Computing and Informatics, Vol. 39, 2020, 925–951, doi: 10.31577/cai_2020_5_925

ASYNCHRONOUS SPIKING NEURAL P SYSTEMS WITH MULTIPLE CHANNELS AND SYMBOLS

Wenmei Yi, Zeqiong Lv, Hong Peng*

School of Computer and Software Engineering
Xihua University, Chengdu, 610039, China
e-mail: wenmeiyee@foxmail.com, ph.xhu@hotmail.com

Xiaoxiao Song, Jun Wang

School of Electrical Engineering and Electronic Information
Xihua University, Chengdu, 610039, China
e-mail: wj.xhu@hotmail.com

Abstract. Spiking neural P systems (SNP systems, in short) are a class of distributed parallel computation systems, inspired from the way that the neurons process and communicate information by means of spikes. A new variant of SNP systems, which works in asynchronous mode, asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems, in short), is investigated in this paper. There are two interesting features in ASNP-MCS systems: multiple channels and multiple symbols. That is, every neuron has more than one synaptic channels to connect its subsequent neurons, and every neuron can deal with more than one type of spikes. The variant works in asynchronous mode: in every step, each neuron can be free to fire or not when its rules can be applied. The computational completeness of ASNP-MCS systems is investigated. It is proved that ASNP-MCS systems as number generating and accepting devices are Turing universal. Moreover, we obtain a small universal function computing device that is an ASNP-MCS system with 67 neurons. Specially, a new idea that can solve "block" problems is proposed in INPUT modules.

Keywords: Membrane computing, spiking neural P systems, asynchronous systems, multiple channels, multiple symbols, Turing universality

1 INTRODUCTION

Abstracted from the structure and functioning of living cells as well as the cooperation of cells in tissue, organs and biological nervous systems, membrane computing is a class of distributed and parallel computation systems [1, 2], known as P systems. There are three main types of P systems: cell-like P systems, tissue-like P systems and neural-like P systems. Abstracted from distinct biological cell mechanisms, a lot of P systems and variants have been proposed, for example, tissue-like P systems [3], population P systems [4], P colonies [5], spiking neural P systems [6], and the latest works can be found on the membrane computing website (`http://ppage.psystems.eu`). In terms of computational theory, most of P systems have been proven to be Turing universal, and some NP-hard problems have been solved in a feasible time [7, 8, 9, 10, 11]. Moreover, there are a various of applications of P systems, for instance, ecology and structural biology [12, 13], function optimization [14], machine learning [15, 16, 17], image and signal processing [18, 19, 20, 21, 22, 23].

Spiking neural P systems (SNP systems, in short) were first proposed by Ionescu et al. [6], inspired by the way of transmitting and exchanging information, by means of spikes, between neurons. SNP systems are also a class of distributed and parallel computation systems. Directed graphs are used to express SNP systems, where the nodes are the neurons and the arcs are used to denote the synapses between these neurons. Each SNP system consists of two components: data and rules. The data is denoted by the number of spikes contained in it and is evolved by rules. There are two forms of rules: spiking rule and forgetting rule. Spiking rule has the form $E/a^c \to a^p$, where E is a regular expression over $\{a\}$, c is the number of spikes consumed by the rule and p is the number of the produced spikes, and $c \geq p \geq 1$. The semantics of spiking rule can be explained as follows. Suppose that neuron σ has a spiking rule $E/a^c \to a^p$ and contains n spikes satisfying $a^n \in L(E)$. The neuron fires and consumes c spikes, and then it produces p spikes and sends them to all subsequent neurons connected with it. Forgetting rule has the form $a^s \to \lambda$, where $s \geq 1$. If forgetting rule $a^s \to \lambda$ is applied in the neuron, then s spikes are removed from it and no spike is produced.

Most SNP systems work in synchronous mode. A global clock is assumed for the synchronization of all neurons, and all neurons in the systems work in parallel, and the rules in each neuron are applied sequentially. When there are more than one rules can be applied in a neuron, one of them must be chosen non-deterministically and applied.

Generally, there are three main research topics:

1. theoretical works,
2. application and
3. simulation systems.

Many variants of SNP systems were proposed, for example, SNP systems with astrocytes [24, 25], SNP systems with anti-spikes [26], SNP systems with weights [27], SNP systems with thresholds [28], SNP systems with rules on synapses [29, 30, 31], SNP systems with multiple channels [32, 33], coupled neural P systems [34], dynamic threshold neural P systems [35], SNP systems with polarizations [36], SNP system with inhibitory rules [37], dendrite P systems [38], nonlinear SNP systems [39], and so on. In addition, several working modes have been investigated, such as asynchronous mode [40], asynchronous mode with local synchronization [41], and sequential mode [42]. Turing universality is one of computational theory of SNP systems. SNP systems and variants can be considered as four devices: number generating/accepting devices, function computing devices as well as language generating devices. In universality investigation, register machines are often regarded as standard model, because it has been proven that register machines can compute/accept any Turing computable number set and obtain a small universal function computing device. Therefore, by simulating register machines, it has been proven that most variants of SNP systems are Turing universal [43, 44]. Moreover, fuzzy logic was introduced into SNP systems to propose a variety of fuzzy spiking neural P systems [45, 46], which have been applied in fault diagnosis [47, 48, 49, 50]. In addition, a number of simulation systems have been developed, for example, P-Lingua [51] and SNP system simulator on GPU [52].

It is no doubt that the synchronization plays a vital role in the proof of above results, however, the assumption of global clock is rather natural from a neurobiological point of view. Therefore, SNP systems working in non-synchronous modes have received much attention in the recent years, especially, in asynchronous mode. The SNP systems working in asynchronous mode are called asynchronous SNP systems (ASNP systems, in short). In asynchronous mode, the global clock is removed and every neuron is not obligatory to use its rules. Therefore, each neuron is free to choose time to fire without any time restriction when its rules are available. New spikes that are received from adjacent neurons may cause the rules to no longer be available. In this case, the computation will continue to work under the new configuration. The result of the computation is no longer relevant with the distance in time because of the asynchronous working mode. Thus, the total number of spikes, which is sent out to the environment, is regarded as the result of the computation. Cavaliere et al. [40] provided a specific description about asynchronization and proved that asynchronous SNP systems with extended rules are equivalent with Turing machines. After that, several asynchronous SNP systems have been investigated, such as asynchronous SNP systems with local synchronization [41], asynchronous SNP systems with rules on synapses [53], asynchronous SNP systems with structural plasticity [54] and asynchronous SNP systems with anti-spikes [55]. These asynchronous systems have been proven to be Turing universal. In addition, Cavaliere et al. [56] investigated the decidability and undecidability of asynchronous SNP systems. The language generating problems of asynchronous SNP systems have been discussed in Zhang et al. [57].

This work discusses a new variant of SNP systems, which works in asynchronous mode and has two interesting features (multiple channels and multiple symbols), asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems, in short). ASNP-MCS systems are different from the existing asynchronous SNP systems in the following two aspects:

1. Every neuron in ASNP-MCS systems has one or more synaptic channels, thus, different sets of subsequent neurons can be connected with it. It is suitable for ASNP-MCS systems, because of the multiple channels feature, to characterize higher-order dynamic systems.
2. More than one symbols are considered in ASNP-MCS systems. Therefore, the rules in ASNP-MCS systems are extended to handle these multiple symbols. It is also suitable for ASNP-MCS systems, due to the multiple symbols feature, to simulate some complicated systems with different parts.

This work investigates the computational power of ASNP-MCS systems. By simulating register machine, it is proven that ASNP-MCS systems as number generating/accepting devices are Turing universal. In addition, we construct an ASNP-MCS system with 67 neurons as a small universal function computing device. The result can be interpreted as the reason that the loss of computational power caused by removing synchronization can be offset by the use of multiple channels and multiple symbols.

The remainder of this paper is organized as follows. In Section 2, we review some basic mathematical knowledge that will be useful for investigation of universality. The definition of ASNP-MCS systems and an illustrative examples are given in Section 3. In Section 4, we first discuss the computational power of ASNP-MCS systems as number generating/accepting devices, and then we construct a small universal ASNP-MCS system for computing functions. Finally, conclusions and future work are drawn in Section 5.

2 PRELIMINARIES

It is assumed that readers have some knowledge with formal language theory and membrane computing. Basic notions and notations are reviewed in this section.

For an alphabet O, the set of all finite strings of symbols from O is denoted by O^*, and the set of all nonempty strings over O is denoted by O^+; the empty string is denoted by λ.

A regular expression over an alphabet O is defined as follows:

1. λ and each $a \in O$ is a regular expression;
2. if E_1 and E_2 are regular expressions over O, then $(E_1)(E_2)$, $(E_1)\bigcup(E_2)$ and $(E_1)^+$ are regular expressions over O, and
3. nothing else is a regular expression over O.

With each regular expression E we associate with a language $L(E)$, defined in the following way:

1. $L(\lambda) = \{\lambda\}$ and $L(a) = \{a\}$, for all $a \in O$;
2. $L((E_1)\bigcup(E_2)) = L(E_1)\bigcup L(E_2)$, $L((E_1)(E_2)) = L(E_1)L(E_2)$, and $L((E_1)^+) = (L(E_1))^+$, for all regular expressions E_1, E_2 over O.

When writing a regular expression, unnecessary parentheses can be omitted. We can simply write $E^+\bigcup\{\lambda\}$ as E^*.

