Notes on Spiking Neural P Systems and Finite Automata

Francis George Carreon-Cabarle[1], Henry N. Adorna[1],
Mario J. Pérez-Jiménez[2]

[1]Department of Computer Science,
University of the Philippines Diliman
Quezon city, 1101, Philippines;
[2]Department of Computer Science and AI
University of Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
fccabarle@up.edu.ph, hnadorna@dcs.upd.edu.ph, marper@us.es

02 to 06 Feb 2015

13th Brainstorming Week on Membrane Computing
Seville, Spain

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

---

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

SNP systems preliminaries

- *Standard rules:* neuron emits at most one pulse (the *spike*, represented by symbol a) each step;
- *Extended rules:* neuron can emit more than one spike each step;
- Generators have output neuron only;
- Acceptors have input neuron only;
- *SNP transducers:* standard and forgetting rules, one input and one output neuron[1]
  - At most one spike can enter or leave the system
- *SNP modules:* extended rules, one or more input neurons and output neurons[2]
  - More than one spike can enter or leave the system

[1] Păun, Gh., Pérez-Jiménez, M.J., Rozenberg, G.: Computing Morphisms by Spiking Neural P Systems. IJCFS. vol. 8(6) pp. 1371-1382 (2007)

[2] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In this work

- Continue SNP modules investigation:
  - Amend construction problem in simulating deterministic finite automata and deterministic finite transducer;[3]
  - Reduce number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extend our construction to simulate DFA with output;
  - Generating automatic sequences

[3] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In this work

- Continue SNP modules investigation:
  - Amend construction problem in simulating deterministic finite automata and deterministic finite transducer;[3]
  - Reduce number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extend our construction to simulate DFA with output;
  - Generating automatic sequences

[3] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In this work

- Continue SNP modules investigation:
  - Amend construction problem in simulating deterministic finite automata and deterministic finite transducer;[3]
  - Reduce number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extend our construction to simulate DFA with output;
  - Generating automatic sequences

[3] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In this work

- Continue SNP modules investigation:
  - Amend construction problem in simulating deterministic finite automata and deterministic finite transducer;[3]
  - Reduce number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extend our construction to simulate DFA with output;
  - Generating automatic sequences

[3] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In this work

- Continue SNP modules investigation:
  - Amend construction problem in simulating deterministic finite automata and deterministic finite transducer;[3]
  - Reduce number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extend our construction to simulate DFA with output;
  - Generating automatic sequences

[3] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

Quick recall

A *deterministic finite automaton* (in short, a DFA) D, is defined by the 5-tuple $D = (Q, \Sigma, q_1, \delta, F)$, where:

- $Q = \{q_1, \ldots, q_n\}$ is a finite set of states,
- $\Sigma = \{b_1, \ldots, b_m\}$ is the input alphabet,
- $\delta : Q \times \Sigma \to Q$ is the transition function,
- $q_1 \in Q$ is the initial state,
- $F \subseteq Q$ is a set of final states.

Quick recall

A *deterministic finite state transducer* (in short, a DFST) with accepting states T, is defined by the 6-tuple $T = (Q, \Sigma, \Delta, q_1, \delta', F)$, where:

- $Q = \{q_1, \ldots, q_n\}$ is a finite set of states,
- $\Sigma = \{b_1, \ldots, b_m\}$ is the input alphabet,
- $\Delta = \{c_1, \ldots, c_t\}$ is the output alphabet,
- $\delta' : Q \times \Sigma \to Q \times \Delta$ is the transition function,
- $q_1 \in Q$ is the initial state,
- $F \subseteq Q$ is a set of final states.

Quick recall

A *deterministic finite automaton with output* (in short, a DFAO) M, is defined by the 6-tuple $M = (Q, \Sigma, \delta'', q_1, \Delta, \tau)$, where:

- $Q = \{q_1, \ldots, q_n\}$ is a finite set of states,
- $\Sigma = \{b_1, \ldots, b_m\}$ is the input alphabet,
- $\delta'' : Q \times \Sigma \to Q$ is the transition function,
- $q_1 \in Q$ is the initial state,
- $\Delta = \{c_1, \ldots, c_t\}$ is the output alphabet,
- $\tau : Q \to \Delta$ is the output function.

