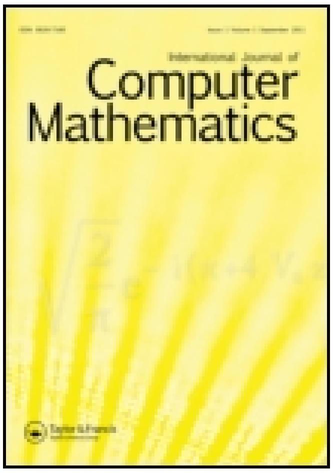

International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcom20

# Homogeneous spiking neural P systems working in sequential mode induced by maximum spike number

Keqin Jiang [a b] , Tao Song [a] , Wei Chen [c] & Linqiang Pan [a]

[a] Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering , Huazhong University of Science and Technology , Wuhan , Hubei , 430074 , China

[b] School of Computer and Information , Anqing Teachers College , Anqing , Anhui , 246133 , China

[c] School of Automation , Wuhan University of Technology , Wuhan , Hubei , 430070 , China

Accepted author version posted online: 10 Oct 2012.Published online: 19 Nov 2012.

To cite this article: Keqin Jiang , Tao Song , Wei Chen & Linqiang Pan (2013) Homogeneous spiking neural P systems working in sequential mode induced by maximum spike number, International Journal of Computer Mathematics, 90:4, 831-844, DOI: 10.1080/00207160.2012.737462

To link to this article: http://dx.doi.org/10.1080/00207160.2012.737462

*International Journal of Computer Mathematics*, 2013
Vol. 90, No. 4, 831–844, http://dx.doi.org/10.1080/00207160.2012.737462

# Homogeneous spiking neural P systems working in sequential mode induced by maximum spike number

Keqin Jiang[a,b], Tao Song[a], Wei Chen[c] and Linqiang Pan[a]*

[a]*Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China;* [b]*School of Computer and Information, Anqing Teachers College, Anqing, Anhui 246133, China;* [c]*School of Automation, Wuhan University of Technology, Wuhan, Hubei 430070, China*

(*Received 21 May 2012; revised version received 21 August 2012; second revision received 10 September 2012; accepted 3 October 2012*)

Spiking neural P systems (SN P systems, for short) are a class of distributed parallel computing devices inspired by the way neurons communicate by means of spikes, where neurons work in parallel in the sense that each neuron that can fire should fire at each computation step, and neurons can be different in the sense that they can have different sets of spiking rules. In this work, we consider SN P systems with the restrictions: (1) all neurons are homogeneous in the sense that each neuron has the same set of rules; (2) at each step the neuron with the maximum number of spikes among the neurons that are active (can spike) will fire. These restrictions correspond to the fact that the system consists of only one kind of neurons and a global view of the whole network makes the system sequential. The computation power of homogeneous SN P systems working in the sequential mode induced by the maximum spike number is investigated. Specifically, it is proved that such systems are universal as both generating and accepting devices.

**Keywords:** membrane computing; spiking neural P system; homogeneity; sequential mode; turing completeness

*2010 AMS Subject Classification*: 68Q05

## 1. Introduction

*Membrane computing* is one of the recent branches of natural computing; it was initiated in [11] and has developed rapidly (already in 2003, ISI considered membrane computing as 'fast emerging research area in computer science', see http://esi-topics.com). The aim is to abstract computing ideas (data structures, operations with data, ways to control operations, computing models, etc.) from the structure and the functioning of a single cell and from complexes of cells, such as tissues and organs including the brain. The models obtained are distributed and parallel computing devices, usually called *P systems*. The present work deals with a class of neural-like P systems, called *spiking neural P systems* (SN P systems, for short), introduced in [5]. For the definition of SN P systems, please refer to Section 2.

Many computational properties of SN P systems have been investigated. SN P systems were proved to be computationally complete (equivalent with Turing machines or other equivalent

*Corresponding author. Email: lqpan@mail.hust.edu.cn computing devices; we also say that SN P systems are universal) as number computing devices [5], language generators [1,16], and function computing devices [10,15]. SN P systems were also used to (theoretically) solve computationally hard problems in a feasible time [6,8]. All systems in [1,5,6,8,10,15,16] consist of different neurons in the sense that neurons can have different sets of rules, and function in the following way: all neurons work in parallel in the sense that at each computation step each neuron that can apply a rule should do it.

In an SN P system, neurons can be different from each other in the sense of having different sets of rules. In [14], SN P systems with the restriction that each neuron has the same set of rules are considered, which are called *homogeneous spiking neural P systems* (for short, HSN P systems). The main motivation behind this restriction is the following biological observation. In the (human) brain, there are about 100 billion neuron cells. Most of the neurons are very similar to each other, they are 'designed' by nature to have the same 'set of rules', 'working' in a uniform way to transform input into output.

Although biological processes in living organisms happen in parallel, they are not synchronized by a universal clock as assumed in SN P systems. Several authors have noticed that the maximal parallelism way of rule application is rather non-realistic and considered various 'strategies' of rule application, e.g. [2,3]. In [4], SN P systems were considered to function in a sequential manner induced by maximum spike number, i.e. if at any computation step there is more than one active neuron, then only the neuron(s) containing the maximum number of spikes (among the currently active neurons) will be able to fire. If there are ties for the maximum number of spikes stored in active neurons, then two distinct strategies are considered: max-pseudo-sequentiality (all the neurons with the maximum number of spikes will fire) and max-sequentiality (only one of the neurons with the maximum number of spikes will fire). It was shown that SN P systems are universal working in max-sequentiality or max-pseudo-sequentiality strategy [4].

In this work, we investigate the computation power of HSN P systems working in max-sequentiality and max-pseudo-sequentiality strategies. It is proved that HSN P systems with weighted synapses working in max-sequentiality or max-pseudo-sequentiality strategy are universal as both generating and accepting devices. Furthermore, we prove that HSN P systems with the usual synapses (the weight of all synapses is one) working in max-pseudo-sequentiality strategy can also achieve the Turing completeness. It remains open whether HSN P systems with the usual synapses working in max-sequentiality strategy are universal.

