Exploiting activity for the modeling and simulation of dynamics and learning processes in hierarchical (neurocognitive) systems

Alexandre Muzy

Abstract—Although modeling and simulation depend on each other, there is no means to simplify formally models and at the same time corresponding simulations. The activity concept elicits the coordination and the number of computations of a system highlighting salient features about its dynamics. I follow here a neurocognitive example linking and applying definitions and algorithms based on a (neuronal) activity measure. At the modeling level, activity state regions are identified dynamically. At the simulation level, I present how to track the activity region at component level. At learning level, I finally present an activity-based search algorithm able to find the best components (actions) into a network (a series of actions). Activity regions are used hierarchically from neurons to actions.

Index Terms—Activity concept, network coordination, spiking neuronal networks, cognition, modeling, simulation, learning.

1 INTRODUCTION

When considering complex systems, their simulation is intrinsically linked to their modeling. In a feedback loop, increasing modeling efforts reduces the search space of possible simulations and conversely simulation results guide modeling efforts. This feedback design loop is usually informal and, as far as we know, very few efforts have been dedicated to formalize and automate the relationship between both simulation and modeling phases [1]. Here, I present the activity concept, which intends to catch in a formal and operational way the dynamics of computations and how to account for it. It is expected that activity can be shared during both simulation and modeling phases virtually detailing and automating each phase.

In mathematical systems theory [2], the structure of dynamic systems has been set to model any kind of systems (being either continuous, discrete or hybrid). These mathematical structures have been specified to computational systems dealing explicitly with time (continuous or discrete) [3]. To follow this diversity of possible models, several definitions of the activity concept have been proposed [3]. As a metrics, the activity has been defined for continuous systems (being a measure of the average accumulated variations of a continuous trajectory) or for discrete systems (being either a discrete trajectory). Based on these metrics, for each kind of system, the simulation execution times were reduced for various applications (fire spread [4], brain simulation [5], Game of Life [6], etc.). Besides, the understanding of system dynamics was enhanced in different domains: describing the apparent equilibria in forest propagations whereas spatial activity occurs [6] or for attention-based agents [7]. At learning level, activity-based algorithms, able to find efficiently the best system structures (accounting for their activity level), have been proposed [8], [9].

In the forest of activity applications and domains (mathematics, algorithms, applications) we present here a continuum of activity definitions and implementations through an example linking neuronal dynamics to the learning of action sequences. A methodology is presented following this bottom-up example with respect to activity, detailing: (i) mathematical abstract models of neurons (in Section 2), (ii) tracking algorithms and coordination mechanisms of neuronal simulations (in Section 3), and (iii) a learning algorithm for component-based networks and cognitive steps based on neuronal networks (in Section 4).

2 DYNAMIC ABSTRACTION OF SPIKING NEURON MODELS

Analyzing the dynamics of systems through their activity is a rough but unexpectedly powerful abstraction to simplify/guide modeling. This abstraction is presented in Figure 1, where a mathematical model $M_A$ is abstracted into another model $M'_A$. The abstraction has to be achieved in coherency with the state, input and output changes. Notice that changes are indicated in red (being coherent with the notion of discrete events that correspond to possible changes of values). We will see here how one can benefit from applying this kind of abstraction, to identify a first set of mathematical model solutions.

Let us define more precisely how to measure the activity of a segment (a map) $\omega_t : \mathbb{R} \rightarrow \mathbb{Z}$, where $\mathbb{R}$ is the duration of the segment and $\mathbb{Z}$ whatever set (either for the inputs, states or outputs of the system). The activity of $\omega_t$ is a total variation norm (cf. Figure 2):

$$A(\omega_t) = \sum \text{height between local extrema} + \sum \text{height at jump}$$

1. Based on this definition, a high-variation trajectory has a higher activity metrics than a low-variation trajectory.
The activity norm by the duration $t$ of the segment: $A(\omega_i) = \frac{A(\omega_i)}{t}$.

