Standardized Proofs of PSPACE-completeness of P Systems with Active Membranes

Petr Sosík[1,2], Alfonso Rodríguez-Patón[1], Lucie Ciencialová[2]

[1] Departamento de Inteligencia Artificial, Facultad de Informática
Universidad Politécnica de Madrid, Campus de Montegancedo s/n
Boadilla del Monte, 28660 Madrid, Spain
[2] Institute of Computer Science, Faculty of Philosophy and Science, Silesian University in Opava, 74601 Opava, Czech Republic

Summary. Two proofs have been shown for P systems with active membranes in previously published papers, demonstrating that these P systems can solve in polynomial time exactly the class of problems **PSPACE**. Consequently, these P systems are equivalent (up to a polynomial time reduction) to Second Machine Class models as the alternating Turing machine or the PRAM computer. These proofs were based on a modified definition of uniform families of P systems. Here we demonstrate that the results remain valid also in the case of standard definitions.

1 Introduction

P systems with active membranes are among computationally most powerful models of P systems. It has been shown that this model, in its standard definition, can solve the PSPACE-complete problem QSAT in a polynomial time [8, 1]. Later on, the paper [10] demonstrated that uniform families of P systems with active membranes can solve in polynomial time exactly the class of problems **PSPACE**. Consequently, these P systems satisfy the *Parallel Computation Thesis* [2]:

$$M\text{-PTIME} = M\text{-NPTIME} = \mathbf{PSPACE}, \tag{1}$$

where M-(N)PTIME is the class of problems solved in polynomial time by a (non-)deterministic machine M. We recall that computers satisfying (1) form the *second machine class*, whose members are the alternating Turing machine, SIMDAG (also known as SIMD PRAM) and other parallel models [2].

However, the papers [8, 10] used a slightly modified version of definition of uniform families of membrane systems. Besides different structure of definitions, the main functional differences between the definition considered standard and presented, e.g., in [5] were these:

1. While both definitions require each P system – a member of a uniform family – to halt, the standard definition requires also the system to produce a distinguished object *yes* or *no* in the last step, telling whether the computation was accepting or not. Our definition, on the contrary, only required the object *yes* in the accepting case.
2. All members of a family must be produced by one and the same Turing machine in the standard definition, while our formulation allowed that different Turing machines might be used for different family members. This possibility, however, was never actually considered and used in our proofs as this would be clearly contra-intuitive to the commonly accepted sense of uniformity.

Therefore, the modification of the proofs presented here focuses on the first mentioned difference and makes the proofs compatible with the standard definition given in the next section.

2 Definitions

A P system with active membranes [7] is a construct

$$\Pi = (V, H, \mu, w_1, \ldots, w_m, R),$$

where:

(i) $m \geq 1$;
(ii) V is an alphabet;
(iii) H is a finite set of *labels* for membranes;
(iv) μ is a *membrane structure*, consisting of m membranes, labelled (not necessarily in a one-to-one manner) with elements of H; all membranes in μ are supposed to be neutral;
(v) $w_1, \ldots, w_m$ are strings over V, describing the *multisets of objects* placed in the regions of μ;
(vi) R is a finite set of *developmental rules*, of the following forms:
   (a) $[_h a \to v]^{\alpha}_h$,
   for $h \in H, \alpha \in \{+, -, 0\}, a \in V, v \in V^*$
   (object evolution rules, associated with membranes and depending on the label and the charge of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part to the application of these rules nor are they modified by them);
   (b) $a[_h\,]^{\alpha_1}_h \to [_h b]^{\alpha_2}_h$,
   for $h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a, b \in V$
   (communication rules; an object is introduced into the membrane, maybe modified during this process; also, the polarization of the membrane can be modified, but not its label);

(c) $[_h a\,]_h^{\alpha_1} \to [_h\,]_h^{\alpha_2} b$,
for $h \in H, \alpha_1, \alpha_2 \in \{+, -, 0\}, a, b \in V$
(communication rules; an object is sent out of the membrane, maybe modified during this process; also, the polarization of the membrane can be modified, but not its label);

(d) $[_h a\,]_h^{\alpha} \to b$,
for $h \in H, \alpha \in \{+, -, 0\}, a, b \in V$
(dissolving rules; in reaction with an object, a membrane can be dissolved, leaving all its object in the surrounding region, while the object specified in the rule can be modified);

