Matrix Representation of Spiking Neural P Systems

Xiangxiang Zeng[1], Henry Adorna[2], Miguel Ángel Martínez-del-Amor[3], Linqiang Pan[1], and Mario J. Pérez-Jiménez[3]

[1] Image Processing and Intelligent Control Key Laboratory of Education Ministry
Department of Control Science and Engineering
Huazhong University of Science and Technology
Wuhan 430074, Hubei, China
xzeng@foxmail.com,
lqpan@mail.hust.edu.cn
[2] Department of Computer Science
(Algorithms and Complexity)
University of the Philippines
Diliman 1101 Quezon City, Philippines
hnadorna@dcs.upd.edu.ph
[3] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
{mdelamor,marper}@us.es

Abstract. Spiking neural P systems (SN P systems, for short) are a class of distributed parallel computing devices inspired from the way neurons communicate by means of spikes. In this work, a discrete structure representation of SN P systems with extended rules and without delay is proposed. Specifically, matrices are used to represent SN P systems. In order to represent the computations of SN P systems by matrices, configuration vectors are defined to monitor the number of spikes in each neuron at any given configuration; transition net gain vectors are also introduced to quantify the total amount of spikes consumed and produced after the chosen rules are applied. Nondeterminism of the systems is assured by a set of spiking transition vectors that could be used at any given time during the computation. With such matrix representation, it is quite convenient to determine the next configuration from a given configuration, since it involves only multiplication and addition of matrices after deciding the spiking transition vector.

1 Introduction

Membrane computing was initiated by Păun [6] and has developed very rapidly (already in 2003, ISI considered membrane computing as "fast emerging

research area in computer science", see `http://esi-topics.com`). It aims to abstract computing ideas (data structures, operations with data, computing models, etc.) from the structure and the functioning of single cell and from complexes of cells, such as tissues and organs, including the brain. The obtained models are distributed and parallel computing devices, called *P systems*. For updated information about membrane computing, please refer to [8].

This work deals with a class of neural-like P systems, called *spiking neural P systems* (SN P systems, for short) [3]. SN P systems were inspired by the neurophysiological behavior of neurons (in brain) sending electrical impulses along axons to other neurons, with the aim of incorporating specific ideas from spiking neurons into membrane computing. Generally speaking, in an SN P system the processing elements are called *neurons* and are placed in the nodes of a directed graph, called the *synapse graph*. The content of each neuron consists of a number of copies of a single object type, namely the *spike*. Each neuron may also contain rules which allow to remove a given number of spikes from it, or send spikes (possibly with a delay) to other neurons. The application of every rule is determined by checking the content of the neuron against a regular set associated with the rule.

Representation of P systems by discrete structures has been one topic in the field of membrane computing. One of the promising discrete structures to represent P systems is matrix. Models with matrices as their representation have been helpful to physical scientists – biologists, chemists, physicists, engineers, statisticians, and economists – solving real world problems. Recently, matrix representation was introduced for represent a restricted form of cell-like P systems without dissolution (where only non-cooperative rules are used) [2]. It was proved that with an algebraic representation P systems can be easily simulated and computed backward (that is, to find all the configurations that produce a given one in one computational step).

In this work, a matrix representation of SN P systems without delay is proposed, where configuration vectors are defined to represent the number of spikes in neurons; spiking vectors are used to denote which rules will be applied; a spiking transition matrix is used to describe the skeleton of a system; the transition net gain vectors are also introduced to quantify the total amount of spikes consumed and produced after the chosen rules are applied. With this algebraic representation, matrix transition can be used to compute the next configuration from a given one.

In addition, we consider another variant of SN P systems, SN P systems with weights (WSN P systems, for short), which were introduced in [9]. In these systems, each neuron has a potential and a given threshold, whose values are real (computable) numbers. Each neuron fires when its potential is equal to its threshold; at that time, part of the potential is consumed and a unit potential is produced (a spike). The unit potential is passed to neighboring neurons multiplied with the weights of synapses. The weights of synapses can also be real (computable) numbers. This variant of SN P system allows real numbers to be computed by the system.

A matrix over $\mathbb{R}_C$ is defined to represent WSN P systems, where vectors are defined to represent the potentials in neurons and the application of rules. In particular, 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). Therefore, a forgetting vector is introduced to denote the potential vanishing from neurons. It is also shown that matrix representation is convenient for deciding the next configuration of the system from a given configuration.

The rest of this paper is organized as follows. In the next section, the definition of SN P systems is introduced. In Section 3 the matrix representation of SN P systems is given, and an example is given to illustrate how to represent the computation of an SN P system by matrix transition. In Section 4 the matrix representation method is extended to WSN P systems. Conclusions and remarks are given in Section 5.

2 Spiking Neural P Systems

In this section, a restricted variant of SN P systems, SN P systems without delay, is introduced.

