Int Jr of Advanced Computer Engineering and Architecture
Vol. 1, No. 1, June 2011

www.ijacea.yolasite.com

Modelling Spiking Neural Activities of Brain with Object Oriented Petri net

Mrs.Padmavati Metta[1], Dr. Deepak Garg[2] and Dr. (Mrs.) Kamala Krithivasan[3]

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

Abstract

We propose object-oriented Petri nets for effective modelling of spiking concept of neurons. Brain is the central part of nervous system and neurons are one-membrane cells, which work concurrently to process information from sensory organs. Modelling neurons provides precise and exact ways of expression, which allow us to go beyond the insights that intuitive or commonsense reasoning alone, can yield. As the neural activities are modular and concurrent in nature we need models like Object Oriented Petri nets that are most suitable for modelling these activities.

Keywords: *object-oriented Petri nets; modelling; brain*

1 Introduction

Brain development and function are inseparable facets of a whole, whose understanding requires both theoretical and modeling approaches. The operation of a neuron is in real time asynchronous, concurrent and nonlinear with an output of the on-off type i.e. binary. In recent years experimental evidence has been accumulating to suggest that biological neural networks, which communicate through spikes, use the timing of these spikes to encode and compute information. Computational neuroscience is oriented towards modeling the spiking nature of

the neurons [1] and retaining the essential elements of the behavior being modeled, while trying to simplify the complexity of the resulting description. The principal motivation for the creation of simplified models is that they allow studying more easily the computational and functional principles of spiking neural systems.

Petri nets [2] offer one means to perform some of the modelling, analysis and design of such a system in terms of its interacting components, concurrent processes and complex logical relationships. Further more analysis methods are needed for the evaluation of the produced neural networks in terms of its performance.

The neural models created capture the spiking nature of the neurons and retain the essentials of the behavior to be modeled, while trying to simplify the description. The structural and behavioural properties of neuron were studied using neural Petri Nets [3]. The neural net architecture was described with Timed Neural Petri nets [4] that considered various chemical changes in the neurons.

However modelling the neural system by an ordinary Petri net was usually organized by analyzing the logic structure, i.e. process-oriented. Object oriented modeling changes this viewpoint. The main idea of object-oriented techniques is breaking down a system into individual objects; each with its own specified externally observable behavior. The actual details of the components are hidden from view of other designers who use the components. We find the activity of each neuron is identical and can be viewed as objects. Message transfer is used to describe the communication between objects. This paper focuses upon the description of a simplified spiking neural model using Object Oriented Petri nets.

The paper is organized such that section 2 reviews the basic operation of spiking neuron; section 3 introduces the Object Oriented Petri nets while section 4 uses the theory of Object Oriented Petri nets for modeling and representing spiking neurons.

2 Spiking Neurons

A typical neuron can be divided into three functionally distinct parts, namely the dendrites, the soma, and the axon. The soma is a main cell body from which runs an axon, a long fibre, which may be regarded as a lossless transmission line. The axon terminates in a series of small dendritic branches, the ends of which lie on or

near the cell bodies of other neurons. The primary function of the soma is to perform the continuous maintenance required to keep the neuron functional.

A neuron receives connections from thousand other neurons. The site where the axon of a pre-synaptic neuron makes contact with the dendrite (or the soma) of a postsynaptic cell is the synapse. In the synapse, the signal can be nonlinearly strengthened or weakened. The strength or efficacy of synaptic transmission is also called a synaptic weight. One neuron receives and sends out signals through from 103 to 105 synapses. Electrical signals transmitted by synapses can have a positive or negative electric sign. In the former case, we speak about excitatory synapses and in the latter case about inhibitory synapses. The voltage response of the postsynaptic neuron to a pre-synaptic action potential is referred to as the postsynaptic potential (PSP). When the sum of positive and negative PSPs (signals) weighted by synaptic weights gets bigger than a particular value, called the excitatory threshold, a neuron fires; that is, it emits an output signal called a spike. A spike is also called an action potential or a nerve impulse. After the firing of an action potential, the neuron enters a refractory period when no further action potentials can be generated. During this period the neuron does not receive any input from other neurons that means the neuron will be in the closed state. The output signal is transmitted down the axon, which delivers it to other neurons. Even with very strong input, it is impossible to excite a second spike during or immediately after a first one. This causes that action potentials in a spike train, the sequence of spikes sent by neuron are usually well separated. Neurons use the spatial and temporal information of incoming spike patterns to encode their message to other neurons.

