Automata Computation with Hodgkin-Huxley Neural Networks Composed of Synfire Rings

Jérémie Cabessa & Aubin Tchaptchet

Laboratoire d'économie mathématique
Université Paris II, France

July 10, 2018, Rio de Janeiro, Brazil

Introduction

- This work focuses on the simulation of digital computers by biological neural networks.
- It is known that first-order discrete-time recurrent neural networks with integer, rational or real weights are computationally equivalent to automata (Kleene 56, Minsky 67), Turing machines (Siegelmann & Sontag 95), and Turing machines with advices (super-Turing) (Siegelmann & Sontag 94, Cabessa & Siegelmann 14), respectively.
- What about the possibility to simulate abstract machines with more biological neural networks?

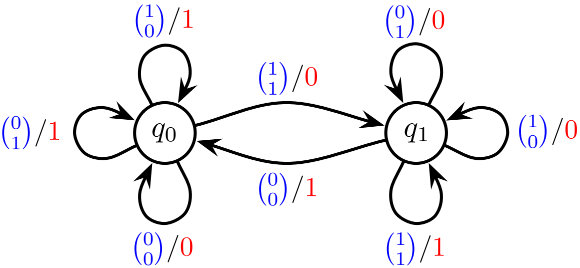
Introduction

- This work focuses on the simulation of digital computers by biological neural networks.
- It is known that first-order discrete-time recurrent neural networks with integer, rational or real weights are computationally equivalent to automata (Kleene 56, Minsky 67), Turing machines (Siegelmann & Sontag 95), and Turing machines with advices (super-Turing) (Siegelmann & Sontag 94, Cabessa & Siegelmann 14), respectively.
- What about the possibility to simulate abstract machines with more biological neural networks?

Introduction

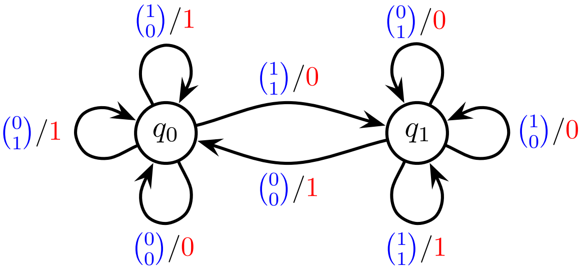
- This work focuses on the simulation of digital computers by biological neural networks.
- It is known that first-order discrete-time recurrent neural networks with integer, rational or real weights are computationally equivalent to automata (Kleene 56, Minsky 67), Turing machines (Siegelmann & Sontag 95), and Turing machines with advices (super-Turing) (Siegelmann & Sontag 94, Cabessa & Siegelmann 14), respectively.
- What about the possibility to simulate abstract machines with more biological neural networks?

Binary Adder Automaton



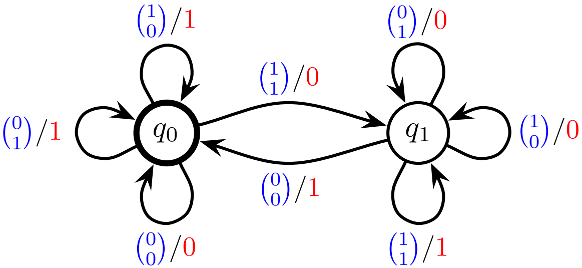
	^{1}	1^1	0^1	1	1	0^1	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



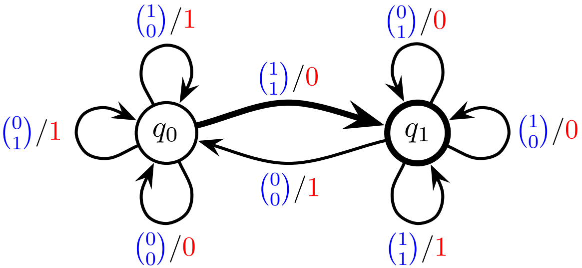
	^{1}	1^1	0^1	1	1	0^1	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



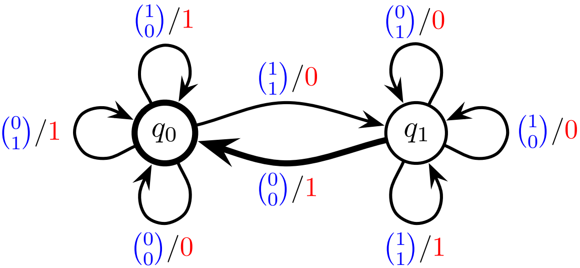
	^{1}	1^1	0^1	1	1	0^1	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



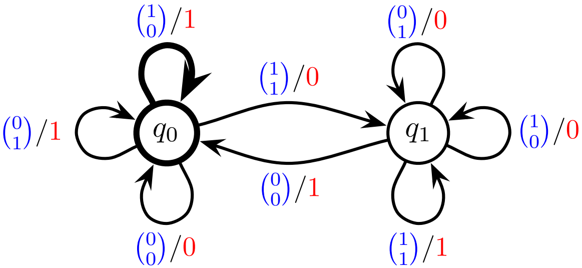
	^{1}	1^{1}	0^{1}	1	1	0^{1}	1
$+$		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



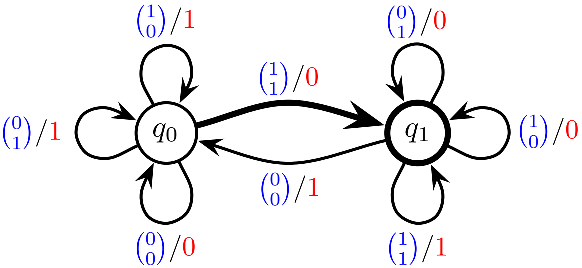
	^{1}	1^1	0^1	1	1	0^1	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



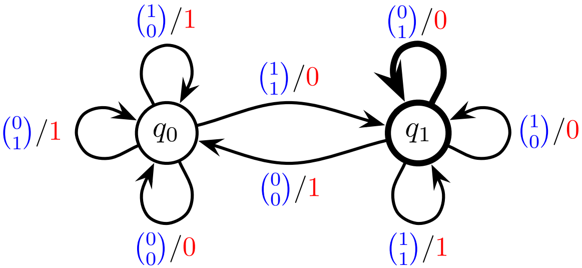
	^{1}	1^1	0^1	1	1	0^1	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



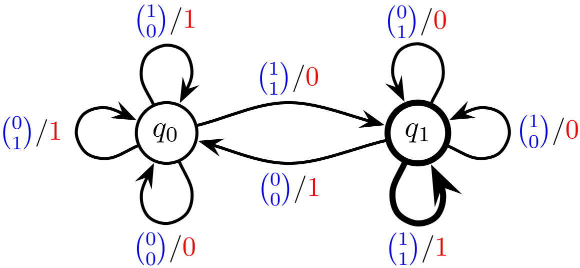
	^{1}	1^{1}	0^{1}	1	1	0^{1}	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



