Representation of Spiking Neural P Systems with Anti-spikes through Petri Nets

Venkata Padmavati Metta[1], Kamala Krithivasan[2], and Deepak Garg[3]

[1] Bhilai Institute of Technology, Durg, India
vmetta@gmail.com
[2] Indian Institute of Technology, Chennai, India
kamala@iitm.ac.in
[3] Thapar University, Patiala, India
deep108@yahoo.com

Abstract. Spiking Neural P(SN P) system with anti-spikes uses two types of objects called spikes and anti-spikes which can encode binary digits in a natural way. We propose a formal method based on Petri nets, which provides a natural and powerful framework to formalize SN P systems with anti-spikes. This enables the use of existing tools for Petri nets to study the computability and behavioural properties of SN P systems with anti-spikes.

1 Introduction

Spiking neural P systems (shortly called SN P systems) introduced in [1] as a variant of P systems, are biologically inspired parallel and distributed computing models inspired by the neurobiological behaviour of neurons sending electrical pulses of identical voltages called spikes to neighbouring neurons.

SN P system with anti spikes (shortly called SN PA system) introduced in [5], is a variant of an SN P system consisting of two types of objects, spikes(denoted as a) and anti-spikes(denoted as $\overline{a}$) participating in spiking and forgetting rules. We propose a relationship between SN P system with anti-spikes and Petri nets to complement the functional characterization of the behaviour of SN PA systems. The relationship between SN P systems and Petri nets is no means a new idea. Behavioural aspects of different variants of membrane systems were studied by translating them into equivalent Petri net models [4,3]. In [8], a formal translation has been given for basic class of SN P systems with delays into a class of Petri nets. In these Petri nets, places are used to represent neurons and spikes are encoded with tokens. Transitions are used to implement rules inside the neurons. Status places are associated with each place to maintain the state(open or closed) of the each neuron. The inhibitor and test arcs are used to test the status of the place and to only send the tokens to the open places.

To represent spikes and anti-spikes in SN PA systems, we use coloured Petri nets [2] in which tokens can of different types. We introduce coloured Petri nets with localities, with each transition is assigned a location based on the input place and locally sequential and globally maximal firing semantics to the

J. Suzuki and T. Nakano (Eds.): BIONETICS 2010, LNICST 87, pp. 651–658, 2012.

sequential firing in a neuron. The annihilation rule present in each neuron is encoded as a highest priority transition with exhaustive firing semantics. It is worth noting that as far as the rules are concerned, SN P systems are highly concurrent in nature and Petri nets play an important role in the modelling, analysis and verification of concurrent systems. As the procedure is direct, it involves less complexity in translation and the rich theoretical concepts and practical tools from well developed Petri nets can be used in the field of SN PA systems. We can also prove the correctness of the computation of SN PA system and also study the some behavioural properties like boundedness by means of Petri nets.

2 Spiking Neural P System with Anti-spikes

First we recall the definition of SN P system with anti-spikes (or SN PA system).

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

$$\Pi=(O, \sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_m, syn, i_0), \text{ where}$$

1. $O = \{ a, \overline{a} \}$ is the alphabet. a is called *spike* and $\overline{a}$ is called anti-spike.
2. $\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_m$ are neurons, of the form

$$\sigma_i=(n_i, R_i), \ 1 \leq i \leq m, \text{ where}$$

   (a) n_i is the *multiset of spikes or anti-spikes* contained by the neuron.
   (b) R_i is a finite set of *rules* of the following two forms:
      i. $E \ / \ b^r \rightarrow b'$ where E is a regular expression over a or $\overline{a}$, while $b, b' \in \{a, \overline{a}\}$, and $r \geq 1$.
      ii. $b^r \rightarrow \lambda$, for some $r \geq 1$, with the restriction that $b^r \notin \mathrm{L}(E)$ for any rule $E \ / \ b^r \rightarrow b'$ of type (1) from R_i;
3. $syn \subseteq \{ 1, 2, 3, \ldots, m\} \times \{ 1, 2, 3, \ldots, m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$ (*synapses* among neurons);
4. $i_0 \in \{ 1, 2, 3, \ldots, m \}$ indicates the *output neuron.*

