

Mini-circuits and LI&F neuron

Model Checking

Fissore Davide

Université Côte d'Azur

Jan. 06, 2023



MASTER
INFORMATIQUE

Objective

- 1 *Choisir deux parmi les mini-circuits proposés dans la figure 1 de l'article FCS.pdf, les implémenter en PRISM et tester des propriétés de logique temporelle concernant ces mini-circuits.*

Objective

- 1 Choisir deux parmi les mini-circuits proposés dans la figure 1 de l'article *FCS.pdf*, les implémenter en PRISM et tester des propriétés de logique temporelle concernant ces mini-circuits.
- 2 Implémenter avec le model checker probabiliste PRISM un neurone biologique de type LI&F.

The LI&F neuron

Definition (LI&F: Leaky Integrate and Fire Model)

A neuronal network represented by a digraph.

- Nodes represent the neurons
- Edges (the synaptic connections) have either positive (activators) or negative (inhibitors) weights
- Nodes contain a membrane potential: if its threshold is overcome, then a spike is emitted
- The leak factor reduces at each time unit the neuron potential

Some existing archetypes

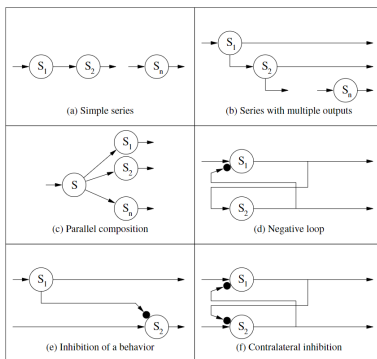


Fig. 1 The basic neuronal archetypes.

¹Img. from *On the Use of Formal Methods to Model and Verify Neuronal Archetypes*

Some existing archetypes

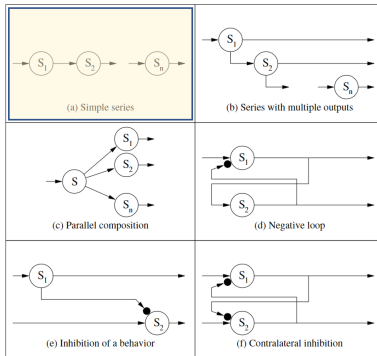


Fig. 1 The basic neuronal archetypes.

¹Img. from *On the Use of Formal Methods to Model and Verify Neuronal Archetypes*

Some existing archetypes

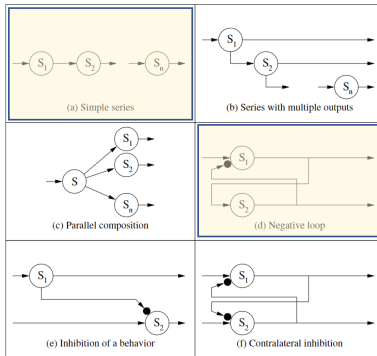


Fig. 1 The basic neuronal archetypes.

¹Img. from *On the Use of Formal Methods to Model and Verify Neuronal Archetypes*

The simple serie archetype

Definition (Simple serie archetype)

A series of n neurons N_1, \dots, N_n , where N_i receives the signal from N_{i-1} and emits its signal to N_{i+1} .

There exists 2 main implementations of this archetype:

- the n -delayer
- the n -delayer/filter

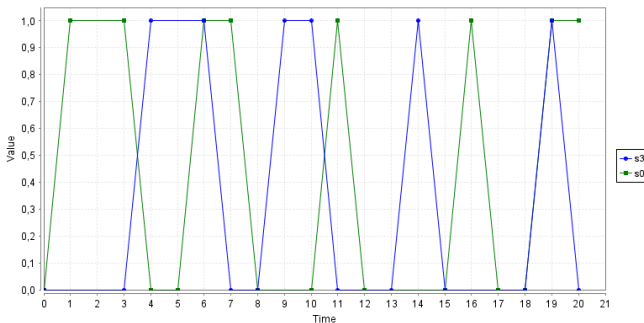
The n -delayer

Given a signal \mathcal{S} , \mathcal{S} is transmitted with a delay of n time units.

