Journal of Membrane Computing (2022) 4:56–67
https://doi.org/10.1007/s41965-021-00088-w

REGULAR PAPER

Universality of spiking neural P systems with polarizations working in sequential mode induced by maximum spike number

Li Liu[1,2] · Keqin Jiang[1,2]

Received: 5 July 2021 / Accepted: 21 October 2021 / Published online: 11 November 2021

Abstract
Based on the inspiration that the communication signal between neurons in biology is composed of short electrical pulses, we investigate a new variant of spiking neural P systems, i.e., spiking neural P systems with polarizations (PSN P systems), which have a rule-triggering condition associated with polarization. In this work, we focus on the computational power of sequential PSN P systems with delay based on the maximum number of spikes, i.e., the ability to preferentially fire the neuron with the maximum number of spikes among the active neurons at each step (except for the neurons in the refractory period) of the computation. Thus, two strategies are considered, i.e., the max-sequentiality strategy and the max-pseudo-sequentiality strategy, and we prove that PSN P systems with delay adopting the max-sequentiality strategy or the max-pseudo-sequentiality strategy are Turing universal as number generating devices. The results give positive answers to the open problem formulated in [Tingfang Wu et al. (2020), Neurocomputing, 401, 392–404].

Keywords Bio-inspired computing · Membrane computing · Spiking neural P system · Sequentiality · Universality

1 Introduction

The human brain is a complex "computing" system, but also the source of inspiration for the generation of efficient computational models and algorithms. Especially neural computation, it simulates the working mechanism of the human brain and proposes an intelligent computation model through the study of the structure and function of the human brain nervous system, information processing, memory storage, etc [9]. It is well known that membrane computing has become popular research in the computing area. In particular, spiking neural P systems are also evaluated as the third generation neural network model [7, 18–20].

Membrane computing is a new branch of biological computing [27]. It mainly studies how to build a distributed parallel computing model from the structure and function of cells, referred to as P systems, and analyzes the computing power of this computing model (whether it is equivalent to the Turing machine) and computational efficiency (whether it is possible to solve difficult computation problems in an effective way). There are three types of P systems: cell-like P systems, tissue-like P systems, and neuron-like P systems, which specific details, readers can refer to [28–30] for more contents on membrane computing.

And then, spiking neural P systems (SN P systems) are a special class of neural-like P systems, which are inspired by the process mechanism that neurons communicate with each other through the same electrical pulse to transmit information [11]. SN P systems employ time series to encode information and consist of three elements: neurons, synapses, and rules to form a network structure of directed graphs (directed arcs represent synapses). The rules include firing rules and forgetting rules, the applicability of the rules is to define the triggering of regular expressions related to the rules by the number of spikes in the neuron.

In addition, SN P systems have been extensively studied in the field of computer science. In terms of theoretical research, the computational performance of various extended models is discussed and analyzed, such as: universal or small universal computing systems [6, 14, 21–26,

✉ Keqin Jiang
jiangkq0519@163.com

Li Liu
lliu150@yeah.net

[1] The University Key Laboratory of Intelligent Perception and Computing of Anhui Province, Anqing Normal University, Anqing 246133, Anhui, China

[2] School of Computer and Information, Anqing Normal University, Anqing 246133, Anhui, China

33–35, 41, 47, 51], language production ability [1, 15, 46, 48, 52], design optimization algorithm [50, 54], etc. In terms of practical applications [5, 49], the SN P system and its variants have also been widely used, such as: pattern recognition [4, 36, 37], fuzzy SN P systems and fault diagnosis of power systems [31, 32, 38–40], computational biology [2], solving computational hard problems [12, 16, 53], performing arithmetic and logical operations, and hardware implementation [3, 8, 17], etc.

SN P systems and its variant models have a very robust computing potential to describe the characteristics of recursively enumerable languages. In recent years, scholars have made significant progress in studying different variants of spiking neural P systems. Especially, in 2018, Tingfang Wu et al. were inspired by the process of depolarization and repolarization in the generation and transmission of nerve impulses. To achieve a more free and regular system, they propose a new mechanism, i.e., using the −, 0, and + charges associated with the neuron to control the applicability of the rule, instead of relying solely on the spike number for rule application, thus avoiding the NP-complete problem of using regular expressions to determine the applicability of rules [44]. The study is known as called the spiking neural P systems with polarizations (PSN P systems for short). After that, Tingfang Wu et al. made a more recent study on PSN P systems, and some new variants were proposed successively [13, 43, 45]. At present, there are still relatively few studies on sequential PSN P systems and only the research of sequential PSN P systems based on the minimum number of spikes that work in the number generation mode [42].

In this work, we will solve the problem raised in the literature [42]. Therefore, we concentrate on the computational power of the PSN P systems with a maximal sequential strategy, where sequentiality is determined by the maximum number of spikes, the delay, and the polarization in the neuron. During module construction, since we are studying a sequential strategy based on the maximum number of spikes, the biggest challenge of this system is when the spike number of some or some neurons reaches the system's maximum consecutively, which then leads to the continuous non-stop firing of that neuron. Therefore, to control the successive excitation of neurons, we added a delay method, which makes it easier to use fewer neurons during the construction of each module. In each step of the system computation, if the active neurons have more than one neuron with the same maximum number of spikes, we should consider two strategies: (1) max-sequentiality (non-deterministically selecting an active neuron that satisfies the condition to start firing); (2) max-pseudo-sequentiality (all active neurons that satisfy the condition fire at the same time).

Furthermore, the spike code in a given neuron is defined as the result of the computation of the system, and it is shown that the sequential PSN P systems based on the delay of the maximum number of spikes are Turing universal as the number generation devices.

The innovations of this research are as follows:

1. The PSN P system with delay mainly solves a problem that constructs the subtractive modules employing the max-sequentiality strategy and max-pseudo-sequentiality strategy proposed in the literature [42].
2. The computational power of PSN P systems with delay is Turing universal regardless of employing the max-sequentiality strategy or max-pseudo-sequentiality strategy as the number generating modes.

The rest of the work is as follows. The Sect. 2 will introduce the formal definition of PSN P systems with delay and the definition of the computation results. Sections 3 and 4 show that PSN P systems with delay have Turing universality when employing two strategies (max-sequentiality strategy and max-pseudo-sequentiality strategy) as the number generation devices. Finally, in Sect. 5, conclusions and open problems are formulated.

2 Spiking neural P systems with polarizations

This section will briefly review the definition of PSN P systems with delay and their applications, and presents the computational results, concepts, and related notations for PSN P systems based on maximum sequential strategies [10, 44].

Formally, a PSN P system with delay has the following structure

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, out),$$

where

(1) $O = \{a\}$ refers to the singleton alphabet, and a denotes spike;

(2) $\sigma_i = (\alpha_i, n_i, R_i)$, $1 \le i \le m$, $(m \ge 1)$ denotes the system with m neurons, there into:

   (a) $\alpha_i \in \{+, 0, -\}$ refers to the initial polarization of neuron σ_i (here polarization can also be referred to as charge);
   (b) $n_i \ge 0$ is the initial number of spikes contained in neuron σ_i;
   (c) R_i is a finite set consisting of rules having the following two forms:

① spiking rules: $\alpha/a^c \to a;\beta;t$, for $\alpha, \beta \in \{+, 0, -\}$, $c \ge 1$ and $t \ge 0$; ② forgetting rules: $\alpha'/a^s \to \lambda;\beta'$, for

α', $\beta' \in \{+, 0, -\}$, $s \geq 1$; If both rules exist in R_i then restrict $\alpha' \neq \alpha$.

(3) $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ denotes the set of synaptic connections between neurons, where $i \neq j$ for each $(i, j) \in syn, 1 \leq i, j \leq m$;

(4) $out \in \{1, 2, \ldots, m\}$ stands for the output neuron.