The rules of type $E \ / \ b^r \rightarrow b'$ are spiking rules, and they are possible only if the neuron contains n b's such that $b^n \in \mathrm{L}(E)$ and $n \geq r$. When neuron σ_i sends a b, it is replicated in such a way that one b' is sent to all neurons σ_j such that $(i, j) \in syn$. The rules of type $b^r \rightarrow \lambda$ are forgetting rules; r number of $b's$ 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, $a^s \rightarrow \lambda$ is applied only if the neuron contains exactly s spikes.

There is an additional fact that a and $\overline{a}$ cannot stay together, so annihilate each other. If a neuron has either objects a or objects $\overline{a}$, and further objects of either type (maybe both) arrive from other neurons, such that we end with a^r and $\overline{a}^s$ inside, then immediately an annihilation rule $a \ \overline{a} \rightarrow \lambda$, which is implicit in each neuron, is applied in a maximal manner, so that either a^{r-s} or $\overline{a}^{s-r}$

remain for the next step, provided that $r \geq s$ or $s \geq r$, respectively. This mutual annihilation of spikes and anti-spikes takes no time and that annihilation rule has priority over spiking and forgetting rules, so the neurons always contains either only spikes or anti-spikes.

$lhs(v)$ and $rhs(v)$ gives the multiset of spikes/anti-spikes present in the left and right hand sides of rule v respectively. Like in [5], we avoid using rules $\overline{a}^c \rightarrow \overline{a}$, but not the other three types, corresponding to the pairs (a, a), $(a, \overline{a})$, $(\overline{a}, a)$. If $E{=}b^r$ then we will write it in the simplified form $b^r \rightarrow b'$.

The standard SN P system works in a similar way but with only one type of object called *spike(a)* and so there exist no annihilation rules.

Definition 2.2 *(Configuration).* The configuration of the system is described $\mathcal{C} =< n_1, n_2, \cdots, n_m >$ where n_i is the multiset written in the form $n_i = a^x \overline{a}^y$, where x is the number of spikes and y is the number of anti-spikes present in neuron σ_i. Because a neuron always contains spikes or anti-spikes, either $n_i(a) = 0$ or $n_i(\overline{a}) = 0$.

Definition 2.3 *(Vector rule).* We define a vector rule V as a mapping with domain Π such that $V(i) = r_{ij}$, r_{ij} is a spiking or forgetting rule from R_i i.e $\mid V(i) \mid = 0$ or 1 where $1 \leq i \leq m$. If no rule is applicable from σ_i then $V(i) = r_{i0}$. If a vector rule V is enabled at a configuration $\mathcal{C}{=}\langle n_1, n_2, \ldots, n_m \rangle$ then $\mathcal{C}$ can evolve to $\mathcal{C}' = \langle n'_1, n'_2, \ldots, n'_m \rangle$ (after applying annihilation rules in each neuron in exhaustive way), where
$n'_i = n_i - lhs(V(i)) + \sum_{(j,i) \in syn} rhs(V(j))$

Definition 2.4 *(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 $\mathcal{C}$ and $\mathcal{C}'$ of Π we denote by $\mathcal{C} \stackrel{V}{\Longrightarrow} \mathcal{C}'$, if there is a direct transition from $\mathcal{C}$ to $\mathcal{C}'$ 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 $\mathcal{C}$ is non-deterministic in the sense that there may be different vector rules applicable to $\mathcal{C}$, as described above. A computation halts if it reaches a configuration where no rule can be used. There are various ways of using such a device [6].

