The Journal of Logic and Algebraic Programming 79 (2010)291–294

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

A quick introduction to membrane computing

Gheorghe Păun

Institute of Mathematics of the Romanian Academy, P.O. Box 1-764, 014700 București, Romania
Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

ARTICLE INFO

Article history:
Available online 6 May 2010

Keywords:
Natural computing
Membrane computing
P system
Turing computability

ABSTRACT

Membrane computing is a branch of natural computing inspired from the architecture and the functioning of biological cells. The obtained computing models are distributed parallel devices, called *P systems*, processing multisets of objects in the compartments defined by hierarchical or more general arrangements of membranes. Many classes of P systems were investigated – mainly from the point of view of computing power and computing efficiency; also, a series of applications (especially in modeling biological processes) were reported. This note is a short and informal introduction to this research area, introducing a few basic notions, research topics, types of results, and pointing out to some relevant references.

1. Introduction

Membrane computing is a branch of natural computing which abstracts computing models from the architecture and the functioning of living cells, as well as from the organization of cells in tissues, organs (brain included) or other higher order structures such as colonies of cells (e.g., of bacteria). The initial goal was to learn something useful (or, at least, interesting) for (theoretical) computer science from biology, following the encouraging examples of, e.g., evolutionary, neural, and DNA computing, and the approach proved to be successful – also providing a systematic framework for devising (discrete, modular, algorithmic) models for various biological processes.

Membrane computing was initiated in 1998, with the final version of the paper first circulated as a research report being published in 2000, [14], and the literature of this area has grown very fast (already in 2003, Thompson Institute for Scientific Information, ISI, has qualified the initial paper as “fast breaking" and the domain as “emergent research front in computer science" – see `http://esi-topics.com`). A comprehensive presentation at the level of year 2002 can be found in the monograph [16] and a recent coverage of the domain can be found in [19]. Details, in particular, many downloadable papers, can be found at the area website from [22].

The first models of membrane computing were starting from the single cell and its organization as a hierarchical structure of compartments, defined by membranes, where a localized biochemistry takes place. Thus, the obtained computing device was a distributed parallel model, with multisets of objects (“chemicals") placed in regions (nodes of a tree) and processed by “reactions" of a biochemical type. The model was extended in various ways, following biological suggestions (e.g., processing the objects by means of other types of operations, such as the symport and antiport operations known in biology), or with mathematical or computational motivations (e.g., passing from single cells to populations of cells, hence from tree arrangements of membranes to arbitrary graphs), trying to cover other biological areas (such as the brain one), etc.

The obtained computing devices proved to be rather powerful, equivalent with Turing machines even when using restricted combinations of features, and also computationally efficient (in certain cases, able to solve computationally hard problems, typically, **NP**-complete problems, in a feasible/polynomial time). Then, a number of applications were reported in

E-mail address: gpaun@us.es

1567-8326/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2010.04.002

several areas – biology, bio-medicine, linguistics, computer graphics, economics, approximate optimization, cryptography, etc. Several software products for simulating P systems and attempts of implementing P systems on a dedicated hardware were reported.

The present text is only a quick introduction to membrane computing, only mentioning some basic ideas, types of results and of applications, and indicating some references where details and further references can be found.

2. Basic classes of P systems

The main ingredients of a P system are (i) *the membrane structure*, delimiting compartments where (ii) *multisets of objects* evolve according to (iii) *(reaction) rules* of a biochemical inspiration. The rules can process both objects and membranes. Thus, membrane computing can be defined as a framework for devising cell-like or tissue-like computing models which process multisets in compartments defined by means of membranes. These models are (in general) distributed and parallel. The objects and the membranes are processed by the rules, thus changing the *configuration* of the system; a sequence of such changes (*transitions*) forms a *computation*, and with a *halting* computation a *result* can be associated in various ways. When a P system is considered as a computing device, hence it is investigated in terms of (theoretical) computer science, the main issues studied concern the *computing power* (in comparison with standard models from computability theory, especially Turing machines/Chomsky grammars and their restrictions) and the *computing efficiency* (the possibility of using parallelism for solving computationally hard problems in a feasible time). Computationally and mathematically oriented ways of using the rules and of defining the result of a computation are considered in this case. For instance, the main way of evolving a P system is based on a non-deterministic maximally parallel use of rules, with the system being synchronized (the clock is universal and in each time unit one uses a maximal multiset of rules in each membrane). Many variants are possible: unsynchronized, with various types of parallelism different from the maximal one (local, bounded, minimal, etc.), deterministic, with local halting, etc. When a P system is constructed as a model of a bio-chemical process, it is examined in terms of dynamical systems, with the (deterministic or probabilistic) evolution in time being the issue of interest, rather than a specific output.

At this moment, there are three main types of P systems: (i) cell-like P systems, (ii) tissue-like P systems, and (iii) neural-like P systems.

