



Spiking neural P systems with polarizations and astrocytes

Suxia Jiang¹ · Zhichao Shen¹ · Bowen Xu¹ · Xiaoliang Zhu¹ · Tao Liang¹

Received: 20 August 2022 / Accepted: 25 February 2023 / Published online: 28 March 2023
© The Author(s), under exclusive licence to Springer Nature Singapore Pte Ltd. 2023

Abstract

Spiking neural P systems with polarizations (PSN P systems) use charges (+, −, 0) instead of regular expression to obtain excellent computation power and broad application prospect. In this work, astrocyte control mechanism is introduced into PSN P systems, spiking neural P systems with polarizations and astrocytes (PASN P systems) are constructed. Astrocytes are both excitatory and inhibitory influences on synapses, which can effectively reduce the consumption of computing resources (the use of fewer neurons). Because of the effects of astrocytes, PASN P systems are proved to have the computation power equivalent to Turing machines in generation and accepting modes. Furthermore, a small universal PASN P system with 82 neurons is given for computing any Turing computable function, that is, fewer neurons are used to construct the relatively simple and universal PASN P systems.

Keywords Bio-inspired computing · Membrane computing · Spiking neural P systems · Polarizations · Astrocytes

1 Introduction

Membrane computing, as a current branch of natural computing, aims to abstract computational models (called membrane systems or P systems) from the structure and function of living cells and from the cooperation of cell groups such as tissues and organs [1, 2]. Membrane computing is characterized by the use of distributed parallel computing to make the computation power greater effective and more power [3, 4]. Membrane computing has developed rapidly since it was proposed and has become an emerging research field [5–7]. There are three main membrane systems that have been extensively studied: *cell-like membrane systems*, *tissue-like*

membrane systems and *neural-like membrane systems*. More detailed information of membrane computing, readers can refer to <http://ppage.psystems.eu>, the latest research results and developments can be found.

Spiking neural P systems (SN P systems) are a type of neural-like membrane systems in the field of membrane computing, which are abstracted by the way that neurons send spikes to other neurons through synapses to realize information communication [8]. SN P systems can be represented as directed graph, where the vertices of the directed graph represent neurons, and the arcs between the vertices of the directed graph represent synapses along which neurons can send spikes to their associated neurons [9]. Each neuron contains a certain number of spikes, firing rules and forgetting rules. By using the firing rule, the neuron sends information in the form of firing spikes to some other neurons, if the neuron makes use of the forgetting rule, a certain number of spikes will be cleared from the neuron. Typically, there will be an output neuron in the system whose spikes can be sent into the environment [10].

SN P systems have been shown to be computationally universal as a kind of digital generating and accepting devices [11], as language generators [12, 13] and function calculators [14, 15]. Inspired by different biological properties, a quantity of variants of SN P systems have been proposed, such as SN P systems with anti-spikes [16], asynchronous SN P systems [17, 18], local synchronized SN P

✉ Suxia Jiang
jiangsx913@126.com
Zhichao Shen
1072867147@qq.com
Bowen Xu
1094614537@qq.com
Xiaoliang Zhu
1426311595@qq.com
Tao Liang
3369590535@qq.com

¹ School of Electrical and Information Engineering,
Zhengzhou University of Light Industry, Zhengzhou 450002,
China

systems [19], SN P systems with rules on synapses [20], homogenous SN P systems [21, 22], axonal SN P systems [23], SN P systems with scheduled synapses [24], etc. In addition, SN P systems can also solve computationally difficult problems in efficient time (polynomial time or linear time), such as subset sum problems [25], SAT problems [26, 27], PSPACE-complete problem [28, 29], and Travelling salesman problem [30]. SN P systems and the variants also have a number of practical applications, such as combinatorial optimization [31], fault diagnosis of power system [32, 33], knowledge expression and fuzzy reasoning [34, 35].

The software implementation of membrane computing is mainly based on the simulation platform developed by C++, Java and other programming languages to simulate membrane systems, such as *Membrane simulator*, *SNUPS*, *P – Lingua* and *MeCoSim* [36–38], etc.

Small universal computing devices can use fewer computing resources (time or space) to construct universal computational models, which have always been a hot research direction in the discipline of computer science. A recurrent neural network of 886 neurons can simulate any Turing machine has been demonstrated in [39]. As computing devices, various variants of SN P systems have been proposed, such as universal SN P systems with 84 neurons [2], SN P systems with anti-spikes with 75 neurons [40], SN P systems with homogenous neurons and synapses having 70 neurons [41], SN P systems with astrocytes with 57 neurons and 19 astrocytes [42], all of the variants have been proved to be Turing universal. Most of variants of SN P systems aim to reduce computing resources by reducing the number of neurons, that is, to reduce the use of space resources in computing resources.

In [43], spiking neural P systems with polarizations (PSN P systems) were constructed by introducing membrane potential, where the rules can be triggered only when the polarization condition is satisfied, such systems are more consistent with biological phenomena, moreover, a small universal PSN P system with 164 neurons was given. However, the number of neurons required by PSN P systems are much larger than the traditional SN P systems and some variants of SN P systems, the construction of this kind of universal computing model is also complex. In [44], spiking neural P systems with polarizations and rules on synapses (PSNRS P systems) were proposed, where spiking rules are placed on synapses, a small universal PSNRS P system with 151 neurons were shown to be able to compute any Turing computable functions. Obviously, compared with PSN P systems, PSNRS P systems can simplify the system structure and reduce the computing resources.

It is known (see [45]) that astrocytes can release D-serine, which plays an important role in the production of long-term enhancement of neurons, and release adenosine triphosphate, which produces heterosynaptic inhibition of neuronal

activity. In short, astrocytes are both excitatory and inhibitory influences on synapses. Thus, astrocytes can be used to reduce the use of complex parameters to a large extent.

In this work, astrocytes as a control mechanism are introduced into PSN P systems, and make use of the biological characteristics of astrocytes to improve PSN P systems. Meanwhile, in order to solve an open problem raised in [43], we consider using astrocytes to optimize the traditional PSN P systems and reduce computing resources, thus a novel variant of PSN P systems called PSN P system with astrocytes (PASN P systems) is proposed, and the computation power of PASN P systems as generating devices and accepting devices is investigated, respectively. In addition, a small universal PASN P system is constructed, the results show that PASN P systems are equivalent to Turing machines. The main contributions of this work are summarized as follows.

- We introduce the control mechanism of astrocytes into PSN P systems, and propose a novel variant called PASN P systems, where astrocytes connecting different neurons are used to optimize computing resources, that is, the flexibility of PASN P systems is greatly improved, and the structure of such systems is relatively simplified.
- In PASN P systems, astrocytes can control not only the spikes at the synapses of neighboring neurons, but also the charge corresponding to the firing rules. Therefore, utilizing the properties of astrocytes, the computation power of PASN P systems is investigated, we prove that PASN P systems, as number generating devices and number accepting devices, are equivalent to Turing machines.
- Simulating the small universal register machine, we give a small universal PASN P system with 82 neurons. As a computing function, PASN P systems reduce computing resources by 82 neurons compared with PSN P systems.

2 PASN P systems

Before introducing the formal definition of PASN P systems, the reader is assumed to have some knowledge of formal languages and automata. For details, please refer to [46]. The conception of polarizations and astrocytes will also be used, the details of which can be found in [43, 47].

A PASN P system of degree $m \geq 1$ is a tuple

$$\Pi = (O, \sigma_1, \sigma_2, \dots, \sigma_m, syn, ast_1, \dots, ast_l, in, out),$$

where:

- (1) $O = \{a\}$ is an alphabet and a indicates the spike;
- (2) σ_i ($1 \leq i \leq m$) indicates the neurons, the number of neurons is m , and the form of each neuron is $\sigma_i = (\alpha_i, n_i, R_i)$, where:

- (1) $\alpha_i \in \{+, 0, -\}$ indicates the initial charge of the neuron σ_i ;
- (2) n_i indicates that there are n spikes in the neuron σ_i at the initial moment;
- (3) R_i is a set of rules for neurons σ_i , which has the following two forms:
 - a. $\alpha/a^c \rightarrow a;\beta$, where $\alpha, \beta \in \{-, 0, +\}$, $c \geq 1$;
 - b. $\alpha/a^s \rightarrow \lambda;\beta$, where $\alpha, \beta \in \{-, 0, +\}$, $s \geq 1$;
- (3) $syn \in \{1, 2, \dots, m\} \times \{1, 2, \dots, m\}$ are synapses between neurons, with $(i, j) \in syn$, and i, j are the label of neurons ($1 \leq i, j \leq m$, $i \neq j$);
- (4) ast_i ($1 \leq i \leq l$) are the label of astrocytes in the form $ast_i \in (syn_{ast_i}, t_i)$, $syn_{ast_i} \subseteq syn$ is the set of synapses controlled by astrocytes, and t_i is the threshold in astrocytes;
- (5) $in, out \in \{1, 2, \dots, m\}$ represent the labels of the input neuron and the output neuron in the system, respectively.

