# Membrane Computing: New Results, New Problems

**Carlos MARTÍN-VIDE[a], Andrei PĂUN[b], Gheorghe PĂUN[a,c]**

[(a)]Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
E-mail: cmv@astor.urv.es

[(b)]Department of Computer Science
University of London Ontario
London, Ontario, Canada N6A 5B7
E-mail: apaun@csd.uwo.ca

[(c)]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro, gp@astor.urv.es

**Abstract.** We briefly discuss some of the recent results in the membrane computing, mainly reported during the Workshop on Membrane Computing, Curtea de Argeş, Romania, August 19–23, 2002 (we refer below to this workshop by WMC2002). Some of these results answer problems which were open in this area, but they also suggest new open problems and research topics.

## 1 Introduction

Membrane computing is a rather young branch of natural computing (the systematic research started in November 1998, when the paper [16] was circulated on the web), but it is already well developed at the theoretical level. However, we do not recall here once again the basic ideas, definitions, central results of this field. A comprehensive information about membrane computing (complete bibliography, papers to be downloaded, addresses of authors, news, etc.) can be found at the web address http://bioinformatics.bio.disco.unimib.it/psystems. Two surveys have appeared also in the present Bulletin ([15], [12]), a friendly guide-introduction is [18], while the recent monograph [17] covers the central notions and results in the area at the level of February-March 2002.

Thus, we assume the reader familiar with membrane computing, we refer to the above mentioned bibliographical sources for details, and we are going directly to mention some recent results and some suggestions for future research in the area derived from these results.

## 2 Catalysts Suffice

We start by considering one of the most surprising recent result, that from [22], stating that P systems with symbol-objects using catalysts (but no other features, such as priorities, membrane permeability control, etc) are computationally complete, they can compute all recursively enumerable sets of natural numbers.

We recall some notations from [17]. We denote by $NOP_m(\alpha, tar, pri)$ the family of sets of natural numbers which can be computed by P systems with at most $m \geq 1$ membranes, using rules of the form $\alpha$, target indications of the form *here, out*, $in_j$, and priorities among the rules; $\alpha$ is one of *ncoo, cat, coo*, indicating that the rules are non-cooperating (context-free), catalytic, cooperative, respectively. If $m = 1$, then we omit the indication *tar*; also, *pri* is omitted when no priority relation is considered. If the number of membranes is not bounded, then the subscript $m$ is replaced with $*$.

For a family $FL$ of languages, we denote by $NFL$ the family of length sets of languages in $FL$. By $CF, RE$ we denote the families of context-free and of recursively enumerable languages, respectively.

The following results appear in [17]:

1. $NOP_*(ncoo, tar) = NOP_1(ncoo) = NCF$ (Theorem 3.3.2),
2. $NOP_1(coo) = NRE$ (Theorem 3.3.3),
3. $NOP_2(cat, tar, pri) = NRE$ (Theorem 3.4.3)

and one formulates the open problem (Q1, pages 63 and 70) of finding the size of the families $NOP_m(cat, tar)$. Moreover, the conjecture was that these families are larger than $NCF$ but not equal to $NRE$. However, the main result of [22] is the following:

**Theorem 1.** $NOP_2(cat, tar) = NRE$.

The proof is carried out by simulating register machines by P systems with catalysts. (The result in [22] is formulated for one membrane only, but the catalysts are ignored when counting the objects in the halting configuration, which is a difference from the definitions from [17]. One further membrane is necessary in order to work with the same definition: use an additional membrane for separating the objects which are counted for the result of a computation from the catalysts.)

Of course, when having two membranes only, the membrane structure can have a unique shape (one inner membrane embedded in the skin membrane), hence the communication is done by using the weak target indications *here, out,in*, hence *tar* can be replaced with *i/o*.

The result from [22] has important implications in the study of the generative power of P systems with symbol-objects: in order to obtain the computational universality we do not need further ingredients, such as controls on rule application (priorities, permitting or forbidding conditions, rule creation, etc) or on communication (membrane permeability control, conditional communication, etc). Several results from [17], if considered in the present formulation, are direct consequences of Theorem 1 above.

The "explanation" of this result lies in the way of defining the computations in a P system: each transition means a maximally parallel use of rules, chosen in a nondeterministic manner, while a computation is successful only if it halts, a configuration is

reached where no rule can be applied. Using these features, and objects and catalysts of different "colours" (primed and non-primed), one can construct a P system which simulates a register machine in such a way that only correct simulations can lead to halting computations. The reader is referred to [22] for the rather non-trivial details of the proof.

