Matrix Representation of Spiking Neural P Systems

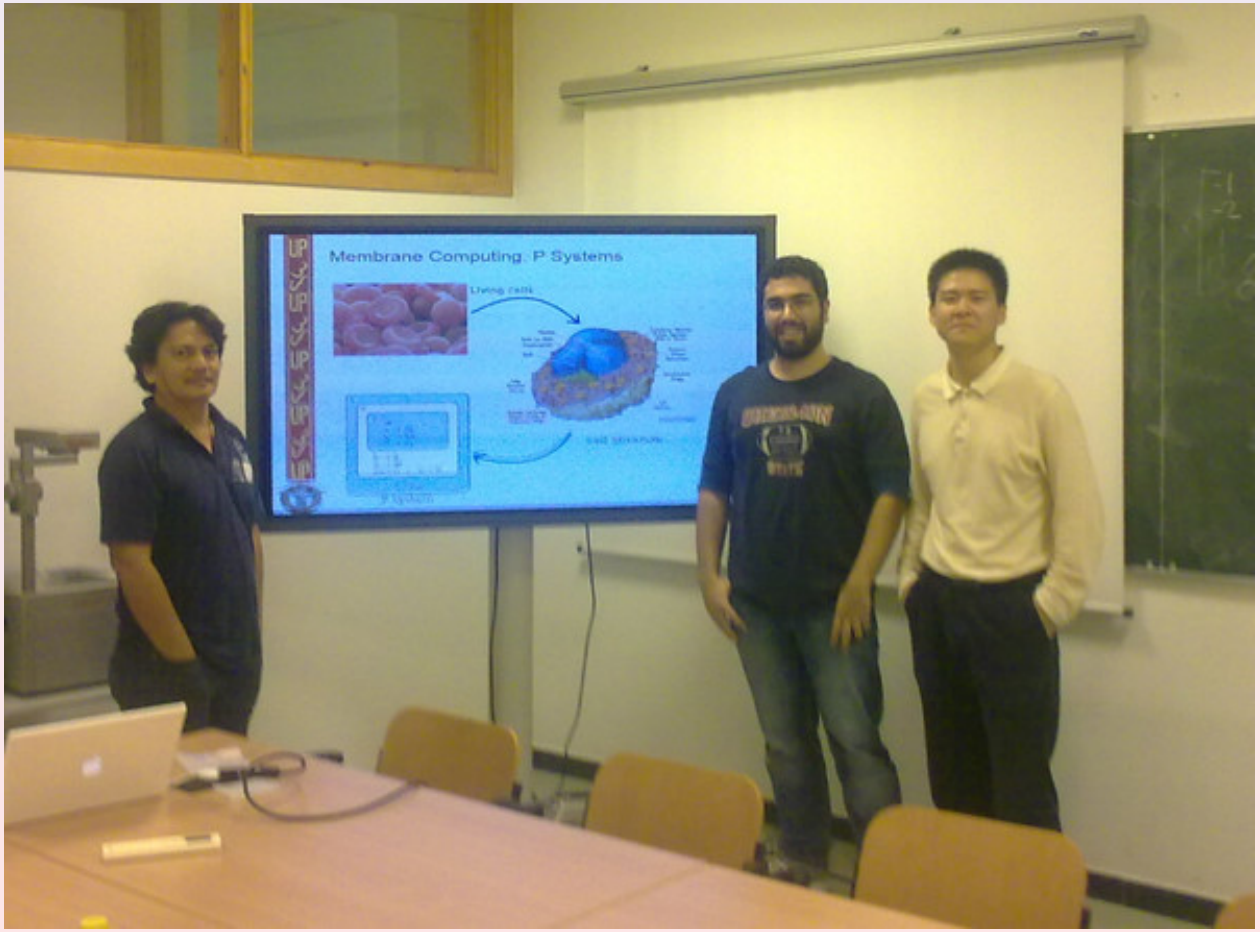
Xiangxiang Zeng, Henry Adorna, Miguel A. Martínez-del-Amor,
Linqiang Pan, Mario J. Pérez-Jiménez

Huazhong University of Science and Technology
University of the Philippines
Universidad de Sevilla

