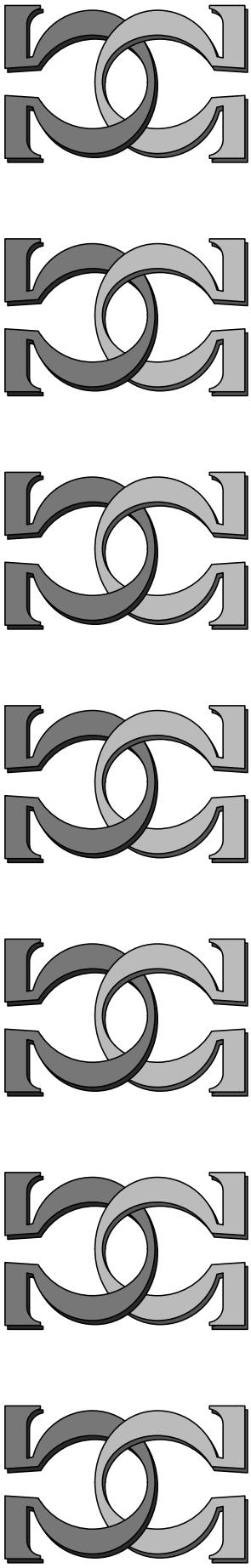

CDMTCS Research Report Series

Computing with Cells and Atoms: After Five Years

C. S. Calude[1] and G. Păun[2,3]

[1]University of Auckland, New Zealand
[2]Institute of Mathematics of the Romanian Academy, Romania
[3]Sevilla University, Spain

CDMTCS-246
August 2004

Centre for Discrete Mathematics and
Theoretical Computer Science

Computing with Cells and Atoms: After Five Years

Cristian S. Calude
Department of Computer Science
The University of Auckland
Private Bag 92019, Auckland, New Zealand
Email: cristian@cs.auckland.ac.nz

Gheorghe Păun
Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
george.paun@imar.ro

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Sevilla University
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
gpaun@us.es

Abstract

This is the text added to the Russian edition of our book *Computing with Cells and Atoms* (Taylor & Francis Publishers, London, 2001) to be published by Pushchino Publishing House. The translation was done by Professor Victor Vladimirovich Ivanov and Professor Robert Polozov.

This new chapter was written for the Russian edition. The whole field of unconventional models of computation has grown enormously in the past five years and there is no way to be able to give here even a glimpse of this evolution. To give just one example, the recent booklet titled "Biologically Motivated Computing and Bioinformatics" lists on 35 pages only books and journals published by Springer-Verlag (see

springeronline.com) in this area. It is far from being complete; two examples of journals, Kluwer *Journal of Natural Computing* and the new stream *Natural Computing* of Elsevier journal *Theoretical Computer Science*. Probably the best way to keep in touch with all recent news is to watch a couple of specialized websites. For DNA computing http://www.iturls.com/English/TechHotspot/TH_DNA.asp is a good start. Just days before finalizing this chapter we learnt that quantum computing is a significant step closer to becoming a reality today with new research published in the journal *Nature* (22 July 2004, 430, 431–435). Three groups of scientists from IBM, UCLA and the Netherlands, have reported the detection of a single electron spin. What does this mean for quantum computing? According to Professor Eli Yablonovitch, the director of UCLA's Center for Nanoscience Innovation for Defense, co-author of the paper, "With 100 transistors, each containing one of these electrons, you could have the implicit information storage that corresponds to all of the hard disks made in the world this year, multiplied by the number of years the universe has been around". And, of course, there is no good reason to stop to 100 transistors...

So, instead of trying in vain to update the content of the whole book we decided to present some recent results in which the authors of the book have been actively involved, namely in *membrane computing* and *trespassing the Turing barrier*.

6.1 Membrane Computing

Membrane computing was initiated in the fall of 1998, when the technical report version of the paper [57][1] was circulated on web. Thus, the present book was written at about one year after that. In the meantime, the area has simply flourished. At the middle of July 2004, the web page http://psystems.disco.unimib.it, maintained in Milano, Italy, under the auspices of European Molecular Computing Consortium (EMCC), contained a bibliography of more than 450 papers, with more than one dozen of collective volumes and one dozen of PhD theses devoted to membrane computing. In 2002 a comprehensive monograph was published by Springer-Verlag, [58]. In February 2003, the Institute for Scientific Information (ISI) has mentioned [57] as a "fast breaking paper" in computer science (see http://esi-topics.com, February 2003); in October 2003, the domain itself was qualified by ISI as "emergent research front in computer science", with the paper [56] mentioned as a "citation leader" in membrane computing.

Consequently, "updating" the first edition of the present book with the current state of the art in membrane computing is simply impossible, as the needed space would be too large. Thus, what we want to do here is to leave the reader having an idea about the directions of research in this area, mentioning only a few relevant notions introduced and results obtained after the publication of the English edition of the present book. We

[1]The references point to the separate bibliography given at the end of this section.

also want to give some bibliographical guidelines to the reader interested in approaching membrane computing – starting with the web address mentioned above, where one can find both the current bibliography of the domain and a series of downloadable papers. In particular, in the page there are available all pre-proceedings volumes of the Workshops on Membrane Computing (the first three editions were organized in Curtea de Argeş, Romania, in 2000, 2001, 2002; the fourth edition was held in Tarragona, Spain, in 2003, the fifth one in Milano, Italy, in 2004, while the sixth one will be held in Vienna, Austria, in July 2005); one can also find in the page the volumes of the Brainstorming Weeks on Membrane Computing (Tarragona 2003, Sevilla 2004).

Some of the most important directions on which membrane computing was developed in the last five years are the following (of course, the list is not exhaustive):

1. improving existing results (especially concerning the size of systems able of universal computations);

2. introducing new classes of P systems, in general, in the attempt to capture further biological features, thus getting closer to the biological reality which has inspired the area; the most important class of this kind is that of symport/antiport systems, introduced in [56];

3. considering not only generative computing devices, but also recognizing devices (automata-like P systems);

4. passing from cell-like systems, with the membranes hierarchically arranged (in the nodes of a tree), to tissue-like systems, with the membranes placed in the nodes of an arbitrary graph;

5. very active investigations in the computational complexity direction, with two main topics of interest: the complexity of problems related to P systems (membership, reachability, etc), and solving computationally hard problems (typically, **NP**-complete, but also **PSPACE** problems) in a feasible time (polynomial, often even linear) time by means of P systems able to generate an exponential working space in a linear time by means of bio-inspired operations (such as cell division, membrane creation, or string replication);

6. implementations/simulations on the electronic computer, on clusters or on networks of computers, or on a dedicated hardware; such programs are both of a didactic interest (in particular, for checking the evolution of given systems) and for applications;

7. applications of membrane computing, in biology, computer graphics, linguistics, computer science, management; most of these applications are preliminary, under current research, but they support the belief that membrane computing can be a useful framework for approaching a large number of problems (in particular,

related to the modelling and the simulation of the living cell, one of the challenges for the bio-informatics – see, e.g., [43], [82]);

8. attempts to use non-crisp mathematics in the study of P systems, mainly probabilistic and fuzzy sets tools, but also rough sets tools;

9. attempts to have dynamic systems approaches, focusing not on the results of computations, but on the evolution itself of a system; related investigations are those concerning P systems with a dynamic structure of membranes, or, in the case of tissue-like P systems, with dynamic communication channels among cells;

10. "advanced" topics, such as reversibility, conservativity (of energy), possibilities to compute beyond the Turing barrier, etc.

