International Journal of Computer Applications (0975 - 8887)
Volume 187 - No.58, November 2025

Spiking Neural P systems with Different Weight Structural Plasticity

Rosini B.
Department of Mathematics
Fatima Mata National College
Kollam, Kerala, India

K.S. Dersanambika
Department of Mathematics
Fatima Mata National College
Kollam, Kerala, India

Anitha P.
Department of Mathematics
Government College, Ambalappuzha
Alappuzha, Kerala, India

ABSTRACT

Spiking neural P systems (in short SNP systems) [4] are the computing devices inspired by the spiking neurons. Spiking neural P systems with Structural Plasticity (SNPSP systems) is a variant of SNP systems in which the biological feature of structural plasticity is incorporated. In this paper, we extend and generalize the spike - travel between neurons. In 2012, L. Pan et. al. [6] proposed weighted synapses inspired from the biological concept that each pair of neurons is connected by several synapses. Different multiple synaptic connections of each pair of neurons extends this study for a variant SNP systems with different weight structural plasticity and its universality.

Keywords

Spiking Neural P systems, Structural plasticity, different weight

1. INTRODUCTION

In membrane computing, Spiking Neural P systems (in short SNP systems) are a emerging research area, with the insight of spiking neurons which communicate each other by means of electrical impulses. This parallel, computing devices in membrane computing, are introduced in [4]. SNP systems are depicted as a directed graph whose nodes are neurons and directed arcs are synapses. The electrical impulse in neuron is known as spike and is denoted by $'a'$. The spikes are triggered with the spiking rules which consume some spikes and produce a spike to all neurons connected by the synapses. The spikes of a neuron are removed by the forgetting rules. One of the neurons called output neuron emits a spike to environment. The output of a SNP system can be defined in several ways: Output is a set of numbers, denotes the time–step difference of two consecutive emission of spikes from output neuron to the environment. In other way, output is binary sequences (binary spike trains) which mark the moments by 1 when output neuron spills a spike to the environment and 0 when no spike is emitted from output neuron. SNP systems can be used in accepting mode also. SNP systems can compute morphisms over $\{0, 1\}$. Such systems are known as SNP transducers. Thus SNP systems can be used as generators, accepters and transducers of sequences of spikes [7], [2].

Inspired from the neuroscience, Structural plasticity, one of the features of the brain, is the motivation for the new variant Spiking neural P systems with Structural Plasticity (SNPSP systems)[1]. Besides the spiking rules, the plasticity rules are used in SNPSP systems, which creates or deletes the synaptic connectivity, so the synapse diagram is vigorous. The study of these systems is developed through [9], [10]. From biological motivation, each pair of neurons is connected by several synapses. In [6], an integer weight is empowered with a synapse to represent the number of synapses for a connected couple of neurons. Synapses are junctions of neurons which allows the transmission of electrical and chemical signals to the adjacent cells. Synapses can be either excitatory or inhibitory. The firing action potential of a neuron is decreased by the inhibitory synapse; while excitatory synapse increase the possibility of the firing action potential of a neuron. The functioning of excitatory and inhibitory synapses are due to the discharge of neurotransmitters Acetylcohline (Ach) and GABA (Gamma Amino Butyric Acid) respectively [3],[8]. In this work, excitatory synapse is endowed with a positive weight; while an inhibitory synapse is effected with a negative weight. Especially the number of spikes passes through the inhibitory synapses becomes negative and they annihilate the same number of spikes of the post synaptic neuron. The plasticity rules are restricted for the creation of either excitatory synapses or inhibitory synapses, but not both. This paper motivates the study of SNPSP systems with different weights; which allows the different number of connections from a neuron so that the neuron can transfer different number of spikes to the neighbouring neurons by creating (for the cases– creation, creation with deletion and deletion with creation) synapses with different weights. A new variant of Spiking neural P systems with structural plasticity is defined in section 2. Section 3, throws light on the universality of such systems both in generating and accepting mode. In [9], it is shown that the universality with the restriction that a neuron is allowed to create or delete instead of both, connections to some of its neighbour neurons in a computation step. It deserves for further research whether there are improved normal forms of universal SNP systems with structural plasticity.

