A Boundary Between Universality and Non-Universality in Extended Spiking Neural P Systems*

Turlough Neary

Boole Centre for Research in Informatics, University College Cork, Ireland.
`tneary@cs.nuim.ie`

Abstract. We solve the problem of finding the smallest possible universal spiking neural P system with extended rules. We give a universal spiking neural P system with extended rules and only 4 neurons. This is the smallest possible 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 (for any reasonable definition of universality). Finally, we show that if we generalise the output technique we can give a universal spiking neural P system with extended rules that has only 3 neurons. This is also the smallest possible universal system of its kind.

1 Introduction

Spiking neural P systems (SN P systems) [1] are quite a new computational model that are a synergy inspired by P systems and spiking neural networks. Here we solve the problem of finding the smallest possible universal spiking neural P system with extended rules; one of the open problems given in [9]. We give a universal extended SN P system that has only only 4 neurons. Following this, we prove 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 (for any reasonable definition of universality). Thus, our 4-neuron system is the smallest possible universal extended SN P system. Finally, we show that if we generalise the output technique we can give a universal SN P system with extended rules that has only 3 neurons. This is also the smallest possible universal system of its kind. Table 1 gives the smallest universal extended SN P systems and their respective simulation time and space overheads. For more on the time/space complexity of small universal SNP systems see [4, 6].

In their paper containing the 49-neuron system, Păun and Păun [8] state that a significant decrease on the number of neurons of their two universal SN P systems is improbable (also stated in [9]). The dramatic improvement on the

number of neurons	simulation time/space	author
49	exponential	Păun & Păun [8]
41	exponential	Zhang et al. [10]
18	exponential	Neary [5, 3]‡
12	double-exponential	Neary [5]
12	exponential	Pan & Zeng [7]*
4	**exponential**	**Section 4**
3	**exponential**	**Section 5†**

Table 1. Small universal extended SN P systems. The "simulation time/space" column gives the overheads used by each system when simulating a standard single tape Turing machine. † A more generalised output technique is used. ‡ The 18 neuron system is not explicitly given in [5]; it was presented at [3] and is easily derived from the other system in [5]. *The system in [7] does not include the input module. This is not the case for all the other systems in this table. Note that if we remove the input module from the 18- and 12-neuron systems in [5] we get 12- and 9-neuron systems, respectively.

size of earlier small universal SN P systems given by Theorem 1 is in part due to the method we use to encode the instructions of the counter machines being simulated. All of the SN P systems given in Table 1 simulate counter machines. The size of previous small universal systems [8, 10] were dependant 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 size of 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 [5] (see Table 1).

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 then e spikes are removed from the neuron. At each timestep t a rule must be applied in each neuron if there is one or more applicable rules. Thus, while the application of rules in each individual neuron is sequential the neurons operate in parallel with each other.

Note from 2b(i) of Definition 1 that there may be two rules of the form $E/s^b \to s; d$, that are applicable in a single neuron at a given time. If this is the case then the next rule to execute is chosen non-deterministically.

An *extended* SN P system [8] has more general rules of the form $E/s^b \to s^p; d$, where $b \geqslant p \geqslant 1$. Thus, a synapse in a SN P system with extended rules may transmit more than one spike in a single timestep. The SN P systems we present in this work use rules without delay, and thus in the sequel we write rules as $E/s^b \to s^p$. Also, if in a rule $E = s^b$ then we write the rule as $s^b \to s^p$.

Spikes are introduced into the system from the environment by reading in a binary sequence (or word) $w \in \{0, 1\}$ via the input neuron σ_1. The sequence w is read from left to right one symbol at each timestep and a spike enters the input neuron from the environment on a given timestep iff the read symbol is 1. The output of a SN P system Π is the time between the first and second timesteps a firing rule is applied in the output neuron.

3   Counter Machines

Definition 2. *A counter machine is a tuple $C = (z, R, c_m, Q, q_1, q_h)$, where z gives the number of counters, R is the set of input counters, c_m is the output counter, $Q = \{q_1, q_2, \cdots, q_h\}$ is the set of instructions, and $q_1, q_h \in Q$ are the initial and halt instructions, respectively.*

Each counter c_j stores a natural number value $y \geqslant 0$. Each instruction q_i is of one of the following two forms:

- $q_i : INC(j)$ increment the value y stored in counter c_j by 1 and move to instruction q_l.
- $q_i : DEC(j)q_k$ if the value y stored in counter c_j is greater than 0 then decrement this value by 1 and move to instruction q_l, otherwise if $y = 0$ move to instruction q_k.

