Further Open Problems in Membrane Computing

Gheorghe PĂUN

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
and
Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Technical Higher School of Computer Science Engineering
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: gpaun@us.es

Abstract. A series of open problems and research topics in membrane computing are pointed out, most of them suggested by recent developments in this area. Many of these problems have several facets and branchings, and further facets and branchings can surely be found after addressing them in a more careful manner.

1 Introduction

The main goal of this note is to challenge the reader–participant to the second Brainstorming Week on Membrane Computing, Sevilla, February 2004, if not to address the problems themselves which follow, at least to formulate and circulate his/her own favourite research topics, thus enhancing the cooperation in this area (in particular, during the Brainstorming).

Open problems can be found in many papers dealing with P systems, sometimes not explicitly stated; for instance, the general question of improving computational power or computational complexity results from the point of view of various descriptional complexity measures can be formulated with respect to many (if not all) results from the literature. This concerns not only the "standard" question about the number of membranes, but also the number and the size of rules, the number of catalysts, promoters, etc. Comprehensive lists of problems appear in [19], as well as in [7] and [20]. In [20] one states 39 open problems – several of them partially or completely solved in the meantime (a useful paper could be one presenting the state-of-the-art of all these questions, with the progresses made since the publication of the book, and the problems suggested by the related developments).

The problems below are either different from those from the sources mentioned above, or presented from a new angle and with new motivations and related references. Some of them are precise technical questions, some others are rather general (research topics, somewhat vaguely formulated; more precise formulations will already be a first step towards answering these topics).

In what follows, we give references only for papers which are not already present in the bibliography of [20]; most of these new papers are available, in most cases in preliminary versions, at `http://psystems.disco.unimib.it`; in general, the reader is adviced to consult this web page for news of any type in the membrane computing area.

2 Determinism Versus Non-Determinism

We start with a very important question for automata and language theory, for computability in general, and which was only partially addressed in membrane computing: the power of non-determinism, the difference (mainly in what concerns the computing power) between deterministic and non-deterministic systems.

Because of the biological inspiration, P systems were introduced as inherently non-deterministic devices and in most papers they were considered in this form. Restrictions about the branching possibilities were considered first when devising membrane computing solutions to decision (and, later, numerical) problems. The first restriction was to *confluent* systems, which, after possible unrestricted branchings, converge to a class of "similar-and-meaningful" configurations, or even to a single specified configuration, which ensures a sound answer to the problem. Then, deterministic systems in the standard sense were considered and constructed (see [23] and many papers of the Sevilla team, O.H. Ibarra, etc.).

An important distinction should be made here: when dealing with generative P systems (with the output associated with halting computations only) the determinism is a strong restriction, leading to non-interesting systems, which either always halt in the same configuration (thus producing a single output) or always diverge (thus producing nothing). The case of accepting/recognizing systems is different, because, in some sense, the non-determinism is moved into the environment, it is "in charge of the user": the computation starts from any given input, which is accepted or not, according to certain conditions (e.g., by halting).

This observation suggests a first research topic: what about considering deterministic generative P systems, with the output defined in such a way to have non-trivial computing devices? The immediate idea is to remove the halting condition, and to collect outputs in a different manner, even from computations which never halt. Suggestions from the area of deterministic L systems can be useful. A possibility was used in a different framework in [1]: working with string-objects with replicated rewriting, and sending strings out of the system at any moment; because of the replication, other strings remain inside, and the computation can continue. Not so clear (hence more challenging) is the case of symbol-objects. Signaling the moments when a number (for instance, of objects present in a given region) is to be considered as "computed" by means of the appearance of certain objects (signals) can be a possible way to obtain a meaningful definition (such signal-objects were used, e.g., in [15]).

Having in mind the restricted generative power of, e.g., D0L systems, it is highly possible that deterministic generative P systems of many types will not be computationally universal, and this would add to the interest for such systems.

As we have mentioned above, determinism is natural for accepting systems. With motivations related to "computing beyond Turing", such systems were considered in [6], where universality proofs were given for deterministic P systems (with symport/antiport). The problem was further investigated in [14], but still there are many classes of P systems for which it remains to be addressed. A challenging case is that of accepting P systems

with catalytic rules. In the generative (non-deterministic) case, two catalysts are know to suffice in order to get universality – see [13], [12]. Are the accepting deterministic catalytic P systems universal? How many catalysts are needed?

