# A universal spiking neural P system with 11 neurons*

Turlough Neary

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

**Abstract.** In this work we offer a significant improvement on the previous smallest spiking neural P system. Păun and Păun [3] gave a universal spiking neural P system with 84 neurons. Subsequently, Zhang et al. [18] reduced the number of neurons used to give universality to 67. Here we give a small universal spiking neural P system that has only 11 neurons and uses rules without delay.

## 1 Introduction

Spiking neural P systems (SN P systems) [4] 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 [4].

In this work we are concerned with the search for the smallest universal SN P system, where size is the number of neurons. Păun and Păun [3] initiated this search by giving an SN P system with 84 neurons. Zhang et al. [18] improved on this result to give a universal SN P system with 67 neurons. Here we give a small universal SN P system that has only 11 neurons and uses rules without delay.

The above mentioned SN P systems in [3, 18] simulate Turing machines in double-exponential time. The 11-neuron system we present here simulates Turing machines with an exponential time overhead. In other work [12], it is shown that universal SN P systems require exponential time to simulate Turing machines, and so significant improvement on the simulation time for our system is not possible. The time/space complexity of small universal SN P systems[1] and a brief history of the area are given in Table 1.

In their paper containing the 84-neuron system, Păun and Păun [3] state that a significant decrease on the number of neurons of their universal SN P system is improbable (also stated in [8]). The dramatic improvement on the 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 machine being simulated. With the exception of the systems in [12, 13], all of the SN P systems given in Table 1

* Turlough Neary is funded by Science Foundation Ireland Research Frontiers Programme grant number 07/RFP/CSMF641.

[1] In a similar Table given in [10] the time/space complexity for a number of the systems is incorrect due to an error copied from Korec's paper [6]. For more see Section 3.

**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. † The 18 neuron system is not explicitly given in [11]; it was presented at [14] and is easily derived from the other system in [11]. ‡ A more generalised output technique is used. * The smallest possible system of its kind, where size is the number of neurons. 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. [17] |
| 18 | polynomial/exponential | extended | yes | Neary [12] |
| 10 | linear/exponential | extended | yes | Neary [13] |
| 49 | double-exponential | extended | no | Păun and Păun [3] |
| 41 | double-exponential | extended | no | Zhang et al. [18] |
| 18 | exponential | extended | no | Neary [11, 14]† |
| 12 | double-exponential | extended | no | Neary [11] |
| 4 | exponential | extended | no | Neary [10]* |
| 3 | exponential | extended | no | Neary [10]*‡ |
| 84 | double-exponential | standard | no | Păun and Păun [3] |
| 67 | double-exponential | standard | no | Zhang et al. [18] |
| 17 | exponential | standard | no | Neary [9] |
| **11** | **exponential** | **standard** | **no** | **Section 4** |

were proved universal by simulating counter machines. The size of previous small universal systems [3, 18] were dependent on the number of instructions in the counter machine being simulated. In our system each unique counter machine instruction is encoded as a unique number of spikes, and thus the size of our SN P system 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 [11].

## 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* $\sigma_i$*,*
   (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* $\lambda$ *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.*

A firing rule $r = E/s^b \rightarrow s; d$ is applicable in a neuron $\sigma_i$ if there are $j \geqslant b$ spikes in $\sigma_i$ 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 $\sigma_i$ fires a spike is sent to each neuron $\sigma_j$ for every synapse $(i, j)$ in $\Pi$. Also, the neuron $\sigma_i$ remains closed and does not receive spikes until time $t + d$ and no other rule may execute in $\sigma_i$ until time $t + d + 1$. A forgetting rule $r' = s^e \rightarrow \lambda$ is applicable in a neuron $\sigma_i$ if there are exactly $e$ spikes in $\sigma_i$. 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 2(b)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 SN P system we present in this work use rules without delay, and thus in the sequel we write rules as $E/s^b \rightarrow s$. Also, if in a rule $E = s^b$ then we write the rule as $s^b \rightarrow s$.

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 $\sigma_1$. 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. In this work the output of an SN P system $\Pi$ is the time between the first and second timesteps a firing rule is applied in the output neuron.

## 3 Counter machines, universality and time/space complexity

**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 $v_j \geqslant 0$. Each instruction $q_i$ is of one of the following two forms:

– $q_i : INC(j), q_l$ increment the value $v_j$ stored in counter $c_j$ by 1 and move to instruction $q_l$.
– $q_i : DEC(j), q_l, q_k$ if the value $v_j$ stored in counter $c_j$ is greater than 0 then decrement this value by 1 and move to instruction $q_l$, otherwise if $v_j = 0$ 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 $v_m$ stored in the output counter $c_m$ when the computation halts.

We now consider some different notions of universality. Korec [6] gives universality definitions that describe some counter machines as weakly universal and other counter machines as strongly universal.

**Definition 3 (Korec [6]).** *A register machine $M$ will be called strongly universal if there is a recursive function $g$ such that for all $x, y \in \mathbb{N}$ we have $\phi_x(y) = \Phi^2_M(g(x), y)$.*

Here $\phi_x$ is the $x^{th}$ unary partial recursive function in a Gödel enumeration of all unary partial recursive functions. Also, $\Phi^2_M(g(x), y)$ is the value stored in the output counter at the end of a computation when $M$ is started with the values $g(x)$ and $y$ in its input counters. Korec's definition insists that the value $y$ should not be changed before passing it as input to $M$. However, if we consider computing an $n$-ary function with a Korec-strong universal counter machine then it is clear that $n$ arguments must be encoded as a single input $y$. Many Korec-strong universal counter machines would not satisfy a definition where the function $\phi_x$ in Definition 3 is replaced with an $n$-ary function with $n > 1$. For example, let us give a new definition where we replace the equation "$\phi_x(y) = \Phi^2_M(g(x), y)$" with the equation "$\phi^n_x(y_1, y_2, \ldots, y_n) = \Phi^{n+1}_M(g(x), y_1, y_2, \ldots, y_n)$" in Definition 3. Note that for any counter machine $M$ with $r$ counters, if $r \leqslant n$ then $M$ does not satisfy this new definition. Perhaps when one uses this definition of universality it would be more appropriate to refer to it as strongly universal for unary partial recursive functions instead of simply strongly universal. In fact towards the end of his paper Korec [6] sketches how to construct a counter machine that is strongly universal for $n$-ary partial recursive functions. It is worth noting that Korec's definition of strong universality deals with input and output only and is not concerned with the time/space efficiency of the computation. As we will see later, the strongly universal counter machines of Korec given in [6] suffer exponential slowdown when simulating counter machines.