PSN P systems differ from the traditional SN P systems in the use of rules, the regular expression is replaced by the charge state (+, -, 0). Firing rules have the form $\alpha/a^c \rightarrow a;\beta$, when the neuron σ_i is charged with α and the quantity of spikes in the neuron is not less than c spikes, the rule satisfies the running conditions, c spikes are consumed, and a spike and β charge are sent to its adjacent neurons. When the rules in the output neuron σ_{out} satisfy the excitation condition, the calculation result is sent to the environment, and the charge sent to the environment does not affect the output result. When the input neuron σ_{in} receives spikes and a charge from the environment, the initial execution state of the system is related to the charge obtained from the environment. If a neuron σ_i in the system executes the forgetting rule $\alpha/a^s \rightarrow \lambda;\beta$, the neuron σ_i consumes all the spikes and does not send spikes, and only sends one of charges over β to its neighbouring neurons. The use of rules within a neuron is determined by the polarity of the charge carried by the neuron itself, and the change of the charge is as follows.

- Multiple positive charges (+) (resp. (0), (-)) are equivalent to a positive charge (resp. a neutral charge, a negative charge), and this charge changes in a way that has the highest priority.
- If a neuron with a positive charge (+) (resp. a negative charge (-)) receives a negative charge (-) (resp. a positive charge (+)), the polarity of the neuron becomes a neutral charge (0).
- If a neuron with a positive charge (+) or a negative charge (-) receives a neutral charge (0), the polarity of the neuron remains the same.

PASN P systems can effectively reduce the use of forgetting rules and delay rules by astrocytes control the spikes on the protrusions of adjacent neurons, which reduces the use of complex parameters to a large extent. At one point, astrocytes in PASN P systems can control not only the spikes at the synapses of neighboring neurons, but also the charge corresponding to the firing rules. Synapse collection syn_{ast_i} can influence the synapse (i, j) between ordinary neurons, a threshold value t_i can determine the excitatory influence or inhibitory influence of the astrocytes synapse syn_{ast_i} to the synapse (i, j) between neurons. When an astrocyte ast_i senses k spikes of communication at the synapses of adjacent neurons, there are three cases as follow: if $k > t_i$ and ast_i has an inhibitory influence on adjacent synapses, then these k spikes and the charge corresponding to the trigger rule will be removed from the system; if $k < t_i$, ast_i has an excitatory influence to the adjacent synapses, then the k spikes and the charge can be sent to the target neuron; if $k = t_i$, the excitatory influence and inhibitory influence of astrocyte ast_i on adjacent synapses are nondeterministic, and one of the two influences is chosen non-deterministically.

In addition, a synapse (i, j) can be effected by two or more astrocytes, if the astrocytes have excitatory influence to the neuron synapse, the spikes can reach the target neuron along the neuron synapse. If one of them has an inhibitory influence on the neuron synapse, the corresponding spikes are terminated and removed from the system.

PASN P systems work in the generating mode, the time interval between two non-zero values are sent to the environment by the output neuron σ_{out} , which is encoded as the calculation result, denoted by $N_2(\Pi)$. If PASN P systems work in the accepting mode, the output neuron σ_{out} is removed, and the input neuron σ_{in} is used to read the external spike sequence. It is necessary to recognize that natural numbers are encoded as the time interval between two neighbouring spikes, such as a spike sequence 10^{n-1} . When the spike sequence is sent to the input neuron, the system starts to calculate until it stops. If the calculation result of the system reaches a final configuration, it is said that the system has recognized the value n . The set of results recognized by Π is denoted by $N_{acc}(\Pi)$ [48].

The family of all sets of numbers generated or accepted by PASN P systems is denoted by $N_{\alpha}PASNP$, where $\alpha \in \{2, acc\}$ indicates that the system is in the generating mode ($\alpha = 2$) or the accepting mode ($\alpha = acc$). Furthermore, the system Π is used as a computational function and the computing resource consumption is calculated [48].

3 The computation power of PASN P systems

It is well known that a register machine with three registers can precisely generate a recursively enumerable set of natural numbers that can characterize NRE [48]. In this section,

the computation power of SN P systems is investigated, we show that PASN P systems can generate all recursively enumerable sets of numbers by simulating the register machine.

A register machine is $M = (m, H, l_0, l_h, I)$, where m indicates the number of registers; H indicates the set of all the instruction labels; l_0 indicates the start instruction; l_h indicates the halt instruction, and I indicates the set of all instructions. The specific instruction forms are as follows. ADD instruction $l_i : (ADD(r), l_j, l_k)$ indicates that a register r is incremented by 1, and non-deterministically jumps to the instruction l_j or l_k . SUB instruction $l_i : (SUB(r), l_j, l_k)$ indicates that if the register r contains a number greater than 0, the number stored in the register r minus 1 and the instruction l_j is executed in the next step; if the register r is empty, the instruction l_k is executed in the next step. Here register 1 is an output register, which stores the computing results and sends the result to the environment when the calculation halts.

Theorem 1 $N_2PASNP = NRE$

Proof We prove the computation power of PASN P systems Π by simulating the register machine M , and only need to confirm the inclusion relation $N_\alpha PASNP \supseteq NRE$. The proof process of the system in the generation mode includes an ADD module, a SUB module, and a FIN module. The calculation result of the system are stored in the register 1. During the system format conversion process, there is no corresponding SUB instruction for the register 1 to perform

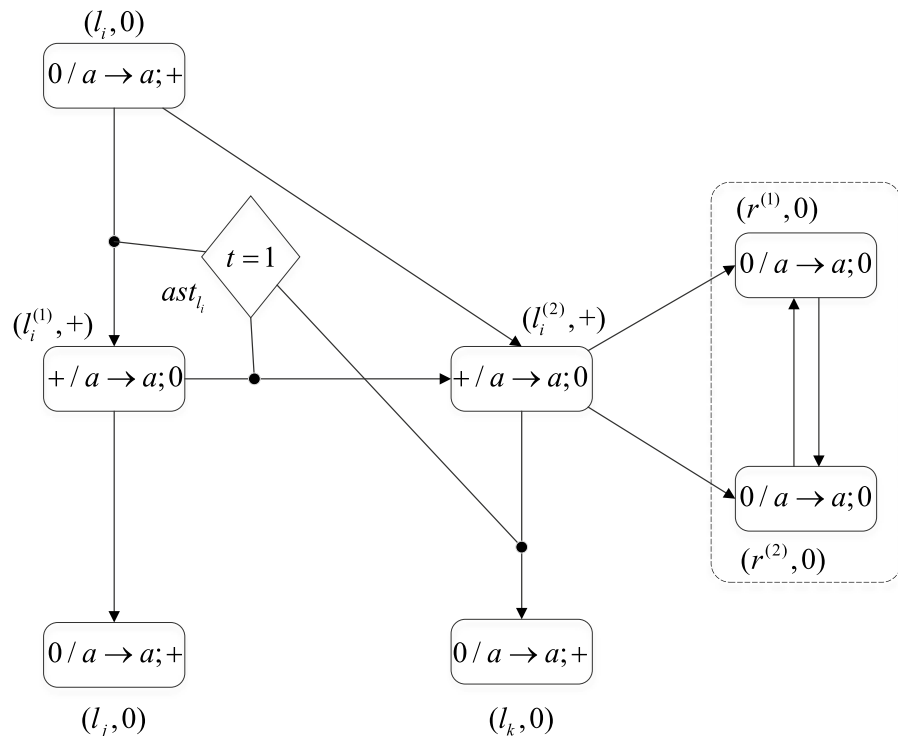
the subtraction operation. When system landscape transfer is stopped, only register 1 is non-empty. In system Π , each register r has a one-to-one correspondence with a neuron σ_r in an ADD or SUB module, and an instruction l_i in H is associated with a neuron σ_{l_i} (subscripts denote neurons in the proof process). It must be clear that when the register r stores a number whose size is n , and then the quantitative relationship contained in the corresponding spike reserve in neuron σ_r corresponds to $2n$. It is worth noting that in each computing module, two auxiliary neurons $\sigma_{r(1)}$ and $\sigma_{r(2)}$ are used to represent the register r .

ADD module: simulating an ADD instruction $l_i : (ADD(r), l_j, l_k)$.

ADD module as shown in Fig. 1. At step t , suppose neuron σ_{l_i} receives a spike and then rule $0/a \rightarrow a;+$ reaches the firing condition, a spike and a positive charge are sent by neuron σ_{l_i} to neuron $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(2)}}$, respectively. It is important to note that the threshold t of the astrocyte ast_{l_i} is equal to the k spikes on synapse $(l_i, l_i^{(1)})$, satisfying the condition $k = t$. Since ast_{l_i} has an excitatory or inhibitory influence on synapse $(l_i, l_i^{(1)})$, which needs to be considered in two cases.

