Modeling Spiking Neural P systems using Timed Petri nets

Venkata Padmavati Metta
Department of Comp.Sc. and Engg.
Thapar University
Patiala, India
vmetta@gmail.com

Kamala Krithivasan
Department of Comp.Sc. and Engg.
Indian Institute of Technology, Madras
Chennai, India
kamala@iitm.ac.in

Deepak Garg
Department of Comp.Sc. and Engg.
Thapar University
Patiala, India
deep108@yahoo.com

Abstract

This paper shows that deterministic P-timed Petri nets with inhibitory and test arcs can simulate an SN P system. A method is proposed to translate an SN P system into Petri net model and is illustrated with an example.

Keywords: *Spiking Neural P System; timed Petri net; inhibitory arc; test arc; membrane computing.*

1. Introduction

Neurons in brain communicate with each other by sending electrical signals of identical voltage called spikes through synapses - links established with neighbouring neurons. The spikes of neurons look alike but the timing and number of spikes entering a neuron determines the way the information is encoded. Spiking Neural P (SN P) systems introduced by Gh.Paun[1] are mathematical models inspired from the above stated concepts of neurobiology. SN P system is a variant of P systems, which are a prominent computational model that has been inspired by the way living cells are divided by membranes into compartments where biochemical reactions may take place[2].
There are several variants of SN P systems which are evolved by adding different ingredients of neurobiology like decaying of membrane potential[3], astrocytes[4] etc. This paper considers the basic form called standard SN P system. Different variants of P systems are translated into Petri nets to complement the functional characterisation of their behaviour in[5], [6]. To depict and simulate the behaviour of an SN P system, we introduced the translation of SN P system into spiking Petri nets in[7]. The spiking Petri net introduced was a new variant of Petri net. To incorporate the refractory period of the neuron, each input place in the spiking petri net is inactive between enabling and firing of the transition and does not receive any tokens from its input transitions. In this paper we try to implement SN P systems using already existing features like timed Petri nets, inhibitory and test arcs to represent the refractory period. So using the notions and tools already developed for Petri nets, one can describe the internal process occurring during a computation in the SN P system. It is worth noting that as far as the rules are concerned, SN P systems are highly concurrent in nature and Petri nets are successful modeling paradigm, which allows concurrent systems to be described in a formal yet graphical and well readable way. Furthermore, Petri nets allow for computer aided simulation and what is more, formal analysis of models based on them is also possible.

2. Spiking Neural P System

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
   σ_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 $a^s \notin$ L(*E*) for any rule $E/a^r \longrightarrow a;t$ of type (1) from R_i;
3. *syn* $\subseteq$ { 1, 2, 3, ..., *m*} × { 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 $a^n \in$ L(*E*) and $n \geq r$. If $E=\phi$ then the 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 $t = 0$, then the spikes are emitted immediately, if $t = 1$, then the spikes are emitted in the next step and so on. In the case $t \geq 1$, if the rule is used in step d, then in step $d, d+1, d+2, ..., d+t-1$, the neuron is closed and it cannot receive new spikes (If a neuron has a synapse to a closed neuron and sends spikes along it, then the spikes are lost, biology calls this the refractory period)). In step $t+d$, the neuron spikes and becomes open again, hence can receive spikes(which can be used in step $t+d+1$). 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 results in no new spikes.

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.

Definition 2.1 *(Configuration)* $\mathcal{C} = \langle n_1/t_1, n_2/t_2, \ldots, n_m/t_m \rangle$ is a configuration where neuron σ_i, $i = 1, 2, 3, \ldots, m$ contains $n_i \geq 0$ spikes and it will open after $t_i \geq 0$ steps. The initial configuration of the system is described by $\mathcal{C}_0 = \langle n_1/0, n_2/0, n_3/0, \ldots, n_m/0 \rangle$ where n_i is the number of spikes present in each neuron σ_i for $1 \leq i \leq m$, which is open initially.

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. The rules are used in the non-deterministic manner, in a maximally parallel way at the level of the system; in each step, all neurons which can use a rule of any type, spiking or forgetting, have to evolve, using a rule.