This question suggests a general problem of a clear interest: find a class of P systems, of any type (generative or accepting), for which the deterministic systems are strictly less powerful than the non-deterministic ones. A positive answer to this question (for a class of P systems already considered in the literature, or defined in a natural/honest manner) would be rather sound and would surely have interesting consequences.

In what concerns the way of introducing the input to be accepted/recognized (hence the way to relate the external non-determinism, coming from the environment/user, with the computation), it is important to mention that besides the two standard ideas, of (1) introducing the number/string to be recognized in a specified region of the system, and to trigger in this way the computation, or (2) considering the sequence of objects/multisets taken into the system during a computation (see, e.g., [10]), an alternative was proposed in [14]: considering signals. In the case of one signal, a special object s is distinguished, not present in the system or in the environment; the computation starts (in step 1) and, if object s appears in the environment in step $t+1$, then the number t is to be recognized. In the case of two signals (different or not), the number of steps elapsed between the occurrence of the two signals is the number to be recognized.

Any other way of introducing the input to be analysed by a P system?

A special case is that of symport/antiport P systems with the result of a computation being the (coding) of the *trace* of a distinguished object. What about accepting-like trace P systems? We work with symbol-objects, hence the "standard" input cannot be anything else than a number (of objects introduced in a given region, or defined by means of signals); the result of a computation is a string, hence in this way we relate an input number n to an output string w_n. We get a sort of transducer, mapping numbers to strings. Which are the properties of such devices? Does the determinism imply a proper decrease in power?

We are convinced that the study of determinism in membrane computing is an important topic, not only natural from a mathematical point of view, leading to many technical questions (only part of them mentioned above), but also from the point of view of applications, for instance, in modeling and simulating biological phenomena/systems.

3 Neural-Like P Systems

This is another general research topic of a clear – theoretical and (possibly) practical – interest. Both the cell and the brain are two masterpieces/miracles of live, especially when considering them as computing machineries; bridging their mathematical–computational approaches could be a great achievement.

Membrane computing has started with a cell-like model, and tissue-like and neural-like models were introduced some years later, when the people working in this area were "busy" with the initial model, hence the new variants have not raised yet the same interest, despite of their generality and versatility. On the other hand, it is rather clear that what are called now "neural-like P systems" are not the most appropriate models, they do not capture the most significant (computationally useful) features of neuronal networks and brain organization.

We do not debate here too much this research topic, as it is quite general, and needs a more detailed biological preparation, but we only mention some possibly useful features to be captured by the model to be defined (these features are considered both having in mind

a "black-box" interpretation of the brain and the existing neural-like P systems, which miss such features). First, it seems necessary to deal with a large number of "neurons" (cells, with or without an internal structure, hence a hierarchical arrangement of membranes), maybe not all of them simultaneously active. (What "non-active" could mean is a matter to define: inhibited because of certain specified conditions – predicates on the contents and/or the environment, or without objects to process, or, in the case of moving rules, without rules able to process the available objects, etc.) Then, of a definite interest would be to have *dynamically defined synapses*; the channels between cells (one-way or two-way channels?) should be established according to the needs of communication, changing from one step to another step. Then, what about having durations of synapses, strenghts, classes of synapses (temporary, permanent, of specified duration)? For instance, in terms of symport/antiport rules, we can consider rules of the form $\langle i, x/y, j\rangle_k$, with the meaning that a synapse is created among cells i and j, when interchanging the multisets x and y among the two cells, and the synapse will last k steps. An existing synapse can be used by any number of standard rules $(i, u/v, j), (i, u, j), (j, u, i)$, but only in the k steps after creating the synapse. For each pair (i, j) a constant $K_{i,j}$ can be given such that, if this channel is used at least $K_{i,j}$ consecutive steps, then the synapse remains established forever. Learning can get in this way a counterpart/meaning. Of interest could be to also have some time delays associated with the communication channels, so that much used synapses will pass more rapidly the "messages" (objects) from a cell to another one, thus speeding-up the computation. Other constants could be associated with the cells (the "neurons") themselves, for instance, expressing the maximal number of objects which can be accommodated by a cell, or the maximal number of rules which can be simultaneously applied in a cell.