Aug 2010

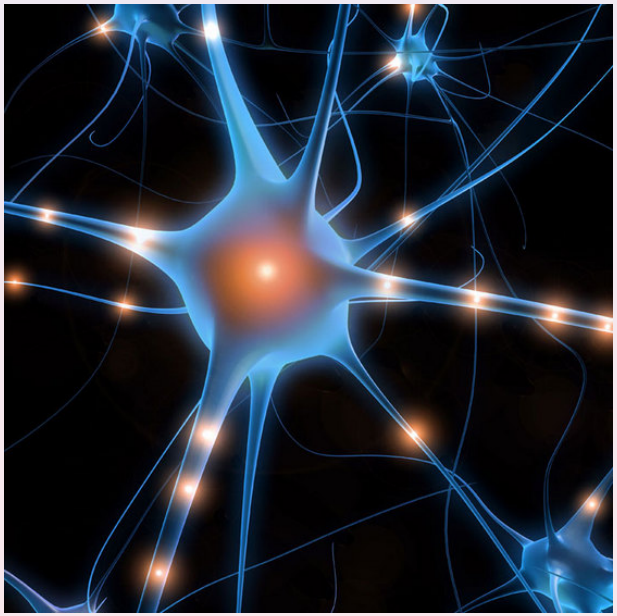
Huazhong University of Science and Technology

Sevilla University, Henry, Miguel, Xiangxiang



Contents

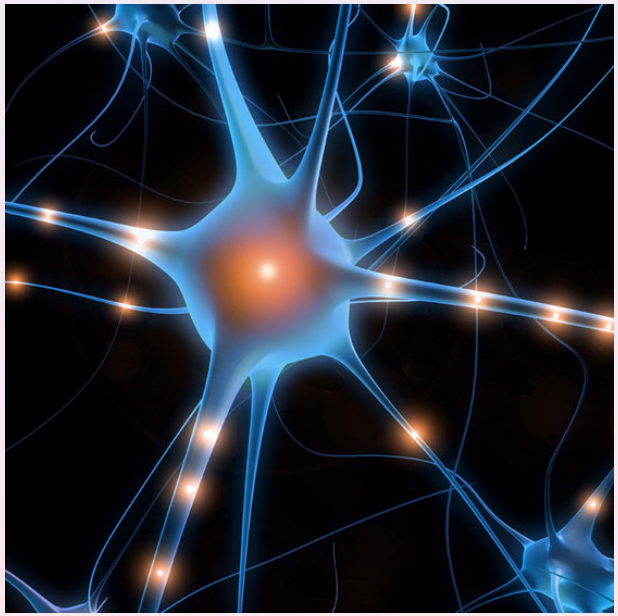
Biological Background



- Biological nervous system
- Neurons communicate by means of spikes

[1] M. Ionescu, Gh. Păun, T. Yokomori. *Fundamenta Informaticae* 2006.

Biological Background



- Biological nervous system
- Neurons communicate by means of spikes

Introduce the feathers of spiking neurons into the membrane computing $\Longrightarrow$ spiking neural P systems(SN P systems)[1].

[1] M. Ionescu, Gh. Păun, T. Yokomori. *Fundamenta Informaticae* 2006.