Definition 2.2 *(Vector rule)* A vector rule v is a mapping with domain Π such that each $v(i)$ is at most one instance of rule from $R(i)$ i.e $\mid v(i) \mid= 1$. A vector rule v is enabled at a configuration $\mathcal{C}$ if, for each neuron σ_i of Π, the following hold:

1. if σ_i is closed and $t_i \geq 2$ then no rule can be used i.e. $v(i)$ = $i0$ but if $t_i = 1$ then $v(i)$ = iS (S stands for spiking of the neuron after being closed for $t-1$ steps).
2. if σ_i is open and if $v(i)$ is of the form ij: *E* / $a^r \longrightarrow$*a*;*t* then $n_i \in$ Parikh set of $L(E)$ and $n_i \geq r$. If $v(i)$ is of the form ij: $a^s \longrightarrow \lambda$ then n_i is exactly s. The number of spikes consumed and produced are called *lhs* and *rhs* of the rule respectively.

if a vector rule v is enabled at a configuration $\mathcal{C}$=$\langle n_1/t_1, n_2/t_2, \ldots, n_m/t_m \rangle$ then $\mathcal{C}$ can evolve to $\mathcal{C}'$=$\langle n'_1/t'_1, n'_2/t'_2, \ldots, n'_m/t'_m \rangle$ such that for every σ_i in Π: if $v(i)$ has a rule then

$$t'_i = \begin{cases} t & \text{if } v(i) \text{ is a spiking rule with delay } t \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

else // $v(i)$ has no rule then

$$t'_i = \begin{cases} t_i - 1 & \text{if } t_i \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Definition 2.3 *(Transition)* Using the vector rule, we pass from one configuration of the system to another configuration, such a step is called a transition. For two configurations C and C' of Π we denote by $C \Rightarrow C'$, if there is a direct transition from C to C' in Π.

A computation of Π is a finite or infinite sequences of transitions starting from the initial configuration, and every configuration appearing in such a sequence is called reachable. Note that the transition of C is non-deterministic in the sense that there may be different vector rules applicable to C, as described above.

A computation halts if it reaches a configuration where no rule can be used. With any computation halting or not we associate a spike train, a sequence of digits of 0 and 1, with 1 appearing in position which indicates the steps when the output neuron sends spikes out of the system. One of the neurons is considered to be the output neuron, and its spikes are sent to the environment. With any spike train we can associate various numbers which are considered as computed by the system. Because of the non-determinism in using the rules, a given system computes in this way a set of numbers.

Example 2.1 Figure.1(a) represents the initial configuration of an SN P system Π_2. It is formally represented as:
Π_2 = ({a},σ_1, σ_2, σ_3, *syn*,3), with
σ_1 = (2, { a^2 /*a*$\longrightarrow$*a*;*0*, *a*$\longrightarrow$ λ }),
σ_2 = (1,{ *a*$\longrightarrow$*a*;*0*, *a*$\longrightarrow$*a*;*1* }),
σ_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. Output neuron 3 sends its spike to the environment. 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*$\longrightarrow$ λ, 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

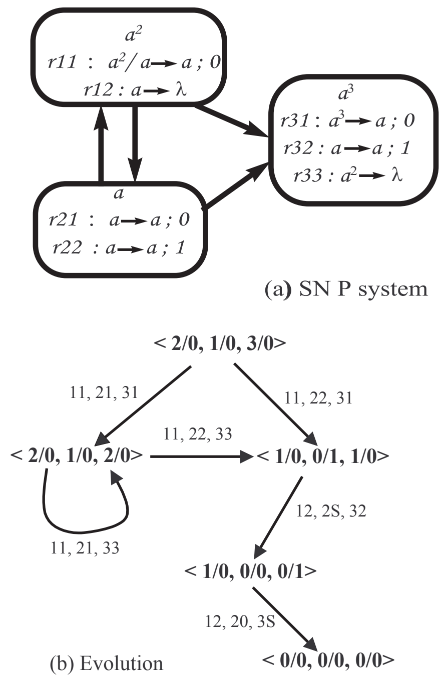

Figure 1.

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.