In what follows, we briefly discuss each of these ten topics, at an informal level, mainly offering bibliographical hints about the progresses reported in the last five years. Details can be found in the monograph [58] (in its turn, covering only the developments in membrane computing until the spring of 2002) and, mainly, in the web page mentioned above. Recent surveys of the domain can be found, e.g., in [59], [49], [61], [60].

6.1.1 Technical Improvements

The classification of evolution rules in cooperative, catalytic (as a particular case of cooperative rules), and non-cooperative raises the natural question about the power of systems using rules of a specific type. Systems with general cooperative rules are easily seen to be computationally complete, while systems using non-cooperative rules can compute only semilinear sets of natural numbers (of vectors of natural numbers) – Theorem 3.3. By adding further ingredients, such as a priority relation on the sets of rules from each region (Theorem 3.2), or the possibility to control the permeability of membranes (by operations δ, τ – Theorem 3.11), we can obtain the universality also for catalytic systems. Whether catalytic systems without any additional controls on the rule application are universal or not was for a while an important open problem in membrane computing.

Surprisingly enough, the answer was positive: P systems with catalytic rules are universal. The first proof was given in [74], and it was using *eight catalysts*. The result was improved several times from the point of view of the number of catalysts; the best current result says that *two catalysts suffice*, see [33]. It is still not known whether one catalyst suffices (the answer is negative for purely catalytic systems [41], those using only rules of the form $ca \to cu$, where c is a catalyts; note than in general in catalytic systems one can also use non-cooperative rules, of the form $a \to u$).

These results do not make uniniteresting the universality results from Theorems 3.2 and 3.11, because in the proofs of these theorems one uses only one catalyst.

Actually, one can obtain the universality without using catalysts, at the expense of using other ingredients, such as promoters (objects which should be present) or inhibitors (objects which should not be present) for rules, or counting the number of copies of rules in the same way as we count the number of copies of objects (that is, we start with multisets of rules available in each region; after applying a rule, it is consumed, but other rules become available). We refer for details to [58].

Such universality results (actually, their proofs) have as a direct consequence the fact that the number of membranes necessary in order to reach the power of Turing machines is bounded (the bound is obtained by starting the proofs from a universal device – grammar or Turing machine; then, the obtained P system is universal, in the sense that it can simulate any given system after introducing in its initial configuration a "code" of the particular system; thus, the hierarchy on the number of membranes collapse at the level of the number of membranes of the universal P system). In this context, an interesting problem which has remained open for a while was that of finding a class of P systems for which the number of membranes induces an infinite hierarchy of the generated sets of numbers. Some answers to this problem were given in [32] and [44], based on classes of systems of a rather restricted forms; a fully satisfactory answer (based on a "neutral" and general enough class of systems) was provided in [40], using a class of P systems where the computation is done by communication only.

6.1.2 Computing by Communication

Communication (the transfer of objects across membranes) plays a central role both in the functioning of P systems and in the cell functioning. The use of target indications of the form *here, in, out* attached to objects and directly introducd by evolution rules reminds the fact that the cell membranes contain many protein channels which are very selective, but in biology, in general, the communication is done by "rules" (mechanisms) different from the evolution (reaction) rules. One of the most interesting ways to pass chemicals from a compartment of a cell to another compartment, or across the cell membrane, is by a *coupled* transport. The process of moving two (or more) objects in the same direction through a protein channel is called *symport*, while the process of moving objects in oposite directions is called *antiport*. Details can be found in [2].

Mathematically, we can consider object processing rules of the following forms: a symport rule (associated with a membrane i) is of the form (ab, in) or (ab, out), stating that the objects a and b enter/exit together membrane i, while an antiport rule is of the form $(a, out; b, in)$, stating that, simultaneously, a exits and b enters membrane i.

We can generalize this idea, to using rules of the forms (x, in), (x, out), $(x, out; y, in)$, where x, y are arbitrary multisets of objects (the size of these multisets is called the *weight* of the respective rules). Note that in the case when we use symport or antiport rules associated with the skin membrane we can send objects out of the system and we can bring into the system objects from the environment. It is also important to note that such systems do not create and do not destroy objects, the objects are only moved across membranes. As a consequence, in order to be able to compute arbitrary numbers, we need a supply of objects into the environment.

Thus, a P system with symport/antiport rules has the same architecture as a system with multiset rewriting rules: alphabet of objects, terminal objects, membrane structure, initial multisets in the regions of the membrane structure, sets of rules associated with the membranes, possibly an output membrane – with one additional component, the set of objects present in the environment. If an object is present in the environment at the beginning of a computation, then it is considered available in arbitrarily many copies (the environment is inexhaustible). We refer to [56] (where symport/antiport P systems were introduced) and to [58] for further details.

The functioning of a P system with symport/antiport rules is the same as for systems with multiset rewriting rules: the transition from a configuration to another configuration is done by applying the rules in a non-deterministic maximally parallel manner, to the objects available in the regions of the system and in the environment, as requested by the used rules. When a halting configuration is reached, we get a result, in a specified output membrane (the environment already contains objects, hence cannot be used for collecting the result). The set of all such numbers computed by Π is denoted by $N(\Pi)$. Of course, we can also consider as a result the vector $Ps(\Pi)$ of numbers representing the multiplicity of objects present in the halting configuration in the output membrane of the system; the family of all such sets of vectors $Ps(\Pi)$, computed by P systems with at most m membranes, with symport rules $(x, in), (x, out)$ with $|x| \leq r$, and antiport rules $(x, out; y, in)$ with $|x|, |y| \leq t$ is denoted by $PsOP_m(sym_r, anti_t)$, $m \geq 1$ and $r, t \geq 0$.

Because the class of symport/antiport P systems is central now in membrane computing,

we give here an **example** of such a system. Let

$$\begin{aligned}
\Pi &= (V, T, \mu, w_1, w_2, E, R_1, R_2, 2), \text{ with} \\
V &= \{a, b, c, c', d, e, e', f, g, Z\}, \\
T &= \{a\}, \\
\mu &= [_1[_2\,]_2]_1, \\
w_1 &= ac, \\
w_2 &= df, \\
E &= \{a, b, c', e, e', g, Z\}, \\
R_1 &= \{(c, out; Z, in),\ (ca, out; cbb, in),\ (ca, out; c'bb, in), \\
&\quad (da, out; Z, in),\ (c'd, out; e, in),\ (eb, out; ea, in), \\
&\quad (eb, out; e'a, in),\ (fb, out; Z, in),\ (e'f, out; cdf, in), \\
&\quad (e'f, out; g, in)\}, \\
R_2 &= \{(d, out; c', in),\ (c', out),\ (f, out; e', in),\ (e', out), \\
&\quad (df, in),\ (Z, in),\ (Z, out),\ (ga, in),\ (g, out)\}.
\end{aligned}$$

