ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **17**, Number 1, 2014, 19–32

Small Universal Spiking Neural P Systems with Astrocytes

Yuan KONG, Keqin JIANG, Zhihua CHEN, Jinbang XU

Key Laboratory of Image Processing and Intelligent Control
School of Automation, Huazhong University of Science and Technology
Wuhan 430074, Hubei, China

E-mail: kongyuan1122@126.com, jiangkq0519@163.com
chenzhihua@hust.edu.cn (Corresponding author)
xujinbang@mail.hust.edu.cn

Abstract. Spiking neural P systems with astrocytes (SNPA systems, for short) are a class of distributed parallel computing devices inspired from the way neurons communicate by means of spikes, where also astrocytes are considered, having an excitatory or an inhibitory influence on synapses. Looking for small universal computing devices is a classical research topic in computer science. In this work, we investigate small universal SNPA systems as both devices computing functions and devices generating sets of numbers. Specifically, as devices of computing functions, a universal SNPA system (without delay and forgetting rules) using 57 neurons and 19 astrocytes is obtained; as generators of sets of numbers, a universal SNPA system (without delay and forgetting rules) using 54 neurons and 17 astrocytes is found. These improve previous results on SN P systems without astrocytes and show the role of these features.

Key-words: Membrane computing; Spiking neural P system; Astrocyte; Universality; Register machine.

1. Introduction

Spiking neural P systems (SN P systems, for short) are a class of distributed and parallel computing devices introduced in [1] and then intensively investigated. Details can be found in [9] and the membrane website [10]. Spiking neural P systems with anti-spikes were introduced in [4], which draw inspiration from the way neurons communicate through both excitatory and inhibitory spikes; a new form of such

systems was given in [15]. The notion of local synchronization was introduced into asynchronous SN P systems in [13], the constructed systems were also proved to be computationally complete in the generating case. Moreover, finite languages and recursively enumerable languages were shown to be characterized by asynchronous SN P systems in [18]. Small universal spiking neural P systems with rules on synapses were constructed in [14] for computing functions.

Briefly, an SN P system consists of a set of *neurons* placed in the nodes of a directed graph, which send signals (*spikes*) along *synapses* (arcs of the graph). Each neuron contains a number of spikes, spiking (also called firing) rules and forgetting rules. Using these rules, the spikes contained in a neuron can be moved or deleted.

Generally, an SN P system can be seen as a network of neurons. Recently, a new variant of SN P systems consisting of neurons and astrocytes was presented in [5], called spiking neural P systems with astrocytes (SNPA systems, for short), where an astrocyte can sense the spike traffic along several neighboring synapses at the same time, and has an excitatory or an inhibitory influence on these neighboring synapses. Actually, the feature of astrocytes was already considered in [8], and the astrocytes were formulated as a control of spikes traffic along axons.

Looking for small universal computing devices is a classical research topic in computer science, see, e.g., [2], [11], and the references therein. This topic has been heavily investigated in the framework of SN P systems [6], where a universal SN P system was obtained using 84 neurons for computing functions, and a system with 76 neurons can generate any Turing computable set of natural numbers. In [19], these results were improved: 67 neurons in the case of computing functions, and 63 neurons in the case of generating sets of numbers. Small universal SN P systems were also considered in [17] when the systems work in an exhaustive manner (that is, if a rule contained by a neuron can be applied, then it is used as many times as possible in the neuron).

In this work, we investigate small universal SNPA systems (without delay and forgetting rules) by simulating a register machine as defined in [2]. We obtain that 57 neurons and 19 astrocytes can compute any Turing computable function; as number generators, 54 neurons and 17 astrocytes can generate any Turing computable set of natural numbers. Note that the number of neurons in our system is smaller than in [19], which proves that the astrocytes are useful in this framework. These results give a positive answer to the open problem formulated in [5].

2. Spiking Neural P Systems with Astrocytes

In this section, we review the definitions of spiking neural P systems with astrocytes as introduced in [5] (in the present work, the feature of delays is not used and we omit it) and of deterministic register machines. For the details of formal language theory and membrane computing, please refer to [12] and [7, 10].

A deterministic register machine is a construct $M = (m, H, l_0, l_h, I)$, where m is the number of registers, H is the set of instruction labels, l_0 is the start label, l_h is the halt label, and I is the set of instructions; each label from H labels only one

instruction from I, thus precisely identifying it. The instructions are of the following forms:

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

A register machine M accepts a number n in the following way: starting with number n in a specified register r_0 and all other registers being empty (i.e., storing the number zero), we apply the instruction with label l_0 and continue to apply instructions as indicated by the labels (and made possible by the contents of registers); the number n is said to be accepted by M if M reaches the halt instruction. If the computation does not halt, then no number is accepted. The set of all numbers accepted by M is denoted by $N(M)$.