Figure.1(b) can be used for analyzing the evolution of the system Π_2. Because the system is finite, the number of configurations reachable from the initial configuration is finite, too. Hence, we can place them at the nodes of a graph, and between two nodes/configurations we draw an arrow if and only if a direct transition is possible between them. In Figure.1(a) we represent the labelled rules used in each neuron, with the following conventions: for each rule we have written only the unique subscript ij; when a neuron σ_i, $i = 1, 2, 3$ uses no rule, we have written $i0$, and when it spikes (after being closed for one step), we write iS.

3. Petri net

A Petri net is a bipartite graph with two types of nodes, place nodes represented with circles containing tokens and transition nodes represented with bars or boxes. The directed arcs connecting places to transitions and transitions to places may be labeled with an integer weight, but if unlabelled are assumed to have a weight equal to 1. A transition has a certain number of input and output places representing the preconditions and post conditions of the event respectively. A transition is enabled if all of its input places have tokens equal to or greater than the weight of the arc connecting that place to the transition. A transition without any output place is called a sink transition. Note that the firing of a sink transition consumes tokens but does not produce any. Similarly a place without any output transition is called output place.

Many extensions to the simple Petri net model have been developed for various modeling and simulation purposes. These high level Petri nets include coloured Petri nets [8], which allow tokens to have internal structure, and transitions can have a guard function to further constrain their enabling. Timed Petri nets in which places and/or transitions may be assigned deterministic/probabilistic time delays [9]. Here we have considered deterministic P-timed Petri nets in which each place is associated with deterministic holding time to represent the refractory period of a neuron. When time is associated with a place; it refers to the length of time that the tokens created in that place by any transition are unavailable to the enabled transition.

Inhibitory arc is a special kind of arc which connects a place with a transition and is used to test the unavailability of tokens in a place. The inhibitor arcs thus provide a "test if zero" condition and are represented by circle headed arcs connecting the place to the transition. Similarly test arc, denoted by dotted directed line connecting a place with a transition, does not consume any content of a place at the source of the arc by firing but it allows firing iff a token is present in the input place.

Now we introduce the class of P-timed Petri nets with transitions having guards, inhibitory and test arcs, to be used in the translation.

Definition 3.1*(Petri net)* A P-timed Petri net is represented by $\mathcal{N} = (P, T, A, H, S, W, \Gamma, G, P_0)$, where

$P = \{P_1, P_2, P_3, \ldots, P_m\}$ is a finite, nonempty set of places.

$T = \{T_1, T_2, T_3, \ldots, T_n\}$ is a finite, nonempty set of transitions.

$A \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs which connect places with transitions and transitions with places such that $P \cap T = P \cap A = A \cap T = \phi$.

$H \subseteq$ (P×T) is a (possibly empty) set of inhibitor arcs which connect places with transitions and A∩H=ϕ.

$S \subseteq$ (P×T) is a (possibly empty) set of test arcs which connect places with transitions and S∩H=S∩A=ϕ.

W: A$\longrightarrow \mathbb{N}$ assigns weight $W(f)$ to elements of $f \in A$ denoting the multiplicity of unary arcs between the connecting nodes.

Γ: P $\longrightarrow \mathbb{R}^+ \cup 0$ assigns holding delay, the delays associated to place. If Γ is not defined for place P_i then place is assumed to have no delay.

G: T $\longrightarrow$ {*true*,*false*}, the guard function maps each transition T_i to boolean expression, which specifies an additional constraint which must be fulfilled before the transition is enabled.

