Finite State Machines and Bio-Inspired Neural Networks

Jérémie Cabessa

Laboratoire d'économie mathématique
Université Paris II, France

April 26, 2022, LISN, Université Paris-Saclay, France

Introduction

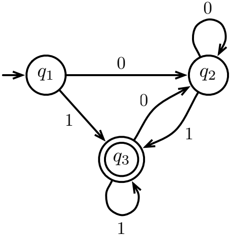
- **Part 1:** We recall some results about the computational capabilities of recurrent neural networks.
- **Part 2:** We introduce a bio-inspired paradigm for neural computation.

Introduction

- **Part 1:** We recall some results about the computational capabilities of recurrent neural networks.
- **Part 2:** We introduce a bio-inspired paradigm for neural computation.

Finite State Automaton (FSA)

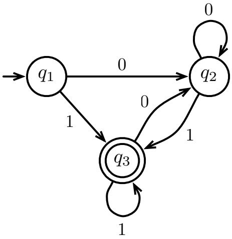
Graph composed of computational states (nodes) and transitions between those (edges).



- Input u is *accepted (rejected)* by $\mathcal{A}$ if $\mathcal{A}(u)$ ends up in a final (non-final) state.
- **Regular languages (REG)**: class of languages recognizable by finite state automata.

Finite State Automaton (FSA)

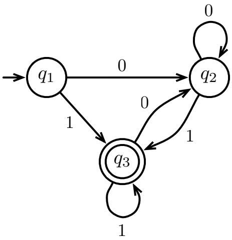
Graph composed of computational states (nodes) and transitions between those (edges).



- Input u is *accepted (rejected)* by $\mathcal{A}$ if $\mathcal{A}(u)$ ends up in a final (non-final) state.
- **Regular languages (REG)**: class of languages recognizable by finite state automata.

Finite State Automaton (FSA)

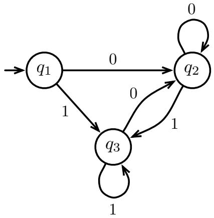
Graph composed of computational states (nodes) and transitions between those (edges).



- Input u is *accepted (rejected)* by $\mathcal{A}$ if $\mathcal{A}(u)$ ends up in a final (non-final) state.
- **Regular languages (REG)**: class of languages recognizable by finite state automata.

Muller Automaton (infinite words)

FSA with Muller acceptance condition: collection of sets of states.

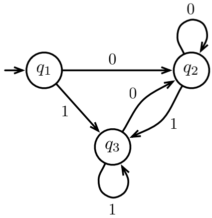


$$\mathcal{T} = \{\{q_2\}, \{q_2, q_3\}\}$$

- Infinite word u *accepted (rejected)* by $\mathcal{A}$ if the infinite run $\mathcal{A}(u)$ satisfies $\inf(\rho_u) \in \mathcal{T}$ $(\inf(\rho_u) \notin \mathcal{T})$.
- **ω-regular languages (ω-REG)**: class of ω-languages recognizable by Muller (Büchi, Rabin, Streett, parity) automata.

Muller Automaton (infinite words)

FSA with Muller acceptance condition: collection of sets of states.

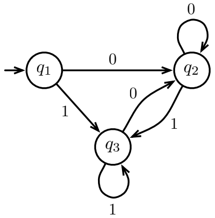


$$\mathcal{T} = \{\{q_2\}, \{q_2, q_3\}\}$$

- Infinite word u *accepted (rejected)* by $\mathcal{A}$ if the infinite run $\mathcal{A}(u)$ satisfies $\inf(\rho_u) \in \mathcal{T}$ ($\inf(\rho_u) \notin \mathcal{T}$).
- **ω-regular languages (ω-REG)**: class of ω-languages recognizable by Muller (Büchi, Rabin, Streett, parity) automata.

Muller Automaton (infinite words)

FSA with Muller acceptance condition: collection of sets of states.

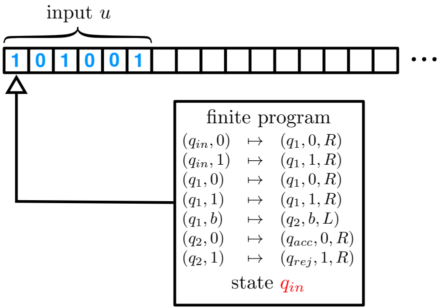


$$\mathcal{T} = \{\{q_2\}, \{q_2, q_3\}\}$$

- Infinite word u *accepted (rejected)* by $\mathcal{A}$ if the infinite run $\mathcal{A}(u)$ satisfies $\inf(\rho_u) \in \mathcal{T}$ ($\inf(\rho_u) \notin \mathcal{T}$).
- **ω-regular languages (ω-REG)**: class of ω-languages recognizable by Muller (Büchi, Rabin, Streett, parity) automata.

Turing Machine (TM)

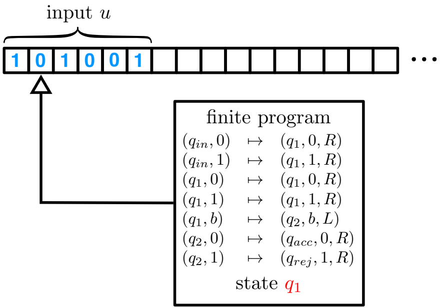
Infinite tape, read-write head, and finite program.



- *u accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- P (NP): class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

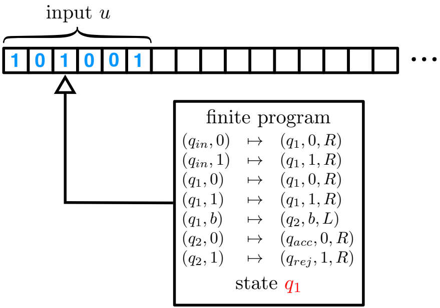
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

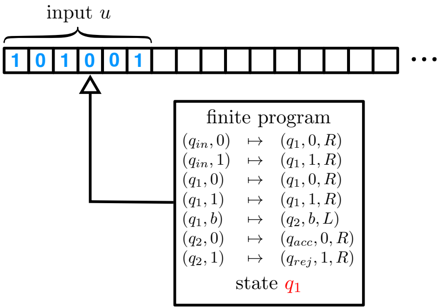
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

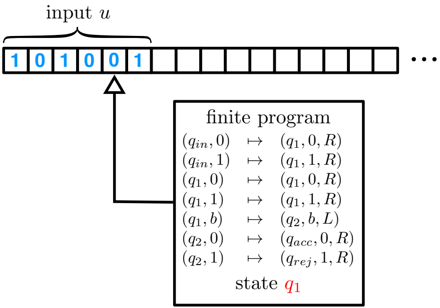
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

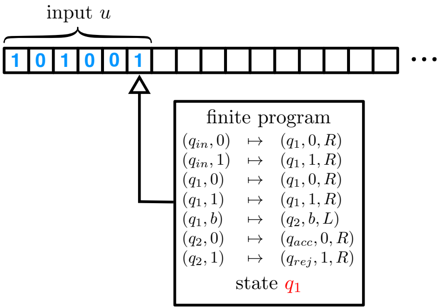
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

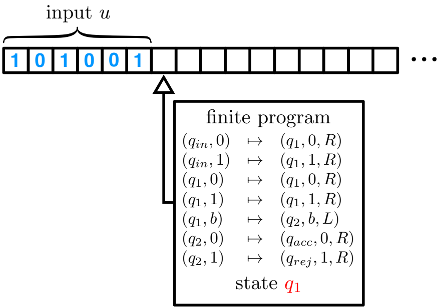
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

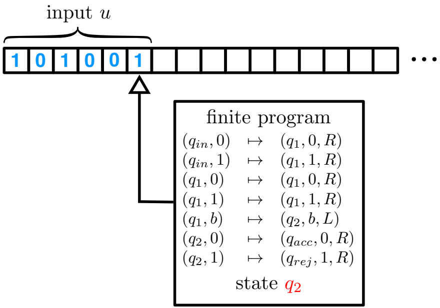
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

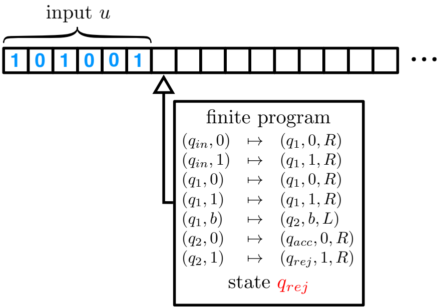
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

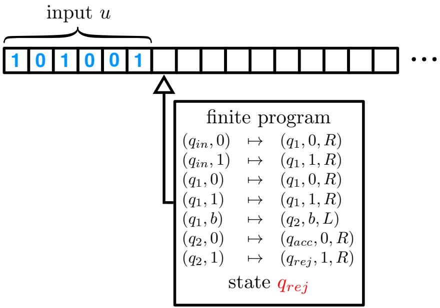
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine (TM)

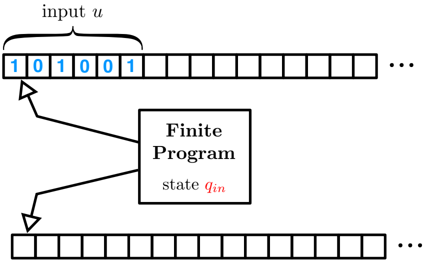
Infinite tape, read-write head, and finite program.



- u *accepted (rejected)* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches state q_{acc} (q_{rej}).
- **P (NP)**: class of languages decidable in poly time by deterministic (non-deterministic) Turing machines.

Turing Machine with Advice (TM/A)

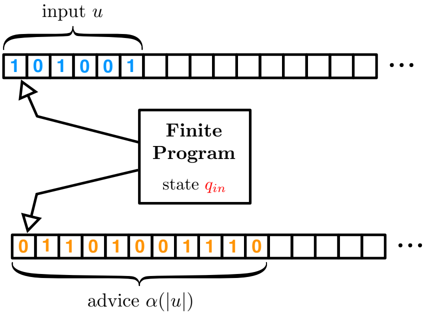
TM with additional tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



- **P/poly**: class of languages decidable in polynomial time by Turing machines with poly long advices (TM/poly(A)).
- P/poly $\supsetneq$ P : TM/poly(A)s are "super-Turing".

Turing Machine with Advice (TM/A)

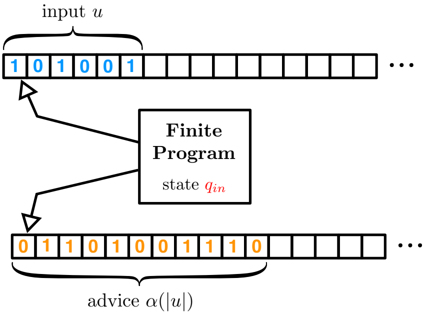
TM with additional tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



- **P/poly**: class of languages decidable in polynomial time by Turing machines with poly long advices (TM/poly(A)).
- **P/poly** $\supsetneq$ **P**. TM/poly(A)s are "super-Turing"...

Turing Machine with Advice (TM/A)

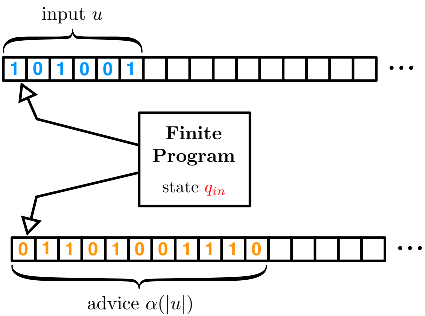
TM with additional tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



- **P/poly**: class of languages decidable in polynomial time by Turing machines with poly long advices (TM/poly(A)).
- **P/poly** $\supsetneq$ **P**. TM/poly(A)s are "super-Turing"...

Turing Machine with Advice (TM/A)

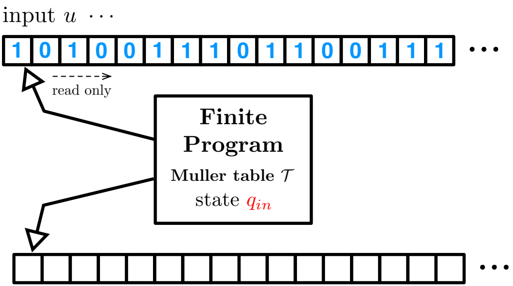
TM with additional tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



- **P/poly**: class of languages decidable in polynomial time by Turing machines with poly long advices (TM/poly(A)).
- **P/poly** $\supsetneq$ **P**. TM/poly(A)s are "super-Turing"...

Muller Turing Machine (infinite words)

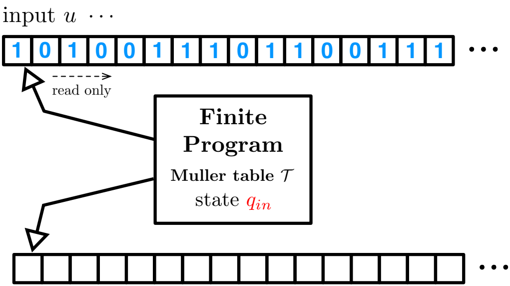
TM with Muller acceptance condition (table $\mathcal{T}$).



- u *accepted (rejected)* by $\mathcal{M}$ if $\inf(\rho_u) \in \mathcal{T}$ ($\inf(\rho_u) \notin \mathcal{T}$).
- $BC(\Pi^0_2)$: class of ω-languages recognizable by Muller Turing machines.