A register machine can also compute any Turing computable function: we place the arguments $n_1, n_2, \ldots, n_k$ in specified registers $r_1, \ldots, r_k$ (without loss of the generality, assume that the first k registers are used), and start with the instruction with label l_0; if the instruction with label l_h is reached (when the register machine stops), then the value of the function is placed in another specified register r_t, and all registers different from r_t are empty. In this way, the partial function computed by the register machine M is denoted by $M(n_1, n_2, \ldots, n_k)$.

A *spiking neural P system with astrocytes* (SNPA system, for short), of degree $m \geq 1$, $l \geq 1$, is a construct of the form

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, ast_1, \ldots, ast_l, in, out), \text{ where:}$$

- $O = \{a\}$ is the singleton alphabet (a is called *spike*);
- $\sigma_1, \ldots, \sigma_m$ are *neurons*, of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq m$, where:
  a) $n_i \geq 0$ is the *initial number of spikes* contained in σ_i;
  b) R_i is a finite set of *rules* of the following two forms:
     (1) $E/a^c \to a$, where E is a regular expression over a, and $c \geq 1$;
     (2) $a^s \to \lambda$, for some $s \geq 1$, verifying that for every rule $E/a^c \to a$ of type (1) from R_i, $a^s \notin L(E)$;
- $syn \subseteq \{1, 2, ..., m\} \times \{1, 2, ..., m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses* between neurons);
- $ast_1, \ldots, ast_l$ are *astrocytes*, of the form $ast_i = (syn_{ast_i}, t_i)$, where $1 \leq i \leq l$, $syn_{ast_i} \subseteq syn$ is the set of synapses controlled by the astrocyte ast_i, $t_i \in \mathbb{N}$ is the *threshold* of the astrocyte ast_i;
- $in, out \in \{1,2,...,m\}$ indicate the *input* and *output* neurons, respectively.

If a rule $E/a^c \to a$ has $E = a^c$, then it can be simplified in the form $a^c \to a$.

The rules of types (1) and (2) are called *firing* (or *spiking*) rules and forgetting rules, respectively – see [1] for more details.

In each time unit, if neuron σ_i can use one of its rules, then a rule from R_i must be used. If two or more rules can be applied in the neuron, only one of them is chosen non-deterministically. Note that if a firing rule is applicable, then no forgetting rule is applicable, and vice versa. From the above description, the rules are used in the sequential manner in each neuron, but the neurons function in parallel.

In SNPA systems, the astrocytes can sense the spikes traffic along the neighboring synapses at the same time. Assume that astrocyte ast_i has a given threshold t_i, and there are k spikes passing along the synapses in syn_{ast_i}. If $k > t_i$, then the astrocyte ast_i has the inhibitory influence on the neighboring synapses and the k spikes are suppressed (namely, the k spikes are lost from the system). If $k < t_i$, then the astrocyte ast_i has the excitatory influence on the neighboring synapses and the k spikes survive and reach their destination neurons. If $k = t_i$, then the astrocyte ast_i non-deterministically chooses the inhibitory or excitatory influence on the neighboring synapses. In this work, we do not consider that two or more astrocytes control the same synapses.

The *configuration* of the system is of the form $\langle r_1, \cdots, r_m \rangle$, which means that neuron σ_i contains $r_i \geq 0$ spikes. With the above notation, the *initial configuration* of an SNPA system can be expressed as $C_0 = \langle n_1, \cdots, n_m \rangle$ with the number $n_1, n_2, \cdots, n_m$ of spikes present in each neuron. Through the application of the rules described above, one can define *transitions* among configurations. Any sequence of transitions which begins with the initial configuration is called a *computation*. A computation is successful if it reaches a configuration where no rule can be used. The *result of a computation* of the system is defined as the distance in time of the first two spikes sent into the environment by the output neuron.

As stated in [6], in order to compute a function $f : N^k \to N$ (N is the set of all positive integers), we suppose k natural numbers $n_1, \cdots, n_k$ in the system by "reading" a binary sequence $z = 10^{n_1-1}10^{n_2-1}1 \cdots 10^{n_k-1}1$ from the environment, which indicates that the input neuron of the system receives a spike in each step corresponding to a digit 1 from the string z; otherwise, no spike is coming into the system. Suppose that exactly $k + 1$ spikes are received by the input neuron. The result of the computation is described as the distance between the first two spikes emitted by the output neuron. If a spike train of the form $0^b10^{r-1}1$ is generated by the system, for some $b \geq 0$, then $r = f(n_1, \cdots, n_k)$.

3. Small Universal SNPA Systems for Computing Functions

Theorem 1. *There is a universal computing SNPA system (without delay and forgetting rules) having 57 neurons and 19 astrocytes.*

Proof. A register machine can compute any Turing computable function (see e.g., [3]). Suppose that $(\varphi_0, \varphi_1, \ldots)$ is a fixed admissible enumeration of the unary partial recursive functions. A register machine M_u is said to be universal if there is a recursive function g such that for all natural numbers x, y we have $\varphi_x(y) = M_u(g(x), y)$. Like in [6], we consider the universal register machine from [2], shown in Fig. 1,

and simulate it by an SNPA system. We add a further register 8 (which is never decremented during the computation), and replace the original halt instruction of M_u with the following instructions: $l_h : (\texttt{SUB}(0), l_{22}, l'_h), l_{22} : (\texttt{ADD}(8), l_h)$, and $l'_h : \texttt{HALT}$. In this way, we have 9 registers (labeled from 0 to 8), 24 ADD and SUB instructions, and 25 labels. The obtained register machine is denoted by M'_u.