Case I: At step $t + 1$, when ast_{l_i} has an excitatory influence on the synapse, then a spike and a positive charge are sent to neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$ along $(l_i, l_i^{(1)})$ and $(l_i, l_i^{(2)})$. At step $t + 2$, ast_{l_i} senses a total of two spikes transmitted on synapses $(l_i^{(1)}, l_i^{(2)})$, $(l_i^{(2)}, l_k)$, and the number of spikes is greater than the threshold in ast_{l_i} , so the two spikes and charge cannot reach the target neuron and are deleted from the system.

Fig. 1 ADD module: simulating $l_i : (ADD(r), l_j, l_k)$



At the same time, a spike and a neutral charge are sent to neurons $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ along $(l_i, r^{(1)})$ and $(l_i, r^{(2)})$, which means that the register r is incremented by 1. And rule $+/a \rightarrow a;0$ in neuron $\sigma_{l_i^{(1)}}$ reaches the firing condition, and after application, a spike and a neutral charge are sent to neuron σ_{l_j} along the synapse $(l_i^{(1)}, l_j)$. In the next step, the neuron σ_{l_j} is fired and the system starts to simulate the l_j instruction.

Case II: At step $t + 1$, a spike and a positive charge are removed when ast_{l_i} has an inhibitory influence on synapse $(l_i, l_i^{(1)})$. At step $t + 2$, a spike and a natural charge is applied by neuron $\sigma_{l_i^{(2)}}$ with rule $+/a \rightarrow a;0$, both of them are sent to neurons σ_{l_k} , $\sigma_{r^{(1)}}$ and $\sigma_{r^{(2)}}$ through synapses $(l_i^{(2)}, l_k)$, $(l_i^{(2)}, r^{(1)})$ and $(l_i^{(2)}, r^{(2)})$. In the next step, the neuron σ_{l_k} is fired and the system starts to simulate the l_k instruction.

So far, the simulation of the ADD module is completed, the register r can be incremented by 1, and the l_k or l_j instruction can be executed non-deterministically. The operation processes of ADD module are shown in Table 1.

SUB module: simulating a SUB instruction $l_i : (\text{SUB}(r), l_j, l_k)$.

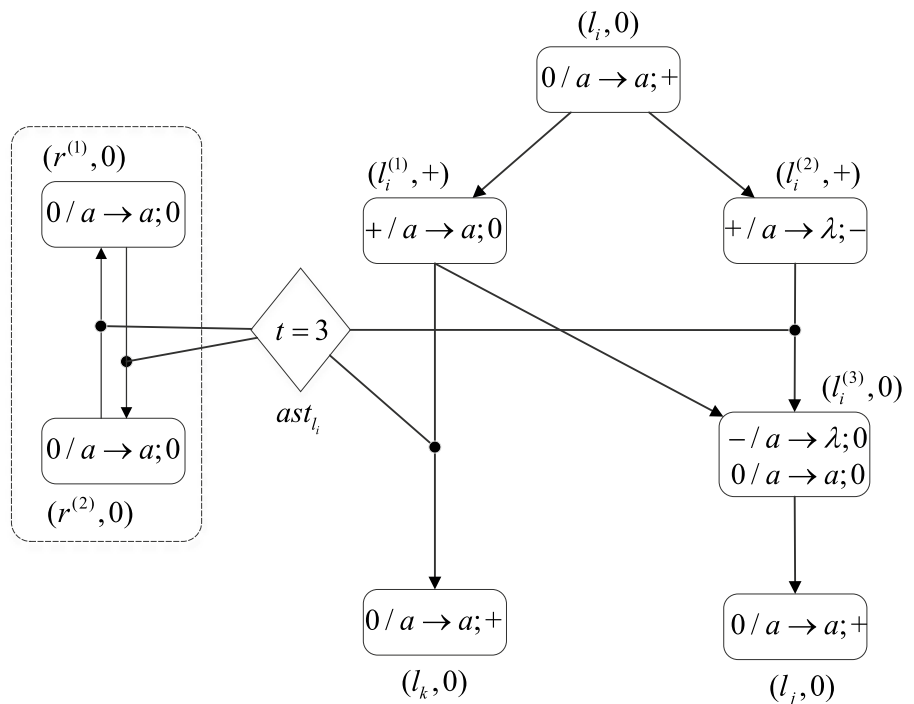
SUB module as shown in Fig. 2, it is assumed that at step t , the neuron σ_{l_i} is activated after receiving a spike. Since it needs to check the empty or non-empty state of the register r , the astrocytes ast_{l_i} check the number of spikes at the adjacent synapses is different, two situations need to be considered.

Case I: When the register r is empty, no spikes are delivered on the synapses $(r^{(1)}, r^{(2)})$ and $(r^{(2)}, r^{(1)})$. At step $t + 1$, rule $0/a \rightarrow a;+$ in neuron σ_{l_i} reaches the firing condition, a spike and positive charge are sent to neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(2)}}$. At step $t + 2$, ast_{l_i} senses two spikes transmission on adjacent synapses $(l_i^{(1)}, l_k)$ and $(l_i^{(2)}, l_i^{(3)})$, the number of spikes is less than the threshold in ast_{l_i} , and the spikes can reach target neuron. Rule $+/a \rightarrow a;0$ in neuron $\sigma_{l_i^{(1)}}$ reaches the firing condition, a spike and a neutral charge are sent to neurons $\sigma_{l_i^{(3)}}$ and σ_{l_k} . Rule $0/a \rightarrow \lambda;-$ in neuron $\sigma_{l_i^{(2)}}$ reaches the firing condition, a negative charge is sent to neuron $\sigma_{l_i^{(3)}}$. When

Table 1 The numbers of spikes and polarity of neurons in add module (see Fig. 1) during the simulation with neuron σ_{l_j} and σ_{l_k} finally activated

Step	Neuron							
	$\sigma_{ast_{l_i}}$	σ_{l_i}	$\sigma_{r^{(1)}}$	$\sigma_{r^{(2)}}$	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	σ_{l_j}	σ_{l_k}
t	–	$(a, 0)$	$(0, 0)$	$(0, 0)$	$(0, +)$	$(0, +)$	$(0, 0)$	$(0, 0)$
$t + 1$	Excitatory	$(0, 0)$	$(a, 0)$	$(a, 0)$	$(a, +)$	$(a, +)$	$(0, 0)$	$(0, 0)$
$t + 2$	Inhibitory	$(0, 0)$	$(a, 0)$	$(a, 0)$	$(0, +)$	$(0, +)$	$(a, 0)$	$(0, 0)$
t	–	$(a, 0)$	$(0, 0)$	$(0, 0)$	$(0, +)$	$(0, +)$	$(0, 0)$	$(0, 0)$
$t + 1$	Inhibitory	$(0, 0)$	$(a, 0)$	$(a, 0)$	$(0, +)$	$(a, +)$	$(0, 0)$	$(0, 0)$
$t + 2$	Excitatory	$(0, 0)$	$(a, 0)$	$(a, 0)$	$(0, +)$	$(0, +)$	$(0, 0)$	$(a, 0)$

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



neuron $\sigma_{l_i^{(3)}}$ received a negative charge and a neutral charge, it takes negative charge and stored a spike. At step $t + 3$, rule $-/a \rightarrow \lambda;0$ in neuron $\sigma_{l_i^{(3)}}$ reaches the firing condition, a neutral charge is sent to neuron σ_{l_j} , but neuron σ_{l_j} does not satisfy the excitation condition in the next step. Simultaneously, the neuron σ_{l_k} is fired and the system starts to simulate the l_k instruction.

Case II: When the state of the register r in the SUB module is non-empty, there are two spikes passing on the synapses $(r^{(1)}, r^{(2)})$ and $(r^{(2)}, r^{(1)})$. Unlike *Case I*, at step $t + 2$, ast_{l_i} senses the adjacent synapses $(r^{(1)}, r^{(2)})$, $(r^{(2)}, r^{(1)})$, $(l_i^{(1)}, l_k)$ and $(l_i^{(2)}, l_i^{(3)})$ each has spike delivery. The number of spikes is greater than the threshold in ast_{l_i} , causing ast_{l_i} to have an inhibitory influence on adjacent synapses, and all spikes involved will be forgotten. Only neuron $\sigma_{l_i^{(3)}}$ receives a spike and a neutral charge from neuron $\sigma_{l_i^{(1)}}$. At step $t + 3$, rule $0/a \rightarrow a;0$ in neuron $\sigma_{l_i^{(3)}}$ reaches the firing condition, a spike and a neutral charge are sent to neuron l_j . At this point, the neuron σ_{l_j} is fired and the system starts to simulate the l_j instruction in the next step.

So far, the SUB module can be accurately simulated, if the register r is in a non-empty state, subtract 1 from the register r and jump to the l_j instruction; if the register r is empty, it directly jumps to the l_k instruction. The operation processes of SUB module are shown in Table 2.