Muller Turing Machine (infinite words)

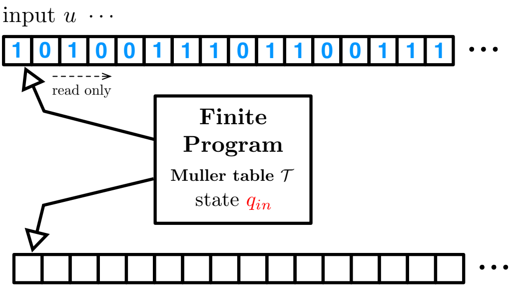
TM with Muller acceptance condition (table $\mathcal{T}$).



- u *accepted (rejected)* by $\mathcal{M}$ if $\inf(\rho_u) \in \mathcal{T}$ ($\inf(\rho_u) \notin \mathcal{T}$).
- $BC(\Pi_2^0)$: class of ω-languages recognizable by Muller Turing machines.

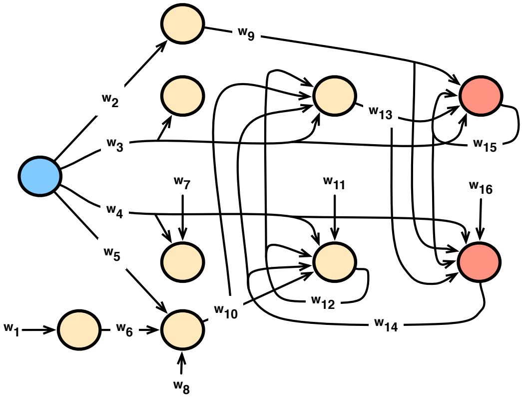
Muller Turing Machine (infinite words)

TM with Muller acceptance condition (table $\mathcal{T}$).



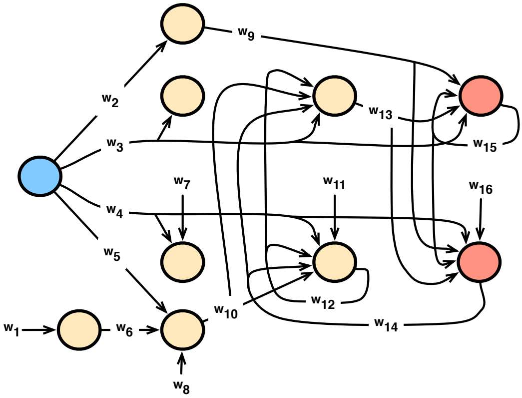
- u *accepted (rejected)* by $\mathcal{M}$ if $\inf(\rho_u) \in \mathcal{T}$ ($\inf(\rho_u) \notin \mathcal{T}$).
- $BC(\Pi_2^0)$: class of ω-languages recognizable by Muller Turing machines.

Recurrent Neural Network (RNN)



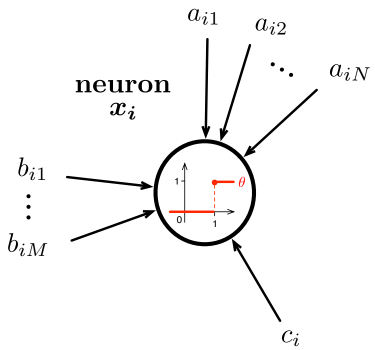
- Recurrence brings *memory*. . .

Recurrent Neural Network (RNN)



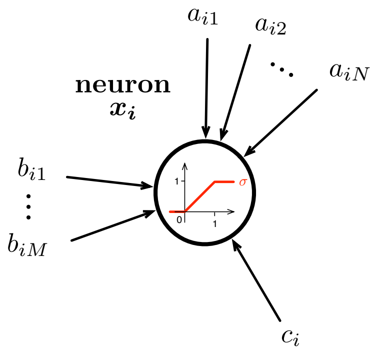
- Recurrence brings *memory*. . . .

Boolean Recurrent Neural Network



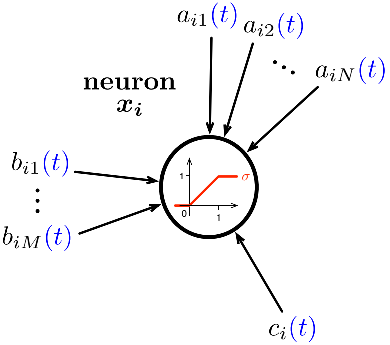
$$x_i(t+1) = \color{red}{\theta}\color{black}{\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i \right)}$$

Sigmoidal Recurrent Neural Network



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

Evolving Recurrent Neural Network



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

Results: classical computation

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca & Si 11,14	Ca & Si 11,14

Results: classical computation

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca & Si 11,14	Ca & Si 11,14

Results: classical computation

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca & Si 11,14	Ca & Si 11,14

Results: classical computation

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca & Si 11,14	Ca & Si 11,14

Results: classical computation

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca & Si 11,14	Ca & Si 11,14

Results: infinite computation / det.

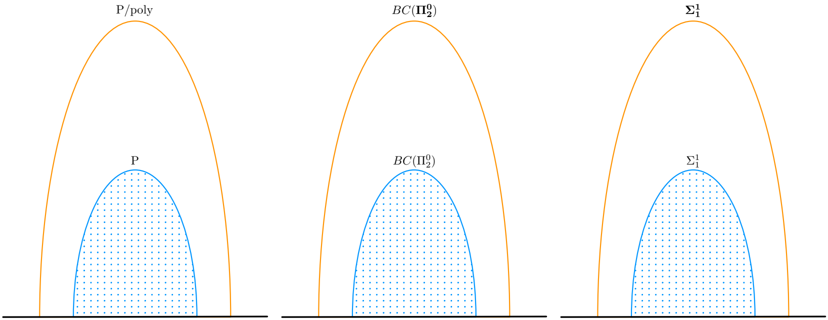
	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	Muller FSA $\in BC(\Pi_2^0)$	Muller TM $= BC(\Pi_2^0)$	super-Turing $= BC(\mathbf{\Pi_2^0})$	super-Turing $= BC(\mathbf{\Pi_2^0})$
$\mathbb{R}$	Muller FSA $\in BC(\Pi_2^0)$	super-Turing $= BC(\mathbf{\Pi_2^0})$	super-Turing $= BC(\mathbf{\Pi_2^0})$	super-Turing $= BC(\mathbf{\Pi_2^0})$

Results: infinite computation / nondet.

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	Muller FSA $\in \Sigma^1_1$	Muller TM $= \Sigma^1_1$	super-Turing $= \boldsymbol{\Sigma}^1_1$	super-Turing $= \boldsymbol{\Sigma}^1_1$
$\mathbb{R}$	Muller FSA $\in \Sigma^1_1$	super-Turing $= \boldsymbol{\Sigma}^1_1$	super-Turing $= \boldsymbol{\Sigma}^1_1$	super-Turing $= \boldsymbol{\Sigma}^1_1$

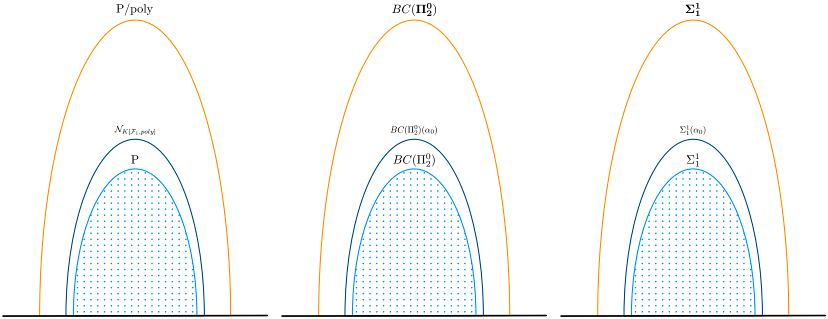
Hierarchy Theorems

- Stratifying the “super-Turing world”.
- Finite and infinite computation: we can define infinitely many complexity classes between the Turing and super-Turing levels.



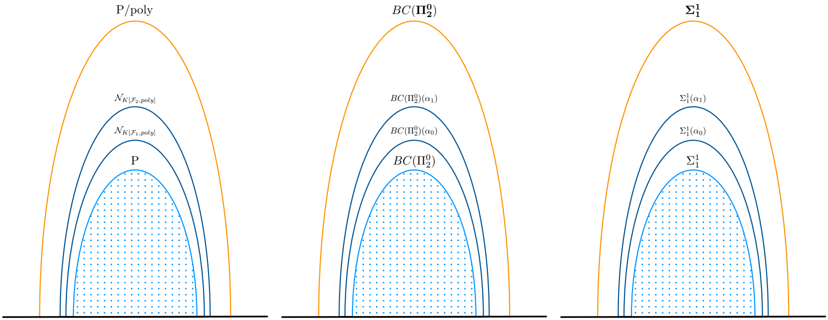
Hierarchy Theorems

- Stratifying the "super-Turing world".
- Finite and infinite computation: we can define infinitely many complexity classes between the Turing and super-Turing levels.



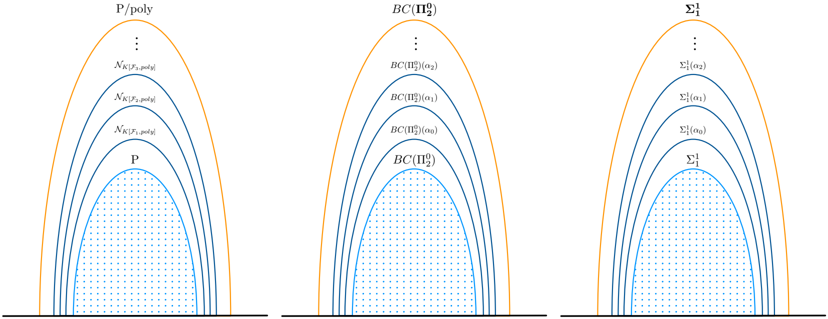
Hierarchy Theorems

- Stratifying the "super-Turing world".
- Finite and infinite computation: we can define infinitely many complexity classes between the Turing and super-Turing levels.



Hierarchy Theorems

- Stratifying the “super-Turing world”.
- Finite and infinite computation: we can define infinitely many complexity classes between the Turing and super-Turing levels.



Intermediate Conclusions

- Recurrent neural networks is a natural model for *oracle-based (super-Turing) computation*.
- In all these results, the simulation of finite state machines by recurrent neural networks is not "biologically plausible".

Intermediate Conclusions

- Recurrent neural networks is a natural model for *oracle-based (super-Turing) computation*.
- In all these results, the simulation of finite state machines by recurrent neural networks is not "biologically plausible".

Limitations

- Computational states of the machines 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*.

Limitations

- Computational states of the machines 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*.

Limitations

- Computational states of the machines 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*.

Limitations

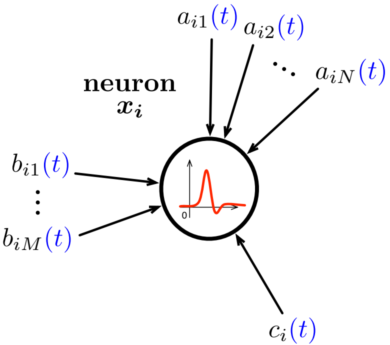
- Computational states of the machines are represented as Boolean states, i.e., spiking configurations of the network.

⋆ 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*.

Bio-Inspired Recurrent Neural Network



- Dynamics governed by the Hodgkin-Huxley equations.

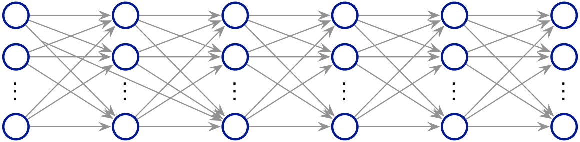
Hodgkin-Huxley Model

$$
\begin{aligned}
C \cdot \frac{dV}{dt} &= -I_L - I_{Na} - I_K - I_C - I_{input} \quad \text{where} \\
I_L &= g_L \cdot (V - V_L) \\
I_{Na} &= g_{Na} \cdot m \cdot h \cdot (V - V_{Na}) \\
& \quad \frac{dm}{dt} = \frac{m_\infty - m}{\tau_m} \qquad m_\infty = \frac{1}{1 + \mathrm{e}^{-s_m \cdot (V - V_{hm})}} \\
& \quad \frac{dh}{dt} = \frac{h_\infty - h}{\tau_h} \qquad h_\infty = 1 - \frac{1}{1 + \mathrm{e}^{-s_h \cdot (V - V_{hh})}} \\
I_K &= g_K \cdot n \cdot (V - V_K) \\
& \quad \frac{dn}{dt} = \frac{n_\infty - n}{\tau_n} \qquad n_\infty = \frac{1}{1 + \mathrm{e}^{-s_n \cdot (V - V_{hn})}} \\
I_C &= w_{intra}^{exc} + w_{inter}^{exc} + w_{inter}^{inh} + w_{output}^{exc} + w_{output}^{inh} \\
I_{input} &= a_{input}^{exc} \cdot \chi t_{input}(t)
\end{aligned}
$$

V: membrane potential; C membrane capacitance; I_L: leakage current; I_{Na}, I_K: sodium and the potassium fast currents; I_C: synaptic currents coming from the neighboring neurons; I_{input}: pulse-like input current. **(show simulator)**

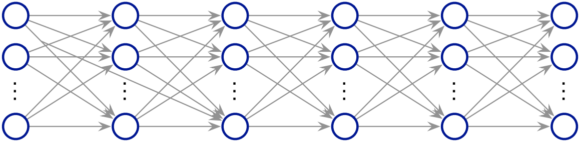
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



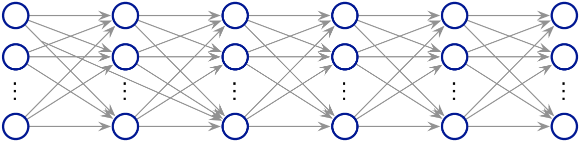
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