However, there is a price to be paid in order to obtain the universality with catalysts only: we need at least six catalysts. In comparison, the number of catalysts used in the proofs of universality from [17] for systems involving further features is in most cases one (see Theorems 3.4.3, 3.6.1, 3.6.4, etc.). There also are several variants of membrane systems for which the universality is obtained without using catalysts; this is the case for controls on rule application of a stronger type (Theorems 3.6.3, 3.6.5 – 3.6.8, 3.7.3, etc). The catalysts corresponds to enzymes which control the cell biochemistry, and it is known that the enzymes need a specific environment (specific reaction conditions) in order to be active, and this makes improbable the simultaneous work of several enzymes in the same compartment. These observations lead to the *open problem* – also formulated in [22] – of finding a proof of Theorem 1 with a smaller number of catalysts. Otherwise stated, in the notation of families $NOP_m(cat, \ldots)$ we also have to take into account the number of catalysts, for instance, writing $NOP_m(cat_s, \ldots)$ in the case when at most $s$ catalysts are used, and then to look for universality results with a small $s$. A trade-off is expected between the number of catalysts and other features of the systems. Does such a trade-off relation exist between the number of membranes and the number of catalysts? For instance, the result from Theorem 1 can be written in the form $NOP_2(cat_6, i/o) = NRE$. Is a result of the form $NOP_m(cat_s, i/o) = NRE$ valid, for some $m \geq 2$ and $s < 6$? If not (or, as expected, if $s$ remains greater than 1), then we still have a motivation for introducing additional features, such as priorities, membrane thickness control, and so on (of course, motivations for having such ingredients can come from other directions: adequacy of P systems to the bio-reality, computational efficiency, length and transparency of proofs, etc.).

# 3 Communication Suffices

The assertion in the title of this section is not new: it is already known that P systems with symbol-objects which are never changed, but only passed through membranes, in a well-specified manner, are computationally universal. This is the case of P systems with symport/antiport rules, and of systems with carriers. Details can be found in [17] – and, of course, in the web page mentioned in the Introduction. However, several of the results from [17] were already improved, while also new types of P systems where the computations are only done by communication, by moving objects through membranes, have been introduced.

This way of computing is interesting from various points of view: it is biologically well motivated (symport and antiport are well known biological processes), it essentially involves the environment of the systems (and this is the case also in reality), it observes the conservation law (no objects is created from nothing or destroyed during a computation), it proves the power of communication.

We remind the fact that a symport rule is of the form $(x, in)$ or $(x, out)$, with the meaning that the objects specified by $x$ enter, respectively, exit the membrane with which the rule is associated, while an antiport rule is of the form $(y, out; z, in)$, with the meaning that the objects specified by $y$ exit and those specified by $z$ enter the membrane. The family of sets of natural numbers computed by P systems with at most $m$ membranes, using symport rules with strings $x$ of length at most $s$ and antiport rules with strings $y, z$ of length at most $r$ is denoted by $NOP_m(sym_s, anti_r)$.

The following results were mentioned in [17]:

1. $NOP_2(sym_2, anti_2) = NRE$ (Theorem 4.2.1),
2. $NOP_5(sym_2, anti_0) = NRE$ (Theorem 4.2.2),
3. $NOP_2(sym_5, anti_0) = NRE$ (Theorem 4.2.3),
4. $NOP_3(sym_4, anti_0) = NRE$ (Theorem 4.2.4),
5. $NOP_3(sym_0, anti_2) = N'RE$ (Theorem 4.2.5),

where $N'RE$ is the family of recursively enumerable sets of non-zero natural numbers.

Problem Q8 (page 140) from [17] asked whether or not these results can be improved – and the conjecture was that the answer is affirmative. The confirmation came from both [6] and [14]. The following results are given in [6]:

**Theorem 2.** (i) $NOP_1(sym_1, anti_2) = NRE$, (ii) $NOP_4(sym_2, anti_0) = NRE$, (iii) $NOP_2(sym_3, anti_0) = NRE$, (iv) $NOP_1(sym_0, anti_2) = N'RE$.

The proofs of all these results are based on the simulation of register machines (called counter automata, in [6]) by means of P systems with symport/antiport.

