# Open problems in Membrane Computing area:
# October 2000

1. Computing with membranes starts from the analogy of processes which take place in the complex structure of a living cell with a computing process. In the style of other branches of Natural Computing, we learn from (alive) nature new computing models and strategies (let us say, paradigms), but it is not clear in this moment whether or not we have to go back to biochemistry for implementing the new computing models (like in DNA Computing) or to the electronic computer (to the general purpose one, or to a specially designed one) for implementation (like in Neural Networks and Genetic Algorithms). This dilemma, in my opinion, is the most fundamental one for membrane computing in this moment.

2. We have also to be prepared for the case that no implementation of P systems will be done (that is, no implementation of a real practical interest), that the attempt will remain forever in info. A sort of a metaquestion appears here: (at least up to now) the kind of problems and, mainly, the kind of proof tools addressed in P systems area are very similar to those in automata and language theory; however, in the basic membrane computing model, we do not deal with strings and languages, but with multisets of atomic objects. It is not the syntax what matters here, but the numerical vectors which characterize the multiplicity of copies of the objects from a given region. Then, P system theory (let me call it so) is a branch of what? Of Formal Multiset Theory, instead of Formal Language Theory? This sounds interesting, especially if we take into account that there are a series of papers devoted to multiset mathematics and manipulation (see, e.g., [i], [ii], [iii] and their references; remark the fact that in some papers one says "bag" instead of "multiset"). Then, P systems can be seen as a part of the generative theory of multisets (with the observation that they are already distributed systems, corresponding to grammar systems in language theory; the simpler, non-distributed case still waits for a systematic study (providing that a mathematical or a "practical" interest for such a study will be identified). Actually, a really huge number of technical problems are in circulation in the P systems area and they already can motivate a Formal Multiset Theory. The whole program of formal language theory can be repeated here, irrespective of the fact whether or not we deal with languages or with multisets: generative/computing power; closure properties (by the way: what corresponds to an Abstract Family of Languages in terms of multisets? how does look an AFM = Abstract Family of Multisets, and the AFM theory?); necessary conditions and counterexamples (such tools are almost completely missing in this moment, but they are urgently necessary in order to settle such problems like finding infinite hierarchies and for separating the power of classes of P systems); decidability (classic – emptiness, finiteness, equivalence, etc – but also specific: is a specified membrane ever

dissolved/divided?); comparison of classes of P systems among them and with other number or language generating devices (maybe taking the equality modulo Parikh images in the case of languages); the power of small systems (first, we have to define "small", hence to consider measures of descriptional complexity); and so on and so forth.I do not persist in this direction, the reader can consult the bibliography of P systems, or can imagine him(her)self such problems.

i) J. P. Banatre, A. Coutant, D. Le Metayer, A parallel machine for multiset transformation and its programming style, Future Generation Computer Systems, 4 (1988), 133–144.

ii) G. Berry, G. Boudol, The chemical abstract machine, Theoretical Computer Sci., 96 (1992), 217–248.

iii) A. Syropoulos, A note on bags, basic pairs and the chemical abstract machine, submitted, 1999.

3. Fundamental to P systems is not only the fact that (in the basic variant) we work with multisets, but also the membrane structure, with all consequences of that, especially the possibility of working in separate regions, arranged in a hierarchical fashion, and to communicate among regions. It is important to emphasize this: the membranes are separators and channels of communication. Having in mind the biological source of inspiration, we can imagine the membranes as physical objects, but this can be a serious limitation when looking for implementations. Any kind of virtual separators which allow (selective) communication among the delimited regions can play the role of membranes. Continuing the previous ideas, computing with membranes should be interpreted in a very general manner, as a framework where distributed processes take place in a membrane structure (a hierarchical arrangement of regions), with an essential role played by communication. Keeping close to the biological source, P systems can be viewed as a general architecture of "living organisms", in the sense of Artificial Life (see, e.g., [i] and [ii]). Whether or not the "life" of a system is to be interpreted as a computation or as a process with another meaning depends on the observer/user. The generality of the approach suggests to consider "incarnations" of P systems in other domains bearing some kind of "life" (in the sense of a development in time), such as logics and formal systems.

i) C. G. Langton, Artificial Life, in vol. Artificial Life, II (C. G. Langton, C. Taylor, J. D. Farmer, S. Rasmussen, eds.), Proc. of the Workshop on Artificial Life, Santa Fe, 1990, Santa Fe Institute Studies in the Science of Complexity, Proc. vol. X, Addison-Wesley, 1990, 1–47.

ii) M. Sipper, Studying Artificial Life using a simple, general cellular model, Artificial Life Journal, 2, 1 (1995), 1–35.