The following is an explanation of the application of the rules:

① a spiking rule $\alpha/a^c \rightarrow a;\beta;t$: if the current polarity of the neuron σ_i is α and it has k spikes, meeting $k \geq c$, assuming the rule can be triggered in step q and spiking in step $q + t$. That is, if $t = 0$, the rule is immediately fired and generating a spiking with β charge. If $t \geq 1$, the generated spike with a charge will leave the neuron σ_i at the $q + t$ step. In the interval between using the rule and releasing the spike, the neuron σ_i is closed (i.e.,neurons can only generate action potentials when they reach a certain threshold during periods of relative refractoriness) and this means another neuron emits a spike with β charge at any moment of $q, q + 1, \cdots, q + t - 1$, then its spike and charge will not be transmitted to the neuron which has used $\alpha/a^c \rightarrow a;\beta;t$ rule in step q, so it cannot receive further spikes, and certainly cannot fire again. When the neuron is in the open moment, that neuron can receive new spikes and charges.

② forgetting rule $\alpha'/a^s \rightarrow \lambda;\beta'$: if the neuron σ_i has exactly s spikes and the polarity is α', the forgetting rule being used, which indicates that s spikes in neuron σ_i will be consumed. Then a charge of β' will be sent immediately to all adjacent neurons that have synaptic connections.

By definition, if a certain neuron of the system can use one or more rules, the neuron selects one of the enabled rules in a nondeterminate way for application. Meanwhile, when the spiking rules are available, no forgetting rules are available, and vice versa.

The neuron σ_{out} represents the output neuron, and the spikes and charges generated by the output neuron will be sent to the environment. Assuming the moment when the output neuron transmits spikes and charges to the environment is marked as 1, and vice versa (the moment when no spikes and charges are generated) is marked as 0, the resulting binary sequence is also the spike train generated by the system.

Spiking neural P systems with polarizations are an application of the rule determined jointly by two conditions: polarization and number of spikes. Moreover, the difference with the application of the regular expression rule is that the neuron in this system receives (sends) not only spikes but also charges, especially in the application of the forgetting rule (as detailed in the application of the forgetting rule definition above). So, when a neuron receives a charge from a neighboring neuron, a charge response occurs and the charge response is without any time consumption, while the polarization (i.e.,0, +, and −) response of that neuron has three cases:

(i) Some positive or some negative charges neutralization response results in the equivalent of one positive or one negative charge, and the polar response between neutral charges remain unchanged. This neutralization reaction has the highest priority level;

(ii) a positive charge and a negative charge interact and the neutralization reaction results in a neutral charge;

(iii) any kind of charge (a positive charge or a negative charge) interacts with a neutral charge, and the polarity is still itself;

In sequential PSN P systems induced the maximum number of spikes and the delay, if there are multiple active neurons in each step of the system, only the active neuron (i.e., neurons are in the open state) with the largest number of spikes can use the corresponding rules. If there is more than one neuron with the largest number of spikes in the active neuron, then the following two strategies can be considered: (1) the max-sequentiality-strategy: non-deterministically select an active neurons to fire; (2) the max-pseudo-sequentiality-strategy: neurons that meet the condition start to fire simultaneously.

For the PSN P system Π defined above, its configuration is determined by the number of spikes contained in the neuron and the polarity of the neuron. At the beginning of the calculation, the initial configuration of the system Π can be expressed as $C_0 = \langle \alpha_1/n_1, \alpha_2/n_2, \cdots, \alpha_m/n_m \rangle$. Through the use of the above rules, we pass from a configuration of the system to another configuration, the process is called the transition of the system configuration. The transition from configuration C_i to configuration C_j is denote by $C_i \Longrightarrow C_j$ in Π. And the computation of the system is also the sequence of transitions of the system from the initial configuration. If the system reaches a certain configuration and no rules are available, the computation is said to stop.

For the system computation in this paper, we use the time distance between the first two spikes delivered by the output neuron to the environment. Specifically, when the system is working in the generation mode, the computation result is encoded in the form of $t_2 - t_1 - 1$, where t_1 and t_2 are the moments when the neuron σ_{out} outputs the first two spikes. The set of all numbers calculated in this way can be represented by $N_2(\Pi)$ (the subscript 2 indicates the time interval between the first two consecutive spikes output to the environment).

When a number can be computed in a limited way, it is called Turing computability. The Turing computable number set family is represented by NRE. $N_2^{\gamma} PSNP_*^n(polar_p)$ is used

to represent the set of PSN P systems with delay generation numbers under the max-sequentiality or max-pseudo-sequentiality strategy, where the subscript 2 indicates the computation result, $\gamma \in \{ms, mps\}$ (ms indicates the max-sequentiality strategy used by the system, while mps denotes the max-pseudo-sequentiality strategy), $*$ is the number of neurons and each one of them having at most n rules ($n \leq 2$), p stands for polarity ($p \leq 3$).

3 The universality of PSN P systems with delay using the max-sequentiality strategy

This section mainly demonstrates that the PSN P system Π (with delay) using the max-sequentiality strategy as the number generating device is Turing universal by building the corresponding module.

Theorem 1 $N_2^{ms}PSNP_*^2(polar_3) = NRE$.

Proof We just need to prove that $NRE \subseteq N_2^{ms}PSNP(polar_3)$ and the inverse inclusion relation is not repeated here and can be found in [28]. The proof of generality is simulating the register to achieve, i.e., it can compute all Turing-computable number sets (equivalent to inscribing the language length set cluster *NRE*).

A register is a five-tuple $M = (m, H, l_0, l_h, I)$, each identifier has a different role, i.e., the m is the number of registers, H represents the set of instruction tags, l_0 is the initial instruction, and l_h is termination instruction (to terminate the computation of the register machine, used to mark the instruction HALT), I represents the instruction set, and a label in H uniquely marks an instruction in I. And there are three forms of I:

- ADD instructions: l_i : (ADD(r), l_j, l_k): add 1 to the value stored in the register r, and then randomly select and execute the instruction l_j or l_k;
- SUB instructions: l_i : (SUB(r), l_j, l_k): if the value stored in the register r is non-zero, subtract the number by 1, and then execute the instruction l_j; if the number in the register r is zero, Then execute the instruction l_k;
- halt instruction: l_h : HALT.

Before the system computes, all registers are empty (that is, the storage number is 0). The register machine first executes the initial instruction l_0, and then runs according to the above-mentioned instructions mode, executing one instruction per step. When the register machine executes the termination instruction l_h, it indicates that computation stops. The value stored in register 1 is the number generated by the register machine M, and the set of all numbers n is denoted as $N(M)$. Suppose that in the shutdown mode, all the other registers except register 1 in M are empty, and during the computation process, the content of register 1 only increases. Next, we construct the ADD, SUB, and Output modules of the PSN P system Π (with delay) that uses the max-sequentiality strategy to simulate register machine M.

For each register r in the register machine M, there is a neuron σ_r corresponding to it in the system Π. Specifically, we divide the number of spikes stored in the register into two categories, one when the register $r = 1$, which contains $2n$ spikes, and the other when the register $r \neq 1$, which then has $2n + 2$ spikes in the neuron σ_r (if $n = 0$, it indicates that the register r is empty). For each instruction label l_i in the register machine M, the neuron σ_{l_i} corresponds to it. In the computation process, if the neuron σ_{l_i} receives a spike with a charge and reaches the firing condition, it starts to simulate the instruction l_i : (OP(r), l_j, l_k) in M: through OP(r) (the register r performs addition or subtraction operations), finally fires the neuron σ_{l_j} or neuron σ_{l_k}. When the termination instruction l_h in the corresponding M fire, the system Π completes the computation of the simulated register M. Simultaneously, the output neuron fire exactly twice, and the result of the time interval between the two firings is stored in register 1.

Module ADD: simulation on an ADD instruction l_i : (ADD(r), l_j, l_k).

In the case of the simulated ADD instructions l_i : (ADD(r), l_j, l_k), Figure 1 shows the initial state of the neuron in the ADD module. Suppose that in step t, the neuron σ_{l_i} receives a spike and starts to simulate the ADD instruction l_i : (ADD(r), l_j, l_k).