Definition 1. *An SN P system without delay, of degree $m \geq 1$, is a construct of the form*

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

where:

1. $O = \{a\}$ *is the singleton alphabet (a is called spike);*
2. $\sigma_1, \ldots, \sigma_m$ *are neurons, of the form*

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m,$$

   where:
   a) $n_i \geq 0$ *is the initial number of spikes contained in σ_i;*
   b) R_i *is a finite set of rules of the following two forms:*
      (1) $E/a^c \to a^p$, *where E is a regular expression over a, and $c \geq 1$, $p \geq 1$, with the restriction $c \geq p$;*
      (2) $a^s \to \lambda$, *for $s \geq 1$, with the restriction that for each rule $E/a^c \to a^p$ of type (1) from R_i, $a^s \notin L(E)$;*
3. $syn = \{(i, j) \mid 1 \leq i, j \leq m, i \neq j\}$ *(synapses between neurons);*
4. $in, out \in \{1, 2, \ldots, m\}$ *indicate the input and output neurons, respectively.*

The rules of type (1) are spiking (also called firing) rules, which are applied as follows. If the neuron σ_i contains k spikes, and $a^k \in L(E), k \geq c$, then the rule $E/a^c \to a^p \in R_i$ can be applied. This means that consuming (removing) c spikes (thus only $k-c$ spikes remain in σ_i), the neuron is fired, and it produces p spikes; these spikes are transported to all neighbor neurons by outgoing synapses. If a rule $E/a^c \to a^p$ has $E = a^c$, then it is written in the simplified form $a^c \to a^p$.

The rules of type (2) are forgetting rules; they are applied as follows: if the neuron σ_i contains exactly s spikes, then the rule $a^s \to \lambda$ from R_i can be used, meaning that all s spikes are removed from σ_i.

In each time unit, if a neuron σ_i can apply one of its rules, then a rule from R_i must be applied. Since two spiking rules, $E_1/a^{c_1} \to a^{p_1}$ and $E_2/a^{c_2} \to a^{p_2}$, can have $L(E_1) \cap L(E_2) \neq \emptyset$, it is possible that two or more rules are applicable in a neuron. In that case, only one of them is chosen and applied nondeterministically. However, note that, by definition, if a spiking rule is applicable, then no forgetting rule is applicable, and vice versa.

Thus, the rules are applied in the sequential manner in each neuron, at most one in each step, but neurons function in parallel with each other. It is important to notice that the applicability of a rule is established based on the total number of spikes contained in the neuron.

The configuration of the system is described by the number of spikes present in each neuron. Using the rules as described above, one can define transitions among configurations. Any sequence of transitions starting in the initial configuration is called a computation. A computation halts if it reaches a configuration where no rule can be applied. The result of a computation is the number of steps elapsed between the first two spikes sent by the output neuron to the environment during the computation.

3 Matrix Representation of SN P Systems

In this section, a matrix representation of SN P systems is given. Based on this representation, it is shown how the computation of SN P systems can be represented by operations with matrices.

As mentioned in the above section, a configuration of the system is described by the number of spikes present in each neuron. Here, vectors are used to represent configurations.

Definition 2 (Configuration Vectors). *Let Π be an SN P system with m neurons, the vector $C_0 = (n_1, n_2, \ldots, n_m)$ is called the* **initial configuration vector** *of Π, where n_i is the amount of the initial spikes present in neuron σ_i, $i = 1, 2, \ldots, m$ before a computation starts.*

*In a computation, for any $k \in \mathbb{N}$, the vector $C_k = (n_1^{(k)}, n_2^{(k)}, \ldots, n_m^{(k)})$ is called the k***th configuration vector** *of the system, where $n_i^{(k)}$ is the amount of spikes in neuron σ_i, $i = 1, 2, \ldots, m$ after the kth step of the computation.*

In order to describe which rules are chosen and applied in each configuration, *spiking vectors* are defined.

Definition 3 (Spiking Vectors). *Let Π be an SN P system with m neurons and n rules, and $C_k = (n_1^{(k)}, n_2^{(k)}, \ldots, n_m^{(k)})$ be the kth configuration vector of Π. Assume a total order $d : 1, \ldots, n$ is given for all the n rules, so the rules can be referred as $r_1, \ldots, r_n$. A* **spiking vector** *$s^{(k)}$ is defined as follows:*

$$s^{(k)} = (r_1^{(k)}, r_2^{(k)}, \ldots, r_n^{(k)}),$$

where:

$$r_i^{(k)} = \begin{cases} 1, \textit{ if the regular expression } E_i \textit{ of rule } r_i \textit{ is satisfied by the} \\ \quad \textit{number of spikes } n_j^{(k)} \textit{ (rule } r_i \textit{ is in neuron } \sigma_j \textit{) and} \\ \quad \textit{rule } r_i \textit{ is chosen and applied;} \\ 0, \textit{ otherwise.} \end{cases}$$

In particular, $s^{(0)} = (r_1^{(0)}, r_2^{(0)}, \ldots, r_n^{(0)})$ *is called the* **initial spiking vector**.