At the beginning of a computation the first instruction executed is q_1. The input to the counter machine is initially stored in the input counters. If the counter machine's control enters instruction q_h, then the computation halts at that timestep. The result of the computation is the value y stored in the output counter c_m when the computation halts.

In earlier work [8], Korec's notion of strong universality was adopted for small SN P systems as follows: if the input to ϕ_x is y then the input to the SN

neuron	rules
σ_1	$(s^{12h})^* s^{6h+8}/s^{6h} \to s^{6h}, \qquad (s^{12h})^* s^{6h+9}/s^{9h+5} \to s^{3h+2},$
	$(s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2}, \quad \text{if } q_i : INC(1), q_i : DEC(1) \notin \{Q\}$
	$(s^{12h})^* s^{6(h+i)+4}/s^{12h+6i+4-6l} \to s^{6l}, \quad \text{if } q_i : INC(1) \in \{Q\}$
	$(s^{12h})^* s^{54h+6i+4}/s^{36h+6i+4-6l} \to s^{6l}, \quad \text{if } q_i : DEC(1) \in \{Q\}$
	$s^{42h+6i+4}/s^{24h+6i+4-6k} \to s^{6k}, \quad \text{if } q_i : DEC(1) \in \{Q\}$
σ_2	$s^{18h+7}/s^{6h} \to s^{6h}, \qquad (s^{12h})^* s^{6h+9}/s^{9h+5} \to s^{3h+2},$
	$(s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2}, \quad \text{if } q_i : INC(2), q_i : DEC(2) \notin \{Q\}$
	$(s^{12h})^* s^{6(h+i)+4}/s^{12h+6i+4-6l} \to s^{6l}, \quad \text{if } q_i : INC(2) \in \{Q\}$
	$(s^{12h})^* s^{54h+6i+4}/s^{36h+6i+4-6l} \to s^{6l}, \quad \text{if } q_i : DEC(2) \in \{Q\}$
	$s^{42h+6i+4}/s^{24h+6i+4-6k} \to s^{6k}, \quad \text{if } q_i : DEC(2) \in \{Q\}$
σ_3	$s^{18h+7}/s^{6h} \to s^{6h}, \; (s^{12h})^*/s^{12h+1} \to s^{12h}, \; s^{18h+2}/s^{6h} \to s^{6h}, \; s^{24h-1} \to s^{12h},$
	$s^{18h+3}/s^{6h+1} \to s^{6h+1}, \quad s^{18h+8}/s^{6h+6} \to s^{6h+1}, \quad (s^{12h})^* s^{36h-1}/s^{12h} \to s^{6h},$
	$(s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2}, \quad \text{if } q_i : INC(3), q_i : DEC(3) \notin \{Q\}$
	$(s^{12h})^* s^{6(h+i)+4}/s^{12h+6i+4-6l} \to s^{6l}, \quad \text{if } q_i : INC(3) \in \{Q\}$
	$(s^{12h})^* s^{54h+6i+4}/s^{36h+6i+4-6l} \to s^{6l}, \quad \text{if } q_i : DEC(3) \in \{Q\}$
	$s^{42h+6i+10}/s^{24h+6i+4-6k} \to s^{6k}, \quad \text{if } q_i : DEC(3) \in \{Q\}$
σ_4	$s^{6h} \to \lambda, \qquad s^{6h+1} \to \lambda, \qquad s^{12h+2} \to \lambda, \qquad s^{6l} \to \lambda, \qquad s^{12h} \to s$

Table 2. Rules for each of the neurons of Π_C. Here $1 \leqslant i < h$, $1 \leqslant l \leqslant h$ and $1 \leqslant k \leqslant h$.

P system is the sequence $10^{y-1}10^{x-1}1$. As with the SN P systems given in [8, 10], the system we give in Theorem 1 satisfies the notion of strong universality adopted from Korec in [8]. However, as we noted in other work [6], it could be considered that Korec's notion [2] of strong universality is somewhat arbitrary and we also pointed out some inconsistency in his notion of weak universality. Hence, in this work we rely on time/space complexity analysis to compare small SN P systems and their encodings (see Table 1).