Then, neurons σ_{l_i} and $\sigma_{l_i^{(1)}}$ fire in turn and the number of spikes in neuron σ_r will be increased by two spikes, i.e., it

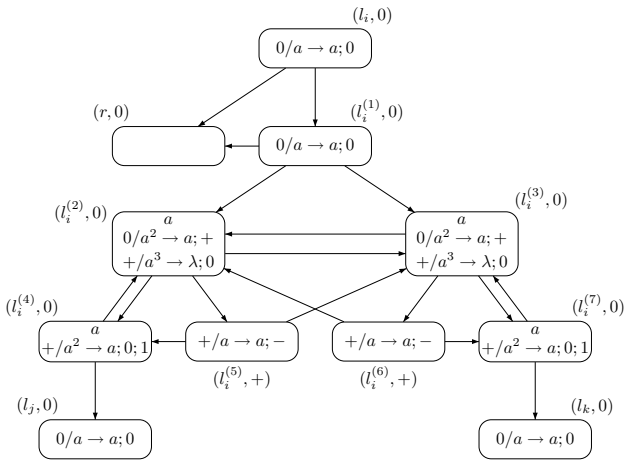

Fig. 1 ADD module of Π to simulate instruction l_i : (ADD(r), l_j, l_k)

simulates the operation of adding 1 to the number in register r.

In step $t+2$, neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ become active after accumulating two spikes. Therefore, according to the max-sequentiality strategy, a randomly selected neuron will be executed.

Case I: If the system selects to use the rule $0/a^2 \to a;+$ of the neuron $\sigma_{l_i^{(2)}}$ in the $t+2$ step, then the neuron $\sigma_{l_i^{(2)}}$ consumes two spikes and sends a spike and a positive charge to the neuron $\sigma_{l_i^{(3)}}$, $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$.

In step $t+3$, the neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(4)}}$ have three and two spikes respectively, both are firing. According to the max-sequentiality strategy, neuron $\sigma_{l_i^{(3)}}$ first uses the spiking rule $+/a^3 \to \lambda;0$, and consumes three spikes. In step $t+4$, the neuron $\sigma_{l_i^{(4)}}$ fires with one step delay. When the neuron $\sigma_{l_i^{(5)}}$ fires in the next step, at the same time neuron $\sigma_{l_i^{(4)}}$ is in the open state and neuron $\sigma_{l_i^{(5)}}$ sends a spike with a negative charge to neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(4)}}$. Meanwhile the neuron $\sigma_{l_i^{(4)}}$ is sending a spike to neuron $\sigma_{l_i^{(2)}}$. Thus, neurons $\sigma_{l_i^{(2)}}$, $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(5)}}$ return to their initial values.

In step $t+6$, the neuron σ_{l_j} receives a spike and a neutral charge from the neuron $\sigma_{l_i^{(4)}}$, neuron σ_{l_j} is in the fired state, and the system will simulate the instruction l_j in the register machine M.

Case II: If the system selects to use the rule $0/a^2 \to a;+$ in the neuron $\sigma_{l_i^{(3)}}$ in step $t+2$, then fires the module associated with the instruction l_k, and finally the neuron σ_{l_k} receives a spike and starts the instruction l_k simulation, and this process is no further elaboration. After the instruction is simulated, all auxiliary neurons in the ADD module are restored to the initial spike number and polarization state, which allows the module to be reused for a long enough time during the simulation of the register machine.

The simulation of the ADD module, as shown in Tables 1 and 2, details the content of the different neurons in the above two cases.

Module SUB: simulation on a SUB instruction l_i : $(\mathtt{SUB}(r), l_j, l_k)$.

The SUB module is shown in Figure 2. Note that in the SUB module, the content of register 1 of the subtraction computation results in the register machine M only increases. Therefore, the corresponding neuron σ_r in the SUB module has $r \neq 1$.

Suppose that at step t, the system starts to simulate the SUB instruction l_i : $(\mathtt{SUB}(r), l_j, l_k)$ in the register M, such that the neurons σ_{l_i}, $\sigma_{l_i^{(1)}}$, and $\sigma_{l_i^{(2)}}$ fire sequentially. In step $t+3$, neurons $\sigma_{l_i^{(3)}}$, $\sigma_{l_i^{(4)}}$, and σ_r have two, four, and $2n+3$ spikes, respectively, all in the excitable state. Depending on the number of spikes in neuron σ_r, the following two cases are discussed.

Case I: If the number of spikes in the neuron σ_r contains $2n+3$ $(n>0)$ (the corresponding register r was originally non-empty). According to the max-sequentiality strategy, then in step $t+3$, neuron σ_r fires preferentially, using the

Table 1 Evolution of the polarization and the number of spikes per neuron during the simulation of the instruction l_j in the ADD module

Step	Neuron						
	σ_{l_i}	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	σ_{l_j}
t	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(+,0)	(0,0)
$t+1$	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(+,0)	(0,0)
$t+2$	(0,0)	(0,0)	(0,2)	(0,2)	(0,1)	(+,0)	(0,0)
$t+3$	(0,0)	(0,0)	(0,0)	(+,3)	(+,2)	(+,1)	(0,0)
$t+4$	(0,0)	(0,0)	(0,0)	(+,0)	(+,2)	(+,1)	(0,0)
$t+5$	(0,0)	(0,0)	(0,0)	(+,0)	—	(+,1)	(0,0)
$t+6$	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(+,0)	(0,1)

Table 2 Evolution of the polarization and the number of spikes per neuron during the simulation of the instruction l_k in the ADD module

Step	Neuron						
	σ_{l_i}	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(7)}}$	$\sigma_{l_i^{(6)}}$	σ_{l_k}
t	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(+,0)	(0,0)
$t+1$	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(+,0)	(0,0)
$t+2$	(0,0)	(0,0)	(0,2)	(0,2)	(0,1)	(+,0)	(0,0)
$t+3$	(0,0)	(0,0)	(+,3)	(0,0)	(+,2)	(+,1)	(0,0)
$t+4$	(0,0)	(0,0)	(+,0)	(0,0)	(+,2)	(+,1)	(0,0)
$t+5$	(0,0)	(0,0)	(+,0)	(0,0)	—	(+,1)	(0,0)
$t+6$	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(+,0)	(0,1)

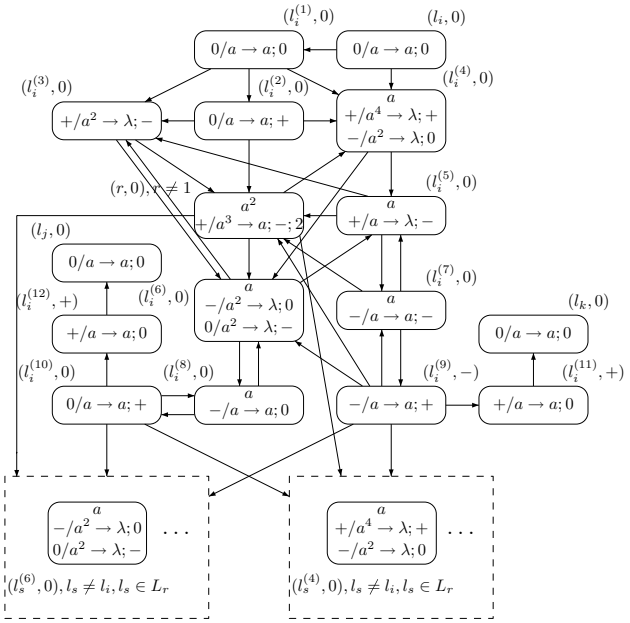

Fig. 2 SUB module of Π to simulate instruction l_i : $(\mathtt{SUB}(r), l_j, l_k)$, $r \neq 1$, where $L_r = \{l \mid l$ is a label of a SUB instruction operating on register $r\}$

rule $+/a^3 \to a; -;2$, and due to the delay used, neuron σ_r is only firing and closed (i.e., it cannot release and receive spikes and charges) during the two-step delay.

In step $t+4$, since the neuron $\sigma_{l_i^{(4)}}$ is the one that currently has the maximum number of spikes four, it fires preferentially, consumes four spikes itself, and sends a positive charge to neurons $\sigma_{l_i^{(5)}}$ and $\sigma_{l_i^{(6)}}$.