## 2. Spiking neural P systems

In this section, we introduce basic SN P systems without delay (this feature is not used in this work) and the notions: weighted synapses, homogeneity, max-sequentiality and max-pseudo-sequentiality. In the definition of SN P systems, the notion of regular expression is used, please refer to [13] for the detail.

A *spiking neural P system* of degree $m \geq 1$ is a construct of the following form:

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, \text{syn}, \text{in}, \text{out}),$$

where

- $O = \{a\}$ is the singleton alphabet ($a$ is called *spike*);
- $\sigma_1, \sigma_2, \ldots, \sigma_m$ are *neurons* of the form $\sigma_i = (n_i, R_i)$ with $1 \leq i \leq m$, where
  (a) $n_i \geq 0$ is the *initial number of spikes* contained in $\sigma_i$;
  (b) $R_i$ is a finite set of *rules* of the following two forms:

(1) $E/a^c \to a$, where $E$ is a regular expression over $O$ and $c \geq 1$;
(2) $a^s \to \lambda$, for some $s \geq 1$, with the restriction that $a^s \notin L(E)$ for any rule $E/a^c \to a$ from $R_i$;

- $\text{syn} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin \text{syn}$ for $1 \leq i \leq m$ (*synapses* between neurons);
- $\text{in}, \text{out} \in \{1, 2, \ldots, m\}$ indicate the *input* and *output* neurons, respectively.

The rules of type (1) are *firing* (we also say *spiking*) rules; and the rules of type (2) are *forgetting* rules. They are applied in the way specified in [5].

The *initial configuration* of the system is described by the numbers $n_1, n_2, \ldots, n_m$ of spikes present in each neuron. It is denoted by $C_0 = \langle n_1, n_2, \ldots, n_m \rangle$. Using the rules as described above, one can define *transitions* among configurations. A series of transitions starting from the initial configuration is called a *computation*. A computation halts if it reaches a configuration where no rule can be used. With any computation, halting or not, we associate a spike train, the binary sequence with occurrences of 1 indicating time instances when the output neuron sends a spike out of the system (we also say that the system itself spikes at that time). The result of a computation can be defined in several ways. In this work, with any spike train containing at least two spikes, the first two being emitted at step $t_1, t_2$, one associates a computation result in the form of the number $t_2 - t_1 - 1$; we say that this number is generated/computed by the SN P system $\Pi$. The set of all numbers computed in this way by $\Pi$ is denoted by $N_2(\Pi)$ (the subscript indicates that the computation result is encoded by the time distance between the first two spikes of any computation). Note that the computation result defined in this work is different with the classical one as given in [5] (specifically, in [5], the computation result is defined as $t_2 - t_1$ instead of $t_2 - t_1 - 1$).

SN P systems can also work in the accepting mode: a computation starts from an initial configuration, and a number $n = t_2 - t_1 - 1$ is introduced through the input neuron by two spikes at steps $t_1$ and $t_2$; number $n$ is said to be accepted by the system if the computation eventually halts. We denote by $N_{\text{acc}}(\Pi)$ the set of numbers accepted by $\Pi$.

SN P systems with *weighted synapses* are introduced in [9], where the weighted synapses are of the form $\text{syn} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \times \mathbb{N}$. SN P systems with weighted synapses have the same components and the similar way of working as the above basic SN P systems except for the synapses. For a synapse $(i, j, r) \in \text{syn}$, if a spike is emitted by neuron $\sigma_i$, then $r$ spikes will be received by neuron $\sigma_j$, where $r$ is called the weight of the synapse. Clearly, basic SN P systems are a special case of SN P systems with weighted synapses, where the weight of each synapse is one.

An SN P system such that each neuron has the same set of rules (that is, $R_1 = R_2 = \cdots = R_m$ for neurons $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$) is said to be *homogeneous*. A homogeneous SN P system is abbreviated as HSN P system for short.

An SN P system works in the *max-sequentiality* manner if the system is choosing non-deterministically as the spiking neuron at each step only one of the neurons that can fire (thus the system works in a sequential way) and, furthermore, the spiking neuron chosen at each step has the maximum number of spikes stored among all the other active neurons in that step. An SN P system can work in *max-pseudo-sequentiality* manner if at each step the system fires all the neurons that store the maximum number of spikes among all the active neurons at that step. Max-sequentiality is forcing the system to work in a sequential manner as at each step at most one neuron can fire, whereas the max-pseudo-sequentiality allows two or more neurons to fire at the same time if all those neurons hold exactly the same number of spikes and that number is the highest value of spikes that is stored among all the active neurons at that moment.

We denote by $N_{\alpha}^{\beta}$ HSN P(weight$_k$) the family of all sets of numbers generated/accepted by HSN P systems with the weight of all the synapses at most $k$, where $\alpha \in \{2, \text{acc}\}$, and $\beta$ is either ms (ms stands for the max-sequentiality) or mps (mps means the max-pseudo-sequentiality). If the weight of the synapses is always one (as the synapses in basic SN P systems), then we remove the indication *weight* from the notation.

## 3. HSN P systems working in the max-sequentiality manner

In this section, we investigate the universality of HSN P systems working in the max-sequentiality manner. It is proved that HSN P systems with weighted synapses working in the max-sequentiality strategy manner are universal as both number generating and accepting devices. As we will see, the weight on synapses significantly simplifies the construction and structure of an HSN P system simulating a register machine. It remains open whether HSN P systems with usual synapses working in the max-sequentiality strategy manner are universal. We conjecture that the answer is positive.