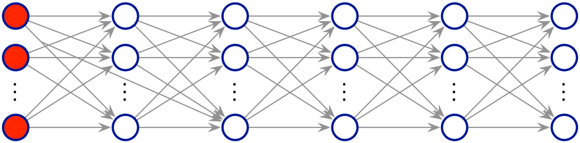
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



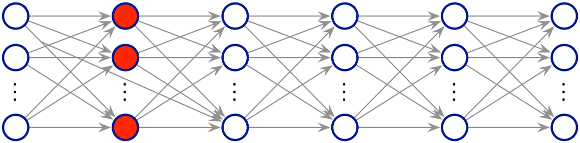
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



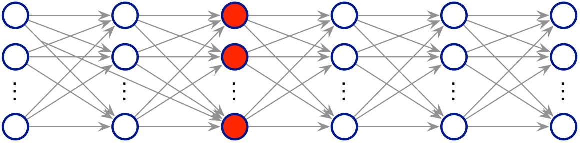
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



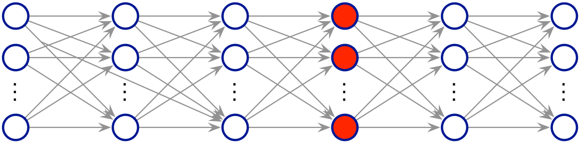
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



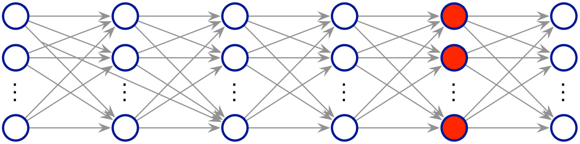
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



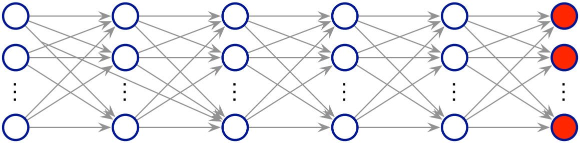
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



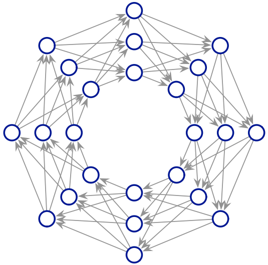
Synfire Chains

- *Synfire chains* have been theorized as fundamental neuronal structures (Abeles 82).
- They allow for robust and highly precise transmission of information in neural networks.



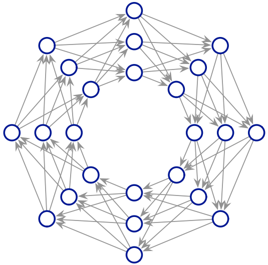
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or attractors; (2) synchronous dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.
- Computational states represented as sustained activities within synfire rings.



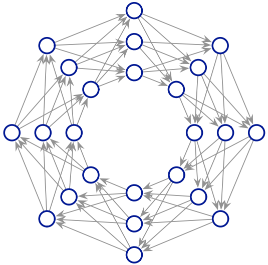
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.
- Computational states represented as sustained activities within synfire rings.



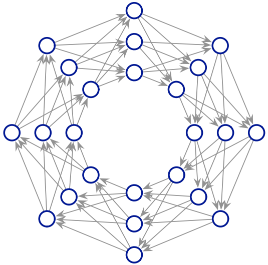
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.
- Computational states represented as sustained activities within synfire rings.



Synfire Rings

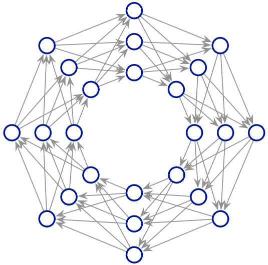
- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.
- Computational states represented as sustained activities within synfire rings.



Synfire Rings

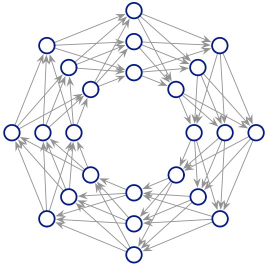
- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

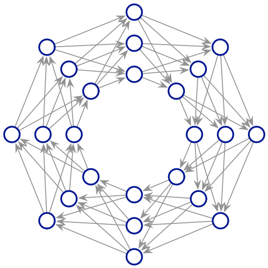
- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.
- Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

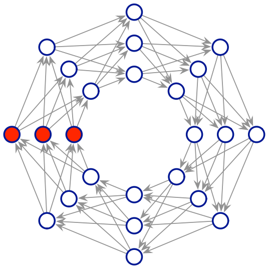
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

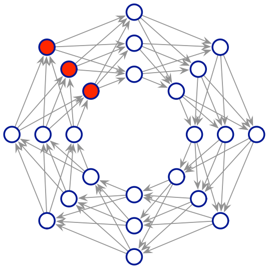
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

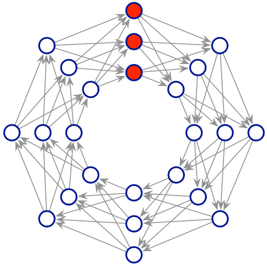
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

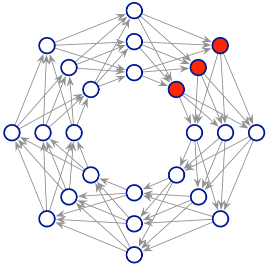
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

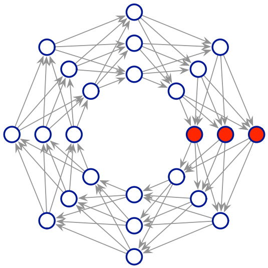
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

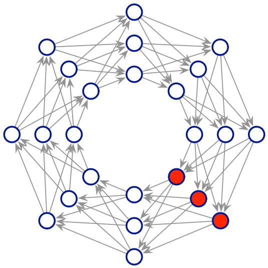
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

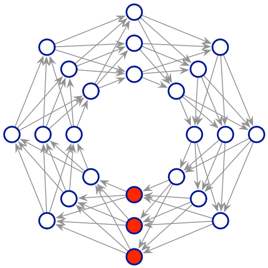
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

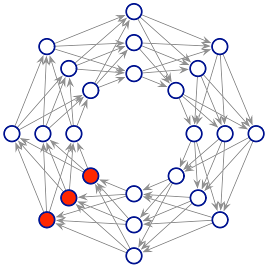
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

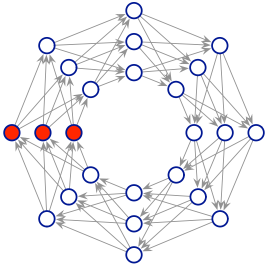
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

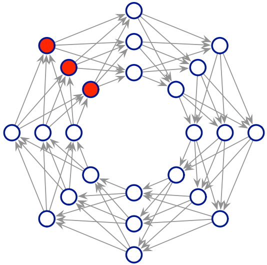
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

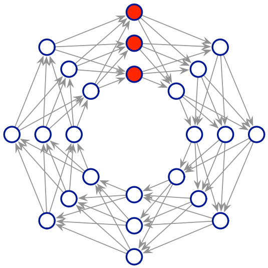
⋆ Computational states represented as sustained activities within synfire rings.



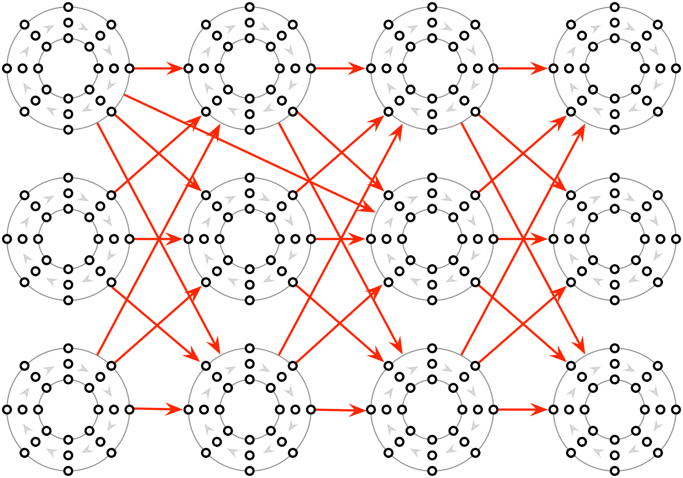
Synfire Rings

- *Synfire rings* are looping synfire chains that have been observed in self-organizing neural networks (Zheng & Triesch 14).
- Synfire rings induce: (1) self-sustained activities or *attractors*; (2) *synchronous* dynamics; (3) discrete temporal structure; (4) Robustness against synaptic failures.

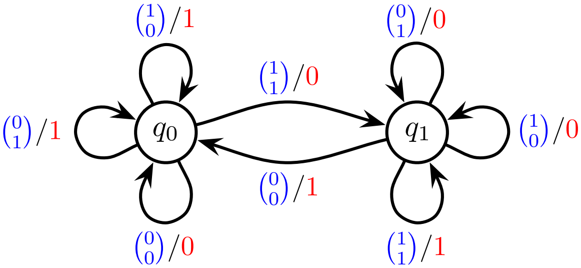
⋆ Computational states represented as sustained activities within synfire rings.



Synfire Ring Architecture

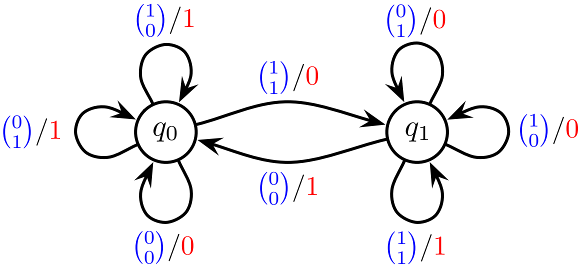


Binary Adder Automaton



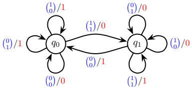
$$\begin{array}{cccccccc} & {}^{1} & 1^{1} & 0^{1} & 1 & 1 & 0^{1} & 1 \\ + & & 1 & 1 & 1 & 0 & 0 & 1 \\ \hline & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

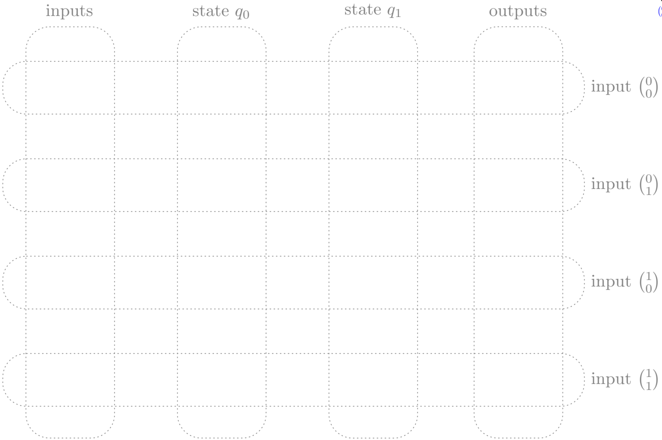
Binary Adder Automaton



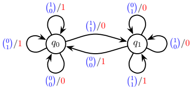
$$
\begin{array}{cccccccc}
 & {}^{1} & 1^{1} & 0^{1} & 1 & 1 & 0^{1} & 1 \\
+ & & 1 & 1 & 1 & 0 & 0 & 1 \\
\hline
 & 1 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}
$$