It is worth noting that there is no mutual influence between ADD module and SUB module, and only the corresponding neuron σ_{l_j} or σ_{l_k} needs to be excited. However, there is mutual influence between the SUB modules. Specifically, if there are multiple SUB instructions l_i acting on register r , then all astrocytes ast_{l_i} can control synapses $(r^{(1)}, r^{(2)})$ and $(r^{(2)}, r^{(1)})$. When simulating the SUB instruction $l_i : (SUB(r), l_j, l_k)$, astrocyte ast_{l_i} has a stimulating effect on the synapses $(r^{(1)}, r^{(2)})$ and $(r^{(2)}, r^{(1)})$, and the spike can reach the target neuron smoothly. When simulating the ADD instruction $l_i : (ADD(r), l_j, l_k)$, the interaction between the

SUB modules does not cause the wrong calculation process of the system Π .

FIN module(outputting the result of the computation)

FIN module as shown in Fig. 3, which is used to output the result of the computation. Assuming that at step t , neuron σ_{l_h} receives a spike. At step $t + 1$, neuron σ_{l_h} is activated, which uses the rule $0/a \rightarrow a;+$ to reach the firing condition, a spike and a positive charge are sent to neurons $\sigma_{l_h^{(1)}}$, $\sigma_{l_h^{(2)}}$, $\sigma_{l_h^{(3)}}$, respectively. When neurons $\sigma_{l_h^{(2)}}$ and $\sigma_{l_h^{(3)}}$ received a positive charge, they take neutral charge and stored a spike. At step $t + 2$, $ast_{l_h^{(1)}}$ checks that the number of spikes delivered on synapses $(r^{(1)}, r^{(2)})$, $(r^{(2)}, r^{(1)})$, $(l_h^{(2)}, out)$ and $(l_h^{(2)}, l_h^{(4)})$ are greater than the threshold in $ast_{l_h^{(1)}}$, then these spikes are removed from the system, that is to say, the register r performs a subtraction operation by 1. Simultaneously, rule $+/a \rightarrow a;0$ in neuron $\sigma_{l_h^{(1)}}$ reaches the firing condition, a spike and a neutral charge are sent to neuron out . At step $t + 3$, a spike is sent to the external environment by the neuron σ_{out} .

From step $t + 2$, neurons $\sigma_{l_h^{(2)}}$, $\sigma_{l_h^{(3)}}$ will transmit spikes cyclically, two neurons keep one spike all the time, the median value of register r is decremented by 1 at each step by $ast_{l_h^{(1)}}$. Until step $t + n + 2$, the number of spikes in the corresponding neuron of the register is exhausted, and the number of spikes on the adjacent synapses sensed by $ast_{l_h^{(1)}}$ is less than its threshold. Rule $0/a \rightarrow a;0$ in neuron $\sigma_{l_h^{(2)}}$ reaches the firing condition that sends a spike and a neutral charge to neurons σ_{out} and $\sigma_{l_h^{(4)}}$, respectively. At step $t + n + 3$, rule $0/a \rightarrow \lambda;-$ in neuron $\sigma_{l_h^{(4)}}$ reaches the firing condition that sends a negative charge to neurons $\sigma_{l_h^{(2)}}$ and $\sigma_{l_h^{(3)}}$ to help them return to the initial polarizations, and neurons $\sigma_{l_h^{(2)}}$ and $\sigma_{l_h^{(3)}}$ stop exchanging spikes. At the same time, the second spike is sent to the outside of the module by neuron σ_{out} , and the system's format conversion is stopped. Time interval calculation of the first two spikes: $(t + n + 3) - (t + 3) = n$,

Table 2 The numbers of spikes and polarity of neurons in SUB module (see Fig. 2) when the simulation with register σ_r is empty or not empty

Neuron	Step of register r is empty				Step of register r is non-empty			
	t	$t + 1$	$t + 2$	$t + 3$	t	$t + 1$	$t + 2$	$t + 3$
$\sigma_{r^{(1)}}$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	$(a^n, 0)$	$(a^n, 0)$	$(a^{n-1}, 0)$	$(a^{n-1}, 0)$
$\sigma_{r^{(2)}}$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	$(a^n, 0)$	$(a^n, 0)$	$(a^{n-1}, 0)$	$(a^{n-1}, 0)$
σ_{l_i}	$(a, 0)$	(0, 0)	(0, 0)	(0, 0)	$(a, 0)$	(0, 0)	(0, 0)	(0, 0)
$\sigma_{ast_{l_i}}$	—	—	Excitatory	—	—	—	Inhibitory	—
$\sigma_{l_i^{(1)}}$	(0, +)	$(a, +)$	(0, +)	(0, +)	(0, +)	$(a, +)$	(0, +)	(0, +)
$\sigma_{l_i^{(2)}}$	(0, +)	$(a, +)$	(0, +)	(0, +)	(0, +)	$(a, +)$	(0, +)	(0, +)
$\sigma_{l_i^{(3)}}$	(0, 0)	(0, 0)	$(a, -)$	(0, 0)	(0, 0)	(0, 0)	$(a, 0)$	(0, 0)
σ_{l_k}	(0, 0)	(0, 0)	$(a, 0)$	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
σ_{l_j}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	$(a, 0)$

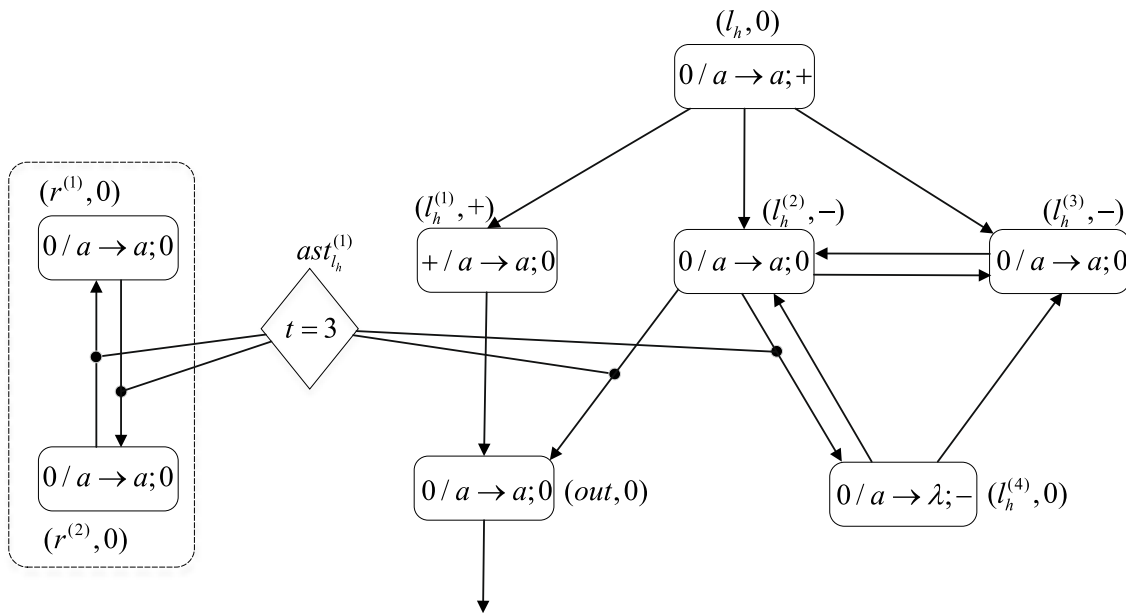


Fig. 3 FIN Module(ending the computation)

which just corresponds to the calculation result in register 1. The operation processes of FIN module are shown in Table 3.

According to the above description of each module and its working principle, it is obvious that the system Π can simulate the calculation process of the register machine M correctly. Therefore, $N_2(\Pi) = N(M)$. The proof is over. \square

Theorem 2 $N_{acc}PASNP = NRE$

Proof In the accepting mode, to prove the computation power of PASN P systems, it is only necessary to prove the inclusion relation $N_{acc}PASNP \supseteq NRE$, which is obtained by the register machine M determined by the simulation of the system Π' . System Π' is improved from system Π in generation mode. The system Π' proof process in the accepting

mode includes an INPUT module, a deterministic ADD module and a SUB module, and the calculation result of the system is stored in the register 1. During the system format conversion process, there is no corresponding SUB instruction for the register 1 to perform the subtraction operation. When system landscape transfer is stopped, only register 1 is non-empty. In the system Π' , each register r has a one-to-one correspondence with a neuron σ_r in the ADD or SUB module, and an instruction l_i in H is associated with a neuron σ_{l_i} . It must be clear that when the register r stores a number whose size is n , then the quantitative relationship contained in the corresponding spike reserve in the neuron σ_r corresponds to $2n$.