The application of each rule will change the number of spikes in some neurons, for example, when the rule $r_i : E/a^c \to a^p$ is applied in neuron σ_j, it consumes c spikes in σ_j, and emits p spikes; these p spikes are immediately delivered to all the neurons σ_s such that $(j, s) \in syn$. Here, a *spiking transition matrix* is defined to denote the amount of spikes consumed (or received) by each neuron via each rule.

Definition 4 (Spiking Transition Matrix). *Let* Π *be an SN P system with* m *neurons and* n *rules, and* $d : 1, \ldots, n$ *be a total order given for all the* n *rules. The* **spiking transition matrix** *of the system* Π*,* M_Π*, is defined as follows:*

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

where:

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

In a spiking transition matrix, the row i is associated with the rule $r_i : E/a^c \to a^p$. Assume that the rule r_i is in neuron σ_j. When the rule r_i is applied, it consumes c spikes in neuron σ_j; neuron σ_s ($s \neq j$ and $(j, s) \in syn$) receives p spikes from neuron σ_j; neuron σ_s ($s \neq j$ and $(j, s) \notin syn$) receives no spike from neuron σ_j. By the definition of spiking transition matrix, the entry in the position (i, j) is a negative number; the other entries in the row i are non-negative numbers. So the following observation holds:

Observation 1: each row of a spiking transition matrix has exactly one negative entry.

In a spiking transition matrix, the column i is associated with neuron σ_i. For an SN P system, without loss of generality, it can be assumed that each neuron has at least one rule inside (if a neuron has no rule inside, it just stores spikes, sending no spikes to other neurons or environment, so it can be deleted without any influence to the computational result of the system). Assume the rules in neuron σ_i are $r_m, r_n, \ldots$. These rules consume spikes of neuron σ_i when they are applied. So the corresponding entries $(m, i), (n, i), \ldots$ in the spiking transition matrix are negative numbers, and the following observation holds:

Observation 2: each column of a spiking transition matrix has at least one negative entry.

In the following, it will be shown that how matrices representing SN P systems can be used to represent the computation of SN P systems by operating with matrices. Before the matrix operations for SN P systems are formally defined, a simple example is provided as follows.

Example 1. Let us consider an SN P system $\Pi = (\{a\}, \sigma_1, \sigma_1, \sigma_3, syn, out)$ that generates the set $\mathbb{N}$ of natural numbers excluding 1, where $\sigma_1 = (2, R_1)$, with $R_1 = \{a^2/a \to a, a^2 \to a\}$; $\sigma_2 = (1, R_2)$, with $R_2 = \{a \to a\}$; $\sigma_3 = (1, R_3)$, with $R_3 = \{a \to a, a^2 \to \lambda\}$; $syn = \{(1,2), (1,3), (2,1), (2,3)\}$; $out = 3$. Π is also represented graphically in Figure 1, which may be easier to understand.

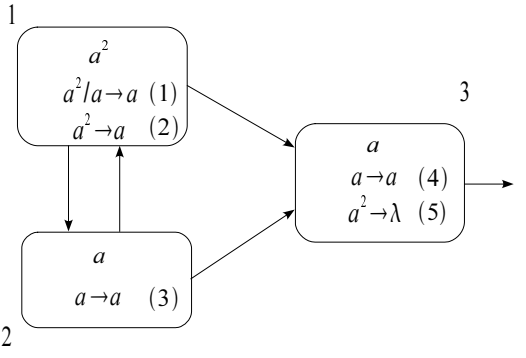

Fig. 1. An SN P system Π that generates the set $\mathbb{N} - \{1\}$

In order to represent the above SN P system Π in a matrix, a total order is set for all the rules in the system, which can be seen in Figure 1. With this order, the rules can be denoted by $r_1, \ldots, r_5$.

Let $M_{\Pi_1} = [a_{ij}]_{5\times 3}$ be the spiking transition matrix for Π. By Definition 4, the row i of M_Π is associated with the rule $r_i : E/a^c \to a^p, c \geq 1, p \geq 0$ in system Π. The entries a_{i1}, a_{i2}, a_{i3} are the amount of spikes which neurons $\sigma_1, \sigma_2, \sigma_3$ will get (or consume) when rule r_i is applied.

