Spiking Neural P systems and Petri nets

Venkata Padmavati Metta
Bhilai Institute of Technology, Durg
vmetta@gmail.com

Kamala Krithivasan
Indian Institute of Technology, Madras
kamala@iitm.ac.in

Deepak Garg
Thapar University, Patiala
deep108@yahoo.com

Abstract

Spiking Neural P (SN P) system characterizes the movement of spikes among the neurons. Spikes have a similarity with tokens in Petri net where tokens (like spikes) are moved through net according to specific rules. This paper proposes the concept of spiking Petri nets, which are isomorphic to spiking neural P systems. It also gives algorithms to construct spiking Petri net for SN P system and vice versa. The Spiking Petri net combines the modeling capabilities of coloured and timed Petri net concepts with the new orientation of SN P system. Examples are given illustrating the way in which the spiking Petri net can be employed to simulate the behaviour of SN P system.
Key words: SN P System, Spiking Petri net, Membrane Computing, Modeling. +

1 Introduction

Biologically inspired computing which is a branch of natural computing is the field of investigation that draws upon metaphors or theoretical models of biological systems in order to design computational tools or systems for solving complex problems. It is motivated by the need to identify alternative media for computing other than silicon. Researchers are now trying to design new computers based on molecules, such as membranes (P Systems), DNA, RNA or Quantum Theory. If biology is such a rich source of inspiration for informatics, then the brain is the gold mine of this intellectual enterprise. Brain is composed of 10^{11} to 10^{12} individual neurons. A neuron consists of synapses, a soma, dendrites, an axon, an axon hillock and axon terminals.

Biological neurons use short and sudden increases in voltage to send information. A neuron receives and sends small electrical pulses usually called spikes from other neurons through its dendrites. A spike is also called an action potential or a nerve impulse. The spikes are stored and processed in the cell body called soma. dendrites are thin numerous bushy extensions of the cell. They receive spikes from the synapses and carry them to soma. When the weight(number of spikes) gets bigger than a particular value, called the threshold, associated with axon hillock, a neuron fires; that is, it emits an output signal called a spike through the axon terminals. Axon is a thin and long channel that carries spikes from the axon hillock to the axon terminals. These axon terminals are the input to other neurons.

Spikes cannot just cross the gap between one neuron and the other. They have to be handled by the most complicated part of the neuron: the synapse, connection between the dendrites of a neuron and the axon terminals from the other neurons. Neurons send out erratic sequences of spikes, or spike-trains, which alter dramatically in frequency over a short period of time. Neurons have to use spatial and temporal information of incoming spike patterns to encode their message to other neurons. There are many different schemes for the use of spike timing information in neural computation.

A neuron is a very powerful cell. They are interconnected with each other and cooperate to efficiently process incoming signals and decide on actions. A typical neuron is able to receive thousands of spikes and transmit thousands of spikes concurrently. The

architecture, size, and capabilities of these cells have created a tremendous research area recently. The functioning of the neurons is well organised, hierarchical, and parallel.

Neuroscience, along with information and mathematical sciences, has developed a variety of theoretical and computational models to model complex brain functions. Along with this development, computational models that adopt principles from the nervous system have been developed to become powerful tools for learning from data and generalization. With the recent advancements of genetic research, more information is becoming available about the interaction between neurons. It is well accepted that brain functions can be better understood if there is a computational model that represents this neural information. Spiking Neural P system (SN P systems, for short) is a complex mathematical model representing the concurrent activities of spiking neurons. It was introduced by Gh. Paun et al.[5] to incorporate membrane computing ideas specific to spiking neurons in a mathematical way. SN P system captures the fact that most of the neural impulses are almost identical electrical signals of a given voltage. A sequence of such impulses, which occur at regular or irregular intervals, is called a spike train that is a useful means to encode information.

In order to give an accurate description of the spiking neural net architecture[9] a model has to be formulated which provides the structure and behaviour of the spiking neural net. The model must serve as the basis for translating brain features into a computer system. It is natural to model this highly parallel system of neurons with a Petri net, as they provide formalism to represent concurrent and parallel events[8]. The structural and behavioural properties of neuron were studied using neural Petri nets[6]. The neural net architecture was described with Timed Neural Petri nets[7] that considered various chemical changes in the neurons. Similarly spiking neural concept was described in a mathematical way using SN P systems. They have a direct (pictorial) similarity with Petri nets, where tokens (like spikes) are moved through the net according to specific rules[2] and Petri nets provide simple graphical representation and can be simulated to analyze the working of the system using appropriate Petri net tool. To complement the functional characterisation of the behaviour of SN P systems, we introduced a translation of SN P systems into Petri nets. In this way, using the notations and tools developed for Petri nets, one can describe what is actually going on during a computation of a SN P system. It is worth noting that as far as the rules of each neuron is concerned, SN P systems are highly concurrent systems and one of the core features of Petri nets is that they support and analyse concurrency in its most fundamental fashion. This paper describes a methodology to develop a Petri net model of a system, by deriving a form of Petri nets called as Spiking Petri nets from SN P systems that use only one object(spike).

