Theoretical Computer Science 623 (2016) 180–189

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Cell-like spiking neural P systems

Tingfang Wu [a], Zhiqiang Zhang [a], Gheorghe Păun [b], Linqiang Pan [a,*]

[a] *Key Laboratory of Image Information Processing and Intelligent Control, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China*
[b] *Institute of Mathematics of the Romanian Academy, PO Box 1-764, 014700 Bucureşti, Romania*

A R T I C L E   I N F O

Article history:
Received 15 October 2015
Received in revised form 19 December 2015
Accepted 27 December 2015
Available online 6 January 2016
Communicated by M.P. Jimenez

Keywords:
Bio-inspired computing
Membrane computing
Spiking neural P system
Cell-like P system

A B S T R A C T

With mathematical motivation, we consider a combination of basic features of multiset-rewriting P systems and of spiking neural P systems, that is, we consider cell-like P systems with spiking rules in their membranes (hence dealing with only one kind of objects, the spikes). The universality of these systems as number generating devices is proved for the two usual ways to define the output (internally or externally) and for various restrictions on the spiking rules. Several research topics are also pointed out.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

Spiking neural (in short, SN) P systems were introduced in [6], as a computing model inspired from the way the neurons communicate by means of spikes, electrical impulses of identical shapes. In short, membranes (called neurons) are placed in the nodes of a graph (whose arcs are called synapses); the membranes contain copies of a unique symbol object *a* (called spike) and rules for processing the spikes: spiking rules, which consume some spikes and produce further spikes, and forgetting rules, which just consume spikes; the produced spikes are replicated and transmitted to the neurons linked by synapses with the neuron where the spiking rule is applied.

Numerous variants of SN P systems were proposed, such as SN P systems with rules on synapses [17–19], SN P systems with anti-spikes [8,20], and SN P systems with thresholds [23]. Most of these systems are Turing complete. When it is possible to create an exponential working space in a polynomial time (e.g., by neuron division or neuron budding), then computationally hard problems can be solved in a polynomial time [7,11]. Some applications of SN P systems can be found in [15,21,22,24] and references therein.

SN P systems have a tissue-like membrane arrangement, corresponding to the graph-like organization in the brain. However, the basic models of membrane computing start from the cell architecture, hence with the membranes arranged hierarchically, in the nodes of a tree. An idea appears in a natural way: what about a cell-like arrangement of membranes, using rules as in SN P systems? The question is also related to another research direction in membrane computing: obtaining universality by systems using a small number of objects. This research topic has been addressed so far only for symport/antiport P systems, first in the cell-like case [13], where three objects were proved to be sufficient, and then the

[*] Corresponding author.
E-mail addresses: wutfhust_2013@hust.edu.cn (T. Wu), zhiqiangzhang@hust.edu.cn (Z. Zhang), gpaun@us.es (G. Păun), lqpan@mail.hust.edu.cn (L. Pan).

http://dx.doi.org/10.1016/j.tcs.2015.12.038
0304-3975/© 2016 Elsevier B.V. All rights reserved.

result was extended to tissue-like P systems in [1] and the initial result has been also improved for cell-like P systems in [5]: one symbol is enough in each case for universality.

In what follows, we abbreviate "cell-like SN P system" by *cSN P system*. Of course, several issues should be fixed when defining cSN P systems, mainly in what concerns the communication among regions (we no longer say neurons, but we use the terminology associated with cell-like P systems). The usual target indications in cell-like P systems are *here, in, out* with the stronger version in_j of *in*, which also indicates the label j of the target membrane. Like in SN P systems, where all neurons connected by synapses with a given neuron receive spikes, the same number each, replicated, we also consider the target indication in_{all}, with the meaning that the associated spikes are replicated and sent to all children membranes. Another question is whether or not we allow the target *here*, having in mind that in standard SN P systems circular synapses are not allowed. As we will see below, in many proofs we can avoid the target *here*, but, as a general strategy, we propose here a general definition, covering all mentioned possibilities, but trying after that to be minimalistic in our constructions.