3 Object Oriented Petri Nets

Petri nets allow us to formalize the behaviour of distributed systems. Petri nets were designed to be able to model true concurrency, key element in parallel and distributed environment. The main advantage to use Petri nets to model parallelism are: graphical nature, well defined semantics that allow formal analysis and support true concurrency, resulting model can be simulated and tested for completeness and correctness using Petri net tools.

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 number of tokens in each place gives the current state of the modeled system. 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). The transitions can be fired concurrently thus representing concurrent activities in the system. In addition tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state charts, algebraic equations and other mathematical models governing the behaviors of system. But Petri nets lack modularity. Furthermore, Petri net approach can be easily combined with other techniques and theories such as object-oriented programming resulting Object Oriented Petri nets [5].

Object-Oriented Petri Nets (OOPN), which combines the maintainability and reusability of Object Orientation (OO) and the advantages of Petri Nets, a graphical interface and a sound theoretical background. They are characterized by object-orientation enriched with concurrency and polymorphic transition execution, which allow for message sending, waiting for and accepting responses, creating new objects, and performing primitive computations. OOPNs are based on viewing objects as active servers, which offer reentrant services to other objects. Services provided by objects, as well as independent activities of objects, are described by OOPNs-services by method nets, object activities by object nets. Tokens in nets represent references to objects.

3.1 The Static Structure of OOPN

An OOPN consists of Petri nets organized in classes. Every class consists of an object net describing the internal activity of objects of this class, a set of dynamically instantiable method nets describing how these objects respond to messages, and a set of predicate methods (or predicates) allowing for testing its objects' states. Particular method and object nets are common high-level nets. They consist of places and transitions interconnected by arcs. Every place has an initial marking and an initial action; every transition has a guard and an action. A guard is an enabling condition associated with a transition. Arcs are inscribed by arc expressions, which specify what tokens, will be removed or added with respect to the concrete binding. From the point of view of transitions, there are input, output, and testing arcs. All method nets of a given class share access to the appropriate object net (places of the object net are accessible for transitions of method nets). Thus objects' behavior can be influenced from outside. Each method net has distinguished parameter places and a return place. There also exist special method nets, which play the role of constructors of objects. Further,

classes can contain predicate methods which have the form of transitions with associated message patterns and which allow for atomic testing their objects' states without any side effects.

