On the computational complexity of spiking neural P systems

Turlough Neary

Boole Centre for Research in Informatics,
University College Cork, Ireland,

Funded by Science Foundation Ireland Research
Frontiers Programme grant number 07/RFP/CSMF641.

August 28, 2008

Introduction

- In this talk we give results regarding the time/space efficiency of spiking neural P systems.
- Spiking neural P systems are the result of a synergy inspired by spiking neural networks and P systems.
- These systems were first presented and proved universal in 2006 by Ionescu, Păun and Yokomori.

Previous spiking neural P systems

- Păun and Păun 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.
- Recently we gave an extended spiking neural P system with 12 neurons that is weakly universal and another with 18 neurons that is strongly universal.
- Spiking neural P systems with exhaustive use of extended rules were proved universal by Ionescu, Păun and Yokomori.
- A number of time efficient solutions to NP-hard problems have been given that rely families of spiking neural P systems.

Our results

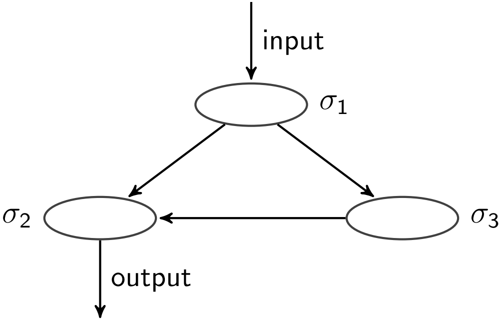
- It is shown that there exists no standard spiking neural P system that simulates Turing machines with less than exponential time and space overheads.
  - This is done by proving counter machines simulate spiking neural P systems in linear time and space.
- Here we present a universal spiking neural P system with exhaustive use of extended rules that has only 18 neurons and simulates Turing machines in polynomial time.
  - This system is shown to be universal by giving an efficient polynomial time simulation of an existing small universal Turing machine.

Spiking neural P systems

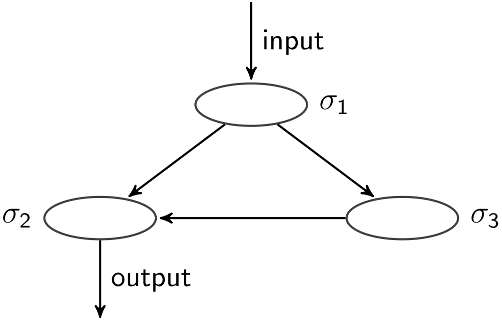
A spiking neural P system is a tuple
$\Pi = (O, \sigma_1, \sigma_2, \cdots, \sigma_m, syn, in, out)$, where:

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:
   - 2.1 $n_i \geqslant 0$ is the initial number of spikes contained in σ_i,
   - 2.2 R_i is a finite set of rules of the following two forms:
     - 2.2.1 $E/s^b \to s; d$, where E is a regular expression over s, $b \geqslant 1$ and $d \geqslant 1$,
     - 2.2.2 $s^e \to \lambda$, 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.

Spiking neural P system



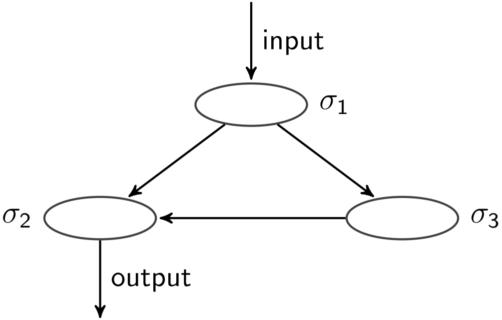
Spiking neural P system



$t_1 : \sigma_1 = 4, \qquad\qquad (s^2)^*/s^3 \to s; 3.$

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_1 if there is an applicable rule at that time.

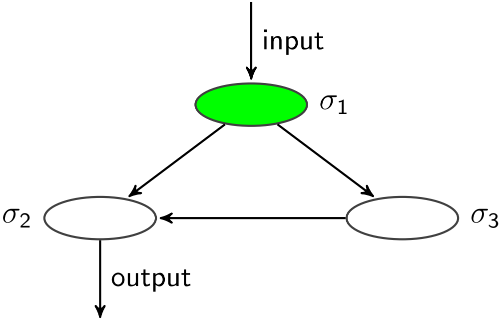
A spiking neural P systems executing rule $E/s^b \rightarrow s; d$



$$t_1 : \sigma_1 = 4, \qquad\qquad (s^2)^*/s^3 \rightarrow s; 3.$$

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_1 if there is an applicable rule at that time.

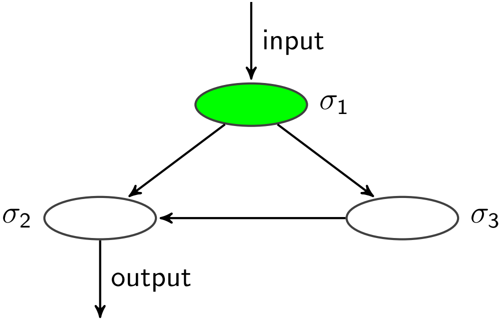
A spiking neural P systems executing rule $E/s^b \to s; d$



$t_1 : \sigma_1 = 4,$ $(s^2)^*/s^3 \to s; 3.$

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_1 if there is an applicable rule at that time.

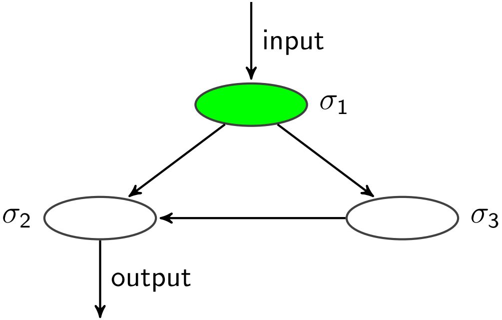
A spiking neural P systems executing rule $E/s^b \rightarrow s; d$



$t_2 : \sigma_1 = 4, \qquad\qquad (s^2)^*/s^3 \rightarrow s; 2.$

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_1 if there is an applicable rule at that time.