The measure is valid for both continuous and discontinuous trajectories. Discontinuities correspond to discrete events. For neuronal electrical activity (cf. Figure 3), the metrics can be applied either to continuous measurements or to the equivalent spiking dynamics.

A state consists of a vector of variable values $v_i \in V_i$, where $V_i$ is the range of values of variable $v_i$. According to the activity metrics, active and inactive states can then be gathered into a state activity region (cf. Figure 4):

$$AR_Q(t) = \{q \in Q \mid Q = \prod_{i} V_i, \exists v_i, A(\omega_{i,t}) > 0\}$$

Defining activity regions in states allows analyzing dynamically the in/activity state space either for modeling purpose (considering the representation of the system through its activated/inactivated states) or for simulation purpose (tracking activity in real time to reduce execution times). To see how to apply activity regions to spiking neurons let first present a model of spiking neurons.

Figure 4 describes graphically the mathematical model of a spiking Leaky Integrate and Fire neuron model [11]. The activity region consists of all the membrane potential values, $m$, having an activity greater than 0:

$$AR_Q(t) = \{\text{active} \in Q \mid Q = V_m, A(\omega_{m,t}) > 0\}$$

Now that we dispose of a spiking neuron model, we would like to be able to represent more biologically plausible bursting neurons. A bursty dynamics consists of "trains of two or more spikes occurring within a relatively short interval and followed by a [longer] period of inactivity" [12] (cf. Figure 5).

To be able to define such models, let us apply the activity metrics of Equation 1 to spikes. As the spikes have value 1, the following density is obtained:

$$A(\omega_i) = \frac{\text{number of events in the segment}}{t}$$

Previous activity regions can now be aggregated into a higher level activity region where discrete event segments $\omega_{i,t}$ are concatenated into packets $p_{i,t}$:

$$AR_Q(t) = \{q \in Q \mid Q = \prod_{i} V_i, \exists v_i, A(p_{i,t}) > 0, p_{i,t} = \omega_{i,t_1} \cdot \omega_{i,t_2} \cdot \ldots \cdot \omega_{i,t}\}$$

In a previous publication [11], we proved that spiking neuron models can be abstracted into bursty neuron models if input/output packets have an activity (or density) greater than a density threshold $D$. With this condition, applying aggregated activity regions to bursty neurons, it is obtained the following activity region with new high level active and inactive states $A$ and $I$:

$$AR_{Q_{A}}(t) = \{q \in Q \mid Q = V_m, A(p_{m,t}) > 0\}$$
Fig. 4. Spiking abstract model. The input/output values $X_S$ and $Y_S$ of the model include null value $\phi$. In input/output, each piece of the trajectories is a continuous segment, where a spike is represented as a discrete event followed by null values (indicated by different blue and red colors) or simply null values (indicated in gray). The state set $Q_S$ is composed of membrane potential $m \in \mathbb{R}_0^+$ and active and inactive Phases. The membrane potential accumulates input spikes. As indicated in Equation 3, active (resp. inactive) phase corresponds to a membrane potential $m > 0$ (resp. $m = 0$). If the potential is greater than a threshold $\theta$, then the neuron instantaneously fires an output spike otherwise as long as no spike is received the membrane potential decreases exponentially representing the membrane potential leak.

Fig. 5. Burst firing (modified from [12]). From top to down, intracellular calcium injection leading to more and more to packets, each packet having an activity greater than a density threshold $D$. The dynamics of a bursty neuron is described in Figure 7. Notice that now macro in/activity states $A$ and $I$ of bursty neurons correspond to packets and non event input segments while micro in/activity states active and inactive of spiky neurons corresponded to spikes and non event input segments in Figure 4.

Now that we are able to map detailed activity states inactive and active to higher active and inactive states $A$ and $I$, as well as to link these new abstract states to density input-output segments, we obtain a new abstract model as introduced in Figure 1. This new model is described at network level in Figure 7, where neurons exchanging packets are in active states $A$.

