Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A dozen of research topics in membrane computing

Gheorghe Păun [a,b,*]

[a] Institute of Mathematics of the Romanian Academy, PO Box 1-764, 014700 Bucureşti, Romania
[b] Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

ARTICLE INFO

Article history:
Received 20 December 2017
Accepted 6 April 2018
Available online xxxx

Keywords:
Natural computing
Membrane computing
Turing computability
Computational complexity

ABSTRACT

This note considers three basic research directions in membrane computing – characterizations of the computing power of Turing machines, computing more than Turing machines, efficiency (solving computationally hard problems in a feasible time) – by basic classes of P systems (cell and tissue multiset rewriting systems, symport/antiport systems, spiking neural P systems, numerical P systems). (Types of) Results reported in the literature are briefly mentioned, several unsolved cases are pointed out, and directions of further research are proposed.

1. Introduction: A synthesis table

Let us start abruptly with the table below, a "map" (and summary) of the whole discussion.

	Power RE		Power $> RE$		Efficiency			
Class	CS	λ	acc	others	2^xspace	precomp	intrinsic	what else?
cell/tissue multiset rew	OK	OK	OK (mcre)	Q5	div/sep	Q7	Q9 (p,i,we)	Q11
S/A	OK	OK	Q2	Q5	div/sep	Q7	Q9 (p,i,we)	Q11
SN P	OK	OK	Q3	Q5	div/bud	OK	Q9 (E)	Q11
numerical	Q1 (expl)		Q4	Q5	Q6 (+,-)	Q8 (+,-)	OK (expl)	Q11 (+,-)
others	Q12	Q12	Q12	Q12	Q12	Q12	Q12	Q12

Here is a **dictionary of abbreviations:** CS = context-sensitivity features, λ = erasing possibilities, acc = acceleration, 2^x space = exponential space (used in a time–space trade-off), precomp = using pre-computed resources, mcre = membrane creation, div = membrane division, sep = membrane separation, p = promoters, i = inhibitors, we = what else?, bud = budding, E = important role of the regular expression in rules of SN P systems, expl = explanations needed, $+, -$ = using only these operations.

* Correspondence to: Institute of Mathematics of the Romanian Academy, PO Box 1-764, 014700 Bucureşti, Romania.
E-mail addresses: gpaun@us.es, curteadelaarges@gmail.com.

https://doi.org/10.1016/j.tcs.2018.04.008

2. A dozen of problems

"Only" one dozen, in order to get a nice title... otherwise the list can be much longer. Moreover, we do not have well formulated, crisp open problems, but, in general, research directions which need to be explored, starting with the technical formulation of ideas and of basic open problems.

2.1. Equivalence with Turing machines

The column "Power RE" refers to characterizations of the Turing computing power. In formal language theory it is well-known the fact that "context-sensitivity plus erasing ensure universality" (for instance, each RE language is the projection of a context-sensitive language). That is why two sub-columns are mentioned, marked with CS and λ.

In "basic" membrane computing, that is, dealing with multiset processing in cell-like, tissue-like or spiking neural P systems, we have a large number of universality results (strictly speaking, computational completeness results). We have mentioned explicitly symport/antiport systems, because, for other issues, this class should be considered separately. For all these types of P systems, context-sensitivity and erasing are provided "for free" by the definition (by the biochemistry of the cell – erasing can be obtained by sending objects into the environment or by making them unable to evolve/react further).

However, also the numerical P systems are known to be universal, and here the *explanation* should be different, as we do not deal with multiset rewriting. This is the first question, Q1 in the table: where lies the power of numerical P systems (with polynomials with integer coefficients in the programs) and how this depends on the production functions (enzymatic/non-enzymatic, using only some of the four operations $+, -, \times, \div$, etc.) Note the crucial importance of operations from the efficiency point of view, see [6].

2.2. Passing beyond "The Turing barrier"

The second column has the header "Power $> RE$", with the meaning that we deal here with *hypercomputing* ideas and results (in spite of the fact that Martin Davis considers hypercomputing a myth, see, e.g., [3]). Two sub-columns were indicated: *acc* and *others*. The first one refers to the idea of *acceleration*, explored in [2] as a way to "compute the uncomputable" in terms of cell-like P systems endowed with rules for *membrane creation*.

Ways to accelerate other classes of P systems were not proposed and examined yet, and the cases of symport/antiport, spiking neural and numerical P systems are associated with questions Q2, Q3, Q4, respectively.

Problem Q5 is a generic notation for the general question of exploring for P systems other ideas known in the hypercomputation area – some surveys of such ideas can be found, e.g., in [9], [10], [14].

Actually, besides acceleration, another idea was also explored in membrane computing as a way to get hypercomputations, namely *evolutionary lineages of P systems*, see [13] and its references; I would include also this idea in "others".

In particular, it would be interesting/intriguing to prove hypercomputing results for numerical P systems, without using "tricks" (a term of Martin Davis), for instance, without using real numbers in the functions involved in the systems.

2.3. Efficiency

Basically, this is about the possibility of solving computationally hard problems, typically, **NP**-complete problems, in a feasible time, typically, in a polynomial time. Plenty of results of this type are known in membrane computing, most of them based on trading-off time for space: a way to create an exponential working space during the computation is considered, and this space is then used, in a massively parallel manner, to solve (decidability, but also numerical) problems in a polynomial time. Two ideas were mainly used: dividing membranes (we include here also separation of membranes and budding of neurons, creating membranes, even the replication of string objects) and using pre-computed resources. The latter idea was considered only for spiking neural P systems – what about other classes of P systems (problem Q7 in the table)?

These two ideas were not considered yet for numerical P systems (hence the problems Q6 and Q8), with an important detail: these systems were proved already to be efficient, in the case of using all the four operations $+, -, \times, \div$, [6], for enzymatic numerical P systems, even without using a space–time trade-off. This is really interesting: numerical P systems have an *intrinsic* efficiency, they do not need to use an exponential working space. Which is the explanation of this fact, where this intrinsic efficiency is placed? The answer seems to be immediate: in each step, one performs complex computations: evaluating production functions, computing the parts of the current "production" which has to be distributed, the distribution itself to the target variables. All these are done "in no time", without any cost.

Then, of course, questions Q6 and Q8 should be adequately formulated for classes of numerical P systems which are not covered by the results in [6]: non-enzymatic, using only a subset of the four operations, etc.

The intrinsic efficiency of numerical P systems suggests a fruitful research direction: do also other classes of P systems possess a similar intrinsic efficiency? Otherwise formulated: do they perform complex computations during the steps which are, by definition, supposed to last only one time unit, irrespective how complex they are? If this is not the case, can we find such features and "hide" them in the definition in order to get efficiency without using the basic "trick" of trading-off

Please cite this article in press as: G. Păun, A dozen of research topics in membrane computing, Theoret. Comput. Sci. (2018), https://doi.org/10.1016/j.tcs.2018.04.008