A SMALL UNIVERSAL SPIKING NEURAL P SYSTEM

Turlough Neary
Boole Centre for Research in Informatics, University College Cork, Ireland.
Email: tneary@cs.may.ie

Abstract
In this work we give a small extended spiking neural P system that is weakly universal. This system is significantly smaller than the smallest strongly universal spiking neural P systems. Strong universality has strict conditions regarding the encoding of input and decoding of output. Weak universality has more relaxed conditions regarding the encoding of input and decoding of output. Păun and Păun [10] gave a strongly universal spiking neural P system with 84 neurons and another that has extended rules with 49 neurons. Subsequently, the number of neurons used for strong universality was reduced from 84 to 67 and from 49 to 41 by Zhang et al. [11]. Here we give a weakly universal spiking neural P system that uses extended rules and has only 12 neurons.

1. Introduction

Spiking neural P systems [2] are quite a new computational model that are a synergy inspired by P systems and spiking neural networks. It has been shown that these systems are computationally universal [2]. Before we discuss results in the area of small universal spiking neural P systems, we note the two different notions of universality given by Korec [5]. Strong universality has strict conditions regarding the encoding of input and decoding of output. Weak universality has more relaxed conditions regarding the encoding of input and decoding of output. Recently, Păun and Păun [10] gave two small strongly universal spiking neural P systems; A spiking neural P system with 84 neurons and an extended spiking neural P system with 49 neurons (and without delay). Păun and Păun conjectured that it is not possible to give a significant decrease in the number of neurons of their two universal systems. Zhang et al. [11] offered such a significant decrease in the number of neurons used to give such small universal systems. They give a strongly universal spiking neural P system with 67 neurons and another, which has extended rules (without delay), with 41 neurons. Here we give an extended spiking neural P system with 12 neurons that is weakly universal and also uses rules without delay.

From a previous result [9] it is known that there exists no universal spiking neural P system that simulates Turing machines in less the exponential time and space. It is a relatively straightforward matter to generalise this result to show that extended spiking neural P systems suffer from the same inefficiencies. It immediately follows that the universal system we present here and those found in [10, 11] have exponential time and space requirements. However, it is possible to give a time efficient spiking neural P system when we allow exhaustive use of rules. A universal extended spiking neural P system with exhaustive use of rules has been given that simulates Turing machines in polynomial time [9]. Furthermore, this system has only 18 neurons. Spiking neural P systems with exhaustive use of rules were originally proved computationally universal by Ionescu et al. [3]. However, the technique used to prove universality suffered from an exponential time overhead.

Using different forms of spiking neural P systems, a number of time efficient (polynomial or constant time) solutions to NP-hard problems have been given [1, 6, 7]. All of these solutions to NP-hard problems rely on families of spiking neural P systems. Specifically, the size of the problem instance determines the number of neurons in the spiking neural P system that solves that particular instance. This is similar to solving problems with circuits families where each input size has a specific circuit that solves it. Ionescu and Sburlan [4] have shown that spiking neural P systems simulate circuits in linear time.

In the next two sections we give definitions for spiking neural P systems and register machines and explain the operation of both. Following this, in Section 4 we give an extended spiking neural P system with 12 neurons that is weakly universal and uses rules without delay.

2. Spiking neural P system

Definition 1 (Spiking neural P systems).
A spiking neural 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. $E/s^b \rightarrow s; d$, *where* E *is a regular expression over* s, $b \geqslant 1$ *and* $d \geqslant 0$,
      ii. $s^e \rightarrow \lambda$, *where* λ *is the empty word,* $e \geqslant 1$, *and for all* $E/s^b \rightarrow s; d$ *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.*

In the same manner as in [10], 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. If the read symbol is 1 then a spike enters the input neuron on that timestep.

A firing rule $r = E/s^b \rightarrow s; d$ 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 forgeting rule $r' = s^e \rightarrow \lambda$ 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 at time t. 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 $E/s^b \rightarrow s; d$, 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. The output is the time between the first and second spike in the output neuron.