Example 2.1. Consider the graphical representation of an SN P system with anti-spikes in Fig.1(a), the neurons are represented by nodes of a directed graph whose arrows represent the synapses; an arrow also exits from the output neuron, pointing to the environment; in each neuron we specify the rules and the spikes present in the initial configuration. It is formally denoted as

$\Pi{=}(\{a, \overline{a}\}, \sigma_1, \sigma_2, \sigma_3, \sigma_4, syn\ , 4)$, with
$\sigma_1 = (a^3, \{a^3/a \rightarrow a, a^3 \rightarrow \overline{a}\})$, $\sigma_2 = (a, \{a \rightarrow a\}\ )$,
$\sigma_3 = (a, \{a \rightarrow a\}\ )$, $\sigma_4 = (a, \{a \rightarrow \overline{a}\ , \overline{a} \rightarrow a\}\ )$,$syn{=}\{(1, 2), (2, 1), (1,4), (4,1), (1,3), (3,1)\}$.

We have four neurons, with labels 1, 2, 3, 4; neuron 4 is the output neuron. Initially neuron 1 has three spikes with non-determinism between its first two

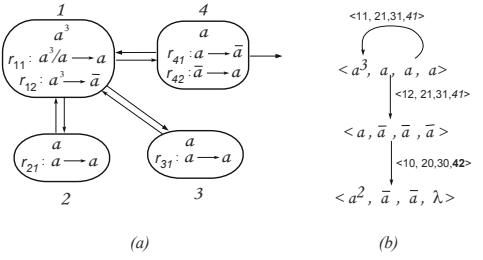

Fig. 1. SN PA system Π

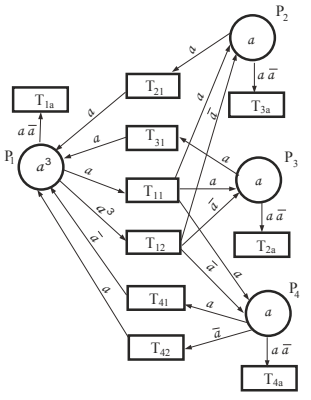

Fig. 2. CPL-net equivalent to Π

rules and neurons 2, 3 and 4 have one spike each. The initial configuration of the system is $< a^3, a, a, a >$.

The evolution of the system Π can be analysed on a transition diagram as that from Fig.1(b) because the number of configurations reachable from the initial configuration is finite, we can place them in 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 the Fig.1(b), we have also indicated the rules used in each neuron with the following conventions; for each r_{ij} we have written only the subscript ij; when a neuron i=1, 2, 3, 4 uses no rule, we have written $i0$.

The functioning can easily be followed on this diagram, so that we only briefly describe it. We start with spikes in all neurons. Neuron 1 can behave non-deterministically choosing one of the two rules. As long as neuron 1 uses the rule $a^3/a \to a$, the computation cycles in the initial configuration sending a spike to neurons 2, 3 and 4; neuron 4 uses its first rule and sends an anti-spike to environment and neuron 1. Neurons 2 and 3 use their rules and send a spike to neuron 1. So neuron 1 receives one anti-spike and two spikes (and two spikes are already present in it), after using annihilation rule, the neuron will have again three spikes. Neuron 2, 3 and 4 will have one spike each.

If neuron 1 uses its second rule $a^3 \to \overline{a}$, the three spikes are consumed and an anti-spike is sent to other three neurons. So neuron 1 will have one spike and neurons 2, 3 and 4 will have one anti-spike each, reaching the configuration $< a, \overline{a}, \overline{a}, \overline{a} >$. In the next step neurons 1, 2 and 3 cannot fire and neuron 4 uses the rule $\overline{a} \to a$ sending a spike to environment and neuron 1 , reaching the configuration $< a^2, \overline{a}, \overline{a}, \lambda >$ and the system halts.

3 Colored Petri Nets

In coloured PN, color refers to the type of data associated with tokens. In other words, tokens can have arbitrary values determined by their type or color. Each

place has an associated color set, which constrains the number and color of tokens that may reside in the place or move along the arc from that place.

Definition 3.1 *(Coloured Petri net).* A coloured Petri net is represented by a tuple $\mathcal{N} \stackrel{df}{=}(\Sigma, P, T, C, A, W, \Gamma, G, M_0)$, where Σ , P and T are finite, non-empty set of colours, places and transitions respectively.