The first type imitates the (eukaryotic) cell, and its basic ingredient is the *membrane structure*, which is a hierarchical arrangement of membranes (understood as three dimensional vesicles), i.e., delimiting compartments where multisets of objects are placed (there also are models where the objects are placed on membranes, like many proteins in the case of the biological cell); the objects are in general described by symbols from a given alphabet; rules for evolving these objects are provided, also localized, acting in specified compartments or on specified membranes. The most common types of rules are multiset rewriting rules (similar to chemical reactions, i.e., of the type $u \to v$, where u and v are multisets of objects) and transport rules, e.g., symport or antiport rules [13], inspired by biological processes (symport rules are of the form (u, in) or (u, out), moving objects of multiset u through a membrane, and antiport rules are of the form $(u, out; v, in)$, moving the objects of multiset u outside a membrane at the same time with moving the objects of multiset v inside). Also the objects produced by a transition rule can pass through membranes (we say that they are "communicated" among compartments), under the control of target indications associated with the objects from v, the multiset of objects produced by the rule $u \to v$. The rules can have several other forms, and their use can be controlled in various ways: promoters, inhibitors, priorities, etc. Also the hierarchy of membranes can evolve, e.g., by creating and destroying membranes, by division, by bio-like operations of exocytosis, endocytosis, phagocytosis, as in [3], and so on. (The "computer" itself evolves during the computation.) Moreover, the objects can be of various types, not only described by letters from a given alphabet, as in the basic class of P systems. For instance, the objects can be described by strings, and then they evolve by string processing rules (for instance, rewriting, splicing, insertion–deletion), they can be pairs name-value, called *conformons* [6], or even more complex data structures (arrays).

In tissue-like P systems [10], several one-membrane cells are considered as evolving in a common environment. They contain multisets of objects, while also the environment contains objects. Certain cells can communicate directly (channels are provided between them) and all cells can communicate through the environment. The channels can be given in advance or they can be dynamically established – this latter case appears in so-called *population P systems* [2]. In the case when the cells are simple, of a limited capacity (as the number of objects they contain or of rules they can use), we obtain the notion of a *P colony*.

Finally, there are two types of neural-like P systems. One is similar to tissue-like P systems: the cells (neurons) are placed in the nodes of an arbitrary graph and they contain multisets of objects, but they also have a *state* which controls the evolution. Another variant was introduced in [9], under the name of *spiking neural P systems*, where one uses only one type of objects, the *spike*, and the main information one works with is the distance between consecutive spikes (for instance, a number is encoded in the number of time units between two consecutive spikes entering a system and the number is recognized if the device eventually halts).

All these types of P systems can be used in the generative mode (starting from the initial configuration, one proceeds until reaching a halting configuration, when a result is provided – in general, as the number of objects in a given membrane), or in the accepting mode (for instance, a number is introduced in a membrane, in the form of the multiplicity of a given object, and this number is accepted if the computation halts) – the latter case leads to the notion of a *P automaton*. Extensions to

vectors of numbers and to strings and languages were also investigated. When also an input and an output are considered, we have a computing P system.

3. Power and efficiency

From a theoretical point of view, P systems are both *powerful* (most classes of P systems are Turing complete, even when using ingredients of a reduced complexity – a small number of membranes, rules of simple forms, ways of controlling the use of rules directly inspired from biology are sufficient for generating/accepting all sets of numbers or languages generated by Turing machines; also small universal P systems of various types were produced, i.e., given systems which can simulate any particular system from a specified class, after introducing a code of the particular system in the universal one) and *efficient* (many classes of P systems, especially those with enhanced parallelism, can solve computationally hard problems – typically **NP**-complete problems, but also harder problems, e.g., **PSPACE**-complete problems – in a feasible time – typically polynomial). Such a speed-up is obtained by trading space for time, with the space grown exponentially in a linear time by means of bio-inspired operations. The most investigated way to obtain such an exponential workspace is membrane division [15], but also membrane creation, membrane separation, string replication and other operations were used. These operations were then extended to cells in a tissue-like P system and even to neurons in a spiking neural P system – and in all cases similar results were obtained: polynomial time solutions to computationally hard problems. Details can be found, e.g., in [20]. Most complexity investigations concern time as the main parameter, but also space complexity was developed, [21], while recently [**?**] also a distributed way to solve a problem was proposed, thus making possible the development of a theory of communication complexity, in the style of [8] (several cell-like P systems linked by communication channels as in tissue-like P systems receive as inputs parts – distributed in a balanced way – of a problem and they cooperate in building the answer by internal work and by communication). Naturally, both in investigations related to the computing power and in any other case when a P system is to be constructed (e.g., when constructing universal P systems of various types), a constant concern is the size of the system in the sense of descriptional complexity already classic in language and automata theory, see, e.g., [7]. A possible way to combine the time complexity (number of steps) with the "effort done in each step" (e.g., the number of rules used in parallel) was proposed in [5], under the name of *Sevilla carpet*.

Details about all these notions and results about them can be found in the references mentioned below, especially in the handbook [19] and at [22].

