The "Catalytic Borderline" Between Universality and Non-Universality of P Systems

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania, and

Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
gpaun@us.es, ghpaun@gmail.com

Summary. *P systems* are computing models inspired by the structure and the functioning of the living cells; they are the basic computing devices of *membrane computing*, a branch of *natural computing*. The present note is an overview of results and open problems related to the borderline between the computationally universal and non-universal catalytic P systems. A short introduction to membrane computing is provided, to the use of the reader not familiar with this research area.

1 A Glimpse to Membrane Computing

Along its evolution, computer science has continuously looked to biology in order to find ideas (data structures, operations with them, ways to control these operations, architectures for computing devices, etc.) useful for improving the use of the existing electronic computers and for developing new computing tools. In the last decades, this tendency became a well defined branch of computer science, called *natural computing*. Besides the goal sketched above, a complementary one is to understand and investigate the processes taking place in nature – especially in biology – as computations.

Membrane computing is one of the research areas of natural computing, having as the starting point the living cell, considered alone or as a part of more complex structures, such as tissues and organs, including the brain. This direction of research was initiated in 1998 ([12]) and it developed rapidly: already in 2003, the Thompson Institute for Scientific Research, ISI, mentioned membrane computing as a fast emerging research front in computer science, with [12] considered a "fast breaking paper" (see http://esi-topics.com). The literature of the domain is now very large, including monographs, collective volumes, PhD theses, research projects. An introduction to membrane computing can be found in [14],

with a comprehensive presentation (at the level of year 2009) in [15]. Up-to-date information (including information about the two yearly meetings in this area, the February Brainstorming Week on Membrane Computing and the summer Conference on Membrane Computing) can be found at the domain website [16].

Very generally speaking, membrane computing deals with processing *objects*, by means of bio-inspired *operations*, in the compartments of a cell-like or tissue-like arrangement of *membranes*. Basically, (i) the objects are symbols from a given alphabet (but they can also be strings or can have a more complex structure), (ii) like in biochemistry, the objects are present in the *multiset* sense (each object has a precise multiplicity in a given compartment), (iii) the operations are abstractions of biochemical *reactions* or other types of operations inspired from the biology of the cell (e.g., symport and antiport, for passing objects across membranes, or operations with membranes – division, separation, phagocytosis, and so on); (iv) the arrangement of membranes is either hierarchical, like in a cell, or "horizontal", like in tissues and other populations of cells (e.g., of bacteria). The operations (reactions) are also called *evolution rules*, or, shortly, *rules*. Like in biochemistry, in the basic model, the operations take place in a *non-deterministic* way (the rules to apply and the objects to react are chosen non-deterministically), in parallel (simultaneously in all compartments, with the objects in each compartment evolving in parallel, according to the local rules). Many variants were investigated, with respect to the types of rules, the ways to use them, the arrangement of membranes. Using the rules, we can pass from a configuration of the system to the next configuration – and in this way we can define *computations*. What is obtained, called a *P system*, was not initially meant as a model of the biological cell, to the use of biologists, but a computing model, of interest for computability.

There are two basic theoretical issues to be addressed for any new computing model, including the P systems: the computing power (competence), and the computing efficiency (performance). Accordingly, two are the reference frameworks: the Turing machines and their restrictions in what concerns the power, and the complexity classes (in particular, the theoretical borderline between tractability and intractability, between polynomial complexity and exponential complexity) in what concerns the efficiency.

From both these two points of view, membrane computing proved to be successful: many classes of P systems are equivalent in power with Turing machines (hence, according to Turing-Church thesis, they can compute whatever an algorithm can compute; we also call this property *computational completeness* or *Turing universality*), while many classes of P systems (especially those equipped with the possibility of producing an exponential working space in polynomial time, e.g., by means of membrane division or string replication) can solve computationally hard problems (typically, **NP**-complete problems, but also harder problems) in a feasible time (typically, polynomial, but often even linear).

A basic question in both these research directions (power and efficiency) is to find the borderline between universality and non-universality, in what concerns the power, and between efficiency and non-efficiency. In this paper we recall some

results about the first issue, namely, considering borderline results concerning the universality of *catalytic P systems* – definitions will be given in the next section.