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

Fig. 1. A small universal register machine from Korec [2].

When a register machine M is simulated by an SNPA system Π, with each register r of M, two neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ are associated; with each label l_i of an instruction of the machine, a neuron σ_{l_i} is associated. Specifically, if register r holds the number $n \geq 0$, then each of the neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ contains n spikes (during a computation, the number of spikes in neuron $\sigma_{r^{(1)}}$ is always equal to the number of spikes in neuron $\sigma_{r^{(2)}}$). An astrocyte is denoted by $ast_j^{(i)}$, which means that the i-th astrocyte presents in Fig. j. The simulations of ADD and SUB instructions by the system are presented in Figs. 2 and 3, respectively.

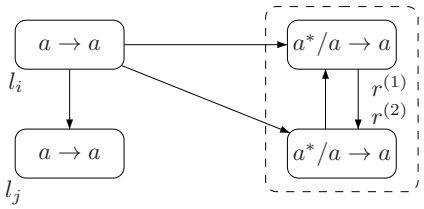

Fig. 2. Module ADD (simulating $l_i : (\text{ADD}(r), l_j)$).

The work of ADD module is very simple: having one spike inside, neuron σ_{l_i} fires by the rule $a \to a$ and simultaneously sends a spike to neurons σ_{l_j}, $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$, respectively. In this way, with one spike inside, neuron σ_{l_j} becomes active and fires; both neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ receive one spike, which simulate the increase of the number stored in register r by one.

The simulation of a SUB instruction is presented in Fig. 3. Let t be the moment when neuron σ_{l_i} fires. At step t, three spikes pass along the synapses (l_i, l_k), $(l_i, l_i^{(1)})$ and $(l_i, l_i^{(2)})$, respectively. Since no astrocyte is controlling the synapse $(l_i, l_i^{(1)})$, the spike passing through it reaches neuron $\sigma_{l_i^{(1)}}$. According to the number of spikes contained by neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$, two cases are considered in the following way.

(1) At step t, each of neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ has n ($n > 0$) spikes (corresponding to the fact that the number stored in register r is n). In this case, neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ send one spike to each other by rule $a^*/a \to a$. Thus, at step t, there are four spikes passing along the synapses controlled by astrocyte $ast_3^{(1)}$ (one spike along each of synapses $(r^{(1)}, r^{(2)})$, $(r^{(2)}, r^{(1)})$, $(l_i, l_i^{(2)})$ and (l_i, l_k)). The number four is greater than the threshold three of astrocyte $ast_3^{(1)}$, so astrocyte $ast_3^{(1)}$ has inhibitory influence on its controlled synapses. These four spikes are suppressed. In this way, each of neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ consumes one spike and receives no spike (that is, the number of spikes in each of neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ is decreased by one), which simulates that the number stored in register r is decreased by one.
At step $t+1$, neuron $\sigma_{l_i^{(1)}}$ spikes and sends out one spike to neuron σ_{l_j}. Because astrocyte $ast_3^{(2)}$ has excitatory influence on its controlled synapses, neuron σ_{l_j} receives the spike from neuron $\sigma_{l_i^{(1)}}$ and becomes active, and the system Π starts to simulate instruction l_j of M.

(2) At step t, each of neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ has no spikes (corresponding to the fact that the number stored in register r is zero). In this case, neuron σ_{l_k} becomes active because of the excitatory influence of astrocyte $ast_3^{(1)}$, and the system Π starts to simulate instruction l_k of M; at step $t+1$, the inhibitory influence of astrocyte $ast_3^{(2)}$ ensures that no spike reaches neuron σ_{l_j}.

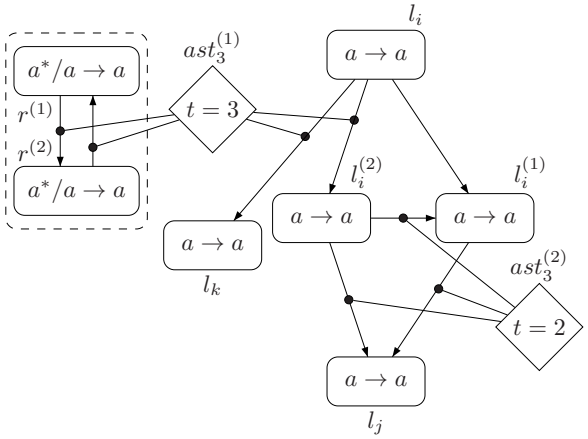

Fig. 3. Module SUB (simulating $l_i : (\texttt{SUB}(r), l_j, l_k)$).

