

LEARNING WITH SYNFIRE RING-BASED RECURRENT NEURAL NETWORKS

DARPA - LIFELONG LEARNING MACHINES (L2M) PI MEETING

Jérémy Cabessa

Laboratoire of Mathematical Economics
Université Paris II, Paris, France

October 24, 2018, Chicago, IL, USA

PROJECT SUMMARY

- ▶ We introduce a bio-inspired paradigm for neural computation based on the concept of **synfire rings**.
- ▶ We propose a **Turing-complete neural architecture** based on synfire rings.
- ▶ We intend to develop **learning algorithms** on this architecture:
 - ▶ Gradient-descent based algorithms
 - ▶ Evolutionary based algorithms
 - ▶ STDP-based algorithms

PROJECT SUMMARY

- ▶ We introduce a bio-inspired paradigm for neural computation based on the concept of **synfire rings**.
- ▶ We propose a **Turing-complete neural architecture** based on synfire rings.
- ▶ We intend to develop **learning algorithms** on this architecture:
 1. Gradient descent-based algorithms
 2. Evolutionary-based algorithms
 3. STDP-based algorithms

PROJECT SUMMARY

- ▶ We introduce a bio-inspired paradigm for neural computation based on the concept of **synfire rings**.
- ▶ We propose a **Turing-complete neural architecture** based on synfire rings.
- ▶ We intend to develop **learning algorithms** on this architecture:
 1. Gradient descent-based algorithms
 2. Evolutionary-based algorithms
 3. STDP-based algorithms

PROJECT SUMMARY

- ▶ We introduce a bio-inspired paradigm for neural computation based on the concept of **synfire rings**.
- ▶ We propose a **Turing-complete neural architecture** based on synfire rings.
- ▶ We intend to develop **learning algorithms** on this architecture:
 1. Gradient descent-based algorithms
 2. Evolutionary-based algorithms
 3. STDP-based algorithms

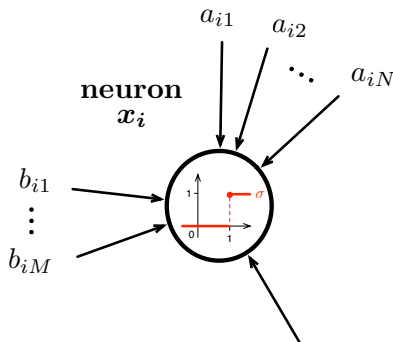
PROJECT SUMMARY

- ▶ We introduce a bio-inspired paradigm for neural computation based on the concept of **synfire rings**.
- ▶ We propose a **Turing-complete neural architecture** based on synfire rings.
- ▶ We intend to develop **learning algorithms** on this architecture:
 1. Gradient descent-based algorithms
 2. Evolutionary-based algorithms
 3. STDP-based algorithms

PROJECT SUMMARY

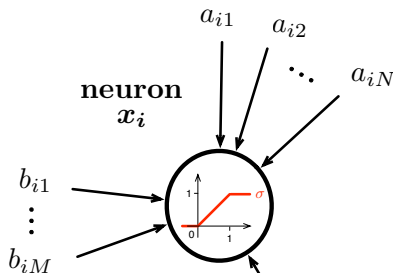
- ▶ We introduce a bio-inspired paradigm for neural computation based on the concept of **synfire rings**.
- ▶ We propose a **Turing-complete neural architecture** based on synfire rings.
- ▶ We intend to develop **learning algorithms** on this architecture:
 1. Gradient descent-based algorithms
 2. Evolutionary-based algorithms
 3. STDP-based algorithms

BOOLEAN RECURRENT NEURAL NETWORKS



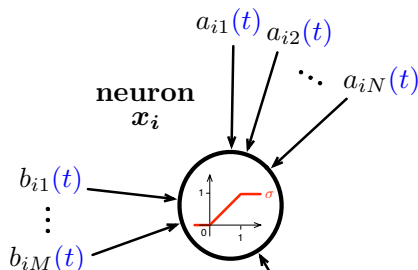
$$x_i(t+1) = \theta \left(\sum_{j=1}^N a_{ij} \cdot x_j(t) + \sum_{j=1}^M b_{ij} \cdot u_j(t) + c_i \right)$$

SIGMOID RECURRENT NEURAL NETWORKS



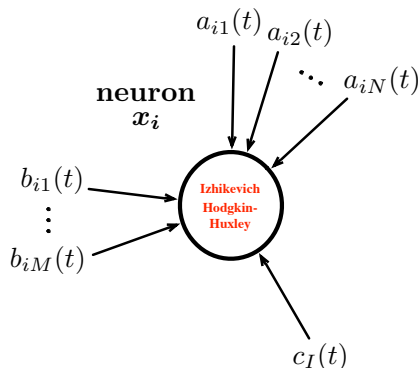
$$x_i(t+1) = \sigma \left(\sum_{j=1}^N a_{ij} \cdot x_j(t) + \sum_{j=1}^M b_{ij} \cdot u_j(t) + c_i \right)$$

SIGMOID RECURRENT NEURAL NETWORKS



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

SPIKING RECURRENT NEURAL NETWORKS



Neurons' activities are modelled by Izhikevich or Hodgkin-Huxley differential equations.

IZHIKEVICH SPIKING NEURAL NETWORKS

Izhikevich differential equations:

$$\begin{cases} v' = 0.04v^2 + 5v + 140 - u + I \\ u' = a(bv - u) \end{cases}$$

with the auxiliary after-spike resetting:

if $v \geq 30$ mV, then $v \leftarrow c$ and $u \leftarrow u + d$

- ▶ v : membrane potential
- ▶ u : membrane recovery variable
- ▶ I : synaptic currents
- ▶ a, b, c, d : dimensionless parameters

HODGKIN-HUXLEY SPIKING NEURAL NETWORKS

$$\alpha_n(V_m) = \frac{0.01(10 - V_m)}{\exp(\frac{10 - V_m}{10}) - 1}$$

$$\beta_n(V_m) = 0.125 \exp(\frac{-V_m}{80})$$

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

$$\beta_m(V_m) = 4 \exp(\frac{-V_m}{18})$$

$$\alpha_h(V_m) = 0.07 \exp(\frac{-V_m}{20})$$

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

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

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

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

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

STATE OF THE ART RESULTS

	BOOLEAN	SIGMOID		
		STATIC	BI-VALUED EVOLVING	EVOLVING
Q	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	P	P/poly	P/poly
	Kleene56	SiegelmannSontag95	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18
R	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	P/poly	P/poly	P/poly
	Kleene56	SiegelmannSontag94	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18

STATE OF THE ART RESULTS

	BOOLEAN	STATIC	SIGMOID BI-VALUED EVOLVING	EVOLVING
\mathbb{Q}	FSA REG	TM P	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag95	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18
\mathbb{R}	FSA REG	TM/poly(A) P/poly	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag94	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18

STATE OF THE ART RESULTS

	BOOLEAN	STATIC	SIGMOID BI-VALUED EVOLVING	EVOLVING
\mathbb{Q}	FSA REG	TM P	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag95	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18
\mathbb{R}	FSA REG	TM/poly(A) P/poly	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag94	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18

STATE OF THE ART RESULTS

	BOOLEAN	STATIC	SIGMOID BI-VALUED EVOLVING	EVOLVING
\mathbb{Q}	FSA REG	TM P	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag95	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18
\mathbb{R}	FSA REG	TM/poly(A) P/poly	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag94	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18

STATE OF THE ART RESULTS

	BOOLEAN	STATIC	SIGMOID BI-VALUED EVOLVING	EVOLVING
\mathbb{Q}	FSA REG	TM P	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag95	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18
\mathbb{R}	FSA REG	TM/poly(A) P/poly	TM/poly(A) P/poly	TM/poly(A) P/poly
	Kleene56	SiegelmannSontag94	CabessaSiegelmann14	CabessaSiegelmann14
	Minsky67	CabessaSiegelmann12	CabessaVilla15,16	CabessaVilla15,16
	CabessaVilla14	CabessaVilla15,16	CabessaDuparc17	CabessaDuparc17
		CabessaDuparc17	CabessaFinkel18	CabessaFinkel18

DRAWBACKS

- ▶ Computational states of the machines are represented by 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 noise*.

DRAWBACKS

- ▶ Computational states of the machines are represented by 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 noise*.

DRAWBACKS

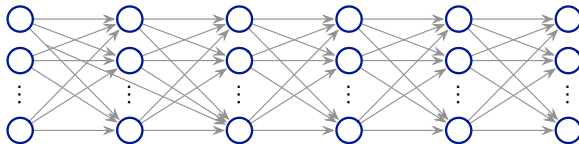
- ▶ Computational states of the machines are represented by spiking configurations of the network.
- ★ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.
- ▶ Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.
- ★ Network should be robust to *architectural plasticity* and *synaptic noise*.

DRAWBACKS

- ▶ Computational states of the machines are represented by 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 noise*.

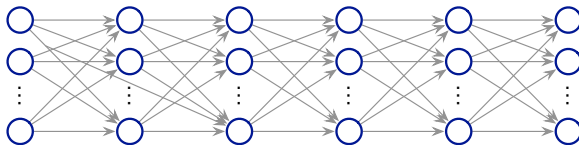
SYNFIRE CHAINS

- ▶ *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.