A given DFAO M defines a function from Σ^* to Δ, denoted as $f_M(w) = \tau(\delta''(q_1, w))$ for $w \in \Sigma^*$. If $\Sigma = \{1, ..., k\}$, denoted as Σ_k, then M is a *k-DFAO*.
A sequence, denoted as $\mathbf{a} = (a_n)_{n \geq 0}$, is *k-automatic* if there exists a k-DFAO, M, such that given $w \in \Sigma_k^*$, $a_n = \tau(\delta''(q_1, w))$, where $[w]_k = n$, $[w]_k = n$ is the base-k representation of $n \in \mathbb{N}$.

Quick recall

A *spiking neural P system* (in short, an SNP system) of degree $m \geq 1$, is a construct of the form
$\Pi = (\{a\}, \sigma_1, \ldots, \sigma_m, syn, in, out)$ where:

- $\{a\}$ is the singleton alphabet (a is called *spike*);
- $\sigma_1, \ldots, \sigma_m$ are *neurons* of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
  - $n_i \geq 0$ is the *initial number of spikes* inside σ_i;
  - R_i is a finite *set of rules* of the general form: $E/a^c \to a^p; d$, where E is a regular expression over $\{a\}$, $c \geq 1$, with $p, d \geq 0$, and $c \geq p$; if $p = 0$, then $d = 0$ and $L(E) = \{a^c\}$;
- $syn \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$, with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses*);
- $in, out \in \{1, \ldots, m\}$ indicate the *input* and *output* neurons, respectively.

Quick recall

A *spiking neural P module* (in short, an SNP module) of degree $m \geq 1$, is a construct of the form $\Pi = (\{a\}, \sigma_1, \ldots, \sigma_m, syn, N_{in}, N_{out})$ where

- $\{a\}$ is the singleton alphabet (a is called *spike*);
- $\sigma_1, \ldots, \sigma_m$ are *neurons* of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
    - $n_i \geq 0$ is the *initial number of spikes* inside σ_i;
    - R_i is a finite *set of rules* of the general form: $E/a^c \rightarrow a^p$, where E is a regular expression over $\{a\}$, $c \geq 1$, and $p \geq 0$, with $c \geq p$; if $p = 0$, then $L(E) = \{a^c\}$
- $syn \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$, with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses*);
- $N_{in}, N_{out} (\subseteq \{1, 2, \ldots, m\})$ indicate the *sets of input* and *output* neurons, respectively.

Previous DFA and DFST simulations

- For some SNP module Π_D simulating finite automata D, $L(\Pi_D) = \{w \in \Sigma^* | \Pi_D(w) \in Q^* F\}$;
- Some previous results:[4]
  - Any regular language L can be expressed as $L = L(\Pi_D)$ for some SNP module Π_D.
  - Any finite transducer T can be simulated by some SNP module Π_T.

[4] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

Previous DFA and DFST simulations

- For some SNP module Π_D simulating finite automata D, $L(\Pi_D) = \{w \in \Sigma^* | \Pi_D(w) \in Q^* F\}$;
- Some previous results:[4]
  - Any regular language L can be expressed as $L = L(\Pi_D)$ for some SNP module Π_D.
  - Any finite transducer T can be simulated by some SNP module Π_T.

[4] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

Previous DFA and DFST simulations

- For some SNP module Π_D simulating finite automata D, $L(\Pi_D) = \{w \in \Sigma^* | \Pi_D(w) \in Q^* F\}$;
- Some previous results:[4]
  - Any regular language L can be expressed as $L = L(\Pi_D)$ for some SNP module Π_D.
  - Any finite transducer T can be simulated by some SNP module Π_T.