The spiking transition matrix for the SN P system Π depicted in Figure 1 is as follows.

$$M_\Pi = \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} \tag{1}$$

Initially, neurons σ_1, σ_2, and σ_3 have 2, 1, and 1 spike(s), respectively. According to Definition 2, the initial configuration vector for system Π would be $C_0 = (2, 1, 1)$. Since neuron σ_1 has two rules r_1 and r_2 that are applicable in

the initial transition, one of the rules could be chosen, the initial spiking transition vector would be $(1,0,1,1,0)$ or $(0,1,1,1,0)$ by Definition 3. Note that the rule r_5 is not applicable because the regular expression a^2 is not satisfied in neuron σ_3.

If the rule $r_1 : a^2/a \rightarrow a$ is applied, it consumes one spike in neuron σ_1 and sends one spike to neurons σ_2 and σ_3, respectively; at the same time, neuron σ_2 sends one spike to each of the neurons σ_1 and σ_3. In this step, the net gain of neuron σ_1 is 0 spike (it consumes 1 spike by r_1 and receives 1 spike from neuron σ_2); the net gain of neuron σ_2 is 0 spike (it consumes 1 spike by r_3 and receives 1 spike from neuron σ_1); the net gain of neuron σ_3 is 1 spike (it consumes 1 spike by rule r_5 and receives 1 spike from each of the neurons σ_1 and σ_2). After this step, the numbers of spikes in neurons σ_1, σ_2 and σ_3 are 2, 1 and 2, respectively.

The illustration above explained intuitively how an SN P system computes from one configuration to the succeeding one. In order to use matrix operations to represent it, the following definitions are needed:

Definition 5 (Transition Net Gain Vector). *Let Π be an SN P system with m neurons and n rules, and $C_k = (n_1^{(k)}, n_2^{(k)}, \ldots, n_m^{(k)})$ be the kth configuration vector of Π. The* **transition net gain vector** *at step k is defined as $NG^{(k)} = C_{k+1} - C_k$.*

Lemma 1. *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, and $s^{(k)}$ the spiking vector at step k. Then the transition net gain vector at step k can be obtained by*

$$NG^{(k)} = s^{(k)} \cdot M_\Pi. \tag{2}$$

Proof. Assume that $M_\Pi = [a_{ij}]_{m \times n}$, $s^{(k)} = (r_1^{(k)}, r_2^{(k)}, \ldots, r_n^{(k)})$, and $NG^{(k)} = (g_1, g_2, \ldots, g_m)$. Note that the spiking vector $s^{(k)}$ is a $\{0,1\}$-vector that identifies the rules that would be applied at step k. Thus, $\Sigma_{i=1}^{n} r_i^{(k)} a_{ij}$ represents the total amount of spikes received and consumed by neuron σ_j after applying the rules identified by $s^{(k)}$. Therefore, the net gain of neuron σ_j is $g_j = \Sigma_{i=1}^{n} r_i^{(k)} a_{ij}$, for all $j = 1, 2, \ldots, m$. That is, $NG^{(k)} = s^{(k)} \cdot M_\Pi$.

Theorem 1. *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*

$$C_{k+1} = C_k + s^{(k)} \cdot M_\Pi. \tag{3}$$

Proof. This results follows directly from the preceding Lemma.

Let us go back to the example shown in Figure 1. Given the initial configuration vector $C_0 = (2,1,1)$, the next configuration of system Π can be computed as follows.

If the rules r_1, r_3, r_4 are chosen to be applied, the spiking vector is $s^{(0)} = (1, 0, 1, 1, 0)$, and the next configuration is

$$C_1 = (2, 1, 1) \ + \ (1, 0, 1, 1, 0) \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} = (2, 1, 2). \tag{4}$$

In the next step, r_1, r_3, r_5 are chosen to be applied, the spiking vector is $(1, 0, 1, 0, 1)$, and the next configuration is

$$C_2 = (2, 1, 2) \ + \ (1, 0, 1, 0, 1) \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} = (2, 1, 2), \tag{5}$$

where the transition net gain vector is

$$NG = (1, 0, 1, 0, 1) \begin{pmatrix} -1 & 1 & 1 \\ -2 & 1 & 1 \\ 1 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \end{pmatrix} = (0, 0, 0). \tag{6}$$

Equation (6) means that the configuration of the system remains unchanged as long as the rules r_1, r_3 and r_5 are chosen to be applied. However, at any moment, starting from the first step of computation, neuron σ_1 can choose to apply the rule $r_2 : a^2 \to a$. In this case, system will go to another configuration. The checking is left to the readers.

The following Corollary is a direct consequence of the preceding Theorem.

Corollary 1. *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 $s^{(k-1)}$ the spiking vector at step $k-1$, then the previous configuration C_{k-1} is*

$$C_{k-1} = C_k \ - \ s^{(k-1)} \cdot M_\Pi. \tag{7}$$

In the above matrix representation, the spikes sent to the environment are not considered. In order to consider the spikes sent to the environment, an *augmented spiking transition matrix* is introduced.

