Neural Computation with Recurrent Neural Networks Composed of Synfire Rings

Jérémie Cabessa

Laboratoire d'économie mathématique
Université Paris II, France

March 30, 2018, Nice, France

Introduction

- We recall important results about the computational power of recurrent neural networks.
- We show that neural networks represent a natural model for oracle-based computation, beyond the Turing limits.
- We introduce a bio-inspired paradigm for neural computation based on the concept of *synfire rings*.

Introduction

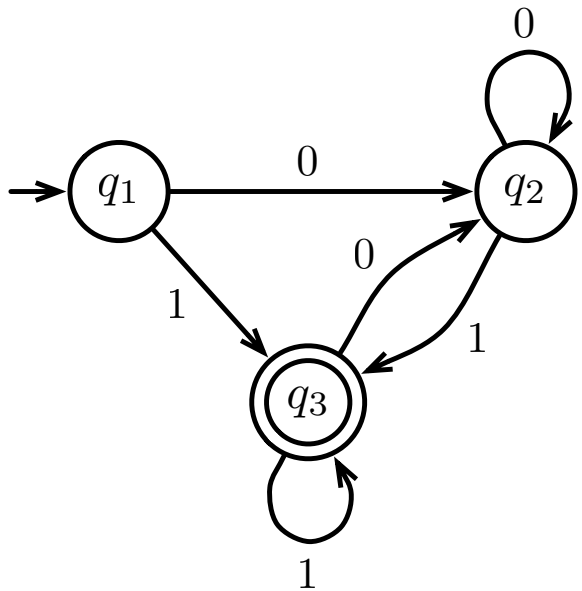
- We recall important results about the computational power of recurrent neural networks.
- We show that neural networks represent a natural model for oracle-based computation, beyond the Turing limits.
- We introduce a bio-inspired paradigm for neural computation based on the concept of *synfire rings*.

Introduction

- We recall important results about the computational power of recurrent neural networks.
- We show that neural networks represent a natural model for oracle-based computation, beyond the Turing limits.
- We introduce a bio-inspired paradigm for neural computation based on the concept of *synfire rings*.

Finite State Automaton

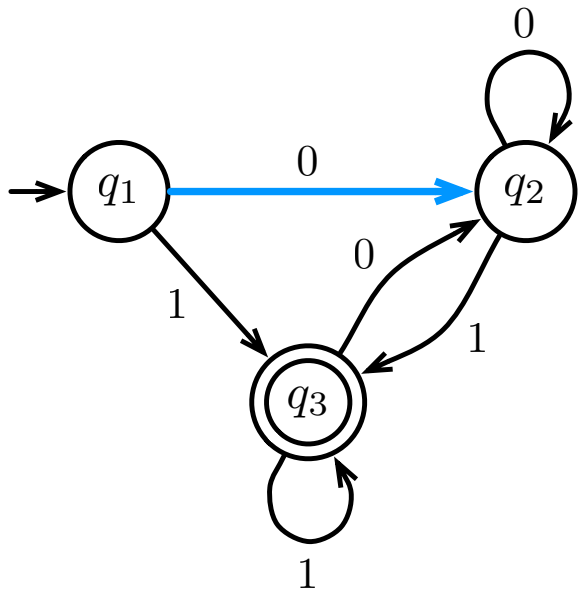
A *finite state automaton* (FSA) can be represented as a graph whose nodes and edges are the computational states and transitions between those.



- input u is *accepted* by $\mathcal{A}$ if $\mathcal{A}(u)$ reaches a final state
- input u is *rejected* by $\mathcal{A}$ if $\mathcal{A}(u)$ otherwise

Finite State Automaton

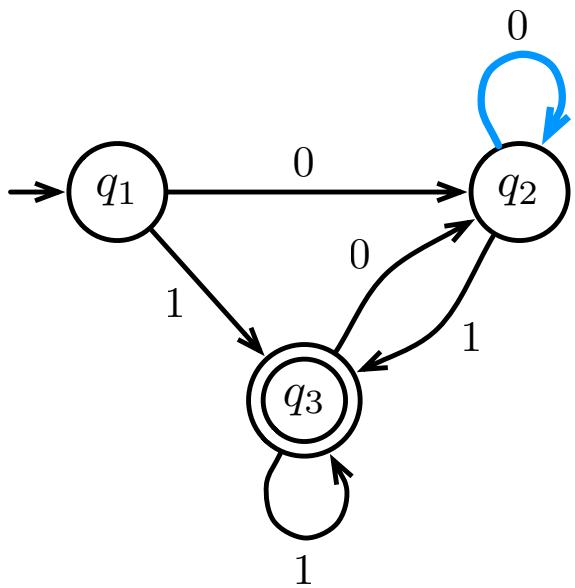
A *finite state automaton* (FSA) can be represented as a graph whose nodes and edges are the computational states and transitions between those.