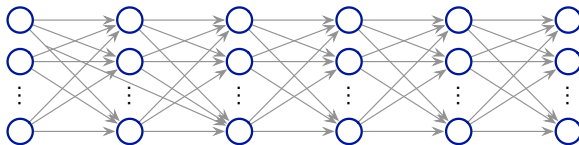
SYNFIRE CHAINS

- ▶ *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.



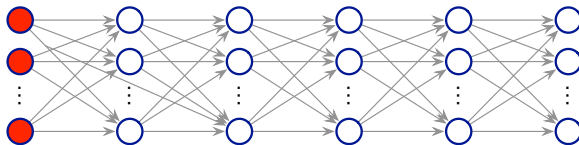
SYNFIRE CHAINS

- ▶ *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.



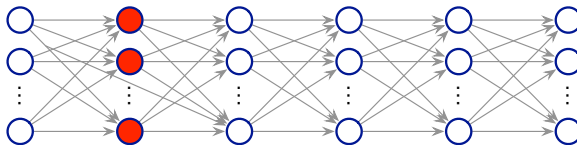
SYNFIRE CHAINS

- ▶ *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.



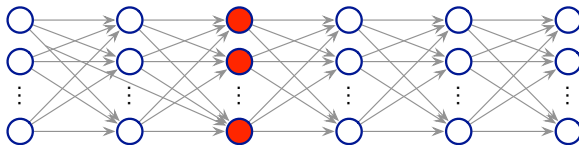
SYNFIRE CHAINS

- ▶ *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.



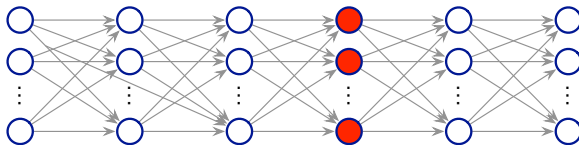
SYNFIRE CHAINS

- ▶ *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.



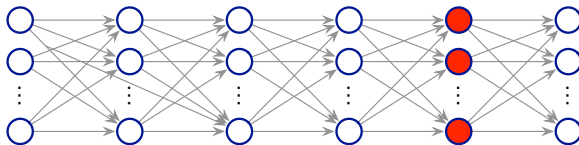
SYNFIRE CHAINS

- ▶ *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.



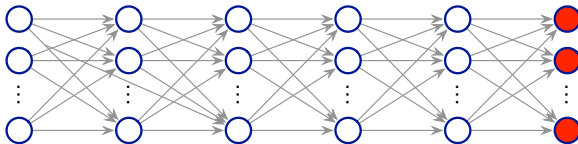
SYNFIRE CHAINS

- ▶ *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.



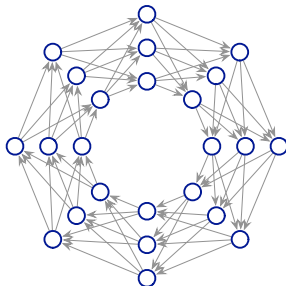
SYNFIRE CHAINS

- ▶ *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.



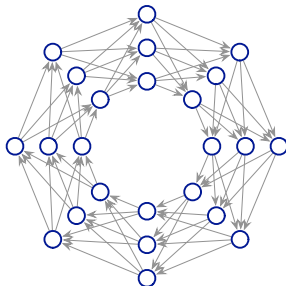
SYNFIRE RINGS

- ▶ *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*.



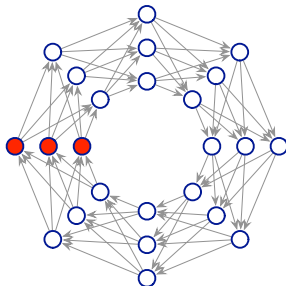
SYNFIRE RINGS

- ▶ *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*.



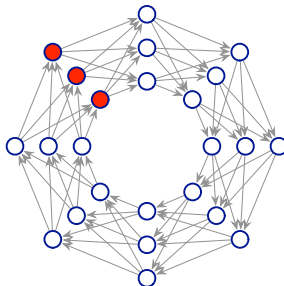
SYNFIRE RINGS

- *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*.



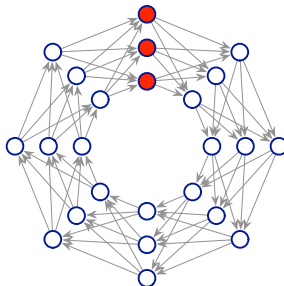
SYNFIRE RINGS

- *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*.



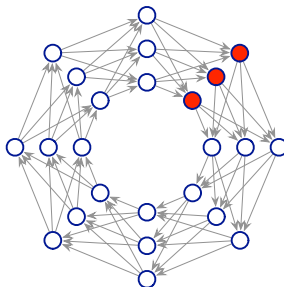
SYNFIRE RINGS

- *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*.



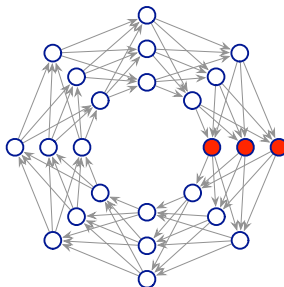
SYNFIRE RINGS

- *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*.



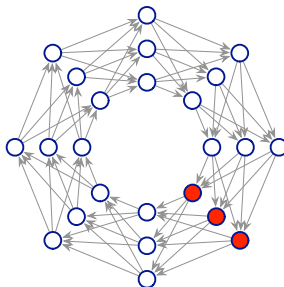
SYNFIRE RINGS

- *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*.



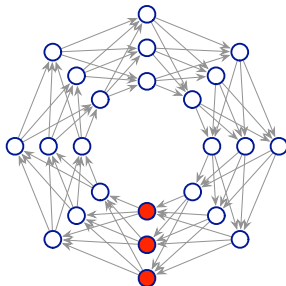
SYNFIRE RINGS

- *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*.



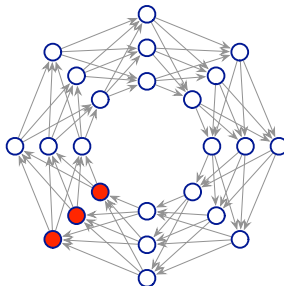
SYNFIRE RINGS

- *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*.



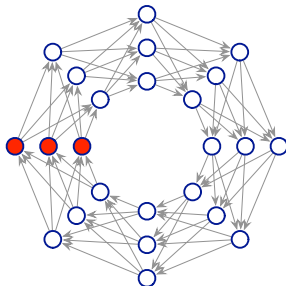
SYNFIRE RINGS

- *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*.

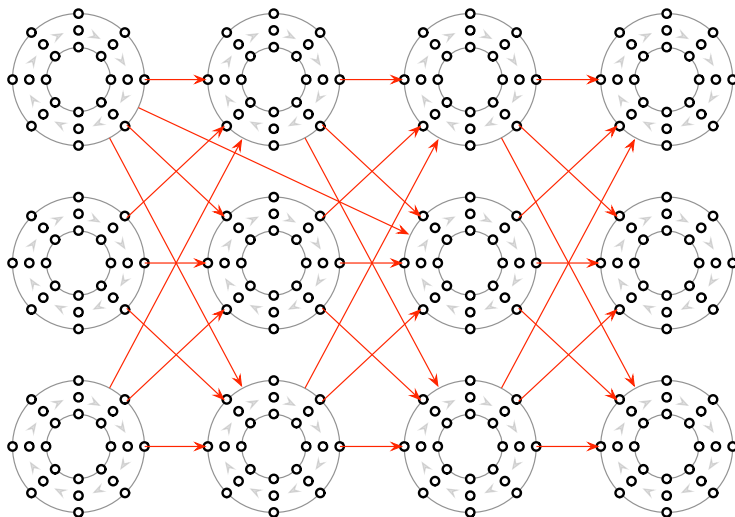


SYNFIRE RINGS

- *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 RING ARCHITECTURE



NEURAL COMPUTATION WITH SYNFIRE RINGS

- ▶ We introduce a paradigm of abstract neural computation based on *synfire rings*.
- ▶ Computational states are represented by sustained activities of synfire rings – i.e., 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 synaptic noises.

NEURAL COMPUTATION WITH SYNFIRE RINGS

- ▶ We introduce a paradigm of abstract neural computation based on *synfire rings*.
- ▶ Computational states are represented by sustained activities of synfire rings – i.e., 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 synaptic noises.

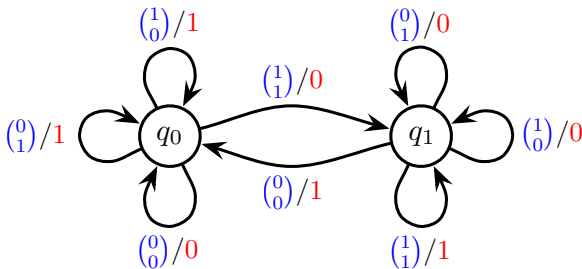
NEURAL COMPUTATION WITH SYNFIRE RINGS

- ▶ We introduce a paradigm of abstract neural computation based on *synfire rings*.
- ▶ Computational states are represented by sustained activities of synfire rings – i.e., 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 synaptic noises.