4. Regarding the central role of communication, the following question is of a clear mathematical and epistemological interest: what about the possibility of considering a class of P systems, meant to compute, where no rule for objects evolution appears, but only rules governing object communication from a region to another one. Because no object can be produced inside the system (there is no rule of the form $a \rightarrow bc$, thus increasing the number of occurrences of objects in the system), we have to consider rules for bringing objects from a specified source, for example, from the outer region. This can be done by either considering rules in the outer region of the form $b \rightarrow b(in)$, $c \rightarrow c(in)$, with the meaning that a copy of $b$ and a copy of $c$ are introduced in the skin membrane, or by considering

a new command associated with objects, besides *here, in, out*, for instance come; rules introducing symbols *b(come)* will be used in the skin membrane only, with the obvious meaning, of bringing a copy of *b* from outside the system. Note that even simple rules of the form $a \rightarrow b$ are not allowed, hence we have either to allow changing of objects when passing through a membrane, or to simulate them, for instance, by sending the symbol *a* outside the system and, at the same time, bringing *b* from the outer region. If the rule $a \rightarrow b$ is to be used in a low level region, this does not look at all trivial. The main mathematical difficulty stands in defining the communication among regions in such a way to get non-trivial computing results by using as elegant and as weak rules as possible. Some conditions should be observed in order to perform communication steps, for instance, in the form of predicates depending on the communicated objects and the contents of regions. If these predicated will be powerful, then the system will be powerful, but too powerful predicates will make the things non-interesting. (Maybe register machines can be used in order to prove results about such purely-communicative P systems.)

5. Let us stay close to biology. Several important laws and characteristics of what happens in biology and biochemistry (at the level of living cells) were practically ignored up to now when defining P systems. For instance, a fundamental law of chemistry is the conservation of matter. Rules of the form $a \rightarrow aa$ are creating a copy of *a* from nothing. In info this can be done, in vivo or in vitro it never happens. Can the conservation law be incorporated in the membrane computing area? An easy solution seems to be to consider that the system brings new objects from its environment (from the outer region), as suggested above for systems completely based on communication and without using evolution rules. Still, the problem is not trivial: if a piece of "raw material" *a'* is meant to become *a* together with one more copy of *a*, this means that we need a rule of the form $aa' \rightarrow aa$, which means cooperation. Cooperative systems are non-interestingly powerful. An elegant way to handle "raw materials" remains to be imagined. Of course, it is much easier to handle "matter destroying rules", of the form $a \rightarrow \lambda$: instead of the empty string we can consider a dummy object, #, just denoting "garbage", which never evolves and which, if necessary, is sent out the system. (As usual in nature/life, the difficulty is to create, not to destroy)

6. The trigger of defining P systems was the assertion, found in several places, that the processes taking place in a living cell, involving manipulations of chemical compounds, energy, and information, can be considered as computing processes. Up to now, almost exclusively chemical compound evolution was captured in P systems, by means of object evolution. What about energy? I said "almost exclusively", because energy is implicitly present in the definition of certain ingredients. For instance, dissolving a membrane when using a certain rule, say, $a \rightarrow b(\delta)$, can be considered to correspond to destroying the membrane because the transformation $a \rightarrow b$ is accompanied by producing a large quantity of energy, which breaks the membrane. Similarly, when defining a priority among rules, the interpretation can be again related to energy: rule $a \rightarrow b$ has priority on rule $c \rightarrow d$ because the reaction it models is more active than the reaction modeled by the latter rule; moreover, rule $c \rightarrow d$ cannot be applied at the same step with rule $a \rightarrow b$ just because the energy available was consumed by $a \rightarrow b$ (and not because the two rules compete for the same objects). Thus: what about explicitly introducing the energy in the

3

model? In some sense, this can be easily done: consider a special object, $e$, as denoting a quanta of energy, and use it in the rules of a system as using any other object. For instance, a rule of the form $aee \to b$ means that object $a$ is transformed into object $b$ at the cost of two energy quanta; on contrary, $a \to bee$ means producing two energy quanta during transforming $a$ into $b$. This again brings cooperation into system, of a well-restricted type. A problem can arise about using or not rules of the form $a \to ee$, or even $e \to ab$. Can energy be converted into substance or conversely? How this can be done? (Which kind of E=mcš like formula we have to observe?)