In parallel with the theoretical investigations, mainly dealing with the previously mentioned questions, power and efficiency, P systems proved to be useful tools for several *applications*, starting with the very field where they originated – biology and biomedicine. This is now one of the main trends of research in this area. For the biologist, membrane computing has several attractive features: the models are directly inspired from biology, they are easy to be understood, P systems deal with discrete mathematical structures (as encountered in many situations in biology, where traditional differential equations are not appropriate), they are easily scalable and programmable, and have an emergent behavior (the evolution cannot be predicted by examining the initial configuration and the evolution rules). For other areas of application (computer graphics, approximate optimization, robot control, etc.) the inherent parallelism, hence computational efficiency, is the central attractive feature. We here do not give details about applications; the reader is referred to the Handbook [15] and to the website mentioned above.

2 Catalytic P Systems

We now introduce the model we consider in this paper, the cell-like P systems, in the catalytic form, stressing once again that, from the biological reality, we abstract a mathematical model suitable for computability investigations, thus ignoring many biological details.

The basic ingredients of a (cell-like) P system are the following ones:

1. The *membrane structure* is a hierarchical arrangement of membranes, understood as 3D vesicles; a membrane without any other membrane inside is said to be *elementary*; each membrane defines a *region/compartment*, the space between the membrane and the immediately inner membranes, if any; the space outside the "skin" membrane is called the *environment*. Each membrane can be labeled, and the label will identify both the membrane and its region. The membrane structure can be represented by a rooted tree (with a membrane in each node and the skin in the root), hence also by an expression of correctly nested labeled parentheses. Sometimes we also use Euler-Venn diagrams (disjoint sets included in a unique external set, the skin one).
2. The *objects* are placed in the compartments of the membrane structure, in the form of multisets (sets with multiplicities associated with the elements). In membrane computing, the multisets are usually represented by strings, like in formal language theory, with the multiplicity of a symbol corresponding to the number of occurrences of that symbol in the string; thus, a string and all its permutations represent the same multiset. For instance, a^2bc^4ab represents the multiset which contains 3 copies of a, 2 copies of b, and 4 copies of c.
3. The *evolution rules* are multiset rewriting rules similar to reactions in chemistry/biochemistry. The basic form is $u \to v$, where u and v are multisets

of objects from a given set O. The use of such a rule means "consuming" the objects of u and "producing" the objects of v. (Note that we do not pay attention to conservation lows, we work with arbitrary multisets, the only restriction is that u is not empty.) In order to link the regions of a system, the objects produced by a rule $u \to v$ can have associated *target indications*, of the forms *here, out, in*, with the meaning that an object with the target *here* remains in the same region where the rule is applied, an object with the target *out* is sent out of the respective membrane (in this way, objects can also be sent to the environment, when the rule is applied in the skin region), while an object with the target *in* is sent to one of the immediately inner membranes, non-deterministically chosen (if there is no such membrane, i.e., if the rule is associated with an elementary membrane, then the rule $u \to v$ with v containing an object (a, in) cannot be applied). The indication *here* in general is omitted when writing the rules.

Both the objects and the rules are associated with compartments of the system; the rules in a given region ("reactor") can be applied only to the objects from the same region.

The way of using the rules which we consider here is the *non-deterministic maximally parallel* one: the rules to be applied are chosen non-deterministically, but in such a way that the choice is maximal, i.e., we apply a multiset of rules (each rule can be applied several times) which is maximal, no further rule can be added to it so that the obtained multiset is still applicable to the existing objects.

The membranes and the objects present in the compartments of a system at a given time form a *configuration*; starting from a given *initial configuration* and using the rules as explained above, we get *transitions* among configurations; a sequence of transitions forms a *computation*. A computation is *halting* if it reaches a configuration where no rule can be applied. With a halting computation we associate a *result*, in the form of the number of objects present in a specified elementary membrane in the halting configuration.

Thus, a (cell-like) P system can be formalized as a construct

$$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o)$$

where O is the alphabet of objects, μ is the membrane structure (with m membranes), $w_1, \ldots, w_m$ are multisets of objects present in the m regions of μ at the beginning of a computation, $R_1, \ldots, R_m$ are finite sets of evolution rules, associated with the regions of μ, and i_o is the label of an elementary membrane, used as the output membrane.

There are many variations of this basic model. For instance, if a rule $u \to v$ has at least two objects in u, then it is called *cooperative*; if u is a single object, then the rule is *non-cooperative*; an intermediate case is that of *catalytic* P systems, whose rules are of the form $ca \to cv$, where c is a special object which never evolves and never passes through a membrane (both these restrictions can be relaxed), but it just assists object a to become the multiset v. A catalytic P system is written in

the form $\Pi = (O, C, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o)$, where all components are as above and $C \subset O$ is the set of catalysts.