3 SIMULATION COORDINATION IN NEURONAL NETWORKS

In previous section, I showed how to set activity regions (for state variables) and how to combine them hierarchically from spikes to bursts. A new higher composition level can now be achieved thanks to component activity regions:

$$\text{AR}_C(t) = \{c \in C \mid q_c \in \text{AR}_Q(t)\} \quad (7)$$

We will see now how to track by simulation these component activity regions as well as how to characterize coordination inside these regions. Coordination is a major issue in brain simulation. Usually, the brain is considered to be a massively parallel computer where each neuron processes in parallel electrical information [13]. However, looking at the top spiking trajectories of Figure 3, spike time occurrences appear to be all aperiodic and changing from neuron to neuron. However, based on actual devices used to measure neuronal activity, it is still not possible to measure precisely the dynamics of a substantial number of neurons. Either it is a very abstract measure over both large space and duration (e.g., as in Magnetic Resonance Imaging (MRI), cf. Figure 13, where it is not possible to distinguish neurons) or tenth of neurons can be measured simultaneously using probes. The best we can do to estimate the total number of simultaneously firing neurons in the brain is to consider the energetical consumption and the maximum firing rate of neurons [14]. Then, the amount of simultaneously firing neurons in the brain is estimated to only $1\%$. This estimation...
thus highly moderates the argument stipulating that the brain is a massively parallel machine.

Beyond the parallel aspect of real neurons, the time coordination of the computational model used should be considered. For example, a modeler can simulate a discrete-time numerical method for discretizing a partial differential equation (as in Human Brain Project [15]), or he can use a usual cellular automaton. Although the parallel aspect of computations does not reflect the actual coordination of biological neurons, the corresponding simulation mechanisms developed can exhibit a large amount of parallel computations.

Figure 8 shows on the left the different possibilities of (neuron) component activity region evolutions according to the computational model chosen. The red curve corresponds to a computational model able to reflect the actual brain activity amount (only 1% of the total number of neurons). On the right, an usual sub-linear parallel simulation is shown. Increasing the number of threads with respect to the number of active neurons obviously leads to a maximum speed-up due to the intrinsic sequential part of the program as well as when using central memory machines, to the increasing waiting times of many threads trying to access the central memory. It is clear that being able to focus on the small amount of active neurons in the brain, few threads are required to obtain interesting speed-up.

Figure 9 shows on the left the different possibilities of (neuron) component activity region evolutions according to the computational model chosen. The red curve corresponds to a computational model able to reflect the actual brain activity amount (only 1% of the total number of neurons). On the right, an usual sub-linear parallel simulation is shown. Increasing the number of threads with respect to the number of active neurons obviously leads to a maximum speed-up due to the intrinsic sequential part of the program as well as when using central memory machines, to the increasing waiting times of many threads trying to access the central memory. It is clear that being able to focus on the small amount of active neurons in the brain, few threads are required to obtain interesting speed-up.

Figure 8. Activity regions wrt. parallelization speed-up. On the left, the number of active components in a simulation is indicated. The top dashed line consists of a fixed activity region where all the components (neurons) remain active during the simulation. The middle gray curve consists of a sub-optimal computational model still focusing on a large amount of active neurons. The bottom red curve consists of an optimal computational model focusing on the actual amount of active neurons in the brain (only around 1%). On the right, an usual sub-linear average speed-up of a simulation parallelizing the computations of the active region with an increasing number of threads is shown (for more information about expected theoretical speed-up in simulation [16]).

Figure 9. Activity regions wrt. parallelization speed-up. On the left, the number of active components in a simulation is indicated. The top dashed line consists of a fixed activity region where all the components (neurons) remain active during the simulation. The middle gray curve consists of a sub-optimal computational model still focusing on a large amount of active neurons. The bottom red curve consists of an optimal computational model focusing on the actual amount of active neurons in the brain (only around 1%). On the right, an usual sub-linear average speed-up of a simulation parallelizing the computations of the active region with an increasing number of threads is shown (for more information about expected theoretical speed-up in simulation [16]).

