Modélisation et Vérification Formelle de la Composition d'Archétypes

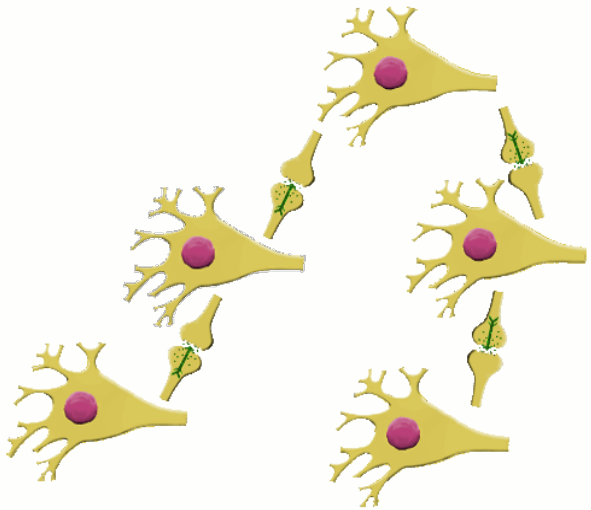
Étudiant : Thibaud L'Yvonnet
Tutrice : Elisabetta De Maria

19 Juin 2017

Archétype

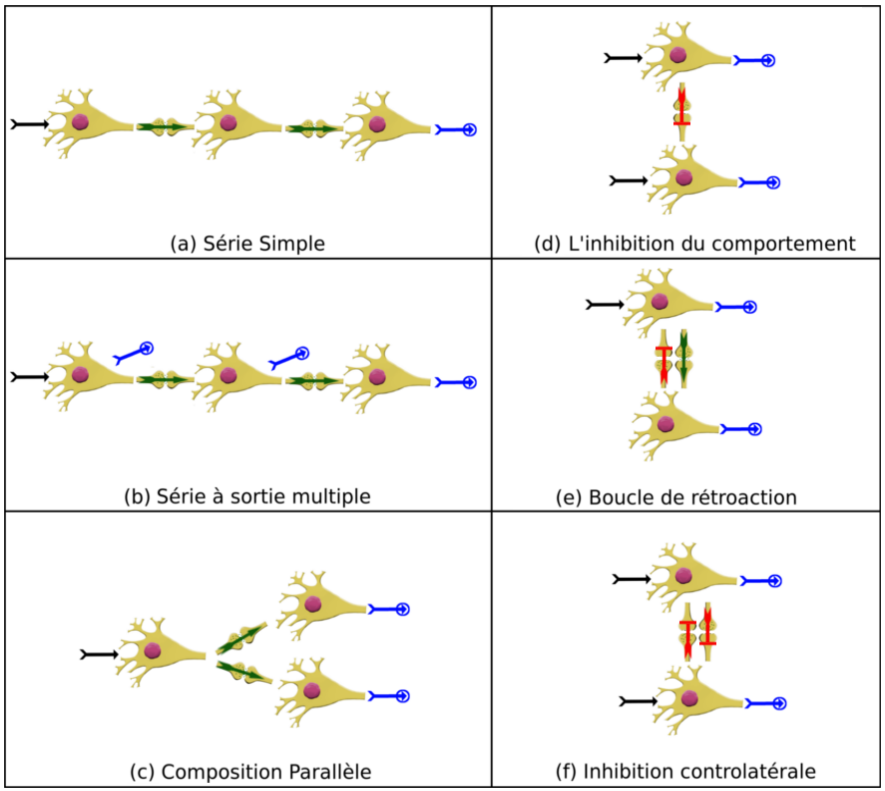
Les Archétypes

Petits réseaux avec une structure et un comportement biologiquement pertinent.



Les Six Archétypes de Base

- Déjà caractérisés au sein du projet MS&N (I3S) :



- **But** : Étudier le comportement de la composition des six archétypes de base.

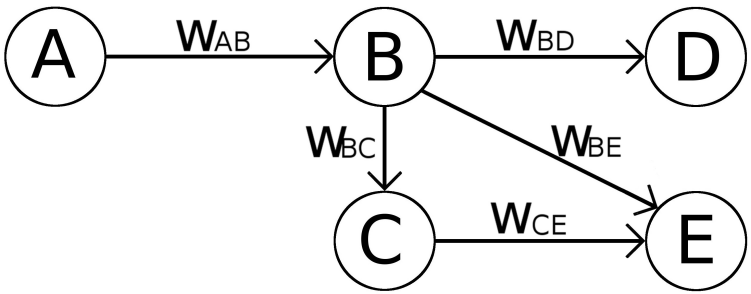
[E. De Maria et al. Verification of temporal properties of neuronal archetypes modeled as synchronous reactive systems. 2016.]

Leaky Integrate and Fire (LIF)

Calcul du potentiel de membrane:

- Intégration des signaux.
- Fuite.
- Fenêtre temporelle.
- Seuil d'activation.

Réseau de neurones = Graphe orienté et pondéré.

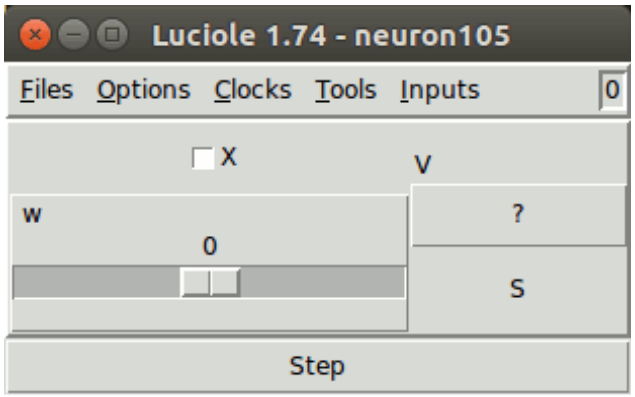


[H. Paugam-Moisy and S. Bohte. Computing with Spiking Neuron Networks, 2012.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.



