ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **20**, Number 1, 2017, 32–41

Spiking Neural P Systems with Anti-Spikes and without Annihilating Priority

Xun Wang[1,2], Tao Song[1,3,*], Pan Zheng[3], Shaohua Hao[1], Tongmao Ma[1]

[1] College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, Shandong, China

[2] Center for Bio-engineering and Bio-technology, China University of Petroleum, Qingdao 266580, Shandong, China

[3] Faculty of Engineering, Computing and Science, Swinburne University of Technology Sarawak Campus, Kuching 93350, Malaysia

Abstract. Spiking neural P systems with anti-spikes (shortly named ASN P systems) are a class of distributed and parallel neural-like computing systems. Besides spikes, neurons in ASN P systems can also contain a number of anti-spikes. Whenever spikes and anti-spikes meet in a neuron, they annihilate each other immediately in a maximal manner, that is, the annihilation has priority over neuron's spiking. In this work, we introduce a variant of ASN P systems, named ASN P systems without annihilating priority. In such systems, when a neuron has both a number of spikes and anti-spikes, the annihilation between spikes and anti-spikes is not obligatory and the neuron can choose non-deterministically spiking or annihilating. The computational power of ASN P systems without annihilating priority as number generators is investigated. As a result, it is obtained that such system with at most two rules per neuron can achieve Turing completeness as number generators. This result gives an answer to an open problem formulated in [INT J COMPUT COMMUN & CONTROL, 3, 273–282, 2009]. As well, the obtained result is optimal in the sense of having a minimal number of rules in neurons of Turing universal ASN P systems.

Key-words: Membrane computing, Spiking neural P system, Turing completeness, Register machine, Anti-spike

1. Introduction

Neurons are one of the most interesting cell-types in the human body. A large number of neurons working in a cooperative manner are able to perform tasks (such as thought, self-awareness, intuition) that are not yet matched by the tools we can build with our current technology. However, it is believed that the distributed manner in which the brain processes information is important in obtaining better performance for electronic computers, that is why we are interested in neural-like computing models.

In 2006, a class of distributed and parallel neural-like computing systems, named spiking neural P systems (SN P systems for short), were proposed in [1]. With terms of motivation, SN P systems fall into the third generation of neural network models [2], and can be used as computing devices in three ways: generating (computing) sets of numbers [1], generating/accepting languages [3] and computing recursive functions [4]. Taking ideas from both mathematics and neurobiological phenomena, some operators and working manners were developed, thus obtaining plenty of variants of SN P systems, see e.g. [5, 6, 7]. Developing new variants of SN P systems was formulated as an attractive research branch in membrane computing [10]. Applications of SN P systems have also arisen widely research interests. For instances, SN P systems have been used to approximately solve combinatorial optimization problems [10], to design neural-like logic gates and circuits [8], to represent knowledge for fault diagnosis [9] and to identify nuclear export signals [11]. More details on SN P systems and their applications can be found in Handbook of Membrane Computing [12], the website of P systems [13], or the IMCS Bulletin [14].

SN P systems with anti-spikes (shortly called ASN P systems) were proposed in [15]. Besides spikes, anti-spikes are also considered in ASN P systems. The neuron in ASN P systems can contain both a number of spikes and anti-spikes. The spikes and anti-spikes will annihilate each other immediately in a maximal manner whenever they meet in certain neuron, thus any neuron can hold only a number of spikes or anti-spikes at any moment. The annihilation happens and completes instantaneously, so it has priority over the neuron's spiking. In the initial literature of ASN P systems, it was proved that ASN P systems can achieve Turing universality as number generating and accepting devices, as well as several problems for further research in ASN P systems were formulated and left open [15].

Till now, most of the open problems left have been solved. In particular, it is proved that one type of neuron is sufficient for ASN P systems with anti-spikes to achieve Turing universality [16, 17]; two categories of spiking rules are sufficient for ASN P systems achieving Turing universality [18]; small universal ASN P systems were obtained in [19]. But, it is still open that whether the computational power ASN P systems as number generators will be reduced or not when the annihilating priority is removed.

To solve the problem, in this work, we deal with ASN P systems without annihilating priority. In the systems, when a neuron contains both a number of spikes and anti-spikes at any moment, the annihilation between spikes and anti-spikes is not obligatory, that is, the neuron can choose non-deterministically spiking (if it has enabled spiking rule) or annihilating (by using annihilating rule). The computational power of ASN P systems without annihilating priority is investigated. As a result, it is obtained that such systems can generate/compute any set of Turing computable natural numbers, even with all the neurons working in a "flip-flop" way (changing spikes in anti-spikes and conversely) and containing a unique spiking rule except for one neuron.