The following proofs are based on the simulation of register machines. A register machine is a construct $M = (m, H, l_0, l_h, I)$, where $m$ is the number of registers, $H$ is the set of instruction labels, $l_0$ is the start label (labelling an ADD instruction), $l_h$ is the halt label (assigned to instruction HALT) and $I$ is the set of instructions; each label from $H$ labels only one instruction from $I$, thus precisely identifying it. The instructions are of the following forms:

- $l_i : (\texttt{ADD}(r), l_j, l_k)$ (add 1 to the register $r$ and then go to one of the instructions with label $l_j$ and $l_k$, non-deterministically chosen),
- $l_i : (\texttt{SUB}(r), l_j, l_k)$ (if register $r$ is non-empty, then subtract 1 from it and go to the instruction with label $l_j$, otherwise go to the instruction with label $l_k$),
- $l_h : \texttt{HALT}$ (the halt instruction).

A register machine can work in the generating or accepting mode, and it is known that register machines compute all sets of numbers which are Turing computable, hence they characterize NRE (see, e.g. [7]).

When the power of two number generating/accpeting devices $D_1$ and $D_2$ is compared, we use the convention that number zero is ignored; that is, $N(D_1) = N(D_2)$ if and only if $N(D_1) - \{0\} = N(D_2) - \{0\}$ (this corresponds to the usual practice of ignoring the empty string when comparing the power of two devices in language and automata theory).

### 3.1 *HSN P systems with weighted synapses working in the generating mode*

THEOREM 3.1 $N_2^{\text{ms}}$ HSN P(weight$_5$) $=$ NRE.

*Proof* We have only to prove the inclusion NRE $\subseteq N_2^{\text{ms}}$ HSN P(weight$_5$); the converse inclusion is straightforward but cumbersome (for similar technical details, please refer to Section 8.1 in [12]).

Let us consider a register machine $M = (m, H, l_0, l_h, I)$. Without any loss of generality, we may assume that in the halting configuration, all registers of $M$ different from register 1 are empty, and that the register 1 is never decremented during the computation, we only add to its content. In what follows, a specific HSN P system with weighted synapses $\Pi$ will be constructed to simulate the register machine $M$.

In the system $\Pi$, each neuron has the same set of rules $R = \{a \to a, a^2 \to \lambda, a^3/a^2 \to a, a^4/a^2 \to a, a^6 \to \lambda, a^9/a \to a, a^6(a^5)^+/a^{11} \to a, a^8(a^5)^+/a^5 \to a\}$, which is shown in Figure 1.

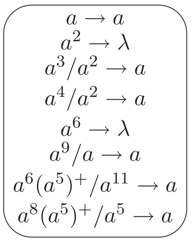

Figure 1. Neurons in system Π.

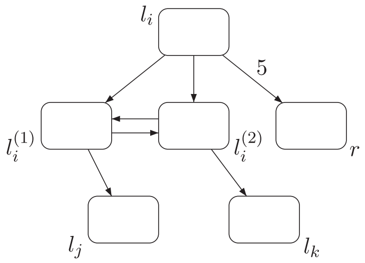

Figure 2. Module ADD for simulating instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

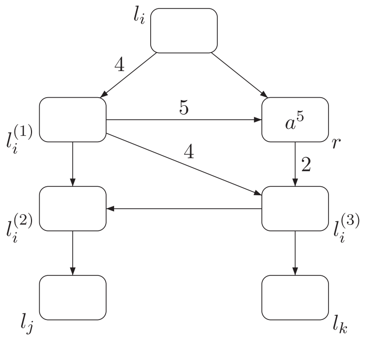

Figure 3. Module SUB for simulating instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$.

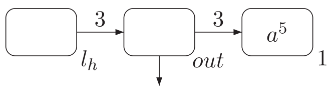

Figure 4. Module FIN (outputting the result of computation).

The system Π consists of three types of modules – ADD module, SUB module and FIN module shown in Figures 2–4, respectively. Of course, each module is composed of neurons as in Figure 1. ADD module and SUB module are used to simulate the ADD and SUB instructions of $M$, respectively; FIN module is used to output a computation result.

In general, for each register $r$ of $M$, a neuron $\sigma_r$ is associated whose content corresponds to the content of the register. Specifically, if register $r$ holds the number $n \geq 0$, then neuron $\sigma_r$ will contain $5n + 5$ spikes. For each label $l_i$ of an instruction in $M$, a neuron $\sigma_{l_i}$ is associated. Initially, all neurons are empty, with the exception that each neuron $\sigma_r$ associated with a register $r$ contains five spikes and neuron $\sigma_{l_0}$ associated with the initial instruction $l_0$ contains one spike (which corresponds to the fact that $M$ starts a computation by applying the instruction with label $l_0$). During the computation, a neuron $\sigma_{l_i}$ which receives one spike will become active and start to simulate an instruction $l_i : (\texttt{OP}(r), l_j, l_k)$ of $M$: starting with neuron $\sigma_{l_i}$ activated, operating the register as requested by $\texttt{OP}$, then introducing one spike in one of the neurons $\sigma_{l_j}$, $\sigma_{l_k}$, which becomes in this way active. When neuron $\sigma_{l_h}$ is activated, associated with the halting label of $M$, the computation in $M$ is completely simulated in $\Pi$; we will have the output neuron spiking twice, at an interval of time which corresponds to the number stored in register 1 of $M$.

*Module ADD*: Simulating an ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

The initial instruction in register machine $M$, labelled with $l_0$, is an ADD instruction. Let us assume that we are at a step $t$ when we have to simulate an ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$, with one spike present in neuron $\sigma_{l_i}$ (like $\sigma_{l_0}$ in the initial configuration) and no spike in any other neurons, except in those neurons associated with registers. Having one spike inside, neuron $\sigma_{l_i}$ is enabled by rule $a \to a$, and this is the only enabled neuron in the system, so $\sigma_{l_i}$ fires sending one spike to each of neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}$, as well as five spikes to neuron $\sigma_r$ (the spike passing along the weighted synapse $(l_i, r, 5)$ is amplified by five due to the weight 5). By receiving the five spikes, the number of spikes in neuron $\sigma_r$ is increased by five, which simulates the increase in the number stored in register $r$ by 1.

At step $t+1$, with one spike inside, neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ are enabled by rule $a \to a$, and no other neurons are enabled in the system. So, neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ have the maximum number of spikes among the active neurons, and one of neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ must fire at this step, which is non-deterministically chosen.