These capacity limits look rather realistic, and can probably lead to PNN ("periodically needed nap") theorems of the following type: systems of a given class can avoid deadlocks (or "damages", to be suitably defined, having in mind the fact that the neurons can "die" during the life of an organism, certain "diseases" are possible, etc.) only if from time to time the "externally useful" work is replaced with some "internal cleaning" sequences of steps[1].

Besides the mathematical and computational interest, a more elaborated neural-like membrane computing model would be important also for applications, not only in biology, but, presumably, as a model of distributed computing – the internet included.

4 Links with Ciliates

One of the most intriguing "computations" held in nature is that done by ciliates, during their reproduction. The permutations of genes in macronuclei–micronuclei are exquisite list processing operations of a surprising power and complexity. Convincing details can be found, e.g., in [18] and, especially, in [11]. In particular, several inter- and intra-molecular operations were proposed in the above mentioned papers (see also their references) for explaining the gene unscrambling and assembly in ciliates.

Ciliates are unicellular organisms. Up to now they were interpreted as single membrane structures, which is not exactly the case in reality. What about "marrying" membrane computing and "ciliate computing"? The first suggestion is to use ciliate–inspired operations with strings in handling string–objects in P systems. Which combinations of

[1] Such PNN theorems would be greeted at least by the author of this note...

operations (in what kinds of membrane structures) lead to universality? Then, a question of a possible interest from a biological point of view is to have a look at the ciliate structure, distinguishing the compartments of their cell–body and the specific operations (including the way of communicating among compartments), and to define a suitable model of this structure. The third issue of interest is to use ciliate-like P systems as computing devices, adding parallelism (and/or further ways to get speeding-up features, such as an exponential workspace) in such a way to solve computationally hard problems in a feasible (polynomial) time. This last idea could hopefully be related to the possibility to "implement" ciliate-P-systems in real–ciliates, thus dreaming at using ciliates as living computers[2].

5 Sevilla Carpet

The time and the space are the main computational complexity measures in both sequential and parallel computing – with the communication complexity also becoming important in parallel (distributed) computing. The time–space duality (materialized, for instance, in trade-off results) is a common sense topic, but still raising mathematically interesting and practically important questions. How much can we trade time for space, or conversely? Is there a sort of minimal threshold for their product in a given circumstance (for a given problem)?

In order to address such questions in membrane computing, the so-called *Sevilla carpets* were proposed in [8], as extensions of the Szilard language from language theory: with each computation in a given P system one associates a rectangle, with the time on the horizontal axis, and with the membranes and the rules from each membrane written on the vertical axis; the membranes are ordered in a given way, and also the rules of each membrane are written in a given order; then, the rules used in a step of a computation are indicated in the vertical line corresponding to that time unit. There are several ways of specifying the used rules: writing 0 for "not used" and 1 for "used"; also specifying the number of times a rule is applied in a given step (depending on the multiplicity of objects to which the rule is applied); using, moreover, a dash for rules which cannot be used, 0 for rules which can be used but are not applied in a given step (because of the non-deterministic use of rules), and 1 or a number (of applications) for rules which are used; associating weights to rules, etc. Further details can be found in [8].

Then, the *weight* of a carpet (the sum of all its entries) gives an indication on the time–space complexity of the computation. Moreover, several characteristics of the computations and of P systems can be identified by means of the carpets. We only point out some of them here, in many cases just recalling their formulation from [8].

First, the important issue of the combined space–time complexity can be investigated in this case. Given a system Π, we can look for an equivalent system Π' which computes the same set of numbers, in a more efficient way from the space–time complexity point of view. Are there general speed-up theorems, of the form "if $N(\Pi)$ is infinite, then a linearly faster (as the surface of the associated Sevilla carpets) system Π' can be constructed equivalent with Π"?

Then, as it is pointed out in [8], the carpet can indicate the "degree of non-determinism" of the system (in the case when we distinguish between rules which can

[2]The above ideas are not all and not completely new: the membrane–ciliate combination is the topic of a research project recently started in Leiden University, The Netherlands, while parallelism in ciliate gene unscrambling was already considered in [16].

be applied but are not, and rules which cannot be applied), and this degree can get a numerical value for each computation. Maybe the ratio of the number of entries 0 and the number of all entries different from "−" can be an accurate measure of non-determinism; this is a feature not so easy to measure in any type of non-deterministic computing systems.