(E is the set of objects which appear in the environment in arbitrarily many copies.) Assume that at some moment we have in the skin region n copies of object a (initially, $n = 1$), and one copy of c, and in region 2 we have one copy of d and one copy of f. If we use the rule $(c, out; Z, in) \in R_1$, then Z gets into region 1 from the environment. In this case the computation will never stop, because of the rules $(Z, in), (Z, out)$ from R_2 which cause Z to oscillate between regions 1 and 2. Thus we have to begin by using the rule $(ca, out; cbb, in) \in R_1$ a number of times. When we replace all copies of a by b, then we have to use the rule $(c, out; Z, in) \in R_1$, and so the computation will not halt, hence we have to save one a and at some time to use the rule $(ca, out; c'bb, in) \in R_1$. The object c' must now go to region 2 and send from there the object d, by the rule $(d, out; c', in) \in R_2$. In the next step, if any copy of a is still present in region 1, then the rule $(da, out; Z, in) \in R_1$ must be used, and thus again the computation will never stop. If no copy of a is present in region 1, that is, the doubling of n was complete (the n copies of a were replaced by $2n$ copies of b), then d will wait one step in the skin region, as c' exits region 2, and then by rule $(c'd, out; e, in) \in R_1$ the two objects exit the system together, while e is brought in.

In the same way as c has changed all a to b, the object e will change now all copies of b into a. When this operation is completed (and only then, since otherwise Z will get into the system), e is replaced by e', and e' will send f out of region 2. The object f checks whether or not all copies of b were replaced by a, and only in the affirmative case it exits the system together with e' and reintroduces the objects c, d, and f; d and f enter then region 2, and the whole process can be iterated. In this way, we can double the number of copies of a as many times as we want, but at least once. At any moment, instead of the rule $(e'f, out; cdf, in) \in R_1$ we can use the rule $(e'f, out; g, in) \in R_1$. The

object g will carry each copy of a into membrane 2 and when this is completed g gets stuck in the skin region. Therefore, $N(\Pi) = \{2^n \mid n \geq 1\}$.

Again somewhat surprisingly, but pleasantly enough from a computability point of view, *computing by communication is universal*, P systems with symport/antiport rules can compute at the level of Turing machines. The currently best results in this respect, in what concerns the number of membranes and the size of symport and antiport rules are the following (proofs can be found in [37], [35], [84]).

Theorem 1 $PsCE = PsOP_m(sym_r, anti_t)$, *for* $(m, r, t) \in \{(1, 1, 2), (3, 2, 0), (2, 3, 0), (3, 1, 1)\}$.

The optimality of these results is not known. In particular, it is an *open problem* whether or not also the families $PsOP_m(sym_r, anti_t)$ with $(m, r, t) \in \{(2, 2, 0), (2, 2, 1)\}$ are equal to $PsCE$.

In the case of systems with symport/antiport rules, one can associate a string with a computation by considering the *trace* of a specified object – the "traveller" – through membranes (the sequence of labels of membranes visited by the traveller during a successful computation), hence a language is associated with a device working with numbers as the internal data structure.

The symport/antiport rules can be used also for defining a class of P systems where the evolution (done through multiset rewriting rules without target indications) is separated from the communication (which is done through symport/antiport rules) – details can be found in [19].

6.1.3 P Automata

The systems considered up to now are generative devices, similar to grammars: starting from an initial configuration (membranes and multisets of objects) we get sequences of transitions, hence computations; because of the non-determinism, we have branching possibilities, and that is why we can associate with a system a *set* of numbers or a set of strings, hence a *language*. In computability, dual to grammars we have automata, devices which recognize/analize strings. A similar strategy has been followed also in membrane computing, by introducing automata-like P systems ([28]). One considers a P system of a given type (membranes, rules, multisets of objects), one inputs a given multiset w in a specified region, and if the computation ever stops, then one says that w is accepted.

The accepting behavior is still more natural in the case of symport/antiport systems considered in [34]): just take a symport/antiport P system and consider the sequence

of symbols it brings inside from the environment during a halting computation; this sequence is said to be the string recognized by the computation (if several objects are taken at the same time, then any permutation of them is allowed).

Several types of P automata were considered: of the types above, with the request to introduce the string to be analyzed symbol by symbol, at the beginning of the computation, with one-way communication among membranes, with states associated with regions. A variant, closer to the way a problem is solved, by introducing (a code of) it in the initial configuration of a system, is to have a system, to introduce in the skin region a number, in the form of the multiplicity of a specified object (e.g., we introduce n copies of an object a), and let the system work; if the computation stops, then the number is accepted/recognized, otherwise the number is rejected.

Of a particular interest are the systems which work in a *deterministic* way, where at each step there is possible at most one transition. Such systems are needed when solving problems, for instance, decidability problems, where we cannot accept branchings, maybe leading to endless computations because of "wrong" choices of rules to apply, not because of the fact that the problem has no solution.

There are several universality results for various classes of P automata dealing with deterministic systems. Currently, it is an open problem whether or not a class of P automata exists such that the deterministic systems are strictly less powerful than the non-deterministic systems.

6.1.4 Tissue-Like P Systems

The cells are in most cases living together in complex organizations, in tissues, organs, organisms, establishing a complex communication net among them. For instance, when two protein channels from two adjacent cells come in contact (and this is enhanced by the fluid-mosaic structure/behavior of the membranes), it often happens that the two proteins establish a common channel, by which a direct communication among the two cells can be done. Having such a channel enhances the realization of further channels, and thus a network of direct channels appears, with a specific functionality in inter-cellular communication (see details in [46]). A rather similar situation appear if we take into account the organization of neurons in nets, with cells (neurons) establishing direct communication links among them through synapses – with the restriction now that we do no longer have the possibility of communication through the environment (one cell expells some objects and, in the next time unit, another cell can take them from the environment); it is also natural to suppose that the communication in a neural-like net is done in a one-way manner.

These observations directly lead to considering a class of P systems which also have a

natural mathematical motivation: instead of placing the membranes in a hierarchical manner, hence in the nodes of a tree, let us place them in the nodes of an arbitrary graph; such systems were introduced in [50].

Actually, making use of symport/antiport rules for direct communication and for communication with the environment, the communication graph is dynamically defined, depending on the rules used during a computation.

Specifically, the rules used for communicating among two cells with labels i and j should specify the targets, hence a symport rule transporting the objects of a multiset x from i to j has the form (i, x, j). If x is moved from i to j in exchange of objects of y, which are moved from j to i (this corresponds to an antiport rule), then we have a rule of the form $(i, x/y, j)$. In all cases, i and j should be different labels. One of i and j can also be equal to 0, identifying the environment.

Thus, a tissue-like P system is given by specifying the alphabet of objects, the list of cells, the sets of inter-cell communication rules, and the objects present initially in the environment; for each cell we have to specify the multiset of objects present in the initial configuration in the cell, as well as the rules for communication with the environment (because the targets are specified in the rules, all rules can be given as a global set for the whole system). The functioning of a tissue-like P system is again governed by the non-deterministic maximally parallel use of rules, with the result of a computation only obtained in a halting configuration. As for cell-like P systems, we can use these devices as generative mechanisms or as recognising mechanisms.

