entropy

MDPI

Article

Spiking Neural P Systems with Membrane Potentials, Inhibitory Rules, and Anti-Spikes

Yuping Liu and Yuzhen Zhao *

Academy of Management Science, Business School, Shandong Normal University, Jinan 250014, China; 2021021027@stu.sdnu.edu.cn

* Correspondence: zhaoyuzhen@sdnu.edu.cn

Abstract: Spiking neural P systems (SN P systems for short) realize the high abstraction and simulation of the working mechanism of the human brain, and adopts spikes for information encoding and processing, which are regarded as one of the third-generation neural network models. In the nervous system, the conduction of excitation depends on the presence of membrane potential (also known as the transmembrane potential difference), and the conduction of excitation on neurons is the conduction of action potentials. On the basis of the SN P systems with polarizations, in which the neuron-associated polarization is the trigger condition of the rule, the concept of neuronal membrane potential is introduced into systems. The obtained variant of the SN P system features charge accumulation and computation within neurons in quantity, as well as transmission between neurons. In addition, there are inhibitory synapses between neurons that inhibit excitatory transmission, and as such, synapses cause postsynaptic neurons to generate inhibitory postsynaptic potentials. Therefore, to make the model better fit the biological facts, inhibitory rules and anti-spikes are also adopted to obtain the spiking neural P systems with membrane potentials, inhibitory rules, and anti-spikes (referred to as the MPAIRSN P systems). The Turing universality of the MPAIRSN P systems as number generating and accepting devices is demonstrated. On the basis of the above working mechanism of the system, a small universal MPAIRSN P system with 95 neurons for computing functions is designed. The comparisons with other SN P models conclude that fewer neurons are required by the MPAIRSN P systems to realize universality.

Keywords: spiking neural P systems; universality; membrane potential; inhibitory rules; anti-spikes

Citation: Liu, Y.; Zhao, Y. Spiking Neural P Systems with Membrane Potentials, Inhibitory Rules, and Anti-Spikes. *Entropy* **2022**, *24*, 834. https://doi.org/10.3390/e24060834

Academic Editor: Fernando Morgado-Dias

Received: 30 April 2022
Accepted: 14 June 2022
Published: 16 June 2022

1. Introduction

Artificial neural networks (ANNs) aim to empower artificial systems with information processing functions consistent with the complex yet efficient human brain system. They are progressively building models with greater functionality and a better fit with biological facts. The third-generation neural network models, spiking neural networks [1], adopt spikes for information encoding and transmission, which provides models with biological features consistent with realistic rationality. Spiking neural P systems (SN P systems) in the field of membrane computing has become a hotspot as one of the third-generation ANNs.

Membrane computing is a branch of natural computing [2]. The distributed parallel computing models obtained from the development of membrane computing are membrane systems, also known as P systems, and the study of P systems has been divided into three types according to the cell membrane structure or cell distribution: cell-like P systems, tissue-like P systems, and neural-like P systems. For the theoretical research of membrane computing, three types of P systems have been extended to obtain several universal computational models [3–7], and the computational complexity of extended P systems has been explored [8–20]. For the application research of membrane computing, existing studies have realized the integration of membrane computing with algorithms for applications in

robot control [21,22], data modeling, and optimization [23–26], algorithms for solving NP problems [27–29], clustering algorithms [30–37], and image processing [38–40].

The spiking neural P system is an important component of the neural-like P system, an abstract simulation of the nervous system in which neurons communicate by sending spikes. Research on SN P systems has been highly dynamic in recent years, and existing research falls into two main areas: theoretical research and application research.

Regarding the theoretical research of SN P systems, there are two main parts including the proposal of variants and the evaluation of computational performance. Several variants of SN P systems have been obtained, mainly via the introduction of various biological mechanisms or features. The purpose is to bring the system more in line with biological principles and facts, as well as to improve computational performance, such as reducing resources without losing computational power. Existing studies abstract biological phenomena and facts, and they extend the systems via the continuous introduction of various biological mechanisms. The extension of the system objects has been achieved via the introduction of calcium-producing astrocytes [41] and the use of five types of spikes [42]. The extension of the system rules has been realized through the control of neuronal firing using a threshold mechanism [43,44], introducing spikes distribution mechanism in rules [45], applying white rules [46], and applying evolutionary rules and communication rules [47]. By introducing the structural plasticity mechanism [48,49], the original SN P systems have achieved a larger degree of extension. The extension of the system operation was achieved via the introduction of the extended channel rule [50], synaptic delay [51], dendritic and axonal computation [52], a generalized use of rules [53] and the application of four sequential working strategies in the system [54,55]. The various variants of the SN P systems obtained still possess computational generality.

Furthermore, theoretical research has focused on evaluating the computational performance of variants of SN P systems, mainly including the computational generality and the computational complexity of the system. The evaluation of the computational generality of the proposed system focuses on proving the Turing universality, also called generality, via the simulation of the computation of the universal register machines with different functions. The extended SN P systems obtained from existing studies have all been proved to be Turing universal as number generation and accepting devices [41,42,44,45,50–52]. The studies addressing the computational complexity of systems include temporal complexity and spatial complexity. The temporal complexity of the system computation is assessed using the ability of the proposed variant to solve the NP-hard problems in polynomial time [56,57]. The spatial complexity is assessed via the construction of a universal system with fewer computational resources, i.e., the neurons. For example, 109 neurons are required to construct a universal dynamic threshold SN P system [44], while a SN P system with target indications requires only 15 neurons to achieve generality [45].

Regarding the application research of SN P systems, studies have focused on combining SN P systems with algorithms to solve real-world problems, and then they have evaluated the performance of the proposed algorithm with the help of experimental results on data sets and comparisons with other algorithms. SN P systems and numerous universal variants have been realized for applications in different real-world domains, such as performing basic arithmetic operations [58], simulating Boolean circuits [59], solving classification problems [60], fault diagnosis [61,62], recognizing English letters [63], image processing [64–66], modeling [67], and time series prediction [68–70]. As one of the third-generation of ANNs, SN P systems are considered to have significant development potential.

Recently, Wu et al. [71] introduced the concept of polarization into SN P systems and obtained the spiking neural P systems with polarization (PSN P systems), which changed the previous control mechanism of neuronal firing in SN P systems. Instead of using regular expressions for the trigger conditions of rules, the neuron firing was controlled by judging the neuron-associated polarizations $(+, 0, -)$ as the rule trigger conditions. The application of polarizations made the information exchange of neurons in the system more consistent

with biological facts, and the systems were shown to still be Turing universal. Wu et al. [72] then introduced a new coding object, the anti-spikes, into the PSN P systems (PASN P systems), in order to provide better computational performance and further simplification. In addition, the computational universality of the PSN P system in asynchronous mode [73] and sequential mode [74,75] is demonstrated. Subsequently, Yang et al. [76] added the feature of multiple channels to the PSN P systems (SNP–MCP systems). By introducing the spiking rules on synapses, Jiang et al. [77] obtained a new variant (PSNRS P systems) requiring less computational resources to realize computational generality.

The new variant of the SN P systems constructed in this paper is motivated by the following two biological mechanisms.

- Neurons of the nervous system contain ions that carry a certain amount of charge (either positive or negative), and the presence of charge forms the transmembrane potential difference (also called potential) of the nerve cell. When a neuron receives a stimulus flow directionally, it forms an electric current, changes the transmembrane potential difference, generates an action potential, and counts to conduct this electrical signal along the cell membrane [78]. Thus, as shown in Figure 1, the conduction of excitation in the nervous system is the process of action potential conduction, and the phenomena of charge aggregation, flow, and transport exist in the cell membrane of neurons. Based on the PSN P systems, we introduce the concept of membrane potential according to the above biological phenomena, as a way to update the number of charges and polarization states of a neuron by considering the aggregation within the neuron and the charge transmission between neurons. Together with the polarization state and the number of charges, the membrane potential of a neuron is composed, and the membrane potential is used as the triggering condition of rules, which provides more powerful control over the systematic computation. The resulting model is constructed to better simulate the characteristics of neurons and the working mechanism of the nervous system.
- There are two main types of synapses between neurons according to their synaptic effects on neuronal activity: excitatory synapses and inhibitory synapses. The presence of excitatory synapses enables the transmission of information between neurons, and the operation mechanism of this synapse can be well modeled by the application of rules of consumption or transmission of spikes by systemic neurons. For inhibitory synapses, such synapses can cause postsynaptic neurons to generate inhibitory postsynaptic potentials, which in turn have an inhibitory effect on the excitation of neurons. Peng et al. [79] formalized the effect mechanism of inhibitory synapses as inhibitory rules within systematic neurons. The extension of the rule-triggering conditions not only made the firing behavior of neurons limited by the contained rules, but also controlled by the state of neurons connected to the current neuron through inhibitory synapses, effectively modeling the mechanism of action of inhibitory synapses in the nervous system.

Based on the above motivation, we also introduced the application of anti-spikes in the new systems to make the systems have better computational performance, and the obtained spiking neural P systems with membrane potentials, inhibitory rules, and anti-spikes (MPAIRSN P systems for short) are more in line with the biological mechanisms. In addition, the new inhibitory rules introduced in the systems consider the neuronal membrane potential as the rule triggering condition, and we update the form of the inhibitory rules as $(A_{en}, \overline{A_{in(i,j)}})/b^c \to b';\beta$; the firing of the current neuron σ_i needs to satisfy that the membrane potential A_i is consistent with the membrane potential A_{en} required by the rule, and that the membrane potential A_j of the inhibitory neuron σ_j, which is connected to the current neuron σ_i by inhibitory synapses, cannot belong to the set of membrane potential $A_{in(i,j)}$ required by the rule. That is, the condition $A_i = A_{en} \wedge A_j \notin A_{in(i,j)}$ needs to be satisfied for the current neuron to fire. The introduction of inhibitory rules achieves a more powerful control over the neurons in the system and it can further reduce the computational resources required by the universal MAIRSN P systems.

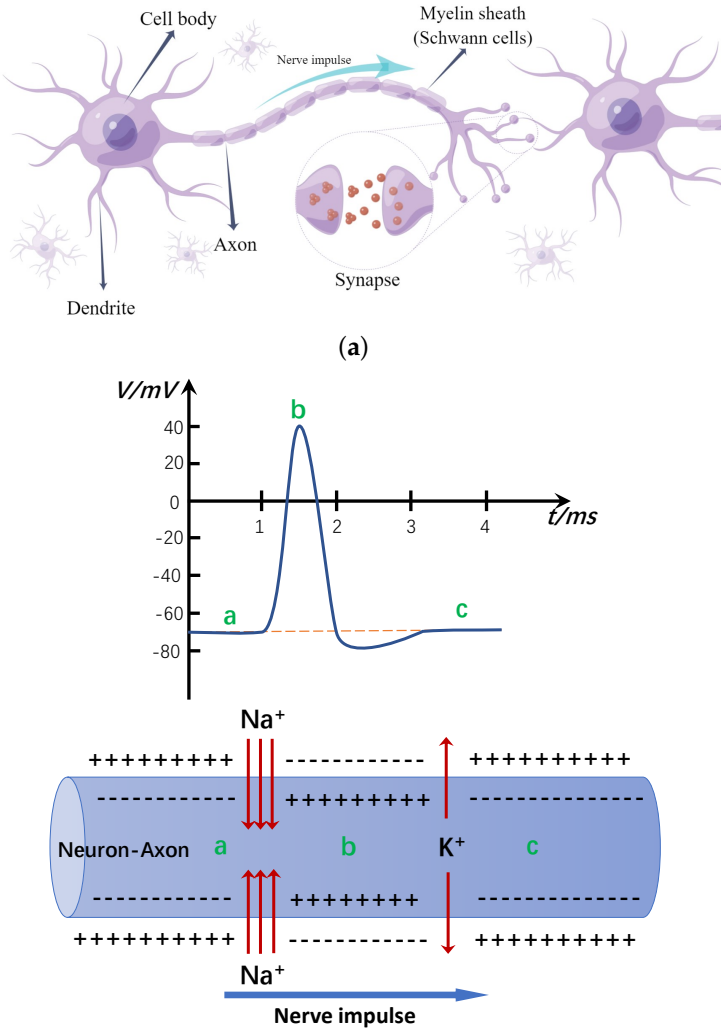

Figure 1. Conduction of the nerve impulse. (**a**) Schematic diagram of the basic structure of a neuron. (By Figdraw, www.figdraw.com accessed on 10 June 2022); (**b**) Schematic diagram of the spike voltage variation of the action potential with time.