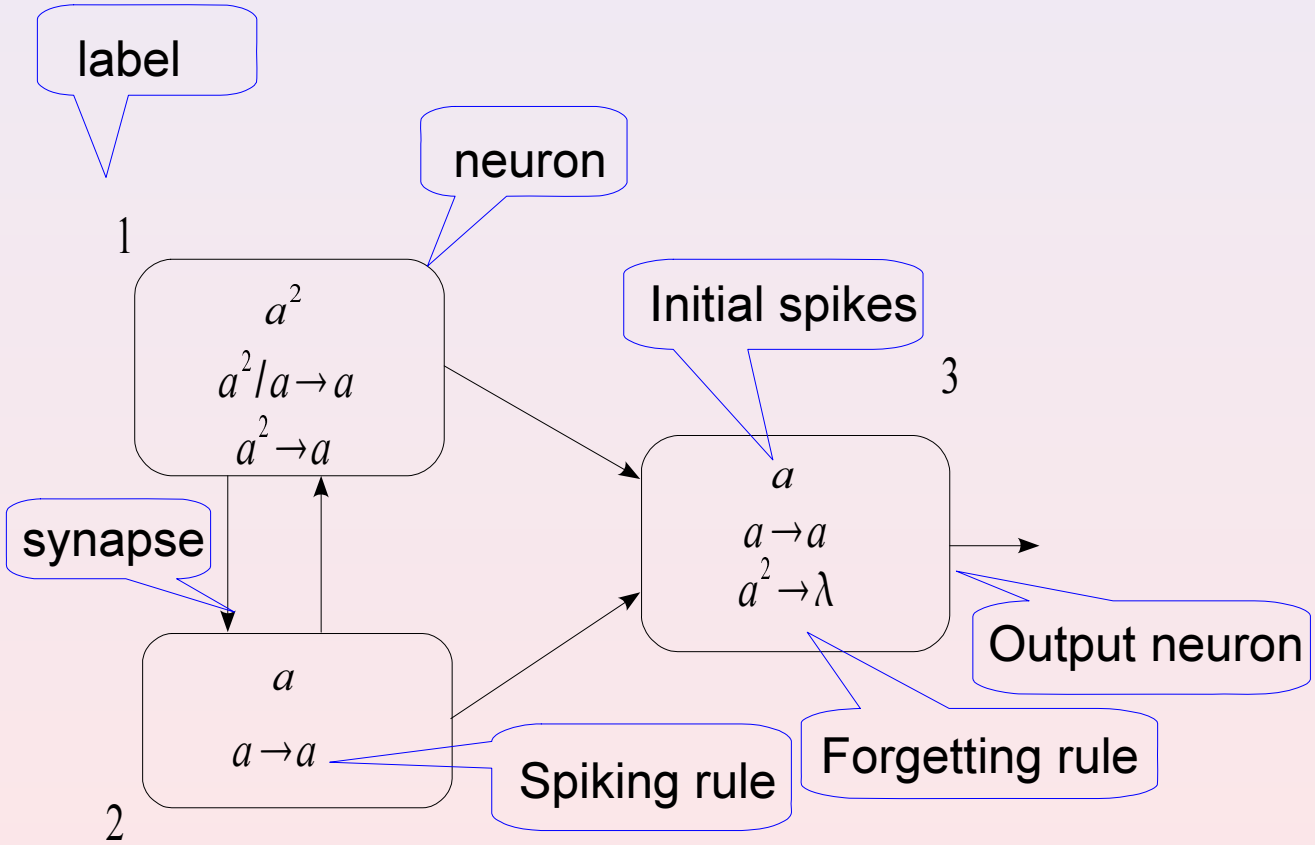
Definition of SN P Systems

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out),$$

1) $O = \{a\}$, the singleton alphabet of spike a;
2) $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$, neurons, where:
   (1) n_i: initial number of spikes
   (2) R_i: a finite set of rules
      a) spiking rule: $E/a^c \rightarrow a^p; d$, E is a regular expression over a;
      b) forgetting rule: $a^s \rightarrow \lambda$, for $s \geq 1$;
3) syn, synapses between neurons;
4) $in, out \in \{1, 2, \ldots, m\}$, the input and the output neurons.

☞ A global clock is used in the synchronous systems.

One simple example



Generate the number set $N - \{1\}$

Configuration and Computation

- *Configuration*
  - $C_k = \langle n_1^{(k)}, n_2^{(k)}, \ldots, n_m^{(k)} \rangle$
    the configuration in step k
  - In particular, we define its **initial configuration** as:

$$C_0 = \langle n_1, n_2, \ldots, n_m \rangle.$$

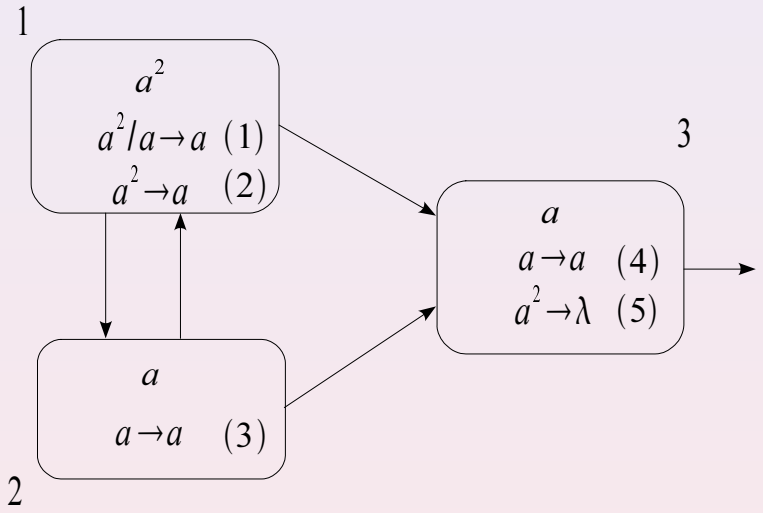
- *Computation*
  A sequence of transitions starting from the initial configuration.