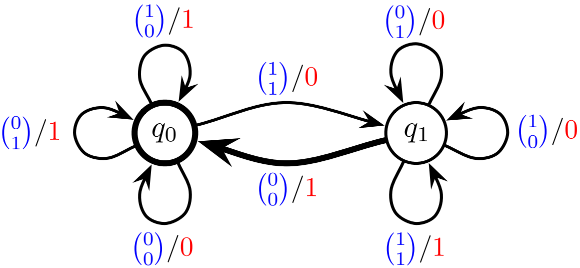
	^{1}	1^1	0^1	1	1	0^1	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



	^{1}	1^{1}	0^{1}	1	1	0^{1}	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Binary Adder Automaton



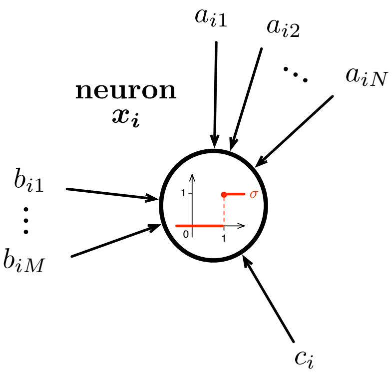
	^{1}	1^{1}	0^{1}	1	1	0^{1}	1
+		1	1	1	0	0	1
	1	1	0	0	1	1	0

Automata & Boolean RNNs

Theorem (Minsky 1967)

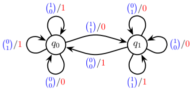
Any finite state automaton can be simulated by some Boolean recurrent neural network.

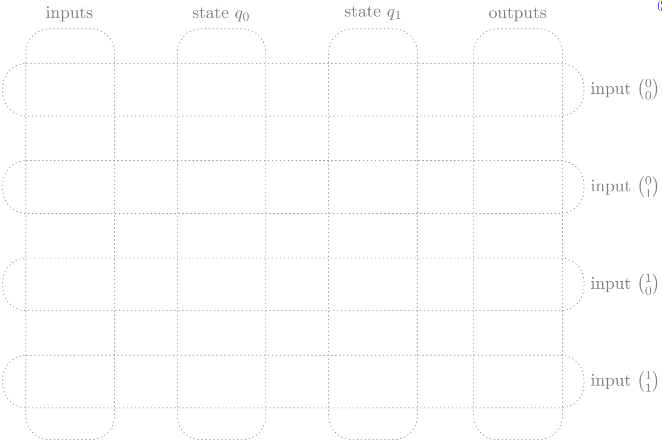
Boolean Recurrent Neural Networks



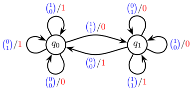
$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