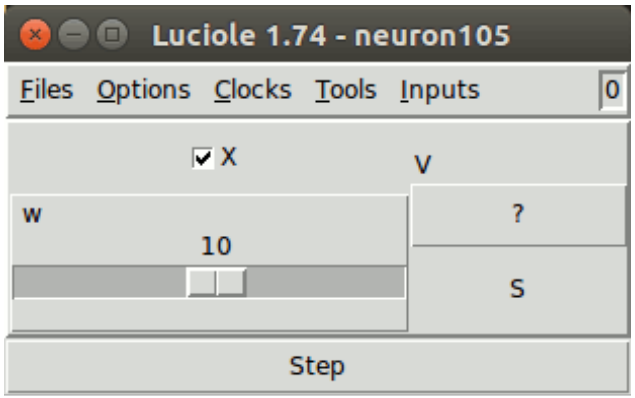

```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

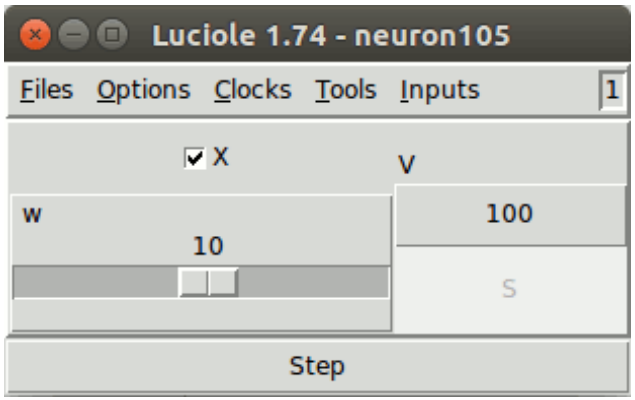
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

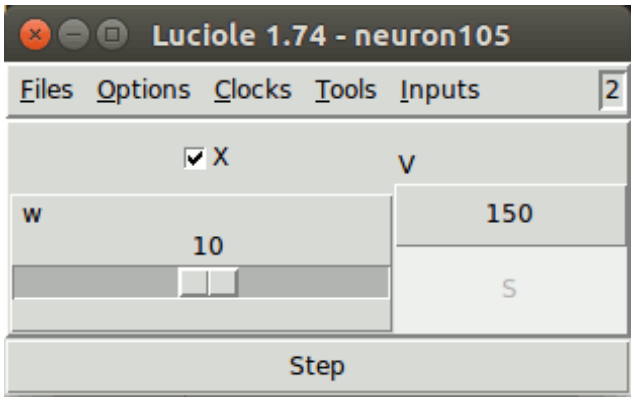
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

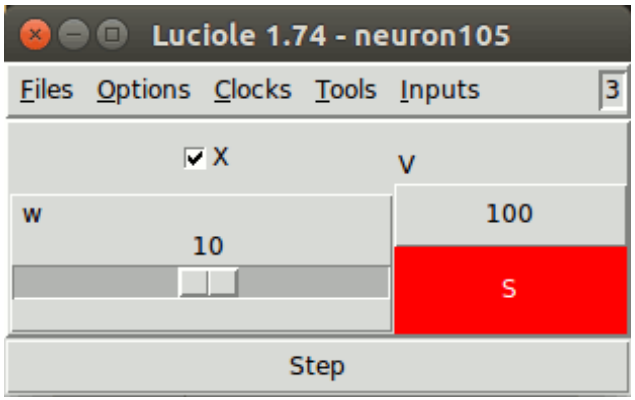
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

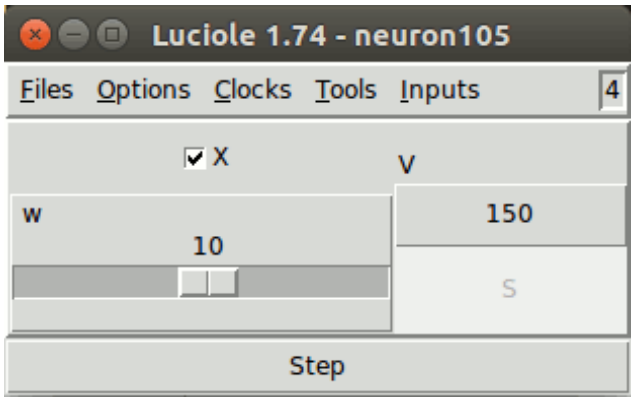
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

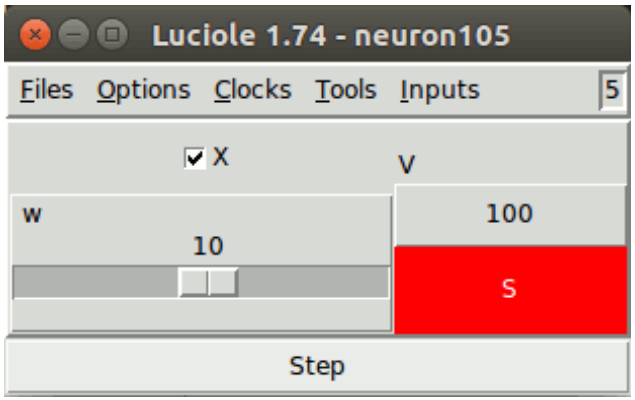
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

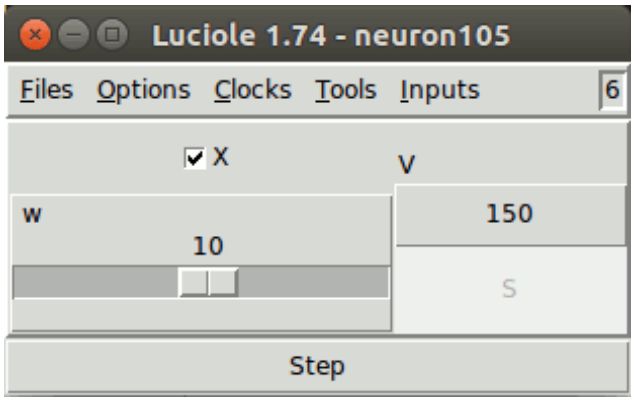
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

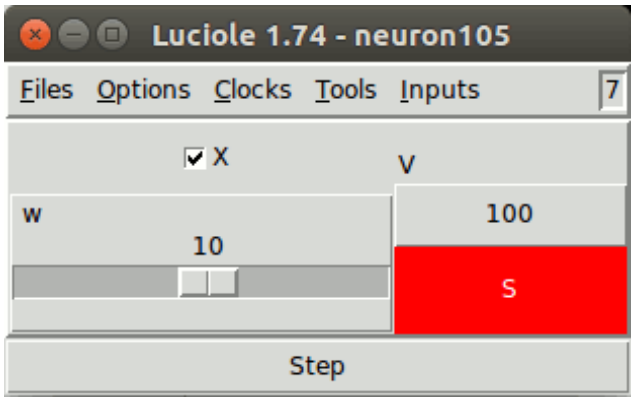
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

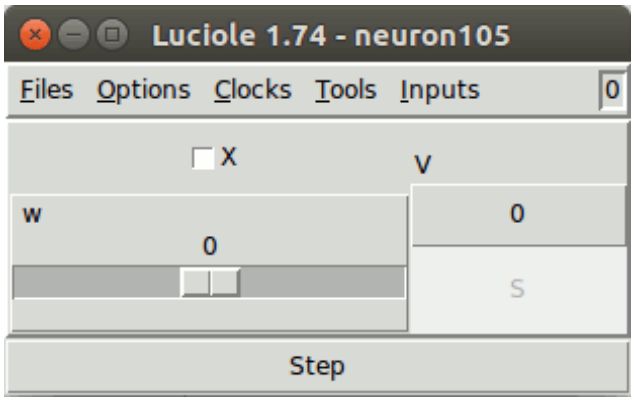
Filtre

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.



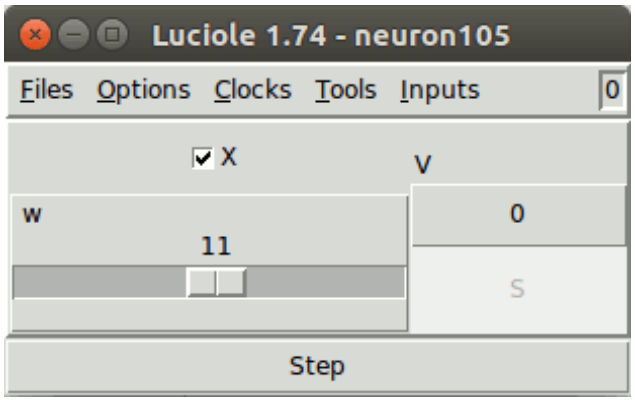