2 Spiking Neural P System

SN P system contains one-membrane cells (called neurons) which can hold any number of spikes; each neuron fires in specified conditions (after collecting a specified number of spikes, which are accumulated, added one after another) and then sends one spike along its axon; this spike passes to all neurons connected by a synapse to the spiking neuron (hence it is replicated into as many copies as many target neurons exist); the rules for spiking should take into account all spikes present in a neuron not only part of them; not all spikes present in a neuron are consumed in this way; after getting fired and before sending the spike to its synapses, the neuron is idle (biology calls this the refractory period) and cannot receive spikes. Between the moment when a neuron fires and the moment when it spikes, each neuron needs a time interval called timed delay. One of the neurons is considered as the output one, and its spikes provide the output of the computation. There are also rules used for "forgetting" some spikes, rules that just remove a specified number of spikes from a neuron.

Definition 2.1 Mathematically, we represent a spiking neural P system (SN P system), of degree $m \geq 1$, in the form

Π=(O, σ_1, σ_2, σ_3 ,. . . , σ_m , *syn* , i_0), where

1. O = $\{a\}$ is the singleton alphabet (a is called *spike*) ;
2. σ_1, σ_2, σ_3 ,. . . , σ_m are neurons, of the form

$$\sigma_i=(n_i\ ,R_i)\ ,\ 1\leq i\leq m,$$

where

a) $n_i\geq 0$ is the *initial number of spikes* contained by the cell;

b) R_i is a finite set of *rules* of the following two forms:

(1) $E\ /\ a^r\longrightarrow a;t$, where E is a regular expression over O, $r\geq 1$, and $t\geq 0$; Number of spikes present in the neuron is described by the regular expression E, r spikes are consumed and it produces a spike, which will be sent to other neurons after t time units

(2) $a^s\longrightarrow\lambda$, for some $s\geq 1$, with the restriction that $a^s\notin \mathrm{L}(E)$ for any rule $E/a^r\longrightarrow a;t$ of type (1) from R_i;

3. *syn* $\subseteq$ { 1, 2, 3, . . . , m} $\times$ { 1, 2, 3, . . . , m} with (i, i) $\notin$ *syn* for $1\leq i\leq m$ (*synapses* among cells);
4. $i_0\in$ { 1, 2, 3, . . . , m} indicates the *output neuron.*

The rules of type $E\ /\ a^r\longrightarrow a;t$ are spiking rules, and they are possible only if the neuron contains n spikes such that $a^n\in \mathrm{L}(E)$ and $n\geq r$. If $E=\phi$ then rule is applied only if the neuron contains exactly r spikes. When neuron σ_i spikes, its spike is replicated in such a way that one spike is sent to all neurons σ_j such that (i, j) $\in$ *syn*, and σ_j is open at that moment. If a neuron σ_i fires and either it has no outgoing synapse, or all neurons σ_j such that (i, j) $\in$ *syn* are closed, then the spike of neuron σ_i is lost; the firing is allowed, it takes place, but it produces no spike.

The rules of type $a^s\longrightarrow\lambda$ are forgetting rules; s spikes are simply removed ("forgotten") when applying. Like in the case of spiking rules, the left hand side of a forgetting rule must "cover" the contents of the neuron, that is, $a^s\longrightarrow\lambda$ is applied only if the neuron contains exactly s spikes.

A global clock is assumed in SN P system and in each time unit each neuron which can use a rule should do it (the system is synchronized), but the work of the system is sequential locally: only (at most) one rule is used in each neuron. One of the neurons is considered to be the output neuron, and its spikes are also sent to the environment. The moments of time when a spike is emitted by the output neuron are marked with 1; the other moments are marked with 0. This binary sequence is called the spike train of the system it might be infinite if the computation does not stop. Many kinds of output can be associated with a computation in an SN P system. This paper considers the distance between first two spikes in the spike train as the value computed by an SN P system.

Example 2.1 Consider spiking neural P system given in Figure.1 and formally it is denoted as Π=(O, σ_1, σ_2, σ_3 , *syn* , 3), with
σ_1 = (7, { $a(aa)^+/a^2\longrightarrow a;2$}),
σ_2 = (0, { $a^2\longrightarrow a;1$}),
σ_3 = (1, { $a\longrightarrow a;0$})
syn={(1, 2), (2, 3)}.

We have three neurons, with labels 1, 2, 3; neuron 3 is the output neuron. Initially neuron 1 has 7(odd number greater than or equal to 3) spikes covered by the regular expression $a(aa)^+$ and neuron 3 has one spike and they fire in the first step. The spike of neuron 3 exits the system, so the number of steps from now until the next spiking of neuron 3 is the number computed by the system. After firing, neurons 3 remains empty and cannot fire until it gets a spike. In turn neuron 2 is empty and cannot fire until collecting exactly two spikes.