We end this section with a simple example, illustrating the architecture and the functioning of a (cell-like) P system, as well as the usual way of graphically representing a P system. Figure 1 indicates the initial configuration (including the rules) of a system which computes the function $n \longrightarrow n^2$, for any natural number $n \geq 1$: we introduce the number n in the initial configuration, in the form of n copies of the object a present in the skin region, and we get the result as the number of copies of object f present in membrane 2 when the computation halts. Besides catalytic and non-cooperating rules, the system also contains a rule with promoters, $e \to e(f, in)|_b$: the object e evolves to ef only if at least one copy of object b is present in the same region.

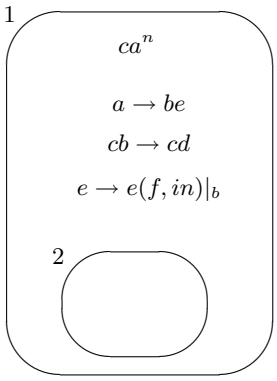

Fig. 1. A P system with catalysts and promoters

Formally, the system is given as follows:

$$
\begin{aligned}
\Pi &= (O, C, \mu, w_1, w_2, R_1, R_2, i_o) \text{ where} \\
O &= \{a, b, d, e, c, f\} \text{ (the set of objects)}, \\
C &= \{c\} \text{ (the set of catalysts)}, \\
\mu &= [\ [\ \ ]_2\ ]_1 \text{ (membrane structure)}, \\
w_1 &= ca^n \text{ (initial objects in region 1)}, \\
w_2 &= \emptyset \text{ (initial objects in region 2)}, \\
R_1 &= \{a \to be,\ cb \to cd,\ e \to e(f, in)|_b\} \text{ (rules in region 1)}, \\
R_2 &= \emptyset \text{ (rules in region 2)}, \\
i_o &= 2 \text{ (the output region)}.
\end{aligned}
$$

The system starts working by using the rule $a \to be$, which has to be applied in parallel to all copies of a; hence, in one step, all objects a are replaced by n

copies of b and n copies of e. From now on, the other two rules from region 1 can be used. The catalytic rule $cb \to cd$ can be used only once in each step, because the catalyst is present in only one copy. This means that in each step one copy of b is replaced by d. Simultaneously (because of the maximal parallelism), the rule $e \to e(f, in)|_b$ should be applied as many times as possible and this means n times, because we have n copies of e. Note the important difference between the promoter b, which allows using the rule $e \to e(f, in)|_b$, and the catalyst c: the catalyst is involved in the rule, it is counted when applying the rule, while the promoter makes possible the use of the rule, but it is not counted; the same (copy of one) object can promote any number of rules. Moreover, the promoter can evolve at the same time by means of another rule (the catalyst is never changed).

In this way, in each step we change one b to d and we produce n copies of f (one for each copy of e); the copies of f are sent to membrane 2 (the indication in from the rule $e \to e(f, in)|_b$). The computation should continue as long as there are applicable rules. This means exactly n steps: in n steps, the rule $cb \to cd$ will exhaust the objects b and in this way neither this rule can be applied, nor $e \to e(f, in)|_b$, because its promoter does no longer exist. The computation halts and in membrane 2, considered as the output membrane, we get n^2 copies of object f.

3 The Power of Catalytic P Systems

We start now to recall results about the computing power of catalytic P systems.

Let us denote by $NP_m(cat_r)$ the family of sets of numbers computed (generated, in the above sense) by P systems with at most m membranes, using catalytic or non-cooperative rules, containing at most r catalysts. We also denote by NRE the family of Turing computable sets of natural numbers ("recursively enumerable", hence the abbreviation), and by $NREG$ the family of semilinear sets of numbers (recognized by finite automata). When all the rules of a system are catalytic, we say that the system is *purely catalytic*, and the corresponding families of sets of numbers are denoted by $NP_m(pcat_r)$. When the number of membranes is not bounded by a specified m (it can be arbitrarily large), then the subscript m is replaced with $*$.

The following fundamental results are known:

Theorem 1. (i) $NP_2(cat_2) = NRE$, [5];
(ii) $NREG = NP_*(pcat_1) \subseteq NP_*(pcat_2) \subseteq NP_2(pcat_3) = NRE$, [7], [8].