The result gives an answer to a problem left open in ASN P systems. As well, the the result is optimal in the sense of having a minimal number of rules in neurons of Turing universal ASN P systems. The result may also provide some theoretical suggestions for the applications of ASN P systems.

2. ASN P Systems without Annihilating Priority

We start by recalling some basic prerequisites of formal language theory from [20]. For an alphabet Σ, it is denoted by Σ^* the set of all finite strings of symbols from Σ; the empty string is denoted by λ, and the set of all nonempty strings over Σ is denoted by Σ^+. When $\Sigma = \{a\}$ is a singleton, we simply write a^* and a^+ instead of $\{a\}^*, \{a\}^+$.

An ASN P system without annihilating priority of degree $m \geq 1$ is a construct:

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

- $O = \{a, \bar{a}\}$ is the alphabet, where a is called spike and $\bar{a}$ is called anti-spike;
- $\sigma_1, \sigma_2, \ldots, \sigma_m$ are neurons of the form $\sigma_i = (n_i, u_i, R_i)$ with $1 \leq i \leq m$, where
  1. $n_i, u_i \in \mathbb{N}$ are the numbers of spikes and anti-spikes initially placed in neuron σ_i, respectively;
  2. R_i is a finite set of rules of the following three forms:
     (a) spiking rule: $E/b^c \to b'$, where E is a regular expression over O, $b, b' \in \{a, \bar{a}\}$ and $c \geq 1$;
     (b) forgetting rule: $b^s \to \lambda, b \in \{a, \bar{a}\}$, for some $s \geq 1$, with the restriction that $b^s \notin L(E)$ for any rule $E/b^c \to b'$ from R_i;
     (c) annihilating rule: $a\bar{a} \to \lambda$;
- $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin syn$, is the set of synapses connecting each pair of neurons;
- $out \in \{1, 2, \ldots, m\}$ indicates the output neuron.

Rules of the form $E/b^c \to b'$ are spiking rules. There are four categories of spiking rules identified by $(b, b') \in \{(a, a), (a, \bar{a}), (\bar{a}, a), (\bar{a}, \bar{a})\}$. The application of a spiking rule is controlled by checking the numbers of spikes and anti-spikes against a regular expression associated with the rule. Specifically, if neuron σ_i contains k spikes and l anti-spikes such that $a^k\bar{a}^l \in L(E)$ and $k, l \geq c$, the rule $E/b^c \to b'$ can be applied. The application of the rule means c spikes or anti-spikes (not both of them) are consumed (thus the remaining content of neuron σ_i is $a^{(k-c)}\bar{a}^l$ or $a^k\bar{a}^{(l-c)}$), meanwhile sending one spike or anti-spike to neuron σ_j such that $(i, j) \in syn$. For two spiking rules, it may hold $L(E_1) \cap L(E_2) \neq \emptyset$, that is, the two spiking rules are both applicable at some moment, but only one of them is non-deterministically chosen to use. For any spiking rule $E/b^c \to b'$, if it has $L(E) = b^c$, the rule is simply written as $b^c \to b'$, and called pure.

The forgetting rule is of the form $b^s \to \lambda, b \in \{a, \bar{a}\}$ with $s \geq 1$. If neuron σ_i contains exactly s spikes or anti-spikes, then forgetting rule $b^s \to \lambda, b \in \{a, \bar{a}\}$ from R_i can be applied, removing the s spikes or anti-spikes. For any regular expression E associated with a spiking rule from R_i, it satisfies that $b^s \notin L(E)$. It means that if a spiking rule is applicable at certain moment, no forgetting rule is applicable, and vice versa.