To simulate the universal register machine M'_u, we start with neurons σ_1 and σ_2 already loaded with $g(x)$ and y spikes, respectively. The work of the system is triggered by introducing one spike in neuron σ_{l_0} (associated with the starting instruction of the register machine). Then, the system begins with simulating the instructions of the universal register machine M'_u by the modules from Figs. 2 and 3. In this way, this system can work in the same way as the universal register machine M'_u. So, neuron σ_0 will contains $2\varphi_x(y)$ spikes if the computation halts.

Two problems should be solved for simulating the small universal register machine M'_u, that is, introducing the mentioned spikes in neurons σ_{l_0}, σ_1, σ_2 and outputting the computed number. To solve the first problem, we design the module INPUT presented in Fig. 4. If we "read" a spike train $10^{g(x)-1}10^{y-1}1$ from the environment, then the module works as follows. The first spike from neuron σ_{in} is sent to each of the neurons σ_{c_1}, σ_{c_2}, σ_{c_3} and σ_{c_4}. With one spike inside, by the spiking rule $a \to a$, each of neurons σ_{c_2} and σ_{c_3} fires and sends a spike to each other. Simultaneously, neuron σ_{c_2} sends a spike to neurons $\sigma_{1^{(1)}}$, $\sigma_{1^{(2)}}$, $\sigma_{2^{(1)}}$ and $\sigma_{2^{(2)}}$, respectively; neuron σ_{c_3} sends a spike to neuron σ_{c_4}. There are three spikes passing along the synapses controlled by astrocyte $ast_4^{(1)}$ (one spike along each of synapses $(c_2, 2^{(1)})$, $(c_2, 2^{(2)})$ and (c_3, c_4)). The number three is greater than the threshold two of astrocyte $ast_4^{(1)}$, so astrocyte $ast_4^{(1)}$ has inhibitory influence on its controlled synapses. These three spikes are suppressed. Without any astrocyte controlling synapses (c_2, c_3), (c_3, c_2), $(c_2, 1^{(1)})$ and $(c_2, 1^{(2)})$, the spikes moving along these synapses can reach their destination neurons. In this way, both neurons $\sigma_{1^{(1)}}$ and $\sigma_{1^{(2)}}$ receive one spike, thus the number stored in register 1 is increased by one. When the second spike enters neuron σ_{in}, the neuron fires again. After receiving two spikes (one from neuron σ_{in}, the other from neuron σ_{c_3}), neuron σ_{c_2} is blocked; neuron σ_{c_4} is activated by the spiking rule $a^2/a \to a$. Note that astrocyte $ast_4^{(1)}$ has excitatory influence on its controlled synapses at this moment, and the number stored in register 2 will be desired by neuron σ_{c_4}. After receiving the third spike from neuron σ_{in}, only neuron σ_{c_1} fires and sends out a spike to neuron σ_{l_0}, hence the simulation of M'_u is triggered.

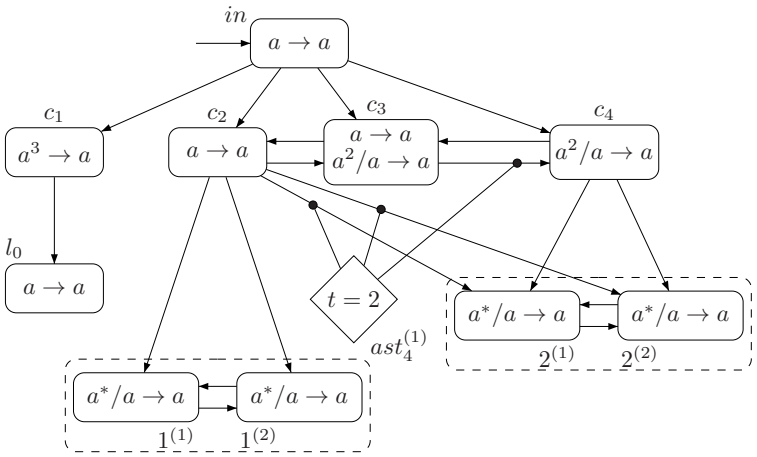

Fig. 4. Module INPUT.

We now solve the second problem by means of the module OUTPUT (register 8 is used for placing the result of the computation), which is served for outputting the computed number. The OUTPUT module is shown in Fig. 5 and it works in the following way. When neuron $\sigma_{l'_h}$ receives one spike, it fires and sends a spike to each of the neurons $\sigma_{8^{(1)}}$ and $\sigma_{8^{(2)}}$; in this way, the spiking rule $aa^+/a \to a$ of

both of the neurons $\sigma_{8^{(1)}}$ and $\sigma_{8^{(2)}}$ can be used, and the two neurons send a spike to each other, while the spikes cannot reach their destination neurons because of the inhibitory influence of astrocyte $ast_5^{(1)}$. Neuron σ_{out} spikes and sends the first spike to the environment after receiving a spike from neuron $\sigma_{8^{(1)}}$. From the next step on until exhausting the spikes from each of neurons $\sigma_{8^{(1)}}$ and $\sigma_{8^{(2)}}$, neuron σ_{out} receives two spikes (one from $\sigma_{8^{(1)}}$ and one from σ_{c_1}), thus it keeps an even number of spikes and cannot fire. When each of neurons $\sigma_{8^{(1)}}$ and $\sigma_{8^{(2)}}$ has only one spike, they cannot fire anymore, but neuron σ_{out} can receive one further spike from neuron σ_{c_1}. Hence, σ_{out} sends out the second spike to the environment.