In step $t+5$, the neuron $\sigma_{l_i^{(3)}}$ fires, while at the same time neuron σ_r returns to the open state, and both neuron $\sigma_{l_i^{(3)}}$ and σ_r will send a charge or a spike with a charge to the neighboring neurons connected by the synapse, respectively.

In step $t+6$, neuron σ_r will receive a negative charge transmitted from neuron $\sigma_{l_i^{(3)}}$, so that the positive and negative charges in neuron σ_r cancel each other and eventually become neutral, containing $2n$ spikes (i.e., the number of stores in register r minus 1), in the unexcitable state. The neurons $\sigma_{l_i^{(4)}}$ reverts to the initial spike and polarity state because they receive a spike with negative charge transmitted by neuron σ_r. According to the rules for the use of electric charge, neuron $\sigma_{l_i^{(6)}}$ has two spikes and is neutral in polarity. In the current system only neurons $\sigma_{l_i^{(6)}}$ and $\sigma_{l_i^{(5)}}$ are active and by the max-sequentiality strategy, neuron $\sigma_{l_i^{(6)}}$ fires preferentially and using the forgetting rule, a negative charge is sent to neurons $\sigma_{l_i^{(3)}}$, $\sigma_{l_i^{(5)}}$ and $\sigma_{l_i^{(8)}}$ and neurons $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(5)}}$will revert to the initial state of the system.

Next, neurons $\sigma_{l_i^{(8)}}$, $\sigma_{l_i^{(10)}}$ and $\sigma_{l_i^{(12)}}$ fire sequentially, and finally a spike with a neutral charge is sent to neuron σ_{l_j} and the system starts to simulate instruction l_j in the register.

Case II: If neuron σ_r has three $(n=0)$ spikes (i.e., corresponding register r turns out to be empty), according to the max-sequentiality strategy, then in step $t+3$, the neuron $\sigma_{l_i^{(4)}}$ is the one that currently has the maximum number of spikes four, so it fires preferentially, consumes four spikes itself, and sends a positive charge to neurons $\sigma_{l_i^{(5)}}$ and $\sigma_{l_i^{(6)}}$.

In step $t+4$, neuron σ_r fires preferentially, using the rule $+/a^3 \to a; -;2$. Due to the delay used, neuron σ_r σ_r is firing and closed (i.e., it cannot release and receive spikes and charges) during the two-step delay.

In step $t+5$, neuron $\sigma_{l_i^{(3)}}$ fires, and since neuron σ_r is in a closed state at this point, a negative charge sent by neuron $\sigma_{l_i^{(3)}}$ to neuron σ_r will be lost. Simultaneously, the polarity of neuron $\sigma_{l_i^{(6)}}$ changes to neutral.

In step $t+6$, neuron $\sigma_{l_i^{(5)}}$ fires and sends a negative charge to neuron $\sigma_{l_i^{(3)}}$, $\sigma_{l_i^{(7)}}$ and σ_r. Simultaneously, neuron σ_r returns to the open state and neuron σ_r will receive a negative charge transmitted from neuron $\sigma_{l_i^{(5)}}$, thus the positive and negative charges in neuron σ_r cancel each other and eventually become neutral.

In step $t+7$, as neuron $\sigma_{l_i^{(6)}}$ and $\sigma_{l_i^{(4)}}$ receive a spike with negative charge from neuron σ_r, the neuron $\sigma_{l_i^{(4)}}$ returns to its initial state. The neuron $\sigma_{l_i^{(6)}}$ contains two spikes and a negative charge, so neuron $\sigma_{l_i^{(6)}}$ fires and consumes two spikes itself.

Next, neurons $\sigma_{l_i^{(7)}}$, $\sigma_{l_i^{(9)}}$ and $\sigma_{l_i^{(11)}}$ fire sequentially and and sends charged spikes to synaptically connected neighboring neurons, so that neurons σ_r and $\sigma_{l_i^{(6)}}$ eventually return to the initial values of this simulation (the corresponding register r is empty). Finally, the neuron σ_{l_k} receives a spike and the system Π starts to simulate the instruction l_k in the register machine M.

The specific SUB module details are shown in Tables 3 and 4.

The following analyzes the interaction between the ADD module and the SUB module. For a certain register r, there may be multiple ADD instructions and SUB instructions acting on the same register r. It can be seen from the above module description that in the ADD module, the neuron σ_r will receive two spikes each time and will not fire the rules in it. Therefore, there is no mutual influence between the ADD modules and the ADD and SUB modules.

Among multiple SUB modules, suppose there are several SUB instructions l_s acting on the register r, then when the SUB instruction l_i : $(\mathtt{SUB}(r), l_j, l_k)$ is simulated, the neuron σ_r uses the spiking rule, send a spike and a negative charge

Table 3 Evolution of polarization and number of spikes in the SUB module when the number of spikes of the neuron $sigma_r$ is $n \neq 0$

Step	Neuron											
	σ_{l_i}	σ_r	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	$\sigma_{l_i^{(6)}}$	$\sigma_{l_i^{(8)}}$	$\sigma_{l_i^{(10)}}$	$\sigma_{l_i^{(12)}}$	σ_{l_j}
t	(0,1)	(0,2n + 2)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(0,0)	(+,0)	(0,0)
$t+1$	(0,0)	(0,2n + 2)	(0,1)	(0,0)	(0,0)	(0,2)	(0,1)	(0,1)	(0,1)	(0,0)	(+,0)	(0,0)
$t+2$	(0,0)	(0,2n + 2)	(0,0)	(0,1)	(0,1)	(0,3)	(0,1)	(0,1)	(0,1)	(0,0)	(+,0)	(0,0)
$t+3$	(0,0)	(+,2n + 3)	(0,0)	(0,0)	(+,2)	(+,4)	(0,1)	(0,1)	(0,1)	(0,0)	(+,0)	(0,0)
$t+4$	(0,0)	—	(0,0)	(0,0)	(+,2)	(+,4)	(0,1)	(0,1)	(0,1)	(0,0)	(+,0)	(0,0)
$t+5$	(0,0)	—	(0,0)	(0,0)	(+,2)	(+,0)	(+,1)	(+,1)	(0,1)	(0,0)	(+,0)	(0,0)
$t+6$	(0,0)	(0,2n)	(0,0)	(0,0)	(+,0)	(0,1)	(+,1)	(0,2)	(0,1)	(0,0)	(+,0)	(0,0)
$t+7$	(0,0)	(0,2n)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,0)	(−,1)	(0,0)	(+,0)	(0,0)
$t+8$	(0,0)	(0,2n)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(−,0)	(0,1)	(+,0)	(0,0)
$t+9$	(0,0)	(0,2n)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(0,0)	(+,1)	(0,0)
$t+10$	(0,0)	(0,2n)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(0,0)	(+,0)	(0,1)

Table 4 Evolution of polarization and number of spikes in the SUB module when the number of spikes of the neuron $sigma_r$ is $n = 0$

Step	Neuron											
	σ_{l_i}	σ_r	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	$\sigma_{l_i^{(6)}}$	$\sigma_{l_i^{(7)}}$	$\sigma_{l_i^{(9)}}$	$\sigma_{l_i^{(11)}}$	σ_{l_k}
t	(0,1)	(0,2)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+1$	(0,0)	(0,2)	(0,1)	(0,0)	(0,0)	(0,2)	(0,1)	(0,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+2$	(0,0)	(0,2)	(0,0)	(0,1)	(0,1)	(0,3)	(0,1)	(0,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+3$	(0,0)	(+,3)	(0,0)	(0,0)	(+,2)	(+,4)	(0,1)	(0,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+4$	(0,0)	(+,3)	(0,0)	(0,0)	(+,2)	(+,0)	(+,1)	(+,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+5$	(0,0)	—	(0,0)	(0,0)	(+,2)	(+,0)	(+,1)	(+,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+6$	(0,0)	—	(0,0)	(0,0)	(+,0)	(+,0)	(+,1)	(0,1)	(0,1)	(−,0)	(+,0)	(0,0)
$t+7$	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(+,0)	(−,2)	(−,1)	(−,0)	(+,0)	(0,0)
$t+8$	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(+,0)	(−,0)	(−,1)	(−,0)	(+,0)	(0,0)
$t+9$	(0,0)	(−,1)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(−,0)	(−,0)	(−,1)	(+,0)	(0,0)
$t+10$	(0,0)	(0,2)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(−,0)	(+,1)	(0,0)
$t+11$	(0,0)	(0,2)	(0,0)	(0,0)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(−,0)	(+,0)	(0,1)

to the neurons $\sigma_{l_s^{(4)}}$ and $\sigma_{l_s^{(6)}}$ in each corresponding module. Therefore, in the SUB module corresponding to the SUB instruction l_s, the neurons $\sigma_{l_s^{(4)}}$ and $\sigma_{l_s^{(6)}}$ both have two spikes, and their polarizations are both negative, which can fire at the same time, using the forgetting rule $-/a^2 \to \lambda;0$, consumes two spikes by itself.