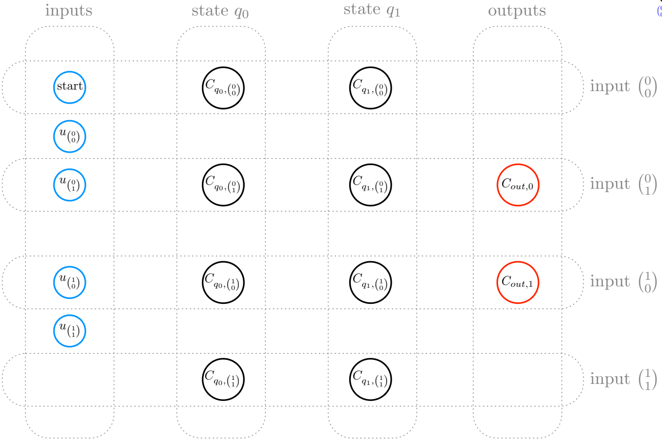
Binary Adder Boolean Neural Network



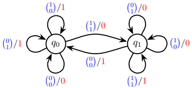


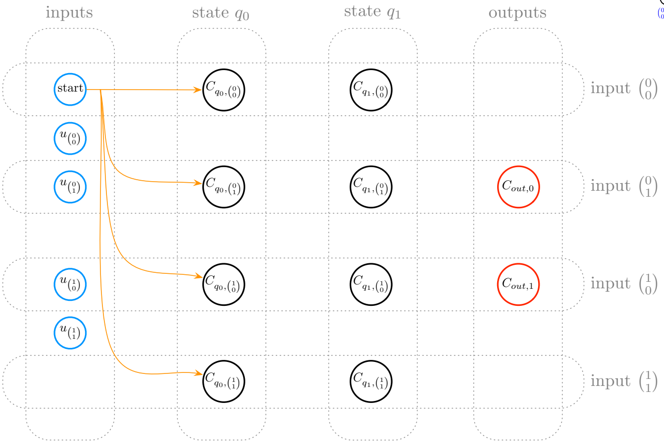
Binary Adder Boolean Neural Network



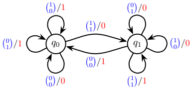


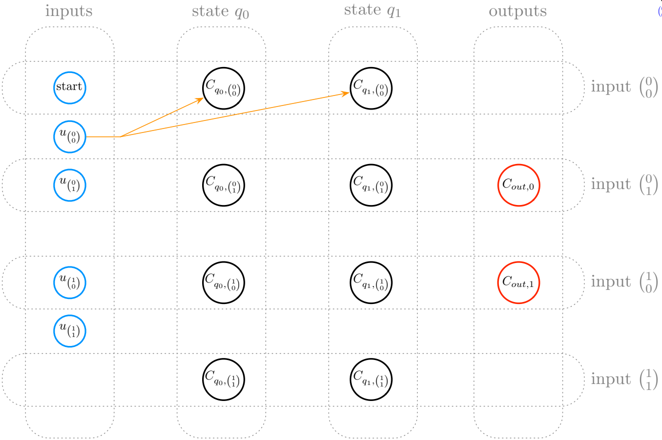
Binary Adder Boolean Neural Network



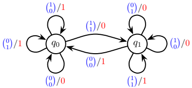


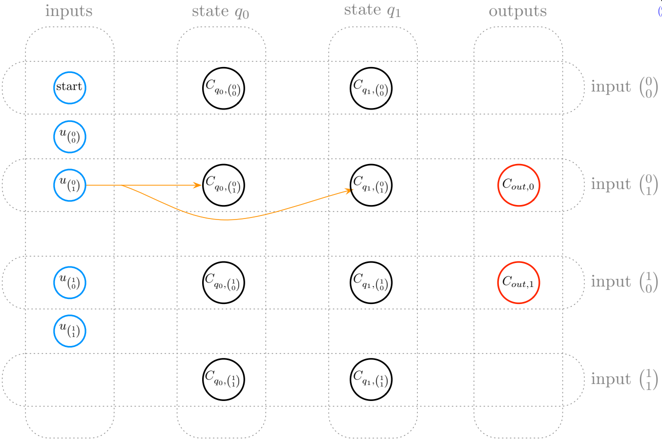
Binary Adder Boolean Neural Network



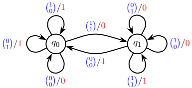


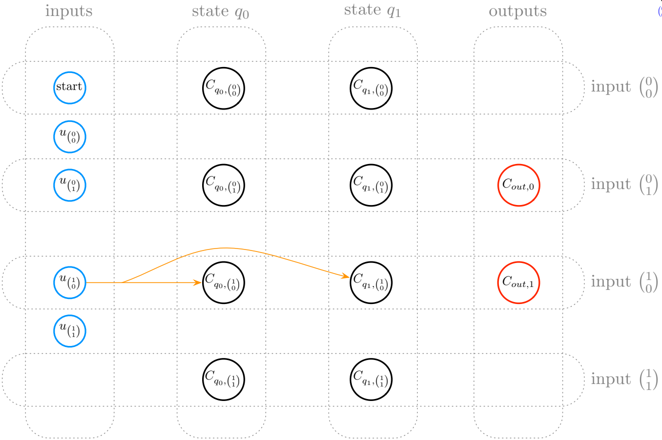
Binary Adder Boolean Neural Network



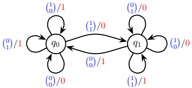


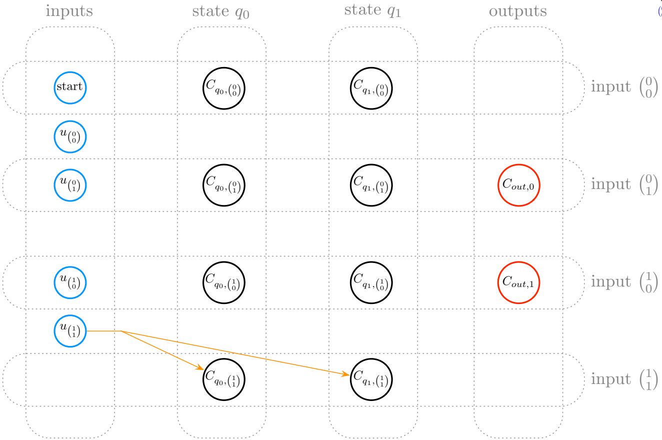
Binary Adder Boolean Neural Network



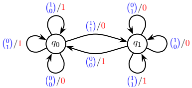


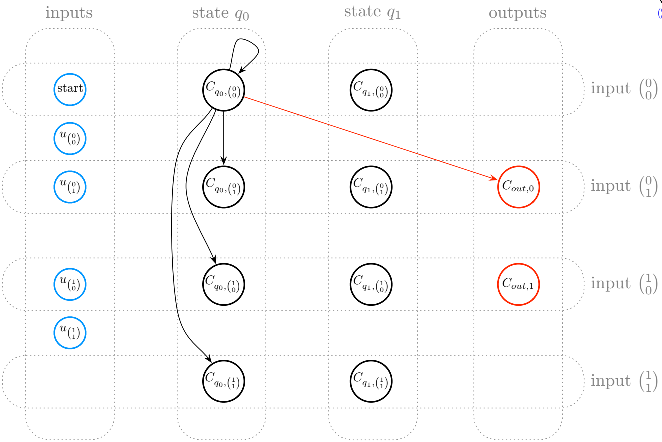
Binary Adder Boolean Neural Network



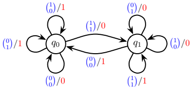


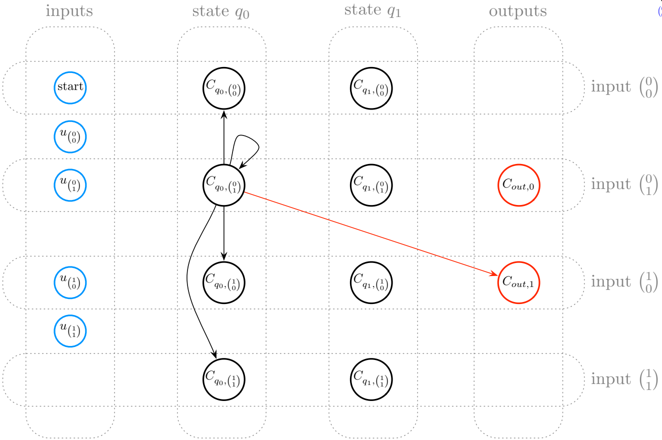
Binary Adder Boolean Neural Network



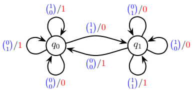


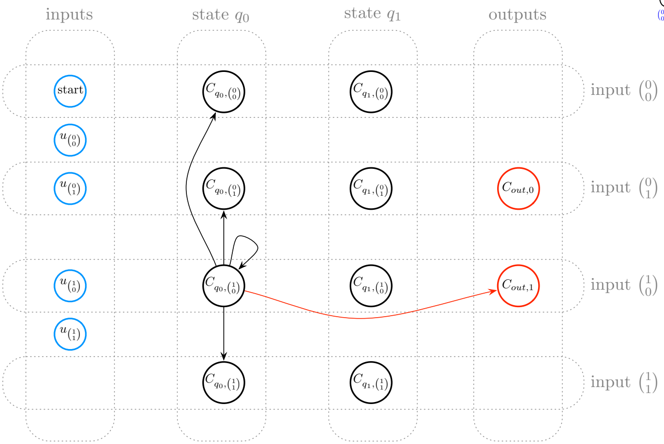
Binary Adder Boolean Neural Network



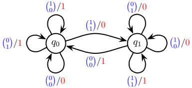


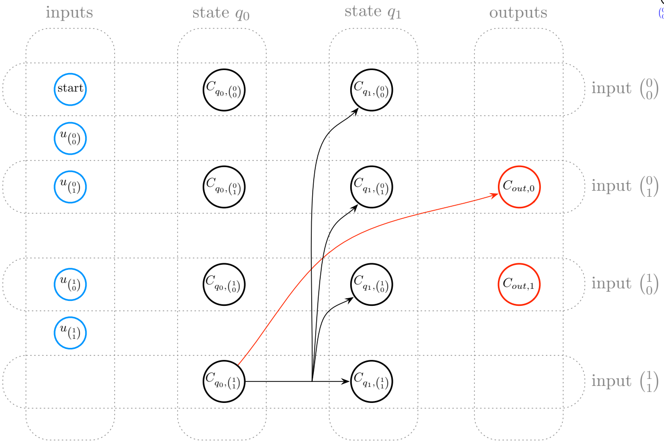
Binary Adder Boolean Neural Network



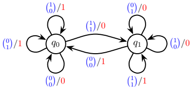


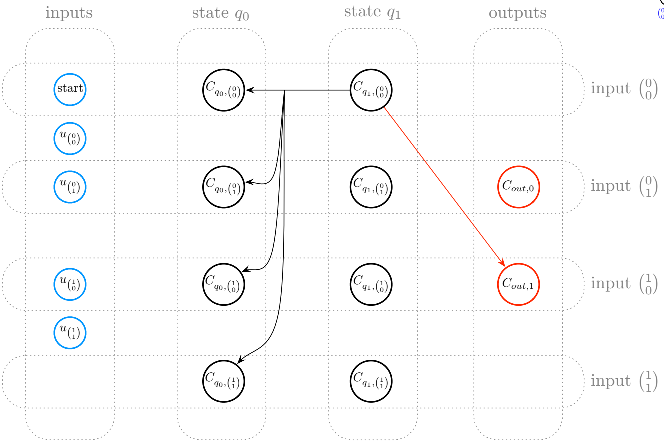
Binary Adder Boolean Neural Network



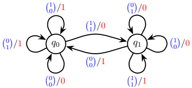


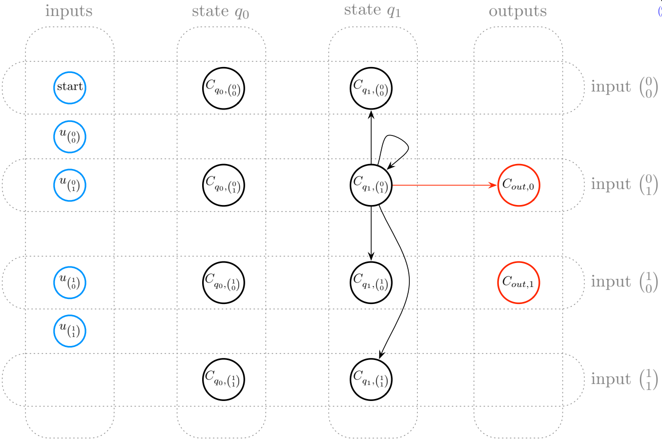
Binary Adder Boolean Neural Network



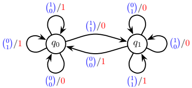


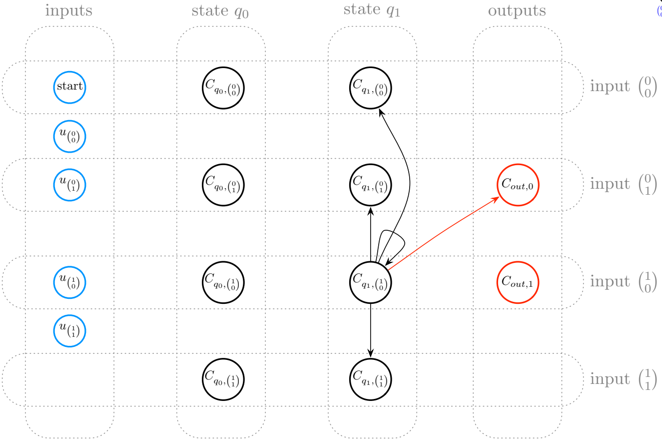
Binary Adder Boolean Neural Network



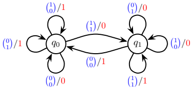


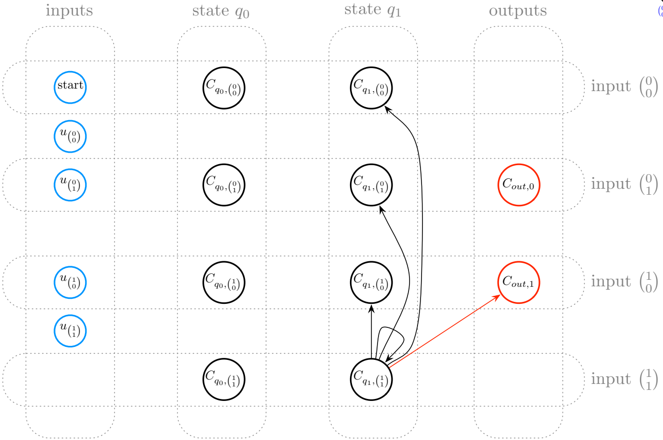
Binary Adder Boolean Neural Network



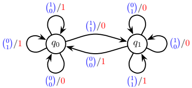