(1) If neuron $\sigma_{l_i^{(1)}}$ fires, then it sends one spike to each of neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_j}$. In this way, neuron $\sigma_{l_i^{(2)}}$ has two spikes and neuron $\sigma_{l_j}$ has one spike, and both of them are enabled by the rules $a^2 \to \lambda$ and $a \to a$, respectively. At this moment, the maximum number of spikes among the active neurons is two. So, at step $t+2$, neuron $\sigma_{l_i^{(2)}}$ applies the rule $a^2 \to \lambda$ removing its two spikes. At step $t+3$, neuron $\sigma_{l_j}$ is the only active one in the system, and it fires, which means that starts to simulate the instruction with label $l_j$.

(2) If neuron $\sigma_{l_i^{(2)}}$ fires, then similar to the case (1), we can check that the number of spikes in neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ is reset to zero, and neuron $\sigma_{l_k}$ is enabled and fires, which means that the system starts to simulate the instruction with label $l_k$.

Therefore, from firing neuron $\sigma_{l_i}$, five spikes are added to neuron $\sigma_r$, and one of neurons $\sigma_{l_j}$ and $\sigma_{l_k}$ non-deterministically fires, which correctly simulates the ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

*Module SUB*: Simulating a SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$.

Instruction $l_i$ is simulated in $\Pi$ in the following way. Initially, there is one spike in neuron $\sigma_{l_i}$, and no spike in other neurons, except for neurons associated with registers. Let $t$ be the moment when neuron $\sigma_{l_i}$ fires. At step $t$, neuron $\sigma_{l_i^{(1)}}$ receives four spikes and neuron $\sigma_r$ receives one spike; so neuron $\sigma_{l_i^{(1)}}$ contains 4 spikes and neuron $\sigma_r$ contains $5n+6$ $(n \geq 0)$ spikes (corresponding to that the number stored in register $r$ is $n$ at step $t$). Both the neurons $\sigma_{l_i^{(1)}}$ and $\sigma_r$ are enabled, and neuron $\sigma_r$ has the maximum number of spikes $5n+6$ between the enabled neurons, so at step $t+1$, neuron $\sigma_r$ fires. For the firing of neuron $\sigma_r$, there are the following two cases.

(1) The number of spikes in neuron $\sigma_r$ is 6 at step $t+1$ (corresponding to that the number stored in register $r$ is zero at step $t$). Then at step $t+1$ neuron $\sigma_r$ removes its 6 spikes by the rule $a^6 \to \lambda$; and at step $t+2$, neuron $\sigma_{l_i^{(1)}}$ is the unique enabled neuron, so it fires by the rule $a^4/a^2 \to a$ sending 1,4 and 5 spikes to neurons $\sigma_{l_i^{(2)}}, \sigma_{l_i^{(3)}}$ and $\sigma_r$, respectively. After receiving the five spikes, neuron $\sigma_r$ contains 5 spikes as in the initial configuration (corresponding to

that the number stored in register $r$ is zero). At step $t+3$, neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ are enabled and contain 2, 1, 4 spikes, respectively; so, neuron $\sigma_{l_i^{(3)}}$ fires sending one spike to each of neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_k}$. At step $t+4$, neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}$, $\sigma_{l_i^{(3)}}$ and $\sigma_{l_k}$ are enabled; each of neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ contains two spikes, neuron $\sigma_{l_k}$ contains one spike. At steps $t+4$, $t+5$, $t+6$, neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ remove its two spikes by the rule $a^2 \rightarrow \lambda$ in a non-deterministic order. At step $t+7$, neuron $\sigma_{l_k}$ is the unique enabled one and fires, which means that the system starts to simulate the instruction with label $l_k$.

(2) The number of spikes in neuron $\sigma_r$ is $5n+6$ ($n > 0$) at step $t+1$. Then at step $t+1$, neuron $\sigma_r$ fires by the rule $a^6(a^5)^+/a^{11} \rightarrow a$ consuming 11 spikes and neuron $\sigma_{l_i^{(3)}}$ receives two spikes. At step $t+2$, neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(3)}}$ are enabled, and they have 4 and 2 spikes, respectively; so neuron $\sigma_{l_i^{(1)}}$ fires sending 1, 4 and 5 spikes to neurons $\sigma_{l_i^{(2)}}, \sigma_{l_i^{(3)}}$ and $\sigma_r$, respectively. In this way, neuron $\sigma_r$ has $5(n-1)+6$ spikes, which simulates that the number stored in register $r$ is decreased by one. At step $t+3$, neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ are enabled, and they have 2, 1, and 6 spikes, respectively; neuron $\sigma_{l_i^{(3)}}$ removes 6 spikes by the rule $a^6 \rightarrow \lambda$. At step $t+4$, neuron $\sigma_{l_i^{(1)}}$ removes 2 spikes by the rule $a^2 \rightarrow \lambda$. At step $t+5$, neuron $\sigma_{l_i^{(2)}}$ fires sending one spike to neuron $\sigma_{l_j}$. With one spike inside, neuron $\sigma_{l_j}$ is enabled and will fire, which means that the system starts to simulate the instruction with label $l_j$.

The simulation of SUB instruction is correct: system $\Pi$ starts from neuron $\sigma_{l_i}$ and ends in neuron $\sigma_{l_j}$ (if the number stored in register $r$ is great than 0 and decreased by one), or in neuron $\sigma_{l_k}$ (if the number stored in register $r$ is 0).