We do not introduce here neural-like P systems – which, actually, do not have a well established definition. For instance, in [58] there is a chapter devoted to such systems, with states associated with neurons and with the synapses pre-established; the evolution rules are rewriting-like, controlled by the states, and the communication is specified by commands *go* (meaning "go along any of the evailable synapses, maybe along all of them, after replication") and *out* (a way to send a result into the environment).

Several papers were devoted to tissue-like and to neural-like P systems; we refer to the Milano web page for details. Recently, a more involved variant of tissue-like P systems was introduced, under the name of population P systems – see [7].

6.1.5 Trading Space for Time in Solving Hard Problems

In the previous Section 3.9 of the book we have shown how SAT problem can be solved in linear time by means of P systems with active membranes, using the membrane division as a basic tool for obtaining an exponential workspace in a linear time. This result was much improved in the meantime.

First, the framework for dealing with related issues was formally better specified, and complexity classes for membrane computing were defined. An initial step in this direction was made in [58], then a more general and a more rigorous theory was started by Sevilla group – see, e.g., [68], [69] and the references therein. Informally speaking, the class $\mathbf{PMC}_X$ of problems which can be solved in polynomial time by a class X of P systems is defined as follows: Take a problem γ, with the *size* described by a number n. For the problem γ we construct a family $\{\Pi_n(\gamma) \mid n \geq 1\}$ of P systems of type X, such that $\Pi_n(\gamma)$ solves the instance $\gamma(n)$ of the problem. The construction should be done in a *uniform* way (that is, starting from γ, not from the instance $\gamma(n)$), in a polynomial time, by a Turing machine. Then, each $\Pi_n(\gamma)$ solves $\gamma(n)$ in the following way. We introduce a code of $\gamma(n)$ in a given region of $\Pi_n(\gamma)$, in the form of a multiset of objects; the system proceeds computing and all its computations should stop in a polynomial time (the system might be non-deterministic, hence branchings are allowed, but all possible computations stop in a number of steps bounded by a given polynomial), moreover, all computations send in the last step the same object to the environment. This object can be one of `yes` and `no`, and the object `yes` is sent out if and only if the instance $\gamma(n)$ has a positive answer. One says that the system is *confluent*, in the sense that all its computations ends with the same answer, *sound* (it gives the correct answer to the problem), and *complete* (it answers all instances of the problem).

Because in a polynomial time one cannot cheat and solve intractable (e.g., **NP**-complete) problems during the construction of the systems $\Pi_n(\gamma)$ (unless if $\mathbf{P} = \mathbf{NP}$), one also allows a *semi-uniform* way of constructing the systems, that is, starting from the instance $\gamma(n)$ itself and constructing directly $\Pi_n(\gamma)$ for solving the given instance.

Many problems were treated in this framework, and, worth mentioning, not only decidability problems, having `yes/no` answers, but also numerical problems, such as the Knapsack problem, the Subset-Sum problem etc. Details can be found, e.g., in [67], [66].

What is interesting is that in these papers one uses P systems with active membranes of a restricted type. For instance, the division of non-elementary membranes was never used (in many cases also the rules for the membrane dissolution are avoided). Another improvement concerns the number of polarizations used for solving **NP**-complete problems: as shown in [4], two polarizations are enough (this is also true for the universality). It is an intriguing open problem whether or not we can completely get rid of polarizations (of course, without introducing further features, such as label changing, priorities, etc).

On the other hand, by using division of non-elementary membranes, it was shown that even **PSPACE** problems can be solved in a polynomial time; this was shown in [76] for `QSAT`, the satisfiability of quantified propositional formulas in the conjunctive normal form. Also in this context there appears an important open problem: is the division of non-elementary membranes necessary for covering **PSPACE**, or division of elementary membranes suffice?

We have menrioned above that membrane division is only one of the ways used in order to solve hard problems in a feasible time by means of P systems. The other possibilities, less investigated than membrane division, are *membrane creation* and *string replication*. Like membrane division, membrane creation is used in systems working with symbol-objects; string replication is used for string-object systems, which process the strings by rewriting. Specifically, new membranes are created by rules of the form $a \to [_h v]_h$ (a membrane with label h is created, starting from an object a; inside the new membrane we place the objects specified by the multiset v – while the set of rules acting in this membrane is indicated by the label h). In turn, the string replication rules are of the form $a \to u_1||u_2$; when rewriting a string xay by such a rule we get two strings, xu_1y and xu_2y.

For both these possibilities of creating working space it was shown that **NP**-complete problems (typically, `SAT` and `HPP` – the Hamiltonian Path Problem) can be solved in polynomial tyme. We refer to [58] for details and references.

6.1.6 Implementations

Up to now there is no attempt to implement P systems in a lab, on a bio-ware. Actually, it is not clear which is the most realistic strategy, to follow the model of other areas of natural computing, such as genetic algorithms (more general, evolutionary computing) and neural computing, and to have implementations on the electronic computer, or to imitate the ambition of DNA computing (in general, molecular computing) and to go to laboratory, trying to have a new type of hardware, based on biochemistry. It is also possible that none of these directions will be successful, and then membrane computing can remain an intellectual aventure, with other types of applications rathen than directly in computer science, in providing new types of hardware or new types of algorithms. Anyway, there are numerous attempts to implement – actually, to simulate – P systems on the existing computers. (Because the P systems are inherently parallel and, in many variants, they also exhibit an intrinsic non-determinism, while the existing computers are basically sequential, we cannot speak of a real implementation when dealing with software products aiming to simulate P systems on electronic computers.) However, there are probably more than two dozens of P systems simulators, some of them simpler, some of them more sophisticated (for instance, providing graphical interfaces, or the possibility to follow several parameters during the evolution of the system). References can be found in the membrane computing web page, where also some programs are freely available. Such programs are not only non-trivial programming exercises, but they have also a didactic or scientific value; some of them were used in checking proofs, where given P systems were simulated during a large number of steps in search of possible unforessen evolutions. Most of these programs deal with symbol-objects P systems, and can incorporate cooperative rules, as well as priority relations or other features. There

are also programs which simulate P systems with active membranes.

While almost all of these programs deal with "standard" P systems, as defined in the previous sections, some of them also deal with probabilities or reaction rates assigned to evolution rules, either given in advance or dynamically computed during the evolution of the system, according, e.g., to the population of objects from a given region of the system at a given time. Of course, such programs are closer to the biochemical reality of the cell and they were the programs used in applications related to the cell biology – see the next section.

Very important is the information provided by the programs. In most cases, one can follow the state of the system along different paths of the computation, with or without graphs displaying the population of certain objects in time. Rather suggestive is the output provided by a program used in [72], based on [26], which provides in a *Mathematica* style the image of the so-called *Sevilla carpet* associated with a computation (the Sevilla carpet is the parallel computing counterpart of the Szilard language from language theory: a rectangle is constructed, indicating in each time unit which rules are used; actually, for each rule one gives the number of times the rule is used, so that a two-variable mapping is obtained, with natural values; Sevilla carpet gives in this way an image of the combined space-time complexity of the computation – details can be found in [23]).

Of a special interest are the implementations/simulations on a distributed hardware, such as those done in [24] and [80], and, especially, that reported in [70], which has used a reconfigurable hardware.