After firing, neuron 1 will be blocked for the next two steps. In the third step it release its two spikes, sending one spike to neuron 2, and in step 4 will fire again. Thus neuron 1 fires in every third step, consuming two spikes having odd number of spikes

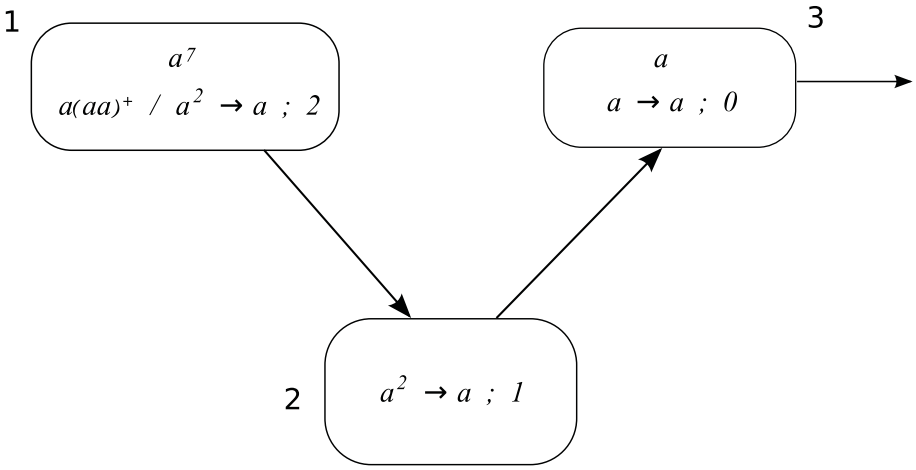

Figure 1: An example of an SN P system.

again. In the sixth step neuron 2 will receive second spike and in the next step, it will fire. The delay between firing and spiking is of one time unit for neuron 2, hence its spike will reach neuron 3 in eighth step and in the ninth step neuron 3 spikes again. Therefore the distance between these two spikes is eight, the value computed by the SN P system.

3 Spiking Petri net

A Petri net consists of ovals representing places, which hold tokens; rectangles representing transitions and arcs connect transitions to places, which change the distribution of the tokens.The current state of the modeled system (the marking) is given by the number of tokens in each place. Transitions are active components. They model the concurrent activities, which can occur thus changing the state of the system. Transitions are only allowed to fire if all the preconditions for the activity must be fulfilled (they are enough tokens available in the input place).

A marking μ of a Petri net is an assignment of tokens to the places in that net. Tokens reside in the places of the net. The number and position of tokens in a net may change during its execution. The vector $\mu = (\mu_1, \mu_2, \mu_3, \ldots, \mu_n)$ gives, for each place in the Petri net, the number of tokens in that place. The number of tokens in place P_i is μ_i, for $1 \leq i \leq$ n. We may also define a marking function μ: P$\longrightarrow N^{|P|}$ from the set of places to a vector on natural numbers, N = { 0, 1, 2,... }. This allows us to use the notation $\mu(P_i)$ to specify the number of tokens in place P_i. The firing of a transition results in a next net marking or state. A reachability graph is a graph where each node represents a Petri net marking, with arcs connecting each marking with all of its next markings. The reachability graph defines a nets state space(i.e the set of reachable states). Formally a PN with a given marking is said to be in deadlock if and only if no transition is enabled in the marking. A PN where no deadlock can occur starting from a given marking is said to be live.

In basic Petri nets, tokens have no identity and are timeless. Therefore, it is arbitrary as to which particular tokens are to be removed without any delay from a transitions input place in the case that the input place contains more tokens than are needed to enable the transition. But Petri nets described real systems tend to be complex and extremely large. Ordinary Petri nets are not always sufficient to represent and analyze complex industrial and other

systems. To solve these problems many authors propose extension of the basic Petri nets. One-way to change this semantic is to introduce the concept of tokens with identity. The type of Petri net is called a colored Petri net[4].

For colored Petri nets (CPNs), tokens have an associated data value. Places are then annotated to indicate what type of values they hold, arcs indicate which values are consumed or generated and transitions can have a guard to further constrain their enabling. In timed extension transition is associated with a time for which no event or firing of a token can occur until this delay time is elapsed. This delay can be deterministic or probabilistic. Petri nets with time extensions, combined with heuristic search techniques, were used to model and study scheduling problems involving manufacturing systems as well as robotic systems. In particular, timed Petri nets can be used to analyze the dynamic behaviour of systems with asynchronous and concurrent processing properties. The timed Petri net has been shown to be a useful tool for the performance evaluation of systems where time is a factor in the decision-making process. The bibliography of Petri nets[3] contains entries dealing with Petri net theory and applications. High level Petri nets(coloured timed extensions etc.) were also used in the modeling, analysis, simulation of the behavior of biological systems.

