Modelling and Formal Verification of Neuronal Archetypes Coupling

Elisabetta De Maria[1] Thibaud L'Yvonnet [1]
Alexandre Muzy[1] Daniel Gaffé[2]
Annie Ressouche[3] Franck Grammont[4]

[1]I3S, Sophia Antipolis - [2]LEAT, Sophia Antipolis
[3]INRIA SAM, Sophia Antipolis - [4]LJAD, Nice

Journée de l'équipe MDSC du laboratoire I3S
June 21th, 2017 Nice, France

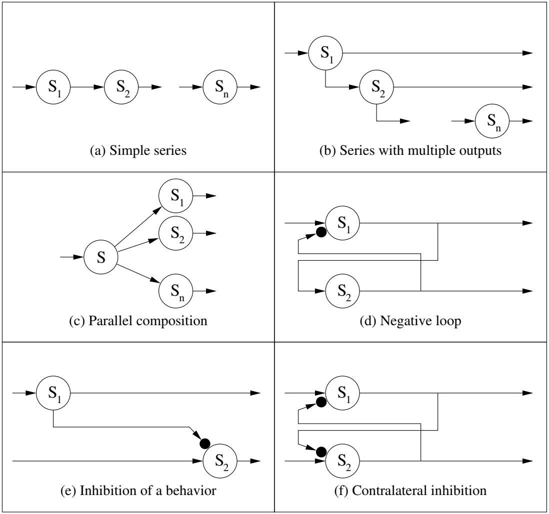
Outline of the presentation

- Neuronal archetypes
- Computational neuronal model
- Synchronous languages and model checking
- Temporal properties of single neurons, archetypes, and their composition in Lustre
- Discussion and future work

Neuronal archetypes

- Neurons tend to form circuits presenting recurrent structures (archetypes)
- Each archetype has a biologically relevant behavior
- Several archetypes can be coupled to constitute the elementary bricks of bigger neuronal circuits (e.g., locomotive motion is controlled by CPGs)
- **Goal of the work**: formally study the behavior of different representative archetypes and their composition

The basic neuronal archetypes



Leaky integrate-and-fire model (1)

- A neuronal network is a weighted directed graph
- **Discrete modeling** (at each instant, a neuron emits a spike if its membrane potential overtakes a given *firing threshold* τ)
- At the instant t, the potential of a given neuron with m inputs can be computed as: $p(t) = r \cdot p(t-1) + \Sigma_{j=1}^{m} x_j(t)$, where $r \in [0, 1]$ is the *remaining potential coefficient*

$$p(t) = \Sigma_{e=0}^{\infty} r^e \Sigma_{j=1}^{m} x_j(t-e)$$

Leaky integrate-and-fire model (2)

Integration time window σ.

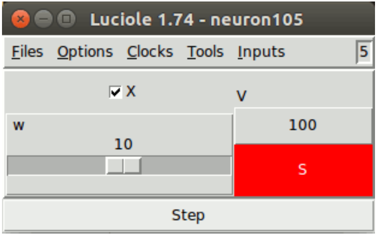
$$\begin{bmatrix} x_1(t) & x_1(t-1) & \cdots & x_1(t-\sigma) \\ x_2(t) & x_2(t-1) & \cdots & x_2(t-\sigma) \\ \vdots & \ddots & \vdots & \vdots \\ x_m(t) & x_m(t-1) & \cdots & x_m(t-\sigma) \end{bmatrix} \begin{bmatrix} 1 \\ r \\ \vdots \\ r^\sigma \end{bmatrix} = \begin{bmatrix} p_1(t) \\ p_2(t) \\ \vdots \\ p_m(t) \end{bmatrix} \tag{1}$$

For each neuron, three important parameters:

- τ: firing threshold
- r: remaining potential
- σ: integration time window

Synchronous languages for reactive systems

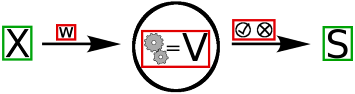
- Spiking networks can be considered as reactive systems
- Synchronous approach based on the notion of a *logical time*
- The synchronous language ***Lustre*** allows to express neuron behaviors easily



Temporal properties and model checking

- Lustre allows to use the technique of *synchronous observers*
- An observer of a property is a program, taking as inputs the inputs/outputs of the program under verification, and deciding at each instant whether the property is violated or not.
- There exists several model checkers for Lustre that are well suited to our purpose: *Lesar*, *Nbac*, *Luke*, *Rantanplan*, and *kind2*.