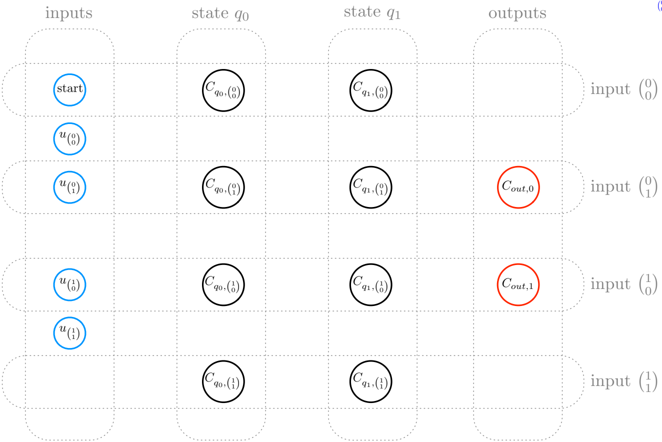
Binary Adder Boolean Neural Network



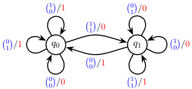


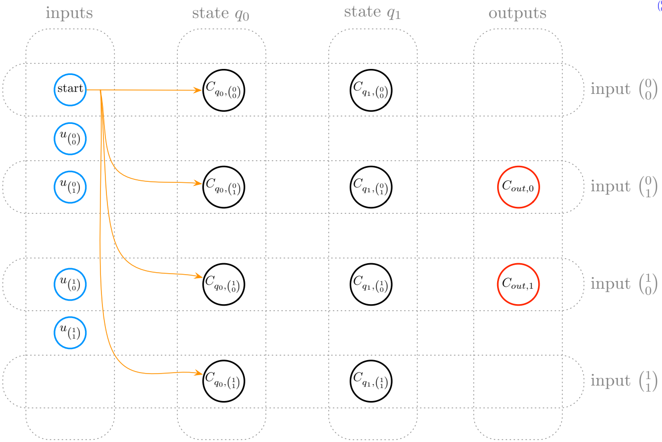
Binary Adder Boolean Neural Network



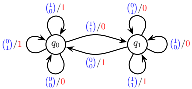


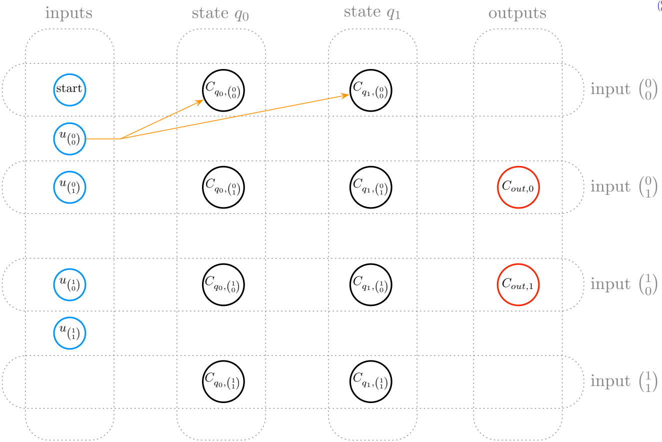
Binary Adder Boolean Neural Network



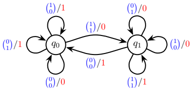


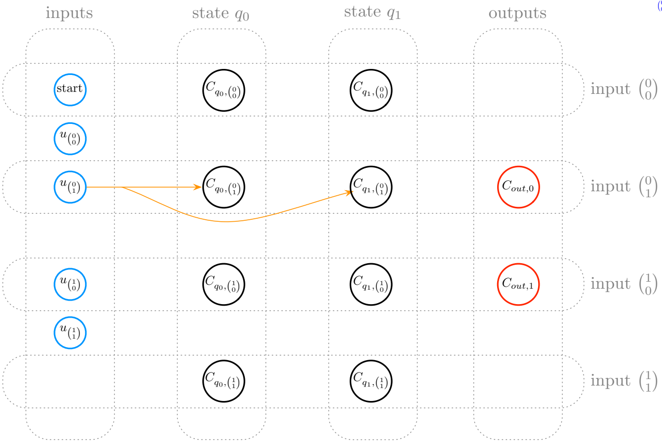
Binary Adder Boolean Neural Network



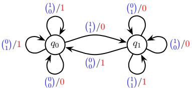


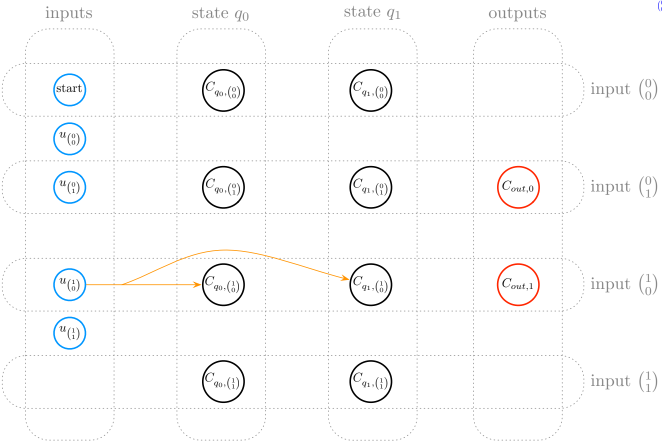
Binary Adder Boolean Neural Network



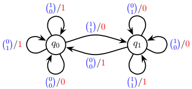


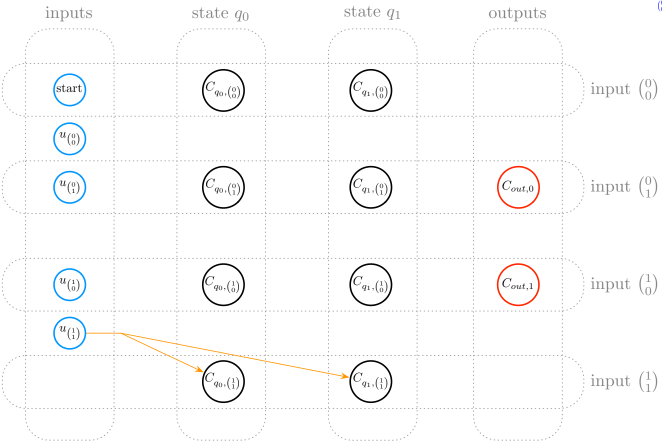
Binary Adder Boolean Neural Network



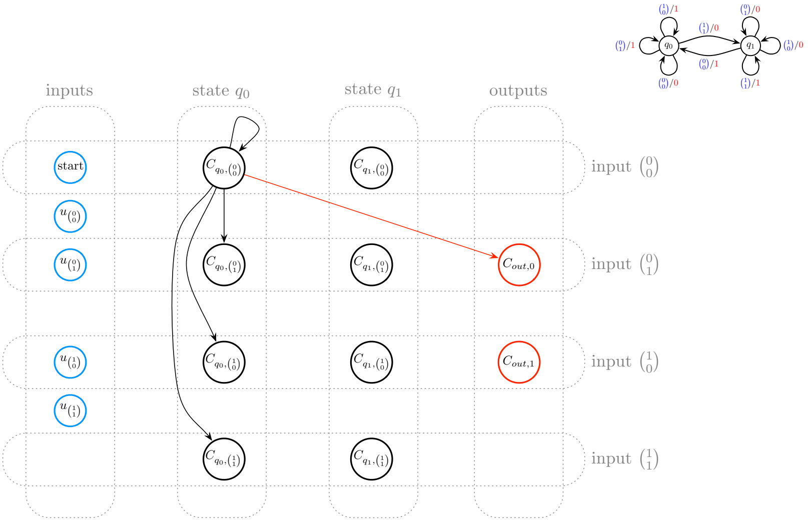


Binary Adder Boolean Neural Network

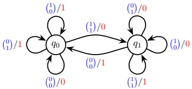


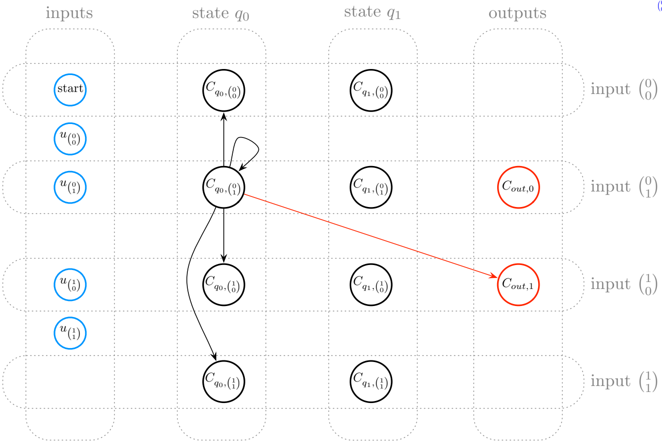


Binary Adder Boolean Neural Network

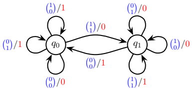


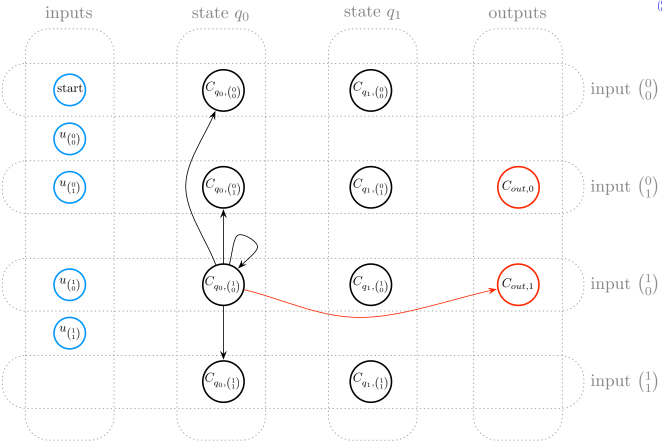
Binary Adder Boolean Neural Network



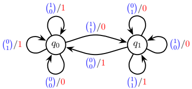


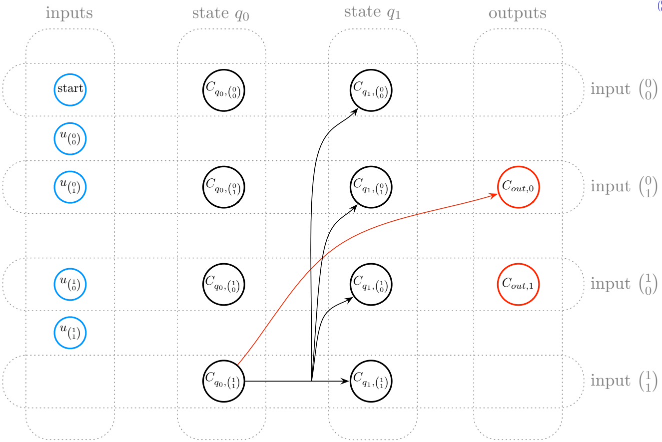
Binary Adder Boolean Neural Network



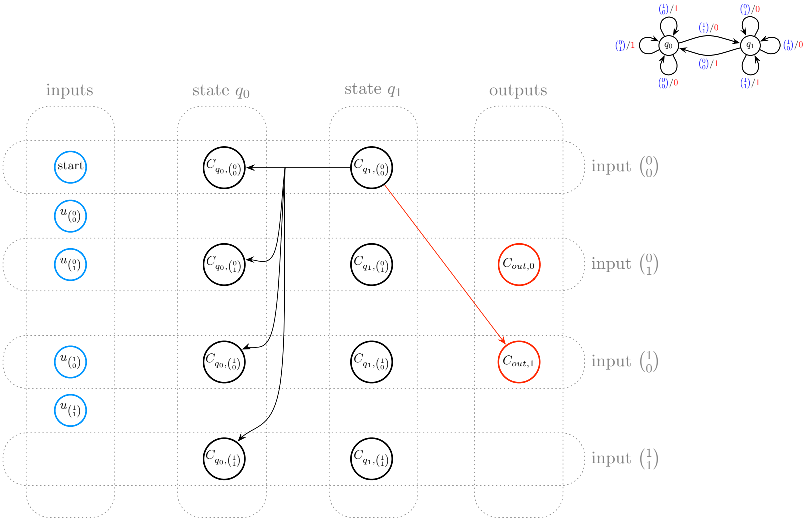