C:$P \rightarrow \Sigma$ is a color function that assigns set of colours to every element of P.

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

W: Assigns a multi-set of coloured tokens from the domain of input places if $f \in (P \times T)$ and from domain of output places if $f \in (T \times P)$.

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.

M_0=$\{m_1, m_2, \cdots m_n\} \in P$, each m_i is a multi set of tokens initially associated with each place P_i and n is the number of places in the net $\mathcal{N}$.

Coloured Petri nets are drawn in a similar way as simple Petri nets with coloured tokens drawn as coloured dots (for tokens of different attributes) or as multi-set inside the places. The directed arcs connecting places to transitions and transitions to places may be labelled with an arc expression having the multi-set of tokens of different colours.

Definition 3.2 *(Marking).* M is a marking which assigns a finite semi-positive multi-set over $C(p)$ to every $p \in P$. M_0 is called the initial marking.

The state or marking of Petri net is changed by the occurrence of transition. Firing rules in the Petri net model are:

1. Transition T_j is enabled iff T_j satisfies the guard condition and its every input place has enough tokens of needed colours as specified in the input arc expression.
2. Upon firing the transition T_j removes number of tokens from each of its input 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.

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. This determines a new marking in a net, a new set of enabled transitions, and so on. 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 in its behaviour. This non-determinism is inherent in Petri nets.

Definition 3.3 *(Step).* A step is a multi-set U of transitions which are enabled at a marking M and we denote this by $M[U\rangle$. The input and output of places of step U are given by $IN_{\mathcal{N}}U(p)$= $\sum_{t \in U} U(t).W(p,t)$ and $OUT_{\mathcal{N}}U(p)$= $\sum_{t \in U} U(t).W(t,p)$ for each $p \in P$

A step U which is enabled at a marking M can be executed leading to the marking $M' = M + OUT_{\mathcal{N}}U$ - $IN_{\mathcal{N}}U$. We denote this by $M\,[U\rangle\,M'$. A step U is a maximal step at a marking M if $M\,[U\rangle$ and there is no transition t' such that $M\,[U + \{t'\}\rangle$ and for every place $p \in P$, transition $t \in U$, t can only be executed if it satisfies the guard function.

A Petri net system $\mathcal{N}$ with maximal concurrency is such that for each markings M and M' if there is a step U such that $M\,[U\rangle\,M'$, then U is a maximal step.

A computation of a Petri net $\mathcal{N}$ 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 reachability problem for Petri net is the problem of finding if a marking M_i is reachable from the initial marking M_0. Formally a Petri net with a given marking is said to be in deadlock if and only if no transition is enabled in the marking. A Petri net where no deadlock can occur starting from a given marking is said to be live. Generally Petri nets are analysed using tools to study important behavioural properties of the system like reachability, liveness, boundedness etc. Before implementation, we introduce the most appropriate Petri net semantics.

Definition 4.1 *(Coloured Petri nets with Localities).* A coloured Petri net with localities (or CPL-net) is a tuple $\mathcal{NL} \stackrel{df}{=} (\Sigma, P, T, C,\ A, W, \Gamma, G, M_0, \psi, L)$, where $\text{UND}(\mathcal{NL}) \stackrel{df}{=} (\Sigma, P, T, C, A, W, \Gamma, G, M_0)$. $\psi : T \to N$ defines the priority function and $L : T \to N$ is a location mapping, for the transition set T.
The markings and steps already defined for coloured Petri nets carry over to CPL-net.

Definition 4.2 *(Enabled step).* A step U is enabled at a marking M if $M \geq IN_{\mathcal{NL}}U$ and, for every place $p \in P$ and transition $t \in U$, $G(t) = true$. Moreover, an enabled U is: *lseq-gmax* (locally sequential globally maximal) enabled if $t \in U$ then there is no transition $u \in U$ such that $L(u) = L(t)$ and there is no transition l such that $U + l$ is enabled at M such there $L(t) \neq L(u)$ for any $u \in U$. The priority of all transitions in a step should be the same. So the priority of the step is same as priority transitions in the step. If two steps are enabled at a marking then the step with higher priority is executed first. In this paper we consider *prioritized locally sequential globally maximal* firing of CPL-nets.

