An event-driven framework for the simulation of networks of spiking neurons

Olivier Rochel, Dominique Martinez

An event-driven framework for the simulation of networks of spiking neurons

Olivier Rochel, Dominique Martinez

LORIA - Campus scientifique B.P. 239
F-54506 Vandœuvre-les-Nancy Cedex
E-mail : {rochel,dmartine}@loria.fr

Abstract. We propose an event-driven framework dedicated to the design and the simulation of networks of spiking neurons. It consists of an abstract model of spiking neurons and an efficient event-driven simulation engine so as to achieve good performance in the simulation phase while maintaining a high level of flexibility and programmability in the modelling phase. Our model of neurons encompasses a large class of spiking neurons ranging from usual leaky integrate-and-fire neurons to more abstract neurons, e.g. defined as complex finite state machines. As a result, the proposed framework allows the simulation of large networks that can be composed of unique or different types of neurons.

1 Introduction

In an event-driven simulation, the simulated time (often called virtual time) is advanced by computing the state of the system at event occurence instants only, whereas in a time-driven simulation it is advanced using arbitrary time steps [2]. Mapping such an event-driven scheme to a pulsed coupled neural network is straightforward : the pulses (or spikes) are instantaneous, can occur at any time, and therefore can be seen as the "events" that determine the evolution of the system. In the context of a spiking neural network simulation, a basic event-driven simulation engine thus follows this scheme :

1. Find the next event to be processed, that is, the next neuron that should fire (or receive) a spike.
2. Update the state of the neuron concerned by this event
3. Schedule possible events induced by that change
4. If some events are pending, return to the first step. Otherwise, the simulation ends.

Previous research has proven that such an event-driven approach is well suited to the simulation of large networks of spiking neurons, since it leads to fast simulations while handling the difficult task of dealing with the high precision required in the computation of spike times [7]. However, the event-driven software simulators that have been developed so far are specific to particular models of neurons or networks. For example, the event-driven simulators in [11, 4, 8, 7] are rather dedicated to integrate-and-fire neurons, the one in [1] is dedicated to neurons similar to automata with a finite number of states.

In contrast, we propose in this paper an event-driven framework in which the neuron models are only limited by the fact that they can be implemented in an event-driven fashion. This encompasses a large class of spiking neurons ranging from usual leaky integrate-and-fire neurons to more abstract neurons, e.g. defined as complex finite state machines. As a result, the proposed framework features a high level of flexibility that allows the simulation of large networks composed of unique or different types of neurons.

2 Spiking neuron models

2.1 Abstract neuron model

We first need to define an abstract model of neurons to be used within our event-driven framework. According to the basic algorithm described above, the following requirements must be fulfilled by such a neuron : we must know how its internal state is affected by the reception of a spike, how its internal state is modified when emitting a spike, and when its next firing will occur.

We therefore define an abstract model of neurons as a set $\{x_i, r_i, s_i, \tilde{t}_i\}$, with

- $x_i \in X$ is the state variable of the neuron and X is a given state space. This variable can change only at the times of some events occuring in the system.
- $r_i : X \times S \times \mathbb{R} \mapsto X$ is the function that describes the change of the state variable driven by the reception of a pulse from a synapse $s \in S$, where S is the set of all synapses, at time $t_r \in \mathbb{R}$. We will be more specific about the synapses in section 2.2.
- $s_i : X \mapsto X$ caracterizes the change of state variable caused by the *firing* of the neuron (reset function).
- $\tilde{t}_i : X \mapsto \mathbb{R}^+ \cup \{+\infty\}$ gives the time of the next firing, given the present state variable, with the additional hypothesis that no event-driven change of state variable will occur until then. We need to provide the special value $+\infty$ as a way to signify that no firing can occur without further events.

The simulation engine uses the $\tilde{t}_i$'s to find the next firing event pending. This, together with a method to take care of (possibly delayed) reception events

scheduled in step 3) of the above-mentioned algorithm, permits the completion of step 1). s_i or r_i will then be used to complete the second step.

2.2 Connectivity

The abstract model described in the previous section is implicitely based on the assumption that the connectivity of the spiking neural networks to be simulated is of a very classical type : there exists a (fixed) set of oriented connections between the neurons, and a neuron has only one output channel (one axon) such that a spike emitted by a neuron will always be transmitted to all its successors (all the neurons linked to its axon). The latter assumption explains why the function s_i does not provide any explicit way of targeting particular neurons.