A register machine, which is a construct $M = (m, H, l_0, l_h, I)$, can be used to prove the universality of ASNP-MCS systems, where m is the number of registers, H is the set of instruction labels, l_0 is the starting label, l_h is the halting label (assigned to instruction HALT), and I is the set of instructions. Each label from H corresponds to an instruction from I. There are instructions of three forms:

1. $l_i : (\mathrm{ADD}(r), l_j, l_k)$ (add 1 to register r then go non-deterministically to one of the instructions with labels l_j, l_k).
2. $l_i : (\mathrm{SUB}(r), l_j, l_k)$ (if register r is non-zero, then subtract 1 from it and go to the instruction with label l_j; otherwise, go to the instruction with label l_k).
3. l_h : HALT (the halting instruction).

It is well-known that a register machine M can generate/accept any Turing computable number set (denoted by NRE).

Number n can be generated by register machine in the following way. The register machine starts with all registers empty (for example, storing the number zero). What instruction activated first is the instruction with label l_0. Then, the subsequent instructions are processed in order. If the register machine reaches the halting instruction, then the computation is completed, and the result of the computation is the number n stored in the first register r_0. We denote by $N_{gen}(M)$ the set of all numbers generated by M.

The register machine M can also accept the numbers. We denote by $N_{acc}(M)$ the set of numbers accepted by M. Its working mechanism can be illustrated as follows. At the beginning, all registers are empty except the first register, and a number is introduced into the first register. In accepting mode, register machine is deterministic, meaning that $l_i : (\mathrm{ADD}(r), l_j)$ is used to substitute $l_i : (\mathrm{ADD}(r), l_j, l_k)$ as the ADD instruction.

Functions of form $f : N^k \to N$ can be computed by register machines. It works as follows: at the beginning, all registers are empty and k parameters are introduced into k specific registers; register machine M starts with instruction l_0, and then continues a series of computations until it reaches the halting instruction l_h. The computed function value will be stored in a specific register r when the system halts. In computing mode, M is deterministic, where ADD instructions have the form $l_i : (\mathrm{ADD}(r), l_j)$.

Korec [58] introduced a small universal register machine, $M_u = (8, H, l_0, l_h, I)$, for computing functions, shown in Figure 1. The register machine contains 8 registers and 23 instructions. By introducing numbers $g(x)$ and y into registers 1 and 2, respectively, the register machine M_u can compute function $\varphi_x(y) = M_u(g(x), y)$, where g is a recursively function for all natural numbers x and y. When register machine M_u halts, the number stored in the register 0 is the computed function value.

In this work, we will discuss the universality of ASNP-MCS systems as function computing devices by means of register machine M_u as a standard model.

l_0: $(SUB(1), l_1, l_2)$	l_1: $(ADD(7), l_0)$	l_2: $(ADD(6), l_3)$
l_3: $(SUB(5), l_2, l_4)$	l_4: $(SUB(6), l_5, l_3)$	l_5: $(ADD(5), l_6)$
l_6: $(SUB(7), l_7, l_8)$	l_7: $(ADD(1), l_4)$	l_8: $(SUB(6), l_9, l_0)$
l_9: $(ADD(6), l_{10})$	l_{10}: $(SUB(4), l_0, l_{11})$	l_{11}: $(SUB(5), l_{12}, l_{13})$
l_{12}: $(SUB(5), l_{14}, l_{15})$	l_{13}: $(SUB(2), l_{18}, l_{19})$	l_{14}:$(SUB(5), l_{16}, l_{17})$
l_{15}: $(SUB(3), l_{18}, l_{20})$	l_{16}: $(ADD(4), l_{11})$	l_{17}: $(ADD(2), l_{21})$
l_{18}: $(SUB(4), l_0, l_h)$	l_{19}: $(SUB(0), l_0, l_{18})$	l_{20}: $(ADD(0), l_0)$
l_{21}: $(ADD(3), l_{18})$	l_h:$HALT$	

Figure 1. A small universal register machine M_u

3 ASYNCHRONOUS SPIKING NEURAL P SYSTEMS WITH MULTIPLE CHANNELS AND SYMBOLS

3.1 Definition

Definition 1. An ASNP-MCS system, of degree $m \geq 1$, is a construct:

$$\Pi = (O, L, \sigma_1, \sigma_2, \dots, \sigma_m, \mathrm{syn}, \mathrm{in}, \mathrm{out})$$

where

1. $O = \{a_1, a_2, \dots, a_k\}$ is the alphabet ($a_1, a_2, \dots, a_k$ denote k types of spikes, respectively);
2. $L = \{1, 2, \dots, N\}$ is the alphabet of channel labels;
3. $\sigma_1, \dots, \sigma_m$ are neurons, of the form $\sigma_i = (\vec{n}_i, L_i, R_i)$, $1 \leq i \leq m$, where
   (a) $\vec{n}_i = (n_{i1}, n_{i2}, \dots, n_{ik})$ is a k-dimensional vector, where $n_{ij} \geq 0$ is the initial number of spikes of jth type a_j contained in neuron σ_i, $1 \leq j \leq k$;
   (b) $L_i \subseteq L$ is a finite set of channel labels used in neuron σ_i;
   (c) R_i is a finite set of rules of the following two forms:
      i Spiking rule $E/a_1^{c_1}a_2^{c_2}\cdots a_k^{c_k} \to a_1^{p_1}a_2^{p_2}\cdots a_k^{p_k}(l)$, where E is a regular expression over O, and $c_j \geq 0$, $p_j \geq 0$ $(1 \leq j \leq k)$, $c_1 + c_2 + \cdots + c_k \geq p_1 + p_2 + \cdots + p_k \geq 1$, $l \in L_i$;
      ii Forgetting rule $a_1^{s_1}a_2^{s_2}\cdots a_k^{s_k} \to \lambda$, where $s_j \geq 0$ $(1 \leq j \leq k)$ and $s_1 + s_2 + \cdots + s_k \geq 1$, with the restriction that for each rule $E/a_1^{c_1}a_2^{c_2}\cdots a_k^{c_k} \to a_1^{p_1}a_2^{p_2}\cdots a_k^{p_k}(l)$ of type (i) from R_i, we have $a_1^{s_1}a_2^{s_2}\cdots a_k^{s_k} \notin L(E)$;

4. syn $= \{(i, j, l)\} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \times L$ with $(i, i, l) \notin$ syn for $\forall 1 \leq i \leq m$ and $\forall l \in L$ (synapse connections);
5. in indicates the input neuron of the system;
6. out indicates the output neuron of the system.

There is only one single type of spikes, denoted by symbol a, in SNP systems, while ASNP-MCS systems proposed in this work have several distinct types of spikes, denoted by symbols a_1, a_2, $\ldots$, a_k, and these multiple symbols can be interpreted as electrical signals with distinct frequencies. The number of spikes in every neuron in an SNP system is denoted by a natural number. However, since there are spikes of k types in an ASNP-MCS system, a k-dimensional vector, $\vec{n}_i = (n_{i1}, n_{i2}, \ldots, n_{ik})$, is considered to indicate the number of each neuron. If $n_{ij} > 0$, one or more spikes of type a_j are contained in the neuron σ_i. If $n_{ij} = 0$, there is no spike of type a_j in the neuron σ_i.

An ASNP-MCS system can be represented by a directed graph, where the nodes are used for labeling m neurons and the arcs are used for denoting the synapses between these neurons. The connection relationships between m neurons are described by syn $= \{(i, j, l)\} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \times L$, meaning that neuron σ_i connects neuron σ_j via channel (l) .

Since ASNP-MCS systems work in asynchronous mode, the use of rules in neurons is not obligatory, which means that a rule can be used immediately or also can be used later if the spikes in the neuron enable the rule. If the number of spikes in the neuron has been changed before using the rule, such as new spikes coming, and the present rule can not be used any more, then the computation continues in the new circumstance. Notice that the produced spikes in the neuron will be sent immediately when the neuron applies the rule in a later step.

Spiking rules and forgetting rules also exist in ASNP-MCS systems. What is the meaning of spiking rules of form (i) can be explained as follows. The rule $E/a_1^{c_1}a_2^{c_2}\cdots a_k^{c_k} \to a_1^{p_1}a_2^{p_2}\cdots a_k^{p_k}(l)$ in neuron σ_i can be applied at some time with the condition of $a_1^{n_1}a_2^{n_2}\cdots a_k^{n_k} \in L(E)$. When neuron σ_i fires, c_j spikes of type a_j are consumed (thus $n_j - c_j$ spikes of type a_j are remained), and p_j spikes of type a_j are produced, $1 \leq j \leq k$. The spikes generated by neuron σ_i are sent to the subsequent neurons via channel (l). The semantics of forgetting rules of form (ii) can be described as follows. If the spikes in neuron σ_i are exactly s_j spikes of type a_j, $1 \leq j \leq k$, then the rule $a_1^{s_1}a_2^{s_2}\cdots a_k^{s_k} \to \lambda$ can be enabled, which means that all s_j spikes of type a_j are removed from neuron σ_i.

We use the following $m \times k$ matrix to describe the initial configuration in ASNP-MCS systems,

$$C_0 = \begin{pmatrix} n_{11} & n_{12} & \cdots & n_{1k} \\ n_{21} & n_{22} & \cdots & n_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ n_{m1} & n_{m2} & \cdots & n_{mk} \end{pmatrix}_{m \times k}$$

where n_{ij} is the initial number of spikes of type a_j contained in neuron σ_i, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, k$. Besides, the similar $m \times k$ matrix also can describe the configuration at any time t during the computation. The transitions among configurations can be therefore defined. One can define a computation with the sequence of transitions starting from the initial configuration. When there is no rule which can be applied and no "block" condition happened, the computation halts.