Figure 9 shows the activity regions of two neurons connected in a feedback loop. If active regions overlap is a prerequisite for determining synchrony, it is not enough. Only during the red part of the dynamics the neurons fire together. Otherwise, their respective firing is asynchronous.

While purely synchronous algorithms are straightforward to implement, purely asynchronous algorithms require sophisticated implementations [17]. Purely asynchronous algorithms can be implemented using a discrete event approach. In latter approach, events drive the simulation. Components in these simulations then require a high autonomy at control level. No control loop is used at network level, rather it is the components that determine themselves their state change according to the reception and sending of external events and their scheduling of internal events. However, implementing discrete event algorithms allows dealing explicitly with time and a/synchronization, automatically focusing computational resources on activity regions.

Figure 10 describes the autonomous coordination of components: Either the components compute autonomously their new state (case (1)), or their new state impacts causally other components (case (2)), or finally interdependent components change simultaneously state leading to the autonomous detection of a synchronization by the component (case (3)).

The activity region of components $AR_C(t)$ (cf. Equation 7) can now be determined based on the state activity region $AR_Q(t) = \{ \text{self-change, send, receive, sync} \}$.
Figure 11 represents the intersynchronization and sequentialization in neuronal simulations. Remembering Figure 8 stipulating that most of the computations are sequential, the precision of time stamps makes natural the rareness of precise synchronous state changes in biological systems, making the discrete event sequential tracking an unexpectedly efficient simulation method from both execution time and memory aspects [18], [19].

Disposing now of a means to determine the activity region of components, Algorithm 1 describes the activity tracking algorithm. Notice that the explicit update of the component activity region makes possible parallelization techniques easy to apply.

**Algorithm 1 Activity tracking**

```plaintext
init. activity region \( AR_C(t = 0) \) // Possible parallelization
repeat
    get senders from \( AR_C(t) \) // Possible parallelization
    route and compute their output events to final receivers
    update \( AR_C(t) \) with receivers // Possible parallelization
    compute transitions of components in \( AR_C(t) \)
    schedule next transitions updating \( AR_C(t_{next}) \)
    \( t \leftarrow t_{next} \)
until \( AR_C(t) = \{\phi\} \) or \( t \geq t_{end} \)
```

**4 LEARNING A SEQUENCE OF ACTIONS**

I presented how to use the activity metrics for modeling and simulation. Activity has been used to combine hierarchically spikes into bursts. I will show now how to use the activity metrics to learn a sequence of actions (cf. Figure 12) [20]. The right sequence of actions to learn is for example “Move hand to object A” → “Catch A” → “Drag A to position Y” → “Rotate A” → ... → “Drop A”. Let us assume that each action component can be right or wrong. Stimuli (discrete events) activate the series of action components unless a wrong action is chosen thus stopping the activation chain (e.g., choosing “Drag A to position X” instead of “Drag A to position Y” in Figure 12). As it is not sending events, a wrong action activated then exhibits less activity than a right action component whereas a non activated component has no activity.

Assuming that each component can be right or wrong in the sequence, the question is then how to develop an algorithm able to automatically select the right action components based on their activity? To answer this question, inspiration from neurosciences can be taken. Figure 13 describes the brain activity results of two motor task achievements (“move your left hand” and “move your right hand”). The experiments aim at correlating the activity of neurons to the task achievements. A direct assumption analogy in simulation-based learning is to consider that local active components during a correct behavior at global network level have more chance to contribute positively to the behavior than other non active components.