An extended spiking neural P system [10] has more general rules of the form $E/s^b \rightarrow s^p; d$, where $b \geqslant p \geqslant 1$. Thus, a synapse in a spiking neural P system with extended rules may transmit more than one spike in a single timestep.

3. Register machines and notions of universality

Definition 2 (Register machine). *A register machine is a tuple $C = (z, r_1, r_m, Q, q_1, q_h)$, where z gives the number of registers, r_1 and r_m are the input and output registers respectively, $Q = \{q_1, q_2, \cdots, q_h\}$ is the set of instructions, $q_1, q_h \in Q$ are the initial and halt instructions, respectively.*

Each register r_j stores a natural number value $x \geqslant 0$. Each instruction q_i is of one of the following two forms $q_i : INC(j)$ or $q_i : DEC(j)q_k$, and is executed as follows:

- $q_i : INC(j)$ increment the value x stored in register r_j by 1 and move to instruction q_{i+1}.
- $q_i : DEC(j)q_k$ if the value x stored in register r_j is greater than 0 then decrement this value by 1 and move to instruction q_{i+1}, otherwise if $x = 0$ move to instruction q_k.

At the beginning of a computation the first instruction executed is q_1. The input to the register machine is initially stored in register r_1. If the register machine's control enters instruction q_h then the computation halts at that timestep. The result of the computation is the value x stored in the output register r_m when the computation halts.

In Section 14.1 of his book, Minsky [8] proves that register machines with only two registers are universal.

Theorem 1 (Minsky [8]). *For any Turing machine T there exists a program machine M_T with just two registers that behaves the same as T (in the sense described in sections 10.1 and 11.2) when started with zero in one register and $2^a 3^m 5^n$ in the other. This machines uses only the operations* $\boxed{'}$ *and* $\boxed{-}$, *assuming that the successor instruction contains the "go" information for the next instruction.*

Minsky refers to register machines as program machines (i.e. M_T satisfies Definition 2). The operations $\boxed{'}$ and $\boxed{-}$ are identical to the instructions INC and DEC, which we defined above. The term "behaves the same as T" is well defined in Minsky's book and the encoding and decoding used by M_T satisfy Korec's notion of weak universality.

Recall that the output of a spiking neural P system Π is the time interval between the first and second spike. If the binary sequence $10^{y-1}1$ is given as input to Π, then the output of the computation is given by $\Pi(y)$. Let $(\phi_0, \phi_1, \phi_2, \ldots)$ be a Gödel enumeration of all unary partial recursive functions. Then we say that a spiking neural P system Π_U is weakly universal if $\phi_x(y) = f(\Pi_U(g(x, y))$ where g and f are recursive functions.

The small universal spiking neural P systems of Păun and Păun [10] and of Zhang et al. [11] simulate the 22 instruction strongly universal register machine of Korec [5]. In addition these spiking neural P systems satisfy the strict encoding and decoding requirements of Korec's [5] notion of strong universality. For weak universality it is sufficient to have encoding and decoding functions that are recursive. For this reason we note only that both the encoding ($2^a 3^m 5^n$) and decoding functions for M_T in Theorem 1 are recursive. It is not necessary for our purposes to discuss the encoding and decoding for M_T in more detail. The extended spiking neural P systems we give in the next section simulates a weakly universal 2 register machine. Thus, our system is also weakly universal.

4. A small universal spiking neural P system

In this section we give our small universal spiking neural P system. We prove the universality of our system by showing that it simulates a weakly universal register machine C_2 that has only two registers. Using Minsky's proof of Theorem 1 finding such a register machine is relatively straightfroward. As noted at the end of the previous section the encoding and decoding for such register machines are recursive and thus it is not necessary to concern ourselves with other details of C_2 here.

Theorem 2. *Let C_2 be a weakly universal register machine with 2 registers. Then there is a weakly universal extended spiking neural P system Π_{C_2} that simulates the computation of C_2 and has only 12 neurons.*

Proof. Let $C_2 = (2, r_1, r_1, Q, q_1, q_h)$ where $Q = \{q_1, q_2, \cdots, q_h\}$. Our spiking neural P system Π_{C_2} is given by Figure 1, and Tables 1 and 2. The algorithm given for Π_{C_2} is deterministic.