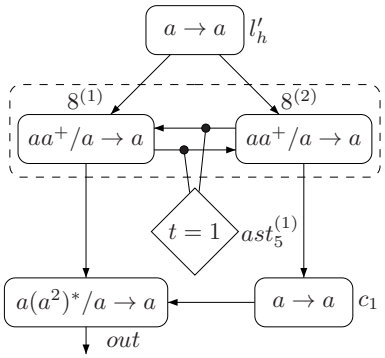

Fig. 5. Module OUTPUT.

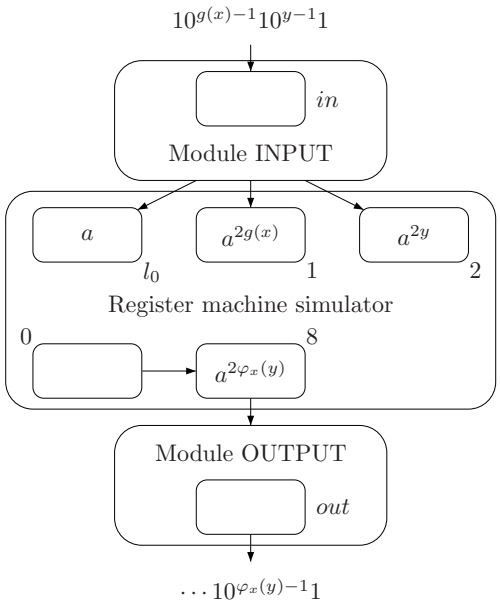

Fig. 6. The general design of the universal SNPA system.

From the above description of the modules and their work, we can find that register machine M'_u is correctly simulated by SNPA system Π. The overall design of the small universal computing system is presented in Fig. 6.

It is clear that we need 18 neurons for the 9 registers, 25 neurons for the 25 labels, 28 neurons and 28 astrocytes for the 24 ADD and SUB instructions, 5 neurons and 1 astrocyte in the INPUT module, 2 neurons and 1 astrocyte in the OUTPUT module. Hence, the total number of neurons and astrocytes are 78 and 30, respectively.

According to some characteristics of the register machine M'_u, the number of neurons can be slightly decreased, by some "code optimization". Specifically, as considered in [6], the sequence of two consecutive ADD instructions: $l_{17} : (\texttt{ADD}(2), l_{21})$ and $l_{21} : (\texttt{ADD}(3), l_{18})$ without any other instruction addressing the label l_{21}, can be combined as in Fig. 7, and in this way we save the neuron associated with l_{21}.

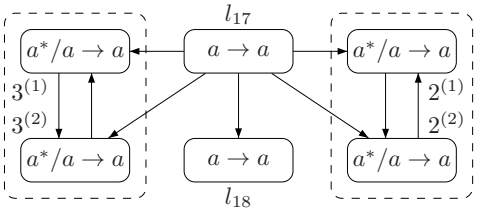

Fig. 7. A module simulating consecutive ADD-ADD instructions.

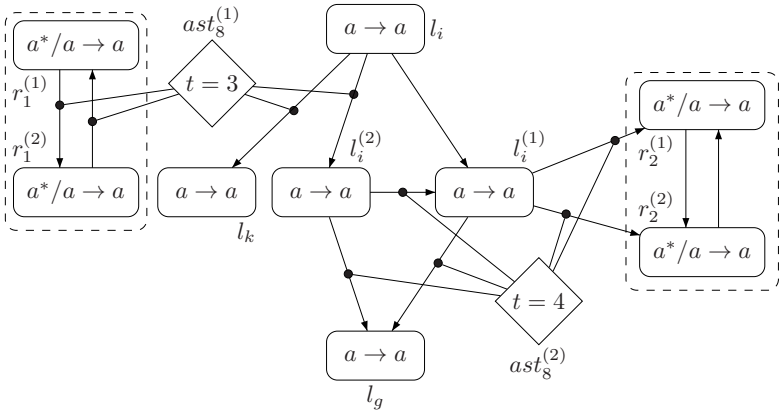

Fig. 8. The module simulating consecutive SUB-ADD instructions.

A similar operation is possible for the following six sequences of SUB-ADD instructions:

$l_0 : (\texttt{SUB}(1), l_1, l_2)$, $l_1 : (\texttt{ADD}(7), l_0)$,
$l_4 : (\texttt{SUB}(6), l_5, l_3)$, $l_5 : (\texttt{ADD}(5), l_6)$,
$l_6 : (\texttt{SUB}(7), l_7, l_8)$, $l_7 : (\texttt{ADD}(1), l_4)$,
$l_8 : (\texttt{SUB}(6), l_9, l_0)$, $l_9 : (\texttt{ADD}(6), l_{10})$,
$l_{14} : (\texttt{SUB}(5), l_{16}, l_{17})$, $l_{16} : (\texttt{ADD}(4), l_{11})$,
$l_h : (\texttt{SUB}(0), l_{22}, l'_h)$, $l_{22} : (\texttt{ADD}(8), l_h)$.