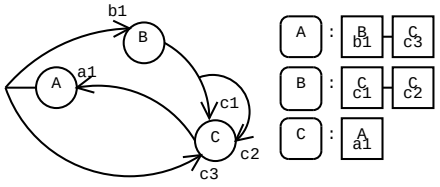

Figure 1

Such a connectivity permits that the step 3) of the event-driven simulation algorithm (reception events scheduling) is performed in a centralized way by the simulation engine, as it only requires the knowledge of the list of successors for each neuron. Indeed, it is possible that more than one connection exists between a pair of neurons (e.g. with different time delays), so a single neuron can appear more than once in the successor list. Then, at firing time, the simulation engine will schedule exactly one event per synapse in the successor list. Moreover, in order for a receiver neuron to react to an incoming spike, it will have to know which synapse is concerned, which requires that all synapses be identified in the successor list. The synapse identity corresponds to the s parameter of the r_i function given in the previous section.

Figure 1 (left) shows a sample network connectivity, with 3 neurons (A,B,C) with the synapse identities expressed as a1,..,c2,c3. On the right, the corresponding successor list for each neuron is represented.

2.3 Relation to other models

Let us now consider a leaky integrate-and-fire neuron i, whose membrane potential V_i obeys the following equation

$$\frac{dV_i}{dt} = -V_i + I_i \tag{1}$$

where I_i corresponds to a constant input current. The neuron is further defined by a threshold mechanism, i.e. it will fire whenever $V_i > \theta_i$ and its potential will be set to zero ($V_i = 0$) at firing times. For the sake of simplicity, we consider that $I_i > \theta_i$ in the following. We further assume in the following that whenever the neuron i receives a spike through a synapse s, at time t_r, the membrane potential $V_i(t_r)$ instantaneously jumps of an amplitude w_s.

To rewrite the above integrate-and-fire neuron according to our model we consider that the state variable x_i is defined as the vector $(V_i^0,\ t_i^0)$ in which $V_i^0 = V_i(t_i^0)$ represents the potential at the time t_i^0 of the latest event (emission or reception of a spike). The integrate-and-fire neuron is now fully described by :

$$r_i(x_i, s, t_r) = \begin{bmatrix} V_i(t_r) + w_s \\ t_r \end{bmatrix} \tag{2}$$

$$s_i(x_i) = \begin{bmatrix} 0 \\ \tilde{t}_i(x_i) \end{bmatrix} \tag{3}$$

$$\tilde{t}_i(x_i) = \begin{cases} t_i^0 & \text{if } V_i^0 \geq \theta_i \ , \\ t_i^0 + log\left(\frac{I_i - V_i^0}{I_i - \theta_i}\right) & \text{otherwise.} \end{cases} \tag{4}$$

where $V_i(t_r)$ is found by integrating Eq. (1) : $V_i(t_r) = I_i + (V_i^0 - I_i) \exp\left(t_i^0 - t_r\right)$.

More generally, to rephrase a threshold-based spiking neuron within the event-driven framework described above, the firing time has to be computed explicitely. This is possible for the leaky integrate-and-fire (see above). However, it is common that no analytical solution is available if one considers more biologically plausible synaptic interactions, such as the use of postsynaptic currents given by alpha functions. In such a case, we can still use a numerical scheme to estimate the next firing time with a given precision, as stated in [7]. A similar technique has been used by Hansel and al. in [5]. Note however that when the focus of a study is mainly on modelling precisely the shape of a postsynaptic potential, a framework such as the Spike Response Model [3] is probably better suited than ours.

Obviously, our abstract model does not require that the state variable of the neuron should be derived from the time course of an underlying membrane potential. Other choices are possible : for example, the neuron models based on finite state automata in [1] could be used as well. Another possible choice is to take the time of the next firing (assuming it always exists) as the state variable. Interactions from other neurons will lead to advance (excitatory connections) or delay (inhibitory connections) that time.

3 Simulation engine

So far, we have described an abstract model of a network of spiking neurons. Throughout the description, the basic interactions with an appropriate simulation engine have been underlined as well. We now need to complete the description of the engine, more precisely to explain how it can handle the event flow in a deterministic way. The basic algorithm for a simulation engine has already been described in section 1, and corresponds to a classical event-driven simulation algorithm. We now focus on some key points that strongly determine the efficiency and correctness of event ordering.