Next, before the neuron σ_{l_k} or σ_{l_j} starts to fire, the neuron $\sigma_{l_i^{(10)}}$ or $\sigma_{l_i^{(9)}}$ respectively transmits a spike and a positive charge to the SUB module of l_s, so that the SUB module of l_s returns to the initial state. Therefore, there is no mutual influence between the SUB modules.

Module Output: Outputting the computation result.

Assume that the computation in the register machine M stops, i.e., the system Π executes to the termination instruction l_h. As shown in Figure 3, the initial state of the neuron in the Output module. At this time, neuron σ_1 contains $2n$ spikes (i.e., the number stored in the register is n), and the neuron σ_{out} contains one spike.

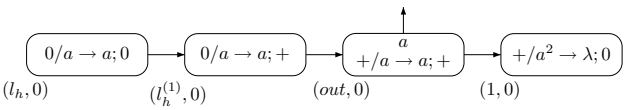

Fig. 3 Output module of of Π to output computation result

Suppose that at step t, the neuron σ_{l_h} receives a spike, i.e., reaches the pause command l_h : HALT, and is in the fired state.

The details are as follows: in step $t+2$, σ_{out} receives a spike with a positive charge transmitted by the neuron $\sigma_{l_h^{(1)}}$, it uses the spiking rule $+/a \to a;+$, and send a spike with positive charge to the neuron σ_1 and the environment. In step $t+3$, at this moment in the whole system, both neurons σ_{out} and σ_1 are in the fired state and contain one, $2n+1$ spikes. Therefore, according to the max-sequentiality strategy, from $t+3$ to $t+n+2$ steps, the neuron σ_1 fired preferentially and consumes two spikes per step. After step $t+n+2$, the

Table 5 Evolution of the polarization and the number of spikes per neuron during the simulation of Output module

Step	Neuron			
	σ_{l_h}	σ_{out}	σ_1	$\sigma_{l_h^{(1)}}$
t	(0,1)	(0,1)	(0,2n)	(0,0)
$t+1$	(0,0)	(0,1)	(0,2n)	(0,1)
$t+2$	(0,0)	(+,2)	(0,2n)	(0,0)
$t+3$	(0,0)	(+,1)	(+,2$n+1$)	(0,0)
...	...	...	...	...
$t+n+3$	(0,0)	(+,1)	(+,1)	(0,0)
$t+n+4$	(0,0)	(+,0)	(+,1)	(0,0)

neuron σ_1 has only one spike and is in a non-excitable. The neuron σ_{out} fires first and will again send a spike into the environment, i.e., the computation in the system Π stops. Therefore, the computation result can be defined and represented by the number $t_2 - t_1 - 1$, where t_2 and t_1 are the moments when the output neuron outputs the first two spikes, so the number computed by the system Π is $(t+n+3)-(t+2)-1=n$.

The specific Output module details are shown in Table 5.

Based on the above description of the three modules of ADD, SUB and Output, we get that under the max-sequentiality strategy, the system Π correctly simulates the register machine M, i.e., $N_2(\Pi)=N(M)$, so $N_2^{ms}PSNP_*^2(polar_3)=NRE$ is established, and the theorem is proved. □

4 The universality of PSN P systems with delay using the max-pseudo-sequentiality strategy

In this section, we prove that the PSN P system Π' (with delay) using the max-pseudo-sequentiality strategy is Turing universality as the number generating device.

Theorem 2 $N_2^{mps}PSNP_*^2(polar_3)=NRE$.

Proof The proof is similar to Theorem 1, we construct the PSN P system Π' (with delay) that uses the max-pseudo-sequentiality strategy to simulate the register machine $M'=(m,H,l_0,l_h,H)$. Likewise, the system Π' is also composed of three modules of ADD, SUB and Output, as shown in Figures 4, 5, and 3 respectively.

For each register r in the register M', there is a neuron σ_r corresponding to it in the system Π', and for each instruction label l_i in the register M', the neuron σ_{l_i} corresponds to it. Specifically, the two cases of the number of spikes stored in the registers, the related execution process, and the

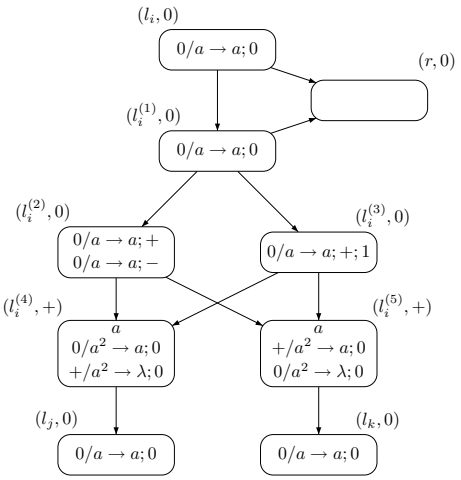

Fig. 4 ADD module of Π' to simulate instruction l_i : (ADD(r), l_j, l_k)

computation of the results in terms of $t_2 - t_1 - 1$ (the output neuron σ_{out} fires at steps t_1 and t_2 respectively, the time interval $t_2 - t_1 - 1$ between the two firings is the number stored in the register 1 of M'.) are described herein the same way as in Theorem 1 above and will not be repeated.

Module ADD: simulation on an ADD instruction l_i : (ADD(r), l_j, l_k).

The system ADD module is shown in Figure 4. Suppose that in step t, the neuron σ_{l_i}receives a spike and starts to simulate the ADD instruction l_i : (ADD(r), l_j, l_k). Next, neurons σ_{l_i} and $\sigma_{l_i^{(1)}}$ fire in turn and the number of spikes in neuron σ_r will be added two spikes, i.e., it simulates the operation of adding 1 to the number in register r. In step $t+2$, neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ receive a spike from the neuron $\sigma_{l_i^{(1)}}$, according to the max-pseudo-sequentiality strategy, both fire simultaneously. But there are two rules in the neuron $\sigma_{l_i^{(2)}}$: $0/a\to a;+$ and $0/a\to a;-$ both can be used, the system will non-deterministically select one of these two rules for implementation.

Case I: In step $t+2$, neurons $\sigma_{l_i^{(2)}}$ and $\sigma_{l_i^{(3)}}$ fire simultaneously, and neuron $\sigma_{l_i^{(3)}}$ uses the spiking rule with a one-step delay. If the neuron $\sigma_{l_i^{(2)}}$ uses the rule $0/a\to a;-$, neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$ are initially positive in polarity, and as a result of receiving a negatively charged spike transmitted from neuron $\sigma_{l_i^{(2)}}$, the positive and negative charges in neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$ cancel each other out and become neutral.

Then in step $t+3$, neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$ fire simultaneously, respectively, using the rule: $0/a^2\to a;0$ and $0/a^2\to\lambda;0$. At which a spike with a positive charge generated within neuron $\sigma_{l_i^{(3)}}$ will also be transmitted to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$, eventually neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$ will return to their initial values.

In step $t+4$, after receiving a spike with a neutral charge transmitted from neuron $\sigma_{l_i^{(4)}}$ by neuron σ_{l_j}, the system starts to simulate the l_j instruction in the register.

Case II: If the neuron $\sigma_{l_i^{(2)}}$ uses the spiking rule: $0/a \to a;+$, then fires the module associated with the instruction l_k, and finally the neuron σ_{l_k} receives a spike and starts the instruction l_k simulation. The execution of the calculation branch is the same as in Case I and will not be described in detail here.