Each consecutive SUB-ADD instruction, $l_i : (\texttt{SUB}(r_1), l_j, l_k)$, $l_j : (\texttt{ADD}(r_2), l_g)$, can be simulated by the module shown in Fig. 8. In this way, we save 6 neurons associated with intermediate labels l_1, l_5, l_7, l_9, l_{16} and l_{22}, respectively.

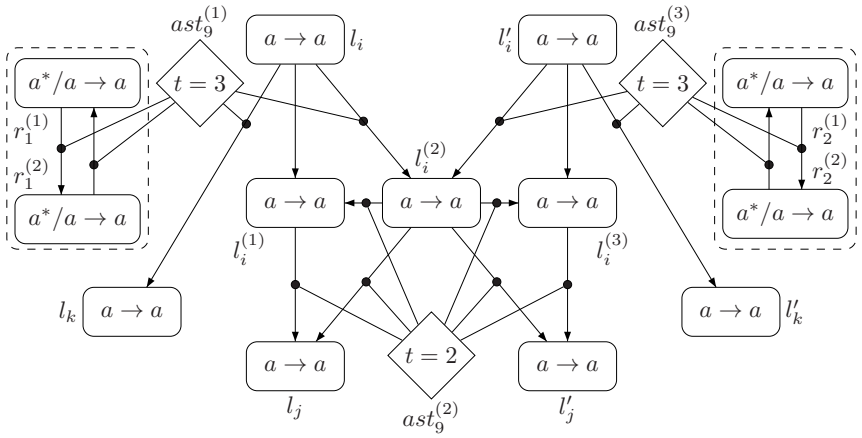

Fig. 9. Module SUB-SUB of instructions which share one neuron and one astrocyte.

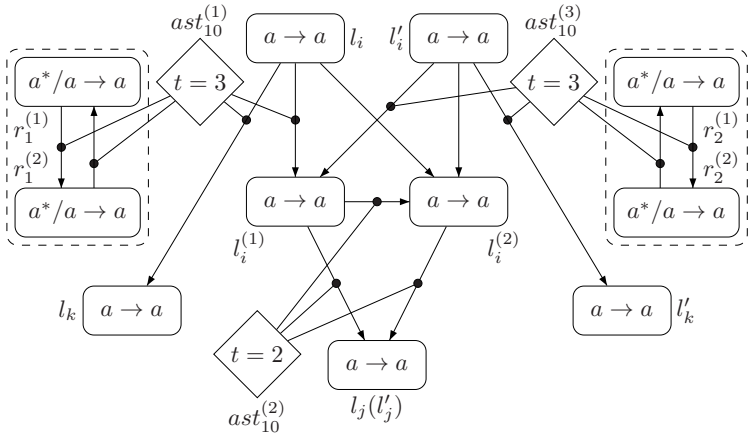

Fig. 10. Module SUB-SUB instructions share two neurons and one astrocyte.

According to the number of shared auxiliary neurons, we can classify the 14 SUB instructions (including $l_h : (\texttt{SUB}(0), l_{22}, l_h')$) to three groups, which are listed as follows:

(1) $l_0 : (\texttt{SUB}(1), l_1, l_2)$, $l_3 : (\texttt{SUB}(5), l_2, l_4)$, $l_4 : (\texttt{SUB}(6), l_5, l_3)$, $l_6 : (\texttt{SUB}(7), l_7, l_8)$, $l_8 : (\texttt{SUB}(6), l_9, l_{10})$, $l_{11} : (\texttt{SUB}(5), l_{12}, l_{13})$, $l_{12} : (\texttt{SUB}(5), l_{14}, l_{15})$, $l_{14} : (\texttt{SUB}(5), l_{16}, l_{17})$, $l_h : (\texttt{SUB}(0), l_{22}, l_h')$;
(2) $l_{10} : (\texttt{SUB}(4), l_0, l_{11})$, $l_{18} : (\texttt{SUB}(4), l_0, l_h)$, $l_{19} : (\texttt{SUB}(0), l_0, l_{18})$;
(3) $l_{13} : (\texttt{SUB}(2), l_{18}, l_{19})$, $l_{15} : (\texttt{SUB}(3), l_{18}, l_{20})$.

For the first group, all modules associated with the instructions can share one auxiliary neuron and one astrocyte (as shown in Fig. 9); for the last two groups,

all modules associated with the instructions can share two auxiliary neurons and one astrocyte (as shown in Fig. 10). So, we save 14 neurons and 11 astrocytes in total.