7. The interpretation of the priority relation in the above "strong" sense reminds to us the use of an order relation in ordered grammars from the regulated rewriting area (see [i]): if $A \to u$ has higher priority over $B \to v$, then the rule $B \to v$ is not allowed to be used if the rule $A \to u$ can be used. What about an interpretation of the priority in a "weaker" way: a rule may be applied if there are objects to which no rule of a higher priority can be applied, irrespective of whether or not such rules of a higher priority can be applied to other objects. An example can clarify the difference between the two interpretations. Consider that we have the rules r1: $ab \to cc$ and r2: $b \to d$, with the priority r1>r2, and the multiset represented by *aabbb*. The rule r1 can be applied to the two pairs of objects *ab*. In the strong interpretation of the priority, the rule r2 cannot be applied to the remaining copy of the object *b*, this *b* passes unchanged to the next configuration; consequently, the result is *ccccb*. In the weak interpretation, the rule r2 can be applied at the same step with r1 to the remaining copy of *b*, because r1 cannot use it; the result is *ccccd*. Up to now, only the strong interpretation has been investigated. The weak one only appears by a mistake in an example from the technical report version of [ii], also used in [iii] (see the correction from [iv]). What about P systems using a priority relation in the weak interpretation sense?

i) J. Dassow, Gh. Paun, Regulated Rewriting in Formal Language Theory, Springer-Verlag, Berlin, 1989.
ii) Gh. Paun, Computing with membranes, *Journal of Computer and System Sciences*, 61 (2000), in press, and Turku Center for computer Science-TUCS Report No 208, 1998 (www.tucs.fi).
iii) Gh. Paun, Computing with membranes. An introduction, *Bulletin of the EATCS*, 67 (Febr. 1999), 139–152.
iv) Gh. Paun, Computing with membranes. A correction, two problems, and some bibliographical remarks, *Bulletin of the EATCS*, 68 (1999), 141–144.

8. The conservation law is fundamental, but also the reversibility is basic to biochemistry. Many reactions are reversible, can go in either way (maybe depending on a change of reaction conditions, for instance, temperature – hence energy availability). In dynamic systems, reversibility is somewhat opposed to nondeterminism: we have to uniquely find the previous configuration of the system starting from the present configuration. Strictly chemically speaking, reversibility means reversing the direction of an equation. What about reversible P systems in the dynamic systems sense, what about "local reversible P systems", with the rules allowed to be used in both senses (for each $u \to v$ we have to also add $v \to u$ to the same region)? (By the way, it is known since many years, [i], that reversible Turing machines are computationally universal. Can reversibility be brought

"for free" to P systems area from Turing machines area?)

i) C. Bennett, Logical reversibility of computation, *IBM J. Res. Dev.*, 17 (1973), 525–532.

9. At the border of biochemistry and mathematics we can find many further questions. One of them concerns the determinism. One of the central interesting features of computing with membranes is the inherent nondeterminism of P systems. This corresponds to what happens in cells and test tubes, where the chemical compounds swim in a solution, in a closed space, and evolve nondeterministically according to certain rules. If we attempt to implement membrane computing on the usual computer, then a big problem (= difficulty) appears: we have to simulate nondeterminism on a deterministic machine (still more: we have to simulate parallelism on a sequential machine). How to do this depends on the computer used and on the programming language used, and this is a practical problem. From a mathematical point of view, the problem is to consider deterministic P systems (what this precisely means, for instance, in the case of cooperating systems, is already a question) and to investigate their power. (The determinism versus nondeterminism question has a glorious history in automata theory and in Lindenmayer system theory, so suggestions can be found in these areas.)

10. A somehow related question: what about P systems with the same rules used in all regions? The use of rules associated with regions is motivated by the fact that rules model chemical reactions and these reactions need specific conditions, which can be assumed to be specific to regions. Various regions have different reaction conditions, hence different rules can act in each of them. Still, the chemistry is universal and the regions of the same cell should have a certain degree of uniformity. Which is the power of "uniform P systems", with the same set of rules acting in all regions? In some sense, in P systems with active membranes we have such a case, but membranes themselves appear in the rules. The problem stated above is of interest for P systems without active membranes, in the sense of the paper [i].

i) Gh. Paun, Computing with membranes, *Journal of Computer and System Sciences*, 61 (2000), in press, and *Turku Center for Computer Science-TUCS Report No 208*, 1998 (www.tucs.fi).