The research contribution of this paper is mainly focused on the following two points.

- We introduce the concept of membrane potential in the SN P systems and propose a new rule-triggering mechanism: using the membrane potential of a neuron as the condition. In addition, the inhibitory rule with membrane potential as the rule triggering condition is updated and applied, which in turn leads to the proposed MPAIRSN P systems.
- The proposed MPAIRSN P systems are shown as Turing universal operating in generating and accepting modes. A small universal MPAIRSN P system is constructed, using 95 neurons and allowing for the computation of functions. Compared with other variants of SN P systems adopting polarizations, the general MPAIRSN P systems require fewer computation resources and have faster computation speed.

For the remainder of this paper, Section 2 proposes the definition of the MPAIRSN P system and displays a small system as an introductory example of how the MPAIRSN P system operates, which has the capability of generating arbitrary nonzero natural numbers. Section 3 proves the Turing universality of the MPAIRSN P systems, mainly operating in generating and accepting modes, respectively. Section 4 constructs a small universal

MPAIRSN P system containing 95 neurons for computing functions. Section 5 gives conclusions and the future research outlook.

2. SN P Systems with Membrane Potentials, Inhibitory Rules, and Anti-Spikes

In this section, a formal definition of the MPAIRSN P system is given to introduce the related concepts. Moreover, we design a small MPAIRSN P system as an example for illustrating the working mechanism. This small system implements the function of generating arbitrary nonzero natural numbers.

2.1. Definition

An MPAIRSN P system consisting of $m \geq 1$ neurons is represented as a tuple:

$$\Pi = (O, \sigma_1, \cdots, \sigma_m, syn, syn', in, out),$$

where:

(1) $O = \{a, \bar{a}\}$ is an alphabet consisting of two characters, where a and $\bar{a}$ denote a spike and an anti-spike, respectively.

(2) σ_1,σ_2,...,σ_m are the m neurons contained in the system, and each neuron can be expressed as a tuple $\sigma_i = (A_i, n_i, R_i)$, $1 \leq i \leq m$, where:

(a) $A_i = (PAR_i, X_i)$ is the initial membrane potential of neuron σ_i, which contains two main components, the polarization of the membrane potential $PAR_i \in \{+, 0, -\}$, and the number of charges $X_i \in N$ contained in the neuron; N is the set of natural numbers.

(b) n_i is the initial number of spikes/anti-spikes contained in the neuron. If $n_i \geq 0$, then the neuron σ_i contains n_i spikes; if $n_i < 0$, then the neuron σ_i contains $-n_i$ anti-spikes.

(c) R_i is the set of rules contained in the neuron. The rules are mainly of the following types.

i. Standard rules: $A/b^c \rightarrow b'; \beta$,

where A is the firing condition, $\beta \in \{+, 0, -\}$, $b, b' \in O$, $c \geq 1$.

ii. Inhibitory rules: $(A_{en}, \overline{IA_{in(i,j)}})/b^c \rightarrow b'; \beta$,

where A_{en} and $IA_{in(i,j)}$ together constitute the firing condition, $\beta \in \{+, 0, -\}$, $b, b' \in O$, $c \geq 1$.

iii. $a\bar{a} \rightarrow \lambda$ is called the annihilating rule. This rule enforces a higher priority than the standard rules and inhibitory rules above.

(3) $syn = \{(i, j) | 1 \leq i \neq j \leq m\}$ is the set of standard synapses between neurons. $syn' = \{(i, j) | 1 \leq i \neq j \leq m\}$ is the set of inhibitory synapses between neurons, where j is the label of the inhibitory neuron and i is the label of the inhibited neuron.

(4) $in, out \in \{1, 2, \cdots, m\}$ are used to distinguish the input and output neurons in the system, respectively. If the system does not contain the input neuron or output neuron, it is omitted.

For the operation of the MPAIRSN P system, since it is a requirement to ascertain neuronal polarization by calculating the number of contained charges, we give the following conventions of charge calculation.

(1) Multiple positive (or negative) charges are allowed to accumulate within a neuron.
(2) A positive charge and a negative can cancel each other out and disappear.
(3) Receiving any number of neutral charges does not change neuronal polarization state.

There are three main types of rules in the MPAIRSN P system: standard rules, inhibitory rules, and annihilating rules.

The standard rule takes the form of $A/b^c \rightarrow b'; \beta$. The triggering condition of this rule restricts the membrane potential of the neuron, i.e., both the polarization and the charge number are required to satisfy the condition. The firing rule is available to be applied when the membrane potential state of neuron σ_i is such that $A_i = A$, and the number of spikes or anti-spikes contained in σ_i is at least c so that sufficient spikes are available to achieve firing.

In particular, if the firing condition of the rule restricts only the polarization of the neuron, then the standard firing rule is simplified to the form $PAR/b^c \rightarrow b'; \beta$. The rule is

applicable if the polarization state of neuron σ_i satisfies $PAR_i = PAR$ and the number of spikes contained in σ_i is at least c.

Moreover, if the standard rule is triggered without generating spikes or anti-spikes, i.e., $b' = \lambda$, this rule is called the standard forgetting rule. In this case, only one charge β is produced and then transmitted.

In summary, if a neuron σ_i satisfies the trigger condition of its standard firing rule, c spikes or anti-spikes are consumed, the neuron generates one spike or anti-spike and a charge β that are simultaneously sent to the successive neuron connected to σ_i via synapse, updating the number of spikes and the membrane potential state of the successive neuron. If the rule triggered is a forgetting rule, only charge β is produced and then transmitted between neurons.

Similarly, standard forgetting rules of the form $A'/b^s \to \lambda; \beta'$ and $PAR'/b^s \to \lambda; \beta'$ consume s spikes or anti-spikes in the neuron, and generate no spikes or anti-spikes but only produce one charge, which is sent to subsequent neurons σ_j connected to σ_i via synapses. This type of rules can be applied as soon as the membrane potential state of neuron σ_i coincides with the membrane potential requirement of the rule, i.e., $A_i = A'$, or the polarization of neuron σ_i satisfies $PAR_i = PAR'$. Moreover, the number of spikes or anti-spikes in σ_i must be greater than s.

The inhibitory rules take the form shown in Figure 2. For the formal representation of the rule, A_{en} indicates the membrane potential condition that needs to be satisfied for the current neuron σ_i to fire, and the subscript (i, j) of $IA_{in(i,j)}$ indicates the presence of a directed inhibitory synapse between the current neuron σ_i and its inhibitory neuron σ_j.

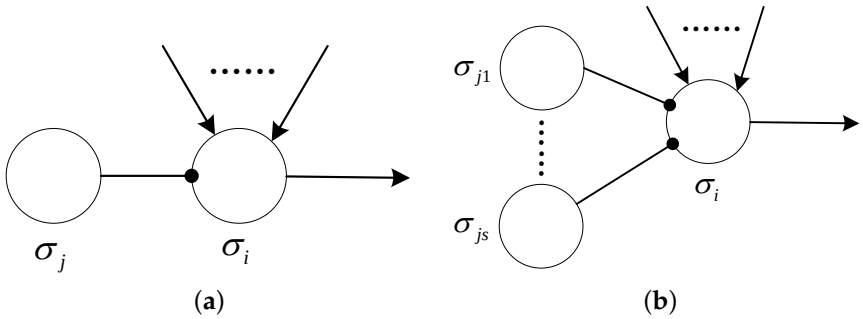

Figure 2. Rules. (**a**) Inhibitory rules: $(A_{en}, \overline{IA_{in(i,j)}})/b^c \to b'; \beta$. (**b**) Extend inhibitory rules: $(A_{en}, \overline{IA_{in(i,js)}}, \cdots, \overline{IA_{in(i,js)}})/b^c \to b'; \beta$.

The application of each inhibitory rule is controlled by the membrane potential A_i or the polarization PAR_i of the current neuron σ_i, and the membrane potential A_j or the polarization PAR_j of the inhibitory neuron σ_j connected to neuron σ_i via the inhibitory synapse. Taking the form of $(A_{en}, \overline{A_{in(i,j)}})/b^c \to b'; \beta$,the inhibitory rule is applicable to neuron σ_i if its membrane potential A_i coincides with the membrane potential condition of the rule A_{en}, i.e., $A_i = A_{en}$, with its inhibitory neuron σ_j containing the membrane potential A_j that does not belong to the set $IA_{in(i,j)}$, i.e., $A_j \notin IA_{in(i,j)}$. In other words, the triggering of the inhibitory rule requires that the expression $A_i = A_{en} \wedge A_j \notin IA_{in(i,j)}$ holds. Additionally, the number of spikes or anti-spikes contained in neuron σ_i is at least c.

Similarly, the simplified inhibitory rule takes the form of $(PAR_{en}, \overline{IPAR_{in(i,j)}})/b^c \to b'; \beta$, which is available to neuron σ_i if the polarization states of current neuron σ_i and its inhibitory neuron σ_j enable the expression $PAR_i = PAR_{en} \wedge PAR_j \notin IPAR_{in(i,j)}$, and the number of spikes or anti-spikes contained in neuron σ_i is at least c. Analogously, this type of rules only restricts the polarizations of the current neuron σ_i and its inhibitory neuron σ_j.

If triggered, the inhibitory firing rule consumes c spikes or anti-spikes in the neuron σ_i, generating one spike or anti-spikes and one charge β, which are sent to the successive neuron connected to σ_i via synapse. For the inhibitory rule, if $b' = \lambda$, it is an inhibitory forgetting rule. Similarly, the triggering of this rule does not generate spikes or anti-spikes, but only a charge.

It is worth noting that, unlike the standard form of synaptic connection, the inhibitory neuron is connected to the current neuron via the inhibitory synapse, also called the inhibitory arc. The inhibitory arc is represented in the directed graph as a directed arc with a solid circle, whereas the standard synapse is represented by a directed arc with an arrow. For the directed inhibitory arc in a graph, there are the following conventions.

(1) If there is an inhibitory arc between neurons σ_i and σ_j , then there is no standard arc between them, i.e., there is no longer a standard synaptic connection between two neurons.
(2) If neuron σ_j is an inhibitory neuron to neuron σ_i, there is no transmission of spikes, and charges between two neurons. Moreover, the application of the inhibitory rules in σ_i performs no effect on its inhibitory neuron σ_j in terms of changing the number of spikes and the membrane potential of neuron σ_j. The transmission of spikes, anti-spikes and charges does not take place in the inhibitory arc. Thus, inhibitory neuron σ_j only functions as a control of neuron σ_i in terms of its firing.

If a neuron is controlled by more than one inhibitory neuron, there are extended inhibitory rules that are applicable. As shown in Figure 2b, neuron σ_i has multiple inhibitory neurons $\sigma_{j1}, \cdots, \sigma_{js}$. Then, for the extended inhibitory firing rule $(A_{en}, \overline{IA_{in(i,js)}}, \cdots, \overline{IA_{in(i,js)}})/b^c \rightarrow b'; \beta$, the firing of the neuron σ_i requires that the neuron σ_i and its inhibitory neurons $\sigma_{j1}, \cdots, \sigma_{js}$ satisfy the condition that $A_i = A_{en} \wedge A_{j1} \notin IA_{in(i,j1)} \wedge \cdots \wedge A_{js} \notin IA_{in(i,js)}$.

The annihilation rule takes the form of $a\bar{a} \rightarrow \lambda$, i.e., one spike and one anti-spike cancel each other out when they meet in the same neuron. The execution of this rule is time independent and uncontrolled by the neuronal membrane potential or polarization. Therefore, the two types of spikes are unable to be present in the same neuron simultaneously.

Generally, any MPARISN P system containing m neurons can be represented by a tuple $\Pi = (O, \sigma_1, \cdots, \sigma_m, syn, syn', in, out)$, or by a topological directed graph. The directed graph representation can help us to visualize the structure and the operation process of the system. Then, in order to facilitate the explanation of the operation process, the MPAIRSN P systems in the latter part of this paper are mainly introduced with the help of directed graphs.

Topologically, an MPAIRSN P system Π can be expressed as a directed graph with inhibitory arcs, where m neurons are shown as m nodes containing initial spikes and rules, represented by rounded rectangles, and the synaptic connections between neurons are denoted as directed arcs, including the standard directed arcs and the newly introduced inhibitory arcs. For each neuron σ_i in the system, we assign a label i, an initial number of spikes, and an initial membrane potential. The neuron label and its initial membrane potential are represented by the tuple $(i, (PAR_i, X_i))$ placed next to the rounded rectangle representing the neuron, where PAR_i represents its polarization and X_i is the corresponding number of charges. The initial spikes are then located in the first row inside the rounded rectangle, and if the neuron contains no initial spikes, only the rule is included.