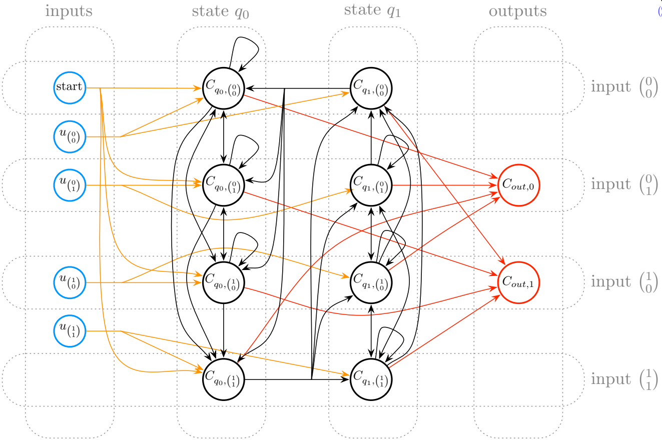
Binary Adder Boolean Neural Network



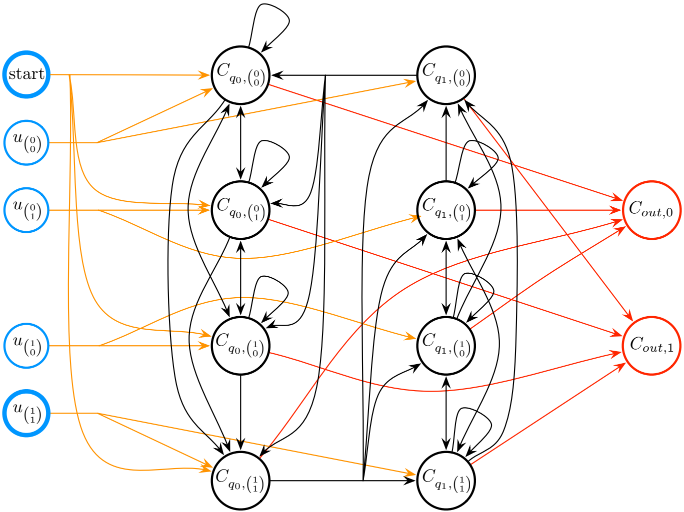


Binary Adder Boolean Neural Network



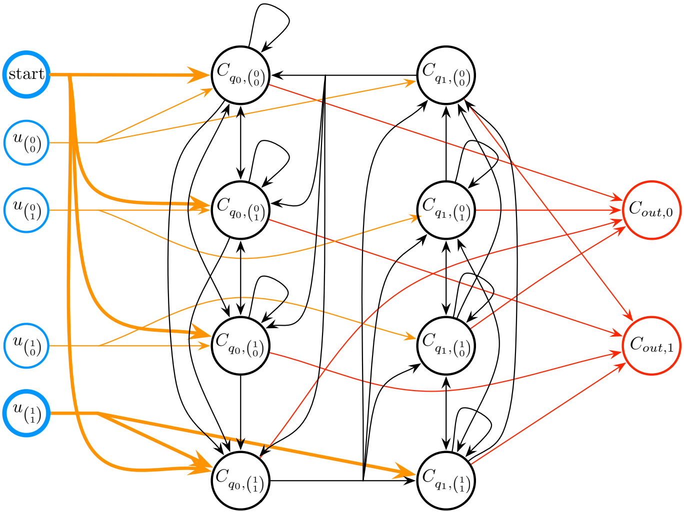


Simulation

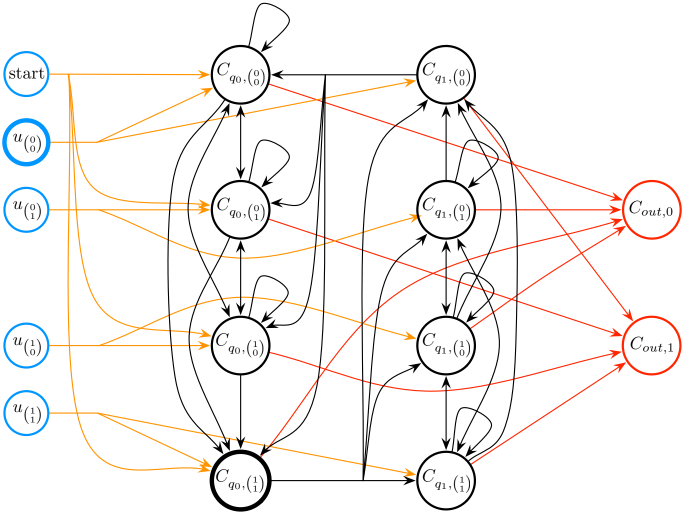


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

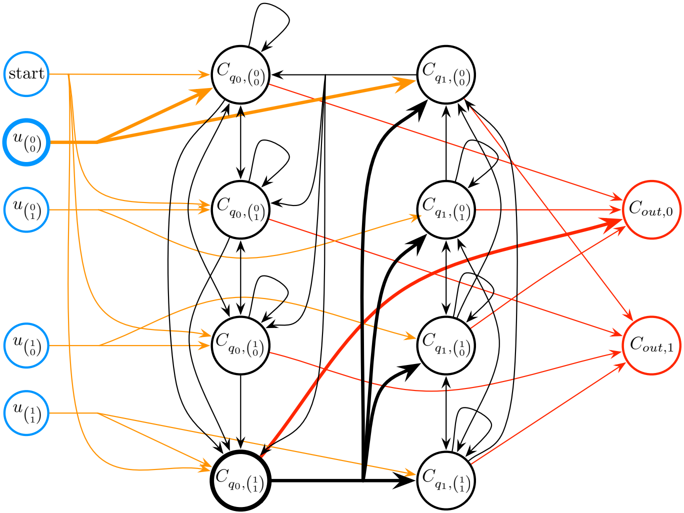
Simulation


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

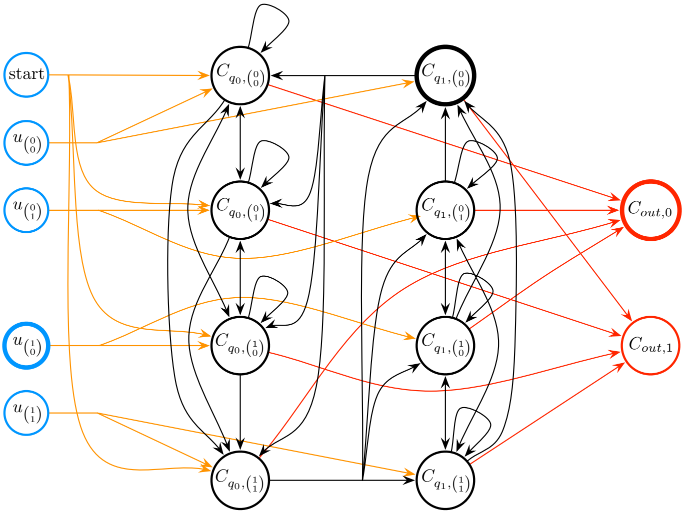
Simulation


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

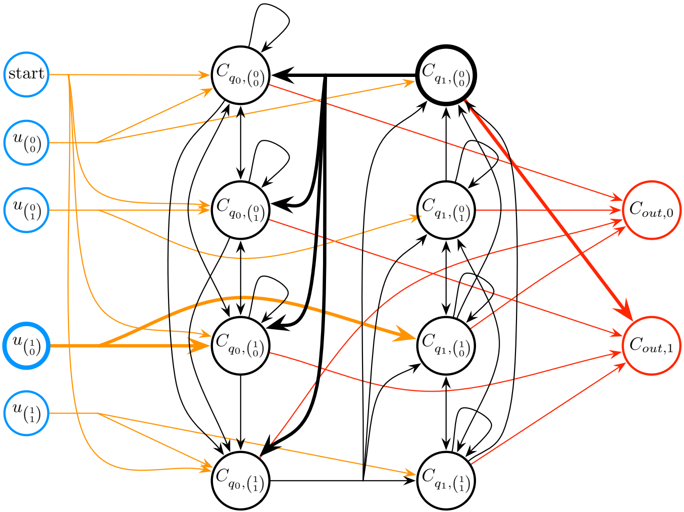
Simulation


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

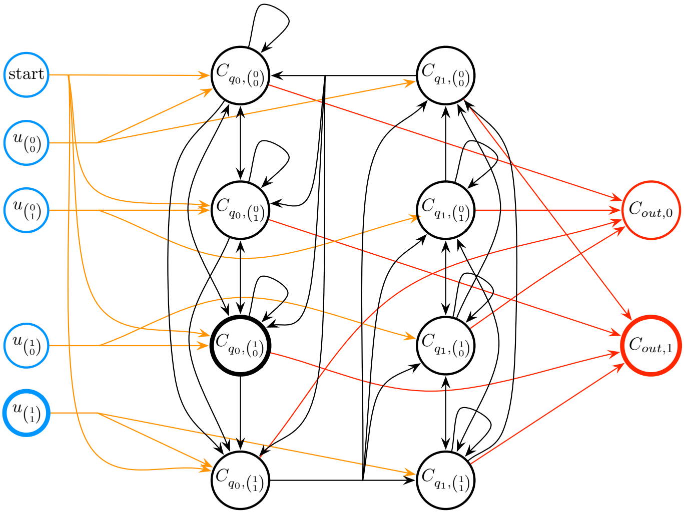
Simulation


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

Simulation

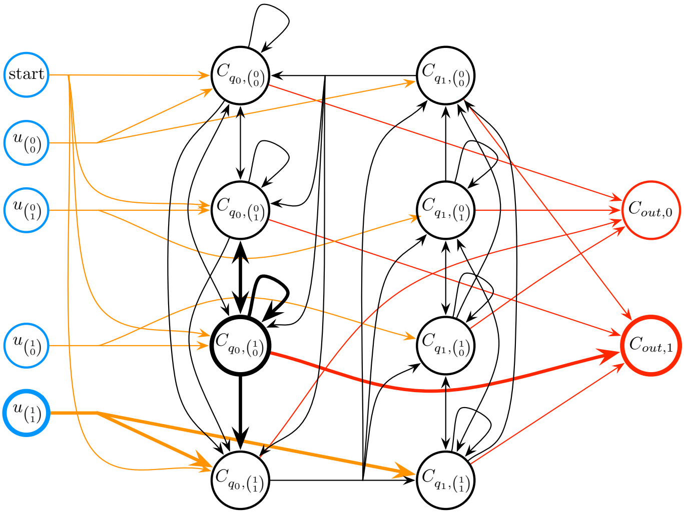

start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

Simulation

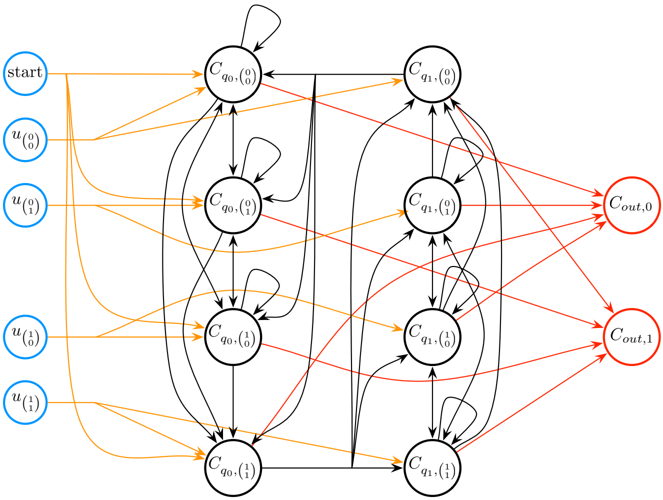


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

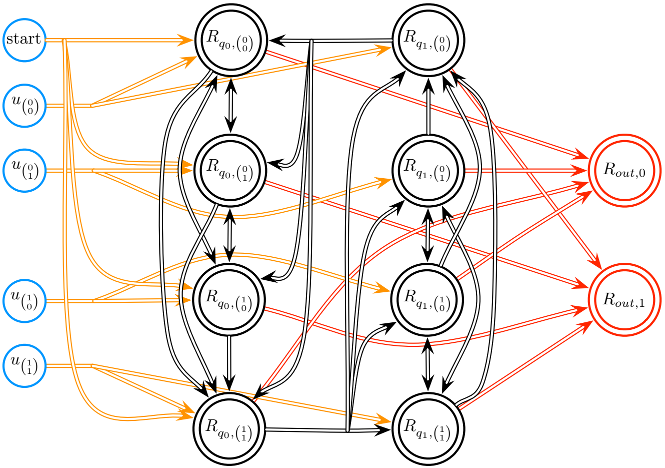
Simulation


start
$u_{\binom{0}{0}}$
$u_{\binom{0}{1}}$
$u_{\binom{1}{0}}$
$u_{\binom{1}{1}}$
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{1}}$
$C_{out,0}$
$C_{out,1}$

Generalization to Synfire Ring RNNs

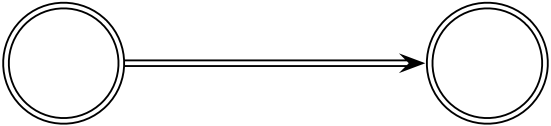


Generalization to Synfire Ring RNNs



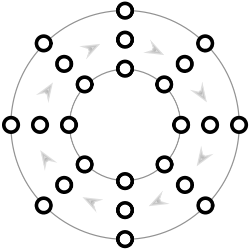
Transition of ring activities: fibres of excitatory & Inhibitory connections

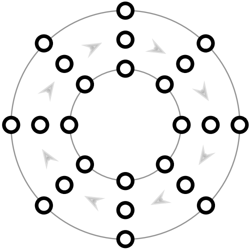
- The *refractory period* of the cells allows for a natural inhibitory system.



Transition of ring activities: fibres of excitatory & Inhibitory connections

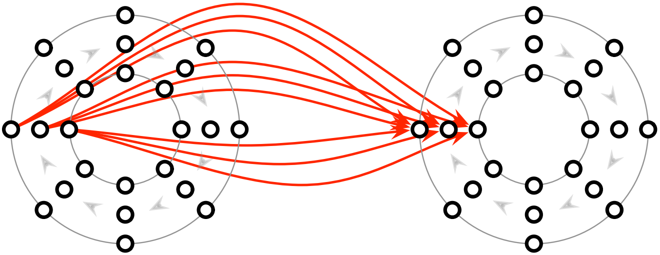
- The *refractory period* of the cells allows for a natural inhibitory system.