11. The previous question concerns, in fact, a normal form of P systems, which is a more general research topic. At least as important as the form of rules is the shape of the membrane structure. In specific circumstances, specific shapes could be more realistic than others, easier to implement than others. The number of different membrane structures grows fast with the number of membranes. In principle, we have to ask whether or not, given a P system with any given membrane structure, we can (effectively) construct an equivalent P system with another membrane structure (supplementary restriction: preserving the number of membranes). Results of this type appear in [i] and [ii]. For systems which are computationally complete, when this result is obtained by using a restricted number of membranes, the form of the system is already well restricted. Still, the question of finding normal forms (especially from the point of view of the shape of the membrane structure) requests further research efforts.

i) Gh. Paun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, submitted, 1999.

ii) I. Petre, A normal form for P systems, *Bulletin of the EATCS*, 67 (Febr. 1999), 165–172.

12. In living cells, the "objects" are moved from a region to another one, through membranes, by making use of the protein channels present in membranes, or because of electrical charges, or randomly, or, mainly, because of different concentrations in neighboring regions. Communication controlled by the concentration of objects was considered only in [i], but its importance in biology motivates a more systematic investigation. Actually, this is related to a more general topic, of defining the processes in a region depending on the contents of that region. For instance, the rules themselves could be chosen according to the objects present in a region. In real cells, division appear also depending on the contents of the cell, not necessarily entailed by a single object (sometimes, the cell divides just because it contains "too much" material, and it is necessary to create a supplementary space). A more extended dependency of evolution on region contents is natural to be considered.
i) J. Dassow, Gh. Paun, Concentration controlled P systems, submitted, 1999.

13. We have already reached the "territory" of mathematical problems, having or not direct biochemical motivations and/or possible practical implications. A very fruitful question, of a rather common place in all computability areas, concerns hierarchies. Is $n+1$ strictly stronger than $n$? In general, it is expected to be so. Many parameters are intimate to the P systems structure, some of them of a classic form (corresponding to descriptional complexity measures in formal language theory, see, e.g., [i]), others are specific to the new devices. The number of catalysts, of membranes, of rules (in total or the maximal number of rules in a region) are of the first type, the depth of the membrane structure (the height of the tree associated with it), its width or "outdegree" (again, defining these notions for the associated tree) are of the second type. When characterizations of the power of Turing machines are obtained by systems of a given size (for instance, as the number of membranes), the hierarchies collapse, but when no such characterizations are known the problem whether or not a given parameter induces an infinite hierarchy remains to be investigated. An important detail: because of the existence of universal Turing machines and of universal type-0 Chomsky grammars (see an explicit construction of such a grammar in [ii]; small universal Turing machines can be found, e.g., in [iii]), a proof that many hierarchies collapse can sometimes be found in a rather easy way. Namely, when a class of P systems is computationally complete and the proof of such a result starts from a Turing machine or a type-0 Chomsky grammar and it constructs an equivalent P system, then, by starting from a universal machine or grammar we get a universal P system, a fixed one, which can generate any given set M of numbers (as usual in P systems area) by changing the objects initially present in the system. The membrane structure and the rules will remain the same, only the objects in the initial configuration will depend on M. In this way, all parameters which deal with membranes and rules will remain unchanged. The size of the universal P system will be an upper bound for all hierarchies on parameters related to membranes and rules (but not on those related to the initial objects present in the system). Finding the smallest value of these parameters, sufficient for obtaining computational completeness, is another question, expected in many cases to be of a serious difficulty. It appears here a really classic question. Many proofs of the

computational completeness of P systems of various types start from matrix grammars with appearance checking (known to characterize the recursively enumerable languages) in the binary normal form (see Lemma 1.3.7 in [iv]). However, no universal grammar of this type is known, so, we cannot get collapsing hierarchies in the way discussed above. Finding a universal matrix grammar, in a sort of binary normal form (with all non-initial matrices containing only two rules) is, therefore, a formal language theory problem of a definite interest for P system theory.

i) J. Gruska, Descriptional complexity of context-free languages, *Proc. Math. Found. Computer Sci. Conf.*, High Tatras, 1973, 71–83.

ii) C. S. Calude, Gh. Paun, Global syntax and semantics for recursively enumerable languages, *Fundamenta Informaticae*, 4, 2 (1981), 245–254.

iii) Y. Rogozhin, Small universal Turing machines, *Theoretical Computer Sci.*, 168 (1996), 215–240

iv) J. Dassow, Gh. Paun, Regulated Rewriting in Formal Language Theory, *Springer-Verlag*, Berlin, 1989.