The specific Output module details are shown in Tables 6 and 7.

Module SUB: simulation on a SUB instruction l_i : (SUB(r), l_j, l_k).

The SUB module is shown in Figure 5. Since the content of the register 1 storing the computation result in the register machine M' only increases and does not decrease, the

Table 6 Evolution of the polarization and the number of spikes per neuron during the simulation of the instruction l_j in the ADD module.

Step	Neuron						
	σ_{l_i}	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	σ_{l_j}
t	(0,1)	(0,0)	(0,0)	(0,0)	(+,1)	(+,1)	(0,0)
$t+1$	(0,0)	(0,1)	(0,0)	(0,0)	(+,1)	(+,1)	(0,0)
$t+2$	(0,0)	(0,0)	(0,1)	(0,1)	(+,1)	(+,1)	(0,0)
$t+3$	(0,0)	(0,0)	(0,0)	—	(0,2)	(0,2)	(0,0)
$t+4$	(0,0)	(0,0)	(0,0)	(0,0)	(+,1)	(+,1)	(0,1)

Table 7 Evolution of the polarization and the number of spikes per neuron during the simulation of the instruction l_k in the ADD module.

Step	Neuron						
	σ_{l_i}	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	σ_{l_k}
t	(0,1)	(0,0)	(0,0)	(0,0)	(+,1)	(+,1)	(0,0)
$t+1$	(0,0)	(0,1)	(0,0)	(0,0)	(+,1)	(+,1)	(0,0)
$t+2$	(0,0)	(0,0)	(0,1)	(0,1)	(+,1)	(+,1)	(0,0)
$t+3$	(0,0)	(0,0)	(0,0)	—	(+,2)	(+,2)	(0,0)
$t+4$	(0,0)	(0,0)	(0,0)	(0,0)	(+,1)	(+,1)	(0,1)

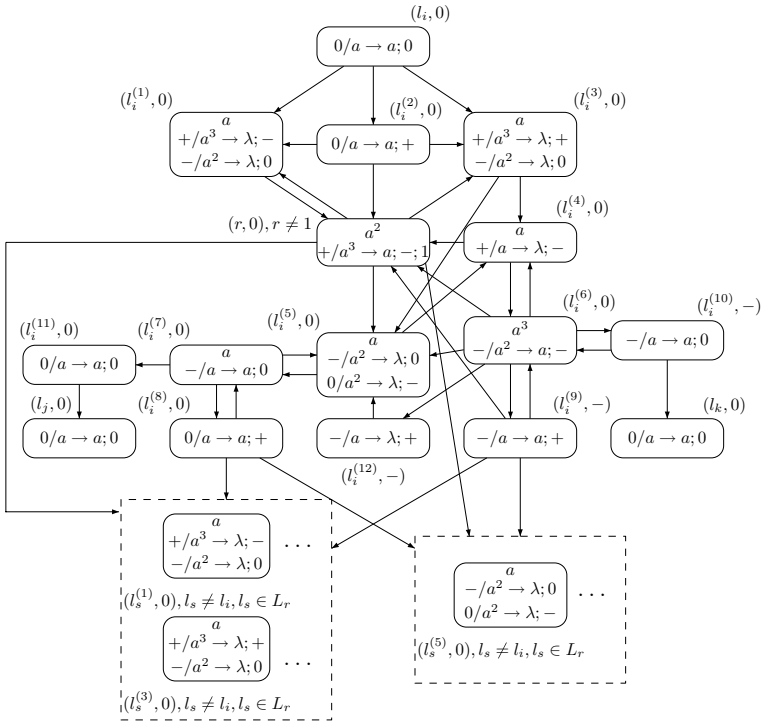

Fig. 5 SUB module of Π' to simulate instruction l_i : (SUB(r), l_j, l_k), $r \neq 1$, where $L_r = \{l \mid l$ is a label of a SUB instruction operating on register $r\}$

corresponding neuron σ_r in the subtraction module has $r \neq 1$. Suppose that in step t, the system Π' starts to simulate the subtraction instruction l_i : (SUB(r), l_j, l_k) in the registration machine M'. Neurons σ_{l_i} and $\sigma_{l_i^{(2)}}$ fire sequentially, thus, both neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(3)}}$ contains three spikes with a positive charge and neuron σ_r contains $2n+3$ spikes ($n \geq 0$) with a positive charge, all can be fired. According to the different number of spikes in the neuron σ_r, the excitation state has the following two situations:

Case I: If the number of spikes in the neuron σ_r contains $2n+3$ ($n > 0$) (the corresponding register r was originally non-empty). According to the max-pseudo-sequentiality strategy, then in step $t+2$, neuron σ_r fires preferentially, using the rule $+/a^3 \rightarrow a; -;1$, and due to the delay used, neuron σ_r is only firing and closed (i.e., it cannot receive and release spikes and charges) during the one-step delay.

In step $t+3$, since the neuron $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(3)}}$ have the current maximum number of spikes, according to the max-pseudo-sequentiality strategy and they fire simultaneously. The neuron $\sigma_{l_i^{(3)}}$ sends a positive charge to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$. At the same time the neuron σ_r delay will end and send a spike with negative charge to the synaptically connected adjacent neurons, while neuron σ_r will also receive the negative charge transmitted from neuron $\sigma_{l_i^{(1)}}$, thus is the positive and negative charges cancel each other in neuron σ_r and finally return to the initial polarity state, and the neuron σ_r contains $2n$ spikes (i.e., the number of stores in register r minus 1). Meanwhile, neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(3)}}$ will also return to the state of the initial value of the system.

In step $t+4$, the neuron $\sigma_{l_i^{(5)}}$ has two spikes and preferentially fires. the neuron $\sigma_{l_i^{(5)}}$ will send a negative charge to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(7)}}$, causing neuron $\sigma_{l_i^{(4)}}$ to return to its initial value.

Next, neurons $\sigma_{l_i^{(7)}}$, $\sigma_{l_i^{(8)}}$ and $\sigma_{l_i^{(11)}}$ fire sequentially, and finally a spike is sent to neuron σ_{l_j} and the system Π' starts to simulate the instruction l_j in the register machine M'.

Case II: If neuron σ_r has three ($n = 0$) spikes (i.e., corresponding register r turns out to be empty). According to the max-pseudo-sequentiality strategy, in step $t+2$, neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(3)}}$ and σ_r fire simultaneously. Since the neuron σ_r has a one-step delay and is in a closed state, here the negative charge transmitted from neuron $\sigma_{l_i^{(1)}}$ to neuron σ_r will be lost. Instead, neuron $\sigma_{l_i^{(3)}}$ will immediately send a positive charge to neurons $\sigma_{l_i^{(4)}}$ and $\sigma_{l_i^{(5)}}$.

In step $t+3$, the neuron $\sigma_{l_i^{(4)}}$ is the only neuron currently active in the system and therefore fires and sends a negative charge to neurons σ_r and $\sigma_{l_i^{(6)}}$. Simultaneously, the delay of neuron σ_r will end and return to initial polarity. A spike with a negative charge will be sent to neurons $\sigma_{l_i^{(1)}}$, $\sigma_{l_i^{(3)}}$ and $\sigma_{l_i^{(5)}}$ from σ_r, so that neurons $\sigma_{l_i^{(1)}}$ and $\sigma_{l_i^{(3)}}$ will return to their initial values in the system.

In step $t+4$, the neuron $\sigma_{l_i^{(6)}}$, which has three spikes with negative polarity, preferentially fires and consumes two spikes and transmit a spike with a negative charge to neuron $\sigma_{l_i^{(5)}}$, $\sigma_{l_i^{(10)}}$, $\sigma_{l_i^{(9)}}$ and $\sigma_{l_i^{(12)}}$.

In step $t+5$, the neuron $\sigma_{l_i^{(5)}}$ preferentially fires and consumes two spikes on its own. Next, neurons $\sigma_{l_i^{(9)}}$, $\sigma_{l_i^{(10)}}$ and $\sigma_{l_i^{(12)}}$ fire simultaneously, sending two and a positive charge to neurons σ_r and $\sigma_{l_i^{(5)}}$, respectively, being they restore to the initial value of the system (i.e., the corresponding register r is empty). Finally, the neuron σ_{l_k} receives a spike and the system Π' starts to simulate the instruction l_k in the register machine M'.