In [6] Korec also gives a number of other definitions of universality. If the equation $\phi_x(y) = \Phi^2_M(g(x), y)$ in Definition 3 above is replaced with any one of the equations $\phi_x(y) = \Phi^1_M(g_2(x, y))$, $\phi_x(y) = f(\Phi^2_M(g(x), y))$ or $\phi_x(y) = f(\Phi^1_M(g_2(x, y)))$ then Korec refers to counter machine $M$ as weakly universal. Korec gives another definition where the equation $\phi_x(y) = \Phi^2_M(g(x), y)$ in Definition 3 is replaced with the equation $\phi_x(y) = f(\Phi^2_M(g(x), h(y)))$. However, he does not include this definition in his list of weakly universal machines even though the equation $\phi_x(y) = f(\Phi^2_M(g(x), h(y)))$ allows for a more relaxed encoding than the equation $\phi_x(y) = f(\Phi^2_M(g(x), y))$ and thus gives a weaker form of universality.

In [3] Korec's notion of strong universality was adopted for SN P systems[2] as follows: An SN P system $\Pi$ is strongly universal if $\Pi(10^{y-1}10^{x-1}1) = \phi_x(y)$ for all $x$ and $y$ (here if $\phi_x(y)$ is undefined so too is $\Pi(10^{y-1}10^{x-1}1)$). Korec

[2] Note that a formal definition of this notion was not explicitly given in [3].

noted that his reason for distinguishing strong universality for counter machines was that becuase they take natural numbers as input, no encoding was necessary when computing unary partial recursive functions. This is clearly not the case for SN P systems as some encoding will always be necessary when computing such functions. For this and the other reasons mentioned above, we rely on time/space complexity analysis to compare small SN P systems and their encodings (see Table 1).

The below definition of universality allows for recursive encoding and decoding functions. It is common to allow such encoding and decoding functions in definition of universality (for example see [1, 15]).

**Definition 4.** *An SN P system $\Pi$ is universal if there are recursive functions $g$ and $f$ such that for all $x, y \in N$ for which $\phi_x(y)$ is defined we have $\phi_x(y) = f(\Pi(g(x, y)))$.*

In the definition above, the function $g$ maps the pair $(x, y)$ to a binary input sequence to be read into $\Pi$. Also, for all values of $x$ and $y$ for which $\phi_x(y)$ is defined, $\Pi(g(x, y))$ gives the output sequence of $\Pi$ when started on the input $g(x, y)$, and if $\phi_x(y)$ is undefined so too is $\Pi(g(x, y))$. Finally, the function $f$ maps the output sequence $\Pi(g(x, y))$ to a natural number. With the exception of the 18 and 10-neuron systems with exhaustive use of rules, all of the SN P system in Table 1 satisfy Definition 4. The 12-neuron system in Table 1 is the only system that (using current algorithms) must encoded $x$ and $y$ together. The other systems in Table 1 may use separate encoding functions to encode $x$ and $y$. Also, excluding the 12-neuron system and the 18-neuron system with exhaustive use of rules, all of the other systems in Table 1 use input and output encodings that are linear in $y$ and $\phi_x(y)$, respectively.

The 84, 67, 49, 18, 17, and 4-neuron systems from Table 1 satisfy the notion of strong universality mentioned above. The 11-neuron system we give in Theorem 1 takes the sequence $10^{4hx}10^{4hy}1$ as input (where $h$ is a constant) and thus does not satisfy the above notion of strong universality. It is interesting to note that such a simple generalisation of the input encoding allows us to significantly reduce the number of neurons used to give a universal system.

For the purposes of explanation, in this work we say that an abstract machine $M'$ simulates another abstract machine $M$ if $M(w) = g(M'(f(w)))$, where $M(w)$ is the output of $M$ when started on the input $w$, $M'(f(w))$ is the output of $M'$ when started on the input $f(w)$, $f$ and $g$ are the encoding and decoding functions, and if $M(w)$ is undefined so too is $M'(f(w))$. Note that we assume that appropriate restrictions are placed on $f$ and $g$ so that they are not to powerful (for example, if $M$ is universal then we insist that they are recursive). In addition to the above, we will say that $M'$ simulates $M$ with an exponential time (space) overhead if for all values of $w$ for which $M$ is defined, $M'$ completes it computation in time (space) $O(2^u)$, where $u$ is the time (space) used by $M$ to complete its computation. Linear $O(u)$, polynomial $O(u^2)$, double-exponential $O(2^{2^u})$ and triple-exponential $O(2^{2^{2^u}})$ simulation overheads are defined in a similar manner. In this work the space used by an SN P system is the maximum number of spikes in the system at any timestep during its computation.

In [6] Korec gives a number of universal counter machines that use very few instructions. At the end of his paper Korec states that his universal counter machine with 32 instructions simulate R3-machines in linear time. This is incorrect and is possibly a typographical error (he may have intended to write "R3a-machines" instead of "R3-machines"). His 32-instruction machine simulates R3a-machines with a linear time overhead, and his R3a-machines simulate counter machines with an exponential time overhead. To see this note that he proves R3a-machines universal by showing that they compute unary partial recursive functions as follows: First the R3a-machine computes $2^y$ from its initial input value $y$. Next the R3a-machine computes the value $2^{f(y)}$ using only 2 of its 3 counters. Finally, the R3a-machine computes the output $f(y)$ from $2^{f(y)}$. Using current algorithms 2-counter machines are exponential slower then 3-counter machines [16], and thus Korec's R3a-machines and 32-instruction machine are exponential slower than 3-counter machines. It is known that counter machines simulate Turing machines with an exponential time overhead [2], and thus Korec's 32-instruction machine simulates Turing machines in double-exponential time. All of the SN P systems in [3, 18, 17] simulate the 23-instruction (the halt instruction is included here) universal counter machine given by Korec in [6]. This 23-instruction machine uses the same algorithm as the 32-instruction machine, and thus also suffers from a double-exponential time overhead when simulating Turing machines. The SN P systems given in [3, 18] simulate Korec's 23-instruction machine with linear time and space overheads, and so have double-exponential time and space overheads when simulating Turing machines. The SN P system given in [17] simulates Korec's 23-instruction machine with linear time and exponential space overheads, and thus has double-exponential time and triple-exponential space overheads when simulating Turing machines. We end our complexity analysis by noting that many of the above simulation overheads (including those of Korec's small counter machines) could be exponentially improved by showing that Korec's R3a-machines simulate 3-counter machines in polynomial time.

In addition to looking at the number of neurons, we could also look at other parameters when considering the size of universal SN P systems. For example, how many different types of rules are used by the system. The systems in [3, 18] do not use as many different types of rules as those in [9, 10] and the 11-neuron system we give in Section 4. However, the 17-neuron system in [9] and the 11-neuron system use a more restricted from of rule than the other *standard* SN P system in Table 1 as they use rules without delay. For the simplification of other aspects of SN P systems see [5, 7], where the authors investigate the computational power of simplified forms of SN P systems.