14. I have several times mentioned the possibility of describing the membrane structure of a P system by means of a tree. This has been already pointed out in [i] and [ii], and considered again in [iii] as a starting point of a generalization of the idea of a membrane structure, to supports of P system-like devices working on graphs different from trees (only planar graphs were considered in [iii]). Here stands a more general question, which is actually a two-fold one: (1) to make use of this mathematically nice description of a membrane structure by a tree and to work on trees in a systematic way, and (2) to consider systematically the idea of working on graphs of more general forms. Of course, the biochemical intuition of a membrane structure is lost when discussing about trees (dissolving a membrane means to remove a node from the tree and to link all direct descendants of the node to the parent node, communicating objects through membranes means to move objects up and down in the tree, etc), but, maybe, tools and suggestions from graph theory will come into the stage.

i) Gh. Paun, Computing with membranes, *Journal of Computer and System Sciences*, 61 (2000), in press, and *Turku Center for Computer Science-TUCS Report No 208*, 1998 (www.tucs.fi).

ii) Gh. Paun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae*, 41, 3 (2000), 259–266.

iii) Gh. Paun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, submitted, 1999.

15. Another large class of problems appears when considering string-objects instead of symbol-objects. This case was already investigated in [i], [ii], [iii], [iv], and recently in [v] and [vi], but a systematic study is still missing. When dealing with strings we need string processing rules; rewriting rules and splicing rules were considered, other types of rules not (such as insertion-deletion or context addition, as common in the area of contextual grammars, see [vii], operations inspired from the genetic area as those considered in [viii], and so on). "Purely biological P systems", involving operations inspired from biology, look of interest (at least aesthetically). What is clear is that string processing plus distributed computing in the sense of P systems is very powerful, "small" and "simple" P systems

are to be expected to characterize the recursively enumerable languages. As usual when inventing a new class of P systems, also when looking for string processing P systems we have to look for systems which are as elegant as possible, using as simple as possible ingredients. A somewhat strange idea concerning the communication of string-objects from a region to another one is the following, leading to a sort of "P systems based on worms processing": when communicating strings from one region to another one, through a membrane, proceed step-by-step (symbol-by-symbol?), in such a way that a string can have a part in one region and another part in another region (maybe, a string can have parts in several regions, more than two); each substring is processed (for instance, rewritten) by the rules in the region where it lies. There are at least two technical questions here: how to define the moving of strings through membranes? how to define the result of a computation?

i) Gh. Paun, Computing with membranes, *Journal of Computer and System Sciences*, 61 (2000), in press, and *Turku Center for Computer Science-TUCS Report No 208*, 1998 (www.tucs.fi).

ii) S. N. Krishna, R. Rama, On the power of P systems with sequential and parallel rewriting, manuscript, 1999.

iii) A. Paun, M. Paun, On the membrane computing based on splicing, submitted, 2000.

iv) Gh. Paun, T. Yokomori, Membrane computing based on splicing, *Preliminary Proc. of Fifth Intern. Meeting on DNA Based Computers* (E. Winfree, D. Gifford, eds.), MIT, June 1999, 213–227.

v) Cl. Zandron, Cl. Ferretti, G. Mauri, Two normal forms for rewriting P systems, manuscript, 2000.

vi) Cl. Zandron, Cl. Ferretti, G. Mauri, Priorities and variable thickness of membranes in rewriting P systems, manuscript, 2000.

vii) Gh. Paun, Marcus Contextual Grammars, Kluwer, Boston, Dordrecht, 1997.

viii) J. Dassow, V. Mitrana, On some operations suggested by genome evolution, *Second Pacific Conf. on Biocomputing*, Hawaii, 1997.