2. SPIKING NEURAL P SYSTEMS WITH DIFFERENT WEIGHT STRUCTURAL PLASTICITY

Now the definition of a new variant namely Spiking Neural P systems with Different weight structural plasticity is given as follows:

DEFINITION 1. *Spiking Neural P systems with Different Weight Structural Plasticity (SNPDWSP systems) of degree ($m \geq 1$) is a construct of the form*

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

where

- $O = \{a\}$*, is an alphabet consists of only a symbol a, which is known as* spike*;*
- $\sigma_1, \sigma_2, \ldots, \sigma_m$ *denotes m neurons. Each neuron σ_i is represented as (n_i, R_i) where $n_i \geq 0$ indicates the number of spikes and R_i is a set of rules consists of two type of rules namely spiking rules and plasticity rules. For a regular expression E over O;*
- ⋆ *Spiking rule $SR_i : E/a^c \rightarrow a^p; d$ with $c \geq p \geq 1$;*
- ⋆ *Plasticity rule*
  $PR_i : E/a^c \rightarrow \alpha k^\beta \{(i, j, w_{ij}) : i, j \in \{1, 2, \ldots, m\}, j \neq i, w_{ij} \in W\}$ *where $c \geq 1$, $\alpha \in \{+, -, \pm, \mp\}, \beta \in \{+, -\}, k \geq 1$ and W is a finite subset of N.*
- syn_t *denotes the set of synapses between the neurons at time t which is a subset of $\{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ with $(i, i) \notin syn_t$ for $1 \leq i \leq m$.*
- $in, out \in \{1, 2, \ldots, m\}$ *denote input and output neurons respectively.*

For a given neuron σ_i, there are two sets which are defined as follows :- $pres_t(i)$ and $pos_t(i)$. $pres_t(i) = \{j : (i, j) \in syn_t\}$, is the set of all neuron labels which has σ_i as presynaptic neuron at time t. $pos_t(i) = \{j : (j, i) \in syn_t\}$, is the set of all neuron labels which has σ_i as postsynaptic neuron at time t. The sets $pres_t(i)$ and $pos_t(i)$ can be determined from synapse graph syn_t at time t. The set of all neighbouring neurons of neuron i is denoted by N_i.
Spiking rules and plasticity rules work as in SNPSP systems [1]. For a neuron σ_i, the implementation of a rule with $\alpha \in \{+, \pm, \mp\}$ means the creation of a synapse (i, j) always involves an inherent transmission of w_{ij} spikes to each neuron connected to. When $\beta = +$, the plasticity rule evolves with excitatory synapses; while $\beta = -$, the plasticity rule evolves with inhibitory synapses. In this work, we incorporates a feature in plasticity rules- creation of synapses with different weights. That is a neuron can create synapses with different weights and thus it can transmit different number of spikes to each connected neuron. When $\beta = +$, it is omitted from that representation for convenience. The creation of an inhibitory synapse (1,2) from neuron 1 to neuron 2 is denoted by $+1^-(1, 2, 2)$ and two spikes reached at neuron 2 causes the annihilation of two spikes in non- empty neuron 2 having two or more spikes.
The configuration, transition, computation and halting computation of SNPDWSP system is defined as that of SNPSP systems. Here we can consider only the first two time instances t and $t + k$ at which first and second spiking of σ_{out} occurs and spikes are emitted to the environment. Their difference $(t + k) - t = k$ is said to be computed or generated by the SNPDWSP system. The set of all numbers computed in this manner by Π as $N_2(\Pi)$. The number 2 denotes the result is the time step difference of the first two spikes emitted by the output neuron. In generating mode, in is ignored. Output of the system is incorporated by the spike train with the binary sequence 1 if output neuron expels a spike to the environment and 0 if no spike is emitted by the output neuron. The SNPDWSP system can also work in accepting mode, where out is ignored. In accepting mode, exactly two spikes are introduced to the system using in at time steps t_1 and t_2. The number $t_2 - t_1$ is accepted or computed by the system if the computation halts. The set of all numbers accepted in this way is denoted by $N_{acc}(\Pi)$. Thus the family of all sets $N_\alpha(\Pi)$ with $\alpha \in \{2, acc\}$ is denoted by $N_\alpha(\Pi)$.

