object which is successively rewritten into all values of the counter, does not work, as the counter may assume exponentially large values (with respect to n). Hence, a more sophisticated solution is needed.

Definition 11. *We define a P system module κ_n, having the following $(n + 1)$-degree membrane structure and initial configuration:*

$$[\overbrace{[\cdots[}^{n-2}[d]_{c_0}^0]_{c_1}^0 \cdots]_{c_{n-1}}^0]_{c_n}^0$$

The device is, essentially, an $(n + 1)$-bit binary counter. Each membrane corresponds to one bit, c_0 and c_n being the least significant and most significant bits respectively. Neutral and positive polarizations represent 0 and 1, respectively. Thus, in the initial configuration, κ_n stores the value 0. By using communication rules, such value is incremented up to 2^n. Since all membranes $c_1, \ldots, c_{n-1}$ have identical behaviour, they can all be given the same label, thus simplifying the structure (and reducing the time required to construct it) as follows:

$$[\overbrace{[\cdots[}^{n-2}[d]_{c_0}^0]_c^0 \cdots]_c^0]_{c_n}^0$$

Recall that incrementing a binary integer is performed by flipping its bits, one by one and starting from the least significant one, until a 0 is flipped into 1. The object d moves inside the membrane structure in order to perform this task. The following rules (which are identical for membranes labeled by c_0 and c) move d outwards, and change it into d' when the current increment operation has finished:

$$[d]_{c_0}^0 \rightarrow [\,]_{c_0}^+ d' \tag{19}$$

$$[d]_{c_0}^+ \rightarrow [\,]_{c_0}^0 d \tag{20}$$

$$[d]_c^0 \rightarrow [\,]_c^+ d' \tag{21}$$

$$[d]_c^+ \rightarrow [\,]_c^0 d \tag{22}$$

The next rules take d' back to the starting position; when d' re-enters the innermost membrane c_0 it is rewritten into d, and the next increment operation may begin:

$$d' [\,]_c^\alpha \rightarrow [d']_c^\alpha \qquad\qquad \forall \alpha \in \{0, +\} \tag{23}$$

$$d' [\,]_{c_0}^\alpha \rightarrow [d]_{c_0}^\alpha \qquad\qquad \forall \alpha \in \{0, +\} \tag{24}$$

Finally, when d crosses the outermost membrane c_n it is left outside (i.e., there is no rule bringing it back inside), as a signal that the counter has reached the value 2^n:

$$[d]_{c_n}^0 \rightarrow [\,]_{c_n}^+ d \tag{25}$$

Lemma 2. *The P system module κ_n can be constructed in linear time, given the unary representation of n; it sends out the object d after at least 2^n steps.*

Proof. The membrane structure is of linear size, and there is a constant number of communication rules, hence the construction can be performed in $O(n)$ time. Since incrementing the binary counter requires at least two applications of communication rules (and $2n$ in the worst case), the object d is not set out before 2^n time steps have passed.

When the object d is sent out from κ_n, we can use it to stop the simulation of the Turing machine, as we know that if it has not halted yet, then it will never do (assuming we have chosen a suitable value for n). The obvious solution is to use d to dissolve the whole membrane structure $\mu'_{M,x,f}$; however, besides requiring the introduction of dissolution rules (recall that we have only used communication rules so far), there might exist computations during which d is not able to enter a membrane in $\mu'_{M,x,f}$ because it is blocked by the head/state object which continuously enters and exits from that membrane (e.g., if the head of the Turing machine is stuck on a single tape cell). Both problems can be solved by slightly changing the definition of module $\mu'_{M,x,f}$.

We define a P system module $\mu''_{M,x,f}$ with the following membrane structure and initial configuration:

$$[[\hat{q}_0]^0_w \overbrace{[[x_1[[x_2\cdots[[x_n[[\cdots[[[\ ]^0_{h_1}\ ]^0_h]^0_j\cdots]^0_h]^0_j}^{2f(n)\ \text{membranes}} \underbrace{}_{2n\ \text{membranes}} \overbrace{]^0_h]^0_j\cdots]^0_h]^0_j]^0_h]^0_j}^{2f(n)\ \text{membranes}} \underbrace{}_{2n\ \text{membranes}}]^0_{h_0}$$

that is, each membrane labelled by h is surrounded by a further membrane labelled by j. The communication rules used to move the head/state object are changed so that, when it crosses a membrane h (in either direction), it also crosses the membrane j immediately outside *whenever it is neutrally charged*, without any further change. However, when the head/state object crosses a membrane labelled by j that is *positively* charged, it is changed into the object *yes*, so that it can be sent outside as if machine M has accepted.