Note that there is no interference between the neurons used in the ADD and the SUB instructions. However, it is possible to have interference between neurons in two SUB modules. Specifically, if there are several SUB instructions $l_s$ which act on register $r$, then when an instruction $l_i$ : $(\mathtt{SUB}(r), l_j, l_k)$ is simulated, it is possible that neuron $\sigma_r$ sends two spikes to all neurons $\sigma_{l_s^{(3)}}$ from modules associated with SUB instructions $l_s$; however, they are forgotten by the rule $a^2 \rightarrow \lambda$ before neuron $\sigma_{l_j}$ or $\sigma_{l_k}$ fires (because neuron $\sigma_{l_s^{(3)}}$ has two spikes while neuron $\sigma_{l_j}$ or $\sigma_{l_k}$ has one spike) (that is, before the simulation of the next instruction starts, the number of spikes in neuron $\sigma_{l_s^{(3)}}$ is reset to zero). Therefore, no undesired effect appears (i.e. steps which do not correspond to correct simulations of instructions of $M$).

*Module FIN*: Outputting the result of computation.

Assume now that the computation in $M$ halts, which means that the halting instruction is reached. This means that neuron $\sigma_{l_h}$ in $\Pi$ gets one spike and fires by rule $a \rightarrow a$. At that moment, neuron $\sigma_1$ contains $5n+5$ spikes, for the number $n$ stored in register 1 of $M$. When $\sigma_{l_h}$ fires, neuron $\sigma_{out}$ receives 3 spikes because of the synapse weight 3. With three spikes inside, neuron $\sigma_{\text{out}}$ fires by the rule $a^3/a^2 \rightarrow a$ sending one spike into the environment and three spikes to neuron $\sigma_1$; we suppose it is at step $t$. In this way, neurons $\sigma_{\text{out}}$ and $\sigma_1$ have 1 and $5n+8$ spikes, respectively, and both of them are enabled. Because of the max-sequentiality, from step $t+1$ to step $t+n$, at each step neuron $\sigma_1$ fires by the rule $a^8(a^5)^+/a^5 \rightarrow a$; there is no outgoing synapse from neuron $\sigma_1$, the emitting spikes are lost. At step $t+n+1$, neuron $\sigma_1$ has 8 spikes, where no rule is applicable; so neuron $\sigma_{\text{out}}$ fires again by the rule $a \rightarrow a$, the system sends the second spike to the environment. The interval between these two spikes sent out to the environment by the system is $(t+n+1)-t = n+1$, which means that the number computed by $\Pi$ is $n$ (recalling that the computation result is defined as $t_2 - t_1 - 1$, where $t_1$ and $t_2$ are the time instance when the first two spikes are sent to the environment).

From the above description of the modules and their work, it is clear that the register machine $M$ is correctly simulated by the HSN P system $\Pi$ in the max-sequentiality manner. We can check that in the system $\Pi$, the maximum weight of the synapses is five. Therefore $N_2^{\text{ms}}$ HSN P(weight$_5$) $=$ NRE. ■

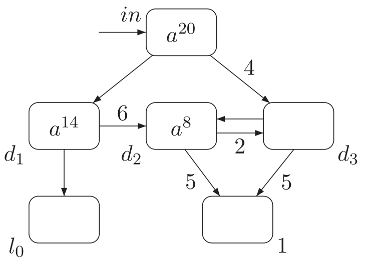

Figure 5. Module INPUT (initiating the computation).

## 3.2 *HSN P systems with weighted synapses working in the accepting mode*

An HSN P system can be also used in the accepting mode: the special neuron $\sigma_{\text{in}}$ is used as an input neuron, which can receive spikes from the environment; we assume that exactly two spikes are entering the system; the $n+1$ steps elapsed between the two spikes encode the number to be analysed; if, after receiving the two spikes, the system halts (not necessarily in the moment of receiving the second spike), then the number $n$ is accepted.

THEOREM 3.2 $N^{\text{ms}}_{\text{acc}}$HSN P(weight$_6$) $=$ NRE.

*Proof* The proof is a direct consequence of the proof of Theorem 3.1. We start from a deterministic register machine $M = (m, H, l_0, l_h, I)$ and construct the HSN P system $\Pi$ as in the proof of Theorem 3.1. We denote by $\Pi'$ the SN P system obtained by modifying the system $\Pi$ – with changes which will be immediately mentioned, as well as a further module, called INPUT, which takes care of initializing the work of $\Pi'$. The system $\Pi'$ is composed of neurons from Figure 1.

The module INPUT is shown in Figure 5. Its functioning is rather clear. All neurons are initially empty, with the exception that neurons $\sigma_{\text{in}}$, $\sigma_{d_1}$ and $\sigma_{d_2}$ contain 20, 14, 8 spikes, respectively. When the first spike enters neuron $\sigma_{\text{in}}$ at step $t$, the rule $a^6(a^5)^+/a^{11} \to a$ is enabled. At step $t+1$, neuron $\sigma_{\text{in}}$ fires consuming 11 spikes (10 spikes remain in $\sigma_{\text{in}}$, which is not enabled); neurons $\sigma_{d_1}$ and $\sigma_{d_3}$ receive one spike and four spikes, respectively. At step $t+2$, neurons $\sigma_{d_1}$, $\sigma_{d_3}$ and $\sigma_{d_2}$ contain 15, 4 and 8 spikes, respectively; so only neuron $\sigma_{d_3}$ is enabled and fires by the rule $a^4/a^2 \to a$; neurons $\sigma_1$ and $\sigma_{d_2}$ receive five spikes and one spike, respectively. At step $t+3$, neurons $\sigma_{d_3}$ and $\sigma_{d_2}$ are enabled, which have 2 and 9 spikes, respectively; neuron $\sigma_{d_2}$ fires; neurons $\sigma_{d_3}$ and $\sigma_1$ receive two and five spikes, respectively. In this way, neurons $\sigma_{d_3}$ and $\sigma_{d_2}$ again have 4 and 8 spikes, respectively, which means that neurons $\sigma_{d_3}$ and $\sigma_{d_2}$ will fire at an alternate step, and the number of spikes in neuron $\sigma_1$ is increased by five at each step. If at step $t+n+1$ ($n \geq 1$ is the number to be accepted), the second spike enters neuron $\sigma_{\text{in}}$ coming from the environment, then at step $t+n+2$ neuron $\sigma_{\text{in}}$ has 11 spikes (more spikes than in neuron $\sigma_{d_2}$ or $\sigma_{d_3}$); it is enabled and fires again by the rule $a^6(a^5)^+/a^{11} \to a$; neuron $\sigma_{d_1}$ receives 1 spike (having 16 spikes inside) and neuron $\sigma_{d_3}$ receives 4 spikes (having 8 or 6 spikes inside). At step $t+n+3$, because of the max-sequentiality, neuron $\sigma_{d_1}$ fires by the rule $a^6(a^5)^+/a^{11} \to a$ consuming 11 spikes (neuron $\sigma_{d_1}$ will be not enabled with 5 spikes remaining inside); neuron $\sigma_{l_0}$ receives one spike and neuron $\sigma_{d_2}$ receives six spikes (having 14 or 15 spikes inside, neuron $\sigma_{d_2}$ is not enabled). At step $t+n+4$, if neuron $\sigma_{d_3}$ has 8 spikes, then it is not enabled and the enabled neuron $\sigma_{l_0}$ fires, which means that the system $\Pi'$ starts to simulate the instructions of $M$; if neuron $\sigma_{d_3}$ has 6 spikes, then it removes its six spikes by the rule $a^6 \to \lambda$, and neuron $\sigma_{l_0}$ fires at step $t+n+5$. Therefore, on the one hand, the number of spike in neuron $\sigma_1$ is $5n+5$ (it consecutively receives 5 spikes at each step from step $t+3$ to $t+n+4$); on the other hand, neuron $\sigma_{l_0}$ triggers the simulation of a computation in $M$, starting with the instruction