```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

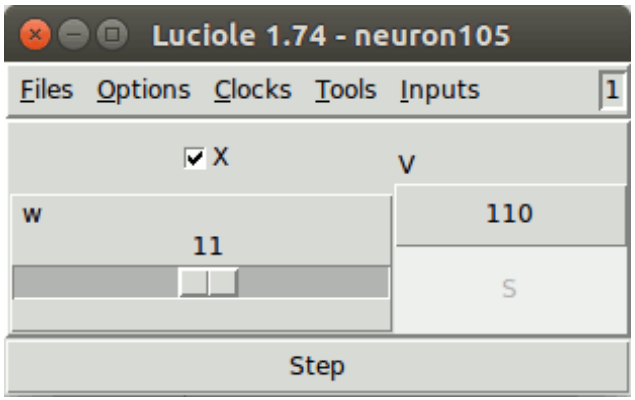
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

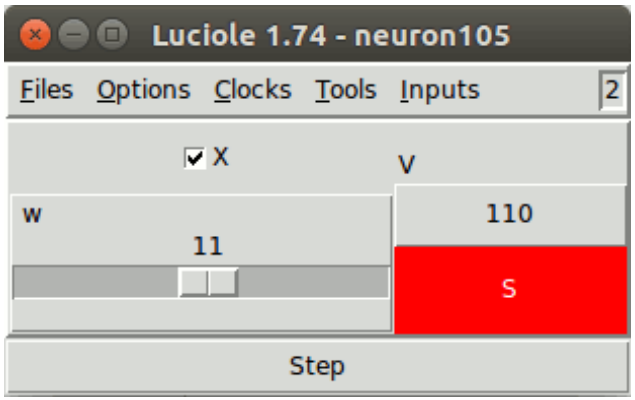
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

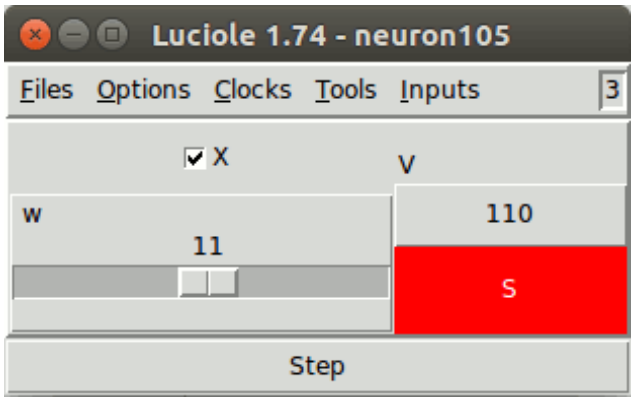
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

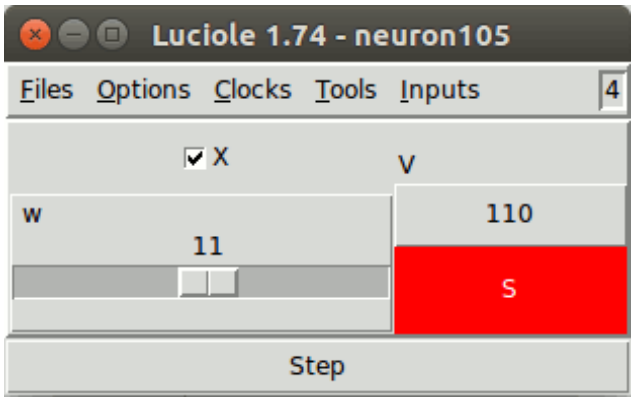
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

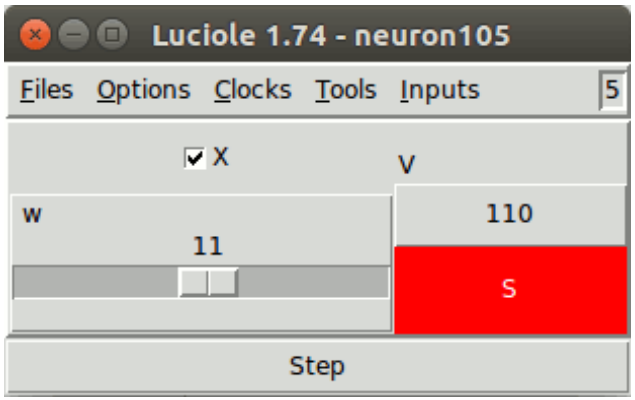
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

