Theoretical Computer Science 567 (2015) 2–20

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Three small universal spiking neural P systems ☆

Turlough Neary

Institute for Neuroinformatics, University of Zürich, ETH Zürich, Switzerland

ARTICLE INFO

Article history:
Received 7 May 2014
Received in revised form 22 August 2014
Accepted 3 September 2014
Available online 16 September 2014
Communicated by L. Kari

Keywords:
Universal spiking neural P system
Computational complexity
Counter machine

ABSTRACT

In this work we give three small spiking neural P systems. We begin by constructing a universal spiking neural P system with extended rules and only 4 neurons. This is the smallest possible number of neurons for a universal system of its kind. We prove this by showing that the set of problems solved by spiking neural P systems with 3 neurons is bounded above by NL, and so there exists no such universal system with 3 neurons. If we generalise the output technique we immediately find a universal spiking neural P system with extended rules that has only 3 neurons. This is also the smallest possible number of neurons for a universal system of its kind. Finally, we give a universal spiking neural P system with standard rules and only 7 neurons. In addition to giving a significant improvement in terms of reducing the number of neurons, our systems also offer an exponential improvement on the time and space overheads of the small universal spiking neural P systems of other authors.

1. Introduction

The search for the simplest Turing universal models of computation helps us to determine the trade-offs between the various parameters of a model that allow computational completeness. In other words, it helps us find the minimum requirements for universality in a model. Since their recent inception, spiking neural P (SN P) systems [1] have been the subject of such endeavours [2–5].

The search to find the smallest universal SN P systems, where size is the number of neurons, was initiated by Păun and Păun [2]. We begin by solving the problem of finding the smallest possible number of neurons needed to give a universal SN P system with extended rules, one of the open problems given in [6]. To achieve this, we begin by giving a universal extended SN P system that has only 4 neurons. Then, we prove that the set of problems solved by SN P systems with 3 neurons is bounded above by NL, and so there exists no such universal system with 3 neurons. Thus, our 4-neuron system has the smallest possible number of neurons for a universal extended SN P system. We also find that if a generalised output technique is used we can give a universal SN P system with extended rules that has only 3 neurons. This is also the smallest possible number of neurons for a universal system of its kind. The results for the extended model previously appeared in [7]. For the standard model, finding a universal SN P system with a small number of neurons is made difficult due to the very limited way in which the neurons can communicate with each other. Here we give a universal SN P system for the standard model that has only 7 neurons. This represents a significant improvement with the lowest number of neurons used to give such a universal system by any other author standing at 67.

The systems we present in this work offer an exponential improvement on the time and space overheads of the small universal SN P systems of other authors [2,12,15]. Our systems simulate Turing machines with exponential time and space

☆ This work was supported by Swiss National Science Foundation grant number 200021-141029.
E-mail address: tneary@ini.phys.ethz.ch.

http://dx.doi.org/10.1016/j.tcs.2014.09.006
0304-3975/

Table 1
Small universal SN P systems. The "simulation time/space" column gives the overheads used by each system when simulating a standard single tape Turing machine. Further explanation of the time/space complexity overheads and comparisons between some of the systems in this table can be found in Section 3.

Number of neurons	Simulation time/space	Type of rules	Exhaustive use of rules	Author
125	double-exponential/triple-exponential	extended	yes	Zhang et al. [8,9]
18	polynomial/exponential	extended	yes	Neary [10]
10	linear/exponential	extended	yes	Neary [11]
49	double-exponential	extended	no	Păun and Păun [2]
41	double-exponential	extended	no	Zhang et al. [12]
18	exponential	extended	no	Neary [13,14][a]
12	double-exponential	extended	no	Neary [13]
9	double-exponential	extended	no	Zeng et al. [15]
4	**exponential**	**extended**	**no**	**Section 4**[c]
3	**exponential**	**extended**	**no**	**Section 5**[c,b]
84	double-exponential	standard	no	Păun and Păun [2]
67	double-exponential	standard	no	Zhang et al. [12]
17	exponential	standard	no	Neary [16]
11	exponential	standard	**no**	Neary [17]
7	**exponential**	**standard**	**no**	**Section 6**

[a] The 18 neuron system is not explicitly given in [13]; it was presented at [14] and is easily derived from the other system in [13].
[b] A more generalised output technique is used.
[c] The smallest possible number of neurons for a universal system of its kind.

overheads whereas the systems in [2,12,15] simulate Turing machines with double-exponential time and space overheads. In other works [10,11], it is shown that universal SN P systems require exponential time to simulate Turing machines, and so significant improvement on the time and space overheads of our systems is not possible. Table 1 gives the state of the art in small universal SN P systems and their respective simulation time and space overheads.[1]

Păun and Păun [2] state that a significant decrease on the number of neurons of their two universal SN P systems is improbable (also stated in [6]). The huge reduction in the number of neurons that is given by Theorems 1 and 4 is in part due to the method we use to encode the instructions of the counter machines being simulated. With the exception of the 18 and 10-neurons systems with exhaustive use of rules, all of the SN P systems given in Table 1 simulate counter machines. The number of neurons in previous small universal systems [2,12] were dependent on the number of instructions in the counter machine being simulated. In our systems each unique counter machine instruction is encoded as a unique number of spikes, and thus the number of neurons in our SN P systems is independent of the number of counter machine instructions. The technique of encoding the instructions as spikes was first used to construct small universal SN P systems in [13] (see Table 1). Another reason for the small number of neurons in our systems is that we have done away with the need for a dedicated input module as the neurons that deal with the input from the environment also perform other functions.

2. SN P systems

Definition 1 *(Spiking neural P system).* A spiking neural P system (SN P system) is a tuple $\Pi = (O, \sigma_1, \sigma_2, \cdots, \sigma_m, syn, in, out)$, where:

1. $O = \{s\}$ is the unary alphabet (s is known as a spike),
2. $\sigma_1, \sigma_2, \cdots, \sigma_m$ are neurons, of the form $\sigma_i = (n_i, R_i)$, $1 \leqslant i \leqslant m$, where:
   (a) $n_i \geqslant 0$ is the initial number of spikes contained in σ_i,
   (b) R_i is a finite set of rules of the following two forms:
      i. $E/s^b \to s; d$, where E is a regular expression over s, $b \geqslant 1$ and $d \geqslant 0$,
      ii. $s^e \to \lambda$, where λ is the empty word, $e \geqslant 1$, and for all $E/s^b \to s; d$ from R_i $s^e \notin L(E)$ where $L(E)$ is the language defined by E,
3. $syn \subseteq \{1, 2, \cdots, m\} \times \{1, 2, \cdots, m\}$ is the set of synapses between neurons, where $i \neq j$ for all $(i, j) \in syn$,
4. $in, out \in \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$ are the input and output neurons, respectively.

A firing rule $r = E/s^b \to s; d$ is applicable in a neuron σ_i if there are $j \geqslant b$ spikes in σ_i and $s^j \in L(E)$ where $L(E)$ is the set of words defined by the regular expression E. If, at time t, rule r is executed then b spikes are removed from the neuron, and at time $t + d$ the neuron fires. When a neuron σ_i fires a spike is sent to each neuron σ_j for every synapse (i, j) in Π. Also, the neuron σ_i remains closed and does not receive spikes until time $t + d$ and no other rule may execute in σ_i until time $t + d + 1$. A forgetting rule $r' = s^e \to \lambda$ is applicable in a neuron σ_i if there are exactly e spikes in σ_i. If r' is executed

[1] In a similar table given in [7] the time/space complexity for a number of the systems is incorrect due to an error copied from Korec's paper [18]. For more see Section 3.