Definition 6 (Augmented Transition Spiking Matrix). *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, and M_Π the $n \times m$ spiking transition matrix of Π. An* **augmented spiking transition matrix** *is defined as*

$$[M_\Pi \,|\, e]_{n \times (m+1)},$$

where the column $e = (e_1, e_2, \ldots, e_n)^T$ *represents the spikes sent to the environment, with:*

$$e_i = \begin{cases} p, & \text{if rule } r_i \text{ is in the output neuron and it is applied producing } p \text{ spikes;} \\ 0, & \text{if rule } r_i \text{ is not in the output neuron.} \end{cases}$$

Correspondingly, the **augmented configuration vector** after the kth step in the computation is defined as

$$C_k = (n_1^{(k)}, n_2^{(k)}, \ldots, n_m^{(k)}, n_e^{(k)}),$$

where $n_i^{(k)}$ is the amount of spikes in neuron σ_i, for all $i = 1, 2, \ldots, m$, $n_e^{(k)}$ is the amount of spikes collected in the environment. Using this vector instead of the configuration vector in Definition 2 allows us to monitor the output of the system.

4 Matrix Representation for WSN P Systems

In this section, matrix representation for spiking neural P systems with weights (WSN P systems, for short) is investigated. Instead of counting spikes as in usual SN P systems, each neuron in WSN P systems contains a potential, which can be expressed by a computable real number. Each neuron fires when its potential is equal to the given threshold. The execution of a rule consumes part of the potential and produces a unit potential. This unit potential passes to neighboring neurons multiplied with the weights of synapses. In an SN P system with weights, the involved numbers – weights, thresholds, potential consumed by each rule – can be real (computable) numbers. Formally, the system is defined as follows.

Definition 7. *An SN P system with weights, of degree* $m \geq 1$*, is a construct of the form*

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

where:

1. $\sigma_1, \ldots, \sigma_m$ *are neurons of the form* $\sigma_i = (p_i, R_i), 1 \leq i \leq m$*, where:*
   a) $p_i \in \mathbb{R}_c$ *is the initial potential in* σ_i*;*
   b) R_i *is a finite set of rules of the form* $T_i/d_s \to 1$*,* $s = 1, 2, \ldots, n_i$ *for some* $n_i \geq 1$*, where* $T_i \in \mathbb{R}_c$*,* $T_i \geq 1$*, is the firing threshold potential of neuron* σ_i*, and* $d_s \in \mathbb{R}_c$ *with the restriction* $0 < d_s \leq T_i$*;*
2. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \times \mathbb{R}_c$ *are synapses between neurons, with* $i \neq j$*,* $w \neq 0$ *for each* $(i, j, w) \in syn, 1 \leq i, j \leq m$*;*
3. $in, out \in \{1, 2, \ldots, m\}$ *indicate the input and output neurons, respectively.*

The spiking rules are applied as follows. Assume that at a given moment, neuron σ_i has the potential equal to p. If $p = T_i$, then any rule $T_i/d_s \to 1 \in R_i$ can be applied. The execution of this rule consumes an amount of d_s of the potential (thus leaving the potential $T_i - d_s$) and prepares one unit potential (we also say a spike) to be delivered to all the neurons σ_j such that $(i, j, w) \in syn$. Specifically, each of these neurons σ_j receives a quantity of potential equal to w, which is added to the existing potential in σ_j. Note that w can be positive or negative, hence the potential of the receiving neuron is increased or decreased. The potential emitted by a neuron σ_i passes immediately to all neurons σ_j such that $(i, j, w) \in syn$, that is, the transition of potential takes no time. If a neuron σ_i spikes and it has no outgoing synapse, then the potential emitted by neuron σ_i is lost.

The main feature of this system is: (1) each neuron σ_i has only one fixed threshold potential T_i; (2) if a neuron has the potential equal to its threshold potential, then all rules associated with this neuron are enabled, and only one of them is non–deterministically chosen to be applied; (3) when a neuron spikes, there is always only one unit potential (a spike) emitted.

If neuron σ_i has the potential p such that $p < T_i$, then the neuron σ_i returns to the resting potential 0. If neuron σ_i has the potential p such that $p > T_i$, then the potential p keeps unchanged.

Summing up, if neuron σ_i has potential p and receives potential k at step t, then at step $t + 1$ it has the potential p', where:

$$p' = \begin{cases} k, & \text{if } p < T_i; \\ p - d_s + k, & \text{if } p = T_i \text{ and rule } T_i/d_s \to 1 \text{ is applied;} \\ \text{p+k}, & \text{if } p > T_i. \end{cases}$$

The configuration of the system is described by the distribution of potentials in neurons. Using the rules described above, one can define transitions among configurations. Any sequence of transitions starting in the initial configuration is called a computation. A computation halts if it reaches a configuration where no rule can be applied. The result of a computation is the number of steps elapsed between the first two spikes sent by the output neuron to the environment during the computation.