- input u is *accepted* by $\mathcal{A}$ if $\mathcal{A}(u)$ reaches a final state
- input u is *rejected* by $\mathcal{A}$ if $\mathcal{A}(u)$ otherwise

Finite State Automaton

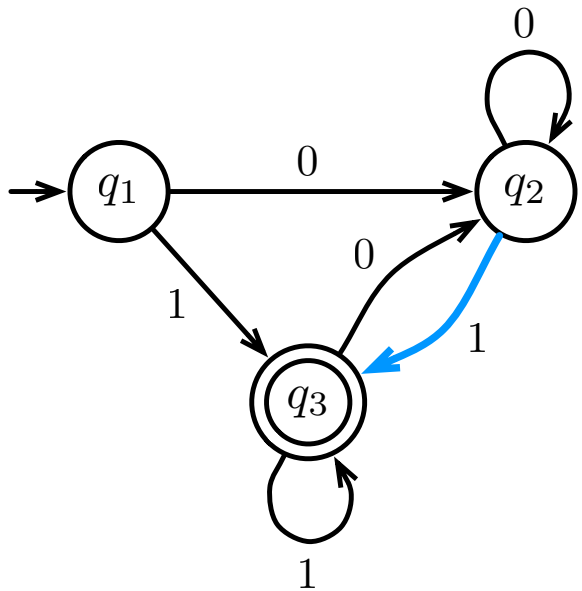
A *finite state automaton* (FSA) can be represented as a graph whose nodes and edges are the computational states and transitions between those.



- input u is *accepted* by $\mathcal{A}$ if $\mathcal{A}(u)$ reaches a final state
- input u is *rejected* by $\mathcal{A}$ if $\mathcal{A}(u)$ otherwise

Finite State Automaton

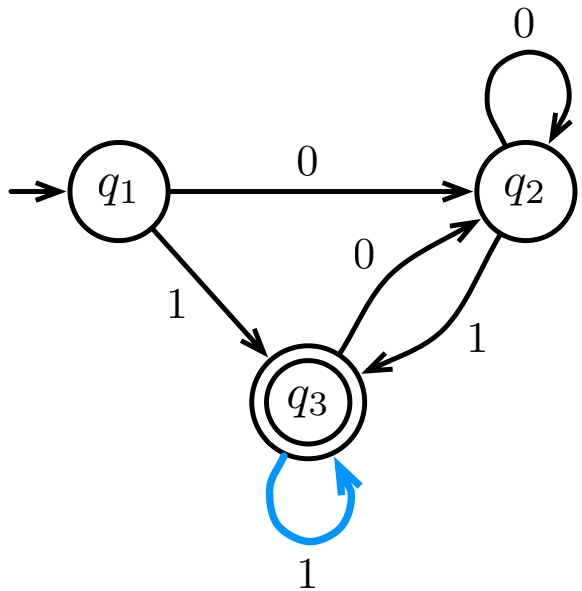
A *finite state automaton* (FSA) can be represented as a graph whose nodes and edges are the computational states and transitions between those.



- input u is *accepted* by $\mathcal{A}$ if $\mathcal{A}(u)$ reaches a final state
- input u is *rejected* by $\mathcal{A}$ if $\mathcal{A}(u)$ otherwise

Finite State Automaton

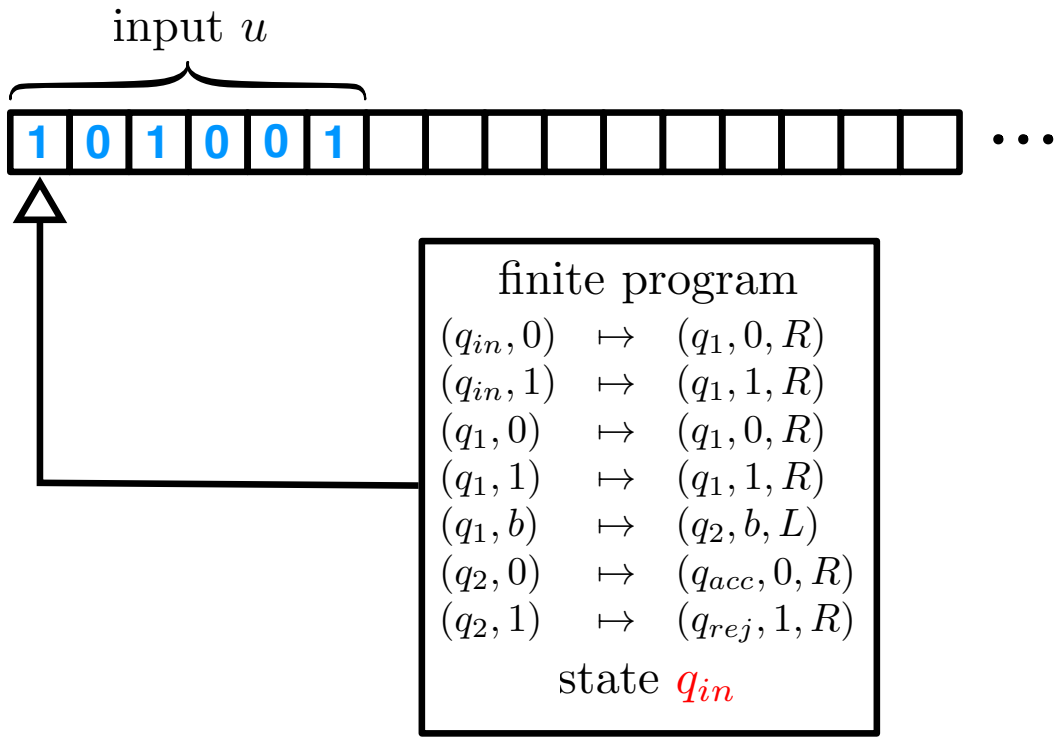
A *finite state automaton* (FSA) can be represented as a graph whose nodes and edges are the computational states and transitions between those.