2.1 Example

An example of SNPDWSP is described in figure 1. At time step 1, the rules of neurons σ_1 and σ_3 fire. The plasticity rule of neuron σ_1 nondeterministically creates a synapse either to the neuron σ_2 with weight 1 or to the neuron σ_3 with weight 2. If the synapse $(1, 3)$ is created with weight 2 and the output neuron σ_3 obtains two spikes and emits a spike to the environment at time step 2. Thus the number $t_2 - t_1 = 1$ is generated. If the synapse $(1, 2)$ with weight 1 is created and one spike is sent to the neuron σ_2 then it applies its rule at step 2. One spike is directed to the neuron σ_1 and the plasticity rule is enabled again. If the synapse $(1, 2)$ is created once, then it will take two time steps for the opportunity of creating the synapse $(1, 3)$ again. Once the synapse $(1, 3)$ is created, it requires one more step for the second spiking of output neuron. If synapse creation $(1, 2)$ repeats $m \geq 1$ times, then $N_2(\Pi_e) = \{2m+1/m \in N\}$.

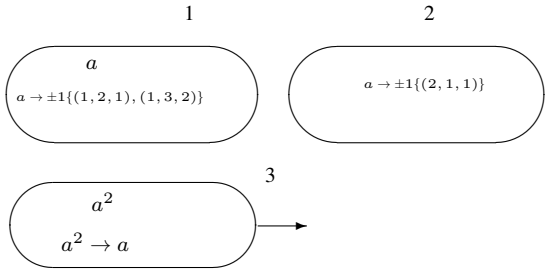

Fig. 1. A SNPDWSP System Π_e

3. UNIVERSALITY OF SNPDWSP SYSTEMS

The universality results of the paper are based on the notion of register machine. The definition of Register machine is given below [5].
Register machine is a construct $M = (m, H, l_0, l_h, I)$ where m is the number of registers, H is the set of instruction labels, l_0 is the start label which labels an ADD instruction, l_h is the label which is assigned to HALT instruction and I is the set of instructions; each label l_i from H is assigned to exactly one instruction $l_i : (\mathtt{OP}(r), l_j, l_k)$ from I. The instructions are of the following forms:

$l_i : (\mathtt{ADD}(r), l_j, l_k)$: Add 1 to the number stored in the register r and go nondeterministically to the instruction l_j or l_k.

$l_i : (\mathtt{SUB}(r), l_j, l_k)$: Subtract 1 from the number stored in the register r and go to the instruction with label l_j if r has nonzero number; otherwise go to the instruction l_k.

l_h: the HALT instruction.

Beginning with all registers empty (that is stored number is 0), and the instruction labelled by l_0 and then continue with the application of instructions by the indicated labels, as the contents of registers. It is assumed that the content of the register 1 is never decremented by the SUB instruction and all other registers become empty in the halting configuration. The register machine M generates a number n, which is the existent number in register 1 at the halting instruction. Such a way, a set of numbers generated by the

register machine M is obtained which is denoted by $N(M)$, which is Turing computable and characterize NRE [5].
For, a number n is introduced into the register r_0, and begins with the instruction l_0. The number n is said to be accepted by the machine M if M reaches the halting instruction. If the numbers $n_1, n_2, \ldots, n_k$ are introduced to the registers $r_1, r_2, \ldots, r_k$ and begins with the instruction label l_0, then gradually M halts and the value of the functions are placed in a designated register r_u, making with all other registers empty. The register machine functions deterministically for both the accepting and computing case. That is deterministic ADD instruction with $l_i : (\texttt{ADD}(r), l_j)$; adding 1 to the number stored in register r and go to the instruction label l_j.

3.1 SNPDWSP systems as generating mode

THEOREM 2. $NRE = N_2SNPDWSP$

PROOF. To prove this, it is enough to show that $NRE \subseteq N_2SNPDWSP$. The reverse inclusion is clear since [5] register machines generates all sets of numbers which are Turing computable. Without loss of generality, assume that for a register machine $M = (m, H, l_0, l_h, I)$ (a) all registers except 1 become empty at halting instruction (b) the register 1 never be decremented by SUB instruction (c) the label l_0 labels an ADD instruction.
For a given register machine $M = (m, H, l_0, l_h, I)$ we can construct a SNPDWSP system Π to simulate M. The three modules - ADD, SUB and FIN shown in figures 2, 3, 4 are contained in the system Π. The first two modules simulate ADD and SUB instructions of M, respectively; FIN module gives the computation result. For the register r in M, the neuron σ_r is considered in Π. If the register has the number $n \geq 0$, then $2n$ spikes are associated with neuron σ_r. For each label l_i of an instruction in M, there is a neuron σ_{l_i} in Π and σ_{l_i} simulates the instruction l_i with a spike. When the neuron σ_{l_h} becomes active (that is the halting instruction l_h of M), the complete computation in M is simulated by Π and gives the output. The neuron σ_{l_h} fires at time steps t_1 and t_2 and the number $t_2 - t_1$ corresponds to the number stored in register 1 of M.