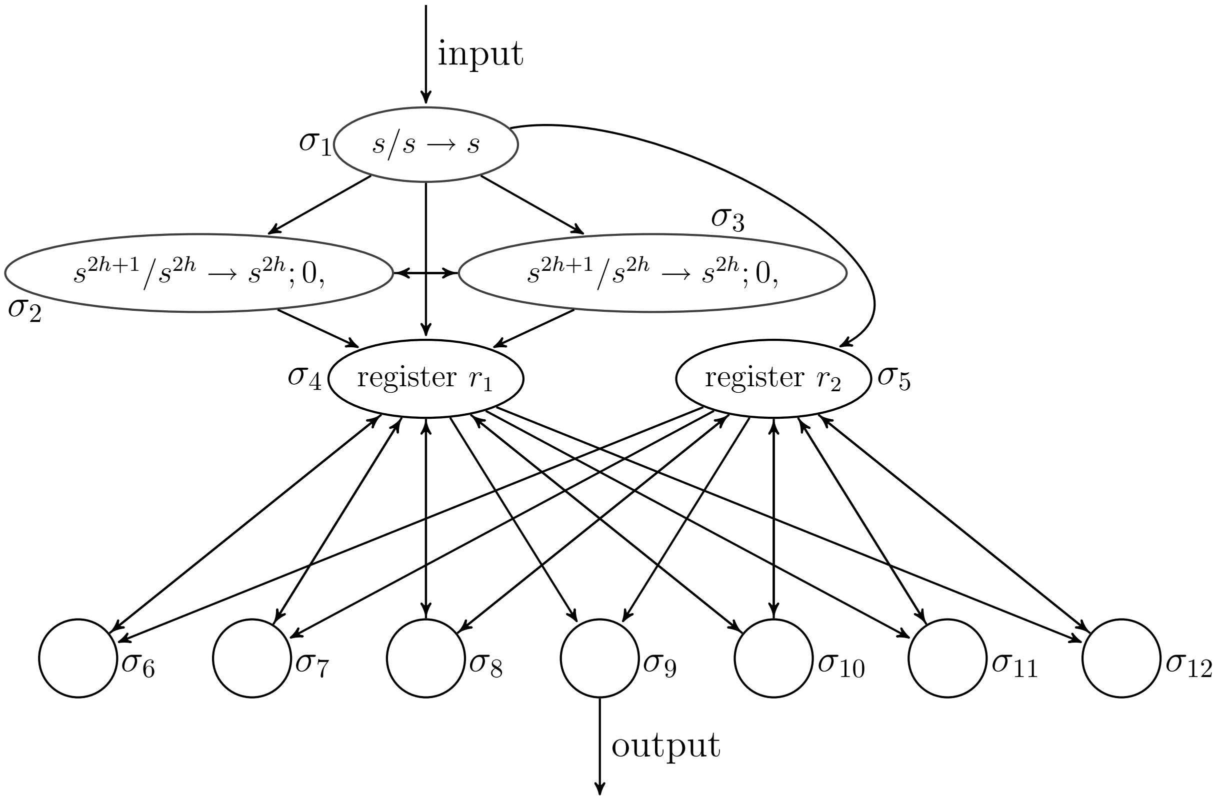

Figure 1: Universal extended spiking neural P system Π_{C_2}.

Encoding of a configuration of C_2 and reading in input to Π_{C_2}. A configuration of C_2 is stored as spikes in the neurons of Π_{C_2}. The next instruction q_i to be executed is stored in each of the neurons σ_4 and σ_5 as $2(h+i)$ spikes. Let x_1 and x_2 be the values stored in registers r_1 and r_2, respectively. Then x_1 and x_2 are stored as $4hx_1$ and $4hx_2$ spikes in neurons σ_4 and σ_5, respectively.