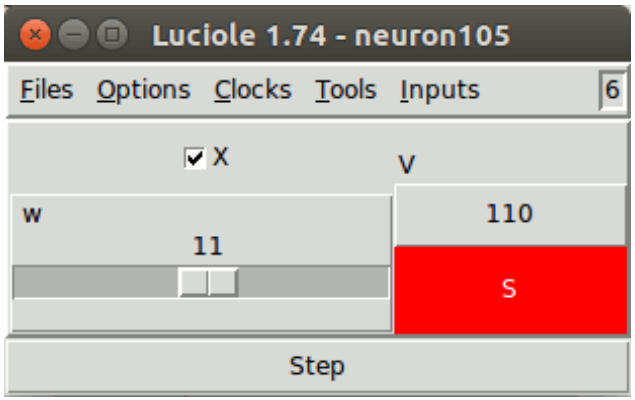
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.




```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

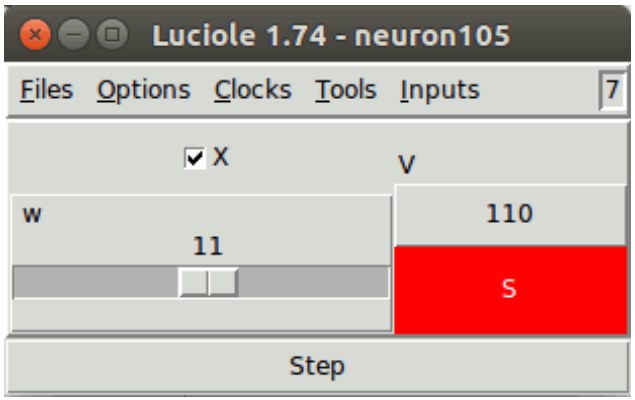
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

Outils Informatiques (1)

Lustre et Luciole.

- Programmation synchrone.
  - Implémentation d'un neurone et d'un réseau.
- Les Observateurs.
  - Un moyen d'implémenter un comportement.



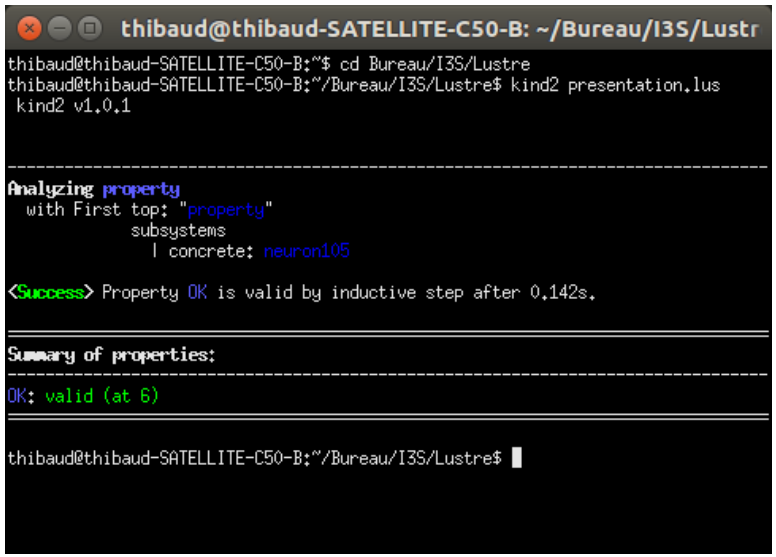