It is denoted by $a\bar{a} \to \lambda$ the annihilating rule. The application of annihilating rule has no priority over spiking and forgetting rules. When a neuron contains both a number of spikes and anti-spikes, the annihilation between spikes and anti-spikes is not obligatory. The neuron can choose non-deterministically to use one of the enabled spiking rules or the annihilating rule, If there is no enabled spiking rule, the neuron must use the annihilating rule. The annihilation takes one time unit. (A global clock is assumed to mark the time of the whole system, and in each time unit, for each neuron with applicable rules one of these rules is non-deterministically chosen, which are then applied on the tick of the clock, for all neurons at the same time.) When the annihilation completes, the neuron ends with only a number of spikes or anti-spikes, but not both of them. Specifically, at some moment, if a neuron contains k spikes and l anti-spikes, it can use the annihilating rule in a maximal manner (ending with $(k-l)$ spikes or $(l-k)$ anti-spikes) or one spiking rule (non-deterministically choosing from the enabled ones). The work of each neuron is sequential: only one rule can be applied in each time unit, but the neurons in the system work in a parallel manner.

A neuron with only one spiking rule is said to be simple. An SN P system is said to be simple if all neurons in the system are simple. An SN P system is said to be almost simple if all neurons in the system are simple except for one neuron.

The configuration of the system at any moment is described by both the number of spikes and the number of anti-spikes in each neuron, hence it is of the form

$$\langle (c_1, d_1), (c_2, d_2), \ldots, (c_m, d_m) \rangle$$

with $c_i, d_i \in \mathbb{N}$. With this notation, the initial configuration can be easily obtained

$$\langle (n_1, u_1), (n_2, u_2), \ldots, (n_m, u_m) \rangle.$$

By using the rules defined above, one can define transitions among configurations. A series of transitions starting from the initial configuration is called a computation. The computation halts if the system reaches a configuration where no rules is enabled in any neuron. With any computation halting or not, a spike train is associated. The computation result is the distance between the first two spikes emitted to the environment by the output neuron.

The set of all numbers generated by system Π is denoted by $N_2(\Pi)$. It is denoted by $N_2ASNP_{nopri}(cate_l, rule_k, forg)$ the family of sets of numbers generated/computed by ASN P systems without annihilating priority, where at most l categories of spiking rules are used among the four possible categories, there are at most k spiking rules in any neuron and forgetting rules are used. If the forgetting rules are not used, indication $forg$ will be removed from the notation.

3. Universality Result

In this section, it is proved that ASN P systems without annihilating priority are Turing universal as number generators, that is, they can generate compute any set of Turing computable natural numbers. The proof is obtained by simulating the computation of the register machine, which is known to characterize the family NRE – Turing computable sets of natural numbers [20].

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 initial instruction (an ADD instruction), l_h is the halt label of instruction HALT; I is a set of instructions of the following forms:

- $l_i : (\mathtt{ADD}(r), l_j, l_k)$ (add 1 to register r and non-deterministically go to instruction l_j or l_k);
- $l_i : (\mathtt{SUB}(r), l_j, l_k)$ (if register r is non-empty, then subtract 1 and go to the instruction l_j, otherwise go to the instruction l_k);
- l_h : $\mathtt{HALT}$ (the halt instruction).

Theorem 0.1 $NRE = N_2ASNP_{nopri}(cate_2, rule_2)$.

Proof It is sufficient to prove the inclusion $NRE \subseteq N_2ASNP_{nopri}(cate_2, rule_2)$, since the converse inclusion is straightforward (we can invoke for it the Turing-Church thesis).

In the following proof, an ASN P system without annihilating priority Π is constructed to simulate the computation of register machine M working in the generative mode. In particular, each register r in register machine M is associated with a neuron σ_r in system Π with $r = 1, 2, \ldots, m$, and each instruction l_i of register machine M is associated with a neuron σ_{l_i}. If the number in register r ($1 < r \leq m$) is n, there are n spikes in neuron σ_r; if the number in register 1 is n, there are $n+1$ spikes in neuron σ_1. Without losing generality, it is assumed that during the computation of register machine M the number in register 1 is never decreased, so the number of spikes in neuron σ_1 cannot be decreased before the simulation reaches the halting instruction l_h. Moreover, at the beginning, neuron σ_{l_0} has one spike and becomes active to simulate register machine M starting to work by operating the initial instruction l_0.

Module ADD indicated in Figure 1 – simulating ADD instruction $l_i : (\mathtt{ADD}\ (r), l_j, l_k)$.