Similar to Section 2, some vectors are defined to represent configurations and the application of rules.

Definition 8 (Configuration Vectors). *Let Π be a WSN P system with m neurons, the vector $C_0 = (p_1, p_2, \ldots, p_m)$ is called the* **initial configuration vector** *of Π, where p_i is the amount of the initial potential present in neuron σ_i, $i = 1, 2, \ldots, m$ before a computation starts.*

*In a computation, for any $k \in \mathbb{N}$, the vector $C_k = (p_1^{(k)}, p_2^{(k)}, \ldots, p_m^{(k)})$ is called the k***th configuration vector** *of the system, where $p_i^{(k)}$ is the amount of spikes in neuron σ_i, $i = 1, 2, \ldots, m$ after the kth step in the computation.*

Definition 9 (Spiking Vectors). *Let Π be a WSN P system with m neurons and n rules, and $C_k = (p_1^{(k)}, p_2^{(k)}, \ldots, p_m^{(k)})$ be the kth configuration vector of Π.*

Assume a total order $d : 1, \ldots, n$ is given for all the n rules, so the rules can be referred as $r_1, \ldots, r_n$. A **spiking vector** $s^{(k)}$ *is defined as follows:*

$$s^{(k)} = (r_1^{(k)}, r_2^{(k)}, \ldots, r_n^{(k)}),$$

where:

$$r_i^{(k)} = \begin{cases} 1, \text{ if } r_i \text{ in neuron } \sigma_j \text{ is enabled (the potential } p_j^{(k)} \text{ in neuron } \sigma_j \text{ is equal} \\ \quad \text{to its spiking threshold } t_j) \text{ and the rule } r_i \text{ is chosen and applied;} \\ 0, \text{ otherwise.} \end{cases}$$

In particular, $s^{(0)} = (r_1^{(0)}, r_2^{(0)}, \ldots, r_n^{(0)})$ *is called the* **initial spiking vector**.

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 10 (Forgetting vector). *Let Π be a WSN P system with m neurons, and $C_k = (p_1^{(k)}, p_2^{(k)}, \ldots, p_m^{(k)})$ be the kth configuration vector of Π. A* **forgetting vector** $forg^{(k)}$ *is defined as follows:*

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

where:

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

In particular, $forg^{(0)} = (f_1^{(0)}, f_2^{(0)}, \ldots, f_m^{(0)})$ *is called the* **initial forgetting vector**.

For a WSN P system, a *spiking transition matrix* is defined in order to store the information of the amount of potential consumed (or received) by each neuron when each rule is applied.

Definition 11 (Spiking Transition Matrix). *Let Π be a WSN P system with m neurons and n rules, $d : 1, \ldots, n$ a total order given for all the n rules. The* **spiking transition matrix** *of the system Π, M_Π, is defined as follows:*

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

where:

$$a_{ij} = \begin{cases} -c, \text{ if rule } r_i \text{ is in neuron } \sigma_j \text{ and it is applied consuming potential } c; \\ w, \text{ if rule } r_i \text{ is in neuron } \sigma_s \ (s \neq j \text{ and } (s, j, w) \in syn) \\ \quad \text{and it is applied;} \\ 0, \text{ if rule } r_i \text{ is in neuron } \sigma_s \ (s \neq j \text{ and } (s, j, w) \notin syn, \text{ for } w \in \mathbb{R}_c). \end{cases}$$

The following example illustrates how to get the matrix representation of a WSN P system.

Example 2. Let us consider a WSN P system $\Pi_1 = (\sigma_1, \sigma_1, \sigma_3, syn, out)$ that generates the set $\mathbb{N}$ of natural numbers excluding 1 and 2, where $\sigma_1 = (3, R_1)$, with $R_1 = \{1.5/1.5 \to 1, 1.5/1 \to 1\}$; $\sigma_2 = (2, R_2)$, with $R_2 = \{1/1 \to 1\}$; $\sigma_3 = (1.5, R_3)$, with $R_3 = \{1.5/1 \to 1\}$; $syn = \{(1,2,1),(1,3,-0.5),(2,1,1),(2,3,1.5),(3,1,-1.5),(3,2,-1)\}$; $out = 3$. Π_1 is also represented graphically in Figure 2. Note that in Figure 2, when the weight on a synapse is one, it is omitted.