Similar to the simulation of the ADD instruction, detailed descriptions are shown in Tables 8 and 9.

In the ADD module, the neuron σ_r receives two spikes each time and does not stimulate the rules in it. Therefore, there is no mutual influence between the ADD modules and between the ADD module and the SUB module.

Among multiple SUB modules, suppose there are several SUB instructions l_s acting on the register r, then when the SUB instruction l_i : (SUB(r), l_j, l_k) is simulated, the neuron σ_r uses the spiking rule which send a spike and negative

Table 8 Evolution of polarization and number of spikes in the SUB module when the number of spikes of the neuron *sigma*$_r$ is $n \neq 0$

Step	Neuron										
	σ_{l_i}	σ_r	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	$\sigma_{l_i^{(8)}}$	$\sigma_{l_i^{(7)}}$	$\sigma_{l_i^{(11)}}$	σ_{l_j}
t	(0,1)	(0,2n + 2)	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(0,0)	(0,1)	(0,0)	(0,0)
$t+1$	(0,0)	(0,2n + 2)	(0,2)	(0,1)	(0,2)	(0,1)	(0,1)	(0,0)	(0,1)	(0,0)	(0,0)
$t+2$	(0,0)	(+,2n + 3)	(+,3)	(0,0)	(+,3)	(0,1)	(0,1)	(0,0)	(0,1)	(0,0)	(0,0)
$t+3$	(0,0)	—	(+,3)	(0,0)	(+,3)	(0,1)	(0,1)	(0,0)	(0,1)	(0,0)	(0,0)
$t+4$	(0,0)	(0,2n)	(0,1)	(0,0)	(0,1)	(+,1)	(0,2)	(0,0)	(0,1)	(0,0)	(0,0)
$t+5$	(0,0)	(0,2n)	(0,1)	(0,0)	(0,1)	(0,1)	(0,0)	(0,0)	(−,1)	(0,0)	(0,0)
$t+6$	(0,0)	(0,2n)	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(0,1)	(−,0)	(0,1)	(0,0)
$t+7$	(0,0)	(0,2n)	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(0,0)	(0,1)	(0,0)	(0,1)

Table 9 Evolution of polarization and number of spikes in the SUB module when the number of spikes of the neuron $sigma_r$ is $n = 0$

Step	Neuron											
	σ_{l_i}	σ_r	$\sigma_{l_i^{(1)}}$	$\sigma_{l_i^{(2)}}$	$\sigma_{l_i^{(3)}}$	$\sigma_{l_i^{(4)}}$	$\sigma_{l_i^{(5)}}$	$\sigma_{l_i^{(6)}}$	$\sigma_{l_i^{(9)}}$	$\sigma_{l_i^{(10)}}$	$\sigma_{l_i^{(12)}}$	σ_{l_k}
t	(0,1)	(0,2)	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(0,3)	(−,0)	(−,0)	(−,0)	(0,0)
$t+1$	(0,0)	(0,2)	(0,2)	(0,1)	(0,2)	(0,1)	(0,1)	(0,3)	(−,0)	(−,0)	(−,0)	(0,0)
$t+2$	(0,0)	(+,3)	(+,3)	(0,0)	(+,3)	(0,1)	(0,1)	(0,3)	(−,0)	(−,0)	(−,0)	(0,0)
$t+3$	(0,0)	—	(+,0)	(0,0)	(+,0)	(+,1)	(+,1)	(0,3)	(−,0)	(−,0)	(−,0)	(0,0)
$t+4$	(0,0)	(0,0)	(0,1)	(0,0)	(0,1)	(+,0)	(0,2)	(−,3)	(−,0)	(−,0)	(−,0)	(0,0)
$t+5$	(0,0)	(−,1)	(0,1)	(0,0)	(0,1)	(0,1)	(−,3)	(−,1)	(−,1)	(−,1)	(−,1)	(0,0)
$t+6$	(0,0)	(−,1)	(0,1)	(0,0)	(0,1)	(0,1)	(−,1)	(−,1)	(−,1)	(−,1)	(−,1)	(0,0)
$t+7$	(0,0)	(0,2)	(0,1)	(0,0)	(0,1)	(0,1)	(0,1)	(0,3)	(−,0)	(−,0)	(−,0)	(0,1)

charge to neurons $\sigma_{l_s^{(1)}}$, $\sigma_{l_s^{(3)}}$ and $\sigma_{l_s^{(5)}}$ in each corresponding module. Therefore, in the SUB module corresponding to the SUB instruction l_s, neurons $\sigma_{l_s^{(1)}}$, $\sigma_{l_s^{(3)}}$ and $\sigma_{l_s^{(5)}}$ can fire simultaneously, using the forgetting rule $-/a^2 \to \lambda;0$, consumes two spikes by itself. Next, before the neuron σ_{l_k} or σ_{l_j} starts to fire, the neuron $\sigma_{l_i^{(8)}}$ or $\sigma_{l_i^{(9)}}$ respectively pass a positively charged spike into the SUB module of l_s, making the SUB module of l_s restore to the initial value, so there is no mutual influence between the SUB modules.

The Output module in the system Π' is shown in Figure 3. Because the Output module used in the proof of Theorem 1 has only one active neuron with the largest number of spikes at each step of execution. Therefore, the module can also be used in the system Π' that uses the max-pseudo-sequentiality strategy.

Consequently, under the max-pseudo-sequentiality strategy, the above description of the three modules of ADD, SUB and Output shows that the system Π' correctly simulates the register machine M', i.e., $N_2(\Pi') = N(M')$. Therefore, $N_2^{mps}PSNP_*^2(polar_3) = NRE$ is established, and the theorem is proved. □

5 Conclusions and discussions

In this work, we focus on the computational power of sequential PSN P systems with delay while overcoming the challenge of register minus one for systems working in the max-sequentiality and max-pseudo-sequentiality strategies mentioned in the literature [42] by introducing delays. More specifically, we construct the corresponding instruction modules and demonstrate that the systems are Turing universal whether it uses the max-sequentiality strategy or the max-pseudo-sequentiality strategy as a digital generation device.

In the proof of Theorems 1 and 2, we have shown that the systems are Turing universal as the number generating device. Whether can also study the computational power of the maximum sequential PSN P system as the number accepting device, but the biggest challenge is how to accept the number n when the system halts computation. From there, it is possible to explore further the small universal PSN P systems adopting sequential mode based on the maximum spike number. And for spiking neural P systems with polarizations working on sequential variants, there are many more, such as rules on synapses, anti-spikes, etc., whether their computational power can also be studied. There is still a lot to study about sequential spiking neural P systems with polarizations.

Acknowledgements This work was supported by Anhui Provincial Natural Science Foundation (No. 1808085MF173).

References

1. 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.
2. Chen, Z., Zhang, P., Wang, X., Shi, X., Wu, T., & Zheng, P. (2018). A computational approach for nuclear export signals identification using spiking neural P systems. *Neural Computing and Applications, 29*(3), 695–705.
3. Diaz, C., Frias, T., Sanchez, G., Perez, H., Toscano, K., & Duchen, G. (2017). A novel parallel multiplier using spiking neural P systems with dendritic delays. *Neurocomputing, 239*, 113–121.
4. Díaz-Pernil, D., Peña-Cantillana, F., & Gutiérrez-Naranjo, M. A. (2013). A parallel algorithm for skeletonizing images by using spiking neural P systems. *Neurocomputing, 115*, 81–91.
5. Fan, S., Paul, P., Wu, T., Rong, H., & Zhang, G. (2020). On applications of spiking neural P systems. *Applied Sciences, 10*(20), 7011.
6. Garcia, L., Sanchez, G., Vazquez, E., Avalos, G., Anides, E., Nakano, M., Sanchez, G., & Perez, H. (2021). Small universal spiking neural P systems with dendritic/axonal delays and dendritic trunk/feedback. *Neural Networks, 138*, 126–139.
7. Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models. Single neurons, populations, plasticity*. Cambridge University Press.
8. Gutiérrez-Naranjo, M. A., & Leporati, A. (2009). First steps towards a cpu made of spiking neural P systems. *International Journal of Computers Communications & Control, 4*(3), 244–252.
9. Hertz, J., Krogh, A., & Palmer, R. G. (2018). *Introduction to the theory of neural computation*. CRC Press.