The INPUT module of the system is shown in Fig. 4. At step t , the first spike is introduced from outside the system by the input neuron σ_{in} . At step $t + 1$, the spikes on the synapses $(in, in_1), (in, in_2)$ are sensed by $ast_{in}^{(1)}$, the number of spikes is

Table 3 The numbers of spikes and polarity of neurons in fin module (see Fig. 3) during the process of outputting the computational result

Neuron	Step						
	t	$t + 1$	$t + 2$	$t + 3$	\dots	$t + n + 2$	$t + n + 3$
$\sigma_{r(1)}$	$(a^n, 0)$	$(a^n, 0)$	$(a^{n-1}, 0)$	$(a^{n-2}, 0)$	\dots	$(0, 0)$	$(0, 0)$
$\sigma_{r(2)}$	$(a^n, 0)$	$(a^n, 0)$	$(a^{n-1}, 0)$	$(a^{n-2}, 0)$	\dots	$(0, 0)$	$(0, 0)$
σ_{l_h}	$(a, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$
$\sigma_{ast_{l_h}^{(1)}}$	—	—	Inhibitory	Inhibitory	Inhibitory	Excitatory	Excitatory
$\sigma_{l_h^{(1)}}$	$(0, 0)$	$(a, +)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$
$\sigma_{l_h^{(2)}}$	$(0, -)$	$(a, 0)$	$(a, 0)$	$(a, 0)$	$(a, 0)$	$(a, 0)$	$(a, -)$
$\sigma_{l_h^{(3)}}$	$(0, -)$	$(a, 0)$	$(a, 0)$	$(a, 0)$	$(a, 0)$	$(a, 0)$	$(a, -)$
$\sigma_{l_h^{(4)}}$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(0, 0)$	$(a, 0)$	$(a, 0)$
σ_{out}	$(0, 0)$	$(0, 0)$	$(a, 0)$	Output	$(0, 0)$	$(a, 0)$	Output

less than the threshold, and rule $0/a \rightarrow a;+$ in neuron σ_{in} reaches the firing condition, a spike and a positive charge are sent to neurons σ_{in_1} , σ_{in_2} , σ_{in_3} and σ_{in_4} . From step $t + 2$, neurons σ_{in_1} , σ_{in_2} exchange spike information with each other in each subsequent time unit, and neurons σ_{in_2} pass through the corresponding synapses $(in_2, 1^{(1)})$ and $(in_2, 1^{(2)})$ send two spikes to register 1.

At the end of step $t + n$, the second spike enters the system. At step $t + n + 1$, $ast_{in}^{(1)}$ checks that the number of spikes delivered on synapses (in, in_1) , (in, in_2) , (in_1, in_2) , (in_2, in_1) are greater than the threshold in $ast_{in}^{(1)}$, then these spikes are removed from the system. At the same moment, rule $0/a \rightarrow a;+$ within neuron σ_{in} reaches the firing condition and sends a spike and a positive charge to neuron σ_{in_3} and neuron σ_{in_4} . At step $t + n + 2$, rule $+/a^2 \rightarrow \lambda;-$ in neuron σ_{in_3} reaches the firing condition, a negative charge are sent to neuron σ_{in_1} and σ_{in_2} , the neurons are reset to their initial charge state. Rule $+/a^2 \rightarrow a;0$ in neuron σ_{in_4} reaches the

firing condition, a spike and neutral charge are sent to neuron $\sigma_{1^{(1)}}$, $\sigma_{1^{(2)}}$, and σ_{l_0} . At this point, the system will jump to the l_0 instruction. The operation processes of INPUT module of Π' are shown in Table 4.

As shown in Fig. 5, the deterministic ADD module is mainly used to simulate the ADD instruction $l_i : (ADD(r, l_j))$. The operation process of the ADD module will not be described here.

The SUB module of the system Π' will continue to use the module shown in Fig. 2. Regarding the stop of the system Π' , the Fin module is removed from the system, and when the register machine runs to the halting instruction l_h in the accepting mode, there is no rule available in the neuron σ_{l_h} , that is, the system halts.

According to the above proof, the register machine can be correctly simulated by the system Π' under the accepting mode, and the theorem is proved. \square

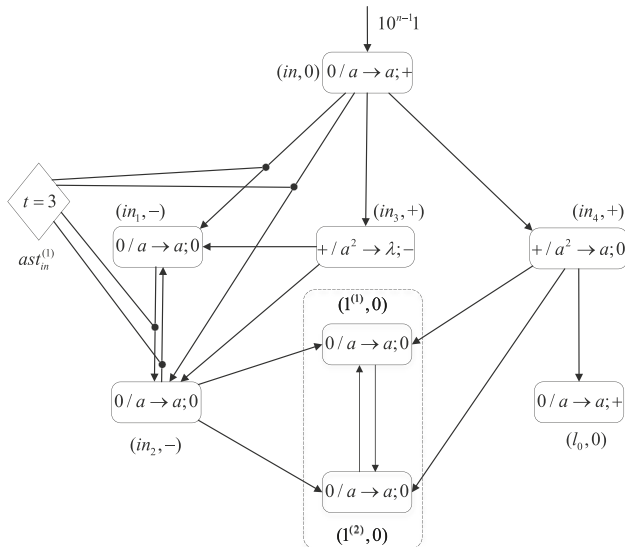


Fig. 4 INPUT Module of Π'

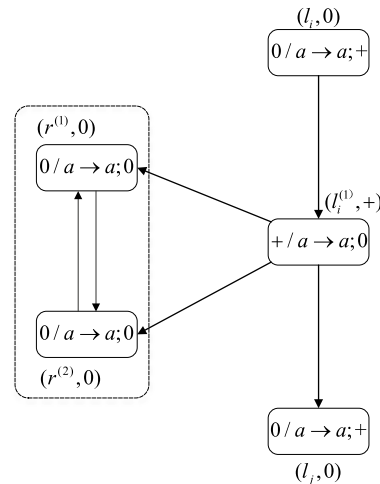


Fig. 5 ADD Module of Π'

Table 4 The numbers of spikes and polarity of neurons in input module (see Fig. 4)

Neuron	Step						
	t	$t + 1$	$t + 2$	\dots	$t + n$	$t + n + 1$	$t + n + 2$
$\sigma_{1^{(1)}}$	(0, 0)	(0, 0)	(a, 0)	\dots	$(a^{n-2}, 0)$	$(a^{n-1}, 0)$	$(a^n, 0)$
$\sigma_{1^{(2)}}$	(0, 0)	(0, 0)	(a, 0)	\dots	$(a^{n-2}, 0)$	$(a^{n-1}, 0)$	$(a^n, 0)$
σ_{in}	(a, 0)	(0, 0)	(0, 0)	(0, 0)	(a, 0)	(0, 0)	(0, 0)
$\sigma_{ast_{in}^{(1)}}$	-	Excitatory	Excitatory	Excitatory	Excitatory	Inhibitory	-
σ_{in_1}	(0, -)	(a, 0)	(a, 0)	(a, 0)	(a, 0)	(0, 0)	(0, -)
σ_{in_2}	(0, -)	(a, 0)	(a, 0)	(a, 0)	(a, 0)	(0, 0)	(0, -)
σ_{in_3}	(0, +)	(a, +)	(a, +)	(a, +)	(a, +)	$(a^2, +)$	(0, +)
σ_{in_4}	(0, +)	(a, +)	(a, +)	(a, +)	(a, +)	$(a^2, +)$	(0, +)
σ_{l_0}	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(a, 0)

4 A small universal PASN P system

In this section, a small universal model of PASN P systems Π'' is constructed as a computational function, which is based on the above research content in this work. The small universal register machine as a calculation function is determined [48], and its structure is shown in Fig. 6.

The universal register machine form is $M_u = (8, H, l_0, l_h, I)$, and the register machine contains eight registers and a total of 23 instructions. The ADD instruction of the register machine M in the accepting or recognition mode is determined, and the ADD instruction is written as $l_i : (ADD(r), l_j)$. According to register 0 in PASN P systems constructed in this paper is used to store data. PASNP systems does not allow subtraction operations acting on the register where the computation result is placed, but instruction l_{19} in Fig. 6 obviously does not meet this requirement, and the register machine M needs to be modified. By adding the register 8, the halting instruction is modified as follows:

$$l_{22} : (SUB(0), l_{23}, l'_h), \quad l_{23} : (ADD(8), l_{22}), \quad l'_h : HLAT$$

The content in register is 0 transferred to register 8 through the above instructions. In the calculation process, the value stored in register 8 only increases. Under the condition that the other definition forms are not changed, the modified register machine M'_u contains 9 registers and 25 instructions, and the result obtained when the system format transfer is terminated will be stored in the register 8.