16. A fruitful idea seems to be that of combining symbol-object and string-object P systems. In symbol-object systems we process multisets of symbols and the result of a computation is a number or a vector of natural numbers. In string-object systems we process strings in a way quite similar to language theory (to grammar system theory, because of distribution), without using multisets, and the result of a computation is a language. Let us consider multisets of strings, processed by rewriting, splicing, or by other string operations, but always taking into account the number of copies of each string. (For instance, when a derivation step of the form $x=(x1)A(x2) => x(1)u(x2)=y$ is performed, the number of copies of the string $x$ is decreased by one and the number of copies of the string $y$ is increased by one.) Consider as the result of a computation the number of strings present at the end of the computation in a specified membrane. Of course, because we need to change the number of strings from a step to another one, rewriting operations (or splicing operations which recombine two strings and produce two new strings) are not enough, we need further operations. Some possibilities are suggested by the biology of the cell (for instance, the DNA biochemistry): cut a string into two new strings, duplicate a string, merge two strings into one new string; we can decrease the number of strings

also by sending them out of the system. The study of such systems was already started in [i], but only a few preliminary results were found: computational completeness and the possibility of solving the Hamiltonian Path Problem in quadratic time and the SAT problem in linear time. The used operations were reduplication, splitting, mutation, and crossing-over at given places. These operations have correspondents in the DNA biochemistry and similar operations are also used in [ii]; crossing-over means passing from *(x1)z(x2), (y1)z(y2)* to *(x1)z(y2), (y1)z(x2)*, because of the block *z* (this corresponds to the operation in [iii] and is a restricted case of the splicing). A particular problem in this framework: consider only string-objects of a bounded length. Does the maximal length induce an infinite hierarchy of the number sets computed by systems of a given type?

i) J. Castellanos, Gh. Paun, A. Rodriguez-Paton, Computing with membranes: P systems with worm-objects, *IEEE Conf. SPIRE*, 2000.

ii) M. Sipper, Studying Artificial Life using a simple, general cellular model, *Artificial Life Journal*, 2, 1 (1995), 1–35.

iii) M. P. Schützenberger: On finite monoids having only trivial subgroups. *Inform. Control*, 8 (1965), 190–194

17. Generalizing the passing from symbol-objects to string-objects, an immediate further step is to consider still more complex structures for describing the objects, such as trees, graphs of arbitrary forms, arrays, etc. Such generalizations are classic in language theory (graph grammar area, array grammars, and even picture grammars are well developed domains); also in the DNA Computing were tried such generalizations, for instance, considering the splicing operation for trees and arrays (references can be found in [i]). In relation with the previously mentioned research topic of exporting membrane computing basic ingredients to other domains, it is natural to start by considering usual systems with complex object descriptions. Maybe the complexity of objects (of data structures we use) can compensate for using systems of a simple form, closer to "reality" and "easier to implement".

i) Gh. Paun, G. Rozenberg, A. Salomaa, DNA Computing. New Computing Paradigms, Springer-Verlag, Berlin, 1998.

18. Of a definite interest are the research topics related to "applications" of P systems. A theoretical application, done only in info, is also called application here. This is the case with solving SAT (in [i] and [ii]), the Hamiltonian Path Problem (in [i] and [iii]), and the Node Covering Problem (in [iii]) in linear time, or to break DES (in [iv]). Find further hard (NP-complete) problems which can be "solved" in the membrane computing framework in a polynomial time (in a direct manner, not by a polynomial reduction to the NP-complete problems mentioned above). What about the primality question, can it be decided in a linear time? P systems deal with numbers, so it is expected that number theory questions are good candidates of problems to be addressed in the P area.

i) J. Castellanos, Gh. Paun, A. Rodriguez-Paton, Computing with membranes: P systems with worm-objects, *IEEE Conf. SPIRE*, 2000.

ii) Gh. Paun, P systems with active membranes: Attacking NP complete problems, *J. Automata, Languages and Combinatorics*, in press, and *Auckland University, CDMTCS Report No 102*, 1999 (www.cs.auckland.ac.nz/CDMTCS).

iii) S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-

complete problems, *Romanian J. of Information Science and Technology*, 2, 4 (1999), 357–367.
iv) S. N. Krishna, R. Rama, Computing with P systems, submitted, 2000.