The input to Π_{C_2} is read into the system via the input neuron σ_1 as shown in Figure 1. If the input to C_2 is x then the binary sequence $w = 10^{x-1}1$ is read in via the input neuron σ_1. In Figure 1 σ_2, σ_3, σ_4, and σ_5 initially contain $2h$ spikes before the computation begins. Neurons σ_2 and σ_3 receive the first spike from σ_1 at time t_1 and the second spike at time t_{x+1}. Thus, σ_2 and σ_3 fire on every timestep between times t_1 and t_{x+2} and send a total of $4hx$ spikes to σ_4. Therefore, when σ_1 fires for the second time, after $x + 2$ timesteps neuron σ_4 contains $4hx + 2(h + 1)$ spikes, the encoding $(4hx)$ of the input x and the encoding $(2(h + 1))$ of the initial instruction q_1. Note that at time t_{x+2} neuron σ_5 also contains the encoding $(2(h+1))$ of the initial instruction q_1.

$\mathbf{\Pi_{C_2}}$ **simulating** $\mathbf{q_i : INC(1)}$. Let there be x_1 spikes in register r_1 and x_2 spikes in register r_2. Then the simulation of $q_i : INC(1)$ begins at time t_j with $4hx_1 + 2(h + i)$ spikes in σ_4 and $4hx_2 + 2(h + i)$ spikes in σ_5. We explain the simulation by giving the number of spikes in each

neuron	rules
σ_1	$s/s \rightarrow s;0$
σ_2,σ_3	$s^{2h+1}/s^{2h} \rightarrow s^{2h};0$
σ_4	$s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s^{2(h+i)-1};0$ if $q_i : INC(1) \in \{Q\}$
	$s^{4h+2(h+i)}(s^{4h})^*/s^{4h+2(h+i)} \rightarrow s^{2(h+i)};0$ if $q_i : DEC(1) \in \{Q\}$
	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2(h+i)-1};0$ if $q_i : DEC(1) \in \{Q\}$
	$s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s;0$ if $q_i : INC(2) \in \{Q\}$ or $q_i : DEC(2) \in \{Q\}$
	$s^{6h+3}(s^{4h})^*/s^{6h} \rightarrow s;0,\quad s^{2h+3} \rightarrow \lambda,\quad s^7(s^{4h})^*/s^{4h} \rightarrow s^{2h};0,\quad s^3/s^3 \rightarrow s;0,$
σ_5	$s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s^{2(h+i)-1};0$ if $q_i : INC(2) \in \{Q\}$
	$s^{4h+2(h+i)}(s^{4h})^*/s^{4h+2(h+i)} \rightarrow s^{2(h+i)};0$ if $q_i : DEC(2) \in \{Q\}$
	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2(h+i)-1};0$ if $q_i : DEC(2) \in \{Q\}$
	$s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s;0$ if $q_i : INC(1) \in \{Q\}$ or $q_i : DEC(1) \in \{Q\}$

Table 1: This table gives the rules in each of the neurons σ_1 to σ_5 of Π_{C_2}.

neuron and the rule that is to be applied in each neuron at time t. For example at time t_j we have

$$\begin{aligned} t_j : \sigma_4 &= 4hx_1 + 2(h+i), & s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s^{2(h+i)-1};0,\\ \sigma_5 &= 4hx_2 + 2(h+i), & s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s;0. \end{aligned}$$

where on the left $\sigma_k = y$ gives the number y of spikes in neuron σ_k at time t_j and on the right is the next rule that is to be applied at time t_j if there is an applicable rule at that time. Thus, from Figure 1, when we apply the rule $s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s^{2(h+i)-1};0$ in neuron σ_4 and the rule $s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s;0$ in σ_5 at time t_j we get

$$\begin{aligned} t_{j+1} : \sigma_4 &= 4hx_1, & \\ \sigma_5 &= 4hx_2, & \\ \sigma_6,\sigma_7,\sigma_8 &= 2(h+i), & s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2h};0,\\ \sigma_9,\sigma_{11},\sigma_{12} &= 2(h+i), & s^{2(h+i)} \rightarrow \lambda,\\ \sigma_{10} &= 2(h+i), & s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2(i+1)};0, \end{aligned}$$

$$\begin{aligned} t_{j+2} : \sigma_4 &= 4h(x_1+1) + 2(h+i+1),\\ \sigma_5 &= 4hx_2 + 2(h+i+1). \end{aligned}$$

