Four (Somewhat Nonstandard) Research Topics

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
gpaun@us.es, curteadelaarges@gmail.com

Summary. Four research directions are suggested, dealing with the following four main ideas: computing along the axon (up to now, this topic was only preliminarily investigated), using pre-computed resources in order to solve computationally hard problems, considering in P systems both objects "of matter" and "of anti-matter" (which annihilate each other when meet), and considering the distance (naturally defined in a membrane structure of a given type) as a support of information.

1 Introduction

Membrane computing is more than fifteen years old, [6], and it is already *a well established branch of natural computing* (it is no longer appropriate to call it "a young branch of NC..."), but, in spite of its large bibliography, there are a lot of open problems and research topics in circulation. It is enough to mention the "mega-paper" [2].

Still, further problems and research ideas can be imagined – four classes are suggested here.

Of course, the reader is assumed to be familiar with membrane computing, so that the presentation is minimal, both in what it concerns the details and the references.

2 Computing Along the Axon

The motivation is related to the fact that the axons are not simple "wires" among neurons, but they carry computations, moving spikes back and forth, amplifying them (in the so-called Ranvier nodes) etc. Starting from this, *axon P systems* were introduced in [1] (and then investigated in [8]). The idea is suggested in Figure 1.

We have [Ranvier] nodes arranged along an axon. Initially, each of them can contain a number of spikes. A rule $E/a^c \to (a^l, a^r)$ is applied as in an SN P system;

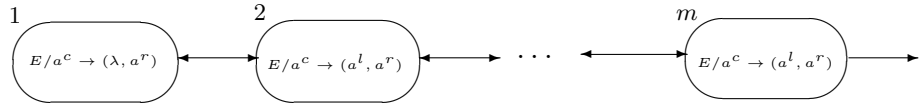

Fig. 1. An axon P system

c spikes are consumed, l spikes are sent to the left node, and r spikes are sent to the right node. The rightmost node is the output one.

The functioning is like that of an SN P system (with a fixed synapses graph and specific spiking rules).

In [1] the axon P systems were used as string/sequence generators: if the output node sends out k spikes, then the symbol b_k is associated with that step of the computation; no spikes means the symbol b_0. In the halting case, only recursive languages can be generated.

Several problems to investigate in this framework are natural (part of them also formulated in the cited papers):

1. generate numbers, by considering the number of steps elapsed between the first two spiking steps (like in SN P systems, halting or not);
2. in the case of strings, look for variants which are computationally universal (what about ignoring the steps when no spike exits, hence taking $b_0 = \lambda$?);
3. consider axon P systems as infinite sequence generators;
4. look for hierarchies (on the number of axons) in the non-universal case;
5. what about also using anti-spikes, like in SN P systems, [5]? (besides the object a, also $\bar{a}$ is considered, called *anti-spike*; spiking rules can consume and/or create either spikes or anti-spikes, but a node/neuron cannot contain at the same time both spikes and anti-spikes, they immediately annihilate themselves, by means of an implicit rule $a\bar{a} \to \lambda$, which is used in no time, in a maximal way).

The axon P systems can be of interest also as a support for the last problem suggested here, i.e., using the distance (between two nodes) as a support of information.

3 Pre-Computed Resources

The idea has been explored in a couple of papers (see, e.g., [3], [4]), but still there is no formal definition of the complexity classes which appear in this context (the problem is, in fact, to define "acceptable" pre-computed resources; intuitively, the idea is to have a pre-computed arbitrarily large support for computation of the form uv^ω, with information only in u; how to pass from this string intuition to SN P systems – or to other types of P systems – remains open; in [7], page 360, it is suggested that the concept of *uniformity* from Circuit Complexity can be useful in this respect).

Besides recalling the attention of the previous problem, I would like to point out here another question which looks of (practical) interest: is it possible to find a pre-computed support of computation which can be used for solving two/several/all problems from a given class? This would be interesting and important: "constructing the computer", whatever large it is, is done only once, then several problems can be solved using the same pre-computed "hardware", where specific information is plug-in.

4 Using Matter and Anti-matter

This is a direct generalization of the idea of anti-spikes from SN P systems, [5]: for each object a, in any type of P systems, to consider an "anti-object" $\bar{a}$. Both objects and anti-objects are handled by usual evolution rules, but matter and anti-matter cannot stay together in a compartment, a rule of the form $a\bar{a} \to \lambda$ is supposed to exist in any place. This rule is applied immediately, in no time, so that in each compartment, in the end of each evolution step either only objects or only anti-objects are present.

Which is the effect, in various classes of P systems, of using anti-objects? In SN P systems, anti-spikes help, so it is expected that this happens also in other cases (and this is one further illustration of the power of annihilation rules $a\bar{a} \to \lambda$, known to be useful in formal language theory, and possibly also in the case of multiset rewriting).

In particular: are one catalytic (purely two catalytic) P systems with anti-objects universal?