The module $\mu''_{M,x,f}$ still simulates M on input x with space bound f; the double membrane structure, besides slowing down the simulation by a multiplicative constant, does not alter the simulated computation of M. However, we can combine this module with $\kappa_{\ell(n)}$ (where $\ell(n)$ is a value large enough to ensure that object d is not sent out prematurely) in such a way that when the object d is sent out, it traverses the membrane structure $\mu''_{M,x,f}$ and changes the polarization of all membranes labelled by j to positive, thus stopping the simulation if it has not already ended. Since the head/state object must always cross at least two membranes in order to simulate a transition of M, the situation in which it continuously crosses the same membrane forward and backward, thus blocking the object d, does never happen.

Definition 12. *The P system $\Pi''_{M,x,f}$ is defined as follows:*

$$[\mu''_{M,x,f}\ \kappa_{\ell(n)}]^0_s$$

that is, a skin membrane containing the P system modules $\mu''_{M,x,f}$ and $\kappa_{\ell(n)}$, with the initial configuration and rules given by those of the two modules together. A further pair of rules is used to send out the result from the skin membrane s:

$$[yes]^0_s \rightarrow [\ ]^+_s\ yes \tag{26}$$

$$[no]^0_s \rightarrow [\ ]^+_s\ no \tag{27}$$

As noticed above, the value of $\ell(n)$ must be large enough to ensure that object d is not sent out from $\kappa_{\ell(n)}$ before the result of a possibly halting computation of M is expelled from $\mu''_{M,x,f}$. Since each transition of M can be simulated by at most six steps of $\mu''_{M,x,f}$, and since M may accept when its head is on the rightmost position (the $f(n)$-th position) of the tape, thus requiring us to wait until the result object has travelled through the whole membrane structure, an appropriate value is

$$\ell(n) = \log\left(6 \cdot f(n) \cdot |Q| \cdot 3^{f(n)} + 2f(n) + 1\right).$$

The system is augmented with a set of communication rules which cause the object d, once it has been sent out from $\kappa_{\ell(n)}$, to traverse the nested membrane structure of $\mu''_{M,x,f}$ while changing the polarization of all membranes labelled by j to positive (without changing any other polarization), thus aborting any non-halting simulated computation.

We denote by $\mathbf{\Pi}''_{M,f}$ the family of P systems $\{\Pi''_{M,f,x} : x \in \{0,1\}^\star\}$.

From this definition, and lemmata 1 and 2, we can prove the following result.

Lemma 3. *Let $f\colon \mathbb{N} \to \mathbb{N}$, with $f(n) = \Omega(n)$ and $f(n) = O(n^k)$ for some fixed k, be time-constructible; let M be a single-tape Turing machine (which does not necessarily halt on every input). Then the family of P systems $\mathbf{\Pi}''_{M,f}$ is semi-uniform, in particular constructible in $O(f(n))$ time, and*

$$L(\mathbf{\Pi}''_{M,f}) = \{x \in \{0,1\}^\star : M \text{ rejects } x \text{ in } f(|x|) \text{ space}\}.$$

We are now finally able to prove that $L(f)$ can be recognised by a family of P systems in $O(f(n))$ space.

Theorem 3. *Let $f\colon \mathbb{N} \to \mathbb{N}$, with $f(n) = \Omega(n)$ and $f(n) = O(n^k)$ for some fixed k, be time-constructible. Then $L(f)$ can be decided by a semi-uniform family of P systems $\mathbf{\Pi}_{L(f)}$ using only communication rules, operating in space $O(f(n))$ and constructible in time $O(f(n))$.*

Proof. We only need to prove that the mapping $\langle M \rangle, x \mapsto \Pi''_{M,x,f}$ (i.e., we are given both M and its input, and not only x) can be computed in $O(f(n))$ time. The only feature of $\Pi''_{M,x,f}$ which depends on M (in contrast with other features depending on x) is the set of communication rules. The number of rules is linear with respect to the length of the encoding of M (due to the rules in (5)–(10) and (17)). Assuming a "reasonable" encoding of M, all the communication rules can be constructed in linear time, hence $O(f(n))$ time.