[4] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

Previous DFA and DFST simulations

- For some SNP module Π_D simulating finite automata D, $L(\Pi_D) = \{w \in \Sigma^* | \Pi_D(w) \in Q^* F\}$;
- Some previous results:[4]
  - Any regular language L can be expressed as $L = L(\Pi_D)$ for some SNP module Π_D.
  - Any finite transducer T can be simulated by some SNP module Π_T.

[4] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

Previous DFA and DFST simulations

Let $D = (Q, \Sigma, \delta, q_1, F)$ be a DFA, where $\Sigma = \{b_1, \ldots, b_m\}$, $Q = \{q_1, \ldots, q_n\}$. An SNP Module Π_D simulating D is as follows:

$$\Pi_D = (\{a\}, \sigma_1, \sigma_2, \sigma_3, syn, \{3\}, \{3\}),$$

where

- $\sigma_1 = \sigma_2 = (n, \{a^n \to a^n\})$,
- $\sigma_3 = (n, \{a^{2n+i+k}/a^{2n+i+k-j} \to a^j | \delta(q_i, b_k) = q_j\})$,
- $syn = \{(1, 2), (2, 1), (1, 3)\}$.

Previous DFA and DFST simulations

Let $T = (Q, \Sigma, \Delta, \delta', q_1, F)$ be a DFST, where $\Sigma = \{b_1, \ldots, b_k\}$, $\Delta = \{c_1, \ldots, c_t\}$, $Q = \{q_1, \ldots, q_n\}$. We construct the following SNP module simulating T:

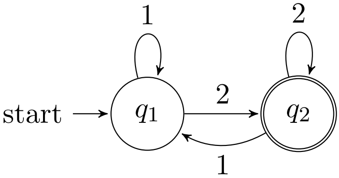
$$\Pi_T = (\{a\}, \sigma_1, \sigma_2, \sigma_3, syn, \{3\}, \{3\}),$$

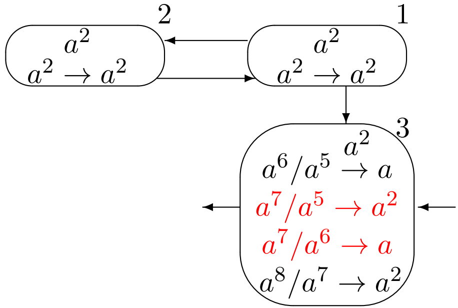
where:

- $\sigma_1 = \sigma_2 = (n, \{a^n \to a^n\})$,
- $\sigma_3 = (n, \{a^{2n+i+k+t}/a^{2n+i+k+t-j} \to a^{n+s} | \delta'(q_i, b_k) = (q_j, c_s)\})$,
- $syn = \{(1,2), (2,1), (1,3)\}$.

An example

Using the previous construction for simulating DFA[5]





[5] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In this work

- Some results:
  - Any regular language L can be expressed as $L = L(\Pi'_D)$ for some 1-neuron SNP module Π'_D.
  - Any finite transducer T can be simulated by some 1-neuron SNP module Π'_T.

In this work

- Some results:
  - Any regular language L can be expressed as $L = L(\Pi'_D)$ for some 1-neuron SNP module Π'_D.
  - Any finite transducer T can be simulated by some 1-neuron SNP module Π'_T.

In this work

- Some results:
  - Any regular language L can be expressed as $L = L(\Pi'_D)$ for some 1-neuron SNP module Π'_D.
  - Any finite transducer T can be simulated by some 1-neuron SNP module Π'_T.

In this work

Given a DFA D, we construct an SNP module Π'_D simulating D as follows:

$$\Pi'_D = (\{a\}, \sigma_1, syn, \{1\}, \{1\}),$$

where

- $\sigma_1 = (1, \{a^{k(2n+1)+i}/a^{k(2n+1)+i-j} \to a^j | \delta(q_i, b_k) = q_j\}),$
- $syn = \emptyset.$

For a given DFST T, we construct an SNP module Π'_T simulating T as follows:

$$\Pi'_T = (\{a\}, \sigma_1, syn, \{1\}, \{1\}),$$

where

- $\sigma_1 = (1, \{a^{k(2n+1)+i+t}/a^{k(2n+1)+i+t-j} \to a^{n+s} | \delta'(q_i, b_k) = (q_j, c_s)\}),$
- $syn = \emptyset.$