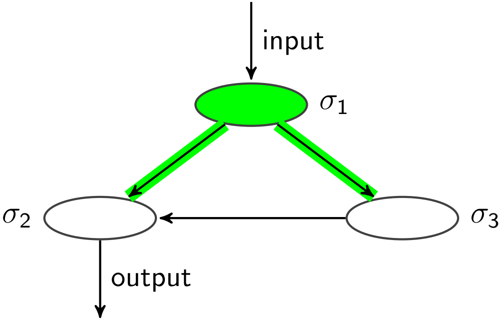
A spiking neural P systems executing rule $E/s^b \rightarrow s; d$



$t_3 : \sigma_1 = 4,$ $\qquad$ $(s^2)^*/s^3 \rightarrow s; 1.$

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_1 if there is an applicable rule at that time.

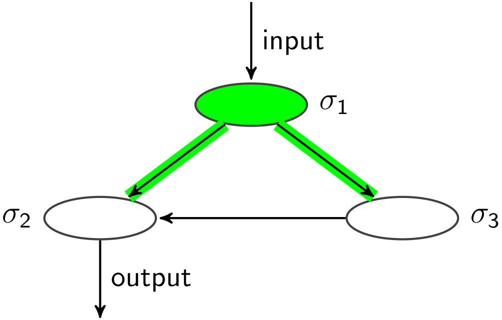
A spiking neural P systems executing rule $E/s^b \rightarrow s; d$



$t_3 : \sigma_1 = 4, \qquad (s^2)^*/s^3 \rightarrow s; 1.$

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_1 if there is an applicable rule at that time.

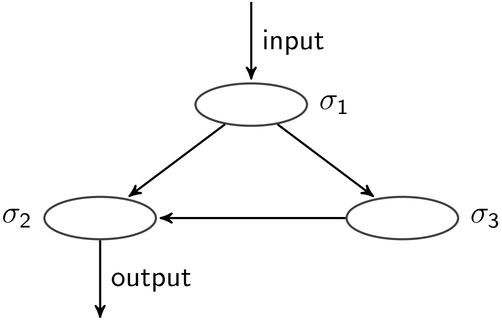
A spiking neural P systems executing rule $E/s^b \rightarrow s; d$



$t_3 : \sigma_1 = 4, \qquad (s^2)^*/s^3 \rightarrow s; 1.$

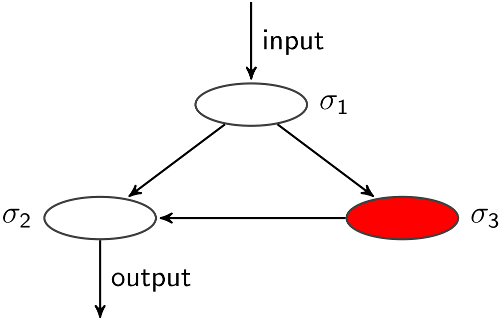
On the left $\sigma_k = y$ gives the number y of spikes in neuron σ_k at time t_j and on the right is the next rule that is to be applied at time t_1 if there is an applicable rule at that time.

A spiking neural P systems executing rule $E/s^b \rightarrow s; d$



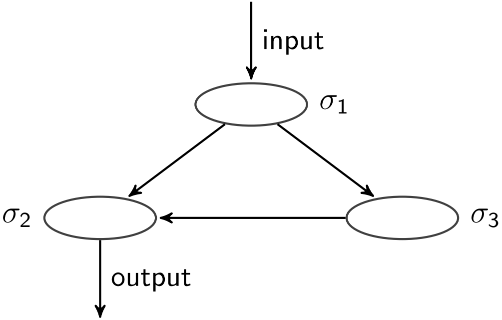
$$
\begin{aligned}
t_4 : \sigma_1 &= 1, \\
\sigma_2 &= 1, \\
\sigma_3 &= 1,
\end{aligned}
$$

A spiking neural P systems executing rule $s^e \to \lambda$



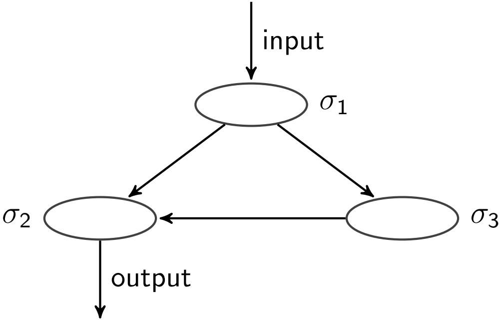
$$t_4 : \sigma_1 = 1,$$
$$\sigma_2 = 1,$$
$$\sigma_3 = 1, \qquad s \to \lambda$$

A spiking neural P systems executing rule $s^e \rightarrow \lambda$



$$
\begin{aligned}
t_5 : \sigma_1 &= 1, \\
\sigma_2 &= 1, \\
\sigma_3 &= 0,
\end{aligned}
$$

Spiking neural P system input and output



The input is a binary sequence $w = \{0, 1\}^*$. The output is the time between the first and second spike.

Counter machines

A counter machine is a tuple $C = (z, c_m, Q, q_0, q_h, \Sigma, f)$, where z gives the number of counters, c_m is the output counter, $Q = \{q_0, q_1, \cdots, q_h\}$ is the set of states, $q_0, q_h \in Q$ are the initial and halt states respectively, Σ is the input alphabet and f is the transition function

$$f : (\Sigma \times Q \times g(i)) \to (\{Y, N\} \times Q \times \{INC, DEC, NULL\})$$

where $g(i)$ is a binary valued function and $0 \leqslant i \leqslant z$, Y and N control the movement of the input read head, and INC, DEC, and $NULL$ indicate the operation to carry out on counter c_i.