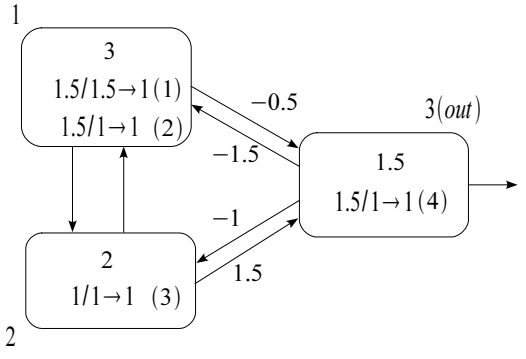

Fig. 2. A WSN P system Π_1 that generates the set $\mathbb{N} - \{1, 2\}$

As shown in Figure 2, a total order for the four rules is set. With this order, the rules can be referred as r_1, r_2, r_3, r_4. According to Definition 11, the spiking transition matrix M_{Π_1} for the WSN P system Π_1 is

$$M_{\Pi_1} = \begin{pmatrix} -1.5 & 1 & -0.5 \\ -1 & 1 & -0.5 \\ 1.5 & -1 & 1.5 \\ -1.5 & -1 & -1.5 \end{pmatrix} \tag{8}$$

The initial configuration is $C_0 = (3, 2, 1.5)$. The initial spiking vector is $(0, 0, 0, 1)$ by Definition 9 and the order of rules.

At step 1, only the output neuron σ_3 spikes, while the other two neurons σ_1, σ_2 maintain their potentials, because their potentials are greater than their corresponding firing thresholds. Neurons σ_1 and σ_2 receive potentials -1.5 and -1, respectively. After this step, the configuration vector becomes $C_1 = (1.5, 1, 0.5)$. At step 2, neurons σ_1 and σ_2 have potentials 1.5 and 1, respectively, which equal to their corresponding firing thresholds, hence both neurons σ_1 and σ_2 spike. Neuron σ_1 has two rules $r_1 : 1.5/1.5 \to 1$ and $r_2 : 1.5/1 \to 1$, and one of them is non-deterministically chosen. If rule r_1 is chosen to be applied, it consumes potential 1.5 and, at the same time, it receives 1.5 unit of potential from neuron σ_2. Hence, in this step, the net gain of potential in neuron σ_1 is 0 and at the next step neuron σ_1 still has potential 1.5. The net gain of potential in neuron

σ_2 is also 0 (one unit of potential is consumed by r_3 and one unit of potential is received from neuron σ_1), neuron σ_3 forgets its potential 0.5 (because it is less than the threshold 1.5), but gets another potential 0.5 from the other two neurons (receives potential -0.5 from σ_1 and potential 1 from σ_2). At the next step, the numbers of spikes in neurons σ_1, σ_2, and σ_3 are still 1.5, 1 and 0.5, respectively.

In order to denote the change of numbers of spikes in each neuron, a *transition net gain vector* is defined.

Definition 12 (Transition Net Gain Vector). *Let Π be a WSN P system with m neurons and n rules, and $C_k = (p_1^{(k)}, p_2^{(k)}, \ldots, p_m^{(k)})$ be the kth configuration vector of Π. The* **transition net gain vector** *at step k is defined as $NG^{(k)} = C_{k+1} - C_k$.*

In order to compute the transition net gain vector, the *Hadamard product* is used.

Definition 13 (Hadamard product). *Let A and B be $m \times n$ matrices. The Hadamard Product of A and B is defined by $[A \odot B]_{ij} = A_{ij}B_{ij}$ for all $1 \leq i \leq m$, $1 \leq j \leq n$.*

To avoid confusion, juxtaposition of matrices will imply the "usual" matrix multiplication, and "$\odot$" is used for the Hadamard product.

Lemma 2. *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 transition net gain vector at step k is*

$$NG^{(k)} = s^{(k)} \cdot M_\Pi - forg^{(k)} \odot C_k. \tag{9}$$

Proof. Assume that $M_\Pi = [a_{ij}]_{m \times n}$, $C_k = (p_1^{(k)}, p_2^{(k)}, \ldots, p_m^{(k)})$, $s^{(k)} = (r_1^{(k)}, r_2^{(k)}, \ldots, r_n^{(k)})$, $forg^{(k)} = (f_1^{(k)}, f_2^{(k)}, \ldots, f_m^{(k)})$, and $NG^{(k)} = (g_1, g_2, \ldots, g_m)$. Note that $r_i^{(k)}$ is a $\{0, 1\}$-value that identifies whether the rule r_i would be applied, $f_i^{(k)}$ is a $\{0, 1\}$-value that identifies whether the neuron σ_j would forget its potential. Thus, $\Sigma_{i=1}^n r_i^{(k)} a_{ij} - f_j^{(k)} p_j^{(k)}$ represents the total amount of potential obtained, consumed and forgotten by neuron σ_j at the kth step. Therefore, the net gain of neuron σ_j is $g_j = \Sigma_{i=1}^n r_i^{(k)} a_{ij} - f_j^{(k)} p_j^{(k)}$, for all $j = 1, 2, \ldots, m$. That is, $NG^{(k)} = s^{(k)} \cdot M_\Pi - forg^{(k)} \odot C_k$.