Binary Adder Boolean Neural Network

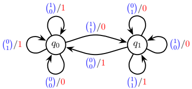


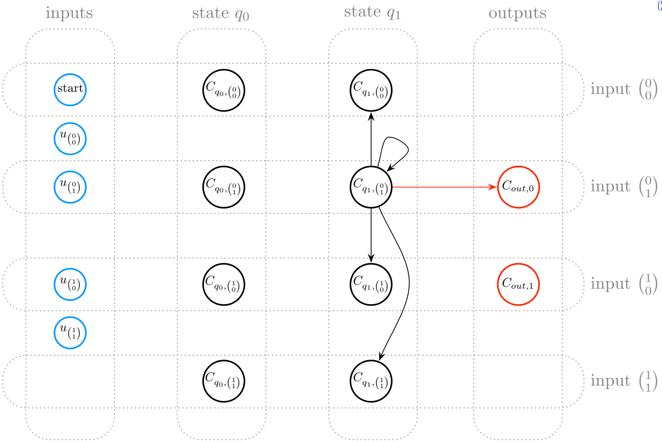


Binary Adder Boolean Neural Network

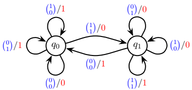


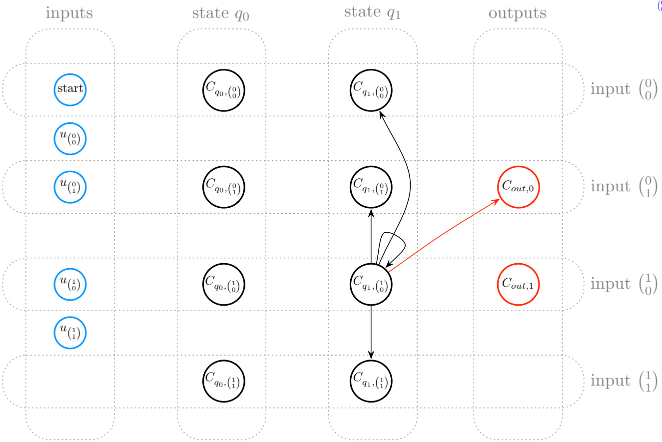
Binary Adder Boolean Neural Network



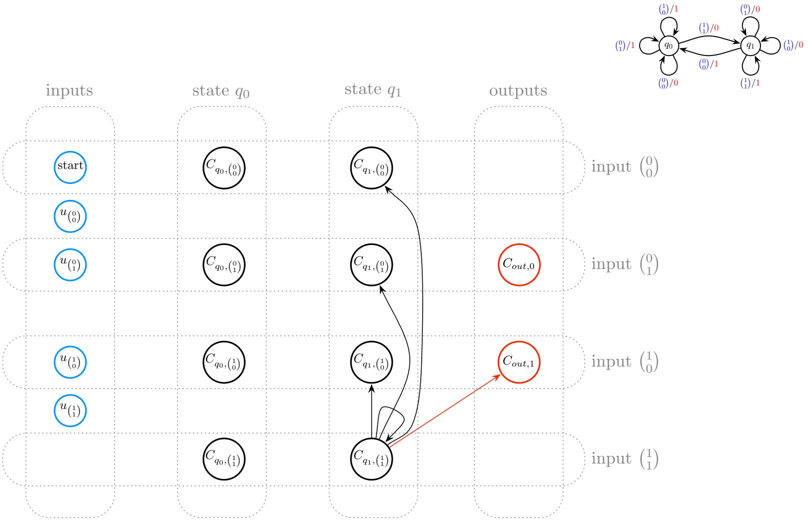


Binary Adder Boolean Neural Network

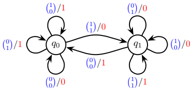


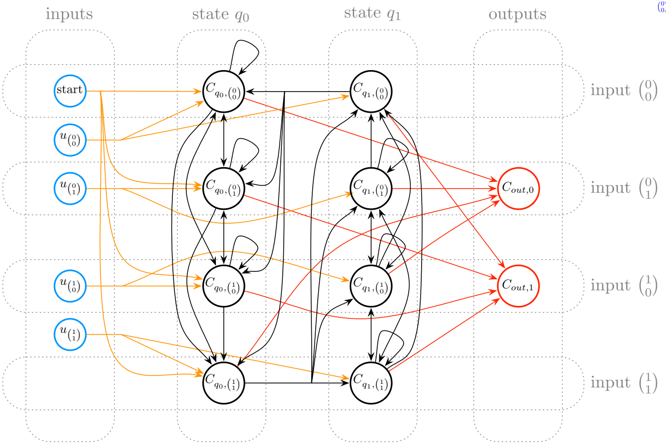


Binary Adder Boolean Neural Network



Binary Adder Boolean Neural Network





Simulation

time	0	1	2	3	4	5	6	7	8
states	q_0	q_1	q_0	q_0	q_1	q_1	q_1	q_0	–
inputs	$\binom{1}{1}$	$\binom{0}{0}$	$\binom{1}{0}$	$\binom{1}{1}$	$\binom{0}{1}$	$\binom{1}{1}$	$\binom{0}{0}$	–	–
outputs	0	1	1	0	0	1	1	–	–
$start$	1	0	0	0	0	0	0	0	0
$u_{\binom{0}{0}}$	0	1	0	0	0	0	1	0	0
$u_{\binom{0}{1}}$	0	0	0	0	1	0	0	0	0
$u_{\binom{1}{0}}$	0	0	1	0	0	0	0	0	0
$u_{\binom{1}{1}}$	1	0	0	1	0	1	0	0	0
$C_{s,i}$	–	$q_0, \binom{1}{1}$	$q_1, \binom{0}{0}$	$q_0, \binom{1}{0}$	$q_0, \binom{1}{1}$	$q_1, \binom{0}{1}$	$q_1, \binom{1}{1}$	$q_1, \binom{0}{0}$	–
$C_{out,0}$	0	0	1	0	0	1	1	0	0
$C_{out,1}$	0	0	0	1	1	0	0	1	1

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.

⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.

- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.

⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.

⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.

- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.

⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.

⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.

- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.

⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.

⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.

- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.

⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Neural Computation with Synfire Rings

- We introduce a paradigm of neural computation based on *synfire rings*.
- Computational states are represented by sustained activities within synfire rings.
- Hence, the successive computational states are encoded into cyclic attractors.
- The transitions between such attractors are perfectly controlled by the inputs.
- The global computational process is robust to various kinds of architectural plasticities and noises.

Neural Computation with Synfire Rings

- We introduce a paradigm of neural computation based on *synfire rings*.
- Computational states are represented by sustained activities within synfire rings.
- Hence, the successive computational states are encoded into cyclic attractors.
- The transitions between such attractors are perfectly controlled by the inputs.
- The global computational process is robust to various kinds of architectural plasticities and noises.

Neural Computation with Synfire Rings

- We introduce a paradigm of neural computation based on *synfire rings*.
- Computational states are represented by sustained activities within synfire rings.
- Hence, the successive computational states are encoded into cyclic attractors.
- The transitions between such attractors are perfectly controlled by the inputs.
- The global computational process is robust to various kinds of architectural plasticities and noises.

Neural Computation with Synfire Rings

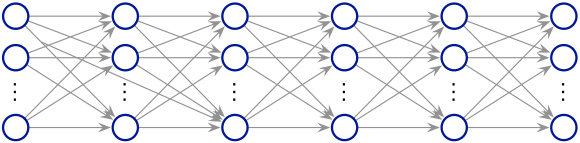
- We introduce a paradigm of neural computation based on *synfire rings*.
- Computational states are represented by sustained activities within synfire rings.
- Hence, the successive computational states are encoded into cyclic attractors.
- The transitions between such attractors are perfectly controlled by the inputs.
- The global computational process is robust to various kinds of architectural plasticities and noises.

Neural Computation with Synfire Rings

- We introduce a paradigm of neural computation based on *synfire rings*.
- Computational states are represented by sustained activities within synfire rings.
- Hence, the successive computational states are encoded into cyclic attractors.
- The transitions between such attractors are perfectly controlled by the inputs.
- The global computational process is robust to various kinds of architectural plasticities and noises.

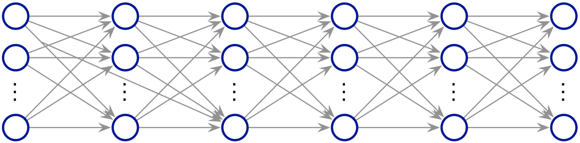
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



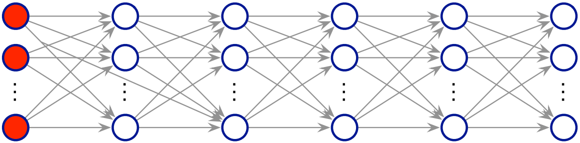
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



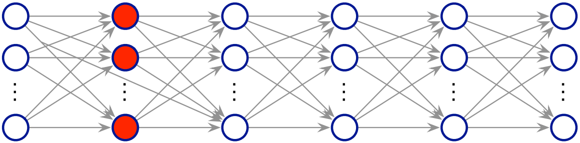
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



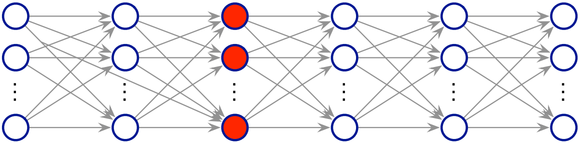
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



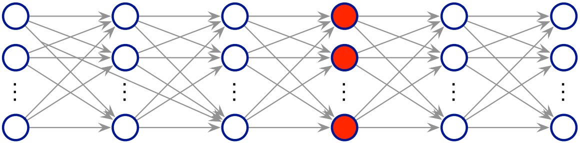
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



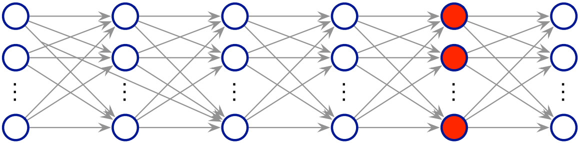
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



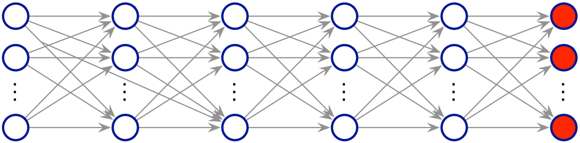
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



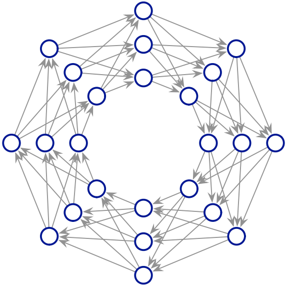
Synfire Chains

- *Synfire chains* allow for robust and highly precise transmission of information in neural networks (Abeles 82).
- *Synfire chains* are likely to be crucially involved in the processing and coding of information in neural networks.



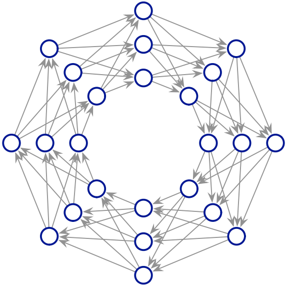
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



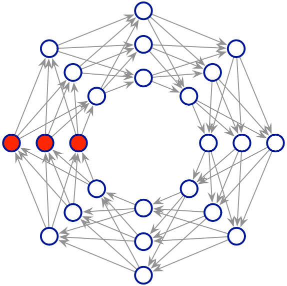
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



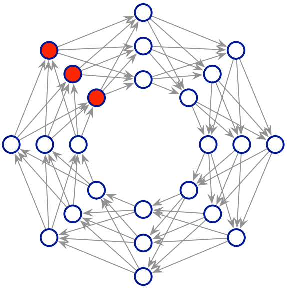
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



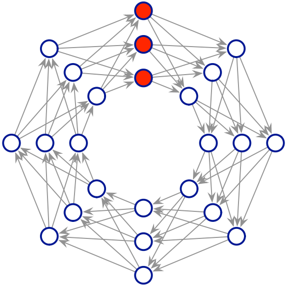
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



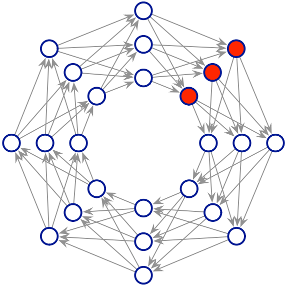
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



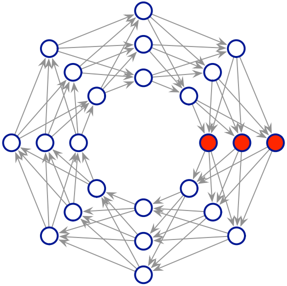
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