Two intriguing open problems appear here, related to the borderline between universality and non-universality: (1) are catalytic P systems with only one catalyst universal? (2) are purely catalytic P systems with two catalysts universal?

We here consider only the first question. In the membrane computing community, the belief is that the answer is negative, one catalyst is not enough in order to equal the power of Turing machines. Preliminary results, supporting this conjecture, can be found, e.g., in [3].

On the other hand, in the membrane computing literature there are many results which show that P systems with only one catalyst are universal if further ingredients are added. Many results of this type can be found in [4], while recent developments can be found in [6] and [10]. We now recall several of these results, without always giving the place where they were reported first; such information can be found in the bibliography of [4]. The overall impression is that "one catalyst is almost universal": features which look "innocent" at the first sight are enough to lead P systems with one catalyst to universality.

4 Universality for P Systems with One Catalyst

Inspired from biochemistry and/or from computability theory, we may add various ingredients to P systems as defined above.

For instance, we may assume that some rules are "more active" than other rules, hence they have *priority* in being applied. This corresponds to considering a partial order relation among the rules in each compartment of a P system. It was proved already in [12] that $NP_2(cat_1, pri) = NRE$, where *pri* indicates the use of priorities.

In the example considered in Section 2 we have also used another feature, the *promoters*: rules can have associated objects which act as promoters, the rule can be applied only if at least one copy of each of the associated promoters is present. The promoters can evolve at the same time, but by other rules than those they promote. Similarly, rules can have associated *inhibitors*, objects whose presence forbids the application of the rule. Catalytic P systems with only one catalyst, using either promoters or inhibitors (one object only associated with a rule, not larger multisets), are universal.

Slightly more sophisticated is the control of the evolution of a P system by means of controlling the *membrane permeability*, [13]. This is achieved by using two operations, associated with usual multiset processing rules: decreasing the thickness (hence increasing the permeability) and increasing the thickness (hence decreasing the permeability) of membranes. Specifically, rules of the forms $u \to v\delta$ and $u \to v\tau$ are used. Initially, each membrane is supposed to be of thickness one. A membrane of thickness one behaves as a usual membrane, objects can be moved across it by means of target indications *in* and *out*. A membrane of thickness 0 is *dissolved*, its objects are left free into the surrounding region and its rules are "lost" (specific biochemistry is active in each membrane, hence, by dissolving a membrane, the respective "reactor" disappears). If a membrane has thickness 2, then it is impermeable, no object can pass across it (hence the rules which ask for such a passage cannot be used). The symbols δ, τ indicate the decrease and the increase, respectively, of the thickness by one. The thickness cannot be greater than 2, a rule $u \to v\tau$ used in such a membrane will lead to a membrane with the same thickness. As already expected, the use of the operations δ, τ (the control

of membrane permeability) again leads to the universality of P systems with one catalyst.

The previous idea actually is part of a larger research area in membrane computing, dealing with the possibility to also evolve the membrane structure during a computation. There are many biologically inspired operations of this type. Here we mention only one, the *membrane creation*, [11]. Besides usual non-cooperating and catalytic rules, we also use rules of the form $ca \to c[\ v\ ]_i$, with the meaning that object a, with the help of catalyst c, produces a new membrane, with the label i, having inside the multiset v; of course, the catalyst is reproduced. Also this feature leads to universality in the case of P systems with only one catalyst.

If instead of "standard" catalysts we use catalysts with additional features, then again we obtain universality. Two such extensions were considered: *bistable* catalysts and *mobile* catalysts, [9]. In the first case, the catalyst can oscillate between two *states* (and this is the only possible transformation the catalysts can have), in the latter case the catalyst can pass through membranes like any other object, by means of target indications *in* and *out*.

Several similar results were recently obtained in [6] and [10].

One of them is the *target restriction*. This restriction has two components, one at the syntactic level (in each rule $u \to v$, all target indications which appear in v are identical), and one at the semantic level (in each step of a computation, in each membrane one uses rules with the same target indication in their right hand member; in different membranes, different target indications may be used, while the choice of rules to apply is done as in general P systems, in the non-deterministic maximally parallel way). Interesting enough, the universality of target restricted one catalyst P systems is obtained in [6] by means of P systems with 7 membranes (it is an open problem whether or not the number of membranes can be diminished).