To evaluate the behavior at network level, let us define the score as the number of output events in a segment, from right steps, accumulated during a simulation trial, at network level \( N \). The network segment is a vector of component segments defined as \( \omega_{N,t} = (\omega_{1,t}, ..., \omega_{n,t}) \). The score of a series of components at one trial thus consists of:

\[
S_{trial}(\omega_{N,t}) = \text{number of output events in the segment} = \text{number of consecutive right steps}
\]

The *credit* is defined as the correlation between task achievement (the score) and local activity. The Simulated Average Credit (SAC) of a component \( i \) consists of:

\[
SAC(i) = \Sigma_{trial} A_{trial}(\omega_i,t) \times S_{trial}(\omega_{N,t})
\]

The credit value of a component over trials represents the evaluation of the confidence that the component can be selected to produce the expected behavior. Thus, although the credit is attributed locally and automatically it is considered to provide a good evaluation for global behavior achievement. At some trial, the activity-based search algorithm can exploit this information to select presupposedly good components with probability:

\[
P(SAC(i)) = \frac{SAC(i)}{\Sigma_{j=1}^{n} SAC(j)}
\]

Where \( n \) is the number of candidate components evaluated.

This approach was called Activity-based Credit Assignment (ACA) [9] (cf. search algorithm in Algorithm 2) in reference to the credit assignment problem identified by Minsky [23] and Holland [24] and discussed in reinforcement learning [25]. ACA finds the best network structure accounting for component credit information.

Figure 14 presents the trial speed-up and activity reduction for unpredictable simulations at each bias trial number. It appears that there is an optimal value of bias start, \( b^* \), to optimally reduce the number of trials (corresponding to value \( \sigma^* \)), before being too early and after too late. As in reinforcement algorithms, finding the optimal value is not straightforward. Besides, it should be remembered that each trial is an actual simulation where components

4. Notice that action activity is defined here at component level for simplicity reasons but can also be defined hierarchically as the sum of the activities of neurons realizing the action [9].
Fig. 12. Neuronal realization of a series of 10 action components. On the left: The neuronal network [5] presented in the previous section. At the bottom: The dynamics of a bursty neuron (in red) is described assuming that all the neurons in the network emit bursts between times $t$ and $t + T_P$ [11]. At the top: Each network corresponds to an action component (whose decreasing activity is indicated from red, orange, yellow to black (with no activity)) connected in a series of 10 components in a network [9]. On the right: The propagation is shown following an activation chain (firing group picture from [21]) where each “group #” corresponds to a network activity and each line to a different neuron (the spiking activity of the bursty neuron in red on the network on the left is indicated by a red line).

Algorithm 2 Activity-based Credit Assignment (ACA) search: During a first exploration phase, the algorithm randomly builds and simulates a network, and evaluates the credit of components composing the network (for a number of trials less than a bias value). Above the bias value, the algorithm exploits the credit of components for biasing the composition of the network.

```plaintext
repeat
  if trial < bias then
    randomly build a new network N // Exploration
  else
    greedily build a new network N choosing each component $c_i$ with probability $P(SAC(i))$ // Exploitation
  end if
  simulate N and collect component credits
  if N is the best then
    exit from the loop
  end if
  trial ++
until trial = trial end
```

Fig. 13. Magnetic Resonance Imaging (MRI) activity wrt. motor task achievements (from [22]). On the left, several patients have been told to achieve the task: “Move your left hand”. Then, the red zones correspond to the neurons particularly firing during this task achievement and thus contributing to the task. The image on the right is similar for the task: “Move your right hand”.

It has been shown that ACA allows also to reduce the computational efforts while achieve real computations (activity).
finding the solution more quickly [9]. The activity reduction
follows the trial speed-up increase, with an optimal value
\( \alpha^* \) corresponding to optimal bias value \( b^* \).

Fig. 14. Trial speed-up \( \sigma \) (i.e., the ratio between the number of trials
using a random search vs. the number of trials using ACA search) vs.
activity reduction \( \alpha \) (i.e., the ratio between the accumulated activity over the
trials using a random search vs. the accumulated activity over the
trials using ACA search).