On the other hand, it is highly possible that a more fruitful approach from the computer science point of view is not to implement a P system as it is, as a whole model, but to incorporate in new types of software products features of membrane computing, more general, features of cell structure and functioning, such as compartmentalization through membranes, localization, decentralization, loose control of program units, and so on. An illustration of this strategy is LMNtal (read "elemental"), the language developed by T. Ueda and his collaborators, where membranes and multisets appear explicitly in the language design – see [83].

6.1.7 Applications

Membrane computing started from the cell biology as an attempt to learn something useful for computer science, not as a model of the "real" cell; after becoming abstract enough and mathematically enough developed, the theory returned to biology, as a possible tool for simulating not only local phenomena from the cell but possibly also simulating the cell as a whole. This is by no means an easy task: in several places it was

said that after completing the genome project, the main challenge for the bio-informatics of our century is to model and simulate the cell as a whole (see, e.g., [43], [82]). On the other hand, membrane computing has some intrinsic features which make it a good candidate for a framework to approach the whole cell simulation: it is a discrete and algorithmic model, easy to implement on a computer and, rather important, easy to scale-up (which is one of the difficulties of dealing with differential ecuations); moreover, the model is easy to understand by biologists, hence easy to check, modify, apply; finally, this is a model which has been initiated with inspiration from the cell structure and functioning, it is not coming "from outside", it is not necessary to adapt it to a new reality, as it happens with models initially used in physics, mechanics, linguistics and then applied to biology.

Actually, up to now, only local processes were modelled with P systems, along a strategy of the following type: one captures in a P systems model the features of a specified biological process (dealing with populations of molecules evolving in the compartments of a cell), one takes or one writes a program for simulating that type of P systems, and one performs experiments with the program, tunning certain parameters and following the evolution of the system, hence of the real process, in various circumstances; in most cases, graphical representations of the evolution in time of multiplicities of certain objects are provided. Respiration in bacteria [5], photosynthesis [52], processes related to the imune system [31], [78], and other processes [24], [77] were studied in this way. We do not recall any detail here, but we refer to the papers mentioned above, to the chapter of [58] devoted to biological applications, as well as to the papers available in the Milano web page.

This is the "standard" type of applications in biology. A less operational one, but tried already also in linguistics and management, is to mainly use the *language* of MC. This means not only the long list of concepts either newly introduced, or related to each other in a new manner in this area, but also the way to represent a cell-like structure, as proposed in membrane computing. This representation is rather attractive for biologist: Euler-Venn diagrams, with labels for membranes, with multisets of objects (chemicals) placed in regions, and with sets of rules placed either in regions (the case of rewriting-like rules) or near membranes, to suggest that they are associated with the membranes (the case of symport/antiport rules).

A promising aplication – and not so expected, although this was happening also for Lindenmayer systems – is in computer graphics, see [38].

6.1.8 Using Fuzzy Set Theory

Althought there are a few papers proposing ways to "fuzzify" P systems, this is mainly a research topic, of a real interest for applications in biology. Counting the multiplicity of

objects is by no means a realistic assumption; the biologists work with concentrations, gradients, with a "linguistic logic" dealing with terms such as "sufficient", "too much", "less than necessary", "less probable", and so on. We mention here only a few papers dealing with fuzzy P systems: [17], [54], [79], as well as [53], [27] for approaches to probabilistic P systems.

6.1.9 Dynamic Systems Approaches

Similar to the previous topic, also investigating P systems as dynamical systems and not as computing devices is mainly a subject for further researches. The interest for applications is obvious: halting is a computer science feature, Turing inspired, while the goal of cells is life, the evolution itself. Duration, periodicity, attractors, reachability of configurations with specified properties, stability, and other related notions become central, moving to a secondary level of interest the computability issues. We again mention only a few titles, [10] and [47], with the mentioning that the second one contains a large number of ideas for future researches, as well as a comprehensive bibliography.

6.1.10 Computing Beyond Turing

We have mentioned above this topic in the framework of other "advanced" issues, such as the reversibility of computations, and the conservation of objects (which happens in symport/antiport systems, but not necessarily in transition P systems with multiset rewriting rules) or of energy related to object evolution. The web page many times mentioned above contains several titles related to these topics.

Is it possible to use P systems to increase the computational power beyond Turing? The answer is affirmative (see [14]); more details will be given in the next section.

6.1.11 Concluding Remarks

As stated in the introduction, this section was only intended to offer a bird eye view of the recent developments in membrane computing, especially pointing out research topics which were or it is plausible/important to be investigated, and providing bibliographical hints of interest. Actually, the bibliography below contains also unrefereed titles, especially of collective volumes and PhD theses, where much more material can be found. The reader interested in membrane computing is strongly advised to follow the developments of the domain through the web page in Milano, in particular, looking for the volumes of the next editions of the workshops and the brainstorming weeks devoted to membrane computing.

6.2 Trespassing the Turing Barrier

For more than fifty years the Turing machine model of computation has defined what it means to "compute" something; the foundations of the modern theory of computing are based on it.

Furthermore, as it has been noted by various authors (see, for example, [13]), the (silicon) computer, whose capacity for handling information has been growing at a rate ten million times faster than information handling did in our nervous system during the more than 600 million years of evolution, seems to be the only important commodity ever to become exponentially better as it gets cheaper. However, this exponential race is essentially "Turing-bounded", it cannot produce feasible solutions to many intractable problems and, more importantly for our investigations here, it cannot solve Turing unsolvable problems.

Hypercomputation or super-Turing computation is a "computation" that transcends the limit imposed by Turing's model. For a recent perspective one can consult the special issues of the journal *Minds and Machines*, [25]; see also [55] for a lucid analysis, [88] for a comprehensive bibliography and the section 'Computation and Turing machines' in [81] (especially the critical paper [29]).

Are these studies just mere idle speculations, pure theoretical abstractions studied for their own sake, with little or no regard to the 'real world'? We dare to answer this rhetorical question in the negative. First, according to [18], "*If* science really is essentially the carrying out of a calculation, then the limits of science are necessarily extended whenever we extend our computational capabilities". Promises are very high: if materialised, science will reach points that have never been seen before. But, what about the case of failure? Even in this case the gain will be also immense (and, arguably, scientifically more interesting), as negative results will reveal new *limits*, with far-reaching implications in mathematics, computer science, physics, biology, and philosophy.

6.2.1 What is Turing Barrier?

Recall the Merchant Problem: we have 10 stacks of coins, each stack containing 100 coins, and we know that *at most one stack contains only false coins, weighting* $1.01\,g$; true coins weight $1\,g$. The problem is to find the stack with false coins (if any) *by only one weighting*. The classical solution reduces the problem to the weighting of a special combination of coins: one coin from the first stack, two coins from the second stack, ..., ten coins from the tenth stack. If the false coins are present in the N-th stack, then the weight of the combination will be $55 + \frac{N}{100}\,g$; otherwise the weight is just $55\,g$. Probably the elegant solution described above was the very first solution of a

computational problem bearing typical features of quantum computing, see an extended discussion in [15].

In the Infinite Merchant Problem we assume that *we have countable many stacks, given in some computable way, all of them, except at most one, containing true coins only.* True coins weight 1 and false coins weight $1 + 2^{-j}$, $j > 0$. Again we are allowed to take a coin from each stack and we want to determine whether all coins are true or there is a stack of false coins.