- input u is *accepted* by $\mathcal{A}$ if $\mathcal{A}(u)$ reaches a final state
- input u is *rejected* by $\mathcal{A}$ if $\mathcal{A}(u)$ otherwise

Turing Machine

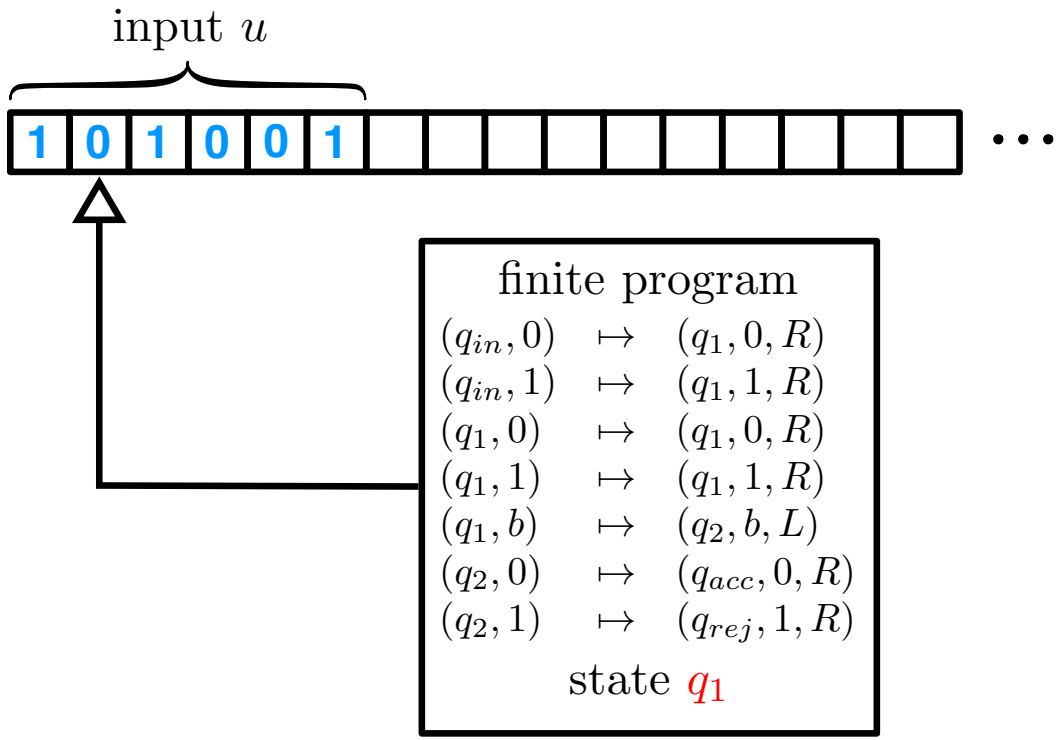
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

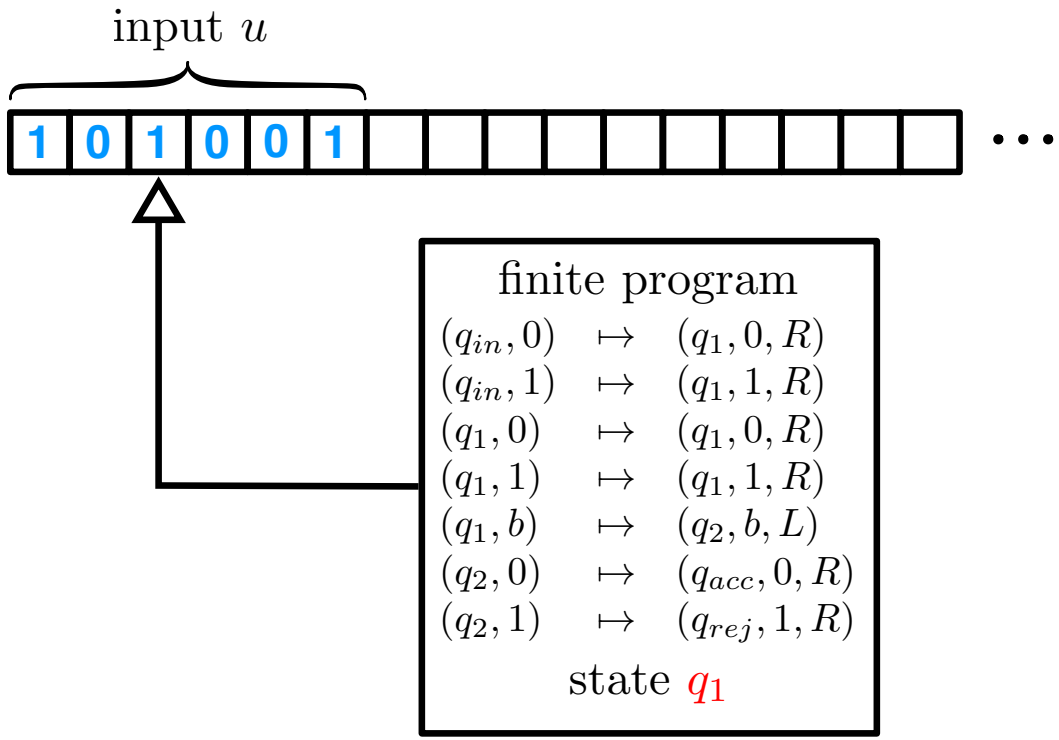
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