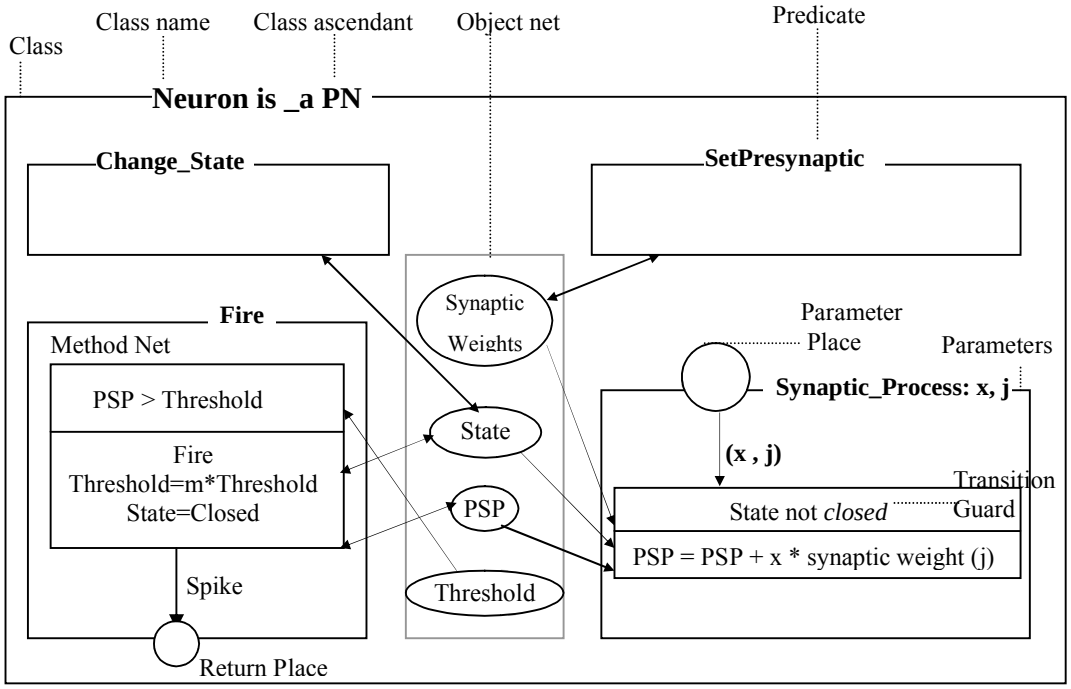

Fig.1: An Object Oriented Petri net model demonstrating the neural activity

3.2 The Dynamic Behaviour of OOPN

Tokens in a running OOPN represent objects. Every object is either trivial (e.g. a number or a string) or it is an instance of some Petri net-described class consisting of one instance of the appropriate object net and currently running instances of method nets. When an object receives a message, a new instance of the corresponding method net is created, parameters are put into the parameter places and the instance of the method net is being executed concurrently with all other net instances until the return place receives a token. Then the value of the token in the return place is passed back to the message sender as the result of the requested service, and the instance of the method net is deleted. Message sending and object creations are specified as actions attached to transitions. In the case of guards, only trivial or predicate methods can be evaluated to assure atomicity. A

transition can be fired for a particular binding if there are enough tokens in its input places with respect to the values of arc expressions within the given binding and if the transition guard evaluates to true. Then tokens from input places are removed, the transition action is performed, and as soon as that is finished, the output marking is produced.

There are tools like PNtalk, which have been developed to support modelling, investigation and prototyping concurrent object oriented Petri nets [6, 7].

4 Representing Spiking Neural Activity with OOPN

A spiking model of a neuron, an element of the SNN, can be for instance inspired by the spike-response model (SRM) of a neuron. Neuron *i* receives input spikes from pre synaptic neurons $j \in \Gamma_i$, where Γ_i is a pool of all neurons pre synaptic to neuron *i*. The state of neuron *i* is described by the state variable $u_i(t)$ that can be interpreted as a total PSP at the membrane of soma. When $u_i(t)$ reaches the firing threshold $v_i(t)$, neuron *i* fires (i.e., emits a spike). The value of the state variable $u_i(t)$ is the sum of all postsynaptic potentials. ***i.e***

$$u_i(t) = \sum_{j \in \Gamma_i} x_j w_{ij}$$

The weight of synaptic connection from neuron *j* to neuron *i* is denoted by w_{ij}. It takes positive (negative) values for excitatory (inhibitory) connections, respectively. Depending on the sign of w_{ij}, a presynaptic spike generated at time t_j increases (or decreases) $u_i(t)$ by x_j, where x_j is an amount expresses an individual PSP evoked by a presynaptic neuron *j* on neuron *i*.

The model in Fig.1 represents an OOPN for the spiking neuron. It has predicates ***SetPresynaptic, Change_State,*** which opens the state of the neuron. It has two method nets *Fire* and *Synaptic_Process*. Fire method has transition which returns a spike if PSP is greater than threshold. Method synaptic_process takes two input parameters *x* and *j* where *x* is the PSP of the presynaptic neuron *j*. The transition increments the PSP of neuron by *x* time the synaptic weight of the connection from neuron *j*.

5 Conclusion

The principal motivation for the creation of simplified models is that they allow studying more easily the computational and functional principles of systems. In this paper we have presented a new model for spiking neural networks based on Object Oriented Petri nets. Such a model can be successfully employed for simulating the behaviour of the spiking neuron. The advantage of this model over others it provides modularity, hence it reduces the overall complexity.

References

[1] W. Gerstner and W. M. Kistler, “Spiking Neuron Models”, Cambridge Univ. Press, Cambridge, 2002.

[2] T. Murata, "Petri Nets: Properties Analysis and Applications ", in proc. of the IEEE, ~0177N, o. 4, pp. 541 -560, 1989.

[3] 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.

[4] 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.

[5] V. Janousek, “Modelling Objects by Petri Nets”, Ph.D thesis, Department of Computer Science and Engineering, FEECS, Technical University of Brno, Czech Republic, 1998.

[6] B. Krena, “Type Analysis in Object-Oriented Petri Nets”, Technical Report, Department of Computer Science and Engineering, FEECS, Technical University of Brno, Czech Republic, 2001.

[7] T.Vojnar, “Evaluating Time-Dependent Properties of Models Described by Object-Oriented Petri Nets”, Ph.D. thesis proposal, Department of Computer Science and Engineering, Technical University of Brno, Czech Republic, 1998.

[8] The Petri Nets World, 2001, an on-line database of Petri net-related conferences, mailing lists, bibliographies, tool databases, newsletters, research groups, etc. URL: http://www.daimi.au.dk/PetriNets/.