Biological spiking neurons have been well studied over the course of the previous century and much has been discovered about the details of action and membrane potential generation. Both spiking itself and the refractory period are thus needed for communications between neurons. However, the refractory period is also an important variable that creates different spiking behaviours in different neurons and also serves an important function in the signal processing performed in brain.

Spiking neural netwoks were mathematically represented using SN P system. In this paper we introduce spiking Petri net, which is an extension of the coloured timed Petri net to accommodate the features of spiking neurons of SN P system. Unlike other Petri net model, in spiking Petri net, during the time interval between enabling and firing of delayed transition, the input place are inactive and do not receive tokens from other transitions. Thus it can very well represent the refractory period in spiking neuron. In spiking Petri net the tokens are of single colour(character type 'a') representing spike in an SN P system.

Definition 3.1 A Spiking Petri net is represented by C=(P, T, A, W, Γ, G, P_0, I_0), where
P=$\{P_0, P_1, P_2, P_3, \ldots, P_m\}$ is a finite set of places.
T=$\{T_1, T_2, T_3, \ldots, T_n\}$ is a finite set of transitions. The immediate transitions are drawn as black bars and delayed transitions as empty boxes.
A$\subseteq$(P$\times$T)$\cup$(T$\times$P) is a finite set of arcs between P elements and T-elements denoting input flows P$\times$T to and output flows T$\times$P from transition such that P$\cap$T= P$\cap$A=A$\cap$T=ϕ.
W: A$\longrightarrow$N assigns weight $W(f)$ to elements of f $\in$A denoting the multiplicity of unary arcs between the connecting nodes.
Γ: T$\longrightarrow$R assigns firing delay, the time gap between enabling and firing of a transition, Γ(T_i) to elements of T_i $\in$T.
G: Guard function that maps each transition T_i, to an expression of type boolean, which must be true before the transition is enabled.
$P_0 \in$ P indicates the output place with no outgoing arcs and is empty in the beginning.
I_0: P$\longrightarrow$ $N^{|P|}$ is a one dimensional vector $(0, n_1, n_2, n_3, \ldots, n_m)$ of $m+1$ elements where n_i is the initial number of tokens in the place P_i.
Firing rules in spiking Petri nets:

1. Transition T_i is enabled immediately if all its input places have sufficient tokens and satisfies the guard function.

2. Upon enabling, the input places remain in the inactive state for the delay specified in the transition. During this interval the place neither sends nor receives any tokens.

3. Upon firing the transition removes tokens from the input places and deposits tokens into the output places. After firing of the transition, input places become active.

Generally Petri nets are analyzed using tools to study important behavioural properties of the system like reachability, liveness, boundedness etc. But here we are using spiking Petri net as a computational model and the computed value is expressed as the time gap between the arrivals of first two tokens in the output place.

4 Spiking Petri Net and SN P System

The similarities between spiking neural P system and spiking Petri net are

1. The place in spiking Petri net corresponds to a neuron (cell body soma) in SN P system. Like neuron, a place is able to receive many inputs and transmits the output through one or more arcs. Output place in spiking Petri net corresponds to environment in SN P system. The place/transition is the storage or waiting element of the Petri net model.
2. The arc between the place and transition represents an axon. An axon works with a threshold value which is represented as spiking rules in the neuron σ_i, $1 \leqslant i \leqslant m$ of the form
   a) $E \,/\, a^r \longrightarrow a;t$. In spiking Petri net we can represent this rule with a transition connecting place P_i and P_j where $(i, j) \in syn$, $1 \leqslant j \leqslant m$ and $i \neq j$, having weight of the input arc as r, weight of output arc as 1 and a guard function that enables the transition when number of tokens in the input place is a member of Parikh set of L(E), language generated by the regular expression E. t will be the delay of the transition. If P_i is a place designating the output neuron then $P_j = P_0$.
   b) $a^s \longrightarrow \lambda$, the forgetting rule of an SN P system. In spiking Petri net, we can have an immediate transition connecting places P_i and P_j where $(i, j) \in syn$ with weight of the input arc as s, weight of the output arc as 0 and a guard function enables the transition if number of tokens in the input place is exactly s.
3. After getting fired and before sending the spike to its synapses, the neuron is idle (biology calls this the refractory period) and cannot receive spikes. The corresponding Petri net model assumes that once the transition is enabled until the transition fires the input place will be in inactive state and neither it can receive tokens from other place nor send tokens to other places. After the completion of firing the input place becomes active.
4. Tokens in Petri net place correspond to spikes in the neuron. The initial marking of the spiking Petri net is $(0, n_1, n_2, n_3, \ldots, n_m)$ where n_i is the number of spikes initially present in the neuron σ_i and 0 is the initial number of tokens in the output place. The initial marking of the Petri net corresponds to initial configuration of SN P system.

The following algorithms makes use of these similarities to construct spiking Petri net model for SN P system and vice versa.