Counter machines simulate spiking neural P systems in linear time and space

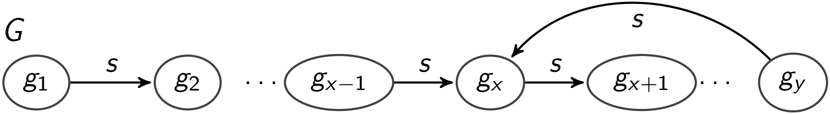
- Let $\Pi = (O, \sigma_1, \sigma_2, \cdots, \sigma_m, syn, in, out)$ be a spiking neural P system that completes it computation in time T and space S.
- We explain the operation of a non-deterministic counter machine C_Π that simulates the operation of Π in time $O(T(x)^2 m + Tm^2)$ and space $O(S)$.

Counter machines simulate spiking neural P systems

- There are $m+1$ counters $c_1, c_2, c_3, \cdots, c_m, c_{m+1}$ in counter machine C_Π.
- Each counter c_i emulates the activity of a neuron σ_i. If σ_i contains y spikes then counter c_i will store the value y.
- The states of the counter machine are used to control which neural rules are simulated in each counter and also to synchronise the operations of the simulated neurons (counters).

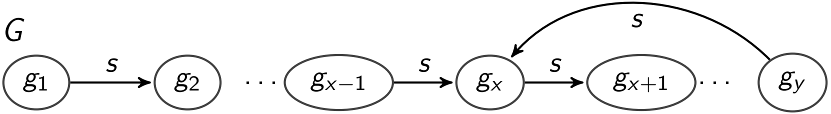
Counter machines simulate spiking neural P systems

Finite state machine G decides if a particular rule $E/s^b \rightarrow s; d$ is applicable in a neuron given the number of spikes in the neuron *at a given time* in the computation.

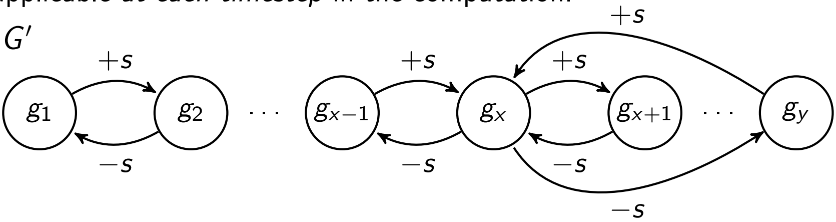


Counter machines simulate spiking neural P systems

Finite state machine G decides if a particular rule $E/s^b \rightarrow s; d$ is applicable in a neuron given the number of spikes in the neuron *at a given time* in the computation.



Machine G' keeps track of the movement of spikes into and out of the neuron and decides whither or not a particular rule is applicable *at each timestep* in the computation.



Counter machines simulate spiking neural P systems

- The algorithm used by counter machine C_Π is presented as three stages. These three stages simulate the synchronous update of all the neurons in Π at an arbitrary timestep.
- Recall that each neuron σ_i in Π is simulated by a counter c_i in C_Π.
- A single iteration of Stage 1 identifies which applicable rule to simulate in a simulated open neuron c_i. If the rule $E/s^b \rightarrow s; d$ is to be executed then b simulated spikes are removed by decrementing the counter b times. Also, the d value for σ_i is recorded in the states of the counter machine.
- Stage 1 is iterated until all simulated open neurons have had the correct number of simulated spikes removed.
- Note that during Stage 1 if a rule of the form $E/s^b \rightarrow s; d$ is executed $d > 0$ this is also recorded in the states of the counter machine.

Counter machines simulate spiking neural P systems

- A single iteration of Stage 2 identifies all the synapses leaving a firing neuron and increments every counter that simulates an open neuron at the end of one of these synapses.
- Stage 2 is iterated until all firing neurons have been simulated by incrementing the appropriate counters.
- Stage 3 synchronises each neuron with the global clock and increments the output counter if necessary.

Counter machines simulate spiking neural P systems in linear time and space

C_Π simulates Π in space of $O(S)$.

- Stage 1. A single iteration of Stage 1 take $O(x^2)$ time. This stage is iterated a maximum of m times per simulated timestep giving $O(x^2m)$ time.
- Stage 2. The maximum number of synapses leaving a neuron σ_i is m. A single spike traveling along a neuron is simulated in one step. Stage 2 is iterated a maximum of m times per simulated timestep giving $O(m^2)$ time.
- Stage 3. Takes a small constant number of steps.

Thus, a single timestep of Π is simulated by C_Π in $O(x^2m + m^2)$ time and T timesteps of Π are simulated in linear time $O(Tx^2m + Tm^2)$ by C_Π.

Counter machines simulate spiking neural P systems in linear time and space

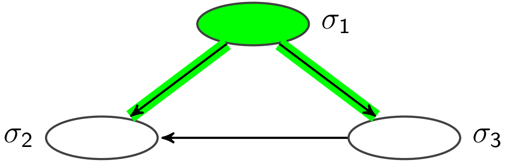
- Fischer et al. have previously shown that counter machines require exponential time and space to simulate Turing machines.
- From our result and Fischer's it immediately follows that spiking neural P systems require exponential time and space to simulate Turing machines.

A universal spiking neural P system that is both small and time efficient

- We present a small universal spiking neural P system with exhaustive use of extended rules and simulates Turing machines in polynomial time.
- This system has only 18 neurons and simulates the computation of an existing small universal Turing machine $U_{6,4}$.