At time t_{j+2} the simulation of $q_i : INC(1)$ is complete. The encoded register value has been incremented by increasing it from $4hx_1$ to $4h(x_1+1)$. The encoding $2(h+i+1)$ of the next instruction q_{i+1} has been established.

neuron	rules
σ_6, σ_7	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2h}; 0$ if $q_i : INC(1) \in \{Q\}$
	$s^{2(h+i)} \rightarrow \lambda$ if $q_i : INC(1) \notin \{Q\}$
	$s^{2(h+i)+1} \rightarrow \lambda, \quad s^{2h} \rightarrow \lambda, \quad s \rightarrow \lambda$
σ_8	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2h}; 0, \quad s^{2(h+i)+1}/s^{2(h+i)+1} \rightarrow s^{2h}; 0, \quad s^{2h} \rightarrow \lambda, \quad s \rightarrow \lambda$
σ_9	$s^{2(h+i)} \rightarrow \lambda, \quad s^{2(h+i)+1} \rightarrow \lambda, \quad s^{2h} \rightarrow \lambda, \quad s/s \rightarrow s; 0$
σ_{10}	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2(i+1)}; 0$ if $q_i : INC \in \{Q\}$ and $q_{i+1} \neq q_h$
	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^3; 0$ if $q_i : INC \in \{Q\}$ and $q_{i+1} = q_h$
	$s^{2(h+i)+1}/s^{2(h+i)+1} \rightarrow s^{2(i+1)}; 0$ if $q_i : DEC \in \{Q\}$ and $q_{i+1} \neq q_h$
	$s^{2(h+i)+1}/s^{2(h+i)+1} \rightarrow s^3; 0$ if $q_i : DEC \in \{Q\}$ and $q_{i+1} = q_h$
	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2k}; 0$ if $q_i : DECq_k \in \{Q\}$ and $q_k \neq q_h$
	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^3; 0$ if $q_i : DECq_k \in \{Q\}$ and $q_k = q_h$
	$s^{2h} \rightarrow \lambda, \quad s \rightarrow \lambda$
σ_{11}, σ_{12}	$s^{2(h+i)}/s^{2(h+i)} \rightarrow s^{2h}; 0$ if $q_i : INC(2) \in \{Q\}$
	$s^{2(h+i)} \rightarrow \lambda$ if $q_i : INC(2) \notin \{Q\}$
	$s^{2(h+i)+1} \rightarrow \lambda, \quad s^{2h} \rightarrow \lambda, \quad s \rightarrow \lambda$

Table 2: This table gives the rules in each of the neurons σ_6 to σ_{12} of Π_{C_2}.

$\mathbf{\Pi_{C_2}}$ **simulating** $\mathbf{q_i : DEC(1)q_k}$. As above the simulation begins at time t_j giving

$$t_j : \sigma_4 = 4hx_1 + 2(h+i), \qquad s^{4h+2(h+i)}(s^{4h})^*/s^{4h+2(h+i)} \rightarrow s^{2(h+i)}; 0,$$
$$\sigma_5 = 4hx_2 + 2(h+i), \qquad s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} \rightarrow s; 0,$$

$$t_{j+1} : \sigma_4 = 4h(x_1 - 1),$$
$$\sigma_5 = 4hx_2,$$
$$\sigma_6, \sigma_7, \sigma_9, \sigma_{11}, \sigma_{12} = 2(h+i) + 1, \qquad s^{2(h+i)+1} \rightarrow \lambda,$$
$$\sigma_8 = 2(h+i) + 1, \qquad s^{2(h+i)+1}/s^{2(h+i)+1} \rightarrow s^{2h}; 0,$$
$$\sigma_{10} = 2(h+i) + 1, \qquad s^{2(h+i)+1}/s^{2(h+i)+1} \rightarrow s^{2(i+1)}; 0,$$

$$t_{j+2} : \sigma_4 = 4h(x_1 - 1) + 2(h+i+1),$$
$$\sigma_5 = 4hx_2 + 2(h+i+1).$$