4.1 SN P system to Spiking Petri net

Theorem 4.1 (Petri net for SN P system) *For every SN P system* Π *there is an equivalent spiking Petri net.*

Proof: Let Π=(O, σ_1, σ_2, σ_3 ,$\ldots$, σ_m , *syn* , i_0) be a spiking neural P system. Construct spiking Petri net C=(P, T, A, W, Γ, G, P_0, I_0) equivalent to Π using Algorithm 1.

To incorporate the features of SN P system, Spiking Petri net model assumes that once the transition is enabled until the transition fires the corresponding input place will be in inactive state and neither it can receive tokens from nor send tokens to other places. The place becomes active after the completion of firing.

Input: Spiking Neural P system Π=(O, σ_1, σ_2, σ_3 ,. . . , σ_m , *syn* , i_0)
Output: Spiking Petri net C=(P, T, A, W, Γ, G, P_0, I_0) equivalent to Π.

Initially the output place P_0 with no tokens in its place is the only place in P. For each neuron σ_i in Π add place P_i to P.
Add (i_0 , 0) to *syn* in P to connect the output neuron with environment.
for all $\sigma_i = (n_i; R_i)$ with $1 \leq i \leq m$ **do**
  Set the number of tokes in place P_i as n_i
  for every rule of the form E / $a^r \longrightarrow a;t$ in σ_i **do**
    for every $(i, j) \in$ *syn* with $1 \leq j \leq m$ **do**
      Add a new transition T_k to T. Insert arcs (P_i , T_k) and (T_k , P_j) with weight r and 1 respectively into A.
      Set $\Gamma(T_k) = t$.
      if $E \neq \phi$ **then**
        Add a guard function G(T_k) that enables the transition if number of tokens in the input place is a member of Parikh set of L(E), language generated by the regular expression E.
      else
        Add a guard function G(T_k) that enables the transition if number of tokens in the input place is exactly r.
      end if
    end for
  end for
  for every rule of the form $a^s \longrightarrow \lambda$ in σ_i **do**
    for every $(i, j) \in$ *syn* with $1 \leq j \leq m$ **do**
      Add a new transition T_k to T. Update A and W by adding arcs (P_i, T_k) and (T_k, P_j) with weight r and 1 respectively.
      Set the transition as an immediate one with $\Gamma(T_k) = 0$.
      Add a guard function G(T_k) that enables the transition if number of tokens in the input place is exactly s.
    end for
  end for
end for

Algorithm 1: Algorithm to construct Spiking Petri net for an SN P system

Now we have to prove that computational power of both the systems is same.
Each neuron in Π corresponds to a place in Petri net. Number of spikes in each neuron σ_i is represented by number of tokens in each place P_i. The words firing and spiking in SN P system are synonymous to enabling and firing respectively in Petri nets. Corresponding to each rule of type $E/a^r \longrightarrow a;t$ in neuron σ_i and $(i, j) \in syn$, in spiking Petri net C, we have a transition T_k with delay t, input arc from place P_i to T_k of weight r, an output arc from T_k to P_j with weight 1 and guard function that enables the transition if number of tokens in the input place is a member of Parikh set of L(E). If σ_i is an output neuron that sends spikes to environment, in the Petri net model we have place P_0 corresponding to environment and transitions that connects the place corresponding to σ_i with the place corresponding to environment(P_0). Similarly forgetting rules are mapped. In this way, we simulate the work of neurons and synapses by using places and transitions in Petri nets.

If the neuron σ_i spikes using the rule $E/\ a^r \longrightarrow a;t$ or $a^s \longrightarrow \lambda$ and $(i,\ j) \in syn$ in Π then in the corresponding Petri net model C, transition connecting the place P_i to P_j will be fired. Similarly if the output neuron spikes and sends spikes to environment then in Petri net C the place corresponding to output place sends tokens to P_0. The sequence in which the output neuron sends tokens to environment in Π is same as the sequence in which the place P_0 receives the tokens in C. It is clear that the computational power of both the systems is the same.
We describe with an examples to illustrate our approach.

Example 4.1 Consider the pictorial representation of SN P system, in Figure. 2 reproduced from [9]. It is formally represented as:

$\Pi = (\{a\},\sigma_1,\sigma_2,\sigma_3,syn,3)$, with

$\sigma_1 = (2,\{\ a^2/a \longrightarrow a;0,\ a \longrightarrow \lambda\})$,

$\sigma_2 = (1,\{\ a \longrightarrow a;0,a \longrightarrow a;1\ \})$,

$\sigma_3 = (3,\{a^3 \longrightarrow a;0,a \longrightarrow a;1,a^2 \longrightarrow \lambda\})$,

syn=$\{(1,2),(2,1),(1,3),(2,3)\}$.

This SN P system works as follows. All neurons can fire in the first step, with neuron 2 choosing non-deterministically between its two rules. Note that neuron 1 can fire only if it contains two spikes; one spike is consumed, the other remains available for the next step.