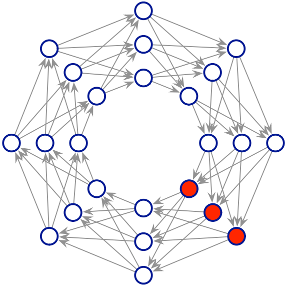
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



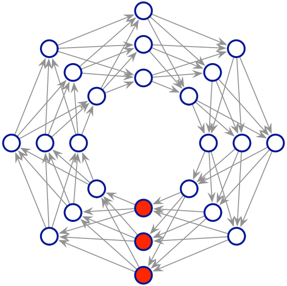
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



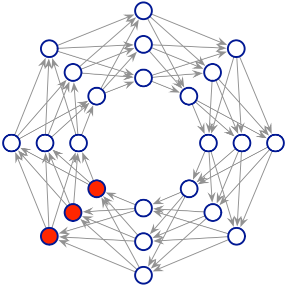
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



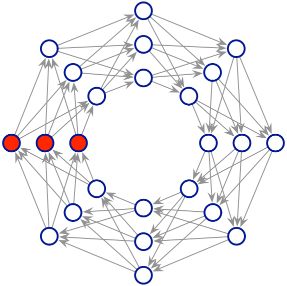
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



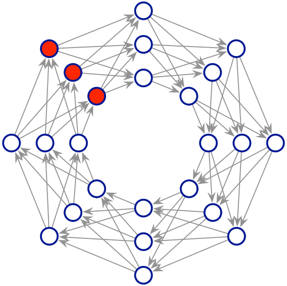
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- *Synfire rings* allow for robust and temporally precise *self-sustained activities*.



Hodgkin-Huxley Neurons (software demo)

$$\alpha_n(V_m) = \frac{0.01(10 - V_m)}{\exp(\frac{10-V_m}{10}) - 1} \qquad \beta_n(V_m) = 0.125\exp(\frac{-V_m}{80})$$

$$\alpha_m(V_m) = \frac{0.1(25 - V_m)}{\exp(\frac{25-V_m}{10}) - 1} \qquad \beta_m(V_m) = 4\exp(\frac{-V_m}{18})$$

$$\alpha_h(V_m) = 0.07\exp(\frac{-V_m}{20}) \qquad \beta_h(V_m) = \frac{1}{\exp(\frac{30-V_m}{10}) + 1}$$

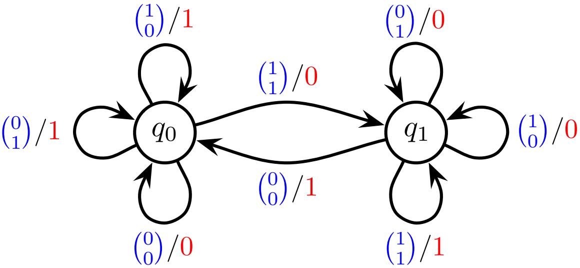
$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

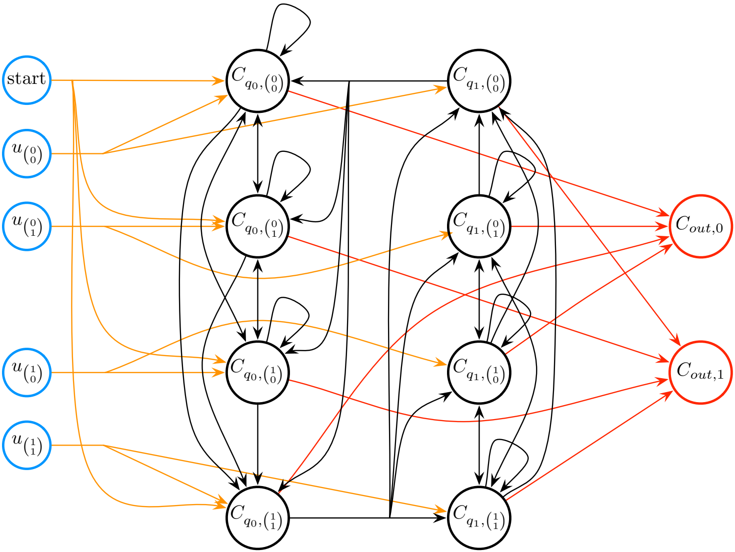
$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h$$

$$C_m\frac{dV_m}{dt} = I - \bar{g}_{\mathsf{K}}n^4(V_m - V_K) - \bar{g}_{\mathsf{Na}}m^3h(V_m - V_{Na}) - \bar{g}_l(V_m - V_l)$$

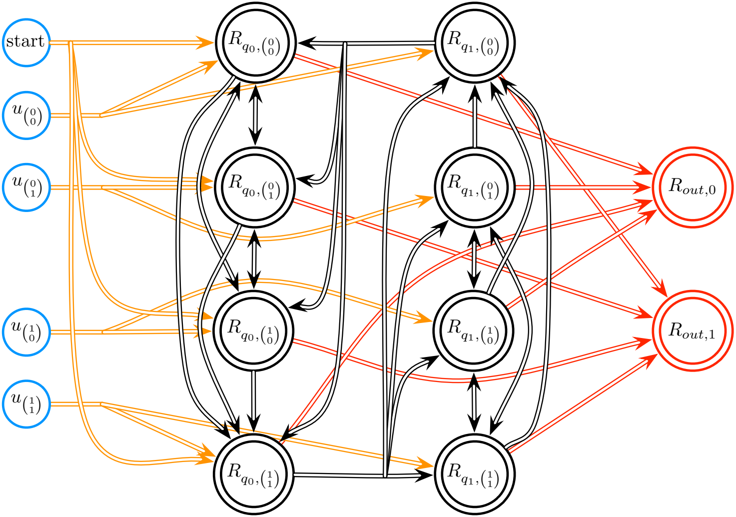
General Construction



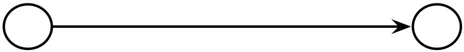
General Construction



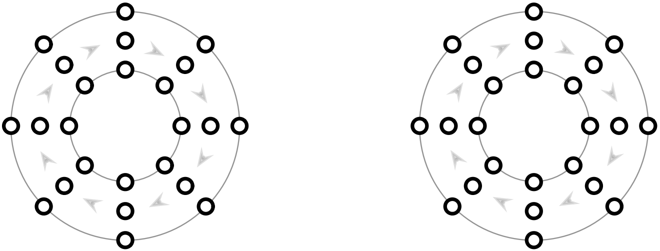
General Construction



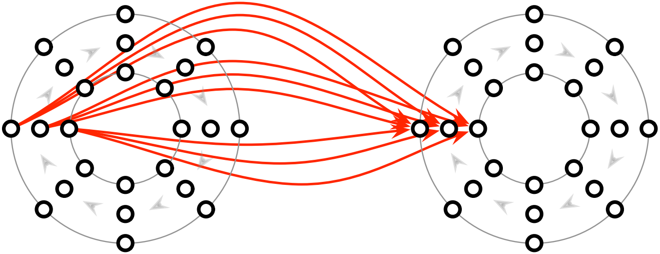
Fibres of connections & Inhibitory System