The following trivial proof of the universality of P systems with two catalysts and with anti-objects supports the conjecture that the answer to the previous question might be positive.

Formally, we can write $NOP_1(2cat, antim) = NRE$, with the obvious meaning.

Indeed, let us consider a 3 register machine, with register 1 (the one where the result is obtained) never decremented, and with registers 2, 3 empty in the end of a computation. Denote it by $M = (3, H, l_0, l_h, I)$ and construct a one-membrane catalytic P system with two catalysts, c_2, c_3. Initially, the system contains the objects l_0, c_2, c_3 in its membrane. The contents of register r is represented by the number of copies of an object a_r, $r = 1, 2, 3$, in the system. For $r = 2, 3$ we also consider the anti-objects $\bar{a}_r$.

For each instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$ in $I, r = 1, 2, 3$, we consider the rules

$$l_i \to l_j a_r d_2 d_3,$$
$$l_i \to l_k a_r d_2 d_3.$$

For each instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$ in $I, r = 2, 3$, we consider the rules

$$l_i \to l_j \bar{a_r} d_2 d_3,$$
$$\bar{a_r} \to \#,$$

$$l_i \to l_k d_{5-r},$$
$$c_r d_r \to c_r,$$
$$c_r a_r \to c_r \#.$$

We also add the rules

$$\# \to \#,$$
$$l_h \to \lambda.$$

The rule $l_i \to l_j \bar{a}_r d_2 d_3$ is used when register r is non-empty; the decrement by 1 is done by means of the implicit rule $a_r \bar{a}_r \to \lambda$. If this rule cannot be applied, then the trap object is introduced, by means of $\bar{a}_r \to \#$. If the register is empty, then the rule $l_i \to l_k d_{5-r}$ should be used.

Note the role of objects d_2, d_3, which "keep busy" the catalysts, not to act in rules of the form $c_r a_r \to c_r \#$. For instance, if the rule $l_i \to l_k d_{5-r}$ is used, but register r is not empty, then, because object d_r is not introduced, but only d_{5-r}, in the next step, the rule $c_r a_r \to c_r \#$ should be used, and the computation never stops.

When the computation in M halts, the object l_h is removed. The number of objects a_1 in the system equals the number computed by M. (The catalysts should be ignored, or the results can be read outside the system, sending a_1 out, etc.)

5 Using the Distance to Encode Numbers

The idea is again suggested by the spiking neurons. If we watch the movie from `http://www.igi.tugraz.at/tnatschl/spike_trains_eng.html`, we see that the distance between various spikes moving along an axon is relevant for the process/computation. How to make this explicit in a P system? Of course, in order to can handle arbitrarily large numbers, we need a membrane structure of an arbitrarily large size, hence we need ways for generating space (membranes in a cell-like or tissue-like P system, neurons in an SN P system, nodes in an axon P system), or we have to deal, again, with pre-computed resources.

Define distance-based P systems, and investigate their power and efficiency. Can universality be reached when using only a bounded number of objects? (I expect a positive answer.) Which are the shapes of the membrane structure necessary/sufficient for computing various classes of numbers, or for the conjectured universality.

6 Final Remarks

Some of the previous questions are more precise, others are less precise. Maybe some of them are trivial. Anyway, a brainstorming is the right place to discuss such issues – my big regret not to be there. I would be very much indebted to the reader present in Sevilla for any reaction.

References

1. H. Chen, T.-O. Ishdorj, and Gh. Păun: Computing along the axon. *Proc. 4th BWMC*, 2006, vol. I, 225-240, and *Progress in Natural Science*, 17(4) (2007), 417-423.
2. M. Gheorghe, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Frontiers of membrane computing: Open problems and research topics, *Intern. J. Found. Computer Sci.*, 2013 (first version in *Proc. Tenth Brainstorming Week on Membrane Computing*, Sevilla, January 30 – February 3, 2012, vol. I, 171–249).
3. T.-O. Ishdorj, A. Leporati: Uniform solutions to `SAT` and `3-SAT` by spiking neural P systems with pre-computed resources. *Natural Computing*, 7 (2008), 519–534.
4. A. Leporati, M.A. Gutiérrez-Naranjo: Solving `SUBSET SUM` by spiking neural P systems with pre-computed resources. *Fundamenta Informaticae*, 87 (2008), 61–77.
5. L. Pan, Gh. Păun: Spiking neural P systems with anti-spikes. *Intern. J. Computers, Comm. Control*, 4, 3 (2009), 273–282.
6. Gh. Păun: Computing with membranes. *J. Comput. Syst. Sci.*, 61 (2000), 108–143 (see also TUCS Report 208, November 1998, `www.tucs.fi`).
7. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
8. X. Zhang, J. Wang, and L. Pan: A note on the generative power of axon p systems. *International Journal of Computers, Communications, and Control*, 4 (2009), 92-98.
9. The P Systems Website: `http://ppage.psystems.eu`.