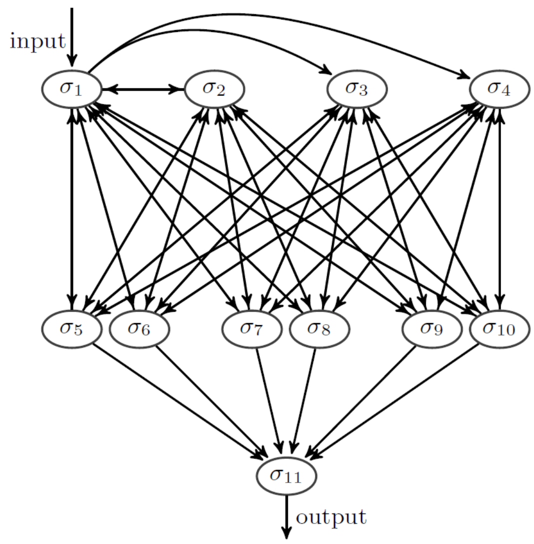


**Fig. 1.** Universal SN P system $\Pi_C$. Each oval labeled $\sigma_i$ is a neuron. An arrow going from neuron $\sigma_i$ to neuron $\sigma_j$ illustrates a synapse $(i, j)$.

## 4 A Small Universal SN P System

**Theorem 1.** *Let $C$ be a universal counter machine with 3 counters and $h$ instructions that completes its computation in time $t$ to give the output value $x_3$ when given the input $(x_1, x_2)$. Then there is a universal SN P system $\Pi_C$ that simulates the computation of $C$ in time $O(ht + hx_1 + hx_2 + x_3)$ and has only 11 neurons.*

*Proof.* Let $C = (3, \{c_1, c_2\}, c_3, Q, q_1, q_h)$ where $Q = \{q_1, q_2, \cdots, q_h\}$. The SN P system $\Pi_C$ is given by Figure 1 and Tables 3 and 4. As a complement to Figure 1, Table 2 may be used to identify all the synapses in $\Pi_C$. The algorithm given for $\Pi_C$ is deterministic.

**Encoding of a configuration of $C$ and reading input into $\Pi_C$.** A configuration of $C$ is stored as spikes in the neurons of $\Pi_C$. The next instruction $q_i$ to be executed is stored in each of the neurons $\sigma_5$, $\sigma_6$, $\sigma_7$, $\sigma_8$, $\sigma_9$, and $\sigma_{10}$ as $8i + 1$ spikes. Let $x_1$, $x_2$ and $x_3$ be the values stored in counters $c_1$, $c_2$ and $c_3$, respectively. Then the value $x_1$ is stored as $16h(x_1 + 1)$ spikes in neurons $\sigma_5$ and $\sigma_6$, $x_2$ is stored as $16h(x_2 + 1)$ spikes in $\sigma_7$ and $\sigma_8$, and $x_3$ is stored as $16h(x_3 + 1)$ spikes in $\sigma_9$ and $\sigma_{10}$.

The input to $\Pi_C$ is read into the system via the input neuron $\sigma_1$ (see Figure 1). If $C$ begins its computation with the input values $x_1$ and $x_2$ in counters $c_1$ and $c_2$, respectively, then the binary sequence $w = 10^{4hx_1}10^{4hx_2}1$ is read in via the input neuron $\sigma_1$. Thus, $\sigma_1$ receives a spike from the environment at times $t_1$, $t_{4hx_1+2}$ and $t_{4hx_1+4hx_2+3}$. 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 $\sigma_1$ contains 2 spikes, $\sigma_5$ and $\sigma_6$ contain $16h+3$ spikes, and $\sigma_7$, $\sigma_8$, $\sigma_9$ and $\sigma_{10}$ contain $16h+11$ spikes. Thus, when $\sigma_1$ receives it first spike at time $t_1$ we have

$$\begin{array}{lll}
t_1: & \sigma_1 : 3, & s^3/s^2 \to s, \\
 & \sigma_5, \sigma_6 : 16h+3, & \\
 & \sigma_7, \sigma_8, \sigma_9, \sigma_{10} : 16h+11, &
\end{array}$$

where on the left $\sigma_j : k$ gives the number $k$ of spikes in neuron $\sigma_j$ at time $t_i$ and on the right is the next rule that is to be applied at time $t_i$ if there is an applicable rule at that time. Thus from Figure 1, when we apply the rule $s^3/s^2 \to s$ in neuron $\sigma_1$ at time $t_1$ we get

$$\begin{array}{llr}
t_2: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 1, & s \to s, \\
 & \sigma_5, \sigma_6 : 16h+4, & \\
 & \sigma_7, \sigma_8, \sigma_9, \sigma_{10} : 16h+12, & s^{16h+12}/s^{10} \to s, \\
 & & \\
t_3: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 5, & s^5 \to s, \\
 & \sigma_5, \sigma_6 : 16h+8, & \\
 & \sigma_7, \sigma_8, \sigma_9, \sigma_{10} : 16h+6, & s^{16h+6}/s^4 \to s, \\
 & \sigma_{11} : 4, & s^4 \to \lambda, \\
 & & \\
t_4: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 5, & s^5 \to s, \\
 & \sigma_5, \sigma_6 : 16h+12, & \\
 & \sigma_7, \sigma_8, \sigma_9, \sigma_{10} : 16h+6, & s^{16h+6}/s^4 \to s, \\
 & \sigma_{11} : 4, & s^4 \to \lambda.
\end{array}$$

Neurons $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$ fire on every timestep between times $t_2$ and $t_{4hx_1+2}$

**Table 2.** The synapses of the SN P system $\Pi_C$. Each origin neuron $\sigma_i$ and target neuron $\sigma_j$ that appear on the same row have a synapse going from $\sigma_i$ to $\sigma_j$.

| origin neurons | target neurons |
|---|---|
| $\sigma_1$ | $\sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}$ |
| $\sigma_2,$ | $\sigma_1, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}$ |
| $\sigma_3, \sigma_4$ | $\sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}$ |
| $\sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}$ | $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_{11}$ |

to send a total of $16hx_1$ spikes to $\sigma_5$ and $\sigma_6$, and thus we get

$$
\begin{array}{lll}
t_{4hx_1+2}: & \sigma_1 : 6, & \\
& \sigma_2, \sigma_3, \sigma_4 : 5, & s^5 \to s, \\
& \sigma_5, \sigma_6 : 16h(x_1+1)+4, & \\
& \sigma_7, \sigma_8, \sigma_9, \sigma_{10} : 16h+6, & s^{16h+6}/s^4 \to s, \\
& \sigma_{11} : 4, & s^4 \to \lambda, \\
\\
t_{4hx_1+3}: & \sigma_1 : 11, & s^{11} \to s, \\
& \sigma_2, \sigma_3, \sigma_4 : 4, & s^4 \to s, \\
& \sigma_5, \sigma_6 : 16h(x_1+1)+7, & (s^{16h})^* s^7/s^6 \to s, \\
& \sigma_7, \sigma_8 : 16h+5, & s^{16h+5}/s \to s, \\
& \sigma_9, \sigma_{10} : 16h+5, & s^{16h+5}/s^4 \to s, \\
& \sigma_{11} : 4, & s^4 \to \lambda, \\
\\
t_{4hx_1+4}: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 7, & s^7/s^5 \to s, \\
& \sigma_5, \sigma_6 : 16h(x_1+1)+5, & (s^{16h})^* s^5/s^4 \to s, \\
& \sigma_7, \sigma_8 : 16h+8, & \\
& \sigma_9, \sigma_{10} : 16h+5, & s^{16h+5}/s^4 \to s, \\
& \sigma_{11} : 6, & s^6 \to \lambda.
\end{array}
$$