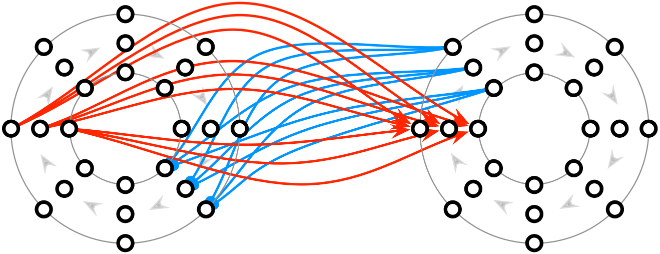
Fibres of connections & Inhibitory System



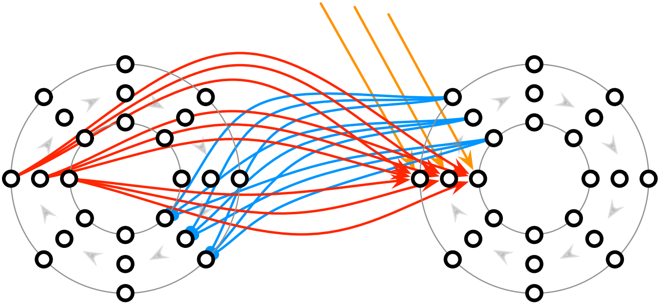
Fibres of connections & Inhibitory System



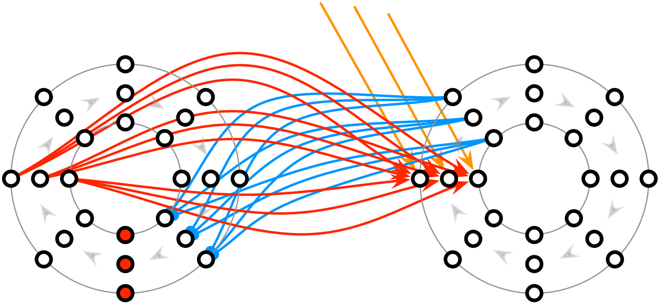
Fibres of connections & Inhibitory System



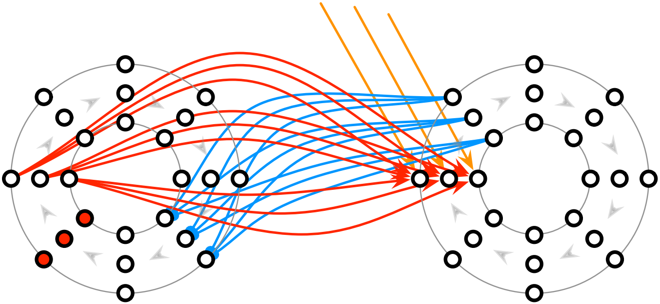
Fibres of connections & Inhibitory System



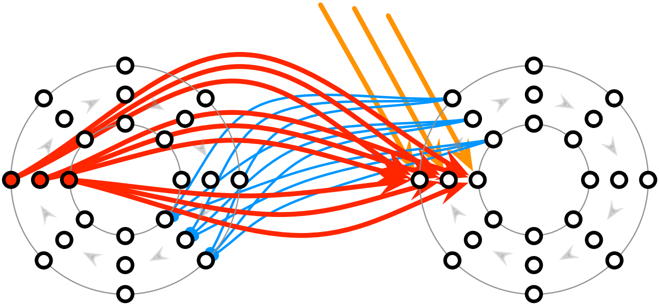
Fibres of connections & Inhibitory System



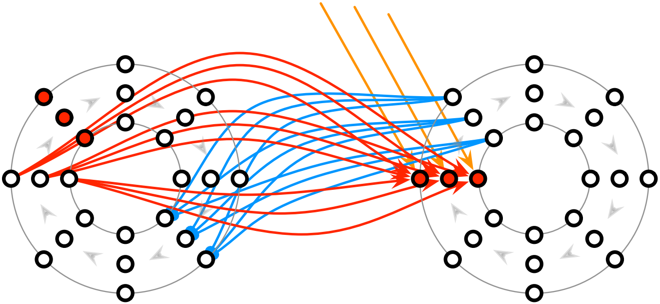
Fibres of connections & Inhibitory System



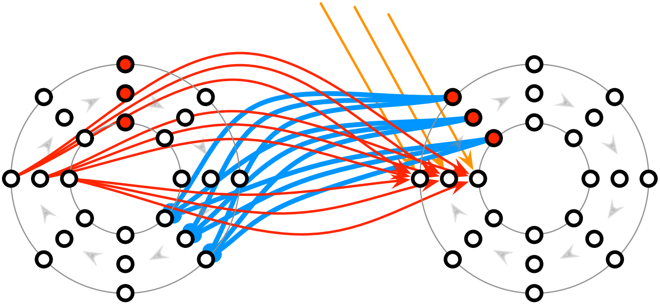
Fibres of connections & Inhibitory System



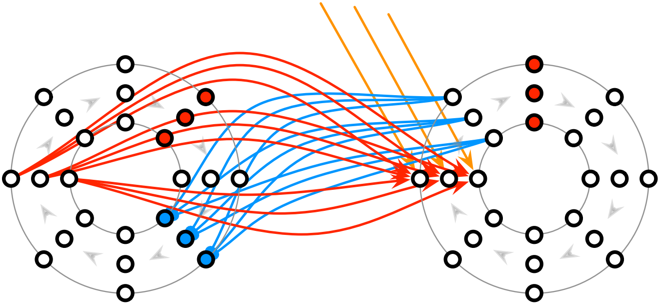
Fibres of connections & Inhibitory System



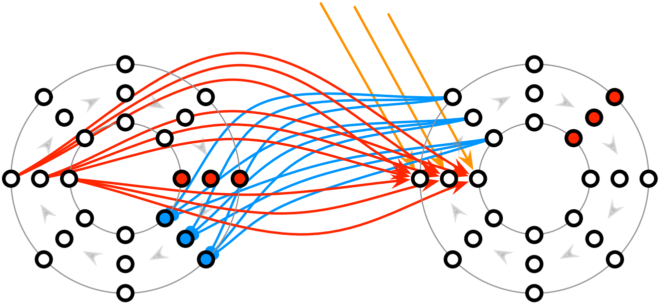
Fibres of connections & Inhibitory System



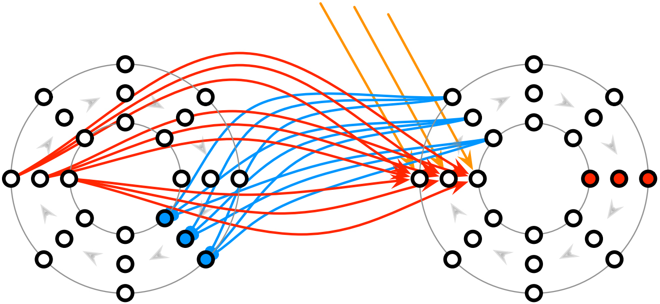
Fibres of connections & Inhibitory System