4   A Small Universal Extended SN P System

Theorem 1. *Let C be a universal counter machine with 3 counters that completes its computation in time t to give the output value x_3 when given the pair of input values (x_1, x_2). Then there is a universal extended SN P system Π_C that simulates the computation of C in time $O(t + x_1 + x_2 + x_3)$ and has only 4 neurons.*

Proof. Let $C = (3, \{c_1, c_2\}, c_3, Q, q_1, q_h)$ where $Q = \{q_1, q_2, \cdots, q_h\}$. Our SN P system Π_C is given by Figure 1 and Table 2. Π_C is deterministic.
Encoding of a configuration of C and reading input into Π_C. A configuration of C is stored as spikes in the neurons of Π_C. The next instruction q_i to be executed is stored in each of the neurons σ_1, σ_2 and σ_3 as $6(h + i)$ spikes. Let x_1, x_2 and x_3 be the values stored in counters c_1, c_2 and c_3, respectively. Then
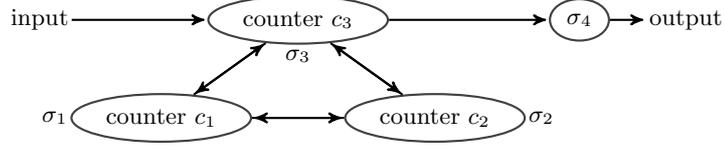
Fig. 1. Universal extended SN P system Π_C. Each oval labeled σ_i is a neuron. An arrow going from neuron σ_i to neuron σ_j illustrates a synapse (i, j).

the values x_1, x_2 and x_3 are stored as $12h(x_1 + 1)$, $12h(x_2 + 1)$ and $12h(x_3 + 1)$ spikes in neurons σ_1, σ_2 and σ_3, respectively. The input to Π_C is read into the system via the input neuron σ_3 (see Figure 1). If C begins its computation with the values x_1 and x_2 in counters c_1 and c_2, respectively, then the binary sequence $w = 10^{x_1-1}10^{x_2-1}1$ is read in via the input neuron σ_3. Thus, σ_3 receives a single spike from the environment at times t_1, t_{x_1+1} and $t_{x_1+x_2+1}$. We explain how the system is initialised to encode an initial configuration of C by giving the number of spikes in each neuron and the rule that is to be applied in each neuron at time t. Before the computation begins neuron σ_1 contains $6h + 7$ spikes, σ_2 contains $18h + 7$ spikes, σ_3 contains $18h + 6$ spikes and σ_4 contains no spikes. Thus, when σ_3 receives it first spike at time t_1 we have

$$t_1: \quad \sigma_1 = 6h + 7,$$
$$\sigma_2, \sigma_3 = 18h + 7, \qquad\qquad s^{18h+7}/s^{6h} \to s^{6h}.$$

where on the left $\sigma_k = z$ gives the number z of spikes in neuron σ_k at time t and on the right is the rule that is to be applied at time t, if there is an applicable rule at that time. Thus, from Figure 1, when we apply the rule $s^{18h+7}/s^{6h} \to s^{6h}$ in neurons σ_2 and σ_3 at time t_1 we get

$$t_2: \quad \sigma_1 = 18h + 7,$$
$$\sigma_2, \sigma_3 = 18h + 7, \qquad\qquad s^{18h+7}/s^{6h} \to s^{6h},$$
$$\sigma_4 = 6h, \qquad\qquad s^{6h} \to \lambda,$$

$$t_3: \quad \sigma_1 = 30h + 7,$$
$$\sigma_2, \sigma_3 = 18h + 7, \qquad\qquad s^{18h+7}/s^{6h} \to s^{6h},$$
$$\sigma_4 = 6h, \qquad\qquad s^{6h} \to \lambda.$$

Neurons σ_2 and σ_3 send $12h$ spikes to neuron σ_1 on each timestep between times t_1 and t_{x_1+1}. This gives a total of $12hx_1$ spikes sent to σ_1 during the time interval t_1 to t_{x_1+1}. Thus when σ_3 receives the second spike from the environment we get

$$t_{x_1+1}: \quad \sigma_1 = 12hx_1 + 6h + 7,$$
$$\sigma_2 = 18h + 7, \qquad\qquad s^{18h+7}/s^{6h} \to s^{6h},$$
$$\sigma_3 = 18h + 8, \qquad\qquad s^{18h+8}/s^{6h+6} \to s^{6h+1},$$
$$\sigma_4 = 6h, \qquad\qquad s^{6h} \to \lambda,$$