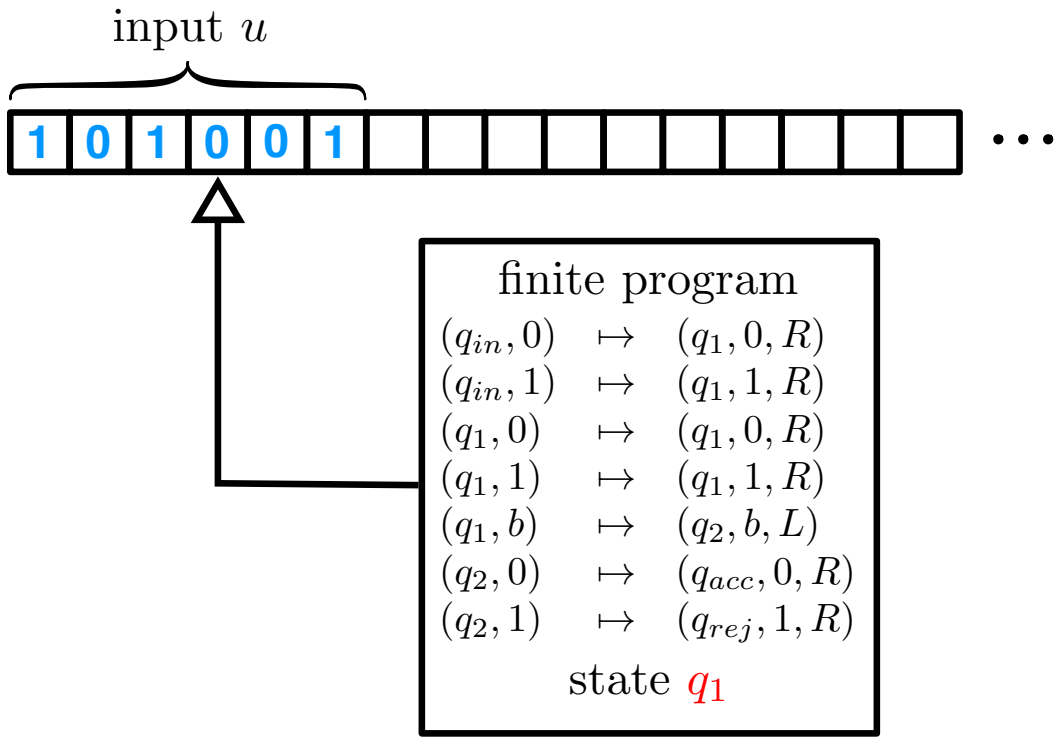
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

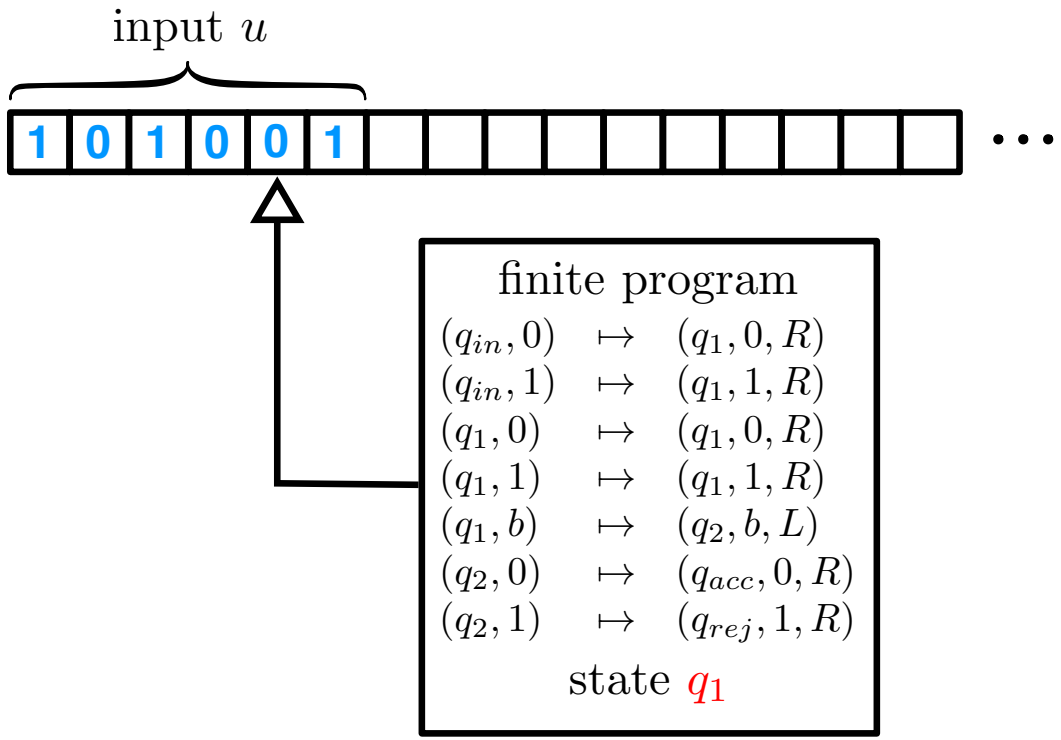
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