As expected, we obtain computing devices which are universal – hence again we have P systems with only one kind of objects which are equivalent in power with Turing machines. The proofs are much simpler than those from [1,5], and this is due to the power of the regular expressions controlling the application of the spiking rules.

Two types of output are considered: internally, as the number of spikes present in a specified region in the end of a computation, and externally, as in usual SN P systems, as the number of steps elapsed between the first and the second spikes sent out of the system.

Of course, many research directions are open in this way: just repeat the research program followed for standard SN P systems – normal forms, considering the spike train as the result of the computation (hence investigating the computing power of cSN P systems as language generators or transducers); in a natural way, we can add membrane creation and membrane division rules, thus the investigation of complexity is natural. All these research topics are left open here, but we will consider them in our future research – while the reader can also imagine further problems in this area.

The paper is organized as follows: in the next section we specify some general definitions and notations; cSN P systems are introduced in Section 3, and the definitions are illustrated with some examples in Section 4. Universality results are provided in Section 5, for the internal mode of defining the result of a computation. The external mode is discussed in Section 6. Final remarks are provided in Section 7, also mentioning further open problems and research topics.

2. Some definitions and notations

The reader is supposed to be familiar with basic elements of membrane computing, from, e.g., [14]. Details can be also found at the membrane computing web site, at http://ppage.psystems.eu. Here we only mention some of the general notations, but in order to make the paper self-contained, we shortly recall the definitions of the main notions that we investigate.

As usual, V^* is the set of all words over the alphabet V, including the empty word λ. The set of non-empty words over V, i.e., $V^* - \{\lambda\}$, is denoted by V^+. By $NREG, NRE$ we denote the family of semi-linear sets of numbers and of recursively enumerable sets of numbers (they are the length sets of regular and of recursively enumerable languages, hence the notation). The multisets over an alphabet V are represented as strings over V, with the convention that a string and all its permutations represent the same multiset. Cell-like membrane structures are represented as strings of matching labeled parentheses (we specify the label only for the right hand parenthesis).

For an alphabet V, a regular expression over V is defined as follows: (i) λ and each $a \in V$ are regular expressions, (ii) if E_1, E_2 are regular expressions over V, then also $(E_1) \cup (E_2), (E_1)(E_2)$, and $(E_1)^+$ are regular expressions over V, and (iii) nothing else is a regular expression over V. With each expression E we associate a language $L(E)$, defined in the following way: (i) $L(\lambda) = \{\lambda\}$ and $L(a) = \{a\}$, for all $a \in V$, (ii) $L((E_1) \cup (E_2)) = L(E_1) \cup L(E_2)$, $L((E_1)(E_2)) = L(E_1)L(E_2)$, and $L((E_1)^+) = L(E_1)^+$, for all regular expressions E_1, E_2 over V. Non-necessary parentheses are omitted when writing a regular expression, and $(E)^+ \cup \{\lambda\}$ is written in the form $(E)^*$.

A (cell-like) catalytic P system is a construct of the form

$$\Pi = (O, C, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o),$$

where O is the alphabet of objects, $C \subset O$ is the set of catalysts, μ is the membrane structure (with m membranes), $w_1, \ldots, w_m$ are multisets of objects present in the m regions of μ at the beginning of a computation, $R_1, \ldots, R_m$ are finite sets of evolution rules associated with the m regions of μ, and i_o indicates the output region (a membrane in μ, or the environment; in the latter case we write $i_o = env$). The rules can be of two forms: either $a \to u$ or $ca \to cu$, where $a \in O - C, c \in C, u \in ((O - C) \times \{here, in, out\})^*$. As usual, the target indication *here* is not explicitly written. When (a, in) is introduced, object a will be sent to one of the membranes immediately inside the membrane where the rule is applied, non-deterministically choosing the destination. One may also use the stronger indication in_j and then the object paired with such an indication is sent to the children membrane with label j, provided that such a membrane exists (otherwise the rule cannot be used).

The rules are used in the maximally parallel manner (a maximal multiset of applicable rules is non-deterministically chosen and applied). A result is associated only with a halting computation, in the form of the number of objects from the region i_o in the halting configuration.