```
36  node property (X:bool; w:int) returns(OK:bool);
37  var Out:bool;
38      V:int;
39  let
40    assert(w<=11);
41    V,Out=neuron105(X,w);
42    OK=true -> (pre(X)=Out); --verification
43    --%MAIN ;
44
45    --%PROPERTY OK;
46
47  tel
```

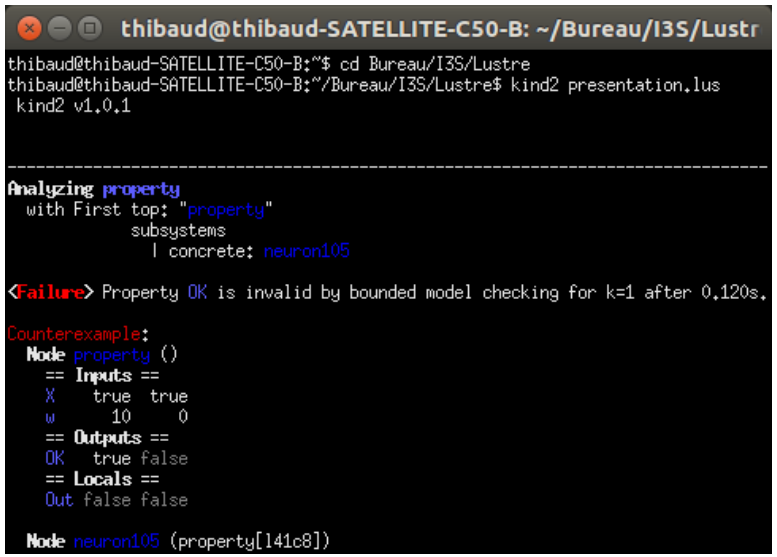
Retardateur

[N. Halbwachs. Synchronous programming of reactive systems. 1998.]

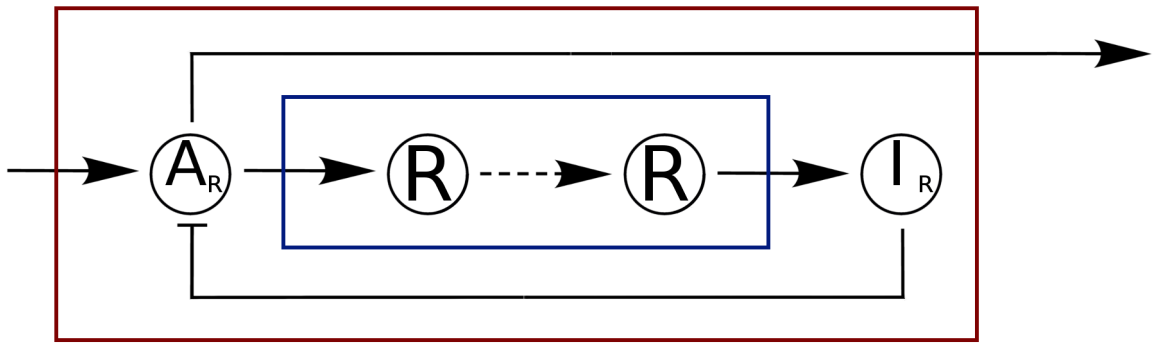
Outils Informatiques (2)

Model Checker : kind2.

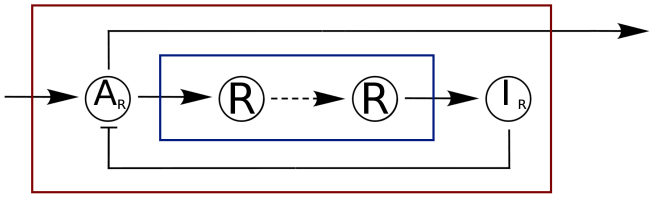




Série imbriquée dans la boucle de rétroaction (1)



Série imbriquée dans la boucle de rétroaction (2)



Comportement dynamique

Figure: Signal en sortie de l'archétype simple

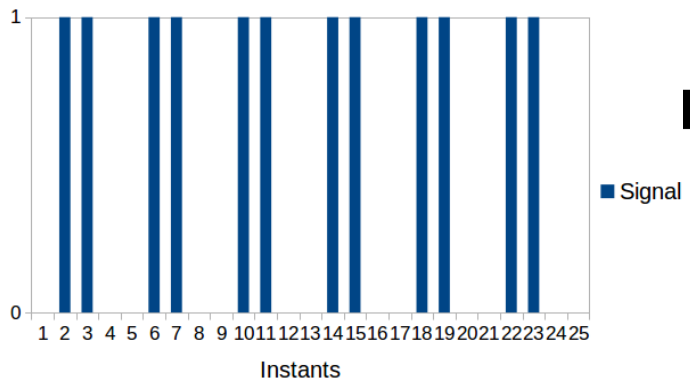
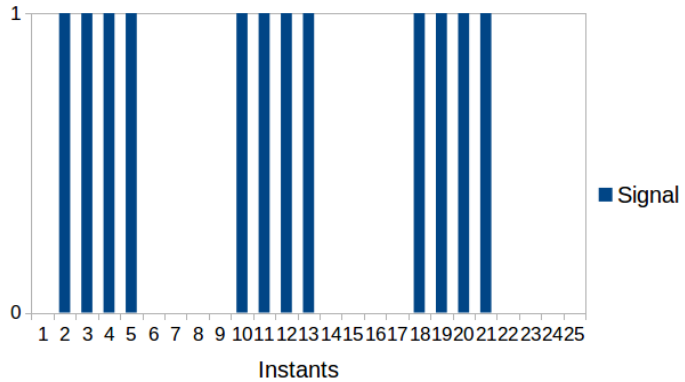
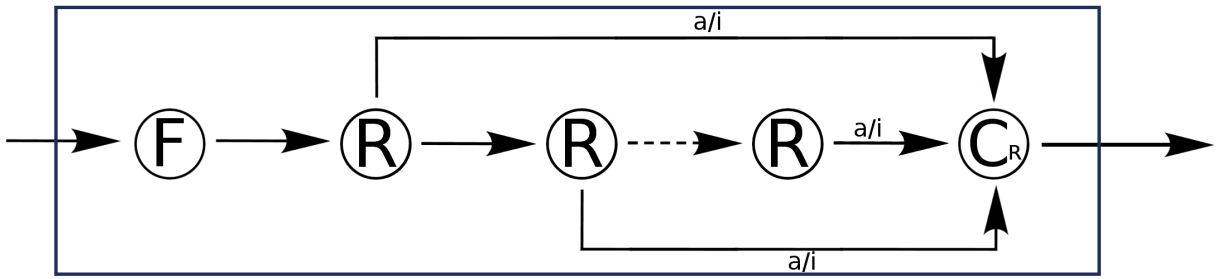


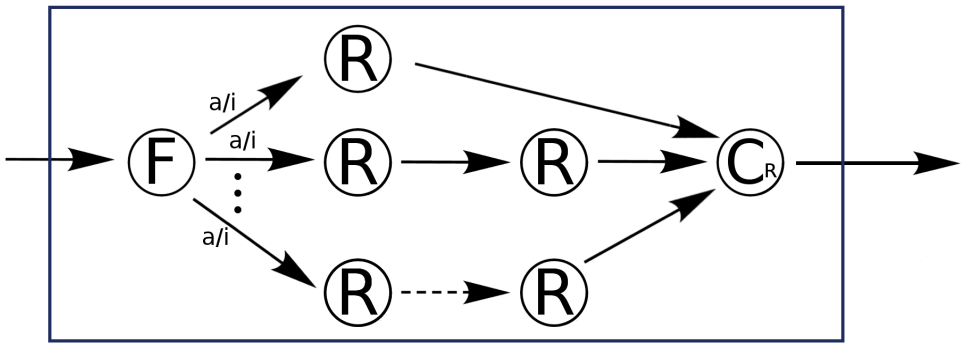
Figure: Signal en sortie de la composition



- La période d'oscillation passe de 2 à $n + 2$.

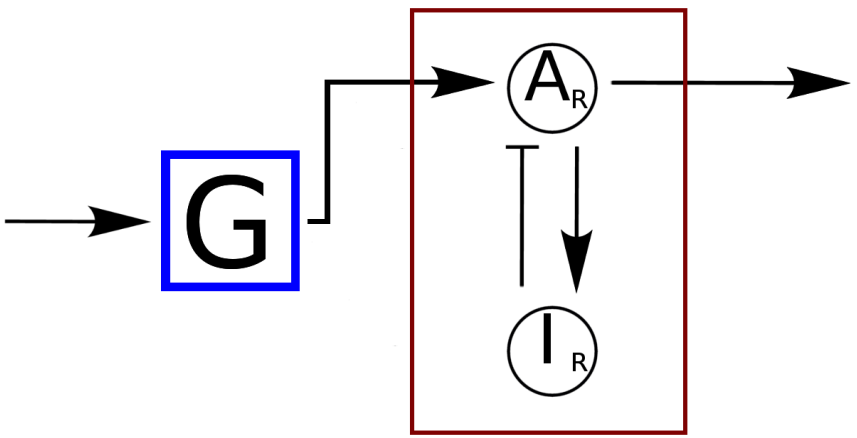
Générateurs de motifs périodiques



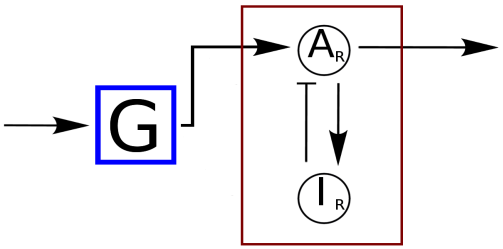