Implementation:

```
1 dtmc
2
3 module S3 = S1 [s1=s3, s0=s2] endmodule
4
5 module S2 = S1 [s1=s2, s0=s1] endmodule
6
7 module S1
8     s1 : [0..1] init 0;
9     [s] true -> (s1'=s0);
10 endmodule
11
12 module initialisation
13     s0 : [0..1] init 0;
14     [s] true -> 0.5:(s0'=1) + 0.5:(s0'=0);
15     //[s] true -> (s0'=1-s0); // alternating sequence
16 endmodule
```

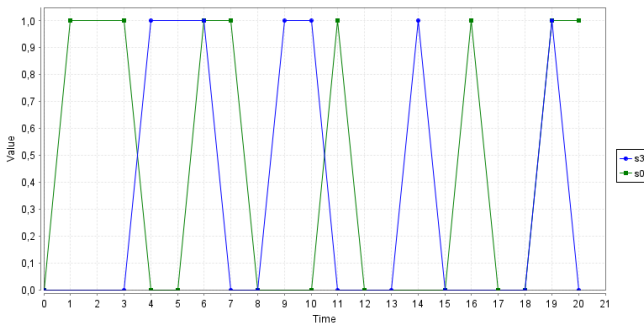
A plot of a n -delayer



An interesting property:

$$\bullet P = [G(s_0 = 1 \Leftrightarrow X^3(s_3 = 1))]?$$

A plot of a n -delayer



An interesting property:

$$\bullet P = [G(s_0 = 1 \Leftrightarrow X^3(s_3 = 1))]? 1.0$$

The n -delayer/filter

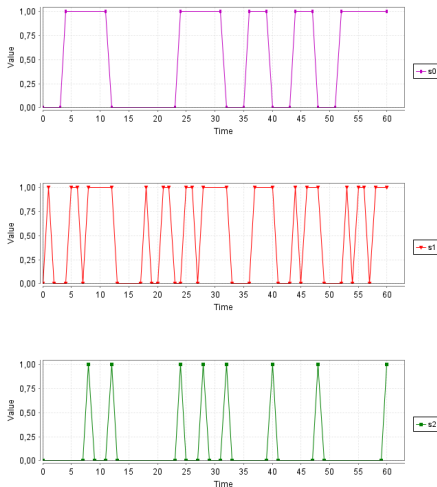
Let's introduce the leak factor and the potential membrane:
An example of **code corrector**.

```

8  const double noise = 0.2;
9  const int rep = 3; // nb of digit repetition
10
11 module S2
12     pot2: [0..10] init 0;
13     s2 : [0..1];
14     [s] s2=0 & cnt < rep -> (pot2'=-ceil(pot2*0.9+s1)) & (s2'=0);
15     [s] s2=0 & cnt = rep & pot2 > 1.5 -> (s2'=1) & (pot2'=0);
16     [s] s2=0 & cnt = rep & pot2 <= 1.5 -> (pot2'=0);
17     [s] s2=1 -> (s2'=0) & (pot2'=0);
18 endmodule
19
20
21 // this neuron applies a random bit flip (with proba equal to the const noise)
22 // on each received signal (the noised channel)
23 module S1
24     s1 : [0..1] init 0;
25     [s] true -> noise: (s1'=1-s0) + [1-noise]: (s1'=s0);
26 endmodule
27
28 // send the original message with the repetition of each bit 3 times
29 module initialisation
30     cnt: [0..rep] init 0;
31     s0 : [0..1] init 0;
32     [s] cnt < rep -> (cnt'=cnt+1);
33     [s] cnt = rep -> 0.5: (s0'=0) & [cnt'=0] + 0.5: (s0'=1) & (cnt'=0);
34     // [s] cnt = rep -> (s0'=1-s0) & (cnt'=0);
35 endmodule

```

A plot of a n -delayer/filter



Test certainty of model by noise

$$P \stackrel{?}{=} (X^4((s_0 = 1) \Leftrightarrow X^4(s_2 = 1)))$$

Test certainty of model by noise

$$P \stackrel{?}{=} (X^4((s_0 = 1) \Leftrightarrow X^4(s_2 = 1)))$$

noise	proba
0.00	1.00

Test certainty of model by noise

$$P \stackrel{?}{=} (X^4((s_0 = 1) \Leftrightarrow X^4(s_2 = 1)))$$

noise	proba
0.00	1.00
0.10	0.90

Test certainty of model by noise

$$P \stackrel{?}{=} (X^4((s_0 = 1) \Leftrightarrow X^4(s_2 = 1)))$$

noise	proba
0.00	1.00
0.10	0.90
0.50	0.50

Test certainty of model by noise

$$P \stackrel{?}{=} (X^4((s_0 = 1) \Leftrightarrow X^4(s_2 = 1)))$$

noise	proba
0.00	1.00
0.10	0.90
0.50	0.50
0.90	0.10

Test certainty of model by noise

$$P \stackrel{?}{=} (X^4((s_0 = 1) \Leftrightarrow X^4(s_2 = 1)))$$

noise	proba
0.00	1.00
0.10	0.90
0.50	0.50
0.90	0.10
1.00	0.00

The negative loop archetype

Definition (Negative loop archetype)