Still more attractive can be to use the carpets in an "operational" manner. A first step would be to observe the "bad characteristics" of a system (just one example: if we have components which work only a few steps, then this indicates a non-fair loading of components, and a small degree of coordination/parallelism), and to try to change the system, improving its behavior. Furthermore, we can pass from just collecting the carpets associated with the computations of a system, to using them for controlling the functioning of the system, in the same way as one can pass from the Szilard language associated with a grammar to a control language which regulates the derivations with respect to a grammar. More precisely, we can consider pairs of the form (Π, K), where Π is a P system and K is a language of carpets with the rows associated with the membranes or the rules of Π; only those computations in Π are accepted which have the Sevilla carpet in K. From the computability power point of view, this approach is of interest only for weak (e.g., non-universal) classes of systems Π and for types of carpets K which are easy to describe/generate.

This makes necessary to consider ways to describe/generate the carpets, hence to link our framework with that of two-dimensional languages. Actually, this link is two-fold: on the one hand, we need "simple" carpets for controlling the behavior of P systems (here "simple" could refer, for instance, to the Chomsky hierarchy of array grammars, or to other classifications of two-dimensional languages), on the other hand, we can consider the P systems themselves as generating two-dimensional languages (like in the case of Szilard string languages, which are by-products of usual grammars). For instance, which types of classic picture generating mechanisms can describe the carpets corresponding to a P system of a given type? Of course, the language of carpets is of interest only for the cases where the alphabet of symbols which mark the pixels of the two-dimensional picture is finite, for instance, consisting only of 0 and 1.

We have not considered here the case where the rules themselves have a weight, a cost; a natural possibility is to take the number of symbols handled by a rule as a measure of its cost (for $u \to v$ we take $|uv|$ as the cost of the rule, which reminds the measure *Symb* from the descriptional complexity of context-free grammars). This case provides further information about the time–space complexity of a computation.

In the same way as above, we can define the Sevilla carpets for P systems with symport/antiport rules (the weight of a rule is clear also in this case).

A non-trivial question seems to be the definition of Sevilla carpets for P systems with the possibility to produce new membranes, by membrane division or by membrane creation. If we want to take into consideration all membranes existing at a certain time, then we either have to add new rows to the carpet during the computation, or we have to provide in advance "dummy rows" (with entries of a new type, for example, $\flat$, as a notation for "blank"), where the newly created rows will be accommodated. A compromise could be to consider in the carpet only the *types of membranes*, one membrane for each type, irrespective of how many copies of each membrane exist at a given moment; this idea is supported also by the fact that the types are known in advance, and each membrane of a given type has a set of rules specified in advance; in each step, for each rule, we have to sum-up all applications of that rule in all copies of the corresponding membrane.

Of course, further related questions and research topics can be formulated, but we

conclude by expressing the belief that the Sevilla carpets associated with a P system deserve a careful investigation, both as a source of data about a given system and about its computations, and as a way to define controls on the functioning of a system.

6 Traces Revisited

Defining the result of a computation by taking into account the itinerary of a specified object through the membranes of a P system (with symport/antiport rules) is a rather "exotic" idea, pointing out to a more general problem: how the input/output of a computation can be defined? In Section 2 we have mentioned several possibilities for introducing the number to be recognized by an automaton-like P system. Some variants were considered also for the output: internal, external, traces. A few papers have also taken the tree structure as the data to handle, thus defining the input or the output of computations in terms of trees (possibly, of certain codifications of them).

Anyway, both the general question of considering new ways of defining the input/output of a computation, and the question of further investigating the trace languages remain of interest.

For instance, because each membrane visited by the traveller–object can contribute with at most one symbol to the alphabet of the computed language, an infinite hierarchy on the number of membranes is obtained in a trivial manner: languages on an alphabet with n symbols cannot be generated by systems with less than n membranes. This result is based on the fact that we use only one traveller–object. A change can be obtained if we distinguish several travellers, say, $t_1, t_2, \ldots, t_k$; for each pair (traveller t_i, membrane j) we get a different "event" (the introduction of t_i in membrane j), in total, km events, for m being the number of membranes. In each time unit, a subset of the set of travellers are crossing membranes, hence we can have 2^{km} different "combined events", with which we can associate distinct symbols. Still, the infinite hierarchy on the number of membranes is directly obtained by considering languages over alphabets with larger and larger cardinalities.