$$\underbrace{C_0 \to C_1 \ldots \to C_k}_{\text{Computation}}$$

Question:

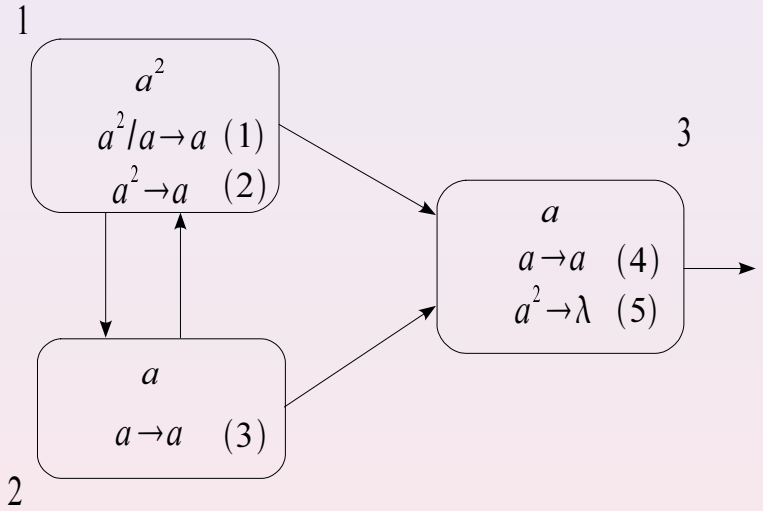
Can we use matrix operation to simulate the computation of an SN P system?

The overview



1. Get the initial configuration vector;
2. Set a total order to all the rules (spiking rules and forgetting rules);
3. Get the spiking transition matrix;
4. Choose the spiking vector;

Initial configuration Vector



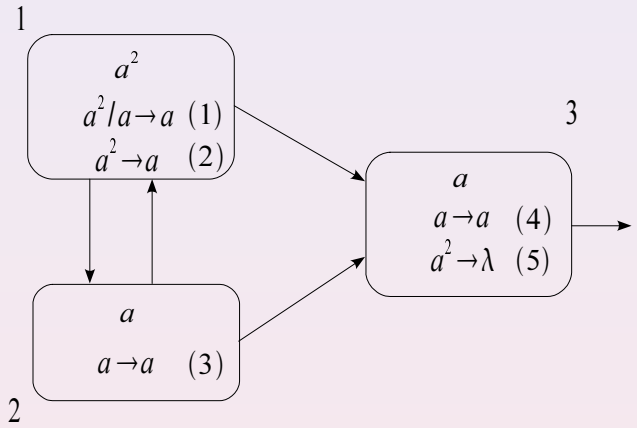
Neuron σ_1 has 2 spikes;

Neuron σ_2 has 1 spike;

Neuron σ_3 has 1 spike.

- The initial configuration vector C_0 is $[2, 1, 1]$.
  - The order of rules is given as above.

Spiking Transition Matrix



The matrix for the above example is

$$\begin{array}{l} r_1: \\ r_2: \\ r_3: \\ r_4: \\ r_5: \end{array} \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} \tag{1}$$

The first row is associated to rule $r_1 : a^2/a \to a$, when r_1 is applied, neuron σ_1 consumes 1 spike, neuron σ_2 gets 1 spike, neuron σ_3 gets 1 spike.

Spiking Transition Matrix

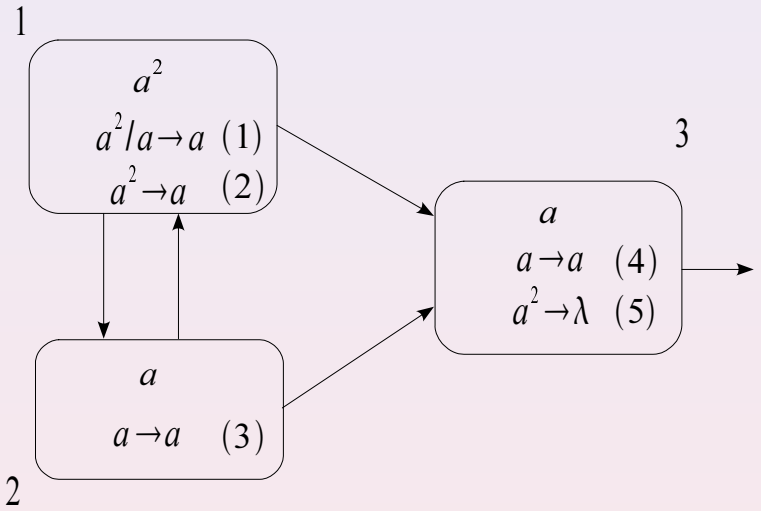
Definition

Let Π be an SN P system with m neurons and n rules. We set a total order $d : 1, \ldots, n$ for all the n rules, so they can be referred as $r_1, \ldots, r_n$. Then, we define the following matrix