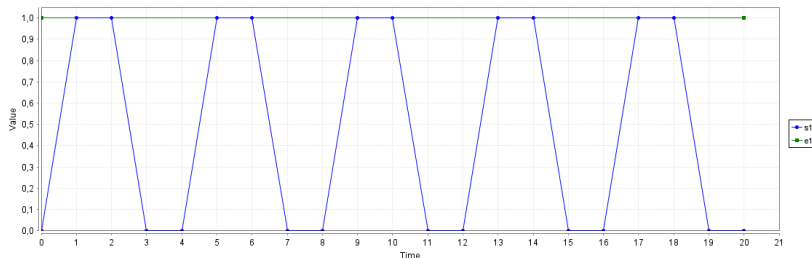
Two neurons where the first receives the input signal and an inhibition from the second and the second is activated by the first.

```
1 dtmc
2
3 const int wS1 = 1;
4 const int wS2 = -1;
5
6 module S2
7   s2 : [0..1] init 0;
8   [s] true -> (s2' = wS1 * s1);
9 endmodule
10
11 module S1
12   s1 : [0..1] init 0;
13   [s] true -> (s1' = max(0, e1 + wS2 * s2));
14 endmodule
15
16 module E1
17   e1 : [0..1] init 1;
18   //[s] true -> 0.5:(e1'=1) + 0.5:(e1'=0);
19   [s] true -> (e1'=1);
20 endmodule
```

Remark

Note the max function → the signal of S1 can only be 0 or 1.

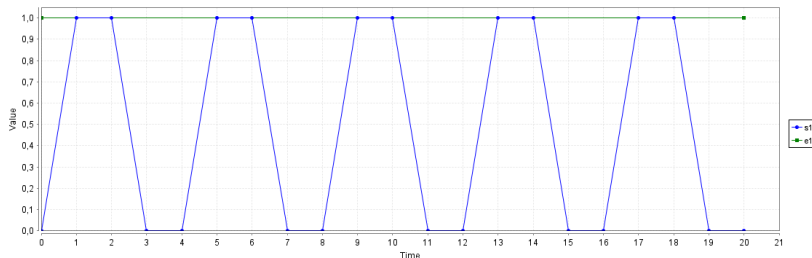
A plot of negative loop



Some properties:

- $P = [G(e_1 = 1)]?$
- $P = [G((s_2 = 1 \wedge (Xs_2 = 1)) \Rightarrow (XXs_2 = 0))]?$
- $P = [G((s_2 = 0 \wedge (Xs_2 = 0)) \Rightarrow (XXs_2 = 1))]?$

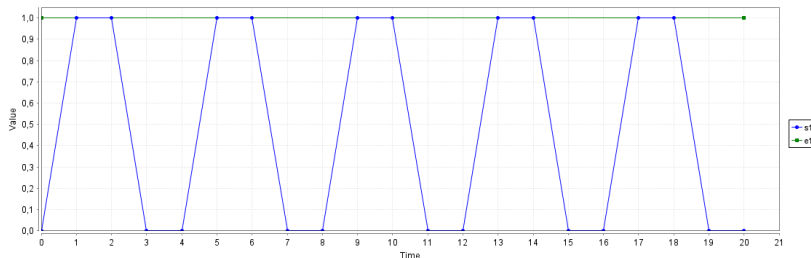
A plot of negative loop



Some properties:

- $P = [G(e_1 = 1)]? 1.0$
- $P = [G((s_2 = 1 \wedge (Xs_2 = 1)) \Rightarrow (XXs_2 = 0))]?$
- $P = [G((s_2 = 0 \wedge (Xs_2 = 0)) \Rightarrow (XXs_2 = 1))]?$

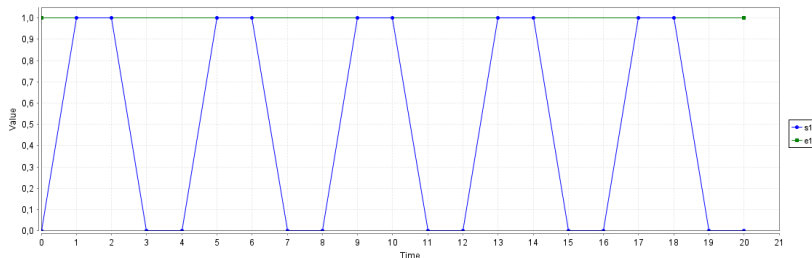
A plot of negative loop



Some properties:

- $P = [G(e_1 = 1)]? 1.0$
- $P = [G((s_2 = 1 \wedge (Xs_2 = 1)) \Rightarrow (XXs_2 = 0))]? 1.0$
- $P = [G((s_2 = 0 \wedge (Xs_2 = 0)) \Rightarrow (XXs_2 = 1))]?$

A plot of negative loop



Some properties:

- $P = [G(e_1 = 1)]? 1.0$
- $P = [G((s_2 = 1 \wedge (Xs_2 = 1)) \Rightarrow (XXs_2 = 0))]? 1.0$
- $P = [G((s_2 = 0 \wedge (Xs_2 = 0)) \Rightarrow (XXs_2 = 1))]? 1.0$

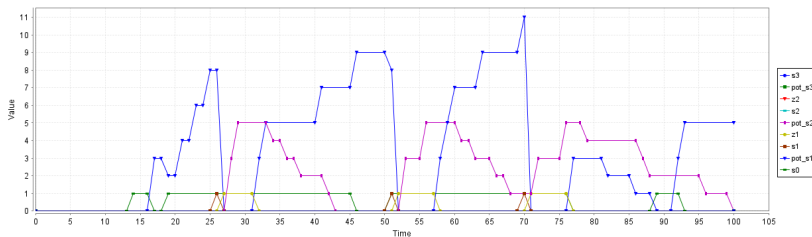
A complete example

```

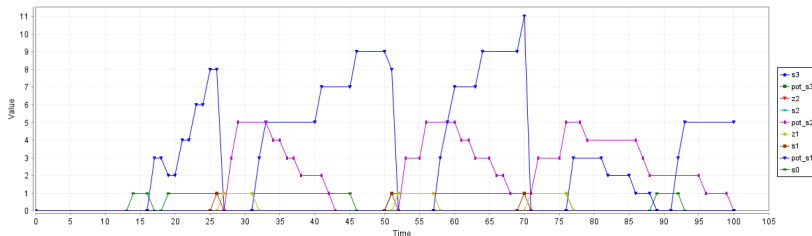
10 ..
11 module S3 = S1 [s1=s3, pot_s1=pot_s3, s0=z2, threshold_s1-threshold_s3, tol1-to3, reset1-reset3] endmodule
12
13 module transf2_3 = transf1_2 [z1=z2, tol1=tol2, reset1-reset2] endmodule
14
15 module S2 = S1 [s1=s2, pot_s1=pot_s2, s0=z1, threshold_s1-threshold_s2, tol1=tol2, reset1-reset2] endmodule
16
17 module transf1_2
18     z1: [0..1] init 0;
19     [tol] true -> (z1'=0);
20     [reset1] true -> (z1'=1);
21 endmodule
22
23 module S1
24
25     s1: [0..1] init 0; // 0 = inactif et 1 = spike
26     pot_s1: [0..m] init 0; // potential of the neuron
27
28     [tol] s1=0 & pot_s1 <= threshold_s1/2 ->
29         (pot_s1'=floor(pot_s1 + fuite + s0 * power));
30     [tol] s1=0 & pot_s1 > threshold_s1/2 & pot_s1 <= threshold_s1 ->
31         0.2:(s1'=1) + 0.8:(s1'=0) & (pot_s1'=floor(pot_s1 + fuite + s0 * power));
32     [tol] s1=0 & pot_s1 > threshold_s1 ->
33         (s1'=1) & (pot_s1'=floor(pot_s1 + fuite + s0 * power));
34     [reset1] s1=1 -> (s1' = 0) & (pot_s1' = 0);
35
36 endmodule
37
38 module initialisation
39     s0: [0..1] init 0;
40     [tol] s0=0 -> 1:(s0' = 1);
41     [tol] s0=1 -> 1:(s0' = 0);
42 endmodule

```

A corresponding plot



A corresponding plot



Manual exploration

Module/[action]	Probability	Update
▶ [to3]	0.3333333333333333	pot_s3'=0
[reset2]	0.3333333333333333	z2'=1, s2'=0, pot_s2'=0
[to1]	0.1666666666666666	z1'=0, pot_s1'=0, s0'=1
[to1]	0.1666666666666666	z1'=0, pot_s1'=0, s0'=0

☒ Generate time automatically

Conclusion

Programming strategy:

- Implement simple and deterministic models
 - Introduce probabilities
 - Add the leak factor along with the potential membrane
-

Conclusion

Programming strategy:

- Implement simple and deterministic models
 - Introduce probabilities
 - Add the leak factor along with the potential membrane
-

About the project:

- Understand the logic behind neuron interactions
- Implement neurons in PRISM
- Test concretely how do they work

Thanks