In SNP systems, we can associate any computation (halting or not) with a spike train: the sequence of zeros and ones describing the behavior whether the output neuron spikes. Since ASNP-MCS systems work in asynchronous mode, zeros sent out by the output neuron are normal. Thus, we define the result of a computation as the number of ones sent out by the output neuron starting from the initial configuration. We denote by $N_{gen}(\Pi)$ the set of numbers generated by Π, where subscript *gen* indicates that the present system works in generating mode. We denote by $N_{gen}\text{ASNP-MCS}_m^n$ the family of all sets $N_{gen}(\Pi)$ computed by ASNP-MCS systems with at most m neurons and at most n rules in every neuron.

An ASNP-MCS system Π can work in accepting mode. There is no output neuron any more. Instead, the input neuron is added to receive a spike train from the environment. The system Π begins the computation by reading a spike train from the environment, and $3\mathrm{T}n$ spikes of type a is stored in a specific neuron. Denote by $N_{acc}(\Pi)$ the set of numbers accepted by system Π, where subscript *acc* represents that the system works in accepting mode. Denote by $N_{acc}\text{ASNP-MCS}_m^n$ the family of all sets $N_{acc}(\Pi)$ accepted by ASNP-MCS systems, which have at most m neurons and at most n rules in every neuron.

3.2 An Example

An example is provided to explain the differences between an SNP-MCS system and an ASNP-MCS system, shown in Figure 2. SNP-MCS system works in synchronous mode, while ASNP-MCS system works in asynchronous mode. There are three neurons, σ_1, σ_2 and σ_3, labeled by 1, 2 and 3 in Figure 2. Neuron σ_1 has the spikes of two types (type a and type b), and it has only one synaptic channel labeled by (1). Neuron σ_3 contains the spikes of two types, however, it has two different synaptic channels, labeled by (1) and (2). There is only one type of spikes, type a, and a single synaptic channel labeled by (1) in neuron σ_2. Neuron σ_1 starts with a spike of type a, and neuron σ_3 starts with a spike of type b. What the mainly difference between an SNP-MCS system and an ASNP-MCS system is their working mode.

In the synchronous mode, since neuron σ_1 initially has a spike of type a and neuron σ_3 initially has a spike of type b, the spikes enable rule $a \to b(1)$ and rule $b \to a(1)$ in neurons σ_1 and σ_3, respectively. Thus, two rules can be applied simultaneously. Neuron σ_1 sends a spike of type b to neuron σ_3 via channel (1). Neuron σ_3 emits a spike of type a to neuron σ_1 through channel (1). When neuron σ_1 receives a spike of type a from neuron σ_3, it can apply rule $a \to b(1)$ again. Similarly, neuron σ_3 also receives a spike of type b from neuron σ_1 and it can apply its rule $b \to a(1)$, too. Therefore, the two neurons simultaneously fire again and

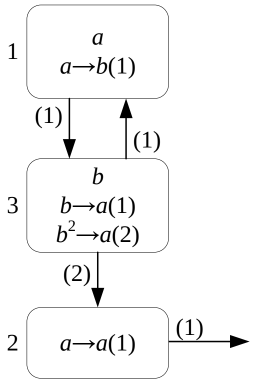


Figure 2. An example of spiking neural P systems with multiple channels and symbols

exchange a spike with each other. Repeatedly, in every step, neurons σ_1 and σ_3 will exchange a spike, and no spike is sent to the environment. Therefore, there is no result of the computation.

For the ASNP-MCS system that works in the asynchronous mode, when the number of spikes in a neuron enables its rule, the neuron can fire its rule at some time, sooner or later. Here, the rules in neurons σ_1 and σ_3 are available separately, but neurons σ_1 and σ_3 are free to choose a time to fire. Thus, there are three cases.

Case 1: Neuron σ_1 fires before neuron σ_3. At first, rule $a \rightarrow b(1)$ in neuron σ_1 can be applied, which means that neuron σ_1 sends a spike of type b to neuron σ_3. The number of spikes in neuron σ_3 has been changed to two spikes of type b, after receiving a spike sent by neuron σ_1. Thus, rule $b^2 \rightarrow a(1)$ can be applied. At a later time, neuron σ_3 fires and sends a spike of type a to neuron σ_2 through channel (2). Rule $a \rightarrow a(1)$ in neuron σ_2 can be applied when it receives a spike, and neuron σ_2 sends a spike of type a to the environment. In this case, only one single spike is sent to the environment. Therefore, the computation result of the system is 1.

Case 2: Neuron σ_1 fires later than neuron σ_3. Since there is a spike of type b in neuron σ_3, rule $b \rightarrow a(1)$ can be applied, and neuron σ_1 receives a spike of type a from neuron σ_3 via channel (1). The number of spikes in neuron σ_1 are two now and there is no rule can be applied, thus the computation is blocked. We can know that there is no any output, thus no any computation result.

Case 3: Neurons σ_1 and σ_3 fire together. In this case, the running of an ASNP-MCS system is the same with the situation in synchronous mode at the first round. So, after the first round, the number of spikes in every neuron is back to its available state and no spike is sent out. Then, neurons σ_1 and σ_3 face a choice again: who want to fire first? And there are also three choices:

1. If it is the Case 1, neuron σ_1 fires before neuron σ_3, then a spike will be sent from the system and the computation will halt. Hence, the final result of the computation is 1;
2. If it is the Case 2, neuron σ_1 fires later than neuron σ_3, then there will be no result in the second round and the computation will be blocked;
3. If it is the Case 3, which means that the two neurons fire together, no spike will be sent to the environment and the numbers of spikes in neurons σ_1 and σ_3 will be back to their available state again.

And then there will be also the three choices again at the next round. The above situation is repeated again. Therefore, if the neurons choose Case 3 at the second round, the final computation result of the system will be one of the following three results: generating the result (number 1), blocked and infinitive repeat.

From the description above, we know that ASNP-MCS systems, which work in the asynchronous mode, have more nondeterministic results compared with SNP-MCS systems.

4 UNIVERSALITY RESULTS

In this section, we will discuss the Turing universality of asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems, in short) as number generating/accepting devices and function computing devices. We prove the universality by simulating the register machine. The systems proposed can generate/accept any sets of recursively enumerable numbers (the family of sets of recursively enumerable numbers is denoted by NRE) and any recursively enumerable computational function.

We construct an ASNP-MCS system Π to simulate register machine M and the result computed by M is presented by the number of spikes that the output neuron sends to the environment. Without loss of generality, two symbols, a and b, are considered in Π, thus $O = \{a, b\}$. INPUT module, ADD module, SUB module and FIN module are constructed to simulate instructions of M, shown in Figures 3, 4, 5, 6, 7 and 8. If t_r is the number of all SUB instructions that act on the same register r, a constant is defined as follows:

$$T = 8 * \max\{t_r | 0 \leq r \leq 7\} = 8 * t_5 = 8 * 4 = 32.$$

The following rule is used to activate instruction neurons: when instruction neurons σ_{l_i}, σ_{l_j} and σ_{l_k} receive $3T$ spikes of type b, they will be activated and execute the corresponding operations. In the process of the computation, the content of register r is coded by the number of spikes in neuron σ_r through the following way: if there is a number $n(\geq 0)$ in register r, then neuron σ_r has 3Tn spikes of type a, vice versa.

Rule $b^{3T} \to a^{\psi(p)}(q)$ is a formal spiking expression, used in INPUT, ADD, SUB and FIN modules. It works as follows. When the rule is available, the neuron consumes 3T spikes of type b and sends $\psi(p)$ spikes of type a to the subsequent neurons connected with the neuron, where the rule uses channel (q). We define the function $\psi(p)$ on the sets of instruction symbols as follows:

$$\psi(p) = \begin{cases} 3T, & \text{if } p \text{ is an ADD instruction;} \\ 2T + s, & \text{if } p \text{ is } i^{\text{th}} \text{ SUB instruction in all SUB instructions on register } r; \\ 1, & \text{if } p \text{ is the output instruction.} \end{cases}$$

In this paper, we use the following way to define the value of s in the SUB instructions which act on the same register r:

- If the instruction is the first SUB instruction in register r, then $s = 1$;
- If the instruction is the second SUB instruction in register r, then $s = 2$;
- If the instruction is the third one in register r, then $s = 4$;
- If the instruction is the forth one, then $s = 9$.

For example, there are four SUB instructions acted on register 5 in universal register machine M_u, $l_3 : (\text{SUB}(5), l_2, l_4)$, $l_{11} : (\text{SUB}(5), l_{12}, l_{13})$, $l_{12} : (\text{SUB}(5), l_{14}, l_{15})$, $l_{14} : (\text{SUB}(5), l_{16}, l_{17})$. Therefore, because the first SUB instruction in register 5 is instruction $l_3 : (\text{SUB}(5), l_2, l_4)$, the function is $\psi(p) = 2T + 1$. The second SUB instruction is instruction l_{11} and its function is $\psi(p) = 2T + 2$. Similarly, the function of the third SUB instruction, $l_{12} : (\text{SUB}(5), l_{14}, l_{15})$, is $\psi(p) = 2T + 4$. The function of the forth SUB instruction l_{14} is $\psi(p) = 2T + 9$.

4.1 ASNP-MCS Systems as Number Generating Devices

Theorem 4.1. $N_{gen}\text{ASNP-MCS}^3_* = \text{NRE}$.

Proof. It is only proven that $\text{NRE} \subseteq N_{gen}\text{ASNP-MCS}^3_*$, because it is obvious for conclusion $N_{gen}\text{ASNP-MCS}^3_* \subseteq \text{NRE}$. To this aim, we characterize NRE by a non-deterministic register machine M working in generating mode.

In order to prove this conclusion, we construct an ASNP-MCS system Π_1 to simulate the register machine M. The system consists of three parts, ADD module, SUB module and FIN module, which simulate ADD instruction, SUB instruction and halting instruction, respectively.