ADD Module

At time step t, an instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$ has to be played by the activation of neuron l_i with a spike. All other neurons are empty other than the neurons related with registers. With a spike, neuron l_i fires at time step t and a spike is transferred to the neuron $\sigma_{l_i^1}$. The neuron $\sigma_{l_i^1}$ fires at step $t+1$ and sends two spikes to neuron σ_r and one spike to $\sigma_{l_i^2}$. At time step $t+2$, the neuron $\sigma_{l_i^2}$ non-deterministically chooses the creation of a synapse (l_i^2, l_j) or (l_i^2, l_k) and thus a spike is transferred to σ_{l_j} or σ_{l_k} and finally that synapse is removed. Therefore the insertion of two spikes to the neuron σ_r corresponds to the number in the register r incremented by 1. One of the neurons σ_{l_j}, σ_{l_k} is activated in next step, hence ADD module simulates the instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$.

SUB Module

Module SUB shown in figure 3, simulates the instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$. Assume that the neuron σ_r represents the register r and σ_r has $2n, (n \geq 0)$ spikes as the number stored n in r and there are neurons corresponds to each instruction label. Initially, the neuron l_i is activated with a spike at time step t. At time step t, neuron σ_{l_i} fires using its rule and sends one spike to neuron σ_r. The neuron σ_r fires with odd number of spikes at the next step. At time step $t+1$, if σ_r contains odd number of spikes more than one, it consumes 3 spikes and creates two synapses to neuron $\sigma_{l_i^2}$ with weight two and to neuron $\sigma_{l_i^3}$ with weight one; hence 2 spikes and 1 spike are reached the neurons respectively. At time step $t+2$, $\sigma_{l_i^2}$ activates and two spikes are transmitted

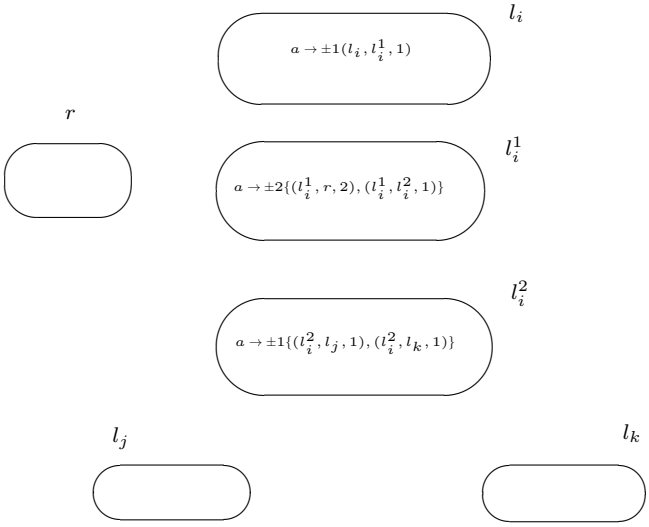

Fig. 2. ADD Module

to σ_{l_j}. Similarly the simultaneous creation of inhibitory synapse $(l_i^3 l_j)$ causes the activation of only one neuron σ_{l_j} with a spike. At time instance $t+1$, if σ_r contains only one spike, then it creates two synapses to the neurons, $\sigma_{l_i^2}$ with weight one and $\sigma_{l_i^3}$ with weight two and hence the neurons $\sigma_{l_i^2}$, $\sigma_{l_i^3}$ receive one spike and two spikes respectively. The creation of both inhibitory and excitatory synapses from $\sigma_{l_i^2}$ and $\sigma_{l_i^3}$ respectively, leads to the activation of only one neuron σ_{l_k} with a spike. At the former case, removing three spikes from σ_r, remains an even number of spikes $2n-2$, corresponds to the number $n-1$ in register r. SUB module simulates the diminishment of the value of r by one and execution of the instruction l_j when r has non-zero value. Similarly it simulates the activation of the instruction l_k with a spike when the stored value in register r is 0.