First, one can show that the Infinite Merchant Problem is classically undecidable by reducing it to the Halting Problem, i.e. the problem to decide whether an arbitrary Turing machine halts on an arbitrary input; hence, the the Infinite Merchant Problem is not solvable by any Turing machine. The undecidability is determined by the impossibility to decide in a finite time the answers to an infinite number of questions, "does the first stack contain a false coin?", "does the second stack contain a false coin?", etc. This might be caused either by the fact that the time of the computation grows indefinitely or by the fact that the space of computation grows indefinitely or both. The classical theories of computability and complexity (see, for example, [12]) do not give any indication in this respect.

6.2.2 A Quantum Approach

The quantum approach to the Infinite Merchant Problem developed in [15, 1] shows that time can be made finite provided we use a specific probabilistic strategy.[2] We are given a probability $\theta = 2^{-n}$ and we assume that we work with a device (described below) with sensitivity given by a real $\varepsilon = 2^{-m}$. Then, we compute classically a time $T = T_{\theta,\varepsilon}$ and run the "device" on a random input for the time T. If we get a click, then the system has false coins; if we do not get a click, then we conclude that with probability greater than $1 - \theta$ all coins are true. An essential part of the method is the requirement that the time limit T is *classically computable.*

The device (with sensitivity ε) will distinguish the values of the iterated quadratic form $\langle \mathbb{Q}^t(\mathbf{x}), \mathbf{x} \rangle = \sum_{i=1}^{\infty} q_i^t \, |x_i|^2$, by observing the difference between averaging over trajectories of two discrete random walks with two non-perturbed and perturbed sequences t_l, $\tilde{t}_l$ of "stops". The non-perturbed sequence corresponds to equal steps $\delta_m = 1$, $t_l = \sum_{m=0}^{l} \delta_m$, and the perturbed corresponds to the varying steps Δ_m, $0 < \Delta_m < \delta_m$, $\tilde{t}_l = \sum_{m=0}^{l} \Delta_m$. The device sensitivity is defined in terms of the Sobolev norm.

Two cases may appear. If for some $T > 0$, $\langle \mathbf{Q}^T(\mathbf{x}), \mathbf{x} \rangle \geq \| \mathbf{x} \|^2 + \varepsilon \| \mathbf{x} \|_1^2$, then the device has clicked and we know for *sure* that there exist false coins in the system. However, it is possible that at some time $T > 0$ the device hasn't (yet?) clicked because

[2] Other "physical" approaches to solve Turing uncomputable problems have been proposed in [30, 42].

$\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x}\rangle < \| \mathbf{x} \|^2 + \varepsilon \| \mathbf{x} \|_1^2$. This may happen because either all coins are true, i.e., $\langle \mathbf{Q}^t(\mathbf{x}), \mathbf{x}\rangle < \| \mathbf{x} \|^2 + \varepsilon \| \mathbf{x} \|_1^2$, for all $t > 0$, or because at time T the growth of $\langle \mathbf{Q}^T(\mathbf{x}), \mathbf{x}\rangle$ hasn't yet reached the threshold $\| \mathbf{x} \|^2 + \varepsilon \| \mathbf{x} \|_1^2$. In the first case the device will *never* click, so at each stage t the test-vector $\mathbf{x}$ produces "true" information; we can call $\mathbf{x}$ a "true" vector. In the second case, the test-vector $\mathbf{x}$ is "lying" at time T as we *do* have false coins in the system, but they were not detected at time T; we say that $\mathbf{x}$ produces "false" information at time T.

If we assume that there exist false coins in the system, say at stack j, but the "device" does not click at the moment T, then the test-vector $\mathbf{x}$ belongs to the *indistinguishable set* $\mathcal{F}_{\varepsilon,T} = \{\mathbf{x} \in l_2^1 \mid ((1+\gamma)^T - 1) \mid x_j \mid^2 < \varepsilon \| \mathbf{x} \|_1^2$, for some $j\}$.

In [15] it was proven that the Wiener measure of the indistinguishable set, $\tilde{W}(\mathcal{F}_{\varepsilon,T})$ *converges computably to zero.* Denote by $P(\mathcal{N})$ the *a priori* probability of absence of false coins in the system. Then, the *a posteriori probability that the system contains only true coins, when the device did not click after running the experiment for the time T, is*

$$\mathbf{P}_{\text{non-click}}(\mathcal{N}) > 1 - \frac{1 - P(\mathcal{N})}{P(\mathcal{N})} \cdot \frac{\sqrt{\varepsilon}}{\sqrt{(1+\gamma)^T - 1 - \varepsilon}\,\sqrt{\prod_{m=1}^{\infty} \Delta_m}}.$$

A Brownian solution based on resonance amplification was discussed in [1]. A special role in the above scenario is played by the ability of producing truly random inputs for the described device. Is this physically possible? The answer is affirmative: not only it is possible to produce reliable truly quantum random bits, but there exist commercial products doing this, for example *QRandom* produced by GAP-Optique, a company affiliated with the University of Geneva, see [71].

6.2.3 A P Systems Approach

In what concerns the possibility of computing beyond Turing by using P systems, there is only one paper devoted to this topic, [14]. Recall that the main problem is *to carry on an infinite computation in a finite amount of time.* The basic tool to achieve this is *acceleration*: as in Figure 1 we have two scales of time, an *external, global* one, of the "user" of the accelerated device (the black box in the figure), and the *internal, local* time of the device. The problem is formulated in global time, at some moment t, and introduced into the accelerated device, which is able to perform an 'inner' infinite computation in a finite number, T, of external time units, when the "user" gets the answer to the problem.

The idea came from [73, 11, 87] who observed that a process that performs its first step in one unit of (global) time, the second step in 1/2 unit of (global) time, and, in general, each subsequent step in half the (global) time of the step before, is able to complete an

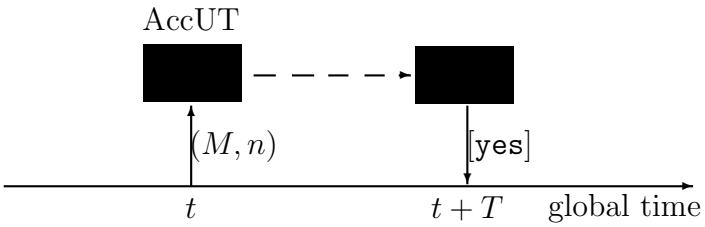

Figure 1: The interplay between local and global time used for solving the Halting Problem by means of an accelerated device

infinity of steps in just two global units of time since $1 + \frac{1}{2} + \frac{1}{4} + \cdots = 2$. A universal Turing machine working in this kind of accelerated manner is capable of deciding the Halting Problem: at some (global) time t one introduces (the code of) any particular Turing machine M and an input n into the accelerated universal Turing machine, and in two (global) time units we have the answer, yes or no, whether $M(n)$ halts or not (here, $T = 2$).

Acceleration does not conflict with the Turing model of computation as the mathematical definition of a Turing machine does not specify how long does it take to perform an individual step. Even more, [86] has shown that no known physical law forbids such an acceleration.