NEURAL COMPUTATION WITH SYNFIRE RINGS

- ▶ We introduce a paradigm of abstract neural computation based on *synfire rings*.
- ▶ Computational states are represented by sustained activities of synfire rings – i.e., 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 synaptic noises.

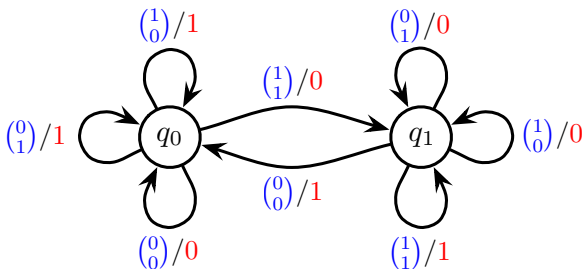
BINARY ADDER AUTOMATON



$$\begin{array}{r}
 \\
 \\
 \\
 \\
 \\
 \\
 \\
 \hline

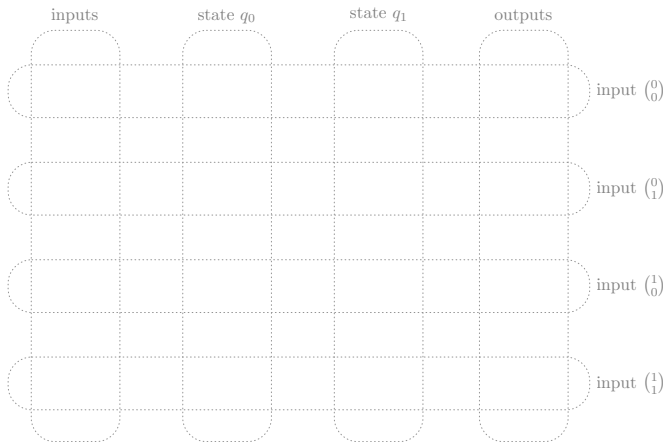
 \end{array}$$

BINARY ADDER AUTOMATON

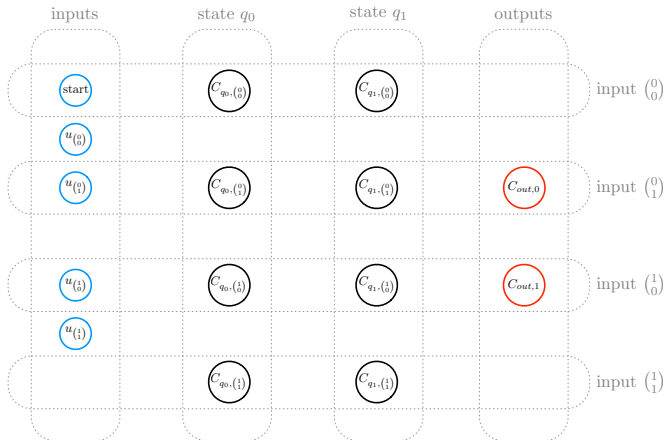


$$\begin{array}{rcccccccc}
 & & 1 & 1^1 & 0^1 & 1 & 1 & 0^1 & 1 \\
 + & & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
 \hline
 & & 1 & 1 & 0 & 0 & 1 & 1 & 0
 \end{array}$$