1. ADD module, simulating $l_i : (\text{ADD}(r), l_j, l_k)$.

   The ADD module is shown in Figure 3. Suppose that we simulate instruction $l_i : (\text{ADD}(r), l_j, l_k)$ at some time, which means that neuron σ_{l_i} has 3T spikes of type b and no any spikes are in other neurons except those associated with registers. The rule $b^{3T} \to a^{3T}(1)$ in neuron σ_{li} is applied at some time and then neuron σ_{li} sends 3T spikes of type a to neurons σ_r and $\sigma_{l_{i1}}$. When neuron σ_r

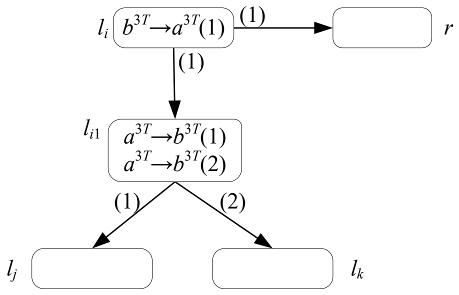


Figure 3. A nondeterministic ADD module

receives the spikes from neuron σ_{l_i}, number 1 is added into the corresponding register. Two rules in neuron $\sigma_{l_{i1}}$ become available after it receives the spikes from neuron σ_{l_i}, but one of the two rules can be applied non-deterministically. If rule $a^{3T} \to b^{3T}(1)$ is applied, neuron σ_{l_j} will receive 3T spikes of type b via channel (1). If rule $a^{3T} \to b^{3T}(2)$ is applied, neuron $\sigma_{l_{i1}}$ will send the spikes to neuron σ_{l_k} through channel (2).

As can be seen from above, ADD module can correctly simulate ADD instruction: starting from $3T$ spikes of type b in neuron σ_{l_i}, $3T$ spikes of type a are added in neuron σ_r (meaning that the corresponding register r is added by 1), and one of two instruction neurons, σ_{l_j} and σ_{l_k}, receives $3T$ spikes of type b non-deterministically.

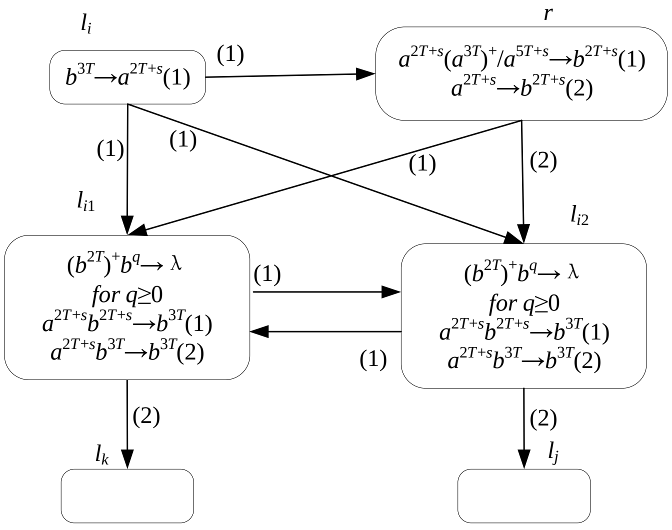

l_i
r
$b^{3T} \to a^{2T+s}(1)$
(1)
$a^{2T+s}(a^{3T})^+/a^{5T+s} \to b^{2T+s}(1)$
$a^{2T+s} \to b^{2T+s}(2)$
(1)
(1)
(1)
(2)
l_{i1}
l_{i2}
$(b^{2T})^+b^q \to \lambda$
$for\ q \geq 0$
$a^{2T+s}b^{2T+s} \to b^{3T}(1)$
$a^{2T+s}b^{3T} \to b^{3T}(2)$
(1)
(1)
$(b^{2T})^+b^q \to \lambda$
$for\ q \geq 0$
$a^{2T+s}b^{2T+s} \to b^{3T}(1)$
$a^{2T+s}b^{3T} \to b^{3T}(2)$
(2)
(2)
l_k
l_j

Figure 4. SUB module

2. SUB module, simulating $l_i : (\mathrm{SUB}(r), l_j, l_k)$.

   This module is shown in Figure 4. Suppose that a SUB instruction is simulated at some time, 3T spikes of type b are in neuron σ_{l_i} and no spike is in other neurons except neuron σ_r (i.e., the multiple of 3T$^{\text{th}}$ spikes in neuron σ_r is the number in the corresponding register r). The SUB instruction works as follows. At some time, neuronσ_{l_i} fires, and then 3T spikes of type b are consumed and $2T+s$ spikes of type a are produced, and the generated spikes are sent to neurons σ_r, $\sigma_{l_{i1}}$ and $\sigma_{l_{i2}}$ via channel (1). Neuron σ_r fires, at a later time, according to the number of spikes in it:

   (a) If there are several spikes in neuron σ_r, indicating that the number in register r is not zero, then rule $a^{2T+s}(a^{3T})^+/a^{5T+s} \to b^{2T+s}(1)$ can be applied;
   (b) If there is no spike in neuron σ_r, indicating that the number in register r is 0, then rule $a^{2T+s} \to b^{2T+s}(2)$ can be applied. There are the following two cases.

   Case 1: If several spikes are in neuron σ_r (i.e., the number in register r is greater than 0), then, at a later time, rule $a^{2T+s}(a^{3T})^+/a^{5T+s} \to b^{2T+s}(1)$ can be applied. Therefore, neuron σ_r consumes $5T+s$ spikes of type a and produces $2T+s$ spikes of type b, and then it sends the produced spikes to neuron $\sigma_{l_{i1}}$ by channel (1). As a result, neuron $\sigma_{l_{i1}}$ has $2T+s$ spikes of type a sent by neuron σ_{l_i} and $2T+s$ spikes of type b sent by neuron σ_r. Hence, rule $a^{2T+s}b^{2T+s} \to b^{3T}(1)$ can be applied. At a later time, neuron $\sigma_{l_{i1}}$ fires and transmits 3T spikes of type b to neuron $\sigma_{l_{i2}}$. Neuron $\sigma_{l_{i2}}$ receives $2T+s$ spikes of type a sent by neuron σ_{l_i} and 3T spikes of type b sent by neuron $\sigma_{l_{i1}}$. So, rule $a^{2T+s}b^{3T} \to b^{3T}(2)$ is enabled. At some time, neuron $\sigma_{l_{i2}}$ fires and emits 3T spikes of type b to neuron σ_{l_j}.

   Case 2: If no spike is in neuron σ_r, indicating that the number in register r is 0, then, at a later time, rule $a^{2T+s} \to b^{2T+s}(2)$ is applied to send $2T+s$ spikes of type b to neuron $\sigma_{l_{i2}}$ via channel (2). Except from receiving $2T+s$ spikes of type b, neuron $\sigma_{l_{i2}}$ also receives $2T+s$ spikes of type a from neuron σ_{l_i}. Therefore, the spikes in the neuron enable rule $a^{2T+s}b^{2T+s} \to b^{3T}(1)$. Then, at some time, the rule in neuron $\sigma_{l_{i2}}$ is applied to send 3T spikes of type b to neuron $\sigma_{l_{i1}}$. With $2T+s$ spikes of type a and 3T spikes of type b in neuron $\sigma_{l_{i1}}$, rule $a^{2T+s}b^{3T} \to b^{3T}(2)$ can be used at a later time. Therefore, neuron $\sigma_{l_{i1}}$ sends 3T spikes of type b to neuron σ_{l_k} via channel (2).

   Note that there is interference between SUB modules and other modules, which means that the same register could be operated by different instructions. Here an example is provided to illustrate this question: instructions $l_{11} : (\mathrm{SUB}(5), l_{12}, l_{13})$ and $l_{12} : (\mathrm{SUB}(5), l_{14}, l_{15})$, for instance, all act on the register 5. The SUB modules that act on the same register, register 5 here, all will receive the spikes via channel (1) (the number in register is not null) or via channel (2) (the number in register is 0) from neuron σ_r. Specifically, suppose that instruction l_{11}

is activated, so the corresponding neurons in SUB module of instruction l_{11} is activated (active neurons, in short), too. Other instructions, like l_{12} here, are non-activated, so the corresponding neurons in SUB modules of other instructions, no instruction l_{11}, are also not activated (passive neurons for short). Instruction l_{11} is the second SUB instruction of register 5. Therefore, the function of its sets of instruction symbols is $\psi(p) = 2T + s = 2T + 2$. So, neuron σ_r will send $2T + 2$ spikes of type b to the following neurons which are also connected with register 5 through channel (1) if register 5 contains numbers. This means, not only active neuron $\sigma_{l_{111}}$ (neuron $\sigma_{l_{i1}}$ connects with neuron σ_r via channel (1) in the module of l_{11}) but also passive neuron $\sigma_{l_{121}}$ (neuron $\sigma_{l_{i1}}$ connects with neuron σ_r via channel (1) in the module of l_{12}) as well as other passive neurons that are connected with neuron σ_r via channel (1) can receive the spikes. For active neurons, the spikes received are exactly they want. However, the passive neurons receive the spikes in a wrong way and cannot refuse to accept them because of the interference between neurons, so we called these spikes, "wrong spikes".