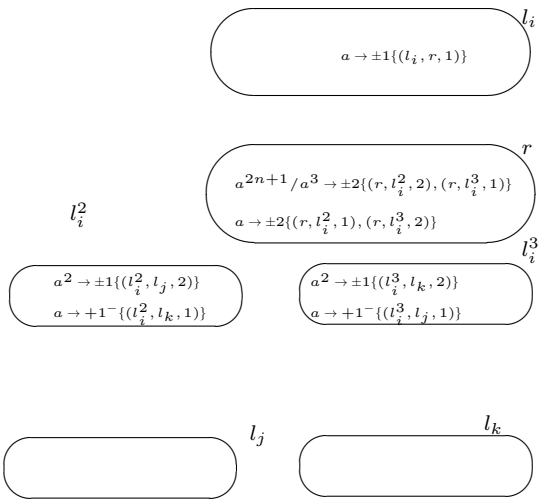

Fig. 3. Module SUB (simulating $l_i : (\texttt{SUB}(r), l_j, l_k)$)

FIN Module

Once the halting instruction l_h is reached in M, module FIN shown in figure 4 is relevant. As in figure 4, neuron σ_1 has $2n$ spikes, corresponds to the number n stored in the register 1 of M when the computation halts. At time t, with one spike, neuron l_h sends one spike to neuron σ_1 and two spikes to neuron σ_{out} by creating synapses (σ_{l_h}, σ_1) with weight 1 and $(\sigma_{l_h}, \sigma_{out})$ with weight 2. At time step $t+1$, σ_{out} fires first time and delivers a spike to the environment. At the same time instance, σ_1 with odd number of spikes more than 3, continuously applies its rule $a^{2n+1}/a^2 \rightarrow -1(1, out, 1)$. Since there is no synapse $(1, out)$, σ_1 consumes two spikes and no synapse is removed. Once the number of spikes reduced into three, σ_1 applies the rule $a^3 \rightarrow \pm 1^-(1, out, 2)$. The creation of this inhibitory synapse and passing of two spikes causes the reduction of number of spikes in σ_{out} by 2. This rule can be executed after the n applications of the previous rule. Thus the neuron σ_{out} fires at time steps $t+1$ and $t+n+1$ and transmits spikes to the environment. The time difference of first and second spiking of σ_{out} is n, the number stored in the register 1 when the computation of M halts.

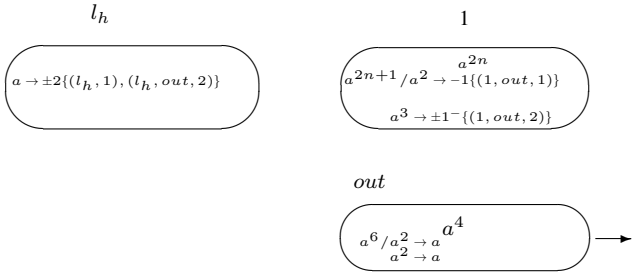

Fig. 4. FIN Module

From the above explanation of the operations of modules ADD, SUB and FIN, it is clear that it simulates the computation of M. Consequently $N_2SNPDWSP = N(M)$. This completes the proof. □

3.2 SNPDWSP systems in accepting mode

Observe that a register machine M is computationally universal even for a deterministic accepting case. The time difference between the first two spikes introduced into the system, is the number which is stored into the register 1, is accepted by the system if it halts.

THEOREM 3. $NRE = N_{acc}SNPDWSP$

PROOF. In accepting mode, for a given deterministic register machine M, a SNPDWSP system Π is constructed as in the proof of theorem 2, with the deterministic ADD instruction and the INPUT module instead of FIN module. In accepting mode, output neuron is ignored and use input neuron to input exactly two input spikes. The INPUT module is shown in the figure 5 and works as follows: Initially all neurons are empty. The first input spike is lead into the input neuron in at the time step t. Using its rule, it creates two synapses with different weights 1 and 2 and passes a spike to neuron I_1 and two spikes to the neuron I_2. At the time $t+1$ I_1 creates two synapses to neuron 1 and I_2, with weight 2 and 1 respectively. From the step $t+1$, the neurons I_1 and I_2 start to exchange their spikes and neuron I_1 puts 2 spikes each to neuron 1. If the last and second spike enters the system at $t+n$, then the spikes are flown to the neurons I_1 and I_2 as detailed above and consequently the transaction of the neurons I_1 and I_2 and the deposition of spikes into the neuron 1 are stopped at the step $t+n+1$. At the same time, I_1 and I_2 activates the neuron l_0 with a spike for the next step which simulates the execution of the instruction l_0. Thus $2n$ spikes are stored in the neuron σ_1 corresponding to the number n in the register 1.