$M_\Pi = [a_{ij}]_{n \times m}, where : a_{ij} =$

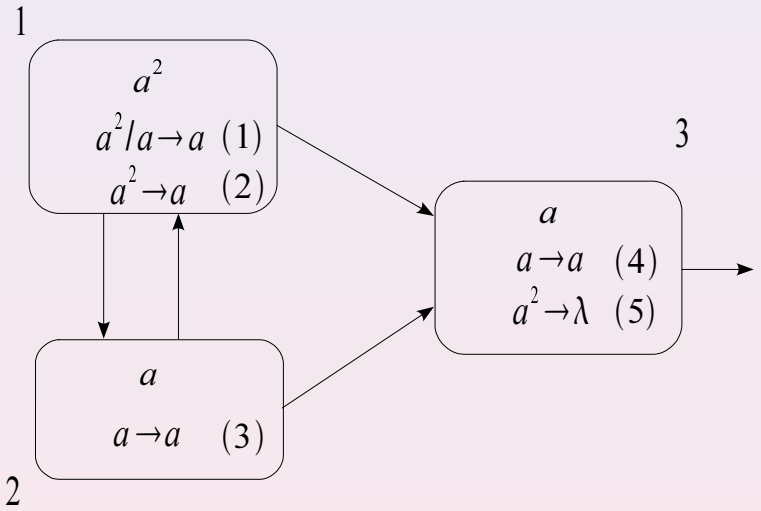
$$\begin{cases} -c, & \text{if rule } r_i \text{ is in neuron } \sigma_j \text{ and it is applied consuming } c \text{ spikes;} \\ p, & \text{if rule } r_i \text{ is in neuron } \sigma_s \ (s \neq j \text{ and } (s, j) \in syn) \\ & \text{and it is applied producing } p \text{ spikes;} \\ 0, & \text{if rule } r_i \text{ is in neuron } \sigma_s \ (s \neq j \text{ and } (s, j) \notin syn). \end{cases}$$

Spiking Vector



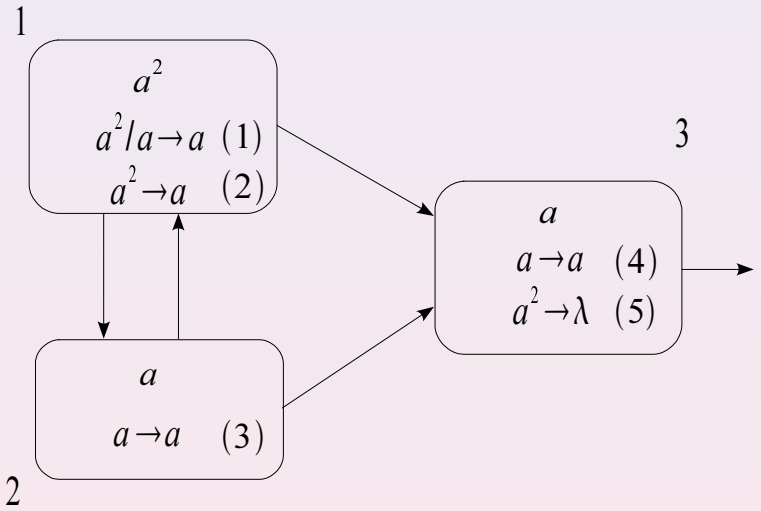
- In neuron σ_1, we can choose rule r_1 or r_2 to apply;