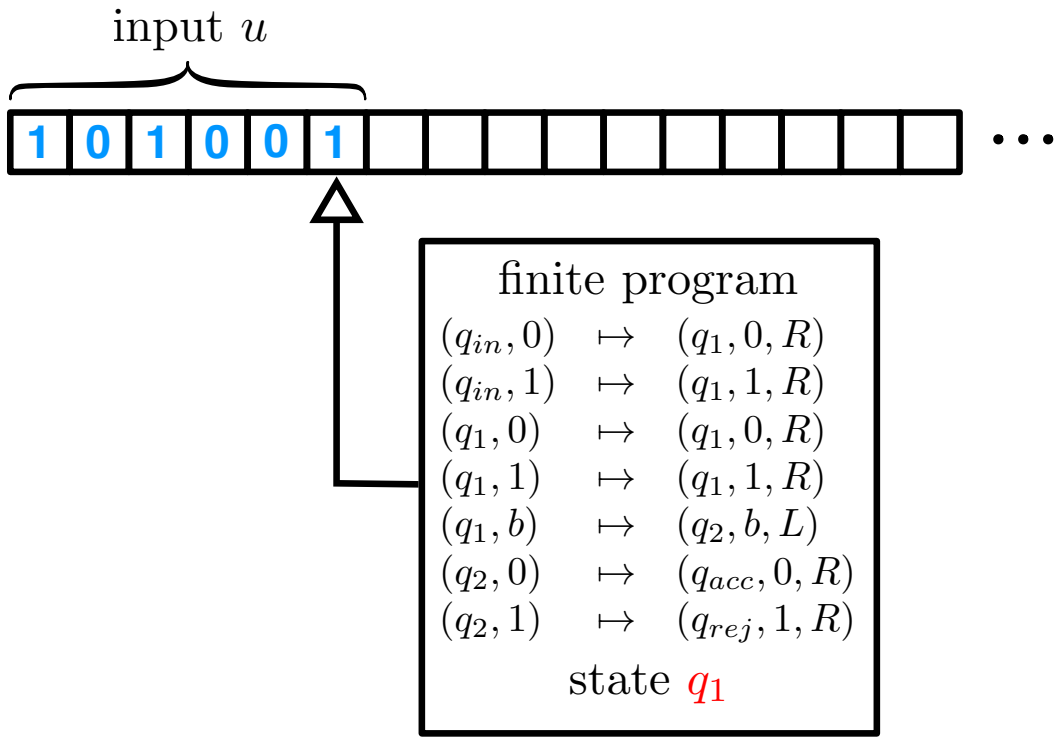
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

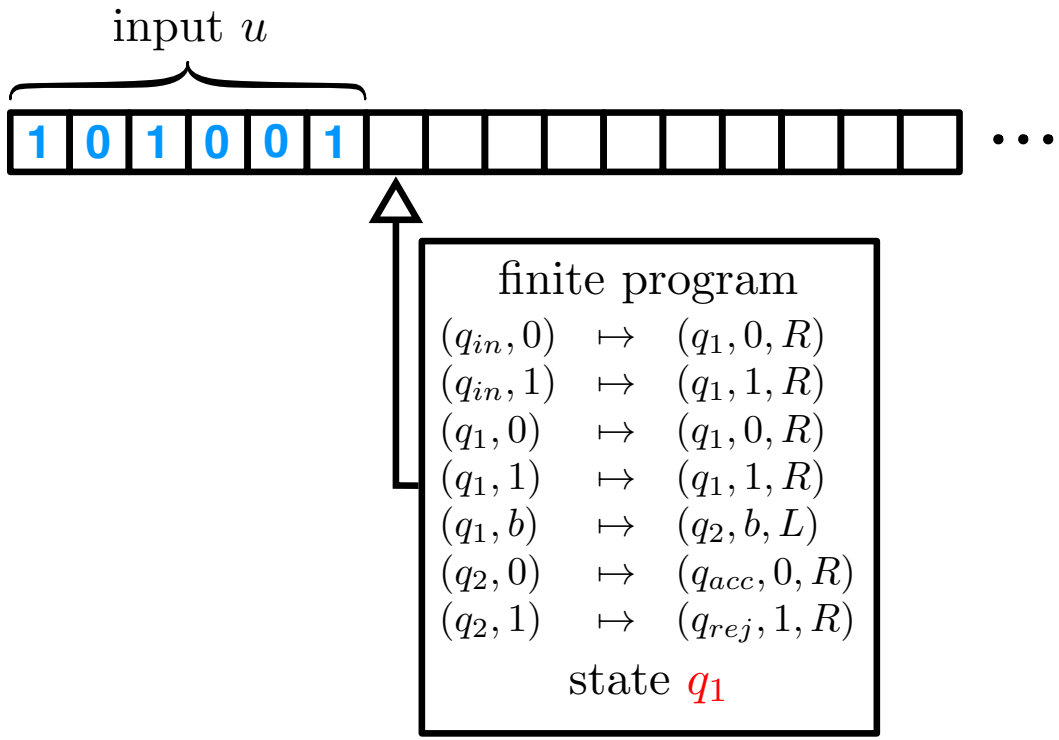
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