(e) $[_h a\,]_h^{\alpha_1} \to [_h b\,]_h^{\alpha_2} [_h c\,]_h^{\alpha_3}$,
for $h \in H, \alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}, a, b, c \in V$
(division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, maybe of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects; all the other objects are copied into both resulting membranes);

(f) $[_{h_0}[_{h_1}\,]_{h_1}^{+} \cdots [_{h_k}\,]_{h_k}^{+}\, [_{h_{k+1}}\,]_{h_{k+1}}^{-} \cdots [_{h_n}\,]_{h_n}^{-}]_{h_0}^{\alpha_2}$
$\to [_{h_0}[_{h_1}\,]_{h_1}^{\alpha_3} \cdots [_{h_k}\,]_{h_k}^{\alpha_3}]_{h_0}^{\alpha_5}\, [_{h_0}[_{h_{k+1}}\,]_{h_{k+1}}^{\alpha_4} \cdots [_{h_n}\,]_{h_n}^{\alpha_4}]_{h_0}^{\alpha_6}$,
for $n > k \geq 1$, $h_i \in H$, $0 \leq i \leq n$, and $\alpha_2, \ldots, \alpha_6 \in \{+, -, 0\}$;
(division of non-elementary membranes; this is possible only if a membrane contains two immediately lower membranes of opposite polarization, + and −; the membranes of opposite polarizations are separated in the two new membranes, but their polarization can change; all membranes of opposite polarizations are always separated by applying this rule;
if the membrane labelled h_0 contains other membranes than $h_1, \ldots, h_n$ specified above, then they must have neutral charges in order to make this rule applicable; these membranes are duplicated and then become part of the content of both copies of membrane h_0).

All the above rules are applied in parallel, but at one step, an object a can be subject to only one rule of type (a)–(e) and a membrane h can be subject to only one rule of type (b)–(f). In the case of type (f) rules, this means that none of the membranes $h_0, \ldots, h_n$ listed in the rule can be simultaneously subject to another rule of type (b)–(f). However, this restriction do not apply to membranes with neutral charge contained in h_0. In general, an application of the rules is performed as follows:

1. In every step, first the rules are assigned to objects and membranes in a maximal way (any object and membrane which can evolve by a rule of any form, should evolve), and then all the rules are simultaneously applied;
2. all objects and membranes which cannot evolve pass unchanged to the next step;

3. if a rule of type (d), (e) or (f) is applied to a membrane, then rules of type (a) are applied first to its objects and then the resulting objects are further copied/moved in accordance with the (d), (e) or (f) type rule;
4. the skin membrane can neither be dissolved nor divided, nor it can introduce an object from outside (unless stated otherwise). Therefore, we assume that there are only rules of types (a) and (c) associated with the skin membrane.

The membrane structure of Π at a given moment, together with all multisets of objects contained in its regions, form the *configuration* of the system. The $(m+1)$-tuple $(\mu, w_1, \ldots, w_m)$ is the *initial configuration*. We can pass from one configuration to another by using the rules from R according to the principles given above. The computation stops when there is no rule which can be applied to objects and membranes in the last configuration.

In this paper we study the accepting (or recognizer) variant of P systems. A *recognizer P system* solving decision problems must comply with the following requirements: (a) the working alphabet contains two distinguished elements *yes* and *no*; (b) all computations halt; and (c) exactly one of the object *yes* (accepting computation) or *no* (rejecting computation) must be sent to the output region of the system, and only at the last step of each computation. In our case of systems with active membranes, the outer environment of the system is taken as the output region.

2.1 Families of membrane systems

Consider a decision problem $X = (I_X, \theta_X)$ where I_X is a language over a finite alphabet (whose elements are called instances) and θ_X is a total boolean function over I_X.

Definition 1. *[5] A family* $\mathbf{\Pi} = \{\Pi(w) : w \in I_X\}$ *of recognizer membrane systems without input membrane is* polynomially uniform by Turing machines *if there exists a deterministic Turing machine working in polynomial time which constructs the system* $\Pi(w)$ *from the instance* $w \in I_X$.

In the sequel we will for short denote such a family just as *uniform*.