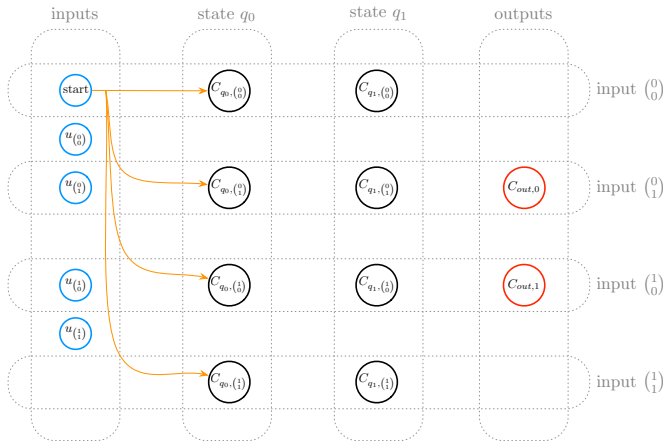
GENERAL CONSTRUCTION



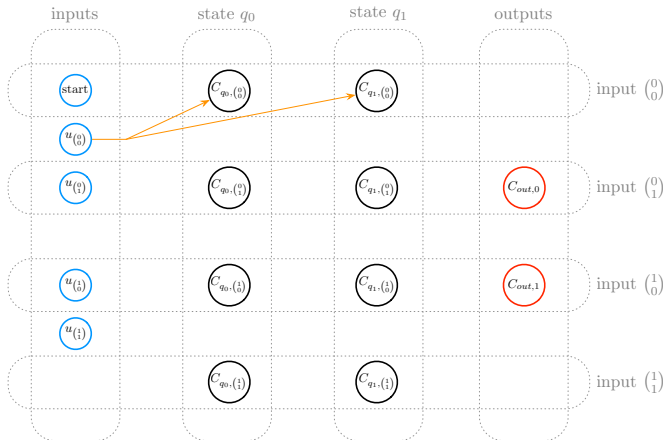
GENERAL CONSTRUCTION



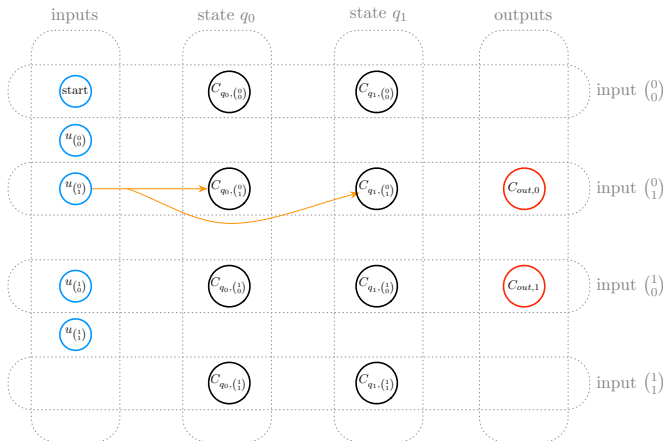
GENERAL CONSTRUCTION



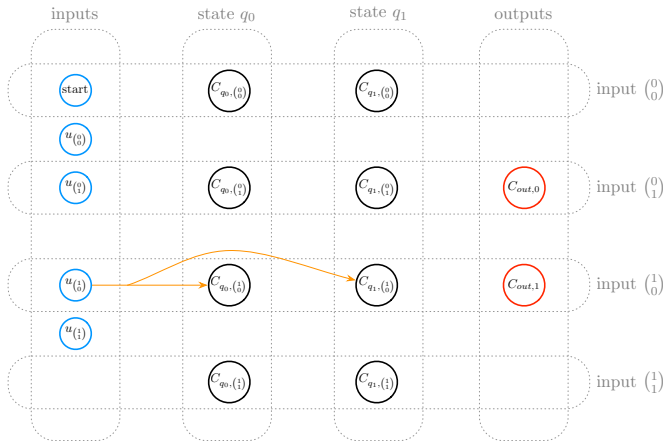
GENERAL CONSTRUCTION



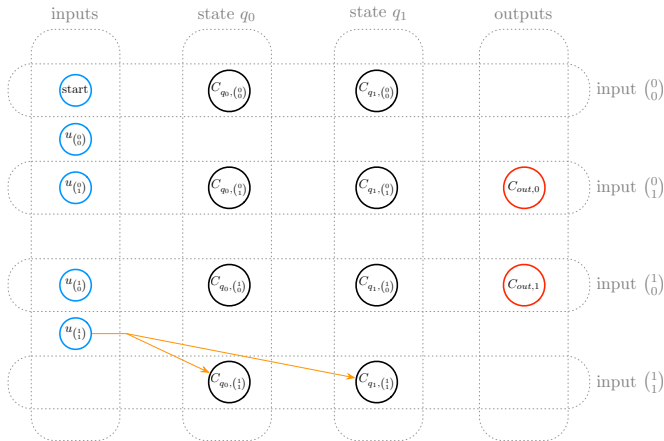
GENERAL CONSTRUCTION



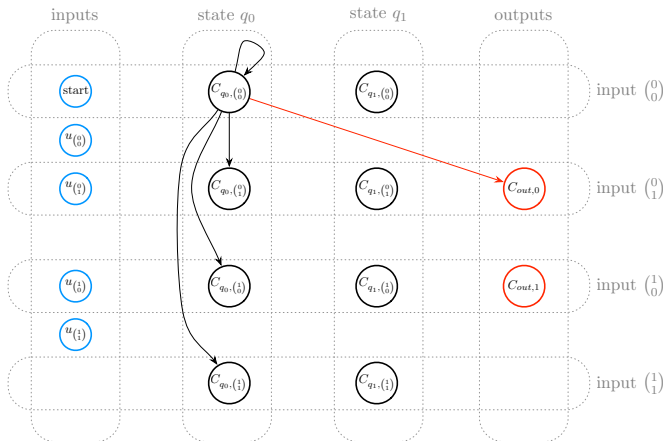
GENERAL CONSTRUCTION



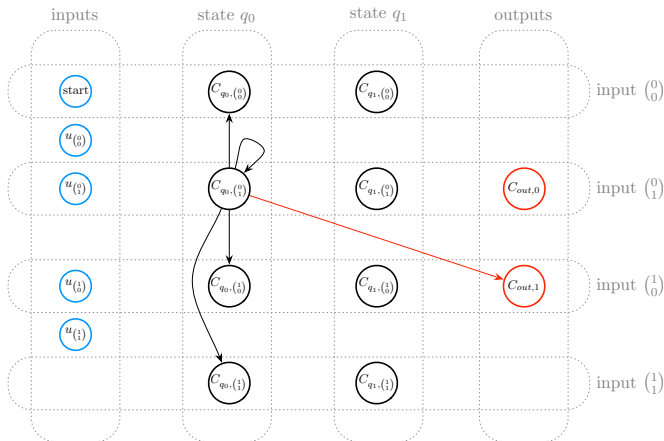
GENERAL CONSTRUCTION



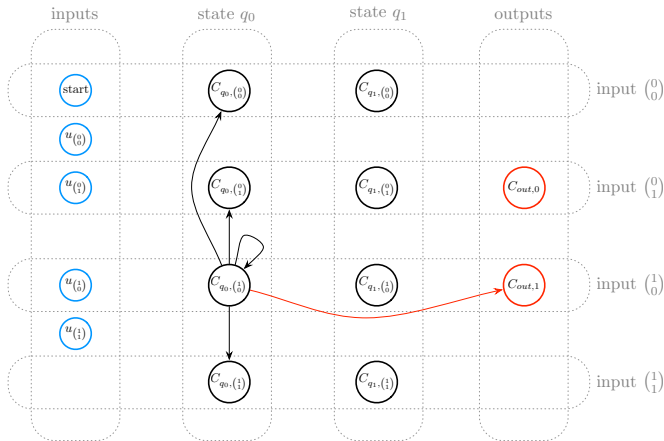
GENERAL CONSTRUCTION



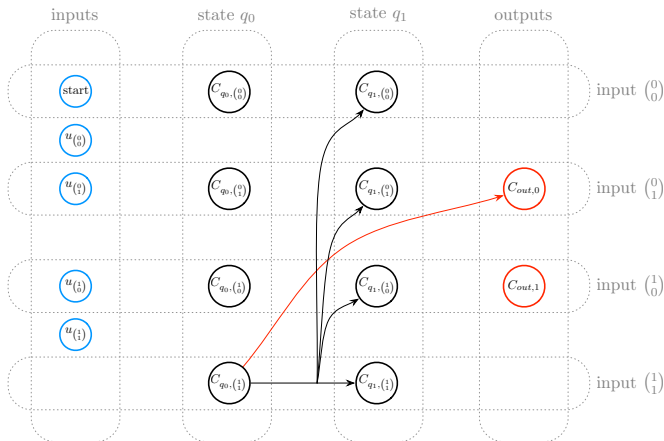
GENERAL CONSTRUCTION



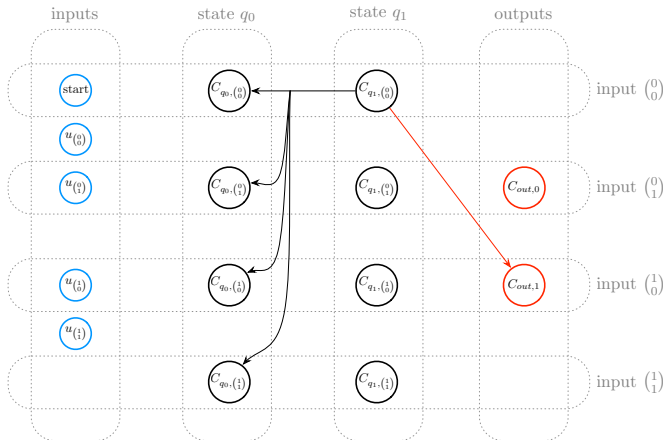
GENERAL CONSTRUCTION



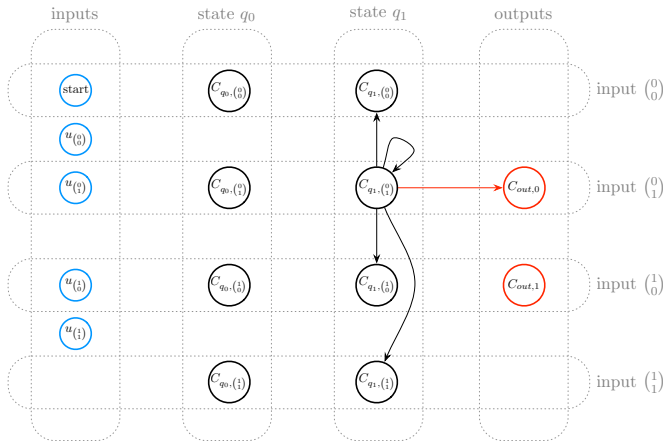
GENERAL CONSTRUCTION



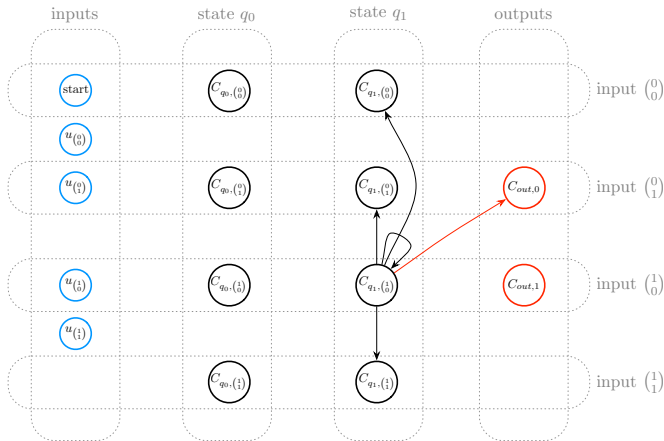
GENERAL CONSTRUCTION



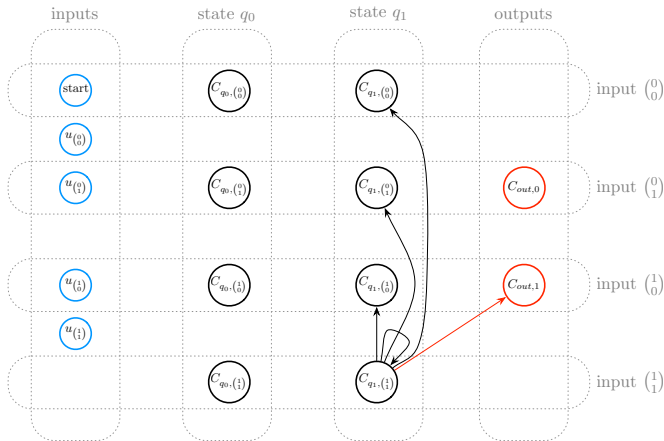
GENERAL CONSTRUCTION



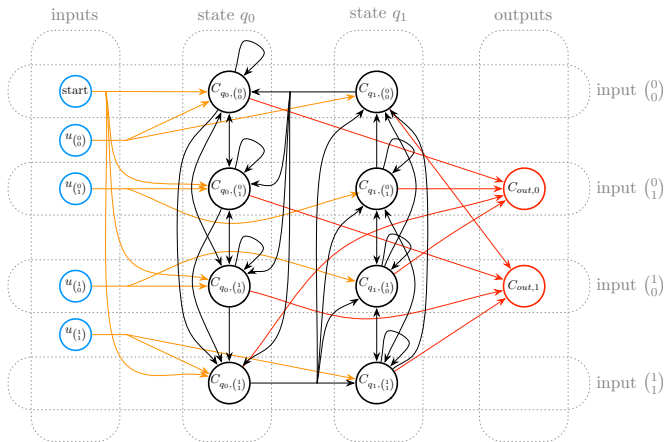
GENERAL CONSTRUCTION



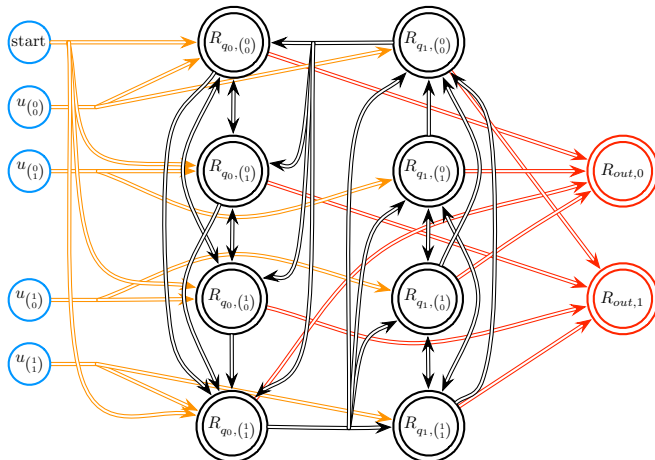
GENERAL CONSTRUCTION



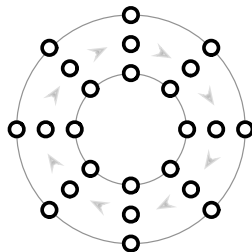
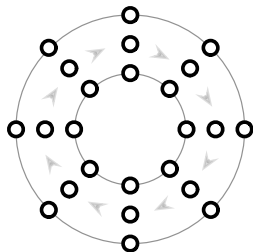
GENERAL CONSTRUCTION