Extended spiking neural P systems with exhaustive use of rules

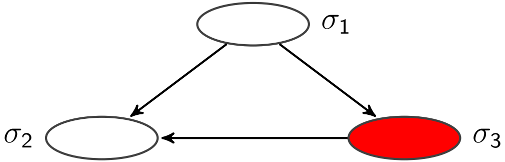
- An extended spiking neural P system has more general rules of the form $E/s^b \to s^p; d$, where $b \geqslant p \geqslant 0$.
- An extended spiking neural P system with exhaustive use of rules applies its rules as follows;



$t_1 : \sigma_1 = 20, \qquad (s^2)^*/s^3 \to s^2; 0.$

Extended spiking neural P systems with exhaustive use of rules

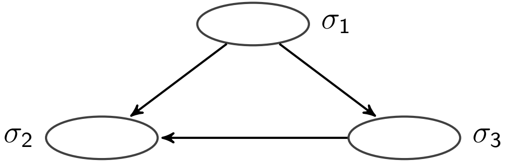
- An extended spiking neural P system has more general rules of the form $E/s^b \to s^p; d$, where $b \geqslant p \geqslant 0$.
- An extended spiking neural P system with exhaustive use of rules applies its rules as follows;



$$t_2 : \sigma_1 = 2,$$
$$\sigma_2 = 12,$$
$$\sigma_3 = 12, \qquad (s^2)^*/s \to \lambda; 0.$$

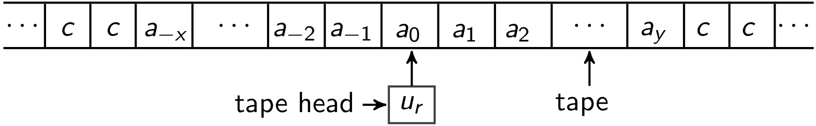
Extended spiking neural P systems with exhaustive use of rules

- An extended spiking neural P system has more general rules of the form $E/s^b \to s^p; d$, where $b \geqslant p \geqslant 0$.
- An extended spiking neural P system with exhaustive use of rules applies its rules as follows;



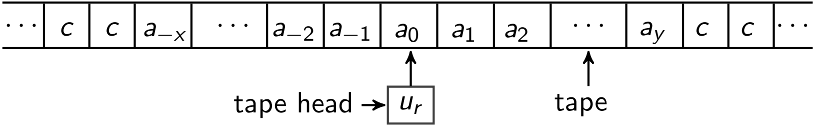
$$
\begin{aligned}
t_3 : \sigma_1 &= 2, \\
\sigma_2 &= 12, \\
\sigma_3 &= 0,
\end{aligned}
$$

Configuration of universal Turing machine $U_{6,4}$



The current state of $U_{6,4}$ is u_r and c is the blank symbol. The cells between a_{-x} and a_y include all of the cells on $U_{6,4}$'s tape that have either been visited by the tape head prior to configuration C_k above or contain part of the input to $U_{6,4}$.

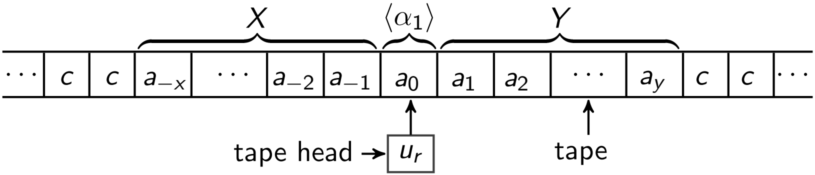
Encoding a configuration of $U_{6,4}$



- The encoding of each object z is given by $\langle z \rangle$.
- The states and symbols of $U_{6,4}$ are encoded as numerical values.
- Each encoded tape cell a_i is encoded as $\langle a_i \rangle = \langle \alpha \rangle$ where α is a tape symbol of $U_{6,4}$. The tape contents to the left and right of the tape head are encoded as the numbers $X = \sum\limits_{i=1}^{x} 32^i \langle a_i \rangle$ and $Y = \sum\limits_{j=1}^{y} 32^j \langle a_j \rangle$, respectively.

Encoding a configuration of $U_{6,4}$

$$X = \sum_{i=1}^{x} 32^i \langle a_i \rangle$$

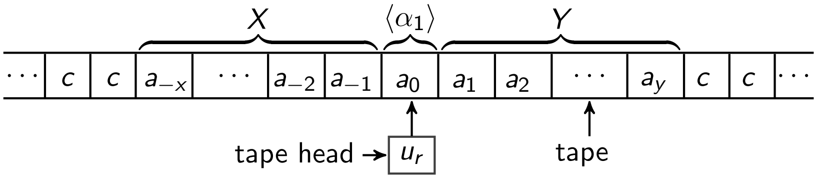


Thus the entire configuration C_k is encoded as three natural numbers via the equation

$$\langle C_k \rangle = (X,\ Y,\ \langle u_r \rangle + \langle \alpha_1 \rangle)$$

Simulating the transition rule $u_r, \alpha_1, \alpha_2, L, u_s$ on $\langle C_K \rangle$

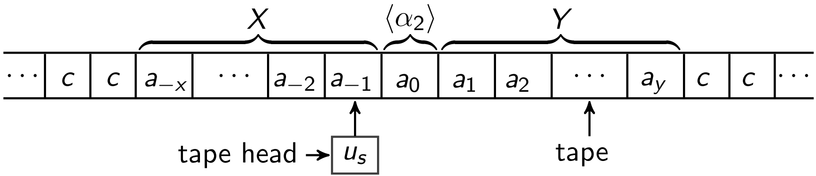
$$X = \sum_{i=1}^{x} 32^i \langle a_i \rangle$$



$$\langle C_k \rangle = (X,\ Y,\ \langle u_r \rangle + \langle \alpha_1 \rangle)$$