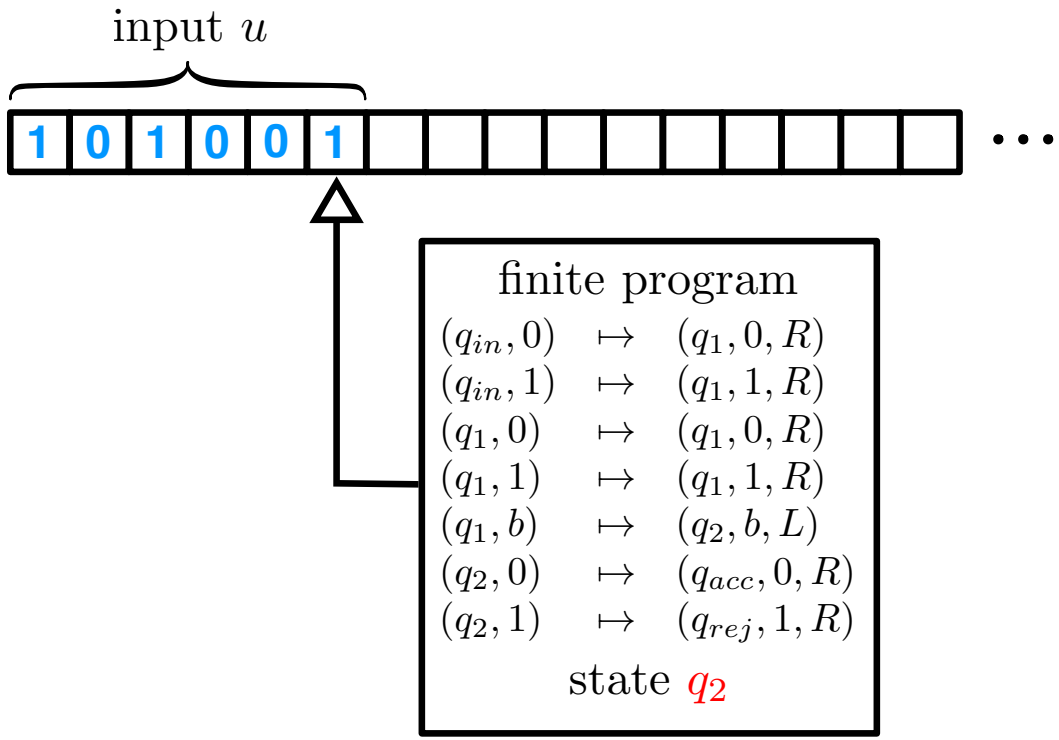
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

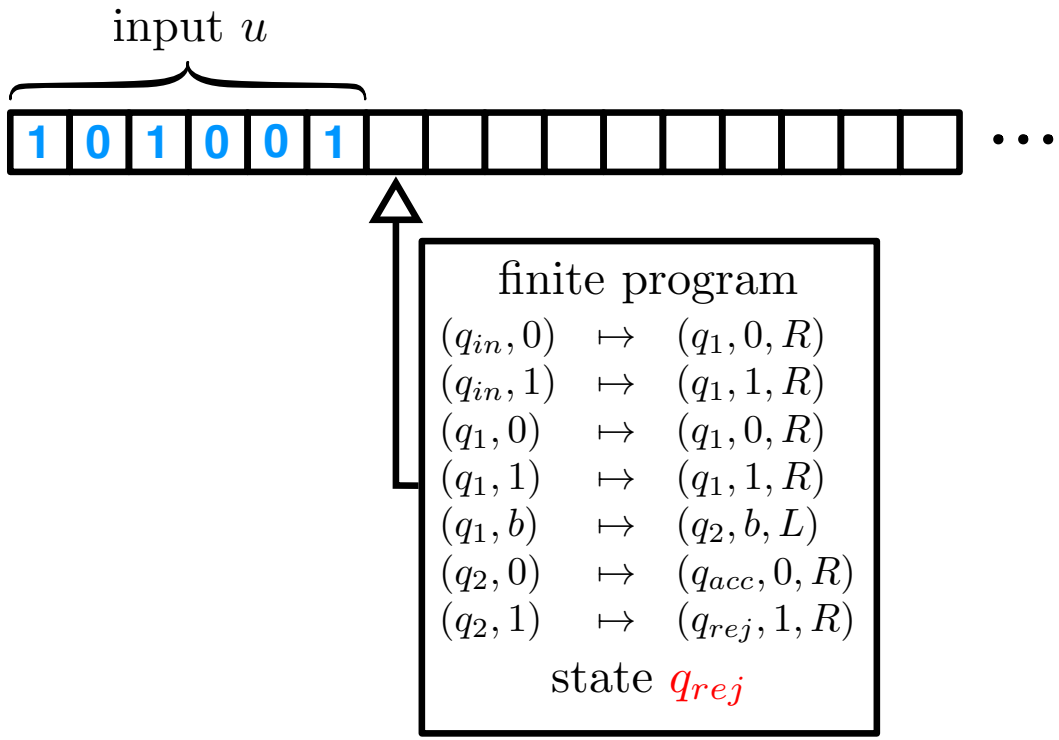
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