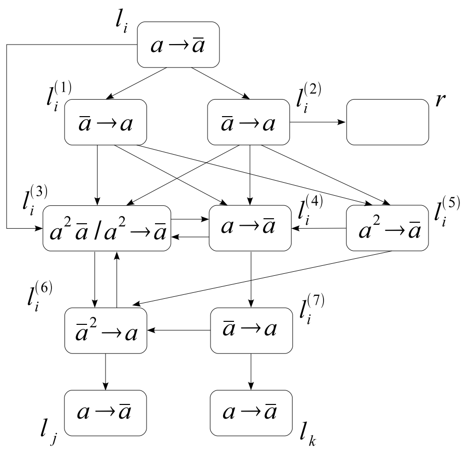

Figure 1: The ADD module of Π

Assumed that at certain step t, neuron σ_{l_i} receives one spike and becomes active. With the spike, it fires by using rule $a \to \bar{a}$, sending one anti-spike to each of neurons $\sigma_{l_i^{(1)}}, \sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$. Having one anti-spike inside, neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ fire at step $t+1$, emitting two spikes to neurons $\sigma_{l_i^{(3)}}, \sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$, respectively. Meanwhile, one spike is emitted to neuron σ_r from neuron $\sigma_{l_i^{(2)}}$ such that the number of spikes in neuron σ_r is increased by one, simulating the operation of adding one on register r of register machine M. Having two spikes and one anti-spike inside, neuron $\sigma_{l_i^{(3)}}$ is able to use both spiking rule $a^2\bar{a}/a^2 \rightarrow \bar{a}$ and annihilating rule $a\bar{a} \rightarrow \lambda$, but only one of them is non-deterministically chosen to use.

– If neuron $\sigma_{l_i^{(3)}}$ chooses firing by using spiking rule $a^2\bar{a}/a^2 \rightarrow \bar{a}$ at step $t+2$, it sends one anti-spike to each of neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(6)}}$. At the same moment, neuron $\sigma_{l_i^{(5)}}$ fires to send one anti-spike to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(6)}}$. In neuron $\sigma_{l_i^{(4)}}$, there are two spikes and two anti-spikes. Since there is no spiking rule can be used, neuron $\sigma_{l_i^{(4)}}$ can only do the annihilation, ending with neither spike nor anti-spike inside. Having two anti-spikes inside, neuron $\sigma_{l_i^{(6)}}$ becomes active at step $t+3$, sending one spike to neuron σ_{l_j}, indicating that system Π passes to simulate instruction l_j of register machine M.

– If neuron $\sigma_{l_i^{(3)}}$ chooses to use annihilating rule $a\bar{a} \rightarrow \lambda$ at step $t+2$, one step later, it has one spike inside and keeps inactive. At that moment, neuron $\sigma_{l_i^{(3)}}$ fires with sending one anti-spike to each of neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(6)}}$. With one anti-spike inside, neuron $\sigma_{l_i^{(6)}}$ keeps inactive, and the anti-spike will be annihilated by the spike from neuron $\sigma_{l_i^{(7)}}$ at step $t+4$. After doing the annihilation, neuron $\sigma_{l_i^{(4)}}$ has one spike and fires to send one anti-spike to each of neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(7)}}$. By using the annihilating rule, neuron $\sigma_{l_i^{(3)}}$ becomes empty (returns to no spike or anti-spike inside as in the initial configuration) Neuron $\sigma_{l_i^{(7)}}$ fires at step $t+4$, emitting a spike to neuron σ_{l_k}, which indicates that system Π starts to simulate instruction l_k of register machine M.

Therefore, from firing neuron σ_{l_i}, system Π adds one spike to neuron σ_r and non-deterministically fires one of neurons σ_{l_j} and σ_{l_k}, which correctly simulates the ADD instruction $l_i : (\mathtt{ADD}(r), l_j, l_k)$.

Module SUB shown in Figure 2 – simulating SUB instruction $l_i : (\mathtt{SUB}\ (r), l_j, l_k)$.

Assume that at certain step t, system Π has to simulate a SUB instruction l_i of register machine M. We have the following two cases.

– If neuron σ_r has n ($n > 0$) spikes (corresponding to the fact that the number stored in register r is n, and $n > 0$), only the annihilating rule can be used for no spiking rule is enabled. One step later, the number of spikes in neuron σ_r is decreased by one, which simulates subtracting one from register r of register machine M. Meanwhile, two spikes are sent to each of neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(4)}}$. In the next step, neuron $\sigma_{l_i^{(3)}}$ fires with sending one anti-spike to neuron $\sigma_{l_i^{(6)}}$. The two spikes in neuron $\sigma_{l_i^{(4)}}$ will be annihilated by two anti-spikes arriving at steps $t+2$ and $t+4$ from neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(8)}}$, respectively. At step $t+6$, neuron σ_{l_j} receives one spike and becomes active, which means system Π starts to simulate instruction l_j of register machine M.