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

Fig. 6 The small universal register machine M_u

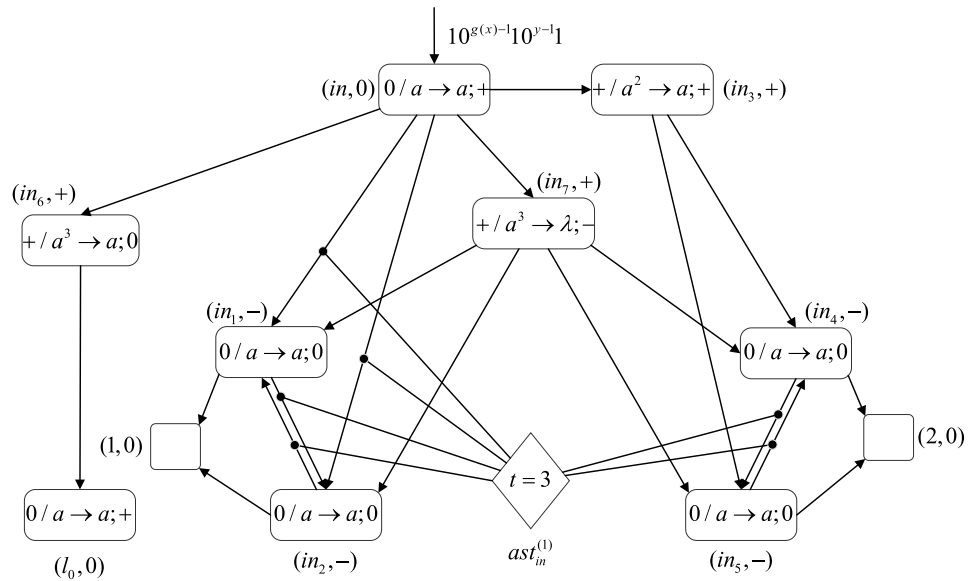
Theorem 3 *There is a universal PASN P system with 65 neurons for computable functions.*

Proof To prove the small universal system Π'' as a function calculation, it is necessary to make it simulate the register machine M'_u . The proof process includes INPUT module, deterministic ADD module, SUB module, ADD–ADD module, ADD–SUB module, SUB–ADD module and OUTPUT module. In the system Π'' , the obtained calculation results are stored in the register 8, and finally output to the outside of the system by the OUTPUT module. In the process of system format conversion, the instruction operation on the register 8 will not reduce its content, that is, there is no corresponding SUB instruction on it to perform the SUB operation. In the system Π'' , each register r has a one-to-one correspondence with a neuron σ_r in the ADD or SUB module and an instruction l_i in H is associated with a neuron σ_{l_i} . It must be clear that when the register r stores a number with a size of n , then the quantitative relationship contained in the corresponding spike reserve in the neuron σ_r corresponds to $2n$.

The INPUT module of the system Π'' as shown in Fig. 7. In the INPUT module, the neuron σ_{in} identifies the spike train $10^{g(x)-1}10^{y-1}1$, and finally $2g(x)$, and $2y$ spikes are stored in the neuron σ_1, σ_2 correspondingly. When neuron σ_{in} receives the first spike in which rule $0/a \rightarrow a;+$ reaches the firing condition, a spike and a positive charge are sent to neuron $\sigma_{in_1}, \sigma_{in_2}, \sigma_{in_3}, \sigma_{in_6}$ and σ_{in_7} . A total of two spikes on the synapses $(in, in_1), (in, in_2)$, are smaller than the threshold in the astrocyte $ast_{in}^{(1)}$ and the spikes can reach the target neuron. At the next moment, rule $0/a \rightarrow a;0$ in the neuron σ_{in_1} and σ_{in_2} reaches the firing condition and transmits a spike and a neutral charge to each other. Moreover, sends two spikes and neutral polarity to neuron σ_1 at each step until neuron σ_{in} receives the second spike, register 1 contains the value $2g(x)$.

When the neuron σ_{in} receives the second spike, astrocytes $ast_{in}^{(1)}$ senses four spikes in adjacent synapses $(in, in_1), (in, in_2), (in_1, in_2), (in_2, in_1)$, that greater than the threshold of astrocytes $ast_{in}^{(1)}$, the synapse is inhibited. At the same time, neuron σ_{in_3} receives the second spike in which rule $+/a^2 \rightarrow a;0$ reaches the firing condition, a spike and a neutral charge are sent to neuron $\sigma_{in_4}, \sigma_{in_5}$. A total of two spikes on the synapses $(in_3, in_4), (in_3, in_5)$, are smaller than the threshold in the astrocyte $ast_{in}^{(1)}$ and the spikes can reach the target neuron. At the next moment, rule $0/a \rightarrow a;0$ in the neuron σ_{in_4} and σ_{in_5} reaches the firing condition and transmits a spike and a neutral charge to each other. Moreover, sends two spikes and neutral charge to neuron σ_2 at each step. When a third spike enters the system, $ast_{in}^{(1)}$ checks that the number of spikes delivered on synapses $(in, in_1), (in, in_2), (in_4, in_5), (in_5, in_4)$ are greater than the threshold in $ast_{in}^{(1)}$, then these spikes are removed from the system. Neuron σ_2

Fig. 7 INPUT Module of Π''



no longer receives spikes and has a number of $2y$ spikes stored in it.

After the third spike enters neuron σ_{in_6} , rule $+/a^3 \rightarrow \lambda; -$ in neuron σ_{in_7} reaches the firing condition, a negative charge are sent to neurons σ_{in_1} , σ_{in_2} , σ_{in_4} and σ_{in_5} , the polarizations of neurons returned to the initial state. Rule $+/a^3 \rightarrow a; 0$ reaches the firing condition, a spike and a neutral charge are sent to neuron σ_{l_0} . At this point, the system will jump to the l_0 instruction of the register machine M'_u .

The deterministic ADD module in the system Π'' is shown in Fig. 5. For the SUB instruction executes the SUB instruction shown in 2. The OUTPUT module is improved on the basis of the module shown in Fig. 3, and the label of neuron σ_r is modified to a certain label $(8, 0)$, other neurons remain unchanged, and the calculation results are output using the improved module.

According to each module of the small universal system Π'' , the small universal system contains a total of 105 neurons. The introduction of system resources is as follows.

- 9×2 neurons correspond to 18 registers,
- 25 neurons correspond to 25 instruction labels,
- 1×10 auxiliary neurons correspond to 10 ADD instruction modules,
- 3×14 auxiliary neurons correspond to 42 SUB instruction modules,
- 5 neurons in the INPUT module,
- 5 neurons in the OUTPUT module,

In view of the above results, it can still be improved through instruction optimization, and the results as follows. For consecutive ADD instructions: $l_{17} : (ADD(2), l_{21})$ and $l_{21} : (ADD(3), l_{18})$, the model constructed after optimization

is recorded as ADD–ADD. In this optimization model, the ADD module corresponding to the l_{21} instruction can be omitted, save 1 neuron. As shown in Fig. 8.

For the continuous ADD and SUB instructions represented as follows, the optimized model is denoted as ADD–SUB, and the form is defined as: $l_i : (ADD(r'), l_g)$, $l_g : (SUB(r''), l_j, l_k)$. After optimization, a total of two neurons are saved. As shown in Fig. 9, the operation process of the improved model will not be described in detail here.

$$l_5 : (ADD(5), l_6), \quad l_6 : (SUB(7), l_7, l_8)$$

$$l_9 : (ADD(6), l_{10}), \quad l_{10} : (SUB(4), l_0, l_{11})$$

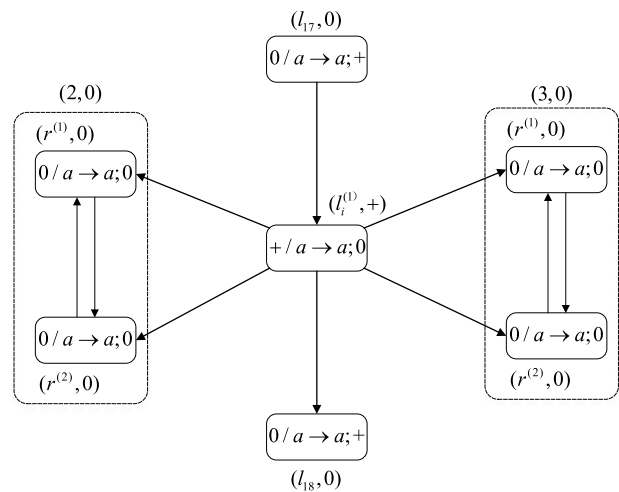


Fig. 8 ADD–ADD Module of Π''

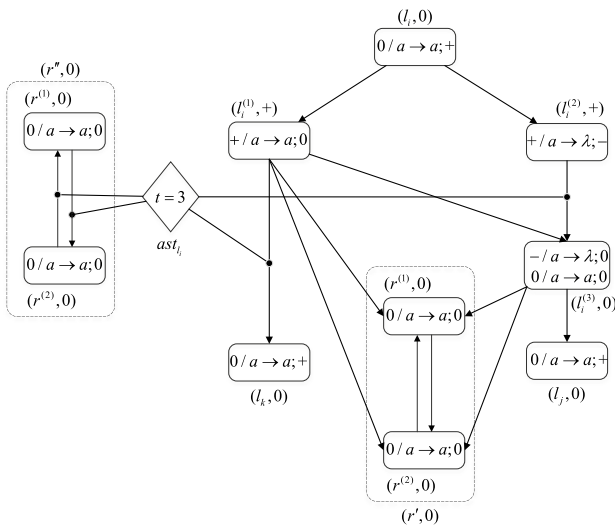


Fig. 9 ADD-SUB Module of Π''