$$t_{x_1+2}: \qquad \sigma_1 = 12h(x_1 + 1) + 6h + 8, \qquad (s^{12h})^* s^{6h+8} / s^{6h} \to s^{6h},$$
$$\sigma_2 = 18h + 8,$$
$$\sigma_3 = 18h + 2, \qquad\qquad s^{18h+2} / s^{6h} \to s^{6h},$$
$$\sigma_4 = 6h + 1, \qquad\qquad\qquad s^{6h+1} \to \lambda,$$

$$t_{x_1+3}: \qquad \sigma_1 = 12h(x_1 + 1) + 6h + 8, \qquad (s^{12h})^* s^{6h+8} / s^{6h} \to s^{6h},$$
$$\sigma_2 = 30h + 8,$$
$$\sigma_3 = 18h + 2, \qquad\qquad s^{18h+2} / s^{6h} \to s^{6h},$$
$$\sigma_4 = 6h, \qquad\qquad\qquad s^{6h} \to \lambda.$$

Neurons σ_1 and σ_3 fire on every timestep between times t_{x_1+2} and $t_{x_1+x_2+2}$ to send a total of $12hx_2$ spikes to σ_2. Thus, when σ_3 receives the last spike from the environment we have

$$t_{x_1+x_2+1}: \qquad \sigma_1 = 12h(x_1 + 1) + 6h + 8, \qquad (s^{12h})^* s^{6h+8} / s^{6h} \to s^{6h},$$
$$\sigma_2 = 12hx_2 + 6h + 8,$$
$$\sigma_3 = 18h + 3, \qquad\qquad s^{18h+3} / s^{6h+1} \to s^{6h+1},$$
$$\sigma_4 = 6h, \qquad\qquad\qquad s^{6h} \to \lambda,$$

$$t_{x_1+x_2+2}: \qquad \sigma_1 = 12h(x_1 + 1) + 6h + 9, \qquad (s^{12h})^* s^{6h+9} / s^{9h+5} \to s^{3h+2},$$
$$\sigma_2 = 12h(x_2 + 1) + 6h + 9, \qquad (s^{12h})^* s^{6h+9} / s^{9h+5} \to s^{3h+2}$$
$$\sigma_3 = 18h + 2, \qquad\qquad s^{18h+2} / s^{6h} \to s^{6h},$$
$$\sigma_4 = 6h + 1, \qquad\qquad\qquad s^{6h+1} \to \lambda,$$

$$t_{x_1+x_2+3}: \qquad \sigma_1 = 12h(x_1 + 1) + 6(h + 1),$$
$$\sigma_2 = 12h(x_2 + 1) + 6(h + 1),$$
$$\sigma_3 = 12h + 6(h + 1).$$

At time $t_{x_1+x_2+3}$ neuron σ_1 contains $12h(x_1 + 1) + 6(h + 1)$ spikes, σ_2 contains $12h(x_2 + 1) + 6(h + 1)$ spikes and σ_3 contains $12h + 6(h + 1)$ spikes. Thus at time $t_{x_1+x_2+3}$ the SN P system encodes an initial configuration of C.

Π_C **simulating** $q_i : INC(1)$. Let counters c_1, c_2, and c_3 have values x_1, x_2, and x_3, respectively. Then the simulation of $q_i : INC(1)$ begins at time t_j with $12h(x_1 + 1) + 6(h + i)$ spikes in σ_1, $12h(x_2 + 1) + 6(h + i)$ spikes in σ_2 and $12h(x_3 + 1) + 6(h + i)$ spikes in σ_3. Thus, at time t_j we have

$$t_j: \qquad \sigma_1 = 12h(x_1 + 1) + 6(h + i),$$
$$\sigma_2 = 12h(x_2 + 1) + 6(h + i), \qquad (s^{12h})^* s^{6(h+i)} / s^{12h+6i+2} \to s^{12h+2},$$
$$\sigma_3 = 12h(x_3 + 1) + 6(h + i), \qquad (s^{12h})^* s^{6(h+i)} / s^{12h+6i+2} \to s^{12h+2}.$$

From Figure 1, when we apply the rule $(s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2}$ in neurons σ_2 and σ_3 at time t_j we get

$$t_{j+1}: \quad \sigma_1 = 12h(x_1+3) + 6(h+i) + 4, \quad (s^{12h})^* s^{6(h+i)+4}/s^{12h+6i+4-6l} \to s^{6l},$$
$$\sigma_2 = 12h(x_2+1) + 6h,$$
$$\sigma_3 = 12h(x_3+1) + 6h,$$
$$\sigma_4 = 12h + 2, \qquad\qquad\qquad\qquad\qquad s^{12h+2} \to \lambda,$$

$$t_{j+2}: \quad \sigma_1 = 12h(x_1+2) + 6(h+l),$$
$$\sigma_2 = 12h(x_2+1) + 6(h+l),$$
$$\sigma_3 = 12h(x_3+1) + 6(h+l).$$