We can find that, if neurons $\sigma_{l_{i1}}$ or $\sigma_{l_{i2}}$ receive the "wrong spikes" before the instruction is activated, then the rule $(b^{2T})^+ \to b^q$, $q \geq 0$, can be applied, and neurons $\sigma_{l_{i1}}$ or $\sigma_{l_{i2}}$ will remove the "wrong spikes". There is also a possible situation that there still are "wrong spikes" until the instruction is activated. Therefore, when SUB module of active instruction starts running, neuron σ_{l_i} emits the spikes to neurons $\sigma_{l_{i1}}$ and $\sigma_{l_{i2}}$, and no rule in the two neurons can be used because of the "wrong spikes". If the "wrong spikes", new or not, cause that no rule can be applied in neurons $\sigma_{l_{i1}}$ or $\sigma_{l_{i2}}$, then the computation is blocked. Since the system works asynchronously, another "bad" situation must be considered. Specifically, before the new computation starts, there are still several "wrong spikes" that are not removed in last computation. In this case, the new computation will be blocked, too, when neuron σ_{l_i} sends the spikes by its rule. When the computation is blocked, no spike is sent out. No error results appear in this situation and the output is the computation result when the system finally reaches instruction l_h. Therefore, the system correctly simulates the SUB instruction of register machine M.

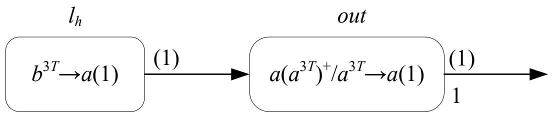


Figure 5. FIN module

3. FIN module, outputting the computation result.

Figure 5 shows this module. Suppose that the computation of M stops at some time, meaning that M receives the halting instruction. For system Π_1, this indicates neuron σ_{l_h} receives 3T spikes of type b. Thus, rule $b^{3T} \to a(1)$ in neuron σ_{l_h} can be applied at a later time, and then a spike of type a is sent to the output neuron by channel (1). After receiving the spike from neuron σ_{l_h}, neuron σ_{out} transmits a spike of type a to the environment constantly when it fires continually, until the moment no rule can be applied in the output neuron. The number of spikes sent to the environment is the results of the computation.

From the description of working modules in system Π_1, we know that the register machine M can be correctly simulated by system Π_1 with at most two kinds of spikes and at most three rules in each neuron. Therefore, the theorem holds. □

4.2 ASNP-MCS Systems as Number Accepting Devices

Theorem 4.2. $N_{acc}\text{ASNP-MCS}^3_* = \text{NRE}$.

Proof. Like the proof of Theorem 4.1, we construct an ASNP-MCS system Π_2 to simulate a deterministic register machine M, $M = (m, H, l_0, l_h, I)$. There are three parts in system Π_2: INPUT module, deterministic ADD module and SUB module. The INPUT module is used to input a spike train from the environment, shown in Figure 6. Spike train $a^{2T}(a^{3T})^n b^T$ can be read by neuron σ_{in} from the environment, and then $3Tn$ spikes of type a are stored in neuron σ_1, where the multiple of $3T$ spikes of type a is n, indicating that the accepted number is n. At some time, neuron σ_{in} reads $2T$ spikes of type a and several groups of $3T$ spikes of type a from the environment. Therefore, rule $a^{2T}(a^{3T})^+/a^{3T} \to a^{3T}(1)$ can be applied from reading the first $3T$ spikes of type a to receiving last ones. As a result, neuron σ_{in} sends $3T$ spikes of type a to neuron σ_1 via channel (1) once the rule is applied. When neuron σ_1 receives $3T$ spikes of type a from neuron σ_{in}, the number in register is added by 1. There are two cases before neuron σ_{in} reads T spikes of type b.

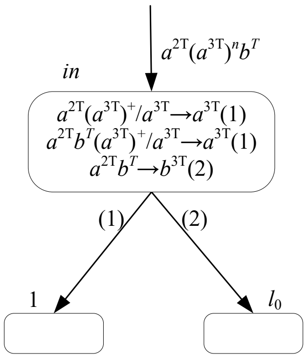

$a^{2T}(a^{3T})^n b^T$
in
$a^{2T}(a^{3T})^+/a^{3T} \to a^{3T}(1)$
$a^{2T}b^T(a^{3T})^+/a^{3T} \to a^{3T}(1)$
$a^{2T}b^T \to b^{3T}(2)$
(1)
(2)
1
l_0

Figure 6. INPUT module

Case 1: Though $3Tn$ spikes of type a have been read from the environment, there are still several groups of $3T$ spikes of type a in neuron σ_{in} because of the asynchronous mode. After T spikes of type b in spike train are received, rule $a^{2T}(a^{3T})^+b^T/a^{3T} \to a^{3T}(1)$ becomes available. Since neuron σ_{in} has read all spikes in the spike train, no new spike comes. Thus, the rule can be applied at a later time and neuron σ_{in} will send the spikes to neuron σ_1 every time as the rule is applied. When all $3Tn$ spikes of type a are stored in neuron σ_1, turn to Case 2.

Case 2: $3Tn$ spikes of type a are all stored in neuron σ_1. When T spikes of type b are read or when T spikes of type b have already been in neuron σ_1 (the situation that Case 1 turns to Case 2), there are only $2T$ spikes of type a and T spikes of type b left in neuron σ_{in}. Hence, rule $a^{2T}b^T \to b^{3T}(2)$ can be applied at some time, and $3T$ spikes of type b will be sent to neuron σ_{l_0} via channel (2).

Therefore, neuron σ_1 receives $3Tn$ spikes of type a, which means that the number stored in register 1 is n. Besides, since neuron σ_{l_0} receives 3T spikes of type b, the system starts to simulate the initial instruction l_0 of M.

We also can get something more. Neuron σ_1 will receive $3Tn$ spikes of type a and neuron σ_{l_0} will receive $3T$ spikes of type b after neuron σ_{in} reads the spike train, $a^{2T}(a^{3T})^+b^T$, from the environment, which means the result of the computation can be gotten normally without "block" situation.

When a register machine works in accepting mode, its ADD instruction of form $l_i : (\text{ADD}(r), l_j)$ is deterministic. The deterministic ADD module is shown in Figure 7. Since $3T$ spikes of type b are in neuron σ_{l_i}, then the neuron fires at a later time and transmits $3T$ spikes of type a to neurons $\sigma_{l_{i1}}$ and σ_r via channel (1). The number of spikes in neuron σ_r is added by $3T$, which indicates that the number of the corresponding register is added by 1. After receiving $3T$ spikes of type a from neuron σ_{l_i}, neuron $\sigma_{l_{i1}}$ applies the rule, at a later time, to converse $3T$ spikes of type a to $3T$ spikes of type b and sends the generated spikes to neuron σ_{l_j}. With $3T$ spikes of type b in neuron σ_{l_j}, the system starts to simulate the instruction l_j of M.

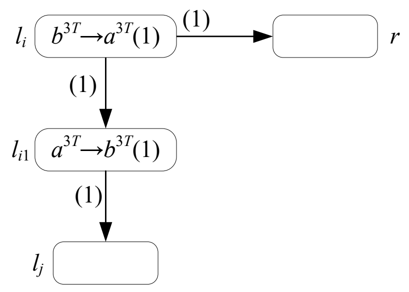
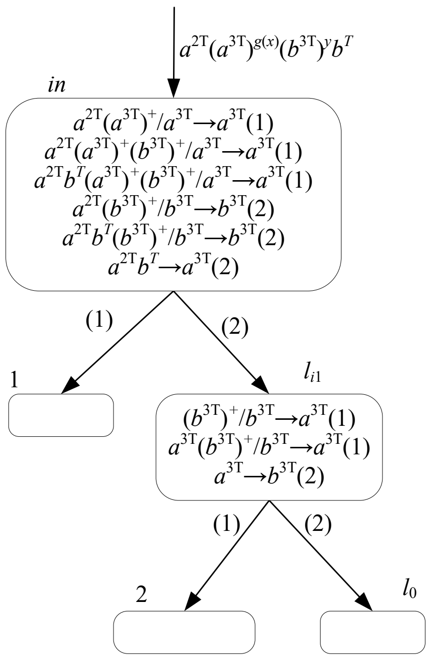


Figure 7. A deterministic ADD module

SUB module remains unchanged, shown in Figure 4. FIN module is removed, but the system remains neuron σ_{l_h}. Since neuron σ_{l_h} has $3T$ spikes of type b, the computation of M reaches instruction l_h and halts.

Based on the discussion above, an ASNP-MCS system with at most three rules in each neuron can correctly simulate the register machine working in accepting mode and no "block" happens in INPUT module. Therefore, the theorem holds. □

4.3 ASNP-MCS Systems as Small Universal Function Computing Devices



$a^{2T}(a^{3T})^{g(x)}(b^{3T})^{y}b^{T}$
in
$a^{2T}(a^{3T})^{+}/a^{3T}\to a^{3T}(1)$
$a^{2T}(a^{3T})^{+}(b^{3T})^{+}/a^{3T}\to a^{3T}(1)$
$a^{2T}b^{T}(a^{3T})^{+}(b^{3T})^{+}/a^{3T}\to a^{3T}(1)$
$a^{2T}(b^{3T})^{+}/b^{3T}\to b^{3T}(2)$
$a^{2T}b^{T}(b^{3T})^{+}/b^{3T}\to b^{3T}(2)$
$a^{2T}b^{T}\to a^{3T}(2)$
(1)
(2)
1
l_{i1}
$(b^{3T})^{+}/b^{3T}\to a^{3T}(1)$
$a^{3T}(b^{3T})^{+}/b^{3T}\to a^{3T}(1)$
$a^{3T}\to b^{3T}(2)$
(1)
(2)
2
l_0

Figure 8. INPUT module

In this section, we will investigate the Turing universality of ASNP-MCS systems as function computing devices. The universality of ASNP-MCS systems will be proved by simulating a small universal register machine M_u.

Theorem 4.3. There is a small universal ASNP-MCS system with 67 neurons for computing functions.