Both neurons 1 and 2 send a spike to the output neuron 3; these two spikes are forgotten in the next step. Neurons 1 and 2 also exchange their spikes; thus, as long as neuron 2 uses the rule $a \longrightarrow a;0$, the first neuron receives one spike, thus completing the needed two spikes for firing again.

However, at any moment, starting with the first step of the computation, neuron 2 can choose to use the rule $a \longrightarrow a;1$. On the one hand, this means that the spike of neuron 1 cannot enter neuron 2, it only goes to neuron 3; in this way, neuron 2 will never work again because it remains empty. On the other hand, in the next step neuron 1 has to use its forgetting rule $a \longrightarrow \lambda$, while neuron 3 fires, using the rule $a \longrightarrow a;1$. Simultaneously, neuron 2 emits its spike, but it cannot enter neuron 3 (it is closed this moment); the spike enters neuron 1, but it is forgotten in the next step. In this way, no spike remains in the system. The computation ends with the expelling of the spike from neuron 3. Because of the waiting moment imposed by the rule $a \longrightarrow a;1$ from neuron 3, the two spikes of this neuron cannot be consecutive, but at least two steps must exist in between.

Thus, we conclude that (remember that number 0 is ignored)

$N_2(\Pi) = N - \{1\} \in Spik_2P_3(rule_3,\ cons_3,forg_2)$.

Where $N_2(\Pi)$ is set of numbers computed by an SN P system Π, with the subscript 2 reminding of the way the result of a computation(distance between first two spikes) is defined, and by $Spik_2P_m(rule_k,\ cons_p,forg_q$ the family of all sets $N_2(\Pi)$ computed as above by spiking neural P

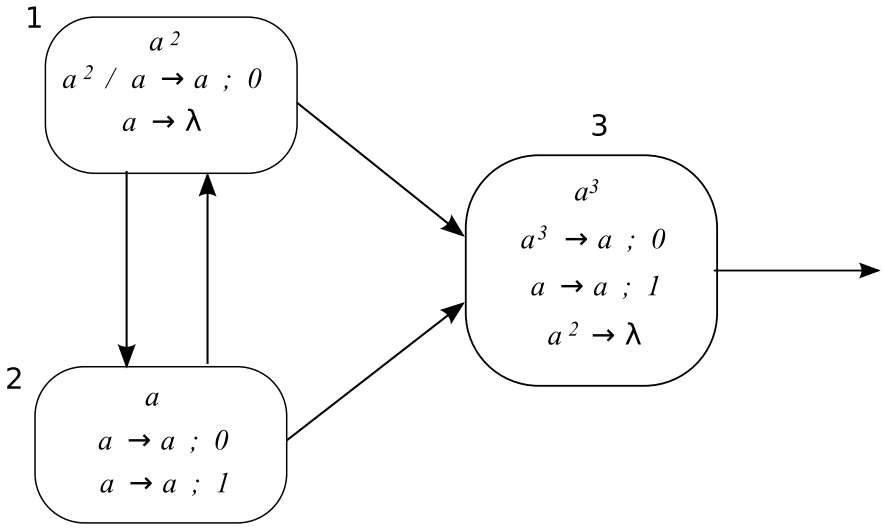

Figure 2: An SN P system generating all natural numbers greater than 1

Rule in Neuron i	Transition	Guard Function for the Transition
$a^2/a \longrightarrow a;0$ in neuron 1	T_1	G (T_1): if $\mu(P_1)$ =2 then true else false
$a \longrightarrow \lambda$ in neuron 1	T_2	G (T_2): if $\mu(P_1)$ =1 then true else false
$a \longrightarrow a;0$ in neuron 2	T_3	G (T_3): if $\mu(P_2)$ =1 then true else false
$a \longrightarrow a;1$ in neuron 2	T_4	G (T_4): if $\mu(P_2)$ =1 then true else false
$a^3 \longrightarrow a;0$ in neuron 3	T_5	G (T_5): if $\mu(P_3)$ =3 then true else false
$a^2 \longrightarrow \lambda$ in neuron 3	T_6	G (T_6): if $\mu(P_3)$ =2 then true else false
$a \longrightarrow a;1$ in neuron 3	T_7	G (T_7): if $\mu(P_3)$ =1 then true else false

Table 1: Mapping between rules in neuron and transitions in Petri net of Example 4.1

systems with at most $m \geq 1$ neurons, using at most $k \geq 1$ rules in each neuron, with all spiking rules $E/\ a^r \longrightarrow a;t$ having $r \leq p$, and all forgetting rules as $a^s \longrightarrow \lambda$ having $s \leq q$. When one of the parameters m; k; p; q is not bounded, then it is replaced with $*$.

Spiking Petri net C corresponding to the SN P system is depicted in Figure.3. Environment as well as neurons are implemented using places and rules inside the neuron are implemented using transition, while synapses are implemented using arcs. The duration between firing and spiking of neuron is represented as a delay in the transition. The mapping between rules in the neuron and transitions is depicted in Table.1. $\mu(P_i)$ denotes the number of tokens in place P_i.