At time t_{j+2} the simulation of $q_i : INC(1)$ is complete. Note that an increment on the value x_1 in counter c_1 was simulated by increasing the $12h(x_1+1)$ spikes in σ_1 to $12h(x_1+2)$ spikes. Note also that the encoding $6(h+l)$ of the next instruction q_l has been established in neurons σ_1, σ_2 and σ_3.

Π_C **simulating** $q_i : DEC(1)q_k$. There are two cases to consider here. Case 1: if counter c_1 has value $x_1 > 0$, then decrement c_1 and move to instruction q_l. Case 2: if counter c_1 has value $x_1 = 0$, then move to instruction q_k. As with the previous example, our simulation begins at time t_j. Thus Case 1 $(x_1 > 0)$ gives

$$t_j: \quad \sigma_1 = 12h(x_1+1) + 6(h+i),$$
$$\sigma_2 = 12h(x_2+1) + 6(h+i), \qquad (s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2},$$
$$\sigma_3 = 12h(x_3+1) + 6(h+i), \qquad (s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2},$$

$$t_{j+1}: \quad \sigma_1 = 12h(x_1+3) + 6(h+i) + 4, \quad (s^{12h})^* s^{54h+6i+4}/s^{36h+6i+4-6l} \to s^{6l},$$
$$\sigma_2 = 12h(x_2+1) + 6h,$$
$$\sigma_3 = 12h(x_3+1) + 6h,$$
$$\sigma_4 = 12h + 2, \qquad\qquad\qquad\qquad\qquad s^{12h+2} \to \lambda,$$

$$t_{j+2}: \quad \sigma_1 = 12hx_1 + 6(h+l),$$
$$\sigma_2 = 12h(x_2+1) + 6(h+l),$$
$$\sigma_3 = 12h(x_3+1) + 6(h+l).$$

At time t_{j+2} the simulation of $q_i : DEC(1)q_k$ for Case 1 $(x_1 > 0)$ is complete. Note that a decrement on the value x_1 in counter c_1 was simulated by decreasing the $12h(x_1+1)$ spikes in σ_1 to $12hx_1$ spikes. Note also that the encoding $6(h+l)$ of the next instruction q_l has been established in neurons σ_1, σ_2 and σ_3.

Alternatively, if we have Case 2 ($x_1 = 0$) then we get

$$t_j : \quad \sigma_1 = 12h + 6(h+i),$$
$$\sigma_2 = 12h(x_2 + 1) + 6(h+i), \quad (s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2},$$
$$\sigma_3 = 12h(x_3 + 1) + 6(h+i), \quad (s^{12h})^* s^{6(h+i)}/s^{12h+6i+2} \to s^{12h+2},$$

$$t_{j+1} : \quad \sigma_1 = 42h + 6i + 4, \qquad s^{42h+6i+4}/s^{24h+6i+4-6k} \to s^{6k},$$
$$\sigma_2 = 12h(x_2 + 1) + 6h,$$
$$\sigma_3 = 12h(x_3 + 1) + 6h,$$
$$\sigma_4 = 12h + 2, \qquad\qquad\qquad\qquad s^{12h+2} \to \lambda,$$

$$t_{j+2} : \quad \sigma_1 = 12h + 6(h+k),$$
$$\sigma_2 = 12h(x_2 + 1) + 6(h+k),$$
$$\sigma_3 = 12h(x_3 + 1) + 6(h+k).$$

At time t_{j+2} the simulation of $q_i : DEC(1)q_k$ for Case 2 is complete. The encoding $6(h+k)$ of the next instruction q_k has been established in σ_1, σ_2 and σ_3.