A more promising idea is to consider one traveller and to allow the change of labels of membranes. For instance, we can consider a list M of possible labels (hence types of membranes), and rules of the following forms: $(x, in|j)$, $(x, out|j)$ (symport), and $(x, out; y, out|j)$ (antiport), associated with membrane i; after using such a rule, the label of the membrane changes from i to j – and now the rules associated with the membrane with label j are applied. In this way, in each moment the system can have a small number of membranes, with labels from a larger list, hence the itinerary of the traveller across these membranes is described by a string over an alphabet with a number of symbols of the size of the number of labels. Therefore, the hierarchy on the number of possible labels is again proved to be infinite in an easy manner – but with how many membranes present in a given moment in the system?

An interesting question (recently formulated by M. Ionescu – personal communication) is to relate trace languages generated by P systems to so-called *Gauss codes* [17]. These codes are actually the traces of a traveller–point along a closed curve which intersects itself in points of simple intersections; these points are marked and the code is the sequence of the point–names. The similarity with the case of trace languages is obvious – but also the differences are clear. Any connection between the two notions seems both non-trivial and of a mathematical interest.

7 Removing Polarizations

The previously mentioned idea of changing the labels of membranes is very powerful, because the labels can be used as a place to store an information useful for the computation. A convincing example: instead of associating polarizations to membranes, we can allow the change of labels; considering labels of the form (l, p), where l is a "usual" label and $p \in \{+, -, 0\}$ is the polarization, we can avoid using explicit polarizations. This idea was already examined in [3], [2], where the problem of removing polarizations from P systems with active membranes was addressed. Several results were obtained in [3], [2], of the following forms: (i) Systems with polarized membranes can be simulated by systems without polarizations, with the possibility of changing labels of membranes by rules of only one of the types (b) (= introducing objects in a region, (c) (= sending an object out of a membrane, (e) (= dividing a membrane); (ii) Universality can be obtained in many cases of using systems without polarizations and with label changing possibilities; (iii) Hard computationally problems can be solved by systems without membrane polarizations and with label changing possibilities. In each case, several problems have remained open – we refer the reader to the mentioned papers for details. Among the most interesting questions we consider those about solving **NP**-complete problems in polynomial time by means of systems without polarizations and "paying instead a small price". The "price" here refers to using label changing, or division of non-elementary membranes, or using non-deterministic confluent systems, maybe constructed in a semi-uniform manner. Can these additional features be completely avoided? Which is the power and efficiency (which families of numbers are generated and which complexity classes are covered/characterized) by P systems with active membranes, without polarizations, and without label changing?

8 Membrane Creation Revisited

In the same way as cell-like P systems have attracted much more attention than tissue-like and neural-like P systems, the membrane division was much more carefully investigated than membrane creation as a way to obtain tractable solutions to hard problems. However, membrane creation is both mathematically and biologically attractive – and polynomial/linear solutions to **NP**-complete problems were obtained also by means of P systems with symbol–objects and membrane creation. The solutions from [20] are based on confluent systems constructed in a semi-uniform manner. Can these results be improved? What about solving other problems than `SAT` and `HPP` – the two considered in [20]? In general, an immediate research topic is to consider the whole research program carried out for P systems with membrane division and to repeat it also for systems with membrane creation (starting with the formal definition of their syntax and semantics, as done in [23] for systems with membrane division).

A possibly interesting question is suggested by the fact that P systems with membrane creation are not universal (Theorem 7.3.1 in [20]). What further ingredients are to be added in order to get universality?

9 Two Technical Problems

In contrast to the generality level of the previous problems, we pass now to two more precise questions.