As the state of neuron σ_i in a system at a given instance is formulated by the number of spikes jointly with the membrane potential, it follows that the overall computational configuration of the system Π at the particular instance t is denoted as $C_t =< A_1/n_1, A_2/n_2, \cdots, A_m/n_m >$, $n_i \in Z$, where A_i is the membrane potential, n_i is the number of spikes, and Z is the set of integers. The computation performed in the system is presented as a transformation of the systematic configuration, starting from the initial configuration C_0. The systematic neurons apply three types of rules to perform the computation, and the process is represented as $C_0 \Rightarrow C_1 \Rightarrow \cdots \Rightarrow C_h$, where C_h is the terminate configuration. The computational process ends when no more rule is available to be applied in the system. Additionally, for any calculation, there is a corresponding spikes sequence encoded by 01 that reflects the behavior of the output neuron, where 0 signifies silence of the output neuron, while 1 indicates that the output neuron fires and transmits spikes.

The MPAIRSN P system Π is operable in both the generating mode and the accepting mode. When the system operates in the generation mode, it can serve as a number

generation device and export computation results by the output neuron σ_{out}. The computation result can be defined as either the time interval between the first two spikes, or the total number of spikes sent by σ_{out}. We denote the computation results generated by the MPAIRSN P system Π as $N(\Pi)$, or $N_2(\Pi)$ in case the computation result is defined in the former manner. When the system performs in accepting mode, an input neuron is introduced into the system, and the output neuron is removed. The number n is encoded as a spikes sequence consisting of 01, and then it is read by the input neuron σ_{in}. If the computation terminates, it indicates that the number n is accepted by the system, and the set of numbers accepted by the system Π is noted as $N_{acc}(\Pi)$.

2.2. An Example

An example is given in Figure 3 to explain the working mechanism of MPAIRSN P systems, which is a small system Π_e capable of generating arbitrary positive natural numbers $n(n > 0)$. The MPAIRSN P system Π_e containing five neurons is represented as a tuple:

$$\Pi_e = (O, \sigma_1, \sigma_2, \cdots, \sigma_5, syn, syn', in, out),$$

where:
$O = \{a, \bar{a}\}$,
$\sigma_1 = ((0,0), 2, \{0/a \to a; - \})$,
$\sigma_2 = ((+,1), 0, \{0/\overline{a \to \bar{a}}; - , -/a \to \lambda; 0 \})$,
$\sigma_3 = ((0,0), 2, \{(0, \overline{\{+\}})/a \to a; 0 , (-, \overline{\{+\}/\bar{a} \to \lambda; 0 \})$,
$\sigma_4 = ((0,0), 0, \{(0, \overline{\{+, 0\}})/a \to a; 0 , (0, \overline{\{+, 0\}})/a \to \bar{a}; - \})$,
$\sigma_5 = ((0,0), 0, \{-/\bar{a} \to a; 0 , -/a \to \lambda; 0 \})$,
$syn = \{(1,2), (3,4), (4,3), (2,5), (4,5)\}$,
$syn' = \{(3,2), (4,2)\}$, and
$out = \sigma_5$.

For ease of understanding and interpretation, we also represent the system Π_e as a directed graph (Figure 3), where both neuron σ_1 and neuron σ_3 contain two initial spikes, and neuron σ_2 carries the initial membrane potential $(+, 1)$. There are, in total, five neurons in the system, where neuron σ_5 is the output neuron. Each neuron is assigned a tuple $(i, (PAR_i, X_i))$. For example, for the tuple $(2, (+, 1))$ assigned to neuron σ_2, 2 is the label of the neuron, and the initial membrane potential is $(+, 1)$, i.e., representing that the neuron σ_2 carries the positive polarization and contains one positive charge. Moreover, the neuron σ_2 is connected to neuron σ_3 and neuron σ_4 via two inhibitory synapses, meaning that the firing of neuron σ_3 and neuron σ_4 is controlled by their neuron σ_2.

The initial configuration of the system in this example is $C_0 =< (0,0)/2, (+,1)/0, (0,0)/2, (0,0)/0, (0,0)/0 >$. The calculation result of this system is defined as the interval between the first two spikes received in the environment.

At the initial step, since neuron σ_1 with neutral polarization contains two spikes, the rule $0/a \to a; -$ is applied to send one spike and one negative charge to neuron σ_2. Meanwhile, neuron σ_3 which remains of neutral polarization and contains two spikes, is unable to apply the inhibitory firing rule as $PAR_2 \in \{+\}$ without satisfying the firing condition. Thereby, the system configuration of step 1 can be obtained as $C_1 =< (0,0)/1, (0,0)/1, (0,0)/2, (0,0)/0, (0,0)/0 >$. At step 1, as the neuron σ_1 contains one spike and remains neutral in polarization, rule $0/a \to a; -$ continues to be applied to send another spike and one negative charge to neuron σ_2. Additionally, neuron σ_2 with neutral polarization contains one spike, where its rule $0/a \to \bar{a}; -$ can be triggered to send one spike and one negative charge to neuron σ_5. Neuron σ_3 satisfies the trigger condition $PAR_3 = 0 \wedge PAR_2 \notin \{+\}$ of the rule $(0, \overline{\{+\}})/a \to a; 0$, and it sends one spike to neuron σ_4 by the application of this inhibitory firing rule. We then have $C_2 =< (0,0)/0, (-,1)/1, (0,0)/1, (0,0)/1, (-,1)/(-1) >$ at step 2. At this step, it becomes empty after neuron σ_2 fires without producing spikes and charges, since the neuron σ_2 satisfies the trigger condition of the forgetting rule $-/a \to \lambda; 0$. Neuron σ_5 with negative polarization contains an anti-spike, allowing the application of the rule to send the first

spike to the environment. Additionally, the application of the rule $(0, \overline{\{+\}})/a \to a; 0$ continues in neuron σ_3, sending one spike to neuron σ_4. The polarization of neuron σ_4 and the polarization of its inhibitory neuron σ_2 fulfill $PAR_4 = 0 \wedge PAR_2 \notin \{+, 0\}$, and then both rule $(0, \overline{\{+, 0\}})/a \to a; 0$ and rule $(0, \overline{\{+, 0\}})/a \to \bar{a}; -$ are applicable. The uncertainty selection of rules applied in neuron σ_4 can be performed, and then the following two cases exist.

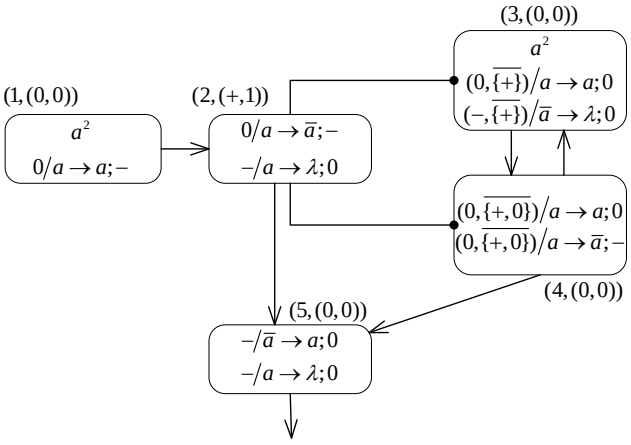

Figure 3. A small MPAIRSN P system Π_e as an illustrative example.

Case 1: Assume that the rule $(0, \overline{\{+, 0\}})/a \to \bar{a}; -$ is applied in neuron σ_4 at step 2, and an anti-spike and a negative charge are sent to neuron σ_3 and neuron σ_5, consuming one spike while still receiving a spike from neuron σ_3, then neuron σ_4 contains one spike and has neutral polarization. Therefore, both neuron σ_3 and neuron σ_5 contain an anti-spike and have negative polarization. Then, there is $C_3 =< (0,0)/0, (-,1)/0, (-,1)/(-1), (0,0)/1, (-,2)/(-1) >$ at step 3. Subsequently, the second spike is sent to the environment from neuron σ_5 by applying rule $-/\bar{a} \to a; 0$ at step 4. Once no rule can be applied in the system, i.e., the computation halts, the time interval between the first spike received in the environment is 1, i.e., the natural number 1 computed by the system is obtained.

Case 2: Suppose that by applying the rule $(0, \overline{\{+, 0\}})/a \to a; 0$ at step 2, neuron σ_4 sends one spike to neuron σ_3 and neuron σ_5. After consuming one spike while receiving one spike from neuron σ_3, there is one spike contained in neuron σ_4 with positive polarization. Then, there is $C_3 =< (0,0)/0, (-,1)/0, (0,0)/1, (0,0)/1, (-,1)/1 >$. The application of rule $(0, \overline{\{+\}})/a \to a; 0$ continues in neuron σ_3 at step 3, sending one spike to neuron σ_4, and neuron σ_5 applies the forgetting rule $-/a \to \lambda; 0$ without producing spikes. Since neuron σ_4 contains one spike and carries neutral polarization, the uncertain selection of rule $(0, \overline{\{+, 0\}})/a \to a; 0$ or $(0, \overline{\{+, 0\}})/a \to \bar{a}; -$ for application can be performed again. In this case, we assume that neuron σ_4 consistently applies rule $(0, \overline{\{+, 0\}})/a \to a; 0$ selectively at each step until step $k (k > 4)$. Then, we have $C_i =< (0,0)/0, (-,1)/0, (0,0)/1, (0,0)/1, (-,1)/1 >, (4 \leq i \leq k)$. The rule $(0, \overline{\{+, 0\}})/a \to \bar{a}; -$ is selected to be applied in neuron σ_4 at step k, and then an anti-spike and a negative charge are generated and sent to neuron σ_3 and neuron σ_5. At moment $k+2$, neuron σ_5 contains one spike for consumption of the rule $-/\bar{a} \to a; 0$ to send the second spike to the environment. Then, the time interval between the first two spikes received in the environment until the calculation stops is $k - 3 (k > 4)$, giving an arbitrary natural number that is greater than 1.

As mentioned above, the small MPAIRSN P system in this example allows for the generation of arbitrary nonzero natural numbers. The set of positive natural numbers generated by this system is denoted as $N_2(\Pi_e) = \{n | n \geq 1\}$.

3. The Computational Universality of MPAIRSN P Systems

Through the design of the MPAIRSN P systems to simulate the register machine in accepting and generating modes, respectively, this section explores the computational universality of the MPAIRSN P system to demonstrate that all recursive enumerated sets of numbers can be generated or accepted by the system.

A register machine is represented as a tuple: $M = (m, H, l_0, l_h, I)$, where m is the number of registers, H is the set of labels corresponding to instructions, while $l_0, l_h \in H$ are the starting and halting instruction labels, respectively. I is the set of instructions distinguished by labels in H, and two types of instructions are included: $l_i : (ADD(r), l_j, l_k)$ and $l_i : (SUB(r), l_j, l_k)$. Typically, the family NRE is used to characterize the set of numbers that can be generated or accepted by the register machine. The computational results of the MPAIRSN P system containing m neurons are denoted by $N_\alpha PAIRSNP_m^n$, $\alpha \in \{2, acc\}$, with at most n rules within each neuron.

3.1. The MPAIRSN P System as a Number Generating Device

The register machine M working in generating mode is available to serve as a number generator. As for its configuration, at the initial state, each register in M is empty, and the machine starts working by executing the instruction l_0. Then, the machine calculates by executing a series of ADD and SUB instructions. Eventually, the machine's computation is terminated by executing the instruction l_h, at which point the number n stored in register 1 is the number generated by M.

Theorem 1. $N_2SNP_*^2(ch_3) = NRE$.

Proof. According to the Turing–Church thesis, the relation $N_2SNP_*^2(ch_3) \subseteq NRE$ holds [80]. It remains only to prove that $NRE \subseteq N_2SNP_*^2(ch_3)$. Therefore, we mainly consider the proof using the MPAIRSN P system Π_1 to simulate the computation of the register machine M working in the generative mode. We assume that for the configuration of the machine at a given step, all its registers are empty and that the value stored in register 1 of them does not decrease during the computation.

In the following contents, an MPAIRSN P system Π_1 is designed to simulate the register machine M_1 working in generation mode, which can act as a number generation device (or a number generator). The system Π_1 is designed with three types of modules containing the ADD module, the SUB module, and the FIN module, as shown in Figures 4–6, respectively.