Proof. We construct an ASNP-MCS system Π_3, including INPUT module, ADD module, SUB module and FIN module, to simulate a small universal register machine M_u. The ADD module is the same with ADD module working in accepting mode, shown in Figure 7. The SUB module is the same with SUB module that works in accepting mode, shown in Figure 4. The FIN module is the same with FIN module working in accepting mode, shown in Figure 5. However, the INPUT module is different from INPUT module that works in accepting mode.

The INPUT module, shown in Figure 8, is used for reading a spike train $a^{2T}(a^{3T})^{g(x)}(b^{3T})^{y}b^{T}$ from the environment and introducing $3Tg(x)$ spikes of type a to neuron σ_1 and $3Ty$ spikes of type a to neuron σ_2. At some moment, neuron σ_{in} reads 2T spikes of type a and several groups of $3T$ spikes of type a from the environment. Thus, rule $a^{2T}(a^{3T})^{+}/a^{3T} \to a^{3T}(1)$ can be applied from first $3T$ spikes of type a to last ones coming. Notice that neuron σ_{in} is free to choose to fire or not once the rule is activated by the spikes in neuron σ_{in}. If neuron σ_{in} fires, then $3T$ spikes of type a are produced and the produced spikes are sent to neuron σ_1 by channel (1). If not, neuron σ_{in} goes on reading the next spike in the spike train. Before the first $3T$ spikes of type b come into neuron σ_{in}, indicating that $3Tg(x)$ spikes of type a already have been read, rule $a^{2T}(a^{3T})^{+}/a^{3T} \to a^{3T}(1)$ in neuron σ_{in} can be:

1. No execution, which means that $3Tg(x)$ spikes of type a are still remained in neuron σ_{in};
2. Partial execution, meaning that some groups of $3T$ spikes of type a are still in neuron σ_{in};
3. Total execution (no 3T spikes of type a in neuron σ_{in}). Hence, there are two cases before the first $3T$ spikes of type b are received: remaining groups of $3T$ spikes of type a or not.

The two cases are considered as follows.

Case 1: All the $3Tg(x)$ spikes of type a are stored in neuron σ_1 by the execution of the rule and there are only $2T$ spikes of type a left in neuron σ_{in}. At the period of time when neuron σ_{in} reads the following groups of $3T$ spikes of type b, rule $a^{2T}(b^{3T})^{+}/b^{3T} \to b^{3T}(2)$ is enabled. Thus, neuron σ_{in} fires, at some time, and then it sends $3T$ spikes of type b to neuron $\sigma_{l_{i1}}$. There are two situations when neuron σ_{in} has read all the spikes in the spike train. The two cases are as follows.

1. All the $3Ty$ spikes of type b are sent to neuron $\sigma_{l_{i1}}$ by the rule. There are 2T spikes of type a and T spikes of type b left in neuron σ_{in}. Therefore, rule $a^{2T}b^{T} \to a^{3T}(2)$ can be applied at a later time. Then, neuron σ_{in} transmits $3T$ spikes of type a to neuron $\sigma_{l_{i1}}$ via channel (2).
2. Several groups of $3T$ spikes of type b are remained in neuron σ_{in}. The spikes in neuron σ_{in} enable rule $a^{2T}b^{T}(b^{3T})^{+}/b^{3T} \to b^{3T}(2)$. Thus, the rule can be applied whenever the rule is available. After all the $3Ty$ spikes of type b are sent to neuron $\sigma_{l_{i1}}$, rule $a^{2T}b^{T} \to a^{3T}(2)$ in neuron σ_{in} can be enabled. Finally, neuron σ_{in} sends $3T$ spikes of type a to neuron σ_{li1} through channel (2).

Case 2: There are still some groups of 3T spikes of type a in neuron σ_{in}. After neuron σ_{in} reads the next groups of $3T$ spikes of type b, rule $a^{2T}(a^{3T})^{+}(b^{3T})^{+}/a^{3T} \to a^{3T}(1)$ can be applied. Then, neuron σ_{in} fires at some time, sending $3T$ spikes of type a to neuron σ_1 through channel (1). At the period of time from reading the first $3T$ spikes of type b to last ones coming, rule $a^{2T}(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ or rule $a^{2T}(b^{3T})^+/b^{3T} \to b^{3T}(2)$ can be applied according to the number of spikes in the neuron. If all $3Tg(x)$ spikes of type a are stored in neuron σ_1, rule $a^{2T}(b^{3T})^+/b^{3T} \to b^{3T}(2)$ can be enabled. If not, rule $a^{2T}(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ can be applied. There are also three situations after T spikes of type b are read. The three cases are as follows.

1. If all the $3Tg(x)$ spikes of type a are stored in neuron σ_1 and $3Ty$ spikes of type b are sent to neuron $\sigma_{l_{i1}}$, then turn to Case 1 (1).
2. If all the $3Tg(x)$ spikes of type a are stored in neuron σ_1 and $3Ty$ spikes of type b are not sent to neuron $\sigma_{l_{i1}}$, then turn to Case 1 (2).
3. If there are still several groups of 3T spikes of type a in neuron σ_{in}, then rule $a^{2T}b^T(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ can be used until no groups of $3T$ spikes of type a are left. Then, turn to Case 1 (2).

After all the $3Tg(x)$ spikes of type a are stored into neuron σ_1, $3Ty$ spikes of type b also will be sent to neuron $\sigma_{l_{i1}}$ sequentially. Thus, rule $(b^{3T})^+/b^{3T} \to a^{3T}(1)$ in neuron $\sigma_{l_{i1}}$ can be applied between the period of time that several groups of $3T$ spikes of type b are received. Neuron $\sigma_{l_{i1}}$ fires at some time, consuming $3T$ spikes of type b, producing $3T$ spikes of type a and transmitting the produced spikes to neuron σ_2. Notice that neuron $\sigma_{l_{i1}}$ is free to fire or not. There are two cases according to the number of spikes in neuron σ_{li1} after $3T$ spikes of type a are received by the neuron.

1. All the $3Ty$ spikes of type b in neuron $\sigma_{l_{i1}}$ are processed by rule $(b^{3T})^+/b^{3T} \to a^{3T}(1)$, which means that neuron σ_2 received $3Ty$ spikes of type a (the corresponding number in register 2 is y). Then, rule $a^{3T} \to b^{3T}(2)$ can be applied, at a later time, and neuron $\sigma_{l_{i1}}$ sends $3T$ spikes of type b to neuron σ_{l_0} via channel (2).
2. There are still some groups of $3T$ spikes of type b in neuron $\sigma_{l_{i1}}$. Thus, rule $a^{3T}(b^{3T})^+/b^{3T} \to a^{3T}(1)$ can be used at a later time. Turn to (1) unless all the $3Ty$ spikes of type b are changed as $3Ty$ spikes of type a and stored in neuron σ_2.

When neuron σ_{l0} receives 3T spikes of type b, the system starts to simulate initial instruction σ_{l_0} of M.

In conclusion, the spikes in neuron σ_{in} are processed as follows. When there are some groups of $3T$ spikes of type a in neuron σ_{in}, rule $a^{2T}(a^{3T})^+/a^{3T} \to a^{3T}(1)$ or rule $a^{2T}(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ or rule $a^{2T}b^T(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ can be used first in order to store all the $3Tg(x)$ spikes of type a in neuron σ_1. Then all the $3Ty$ spikes of type b in neuron σ_{in} are passing into neuron $\sigma_{l_{i1}}$ by rule $a^{2T}(b^{3T})^+/b^{3T} \to b^{3T}(2)$ or rule $a^{2T}b^T(b^{3T})^+/b^{3T} \to b^{3T}(2)$. When all this is done, neuron σ_{in} finally sends $3T$ spikes of type a to neuron $\sigma_{l_{i1}}$. Besides, the spikes in neuron $\sigma_{l_{i1}}$ are processed as follows. $3Ty$ spikes of type a are sent to neuron σ_2 first by rule $(b^{3T})^+/b^{3T} \to a^{3T}(1)$ or rule $a^{3T}(b^{3T})^+/b^{3T} \to a^{3T}(1)$ in neuron $\sigma_{l_{i1}}$. When there are only $3T$ spikes of type a left in neuron $\sigma_{l_{i1}}$, rule $a^{3T} \rightarrow b^{3T}(2)$ can be applied and the neuron emits $3T$ spikes of type b to neuron σ_{l_0}.

All in all, we can find that $3Tg(x)$ spikes of type a will be stored in neuron σ_1 and $3Ty$ spikes of type a will be stored in neuron σ_2 after neuron σ_{in} reads the spike train $a^{2T}(a^{3T})^{g(x)}(b^{3T})^y b^T$ from the environment. And finally, neuron σ_{l_0} will receive 3T spikes of type b. This means there is no "block" situation in this INPUT module.

A total of 67 neurons are used in this system: 8 neurons for 8 registers; 23 neurons for 23 instruction labels; 1 auxiliary neuron for each ADD module, 9 in total; 2 auxiliary neurons for each SUB module, 26 in total; INPUT module uses 1 auxiliary neuron.

From the description above, we know that ASNP-MCS systems with 67 neurons can correctly simulate register machine M_u working in computing mode. Therefore, Theorem 4.3 holds. □

5 CONCLUSIONS AND FUTURE WORK

In this work, we investigated a variant of SNP systems working in the asynchronous mode, asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems for short). Different from regular SNP systems, ASNP-MCS systems work in the asynchronous mode: neurons are free to fire when their rules can be applied; if the coming of new spikes causes the rules to be unavailable, the computation will continue in the new configuration. Different from the existing asynchronous SNP systems, ASNP-MCS systems have two interesting characters: multiple channels and multiple symbols. We proved that ASNP-MCS systems as number generating and accepting devices are Turing universal. Then, we constructed an ASNP-MCS system with 67 neurons as a small universal function computing device.