The first one refers to the universality of P systems with minimal symport/antiport rules, that is, with rules of the forms $(a, in), (a, out), (a, out; b, in)$, where a, b are symbol–objects. Surprisingly enough, such systems with nine membranes were proven in [4] to be universal. The result was improved first to six membranes, then to five – this last result is from [5], where also the problem was formulated whether or not this bound is optimal. More generally: which is the size of families of numbers computed by P systems with minimal symport/antiport rules with 1, 2, 3, 4 membranes? What about adding some control features on the use of rules, such as promoters or inhibitors (which is the number of membranes which ensures the universality in such a case)?

A related topic concerns the trace languages generated by P systems with minimal symport/antiport rules? (Because the movement of the traveller should be controlled by other objects from the system, and the only possibility now is to exchange the traveller with another object through a minimal antiport rule, it is highly possible that we do not get universality.)

Let us now pass to another unexpected result from membrane computing, the universality of catalytic systems (thus solving the first problem from [20], and having implications on several other open problems from the book), even for the case of using only two catalysts. In the systems used in [13], [12], the same catalyst c can assist several objects to evolve, that is, the number of objects a for which we have rules of the form $ca \to cv$ is arbitrary. An immediate suggestion is to bound this number, that is, to consider a constant H such that $h(c) \leq H$ for all catalyst c, where $h(c) = card\{a \mid ca \to cv \text{ is a rule of the system}\}$. A trade-off is expected between the number of catalysts and the threshold H. Problem: find pairs (H, cat), where cat is the number of catalysts, for which we have universality. Which is the smallest H for which we get universality, by means of systems with an arbitrary number of catalysts?

10 The "Pink-World" Postulate

All previous problems were rather "classic", both mathematically speaking and from the point of view of membrane computing; the one which follows is, instead, an informal suggestion toward a possibly new approach of problem solving in such frameworks as DNA and membrane computing, where we can rely on a large population of "molecules" when looking, in a parallel manner, for the solution. The idea was already formulated in [21], we just recall it here in a brief manner, with some additional proposals (and with a new name...).

The starting observation is that there is a striking difference between classic computer science (complexity theory included), based on (assumed) deterministic functioning of computers, and bio-computations, as carried in DNA computing and as imagined in membrane computing: if we have "enough" molecules in a test tube, then all possible reactions will actually happen. For instance, assume that we have some copies of an object a and two rules, $a \to b$ and $a \to c$, in the same region. Then, some copies of a will evolve by means of the first rule and some others by means of the second rule. All combinations are possible, from "all copies of a go to b" to "all copies of a go to c". In biology and chemistry the range of possibilities is not so large: if we have "enough" copies of a, then "for sure" part of them will become b and "for sure" the other part will become c. What "enough" can mean depends on the circumstances. At our symbolic level, if we have two copies of a we cannot expect that "for sure" one will become b and one c, but starting from, say, some dozens of copies can ensure that both rules are applied.

In our framework, we formulate this optimistic point of view (let us call it *the pink-world postulate*) in the following way: if we have *enough* objects in a membrane, then *all* applicable rules are actually applied in each transition. Thus, the choices due to non-determinism are restricted, if a rule *can* be applied, then it is actually applied – provided that we have enough objects. The problem is to define "enough"... In [21], a "polynomially optimistic" solution was proposed: if each combination of rules has polynomially many chances with respect to the number of combinations, then each combination is applied. For instance, in the case of string-objects, assume that in a given region we can distinguish n combinations of rules which can be applied to the local strings ("combination" here means sets of rules, with the number of times a rule is applied being ignored); then each combination will be applied to at least one string, providing that we have at least $n \cdot p(n)$ copies of the string available, for some polynomial $p(x)$ specific to the system we deal with. A P system having this property is said to be *polynomially reliable*.

The idea from [21] can be extended to *exponentially, linearly, logarithmically reliable*, or even to *constant–reliable* systems, in the natural way, replacing the above polynomial $p(x)$ with a suitable function $f(x)$. For instance, a deterministic system is constant–reliable, with $f(n) = 1$, so of more interest is to assume this property for non-deterministic P systems.

In [21] it was shown how polynomially reliable non-deterministic systems (with string-objects and able to replicate strings of length one) can solve **NP**-complete problems in linear time (this is illustrated for `SAT`).