Spiking Vector



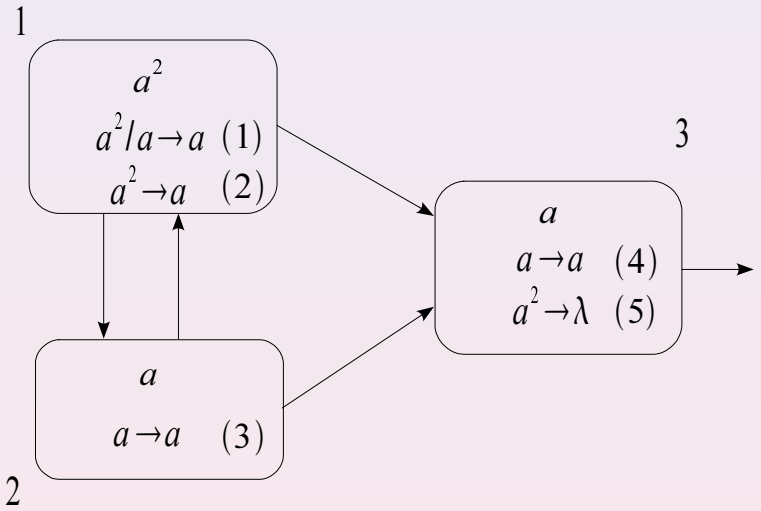
- In neuron σ_1, we can choose rule r_1 or r_2 to apply;
- In neuron σ_2, r_3 will be applied;

Spiking Vector



- In neuron σ_1, we can choose rule r_1 or r_2 to apply;
- In neuron σ_2, r_3 will be applied;
- In neuron σ_3, we apply rule r_4.

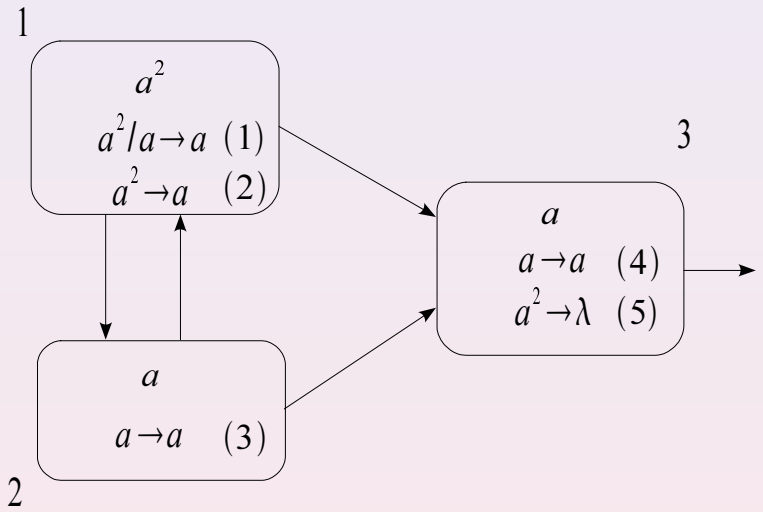
Spiking Vector



- In neuron σ_1, we can choose rule r_1 or r_2 to apply;
- In neuron σ_2, r_3 will be applied;
- In neuron σ_3, we apply rule r_4.

  Note that we cannot use yet rule r_5 because the regular expression a^2 is not yet satisfied.

Spiking Vector



- In neuron σ_1, we can choose rule r_1 or r_2 to apply;
- In neuron σ_2, r_3 will be applied;
- In neuron σ_3, we apply rule r_4.

  Note that we cannot use yet rule r_5 because the regular expression a^2 is not yet satisfied.

  Therefore, our spiking transition vectors would be $[1, 0, 1, 1, 0]$, or $[0, 1, 1, 1, 0]$

Theorem

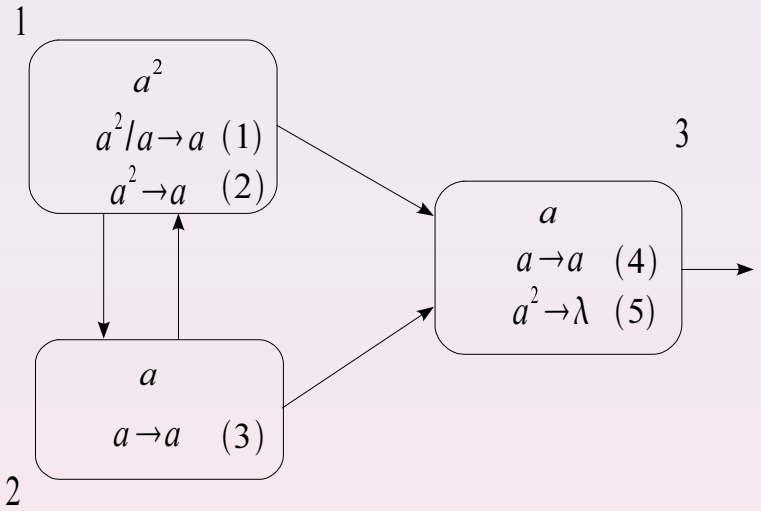
By using the above matrix representation, we can compute an SN P system from one configuration to the next one.