Encoding neurons in Lustre




```
node neuron1 (X:bool) returns(S:bool);
  var
    V:int;
    threshold:int;
    w:int;
    rvector: int^5;
    mem:int^1*5;
    localS: bool;
  let
    w=10; threshold=105; rvector=[10,5,3,2,1];
    mem[0]=if X then w else 0;
    mem[1..4]=0^4->if pre(S) then 0^4 else pre(mem[0..3]);
    V=mem[0]*rvector[0]+mem[1]*rvector[1]+mem[2]*rvector[2]
        +mem[3]*rvector[3]+mem[4]*rvector[4];
    localS=(V>=threshold);
    S= false -> pre(localS);
  tel
```

Single neuron

- **Delayer:** the sum of the current input signals (multiplied by their weights) is enough to overtake the threshold
- $1/x$ **Filter:** more than one time unit is needed to overtake the threshold

Delayer or filter

Given a neuron receiving an input stream on the alphabet $\{0, 1\}$, it can only express one of the two following behaviors:

- *Delayer effect*. It emits a 0 followed by a stream identical to the input one.
- *Filter effect*. It emits at least two 0 at the beginning and can never emit two consecutive 1.

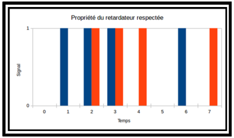
Limit case of the filter effect : *wall* effect

Property of a single neuron

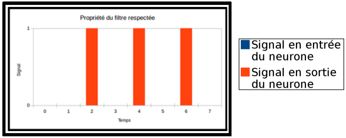
```
node retardateur (X: bool; w: int) returns(OK: bool);
  var
    Out: bool;
    SX: bool;
    S1: bool;
  let
    Out = neuron100(X, w);
    S1 = true->Out;
    OK = S1 or pre(X)=false;
  tel
```

```
node filtre (X: bool; w: int) returns(OK: bool);
  var
    Out: bool;
  let
    Out = neuron100(X, w);
    OK = true -> if Out then not pre(Out) else not Out;
  tel
```

```
node prop1 (X: bool; w: int) returns (OK: bool);
  var
    S1: bool;
    S2: bool;
  let
    S1 = retardateur(X, w);
    S2 = filtre(X, w);
    OK = if S1 xor S2 then true else false;
  tel
```



$\oplus$



$=$

Propriété 1

Simple series of length n

n-delayer or n-delayer/filter

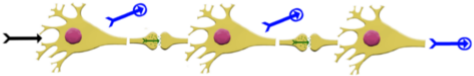
Given a series of length n receiving an input stream on the alphabet $\{0, 1\}$, it can only express one of the two following behaviors:

n-delayer effect. It emits a sequence of 0 of length n followed by a stream identical to the input one.

n-delayer/filter effect. It emits a sequence of 0 of length at least $n+1$ and can never emit two consecutive 1.

A simple series is not able to reconstitute a permanent signal.
Filter neurons do *not commute* in a simple series.

Series with Multiple Outputs



Exclusive temporal activation in a series with multiples outputs

When a series of n delayers with multiples outputs receives the output of a $1/n$ filter, only one neuron at a time overtakes its threshold (and thus emits).

Parallel Composition

Lower/upper firing bounds in a parallel composition

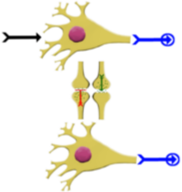
Given a parallel composition of neurons, at each time unit the number of emitted spikes is in between a given interval.

Parallel composition of n filters, sequence of 1 as input

Given a parallel composition with a delayer connected to n filters of different selectivity connected to a delayer, it is possible to emit as output a sequence of 1 of length k, with $k \geq n$.

A parallel composition is able to reconstitute a permanent signal from a not permanent one.

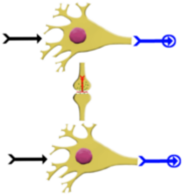
Negative Loop



Oscillation in a negative loop

Given a negative loop composed of two delayers, when a sequence of 1 is given as input, the inhibited neuron oscillates with a pattern of the form 1100 (and the inhibitor expresses the same behavior delayed of one time unit).

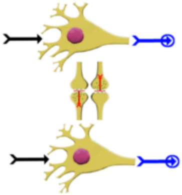
Inhibition of a Behavior



Fixed point inhibition

Given an inhibition archetype, if a sequence of 1 is given as input, at a certain time the inhibited neuron can only emit 0 values.