Turing Machine

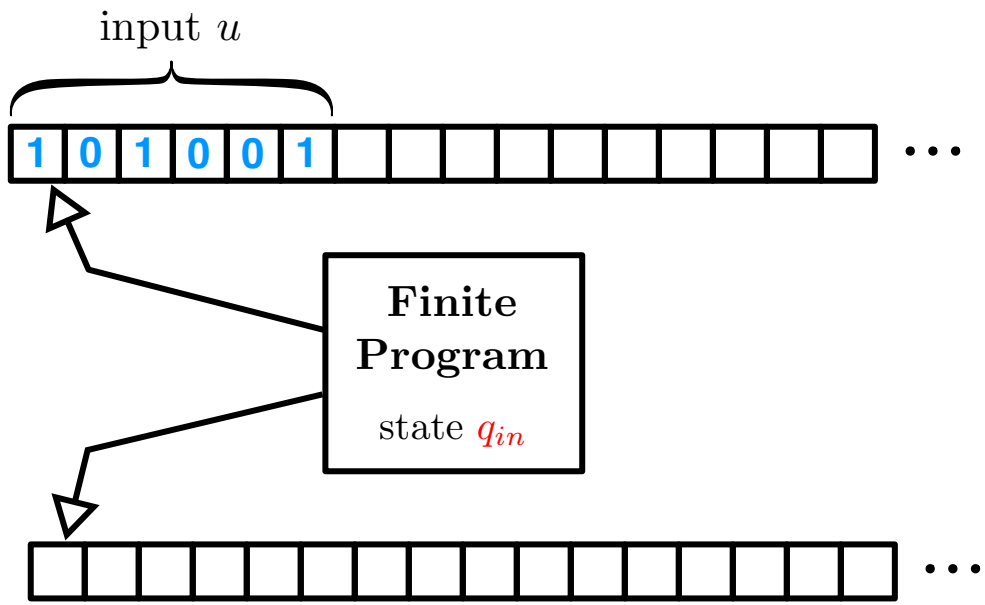
A *Turing machine* (TM) consists of an infinite tape, a read-write head, and a finite program.



- input u is *accepted* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{acc}
- input u is *rejected* by $\mathcal{M}$ if $\mathcal{M}(u)$ reaches the state q_{rej}