- On génère tous les motifs de longueur n sur $\{0, 1\}$.

Générateur suivi de la boucle de rétroaction (1)



Générateur suivi de la boucle de rétroaction (2)



Comportement dynamique

Figure: Motif entrant dans la boucle (011001)

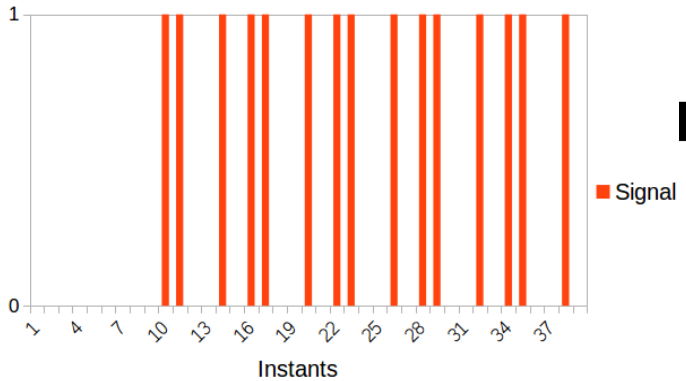
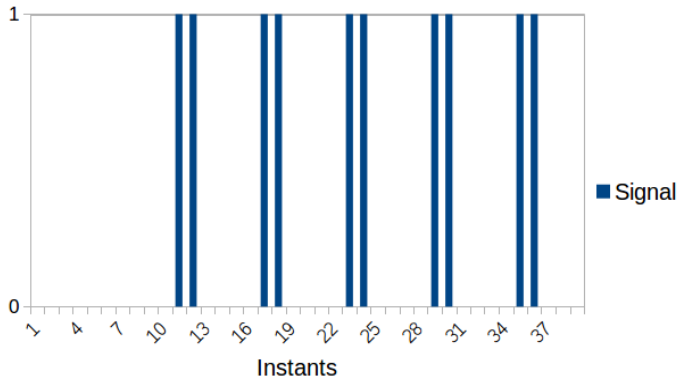
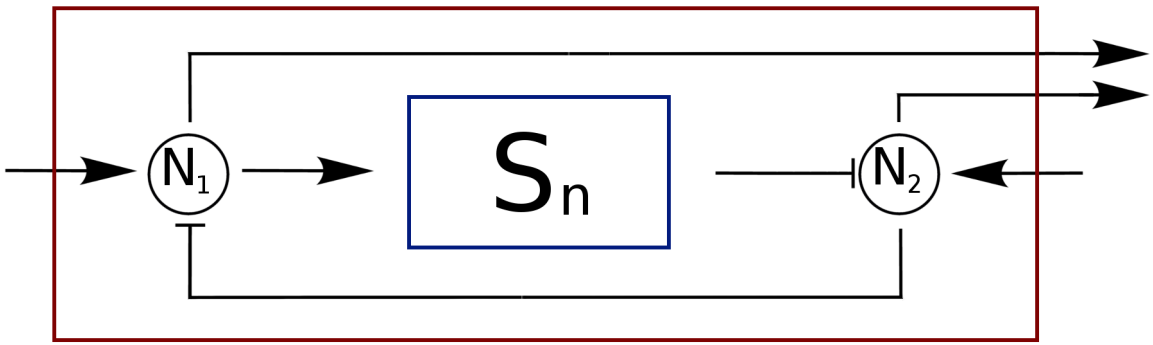


Figure: Signal en sortie de la composition(110000)

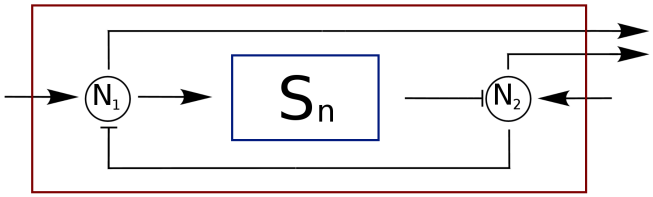


- Identification de tous les motifs capables de générer une oscillation.

Série imbriquée dans l'inhibition controlatérale (1)



Série imbriquée dans l'inhibition controlatérale (2)



Comportement dynamique

Figure: Vérification du "Winner takes all" (archétype simple)

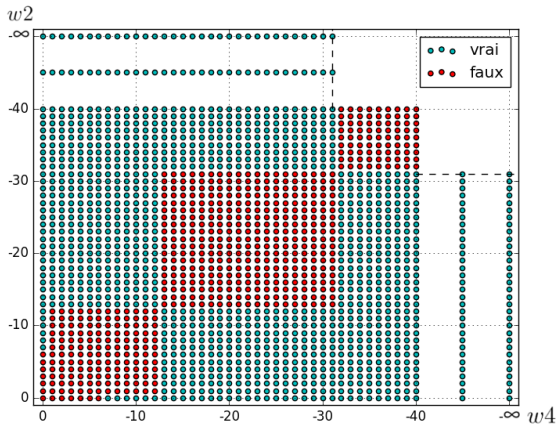
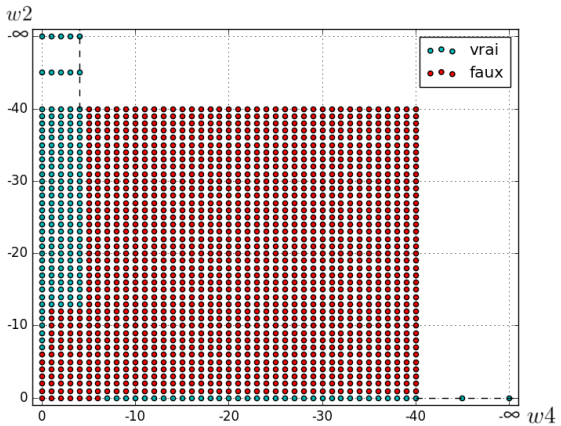
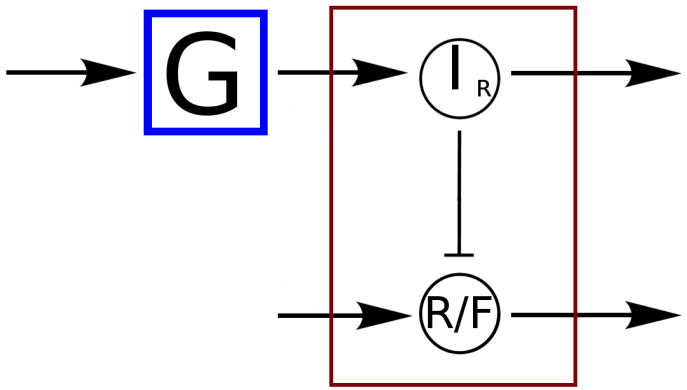


Figure: Vérification du "Winner takes all" (composition)

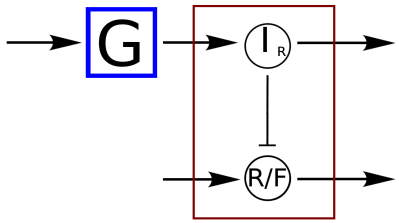


- Agrandissement de la zone rouge et apparition d'une asymétrie.

Générateur suivi de l'inhibition de comportement (1)

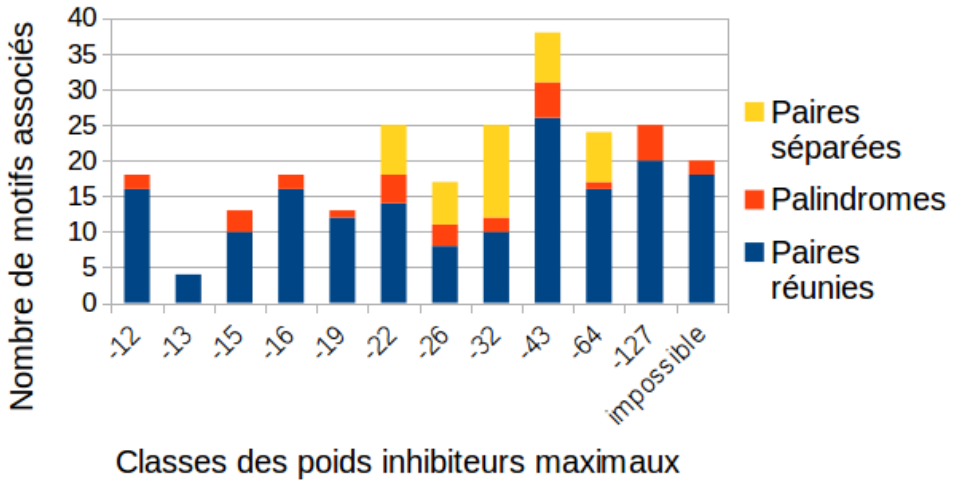