5 Conclusion and Perspectives

Based on previous activity successes [4], [5], [6], [8], [9], [11],
I proposed here a line of definitions integrating activity con-
cepts through a single neurocognitive example \( ^5 \). At modeling
level, the activity metrics has been used to aggregate dy-
namically system’s variable changes (membrane potentials
of neurons) into micro inactive and activity states (gathered
into activity regions). The activity-based states have been
then abstracted into macro activity regions in coherency
between input-state-output changes (of bursty neurons). At
simulation level, the definition of dynamic activity regions
has been used to focus computations and elicit coordination
between the components. Local activity a/synchronization
and related parallelization techniques can then be applied
to minimal (synchronized) activity regions. At learning level,
the bursty activity regions obtained at modeling level have
been used for (neuronal) networks abstracted into com-
ponents (representing actions) connected in an activation
chain. Based on the activity of each (action) component
and on the global network behavior evaluation, the ACA
algorithm has been used to find quickly the best network
structure (or series of actions) reducing also the activity
(therefore computational effort).

Concerning the activity concept, the main perspectives
can be discussed with respect to the elements of Figure 12.
First, the relationship between (neuronal) network activity
and (action) activity has to be formally set. Second, although
ACA simulation results have been generally presented for
whatever network structures (a/cyclic, probabilistic, etc.),
taking into account particular structures of the network
(series, parallel, or a composition of both) allows normal
analyses of the results (in terms of predicted activity, al-
gorithm efficiency, etc.) [20]. Third, ACA constitutes a new
means to solve collective structural credit assignment in
networks [25] in relation to the timing constraints from
(neuronal) networks to (action) components (using timed
iterative specifications [3]). Based on the fine-grain coor-
dination described in Section 3, ACA can be used as an
integrative framework for building neuronal networks of
temporally constrained actions comparing the two current
main stream theories of asynchronous spikes [26] (generat-
ing networks with shortest temporal paths corresponding
to action temporal constraints) and synchronous [27] spikes
(generating networks with burst durations (as previously
described) corresponding to action temporal constraints).
Finally, in the field of neuromorphic implementations activity-
based algorithms are faithful candidates to reduce energy
consumption and execution times [28].

Acknowledgments

Many thanks to Jean-Pierre Briot, the “AI expert”, who
impelled me to rephrase activity definitions!

References

in Proceedings of MODELS 2017 Satellite Events, workshop on Model
Driven Engineering, Verification and Validation (MoDeVt), 2017, pp.
315–320.

[2] M. D. Mesarovic and Y. Takahara, General systems theory: mathemat-

ulation: Discrete Event & Iterative System Computational Foundations.

D. R. Hill, “Modelling and simulation of ecological propagation
processes: application to fire spread,” Environmental Modelling &

“Parallel and pseudorandom discrete event system specification
vs. networks of spiking neurons: Formalization and preliminary
implementation results,” in High Performance Computing & Simu-
925–934.

L. Touraille, D. Prunetti, P. Caillou, O. Michel, and D. R. Hill,
“Re-focusing of the activity concept? towards a federative paradigm
for modeling and simulation,” Simulation, vol. 89, no. 2, pp. 156–
177, 2013.

design of emergent behavior in system of systems engineering,”

plasticity in a stochastic environment minimises the cost/benefit

heuristic for simulation-based stochastic search in a hierarchical
model base of systems,” IEEE Systems Journal, vol. 11, no. 4,

and D. R. Hill, “Activity regions for the specification of discrete
event systems,” in Proceedings of the 2010 Spring Simulation Multi-
cconference. Society for Computer Simulation International, 2010,
p. 136.

as a modeling and simulation formalism for f/o general systems,”

[12] A. A. Grace and B. S. Bunney, “The control of firing pattern in
nigral dopamine neurons: burst firing,” Journal of neuroscience,

2012.


Alexandre Muzy is research fellow at CNRS, in charge of the Modélisation, Simulation & Neurocognition (MS&N) group. He is a specialist of discrete-event modeling and simulation and their application to neurocognitive systems.