$P_0 \in$ P indicates the output place with no outgoing arcs and

is empty in the beginning.
The sets of all input and output places of a transition T_j are denoted by $I(T_j)$={P_i : (P_i, T_j)∈ A} and $O(T_j)$={P_i : (T_j, P_i)∈ A} respectively. Similarly the sets of input and output transitions of a place P_i are denoted by $I(P_i)$={T_j : (T_j, P_i)∈ A} and $O(P_i)$={T_j : (P_i, T_j)∈ A} respectively. The set of all inhibitor places of T_j is denoted by $Inh(T_j)$={P_i : (P_i, T_j)∈ H} and the set of transitions connected by inhibitor arcs with a place P_i is denoted by $Inh(P_i)$={T_j : (P_i, T_j)∈ H}. Similarly a place P_i is a test place of a transition T_j iff $(P_i, T_j) \in$ S. The set of all test places of T_j is denoted by $Tst(T_j)$={P_i : (P_i, T_j)∈ S} and the set of transitions connected by test arcs with a place P_i is denoted by $Tst(P_i)$={T_j : (P_i, T_j)∈ S}.
Definition 3.2 *(Marking)* A marking(state) assigns to each place P_i a non negative integer *k*, we say that place P_i is marked with *k* tokens. Pictorially we place *k* black dots (tokens) in place P_i. A marking is denoted by M, an *m*-vector where *m* is the total number of places. M_0 is the initial marking, the initial number of tokens in each place P_i.
The state or marking of Petri net is changed by the occurrence of transition. Transition T_j is enabled iff T_j satisfies the guard condition and its every input place has at least as many tokens as the weight of the input arcs, inhibitor places have no tokens and each of the test place has a token. Upon firing the transition T_j removes number of tokens from each of its input places(but not inhibitory places and test places) equal to the weight of the input arcs and deposits number of tokens into the output places equal to the weight of output arcs. if the output place is associated with time delay, the tokens are unavailable to its transitions for the length of time equal to delay.
Concurrency is also a concept that Petri net systems represent in an extremely natural way. Two transitions are concurrent at a given marking if they can be fired at the same time i.e. simultaneously. Every transition enabled by a marking M can fire but is never forced to fire. An important concept in Petri nets is that of conflict. Conflict occurs between transitions that are enabled by the same marking, where the firing of one transition disables the other. A major feature of net is that they do not define in any way how and when a given conflict should be resolved, leading to non-determinism on its behaviour.
Definition 3.3 *(Step)* A step is a set U of transitions which are free enabled in a marking. A step U is free enabled at a marking M if every input place of $t \in U$ has sufficient tokens, each inhibitor place does not contain any token, test places has only one token each and satisfies the guard function. A step U which is enabled at a marking M can be executed leading to the marking M'. We denote this by $M\left[U\right\rangle M'$. A computation of a Petri net N is a finite or infinite sequences of executions starting from the initial marking and every marking appearing in such a sequence is called reachable.
A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems such as reachability, boundedness and liveness. The firing of an enabled transition will change the token distribution in a net according to the transition. A sequence of firings will result in a sequence of markings. A marking M_n is reachable from initial marking M_0 if a sequence of firings that transforms M_0 to M_n. The reachability problem for Petri net is the problem of finding if a marking M_i is reachable from the initial marking M_0.

4. SN P Systems and Petri nets

In this section, we translate an SN P system into a behaviourally equivalent Petri net. We also translate some classe of Petri nets into equivalent SN P systems.

4.1. SN P System to Petri net

Let Π=(O, σ_1, σ_2, σ_3 ,..., σ_m , *syn* , i_0) be a spiking neural P system. Construct Petri net $\mathcal{N}$ equivalent to Π using the following procedure.

1. Each neuron σ_i in an SN P system is represented with a place P_i. Output place P_0 in Petri net corresponds to environment in an SN P system. So add the set of neuron places $\{P_0, P_1, P_2, \cdots, P_n\}$ to P. Set $M_0(P_i) = n_i$ and $M_0(P_0) = 0$ where n_i is the initial number of spikes in σ_i. Similar to the state of the neuron, the state of a place P_i is maintained by adding a status place P_i'. Initially $M_0(P_i') = 1$. The absence of token in place P_i' indicates that place P_i is closed, otherwise open. Status place is not maintained for neuron having no timed rules (with $t \geq 1$) as it is always open.
2. The arcs between a place and transition and transition and place represents an axon. The neuron σ_i spikes using rules ij. This is represented with Petri net as follows:
   a) ij : $a^s \longrightarrow \lambda$, the forgetting rule of an SN P system. In Petri net, we have a sink transition T_{ij} with $W(P_i, T_{ij}) = s$ and $G(T_{ij})$= true if $M(P_i) = s$.
   b) ij : E / $a^r \longrightarrow a$;0. Let $k_1, k_2, ..., k_l$ be neurons(called timed neurons) with at least one timed rule such that (i, k_i)∈ *syn* where $1 \leq i \leq l$. Let $y_1, y_2, ..., y_p$ be neurons having no timed rules such that (i, y_i)∈ *syn* where $1 \leq i \leq p$. Create and add T_U to T, a set of 2^l mutually exclusive transitions for checking each combination of states of timed neurons i.e. T_U=∪ $\left\{t_{ijy_1y_2y_3...y_px_1x_2x_3...x_l}\right\}$ where x_i can be k_i(for open state) or k_i'(for closed state). For