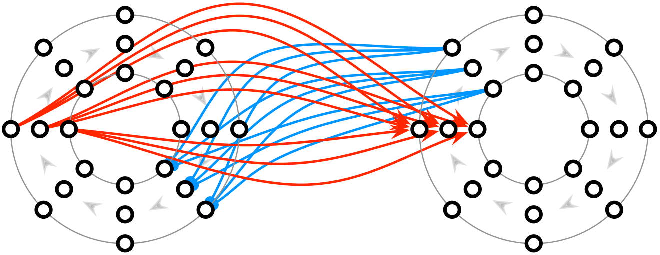
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



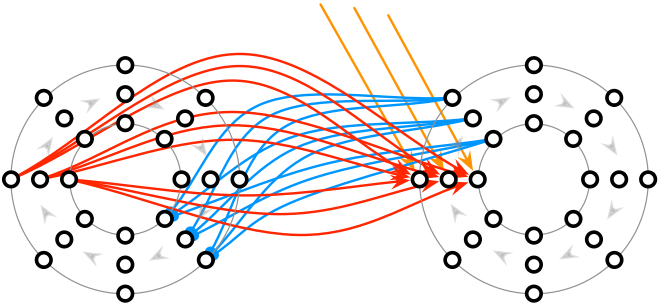
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



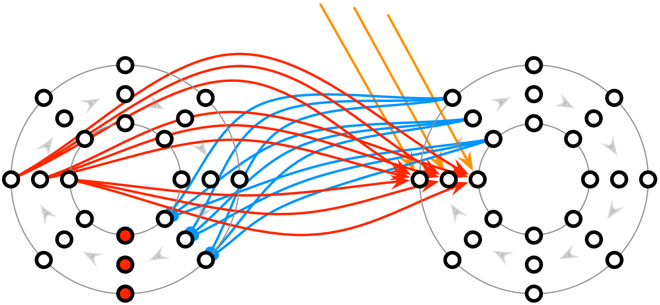
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



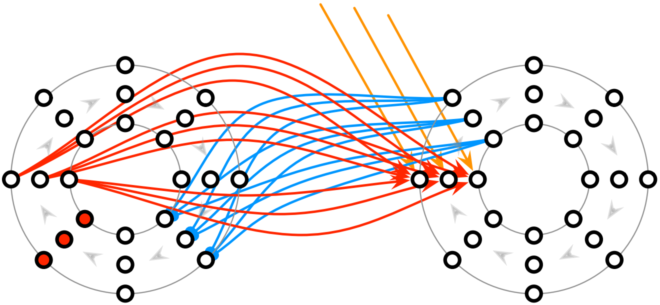
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



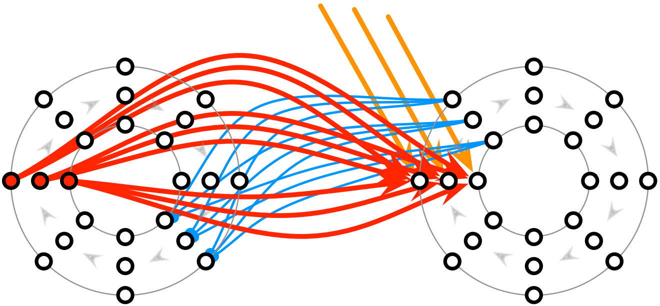
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



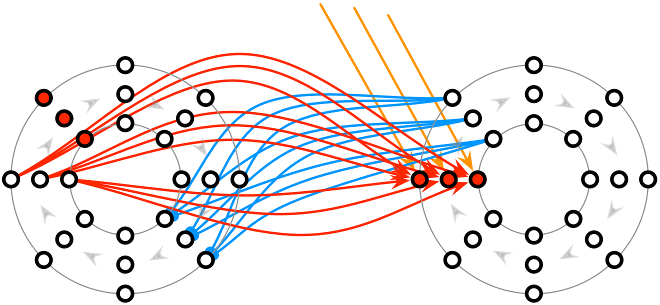
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



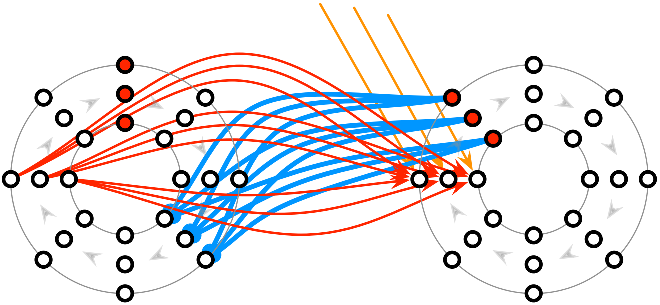
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



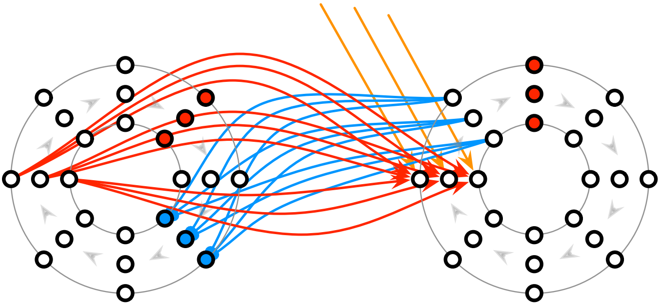
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



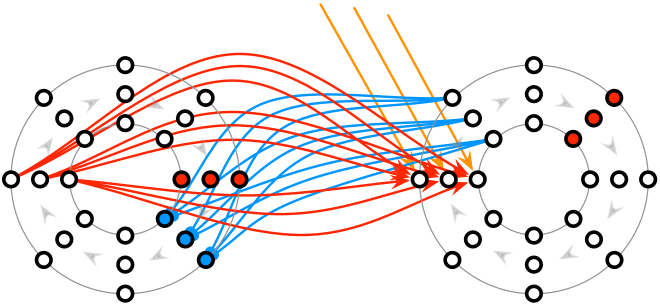
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



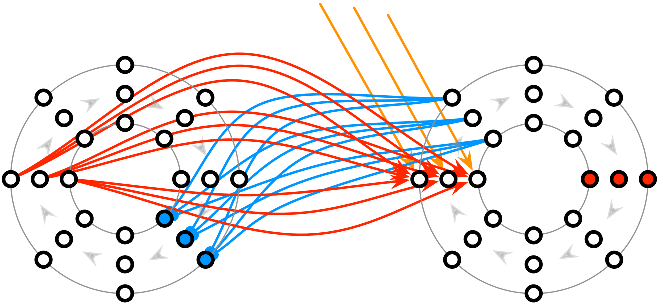
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



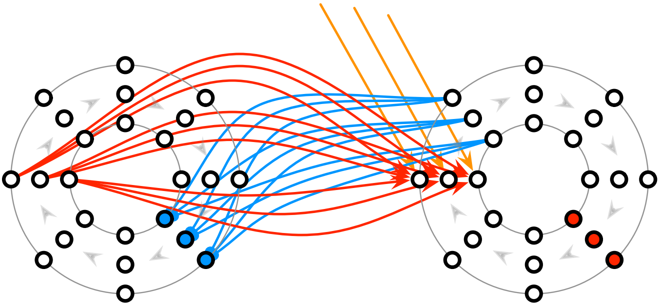
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



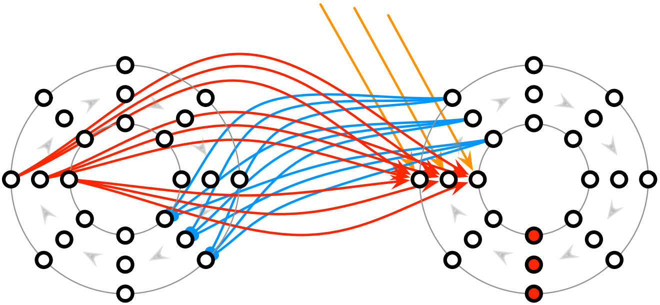
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



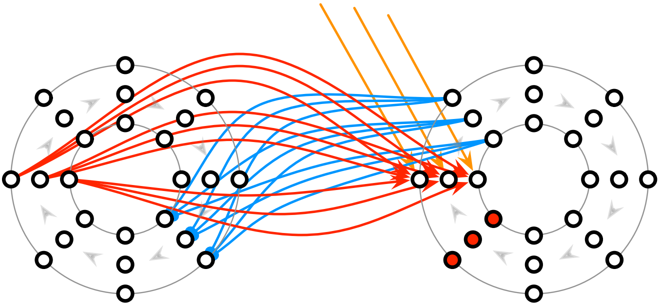
Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.

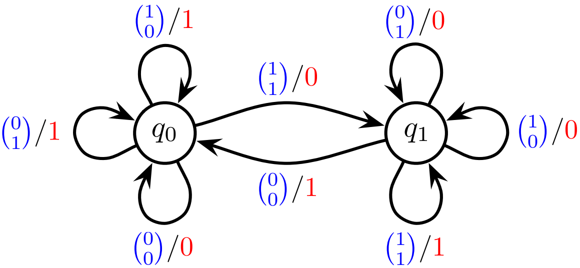


Transition of ring activities: fibres of excitatory & Inhibitory connections

- The *refractory period* of the cells allows for a natural inhibitory system.



Simulation



Play movie

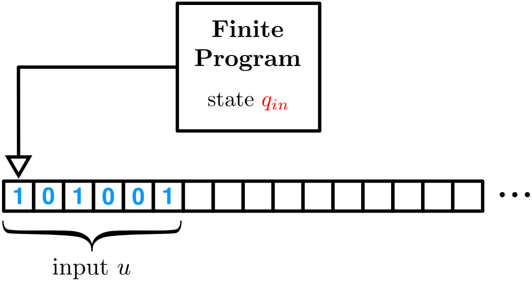
Automata & Hodgkin-Huxley RNNs with Synfire Rings

Since the construction is generic, the following result ensues:

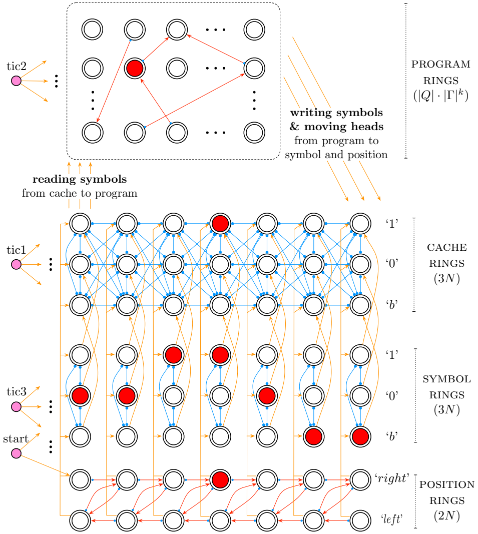
Theorem

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

TM and RNN

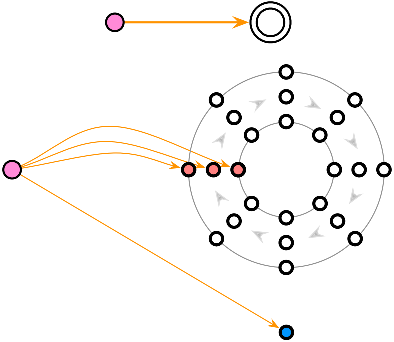


TM and RNN


tic2
writing symbols
& moving heads
from program to
symbol and position
PROGRAM
RINGS
$(|Q| \cdot |\Gamma|^k)$
reading symbols
from cache to program
tic1
tic3
start
'1'
'0'
'b'
CACHE
RINGS
$(3N)$
'1'
'0'
'b'
SYMBOL
RINGS
$(3N)$
'right'
'left'
POSITION
RINGS
$(2N)$

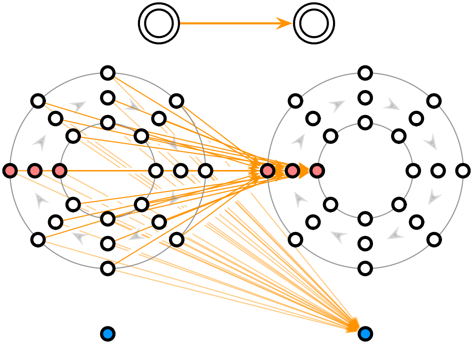
Fibres of connections

- Cell-to-ring one-shot excitation



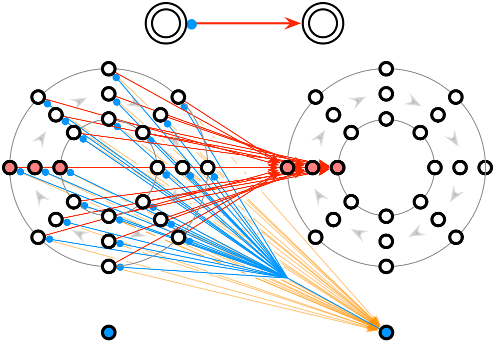
Fibres of connections

- Ring-to-ring constant excitation



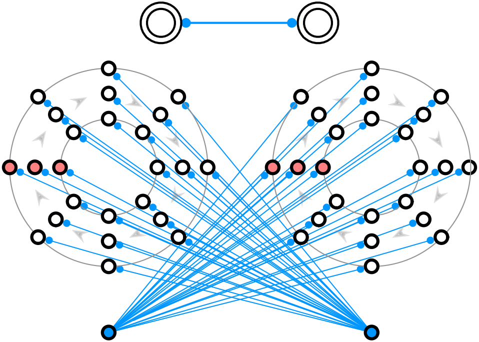
Fibres of connections

- Ring-to-ring constant excitation / one-shot inhibition

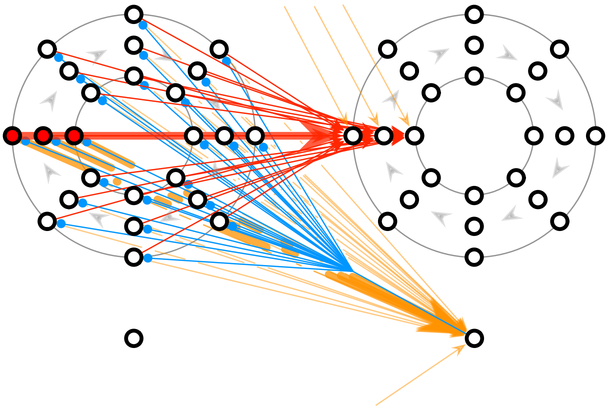


Fibres of connections

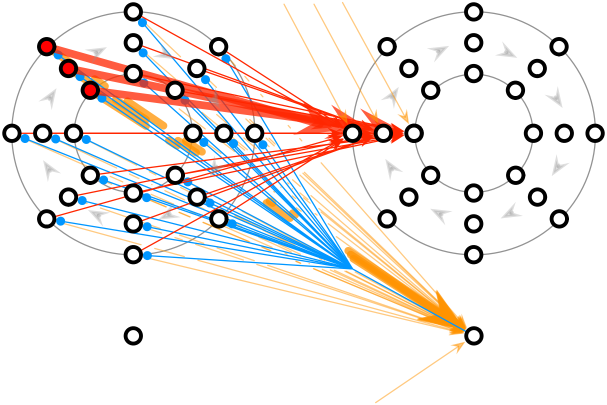
- Ring-to-ring bidirectional one-shot inhibition