– If neuron σ_r has no spike inside (corresponding to the fact that the number stored in register r is 0), it fires by using spiking rule $\bar{a} \rightarrow a$ at step $t+1$. Neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(4)}}$ get three spikes, meanwhile neuron $\sigma_{l_i^{(5)}}$ receives one spike. With three spikes inside, neuron $\sigma_{l_i^{(4)}}$ fires at step $t+2$, sending one anti-spike to neuron $\sigma_{l_i^{(7)}}$. The three spikes in neuron $\sigma_{l_i^{(3)}}$ will be annihilated by three anti-spikes from neurons $\sigma_{l_i^{(4)}}, \sigma_{l_i^{(5)}}$ and $\sigma_{l_i^{(9)}}$, respectively. At step $t+2$, neuron $\sigma_{l_i^{(5)}}$ fires and emits one anti-spike to neuron $\sigma_{l_i^{(4)}}$, which will be annihilated by the spike from neuron $\sigma_{l_i^{(7)}}$ at step $t+3$. At step $t+6$, neuron σ_{l_k} is activated by receiving one spike from neuron $\sigma_{l_i^{(11)}}$, which simulates system Π starts the simulation to instruction l_k of register machine M.

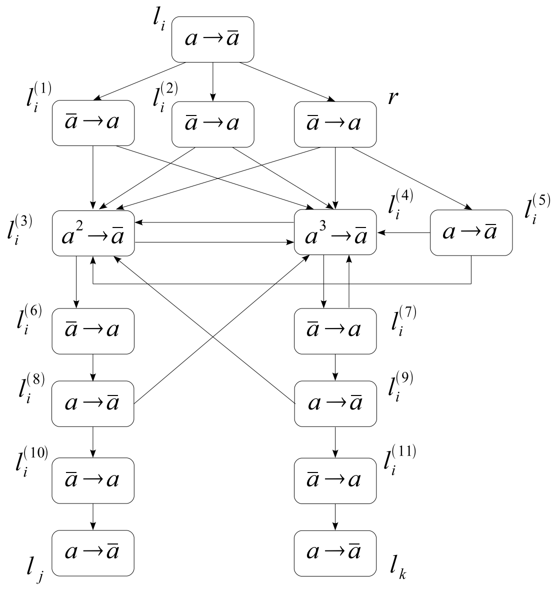

Figure 2: The SUB module of Π

The simulation of SUB instruction is correct: system Π starts from neuron σ_{l_i} becoming active and ends in spiking neuron σ_{l_j} (if the number stored in register r is greater than 0 and decreased by one), or in spiking neuron σ_{l_k} (if the number stored in register r is 0).

Note that there is no interference between two ADD modules. However, it is possible to have interference between two SUB modules. Specifically, if there are several SUB instructions l_v acting on the same register r, neuron σ_r has synapse connections to all neurons $\sigma_{l_v^{(3)}}, \sigma_{l_v^{(4)}}$ and $\sigma_{l_v^{(5)}}$. When a SUB instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$ is simulated, neurons $\sigma_{l_v^{(3)}}, \sigma_{l_v^{(4)}}$ and $\sigma_{l_v^{(5)}}$ in the SUB module associated with l_v ($l_v \neq l_i$) may receive one spike from neuron σ_r. With one spike, neuron $\sigma_{l_v^{(5)}}$ fires and emits one anti-spike to each of neurons $\sigma_{l_v^{(3)}}$ and $\sigma_{l_v^{(4)}}$, which will annihilate the spike in the two neurons. In this way, the interference among SUB modules will not cause undesired steps in system Π (i.e., steps that do not correspond to correct simulations of instructions of register machine M).

Module FIN – outputting the result of computation.

The task of outputting the result is covered by the FIN module shown in Figure 3. Assume that the computation of register machine M halts (that is, the halting instruction l_h is reached).

At that moment, if the number stored in register 1 of register machine M is n ($n \geq 0$), there are $n+1$ spikes in neuron σ_1.