each t of the form $t_{ijy_1y_2y_3\ldots y_px_1x_2x_3\ldots x_l} \in T_U$, set $W(P_i,t) = r$ and $G(t)$=true if $M(P_i)$ is a member of Parikh set of $L(E)$. Also

- for each y_i, set $W(t, P_{y_i}) = 1$.
- for each x_i, if it is k_i then add (P'_{k_i}) to $Tst(t)$ and P_{k_i} to $O(t)$ else add (P'_{k_i}) to $Inh(t)$.

Here the group of transitions check the status of output places through test and inhibitory arcs, so that the tokens can be sent to the places which are open at that time.

c) ij : *E / a^r ⟶a;t* where $t \geq 1$. Here the refractory period of the neuron is maintained through timed place P_{ij} with delay $\Gamma(P_{ij}) = t - 1$ that keeps the token unavailable to the transitions for $t-1$ steps. Add T_{ij} to T and update A and W by adding arcs (P_i,T_{ij}), (P'_i,T_{ij}) and (T_{ij},P_{ij}) with $W(P_i,T_{ij})$=*r* and $G(T_{ij})$=true if $M(P_i)$ is a member of Parikh set of $L(E)$.
Similar to the above rule let σ_i be connected to l timed neurons(called timed neurons) and p non-timed neurons through its synapses. Create and add T_U to T. For each t of the form $t_{ijy_1y_2y_3\ldots y_px_1x_2x_3\ldots x_l} \in T_U$

- set $W(t, P'_i) = W(P_{ij}, t) = 1$.
- for each y_i, set $W(t, P_{y_i}) = 1$.
- for each x_i, if it is k_i then add (P'_{k_i}) to $Tst(t)$ and P_{k_i} to $O(t)$ else add (P'_{k_i}) to $Inh(t)$.

The initial marking of the Petri net corresponds to initial configuration of an SN P system C_0. To establish the behavioural equivalence of Π and $\mathcal{N}_\Pi$, we first capture the correspondence between configurations and markings and between enabledness of vector rule and steps.

Definition 4.1 For each marking M of $\mathcal{N}_\Pi$ the configuration C_M is such that for every neuron σ_i for Π, state of

$$\sigma_i = \begin{cases} \textit{closed} & \text{if } M(P'_i)\text{=0} \\ \textit{open} & \text{if } M(P'_i)\text{=1} \end{cases}$$

Moreover for every neuron σ_i of Π, we have M(P_i)=n_i $\forall \sigma_i$ For each step of transitions U of $\mathcal{N}_\Pi$, there is vector rule v_U such that for every neuron $\sigma_i \in \Pi$ and rule $v(i) \in v_U$, if t in U is a transition of the form

1. T_{ij} with $W(P_i, T_j) = s$ with no out going arcs then $\exists$ a rule $v(i)$ of the form $ij : a^s \longrightarrow \lambda$.
2. $T_{ijy_1y_2\ldots.y_px_1x_2\ldots.x_l}$ with input arc from P_i then $\exists$ a rule $v(i)$ of the form $ij : E/a^r \longrightarrow a; 0$.
3. $T_{ijy_1y_2\ldots.y_px_1x_2\ldots.x_l}$ with input arc from P_{ij} then neuron i spikes and $v(i)$ is of the form is (s stands for spiking).
4. T_{ij} with input arcs from P_i and P'_i then $\exists$ a spiking rule $v(i)$ of the form $ij : E/a^r \longrightarrow a; t$.

if U has no transition beginning with subscript i then $v(i) = i0$.