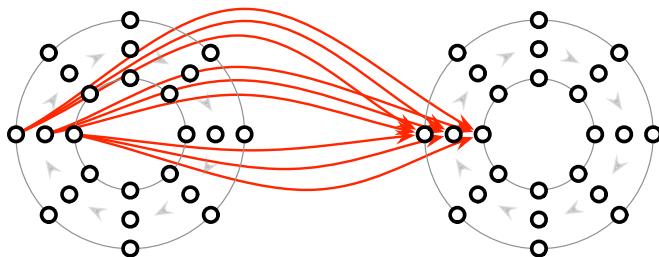
GENERAL CONSTRUCTION



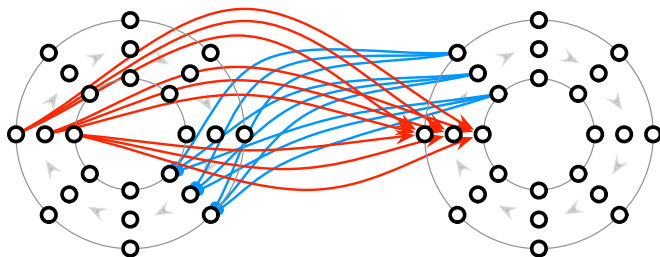
RING TRANSITION



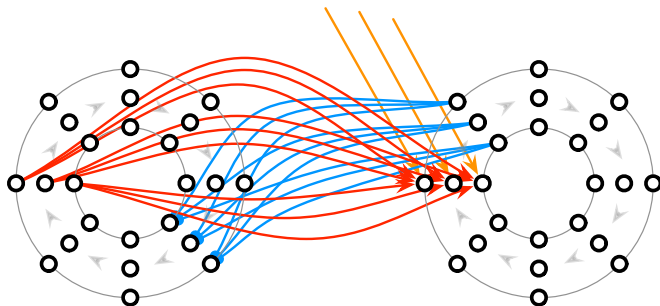
RING TRANSITION



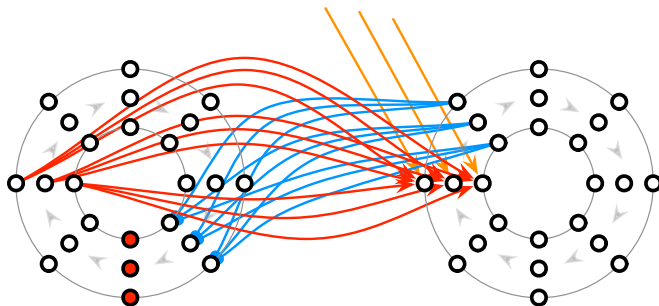
RING TRANSITION



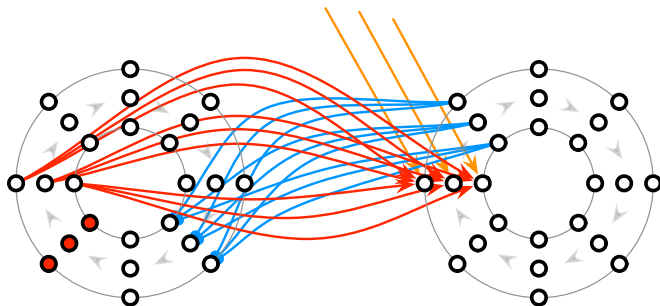
RING TRANSITION



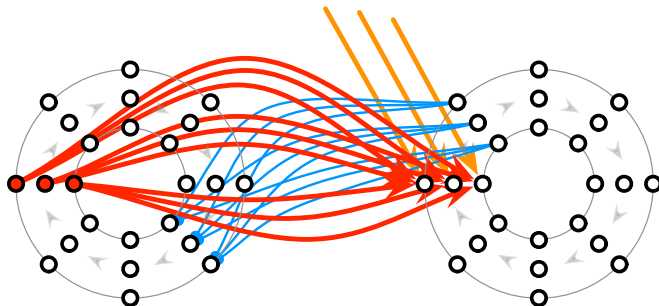
RING TRANSITION



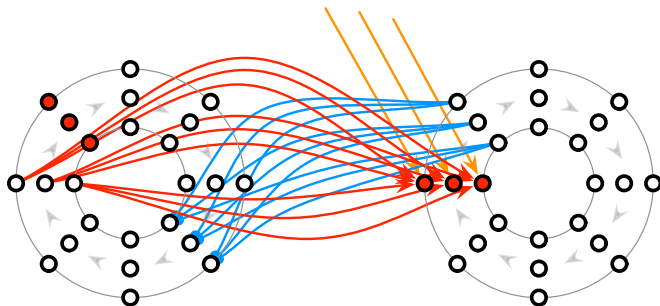
RING TRANSITION



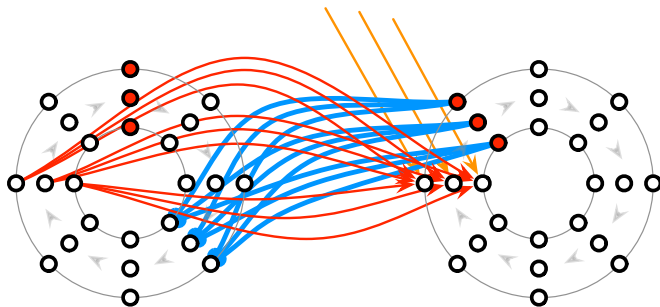
RING TRANSITION



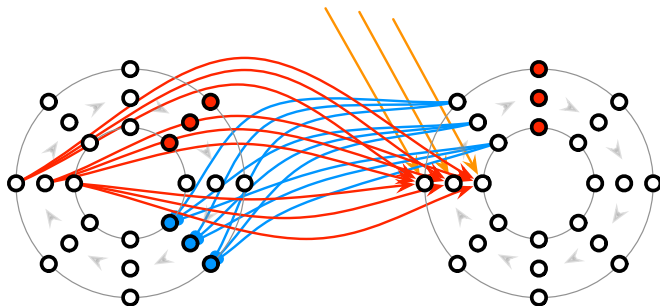
RING TRANSITION



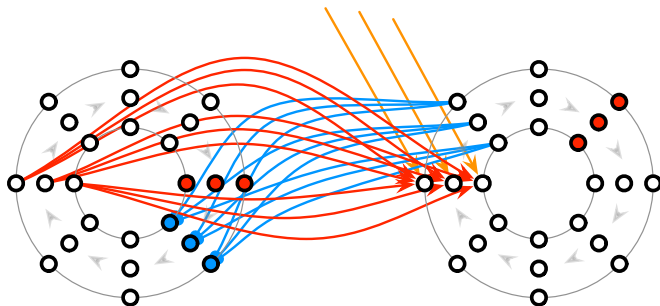
RING TRANSITION



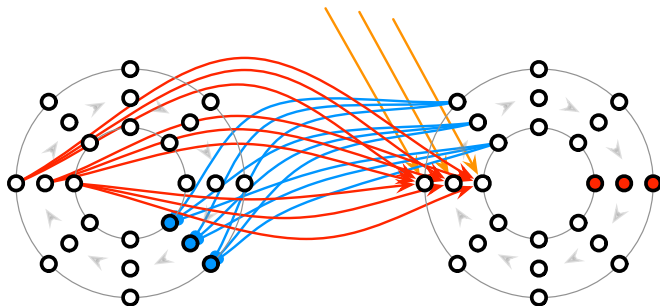
RING TRANSITION



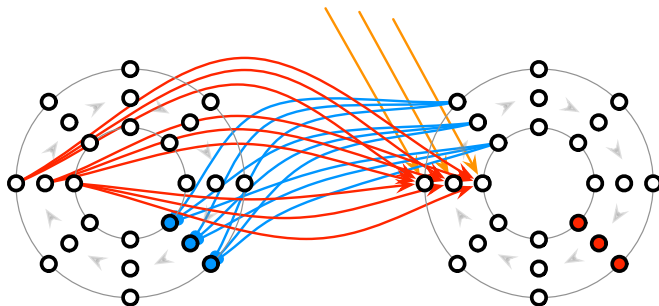
RING TRANSITION



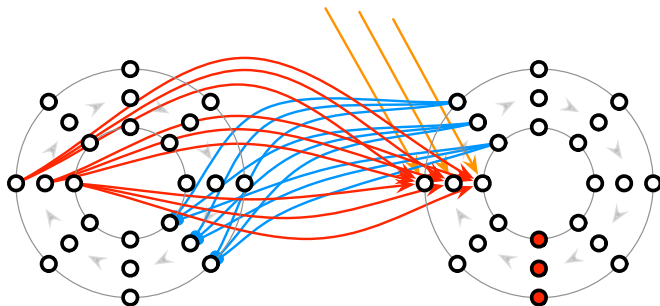
RING TRANSITION



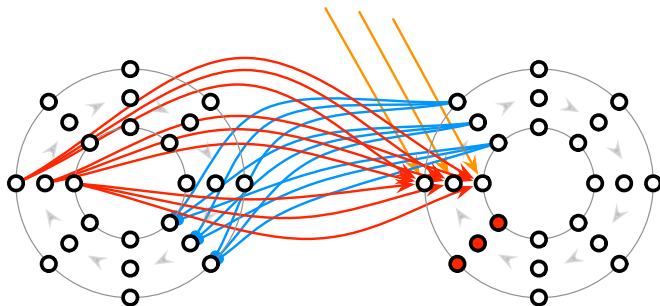
RING TRANSITION



RING TRANSITION



RING TRANSITION



AUTOMATA & BOOLEAN, IZHIKEVICH AND HODGKIN-HUXLEY RNNs WITH SYNFIRE RINGS

Play movie...