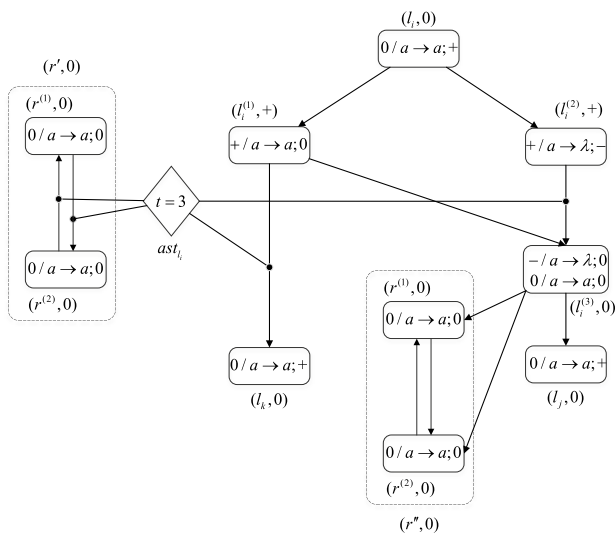
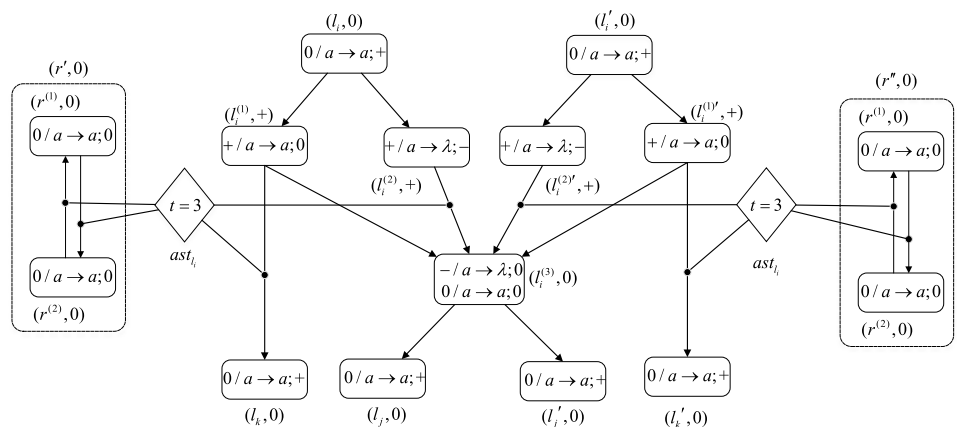


Fig. 10 SUB-ADD Module of Π''

Fig. 11 SUB-SUB Module of Π'' share one neurons



Regarding continuous SUB and ADD instructions, there are six related instructions, which are usually formally defined as $l_g : (SUB(r'), l_j, l_k)$, $l_i : (ADD(r''), l_g)$, the ADD instructions which can be optimized are $l_1, l_5, l_7, l_9, l_{16}, l_{22}$, save a total of six neurons. Continuous SUB and ADD instructions, the optimized model is recorded as SUB-ADD, as shown in Fig. 10.

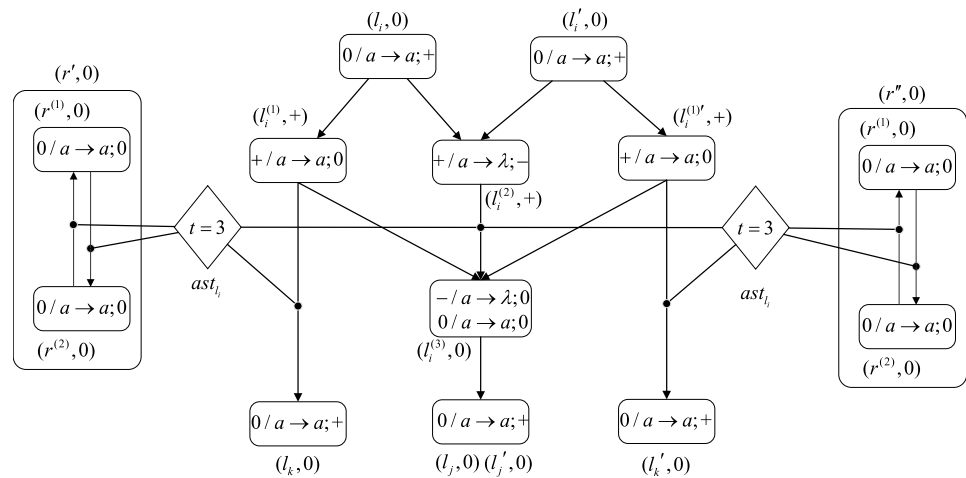
Subtraction instructions can be divided into three categories:

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

For the first set of SUB instructions, an auxiliary neuron can be shared between different SUB instructions, as shown in Fig. 11. For the second and third groups of subtraction SUB, two auxiliary neurons can be shared between different SUB instructions, as shown in Fig. 12. By grouping subtraction instructions. Save a total of 14 neurons.

The proof process of the module will not be repeated, but it should be pointed out that the two modules SUB-ADD and ADD-SUB are both optimization models based on the SUB module. Therefore, by using ADD-ADD, ADD-SUB, SUB-ADD, and SUB-SUB optimization instructions, a total of 23 neurons are saved. Reduced computing resource consumption of small universal systems from 105 to 82. \square

Fig. 12 SUB–SUB Module of Π'' share two neurons



5 Conclusion and discussion

In this work, the control mechanism of astrocytes is added to PSN P systems to construct PASN P systems. As number generating devices and number accepting devices, PASN P systems are proved to be equivalent to Turing machines. In addition, a small universal PASN P system is constructed as a computing function, which is used to simulate the modified small universal register machine M''_u , we demonstrate that the small universal system uses 82 neurons, a reduction of 82 neurons compared to the small universal SN P system with polarization proposed in [43] and a reduction of 69 neurons compared to the small universal SN P system with polarization and rules on synapses proposed in [44]. Therefore, it is of great significance to put forward PASN P systems and prove its Turing universality.

PASN P systems inherit the advantages of ASN P systems and PSN P systems. In applications, the advantage of PASN P systems can also be used to characterize NRE in both synchronous mode and asynchronous mode, or to build basic logic gates and Boolean circuits [42]. PAS P systems have great advantages to solve the real-life problems, such as recognition of handwritten digit letters recognition of handwritten numerals [49], autonomous robot control [50], and information processing and learning [51], that is worth investigating.

Based on PASN P systems, some topics and open problems are suggested. For instance, whether the number of neurons can be reduced by using extended rules or increasing the use of astrocytes. It would also be interesting to design regular expressions that express polarization behavior, or try to incorporate learning strategies into PASN P systems to solving practical engineering problems.

References

1. Păun, G., & Rozenberg, G. (2002). A guide to membrane computing. *Theoretical Computer Science*, 287(1), 73–100.
2. Păun, A., & Păun, G. (2007). Small universal spiking neural P systems. *Biosystems*, 90(1), 48–60.
3. Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences*, 61(1), 108–143.
4. Song, B., Li, K., Orellana-Martín, D., Pérez-Jiménez, M. J., & Pérez-Hurtado, I. (2021). A survey of nature-inspired computing: Membrane computing. *ACM Computing Surveys (CSUR)*, 54(1), 1–31.
5. Pan, L., Orellana-Martín, D., Song, B., & Pérez-Jiménez, M. J. (2020). Cell-like P systems with polarizations and minimal rules. *Theoretical Computer Science*, 816, 1–18.
6. Wu, T., Pan, L., Yu, Q., & Tan, K. C. (2020). Numerical spiking neural P systems. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6), 2443–2457.
7. Wu, T., & Pan, L. (2020). The computation power of spiking neural P systems with polarizations adopting sequential mode induced by minimum spike number. *Neurocomputing*, 401, 392–404.
8. Wu, T., & Jiang, S. (2021). Spiking neural P systems with a flat maximally parallel use of rules. *Journal of Membrane Computing*, 3(3), 221–231.
9. Ibarra, O. H., Păun, A., Păun, G., Rodríguez-Patón, A., Sosík, P., & Woodworth, S. (2007). Normal forms for spiking neural P systems. *Theoretical Computer Science*, 372(2–3), 196–217.
10. Pan, L., & Păun, G. (2010). Spiking neural P systems: An improved normal form. *Theoretical Computer Science*, 411(6), 906–918.
11. Ionescu, M., Păun, G., & Yokomori, T. (2006). Spiking neural P systems. *Fundamenta Informaticae*, 71(2–3), 279–308.
12. Chen, H., Freund, R., Ionescu, M., Păun, G., & Pérez-Jiménez, M. J. (2007). On string languages generated by spiking neural P systems. *Fundamenta Informaticae*, 75(1–4), 141–162.
13. Liu, X., & Ren, Q. (2021). Spiking neural membrane computing models. *Processes*, 9(5), 733.
14. Cabarle, F. G. C., Adorna, H. N., Pérez-Jiménez, M. J., & Song, T. (2015). Spiking neural P systems with structural plasticity. *Neural Computing and Applications*, 26(8), 1905–1917.