Using a good data structure for event ordering (priority queue) is often seen as the critical point when implementing general-purpose event-driven simulation frameworks [10]. In the context of spiking neuron simulation, that issue has been addressed in recent works on simulation algorithms for some specific spiking neurons models, such as integrate-and-fire neurons in [7] or finite state automata-based neurons in [1]. It must be noted however that the simulation engine has to deal with two rather different event types : the reception events, once scheduled, cannot be cancelled nor rescheduled at another time, while the firing events can be rescheduled or cancelled by forthcoming reception events. That particular point means that it is almost essential to design two different data structures aimed at proper ordering of each event type. As an exhaustive study of the possible data structures for implementing good priority queues in each case is beyond the scope of this paper, we will just point out a worthwhile optimization which is related to the way the neurons interact, as explained before in section 2.2. The successor list provides the basics to schedule, at firing time, the reception events for each successor neuron. When using time delayed receptions, it is generally worthwhile to maintain these lists ordered, so as to insert in the pending event list only the event associated with the smaller time delay, thus effectively limiting the priority queue length. As soon as this event will be processed, the next event in the ordered list will be explicitely scheduled until no remaining connection is left.

Another reason that complicates the design of the underlying data structures relies on the fact that some events (e.g. pulse receptions) can share the same timestamp (synchrony). Until now, we assumed implicitely that the simulation engine was provided a way of ordering the events, i.e. sorting events by their timestamps. In order to fully define the simulation of a network of spiking neurons, we have then to provide an explicit rule for ordering events with equal timestamps. Multiple rules can be used, depending on the choice of the user : data-structure based (FIFO[1]-like), random choice, or more specific rules defined from the available parameters (synapse identity, neuron identity, type of event...).

4 Conclusion

We have presented an event-driven framework that consists of an abstract model of spiking neurons and an efficient event-driven simulation engine. This framework is dedicated to the design and the simulation of networks of spiking neurons and presents a high level of flexibility and programmability. This allows to build and simulate networks of classical spiking neurons such as integrate-and-fire neurons or of more abstract neurons specifically designed for the application at hand. We have used this event-driven framework in these two situations : (1) for the simulation of leaky integrate-and-fire neurons with the aim of contour detection by synchronization [6] and (2) for the simulation of

[1] First in, first out

more abstract neurons specifically designed for detecting an odor independent of its concentration [9]. A simulator has been developped and the software should be soon available at http://www.loria.fr/~rochel/ .

Besides the need of a more in-depth study of the data structures used by the simulation engine, future works will include the design of a hierarchical abstract model that will permit easier modelling of complex networks and more efficient simulations of homogeneous population of neurons.

References

[1] E. T. Claverol, A. D. Brown, and J. E. Chad. Discrete simulation of large aggregates of neurons. *Neurocomputing 47, 277–297*, 2002.

[2] A. Ferscha. *Parallel and Distributed Computing Handbook*, chapter Parallel and Distributed Simulation of Discrete Event Systems. McGraw Hill, 1996.

[3] W. Gerstner and W. Kistler. *Spiking Neuron Models : Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[4] C. Grassmann and J. K. Anlauf. Fast digital simulation of spiking neural networks and neuromorphic integration with spikelab. *International Journal of Neural Systems, Vol9, No. 5 473–478*, 1999.

[5] D. Hansel, G. Mato, C. Meunier, and L. Nelter. On numerical simulations of integrate-and-fire neural networks. *Neural Computation 10,467-483*, 1998.

[6] E. Hugues, F. Guilleux, and O. Rochel. Contour detection by synchronization of integrate-and-fire neurons. *2nd Intl. Workshop on Biologically Motivated Computer Vision (BMCV)*, 2002.

[7] G. Lee and N. H. Farhat. The double queue method: a numerical method for integrate-and-fire neuron networks. *Neural Networks 14 921–932*, 2001.

[8] M. Mattia and P. Del Giudice. Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation 12,2305-2329*, 2000.

[9] O. Rochel, D. Martinez, E. Hugues, and F. Sarry. Stereo-olfaction with a sniffing neuromorphic robot using spiking neurons. *16th European Conference on Solid-State Transducers (Eurosensors)*, 2002.

[10] R. Ronngren and R. Ayani. A comparative study of parallel and sequential priority queue algorithm. *ACM Transactions on Modeling and Computer Simulation, Vol. 7, No. 2, pp. 157–209*, 1997.

[11] L. Watts. Event-driven simulation of networks of spiking neurons. *Proceedings of the Sixth Neural Information Processing Systems Conference, pp. 927-934.*, 1993.