In this work

- For some finite string $w = a_1 a_2 \ldots a_n$, let $w^R = a_n a_{n-1} \ldots a_2 a_1$ (we read w in reverse)
- Some additional results:
  - Any k-DFAO M can be simulated by some 2-neuron SNP module Π_M.
  - Any k-automatic sequence $\mathbf{a} = (a_n)_{n \geq 0}$ can be generated by some 2-neuron SNP module Π.
  - Let $\mathbf{a} = (a_n)_{n \geq 0}$ be a k-automatic sequence. Then there is some 2-neuron SNP module Π where $\Pi(w^R) = a_n$, $w \in \Sigma_k^*$, $[w]_k = n$, and $n \geq 0$.

In this work

- For some finite string $w = a_1 a_2 \ldots a_n$, let $w^R = a_n a_{n-1} \ldots a_2 a_1$ (we read w in reverse)
- Some additional results:
  - Any k-DFAO M can be simulated by some 2-neuron SNP module Π_M.
  - Any k-automatic sequence $\mathbf{a} = (a_n)_{n \geq 0}$ can be generated by some 2-neuron SNP module Π.
  - Let $\mathbf{a} = (a_n)_{n \geq 0}$ be a k-automatic sequence. Then, there is some 2-neuron SNP module Π where $\Pi(w^R\$) = a_n$, $w \in \Sigma_k^*$, $[w]_k = n$, and $n \geq 0$.

In this work

- For some finite string $w = a_1 a_2 \ldots a_n$, let $w^R = a_n a_{n-1} \ldots a_2 a_1$ (we read w in reverse)
- Some additional results:
    - Any k-DFAO M can be simulated by some 2-neuron SNP module Π_M.
    - Any k-automatic sequence $\mathbf{a} = (a_n)_{n \geq 0}$ can be generated by some 2-neuron SNP module Π.
    - Let $\mathbf{a} = (a_n)_{n \geq 0}$ be a k-automatic sequence. Then, there is some 2-neuron SNP module Π where $\Pi(w^R\$) = a_n$, $w \in \Sigma_k^*$, $[w]_k = n$, and $n \geq 0$.

In this work

- For some finite string $w = a_1 a_2 \ldots a_n$, let $w^R = a_n a_{n-1} \ldots a_2 a_1$ (we read w in reverse)
- Some additional results:
  - Any k-DFAO M can be simulated by some 2-neuron SNP module Π_M.
  - Any k-automatic sequence $\mathbf{a} = (a_n)_{n \geq 0}$ can be generated by some 2-neuron SNP module Π.
  - Let $\mathbf{a} = (a_n)_{n \geq 0}$ be a k-automatic sequence. Then, there is some 2-neuron SNP module Π where $\Pi(w^R\$) = a_n$, $w \in \Sigma_k^*$, $[w]_k = n$, and $n \geq 0$.

In this work

- For some finite string $w = a_1 a_2 \dots a_n$, let $w^R = a_n a_{n-1} \dots a_2 a_1$ (we read w in reverse)
- Some additional results:
  - Any k-DFAO M can be simulated by some 2-neuron SNP module Π_M.
  - Any k-automatic sequence $\mathbf{a} = (a_n)_{n \geq 0}$ can be generated by some 2-neuron SNP module Π.
  - Let $\mathbf{a} = (a_n)_{n \geq 0}$ be a k-automatic sequence. Then, there is some 2-neuron SNP module Π where $\Pi(w^R \$) = a_n$, $w \in \Sigma_k^*$, $[w]_k = n$, and $n \geq 0$.