Simulating the transition rule $u_r, \alpha_1, \alpha_2, L, u_s$ on C_K

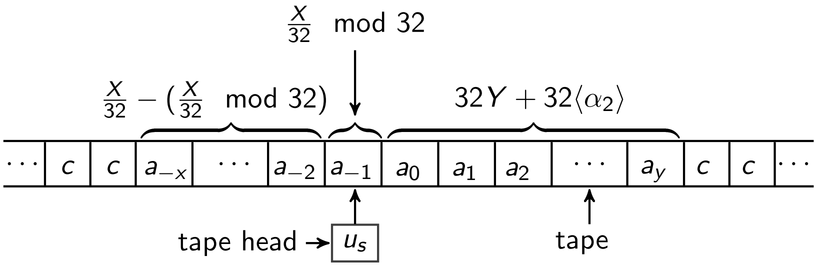
$$X = \sum_{i=1}^{x} 32^i \langle a_i \rangle$$



$$\langle C_k \rangle = (X,\ Y,\ \langle u_r \rangle + \langle \alpha_1 \rangle)$$

Simulating the transition rule $u_r, \alpha_1, \alpha_2, L, u_s$ on C_K

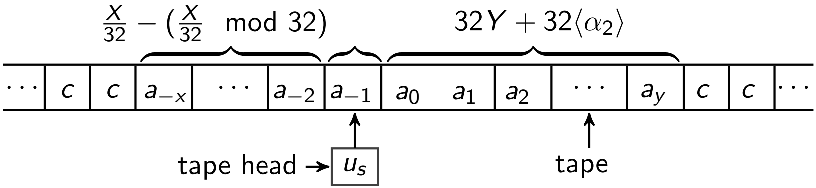
$$X = \sum_{i=1}^{x} 32^i \langle a_i \rangle$$



$$\langle C_k \rangle = (X,\ Y,\ \langle u_r \rangle + \langle \alpha_1 \rangle)$$

$$\langle C_{k+1} \rangle = \left(\frac{X}{32} - (\frac{X}{32} \mod 32),\ 32Y + 32\langle \alpha_2 \rangle,\ (\frac{X}{32} \mod 32) + \langle u_s \rangle \right)$$

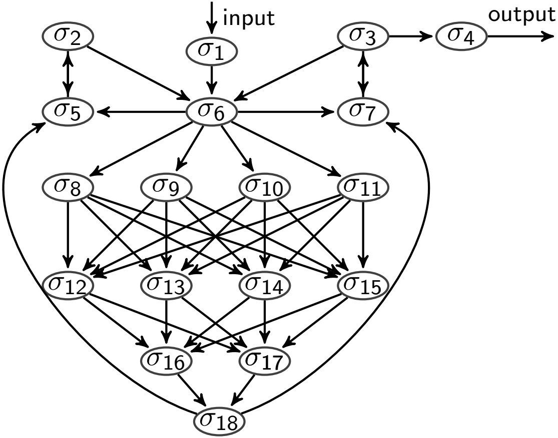
Simulating the transition rule $u_r, \alpha_1, \alpha_2, D, u_s$ on C_K



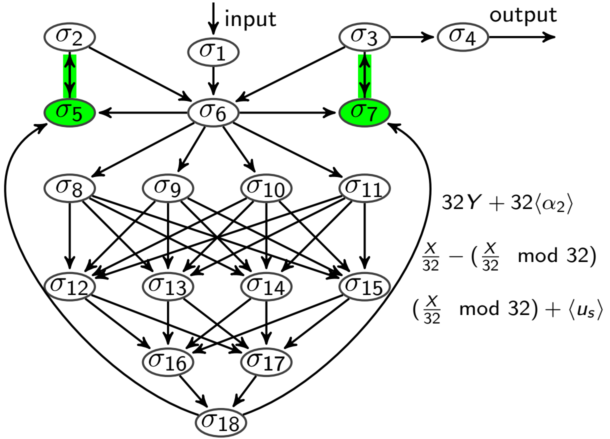
$$\langle C_k \rangle = (X,\ Y,\ \langle u_r \rangle + \langle \alpha_1 \rangle)$$

$$\langle C_{k+1} \rangle = \begin{cases} (\frac{X}{32} - (\frac{X}{32} \mod 32),\ 32Y + 32\langle \alpha_2 \rangle,\ (\frac{X}{32} \mod 32) + \langle u_s \rangle) \\ (32X + 32\langle \alpha_2 \rangle,\ \frac{Y}{32} - (\frac{Y}{32} \mod 32),\ (\frac{Y}{32} \mod 32) + \langle u_s \rangle) \end{cases}$$

Universal spiking neural P system



Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



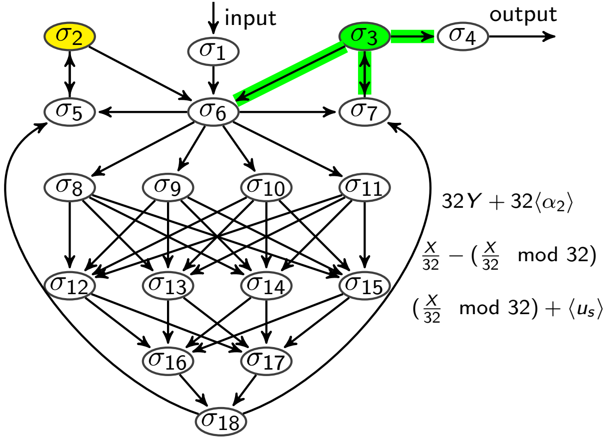
$t_i : \sigma_2 = X,$

$\sigma_3 = Y,$

$\sigma_5 = \langle u_r\rangle + \langle\alpha_1\rangle,$ $\quad s^{\langle u_r\rangle + \langle\alpha_1\rangle}/s \to s; 1,$

$\sigma_7 = \langle u_r\rangle + \langle\alpha_1\rangle,$ $\quad s^{\langle u_r\rangle + \langle\alpha_1\rangle}/s \to s; 1.$