Halting The halt instruction q_h is encoded as $12h$ spikes. Thus if C halts we get

$$t_j : \quad \sigma_1 = 12h(x_1 + 2),$$
$$\sigma_2 = 12h(x_2 + 2),$$
$$\sigma_3 = 12h(x_3 + 2), \qquad\qquad (s^{12h})^*/s^{12h+1} \to s^{12h},$$

$$t_{j+1} : \quad \sigma_1 = 12h(x_1 + 3),$$
$$\sigma_2 = 12h(x_2 + 3),$$
$$\sigma_3 = 12h(x_3 + 1) - 1, \qquad (s^{12h})^* s^{36h-1}/s^{12h} \to s^{6h},$$
$$\sigma_4 = 12h, \qquad\qquad\qquad\qquad s^{12h} \to s,$$

$$t_{j+2} : \quad \sigma_1 = 12h(x_1 + 3) + 6h,$$
$$\sigma_2 = 12h(x_2 + 3) + 6h,$$
$$\sigma_3 = 12hx_3 - 1, \qquad\qquad (s^{12h})^* s^{36h-1}/s^{12h} \to s^{6h},$$
$$\sigma_4 = 6h, \qquad\qquad\qquad\qquad s^{6h} \to \lambda.$$

The rule $(s^{12h})^* s^{36h-1}/s^{12h} \to s^{6h}$ is applied a further $x_3 - 2$ times in σ_3 to give

$$t_{j+x_3} : \quad \sigma_1 = 12h(x_1 + 3) + 6h(x_3 - 1),$$
$$\sigma_2 = 12h(x_2 + 3) + 6h(x_3 - 1),$$
$$\sigma_3 = 24h - 1, \qquad\qquad s^{24h-1} \to s^{12h},$$
$$\sigma_4 = 6h, \qquad\qquad\qquad s^{6h} \to \lambda,$$

$$t_{j+x_3+1}: \qquad \sigma_1 = 12h(x_1 + 4) + 6h(x_3 - 1),$$
$$\sigma_2 = 12h(x_2 + 4) + 6h(x_3 - 1),$$
$$\sigma_4 = 12h, \qquad\qquad\qquad s^{12h} \rightarrow s.$$

As usual the output is the time interval between the first and second timesteps when a firing rule is applied in the output neuron. Note from above that the output neuron σ_4 fires for the first time at timestep t_{j+1} and for the second time at timestep t_{j+x_3+1}. Thus, the output of Π_C is x_3 the value of the output counter c_3 when C enters the halt instruction q_h. Note that if $x_3 = 0$ then the rule $s^{24h-1} \rightarrow s^{12h}$ can not be executed as there is only $12h - 1$ spikes in σ_3 at timestep t_{j+1}. Thus if $x_2 = 0$ the output neuron will fire only once.

We have shown how to simulate arbitrary instructions of the form $q_i : INC(1)$ and $q_i : DEC(1)q_k$ that operate on counter c_1. Instructions which operate on counters c_2 and c_3 are simulated in a similar manner. Immediately following the simulation of an instruction Π_C is configured to simulate the next instruction. Each instruction of C is simulated in 2 timesteps. The pair of input values (x_1, x_2) is read into the system in $x_1 + x_2 + 3$ timesteps and sending the output value x_3 out of the system takes $x_3 + 1$ timesteps. Thus, if C completes it computation in time t, then Π_C simulates the computation of C in linear time $O(t + x_1 + x_2 + x_3)$.

$\square$

5   Lower Bounds for Small Universal SN P Systems

In this and other works [8, 10] on small SN P systems the input neuron only receives a constant number of spikes from the environment and the output neuron fires no more than a constant number of times. Hence, we call the input standard if the input neuron receives no more than x spikes from the environment, where x is a constant independent of the input (i.e. the number of 1s in its input sequence is $< x$). Similarly, we call the output standard if the output neuron fires no more than y times, where y is a constant independent of the input. Here we say a SN P system has generalised input if the input neuron is permitted to receive $\leqslant n$ spikes from the environment where $n \in \mathbb{N}$ is the length of its input sequence.

Theorem 2. *Let Π be any extended SN P system with only 3 neurons, generalised input and standard output. Then there is a non-deterministic Turing machine T_Π that simulates the computation of Π in space $O(\log n)$ where n is the length of the input to Π.*

Proof. Let Π be any extended SN P system with generalised input, standard output, and neurons σ_1, σ_2 and σ_3. Also, let y be the maximum number of times the output neuron σ_3 is permitted to fire and let q and r be the maximum value for b and p respectively, for all $E/s^b \rightarrow s^p; d$ in Π.