In this work

For a given k-DFAO $M = (Q, \Sigma, \Delta, \delta'', q_1, \tau)$, we have $1 \leq i, j \leq n$, $1 \leq s \leq t$, and $1 \leq k \leq m$. Construction of an SNP module Π_M simulating M, is as follows:

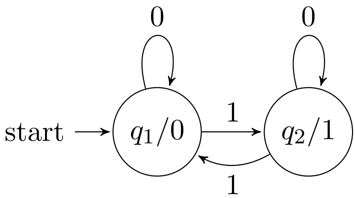
$$\Pi = (\{a\}, \sigma_1, \sigma_2, syn, \{1\}, \{2\}),$$

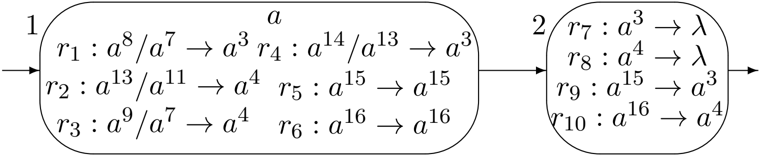
where

- $\sigma_1 = (1, R_1), \sigma_2 = (0, R_2)$,
- $R_1 = \{a^{k(2n+1)+i+t}/a^{k(2n+1)+i+t-j} \rightarrow a^{n+s} | \delta''(q_i, b_k) = q_j, \tau(q_j) = c_s\}$ $\cup \{a^{m(2n+1)+n+t+i} \rightarrow a^{m(2n+1)+n+t+i} | 1 \leq i \leq n\}$,
- $R_2 = \{a^{n+s} \rightarrow \lambda | \tau(q_i) = c_s\} \cup \{a^{m(2n+1)+n+t+i} \rightarrow a^{n+s} | \tau(q_i) = c_s\}$,
- $syn = \{(1, 2)\}$.

In this work

An example.





In summary

- We continued SNP modules investigation:
  - Amend construction problem in simulating DFA and DFST;[6]
  - Reduced number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extended our construction to simulate DFAO;
  - Generating automatic sequences

[6] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In summary

- We continued SNP modules investigation:
  - Amend construction problem in simulating DFA and DFST;[6]
  - Reduced number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extended our construction to simulate DFAO;
  - Generating automatic sequences

[6] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In summary

- We continued SNP modules investigation:
  - Amend construction problem in simulating DFA and DFST;[6]
  - Reduced number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extended our construction to simulate DFAO;
  - Generating automatic sequences

[6] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In summary

- We continued SNP modules investigation:
  - Amend construction problem in simulating DFA and DFST;[6]
  - Reduced number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extended our construction to simulate DFAO;
  - Generating automatic sequences

[6] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

In summary

- We continued SNP modules investigation:
  - Amend construction problem in simulating DFA and DFST;[6]
  - Reduced number of neurons in simulation: from 3 neurons down to 1 neuron;
  - Extended our construction to simulate DFAO;
  - Generating automatic sequences

[6] Ibarra, O., Peréz-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

Final remarks

- More research on SNP modules:
  - Other finite and infinite automata[7]
  - A possibility for "going beyond Turing"?
    - Interactive computations: Persistent Turing Machines[8], interactive components[9]
  - Applications?
    - Human Brain Project[10] (EU), The Brain Initiative[11] (USA)

[7] Freund, R., Oswald, M.: Regular ω-languages defined by finite extended spiking neural P systems. Fundamenta Informaticae, vol. 81(1-2), pp. 65-73 (2008)

[8] Goldin, D.: Persistent Turing Machines as a Model of Interactive Computation. FoIKS 2000, LNCS 1762, pp. 116 – 135.

[9] van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. in Goldin et al. (Eds.) Interactive Computation: The New Paradigm. Springer-Verlag (2006).

[10] https://www.humanbrainproject.eu/

[11] http://braininitiative.nih.gov/

Final remarks

- More research on SNP modules:
  - Other finite and infinite automata[7]
  - A possibility for "going beyond Turing"?
    - Interactive computations: Persistent Turing Machines[8], interactive components[9]
  - Applications?
    - Human Brain Project[10] (EU), The Brain Initiative[11] (USA)

---

[7] Freund, R., Oswald, M.: Regular ω-languages defined by finite extended spiking neural P systems. Fundamenta Informaticae, vol. 81(1-2), pp. 65-73 (2008)

[8] Goldin, D.: Persistent Turing Machines as a Model of Interactive Computation. FoIKS 2000, LNCS 1762, pp. 116 - 135.