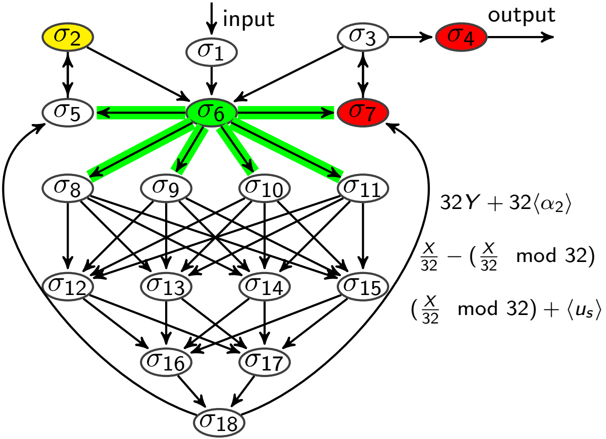
Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$t_{i+1} : \sigma_2 = X + \langle u_r\rangle + \langle\alpha_1\rangle, \qquad s^{64}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s^{32} \to s; 9,$$

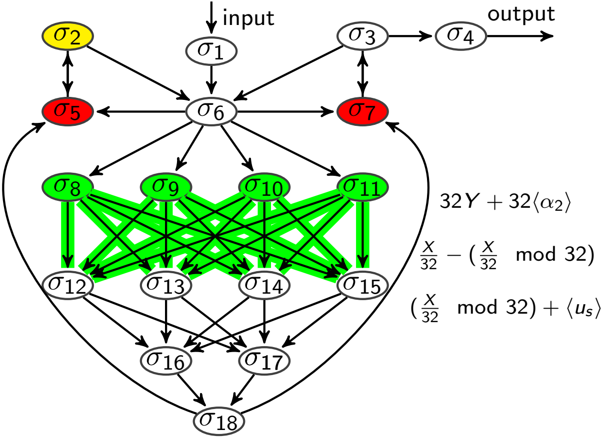
$$\sigma_3 = Y + \langle u_r\rangle + \langle\alpha_1\rangle, \qquad (s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s \to s; 1.$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$
\begin{aligned}
t_{i+2}: \sigma_2 &= X + \langle u_r\rangle + \langle\alpha_1\rangle, & s^{64}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s^{32} &\to s; 8,\\
\sigma_4 &= Y + \langle u_r\rangle + \langle\alpha_1\rangle, & s &\to \lambda; 0,\\
\sigma_6 &= Y + \langle u_r\rangle + \langle\alpha_1\rangle, & (s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s &\to s; 1,\\
\sigma_7 &= Y + \langle u_r\rangle + \langle\alpha_1\rangle, & s^{32}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s &\to \lambda; 0.
\end{aligned}
$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$

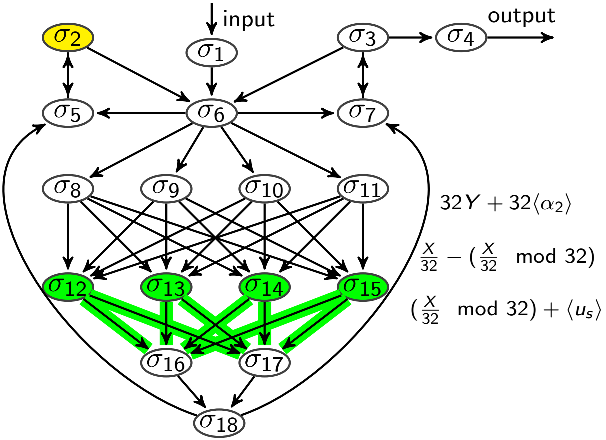


$$t_{i+3} : \sigma_2 = X + \langle u_r\rangle + \langle\alpha_1\rangle, \qquad s^{64}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s^{32} \to s; 7,$$

$$\sigma_5, \sigma_7 = Y + \langle u_r\rangle + \langle\alpha_1\rangle, \qquad s^{32}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s \to \lambda; 0,$$

$$\sigma_8, \sigma_9, \sigma_{10}, \sigma_{11} = Y + \langle u_r\rangle + \langle\alpha_1\rangle, \qquad s^{32}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s \to s; 1.$$

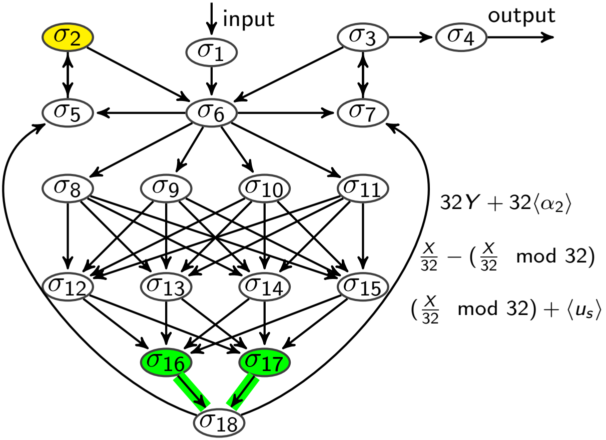
Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$t_{i+4} : \sigma_2 = X + \langle u_r\rangle + \langle\alpha_1\rangle, \qquad s^{64}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s^{32} \longrightarrow s; 6,$$

$$\sigma_{12}, \sigma_{13}, \sigma_{14}, \sigma_{15} = 4(Y + \langle u_r\rangle + \langle\alpha_1\rangle), \qquad (s^{128})^* s^{4(\langle u_r\rangle+\langle\alpha_1\rangle)}/s \longrightarrow s; 1.$$