AUTOMATA & HODGKIN-HUXLEY RNNs WITH SYNFIRE RINGS

The construction is generic, therefore, the following results hold:

THEOREM (CABESSA & MASULLI 17, CABESSA ET AL. 17, CABESSA & TCHAPTCHET 18)

- ▶ *Any finite state automaton can be simulated by some **Boolean** neural network composed of synfire rings.*
- ▶ *Any finite state automaton can be simulated by some (noisy) **Izhikevich** spiking neural network composed of synfire rings.*
- ▶ *Any finite state automaton can be simulated by some **Hodgkin-Huxley** spiking neural network composed of synfire rings.*

AUTOMATA & HODGKIN-HUXLEY RNNs WITH SYNFIRE RINGS

The construction is generic, therefore, the following results hold:

THEOREM (CABESSA & MASULLI 17, CABESSA ET AL. 17, CABESSA & TCHAPTCHET 18)

- ▶ *Any finite state automaton can be simulated by some **Boolean** neural network composed of synfire rings.*
- ▶ *Any finite state automaton can be simulated by some (noisy) **Izhikevich** spiking neural network composed of synfire rings.*
- ▶ *Any finite state automaton can be simulated by some **Hodgkin-Huxley** spiking neural network composed of synfire rings.*

AUTOMATA & HODGKIN-HUXLEY RNNs WITH SYNFIRE RINGS

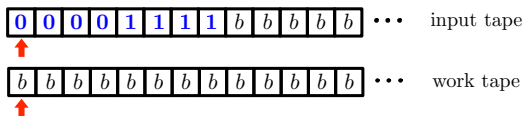
The construction is generic, therefore, the following results hold:

THEOREM (CABESSA & MASULLI 17, CABESSA ET AL. 17, CABESSA & TCHAPTCHET 18)

- ▶ *Any finite state automaton can be simulated by some **Boolean** neural network composed of synfire rings.*
- ▶ *Any finite state automaton can be simulated by some (noisy) **Izhikevich** spiking neural network composed of synfire rings.*
- ▶ *Any finite state automaton can be simulated by some **Hodgkin-Huxley** spiking neural network composed of synfire rings.*

TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$

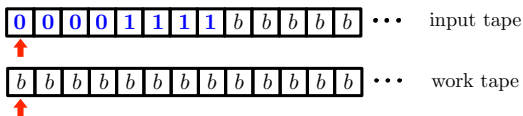


Program *current state: q_{in}*

$(q_{in}, b, b) \mapsto (q_{acc}, b, b, S, S)$	
$(q_{in}, 0, b) \mapsto (q_0, 0, b, R, S)$	
$(q_{in}, 1, b) \mapsto (q_{rej}, 1, b, S, S)$	
$(q_0, 0, b) \mapsto (q_{0bis}, 0, 0, R, R)$	$(q_{bis}, 0, b) \mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b) \mapsto (q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b) \mapsto (q_1, 1, 1, R, L)$
$(q_0, b, b) \mapsto (q_{rej}, b, b, S, S)$	$(q_{0bis}, b, b) \mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0) \mapsto (q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0) \mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1) \mapsto (q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1) \mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0) \mapsto (q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0) \mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$

TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$

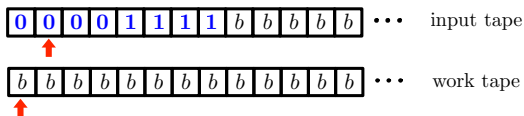


Program *current state: q_{in}*

(q_{in}, b, b)	\mapsto	(q_{acc}, b, b, S, S)		
$(q_{in}, 0, b)$	\mapsto	$(q_0, 0, b, R, S)$		
$(q_{in}, 1, b)$	\mapsto	$(q_{rej}, 1, b, S, S)$		
$(q_0, 0, b)$	\mapsto	$(q_{0bis}, 0, 0, R, R)$	$(q_{bis}, 0, b)$	\mapsto $(q_0, 0, 0, R, R)$
$(q_0, 1, b)$	\mapsto	$(q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b)$	\mapsto $(q_1, 1, 1, R, L)$
(q_0, b, b)	\mapsto	(q_{rej}, b, b, S, S)	(q_{0bis}, b, b)	\mapsto (q_{rej}, b, b, S, S)
$(q_1, 1, 0)$	\mapsto	$(q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0)$	\mapsto $(q_1, 1, 1, R, L)$
$(q_1, b, 1)$	\mapsto	$(q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1)$	\mapsto $(q_{acc}, b, 1, S, S)$
$(q_1, b, 0)$	\mapsto	$(q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0)$	\mapsto $(q_{rej}, b, 0, S, S)$
$(q_1, 1, 1)$	\mapsto	$(q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1)$	\mapsto $(q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1)$	\mapsto	$(q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1)$	\mapsto $(q_{rej}, 0, 1, S, S)$

TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$

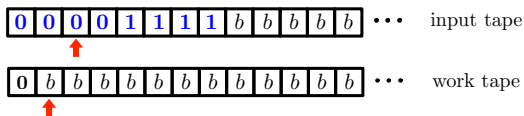


Program *current state:* q_0

$(q_{in}, b, b) \mapsto (q_{acc}, b, b, S, S)$	
$(q_{in}, 0, b) \mapsto (q_0, 0, b, R, S)$	
$(q_{in}, 1, b) \mapsto (q_{rej}, 1, b, S, S)$	
$(q_0, 0, b) \mapsto (q_{bis}, 0, 0, R, R)$	$(q_{bis}, 0, b) \mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b) \mapsto (q_1, 1, 1, R, L)$	$(q_{bis}, 1, b) \mapsto (q_1, 1, 1, R, L)$
$(q_0, b, b) \mapsto (q_{rej}, b, b, S, S)$	$(q_{bis}, b, b) \mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0) \mapsto (q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0) \mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1) \mapsto (q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1) \mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0) \mapsto (q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0) \mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$

TURING MACHINE

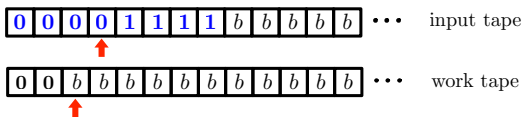
Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$



Program <i>current state:</i> q_{0bis}			
$(q_{in}, b, b) \mapsto$	(q_{acc}, b, b, S, S)		
$(q_{in}, 0, b) \mapsto$	$(q_0, 0, b, R, S)$		
$(q_{in}, 1, b) \mapsto$	$(q_{rej}, 1, b, S, S)$		
$(q_0, 0, b) \mapsto$	$(q_{0bis}, 0, 0, R, R)$	$(q_{0bis}, 0, b) \mapsto$	$(q_0, 0, 0, R, R)$
$(q_0, 1, b) \mapsto$	$(q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b) \mapsto$	$(q_1, 1, 1, R, L)$
$(q_0, b, b) \mapsto$	(q_{rej}, b, b, S, S)	$(q_{0bis}, b, b) \mapsto$	(q_{rej}, b, b, S, S)
$(q_1, 1, 0) \mapsto$	$(q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0) \mapsto$	$(q_1, 1, 1, R, L)$
$(q_1, b, 1) \mapsto$	$(q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1) \mapsto$	$(q_{acc}, b, 1, S, S)$
$(q_1, b, 0) \mapsto$	$(q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0) \mapsto$	$(q_{rej}, b, 0, S, S)$
$(q_1, 1, 1) \mapsto$	$(q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1) \mapsto$	$(q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1) \mapsto$	$(q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1) \mapsto$	$(q_{rej}, 0, 1, S, S)$

TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$

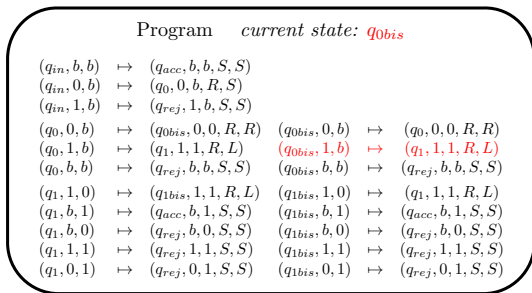
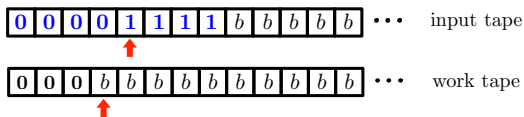