We begin by explaining how the activity of σ_3 may be simulated using only the states of T_Π (i.e. no workspace is required to simulate σ_3). Recall that the
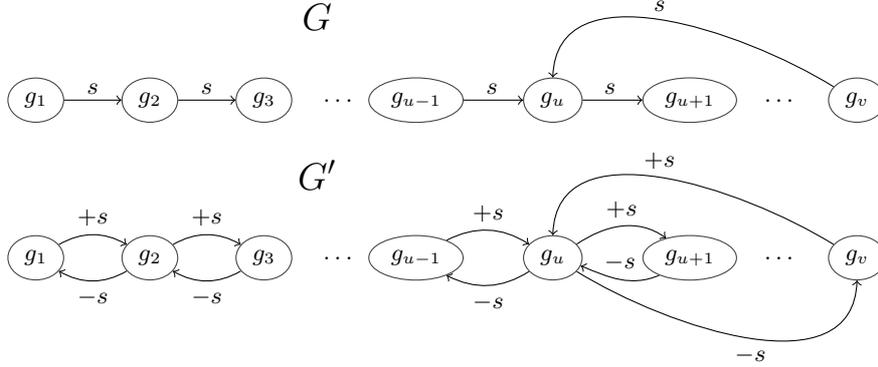
$$G$$

$$G'$$

Fig. 2. Finite state machine G decides if there is any rule applicable in a neuron given the number of spikes in the neuron *at a given time* in the computation. Each s represents a spike in the neuron. Machine G' keeps track of the movement of spikes into and out of the neuron and decides whither or not any rule is applicable *at each timestep* in the computation. $+s$ represents a single spike entering the neuron and $-s$ represents a single spike exiting the neuron.

applicability of each rule is determined by a regular expression over a unary alphabet. We can give a single regular expression R that is the union of all the regular expressions for the firing rules of σ_3. This regular expression R determines whither or not there is any applicable rule in σ_3 at each timestep. Figure 2 gives the deterministic finite automata G that accepts $L(R)$ the language generated by R. During a computation we may use G to decide which rules are applicable in σ_3 by passing an s to G each time a spike enters σ_3. However, G may not give the correct result if spikes leave the neuron as it does not record spikes leaving σ_3. Thus, using G we may construct a second machine G' such that G' records the movement of spikes going into and out of the neuron. G' is constructed as follows: G' has all the same states (including accept states) and transitions as G along with an extra set of transitions that record spikes leaving the neuron. This extra set of transitions are given as follows: for each transition on s from a state g_i to a state g_j in G there is a new transition on $-s$ going from state g_j to g_i in G' that records the removal of a spike from σ_3. By recording the dynamic movement of spikes, G' is able to decide which rules are applicable in σ_3 at each timestep during the computation. G' is also given in Figure 2. To simulate the operation of σ_3 we emulate the operation of G' in the states of T_{Π}. Note that there is a single non-deterministic choice to be made in G'. This choice is at state g_u if a spike is being removed ($-s$). It would seem that in order to make the correct choice in this situation we need to know the exact number of spikes in σ_3. However, we need only store at most $u + yq$ spikes. The reason for this is that if there are $\geqslant u + yq$ spikes in σ_3, then G' will not enter state g_{u-1} again. To see this, note that σ_3 spikes a maximum of y times using at most q spikes each

time, and so once there are $> u + yq$ spikes the number of spikes in σ_3 will be $> u - 1$ for the remainder of the computation. Thus, T_Π simulates the activity of σ_3 by simulating the operation of G' and encoding at most $u + yq$ spikes in its states.

In this paragraph we explain the operation of T_Π. Following this, we give an analysis of the space complexity of T_Π. T_Π has 4 tapes including an output tape, which is initially blank, and a read only input tape. The tape head on both the input and output tapes is permitted to only move right. Each of the remaining tapes, tapes 1 and 2 simulate the activity of the neurons σ_1 and σ_2, respectively. These tapes record the number of spikes in σ_1 and σ_2. A timestep of Π is simulated as follows: T_Π scans tapes 1 and 2 to determine if there are any applicable rules in σ_1 and σ_2 at that timestep. The applicability of each neural rule in Π is determined by a regular expression and so a decider for each rule is easily implemented in the states of T_Π. Recall from the previous paragraph that the applicability of the rules in σ_3 is already recorded in the states of T_Π. Also, T_Π is non-deterministic and so if more than one rule is applicable in a neuron T_Π simply chooses the rule to simulate in the same manner as Π. Once T_Π has determined which rules are applicable in each of the three neurons at that timestep it changes the encodings on tapes 1 and 2 to simulate the change in the number of spikes in neurons σ_1 and σ_2 during that timestep. As mentioned in the previous paragraph any change in the number of spikes in σ_3 is recorded in the states of T_Π. The input sequence of Π may be given as binary input to T_Π by placing it on its input tape. Also, if at a given timestep a 1 is read on the input tape then T_Π simulates a spike entering the simulated input neuron. At each simulated timestep, if the output neuron σ_3 spikes then a 1 is place on the output tape, and if σ_3 does not spike a 0 is placed on the output tape. Thus the output of Π is encoded on the output tape when the simulation ends.