In order to associate with the register machine M_1, we set the neuron σ_r in system Π_1 to correspond to a register r in machine M_1, and the number of spikes contained in the neuron σ_r is equal to the value stored in the corresponding register r. The neuron σ_{l_i} in system Π_1 corresponds to instruction l_i in the machine. In addition, the auxiliary neurons $\sigma_{c_i} (i = 1, 2, \cdots)$ of the module associated with a neuron σ_{l_i} are added, and they are uniquely related.

In the initial state, each neuron in the system has an initial membrane potential; the neuron σ_{l_0} receives a spike for triggering the system computation, and the rest of the neurons are empty. During the computation steps, neuron σ_{l_i} with a neutral polarization fires as soon as it receives a spike, i.e., the system begins to simulate instruction $l_i : (OP(r), l_j, l_k)(OP = ADD, SUB)$, which triggers the work of the relevant module. Following that, neuron σ_{l_j} or neuron σ_{l_k} receives one spike and starts to simulate instruction l_j or l_k, which triggers the calculation of the corresponding module. If neuron σ_{l_h} receives one spike and fires, then system Π_1 successfully simulates the computation of the register machine M_1 in generative mode, and the computation result is output by the FIN module where neuron σ_{l_h} is located. We define the time interval $t_2 - t_1$ between the first two spikes output by the neuron σ_{out} to the environment as the computation result, and the value corresponds to the number stored in register 1.

The procedure and the details of the proofs for the simulations of the ADD, SUB, and FIN modules of the system Π_1 are given below.

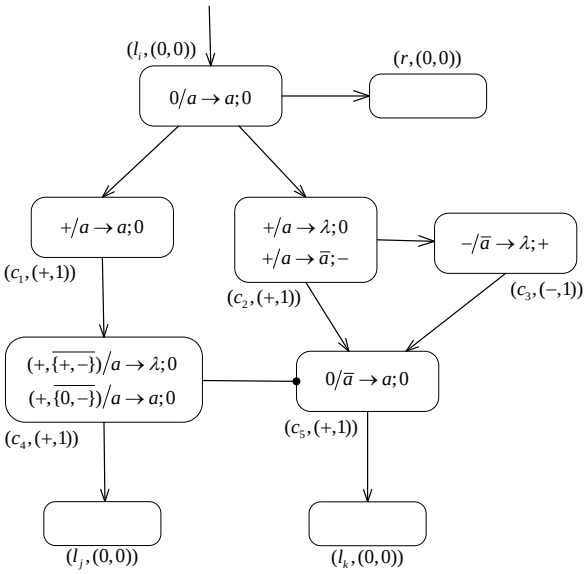

Figure 4. The ADD module of system Π_1.

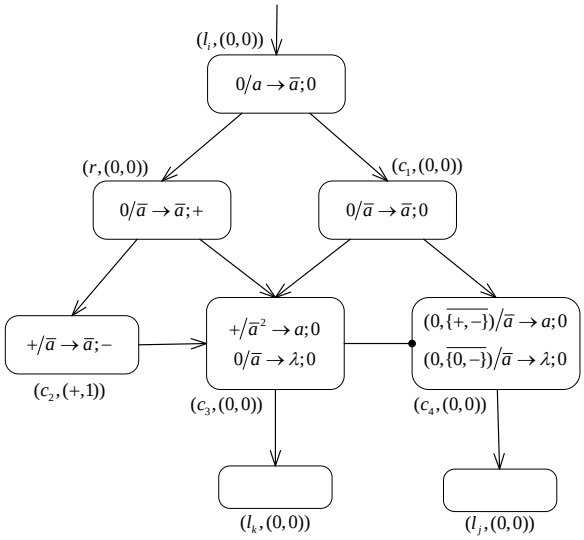

Figure 5. The SUB module of system Π_1.

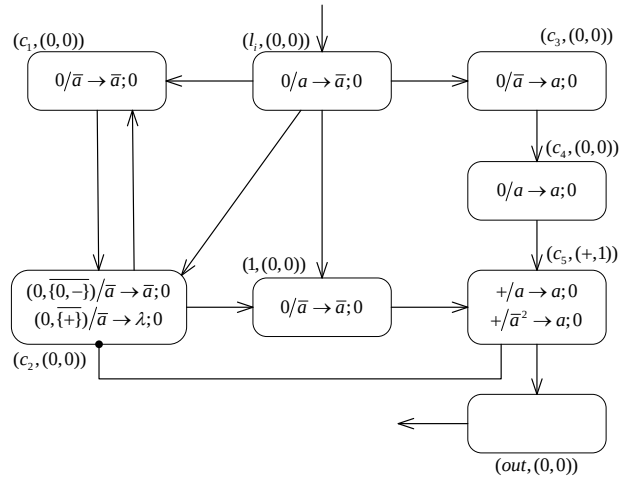

Figure 6. The FIN module of system Π_1.

(1) The ADD module (Figure 4) simulates an ADD instruction $l_i : (ADD(r), l_j, l_k)$.

Suppose that at step t, an ADD instruction l_i is triggered and neuron σ_{l_i} picks up one spike available for firing. Then, at step $t+1$, σ_{c_1}, σ_{c_2} and σ_r all contain one spike derived from the firing of neuron σ_{l_i}. Following this, neuron σ_{c_1} fires by applying the rule $+/a \to a;0$, while one of rules $+/a \to \lambda;0$ and $+/a \to \bar{a};-$ can be applied indeterministically in neuron σ_{c_2}. The following two cases are possible.

- If the forgetting rule $+/a \to \lambda;0$ is applied, then there is one spike in neuron σ_{c_4} with positive polarization, as well as its inhibitory neuron σ_{c_5} also carries positive polarization. Then, the triggering condition of the inhibitory spiking rule $(+, \overline{\{0,-\}})/a \to a;0$ is satisfied, i.e., $PAR_4 = + \wedge PAR_2 \notin \{0,-\}$. As soon as neuron σ_{l_j} receives one spike from neuron σ_{c_4}, this simulated computational procedure activates the module associated with instruction l_j.
- If the rule applied in neuron σ_{c_2} is $+/a \to \bar{a};-$, then at step $t+2$, there is an anti-spike contained in neuron σ_{c_5} with neutral polarization. That is, rule $0/\bar{a} \to a;0$ is applied inside neuron σ_{c_5} and one spike generated is sent to neuron σ_{l_k}. Under such a case, neuron σ_{c_4} is reset to empty by applying the inhibitory forgetting rule $(+, \overline{\{+,-\}})/a \to a;0$. The neuron σ_{c_5} then receives one positive charge generated by the application of the forgetting rule $-/\bar{a} \to \lambda;+$ in neuron σ_{c_3} to restore its initial polarized state. Thus, this simulation computation activated the module associated with instruction l_k.

In summary, the ADD module successfully simulates the execution of the ADD instruction $l_i : (ADD(r), l_j, l_k)$. The acceptance of one spike by neuron σ_{l_i} starts the computation of the ADD module, followed by the implementation of adding one to the number of spikes contained in the neuron σ_r, corresponding to adding one to the value in the register r in the machine. Sending one spike to neuron σ_{l_j} or σ_{l_k} by indeterminacy corresponds to the indeterminate execution of instruction l_j or l_k.

(2) The SUB module (Figure 5) simulates a SUB instruction $l_i : (SUB(r), l_j, l_k)$.

Suppose that at step t, neuron σ_{l_i} receives one spike and fires. Following that, as neuron σ_{c_1} fires, both neurons σ_{c_3} and σ_{c_4} contain one anti-spike from neuron at step $t+2$. Again, depending on the value stored in register r, corresponding to changes occurring in neuron σ_r and the operation of the module, the following two cases exist.

- If the value in register r is 0, and accordingly, the neuron σ_r is empty, then at step $t+1$, neuron σ_r consumes one anti-spike using the rule $0/\bar{a} \to \bar{a};+$ to transmit one positive charge and one anti-spike to neurons σ_{c_2} and σ_{c_3}. At moment $t+2$, neuron σ_{c_3} has two unconsumed anti-spikes with positive polarization, satisfying the firing condition of rule $+/\bar{a}^2 \to a;0$, enabling one spike to be sent to neuron σ_{l_k}. In addition, the neuron σ_{c_4}, which contains one anti-spike with neutral polarization, has the triggering condition of its inhibitory forgetting rule $(0, \overline{\{0,-\}})/\bar{a} \to \lambda;0$ satisfied, i.e., $PAR_{c_4} = 0 \wedge PAR_{c_3} \notin \{0,-\}$, and then it becomes empty without generating spikes or charges upon the application of the rule. At this point, neuron σ_{c_2} also fires with the application of the rule $+/\bar{a} \to \bar{a};-$. At the next step, neuron σ_{c_3} receives one negative charge from neuron σ_{c_2} and becomes neutral, which means that it reverts to its initial polarization. The received anti-spike is then consumed by the application of the forgetting rule $0/\bar{a} \to \lambda;0$, and neuron σ_{c_3} eventually becomes empty. In general, this process activates the computational module associated with the instruction l_k.
- If the value in register r is $n \geq 1$ and correspondingly neuron σ_r contains n spikes, then at step $t+1$, after receiving an anti-spike from neuron σ_{l_i}, neuron σ_r consumes one spike by the application of the annihilation rule $a\bar{a} \to \lambda$, and the number of spikes it contains becomes $n-1$. At step $t+2$, neuron σ_{c_3} only receives one anti-spike from neuron σ_{c_3} and maintains neutral polarization, and then it becomes empty after applying the forgetting rule $0/\bar{a} \to \lambda;0$. In contrast, the neuron σ_{c_4}, which contains one anti-spike and carries neutral polarization, satisfies simultaneously with its inhibitory neuron the trigger condition $PAR_{c_4} = 0 \wedge PAR_{c_3} \notin \{+,-\}$ of the inhibitory spiking

rule $(0, \overline{\{+,-\}})/\bar{a} \to a;0$, so that one spike can be sent to neuron σ_{l_j}. Then, the process activates the computational module associated with the instruction l_j.

In summary, the SUB module successfully simulates the SUB instruction $l_i : (SUB(r), l_j, l_k)$. Similarly, starting from the acceptance of one spike by neuron σ_{l_i}, the instruction l_j or l_k successively is simulated depending on the number of spikes ($n = 0$ or $n \geq 1$) contained in neuron σ_r, respectively.

Remarkably, during the simulated computation of systematic instructions $l_i : (SUB(r), l_j, l_k)$ by system Π_1, there is an unavoidable situation that multiple instructions act on the same register according to the different instruction labels, so that mutual interference exists in the computation between the SUB modules. Specifically, for the simulated instruction $l_i : (SUB(r), l_j, l_k)$ of the SUB module shown in Figure 4, there may be another instruction l_s acting on the register r as well. Then, during the simulation of instruction l_i, for the case where the value stored in register r is 0, the corresponding neuron σ_r sends both an anti-spike and a positive charge to auxiliary neurons σ_{c_2} and σ_{c_3} in the SUB module associated with instruction l_s. Thus neuron σ_{c_3} contains one anti-spike and carries positive polarization, which is not enough to apply the spiking rule $+/\bar{a}^2 \to a;0$. As for neuron σ_{c_2}, it can apply rule $+/\bar{a} \to \bar{a};-$ to send an anti-spike and a negative charge to neuron σ_{c_3}. Subsequently, neuron σ_{c_3} contains two anti-spikes with neutral polarization, turning empty after applying the standard forgetting rule $0/\bar{a} \to \lambda;0$. As a result, the auxiliary neurons σ_{c_2} and σ_{c_3} in the SUB module associated with instruction l_s affected by the firing of neuron σ_r can be restored to their initial state. In summary, the interference between the SUB modules does not affect the correctness of the computation process.

(3) The FIN module (Figure 6) outputs the computation result.