Neurons $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$ fire on every timestep between times $t_{4hx_1+3}$ and $t_{4hx_1+4hx_2+3}$ to send a total of $16hx_2$ spikes to $\sigma_7$ and $\sigma_8$. Thus, when $\sigma_1$ receives the last spike from its environment we have

$$
\begin{array}{lll}
t_{4hx_1+4hx_2+3}: & \sigma_1: 8, & ,\\
& \sigma_2,\sigma_3,\sigma_4: 7, & s^7/s^5 \to s,\\
& \sigma_5,\sigma_6: 16h(x_1+1)+5, & (s^{16h})^* s^5/s^4 \to s,\\
& \sigma_7,\sigma_8: 16h(x_2+1)+4, & \\
& \sigma_9,\sigma_{10}: 16h+5, & s^{16h+5}/s^4 \to s,\\
& \sigma_{11}: 4, & s^4 \to \lambda,
\end{array}
$$

$$
\begin{array}{lll}
t_{4hx_1+4hx_2+4}: & \sigma_1: 13, & s^{13}/s^4 \to s,\\
& \sigma_2,\sigma_3,\sigma_4: 6, & s^6/s \to s,\\
& \sigma_5,\sigma_6: 16h(x_1+1)+4, & \\
& \sigma_7,\sigma_8: 16h(x_2+1)+7, & (s^{16h})^* s^7/s^3 \to s,\\
& \sigma_9,\sigma_{10}: 16h+4, & \\
& \sigma_{11}: 4, & s^4 \to \lambda,
\end{array}
$$

$$
\begin{array}{lll}
t_{4hx_1+4hx_2+5}: & \sigma_1: 12, & s^{12}/s^3 \to s,\\
& \sigma_2,\sigma_3,\sigma_4: 8, & \\
& \sigma_5,\sigma_6: 16h(x_1+1)+8, & \\
& \sigma_7,\sigma_8: 16h(x_2+1)+8, & \\
& \sigma_9,\sigma_{10}: 16h+8, & \\
& \sigma_{11}: 2, & s^2 \to \lambda,
\end{array}
$$

$$
\begin{array}{ll}
t_{4hx_1+4hx_2+6}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4: 9,\\
& \sigma_5,\sigma_6: 16h(x_1+1)+9,\\
& \sigma_7,\sigma_8: 16h(x_2+1)+9,\\
& \sigma_9,\sigma_{10}: 16h+9.
\end{array}
$$

At time $t_{4hx_1+4hx_2+6}$ the neurons of $\Pi_C$ encode an initial configuration of $C$. To see this note that neurons $\sigma_5$ and $\sigma_6$ encode the initial value $x_1$ of counter $c_1$ with $16h(x_1+1)$ spikes and the instruction $q_1$ with 9 spikes, $\sigma_7$ and $\sigma_8$ encode the initial value $x_2$ of counter $c_2$ with $16h(x_2+1)$ spikes and instruction $q_1$ with 9 spikes, and $\sigma_9$ and $\sigma_{10}$ encode the initial value 0 of counter $c_3$ with $16h$ spikes and instruction $q_1$ with 9 spikes.

$\boldsymbol{\Pi_C}$ **simulating** $\boldsymbol{q_i : INC(1), q_l}$**.** Let $x_1$, $x_2$ and $x_3$ be the values in counters $c_1$, $c_2$ and $c_3$, respectively. Then our simulation of $q_i$ begins with $16h(x_1+1)+8i+1$ spikes in $\sigma_5$ and $\sigma_6$, $16h(x_2+1)+8i+1$ spikes in $\sigma_7$ and $\sigma_8$, and

$16h(x_3+1)+8i+1$ spikes in $\sigma_9$ and $\sigma_{10}$. Beginning our simulation at time $t_j$, we have

$$
\begin{array}{lll}
t_j: & \sigma_1,\sigma_2,\sigma_3,\sigma_4: 9, & \\
& \sigma_5,\sigma_6: 16h(x_1+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{8i+1} \to s, \\
& \sigma_7,\sigma_8: 16h(x_2+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s, \\
& \sigma_9: 16h(x_3+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s \\
& \sigma_{10}: 16h(x_3+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{4h+4i+3} \to s.
\end{array}
$$

Thus, from Figure 1 we get

$$
\begin{array}{lll}
t_{j+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4: 15, & \\
& \sigma_5,\sigma_6: 16h(x_1+1), & \\
& \sigma_7,\sigma_8: 16hx_2+12h+4i+2, & (s^{16h})^* s^{4m+2}/s^4 \to s, \\
& \sigma_9: 16hx_3+12h+4i+2, & (s^{16h})^* s^{4m+2}/s^4 \to s, \\
& \sigma_{10}: 16hx_3+12h+4i-2, & (s^{16h})^* s^{4m+2}/s^4 \to s, \\
& \sigma_{11}: 6, & s^6 \to \lambda, \\
 & & \\
t_{j+2}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4: 19, & \\
& \sigma_5,\sigma_6: 16h(x_1+1), & \\
& \sigma_7,\sigma_8: 16hx_2+12h+4i-2, & (s^{16h})^* s^{4m+2}/s^4 \to s, \\
& \sigma_9: 16hx_3+12h+4i-2, & (s^{16h})^* s^{4m+2}/s^4 \to s, \\
& \sigma_{10}: 16hx_3+12h+4i-6, & (s^{16h})^* s^{4m+2}/s^4 \to s, \\
& \sigma_{11}: 4, & s^4 \to \lambda.
\end{array}
$$

Note that the variable $m$ only has natural number values in the range $3 \leqslant m \leqslant 3h+i$. Neurons $\sigma_7$, $\sigma_8$, $\sigma_9$ and $\sigma_{10}$ fire on every timestep between times $t_j$ and $t_{j+3h+i-2}$ to send a total of $12h+4i-8$ spikes to $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$. In addition,

$\sigma_7$ and $\sigma_8$ each send a spike at the beginning of the simulation thus giving

$$
\begin{array}{lll}
t_{j+3h+i-2}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4:\ 12h+4i+3, & \\
& \sigma_5,\sigma_6:\ 16h(x_1+1), & \\
& \sigma_7,\sigma_8:\ 16hx_2+14, & (s^{16h})^*s^{4m+2}/s^4\rightarrow s,\\
& \sigma_9:\ 16hx_3+14, & (s^{16h})^*s^{4m+2}/s^4\rightarrow s,\\
& \sigma_{10}:\ 16hx_3+10, & (s^{16h})^*s^{10}/s^{10}\rightarrow s,\\
& \sigma_{11}:\ 4, & s^4\rightarrow\lambda,
\end{array}
$$

$$
\begin{array}{lll}
t_{j+3h+i-1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4:\ 12h+4i+7, & \\
& \sigma_5,\sigma_6:\ 16h(x_1+1), & \\
& \sigma_7,\sigma_8:\ 16hx_2+10, & (s^{16h})^*s^{10}/s^{10}\rightarrow s,\\
& \sigma_9:\ 16hx_3+10, & (s^{16h})^*s^{10}/s^{10}\rightarrow s,\\
& \sigma_{10}:\ 16hx_3, & \\
& \sigma_{11}:\ 4, & s^4\rightarrow\lambda,
\end{array}
$$

$$
\begin{array}{lll}
t_{j+3h+i}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4:\ 12h+4i+10, & s^{12h+4i+10}/s^{4(h+i-l)+1}\rightarrow s,\\
& \sigma_5,\sigma_6:\ 16h(x_1+1), & \\
& \sigma_7,\sigma_8:\ 16hx_2, & \\
& \sigma_9,\sigma_{10}:\ 16hx_3, & \\
& \sigma_{11}:\ 3, & s^3\rightarrow\lambda.
\end{array}
$$