15. Peng, H., & Wang, J. (2018). Coupled neural P systems. *IEEE Transactions on Neural Networks and Learning Systems*, 30(6), 1672–1682.
16. Liu, Y., & Zhao, Y. (2022). Spiking neural P systems with membrane potentials, inhibitory rules, and anti-spikes. *Entropy*, 24(6), 834.
17. Wu, T., Zhang, L., Lyu, Q., & Jin, Y. (2022). Asynchronous spiking neural P systems with local synchronization of rules. *Information Sciences*, 588, 1–12.
18. Jiang, S., Liu, Y., Xu, B., Sun, J., & Wang, Y. (2022). Asynchronous numerical spiking neural P systems. *Information Sciences*, 605, 1–14.
19. Aman, B., & Ciobanu, G. (2022). The power of synchronizing rules in membrane computing. *Information Sciences*, 594, 360–370.
20. Song, T., Pan, L., & Păun, G. (2014). Spiking neural P systems with rules on synapses. *Theoretical Computer Science*, 529, 82–95.
21. Zeng, X., Zhang, X., & Pan, L. (2009). Homogeneous spiking neural P systems. *Fundamenta Informaticae*, 97(1–2), 275–294.
22. de la Cruz, R. T. A., Cabarle, F. G. C., Macababayao, I. C. H., Adorna, H. N., & Zeng, X. (2021). Homogeneous spiking neural P systems with structural plasticity. *Journal of Membrane Computing*, 3(1), 10–21.
23. Zhang, X., Pan, L., & Păun, A. (2015). On the universality of axon P systems. *IEEE Transactions on Neural Networks and Learning Systems*, 26(11), 2816–2829.
24. Cabarle, F. G. C., Adorna, H. N., Jiang, M., & Zeng, X. (2017). Spiking neural P systems with scheduled synapses. *IEEE Transactions on Nanobioscience*, 16(8), 792–801.
25. Song, T., Luo, L., He, J., Chen, Z., & Zhang, K. (2014). Solving subset sum problems by time-free spiking neural P systems. *Applied Mathematics & Information Sciences*, 8(1), 327.
26. Ishdorj, T.-O., Leporati, A., Pan, L., Zeng, X., & Zhang, X. (2010). Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources. *Theoretical Computer Science*, 411(25), 2345–2358.
27. Noguchi, T., & Fujiwara, A. (2022). An asynchronous P system with a DPLL algorithm for solving SAT. *International Journal of Networking and Computing*, 12(2), 238–252.
28. Song, B., & Zeng, X. (2021). Solving a PSPACE-complete problem by symport/antiport P systems with promoters and membrane division. *Journal of Membrane Computing*, 3(4), 296–302.
29. Henderson, A., Nicolescu, R., & Dinneen, M. J. (2020). Solving a PSPACE-complete problem with cP systems. *Journal of Membrane Computing*, 2(4), 311–322.
30. Aman, B., & Ciobanu, G. (2021). Travelling salesman problem in tissue p systems with costs. *Journal of Membrane Computing*, 3(2), 97–104.
31. Dong, J., Zhang, G., Luo, B., Yang, Q., Guo, D., Rong, H., Zhu, M., & Zhou, K. (2022). A distributed adaptive optimization spiking neural P system for approximately solving combinatorial optimization problems. *Information Sciences*, 596, 1–14.
32. Tao, C., Yu, W., Wang, J., Peng, H., Chen, K., & Ming, J. (2016). Fault diagnosis of power systems based on triangular fuzzy spiking neural P systems. In *International conference on bio-inspired computing: Theories and applications* (pp. 385–398). Springer.
33. Liu, Y., Chen, Y., Paul, P., Fan, S., Ma, X., & Zhang, G. (2021). A review of power system fault diagnosis with spiking neural P systems. *Applied Sciences*, 11(10), 4376.
34. Peng, H., Wang, J., Pérez-Jiménez, M. J., Wang, H., Shao, J., & Wang, T. (2013). Fuzzy reasoning spiking neural P system for fault diagnosis. *Information Sciences*, 235, 106–116.
35. Lv, Z., Yang, Q., Peng, H., Song, X., & Wang, J. (2021). Computational power of sequential spiking neural P systems with multiple channels. *Journal of Membrane Computing*, 3(4), 270–283.
36. Valencia-Cabrera, L., & Song, B. (2020). Tissue P systems with promoter simulation with MeCoSim and P-Lingua framework. *Journal of Membrane Computing*, 2(2), 95–107.
37. Pérez-Hurtado, I., Orellana-Martín, D., Martínez-del-Amor, M. A., Valencia-Cabrera, L., & Riscos-Núñez, A. (2022). A new P-Lingua toolkit for agile development in membrane computing. *Information Sciences*, 587, 1–22.
38. Buiu, C., Arsene, O., Cipu, C., & Patrascu, M. (2011). A software tool for modeling and simulation of numerical P systems. *Biosystems*, 103(3), 442–447.
39. Siegelmann, H. T., & Sontag, E. D. (1992). On the computational power of neural nets. In: *Proceedings of the fifth annual workshop on computational learning theory* (pp. 440–449).
40. Song, T., Jiang, Y., Shi, X., & Zeng, X. (2013). Small universal spiking neural P systems with anti-spikes. *Journal of Computational and Theoretical Nanoscience*, 10(4), 999–1006.
41. Wu, T., Wang, Y., Jiang, S., & Shi, X. (2016). Small universal spiking neural P systems with homogenous neurons and synapses. *Fundamenta Informaticae*, 149(4), 451–470.
42. Kong, Y., Jiang, K., Chen, Z., & Xu, J. (2014). Small universal spiking neural P systems with astrocytes. *Science and Technology*, 17(1), 19–32.
43. Wu, T., Păun, A., Zhang, Z., & Pan, L. (2017). Spiking neural P systems with polarizations. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8), 3349–3360.
44. Jiang, S., Fan, J., Liu, Y., Wang, Y., & Xu, F. (2020). Spiking neural P systems with polarizations and rules on synapses. *Complexity* 2020.
45. Volterra, A., & Meldolesi, J. (2005). Astrocytes, from brain glue to communication elements: The revolution continues. *Nature Reviews Neuroscience*, 6(8), 626–640.
46. Păun, G. (2010). *The Oxford Handbook of Membrane Computing*. New York: Oxford University Press.
47. Cavaliere, M., Ibarra, O. H., Păun, G., Egecioglu, O., Ionescu, M., & Woodworth, S. (2009). Asynchronous spiking neural P systems. *Theoretical Computer Science*, 410(24–25), 2352–2364.
48. Korec, I. (1996). Small universal register machines. *Theoretical Computer Science*, 168(2), 267–301.
49. Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12), 3207–3220.
50. Trhan, P. (2010). The application of spiking neural networks in autonomous robot control. *Computing and Informatics*, 29(5), 823–847.
51. Khademian, F., Khanbabaie, R., & Babol, I. (2015). Practical applications of spiking neural network in information processing and learning. *AcSIJ Advances in Computer Science: An International Journal*, 4(4), 133–137.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Suxia Jiang received the Master's degree in Detection Technology and Automatic Equipment from Zhengzhou University of Light Industry, Zhengzhou, China, in 2008. She is currently a Professor with the School of Electrical and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou, China. Her research interests include membrane computing, spiking neural networks, and artificial intelligence.



Xiaoliang Zhu received the bachelor's degree in Business and Management from Zhengzhou Shengda University, Xinzheng, China, in 2021. He is currently studying for his master's degree in Zhengzhou University of Light Industry, Zhengzhou, China. His research direction is spiking neural P systems.



Zhichao Shen received the bachelor's degree from Xuchang University, Xuchang, China, in 2021. He is currently studying for his master's degree in Zhengzhou University of Light Industry, Zhengzhou, China. His research direction is Path planning research and Spiking Neural P Systems.



Tao Liang received the bachelor's degree from Huanghe University of Science and Technology, Zhengzhou, China, in 2020. He is currently studying for his master's degree in Zhengzhou University of Light Industry, Zhengzhou, China. His research direction is cell-like membrane systems.



Bowen Xu received the bachelor's degree from Henan Polytechnic University, Jiaozuo, China, in 2019. He is currently studying for his master's degree in Zhengzhou University of Light Industry, Zhengzhou, China. His research direction is numerical spiking neural P systems.