The family $\mathbf{\Pi}_{L(f)}$ of P systems is then constructed in $O(f(n))$ time by the following Turing machine (here described informally):

> If the input x is not of the form $\langle M \rangle 10^\star$, then construct a P system which rejects immediately. Otherwise, construct $\Pi''_{M,x,f}$.

The P systems constructed by this Turing machine work in $O(f(n))$ space, and the thesis follows.

In [17] a simulation algorithm for P systems with active membranes is described. Although the precise space requirements are not detailed (only an asymptotic upper bound is given), by looking at the description of the algorithm one can observe that, essentially, in order to simulate a P system Π we need to store its current configuration, step by step (some auxiliary space is needed; however, it does not exceed the space required by the configuration). Notice that a Turing machine storing the configuration of Π does *not* have the same space requirements as Π itself: indeed, a membrane structure of degree n may require up to $n \log n$ space, since the labels of the membranes (which do not contribute to the space required by Π) need to be stored in order to correctly apply the rules (especially non-elementary division rules); all the labels may be different in the worst case. Keeping in mind this detail, we can prove the following result.

Theorem 4. *Let* $f \colon \mathbb{N} \to \mathbb{N}$*, with* $f(n) = \Omega(n)$ *and* $f(n) = O(n^k)$ *for some fixed* k*, be time-constructible. Then no family* $\mathbf{\Pi}$ *of P systems with active membranes, constructible in* $o(f(n))$ *time and operating in* $o(f(n)/\log f(n))$ *space, decides* $L(f)$*.*

Proof. Suppose otherwise. Let M be the Turing machine constructing $\mathbf{\Pi}$ and consider a Turing machine M' implementing the following algorithm:

> On input x, simulate M on x thus obtaining a description of a P system Π_x deciding whether $x \in L(f)$. Then simulate Π_x and return the same result.

Then $L(M') = L(\mathbf{\Pi}) = L(f)$, and M' has the following space requirements:

$$o\left(\overbrace{f(n)}^{\text{construction}} + \overbrace{\frac{f(n)}{\log f(n)} \log \frac{f(n)}{\log f(n)}}^{\text{simulation}} \right) = o(f(n))$$

This means that M' decides $L(f)$ in $o(f(n))$ space, thus contradicting the space hierarchy theorem.

Notice that there is no restriction on the kind of rules the family $\mathbf{\Pi}$ can use. By combining theorems 3 and 4 we obtain:

Theorem 5. *Let* $f \colon \mathbb{N} \to \mathbb{N}$*, with* $f(n) = \Omega(n)$ *and* $f(n) = O(n^k)$ *for some fixed* k*, be time-constructible. Then there exists a language* L *which can be decided by a semi-uniform family of P systems with active membranes (and using only communication rules) that can be built in* $O(f(n))$ *time and works in* $O(f(n))$ *space. On the other hand,* L *cannot be decided by any family of P systems constructible in* $o(f(n))$ *time and working in* $o(f(n)/\log f(n))$ *space.*

5 Conclusions

In this paper we showed that a deterministic single-tape Turing machine, which operates in polynomial space with respect to the input length, can be efficiently

simulated (both in terms of time and space) by a semi-uniform family of P systems with active membranes and three polarizations. The proposed simulation contains, in our opinion, a very interesting construction which has never been considered before (to the best of our knowledge), and which is exploited to obtain the result: the contents of the cells of the simulated Turing machine are stored in the polarization of the membranes. This allowed us to use only communication rules to compute the result.

Basing upon the above simulation, we proved that a result similar to the *space hierarchy theorem* can be obtained for P systems with active membranes: the larger the amount of space we can use during the computations, the harder the problems we are able to solve.

Several open problems and research directions still remain to be investigated. First of all, the result related to the space (pseudo)-hierarchy for P systems contains a logarithmic factor, which arises from the simulation we proposed. Can we avoid such a factor, thus obtaining a theorem which exactly corresponds to the space hierarchy theorem related to Turing machines?

Following this direction, we could also consider different classes of P systems with active membranes (e.g., using different parallel semantics), and check whether the space (pseudo)-hierarchy theorem still holds for such classes.