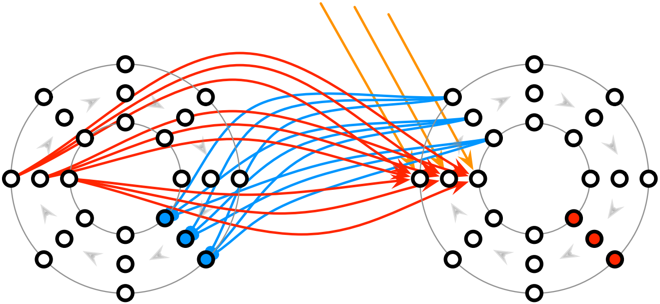
Fibres of connections & Inhibitory System



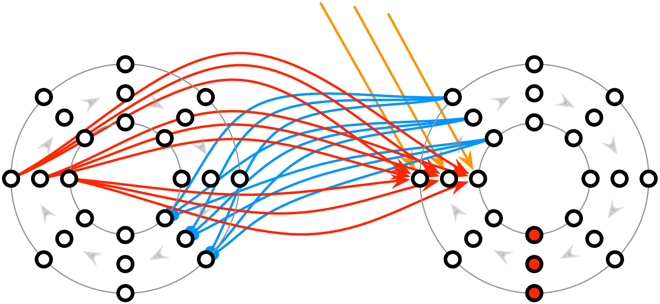
Fibres of connections & Inhibitory System



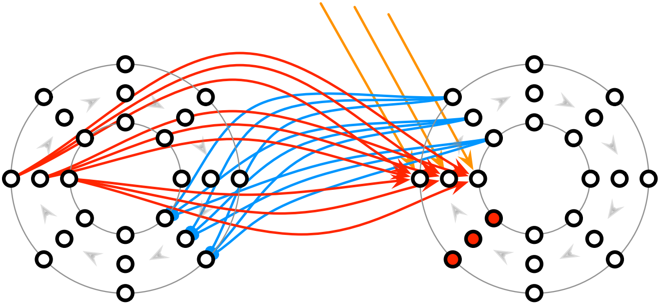
Fibres of connections & Inhibitory System



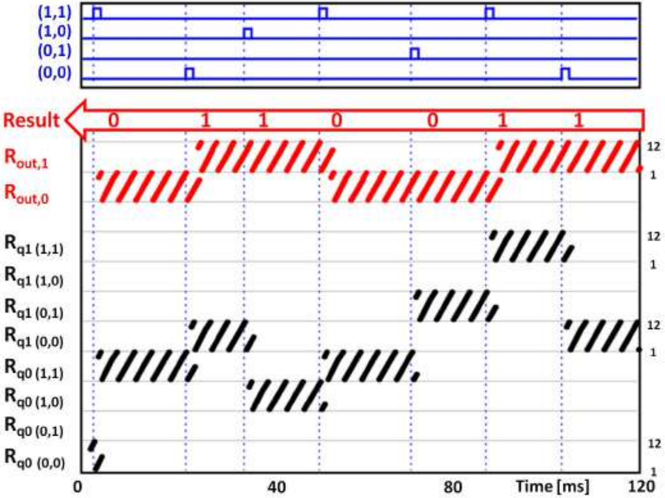
Fibres of connections & Inhibitory System



Fibres of connections & Inhibitory System



Simulation


(1,1)
(1,0)
(0,1)
(0,0)
Result
0 1 1 0 0 1 1
$R_{out,1}$
$R_{out,0}$
$R_{q1\,(1,1)}$
$R_{q1\,(1,0)}$
$R_{q1\,(0,1)}$
$R_{q1\,(0,0)}$
$R_{q0\,(1,1)}$
$R_{q0\,(1,0)}$
$R_{q0\,(0,1)}$
$R_{q0\,(0,0)}$
12
1
0
40
80
Time [ms]
120

Simulation

Play movie...

Results

Algorithm 1

Require: DFSA $\mathcal{A} = (Q, \Sigma, \delta_{\mathcal{A}}, q_0, F)$ (resp. DFST $\mathcal{T} = (Q, \Sigma, \delta_{\mathcal{T}}, q_0)$)

consider K input cells $(u_a)_{a \in \Sigma}$, where $K = |\Sigma|$
consider $I \times J$ synfire rings $(R_{q,a})_{q \in Q, a \in \Sigma}$, where $I = |\Sigma|$ and $J = |Q|$
consider K synfire rings $(R_{out,a})_{a \in \Sigma}$, where $K = |\Sigma|$
for all state $q \in Q$ **do**
  for all input symbol $a \in \Sigma$ **do**
    add a fibre of input connections from u_a to $R_{q,a}$
  end for
end for
for all transition $(q, a, q') \in graph(\delta_{\mathcal{A}})$ (resp. $(q, a, q', o) \in graph(\delta_{\mathcal{T}})$) **do**
  for all input symbol $a' \in \Sigma$ **do**
    add a fibre of inter-ring connections from $R_{q,a}$ to $R_{q',a'}$
    add a fibre of output connections from $R_{q,a}$ to $R_{out,o}$
  end for
end for
set weights w^{exc}_{input}, w^{exc}_{inter} and w^{exc}_{output} appropriately
set weights w^{inh}_{inter} and w^{inh}_{output} appropriately

Automata & Hodgkin-Huxley RNNs with Synfire Rings

Since the construction is generic, one has the following result:

Theorem

Any finite state automaton can be simulated by some Hodgkin-Huxley based neural network composed of synfire rings.

Conclusions

- We introduced a *bio-inspired paradigm of neural computation* based on the concept of *synfire rings*.
- We intend to extend the results towards the simulation of *Turing machines*.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of *biological neural computers*.

Thank you!

Conclusions

- We introduced a *bio-inspired paradigm of neural computation* based on the concept of *synfire rings*.
- We intend to extend the results towards the simulation of *Turing machines*.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of *biological neural computers*.

Thank you!

Conclusions

- We introduced a *bio-inspired paradigm of neural computation* based on the concept of *synfire rings*.
- We intend to extend the results towards the simulation of *Turing machines*.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of *biological neural computers*.

Thank you!

Conclusions

- We introduced a *bio-inspired paradigm of neural computation* based on the concept of *synfire rings*.
- We intend to extend the results towards the simulation of *Turing machines*.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of *biological neural computers*.

Thank you!

Conclusions

- We introduced a *bio-inspired paradigm of neural computation* based on the concept of *synfire rings*.
- We intend to extend the results towards the simulation of *Turing machines*.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of *biological neural computers*.

Thank you!

Conclusions

- We introduced a *bio-inspired paradigm of neural computation* based on the concept of *synfire rings*.
- We intend to extend the results towards the simulation of *Turing machines*.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of *biological neural computers*.

Thank you!