If the computation process operates up to the instruction l_h, the computation is finished, and the result is output by the FIN module. At step t, neuron σ_{l_h} receives one spike, and the FIN module is activated. At step $t+1$, neuron σ_{c_3} receives the anti-spikes from neuron σ_{l_h} , and the execution of the rule $0/\bar{a} \to a;0$ generates one spike. Neurons σ_{c_4}, σ_{c_5}, and the output neuron σ_{out} fire in the subsequent step. At step $t+4$, the output neuron σ_{out} sends its first generating spike into the environment. In addition, starting from step $t+1$, neuron σ_{c_1} consumes one anti-spike for firing, neuron σ_{c_2} and its inhibitory neuron σ_{c_5} also satisfy the triggering condition of the inhibitory spiking rule $(0, \overline{\{0,-\}})/\bar{a} \to \bar{a};0$, i.e., $PAR_{c_2} = 0 \wedge PAR_{c_5} \notin \{0,-\}$. Then neurons σ_{c_1} and σ_{c_2} begin the process of exchanging one anti-spike with each other, and they continuously fire. Additionally, at each subsequent step, neuron σ_{c_2} sends one anti-spike to neuron σ_1, gradually annihilating the n spikes contained inside it.

Until step $t+n$, neuron σ_1 becomes empty, but continues to receive anti-spikes from neuron σ_{c_2}. Then, at step $t+n+1$, the neuron σ_1 is able to trigger the rule $0/\bar{a} \to \bar{a};0$, and sends one anti-spike to neuron σ_{c_5}. The excitation condition of the inhibitory spiking rule in neuron σ_{c_2} is then not satisfied at step $t+n+2$, as the polarization of neuron σ_{c_5} changes to neutral. However, the inhibitory forgetting rule $(0, \overline{\{+\}})/\bar{a} \to \lambda;0$ is eventually triggered, leaving the neuron σ_{c_2} empty. At step $t+n+3$, the neuron σ_{c_5} contains a total of two anti-spikes and carries negative polarization, allowing the rule $-/\bar{a}^2 \to a;0$ to be applied. Thus, the second spike is sent to the environment by the neuron σ_{out} at step $t+n+4$. The calculation result output by this module is $(t+n+4)-(t+4) = n$, which corresponds to the value in register 1.

In summary, the MPAIRSN P system Π_1 correctly simulates the register machine M_1 working in the generating mode, which applies three types of polarizations, with, at most, two rules per neuron in the system. Therefore, Theorem 1 holds. □

The proposal of the MPAIRSN P system is an improvement of the spiking neural P system with polarization, by introducing membrane potentials to complete the application of polarization in the SN P system, which makes the model more consistent with biological facts. Therefore, we compare the required resources (Table 1) of the four SN P systems, using polarizations as number generating devices. The comparison data are taken from

the references. For the computation resources, the comparisons are the maximum number of rules contained in each neuron, and the number of auxiliary neurons required for each instruction module.

Table 1. Comparison of the extended SN P systems, with polarizations as number generating devices.

Computing Models	Configuration			
	Maximum Number of Rules per Neuron	Auxiliary Neurons (ADD)	Auxiliary Neurons (SUB)	Auxiliary Neurons (FIN)
MPAIRSN P systems	2	5	4	5
PSN P systems [71]	2	5	5	7
PASN P systems [72]	2	4	6	6
SNP–MCP systems [76]	2	4	6	1
PSNRS P systems [77]	2	7	8	5

In terms of the number of neurons needed in each module, the MPAIRSN P system and the SNP–MCP system use a fewer number of neurons compared to the remaining three SN P systems with polarizations. The FIN module, which is used to output the result, is applied only once at the end of the computation process. Then, it is mainly the number of neurons in the ADD and SUB modules that affect the size of the computation resources. From the average value of the auxiliary neurons in the two modules, the MPAIRSN P systems require fewer computational resources.

3.2. The MPAIRSN P System as a Number Accepting Device

The register machine M, operating in accepting mode, serves to receive the natural number n, with all registers contained being empty in the initial state. The machine introduces the number n to be analyzed from the environment, stores the value n in the first register, and then starts the computational work from the execution instruction l_0. If the computation processing of the machine reaches the termination instruction l_h, the number n is considered to be received by the machine M_2. The set of numbers that can be received by the machine M is recorded as $N_{acc}(M)$.

Theorem 2. $N_2SNP^2_*(ch_3) = NRE$.

Proof. As is similar to the proof of Theorem 1, we only need to prove that the expression $N_{acc}SNP^2_* \subseteq NRE$ holds. To simulate the register machine $M_2 = (m, H, l_0, l_h, I)$ operating in accepting mode, we design the MPAIRSN P system Π_2, which serves as a number accepting device. The system Π_2 mainly consists of three types of computing modules: the deterministic ADD module, the SUB module, and the INPUT module.

(1) The INPUT module (Figure 7) introduces the number n into the system.

Suppose that at step t, the spikes sequence $10^{n-1}1$ is introduced into the INPUT module of system Π_2. Then, the neuron σ_{in} receives the first spike from the environment and applies the rule $0/a \to a; -$ to fire. After receiving a negative charge from neuron σ_{in}, neuron σ_{c_1} takes the neutral polarization. The membrane potentials of neurons σ_{c_2} and σ_{c_3} both become -2, where one spike is contained in every neuron. Then, during step $t+2$ and subsequent computation steps, since the formula $PAR_{c_2} = - \wedge PAR_{c_1} \notin \{-\}$ always holds, neuron σ_{c_2} can continuously apply the inhibitory spiking rule $(-, \overline{\{-\}})/a \to a; 0$ to fire and transmit one spike to the neurons σ_{c_2} and σ_1. At the same time, neuron σ_{c_3} also applies the rule $-/a \to a; 0$ to continuously supply spikes to neuron σ_{c_2}. Thus, there is a constant exchange of spikes between neurons σ_{c_2} and σ_{c_3}, and each step adds one to the number of spikes contained in neuron σ_1.

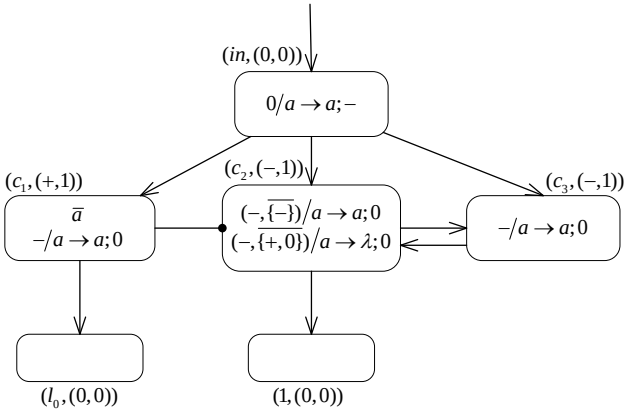

Figure 7. The INPUT module of system Π_2.

At step $t+n$, neuron σ_{in} receives the second spike from the spikes sequence, and the rule $0/a \to a;-$ is applied again to generate one negative charge and one spike. Then at step $t+n+1$, neuron σ_{c_1} contains one spike and carries the negative polarization. The rule $-/a \to a;0$ is triggered, which sends a generated spike to neuron σ_{l_0}. However, neuron σ_{c_2} is no longer able to apply the inhibitory spiking rule, but satisfies $PAR_{c_2} = - \wedge PAR_{c_1} \notin \{+,0\}$. The inhibitory forgetting rule $(-,\overline{\{+,0\}})/a \to \lambda;0$ is triggered and leaves the neuron σ_{c_2} eventually empty. At this point, there are, in total, n spikes stored in neuron σ_1. At the step $t+n+2$, neuron σ_{l_0} receives one spike, which excites the module associated with instruction l_0 and thus starts the analog computation, while the number n is introduced into the neuron σ_1 in the system.

(2) The deterministic ADD module (Figure 8) to simulate a deterministic ADD instruction $l_i : (ADD(r), l_j)$.

The computation process of the register machine M_2 in the accepting mode is deterministic and uses deterministic ADD instructions. Therefore, to achieve system simplification, the ADD module no longer applies inhibitory rules. Suppose that at step t neuron σ_{l_i} receives one spike, and the system starts to simulate instruction $l_i : (ADD(r), l_j)$. by applying rule $0/a \to a;0$ to send one spike to neurons σ_{l_j} and σ_r. Then, at step $t+1$, neuron σ_{l_j} is activated, and the system starts to simulate instruction l_j while the number of spikes contained in neuron σ_r is added by one.

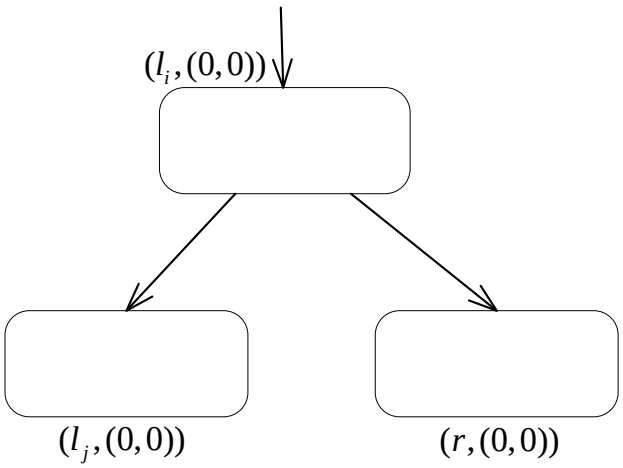

Figure 8. The deterministic ADD module of system Π_2.

The SUB module, as shown in Figure 5, remains available for system Π_2. Meanwhile, in system Π_2, the FIN module is omitted, but the neuron σ_{l_h} corresponding to the halt

instruction is still retained to determine the acceptance result. The simulation of the system is terminated when one spike is received by neuron σ_{l_h}.

In summary, system Π_2 successfully simulates the register machine M_2 operating in the accepting mode. In total, three types of polarizations are applied in the system, and there are, at most, two rules contained in each neuron. Therefore, Theorem 2 is proven. □

Similarly, we present Table 2 to compare the resources of the MPAIRSN P systems with three SN P systems containing polarizations as number accepting devices. As can be seen, the MPAIRSN P system and the PASN P system use the least number of neurons.

Table 2. Comparison of the extended SN P systems with polarizations as number accepting devices.

Computing Models	Computation Resources		
	Maximum Number of Rules per Neuron	Auxiliary Neurons (ADD)	Auxiliary Neurons (INPUT)
MPAIRSN P systems	2	0	3
PASN P systems [72]	2	0	3
SNP–MCP systems [76]	2	2	5
PSNRS P systems [77]	2	2	7

4. A Small General MPAIRSN P System for Function Computation

This section constructs a small general MPAIRSN P system that performs function calculations.

Theorem 3. *There is a Turing universal MPAIRSN P system using three types of polarizations with 95 neurons for computing functions.*

Proof. Similarly, we prove the theorem by designing the MPAIRSN P system to simulate the register machine $M_c = (m, H, l_0, l_h, I)$. The machine serves to compute a function $f : N^k \to N$, in which, in the initial state, k specified registers are used to store k parameters (usually, the first k registers $r_1, r_2, \cdots r_k$ are selected), and the rest of the registers are empty. The machine operates from the start instruction l_0, followed by the execution of different instructions for processing, and the computation stops when it reaches the halt instruction l_h. At this point, the value of function f is stored in the specified register r_t, denoted by $M_c(n_1, n_2, \cdots, n_k)$, where $n_1, n_2, \cdots, n_k$ is k parameters. In addition, the machine used for the functions calculation works in a deterministic manner, so the ADD instruction form used is $l_i : (ADD(r), l_j)$. If it is assumed that $(\varphi_0, \varphi_1, \cdots)$ a fixed enumeration of unary partial recursive functions, for the machine M_c to be universal as a function computing device, it is necessary that there exists a recursive function g, such that the equation $\varphi_x(y) = M_c(g(x), y)$ holds for all natural numbers x, y.

According to the small universal register machine $M_u = (8, H, l_0, l_h, I)$ proposed by Korec [81], it contains eight registers (marked from 0 to 7) and 23 instructions. The machine firstly stores the parameters $g(x)$ and y in registers 1 and 2, respectively, and the computation result $\varphi_x(y)$ is stored in register 0. Similarly, an MPAIRSN P system Π_3 is constructed to simulate the register machine M_u. To facilitate the system construction, the machine M_u is modified by adding a new register 8 and replacing the original halt instruction with the instructions $l_{22} : (SUB(0), l_{23}, l_h), l_{23} : (ADD(8), l_{22}), l_h : HALT$. The modified register machine M'_u is shown in Figure 9, containing 24 ADD instructions and SUB instructions, 9 registers, and 25 labels.