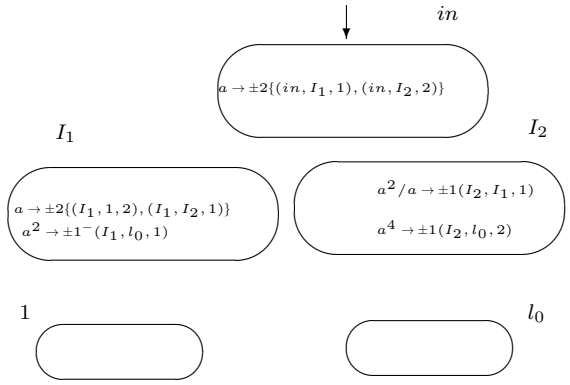

Fig. 5. INPUT Module

A deterministic ADD module is constructed for simulating $l_i : (\texttt{ADD}(r), l_j)$ as shown in figure 6, which is simpler than non-deterministic ADD module. SUB module resides same as in the

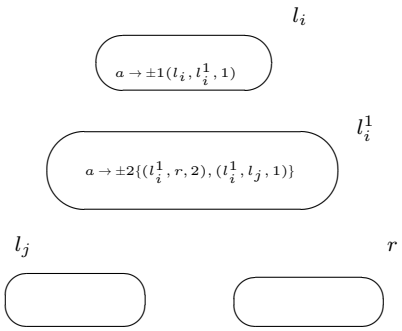

Fig. 6. Deterministic ADD Module

proof of theorem 2. Although the module FIN is irrelevant in this case, the neuron σ_{l_h} remains in the system with $pres_t(l_h) = \emptyset$ for all t. Once σ_{l_h} receives a spike indicates that the computation of M has halted. The neuron applies its rule but does not send its spike to any other neurons. The computation of Π halts according to the computation of M halts. If the computation halts, the number $n = (t+n) - t$ is accepted. Thus we have $N_{acc}(\Pi) = N(M)$. □

4. CONCLUSION

SNPDWSP system adopts transducer mode by considering both input and output neurons. Spikes are introduced into the input neuron and while output neuron emits its spikes to the environment. With a computation, halting or not, we associate a binary digit 1 or 0 with the moment at which output neuron emits spikes or (no spikes) to the environment. Similarly the binary sequence associated with spikes is considered to the input of the system. A large class of Boolean functions can be computed in this mode.

This paper enlightens the universality of SNPDWSP systems both in generating and accepting mode, with the following features: (a) only plasticity rules are used in all neurons except output neuron (b) forgetting rules and rules with delay are omitted (c) creation of synapses with different weights which helps to transfer more than one spike while connected to (d) initial synapse set is empty (e) each neuron (except output neuron) contains at most two plasticity rules. The capability of systems as transducers is also discussed, in which further study is improved with simpler structures for exact computations of real numbers with the potential of different weight structural plasticity and excitatory– inhibitory nature of synapses.

5. REFERENCES

[1] Cabarle F. G. C., Henry N. Adorna, Mario J. Pérez-Jiménez, and Tao Song. Spiking neural p systems with structural plasticity. *Neural Computing and Applications*, 26(8):1905–1917, 2015.

[2] S. Fan, P. Paul, T. Wu, H. Rong, and G.Zhang. On applications of spiking neural p systems. *Application of Science*, 10(20):1–26, October 2020.

[3] https: //my.clevelandclinic.orghealth.

[4] M. Ionescu, G. Păun, and T.Yokomori. Spiking neural p systems. *Fundamenta Informaticae*, 71(2–3):279–308, 2006.

[5] M.L. Minsky. Computation: Finite and infinite machines. *Englewood Cliffs, NJ: Prentince Hall*, 1967.

[6] L. Pan, X. Zeng, X. Zhang, and Yun Jiang. Spiking neural p systems with weighted synapses. *Neural Process Letteers*, 35:13–27, 2012.

[7] G. Paun, G. Rozenberg, and A. Salomma. *The Handbook of Membrane Computing*. Oxford University Press, 2010.

[8] M.R. Picciotto, M. Higley, and Y.S. Mineur. Acetylcholine as a neuromodulator: cholinergic signaling shapes nervous system function and behaviour. *Neuron*, 76(1):116–129, 2012.

[9] T. Song and L. Pan. A normal form of spiking neural p systems with structural plasticity. *Swarm Intelligence*, 1(4):344–357, 2015.

[10] M. Sun and Jinhua Qu. Weighted spiking neural p systems with structural plasticity working in maximum spiking strategy. *8th International conference on Information Technology in Medicine and Education*, pages 355–359, 2016.