In this paper we deal with recognizer systems without input membrane, i.e., an instance w of a problem X is encoded into the structure of the P system $\Pi(w)$. The system $\Pi(w)$ is supposed to solve the instance w. Formally, [5] defines the conditions of *soundness* and *completeness* of $\mathbf{\Pi}$ with respect to X. A conjunction of these two conditions ensures that for every $w \in I_X$, if $\theta_X(w) = 1$, then every computation of $\Pi(w)$ is accepting, and if $\theta_X(w) = 0$, then every computation of $\Pi(w)$ is rejecting.

Note that the system $\Pi(w)$ can be generally nondeterministic, i.e, it may have different possible computations, but with the same result. Such a P system is also called *confluent*.

Definition 2. *[5] A decision problem X is solvable in polynomial time by a family of recognizer P systems belonging to a class $\mathcal{R}$ without input membrane $\mathbf{\Pi} = \{\Pi(w) : w \in I_X\}$, denoted by $X \in \mathbf{PMC}^*_{\mathcal{R}}$, if the following holds:*

- *The family $\mathbf{\Pi}$ is polynomially uniform by Turing machines.*
- *The family $\mathbf{\Pi}$ is polynomially bounded; that is, there exists a natural number $k \in \mathbb{N}$ such that for each instance $w \in I_X$, every computation of $\Pi(w)$ performs at most $|w|^k$ steps.*
- *The family $\mathbf{\Pi}$ is sound and complete with respect to X.*

The family $\mathbf{\Pi}$ is said to provide a *semi-uniform solution* to the problem X. In this case, for each instance of X we have a special P system.

3 P Systems with Active Membranes Solving QSAT

The QSAT (satisfiability of quantified propositional formulas) is a well-known **PSPACE**-complete problem. It asks whether or not a given quantified boolean formula in the conjunctive normal form assumes the value *true*. A formula as above is of the form

$$\gamma = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n (C_1 \wedge C_2 \wedge \ldots \wedge C_m), \tag{2}$$

where each Q_i, $1 \leq i \leq n$, is either $\forall$ or $\exists$, and each C_j, $1 \leq j \leq m$, is a *clause* of the form of a disjunction

$$C_j = y_1 \vee y_2 \vee \ldots \vee y_r,$$

with each y_k being either a propositional variable, x_s, or its negation, $\sim x_s$. For example, let us consider the propositional formula

$$\beta = Q_1 x_1 Q_2 x_2 [(x_1 \vee x_2) \wedge (\sim x_1 \vee \sim x_2)]$$

It is easy to see that it is *true* when $Q_1 = \forall$ and $Q_2 = \exists$, but it is *false* when $Q_1 = \exists$ and $Q_2 = \forall$.

By adding dummy variables, each such formula can be rewritten such that the quantifiers alternate: $Q_1 = \exists$, $Q_2 = \forall$, $Q_3 = \exists$, $Q_4 = \forall$ etc. We assume this normal form for the formulas considered in the sequel.

Theorem 1 ([8]). *There exists a uniform family of recognizer P systems with active membranes providing a semi-uniform solution to QSAT in a time linear in the number of variables and the number of clauses.*

Proof. The following proof differs from the original one published in [8] mostly in omitting the original paragraph 1, modifying paragraph 8 (here paragraph 7) and adding a new paragraph 8.

Consider a propositional formula γ of the form (2) with

$$C_i = y_{i,1} \vee \ldots \vee y_{i,p_i},$$

for some $p_i \geq 1$, and $y_{i,j} \in \{x_k, \sim x_k \mid 1 \leq k \leq n\}$, for each $1 \leq i \leq m$, $1 \leq j \leq p_i$. We construct the P system

$$\Pi = (V, H, \mu, w_0, w_1, \ldots, w_m, w_{m+n+1}, R)$$

with the components

$$\begin{aligned} V &= \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{c_i \mid 0 \leq i \leq 4n + 2m + 2\} \cup \{t, s, yes, no\}, \\ H &= \{0, 1, \ldots, m + n + 1\}, \\ \mu &= [_{m+n+1}[_{m+n} \cdots [_1[_0]_0^0]_1^0 \cdots]_{m+n}^0]_{m+n+1}^0, \\ w_0 &= c_0, \\ w_i &= \lambda, \text{ for all } i = 1, 2, \ldots, m + n, \\ w_{m+n+1} &= b, \end{aligned}$$

while the set R contains the following rules:

1. $[_0 c_i \to a_{i/2+1} c_{i+1}]_0^\alpha$, for all $0 \leq i < 2n$, i even, $\alpha \in \{+, -, 0\}$, and