At time $t_{j+3h+i}$ the rule $s^{12h+4i+10}/s^{4(h+i-l)+1}\rightarrow s$ is executed in $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$ in order to move from instruction $q_i$ to the next instruction $q_l$. After time $t_{j+3h+i}$ the simulation of $INC(1)$ is completed by sending $16h+8l+1$ spikes from $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$ to the neurons encoding the counters of $C$ (i.e. $\sigma_5$, $\sigma_6$, $\sigma_7$, $\sigma_8$, $\sigma_9$ and $\sigma_{10}$ ). This simulates an increment on counter $c_1$ and establishes the encoding of the next instruction. (In the below configurations $8\leqslant r\leqslant 6h+5$.)

$$
\begin{array}{lll}
t_{j+3h+i+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4:\ 8h+4l+10, & s^{2r}/s^3\rightarrow s,\\
& \sigma_5,\sigma_6:\ 16h(x_1+1)+4, & \\
& \sigma_7,\sigma_8:\ 16hx_2+4, & \\
& \sigma_9,\sigma_{10}:\ 16hx_3+4, &
\end{array}
$$

$$
\begin{array}{lll}
t_{j+3h+i+2}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4:\ 8h+4l+8, & s^{2r}/s^3\rightarrow s,\\
& \sigma_5,\sigma_6:\ 16h(x_1+1)+8, & \\
& \sigma_7,\sigma_8:\ 16hx_2+8, & \\
& \sigma_9,\sigma_{10}:\ 16hx_3+8. &
\end{array}
$$

Neurons $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$ continue to fire on every timestep between times $t_{j+3h+i}$ and $t_{j+7h+i+2l}$ to send a total of $16h+8l$ spikes to $\sigma_5$, $\sigma_6$, $\sigma_7$, $\sigma_8$, $\sigma_9$ and $\sigma_{10}$. Thus we get

$$
\begin{array}{rll}
t_{j+7h+i+2l-1}: & \sigma_1: 14, & s^{14}/s^3 \to s, \\
& \sigma_2,\sigma_3,\sigma_4: 14, & s^{14}/s^7 \to s, \\
& \sigma_5,\sigma_6: 16h(x_1+2)+8l-4, & \\
& \sigma_7,\sigma_8: 16h(x_2+1)+8l-4, & \\
& \sigma_9,\sigma_{10}: 16h(x_3+1)+8l-4, & \\
\\
t_{j+7h+i+2l}: & \sigma_1: 12, & s^{12}/s^3 \to s, \\
& \sigma_2,\sigma_3,\sigma_4: 8, & \\
& \sigma_5,\sigma_6: 16h(x_1+2)+8l, & \\
& \sigma_7,\sigma_8: 16h(x_2+1)+8l, & \\
& \sigma_9,\sigma_{10}: 16h(x_3+1)+8l, & \\
\\
t_{j+7h+i+2l+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4: 9, & \\
& \sigma_5,\sigma_6: 16h(x_1+2)+8l+1, & \\
& \sigma_7,\sigma_8: 16h(x_2+1)+8l+1, & \\
& \sigma_9,\sigma_{10}: 16h(x_3+1)+8l+1. &
\end{array}
$$

At time $t_{j+7h+i+2l+1}$ the simulation of $q_i : INC(1), q_l$ is complete. Note that an increment on the value $x_1$ in counter $c_1$ was simulated by increasing the $16h(x_1+1)$ spikes in $\sigma_5$ and $\sigma_6$ to $16h(x_1+2)$ spikes. Note also that the encoding $8l+1$ of the next instruction $q_l$ has been established in neurons $\sigma_5$, $\sigma_6$, $\sigma_7$, $\sigma_8$, $\sigma_9$ and $\sigma_{10}$.

***$\Pi_C$* simulating $q_i : DEC(1), q_l, q_k$ when $x_1 > 0$.** If we are simulating $DEC(1)$ for $x_1 > 0$ then we get

$$
\begin{array}{rlr}
t_j: & \sigma_1,\sigma_2,\sigma_3,\sigma_4: 9, & \\
& \sigma_5,\sigma_6: 16h(x_1+1)+8i+1, & (s^{16h})^* s^{32h+8i+1}/s^{32h+8i+1} \to s, \\
& \sigma_7,\sigma_8: 16h(x_2+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s, \\
& \sigma_9: 16h(x_3+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s, \\
& \sigma_{10}: 16h(x_3+1)+8i+1, & (s^{16h})^* s^{8i+1}/s^{4h+4i+3} \to s.
\end{array}
$$

The only difference between the simulation of $INC(1), q_l$ and $DEC(1), q_l, q_k$ (for $x_1 > 0$) is that an extra $32h$ spikes are used up in each of the neurons $\sigma_5$ and $\sigma_6$ at time $t_j$. To see this note the difference in the number of spikes used by the rules executed in $\sigma_5$ and $\sigma_6$ at time $t_j$ in this example and the previous example. The remainder of the simulation of $DEC(1), q_l, q_k$ for $x_1 > 0$ between times

$t_{j+1}$ and $t_{j+7h+i+2l+1}$ proceeds in the same manner as in the previous example $INC(1), q_l$. Thus at time $t_{j+7h+i+2l+1}$ we have

$$
\begin{array}{ll}
t_{j+7h+i+2l+1}: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 9, \\
& \sigma_5, \sigma_6 : 16hx_1 + 8l + 1, \\
& \sigma_7, \sigma_8 : 16h(x_2 + 1) + 8l + 1, \\
& \sigma_9, \sigma_{10} : 16h(x_3 + 1) + 8l + 1.
\end{array}
$$

At time $t_{j+7h+i+2l+1}$ the simulation of $q_i : DEC(1), q_l, q_k$ for $x_1 > 0$ is complete. Note that a decrement on the value $x_1$ in counter $c_1$ was simulated by decreasing the $16h(x_1+1)$ spikes in $\sigma_5$ and $\sigma_6$ to $16hx_1$ spikes. Note also that the encoding $8l+1$ of the next instruction $q_l$ has been established in neurons $\sigma_5$, $\sigma_6$, $\sigma_7$, $\sigma_8$, $\sigma_9$ and $\sigma_{10}$.