19. When attacking the previous NP-complete problems, one uses P systems with the possibility of dividing membranes. When a membrane is divided, its contents is almost completely replicated in the membranes obtained by division; in one step, does not matter how many objects and lower level membranes exist, all of them are replicated (in the case of [i] in two copies, but in [ii] the number of copies is not bounded). This is a very powerful operation, not only because it enhances the parallelism (an exponential number of membranes can be obtained in a linear number of steps), but also because of this one-step replication of arbitrarily many objects. On the other hand, the initial model, that without membrane division already possesses a great amount of parallelism (somehow, of the type known from Lindenmayer systems). Is this parallelism sufficient in order to solve complex problems in an efficient way? Finding a specific NP-complete problem which can be solved in polynomial time by a P system without membrane division (and using symbol-objects) would be a very important result (the previous formulation suggests that I am a little bit pessimistic about this possibility: the symbols cannot contain "too much" information, as it is the case with membranes and with strings).
i) Gh. Paun, P systems with active membranes: Attacking NP complete problems, *J. Automata, Languages and Combinatorics*, in press, and *Auckland University, CDMTCS Report No 102*, 1999 (www.cs.auckland.ac.nz/CDMTCS).
ii) S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-complete problems, *Romanian J. of Information Science and Technology*, 2, 4 (1999), 357–367.

20. A "local" problem, related to the previous two points. In [i] one deals with the case when the division of a membrane leads to two new membranes, while in [ii] one imposes no bound on the number of the new membranes. Is this latter feature necessary, or any system with, say, *k-division* can be simulated by a system with *2-division*, maybe with a controlled slowdown? The intuition says that this would be the case: $k$ descendant membranes of a given membrane can be obtained by *k-1* divisions into two membranes; all other membranes and all objects not involved in these divisions can be "kept busy" for a number of steps depending on $k$ by "timing rules" of the form $a \rightarrow a_1, a_1 \rightarrow a_2, \ldots, a_r \rightarrow a$. It happens that this intuition is not at all easy to be implemented. One of the main difficulties appears with the division of non-elementary membranes, when no object controls the division and, moreover, we do not know in advance how many copies of a membrane should be produced. The *2-division* systems can be considered as a normal form. A more general question for systems with membrane division is to look also for normal forms concerning other features, starting with the shape of the membrane structure; restricting the value of parameters related to the tree describing a membrane structure might be one of the directions for looking for such normal forms.
i) Gh. Paun, P systems with active membranes: Attacking NP complete problems, *J. Automata, Languages and Combinatorics*, in press, and *Auckland University, CDMTCS Report No 102*, 1999 (www.cs.auckland.ac.nz/CDMTCS).
ii) S. N. Krishna, R. Rama, A variant of P systems with active membranes: Solving NP-

complete problems, *Romanian J. of Information Science and Technology*, 2, 4 (1999), 357–367.

21. In what concerns the handling of membranes and the possibility of increasing their number, of interest can be one more idea: to create new membranes directly under the influence of objects, not starting from a previous membrane. In nature, it happens that membranes emerge in certain circumstances just by the organization of certain chemical compounds (lipid molecules). What about rules of the form $u \to v_i$, which mean that the objects identified by $u$ are transformed into the objects identified by $v$, surrounded by a fresh membrane having the label $i$. Because of the label, we precisely know the rules which can act in the region of this membrane $i$ (we associate in advance a set of evolution rules with each membrane label). This can be a very powerful computing tool: by a rule of the form $a \to aa$, in $n$ steps we get $2^n$ copies of the object $a$; using then the rule $a \to b_i$ we can get $2^n$ copies of the membrane with the label $i$, which is very much similar to the way of producing exponentially many membranes in the case of systems with membrane division.

22. Also with respect to membrane handling, one can consider the possibility of communicating from a region to another one not only separate objects, but also membranes as a whole. For instance, by applying a rule of the form *[ [ ]_j ]_i -> [ ]_i [ ]_j* the membrane with the label *j* is sent out of the membrane with the label *i* (if the latter one is the skin membrane, then membrane j leaves the system). Of course, by moving membrane j outside membrane i, the contents of membrane j is moved as a whole (an arbitrarily large number of objects can be communicated at the same time outside membrane i; this is not at all similar to dissolving membrane i, because the communicated objects are now together in membrane j, which is placed in the membrane directly above membrane i, and both membranes i and j are still present in the system. By using the previous rule in the reverse order (remember the reversibility problem; here we just look for rules which define the introduction of a membrane into another one), we can introduce membrane j, together with its contents, inside the region of membrane i. The computing possibilities open by using such rules, able to modify the very topology of the membrane structure, of "rewriting" it, seem to be very powerful (and intricate from a mathematical point of view).