Several open problems still need to be discussed. For example, how to avoid the "block" situation in asynchronous systems. In the existing work, Song et al. [41] provided a scheme for solving this problem with the mode of "synchronization". This paper has made some attempt to deal with this "block" problem. The features of multiple channels and multiple symbols can give some help, in accepting/computing mode, to solve the "block" situation in INPUT module. Some issues remain to investigate: Can the "block" be solved in SUB module? Is there a small universal ASNP-MCS system with less neurons?

From the perspective of application, ASNP-MCS systems are the distributed and parallel computation systems working in the asynchronous mode and have the feature of multiple channels and the power of dealing with multiple symbols. In this work, only the discussion of Turing universality of the systems was in our focus. In the future, we will consider the application of ASNP-MCS systems in complex problems, for example, high-order dynamic systems and social network in real-world.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 62076206), and the Research Foundation of the Education Department of Sichuan Province (No. 17TD0034), China.

REFERENCES

[1] Păun, G.: Computing with Membranes. Journal of Computer and System Sciences, Vol. 61, 2000, No. 1, pp. 108–143, doi: 10.1006/jcss.1999.1693.

[2] Păun, G.—Rozenberg, G.—Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, New York, 2010.

[3] Freund, R.—Păun, G.—Pérez-Jiménez, M. J.: Tissue P Systems with Channel States. Theoretical Computer Science, Vol. 330, 2005, No. 1, pp. 101–116, doi: 10.1016/j.tcs.2004.09.013.

[4] Bernardini, F.—Gheorghe, M.: Population P Systems. Journal of Universal Computer Science, Vol. 10, 2004, No. 5, pp. 509–539, doi: 10.3217/jucs-010-05-0509.

[5] Cienciala, L.—Ciencialová, L.: Some New Results of P Colonies with Bounded Parameters. Natural Computing, Vol. 17, 2018, No. 2, pp. 321–332, doi: 10.1007/s11047-016-9591-0.

[6] Ionescu, M.—Păun, G.—Yokomori, T.: Spiking Neural P Systems. Fundamenta Informaticae, Vol. 71, 2006, No. 2, pp. 279–308.

[7] Ciencialová, L.—Csuhaj-Varjú, E.—Kelemenová, A.—Vaszil, G.: Variants of P Colonies with Very Simple Cell Structure. International Journal of Computers Communications and Control, Vol. 4, 2009, No. 3, pp. 224–233, doi: 10.15837/ijccc.2009.3.2430.

[8] Păun, G.—Pérez-Jiménez, M. J.: Solving Problems in a Distributed Way in Membrane Computing: dP Systems. International Journal of Computers Communications and Control, Vol. 5, 2010, No. 2, pp. 238–250, doi: 10.15837/ijccc.2010.2.2478.

[9] Song, B.—Pan, L.: The Computational Power of Tissue-Like P Systems with Promoters. Theoretical Computer Science, Vol. 641, 2016, pp. 43–52, doi: 10.1016/j.tcs.2016.05.022.

[10] Valencia-Cabrera, L.—Orellana-Martín, D.—Martínez-Del-Amor, M. A.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Computational Efficiency of Minimal Cooperation and Distribution in Polarizationless P Systems with Active Membranes. Fundamenta Informaticae, Vol. 153, 2017, No. 1-2, pp. 147–172, doi: 10.3233/fi-2017-1535.

[11] Zhang, X.—Liu, Y.—Luo, B.—Pan, L.: Computational Power of Tissue P Systems for Generating Control Languages. Information Sciences, Vol. 278, 2014, No. 10, pp. 285–297, doi: 10.1016/j.ins.2014.03.053.

[12] García-Quismondo, M.—Nisbet, I. C. T.—Mostello, C.—Reed, M. J.: Modeling Population Dynamics of Roseate Terns (Sterna dougallii) in the Northwest Atlantic Ocean. Ecological Modelling, Vol. 368, 2018, pp. 298–311, doi: 10.1016/j.ecolmodel.2017.12.007.

[13] Gheorghe, M.—Manca, V.—Romero-Campero, F. J.: Deterministic and Stochastic P Systems for Modelling Cellular Processes. Natural Computing, Vol. 9, 2010, No. 2, pp. 457–473, doi: 10.1007/s11047-009-9158-4.

[14] Zhang, G.—Rong, H.—Neri, F.—Pérez-Jiménez, M. J.: An Optimization Spiking Neural P System for Approximately Solving Combinatorial Optimization Problems. International Journal of Neural Systems, Vol. 24, 2014, No. 5, Art. No. 1440006, pp. 1–16, doi: 10.1142/s0129065714400061.

[15] Peng, H.—Shi, P.—Wang, J.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Multiobjective Fuzzy Clustering Approach Based on Tissue-Like Membrane Systems. Knowledge-Based Systems, Vol. 125, 2017, pp. 74–82, doi: 10.1016/j.knosys.2017.03.024.

[16] Peng, H.—Wang, J.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: An Unsupervised Learning Algorithm for Membrane Computing. Information Sciences, Vol. 304, 2015, No. 20, pp. 80–91, doi: 10.1016/j.ins.2015.01.019.

[17] Peng, H.—Wang, J.—Shi, P.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: An Extended Membrane System with Active Membranes to Solve Automatic Fuzzy Clustering Problems. International Journal of Neural Systems, Vol. 26, 2016, No. 3, Art. No. 1650004, pp. 1–17, doi: 10.1142/s0129065716500040.

[18] Díaz-Pernil, D.—Berciano, A.—Peña-Cantillana, F.—Gutiérrez-Naranjo, M. A.: Segmenting Images with Gradient-Based Edge Detection Using Membrane Computing. Pattern Recognition Letters, Vol. 34, 2013, No. 8, pp. 846–855, doi: 10.1016/j.patrec.2012.10.014.

[19] Díaz-Pernil, D.—Gutiérrez-Naranjo, M. A.—Peng, H.: Membrane Computing and Image Processing: A Short Survey. Journal of Membrane Computing, Vol. 1, 2019, pp. 58–73, doi: 10.1007/s41965-018-00002-x.

[20] Wang, J.—Shi, P.—Peng, H.: Membrane Computing Model for IIR Filter Design. Information Sciences, Vol. 329, 2016, pp. 164–176, doi: 10.1016/j.ins.2015.09.011.

[21] Li, B.—Peng, H.—Wang, J.—Huang, X.: Multi-Focus Image Fusion Based on Dynamic Threshold Neural P Systems and Surfacelet Transform. Knowledge-Based Systems, Vol. 196, 2020, Art. No. 105794, pp. 1–12, doi: 10.1016/j.knosys.2020.105794.

[22] Li, B.—Peng, H.—Luo, X.—Wang, J.—Song, X.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Medical Image Fusion Method Based on Coupled Neural P Systems in Nonsubsampled Shearlet Transform Domain. International Journal of Neural Systems, Vol. 31, 2021, No. 1, Art. No. 2050050, doi: 10.1142/S0129065720500501.

[23] Li, B.—Peng, H.—Wang, J.: A Novel Fusion Method Based on Dynamic Threshold Neural P Systems and Nonsubsampled Contourlet Transform for Multi-Modality Medical Images. Signal Processing, Vol. 178, 2021, Art. No. 107793, pp. 1–13, doi: 10.1016/j.sigpro.2020.107793.

[24] Pan, L.—Wang, J.—Hoogeboom, H. J.: Spiking Neural P Systems with Astrocytes. Neural Computation, Vol. 24, 2012, No. 3, pp. 805–825, doi: 10.1162/neco_a_00238.

[25] Păun, G.: Spiking Neural P Systems with Astrocyte-Like Control. Journal of Universal Computer Science, Vol. 13, 2007, No. 11, pp. 1707–1721.

[26] Pan, L.—Păun, G.: Spiking Neural P Systems with Anti-Spikes. International Journal of Computers Communications and Control, Vol. 4, 2009, No. 3, pp. 273–282, doi: 10.15837/ijccc.2009.3.2435.

[27] Wang, J.—Hoogeboom, H. J.—Pan, L.—Păun, G.—Pérez-Jiménez, M. J.: Spiking Neural P Systems with Weights. Neural Computation, Vol. 22, 2010, No. 10, pp. 2615–2646, doi: 10.1162/neco_a_00022.

[28] Zeng, X.—Zhang, X.—Song, T.—Pan, L.: Spiking Neural P Systems with Thresholds. Neural Computation, Vol. 26, 2014, No. 7, pp. 1340–1361, doi: 10.1162/neco_a_00605.

[29] Peng, H.—Chen, R.—Wang, J.—Song, X.—Wang, T.—Yang, F.—Sun, Z.: Competitive Spiking Neural P Systems with Rules on Synapses. IEEE Transactions on NanoBioscience, Vol. 16, 2017, No. 8, pp. 888–895, doi: 10.1109/tnb.2017.2783890.

[30] Song, T.—Pan, L.—Păun, G.: Spiking Neural P Systems with Rules on Synapses. Theoretical Computer Science, Vol. 529, 2014, pp. 82–95, doi: 10.1016/j.tcs.2014.01.001.

[31] Song, T.—Pan, L.: Spiking Neural P Systems with Rules on Synapses Working in Maximum Spiking Strategy. IEEE Transactions on NanoBioscience, Vol. 14, 2015, No. 4, pp. 465–477, doi: 10.1109/TNB.2015.2402311.