Let t be the moment when neuron σ_{l_h} fires. It sends one anti-spike to neurons $\sigma_{l_h^{(1)}}, \sigma_{l_h^{(2)}}$ and $\sigma_{l_h^{(8)}}$. By receiving one anti-spike from neuron σ_{l_h}, neurons $\sigma_{l_h^{(1)}}$ and $\sigma_{l_h^{(8)}}$ fire at step $t+1$ sending two spikes to neuron $\sigma_{l_h^{(10)}}$. At step $t+7$, neuron σ_{out} fires for the first time by using spiking rule $\bar{a} \rightarrow a$, emitting one spike into the environment.

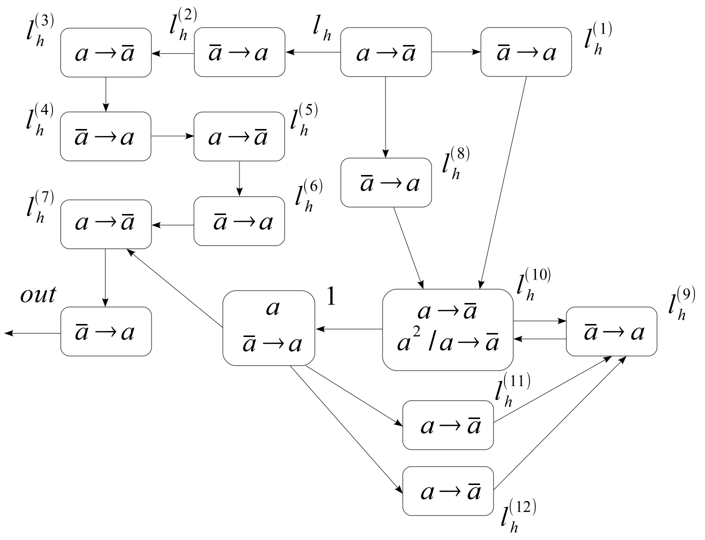

Figure 3: The FIN module of Π

At step $t+2$, by using spiking rule $a^2/a \rightarrow \bar{a}$, neuron $\sigma_{l_h^{10)}}$ fires with sending one anti-spike to neurons σ_1 and $\sigma_{l_h^{(9)}}$. One step later, neurons $\sigma_{l_h^{(9)}}$ and $\sigma_{l_h^{(10)}}$ begin to exchange one spike and one anti-spike among each other. In this way, from that moment on, neuron $\sigma_{l_h^{(10)}}$ continuously sends one anti-spike to neuron σ_1. In each step, neuron σ_1 annihilates one of its spikes by the anti-spike received last step and gets one anti-spike again from neuron $\sigma_{l_h^{(10)}}$. (Before spiking rule $\bar{a} \rightarrow a$ is enabled in neuron σ_1, in total $n+1$ spikes need to be annihilated, which costs $n+1$ steps from $t+3$.) At step $t+n+5$, neuron σ_1 contains one anti-spike and fires, sending one spike to neuron $\sigma_{l_h^{(7)}}$. One step later, neuron $\sigma_{l_h^{(7)}}$ fires again, sending the second spike to neuron σ_{out}. With the spike, neuron σ_{out} fires for the second time at step $t+n+7$ emitting out the second spike to the environment. It is easy to check that the interval between the two spikes emitted by the output neuron equals to $(t+n+7)-(t+7)=n$, which is exactly the number stored in register 1 of register machine M at the moment when the computation halts.

From the above description of the modules and their works, it is clear that register machine M can be correctly simulated by ASN P system without annihilating priority Π, i.e.,

$N(M) = N_2(\Pi)$.

We can check that all neurons in system Π have only one spiking rule inside, except for neuron $\sigma_{l_h^{(10)}}$ in the FIN module. All the spiking rules are of two categories $(a, \bar{a})$ and $(\bar{a}, a)$, that is, each neuron works in a 'flip-flop' way (change spikes into anti-spikes or change anti-spikes into spikes), and no forgetting rule is used. Hence, it is obtained that $NRE \subseteq N_2ASNP_{nopri}(cate_2, rule_2)$. This concludes the proof.

Clearly, the following corollary holds.

Corollary 0.1 *Almost simple ASN P systems without annihilating priority can characterize* NRE *by using spiking rules of categories* $(a, \bar{a})$ *and* $\bar{a}, a)$ *and no forgetting rule.*

4. Final Remarks

In this work, we deal with ASN P systems without annihilating priority. In the systems, when a neuron has both a number of spikes and anti-spikes, the annihilation between the spikes and anti-spikes is not obligatory, and the neuron can choose non-deterministically spiking (if there are enabled spiking rules) or annihilating. The main contribution of this work is to solve an open problem left in ASN P systems formulated in [15]: whether the computational power of ASN P systems as number generators will be reduced or not when the annihilating priority is removed. It is obtained ASN P system without annihilating priority can achieve Turing completeness as number generators, even having at most two rules per neuron.