Theorem 4.1: Let M be a reachable marking of $\mathcal{N}_\Pi$, for any execution

1. If $M\,[U\rangle\, M'$ then $C_M \overset{v_U}{\Longrightarrow} C_{M'}$
2. If $C_M \overset{v}{\Longrightarrow} C'$ then there is a step U such that $v = v_U$, $M\,[U\rangle\, M'$ and $C_{M'} \Longrightarrow C'$

Proof: Let C_M be a configuration of Π.
(1) We first show that v_U is enabled at C_M. As U is enabled at M, by the definition 4.1, for every $t \in U$, $\exists$ a rule $v(i) \in v_U$ such that $v(i)$ is enabled at C_M. Hence there is C such that $C_M \overset{v_U}{\Longrightarrow} C$. Moreover $C = C_{M'}$ follows from the algorithm and definition 4.1.
(2) Let σ_i be a neuron such that $v(i)$ is a rule of any one of the four forms (a) ij: *E/ a^r ⟶a;t* then the number of spikes in the neuron σ_i, $n_i \geq r$ and $n_i \in L(E)$. That is the rule is enabled. (b) ij:$a^s \longrightarrow \lambda$ then by the definition 4.1 there exists a sink transition T_{ij} in U_i with $W(P_i, T_{ij}) = s$. (c) is: Neuron σ_i spikes and sends a spike to its neighbouring neurons. (d) $i0$: No rule can be used. By the definition 4.1, $\exists$ a transition $U_i \in U$ for $v(i)$. It therefore follows that $U = \sum_{\sigma_i \in \Pi} U_i$, if $v(i) = i0$ then $U_i = \lambda$. Hence there is M' such that $M\,[U\rangle\, M'$. Moreover $C' = C_{M'}$ follows from the algorithm and definition 4.1.

Example 4.1 Petri net $\mathcal{N}_\Pi$ corresponding to the SN P system in Figure.1, is depicted in Figure.2. The transitions are enabled only if they satisfy the guard functions, which consider the count of tokens in the input place. The Petri net $\mathcal{N}_\Pi$ has eight places, P_0 corresponds to the environment and P_1, P_2, P_3 correspond the neurons 1, 2 and 3 respectively. As neurons 2 and 3 have spiking rules with $t \geq 1$, the neurons will be in closed state for one step after firing these rules. In Petri net model the state of these places are represented using status places P'_2 and P'_3. At the first step transitions $T_{1123}, T_{2113}, T_{22}$ and T_{310} corresponding to the spiking rules used by the neurons in Π in the beginning, are enabled with non-determinism between T_{2113} and T_{22}. Transition T_{310} deposits a token in the output place. Transition T_{1123} 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. These tokens are consumed by transition T_{33} in the next step. Places P_1 and P_3 also exchange their tokens through the transitions T_{1123} and T_{2113}; Thus as long as transition T_{2113} gets priority over T_{22}, the place P_1 gets one token, thus completing the needed 2 tokens for enabling the transition T_{1123} again.
Like in SN P system, at any moment starting with first step of the computation, transition T_{22} 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 with input place as P_2 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_{32} will be enabled and fired in the next step. At the same time transitions T_{12} will be fired, which consumes token from

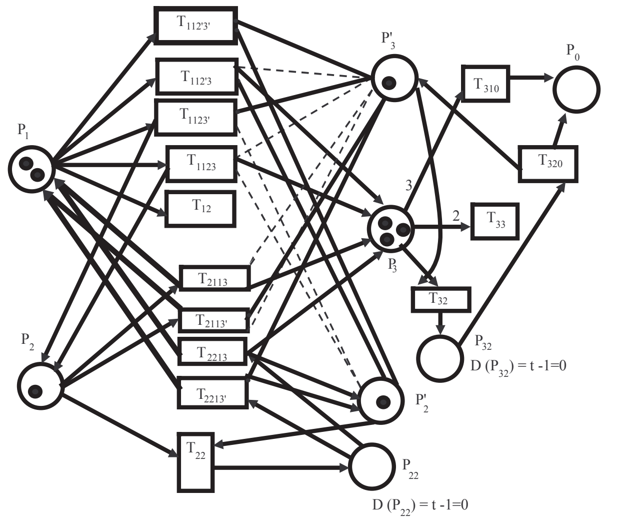

Figure 2. Petri net model equivalent to SN P system

place P_1. Now the place P_1 is also empty. Transition T_{32} deposits the second token in place P_0. Because there is a delay in transition T_{32}, the place P_0 cannot get two tokens consecutively, but requires at least two time units. The Petri net reaches dead marking, where no transition is enabled.