   $[_0 c_i \to c_{i+1}]_0^\alpha$, for all $0 \leq i < 2n$, i odd, or $2n \leq i \leq 2n + m - 1$, $\alpha \in \{+, -, 0\}$

   (we count to $2n + m$, which is the time needed for producing all 2^n truth-assignments for the n variables, as well as 2^n membrane sub-structures which will examine the truth value of formula γ for each of these truth-assignments; this counting is done in the central membrane, irrespective which is its polarity; moreover during first n odd steps, symbols $a_1, \ldots a_n$ are subsequently produced);
2. $[_0 a_i]_0^0 \to [_0 t_i]_0^+ [_0 f_i]_0^-$, for all $1 \leq i \leq n$

   (in membrane 0, when it is "electrically neutral" we subsequently choose each variable x_i, $1 \leq i \leq n$, and both values *true* and *false* are associated with it, in the form of objects t_i, f_i, which are separated in two membranes with the label 0 which differ only by these objects t_i, f_i and by their charge);
3. $[_{i+1}[_i]_i^+ [_i]_i^-]_{i+1}^0 \to [_{i+1}[_i]_i^0]_{i+1}^+ [_{i+1}[_i]_i^0]_{i+1}^-$, for all $0 \leq i \leq m + n - 1$

   (division rules for membranes labeled with $0, 1, \ldots, m + n$; the opposite polarization introduced when dividing a membrane 0 is propagated from lower levels to upper levels of the membrane structure and the membranes are continuously divided until also membrane $m+n$ has been divided; this membrane remains polarized and hence may be never divided again; in the following cycle of the division process, the same holds for the membrane $m+n-1$ and so on, resulting in the structure at Figure 1 after $2n + m$ steps);
4. $[_0 c_{2n+m}]_0^0 \to t$

   (after $2n + m$ steps, each copy of membrane 0 is dissolved and the contents is released into the surrounding membrane, which is labeled with 1);

5. $[_j t_i]^0_j \to t_i$, if x_i appears in clause C_j, $1 \leq i \leq n, 1 \leq j \leq m$, and

   $[_j f_i]^0_j \to f_i$, if $\sim x_i$ appears in clause C_j, $1 \leq i \leq n, 1 \leq j \leq m$

   (a membrane with label j, $1 \leq j \leq m$, is dissolved if and only if clause C_j is satisfied by the current truth-assignment; if this is the case, then the truth values associated with the variables are released in the surrounding membrane, that associated with the next clause, C_{j+1}, otherwise these truth values remain blocked in membrane j and never used at the next steps by the membranes placed above; note that, as we will see immediately, after $2n+m$ steps we have 2^n membrane sub-structures of the form $[_m[_{m-1} \cdots [_1]^0_1 \cdots]^0_{m-1}]^0_m$ working in parallel; each of them is connected to a leaf of the binary tree membrane structure as in Figure 1);
6. $[_{m+1} t]^\alpha_{m+1} \to [_{m+1}]^\alpha_{m+1} t$, $\alpha \in \{+, -\}$

   (together with the truth-assignments, we also have the object t, which can be passed from a level to the upper one only by dissolving membranes; this object reaches the level $m+1$ if only if all membranes in a sub-structure of the form $[_m[_{m-1} \cdots [_1]^0_1 \cdots]^0_{m-1}]^0_m$ are dissolved, which means that the associated truth-assignment has satisfied all the clauses);
7. $[_i t]^\alpha_i \to [_i]^0_i s$, $[_i t]^0_i \to [_i]^0_i t$, if $Q_{m+n+2-i} = \forall$, $\alpha \in \{+, -\}$, $m+2 \leq i \leq m+n$, and

   $[_i t]^\alpha_i \to [_i]^0_i t$, if $Q_{m+n+2-i} = \exists$, $\alpha \in \{+, -\}$, $m+2 \leq i \leq m+n$

   (a membrane $[_i]_i$ corresponds to the quantifier $Q_j x_j$, where $j = m+n+2-i$; if $Q_j = \forall$, the object t is passed to the upper level only if it comes from both lower level membranes, i.e. the respective clauses are satisfied for both truth values of x_j; if $Q_j = \exists$, then the object t coming from lower level is sent up);
8. $[_{m+n+1} c_i \to c_{i+1}]^0_{m+n+1}$, for all i, $0 \leq i < 4n + 2m + 2$,

   $[_{m+n+1} c_{4n+2m+2}]^0_{m+n+1} \to [_{m+n+1}]^-_{m+n+1} no$,

   $[_{m+n+1} t]^0_{m+n+1} \to [_{m+n+1}]^+_{m+n+1} yes$

   (objects c_i in the region enclosed by the skin membrane act as a clock; if the object t reaches this region within $4n + 2m + 2$ steps, signalling that the formula evaluates to *true*, then the object *yes* is expeled from the system, otherwise the object *no* is expeled after $4n + 2m + 2$ steps. In both cases the systems immediately halts.