The equality $NOP_2(sym_3, anti_0) = NRE$ has been also obtained in [14], this time with the proof carried out in the "usual" way of universality proofs in membrane computing: starting from a matrix grammar with appearance checking in the Z-binary normal form.

As one can see, the results from Theorem 2 improve all results from [17] about P systems with symport/antiport rules – but still one does not know whether these new results are optimal. Thus, question Q8 from [17] is still open, with respect to the new combinations of the number of membranes, size of symport rules, and size of antiport rules as specified by Theorem 2.

# 4 Errata-Remark About the Z-Normal Form

We have mentioned that the proof from [14] is based on the so-called Z-normal form for matrix grammars with appearance checking. This normal form was given in [10] and it is also used in many proofs from [17].

However, the formulation of this normal form has a flaw, pointed out to us by R. Freund during WMC2002. Let us first recall this formulation as it appears in [17] (page 35):

A matrix grammar with appearance checking $G = (N, T, S, M, F)$ is said to be in the *Z-binary normal form* if $N = N_1 \cup N_2 \cup \{S, Z, \#\}$, with these three sets mutually disjoint, and the matrices in $M$ are in one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*, |x| \leq 2$,
3. $(X \to Y, A \to \#)$, with $X \in N_1, Y \in N_1 \cup \{Z\}, A \in N_2$,
4. $(Z \to \lambda)$.

Moreover, there is only one matrix of type 1, $F$ consists exactly of all rules $A \to \#$ appearing in matrices of type 3, and if a sentential form generated by $G$ contains the object $Z$, then it is of the form $Zw$, for some $w \in T^*$ (that is, the appearance of $Z$ makes sure that, except for $Z$, all objects are terminal); # is a trap-object, and the (unique) matrix of type 4 is used only once, in the last step of a derivation.

For this formulation, the assertion that for each recursively enumerable language $L$ there is a matrix grammar with appearance checking in the Z-normal form which generates $L$ is not true. In fact, the proof from [10] gives a slightly different normal form: the symbol $Z$ appears only in strings of the form $Zw$, where *$w$ is either a terminal string, or it contains occurrences of the trap-symbol #*. Therefore, the appearance of $Z$ makes sure that, except for $Z$, the sentential form is either terminal, or it will never lead to a terminal string, because of the presence of the trap-symbol.

Fortunately, all proofs from [17] (and the papers cited in [17]) which are based on the Z-normal form remain valid: in all these proofs one constructs a P system which simulates the initial matrix grammar in the Z-normal form; all these systems contain the object # and rules of the form $\# \to \#$ $((\#, in), (\#, out)$, in the symport case, etc). This means that in the case when $Z$ appears together with a terminal string, the computation in the associated P system halts, while if $Z$ appears together with a string which contains occurrences of #, then the computation will never halt. In short, $Z$ signals correctly the end of successful computations.

# 5 Back to Computing by Communication

The results from Theorem 2 have some direct implications about the results concerning the recently considered mode of defining the output of a computation, by recording the traces of certain objects through the membranes of a P system, [8]. We refer to [6] for details, and we pass to mentioning one new class of P systems with the computation based on communication only. It has been introduced in [22] under the name of *communicating P systems*, and uses rules of the following three forms:

$$a \to a_t,$$
$$ab \to a_{t_1} b_{t_2},$$
$$ab \to a_{t_1} b_{t_2} c_{come},$$

where $a, b, c$ are objects, $t, t_1, t_2$ are target indications of the forms *here, out*, $in_j$ ($j$ is the label of a membrane). These rules are associated with membranes, with the restriction that rules of the third type can only be associated with the skin membrane (by means of rules of this form we can bring new objects into the system, from the environment).

Note that, for instance, rules of the form $ab \to a_{in}b_{out}$ do not correspond to symport or antiport rules: the objects $a, b$ are from the same region, but after the use of the rule, they go into different regions, adjacent to the starting region.

In [22] it is proved that systems with rules as above are computationally universal, but the proof does not provide a bound on the number of membranes. The result has been improved in [14] to the following form ($NOP_m(com)$ is the family of sets of numbers computed by P systems with rules as above, using at most $m$ membranes):

**Theorem 3.** $NOP_2(com) = NRE$.

The proof from [22] starts from register machines, the one from [14] starts from matrix grammars with appearance checking in the Z-binary normal form. By combining the two techniques, it seems that the result can be improved to systems with only one membrane – this is a remark of R. Freund, made during WMC2002.