As for the simulation of the single-tape deterministic Turing machine we presented in section 3, we conjecture that it can be extended to consider nondeterministic Turing machines, as well as multi-tape Turing machines, to obtain efficient simulations both in terms of time and space also in these cases. It would also be interesting to consider if such an efficient simulation can be performed for other different computational models.

References

1. A. Alhazov, C. Martín-Vide, L. Pan: Solving a **PSPACE**-complete problem by recognizing P systems with restricted active membranes. *Fundamenta Informaticae*, **58**(2):67–77, 2003.
2. A. Alhazov, R. Freund: On efficiency of P systems with active membranes and two polarizations. In: G. Mauri, Gh. Păun, M. J. Pérez-Jiménez, G. Rozenberg, A. Salomaa, eds., *Membrane Computing, Fifth International Workshop, WMC 2004*, LNCS 3365, Springer-Verlag, Berlin, 2005, pp. 81–94.
3. J. L. Balcázar, J. Díaz, J. Gabarró: *Structural complexity I* (second edition), Springer, 1995.
4. 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, LNCS 3699, Springer-Verlag, 2005, pp. 105–116.
5. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, F. J. Romero-Campero: On the power of dissolution in P systems with active membranes. In: R. Freund, Gh. Păun, G. Rozenberg, A. Salomaa, eds., *Membrane Computing, Sixth*

International Workshop, WMC 2005, LNCS 3850, Springer-Verlag, Berlin, 2006, pp. 224–240.

6. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, F. J. Romero-Campero, A. Romero-Jiménez: Characterizing tractability by cell-like membrane systems. In: K. G. Subramanian, K. Rangarajan, M. Mukund, eds., *Formal models, languages and applications*, Series in Machine Perception and Artificial Intelligence, Vol. 66, World Scientific, 2006, pp. 137–154.

7. O. H. Ibarra: On the computational complexity of membrane systems, *Theoretical Computer Science*, **320**:89–104, 2004.

8. K. Kobayashi: On proving time constructibility of functions. *Theoretical Computer Science*, **35**:215–225, 1985.

9. S. N. Krishna, R. Rama: A variant of P systems with active membranes: solving **NP**-complete problems. *Romanian Journal of Information Science and Technology*, **2**(4):357–367, 1999.

10. A. Obtulowicz: Deterministic P systems for solving SAT problem. *Romanian Journal of Information Science and Technology*, **4**(1–2):551–558, 2001.

11. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, **1**(61):108–143, 2000. See also Turku Centre for Computer Science – TUCS Report No. 208, 1998.

12. Gh. Păun: P Systems with active membranes: attacking **NP**-complete problems. *Journal of Automata, Languages and Combinatorics*, **6**(1):75–90, 2001.

13. Gh. Păun: *Membrane computing. An introduction.* Springer-Verlag, Berlin, 2002.

14. M. J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: The **P** versus **NP** problem through cellular computing with membranes. In: N. Jonoska, Gh. Păun, G. Rozenberg, eds., *Aspects of Molecular Computing*, LNCS 2950, Springer-Verlag, Berlin, 2004, pp. 338–352.

15. M. J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: A polynomial complexity class in P systems using membrane division. In: E. Csuhaj-Varjú, C. Kintala, D. Wotschke, G. Vaszil, eds., *Proceedings of the Fifth Workshop on Descriptional Complexity of Formal Systems, DCFS 2003*, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, 2003, pp. 284–294.

16. A. E. Porreca, A. Leporati, G. Mauri, C. Zandron: Introducing a space complexity measure for P systems, *International Journal of Computers, Communications & Control*, **4**(3):301–310, 2009.

17. A. E. Porreca, G. Mauri, C. Zandron: Complexity classes for membrane systems. *RAIRO Theoretical Informatics and Applications*, **40**(2):141–162, 2006.

18. M. Sipser: *Introduction to the theory of computation* (second edition), Course Technology, 2005.

19. P. Sosík: The computational power of cell division. *Natural Computing*, **2**(3):287–298, 2003.

20. C. Zandron, C. Ferretti, G. Mauri: Solving **NP**-complete problems using P systems with active membranes. In: I. Antoniou, C. S. Calude, M. J. Dinneen, eds., *Unconventional Models of Computation*, Springer-Verlag, Berlin, 2000, pp. 289–301.

21. 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*, **87**(1):79–91, 2008.

22. The P systems Web page: `http://ppage.psystems.eu`