Theorem

Let Π be an SN P system with m neurons and n rules, $d : 1, \ldots, n$ be a total order for the n rules, M_Π the spiking transition matrix of Π, C_k the kth configuration vector, and $s^{(k)}$ the spiking vector at step k, then the configuration C_{k+1} of Π can be obtained by

$$C_{k+1} = C_k \ + \ s^{(k)} \cdot M_\Pi. \tag{2}$$

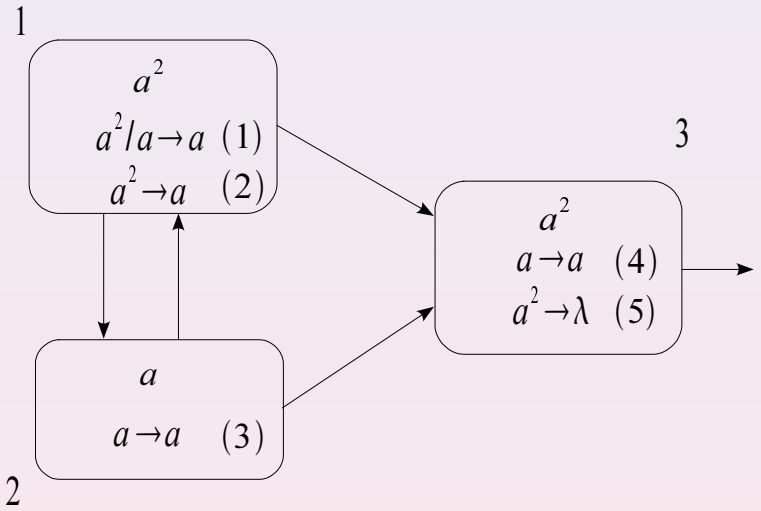
The example



If we choose the rules r_1, r_3, r_4 to apply, that is, $C_0 = [2, 1, 1]$, $s^{(0)} = [1, 0, 1, 1, 0]$, then the next configuration is:

$$C_1 = (2 \quad 1 \quad 1) + (1 \quad 0 \quad 1 \quad 1 \quad 0) \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} = (2 \quad 1 \quad 2) \tag{3}$$

The example



In the next step, we can choose r_1, r_3, r_5 to apply, so our spiking vector is $[1, 0, 1, 0, 1]$, then the next configuration is:

$$C_2 = (2 \quad 1 \quad 2) + (1 \quad 0 \quad 1 \quad 0 \quad 1) \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} = (2 \quad 1 \quad 2) \tag{4}$$

Conclusion

In this way:

- the skeleton of an SN P system can be represented by a spiking transition matrix;
- configuration vector are defined for configuration;
- spiking vector are defined for choosing the rules

With these algebraic representation, we can use matrix operations to compute from one configuration to the next configuration.

Observation 1

we give the following observations on our definition of spiking transition matrices:

Observation

Every row of our spiking transition matrix has **exactly one** *negative entry.*

For row i which is related to rule $r_i : E/a^c \rightarrow a^p$ of neuron σ_j, it will consume $c \geq 1$ spikes in neuron σ_j, so the entry in the position (i, j) will be negative.

Observation 2

Observation

each column of a spiking transition matrix has at least one negative entry.

The column j is associated with neuron σ_j. Assume the rules in neuron σ_j are r_m, $r_n, \ldots$. These rules consume spikes of neuron σ_i when they are applied. So the corresponding entries $(m, j), (n, j), \ldots$ in the spiking transition matrix are negative numbers.

Matrix Representation for WSN P Systems

Next, we will consider matrix representation for a special variant of SN P systems – SN P systems with weights (or, WSN P systems), which was introduced in:
J. Wang, H. J. Hoogeboom, L. Pan, Gh. Păun and M. J. Pérez-Jiménez, Spiking neural P systems with weights, WMC10, Curtea de Arges, Romania, 2009, 514–533, Neural Computation, in press.