The spiking Petri net C has four places, P_0 corresponds to environment and P_1, P_2, P_3 correspond neurons 1, 2 and 3 respectively. Number of tokens in each place is same as the number of spikes in the corresponding neuron. At first step transitions T_1, T_3, T_4 and T_5 corresponding to the spiking rules used by the neurons in Π in the beginning, are enabled with non-determinism between T_3 and T_4. Transition T_5 deposits a token in the output place. Hence we have to count the time until the next token is deposited in the output place, to define the result of computation. Transition T_1 removes one token from place P_1 and deposits in places P_2 and P_3.Similarly place P_2 also send tokens to P_1 and P_3.

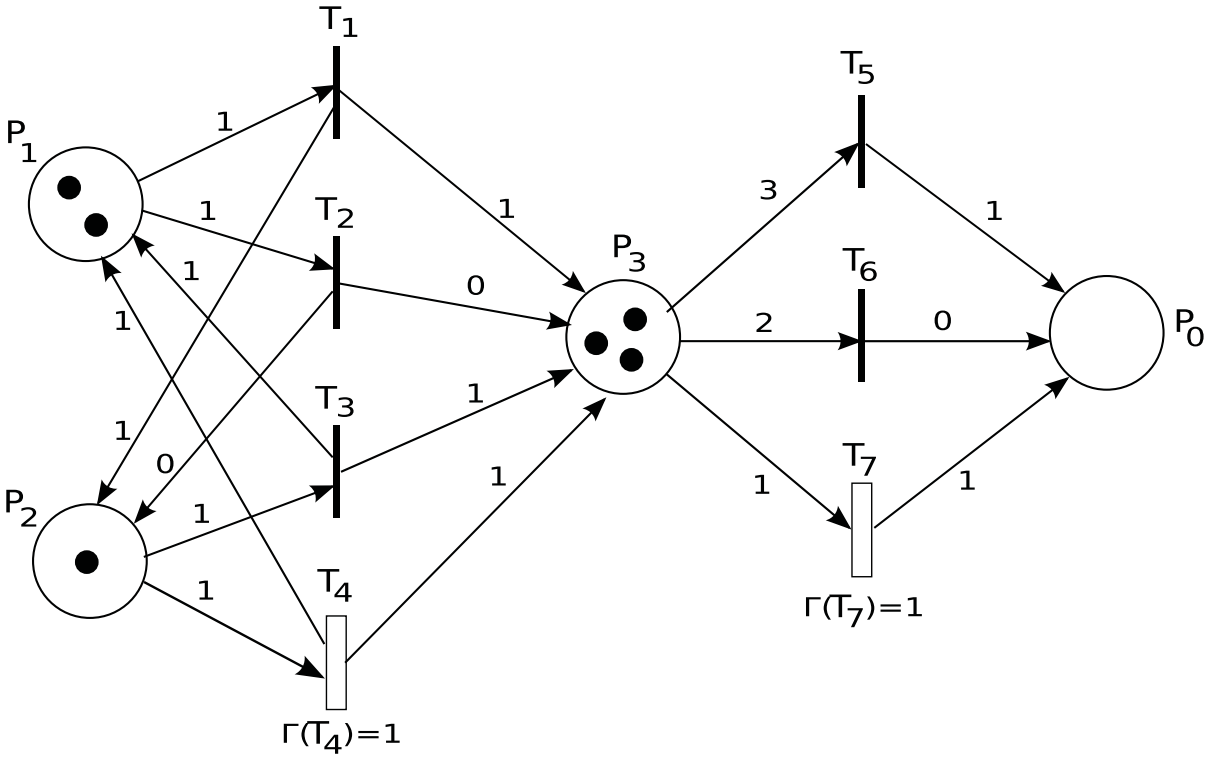

Figure 3: Spiking Petri net C equivalent to SN P system in Figure.2

These tokens are consumed by transition T_6 in the next step. Places P_1 and P_3 also exchange their tokens through the transitions T_1 and T_3; Thus as long as transition T_3 gets priority over T_4, the place P_1 gets one token, thus completing the needed 2 tokens for enabling the transition T_1 again.

Like in SN P system, at any moment starting with first step of the computation, transition T_4 can be enabled. That means that token of place P_1 cannot be deposited in place P_2 as it is in inactive state for one unit of time but can be deposited in place P_3. The transitions T_3 and T_4 cannot be live after this marking as place P_2 will never get a token. Thus place P_3 gets one token, so the transition T_7 will be enabled and fired in the next step. At the same time transitions T_2 will be fired, which consumes token from place P_1. Now the place P_1 is also empty. Transition T_7 deposits the second token in place P_0. Because there is a delay in transition T_4, the place P_0 cannot get two tokens consecutively, but requires at least two time units. It is clear that the computational power of the Petri net model C is equivalent to that of Π.