$l_0 : (SUB(1), l_1, l_2)$ $l_1 : (ADD(7), l_0)$

$l_2 : (ADD(6), l_3)$ $l_3 : (SUB(5), l_2, l_4)$

$l_4 : (SUB(6), l_5, l_3)$ $l_5 : (ADD(5), l_6)$

$l_6 : (SUB(7), l_7, l_8)$ $l_7 : (ADD(1), l_4)$

$l_8 : (SUB(6), l_9, l_0)$ $l_9 : (ADD(6), l_{10})$

$l_{10} : (SUB(4), l_0, l_{11})$ $l_{11} : (SUB(5), l_{12}, l_{13})$

$l_{12} : (SUB(5), l_{14}, l_{15})$ $l_{13} : (SUB(2), l_{18}, l_{19})$

$l_{14} : (SUB(5), l_{16}, l_{17})$ $l_{15} : (SUB(3), l_{18}, l_{20})$

$l_{16} : (ADD(4), l_{11})$ $l_{17} : (ADD(2), l_{21})$

$l_{18} : (SUB(4), l_0, l_{22})$ $l_{19} : (SUB(0), l_0, l_{18})$

$l_{20} : (ADD(0), l_0)$ $l_{21} : (ADD(3), l_{18})$

$l_{22} : (SUB(0), l_{23}, l_h)$ $l_{23} : (ADD(8), l_{22})$

$l_h : HALT$

Figure 9. The universal register machine M'_u.

The overall composition of the small general MPAIRSN P system Π_3 (Figure 10) contains three main parts: the first is the INPUT module for reading the parameters encoded as spike sequences from the environment, the second is the simulation part of the register machine, which contains several deterministic ADD modules and SUB modules, and the third is the OUTPUT module for outputting the calculation results. The operation of the system begins with the parameters introduction phase, in which the neuron σ_{in} in the INPUT module reads the spikes sequence. The spikes of amount $g(x)$ and the spikes of amount y are introduced into neurons σ_1 and σ_2, respectively, as well as activating neuron σ_{l_0}. The system then proceeds to the register machine simulation phase. In this phase, a series of calculations are performed on the introduced parameters by simulating various instructions, and the final result is stored in neuron σ_8. Until neuron σ_{l_h} is activated, the system calculation enters the result output phase. The neuron σ_{out} outputs spikes to the environment in the OUTPUT module. We record the entire spikes received in the environment as the output result, corresponding to the value of the calculation result stored in register 8.

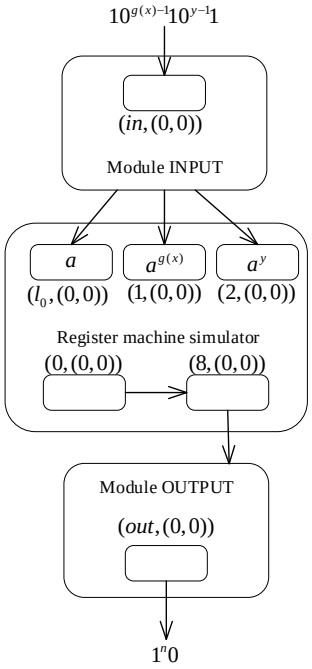

Figure 10. The overall construction of the MPAIRSN P system Π_3.

(1) The INPUT module (Figure 11) for reading the spikes sequence $10^{g(x)}10^{y}1$ of encoded parameters from the environment.

Suppose that at step t, neuron σ_{in} receives the first spike in the sequence from the environment and fires. Then at step $t+1$, influenced by the firing of neuron σ_{in}, neurons σ_{c_2}, σ_{c_3}, and σ_{c_4} all change to carry the neutral polarization. Thus, neurons σ_{c_3} and σ_{c_4} are available to trigger rules $0/a \to \bar{a};0$ and $0/\bar{a} \to a;0$, respectively, and the process of exchanging a spike and an anti-spike is continuously implemented in the subsequent steps. At the same time, one spike is continuously transmitted to neuron σ_1 at each step. However, for neuron σ_{c_2}, although it continuously receives spikes from neuron σ_{c_3}, all these spikes are deleted by the application of its internal forgetting rule $0/\bar{a} \to \lambda;0$.

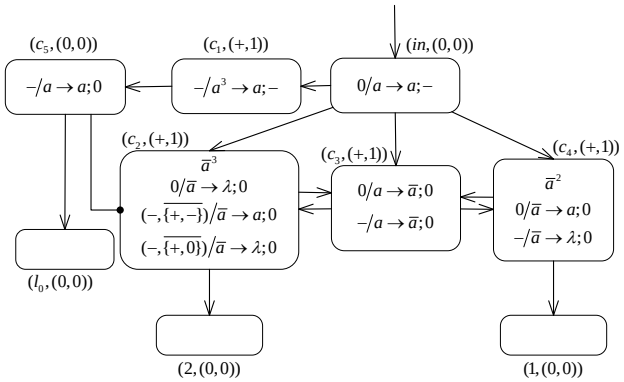

Figure 11. The INPUT module of system Π_3.

The above process proceeds until step $t+g(x)$, neuron σ_{in} receives the second spike in the spikes sequence and fires, and its firing produces one spike and a negative charge sent to neurons σ_{c_1}, σ_{c_2}, σ_{c_3}, and σ_{c_4}. Subsequently, the polarization states of neurons σ_{c_2}, σ_{c_3}, and σ_{c_4} all turn negative. Then, the rule capable of triggering in neuron σ_{c_4} is the forgetting rule $-/\bar{a} \to \lambda;0$, and it is no longer sending spikes to neuron σ_1. At this point, the number of stored spikes in neuron σ_1 is exactly g(x). Additionally, the neuron σ_{c_2} satisfies the triggering condition of its internal inhibitory rule $(-,\overline{\{+,-\}})/\bar{a} \to a;0$, i.e., the formula $PAR_{c_2} = - \wedge PAR_{c_5} \notin \{+,-\}$ holds. Meanwhile, the rule applied by neuron σ_{c_3} with negative polarization is $-/a \to \bar{a};0$. Then, in the subsequent steps, the mutual replenishment of spikes and ani-spikes between neurons σ_{c_2} and σ_{c_3} is achieved, and one spike is sent from neuron σ_{c_2} to neuron σ_2 at each step.

At step $t+g(x)+y$, the neuron σ_{in} receives the last spike in the spikes sequence and fires. Then, the polarization state of neuron σ_{c_1} and the number of spikes contained are sufficient for triggering the rule $-/a^3 \to a;-$. At step $t+g(x)+y+2$, the polarization state of neuron σ_{c_5} becomes negatively influenced by the firing of the neuron σ_{c_1}. Additionally, the received spike in neuron σ_{c_1} is subsequently consumed by the application of rule $-/a \to a;0$. In the subsequent computation steps, the spikes or anti-spikes contained in neurons σ_{c_2}, σ_{c_3}, and σ_{c_4} are cleared by the application annihilation rule, the forgetting rule, and the inhibitory forgetting rule, respectively. The neuron σ_{l_0} is activated at step $t+g(x)+y+3$, while the numbers of spikes stored in neurons σ_1 and σ_2 at this step are $g(x)$ and y, respectively. The above procedure successfully simulates the introduction of parameters $g(x)$ and y into registers 1 and 2, respectively, and starts the simulation of the system Π_3.

Similarly, the ADD module, as shown in Figure 7, is applied in system Π_3 to simulate the deterministic ADD instruction of the machine M'_u. The SUB module, as shown in Figure 4, remains available for system Π_3 to simulate the SUB instruction. The proof process has been illustrated in detail in the previous section.

(2) The OUTPUT module (Figure 12) outputs the result of the system calculation.

Assuming that the neuron σ_{l_h} is triggered and fires at step t, then at step $t+1$, after receiving an anti-spike from neuron σ_{l_h}, the neuron σ_8 consumes one spike by annihilation,

where the number of spikes contained turns to $n-1$. Neurons σ_{c_1} and σ_{c_2} are allowed to trigger the spiking rule $0/\bar{a} \to \bar{a};0$ and the inhibitory spiking rule $(0,\overline{\{0,-\}})/\bar{a} \to \bar{a};0$, respectively, thus enabling the two neurons to replenish spikes to each other and to fire at each step. One spike is sent by neuron σ_{c_2} to neuron σ_8 at each step to successively annihilate its internal spikes. In addition, neuron σ_{c_3} also receives an anti-spike from neuron σ_{c_1} at each step, continuously applying rule $+/\bar{a} \to a;0$ to fire. Consequently, the neuron σ_{c_3} feeds one spike to neuron σ_{out} at each step, which subsequently triggers the rule $0/a \to a;0$ in neuron σ_{out}. The spikes generated by neuron σ_{out} at each step are received in the environment, starting at step $t+4$.

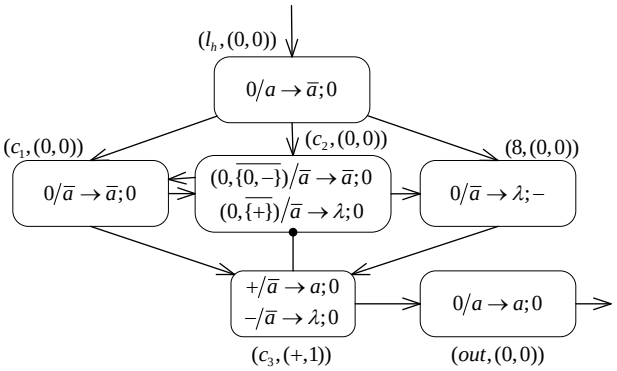

Figure 12. The OUTPUT module of system Π_3.

The above process continues until step $t+n$, when the spikes contained in neuron σ_8 are completely annihilated and left empty, but one anti-spike from neuron σ_{c_2} is still received. Subsequently, the neuron σ_8 applies rule $0/\bar{a} \to \lambda;-$ to fire, and delivers a negative charge to neuron σ_{c_3} to change its polarization. The neuron σ_{c_3} only applies the forgetting rule $-/\bar{a} \to \lambda;0$ after its polarization changes to negative without sending any spikes to the neuron σ_{out}. In addition, influenced by the changed polarization state of neuron σ_{c_3}, i.e., the formula $\alpha_{c_2} = 0 \wedge \alpha_{c_3} \notin \{+\}$ holds, only the inhibitory forgetting rule $(0,\overline{\{+\}})/\bar{a} \to \lambda;0$ is triggered in neuron σ_{c_2}, ensuring subsequent steps to clear neurons σ_{c_1} and σ_{c_2}. At step $t+n+3$, the environment receives the last spike from neuron σ_{out}. The cumulative number of spikes received is $(t+n+3)-(t+4)+1=n$, which is exactly the number of spikes contained in neuron σ_8. That is, the module successfully outputs the computation result stored in register 8.

To summarize, the small universal proposed MPAIRSN P system Π_3 is capable of correctly simulating the universal register machine M'_u used for function computation. The required computational resources for this system are 98 neurons, which can be subdivided as shown in Table 3.

Table 3. Computing resources required for the PAWSN P system Π_3.

Components of the PAWSN P System Π_3	Number of Neurons
Registers	9
Instruction labels	25
Auxiliary neurons required for SUB modules	56
Neurons required for INPUT modules	5
Neurons required for OUTPUT modules	3

Moreover, observing the characteristics of the instructions of the universal register machine M'_u applied in Figure 9, the number of neurons needed for system Π_3 is reduced by integrating the SUB and ADD modules, which correspond to consecutive pairs of instructions.

(1) As for a pair of consecutive ADD instructions: $l_{17} : (ADD(2), l_{21})$ and $l_{21} : (ADD(3), l_{18})$, the ADD-ADD module, as shown in Figure 13, is constructed for simulation. In this way, the neuron $\sigma_{l_{21}}$ in the system is omitted.

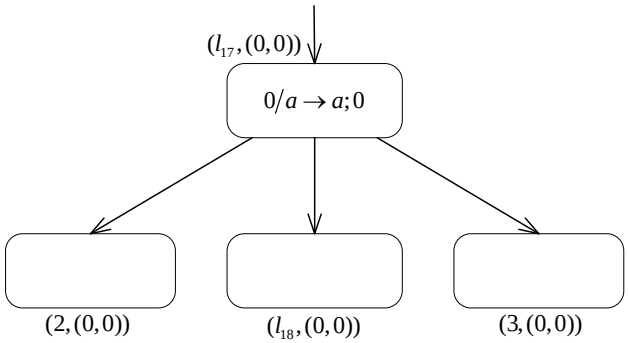

Figure 13. The ADD-ADD module of system Π_3.