In our proofs, spiking rules of two categories $(a, \bar{a})$ and $\bar{a}, a)$ are used. It is of interests to develop Turing universal ASN P systems without annihilating priority by using spiking rules of another categories, such as (a, a) and $(a, \bar{a})$, and using no forgetting rule. When anti-spikes are not used, pure spiking rules can only handle finite numbers of spikes hence the system is bounded and can generates only semi-linear sets. Is it possible to obtain the universality of ASN P systems (with or without annihilating priority) by using only pure spiking rules and anti-spikes, and if there are simple universal such systems? How to construct small universal ASN P systems without annihilating priority is also worthy for future works.

Acknowledgements

This work was supported by National Natural Science Foundation of China (61402187, 61502535, 61572522, 61572523, 61672248 and 61672033), China Postdoctoral Science Foundation funded project (2016M592267), Fundamental Research Funds for the Central Universities (2472016010136), Program for New Century Excellent Talents in University (NCET-13-1031) and 863 Program (2015AA020925).

References

[1] Ionescu M, Paun Gh, Yokomori T (2006) Spiking neural P systems, Fundamenta Informaticae, 71(2): 279–308.

[2] Maass W. (1997) Networks of spiking neurons: the third generation of neural network models, Neural Networks, 10(9): 1659–1671.

[3] Chen H, Freund R, Ionescu M (2007) On string languages generated by spiking neural P systems. Fundamenta Informaticae, 75(1-4): 141–162.

[4] Paun A, Paun Gh (2007) Small universal spiking neural P systems. Biosystems, 90: 48–60.

[5] Cavaliere M, Ibarra OH, Paun Gh, Egecioglu O, Ionescu M, Wood S (2009) Asynchronous spiking neural P systems, Theor Comput Sci, 410(24): 2352–2364.

[6] Song T, Pan L, Paun Gh (2012) Asynchronous spiking neural P systems with local synchronization, Information Sciences, 219: 197–207.

[7] Wang X, Song T, Gong F, Zheng P (2016) On the computational power of spiking neural P systems with self-orgniaztion, Scientific Reports, 6 Article number: 27624.

[8] Song T, Pan Z Wong DM, Wang X (2016) Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control, Inform Sciences, 372: 380–391

[9] Peng H, Wang J, Perez-Jimenez MJ (2013) Fuzzy reasoning spiking neural P system for fault diagnosis, Inform Sciences, 235: 106–116.

[10] Zhang G, Rong H, Neri F, Perez-Jimenez MJ (2014) An optimization spiking neural P system for approximately solving combinatorial optimization problems, Int J Neural Syst, 24(05).

[11] Chen Z, Zhang P, Wang X (2016) A computational approach for nuclear export signals identification using spiking neural P systems, Neural Computing and Applications, DOI: 10.1007/s00521-016-2489-z.

[12] G. Păun, G. Rozenberg, and A. Salomaa, eds. The Oxford handbook of membrane computing, Oxford University Press, 2010.

[13] http://ppage.psystems.eu/

[14] http://membranecomputing.net/IMCSBulletin/

[15] Pan L, Paun Gh (2009) Spiking neural P systems with anti-spikes. International Journal of Computers Communications & Control, 4(3): 273–282.

[16] Song T, Liu X, Zhao Y, Zhang X (2016) Spiking neural P systems with white hole neurons, IEEE T Nanobiosci, 15(7): 666–673.

[17] Song T, Wang X (2014) Homogenous spiking neural P systems with inhibitory synapses. Neural Process Lett, 42(1): 199–214

[18] Song T, Pan L, Wang J (2012) Normal forms of spiking neural P systems with anti-spikes. IEEE T Nanobiosci, 4(11): 352–359.

[19] Song T, Jiang Y, Shi X (2013) Small universal spiking neural P systems with anti-spikes. J Comput Theor Nanos, 10(4): 999–1006.

[20] Rozenberg G, Salomaa A eds. (1997) Handbook of formal languages, Berlin: Springer-Verlag.