4.2 Spiking Petri net to SN P system

Theorem 4.2 (SN P system for spiking Petri net) *Let C=(P, T, A, W, Γ, G, P_0, I_0) be a spiking Petri net with*

1. *Transitions having only one incoming arc and all outgoing arcs labeled with either 0 or 1.*
2. *The guard functions are membership functions that checks whether a number is in a Parikh set of regular language over single alphabet.*
3. *C has only one place connected to P_0.*

Input: Spiking Petri net C=(P, T, A, W, Γ, G, P_0, I_0)
Output: Spiking Neural P system Π=(O, σ_1, σ_2, σ_3 ,. . . , σ_m , *syn* , i_0) equivalent to C.
Set O={a}
Make the output place P_0 as environment in Π. If P_k∈P is the place connected to P_0 then set i_0 as k.
for each place $P_i \neq P_0$ with $1 \leq i \leq m$ **do**
  add neuron $\sigma_i = (n_i; R_i)$ to Π
  Set n_i=initial number of tokens in place P_i
  for each transition T_k with P_i as input place **do**
    if weight of any outgoing arc is equal to one **then**
      Derive regular expression E over single alphabet {a} for the Parikh set in the guard function G(T_k)
      Let r be the weight of the incoming arc (P_i,T_k)
      Add spiking rule E / $a^r \longrightarrow a;t$ to R_i
    else
      Let s be the weight of the incoming arc (P_i,T_k)
      Add forgetting rule $a^s \longrightarrow \lambda$ to R_i
    end if
    for each output place $P_j \neq P_0$ of T_k **do**
      add (i, j) to *syn*
    end for
  end for
end for

Algorithm 2: Algorithm for constructing SN P system for Spiking Petri net

Then we can have an equivalent SN P system Π *for C.*

Proof: Given spiking Petri net C=(P, T, A, W, Γ, G, P_0, I_0) with $m + 1$ places . Construct SN P system Π=(O, σ_1, σ_2, σ_3 ,. . . , σ_m , *syn* , i_0) using Algorithm 2.

we can prove the computational equivalence of both systems is similar way as we proved in Theorem 4.1.

If we construct the SN P system for the spiking Petri net in Figure.3 using the Theorem 4.2 we get back the original SN P system in Figure.2.

Conclusion

In this paper, we have introduced spiking Petri nets, which incorporated the features of SN P system. Also presented a methodology to derive a Spiking Petri net model from spiking neural P system and the other way round. Spiking Petri nets can also represent extended SN P system where the rules in the neuron are of the form E / $a^r \longrightarrow a^p;t$, with the meaning that when when using the rule, r spikes are consumed and p spikes are produced($r \geq p$). Because p can be 0 or greater then 0, we obtain a generalization of both spiking and forgetting rules, while forgetting rules also have a regular expression associated with them. This rule can be implemented using spiking Petri net with a transition of delay t having input arc and output arc labeled by r and p respectively. The guard function enables the transition when the number of tokens in the input place is a member of the Parikh set of L(E). The main idea is to generate a Petri net model for SN P systems to allow the use of existing net analysis techniques to study the behaviour of SN P system.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Rao, K. Roberts and P. Walter, "Molecular Biology of the Cell", 4th Ed. Garland Science, New York, 2002.

[2] G. Paun, "Twenty Six Research Topics About Spiking Neural P Systems", http://www.gcn.us.es/5BWMC/volume/snproblems.pdf.

[3] H. Plunnecke and W. Reisig, "Bibliography of Petri nets", in Advances on Petri Nets, Lecture Notes in Computer Science, G.Rozenberg, Ed.Berlin: Springer-Verlag, vol.524, pp.317-572, 1991.

[4] K. Jenson, "Coloured Petri nets: Basic Concepts, Analysis, Methods and Practical Use", EACTS, Monographs on Theoretical Computer Science. Springer-Verlag, 1992.

[5] M. Ionescu, G. Paun and T. Yokomori, "Spiking Neural P Systems", Fundamenta Informaticae, vol.71, No.2-3, pp.279-308, 2006

[6] M.G. Kadjinicolaou, M.B.E. Abdelrazik and G. Musgrave, "Structured Analysis for Neural Networks using Petri nets", Proceedings of 33rd Midwest Symposium on Circuits and Systems, vol.2, pp.770-773, 1990.

[7] N. Chamas, L. Anneberg and E. Yaprak, "Timed Neural Petri nets", Proceedings of the 36th Midwest symposium on Circuits and Systems, vol.2, pp.926-929, 1993.

[8] T. Murata, "Petri nets: Properties, Analysis and Applications", Proceedings of the IEEE, vol 77, No.4, pp.541-580, 1989.

[9] W. Gerstner and W. Kistler, "Spiking Neuron Models Single Neurons, Populations, Plasticity", Cambridge Univ. Press, 2002.