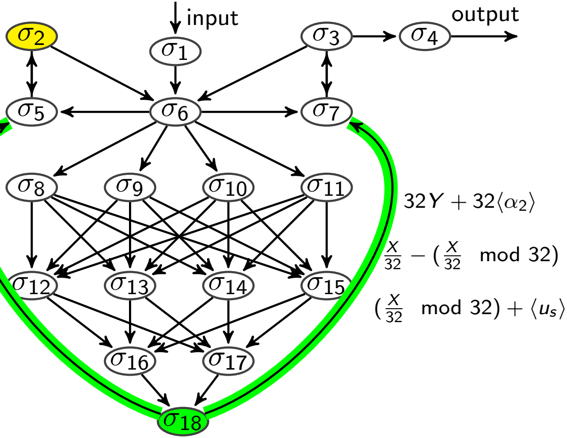
Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$t_{i+5} : \sigma_2 = X + \langle u_r \rangle + \langle \alpha_1 \rangle, \qquad s^{64}(s^{32})^* s^{\langle u_r \rangle + \langle \alpha_1 \rangle}/s^{32} \to s; 5,$$

$$\sigma_{16}, \sigma_{17} = 16(Y + \langle u_r \rangle + \langle \alpha_1 \rangle), \qquad (s^{512})^* s^{16(\langle u_r \rangle + \langle \alpha_1 \rangle)}/s \to s; 1.$$

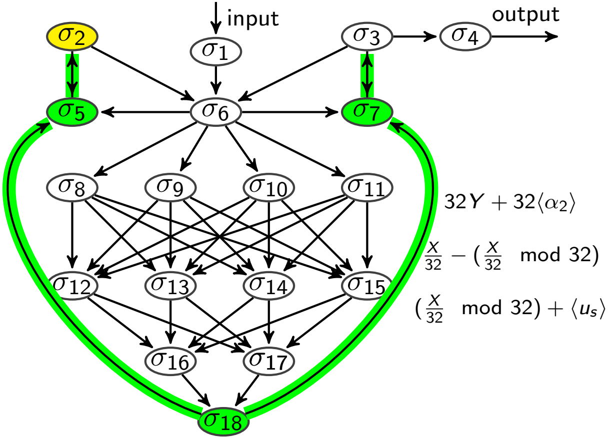
Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$t_{i+6} : \sigma_2 = X + \langle u_r \rangle + \langle \alpha_1 \rangle, \qquad s^{64}(s^{32})^* s^{\langle u_r \rangle + \langle \alpha_1 \rangle} / s^{32} \to s; 4,$$

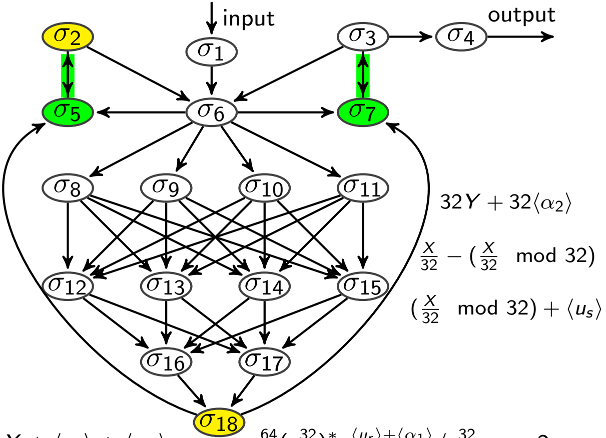
$$\sigma_{18}, = 32(Y + \langle u_r \rangle + \langle \alpha_1 \rangle), \qquad (s^{32^2})^* s^{32(\langle u_r \rangle + \langle \alpha_1 \rangle)} / s^{32^2} \to (s^{32^2}); 1.$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



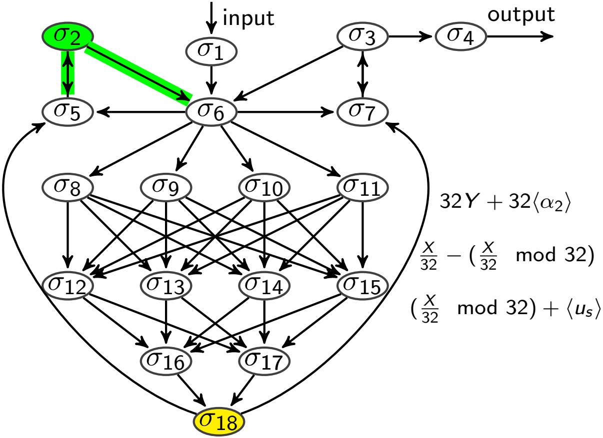
$$
\begin{aligned}
t_{i+7}: \sigma_2 &= X + \langle u_r\rangle + \langle\alpha_1\rangle, & s^{64}(s^{32})^* s^{\langle u_r\rangle+\langle\alpha_1\rangle}/s^{32} &\to s; 3,\\
\sigma_5 &= 32Y, & (s^{32})^*/s^{32} &\to s; 1,\\
\sigma_7 &= 32Y, & (s^{32})^*/s^{32} &\to s; 1,\\
\sigma_{18}, &= 32(\langle u_r\rangle + \langle\alpha_1\rangle), & s^{32(\langle u_r\rangle+\langle\alpha_1\rangle)}/s^{32(\langle u_r\rangle+\langle\alpha_1\rangle)-\langle u_s\rangle} &\to s^{32\langle\alpha_2\rangle}; 1.
\end{aligned}
$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



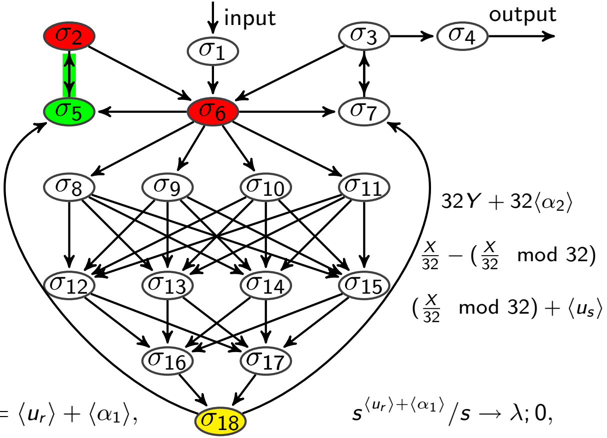
$$
\begin{aligned}
t_{i+8} : \sigma_2 &= X + \langle u_r \rangle + \langle \alpha_1 \rangle, & s^{64}(s^{32})^* s^{\langle u_r \rangle + \langle \alpha_1 \rangle}/s^{32} &\to s; 2, \\
\sigma_3 &= 32Y, & & \\
\sigma_5 &= 32\langle \alpha_2 \rangle, & (s^{32})^*/s^{32} &\to s; 1, \\
\sigma_7 &= 32\langle \alpha_2 \rangle, & (s^{32})^*/s^{32} &\to s; 1, \\
\sigma_{18}, &= \langle u_s \rangle, & s^{\langle u_s \rangle}/s &\to s; 4.
\end{aligned}
$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