23. Although this was mentioned several times before, I formulate as an explicit research topic the need of implementing P systems on the electronic computer, to simulate them by programs written in appropriate programming languages. Simulating parallelism and nondeterminism on a sequential deterministic machine can lose all the power and attractiveness of computing with P systems, while solving NP-complete problems in linear time means using an exponentially large space, but still the attempt should be done. The papers [i] and [ii] have already tried this, but for systems without membrane division; moreover, the papers do not include actual programs (maybe the authors have such working programs and they have not distributed them). Even if in this way we do not get an implementation of P systems in silico, we can get tools for related problems, such as simulating the life of a system, in the sense of Artificial Life. (For systems with only one membrane and with the number of possible objects bounded, this was already done – see

[iii] and its references.)

i) M. Malita, Membrane computing in Prolog, manuscript, 1999.

ii) Y. Suzuki, H. Tanaka, On a LISP implementation of a class of P systems, *Romanian J. of Information Science and Technology*, 3, 2 (2000).

iii) Y. Suzuki, J. Takabayashi, H. Tanaka, Investigation of an ecological system by using an abstract rewriting system on multisets, *in Recent Topics in Mathematical and Computational Linguistics* (Gh. Paun, ed.), The Publ. House of the Romanian Academy, Bucharest, 2000, 300–309.

24. Now, I come to an important point, the trade-off between programmability, efficiency, and adaptability/learnability, as pointed out by M. Conrad in several papers, see, e.g., [i] and [ii]. Looking for computing devices equal in power to Turing machines and also having universality properties (in the sense of the existence of universal devices in the considered class, which makes possible the programmability) is natural and seems desirable from a "classic" (= mathematical) point of view, but bio-computing seems to make possible an old dream of computer science, adaptable computers, which can "learn" both at the level of the hardware and at the level of the software. Even if the price of adaptability is the loss of programmability, maybe also of efficiency, it is highly possible that classes of problems exist for which it is preferable to have computers with an evolving hardware, adapted to the problem they have to solve. What this could mean in the membrane computing area is not at all clear, but the biological origin of P systems supports the question of investigating this topic. Suggestions from areas of Natural Computing where evolution and adaptation/learning are already central issues, like Neural Networks and Evolutionary Computing, will probably be useful.

i) M. Conrad, Information processing in molecular systems, *Currents in Modern Biology*, 5 (1972), 1–14.

ii) M. Conrad, The price of programmability, The Universal Turing Machine: A Half-Century Survey (R. Herken, ed.), Kammerer and Unverzagt, Hamburg, 1988, 285–307.

25. In general, a more systematic osmosis with other areas of Natural Computing or with other approaches to distributed computing (such as systolic systems, actor systems, etc) is of interest and, expectedly, fruitful for both sides: ideas from other domains can be added (reformulated) to membrane computing area, while ideas from membrane computing area can (hopefully) be useful to other fields. P system theory has already imported many ingredients from grammar system theory (see, e.g., [i]) and from the DNA Computing area (see, e.g., [ii]). A close correspondence with Cardelli's ambient calculus (see, [iii], [iv]) was found in [v].

i) E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Paun, Grammar Systems. A Grammatical Approach to Distribution and Cooperation, Gordon and Breach, London, 1994.

ii) Gh. Paun, G. Rozenberg, A. Salomaa, DNA Computing. New Computing Paradigms, Springer-Verlag, Berlin, 1998.

iii) L. Cardelli, Abstractions for mobile computation, in Secure Internet Programming (J. Vitek, Ch. Jensen, eds.), *Lecture Notes in Computer Science*, 1603, Springer-Verlag, 1999.

iv) L. Cardelli, A. Gordon, Mobile ambients, *in Proceedings of FoSSaCS'98 (M. Nivat, ed.), Lecture Notes in Computer Science*, 1378, Springer-Verlag, 1998, 140–155.

v) I. Petre, L. Petre, Mobile ambients and P systems, *Workshop on Formal Languages, FCT99*, Iasi, 1999, *J. Universal Computer Sci.*, 5, 9 (1999), 588–598.

26. Membrane Computing comes from biology and in biology and biochemistry the processes are nondeterministic, the result is only approximately/probabilistically true. Up to now, only "crisp" mathematics was used in P system area (the same is in a great extent true also for DNA Computing). What about "approximate" mathematical approaches, using probabilities, fuzzy sets, or rough sets theory? (For the latter one, see [i]; the basic idea is to approximate a set from the interior and from the exterior, by converging unions of equivalence – or, more general, tolerance – classes.) What about "approximate" computing, whatever this can mean? All these can be reformulated in terms of soft computing, to which, in my opinion, Molecular Computing (DNA and membranes included) belongs.

    i) Z. Pawlak, Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer, Dordrecht, 1992.