[9] van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. in Goldin et al. (Eds.): Interactive Computation: The New Paradigm. Springer-Verlag (2006).

[10] https://www.humanbrainproject.eu/

[11] http://braininitiative.nih.gov/

Final remarks

- More research on SNP modules:
  - Other finite and infinite automata[7]
  - A possibility for “going beyond Turing”?
    - Interactive computations: Persistent Turing Machines[8], interactive components[9]
  - Applications?
    - Human Brain Project[10] (EU), The Brain Initiative[11] (USA)

---

[7] Freund, R., Oswald, M.: Regular ω-languages defined by finite extended spiking neural P systems. Fundamenta Informaticae, vol. 81(1-2), pp. 65-73 (2008)

[8] Goldin, D.: Persistent Turing Machines as a Model of Interactive Computation. FoIKS 2000, LNCS 1762, pp. 116 - 135.

[9] van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. in Goldin et al. (Eds.): Interactive Computation: The New Paradigm. Springer-Verlag (2006).

[10] https://www.humanbrainproject.eu/

[11] http://braininitiative.nih.gov/

Final remarks

- More research on SNP modules:
  - Other finite and infinite automata[7]
  - A possibility for "going beyond Turing"?
    - Interactive computations: Persistent Turing Machines[8], interactive components[9]
  - Applications?
    - Human Brain Project[10] (EU), The Brain Initiative[11] (USA)

---

[7] Freund, R., Oswald, M.: Regular ω-languages defined by finite extended spiking neural P systems. Fundamenta Informaticae, vol. 81(1-2), pp. 65-73 (2008)

[8] Goldin, D.: Persistent Turing Machines as a Model of Interactive Computation. FoIKS 2000, LNCS 1762, pp. 116 - 135.

[9] van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. in Goldin et al. (Eds.): Interactive Computation: The New Paradigm. Springer-Verlag (2006).

[10] https://www.humanbrainproject.eu/

[11] http://braininitiative.nih.gov/

Final remarks

- More research on SNP modules:
  - Other finite and infinite automata[7]
  - A possibility for "going beyond Turing"?
    - Interactive computations: Persistent Turing Machines[8], interactive components[9]
  - Applications?
    - Human Brain Project[10] (EU), The Brain Initiative[11] (USA)

[7]Freund, R., Oswald, M.: Regular ω-languages defined by finite extended spiking neural P systems. Fundamenta Informaticae, vol. 81(1-2), pp. 65-73 (2008)

[8]Goldin, D.: Persistent Turing Machines as a Model of Interactive Computation. FoIKS 2000, LNCS 1762, pp. 116 - 135.

[9]van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. in Goldin et al. (Eds.): Interactive Computation: The New Paradigm. Springer-Verlag (2006).

[10]https://www.humanbrainproject.eu/

[11]http://braininitiative.nih.gov/

Final remarks

- More research on SNP modules:
  - Other finite and infinite automata[7]
  - A possibility for "going beyond Turing"?
    - Interactive computations: Persistent Turing Machines[8], interactive components[9]
  - Applications?
    - Human Brain Project[10] (EU), The Brain Initiative[11] (USA)

---

[7] Freund, R., Oswald, M.: Regular ω-languages defined by finite extended spiking neural P systems. Fundamenta Informaticae, vol. 81(1-2), pp. 65-73 (2008)

[8] Goldin, D.: Persistent Turing Machines as a Model of Interactive Computation. FoIKS 2000, LNCS 1762, pp. 116 - 135.

[9] van Leeuwen, J., Wiedermann, J.: A Theory of Interactive Computation. in Goldin et al. (Eds.): Interactive Computation: The New Paradigm. Springer-Verlag (2006).

[10] https://www.humanbrainproject.eu/

[11] http://braininitiative.nih.gov/

End!

Thank you for your attention!