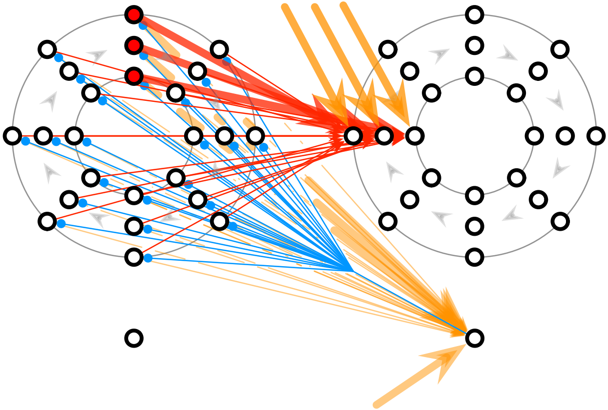
Transition of ring activities



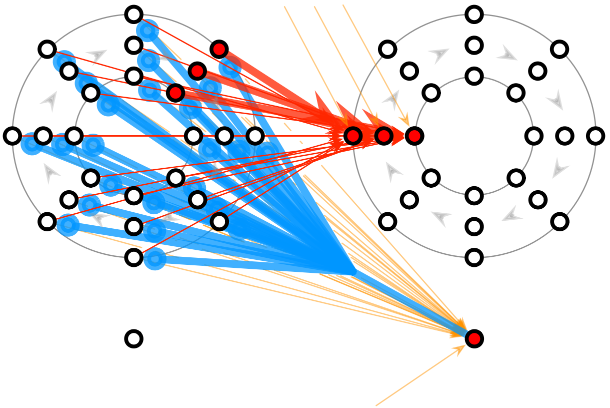
Transition of ring activities



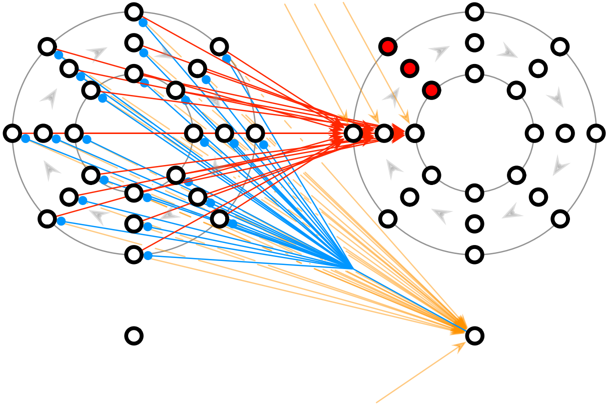
Transition of ring activities



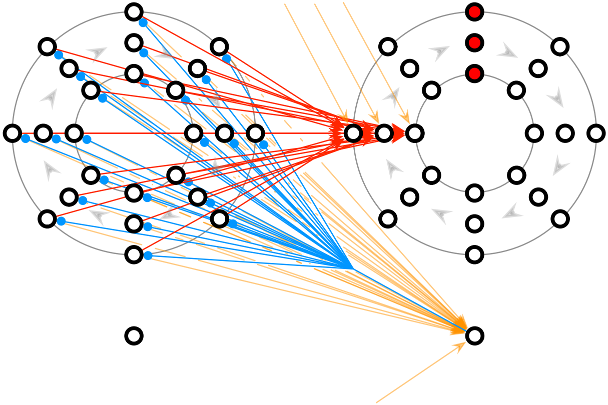
Transition of ring activities



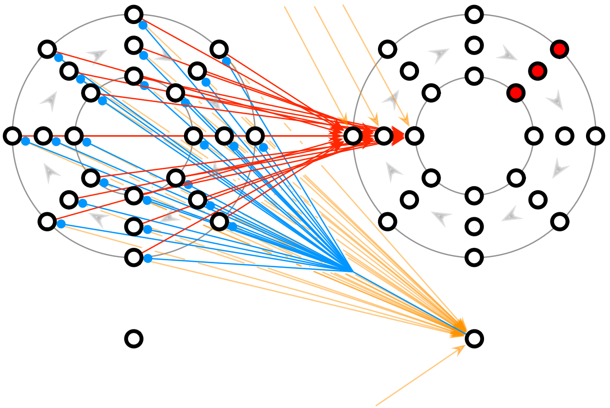
Transition of ring activities



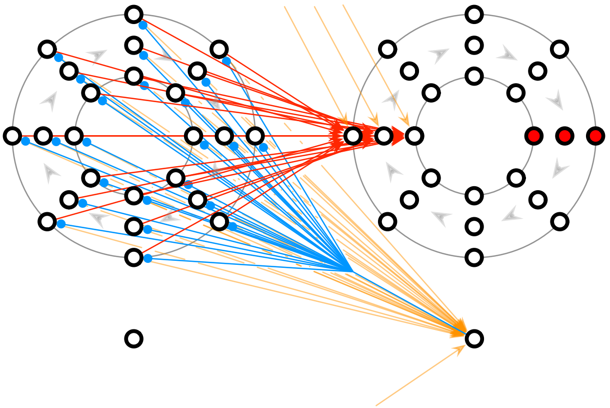
Transition of ring activities



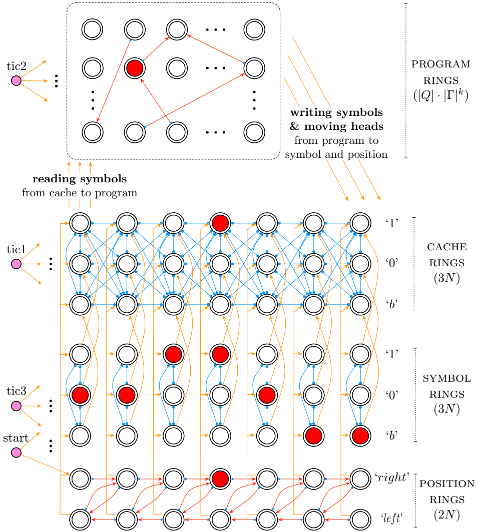
Transition of ring activities



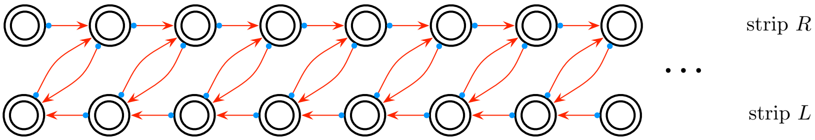
Transition of ring activities



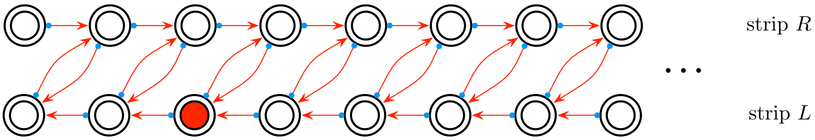
Position Rings



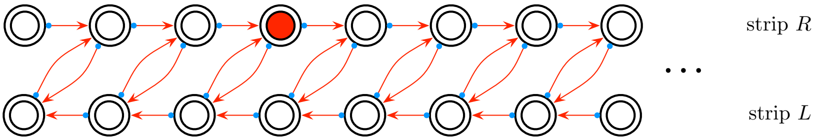
Position Rings



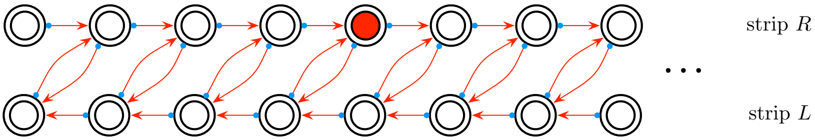
Position Rings



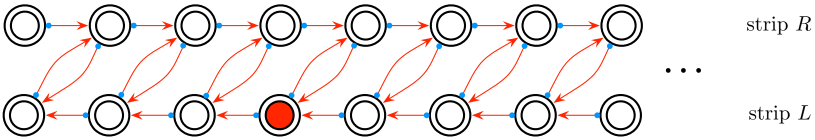
Position Rings



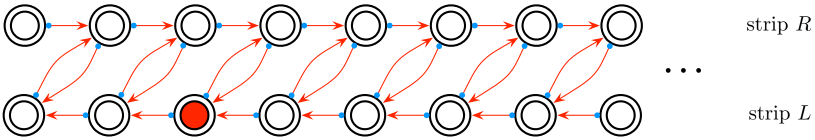
Position Rings



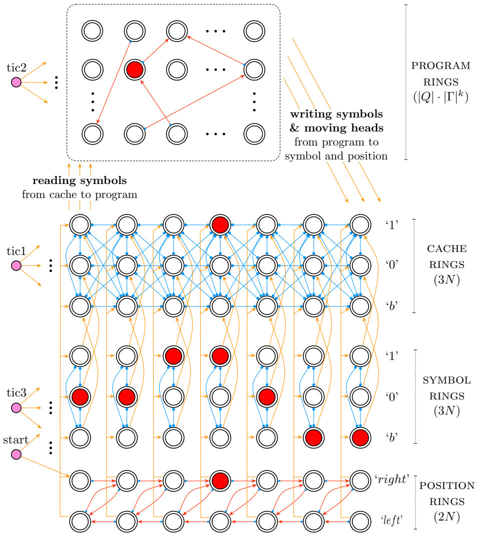
Position Rings



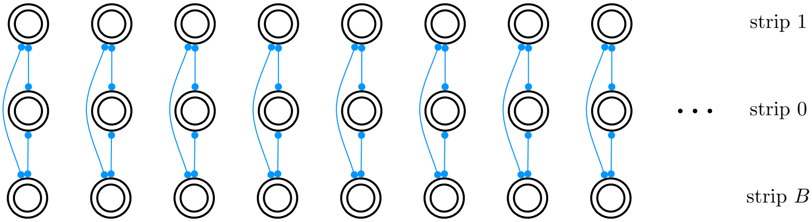
Position Rings



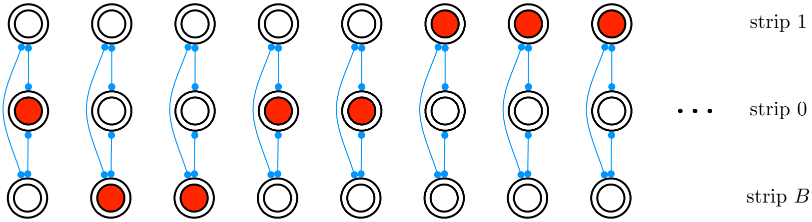
Symbol Rings



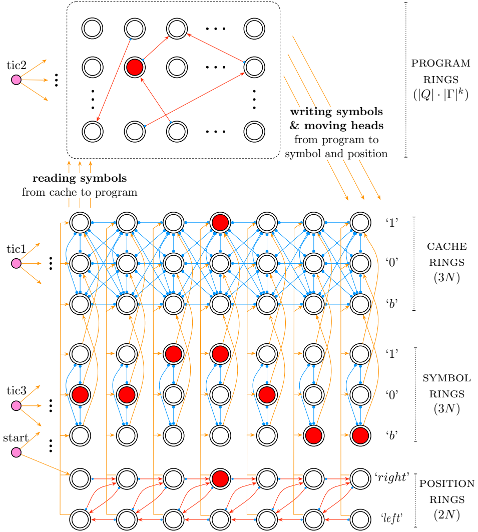
Symbol Rings



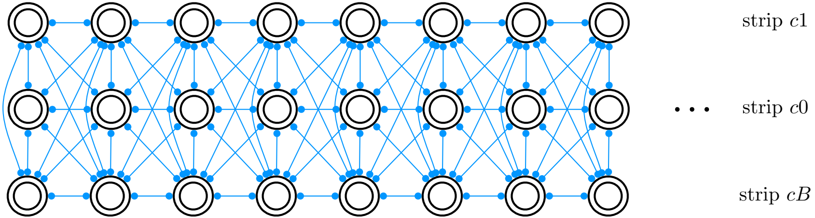
Symbol Rings



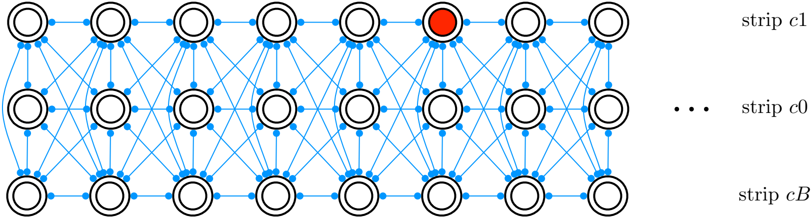
Cache Rings


tic2
tic1
tic3
start
PROGRAM
RINGS
$(|Q| \cdot |\Gamma|^k)$
writing symbols
& moving heads
from program to
symbol and position
reading symbols
from cache to program
'1'
'0'
'b'
CACHE
RINGS
$(3N)$
'1'
'0'
'b'
SYMBOL
RINGS
$(3N)$
'*right*'
'*left*'
POSITION
RINGS
$(2N)$