4. Applications

As a modeling framework, membrane computing is rather adequate for handling discrete (especially biological) processes, having many features which are attractive from this point of view: easy understandability, scalability and programmability, inherent compartmentalization and ability to handle discrete data, etc. Most applications use cell-like P systems and tissue-like P systems, and the general protocol is the following: a P system is written which models a given (biological – but not only: also economic processes, evolution of ecosystems, and of other processes were addressed in this framework) process, capturing the objects, compartments, and evolution rules, then a program is written to simulate this P system, or a program available on internet is used, after that computer simulations are performed, tuning parameters until correctly describing the real process; related processes are then investigated within this framework. Details can be found in [4], including case studies and a description of existing software products (at the level of 2005), and at [22]. Besides programs to simulate P systems on the existing computers, sometimes on grids, clusters, networks of computers (such programs are currently used in applications), there are several attempts to implement P systems on a dedicated hardware; we mention here only [11], where further references can be found.

There also are applications of other types, e.g., in computer graphics, cryptography, approximate optimization (this last direction of research was initiated in [12] and continued by many researchers and the results are rather encouraging; basically, we have a distributed evolutionary computing approach, with the distribution and the computation organized like in a P system), and so on.

Comprehensive and up-dated information (at the level of year 2009) about all issues mentioned above, from mathematical theory of P systems, power and efficiency included, for various different classes of P systems, to applications, available software and implementations, can be found in [19] (chapters are dedicated to each significant issue) and, providing the state-of-the-art, in the membrane computing website [22]. In particular, Chapter 1 of [19] is a friendly and comprehensive introduction to membrane computing.

Further (somewhat preliminary) connections between membrane computing and programming (languages, engineering, etc.) can be found in several papers, e.g., presented during the series of Workshops on Membrane Computing (details at [22]; the tenth edition of WMC took place in August 2009, [18]), as well as in [1].

References

[1] J.-P. Banatre, P. Fradet, J.-L. Giavitto, O. Michel (Eds.), Unconventional Programming Paradigms. International Workshop UPP 2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers, LNCS, vol. 3566, Springer, Berlin, 2005.
[2] F. Bernardini, M. Gheorghe, Population P systems, J. UCS, 10 (5) (2004) 509–539.

[3] L. Cardelli, Brane calculus, in: Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 2004, Revised Selected Papers, LNCS, vol. 3082, Springer, Berlin, 2005, pp. 257–280.
[4] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (Eds.), Applications of Membrane Computing, Springer, Berlin, 2006.
[5] G. Ciobanu, Gh. Păun, Gh. Ştefănescu, Sevilla carpets associated with P systems, in: M. Cavaliere et al. (Eds.), Proc. Brainstorming Week on Membrane Computing, Tarragona Univ., TR 26/03, 2003, pp. 135–140.
[6] P. Frisco, Advances in Membrane Computing, Oxford University Press, 2008.
[7] J. Gruska, Descriptional complexity of context-free languages, in: Proceedings of the Symposium on Mathematical Foundations of Computer Science, MFCS, High Tatras, 1973, pp. 71–83.
[8] J. Hromkovic, Communication Complexity and Parallel Computing: The Application of Communication Complexity in Parallel Computing, Springer, Berlin, 1997.
[9] M. Ionescu, Gh. Păun, T. Yokomori, Spiking neural P systems, Fund. Inform. 71 (2–3) (2006) 279–308.
[10] C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón, Tissue P systems, Theoret. Comput. Sci. 296 (2) (2003) 295–326.
[11] V. Nguyen, D. Kearney, G. Gioiosa, A region-oriented hardware implementation for membrane computing applications, in [18], 388–412.
[12] T.Y. Nishida, An application of P systems: a new algorithm for NP-complete optimization problems, in: N. Callaos et al. (Eds.), Proceedings of the 8th World Multi-Conference on Systems, Cybernetics and Informatics, vol. V, 2004, pp. 109–112.
[13] A. Păun, Gh. Păun, The power of communication: P systems with symport/antiport, New Gener. Comput. 20 (2002) 295–306.
[14] Gh. Păun, Computing with membranes, J. Comput. System Sci. 61 (1) (2000) 108–143 (and Turku Center for Computer Science-TUCS Report 208, November 1998. <www.tucs.fi>).
[15] Gh. Păun, P systems with active membranes: attacking NP-complete problems, J. Autom. Lang. Comb. 6 (1) (2001) 75–90.
[16] Gh. Păun, Membrane Computing. An Introduction, Springer, Berlin, 2002.
[17] Gh. Păun, M.J. Pérez-Jiménez, Solving problems in a distributed way in membrane computing: dP systems, Int. J. Comput. Commun. Control 5 (2) (2010) 238–252.
[18] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, Tenth International Workshop, WMC 2009, Curtea de Argeş Romania, August 2009, Selected and Invited Papers, LNCS, vol. 5957, Springer, Berlin, 2009.
[19] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Handbook of Membrane Computing, Oxford University Press, 2009.
[20] M.J. Pérez-Jiménez, A computational complexity theory in membrane computing, in [18], 125–148.
[21] A.E. Porreca, A. Leporati, G. Mauri, C. Zandron, Introducing a space complexity measure for P systems, Int. J. Comput. Commun. Control 4 (3) (2009) 301–310.
[22] The P Systems Website: <http://ppage.psystems.eu>.