Program *current state: q_0*

$(q_{in}, b, b) \mapsto (q_{acc}, b, b, S, S)$	
$(q_{in}, 0, b) \mapsto (q_0, 0, b, R, S)$	
$(q_{in}, 1, b) \mapsto (q_{rej}, 1, b, S, S)$	
$(q_0, 0, b) \mapsto (q_{bis}, 0, 0, R, R)$	$(q_{bis}, 0, b) \mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b) \mapsto (q_1, 1, 1, R, L)$	$(q_{bis}, 1, b) \mapsto (q_1, 1, 1, R, L)$
$(q_0, b, b) \mapsto (q_{rej}, b, b, S, S)$	$(q_{bis}, b, b) \mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0) \mapsto (q_{bis}, 1, 1, R, L)$	$(q_{bis}, 1, 0) \mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1) \mapsto (q_{acc}, b, 1, S, S)$	$(q_{bis}, b, 1) \mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0) \mapsto (q_{rej}, b, 0, S, S)$	$(q_{bis}, b, 0) \mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$	$(q_{bis}, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$	$(q_{bis}, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$

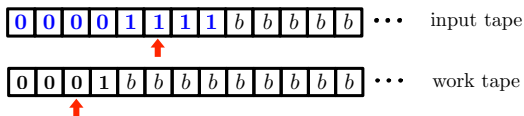
TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$



TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$

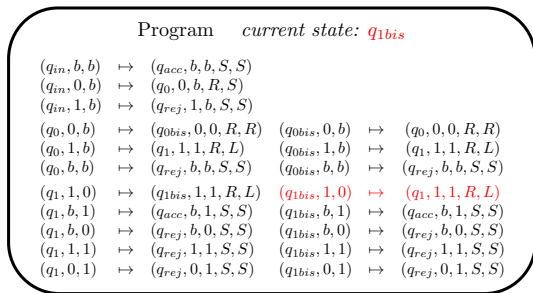
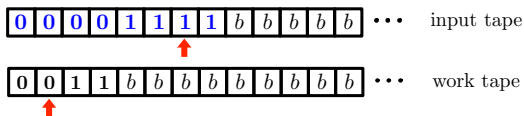


Program *current state:* q_1

$(q_{in}, b, b) \mapsto (q_{acc}, b, b, S, S)$	
$(q_{in}, 0, b) \mapsto (q_0, 0, b, R, S)$	
$(q_{in}, 1, b) \mapsto (q_{rej}, 1, b, S, S)$	
$(q_0, 0, b) \mapsto (q_{0bis}, 0, 0, R, R)$	$(q_{0bis}, 0, b) \mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b) \mapsto (q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b) \mapsto (q_1, 1, 1, R, L)$
$(q_0, b, b) \mapsto (q_{rej}, b, b, S, S)$	$(q_{0bis}, b, b) \mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0) \mapsto (q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0) \mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1) \mapsto (q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1) \mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0) \mapsto (q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0) \mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$

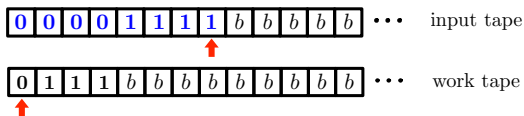
TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$



TURING MACHINE

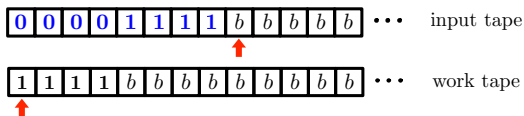
Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$



Program current state: q_1			
(q_{in}, b, b)	\mapsto	(q_{acc}, b, b, S, S)	
$(q_{in}, 0, b)$	\mapsto	$(q_0, 0, b, R, S)$	
$(q_{in}, 1, b)$	\mapsto	$(q_{rej}, 1, b, S, S)$	
$(q_0, 0, b)$	\mapsto	$(q_{0bis}, 0, 0, R, R)$	$(q_{0bis}, 0, b) \mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b)$	\mapsto	$(q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b) \mapsto (q_1, 1, 1, R, L)$
(q_0, b, b)	\mapsto	(q_{rej}, b, b, S, S)	$(q_{0bis}, b, b) \mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0)$	\mapsto	$(q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0) \mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1)$	\mapsto	$(q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1) \mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0)$	\mapsto	$(q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0) \mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1)$	\mapsto	$(q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1) \mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1)$	\mapsto	$(q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1) \mapsto (q_{rej}, 0, 1, S, S)$

TURING MACHINE

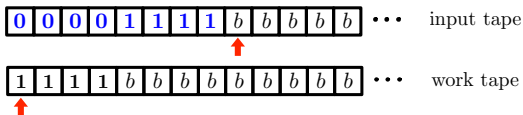
Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$



Program		current state: q_{1bis}	
(q_{in}, b, b)	$\mapsto (q_{acc}, b, b, S, S)$		
$(q_{in}, 0, b)$	$\mapsto (q_0, 0, b, R, S)$		
$(q_{in}, 1, b)$	$\mapsto (q_{rej}, 1, b, S, S)$		
$(q_0, 0, b)$	$\mapsto (q_{0bis}, 0, 0, R, R)$	$(q_{0bis}, 0, b)$	$\mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b)$	$\mapsto (q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b)$	$\mapsto (q_1, 1, 1, R, L)$
(q_0, b, b)	$\mapsto (q_{rej}, b, b, S, S)$	(q_{0bis}, b, b)	$\mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0)$	$\mapsto (q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0)$	$\mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1)$	$\mapsto (q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1)$	$\mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0)$	$\mapsto (q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0)$	$\mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1)$	$\mapsto (q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1)$	$\mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1)$	$\mapsto (q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1)$	$\mapsto (q_{rej}, 0, 1, S, S)$

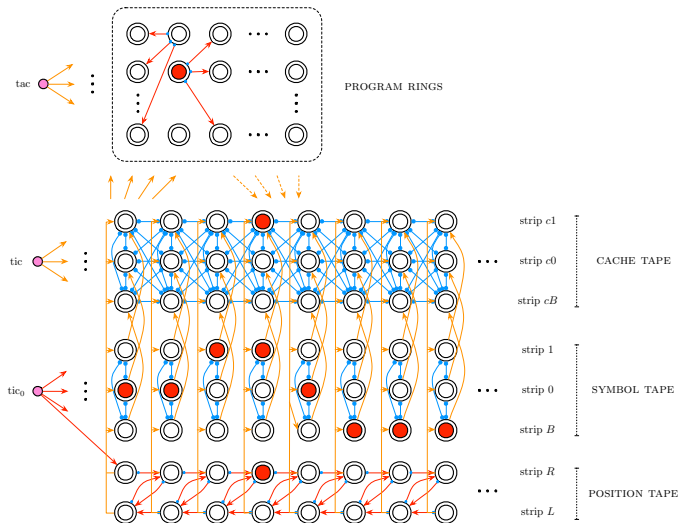
TURING MACHINE

Turing machine recognizing the non regular and non context-free language $\{0^n 1^n : n \geq 0\}$



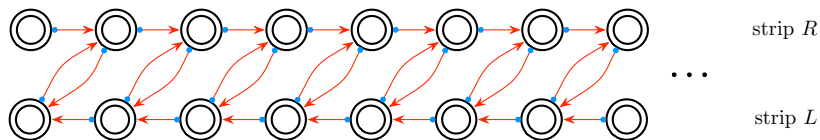
Program		current state: q_{acc}	
(q_{in}, b, b)	$\mapsto (q_{acc}, b, b, S, S)$		
$(q_{in}, 0, b)$	$\mapsto (q_0, 0, b, R, S)$		
$(q_{in}, 1, b)$	$\mapsto (q_{rej}, 1, b, S, S)$		
$(q_0, 0, b)$	$\mapsto (q_{0bis}, 0, 0, R, R)$	$(q_{0bis}, 0, b)$	$\mapsto (q_0, 0, 0, R, R)$
$(q_0, 1, b)$	$\mapsto (q_1, 1, 1, R, L)$	$(q_{0bis}, 1, b)$	$\mapsto (q_1, 1, 1, R, L)$
(q_0, b, b)	$\mapsto (q_{rej}, b, b, S, S)$	(q_{0bis}, b, b)	$\mapsto (q_{rej}, b, b, S, S)$
$(q_1, 1, 0)$	$\mapsto (q_{1bis}, 1, 1, R, L)$	$(q_{1bis}, 1, 0)$	$\mapsto (q_1, 1, 1, R, L)$
$(q_1, b, 1)$	$\mapsto (q_{acc}, b, 1, S, S)$	$(q_{1bis}, b, 1)$	$\mapsto (q_{acc}, b, 1, S, S)$
$(q_1, b, 0)$	$\mapsto (q_{rej}, b, 0, S, S)$	$(q_{1bis}, b, 0)$	$\mapsto (q_{rej}, b, 0, S, S)$
$(q_1, 1, 1)$	$\mapsto (q_{rej}, 1, 1, S, S)$	$(q_{1bis}, 1, 1)$	$\mapsto (q_{rej}, 1, 1, S, S)$
$(q_1, 0, 1)$	$\mapsto (q_{rej}, 0, 1, S, S)$	$(q_{1bis}, 0, 1)$	$\mapsto (q_{rej}, 0, 1, S, S)$

GENERAL CONSTRUCTION



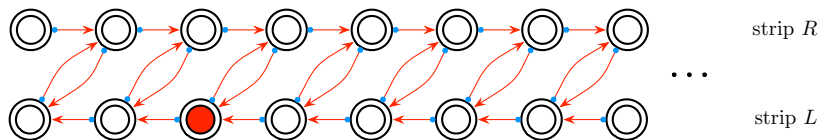
POSITION TAPE

- Stores current head's position.