In a two neuron system each neuron has at most one out-going synapse and so the number of spikes in the system does not increase over time. Thus, the total number of spikes in neurons σ_1 and σ_2 can only increase when σ_3 fires or a spike is sent into the system from the environment. The input is of length n, and so σ_1 and σ_2 receive a maximum of n spikes from the environment. Neuron σ_3 fires no more than y times sending at most r spikes each time to σ_1 and σ_2. Thus the maximum number of spikes in σ_1 and σ_2 during the computation is $n + 2ry$. Using a binary encoding tapes 1 and 2 of T_Π encode the number of spikes in σ_1 and σ_2 using space of $\log_2(n + 2ry)$. As mentioned earlier no space is used to simulate σ_3, and thus T_Π simulates Π using space of $O(\log n)$. $\square$

If we remove the restriction that allows the output neuron to fire only a constant number of times then we may construct a universal system with 3 neurons.

Theorem 3. *Let C be a universal counter machine with 3 counters that completes its computation in time t to give the output value x_3 when given the pair of input values (x_1, x_2). Then there is a universal extended SN P system Π'_C with standard input and generalised output that simulates the computation of C in time $O(t + x_1 + x_2 + x_3)$ and has only 3 neurons.*

Proof. A graph of Π'_C is constructed by removing the output neuron σ_4 from the graph in Figure 1 and making σ_3 the new output neuron by adding a synapse to the environment. The rules for Π'_C are given by the first 3 rows of Table 2. The operation of Π'_C is identical to the operation of Π_C from Theorem 1 with the exception of the new output technique. The output of Π'_C is the time interval between the first and second timesteps where exactly $12h$ spikes are sent out of the output neuron σ_3.   □

From the last paragraph of the proof of Theorem 2 we get Corollary 1.

Corollary 1. *Let Π be any extended SN P system with only 2 neurons and generalised input and output. Then there is a non-deterministic Turing machine T_Π that simulates the computation of Π in space $O(\log n)$ where n is the length of the input to Π.*

6   Conclusion

Our results show that there is no significant trade-off between the time/space complexity and the number of neurons in universal extended SN P systems; our new systems suffer no exponential slow-down when compared with universal SN P systems with a greater number of neurons (see Table 1). The size of a SN P system could also be measured by the number of neural rules in the system. It would be interesting to explore possible trade-offs between the number of neuron, the number of rules and the time/space complexity of universal SN P systems.

References

1. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae **71**(2-3) (2006) 279–308
2. Korec, I.: Small universal register machines. Theoretical Computer Science **168**(2) (1996) 267–301
3. Neary, T.: (2008) Presentation at Computing with Biomolecules (CBM 2008). Available at http://www.emcc.at/UC2008/Presentations/CBM5.pdf .
4. Neary, T.: On the computational complexity of spiking neural P systems. In Calude, C.S., Costa, J.F., Freund, R., Oswald, M., Rozenberg, G., eds.: UC 2008. Volume 5204 of LNCS., Springer (2008) 189–205
5. Neary, T.: A small universal spiking neural P system. In Csuhaj-Varjú, E., Freund, R., Oswald, M., Salomma, K., eds.: International Workshop on Computing with Biomolecules, Vienna, Austrian Computer Society (August 2008) 65–74
6. Neary, T.: A boundary between universality and non-universality in spiking neural P systems. arXiv:0912.0741v1 [cs.CC] (December 2009)
7. Pan, L., Zeng, X.: A note on small universal spiking neural P systems. In Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A., eds.: Tenth Workshop on Membrane Computing (WMC10), Curtea de Argeş, Romania (August 2009) 464–475
8. Păun, A., Păun, G.: Small universal spiking neural P systems. BioSystems **90**(1) (2007) 48–60

9. Păun, G., Pérez-Jiménez, M.J.: Spiking Neural P Systems. Recent Results, Research Topics. Natural Computing Series. In: Algorithmic Bioprocesses. Springer (2009) 273–291
10. Zhang, X., Zeng, X., Pan, L.: Smaller universal spiking neural P systems. Fundamenta Informaticae **87**(1) (2008) 117–136