# Computing with Spiking Neural P Systems: Traces and Small Universal Systems

Mihai Ionescu[1], Andrei Păun[2], Gheorghe Păun[3,4], Mario J. Pérez-Jiménez[4]

[1]Research Group on Mathematical Linguistics
Universitat Rovira i Virgili
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
armandmihai.ionescu@urv.net

[2]Department of Computer Science, Louisiana Tech University
Ruston, PO Box 10348, Louisiana, LA-71272 USA
apaun@latech.edu

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

[4]Department of Computer Science and AI, University of Sevilla
Avda Reina Mercedes s/n, 41012 Sevilla, Spain
gpaun@us.es, marper@us.es

**Abstract**

Recently, the idea of spiking neurons and thus of computing by spiking was incorporated into membrane computing, and so-called spiking neural P systems were introduced. Very shortly, in these systems neurons linked by synapses communicate by exchanging identical signals (spikes), with the information encoded in the distance between consecutive spikes. Several ways of using such devices for computing were considered in a series of papers, with universality results obtained in the case of computing numbers, both in the generating and the accepting mode; generating, accepting, or processing strings or infinite sequences was also proved to be of interest.

In the present paper, after a short survey of central notions and results related to spiking neural P systems, we contribute to this area with two (types of) results: (i) we produce small universal spiking neural P systems (84 neurons are sufficient in the basic definition, but this number is decreased to 49 neurons if a slight generalization of spiking rules is adopted), and (ii) we investigate the possibility of generating a language by following the trace of a designated spike in this way through the neurons.

## 1 Introduction

Spiking neural P systems (in short, SN P systems) were introduced in [4], with the motivation coming from two directions: the attempt of membrane computing to pass from cell-like architectures to tissue-like or neural-like architectures (see [11], [8]), and the intriguing possibility of encoding information in the duration of events, or in the interval of time elapsed between events, as in recent research in neural computing (of "third generation") [6].

This double challenge led to a class of P systems based on the following simple ideas: let us use only one object, the symbol denoting a spike, and one-membrane cells (called *neurons*) which can hold any number of spikes; each neuron fires in specified conditions (after collecting a specified number of spikes) and then sends one spike along its axon; this spike passes to all neurons connected by a *synapse* to the spiking neuron (hence it is replicated into as many copies as many target neurons exist); between the moment when a neuron fires and the moment when it spikes, each neuron needs a time interval, and this time interval is the essential ingredient of the system functioning (the basic information carrier – with the mentioning that also the number of spikes accumulated in each moment in the neurons provides an important information for controlling the functioning of the system); one of the neurons is considered the output one, and its spikes provide the output of the computation. The rules for spiking take into account *all* spikes present in a neuron not only part of them, but not all spikes present in a neuron are consumed in this way; after getting fired and before sending the spike to its synapses, the neuron is idle (biology calls this the refractory period) and cannot receive spikes. There are also rules used for "forgetting" some spikes, rules which just remove a specified number of spikes from a neuron.

In the spirit of spiking neurons, as the result of a computation (not necessarily a halting one) in [4] one considers the number of steps elapsed between the first two spikes of the output neuron. Even in this restrictive framework, SN P systems turned out to be Turing complete, *able to compute all Turing computable sets of natural numbers*, both in the generative mode (as sketched above, a number is computed if it represents the interval between the two consecutive spikes of the output neuron) and in the accepting mode (a number is introduced in the system in the form of the interval of time between the first two spikes entering a designated neuron, and this number is accepted if the computation halts). If a bound is imposed on the number of spikes present in any neuron during a computation, then *a characterization of semilinear sets of numbers is obtained.*

These results were extended in [9] to several other ways of associating a set of numbers with an SN P system: taking into account the interval between the first $k$ spikes of each spike train, or all spikes, taking only alternately the intervals, or all of them, considering halting computations. Then, the spike train itself (the sequences of symbols 0, 1 describing the activity of the output neuron: we write 0 if no spike exits the system in a time unit and 1 if a spike is emitted) was considered as the result of a computation; the infinite case is investigated in [10], the finite one in [2]. A series of possibilities of handling infinite sequences of bits are discussed in [10], while morphic representations of regular and of recursively enumerable languages are found in [2].

In this paper we directly continue these investigations, contributing in two natural directions. First, the above mentioned universality results (the possibility to compute all Turing computable sets of numbers) do not give an estimation on the number of neurons sufficient for obtaining the universality. Which is the size of the smallest universal "brain" (of the form of an SN P system)? This is both a natural and important (from computer science and, also, from neuro-science point of view) problem, reminding the extensive efforts paid for finding small universal Turing machines – see, e.g., [12] and the references therein.

Our answer is rather surprising/encouraging: *84 neurons ensure the universality* in the basic setup of SN P systems, as they were defined in [4], while this number is decreased to 49 if slightly more general spiking rules are used (providing the possibility to produce not only one spike, *but also two or more spikes at the same time*). The proof is based on simulating a small universal register machine from [5].

Then, another natural issue is to bring to the SN P systems area a notion introduced for symport/antiport P systems in [3]: mark a spike and follow its path through the system, recording the labels of the visited neurons until either the marking disappears or the computation halts. Because of the very restrictive way of generating strings in this way, there are simple languages

which cannot be computed, but, on the other hand, there are rather complex languages which can be obtained in this framework.

Due to space restrictions, we do not give formal definitions/results for SN P systems (we refer to the above mentioned papers for that); also, we only present the ideas of the proofs, with full details to be given in separate papers to be circulated/announced through [13].

## References


[1] M. Cavaliere, P. Leupold: Evolution and observation – A new way to look at membrane systems. In *Membrane Computing. Intern. Workshop'WMC 2003, Tarragona, Spain, July 2003. Revised Papers* (C. Martin-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), LNCS 2933, Springer, Berlin, 2004, 70–87.

[2] H. Chen, R. Freund, M. Ionescu, Gh. Păun, M.J. Pérez-Jiménez: On string languages generated by spiking neural P systems. Submitted, 2006.

[3] M. Ionescu, C. Martin-Vide, A. Păun, Gh. Păun: Membrane systems with symport/antiport: (unexpected) universality results. In *Proc. 8th International Meeting of DNA Based Computing* (M. Hagiya, A. Ohuchi, eds.), Japan, 2002, 151–160.

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

[5] I. Korec: Small universal register machines. *Theoretical Computer Science*, 168 (1996), 267–301.

[6] W. Maass, C. Bishop, eds.: *Pulsed Neural Networks*, MIT Press, Cambridge, 1999.

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

[8] Gh. Păun: *Membrane Computing – An Introduction*. Springer-Verlag, Berlin, 2002.

[9] Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Spike trains in spiking neural P systems. *Intern. J. Found. Computer Sci.*, to appear (also available at [13]).

[10] Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Infinite spike trains in spiking neural P systems. Submitted, 2006.

[11] Gh. Păun, Y. Sakakibara, T. Yokomori: P systems on graphs of restricted forms. *Publicationes Mathematicae Debrecen*, 60 (2002), 635–660.

[12] Y. Rogozhin: Small universal Turing machines. *Theoretical Computer Science*, 168 (1996), 215–240.

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