(2) For two consecutive pairs of ADD-SUB instructions in M'_u: $l_5 : (ADD(5), l_6)$ and $l_6 : (SUB(7), l_7, l_8)$, $l_9 : (ADD(6), l_{10})$ and $l_{10} : (SUB(4), l_0, l_{11})$, both having the same form: $l_i : (ADD(r_1), l_j)$ and $l_j : (SUB(r_2), l_g, l_k)$. For this purpose, the ADD-SUB module, as shown in Figure 14, allows the for merging of two modules. In this way, neurons σ_{l_6} and $\sigma_{l_{10}}$ are omitted.

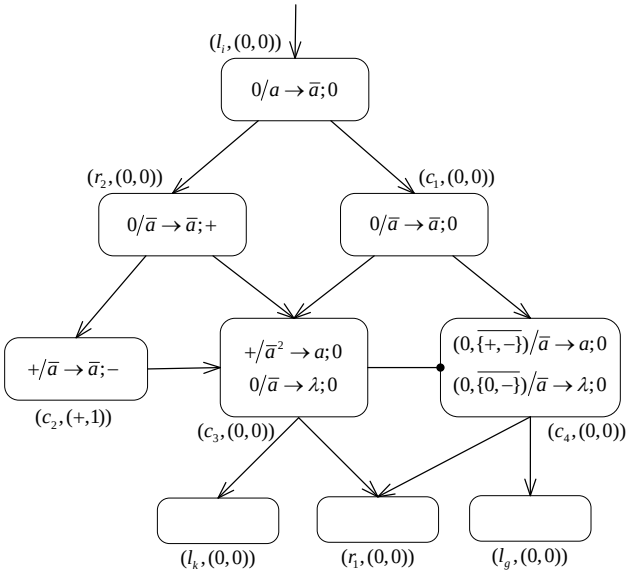

Figure 14. The ADD-SUB module of system Π_3.

Thus, with the construction of the corresponding composite modules for consecutive pairs of instructions, the total number of neurons used by the system can be reduced by 3: one neuron is saved by the application of the ADD-ADD module, and two neurons are saved by the application of the ADD-SUB module. Thus, the system for function computation achieves universality by applying three types of polarizations and 95 neurons. In summary, Theorem 3 is proven. □

For the required computation resources and the computation speed, Table 4 shows the result of the proposed MPAIRSN P systems with four extended SN P systems, with polarizations for function computing. It should be noted that the computation resources here are the total number of neurons required for the systems to implement the function computation. The computation speed is the number of steps required by completing the

computation. As is seen, compared to the remaining four SN P systems, the number of neurons used in the proposed MPAIRSN P system is still minimal. Moreover, the MPAIRSN P systems require the least number of steps to perform the computation, i.e., the computation is relatively faster.

Table 4. Comparison of the extended SN P systems with polarizations for computing functions.

Computing Models	Number of Neurons	Computation Speed
MPAIRSN P systems	95	g(x) + y + n + 58
PSN P systems [71]	164	g(x) + y + n + 128
PASN P systems [72]	121	g(x) + y + n + 72
SNP–MCP systems [76]	150	g(x) + y + n + 66
PSNRS P systems [77]	151	g(x) + y + n + 93

5. Conclusions

In this paper, the concept of the membrane potential of nerve cells is introduced into the spiking neural P system, and the biological phenomena of charge accumulation and transmission between neurons are considered based on the spiking neural P systems with polarizations. Combining these ideas, a new variant called spiking neural P systems with membrane potential, inhibitory rules, and anti-spikes (MPAIRSN P systems for short) is proposed. The system adopts membrane potential as the rule-triggering condition and uses inhibitory rules to simulate the role of inhibitory synapses in the nervous system, which makes the construction and operation of the system more consistent with biological facts. The application of anti-spikes makes the system available with another information-encoding object, which simplifies the system construction and enhances the information representation capability of the system. The introduction of membrane potentials and the application of inhibitory rules provide the MPAIRSN P systems with more powerful control in computation.

We first give a formal definition of the MPAIRSN P system, and we update the novel inhibitory rule with membrane potential as the trigger condition. The conventions for neuronal charge calculation are given for determining neuron-associated polarization. By simplifying rules with polarization as the trigger condition, we design a small MPAIRSN P system as an illustrative example to detail its operation, which is equipped with the ability to generate arbitrary non-zero natural numbers. We demonstrate that MPAIRSN P systems are Turing-universal, as both a number generating and accepting device. A small universal MPAIRSN P system using 95 neurons is obtained, which saves 69 neurons compared to the initial spiking neural P systems with polarizations. By comparing the MPAIRSN P systems with other variants of the SN P systems with polarizations, it is shown that the MPAIRSN P systems have better performance, both in space efficiency and computation speed.

The MPAIRSN P system constructed in this paper mainly applies the standard spiking rules, and it is worth considering the use of extended rules in the system. It is also worthwhile to further investigate the computational power of the MPAIRSN P systems operating in other modes, such as sequential mode, asynchronous mode, etc. To reduce the number of polarizations and neurons needed, further research could consider introducing more biological features into the system without losing computational power, which allows the system to better approximate the biological facts. The spiking neural P systems with membrane potential, as a new variant, introduce a novel systematic rule-triggering mechanism, which is of novelty and development potential. It is worth considering the application of the spiking neural P systems with membrane potential to solve NP problems. Moreover, given that the MPAIRSN P systems provide powerful control over computation, it is practicable to apply them to real-world problems such as supervisory control and fault diagnosis.

Author Contributions: Conceptualization, Y.L. and Y.Z.; methodology, Y.L. and Y.Z.; software, Y.L.; validation, Y.L.; formal analysis, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and Y.Z.; supervision, Y.L.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 61806114, 61472231, 61876101, 61602282, 61402187, 61502283, 61802234, and 61703251), and the China Postdoctoral Science Foundation (No. 2018M642695 and 2019T120607).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No datasets were used in this article.

Conflicts of Interest: The authors declare no conflict to interest.

References

1. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. [CrossRef]
2. Păun, G. Computing with membranes. *J. Comput. Syst. Sci.* **2000**, *61*, 108–143. [CrossRef]
3. Liu, L.; Yi, W.; Yang, Q.; Peng, H.; Wang, J. Small Universal Numerical P Systems with Thresholds for Computing Functions. *Fundam. Inform.* **2020**, *176*, 43–59. [CrossRef]
4. Zhang, Z.; Su, Y.; Pan, L. The computational power of enzymatic numerical P systems working in the sequential mode. *Theor. Comput. Sci.* **2018**, *724*, 3–12. [CrossRef]
5. Liu, L.; Yi, W.; Yang, Q.; Peng, H.; Wang, J. Numerical P systems with Boolean condition. *Theor. Comput. Sci.* **2019**, *785*, 140–149. [CrossRef]
6. Jiang, S.; Wang, Y.; Xu, F.; Deng, J. Communication P Systems with Channel States Working in Flat Maximally Parallel Manner. *Fundam. Inform.* **2019**, *168*, 1–24. [CrossRef]
7. Zeng, X.; Pan, L.; Perez-Jimenez, M.J. Small universal simple spiking neural P systems with weights. *Sci. China-Inf. Sci.* **2014**, *57*, 1–11. [CrossRef]
8. Zhang, Z.; Wu, T.; Păun, A.; Pan, L. Universal enzymatic numerical P systems with small number of enzymatic variables. *Sci. China-Inf. Sci.* **2018**, *61*, 1–12. [CrossRef]
9. Song, B.; Kong, Y. Solution to PSPACE-Complete Problem Using P Systems with Active Membranes with Time-Freeness. *Math. Probl. Eng.* **2019**, *2019*, 5793234. [CrossRef]
10. Pan, L.; Song, B. P Systems with Rule Production and Removal. *Fundam. Inform.* **2020**, *171*, 313–329. [CrossRef]
11. Orellana-Martín, D.; Valencia-Cabrera, L.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. Membrane Creation in Polarizationless P Systems with Active Membranes. *Fundam. Inform.* **2020**, *171*, 297–311. [CrossRef]
12. Orellana-Martín, D.; Valencia-Cabrera, L.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. The Unique Satisfiability Problem from a Membrane Computing Perspective. *Rom. J. Inf. Sci. Technol.* **2018**, *21*, 288–297.
13. Orellana-Martín, D.; Martínez-del Amor, M.A.; Pérez-Hurtado, I.; Riscos-Núñez, A.; Valencia-Cabrera, L.; Pérez-Jiménez, M.J. When object production tunes the efficiency of membrane systems. *Theor. Comput. Sci.* **2020**, *805*, 218–231. [CrossRef]
14. Buño, K.C.; Cabarle, F.G.C.; Calabia, M.D.; Adorna, H.N. Solving the N-Queens problem using dP systems with active membranes. *Theor. Comput. Sci.* **2018**, *736*, 1–14. [CrossRef]
15. Luo, Y.; Tan, H.; Zhang, Y.; Jiang, Y. The computational power of timed P systems with active membranes using promoters. *Math. Struct. Comput. Sci.* **2019**, *29*, 663–680. [CrossRef]
16. Guo, P.; Zhu, J.; Chen, H.; Yang, R. A Linear-Time Solution for All-SAT Problem Based on P System. *Chin. J. Electron.* **2018**, *27*, 367–373. [CrossRef]
17. Díaz-Pernil, D.; Christinal, H.A.; Gutiérrez-Naranjo, M.A. Solving the 3-COL problem by using tissue P systems without environment and proteins on cells. *Inf. Sci.* **2018**, *430–431*, 240–246. [CrossRef]
18. Ye, L.; Zheng, J.; Guo, P.; Pérez-Jiménez, M.J. Solving the 0–1 Knapsack Problem by Using Tissue P System with Cell Division. *IEEE Access* **2019**, *7*, 66055–66067. [CrossRef]
19. Song, B.; Zeng, X.; Rodríguez-Patón, A. Monodirectional tissue P systems with channel states. *Inf. Sci.* **2021**, *546*, 206–219. [CrossRef]
20. Song, B.; Zeng, X. Solving a PSPACE-complete problem by symport/antiport P systems with promoters and membrane division. *J. Membr. Comput.* **2021**, *3*, 296–302. [CrossRef]
21. Wang, X.; Zhang, G.; Gou, X.; Paul, P.; Neri, F.; Rong, H.; Yang, Q.; Zhang, H. Multi-behaviors coordination controller design with enzymatic numerical P systems for robots. *Integr.-Comput.-Aided Eng.* **2021**, *28*, 119–140. [CrossRef]
22. Xu, J.; Huang, Y.; Liu, Y. Attitude Optimization Control of Unmanned Helicopter in Coal Mine Using Membrane Computing. *Math. Probl. Eng.* **2020**, *2020*, 3820896. [CrossRef]