A further variant of P systems using rules which are either communication rules or transformation rules was proposed in [2]. For instance, the communication rules are of the form $xx'[_i y'y \to xy'[_i x'y$, with the meaning that the objects specified by $x'$ enter membrane $i$ and the objects specified by $y'$ exits, at the same time, membrane $i$; the multisets $x$ and $y$ should be present outside and inside membrane $i$, respectively. Note that such rules also involve the membranes of the systems, and that they are rather general, they can encode both symport and antiport rules. Consequently, no further rules are necessary in order to achieve the computational universality, in particular, no object transforming rules (providing that the environment is supposed to contain sufficient copies of the needed objects.

# 6 P Automata

The problem of considering classes of P systems which do not generate/compute sets of numbers, but *accept* sets of numbers, in the sense of accepting grammars or of automata, was formulated several times in the membrane computing literature. A first answer was provided recently in [5], in the form of a class of P systems – called P automata – which work by communication only, "absorbing" multisets of objects from the environment and pushing them down in the membranes of the system. That is, the communication is performed in a one-way manner, top-down. A computation is controlled by the contents of the regions of the membrane structure, and it is successful when it reaches a configuration where each region contains a multiset which is considered *terminal*; the result of a successful computation consists of a string associated with the sequence of multisets which enter the system during the computation.

The form of the rules used in [5] is $x : y \to in$, having the same meaning as the conditional symport rule $(y, in)_x$: if the multiset of objects $x$ is present in region with which the rule is associated, then the objects specified by $y$ enter this region, coming from the immediately upper region (the environment, in the case of the skin membrane).

Note that the communication is done in a one-way manner, which is rather restrictive, but the rules are applied in a permitting context mode, depending on the contents

of the region where we are going to bring objects from the upper region. The multisets of objects present in each region are called *states*; final states are provided in advance, defining the end of a computation (a computation ends when all regions contain a final state). This way of defining the successful computations is new (it also correspond to the definition of successful computations by checking the presence of certain objects in certain "acknowledging" membranes, as in [7]).

Somewhat surprising, the devices defined in [5] are shown to be computationally universal: P automata with seven membranes characterize the recursively enumerable languages, modulo a certain way to "translate" a sequence of multisets into a string (the corresponding mapping is called an l-projection in [5]). The proof is again based on the characterization of recursively enumerable languages by means of counter automata.

Of course, seven looks "too large", so a natural *open problem* is to improve the result from [5]. Several related research topics can be formulated (some of them are already stated in [5]): consider two-way communication, consider rules of a restricted form (what about non-conditional rules, maybe using two-way communication), consider usual symport/antiport rules, etc. We forecast significant continuations of this idea of P automata (accepting P systems), due to its interest from the point of view of parsing, decidability, etc.

# 7 Other Recent Developments

We now briefly mention some other recent interesting ideas and (preliminary) results, all of them suggesting appealing directions of research. We consider these subjects in the ordering of the corresponding papers in the bibliography which ends this note.

The paper [1] proposes some applications of P systems in linguistics, specifically, to pragmatics, phonetic evolution, and dialogue. To this aim, the so-called *linguistic P systems* are introduced, where a series of new operations with membranes (and their contents) are considered, such as absorption, invasion, and cloning, while the rules can change the region, can be active or sleeping, etc. The model proposed in [1] remains to be investigated from a theoretical point of view, and further considered from the point of view of application in linguistics and related areas.

Another unexpected link with linguistics is proposed in [3]: the so-called structured contextual grammars, introduced in [11], generate strings which also contain parentheses associated with the rules used during the derivation; a string of parentheses contained in a unique external parenthesis describes a membrane structure; the strings introduced by each contextual rule can be naturally considered as symport rules associated with the membranes. In this way (with some technical details which we omit here), a structured contextual grammar can be used for generating an infinite family of P systems. These systems are "syntactically similar", they grow in a way corresponding to the derivation in the "pattern grammar", and can be considered as snapshots of a computing device which evolves during solving a given problem (derivation steps in the contextual grammar can be intercalated with transitions in the generated P systems).