At time t_{j+2} the simulation of $q_i : DEC(1)q_k$ is complete. The encoded register value has been decremented by decreasing it from $4hx_1$ to $4h(x_1 - 1)$. The encoding $2(h+i+1)$ of the next instruction q_{i+1} has been established. In the above example we assume that register r_1 has value $x_1 > 0$. If $x_1 = 0$ then we get the following

$$\begin{aligned}
t_j : \sigma_4 &= 2(h+i), & s^{2(h+i)}/s^{2(h+i)} &\rightarrow s^{2(h+i)-1}; 0,\\
\sigma_5 &= 4hx_2 + 2(h+i), & s^{2(h+i)}(s^{4h})^*/s^{2(h+i)} &\rightarrow s; 0,
\end{aligned}$$

$$\begin{aligned}
t_{j+1} : \sigma_5 &= 4hx_2,\\
\sigma_6, \sigma_7, \sigma_9, \sigma_{11}, \sigma_{12} &= 2(h+i), & s^{2(h+i)} &\rightarrow \lambda,\\
\sigma_8 &= 2(h+i), & s^{2(h+i)}/s^{2(h+i)} &\rightarrow s^{2h}; 0,\\
\sigma_{10} &= 2(h+i), & s^{2(h+i)}/s^{2(h+i)} &\rightarrow s^{2k}; 0,
\end{aligned}$$

$$\begin{aligned}
t_{j+2} : \sigma_4 &= 2(h+k),\\
\sigma_5 &= 4hx_2 + 2(h+k).
\end{aligned}$$

Note that at time t_{j+2}, when the simulation is complete, the encoding $2(h+k)$ of the next instruction q_{i+1} has been established.

Halting. The halt instruction q_h is encoded as $2h+3$ spikes. Thus, if C_2 enters the halt instruction q_h we get the following

$$\begin{aligned}
t_j : \sigma_4 &= 4hx_1 + 2h + 3, & s^{6h+3}(s^{4h})^*/s^{6h} &\rightarrow s; 0,\\
\sigma_5 &= 4hx_2 + 2h + 3,
\end{aligned}$$

$$\begin{aligned}
t_{j+1} : \sigma_4 &= 4h(x_1 - 1) + 3, & s^7(s^{4h})^*/s^{4h} &\rightarrow s^{2h}; 0,\\
\sigma_5 &= 4hx_2 + 2h + 3,\\
\sigma_6, \sigma_7, \sigma_8, \sigma_{10}, \sigma_{11}, \sigma_{12} &= 1, & s &\rightarrow \lambda,\\
\sigma_9 &= 1, & s/s &\rightarrow s; 0,
\end{aligned}$$

$$\begin{aligned}
t_{j+2} : \sigma_4 &= 4h(x_1 - 2) + 3, & s^7(s^{4h})^*/s^{4h} &\rightarrow s^{2h}; 0,\\
\sigma_5 &= 4hx_2 + 2h + 3,\\
\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}, \sigma_{12} &= 2h, & s^{2h} &\rightarrow \lambda.
\end{aligned}$$

The rule $s^7(s^{4h})^*/s^{4h} \rightarrow s^{2h}; 0$, is applied a further $x_1 - 2$ times in σ_9 until we get

$$\begin{aligned}
t_{j+x_1} : \sigma_4 &= 3, & s^3/s^3 &\rightarrow s; 0,\\
\sigma_5 &= 4hx_2 + 2h + 3,\\
\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \sigma_{11}, \sigma_{12} &= 2h, & s^{2h} &\rightarrow \lambda,
\end{aligned}$$

$$\begin{aligned}
t_{j+x_1+1} : \sigma_5 &= 4hx_2 + 2h + 3,\\
\sigma_6, \sigma_7, \sigma_8, \sigma_{10}, \sigma_{11}, \sigma_{12} &= 1, & s &\rightarrow \lambda,\\
\sigma_9 &= 1, & s/s &\rightarrow s; 0.
\end{aligned}$$