***$\Pi_C$* simulating $q_i : DEC(1), q_l, q_k$ when $x_1 = 0$.** If we are simulating $DEC(1)$ for $x_1 = 0$ then we get

$$
\begin{array}{lll}
t_j: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 9, & \\
& \sigma_5, \sigma_6 : 16h + 8i + 1, & s^{16h+8i+1}/s^{16h+8i} \rightarrow s, \\
& \sigma_7, \sigma_8 : 16h(x_2 + 1) + 8i + 1, & (s^{16h})^* s^{8i+1}/s^{4h+4i-1} \rightarrow s, \\
& \sigma_9 : 16h(x_3 + 1) + 8i + 1, & (s^{16h})^* s^{8i+1}/s^{4h+4i-1} \rightarrow s, \\
& \sigma_{10} : 16h(x_3 + 1) + 8i + 1, & (s^{16h})^* s^{8i+1}/s^{4h+4i+3} \rightarrow s, \\
& & \\
t_{j+1}: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 15, & \\
& \sigma_5, \sigma_6 : 1, & s \rightarrow s, \\
& \sigma_7, \sigma_8 : 16hx_2 + 12h + 4i + 2, & (s^{16h})^* s^{4m+2}/s^4 \rightarrow s, \\
& \sigma_9 : 16hx_3 + 12h + 4i + 2, & (s^{16h})^* s^{4m+2}/s^4 \rightarrow s, \\
& \sigma_{10} : 16hx_3 + 12h + 4i - 2, & (s^{16h})^* s^{4m+2}/s^4 \rightarrow s, \\
& \sigma_{11} : 6, & s^6 \rightarrow \lambda, \\
& & \\
t_{j+2}: & \sigma_1, \sigma_2, \sigma_3, \sigma_4 : 21, & \\
& \sigma_5, \sigma_6 : 0, & \\
& \sigma_7, \sigma_8 : 16hx_2 + 12h + 4i - 2, & (s^{16h})^* s^{4m+2}/s^4 \rightarrow s, \\
& \sigma_9 : 16hx_3 + 12h + 4i - 2, & (s^{16h})^* s^{4m+2}/s^4 \rightarrow s, \\
& \sigma_{10} : 16hx_3 + 12h + 4i - 6, & (s^{16h})^* s^{4m+2}/s^4 \rightarrow s, \\
& \sigma_{11} : 6, & s^6 \rightarrow \lambda.
\end{array}
$$

Note that unlike the previous examples $\sigma_5$ and $\sigma_6$ fire twice (instead of once) thus sending an extra 2 spikes to neurons $\sigma_1$, $\sigma_2$, $\sigma_3$ and $\sigma_4$. This only occurs during the simulation of a $DEC(1)$ instruction when $x_1 = 0$, and this allows neurons $\sigma_1$ to $\sigma_4$ to determine when the counter is empty. Following time $t_{j+1}$, the computation proceeds in the same manner as the previous examples until we get to timestep $t_{j+3h+i}$ when the rule $s^{12h+4i+12}/s^{4(h+i-k)+3} \to s$ is executed (instead of the rule $s^{12h+4i+10}/s^{4(h+i-l)+1} \to s$).

$$\begin{array}{lll}
t_{j+3h+i}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 12h+4i+12, & s^{12h+4i+12}/s^{4(h+i-k)+3} \to s,\\
& \sigma_5,\sigma_6 : 0, & \\
& \sigma_7,\sigma_8 : 16hx_2, & \\
& \sigma_9,\sigma_{10} : 16hx_3, & \\
& \sigma_{11} : 3, & s^3 \to \lambda,
\end{array}$$

$$\begin{array}{lll}
t_{j+3h+i+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 8h+4k+10, & s^{2r}/s^3 \to s,\\
& \sigma_5,\sigma_6 : 4, & \\
& \sigma_7,\sigma_8 : 16hx_2+4, & \\
& \sigma_9,\sigma_{10} : 16hx_3+4. &
\end{array}$$

The remainder of the simulation proceeds in the same manner as in the previous examples to give

$$\begin{array}{ll}
t_{j+7h+i+2l+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 9,\\
& \sigma_5,\sigma_6 : 16h+8k+1,\\
& \sigma_7,\sigma_8 : 16h(x_2+1)+8k+1,\\
& \sigma_9,\sigma_{10} : 16h(x_3+1)+8k+1.
\end{array}$$

At time $t_{j+7h+i+2l+1}$ the simulation of $q_i : DEC(1), q_l, q_k$ for $x_1 = 0$ is complete. The encoding $8k+1$ of the next instruction $q_k$ has been established in neurons $\sigma_5$ to $\sigma_{10}$.

**Halting.** If $C$ enters the halt instruction $q_h$ at time $t_j$ then we get the following

$$\begin{array}{lll}
t_j: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 9, & \\
& \sigma_5 : 16h(x_1+1)+8h+1, & (s^{16h})^* s^{8h+1}/s^{24h+1} \to s,\\
& \sigma_6 : 16h(x_1+1)+8h+1, & (s^{16h})^* s^{8h+1}/s^{14h} \to s,\\
& \sigma_7,\sigma_8 : 16h(x_2+1)+8h+1, & (s^{16h})^* s^{8h+1}/s^{14h} \to s\\
& \sigma_9,\sigma_{10} : 16h(x_3+1)+8h+1, & (s^{16h})^* s^{8h+1}/s^{12h} \to s,
\end{array}$$

$$
\begin{array}{lll}
t_{j+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 15, & \\
 & \sigma_5 : 16hx_1, & \\
 & \sigma_6 : 16hx_1+10h+1, & (s^{16h})^*s^{10h+1}/s^{10h+1}\rightarrow s, \\
 & \sigma_7,\sigma_8 : 16hx_2+10h+1, & (s^{16h})^*s^{10h+1}/s^{10h+1}\rightarrow s \\
 & \sigma_9,\sigma_{10} : 16hx_3+12h+1, & (s^{16h})^*s^{12h+1}/s^{2h}\rightarrow s, \\
 & \sigma_{11} : 6, & s^6\rightarrow\lambda,
\end{array}
$$

$$
\begin{array}{lll}
t_{j+2}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 20, & s^{2r}/s^3\rightarrow s \\
 & \sigma_5,\sigma_6 : 16hx_1, & \\
 & \sigma_7,\sigma_8 : 16hx_2, & \\
 & \sigma_9,\sigma_{10} : 16hx_3+10h+1, & (s^{16h})^*s^{42h+1}/s^{16h+4}\rightarrow s, \\
 & \sigma_{11} : 5, & s^5\rightarrow s,
\end{array}
$$

$$
\begin{array}{lll}
t_{j+3}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 20, & s^{2r}/s^3\rightarrow s \\
 & \sigma_5,\sigma_6 : 16hx_1+4, & \\
 & \sigma_7,\sigma_8 : 16hx_2+4, & \\
 & \sigma_9,\sigma_{10} : 16h(x_3-1)+10h+1, & (s^{16h})^*s^{42h+1}/s^{16h+4}\rightarrow s, \\
 & \sigma_{11} : 2, & s^2\rightarrow\lambda.
\end{array}
$$