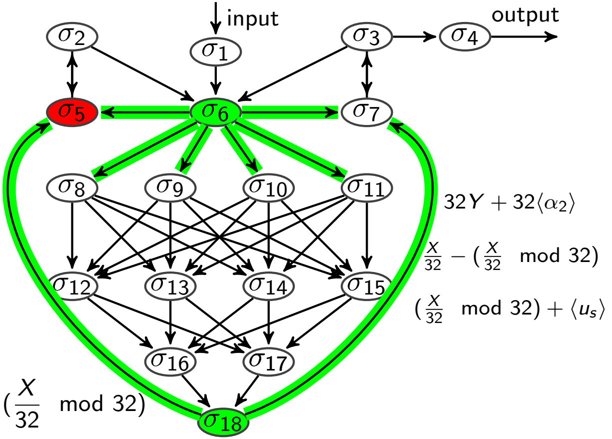
$$
\begin{aligned}
t_{i+9} : \sigma_2 &= X + \langle u_r\rangle + \langle\alpha_1\rangle, && s^{64}(s^{32})^* s^{\langle u_r\rangle + \langle\alpha_1\rangle}/s^{32} \to s; 1,\\
\sigma_3 &= 32Y + 32\langle\alpha_2\rangle, && \\
\sigma_{18}, &= \langle u_s\rangle, && s^{\langle u_s\rangle}/s \to s; 3.
\end{aligned}
$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$
\begin{aligned}
t_{i+10} : \sigma_2 &= \langle u_r \rangle + \langle \alpha_1 \rangle, & s^{\langle u_r \rangle + \langle \alpha_1 \rangle}/s &\to \lambda; 0, \\
\sigma_3 &= 32Y + 32\langle \alpha_2 \rangle & & \\
\sigma_5 &= \frac{X}{32}, & (s^{32})^* s^{(\frac{X}{32} \mod 32)}/s^{32} &\to s^{32}; 1, \\
\sigma_6 &= \frac{X}{32}, & (s^{32})^* s^{(\frac{X}{32} \mod 32)}/s^{32} &\to \lambda; 0, \\
\sigma_{18}, &= \langle u_s \rangle, & s^{\langle u_s \rangle}/s &\to s; 2.
\end{aligned}
$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$t_{i+11} : \sigma_2 = \frac{X}{32} - (\frac{X}{32} \mod 32)$$

$$\sigma_3 = 32Y + 32\langle \alpha_2 \rangle$$

$$\sigma_5 = \frac{X}{32} \mod 32$$

$$\sigma_6 = \frac{X}{32} \mod 32$$

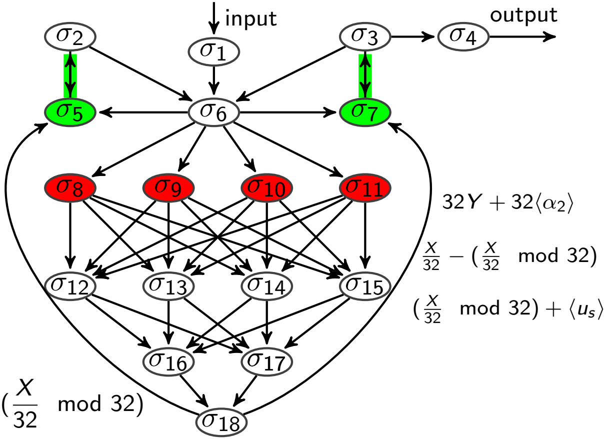
$$\sigma_{18}, = \langle u_s \rangle,$$

$$s^{\frac{X}{32} \mod 32} / s^{\frac{X}{32} \mod 32} \rightarrow \lambda; 0$$

$$s^{\frac{X}{32} \mod 32} / s^{\frac{X}{32} \mod 32} \rightarrow s; 1$$

$$s^{\langle u_s \rangle} / s \rightarrow s; 1$$

Simulating $u_r, \alpha_1, \alpha_2, L, u_s$



$$t_{i+12} : \sigma_2 = \frac{X}{32} - (\frac{X}{32} \mod 32)$$

$$\sigma_3 = 32Y + 32\langle \alpha_2 \rangle$$

$$\sigma_5 = (\frac{X}{32} \mod 32) + \langle u_s \rangle \qquad s^{(\frac{X}{32} \mod 32)+\langle u_s \rangle}/s \rightarrow s; 1$$

$$\sigma_7 = (\frac{X}{32} \mod 32) + \langle u_s \rangle \qquad s^{(\frac{X}{32} \mod 32)+\langle u_s \rangle}/s \rightarrow s; 1,$$

$$\sigma_8, \sigma_9, \sigma_{10}, \sigma_{11} = \frac{X}{32} \mod 32 \quad s^{\frac{X}{32} \mod 32}/s^{\frac{X}{32} \mod 32} \rightarrow \lambda; 0.$$

Conclusions

- Standard (extended) spiking neural P systems require exponential time and space to simulate Turing machines.
- Extended spiking neural P systems with exhaustive use of rules simulate Turing machines in polynomial time and exponential space.
- The simulation technique given for the universal spiking neural P system is easily adapted to give other more time efficient spiking neural P systems.