Handling Languages with Spiking Neural P Systems with Extended Rules

Haiming Chen[1], Tseren-Onolt Ishdorj[3], Gheorghe Păun[2,3], Mario J. Pérez-Jiménez[3]

[1]Computer Science Laboratory, Institute of Software
Chinese Academy of Sciences
100080 Beijing, China
chm@ios.ac.cn

[2]Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucharest, Romania

[3]Research Group on Natural Computing
Department of Computer Science and AI
University of Sevilla
Avda Reina Mercedes s/n, 41012 Sevilla, Spain
tseren@yahoo.com, gpaun@us.es, marper@us.es

November 18, 2006

Abstract

We consider spiking neural P systems with spiking rules allowed to introduce zero, one, or more spikes at the same time. A tool-kit for computing (some) operations with languages generated by such systems is provided. Computing the union of languages is easy. However, computing the concatenation or the intersection with a regular language is not so easy. A way to compute weak encoding is also provided. The main results of the computing power of the obtained systems are then presented, when considering them as number generating and as language generating devices. In particular, we find direct characterizations of finite and recursively enumerable languages (without using any squeezing mechanism, as it was necessary in the case of restricted rules).

1 Introduction

We combine here two ideas recently considered in the study of the spiking neural P systems (in short, SN P systems) introduced in [3], namely the *extended* rules from [5] and the *string generation* from [1].

For the reader's convenience, we shortly recall that an SN P system consists of a set of neurons placed in the nodes of a graph and sending signals (spikes) along synapses (edges of the graph), under the control of firing rules. One neuron is designated as the *output* neuron of the system and its spikes can exit into the environment, thus producing a *spike train*. Two main kinds of outputs can be associated with a computation in an SN P system: a set of numbers, obtained by considering the number of steps elapsed between consecutive spikes which exit the output neuron, and the string corresponding to the sequence of spikes which exit the output neuron. This sequence is a binary one, with 0 associated with a step when no spike is emitted and 1 associated with a step when a spike is emitted.

The case of SN P systems as number generators was investigated in several papers, starting with [3], where it is proved that such systems are Turing complete (hence also universal, because the proof is constructive; universality in a rigorous framework was investigated in [5]). In turn, the string case is investigated in [1], where representations of finite, regular, and recursively enumerable languages were obtained, but also finite languages were found which cannot be generated in this way.

Here we consider an extension of the rules, already used in [5], namely we allow rules of the form $E/a^c \to a^p$, with the following meaning: if the content of the neuron is described by the regular expression E, then c spikes are consumed and p are produced and sent to the neurons to which there exist synapses leaving the neuron where the rule is applied (more precise definitions will be given in the next section). Thus, these rules cover and generalize at the same time both spiking rules and forgetting rules as considered so far in this area – with the mentioning that we do not also consider here a delay between firing and spiking, because in the proofs we never need such a delay.

In Section 3 we present constructions of SN P systems for computing some usual operations with languages: union, concatenation, weak coding, intersection with regular languages. Computing the union of languages is easy, but computing the concatenation or the intersection with a regular language is not so easy. A way to compute weak encoding is also provided. The main results of the computing power of these systems are recalled in Section 4. As expected, the use of extended rules allows much simpler constructions for the proof of universality in the case of considering SN P systems as number generators. More interesting is the case of strings produced by SN P systems with extended rules: we associate a symbol b_i to a step when the system sends i spikes into the environment, with two possible cases – b_0 is used as a separated symbol, or it is replaced by λ (sending no spike outside is interpreted as a step when the generated string is not grown). The first case is again restrictive: not all minimal linear languages can be obtained, but still results stronger than those from

[1] can be proved in the new framework because of the possibility of removing spikes under the control of regular expressions. The freedom provided by the existence of steps when we have no output makes possible direct characterizations of finite and recursively enumerable languages (not only representations, modulo various operations with languages, as obtained in [1] for the standard binary case).

References

[1] H. Chen, R. Freund, M. Ionescu, Gh. Păun, M.J. Pérez-Jiménez: On string languages generated by spiking neural P systems. *Proc. Fourth Brainstorming Week on Membrane Computing, vol. I*, Sevilla, 2006, 169–193. Available at [9].

[2] H. Chen, T.-O. Ishdorj, Gh. Păun, M.J. Pérez-Jiménez. Spiking neural P systems with extended rules. *Proc. Fourth Brainstorming Week on Membrane Computing, vol. I*, Sevilla, 2006, 241–265. Available at [9].

[3] M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.

[4] M. Minsky: *Computation – Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, NJ, 1967.

[5] A. Păun, Gh. Păun: Small universal spiking neural P systems. *BioSystems*, to appear, 2006.

[6] Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Spike trains in spiking neural P systems. *Intern. J. Found. Computer Sci.*, 17, 4 (2006), 975–1002.

[7] G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*, 3 volumes. Springer-Verlag, Berlin, 1997.

[8] A. Salomaa: *Formal Languages*. Academic Press, New York, 1973.

[9] The P Systems Web Page: `http://psystems.disco.unimib.it`.