With each timestep the simulated output counter is decrement by removing $16h$ spikes from neurons $\sigma_9$ and $\sigma_{10}$. This continues until we get

$$
\begin{array}{lll}
t_{j+x_3+1}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 20, & s^{2r}/s^3\rightarrow s \\
 & \sigma_5,\sigma_6 : 16hx_1+4(x_3-1), & \\
 & \sigma_7,\sigma_8 : 16hx_2+4(x_3-1), & \\
 & \sigma_9 : 16h+10h+1, & \\
 & \sigma_{10} : 16h+10h+1, & s^{26h+1}\rightarrow s, \\
 & \sigma_{11} : 2, & s^2\rightarrow\lambda
\end{array}
$$

$$
\begin{array}{lll}
t_{j+x_3+2}: & \sigma_1,\sigma_2,\sigma_3,\sigma_4 : 19, & \\
 & \sigma_5,\sigma_6 : 16hx_1+4x_3, & \\
 & \sigma_7,\sigma_8 : 16hx_2+4x_3, & \\
 & \sigma_9 : 26h+5, & \\
 & \sigma_{10} : 4, & \\
 & \sigma_{11} : 1, & s\rightarrow s.
\end{array}
$$

Recall from Section 2 that the output of an SN P system 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 $\sigma_{11}$ fires for the first time at timestep $t_{j+2}$ and for the second time at timestep $t_{j+x_3+2}$. Thus, the output of $\Pi_C$ is $x_3$ the value of the output counter $c_3$ when $C$ enters the halt instruction $q_h$. If $x_3 = 0$ then the output neuron $\sigma_{11}$ fires only once.

We have shown how to simulate arbitrary instructions of the form $q_i : INC(1), q_l$ and $q_i : DEC(1)q_l, q_k$. Instructions that operate on counters $c_2$ and $c_3$ are simulated in a similar manner. Immediately following the simulation of an instruction $\Pi_C$ is configured to begin simulation of the next instruction. Each instruction of $C$ is simulated in $7h+i+2l+1$ timesteps. The pair of input values $(x_1, x_2)$ is read into the system in $4hx_1 + 4hx_2 + 6$ timesteps and sending the output value $x_3$ out of the system takes $x_3 + 2$ timesteps. Thus, if $C$ completes its computation in time $t$ then $\Pi_C$ simulates the computation of $C$ in linear time $O(ht + hx_1 + hx_2 + x_3)$. □

## 5 Conclusion

In [10] it is shown that there exists no universal extended SN P system with 3 neurons. This lower bound is also applicable to standard SN P systems, and so our 11-neuron system shows that the smallest universal standard SN P system that is possible has between 4 and 11 neurons. It seems likely that this lower bound of 4 neurons can be increased due to the fact that the rules used in standard systems are quite limited when compared with those used in extended SN P systems. The fact that each neuron with standard rules can send no more than one spike along a synapse in a single timestep makes communication between neurons difficult for standard systems with a small number of neurons.

A related problem that is of interest is that of a possible trade-off between the time efficiency and the size of small standard SN P systems. To date all of the small standard and extended SN P systems were proved universal by simulating counter machines. The number of neurons in these SN P systems is partially dependent on the number of counters in the counter machine that is simulated to prove universality. As a result, one might expect a trade-off between the time efficiency and the number of neurons as (currently) 2-counter machines are exponentially slower than 3-counter machines when simulating Turing machines. However, as seen in [10] there is no significant trade-off between the number of neurons and the time efficiency for the extended model. It remains to be seen if this is the case for the standard model.

## References


1. Davis, M.: The definition of universal turing machine. Proceedings of the American Mathematical Society **8**(6) (1957) 1125–1126
2. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Counter machines and counter languages. Mathematical Systems Theory **2**(3) (1968) 265–283
3. Păun, A., Păun, G.: Small universal spiking neural P systems. BioSystems **90**(1) (2007) 48–60

4. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae **71**(2-3) (2006) 279–308
5. Ibarra, O.H., Păun, A., Păun, G., Rodríguez-Patón, A., Sosík, P., Woodworth, S.: Normal forms for spiking neural P systems. Theoretical Computer Science **372**(2-3) (2010) 196–217
6. Korec, I.: Small universal register machines. Theoretical Computer Science **168**(2) (1996) 267–301
7. Pan, L., Păun, G.: Spiking neural P systems: An improved normal form. Theoretical Computer Science **411**(6) (2010) 906–918
8. Păun, G., Pérez-Jiménez, M.J.: Spiking Neural P Systems. Recent Results, Research Topics. Natural Computing Series. In: Algorithmic Bioprocesses. Springer (2009) 273–291
9. Neary, T.: A boundary between universality and non-universality in spiking neural P systems. Technical Report arXiv:0912.0741v1 [cs.CC] (December 2009)
10. Neary, T.: A boundary between universality and non-universality in extended spiking neural P systems. In: In Language and Automata Theory and Applications, 4th International Conference, LATA 2010. Volume 6031 of LNCS., Trier, Springer (May 2010) 475–487
11. 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
12. 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.: Unconventional Computation, 7th International Conference, UC 2008. Volume 5204 of LNCS., Springer (2008) 189–205
13. Neary, T.: On the computational complexity of spiking neural P systems. Technical Report arXiv:0912.0928v1 [cs.CC] (December 2009)
14. Neary, T.: (2008) Presentation at Computing with Biomolecules (CBM 2008). Available at http://www.emcc.at/UC2008/Presentations/CBM5.pdf .
15. Rogozhin, Y.: Small universal Turing machines. Theoretical Computer Science **168**(2) (November 1996) 215–240
16. Schroeppel, R.: A two counter machine cannot calculate $2^n$. Technical Report AIM-257, A.I. memo 257, Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA (1972)
17. Zhang, X., Jiang, Y., Pan, L.: Small universal spiking neural P systems with exhaustive use of rules. In: 3rd International Conference on Bio-Inspired Computing: Theories and Applications(BICTA 2008), Adelaide, Australia, IEEE (October 2008) 117–128
18. Zhang, X., Zeng, X., Pan, L.: Smaller universal spiking neural P systems. Fundamenta Informaticae **87**(1) (2008) 117–136

**Table 3.** This table gives the rules for neurons $\sigma_1$ to $\sigma_6$ of $\Pi_C$, where $(1 \leqslant i < h)$, $(1 \leqslant l, k \leqslant h)$, $(8 \leqslant r \leqslant 6h+5)$ and $(3 \leqslant m \leqslant 3h+i)$.