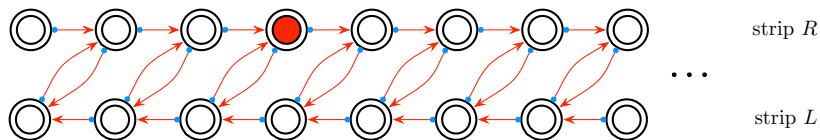
POSITION TAPE

- Stores current head's position.



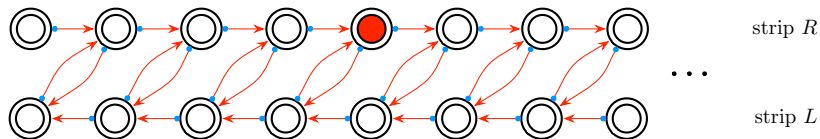
POSITION TAPE

- Stores current head's position.



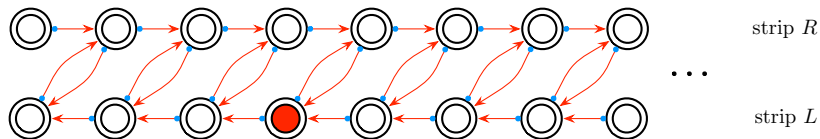
POSITION TAPE

- Stores current head's position.



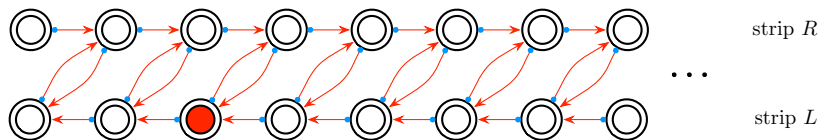
POSITION TAPE

- Stores current head's position.



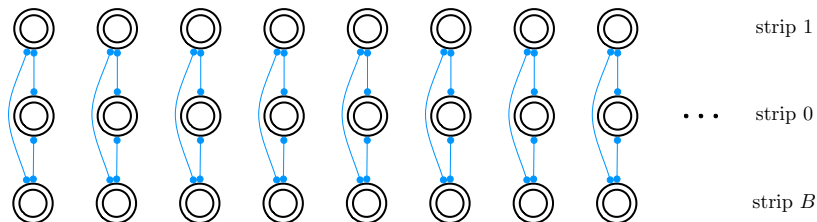
POSITION TAPE

- Stores current head's position.



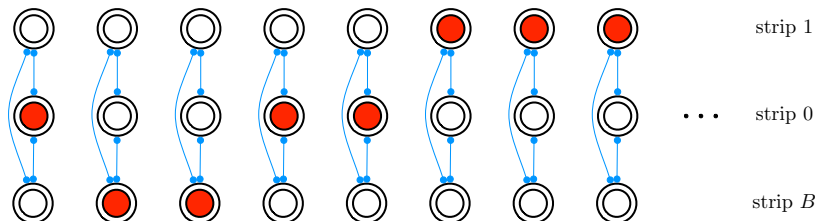
SYMBOL TAPE

- Stores symbols written on the tape.



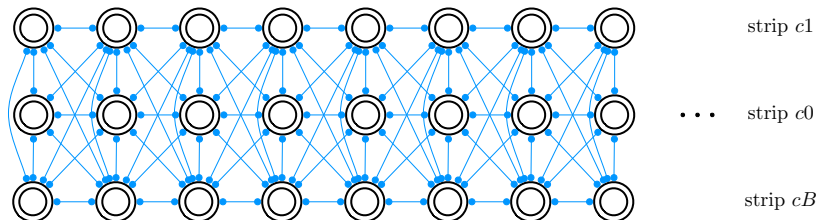
SYMBOL TAPE

- Stores symbols written on the tape.



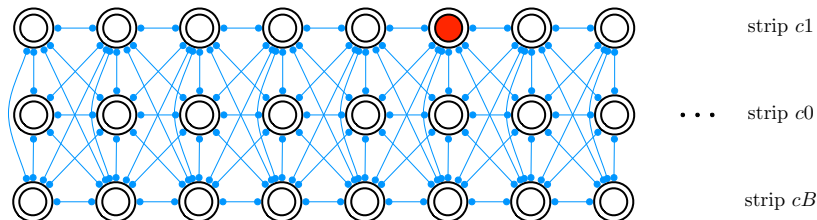
CACHE TAPE

- Store symbol under the current head's position.

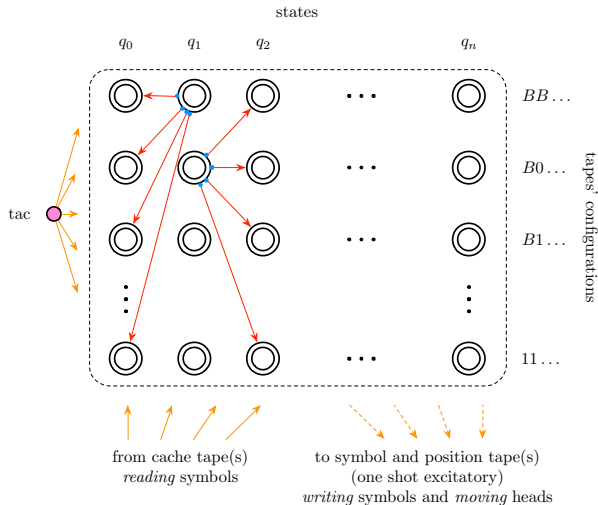


CACHE TAPE

- Store symbol under the current head's position.



PROGRAM RINGS



TURING MACHINES & BOOLEAN RNNs WITH SYNFIRE RINGS

Play movie...

TURING MACHINES & BOOLEAN RNNs WITH SYNFIRE RINGS

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

THEOREM

Any Turing machine can be simulated by some Boolean neural network composed of synfire rings (up to assuming that the number of rings can be extended in an unbounded manner).

FUTURE WORK

1. Generalize the Turing-complete synfire ring architecture to Izhikevich and Hodgkin-Huxley recurrent neural networks.
2. Develop learning algorithms on this architecture (phase 2):
 - 2.1. Gradient descent-based algorithms
 - 2.2. Hebbian learning-based algorithms
 - 2.3. STDP-based algorithms

FUTURE WORK

1. Generalize the Turing-complete synfire ring architecture to Izhikevich and Hodgkin-Huxley recurrent neural networks.
2. Develop learning algorithms on this architecture (phase 2):
 - 2.1 Gradient descent-based algorithms
 - 2.2 Evolutionary-based algorithms
 - 2.3 STDP-based algorithms

FUTURE WORK

1. Generalize the Turing-complete synfire ring architecture to Izhikevich and Hodgkin-Huxley recurrent neural networks.
2. Develop learning algorithms on this architecture (phase 2):
 - 2.1 Gradient descent-based algorithms
 - 2.2 Evolutionary-based algorithms
 - 2.3 STDP-based algorithms

FUTURE WORK

1. Generalize the Turing-complete synfire ring architecture to Izhikevich and Hodgkin-Huxley recurrent neural networks.
2. Develop learning algorithms on this architecture (phase 2):
 - 2.1 Gradient descent-based algorithms
 - 2.2 Evolutionary-based algorithms
 - 2.3 STDP-based algorithms

FUTURE WORK

1. Generalize the Turing-complete synfire ring architecture to Izhikevich and Hodgkin-Huxley recurrent neural networks.
2. Develop learning algorithms on this architecture (phase 2):
 - 2.1 Gradient descent-based algorithms
 - 2.2 Evolutionary-based algorithms
 - 2.3 STDP-based algorithms

CONCLUSIONS

- What is lifelong learning? What is bio-inspired learning?

Classical Machine Learning

external intervention
of the modeller
exogenous

Bio-Inspired

internal biological rules
evolution, STDP, etc.
endogenous

- Bio-inspired learning: emergent collective property of internal biological rules.
- Towards more bio-inspired learning learning...
- Long-term goal (utopia?): towards neuronal computers...

Thank you!

CONCLUSIONS

- ▶ What is lifelong learning? What is bio-inspired learning?

Classical Machine Learning

external intervention
of the modeller
exogenous

Bio-Inspired

internal biological rules
evolution, STDP, etc.
endogenous

- ▶ Bio-inspired learning: emergent collective property of internal biological rules.
- ▶ Towards more bio-inspired learning learning...
- ▶ Long-term goal (utopia?): towards neuronal computers...

Thank you!

CONCLUSIONS

- What is lifelong learning? What is bio-inspired learning?

Classical Machine Learning

external intervention
of the modeller
exogenous

Bio-Inspired

internal biological rules
evolution, STDP, etc.
endogenous

- Bio-inspired learning: emergent collective property of internal biological rules.
- Towards more bio-inspired learning learning...
- Long-term goal (utopia?): towards neuronal computers...

Thank you!