Theorem 2. *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*

$$C_{k+1} = C_k \ + \ s^{(k)} \cdot M_\Pi - forg^{(k)} \odot C_k. \tag{10}$$

Proof. This result follows directly from the preceding lemma.

In Example 2 shown in Figure 2, $C_0 = (3, 2, 1.5)$, at step 1 only the rule r_4 is applicable. As the potentials in all the neurons are higher than their thresholds, no neuron forgets its potential. Therefore, $s^{(0)} = (0, 0, 0, 1)$, $forg^{(0)} = (0, 0, 0)$, the next configuration can be obtained by

$$C_1 = (3, 2, 1.5) + (0, 0, 0, 1) \begin{pmatrix} -1.5 & 1 & -0.5 \\ -1 & 1 & -0.5 \\ 1.5 & -1 & 1.5 \\ -1.5 & -1 & -1.5 \end{pmatrix} - (0, 0, 0) \odot (3, 2, 1.5) \qquad (11)$$

That is, $C_1 = (2, 1, 0.5)$.

In the next step, rules r_1, r_3 can be applied, so the spiking vector is $(1, 0, 1, 0)$. Because the potential in neuron σ_3 is less than its threshold, neuron σ_3 forgets its potential, and the forgetting vector is $forg^{(0)} = (0, 0, 1)$. Hence, the next configuration is

$$C_2 = (2, 1, 0.5) + (1, 0, 1, 0) \begin{pmatrix} -1.5 & 1 & -0.5 \\ -1 & 1 & -0.5 \\ 1.5 & -1 & 1.5 \\ -1.5 & -1 & -1.5 \end{pmatrix} - (0, 0, 1) \odot (2, 1, 0.5) \qquad (12)$$

That is, $C_2 = (2, 1, 0.5)$.

This example clearly shows how matrix representation and operation can describe the computation of the system. Such matrix representation is useful for the simulation of the system in computer.

5 Conclusions and Remarks

In this work, an algebraic representation for SN P systems is introduced. For every SN P system without delay, configuration vectors are defined to represent the number of spikes in each neuron; spiking vectors are used to denote which rules will be applied; a spiking transition matrix is used to describe the skeleton of system. Such algebraic representation is also extended for another variant of SN P systems – WSN P systems.

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 [1] and SN P systems with exhaustive use of rules [4]. 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.

The systems considered in this paper 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.

Acknowledgements

The work of X. Zeng and L. Pan was supported by National Natural Science Foundation of China (61033003 and 30870826) and the Fundamental Research Funds for the Central Universities (2010ZD001). The work of H. Adorna is supported by Engineering Research and Development for Technology of the DOST, Philippines. The work of M.A. Martínez-del-Amor and M.J. Pérez-Jiménez is supported by the project TIN2009-13192 of the Ministerio de Ciencia e Innovación of Spain, cofinanced by FEDER funds, and the "Proyecto de Excelencia con Investigador de Reconocida Valía" of the Junta de Andalucía under grant P08-TIC04200.

References

1. Cavaliere, M., Egecioglu, O., Ibarra, O.H., Woodworth, S., Ionescu, M., Păun, G.: Asynchronous Spiking Neural P Systems. Theoretical Computer Science 410, 2352–2364 (2009)
2. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Searching Previous Configurations in Membrane Computing. In: Păun, G., Pérez-Jiménez, M.J., Riscos-Núñez, A., Rozenberg, G., Salomaa, A. (eds.) WMC 2009. LNCS, vol. 5957, pp. 301–315. Springer, Heidelberg (2010)
3. Ionescu, M., Păun, G., Yokomori, T.: Spiking Neural P Systems. Fundamenta Informaticae 71(2–3), 279–308 (2006)
4. Ionescu, M., Păun, G., Yokomori, T.: Spiking Neural P Systems with Exhaustive Use of Rules. International Journal of Unconventional Computing 3, 135–154 (2007)
5. Nelson, J.K., McCormac, J.C.: Structural Analysis: Using Classical and Matrix Methods, 3rd edn. Wiley, Chichester (2003)
6. Păun, G.: Computing with Membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
7. Păun, G.: Membrane Computing – An Introduction. Springer, Berlin (2002)
8. Păun, G., Rozenberg, G., Salomaa, A. (eds.): Handbook of Membrane Computing. Oxford University Press, Oxford (2010)
9. Wang, J., Hoogeboom, H.J., Pan, L., Păun, G., Pérez-Jiménez, M.J.: Spiking Neural P Systems with Weights. Neural Computation 22(10), 2615–2646 (2010)
10. The P System Web Page, `http://ppage.psystems.eu`