Up to now, the P systems were mainly considered as computing models, where 'computing' is understood in the Turing sense, as an algorithmic connection between

an input and an output (in particular, the input can be a decision problem, and the output can be the answer, yes or no, to the problem). Relationships between membrane computing and distributed computing, e.g., in computer networks, were investigated in [21]. A recent approach to this question, of the possible usefulness of membrane systems as models of distributed computing, can be found in [4]. The purpose is to prove that "P systems can become a primary model for distributed computing, particularly for message-passing algorithms". Algorithms are presented for several communication tasks, such as skin membrane broadcast, generalized broadcast, convergecast, leader election, fault toterant consensus, etc. Message complexity and time complexity are evaluated for the membrane algorithms corresponding to these problems.

Two recent papers, [9] and [13], answer an important open problem of membrane computing area: to consider probabilistic systems, able to take into consideration the probabilistic nature of biochemical reactions. Actually, in [13] two classes of P systems are introduced: stochastic systems, and randomized systems (the latter being used to implement randomized algorithms).

Especially interesting are the second type of systems, because they allow to implement algorithms which solve computationally hard problems, with a low error probability, in polynomial time, making use of a subexponential number of membranes. (This should be compared with the P algorithms based on membrane division, which solve hard problems in polynomial time by using an exponential number of membranes.) The paper presents a family of randomized P systems which are used for implementing the Miller-Rabin randomized algorithms for primality testing.

A reduction of the $\mathbf{P} \neq \mathbf{NP}$ conjecture to the existence of an **NP**-complete problem which cannot be solved in polynomial time by a deterministic "decision P system" constructed in polynomial time is discussed in [20] (roughly speaking, a *decision P system* is a system which accepts/rejects an input multiset if it stops by sending out the special object YES/NO). We do not enter into details, and refer the reader to [20].

# References


[1] G. Bel Enguix, Preliminaries about some possible applications of P systems in linguistics, in [19], 81–96.

[2] F. Bernardini, V. Manca, P systems with boundary rules, in [19], 97–102.

[3] R. Ceterchi, C. Martin-Vide, Generating P systems with contextual grammars, in [19], 119–144.

[4] G. Ciobanu, R. Desai, A. Kumar, Membrane systems and distributed computing, in [19], 145–162.

[5] E. Csuhaj-Varju, G. Vaszil, P automata, in [19], 177–192.

[6] P. Frisco, H.J. Hoogeboom, Simulating counter automata by P systems with symport/antiport, in [19], 237–248.

[7] P. Frisco, S. Ji, Info-energy P systems, *Proc. Eight Intern. Meeting on DNA Based Computers*, Sapporo 2002 (M. Hagiya, A. Obuchi, eds.), 161–170.

[8] M. Ionescu, C. Martin-Vide, Gh. Păun, P systems with symport/antiport rules: The traces of objects, in [19], 283–296.

[9] M. Madhu, Probabilistic P systems, submitted, 2002.

[10] C. Martin-Vide, A. Păun, Gh. Păun, G. Rozenberg, Mebrane systems with coupled transport: Universality and normal forms, *Fundamenta Informaticae*, 49, 1-3 (2002), 1–15.

[11] C. Martin-Vide, Gh. Păun, Structured contextual grammars, *Grammars*, 1, 1 (1998), 33–35.

[12] C. Martin-Vide, Gh. Păun, Language generating by means of membrane systems, *Bulletin of the EATCS*, 75 (Oct. 2001), 199–218.

[13] A. Obtulowicz, Probabilistic P systems, in [19], 331–332 (extended abstract; the complete paper was circulated during the workshop).

[14] A. Păun, Membrane systems with symport/antiport: universality results, in [19], 333–344.

[15] Gh. Păun, Computing with membranes; An introduction, *Bulletin of the EATCS*, 68 (Febr. 1999), 139–152.

[16] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and *Turku Center for Computer Science-TUCS Report* No 208, 1998 (www.tucs.fi).

[17] Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

[18] Gh. Păun, G. Rozenberg, A guide to membrane computing, *Theoretical Computer Sci.*, 2002.

[19] Gh. Păun, C. Zandron, eds., *Pre-proceedings of Workshop on Membrane Computing*, Curtea de Argeş, August 19-23, 2002 (Publication No. 1 of MolCoNet Project – IST-2001-32008).

[20] M.J. Perez-Jimenez, A. Romero-Jimenez, F. Sancho-Caparrini, Decision P systems and the $\mathbf{P} \neq \mathbf{NP}$ conjecture, in [19], 345–354.

[21] I. Petre, L. Petre, Mobile ambients and P systems, *J. Univ. Computer Science*, 5, 9 (1999), 588–598.

[22] P. Sosik, P systems versus register machines: two universality proofs, in [19], 371–382.