Some possible ways to address the reliability property in mathematical terms were suggested in [21]: give a fuzzy sets, rough sets, or probabilistic definition of reliability; provide some sufficient conditions for it, a sort of axioms entailing reliability. Of course, of a clear interest would be to investigate the usefulness of the *pink-world postulate* at the theoretical level and its adequacy/limits in practical applications/experiments. Which is the computing power of reliable systems of a specified type? Which are the implications from the computational efficiency point of view? Further investigations in this area are worth pursuing.

11 Final Remarks

As we have mentioned from the very beginning, this note is mainly a provocation for the participants to the second Brainstorming Week on Membrane Computing, Sevilla 2004, both showing that there are numerous research topics in membrane computing which deserve to be investigated, and suggesting to the reader to formulate, circulate, and address such problems. The previous list is just a subjective and quick selection, with rather preliminary formulations, most probably to be adequately changed after considering these problems in a more careful way. Anyway, any reaction of the reader is very much welcome.

References

[1] A. Alhazov, Generating classes of languages by P systems and other devices, *Brainstorming Week on Membrane Computing*, Tarragona, February 2003, TR 26/03, URV, 2003, 18–22.

[2] A. Alhazov, L. Pan, Polarizationless P systems with active membranes, *Grammars*, 7, 1 (2004).

[3] A. Alhazov, L. Pan, Gh. Păun, Trading polarizations for labels in P systems with active membranes, submitted 2003.

[4] F. Bernardini, M. Gheorghe, On the power of minimal symport/antiport, *Pre-proceedings of Workshop on Membrane Computing, WMC2003*, Tarragona, GRLMC Report 28/03, 72–83.

[5] F. Bernardini, A. Păun, Universality of minimal symport/antiport: Five membranes suffice, *Aspects of Molecular Computing. Essays Dedicated to Tom Head on the Occasion of His 70th Birthday* (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), LNCS 2950, Springer-Verlag, Berlin, 2004, 43–54.

[6] C. Calude, Gh. Păun, Bio-steps beyond Turing, *CDMTCS Research Report* 226, Univ. of Auckland, November 2003.

[7] 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.

[8] G. Ciobanu, Gh. Păun, Gh. Ştefănescu, Szilard carpets associated with P systems, *Proc. Brainstorming Week on Membrane Computing* (M. Cavaliere, et al, eds.), Tarragona Univ., TR 26/03, 2003, 135–140.

[9] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[10] E. Csuhaj-Varju, G. Vaszil, P automata or purely communicating accepting P systems, in [22], 219–233.

[11] A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott, G. Rozenberg, *Computations in Living Cells*, Springer-Verlag, Berlin, 2004.

[12] R. Freund, L. Kari, M. Oswald, P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient, submitted, 2003.

[13] R. Freund, M. Oswald, P. Sosik, Reducing the number of catalysts needed in computationally universal P systems without priorities, *Proc. DCFS Workshop*, Budapest, Hungary, 2003, 102–113.

[14] R. Freund, Gh. Păun, On deterministic P systems, submitted, 2003.

[15] P. Frisco, The conformon-P system: A molecular and cell biology-inspired computability model, *Theoretical Computer Sci.*, 2004.

[16] T. Harju, I. Petre, G. Rozenberg, Parallel gene assembly in ciliates, *EMCC Workshop*, Vienna, November 2003.

[17] L. Kari, S. Marcus, Gh. Păun, A. Salomaa, In the prehistory of formal languages: Gauss codes, *Bulletin of the EATCS*, 46 (1992), 124–139.

[18] L.F. Landweber, L. Kari, Universal molecular computation in ciliates, in *Evolution as Computation* (L.F. Landweber, E. Winfree, eds.), Springer-Verlag, Berlin, 2002, 257–274.

[19] Gh. Păun, Computing with membranes (P systems): Twenty six research topics, *CDMTCS Technical Report* 119, University of Auckland, 2000.

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

[21] Gh. Păun, Membrane computing: Some non-standard ideas, in *Aspects of Molecular Computing* (N. Jonoska, Gh. Păun, G. Rozenberg, eds.), LNCS 2950, Springer-Verlag, Berlin, 2004, 322–337.

[22] Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds., *Membrane Computing 2002*, *Lecture Notes in Computer Science* 2597, Springer-Verlag, Berlin, 2002.

[23] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, *Teoria de la complejidad en modelos de computación celular con membranas*, Kronos Editorial, Sevilla, 2002.