Since 1 neuron is saved by the ADD-ADD module in Fig. 7, 6 neurons are saved by the SUB-ADD modules in Fig. 8, we can totally save 21 neurons and 11 astrocytes. Hence, the number of neurons can be decremented from 78 to 57, the number of astrocytes from 30 to 19. This completes the proof. □

4. Small Universal SNPA Systems as Number Generators

As mentioned in [6], a generating SN P system Π_u is universal if given a fixed admissible enumeration of the unary partial recursive function, $(\varphi_0, \varphi_1, \cdots)$, there is a recursive function g such that for each natural number x, if we input the number $g(x)$ in Π_u, the set of numbers generated by the system is equal to $\{n \in N \mid \varphi_x(n)\text{is defined}\}$. Otherwise stated, after introducing the "code" $g(x)$ of the partial recursive function φ_x in a specified neuron, the system generates all numbers n for which $\varphi_x(n)$ is defined.

We here consider the same methods from [6]:

(1) Read the string $10^{g(x)-1}1$ from the environment and load $2g(x)$ spikes in neuron σ_1.
(2) Load neuron σ_2 non-deterministically with an arbitrary natural number n (introducing $2n$ spikes in neuron σ_2); at the same time, output the spike train $10^{n-1}1$ (hence the number n).
(3) Check whether the function φ_x is defined for n. To this aim, start the register machine M_u from Fig. 1, with $g(x)$ in register 1 and n in register 2. If the computation in M_u halts, then also the computation in our SNPA system halts, so n is introduced in the set of generated numbers.

We need to point that there is an important difference between number generating and function computing: it is not necessary to output a result after halting the computation, while we have to randomly generate a number at the beginning of the computation. In this way, some modifications are made in the construction of the universal computing SNPA systems: (1) no separate OUTPUT module is necessary, the further register 8 and label l_h are omitted; (2) an INPUT-OUTPUT module (the INPUT module together with the output one) is used, at the same time non-deterministically producing the number n. The INPUT-OUTPUT module is shown in Fig. 11. The modified module for simulating the instruction $l_{18} : (\texttt{SUB}(4), l_0, l_h)$ is shown in Fig. 12. We find that l_{18} needs three auxiliary neurons; when substraction is no longer possible, the system halts.

When loading neuron σ_1 with $2g(x)$ spikes, we start loading neuron σ_2 with an arbitrary number of spikes through neuron σ_{c_4}, at the same time emitting the respective number as the distance between two spikes sent out by neuron σ_{out}. The work of neurons σ_{c_3}, σ_{c_4} stops when astrocyte $ast_{11}^{(1)}$ has inhibitory influence on synapses (c_3, c_4) and (c_4, c_3). After having two spikes emitted by σ_{out}, we load neuron σ_{l_0}

with one spikes via neuron σ_{c_6}. In this way, the work of the register machine M_u is triggered.

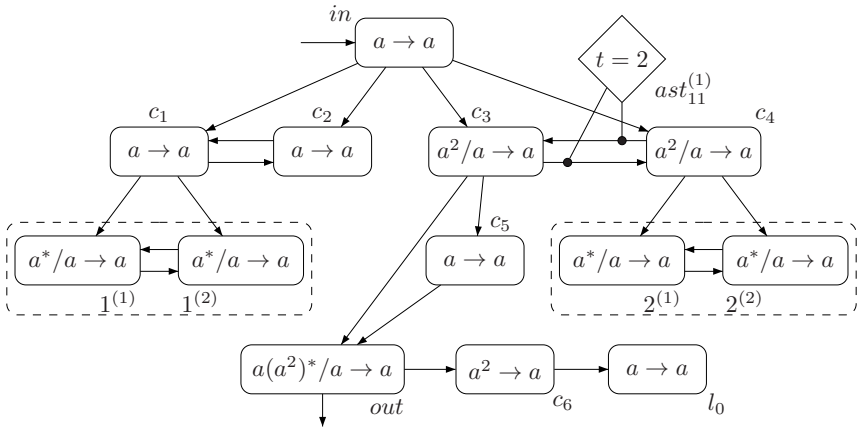

Fig. 11. The INPUT–OUTPUT module for number generating universal systems.

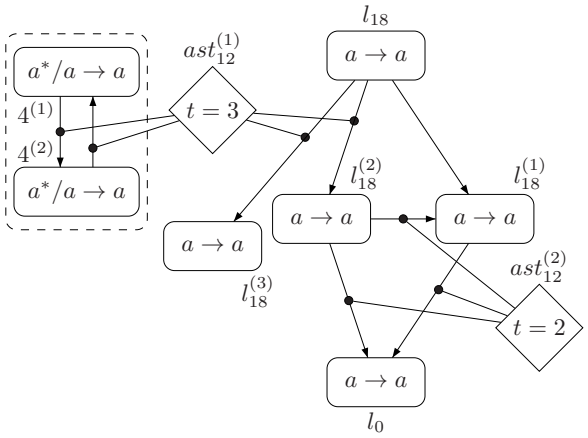

Fig. 12. The modified module for simulating $l_{18} : (\texttt{SUB}(4), l_0, l_h)$.