Générateur suivi de l'inhibition de comportement (2)



Comportement dynamique

Figure: Quantité de motifs capables de provoquer une inhibition regroupés par classe de poids et par type.



- Pertinence du codage temporel.

Conclusion et Perspectives

CONCLUSION	
Composition : moduler un comportement	
Composition : créer de nouveaux comportements	
Plusieurs compositions pour le même comportement	
Codage temporel vs. codage en débit	

- Résultats présentés à la rencontre C@UCA.
- Rédaction d'un article en cours avec F. Grammont et A. Muzy.

Remerciements

- À Cyrille Mascart.
- À toute l'équipe MDSC.

Conclusion et Perspectives

CONCLUSION	PERSPECTIVES
Composition : moduler un comportement	Tester d'autres compositions
Composition : créer de nouveaux comportements	Complexifier le modèle
Plusieurs compositions pour le même comportement	Apprentissage automatique de paramètres
Codage temporel vs. codage en débit	FPGA

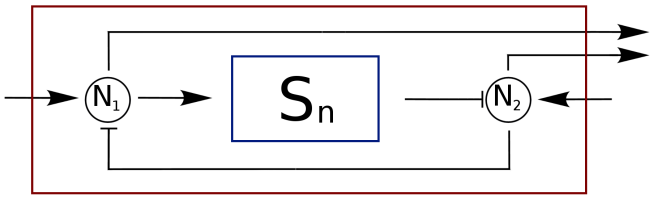
- Résultats présentés à la rencontre C@UCA.
- Rédaction d'un article en cours avec F. Grammont et A. Muzy.