| neuron | rules |
|---|---|
| $\sigma_1$ | $s \to s, \quad s^3/s^2 \to s, \quad s^5 \to s, \quad s^7/s^5 \to s, \quad s^{11} \to s,$ <br> $s^{12}/s^3 \to s, \quad s^{13}/s^4 \to s, \quad s^{14}/s^3 \to s, \quad s^{2r}/s^3 \to s,$ <br> $s^{12h+4i+10}/s^{4(h+i-l)+1} \to s, \quad s^{12h+4i+12}/s^{4(h+i-k)+3} \to s,$ |
| $\sigma_2, \sigma_3, \sigma_4$ | $s \to s, \quad s^4 \to s, \quad s^5 \to s, \quad s^6/s \to s, \quad s^7/s^5 \to s,$ <br> $s^{14}/s^7 \to s, \quad s^{2r}/s^3 \to s, \quad s^{12h+4i+10}/s^{4(h+i-l)+1} \to s,$ <br> $s^{12h+4i+12}/s^{4(h+i-k)+3} \to s,$ |
| $\sigma_5$ | $s \to s, \quad (s^{16h})^* s^5/s^4 \to s, \quad (s^{16h})^* s^7/s^6 \to s,$ <br> $(s^{16h})^* s^{10}/s^{10} \to s, \quad (s^{16h})^* s^{4m+2}/s^4 \to s,$ <br> $(s^{16h})^* s^{8h+1}/s^{24h+1} \to s,$ <br> $(s^{16h})^* s^{8i+1}/s^{8i+1} \to s$ if $q_i : INC(1) \in \{Q\}$ <br> $(s^{16h})^* s^{32h+8i+1}/s^{32h+8i+1} \to s$ if $q_i : DEC(1) \in \{Q\}$ <br> $s^{16h+8i+1}/s^{16h+8i} \to s$ if $q_i : DEC(1) \in \{Q\}$ <br> $(s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s$ if $q_i : INC(1), q_i : DEC(1) \notin \{Q\}$ |
| $\sigma_6$ | $s \to s, \quad (s^{16h})^* s^5/s^4 \to s, \quad (s^{16h})^* s^7/s^6 \to s,$ <br> $(s^{16h})^* s^{10}/s^{10} \to s, \quad (s^{16h})^* s^{4m+2}/s^4 \to s,$ <br> $(s^{16h})^* s^{8h+1}/s^{14h} \to s, \quad (s^{16h})^* s^{10h+1}/s^{10h+1} \to s$ <br> $(s^{16h})^* s^{8i+1}/s^{8i+1} \to s$ if $q_i : INC(1) \in \{Q\}$ <br> $(s^{16h})^* s^{32h+8i+1}/s^{32h+8i+1} \to s$ if $q_i : DEC(1) \in \{Q\}$ <br> $s^{16h+8i+1}/s^{16h+8i} \to s$ if $q_i : DEC(1) \in \{Q\}$ <br> $(s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s$ if $q_i : INC(2), q_i : DEC(2) \in \{Q\}$ <br> $(s^{16h})^* s^{8i+1}/s^{4h+4i+3} \to s$ if $q_i : INC(3), q_i : DEC(3) \in \{Q\}$ |

**Table 4.** This table gives the rules for neurons $\sigma_7$ to $\sigma_{11}$ of $\Pi_C$, where $(1 \leqslant i < h)$, $(1 \leqslant l, k \leqslant h)$, $(8 \leqslant r \leqslant 6h+5)$ and $(3 \leqslant m \leqslant 3h+i)$.

| neuron | rules |
|---|---|
| $\sigma_7, \sigma_8$ | $s \to s,\quad s^{16h+5}/s \to s,\quad s^{16h+6}/s^4 \to s,\quad s^{16h+12}/s^{10} \to s,$ <br> $(s^{16h})^* s^7/s^3 \to s,\quad (s^{16h})^* s^{10}/s^{10} \to s,$ <br> $(s^{16h})^* s^{4m+2}/s^4 \to s,\quad (s^{16h})^* s^{8h+1}/s^{14h} \to s,$ <br> $(s^{16h})^* s^{10h+1}/s^{10h+1} \to s$ <br> $(s^{16h})^* s^{8i+1}/s^{8i+1} \to s$ if $q_i : INC(2) \in \{Q\}$ <br> $(s^{16h})^* s^{32h+8i+1}/s^{32h+8i+1} \to s$ if $q_i : DEC(2) \in \{Q\}$ <br> $s^{16h+8i+1}/s^{16h+8i} \to s$ if $q_i : DEC(2) \in \{Q\}$ <br> $(s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s$ if $q_i : INC(2), q_i : DEC(2) \notin \{Q\}$ |
| $\sigma_9$ | $s \to s,\quad s^{16h+5}/s^4 \to s,\quad s^{16h+6}/s^4 \to s,\quad s^{16h+12}/s^{10} \to s,$ <br> $(s^{16h})^* s^{10}/s^{10} \to s,\quad (s^{16h})^* s^{4m+2}/s^4 \to s,$ <br> $(s^{16h})^* s^{8h+1}/s^{12h} \to s,\quad (s^{16h})^* s^{12h+1}/s^{2h} \to s,$ <br> $(s^{16h})^* s^{42h+1}/s^{16h+4} \to s,$ <br> $(s^{16h})^* s^{8i+1}/s^{8i+1} \to s$ if $q_i : INC(3) \in \{Q\}$ <br> $(s^{16h})^* s^{32h+8i+1}/s^{32h+8i+1} \to s$ if $q_i : DEC(3) \in \{Q\}$ <br> $s^{16h+8i+1}/s^{16h+8i} \to s$ if $q_i : DEC(3) \in \{Q\}$ <br> $(s^{16h})^* s^{8i+1}/s^{4h+4i-1} \to s$ if $q_i : INC(3), q_i : DEC(3) \notin \{Q\}$ |
| $\sigma_{10}$ | $s \to s,\quad s^{16h+5}/s^4 \to s,\quad s^{16h+6}/s^4 \to s,\quad s^{16h+12}/s^{10} \to s,$ <br> $(s^{16h})^* s^{10}/s^{10} \to s,\quad (s^{16h})^* s^{4m+2}/s^4 \to s,\quad s^{26h+1} \to s,$ <br> $(s^{16h})^* s^{8h+1}/s^{12h} \to s,\quad (s^{16h})^* s^{12h+1}/s^{2h} \to s,$ <br> $(s^{16h})^* s^{42h+1}/s^{16h+4} \to s,$ <br> $(s^{16h})^* s^{8i+1}/s^{8i+1} \to s$ if $q_i : INC(3) \in \{Q\}$ <br> $(s^{16h})^* s^{32h+8i+1}/s^{32h+8i+1} \to s$ if $q_i : DEC(3) \in \{Q\}$ <br> $s^{16h+8i+1}/s^{16h+8i} \to s$ if $q_i : DEC(3) \in \{Q\}$ <br> $(s^{16h})^* s^{8i+1}/s^{4h+4i+3} \to s$ if $q_i : INC(3), q_i : DEC(3) \notin \{Q\}$ |
| $\sigma_{11}$ | $s \to s,\quad s^2 \to \lambda,\quad s^3 \to \lambda,\quad s^4 \to \lambda,\quad s^5 \to s,\quad s^6 \to \lambda,$ |