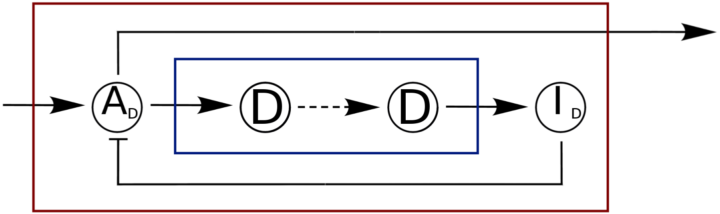
Contralateral Inhibition



Winner takes all in a contralateral inhibition

Given a contralateral inhibition archetype with two neurons (where the two neurons do not necessarily have the same parameters), if a sequence of 1 is given as input, at a given time one neuron is activated and the other one is inhibited.

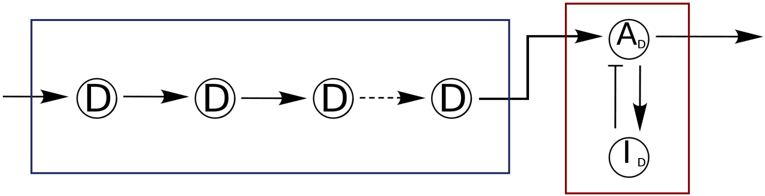
Simple series within a negative loop



Oscillation period extension

Given a simple series of length n within a negative loop, if a sequence of 1 is given as input, the output of the activator is of the form : $(1^{n+2}0^{n+2})^{\omega}$.

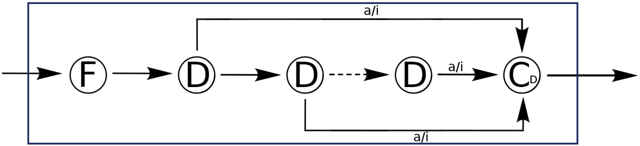
Simple series followed by a negative loop

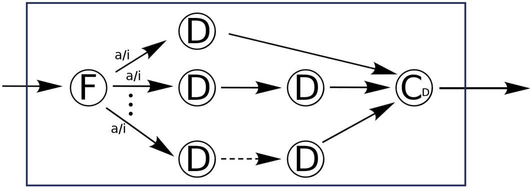


Oscillation delay

Given a simple series of delayers of length n connected to the activator of a negative loop, if a sequence of 1 is given as input, the output of the activator is of the form : $0^n(1100)^\omega$.

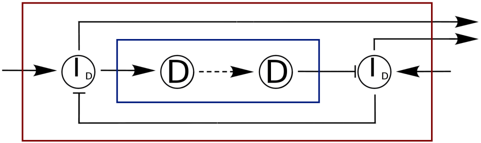
Generators of periodic patterns

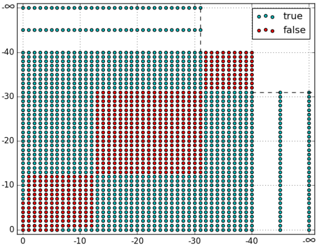


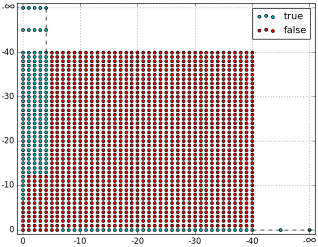


Example: For $n = 5$, the pattern 11001 generates oscillations of the form 11000 as output of the negative loop.

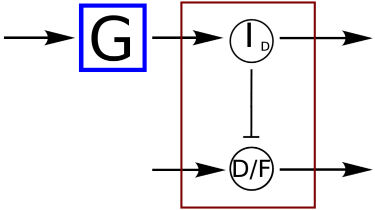
Series within a contralateral inhibition

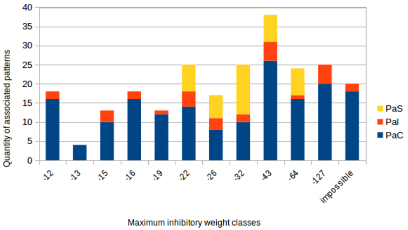






Pattern generator followed by inhibition





Example: The patterns 0110100 and 0010110 are in two different (neighbor) classes

Discussion and future work

- Study more sophisticated compositions
- Integrate LDDs to model-checkers in order to automatically infer parameters
- Translate our Lustre code into VHDL one to make it run on FPGAs
- Express all neural networks as archetype compositions