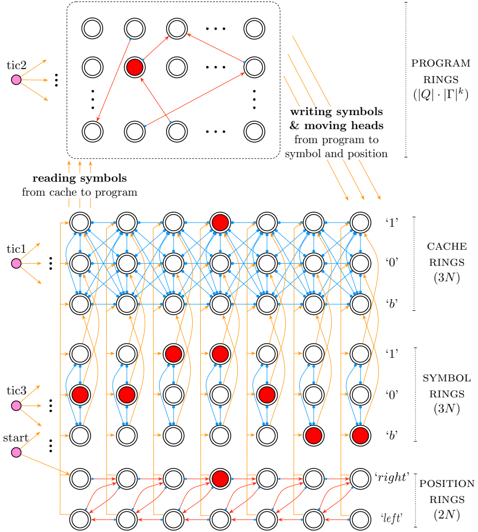
Cache Rings



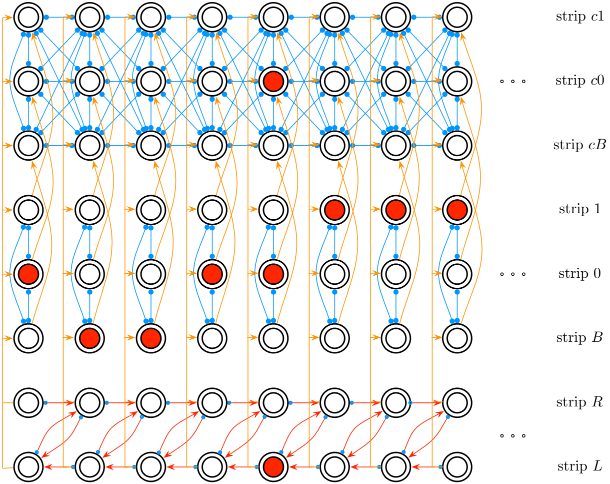
Cache Rings



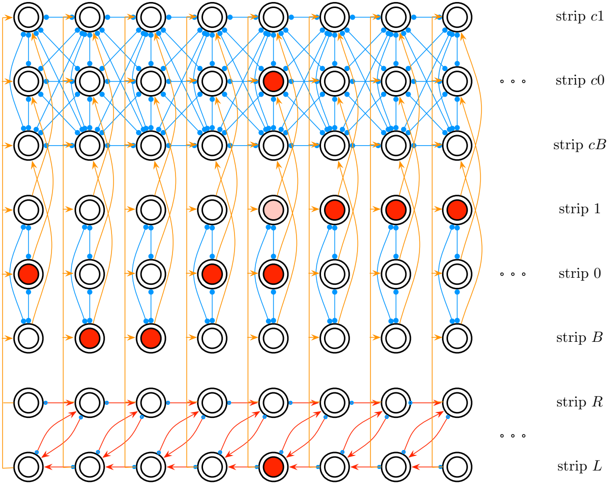
Position-Symbol-Cache Connections



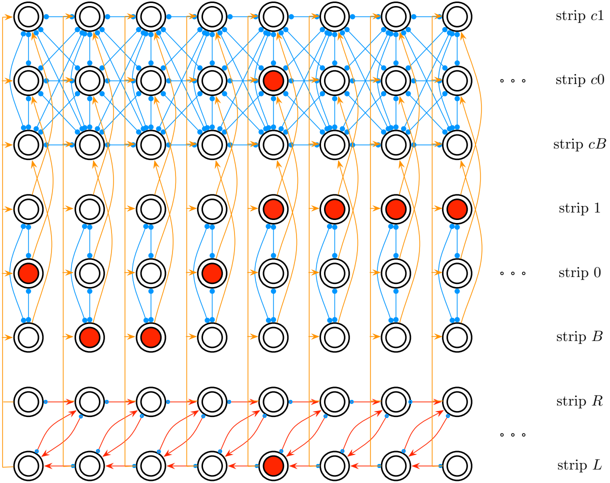
Position-Symbol-Cache Connections



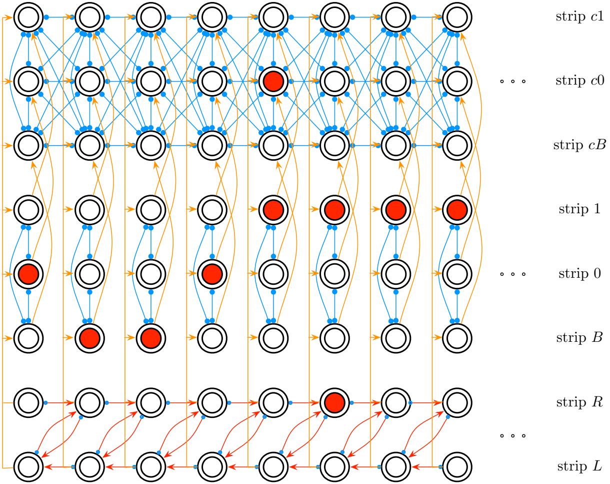
Position-Symbol-Cache Connections



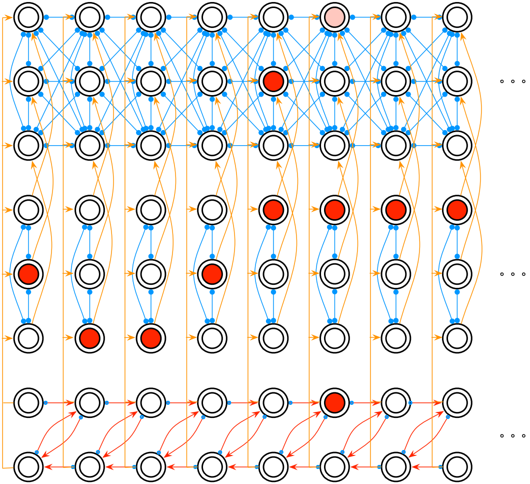
Position-Symbol-Cache Connections



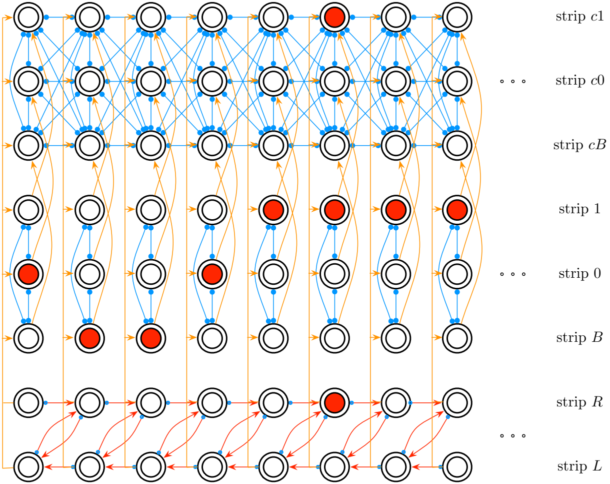
Position-Symbol-Cache Connections



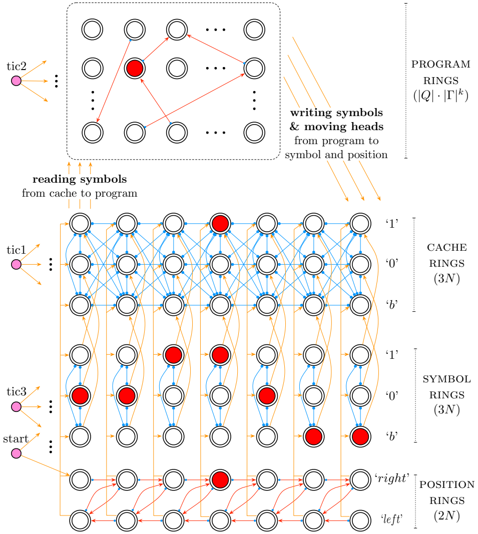
Position-Symbol-Cache Connections



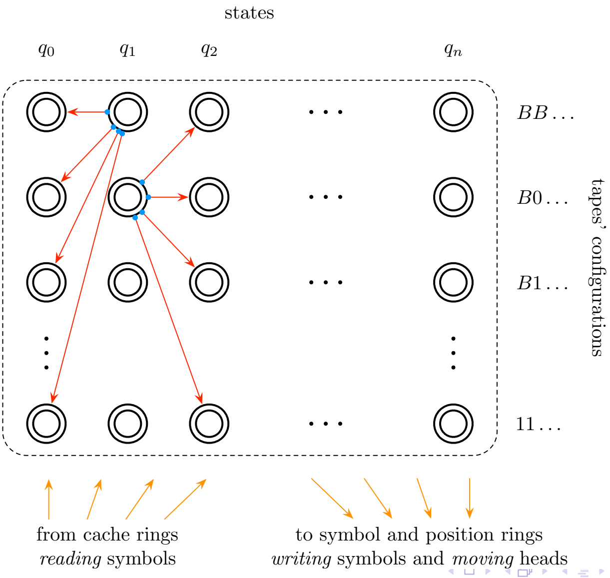
Position-Symbol-Cache Connections



Program Rings



Program Rings



Turing machines & Boolean RNNs with Synfire Rings

Since the construction is generic, the following results hold:

Theorem

- *Let $\mathcal{M}$ be a fixed-space k-tape TM whose every tape is of length N. Then, there exists some B-RNN composed of $\mathcal{O}(N)$ synfire rings and cells that simulates $\mathcal{M}$ in linear time.*
- *Let $\mathcal{M}$ be a k-tape TM. Then, there exists a B-RNN composed of infinitely many synfire rings that simulates $\mathcal{M}$ in linear time.*

Turing machines & Boolean RNNs with Synfire Rings

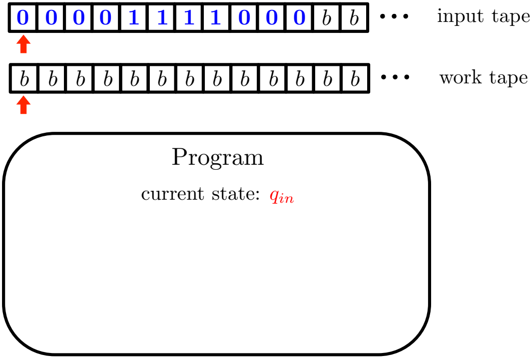
Since the construction is generic, the following results hold:

Theorem

- *Let $\mathcal{M}$ be a fixed-space k-tape TM whose every tape is of length N. Then, there exists some B-RNN composed of $\mathcal{O}(N)$ synfire rings and cells that simulates $\mathcal{M}$ in linear time.*
- *Let $\mathcal{M}$ be a k-tape TM. Then, there exists a B-RNN composed of infinitely many synfire rings that simulates $\mathcal{M}$ in linear time.*

Simulation

- TM recognizing the non-regular language $L = \{0^n 1^n 0^n : n \geq 0\}$.



Play movie

Future Works

1. Reservoir computing: echo state networks (ESN) / liquid state machines (LSM)
   - Introduce *learning* – via synaptic plasticity, intrinsic plasticity, etc. – within this bio-inspired neural architecture.
2. Neuromorphic computing
   - Implement bio-inspired into neuromorphic hardwares.
3. Biology
   - Implement this architecture into cultured neural networks (*in vitro*): towards neuronal computers. . .

Future Works

1. **Reservoir computing: echo state networks (ESN) / liquid state machines (LSM)**

- Introduce *learning* – via synaptic plasticity, intrinsic plasticity, etc. – within this bio-inspired neural architecture.

2. **Neuromorphic computing**

- Implement bio-inspired into neuromorphic hardwares.

3. **Biology**

- Implement this architecture into cultured neural networks (*in vitro*): towards neuronal computers...

Future Works

1. Reservoir computing: echo state networks (ESN) / liquid state machines (LSM)

- Introduce *learning* – via synaptic plasticity, intrinsic plasticity, etc. – within this bio-inspired neural architecture.

2. Neuromorphic computing

- Implement bio-inspired into neuromorphic hardwares.

3. Biology

- Implement this architecture into cultured neural networks (*in vitro*): towards neuronal computers...

Future Works

1. Reservoir computing: echo state networks (ESN) / liquid state machines (LSM)

- Introduce *learning* – via synaptic plasticity, intrinsic plasticity, etc. – within this bio-inspired neural architecture.

2. Neuromorphic computing

- Implement bio-inspired into neuromorphic hardwares.

3. Biology

- Implement this architecture into cultured neural networks (*in vitro*): towards neuronal computers. . .

Future Works

1. Reservoir computing: echo state networks (ESN) / liquid state machines (LSM)

- Introduce *learning* – via synaptic plasticity, intrinsic plasticity, etc. – within this bio-inspired neural architecture.

2. Neuromorphic computing

- Implement bio-inspired into neuromorphic hardwares.

3. Biology

- Implement this architecture into cultured neural networks (*in vitro*): towards neuronal computers. . .

Future Works

1. Reservoir computing: echo state networks (ESN) / liquid state machines (LSM)

- Introduce *learning* – via synaptic plasticity, intrinsic plasticity, etc. – within this bio-inspired neural architecture.

2. Neuromorphic computing

- Implement bio-inspired into neuromorphic hardwares.

3. Biology

- Implement this architecture into cultured neural networks (*in vitro*): towards neuronal computers. . .