Another restriction considered in [6] is the *time-varying*: a sequence $U_1, \ldots, U_p$ of sets of rules is given, the computation starts with a step when rules from the set U_1 are used, then we use rules from U_2, and so on; after step p, when rules from U_p are used, we return to U_1 and continue (in step $pn + i, n \geq 0$, one uses rules from set U_i). The universality of time-varying P systems is obtained for one catalyst P systems with only one membrane, having the *period* p equal to 6.

In [10], so-called *label restricted* P systems are considered: each rule has a label, which can be a symbol from a given alphabet, or it can be the empty label; a computation is label restricted if in each transition one applies only rules with the same label, possibly also rules with the empty label. Although this restriction does not look too strong, it is sufficient to get universality of P systems with only one catalyst.

5 Final Remarks

This paper is only a hint to one of the research directions in membrane computing. After a brief introduction to membrane computing, we have recalled several cases where P systems with only one catalyst are computationally equivalent with Turing machines. Various ingredients were considered: a priority relation among rules, promoters, inhibitors, the control of membrane permeability, mobile catalysts, bistable catalysts, membrane creation, label restricted transitions, selection of rules by the target indications, time varying sets of rules. Such results are of interest in view of the conjecture that P systems with only one catalyst are not universal (two catalysts are known to lead to universality).

Several problems remain open. For instance, because the conjecture is that purely catalytic P systems with two catalysts are not universal, all the results mentioned above for the one catalyst case should also be examined for the purely catalytic P systems with two catalysts.

Furthermore, other additional features remain to be considered, with the aim of increasing the power of P systems with one catalyst – for instance, inspired from the regulated rewriting area [2] or the grammar systems area [1] in formal language theory.

Acknowledgements. Work supported by Proyecto de Excelencia con Investigador de Reconocida Valía, de la Junta de Andalucía, grant P08 – TIC 04200. A careful reading of the first version of the text by Rudi Freund is gratefully acknowledged.

References

1. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun: *Grammar Systems. A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach, London, 1994.
2. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory.* Springer-Verlag, Berlin, 1989.
3. R. Freund: Particular results for variants of P systems with one catalyst in one membrane. *Proc. Fourth Brainstorming Week on Membrane Computing*, Fénix Editora, Sevilla, 2006, vol. II, 41–50.
4. R. Freund, O.H. Ibarra, A. Păun, P. Sosík, H.-C. Yen: Catalytic P systems. Chapter 4 of [15].
5. R. Freund, L. Kari, M. Oswald, P. Sosík: Computationally universal P systems without priorities: two catalysts are sufficient. *Th. Computer Sci.*, 330 (2005), 251–266.
6. R. Freund, Gh. Păun: Universal P Systems: One Catalyst Can Be Sufficient. *Proc. 11th Brainstorming Week on Membrane Computing*, Sevilla, 4-8 February 2013, Fénix Editora, Sevilla, 2013, to appear.
7. O.H. Ibarra, Z. Dang, O. Egecioglu: Catalytic P systems, semilinear sets, and vector addition systems. *Th. Computer Sci.*, 312 (2004), 379–399.
8. O.H. Ibarra, Z. Dang, O. Egecioglu, G. Saxena: Characterizations of catalytic membrane computing systems. *28th Intern. Symp. Math. Found. Computer Sci.*, 2003 (B. Rovan, P. Vojtás, eds.), LNCS 2747, Springer, 2003, 480–489.

9. S.N. Krishna, A. Păun: Results on catalytic and evolution-communication P systems. *New Generation Computing*, 22 (2004), 377–394.
10. K. Krithivasan, Gh. Păun, A. Ramanujan: On controlled P systems. *Fundamenta Informaticae*, to appear.
11. M. Mutyam, K. Krithivasan: P systems with membrane creation: Universality and efficiency. *Proc. MCU 2001* (M. Margenstern, Y. Rogozhin, eds.), LNCS 2055, Springer, Berlin, 2001, 276–287.
12. Gh. Păun: Computing with membranes. *J. Comput. Syst. Sci.*, 61 (2000), 108–143 (see also TUCS Report 208, November 1998, `www.tucs.fi`).
13. Gh. Păun: Computing with membranes – A variant. *Intern. J. Found. Computer Sci.*, 11, 1 (2000), 167–182.
14. Gh. Păun: *Membrane Computing. An Introduction*. Springer, Berlin, 2002.
15. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
16. The P Systems Website: `http://ppage.psystems.eu`.