From the previous explanations one can see that the object *yes* (*no*) leaves the system in the last step if and only if formula γ evaluates to *true* (*false*, respectively). This is achieved in $3n + \lfloor n/2 \rfloor + 2m + 2$ steps:

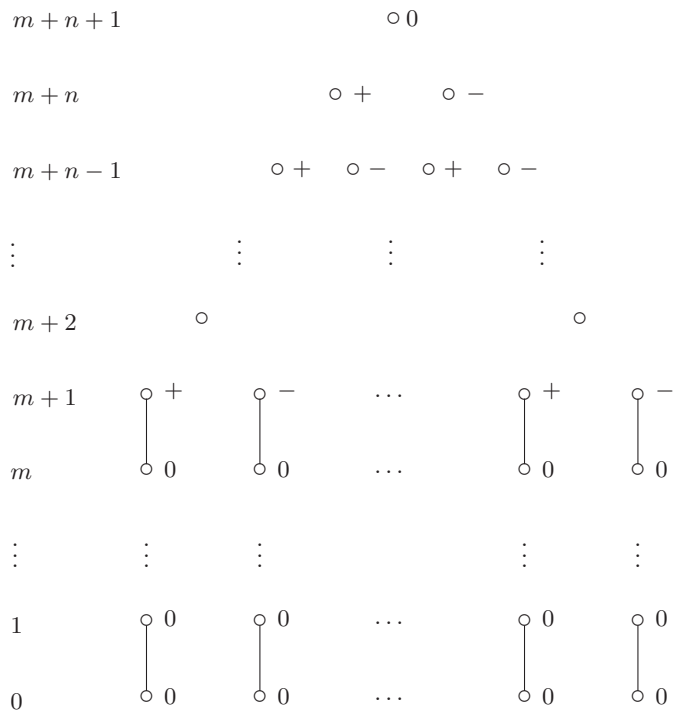

Fig. 1. The membrane structure of the system Π after $2n+m$ steps.

- in $2n+m$ steps we create the membrane structure at Figure 1 (as well as the 2^n different truth-assignments)
- then we dissolve all membranes 0 (one step)
- we check the satisfiability of each clause for each truth-assignment, in parallel in the 2^n sub-structures ($m+1$ steps)
- we check whether all quantifiers are satisfied by propagating objects t through the indicated binary tree structure; one step is need for each of $\lceil n/2 \rceil$ quantifiers $\exists$, while two steps are necessary for each of $\lfloor n/2 \rfloor$ quantifiers $\forall$.

The arguments given above ensure that the system Π is polynomially bounded and that the family of these P systems is complete and sound with respect to the problem QSAT. Finally, the family is polynomially uniform by Turing machines as the above construction can be performed by an algorithm which would run on a classical computer (and hence also on Turing machine) in a polynomial time, having as input a propositional formula γ (an instance of QSAT) and which would output the description of the system Π. Note that both the size of the alphabet V and the number of membranes in the initial configuration is $\mathcal{O}(n+m)$, and the number of rules is $\mathcal{O}(nm)$, which determines the time necessary for the con-

struction. Since the construction can be done step-by step without a need to store previous steps, the space needed is $\mathcal{O}(\log n + \log m)$.

Corollary 1 ([8]). PSPACE $\subseteq$ PMC$^S_{\mathcal{AM}}$.

4 Simulation of P Systems with Active Membranes in Polynomial Space

In this section we show that the inclusion reverse to Corollary 1 hold as well. We employ the technique of reverse-time simulation. Instead of simulating a computation of a P system from its initial configuration onwards (which would require an exponential space for storing configurations), we create a recursive function which returns the state of any membrane h after a given number of steps. The recursive calls evaluate contents of the membranes interacting with h in a reverse time order (towards the initial configuration). In such a manner we do not need to store a state of any membrane, but instead we calculate it recursively whenever it is needed. In this way a result of any $T(n)$-time-bounded computation of a recognizer P system with active membranes can be found in a space polynomial to $T(n)$.