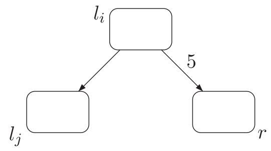

Figure 6. Module ADD in the deterministic case.

labelled with $l_0$. From now on, we start to use modules ADD and SUB associated with register machine $M$.

Because ADD instructions in accepting mode are deterministic (of the form $l_i : (\texttt{ADD}(r), l_j)$), the corresponding module ADD is now much simpler than the one in Figure 2, which is shown in Figure 6.

Module SUB remains unchanged as shown in Figure 3, while module FIN is removed, with neuron $\sigma_{l_h}$ remaining in the system, but without outgoing synapses. When neuron $\sigma_{l_h}$ receives one spike, it means that the computation of register machine $M$ reaches instruction $l_h$ and stops. Having one spike inside, neuron $\sigma_{l_h}$ fires by rule $a \rightarrow a$, but it sends no spike out because it has no outgoing synapses, and in this way the HSN P system $\Pi'$ halts. ■

## 4. HSN P systems working in the max-pseudo-sequential manner

In this section, we investigate the computation power of HSN P systems working in the max-pseudo-sequential manner.

### 4.1 *HSN P systems with weighted synapses*

THEOREM 4.1 $N_2^{\text{mps}}\,\text{HSN P}(\text{weight}_7) = \text{NRE}$.

*Proof* We only have to prove that $\text{NRE} \subseteq N_2^{\text{mps}}\,\text{HSN P}(\text{weight}_7)$. For a register machine $M = (m, H, l_0, l_h, I)$, we construct an HSN P system $\Pi''$ working in max-pseudo-sequentiality to simulate $M$. The system $\Pi''$ consists of neurons shown in Figure 7.

Since the SUB module and FIN module given in the proof of Theorem 3.1 have always a unique enabled neuron with the maximum number of spikes at each step, the working manner of these modules can also be considered as max-pseudo-sequentiality. Hence, we only need to construct ADD module to simulate the ADD instruction. For an HSN P system working in the

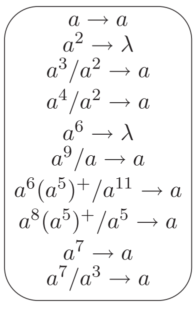

Figure 7. Neurons in HSN P system $\Pi''$.

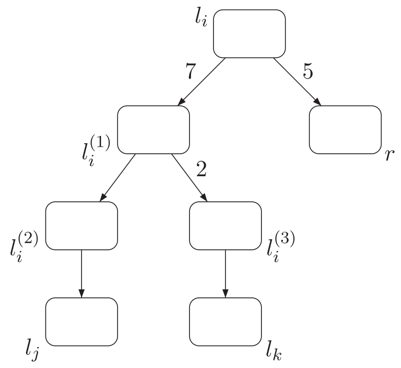

Figure 8. The ADD module of $\Pi''$.

max-pseudo-sequentiality manner, there is no non-determinism at the level of the system: from each configuration to the next one, we know for sure which neuron(s) will fire. Here, we achieve the non-determinism of ADD module at the level of neurons. The ADD module is shown in Figure 8. In what follows, we describe the work of ADD module.

Assume that at step $t$ neuron $\sigma_{l_i}$ has one spike inside; it is enabled and fires by the rule $a \rightarrow a$; neurons $\sigma_{l_i^{(1)}}$ and $\sigma_r$ receive 7 spikes and 5 spikes, respectively. In this way, the number of spikes in neuron $\sigma_r$ increases by five, which simulates that the number stored in register $r$ is added by one. At step $t+1$, neuron $\sigma_{l_i^{(1)}}$ is enabled and fires by non-deterministically choosing the rules $a^7 \rightarrow a$ or $a^7/a^3 \rightarrow a$.