Hence, the obtained system contains: 16 neurons for the 8 registers, 22 neurons for the 22 labels, 27 neurons and 26 astrocytes for the 22 ADD and SUB instructions, 8 neurons and 1 astrocyte in the INPUT-OUTPUT module. The total number of neurons and astrocytes are 73 and 27, respectively. From the observation in Section 3, we can save 19 neurons (1 for ADD-ADD instruction, 5 for SUB-ADD instructions and 13 for SUB-SUB instructions) and 10 astrocytes (for SUB-SUB instructions). So, the following result is obtained:

Theorem 2. *There is a universal number generating SNPA system (without delay and forgetting rules) having 54 neurons and 17 astrocytes.*

5. Conclusions and Remarks

In this work, we investigate universal SNPA systems to compute functions as well as to generate/compute sets of natural numbers. Specifically, it is obtained that SNPA systems with 57 neurons and 19 astrocytes can compute any Turing computable functions and SNPA systems with 54 neurons and 17 astrocytes can generate any Turing number sets. It is emphasized that no delay and forgetting rules are used. These results give a positive answer to an open problem left in [5], and show that astrocytes can improve the computing power of SN P systems.

There are several open problems and research topics suggested by the previous results. For instance, is it possible to significantly reduce the number of neurons using more astrocytes? Moreover, in [16], a special neuron is used for each instruction of the register machine, and the number of neurons in universal SN P systems with extended rules can be significantly reduced to 9. It is of interest to investigate what results can be obtained if a similar special neuron is used in universal SNPA systems. It also deserves to be checked whether the feature of delay and forgetting rules is useful in improving the current results.

Acknowledgements. This work was supported by National Natural Science Foundation of China (61033003, 91130034, 61320106005, and 61370105). Fundamental Research Funds for the Central Universities (2013TS124). Natural Science Foundation of Hubei Province (2013CFB159), Anhui Provincial Natural Science Foundation (1408085MF131), Natural Science Research Project for Higher Education Institutions of Anhui Province(KJ2014A140).

References

[1] IONESCU M., PĂUN Gh., YOKOMORI T., *Spiking neural P system*, Fundamenta Informaticae, 2006, Vol. **71**(2-3), pp. 279–308.

[2] KOREC I., *Small universal register machines*, Theoretical Computer Science, 1996, Vol. **168**(2), pp. 267–301.

[3] MINSKY M., *Computation – Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.

[4] PAN L., PĂUN Gh., *Spiking neural P systems with anti-spikes*, International Journal of Computers, Communications & Control, 2009, Vol. **4**(3), pp. 273–282.

[5] PAN L., WANG J., HOOGEBOOM H.J., *Spiking neural P systems with astrocytes*, Neural Computation, 2012, Vol. **24**(3), pp. 805–825.

[6] PĂUN A., PĂUN Gh., *Small universal spiking neural P systems*, BioSystems, 2007, Vol. **90**(1), pp. 48–60.

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

[8] PĂUN Gh., *Spiking neural P systems with astrocyte-like control*, Journal of Universal Computer Science, 2007, Vol. **13**, pp. 1707–1721.

[9] PĂUN Gh., Rozenberg G., Salomaa A. (Eds.), *The Oxford Handbook of Membrane Computation*, Oxford Univ. Press., 2010.

[10] P systems web page `http://ppage.psystems.eu`

[11] ROGOZHIN Y., *Small universal Turing machines*, Theoretical Computer Science, 1996, Vol. **168**, pp. 215–240.

[12] ROZENBERG G., SALOMAA A. (Eds.), *Handbook of Formal Languages, 3 volumes*, Springer-Verlag, Berlin, 1997.

[13] SONG T., PAN L., PĂUN Gh., *Asynchronous spiking neural P systems with local synchronization*, Information Sciences, 2013, Vol. **219**, pp. 197–207.

[14] SONG T., PAN L., PĂUN Gh., *Spiking neural P systems with rules on synapses*, Theoretical Computer Science, 2014, DOI: 10.1016/j.tcs.2014.01.001.

[15] SONG T., PAN L., WANG J., VENKAT I., SUBRAMANIAN K.G., ABDULLAH R., *Normal forms of spiking neural P systems with anti-spikes*, IEEE Transactions Nanobioscience, 2012, Vol. **11**(4), pp. 352–359.

[16] ZENG X., LU C., PAN L., *A weakly universal spiking neural P system*, Mathematical and Computer Modelling, 2010, Vol. **52**, pp. 1940–1946.

[17] ZHANG X., JIANG Y., PAN L., *Small universal spiking neural P systems with exhaustive use of rules*, Journal of Computational and Theoretical Nanoscience, 2010, Vol. **7**, pp. 1–10.

[18] ZHANG X., ZENG X., PAN L., *On languages generated by asynchronous spiking neural P systems*, Theoretical Computer Science, 2009, Vol. **410**, pp. 2478–2488.

[19] ZHANG X., ZENG X., PAN L., *Smaller universal spiking neural P systems*, Fundamenta Informaticae, 2008, Vol. **87**, pp. 117–136.