We take here a different approach, grounded on suggestions coming from cell and brain biology: starting from the observation that nature creates smaller and smaller compartments in a cell in ordet to enhance the reactivity of the enclosed chemicals, we can introduce the hypothesis that the time is not measured with the same unit in all compartments of a P system, but lower compartments have smaller units. More exactly, we assume that the time unit decreases in such a way that, if the membrane structure becomes deeper and deeper (for instance, by means of membrane creation), then an arbitrarily deep structure can be obtained, with the time units smaller and smaller, such that arbitrarily many steps of computation can be done in an inner membrane while only one time unit elapses outside the system. In this way, in a constant number of units of external time (time of observer), the system can perform arbitrarily many steps in its compartments, and this can lead to computations which cannot be done by Turing machines.

The same result is obtained for tissue-like P systems or neural-like P systems if we suppose that the channels among cells/neurons are faster and faster during the computation (they "learn", hence become continuously more efficient). It was in this context that the important idea of deterministic computations was considered for the first time for recognizing P systems. In both models acceleration is a part of the hardware not a quality of the environment; ultimately, these hypotheses should be validated or not by biophysics. Details can be found in [14].

References

[1] V. A. Adamyan, C. S. Calude, B. S. Pavlov, Transcending the limits of Turing computability, in T. Hida (ed.). *Proceedings of Winter School, Meijo University, Japan*, World Scientific, Singapore, to appear.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K Roberts, P. Walter, *Molecular Biology of the Cell*, 4th ed. Garland Science, New York, 2002.

[3] A. Alhazov, C. Martin-Vide, Gh. Păun, eds., *Pre-Proceedings of Workshop on Membrane Computing*, WMC 2003. Tarragona, Spain, July 2003, Technical Report 28/03, Rovira i Virgili University, Tarragona, 2003.

[4] A. Alhazov, R. Freund, Gh. Păun, P systems with active membranes and two polarizations. In [62], 20–36.

[5] I.I. Ardelean, M. Cavaliere, Modelling biological processes by using a probabilistic P system software. *Natural Computing*, 2, 2 (2003), 173–197.

[6] G. Bel-Enguix, R. Gramatovici, Parsing with active P automata. In [48], 31–42.

[7] F. Bernardini, M. Gheorghe, Population P systems. *Journal of Universal Computer Science*, 10, 5 (2004), 509–539.

[8] D. Besozzi, *Computational and modelling power of P systems*. PhD Thesis, Univ. degli Studi di Milano, 2004.

[9] D. Besozzi, I.I. Ardelean, G. Mauri, The potential of P systems for modeling the activity of mechanosensitive channels in E. Coli. In [3], 84–102.

[10] L. Bianco, F. Fontana, G. Franco, V. Manca, P systems in bio systems, in [22], 2004.

[11] R. M. Blake, The paradoxes of temporal process, *Journal of Philosophy*, 23 (1926), 645–654.

[12] C. S. Calude, *Information and Randomness: An Algorithmic Perspective*, 2nd Edition, Revised and Extended, Springer Verlag, Berlin, 2002.

[13] C.S. Calude, J.L. Casti, Parallel thinking, *Nature*, 392 (9 April 1998), 549–551.

[14] C.S. Calude, Gh. Păun, Bio-steps beyond Turing. *CDMTCS Research Report* 226, Univ. of Auckland (November 2003), and *BioSystems*, 2004.

[15] C. S. Calude, B. Pavlov, Coins, quantum measurements, and Turing's barrier, *Quantum Information Processing* 1, 1–2 (2002), 107–127.

[16] C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds., *Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View. Lecture Notes in Computer Science*, 2235, Springer-Verlag, Berlin, 2001.

[17] J. Casasnovas, J. Miro, M. Moya, F. Rossello, An approach to membrane computing under inexactitude, *Intern. J. Foundations of Computer Sci.*, to appear.

[18] J.L. Casti, *The One True Platonic Haven*, Joseph Henry Press, New York, 2003.

[19] M. Cavaliere, Evolution-communication P systems. In [64], 134–145.

[20] M. Cavaliere, C. Martin-Vide, Gh. Păun, eds., *Proceedings of the Brainstorming Week on Membrane Computing; Tarragona, February 2003*. Technical Report 26/03, Rovira i Virgili University, Tarragona, 2003.

[21] R. Ceterchi, R. Gramatovici, N. Jonoska, K.G. Subramanian, Tissue-like P systems with active membranes for picture generation. *Fundamenta Informaticae*, 56, 4 (2003), 311–328.

[22] G. Ciobanu, Gh. Păun, M.J. Perez-Jimenez, eds. *Applications of Membrane Computing*. Springer-Verlag, Berlin (in preparation).

[23] G. Ciobanu, Gh. Păun, Gh. Ştefănescu, Sevilla carpets associated with P systems. In [20], 135–140.

[24] G. Ciobanu, G. Wenyuan, A P system running on a cluster of computers. In [48], 123–139.

[25] B.J. Copeland, Hypercomputation, *Minds and Machines*, 12, 4 (2002), 461–502.

[26] A. Cordon-Franco, M.A. Gutierrez-Naranjo, M. Perez-Jimenez, F. Sancho-Caparrini, Implementing in Prolog an effective cellular solution to the Knapsack problem. In [48], 140–152.

[27] A. Cordon-Franco, F. Sancho-Caparrini, A note on complexity measures for probabilistic P systems. *Journal of Universal Computer Science*, 10, 5 (2004), 559–539.

[28] E. Csuhaj-Varju, G. Vaszil, P automata or purely communicating accepting P systems. In [3], 219–233.

[29] M. Davis, The myth of hypercomputation, in [81], 195–211.

[30] G. Etesi, I. Németi, Non-Turing computations via Malament-Hogarth space-times, *International Journal of Theoretical Physics*, 41 (2002), 341–370.

[31] G. Franco, V. Manca, A membrane system for the leukocyte selective recruitment. In [48], 180–189.

[32] R. Freund, Special variants of P systems inducing an infinite hierarchy with respect to the number of membranes. *Bulletin of the EATCS*, 75 (October 2001), 209–219.

[33] R. Freund, L. Kari, M. Oswald, P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient. *Theoretical Computer Science*, in press.

[34] R. Freund, M. Oswald, A short note on analysing P systems. *Bulletin of the EATCS*, 78 (2003), 231–236.

[35] R. Freund, A. Păun, Membrane systems with symport/antiport: universality results. in [64], 270–287.

[36] P. Frisco, *Theory of Molecular Computing. Splicing and Membrane Systems*. PhD Thesis, Leiden University, The Netherlands, 2004.

[37] P. Frisco, H.J. Hoogeboom, Simulating counter automata by P systems with symport/antiport. In [64], 288–301.

[38] A. Georgiou, M. Gheorghe, F. Bernardini, Generative devices used in graphics. In [22].

[39] O.H. Ibarra, On the computational complexity of membrane computing systems. *Theoretical Computer Science*, 320, 1 (2004), 98–109.

[40] O.H. Ibarra, The number of membranes matters. In [3], 218–231.

[41] O.H. Ibarra, Z. Dang, O. Egecioglu, Catalytic membrane systems, semilinear sets, and vector addition systems. *Theoretical Computer Sci.*, 312, 2-3 (2004), 378–400.