(1) If the rule $a^7 \rightarrow a$ is applied, then neuron $\sigma_{l_i^{(1)}}$ consumes its all 7 spikes; neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ receive one spike and two spikes, respectively. At step $t+2$, neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ are enabled, and neuron $\sigma_{l_i^{(3)}}$ removes its two spikes by the forgetting rule $a^2 \rightarrow \lambda$. At step $t+3$, neuron $\sigma_{l_i^{(2)}}$ fires by the rule $a \rightarrow a$; neuron $\sigma_{l_j}$ receives one spike. At step $t+4$, neuron $\sigma_{l_j}$ is enabled and fires, which means that the system starts to simulate the instruction $l_j$.
(2) If the rule $a^7/a^3 \rightarrow a$ is applied, then neuron $\sigma_{l_i^{(1)}}$ consumes 3 spikes and there are 4 spikes remaining in neuron $\sigma_{l_i^{(1)}}$; neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ receive one spike and two spikes, respectively. At step $t+2$, neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ are enabled, and neuron $\sigma_{l_i^{(1)}}$ fires by the rule $a^4/a^2 \rightarrow a$; neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ receive one spike and two spikes, respectively. At step $t+3$, each of neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ has two spikes, and neuron $\sigma_{l_i^{(3)}}$ has four spikes, all of them are enabled; neuron $\sigma_{l_i^{(3)}}$ fires by the rule $a^4/a^2 \rightarrow a$ sending one spike to neuron $\sigma_{l_k}$. At step $t+4$, each of neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ has two spikes, neuron $\sigma_{l_k}$ has one spike; all of them are enabled; each of the neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ removes its two spikes by the rule $a^2 \rightarrow \lambda$. At step $t+5$, neuron $\sigma_{l_k}$ fires, which means that the system starts to simulate the instruction $l_k$.

Therefore, from firing neuron $\sigma_{l_i}$, one of the neurons $\sigma_{l_j}$, $\sigma_{l_k}$ non-deterministically fires, which correctly simulates the ADD instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

We can check that in the system $\Pi''$, the maximum weight of the synapses is 7. Therefore, $N_2^{\text{mps}}$ HSN P(weight$_7$) $=$ NRE. ■

THEOREM 4.2 $N_{\text{acc}}^{\text{mps}}$ HSN P(weight$_6$) $=$ NRE.

*Proof* In the proof of Theorem 3.2, we can check that the INPUT module shown in Figure 5, ADD module shown in Figure 6 and SUB module shown in Figure 3 also work in the max-pseudo-sequentiality manner because at each step there is a unique enabled neuron with the maximum number of spikes. ■

## 4.2 *HSN P systems with usual synapses*

Let us now consider HSN P systems with usual synapses (that is, the weight on the synapses is one) working in the max-pseudo-sequentiality manner. It is proved that HSN P systems with usual synapses working in the max-pseudo-sequentiality manner are universal as both generating and accepting devices, which are stated as Theorems 4.3 and 4.4. In the proof of Theorems 4.3 and 4.4, we only provide the modules that the systems consist of, without any explanation about their functioning.

Theorem 4.3 $N_2^{\text{mps}}$ HSN P = NRE.

*Proof* The ADD, SUB, FIN modules are shown in Figures 9–11, respectively, where all modules consist of neurons shown in Figure 7. ■

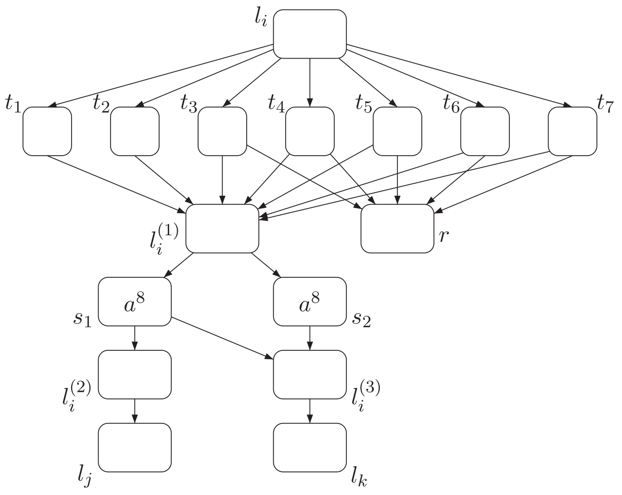

Figure 9. The ADD module with usual synapses.

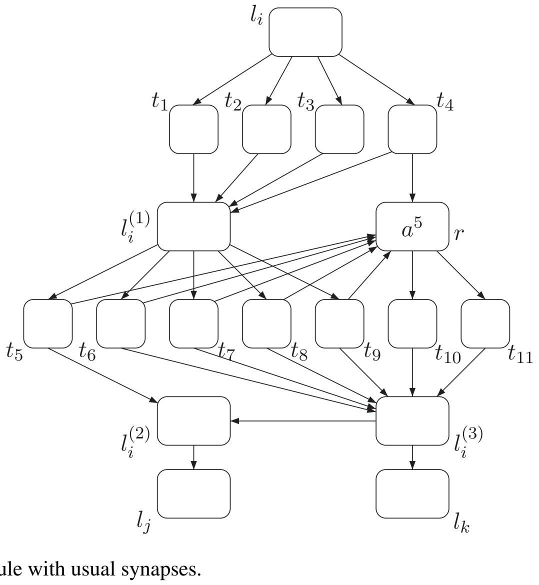

Figure 10. The SUB module with usual synapses.

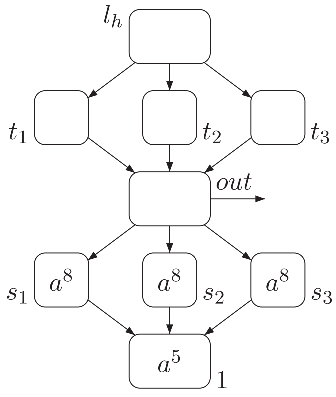

Figure 11. The FIN module with usual synapses.

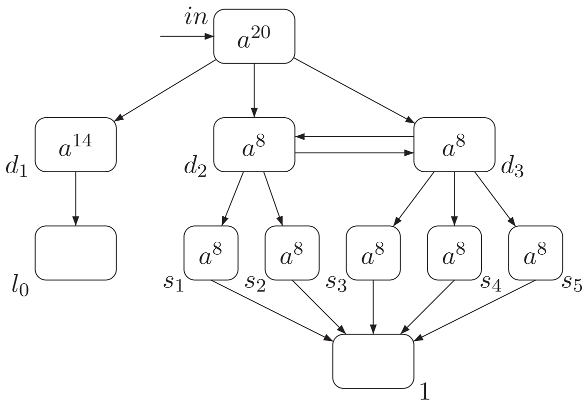

Figure 12. The INPUT module with usual synapses.