23. Niu, Y.; Zhang, Y.; Zhang, J.; Xiao, J. Running Cells with Decision-Making Mechanism: Intelligence Decision P System for Evacuation Simulation. *Int. J. Comput. Commun. Control* **2018**, *13*, 865–880. [CrossRef]
24. Sapp, K.; Sodt, A.J.; Maibaum, L. Modeling Relaxation Timescales of Coupled Membrane/Protein Systems. *Biophys. J.* **2019**, *116*, 363a. [CrossRef]
25. Liu, W.; Wang, T.; Zang, T.; Huang, Z.; Wang, J.; Huang, T.; Wei, X.; Li, C. A Fault Diagnosis Method for Power Transmission Networks Based on Spiking Neural P Systems with Self-Updating Rules considering Biological Apoptosis Mechanism. *Complexity* **2020**, *2020*, 2462647. [CrossRef]
26. Gheorghe, M.; Ceterchi, R.; Ipate, F.; Konur, S.; Lefticaru, R. Kernel P systems: From modelling to verification and testing. *Theor. Comput. Sci.* **2018**, *724*, 45–60. [CrossRef]
27. Hao, L.; Liu, J. Enhanced Membrane Computing Algorithm for SAT Problems Based on the Splitting Rule. *Symmetry* **2019**, *11*, 1412. [CrossRef]
28. Zhu, M.; Yang, Q.; Dong, J.; Zhang, G.; Gou, X.; Rong, H.; Paul, P.; Neri, F. An Adaptive Optimization Spiking Neural P System for Binary Problems. *Int. J. Neural Syst.* **2021**, *31*, 20500549. [CrossRef]
29. Ramachandranpillai, R.; Arock, M. Spiking neural firefly optimization scheme for the capacitated dynamic vehicle routing problem with time windows. *Neural Comput. Appl.* **2021**, *33*, 409–432. [CrossRef]
30. Zhao, Y.; Liu, X.; Li, X. An improved DBSCAN algorithm based on cell-like P systems with promoters and inhibitors. *PLoS ONE* **2018**, *13*, e0200751. [CrossRef]
31. Yang, J.; Peng, H.; Luo, X.; Wang, J. Stochastic Numerical P Systems With Application in Data Clustering Problems. *IEEE Access* **2020**, *8*, 31507–31518. [CrossRef]
32. Luo, Y.; Guo, P.; Zhang, M. A Framework of Ant Colony P System. *IEEE Access* **2019**, *7*, 157655–157666. [CrossRef]
33. Liu, X.; Wang, L.; Qu, J.; Wang, N. A Complex Chained P System Based on Evolutionary Mechanism for Image Segmentation. *Comput. Intell. Neurosci.* **2020**, *2020*, 6524919. [CrossRef] [PubMed]
34. Jiang, Z.; Liu, X.; Sun, M. A Density Peak Clustering Algorithm Based on the K-Nearest Shannon Entropy and Tissue-Like P System. *Math. Probl. Eng.* **2019**, *2019*, 1713801. [CrossRef]
35. Jiang, Z.; Liu, X. A Novel Consensus Fuzzy K-Modes Clustering Using Coupling DNA-Chain-Hypergraph P System for Categorical Data. *Processes* **2020**, *8*, 1326. [CrossRef]
36. Guo, P.; Jiang, W.; Liu, Y. A P system for hierarchical clustering. *Int. J. Mod. Phys. C* **2019**, *30*, 1950062. [CrossRef]
37. Chen, G.; Hu, J.; Peng, H.; Wang, J.; Huang, X. A Spectral Clustering Algorithm Improved by P Systems. *Int. J. Comput. Commun. Control* **2018**, *13*, 759–771. [CrossRef]
38. Gou, X.; Liu, Q.; Rong, H.; Hu, M.; Paul, P.; Deng, F.; Zhang, X.; Yu, Z. A Novel Spiking Neural P System for Image Recognition. *Int. J. Unconv. Comput.* **2021**, *16*, 121–139.
39. Yuan, J.; Guo, D.; Zhang, G.; Paul, P.; Zhu, M.; Yang, Q. A Resolution-Free Parallel Algorithm for Image Edge Detection within the Framework of Enzymatic Numerical P Systems. *Molecules* **2019**, *24*, 1235. [CrossRef]
40. Li, B.; Peng, H.; Luo, X.; Wang, J.; Song, X.; Pérez-Jiménez, M.J.; Riscos-Núñez, A. Medical Image Fusion Method Based on Coupled Neural P Systems in Nonsubsampled Shearlet Transform Domain. *Int. J. Neural Syst.* **2021**, *31*, 2050050. [CrossRef]
41. Aman, B.; Ciobanu, G. Spiking Neural P Systems with Astrocytes Producing Calcium. *Int. J. Neural Syst.* **2020**, *30*, 2050066. [CrossRef] [PubMed]
42. Bîlbîe, F.D.; Păun, A. Small SNQ P Systems with multiple types of spikes. *Theor. Comput. Sci.* **2021**, *862*, 14–23. [CrossRef]
43. Zeng, X.; Zhang, X.; Song, T.; Pan, L. Spiking Neural P Systems with Thresholds. *Neural Comput.* **2014**, *26*, 1340–1361. [CrossRef] [PubMed]
44. Peng, H.; Wang, J.; Pérez-Jiménez, M.J.; Riscos-Núñez, A. Dynamic threshold neural P systems. *Knowl.-Based Syst.* **2019**, *163*, 875–884. [CrossRef]
45. Wu, T.; Zhang, L.; Pan, L. Spiking neural P systems with target indications. *Theor. Comput. Sci.* **2021**, *862*, 250–261. [CrossRef]
46. Song, T.; Gong, F.; Liu, X.; Zhao, Y.; Zhang, X. Spiking Neural P Systems with White Hole Neurons. *IEEE Trans. Nanobiosci.* **2016**, *15*, 666–673. [CrossRef]
47. Wu, T.; Lyu, Q.; Pan, L. Evolution-Communication Spiking Neural P Systems. *Int. J. Neural Syst.* **2021**, *31*, 20500641. [CrossRef]
48. Yang, Q.; Li, B.; Huang, Y.; Peng, H.; Wang, J. Spiking neural P systems with structural plasticity and anti-spikes. *Theor. Comput. Sci.* **2020**, *801*, 143–156. [CrossRef]
49. Cabarle, F.G.C.; Adorna, H.N.; Pérez-Jiménez, M.J.; Song, T. Spiking neural P systems with structural plasticity. *Neural Comput. Appl.* **2015**, *26*, 1905–1917. [CrossRef]
50. Lv, Z.; Bao, T.; Zhou, N.; Peng, H.; Huang, X.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. Spiking Neural P Systems with Extended Channel Rules. *Int. J. Neural Syst.* **2021**, *31*, 20500495. [CrossRef]
51. Song, X.; Valencia-Cabrera, L.; Peng, H.; Wang, J.; Pérez-Jiménez, M.J. Spiking Neural P Systems with Delay on Synapses. *Int. J. Neural Syst.* **2021**, *31*, 2050042. [CrossRef] [PubMed]
52. Garcia, L.; Sanchez, G.; Vazquez, E.; Avalos, G.; Anides, E.; Nakano, M.; Sanchez, G.; Perez, H. Small universal spiking neural P systems with dendritic/axonal delays and dendritic trunk/feedback. *Neural Netw. Off. J. Int. Netw. Soc.* **2021**, *138*, 126–139. [CrossRef] [PubMed]
53. Zhang, X.; Wang, B.; Pan, L. Spiking Neural P Systems with a Generalized Use of Rules. *Neural Comput.* **2014**, *26*, 2925–2943. [CrossRef] [PubMed]

54. Lv, Z.; Kou, J.; Yi, W.; Peng, H.; Song, X.; Wang, J. Sequential Coupled Neural P Systems. *Int. J. Unconv. Comput.* **2020**, *15*, 157–191.
55. Zhang, X.; Luo, B.; Fang, X.; Pan, L. Sequential spiking neural P systems with exhaustive use of rules. *Biosystems* **2012**, *108*, 52–62. [CrossRef]
56. Wang, L.; Liu, X.; Zhao, Y. Universal Nonlinear Spiking Neural P Systems with Delays and Weights on Synapses. *Comput. Intell. Neurosci.* **2021**, *2021*, 3285719. [CrossRef]
57. Song, T.; Luo, L.; He, J.; Chen, Z.; Zhang, K. Solving Subset Sum Problems by Time-free Spiking Neural P Systems. *Appl. Math. Inf. Sci.* **2014**, *8*, 327–332. [CrossRef]
58. Zeng, X.; Song, T.; Zhang, X.; Pan, L. Performing Four Basic Arithmetic Operations with Spiking Neural P Systems. *IEEE Trans. Nanobiosci.* **2012**, *11*, 366–374. [CrossRef]
59. Zhang, X.; Zeng, X.; Luo, B.; Xu, J. Several Applications of Spiking Neural P Systems with Weights. *J. Comput. Theor. Nanosci.* **2012**, *9*, 769–777. [CrossRef]
60. Zhang, G.; Zhang, X.; Rong, H.; Paul, P.; Zhu, M.; Neri, F.; Ong, Y.S. A Layered Spiking Neural System for Classification Problems. *Int. J. Neural Syst.* **2022**, *2022*, 2250023. [CrossRef]
61. Huang, Z.; Wang, T.; Liu, W.; Valencia-Cabrera, L.; Perez-Jimenez, M.J.; Li, P. A Fault Analysis Method for Three-Phase Induction Motors Based on Spiking Neural P Systems. *Complexity* **2021**, *2021*, 2087027. [CrossRef]
62. Rong, H.; Yi, K.; Zhang, G.; Dong, J.; Paul, P.; Huang, Z. Automatic Implementation of Fuzzy Reasoning Spiking Neural P Systems for Diagnosing Faults in Complex Power Systems. *Complexity* **2019**, *2019*, 2635714. [CrossRef]
63. Song, T.; Pan, L.; Wu, T.; Zheng, P.; Wong, M.L.D.; Rodriguez-Paton, A. Spiking Neural P Systems with Learning Functions. *IEEE Trans. Nanobiosci.* **2019**, *18*, 176–190. [CrossRef] [PubMed]
64. Xue, J.; Wang, Z.; Kong, D.; Wang, Y.; Liu, X.; Fan, W.; Yuan, S.; Niu, S.; Li, D. Deep ensemble neural-like P systems for segmentation of central serous chorioretinopathy lesion. *Inf. Fusion* **2021**, *65*, 84–94. [CrossRef]
65. Cai, Y.; Mi, S.; Yan, J.; Peng, H.; Luo, X.; Yang, Q.; Wang, J. An unsupervised segmentation method based on dynamic threshold neural P systems for color images. *Inf. Sci.* **2022**, *587*, 473–484. [CrossRef]
66. Song, T.; Pang, S.; Hao, S.; Rodriguez-Paton, A.; Zheng, P. A Parallel Image Skeletonizing Method Using Spiking Neural P Systems with Weights. *Neural Process. Lett.* **2019**, *50*, 1485–1502. [CrossRef]
67. Song, T.; Zeng, X.; Zhang, P.; Jiang, M.; Rodríguez-Patón, A. A Parallel Workflow Pattern Modeling Using Spiking Neural P Systems With Colored Spikes. *IEEE Trans. Nanobiosci.* **2018**, *17*, 474–484. [CrossRef]
68. Liu, Q.; Long, L.; Yang, Q.; Peng, H.; Wang, J.; Luo, X. LSTM-SNP: A long short-term memory model inspired from spiking neural P systems. *Knowl.-Based Syst.* **2022**, *235*, 107656. [CrossRef]
69. Long, L.; Liu, Q.; Peng, H.; Yang, Q.; Luo, X.; Wang, J.; Song, X. A Time Series Forecasting Approach Based on Nonlinear Spiking Neural Systems. *Int. J. Neural Syst.* **2022**, *2022*, 2250020. [CrossRef]
70. Liu, Q.; Long, L.; Peng, H.; Wang, J.; Yang, Q.; Song, X.; Riscos-Nunez, A.; Perez-Jimenez, M.J. Gated Spiking Neural P Systems for Time Series Forecasting. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**. [CrossRef]
71. Wu, T.; Paun, A.; Zhang, Z.; Pan, L. Spiking Neural P Systems with Polarizations. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 3349–3360. [CrossRef] [PubMed]
72. Wu, T.; Zhang, T.; Xu, F. Simplified and yet Turing universal spiking neural P systems with polarizations optimized by anti-spikes. *Neurocomputing* **2020**, *414*, 255–266. [CrossRef]
73. Wu, T.; Pan, L.; Alhazov, A. Computation power of asynchronous spiking neural P systems with polarizations. *Theor. Comput. Sci.* **2019**, *777*, 474–489. [CrossRef]
74. Wu, T.; Pan, L. The computation power of spiking neural P systems with polarizations adopting sequential mode induced by minimum spike number. *Neurocomputing* **2020**, *401*, 392–404. [CrossRef]
75. Liu, L.; Jiang, K. Universality of spiking neural P systems with polarizations working in sequential mode induced by maximum spike number. *J. Membr. Comput.* **2021**, *4*, 56–67. [CrossRef]
76. Yang, Q.; Lv, Z.; Liu, L.; Peng, H.; Song, X.; Wang, J. Spiking neural P systems with multiple channels and polarizations. *Biosystems* **2019**, *185*, 104020. [CrossRef]
77. Jiang, S.; Fan, J.; Liu, Y.; Wang, Y.; Xu, F. Spiking Neural P Systems with Polarizations and Rules on Synapses. *Complexity* **2020**, *2020*, 8742308. [CrossRef]
78. Gutkin, B.; Ermentrout, G.B. Neuroscience - Spikes too kinky in the cortex? *Nature* **2006**, *440*, 999–1000. [CrossRef]
79. Peng, H.; Li, B.; Wang, J.; Song, X.; Wang, T.; Valencia-Cabrera, L.; Pérez-Hurtado, I.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. Spiking neural P systems with inhibitory rules. *Knowl.-Based Syst.* **2020**, *188*, 105064. [CrossRef]
80. Ionescu, M.; Păun, G.; Yokomori, T. Spiking Neural P Systems. *Fundam. Inform.* **2006**, *71*, 279–308.
81. Korec, I. Small universal register machines. *Theor. Comput. Sci.* **1996**, *168*, 267–301. [CrossRef]