The features of WSN P Systems

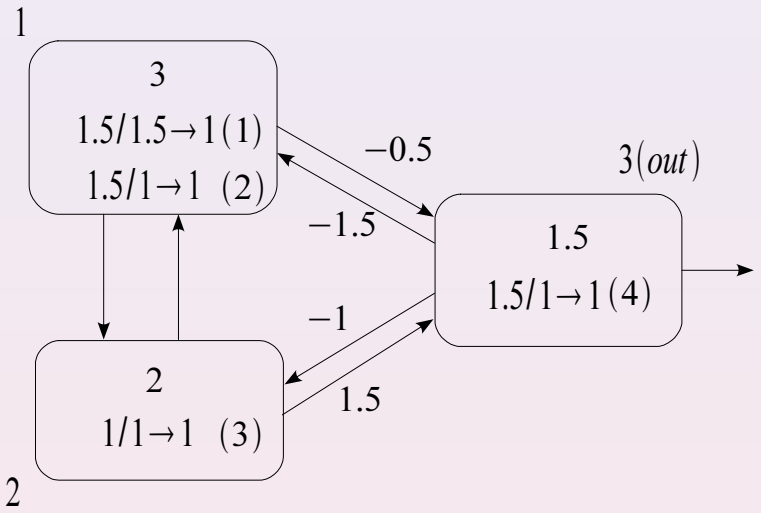
In this variant of SN P systems:

- Neurons have (Positive or negative)potential inside;
- (Positive or negative) weights are considered;
- The rules do not have regular expression, a neuron fires when the potential of that neuron equals to a given threshold.
- If a neuron has the potential less than its threshold, then the neuron returns to the resting potential 0.
- The involved numbers – weights, thresholds, potential consumed by each rule – can be real (computable) numbers.

Matrices for WSN P systems

Similar to SN P systems, some matrices are defined:

- configuration vectors are defined to represent configurations;
- Spiking vectors are defined to represent the application of rules;
- Spiking transition matrix is defined to store the information of the amount of potential consumed (or received) by each neuron when each rule is applied.



Spiking transition matrix for the above system

$$M_{WSNP} = \begin{pmatrix} -1.5 & 1 & -0.5 \\ -1 & 1 & -0.5 \\ 1.5 & -1 & 1.5 \\ -1.5 & -1 & -1.5 \end{pmatrix} \tag{5}$$

In WSN P systems, when the potential of a neuron is smaller than its spiking threshold, then this potential vanishes, the potential of the neuron is set to zero.

In order to describe which neurons forget their potentials, *forgetting vector* is defined.

Definition (Forgetting vector)

$$forg^{(k)} = (f_1^{(k)}, f_2^{(k)}, \ldots, f_m^{(k)}),$$

where:

$$f_i^{(k)} = \begin{cases} 1 & \text{if the potential in neuron } \sigma_j \text{ is} \\ & \text{less than its spiking threshold;} \\ 0 & \text{otherwise.} \end{cases}$$

For WSN P systems, the next configuration C_{k+1} can be computed using the following theorem.

Theorem

Let Π be a WSN P system with m neurons and n rules, $d : 1, \ldots, n$ be a total order for the n rules, M_Π the spiking transition matrix of Π, C_k the kth configuration vector, $s^{(k)}$ the spiking vector at step k, and $forg^{(k)}$ the forgetting vector at step k. Then the configuration C_{k+1} of Π can be obtained by

$$C_{k+1} = C_k \ + \ s^{(k)} \cdot M_\Pi - forg^{(k)} \odot C_k. \tag{6}$$

Conclusion and Future Work

- It is not difficult to see that such matrix representation is also suitable for other variants of SN P systems, such as asynchronous SN P systems and SN P systems with exhaustive use of rules.

Conclusion and Future Work

- It is not difficult to see that such matrix representation is also suitable for other variants of SN P systems, such as asynchronous SN P systems and SN P systems with exhaustive use of rules.
- The spiking transition matrix is related to the structure of system only, so the elements of the matrix are determined initially. During the computation of a system, it is only needed to decide the spiking vector by checking the current configuration vector and the regular expressions of rules. In general, such algebraic representation is easy to be programmed for computer simulation.

Future Works 2

The systems considered in this work have no delay, which corresponds to the biological feature that neurons have refractory time. It is open how to represent the computations of SN P systems with delay by matrices.

Future Works 3

Implement of SN P systems

- Matrix computation is convenient for computer programming. With matrix, it is easy to implement SN P systems (especially WSN P systems) in some parallel computing architecture, such as CUDA (an acronym for Compute Unified Device Architecture) for graphic card (GPU). Because the matrix computation can be processed in parallel way. That is, every processor can deal with a part of matrix.

Thank you

Thank you!!
Questions?