Definition 4.3 *(Executed step and Computation).* We introduce two modes of execution of *lseq-gmax* enabled step U. In *minimal execution mode* every $t \in U$ is fired only once and is denoted as U_{min} where as in *exhaustive execution mode* every $t \in U$ is fired as many times as possible with the same binding for its controlled variables and we denote this as U_{exh}.

Let $m \in \{min, exh\}$ be a mode of execution. A *lseq-gmax* enabled step U at a marking M can be *m-executed* leading to the marking M'. We denote this by $M[U >_m M'$. A *computation* of a CPL-net $\mathcal{NL}$ is a finite or infinite sequence

of executions starting from the initial marking, and every marking appearing in such a sequence is called reachable. The step semantics and the execution modes defined above corresponds to the way rules are fired in the SN PA systems. Using this correspondence we will give in the next section a faithful translation of SN PA systems into CPL-nets.

4 SN P System with Anti-spikes to Petri Net

In this section, we propose a formal method to translate SN PA systems into CPL-nets.

Definition 5.1 *(SN PA system to CPL-net).* Let Π=(O, σ_1, σ_2, σ_3 ,..., σ_m , syn , i_0) be an SN P system with anti-spikes, then the corresponding CPL-net $\mathcal{NL}_\Pi \stackrel{df}{=} (\Sigma, P, T, C, A, W, \Gamma, G, M_0, \psi, L)$, where

1. Σ=O
2. $P = \{P_1, P_2, \cdots, P_m\}$ is the set of places. P_{i_0} is the output place.
3. $T = T_1 \cup T_2 \cup \cdots T_m$ where each T_i contains a distinct transition T_{ij} for every rule of $r_{ij} \in R_i$ and T_{ia}, a transition corresponding to the annihilation rule implicitly present in each neuron P_i.
   For every place $p = P_i \in P$ and every transition $t = T_{jk} \in T$ and $k \neq a$, $W(p,t) = lhs(r_jk)$ if $i = j$, $W(t,p) = rhs(r_jk)$ if $i \neq j$ and $(i,j) \in syn$, $\psi(t) = 1$, $L(t) = i$ and
   for every $p = P_i \in P$, $W(P, T_{ia}) = a\overline{a}$, $\psi(T_{ia}) = 2$, $L(T_{ia}) = i$.
4. For every place $P_i \in P$, its initial marking is $M_0(P_i) \stackrel{df}{=} n_i$.

To capture the very tight correspondence between the SN P system with anti-spikes Π and Petri nets $\mathcal{NL}_\Pi$, we introduce a straight forward bijection between configurations of Π and markings of $\mathcal{NL}_\Pi$, based on the correspondence between places and neurons.

Let $\mathcal{C} =< n_1, n_2, \cdots, n_m >$ be a configuration of an SN PA system Π. Then the corresponding marking $\phi(\mathcal{C})$ of $\mathcal{NL}_\Pi$ is given by $\phi(\mathcal{C}))(P_i) \stackrel{df}{=} n_i$ for every place P_i of $\mathcal{NL}_\Pi$.

Similarly, for any vector rule $V = (r_{1j_1}, r_{2j_2}, \cdots, r_{mj_m})$ of Π, we define a *lseq-gmax* enabled step $\xi(V)$ of transitions of $\mathcal{NL}_\Pi$ such that $\xi(V)(T_{ij}) \stackrel{df}{=} r_{ij}$ for every $T_{ij} \in T$ and $j \neq a$. It is clear that ϕ is a bijection from the configurations of Π to the markings of $\mathcal{NL}_\Pi$, and that ξ is a bijection from vector rules of Π to *lseq-gmax* enabled steps of $\mathcal{NL}_\Pi$.