10. Ibarra, O. H., Păun, A., & Rodríguez-Patón, A. (2009). Sequential SNP systems based on min/max spike number. *Theoretical Computer Science, 410*, 2982–2991.
11. Ionescu, M., Păun, G., & Yokomori, T. (2006). Spiking neural P systems. *Fundamenta Informaticae, 71*(2—-3), 279–308.
12. 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.
13. Jiang, S., Fan, J., Liu, Y., Wang, Y., & Xu, F. (2020). Spiking neural P systems with polarizations and rules on synapses. *Complexity, 2020*(1), 12.
14. Jiang, Y., Su, Y., & Luo, F. (2019). An improved universal spiking neural P system with generalized use of rules. *Journal of Membrane Computing, 1*(4), 270–278.
15. Krithivasan, K., Metta, V. P., & Garg, D. (2011). On string languages generated by spiking neural P systems with anti-spikes. *International Journal of Foundations of Computer Science, 22*(1), 15–27.
16. Leporati, A., Mauri, G., Zandron, C., Păun, G., & Pérez-Jiménez, M. J. (2009). Uniform solutions to SAT and Subset Sum by spiking neural P systems. *Natural Computing, 8*(4), 681–702.
17. Li, J., Huang, Y., & Xu, J. (2016). Decoder design based on spiking neural P systems. *IEEE Transactions on Nanobioscience, 15*(7), 639–644.
18. Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks, 10*(9), 1659–1671.
19. Maass, W. (1997). *The Third generation of neural network models*. Technische Universitat Gräz.
20. Maass, W., & Bishop, C. M. (2001). *Pulsed neural networks*. MIT Press.
21. Neary, T. (2015). Three small universal spiking neural P systems. *Theoretical Computer Science, 567*, 2–20.
22. Pan, L., Păun, G., Zhang, G., & Neri, F. (2017). Spiking neural P systems with communication on request. *International Journal of Neural Systems, 27*(08), 1750042.
23. Pan, L., Wu, T., Su, Y., & Vasilakos, A. V. (2017). Cell-like spiking neural P systems with request rules. *IEEE Transactions on NanoBioscience, 16*(6), 513–522.
24. Pan, L., & Zeng, X. (2011). Small universal spiking neural P systems working in exhaustive mode. *IEEE Transactions on NanoBioscience, 10*(2), 99–105.
25. Pan, T., Shi, X., Zhang, Z., & Xu, F. (2018). A small universal spiking neural P system with communication on request. *Neurocomputing, 275*, 1622–1628.
26. Păun, A., & Păun, G. (2007). Small universal spiking neural P systems. *BioSystems, 90*(1), 48–60.
27. Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences, 61*(1), 108–143.
28. Păun, G. (2002). *Membrane computing: An introduction*. Springer-Verlag.
29. Păun, G. (2010). A quick introduction to membrane computing. *The Journal of Logic and Algebraic Programming, 79*(6), 291–294.
30. Păun, G., Rozenberg, G., & Salomaa, A. (2010). *The Oxford handbook of membrane computing*. Oxford University Press.
31. Peng, H., Wang, J., Ming, J., Shi, P., Pérez-Jiménez, M. J., Yu, W., & Tao, C. (2017). Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems. *IEEE Transactions on Smart Grid, 9*(5), 4777–4784.
32. Peng, H., Wang, J., Pérez-Jiménez, M. J., Wang, H., Shao, J., & Wang, T. (2013). Fuzzy reasoning spiking neural P systems for fault diagnosis. *Information Sciences, 235*, 106–116.
33. Song, T., & Pan, L. (2016). Spiking neural P systems with request rules. *Neurocomputing, 193*, 193–200.
34. Song, T., Pan, L., & Păun, G. (2013). Asynchronous spiking neural P systems with local synchronization. *Information Sciences, 219*, 197–207.
35. Song, T., Pan, L., & Păun, G. (2014). Spiking neural P systems with rules on synapses. *Theoretical Computer Science, 529*, 82–95.
36. Song, T., Pan, L., Wu, T., Zheng, P., Wong, M. D., & Rodríguez-Patón, A. (2019). Spiking neural P systems with learning functions. *IEEE Transactions on Nanobioscience, 18*(2), 176–190.
37. Song, T., Pang, S., Hao, S., Rodríguez-Patón, A., & Zheng, P. (2019). A parallel image skeletonizing method using spiking neural P systems with weights. *Neural Processing Letters, 50*(2), 1485–1502.
38. Wang, J., Peng, H., Yu, W., Ming, J., Pérez-Jiménez, M. J., Tao, C., & Huang, X. (2019). Interval-valued fuzzy spiking neural P systems for fault diagnosis of power transmission networks. *Engineering Applications of Artificial Intelligence, 82*, 102–109.
39. Wang, J., Shi, P., Peng, H., Pérez-Jiménez, M. J., & Wang, T. (2013). Weighted fuzzy spiking neural P systems. *IEEE Transactions on Fuzzy Systems, 21*(2), 209–220.
40. Wang, T., Zhang, G., Zhao, J., He, Z., Wang, J., & Pérez-Jiménez, M. J. (2014). Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. *IEEE Transactions on Power Systems, 30*(3), 1182–1194.
41. Wu, T., Bîlbîe, F. D., Păun, A., Pan, L., & Neri, F. (2018). Simplified and yet turing universal spiking neural P systems with communication on request. *International Journal of Neural Systems, 28*(08), 1850013.
42. 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.
43. Wu, T., Pan, L., & Alhazov, A. (2019). Computation power of asynchronous spiking neural P systems with polarizations. *Theoretical Computer Science, 777*, 474–489.
44. Wu, T., Păun, A., Zhang, Z., & Pan, L. (2018). Spiking neural P systems with polarizations. *IEEE Transactions on Neural Networks and Learning Systems, 29*(8), 3349–3360.
45. Wu, T., Zhang, T., & Xu, F. (2020). Simplified and yet turing universal spiking neural P systems with polarizations optimized by anti-spikes. *Neurocomputing, 414*, 255–266.
46. Wu, T., Zhang, Z., & Pan, L. (2016). On languages generated by cell-like spiking neural P systems. *IEEE Transactions on Nanobioscience, 15*(5), 455–467.
47. Wu, T., Zhang, Z., Păun, G., & Pan, L. (2016). Cell-like spiking neural P systems. *Theoretical Computer Science, 623*, 180–189.
48. Zeng, X., Xu, L., Liu, X., & Pan, L. (2014). On languages generated by spiking neural P systems with weights. *Information Sciences, 278*, 423–433.
49. Zhang, G., Pérez-Jiménez, M. J., & Gheorghe, M. (2017). *Real-life applications with membrane computing*. Springer.
50. Zhang, G., Rong, H., Neri, F., & Pérez-Jiménez, M. J. (2014). An optimization spiking neural P system for approximately solving combinatorial optimization problems. *International Journal of Neural Systems, 24*(05), 1440006.
51. Zhang, X., Zeng, X., & Pan, L. (2008). Smaller universal spiking neural P systems. *Fundamenta Informaticae, 87*(1), 117–136.
52. Zhang, X., Zeng, X., & Pan, L. (2009). On languages generated by asynchronous spiking neural P systems. *Theoretical Computer Science, 410*(26), 2478–2488.
53. Zhao, Y., Liu, X., & Wang, W. (2016). Spiking neural P systems with neuron division and dissolution. *Plos One, 11*(9), 0162882.
54. Zhu, M., Yang, Q., Dong, J., Zhang, G., Gou, X., Rong, H., Paul, P., & Neri, F. (2021). An adaptive optimization spiking neural P system for binary problems. *International Journal of Neural Systems, 31*(01), 2050054.