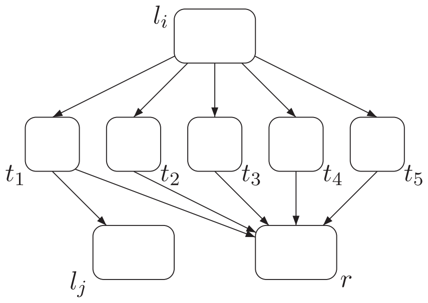

Figure 13. The ADD module with usual synapses (in the deterministic case).

THEOREM 4.4 $N_{\text{acc}}^{\text{mps}}$ HSN P = NRE.

*Proof* The INPUT, ADD, SUB modules are shown in Figures 10, 12, 13, respectively, where all modules consist of neurons shown in Figure 7. ■

## 5. Conclusions and remarks

In this work, we investigate the computation power of a restricted variant of SN P systems – homogeneous SN P systems working in a sequential manner induced by the maximum number of spikes. It is proved that such systems with weighted synapses are universal as both generative and acceptive devices. In the max-pseudo-sequentiality case, such systems with usual synapses (weight on all synapses is one) are also universal. These results can have a nice interpretation: the structure of a neural system is crucial for the functioning of the system. Although neurons are homogeneous, by cooperating with each other, a network of neurons can be powerful – 'complete (Turing) creativity' even in the sequential manner.

In this work, although the computation result is encoded by the time distance between the first two spikes sent to the environment by the system, it is defined in the form of the number $t_2 - t_1 - 1$ instead of $t_2 - t_1$ (the classical definition of a computation result as in [5]), where $t_1, t_2$ are the time instances when the first two spikes sent to the environment. The reason of defining a computation result in such way is to avoid discussing the generation of number one case by case (actually, if numbers zero and one are ignored when the power of two devices is compared, then the proofs given in this work are easy to be modified to follow the classical definition of computation result).

In the proofs of Theorems 4.3 and 4.4 (HSN P systems with usual synapses working in the max-pseudo-sequentiality manner are universal), a group of neurons are used to replace the weight on a synapse. Although the universality of HSN P systems with usual synapses working in the max-pseudo-sequentiality manner is obtained, the following problem is still of interest. Can we directly construct an HSN P system with usual synapses working in the max-pseudo-sequentiality manner to simulate a register machine (instead of achieving the universality by modifying a universal HSN P system with weighted synapses working in the max-pseudo-sequentiality manner), where each instruction is associated with one neuron? Of course, it is expected that in such universal HSN P systems, the set of rules is not too complicated.

It remains open whether HSN P systems with usual synapses working in the max-sequentiality manner are universal.

## Acknowledgements

This work was supported by National Natural Science Foundation of China (61033003, 91130034 and 30870826), PhD Programs Foundation of Ministry of Education of China (20100142110072), Fundamental Research Funds for the Central Universities (2011TS005 and 2011TS006) and Natural Science Foundation of Hubei Province (2011CDA027).

## References

[1] H. Chen, M. Ionescu, T.-O. Ishdorj, A. Păun, Gh. Păun, and M.J. Pérez-Jiménez, *Spiking neural P systems with extended rules: Universality and languages*, Nat. Comput. 7(2) (2008), pp. 147–166.

[2] G. Ciobanu, L. Pan, Gh. Păun, and M.J. Pérez-Jiménez, *P systems with minimal parallelism*, Theoret. Comput. Sci. 378 (2007), pp. 117–130.

[3] R. Freund, *Asynchronous P systems and P systems working in the sequential mode*, in WMC5 2004, Lecture Notes in Computer Science, vol. 3365, G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, and A. Salomaa (eds.), Springer, Berlin, 2005, pp. 36–62.

[4] O.H. Ibarra, A. Păun, and A. Rodríguez-Patón, *Sequential SNP systems based on min/max spike number*, Theoret. Comput. Sci. 410 (2009), pp. 2982–2991.

[5] M. Ionescu, Gh. Păun, and T. Yokomori, *Spiking neural P systems*, Fund. Inform. 71 (2006), pp. 279–308.

[6] T.-O. Ishdorj, A. Leporati, L. Pan, X. Zeng, and X. Zhang, *Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources*, Theoret. Comput. Sci. 411 (2010), pp. 2345–2358.

[7] M. Minsky, *Computation – Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[8] L. Pan, Gh. Păun, and M.J. Pérez-Jiménez, *Spiking neural P systems with neuron division and budding*, Sci. China Inf. Sci. 54 (2011), pp. 1596–1607.

[9] L. Pan, X. Zeng, X. Zhang, and Y. Jiang, *Spiking neural P systems with weighted synapses*, Neural Process. Lett. 35 (2012), pp. 13– 27.

[10] A. Păun and Gh. Păun, *Small universal spiking neural P systems*, BioSystems. 90 (2007), pp. 48–60.

[11] Gh. Păun, *Computing with membranes*, J. Comput. System Sci. 61 (2000), pp. 108–143 (first circulated as TUCS Research Report No. 208, November 1998).

[12] Gh. Păun, *Membrane Computing – An Introduction*, Springer-Verlag, Berlin, 2002.

[13] G. Rozenberg and A. Salomaa (eds.), *Handbook of Formal Languages, 3 volumes*, Springer-Verlag, Berlin, 1997.

[14] X. Zeng, X. Zhang, and L. Pan, *Homogeneous spiking neural P systems*, Fund. Inform. 97 (2009), pp. 1–20.

[15] X. Zhang, X. Zeng, and L. Pan, *Smaller universal spiking neural P systems*, Fund. Inform. 87 (2008), pp. 117–136.

[16] X. Zhang, X. Zeng, and L. Pan, *On languages generated by asynchronous spiking neural P systems*, Theoret. Comput. Sci. 410 (2009), pp. 2478–2488.