We now can formulate a fundamental property concerning the relationship between the dynamics of the SN PA system Π and that of the corresponding CPL-net: $\mathcal{C} \stackrel{V}{\Longrightarrow} \mathcal{C}'$ *if and only if* $\phi(\mathcal{C})[\xi(V >_{min} [H >_{exh} \phi(\mathcal{C}')$.

where H is *lseq-gmax* step only containing $T_{ia} \in T$ for every $1 \leq i \leq m$.

Since the initial configuration of Π corresponds through ϕ to the initial marking of $\mathcal{NL}_\Pi$, the above immediately implies that the computations of Π coincide with the locally sequential and globally maximal concurrency semantics of the CPL-net $\mathcal{NL}_\Pi$.

The reader might by now have observed that the structure of neurons in Π is used in the definitions of the structure of the CPL-net $\mathcal{NL}_\Pi$ (i.e., in the definitions of places, transitions and the weight function). Let $\mathcal{C}$ be a configuration of Π and there is a vector rule V enabled at $\mathcal{C}$ reaching a configuration $\mathcal{C}'$. As there is a mapping between configuration and markings, $\phi(\mathcal{C})$ is marking of CPL-net $\mathcal{NL}_\Pi$ corresponding to the configuration $\mathcal{C}$ of Π. There is a one-to-one mapping between the rules in the SN PA system and transitions in CPL-net. For locally sequential and globally maximal concurrency firing semantics of SN PA systems, the *prioritized locally sequential globally maximal steps* are defined CPL-nets. Two execution modes *minimal* and *exhaustive* are defined to encode the working spiking rules and annihilations rules in the SN PA system. So there exists a step $\xi(V >$ enabled at the marking $\phi(\mathcal{C})$. After the execution of the step in the minimal mode, and after firing the step H containing the transitions corresponding the the annihilation rules in exhaustive way, the system reaches the configuration $\phi(\mathcal{C}')$. We can prove only if part in the similar way. So the evolution of the CPL-net $\mathcal{NL}_\Pi$ is same as the evolution of the SN PA system Π. The CPL-net $\mathcal{NL}_\Pi$ corresponding to the SN PA system Π is given in the Fig.2.

5 Conclusion

In this paper we have proposed an approach to the performance modeling of the behaviour of SN P systems with anti-spikes through a class of Petri nets, called coloured Petri nets with localities. The annihilation rule implicitly present in a neuron is successfully encoded as highest priority sink transition and its exhaustive firing enables to represent working of annihilation rule. Based upon the introduction these features, the neural structure can be successfully encoded as a Petri nets model which permit the description the behavioural state based process run-time structure change of SN P system with anti-spikes.

References

1. Ionescu, M., Păun, G., Yokomori, T.: Spiking Neural P Systems. Fund. Infor. 71, 279–308 (2006)
2. Jensen, K.: A Brief Introduction to Coloured Petri Nets. In: Brinksma, E. (ed.) TACAS 1997. LNCS, vol. 1217, pp. 203–208. Springer, Heidelberg (1997)
3. Kleijn, J., Koutny, M.: A Petri net model for membrane systems with dynamic structure. Natural Computing 8(4), 781–796 (2009)
4. Kleijn, J., Koutny, M., Rozenberg, G.: Towards a Petri Net Semantics for Membrane Systems. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 292–309. Springer, Heidelberg (2006)
5. Linqiang, P., Păun, G.: Spiking Neural P Systems with Anti-Spikes. Int. J. of Computers, Communications and Control 4, 273–282 (2009)
6. Păun, G.: Spiking Neural P Systems Used as Acceptors and Transducers. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 1–4. Springer, Heidelberg (2007)
7. Reisig, W., Rozenberg, G. (eds.): APN 1998. LNCS, vol. 1491, 1492. Springer, Heidelberg (1998)
8. Venkata Padmavati, M., Kamala, K., Deepak, G.: Modeling Spiking Neural P systems using Timed Petri nets. In: Int. Conf. on Nature and Biologically Inspired Computing (NaBIC 2009). IEEE Xplore (2009)