4.2. Petri net to SN P system

Here we prove that every non-timed basic Petri net $N = (P, T, A, W, P_0)$ with $|I(P_0)| = 1$(i.e. number of input transitions for place P_0=1) and every transition $t \in T$ such that $|I(t)| = 1$ and $W(t, P_j) = 1$. N should also have the property that for every $t_i, t_j \in T$ if $I(t_i) = I(t_j)$ then $O(t_i) = O(t_j)$. N is converted into behaviourally equivalent Spiking Neural P system Π=(O, σ_1, σ_2, σ_3 ,..., σ_m , *syn* , i_0) having no timed rules using the following procedure.

1. Set $O = \{a\}$. We have environment in Π for place P_0. Let P_k is the only place connected to P_0. Set i_0 as k.
2. For each place $P_i \in P$ except P_0, add $\sigma_i = (n_i; R_i)$ with $n_i = M_0(P_i)$ to Π. For each $t \in O(P_i)$ do, (i). if $O(t) = \phi$ then add $a^r \longrightarrow \lambda$ to R_i else add $E/a^r \longrightarrow a; 0$ to R_i where $E = a^*$ and $r = W(P_i, t)$ (ii). for each $P_j \in O(t)$ add (i, j) to *syn* if (i, j) is not in *syn*.

We can prove the behavioural equivalence of both systems in a similar way as we proved in Theorem 4.1. To construct an SN P system for a P-timed Petri net with inhibitory and test arcs would be more challenging.

Conclusion

In this paper, we have presented a methodology to derive a Petri net model for spiking neural P system. We gave a formal translation for basic class of SN P systems and proved the equivalence between their behaviours. P-timed Petri nets can also represent extended SN P system in[10], where the rules in the neuron are of the form E / $a^r \longrightarrow a^p$; t, with the meaning that 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 similar to the spiking rule E / $a^r \longrightarrow a^p$; t of standard SN P system but with output arc labeled as p. The main idea of this paper is to generate a Petri net model for SN P systems to allow the use of existing net analysis techniques to study the behaviour and properties of SN P system. We have also given translation for restricted class of Petri nets into SN P systems. It would be interesting to consider different variations and normal forms of SN P systems and find out the suitable class of Petri nets which could simulate these variations and normal forms.

References

[1] Ionescu M, Paun Gh, Yokomori T, *Spiking Neural P Systems*, Fundamenta Informaticae, vol.71, No.2-3, pp.279-308, 2006.

[2] Paun Gh, *Computing with Membranes*, Journal of Computer and System Sciences, vol.61, pp.108-143, 2000.

[3] Freund R, Ionescu M, Oswald M, *Extended Spiking Neural P Systems with Decaying Spikes and-or Total Spiking*, ACME FCT Workshop, Budapest, 2007.

[4] Binder A, Freund R, Oswald M, Vock L, *Extended Spiking Neural P systems with Excitatory and Inhibitory Astrocytes*, Proceedings of the 8th WSEAS international conference on Evolutionary Computing, British Columbia, Canada , June 19-21, 2007.

[5] Kleijn J, Koutny M, Rozenberg G *Process Semantics for Membrane System*, Journal of Automata, Languages and Combinatorics , vol 11, pp.321-340, 2006.

[6] Kleijn J, Koutny M *A Petri net model for membrane system with dynamic structure*, Journal of Natural Computing, 2008.

[7] Padmavati M, Kamala K, Deepak G, *Spiking Neural P Systems and Petri nets*, Int. Workshop on MIR day, Nagpur, 2009.

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

[9] Fred, Bowden D J, *A Brief Survey and Synthesis of the Roles of Time in Petri nets*, Mathematical and Computer Modelling, vol.31, No.10-12, pp.55-68, 2000.

[10] Chen H, Ishdorj T O, Paun Gh, Perez-Jimenez M J, *Spiking Neural P Systems with Extended Rules*, Proceedings of Fourth Brainstorming Week on Membrane Computing, Fenix Editora, Sevilla, vol. I, pp. 241-265, 2006.