Theorem 2 ([10]). PMC$^S_{\mathcal{AM}} \subseteq$ **PSPACE**.

Proof. The proof of this result published in [10] remains unchanged under the standard definition, except the paragraph at p. 149 under the subtitle "Space complexity of the simulation", starting with "Consider an instance of a size s...". This paragraph should be reformulated as follows:

Consider a decision problem which is, by assumption, solved by a uniform family of P system with active membranes in a semi-uniform way. Each instance of a size s is solved by a P system $\Pi = (V, H, \mu, w_1, \ldots, w_m, R)$ of size $s^{\mathcal{O}(1)}$, a member of the family. The result of computation of Π can be calculated with the aid of the function `State`. Let h_0 be the skin membrane of Π. One can subsequently calculate `State`(h_0, n) for $n = 0, 1, 2 \ldots$ until the object *yes* or *no* is expelled from h_0 using the rule of type (c). We determine the space complexity of the function `State`. Let... □

Together with Corollary 1 we obtain the parallel computation thesis for uniform families of recognizer P systems with active membranes:

Corollary 2. PMC$^S_{\mathcal{AM}} =$ **PSPACE**.

5 Concluding Remarks

Since the publication of papers [8, 10], similar results linking the class PSPACE with other types of membrane systems have been presented, see, e.g., [3, 9]. The

proof technique we have used in Theorem 2 is applicable also to other variants of P systems.

Finally, we note that the characterization of power of non-confluent P systems with active membranes remains still open. The presented proof cannot be simply adapted to this case by using a non-deterministic Turing machine. The reason is that we cannot store non-deterministic choices of such a P system along a chosen trace of computation, as this would require an exponential space. It is possible that non-confluent P systems with active membranes might capture in polynomial time the class **NEXPTIME**.

Acknowledgements

Authors are grateful to Mario Pérez-Jiménez for discussions and suggestions concerning complexity issues and uniform families of membrane systems. The research was partially supported by the Ministerio de Ciencia e Innovación (MICINN), Spain, under project TIN2009–14421, by the program I3, by the Comunidad de Madrid (grant No. CCG06-UPM/TIC-0386 to the LIA research group), by the Czech Science Foundation, grant No. 201/09/P075, and by the Silesian University in Opava, grant No. SGS/4/2010.

References

1. A. Alhazov, C. Martín-Vide, and L. Pan. Solving a PSPACE-complete problem by P systems with restricted active membranes. *Fundamenta Informaticae*, 58(2):67–77, 2003.
2. P. van Emde Boas. Machine models and simulations. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A: Algorithms and Complexity, pages 1–66. Elsevier, Amsterdam, 1990.
3. T.-O. Ishdorj, A. Leporati, L. Pan, X. Zeng, and X. Zhang. Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources. In Martínez-del-Amor et al. [4], pages 1–27. Volume 2.
4. M.A. Martínez-del-Amor, E.F. Orejuela-Pinedo, G. Păun, I. Pérez-Hurtado, and A. Riscos-Núnez, editors. *Seventh Brainstorming Week on Membrane Computing*, Sevilla, 2009. Fenix Editora.
5. M.J. Pérez-Jiménez. A computational complexity theory in membrane computing. In Păun et al. [6], pages 125–148.
6. G. Păun, M.J. Pérez-Jiménez, A. Riscos-Núnez, G. Rozenberg, and A. Salomaa, editors. *Membrane Computing, 10th International Workshop, WMC 2009*, volume 5957 of *Lecture Notes in Computer Science*, Berlin, 2010. Springer.
7. Gh. Păun. P systems with active membranes: attacking NP-complete problems. *J. Automata, Languages and Combinatorics*, 6 (1):75–90, 2001.
8. P. Sosík. The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing*, 2(3):287–298, 2003.
9. P. Sosík, A. Păun, A. Rodríguez-Patón, and D. Pérez. On the power of computing with proteins on membranes. In Păun et al. [6], pages 448–460.
10. P. Sosík and A. Rodríguez-Patón. Membrane computing and complexity theory: A characterization of PSPACE. *J. Comput. System Sci.*, 73(1):137–152, 2007.