Remerciements

- À Cyrille Mascart.
- À toute l'équipe MDSC.

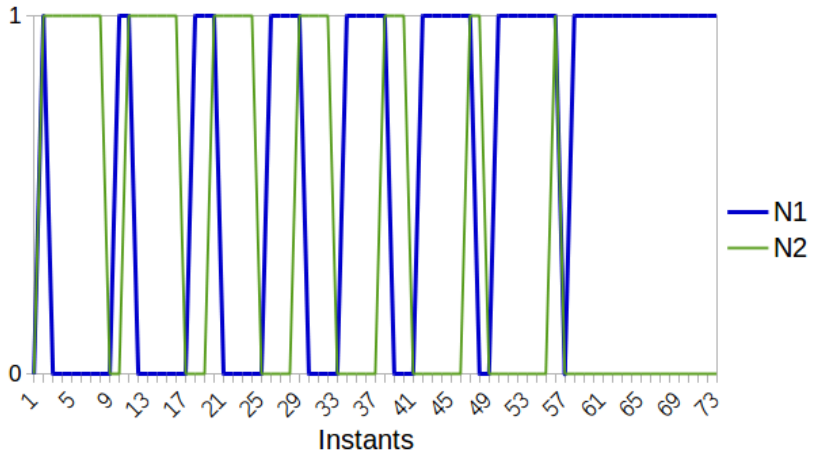
Annexe (1)

Série imbriqué dans l'inhibition controlatérale



Comportement dynamique

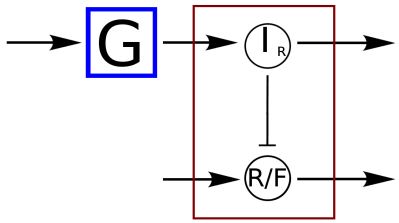
Figure: Signaux en sortie de N_1 et de N_2



- Le comportement "Winner takes all" est précédé d'une oscillation amortie.

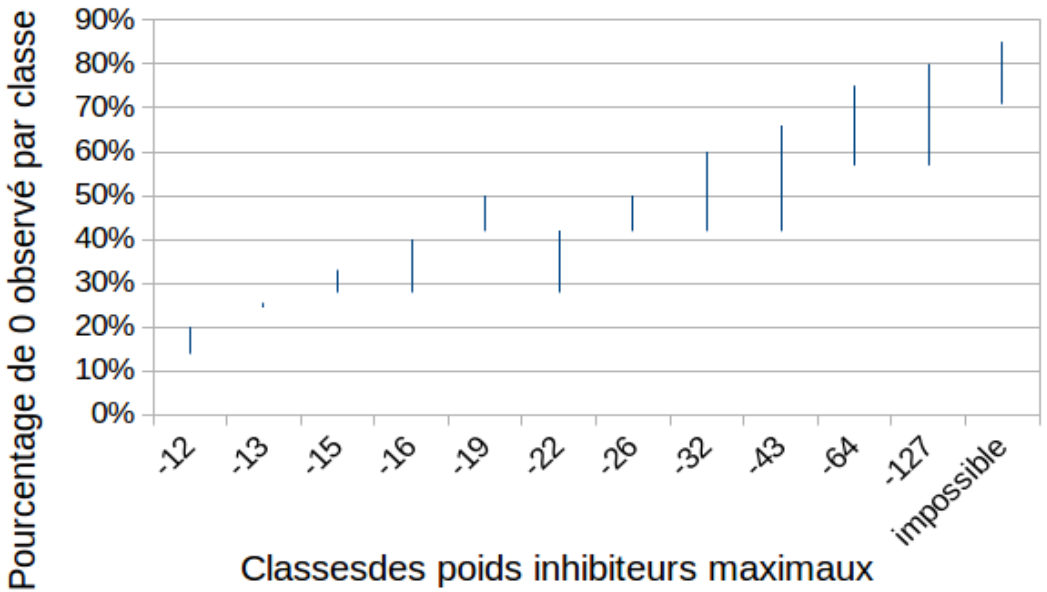
Annexe (2)

Générateur suivi de l'inhibition de comportement



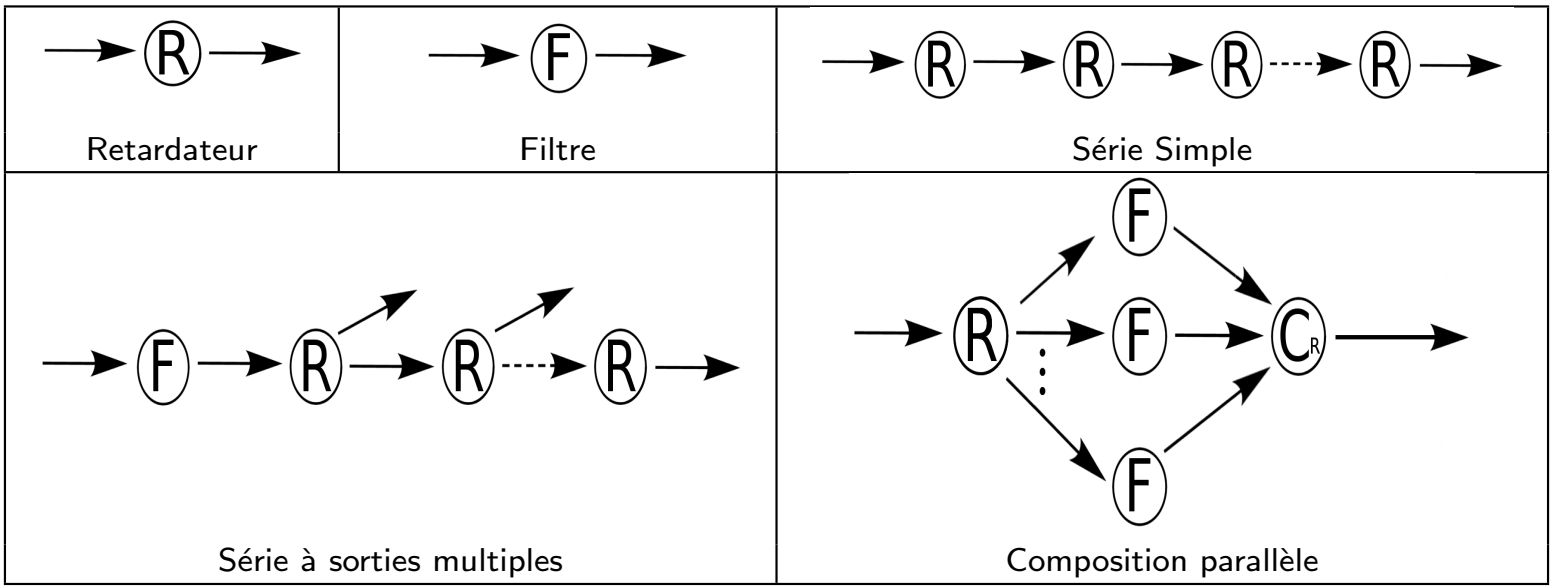
Comportement dynamique

Figure: Proportion de 0 dans les motifs par classes de poids inhibiteurs



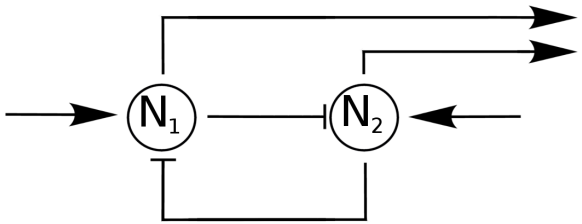
Annexe (3)

Autres Résultats (1)

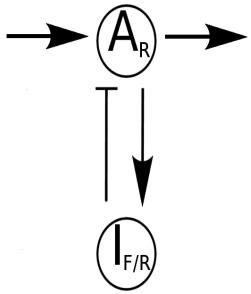


Annexe (3)

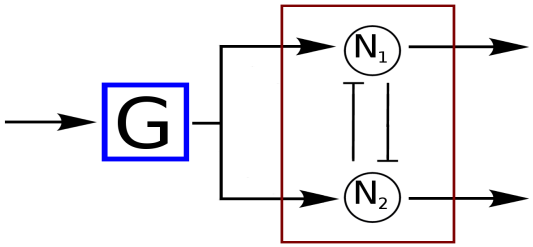
Autres Résultats (2)



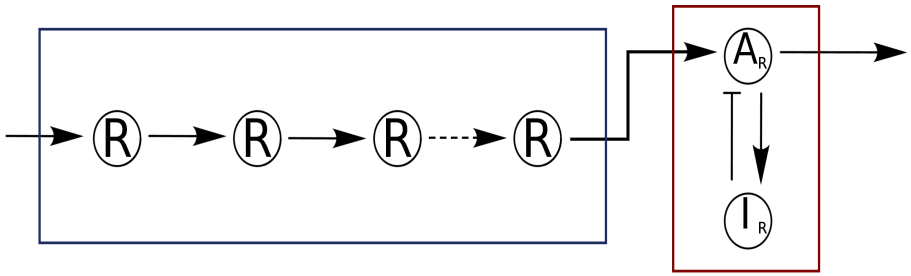
Inhibition controlatérale



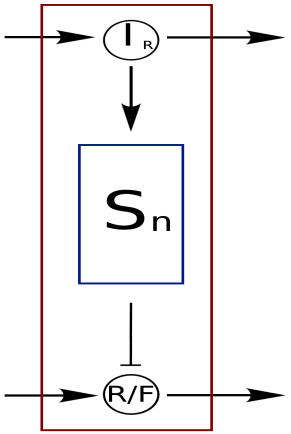
Boucle de rétroaction



Générateur de motifs suivi de l'inhibition controlatérale



Série suivie de la boucle de rétroaction



Série imbriquée dans l'inhibition de comportement