[32] Peng, H.—Yang, J.—Wang, J.—Wang, T.—Sun, Z.—Song, X.—Lou, X.—Huang, X.: Spiking Neural P Systems with Multiple Channels. Neural Networks, Vol. 95, 2017, pp. 66–71, doi: 10.1016/j.neunet.2017.08.003.

[33] Song, X.—Wang, J.—Peng, H.—Ning, G.—Sun, Z.—Wang, T.—Yang, F.: Spiking Neural P Systems with Multiple Channels and Anti-Spikes. Biosystems, Vol. 167–170, 2018, pp. 13–19, doi: 10.1016/j.biosystems.2018.05.004.

[34] Peng, H.—Wang, J.: Coupled Neural P Systems. IEEE Transactions on Neural Networks and Learning Systems, Vol. 30, 2019, No. 6, pp. 1672–1682, doi: 10.1109/TNNLS.2018.2872999.

[35] Peng, H.—Wang, J.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Dynamic Threshold Neural P Systems. Knowledge-Based Systems, Vol. 163, 2019, pp. 875–884, doi: 10.1016/j.knosys.2018.10.016.

[36] Wu, T.—Păun, A.—Zhang, Z.— Pan, L.: Spiking Neural P Systems with Polarizations. IEEE Transactions on Neural Networks and Learning Systems, Vol. 29, 2018, No. 8, pp. 3349–3360, doi: 10.1109/TNNLS.2017.2726119.

[37] Peng, H.—Li, B.—Wang, J.—Song, X.—Wang, T.—Valencia-Cabrera, L.—Pérez-Hurtado, I.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Spiking Neural P Systems with Inhibitory Rules. Knowledge-Based Systems, Vol. 188, 2020, Art. No. 105064, pp. 1–10, doi: 10.1016/j.knosys.2019.105064.

[38] Peng, H.—Bao, T.—Luo, X.—Wang, J.—Song, X.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Dendrite P Systems. Neural Networks, Vol. 127, 2020, pp. 110–120, doi: 10.1016/j.neunet.2020.04.014.

[39] Peng, H.—Lv, Z.—Li, B.—Luo, X.—Wang, J.—Song, X.—Wang, T.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Nonlinear Spiking Neural P Systems. International Journal of Neural Systems, Vol. 30, 2010, No. 10, Art. No. 2050008, pp. 1–17, doi: 10.1142/S0129065720500082.

[40] Cavaliere, M.—Ibarra, O. H.—Păun, G.—Egecioglu, O.—Ionescu, M.—Woodworth, S.: Asynchronous Spiking Neural P Systems. Theoretical Computer Science, Vol. 410, 2009, No. 24-25, pp. 2352–2364, doi: 10.1016/j.tcs.2009.02.031.

[41] Song, T.—Pan, L.—Păun, G.: Asynchronous Spiking Neural P Systems with Local Synchronization. Information Sciences, Vol. 219, 2013, No. 10, pp. 197–207, doi: 10.1016/j.ins.2012.07.023.

[42] Song, T.—Pan, L.—Jiang, K.—Song, B.—Chen, W.: Normal Forms for Some Classes of Sequential Spiking Neural P Systems. IEEE Transactions on NanoBioscience, Vol. 12, 2013, No. 3, pp. 255–264, doi: 10.1109/tnb.2013.2271278.

[43] Pan, L.—Păun, G.—Zhang, G.—Neri, F.: Spiking Neural P Systems with Communication on Request. International Journal of Neural Systems, Vol. 27, 2017, No. 8, Art. No. 1750042, pp. 1–13, doi: 10.1142/s0129065717500423.

[44] Zhang, X.—Pan, L.—Păun, A.: On The Universality of Axon P Systems. IEEE Transactions on Neural Networks and Learning Systems, Vol. 26, 2015, No. 11, pp. 2816–2829, doi: 10.1109/tnnls.2015.2396940.

[45] Peng, H.—Wang, J.—Pérez-Jiménez, M. J.—Wang, H.—Shao, J.—Wang, T.: Fuzzy Reasoning Spiking Neural P System for Fault Diagnosis. Information Sciences, Vol. 235, 2013, No. 20, pp. 106–116, doi: 10.1016/j.ins.2012.07.015.

[46] Wang, J.—Shi, P.—Peng, H.—Pérez-Jiménez, M. J.—Wang, T.: Weighted Fuzzy Spiking Neural P Systems. IEEE Transactions on Fuzzy Systems, Vol. 21, 2013, No. 2, pp. 209–220, doi: 10.1109/tfuzz.2012.2208974.

[47] Peng, H.—Wang, J.—Ming, J.—Shi, P.—Pérez-Jiménez, M. J.—Yu, W.—Tao, C.: Fault Diagnosis of Power Systems Using Intuitionistic Fuzzy Spiking Neural P Systems. IEEE Transactions on Smart Grid, Vol. 9, 2018, No. 5, pp. 4777–4784, doi: 10.1109/tsg.2017.2670602.

[48] Peng, H.—Wang, J.—Shi, P.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Fault Diagnosis of Power Systems Using Fuzzy Tissue-Like P Systems. Integrated Computer-Aided Engineering, Vol. 24, 2017, No. 4, pp. 401–411, doi: 10.3233/ica-170552.

[49] Wang, T.—Zhang, G.—Zhao, J.—He, Z.—Wang, J.—Pérez-Jiménez, M. J.: Fault Diagnosis of Electric Power Systems Based on Fuzzy Reasoning Spiking Neural P Systems. IEEE Transactions on Power Systems, Vol. 30, 2015, No. 3, pp. 1182–1194, doi: 10.1109/TPWRS.2014.2347699.

[50] Wang, J.—Peng, H.—Yu, W.—Ming, J.—Pérez-Jiménez, M. J.—Tao, C.—Huang, X.: Interval-Valued Fuzzy Spiking Neural P Systems for Fault Diagnosis of Power Transmission Networks. Engineering Applications of Artificial Intelligence, Vol. 82, 2019, pp. 102–109, doi: 10.1016/j.engappai.2019.03.014.

[51] Macías, L. F.—Pérez-Hurtado, I.—García-Quismondo, M.—Valencia-Cabrera, L.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: A P-Lingua Based Simulator for Spiking Neural P Systems. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (Eds.): Membrane Computing (CMC 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7184, 2012, pp. 257–281, doi: 10.1007/978-3-642-28024-5_18.

[52] Cabarle, F. G. C.—Adorna, H.—Martínez, M. A.: A Spiking Neural P System Simulator Based on CUDA. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (Eds.): Membrane Computing (CMC 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7184, 2012, pp. 87–103, doi: 10.1007/978-3-642-28024-5_8.

[53] Song, T.—Zou, Q.—Liu, X.—Zeng, X.: Asynchronous Spiking Neural P Systems with Rules on Synapses. Neurocomputing, Vol. 151, 2015, Part 3, pp. 1439–1445, doi: 10.1016/j.neucom.2014.10.044.

[54] Cabarle, F. G. C.—Adorna, H. N.—Pérez-Jiménez, M. J.—Song, T.: Spiking Neural P Systems with Structural Plasticity. Neural Computing and Applications, Vol. 26, 2015, No. 8, pp. 1905–1917, doi: 10.1007/s00521-015-1857-4.

[55] Song, T.—Liu, X.—Zeng, X.: Asynchronous Spiking Neural P Systems with Anti-Spikes. Neural Processing Letters, Vol. 42, 2015, No. 3, pp. 633–647, doi: 10.1007/s11063-014-9378-1.

[56] Cavaliere, M.—Egecioglu, O.—Ibarra, O. H.—Ionescu, M.—Păun, G.—Woodworth, S.: Asynchronous Spiking Neural P Systems: Decidability and Undecidability. In: Garzon, M. H., Yan, H. (Eds.): DNA Computing (DNA 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4848, 2007, pp. 246–255, doi: 10.1007/978-3-540-77962-9_26.

[57] Zhang, X.—Zeng, X.—Pan, L.: On Languages Generated by Asynchronous Spiking Neural P Systems. Theoretical Computer Science, Vol. 410, 2009, No. 26, pp. 2478–2488, doi: 10.1016/j.tcs.2008.12.055.

[58] Korec, I.: Small Universal Register Machines. Theoretical Computer Science, Vol. 168, 1996, No. 2, pp. 267–301, doi: 10.1016/s0304-3975(96)00080-1.

Wenmei Yi received her B.Sc. degree in computer science from Sichuan Police College, Luzhou, China in 2016. She is currently pursuing her M.Sc. degree with School of Computer and Software Engineering, Xihua University, Chengdu, China. Her current research interests include membrane computing and machine learning.

Zeqiong Lv received her B.Sc. degree in computer science from Xihua University, Chengdu, China, in 2018. She is currently pursuing her M.Sc. degree with School of Computer and Software Engineering, Xihua University, Chengdu, China. Her current research interests include membrane computing and machine learning.

Hong Peng received his Ph.D. degree in signal and information processing from the University of Electronic Science and Technology of China, Chengdu, China in 2010. He has been Professor with School of Computer and Software Engineering since 2005. His research interests include membrane computing, machine learning, image processing.

Xiaoxiao Song received his Ph.D. degree in electrical engineering from the Chongqing University, Chongqing, China in 2010. He is currently Associate Professor with School of Electrical and Information Engineering, Xihua University, China. His research interests include membrane computing and image processing.

Jun Wang received her Ph.D. degree in electrical engineering from the Southwest Jiaotong University, Chengdu, China in 2006. She was Lecturer with the Sichuan College of Science and Technology, China, from 1991 to 2003. She was Associate Professor with the Xihua University, China, from 1998 to 2003. Since 2004 she has been Professor with the School of Electrical and Information Engineering. Her research interests include electrical automation, intelligent control, and membrane computing.