[42] T.D. Kieu, Quantum hypercomputation, *Minds and Machines*, 12, 4 (2002), 541–561.

[43] H. Kitano, Computational systems biology. *Nature*, 420, 14 (2002), 206–210.

[44] S.N. Krishna, Infinite hierarchies on some variants of P systems. Submitted, 2003.

[45] A. Leporati, C. Zandron, G. Mauri, Conservative computations in energy-based P systems. [51], 268–282.

[46] W.R. Loewenstein, *The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life*. Oxford University Press, New York, Oxford, 1999.

[47] V. Manca, On the dynamics of P systems. In [51], 29–43.

[48] C. Martin-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa, eds., *Membrane Computing. International Workshop, WMC2003, Tarragona, Spain, Revised Papers. Lecture Notes in Computer Science*, 2933, Springer-Verlag, Berlin, 2004.

[49] C. Martin-Vide, Gh. Păun, P systems with symport/antiport rules; A survey. In *Modeling in Molecular Biology* (G. Ciobanu, G. Rozenberg, eds.), Springer-Verlag, Berlin, 2004, 175–192.

[50] C. Martin-Vide, Gh. Păun, J. Pazos, A. Rodriguez-Paton, Tissue P systems. *Theoretical Computer Sci.*, 296, 2 (2003), 295–326 (see also TUCS Technical Report No. 421, September 2001).

[51] G. Mauri, Gh. Păun, C. Zandron, eds., *Pre-proceedings of Fifth Workshop in Membrane Computing, WMC5*, Milano, MolCoNet Publication, 2004.

[52] T.Y. Nishida, Simulations of photosynthesis by a K-subset transforming system with membranes. *Fundamenta Informaticae*, 49, 1-3 (2002), 249–259.

[53] A. Obtulowicz, Probabilistic P systems. In [64], 377–387.

[54] A. Obtulowicz, General multi-fuzzy sets and fuzzy membrane systems. In [51], 316–326.

[55] T. Ord, *Hypercomputation: Computing more than the Turing Machine*, Honours Thesis, Computer Science Department, University of Melbourne, Australia, 2002; `arxiv.org/ftp/math/papers/0209/0209332.pdf`.

[56] A. Păun, Gh. Păun, The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–306.

[57] Gh. Păun, Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998, `www.tucs.fi`).

[58] Gh. Păun, *Computing with Membranes: An Introduction*. Springer-Verlag, Berlin, 2002.

[59] Gh. Păun, Membrane computing: Main ideas, basic results, applications. In *Molecular Computational Models: Unconventional Approaches* (M. Gheorghe, ed.), Idea Group Publ., London, 2004.

[60] Gh. Păun, Membrane computing: Power and efficiency (A quick overview). *Proc DNA10*, Milano, 2004 (C. Ferretti, G. Mauri, C. Zandron, eds.), Univ. Milano-Bicocca, 2004, 1–15.

[61] Gh. Păun, Membrane computing (after the second Brainstorming Week, Sevilla, February 2004). *Bulletin of the EATCS*, 83 (June 2004), 159–170.

[62] Gh. Păun, A. Riscos-Nunez, A. Romero-Jimenez, F. Sancho-Caparrini, eds., *Proceedings of the Second Brainstorming Week on Membrane Computing, Sevilla, February 2004*. Technical Report 01/04 of Research Group on Natural Computing, Sevilla University, Spain, 2004

[63] Gh. Păun, G. Rozenberg, A guide to membrane computing. *Theoretical Computer Science*, 287, 1 (2002), 73–100.

[64] Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds. *Membrane Computing. International Workshop, WMC-CdeA 2002, Curtea de Argeş, Romania, Revised Papers. Lecture Notes in Computer Science*, 2597, Springer-Verlag, Berlin, 2003.

[65] Gh. Păun, C. Zandron, eds., *Pre-proceedings of Workshop on Membrane Computing*. Curtea de Argeş, Romania, August 2002, MolCoNet Publication No 1, 2002.

[66] M.J. Perez-Jimenez, A. Riscos-Nunez, A linear-time solution for the Knapsack problem using active membranes. In [48], 250–268.

[67] M.J. Perez-Jimenez, A. Riscos-Nunez, Solving SUBSET-SUM problem by P systems with active membranes. *New Generation Computing*, 2004.

[68] M. Perez-Jimenez, A. Romero-Jimenez, F. Sancho-Caparrini, *Teoría de la Complejidad en Modelos de Computatión Celular con Membranas*. Editorial Kronos, Sevilla, 2002.

[69] M. Perez-Jimenez, A. Romero-Jimenez, F. Sancho-Caparrini, Complexity classes in cellular computation with membranes. *Natural Computing*, 2, 3 (2003), 265–285.

[70] B. Petreska, C. Teuscher, A reconfigurable hardware membrane system. In [48], 267–283.

[71] `http://www.idquantique.com/.`

[72] A. Riscos-Nunez, *Programacion celular. Resolucion eficiente de problemas numericos NP-complete*. PhD Thesis, Univ. Sevilla, 2004.

[73] B.A.W. Russell, The limits of empiricism, *Proceedings of the Aristotelian Society*, 36 (1936), 131–150.

[74] P. Sosik, P systems versus register machines: two universality proofs. In [65], 371–382.

[75] P. Sosik, The power of catalysts and priorities in membrane systems. *Grammars*, 6, 1 (2003), 13–24.

[76] P. Sosik, The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing*, 2, 3 (2003), 287–298.

[77] Y. Suzuki, Y. Fujiwara, H. Tanaka, J. Takabayashi, Artificial life applications of a class of P systems: Abstract rewriting systems on multisets. In [16], 299–346.

[78] Y. Suzuki, S. Ogishima, H. Tanaka, Modeling the p53 signaling network by using P systems. In [20], 449–454.

[79] A. Syropoulos, Fuzzifying P systems. Submitted, 2004.

[80] A. Syropoulos, P.C. Allilomes, E.G. Mamatas, K.T. Sotiriades, A distributed simulation of P systems. In [48], 355–366.

[81] C. Teuscher (ed.), *Alan Turing: Life and Legacy of a Great Thinker*, Springer-Verlag, Heidelberg, 2003.

[82] M. Tomita, Whole-cell simulation: A grand challenge of the 21st century. *Trends in Biotechnology*, 19 (2001), 205–210.

[83] K. Ueda, N. Kato, LNMtal: a language model with links and membranes. In [51], 65–79.

[84] G. Vaszil, On the size of P systems with minimal symport/antiport. In [51], 422–431.

[85] C. Zandron, C. Ferretti, G. Mauri, Solving NP-complete problems using P systems with active membranes. In *Unconventional Models of Computation* (I. Antoniou, C.S. Calude, M.J. Dinneen, eds.), Springer-Verlag, London, 2000, 289–301.

[86] K. Svozil, The Church-Turing Thesis as a guiding principle for physics, in C.S. Calude, J. Casti, M.J. Dinneen (eds.). *Unconventional Models of Computation*, Springer-Verlag, Singapore, 1998, 371–385.

[87] H. Weyl, *Philosophie der Mathematik und Natureissenschaft*, R. Oldenburg, Munich, 1927.

[88] `www.hypercomputation.net/bib`.