As usual the output is the time interval between the first and second spikes that are sent out of the output neuron. Note from above that the output neuron σ_9 fires for the first time at timestep t_{j+1} and for the second time at timestep t_{j+x_1+1}. Thus, the output of Π_{C_2} is x_1 the value of the output register r_1 when C_2 enters the halt instruction q_h. Note that if $x_1 = 0$ then the rule $s^{2h+3} \rightarrow \lambda$ is executed at timestep t_j and thus no spikes will be sent out of the output neuron.

We have shown how to simulate arbitrary instructions of the form $q_i : INC(1)$ and $q_i : DEC(1)q_k$. Instructions of the form $q_i : INC(2)$ and $q_i : DEC(2)q_k$, which operate on register r_2, are simulated in a similar manner Immediately following the simulation of an instruction Π_{C_2} is configured to simulate the next instruction. Thus, Π_{C_2} simulates the computation of C_2. □

The algorithm used by Π_{C_2} could be easily modified to simulate strongly universal register machines thus giving small extended spiking neural P systems that are strongly universal. Each additional register would require an extra three neurons. Also, if we wish to simulate a register machine that has two input registers we would require a further three neurons.

The reachability question for spiking neural P systems is as follows; Given a configuration c_x of a spiking neural P systems does it ever enter a configuration c_y. It is worth noting that using the results in Theorem 2 smaller spiking neural P systems with undecidable reachability questions may be given. Such systems may be given by removing the output neurons and the neurons for initialising the system (the input module) from Π_{C_2}. Thus, there exist spiking neural P systems with 8 neurons which have undecidable reachability questions.

Acknowledgements

The author would like to thank the anonymous reviewers and Professor Rudolf Freund for their careful reading of the paper, and for their helpful suggestions and comments. The author is funded by Science Foundation Ireland Research Frontiers Programme grant number 07/RFP/CSMF641.

References

[1] H. Chen, M. Ionescu, and T. Ishdorj. On the efficiency of spiking neural P systems. In M.A. Gutiérrez-Naranjo et al., editor, *Proceedings of Fourth Brainstorming Week on Membrane Computing*, pages 195–206, Sevilla, Feb. 2006.

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

[3] M. Ionescu, G. Păun, and T. Yokomori. Spiking neural P systems with exhaustive use of rules. *International Journal of Unconventional Computing*, 3(2):135–153, 2007.

[4] M. Ionescu and D. Sburlan. Some applications of spiking neural P systems. In George Eleftherakis et al., editor, *Proceedings of the Eighth Workshop on Membrane Computing*, pages 383–394, Thessaloniki, June 2007.

[5] I. Korec. Small universal register machines. *Theoretical Computer Science*, 168(2):267–301, Nov. 1996.

[6] A. Leporati, C. Zandron, C. Ferretti, and G. Mauri. On the computational power of spiking neural P systems. In M.A. Gutiérrez-Naranjo et al., editor, *Proceedings of the Fifth Brainstorming Week on Membrane Computing*, pages 227–245, Sevilla, Jan. 2007.

[7] A. Leporati, C. Zandron, C. Ferretti, and G. Mauri. Solving numerical NP-complete problems with spiking neural P systems. In George Eleftherakis et al., editor, *Proceedings of the Eighth Workshop on Membrane Computing*, pages 405–423, Thessaloniki, June 2007.

[8] M. Minsky. *Computation, finite and infinite machines.* Prentice-Hall, 1967.

[9] T. Neary. On the computational complexity of spiking neural P systems. In *7th International Conference on Unconventional Computation (UC 2008)*, volume 5204 of *LNCS*, pages 190–206, Vienna, Aug. 2008. Springer.

[10] A. Păun and G. Păun. Small universal spiking neural P systems. *BioSystems*, 90(1):48–60, 2007.

[11] X. Zhang, X. Zeng, and L. Pan. Smaller universal spiking neural P systems. In submission, 2008.