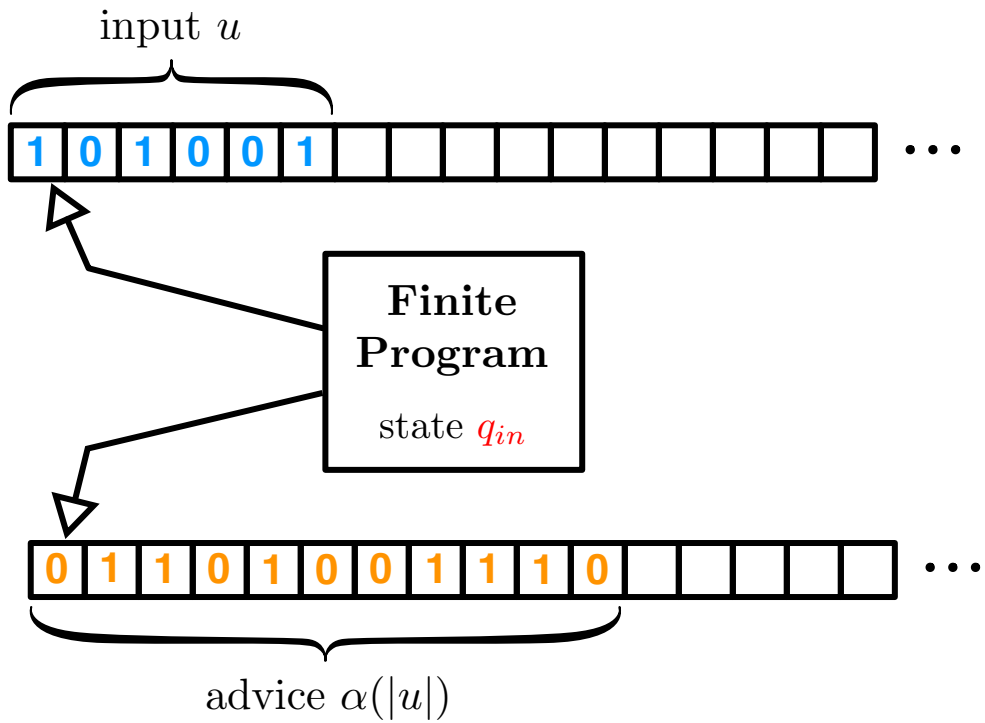
Turing Machine with Advice

A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



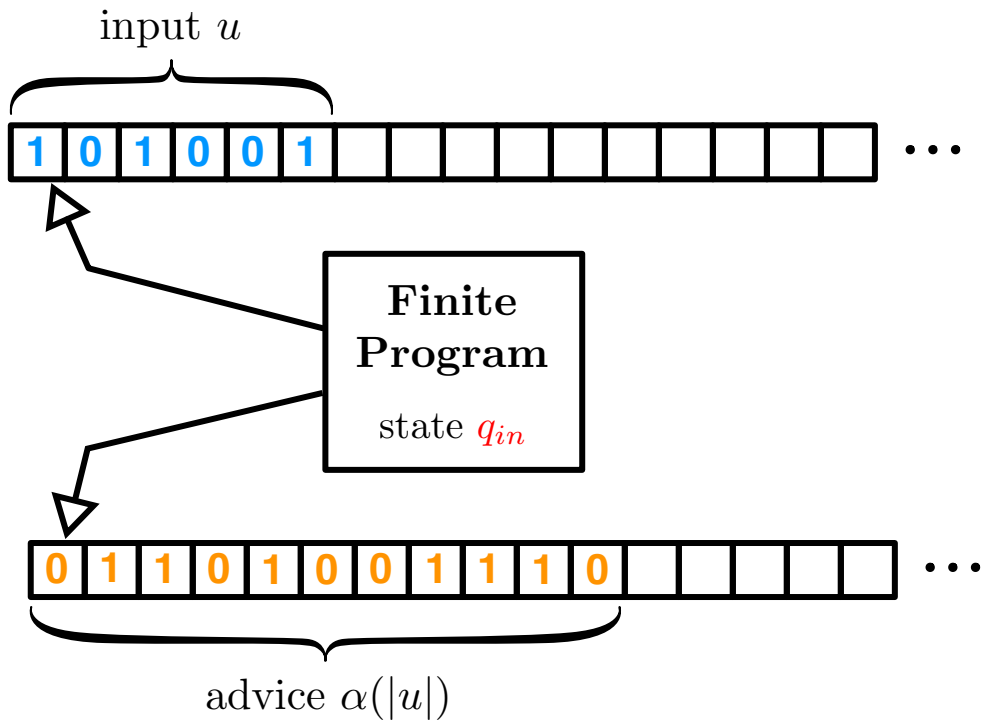
Turing Machine with Advice

A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



Turing Machine with Advice

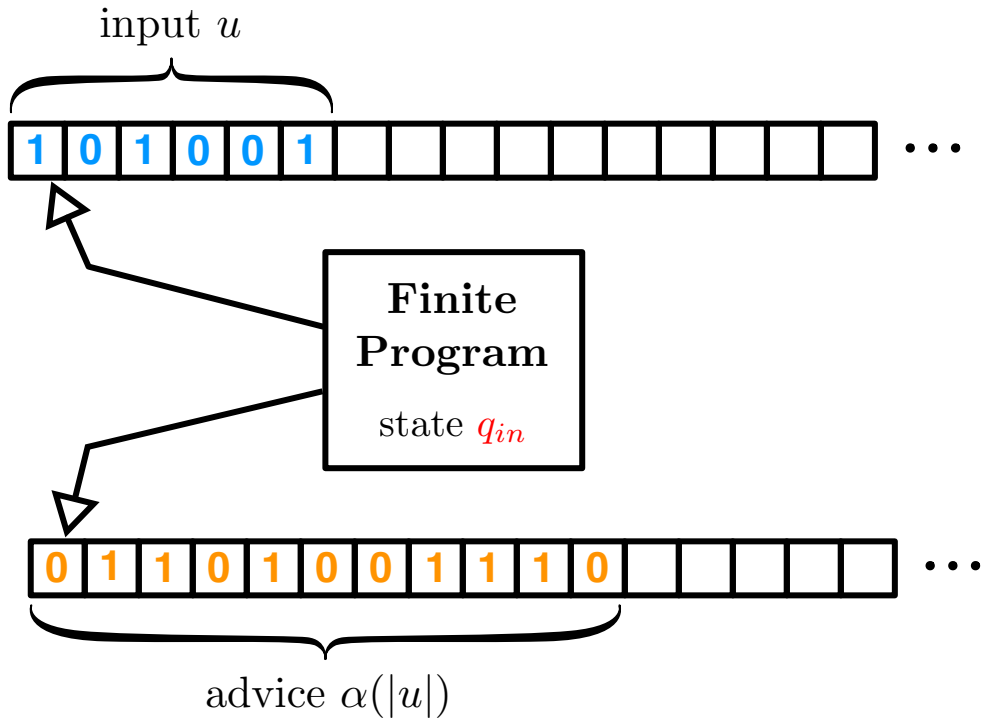
A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.



- The class of languages recognized in polynomial time by Turing machines with polynomial advices (TM/poly(A)) is **P/poly**.

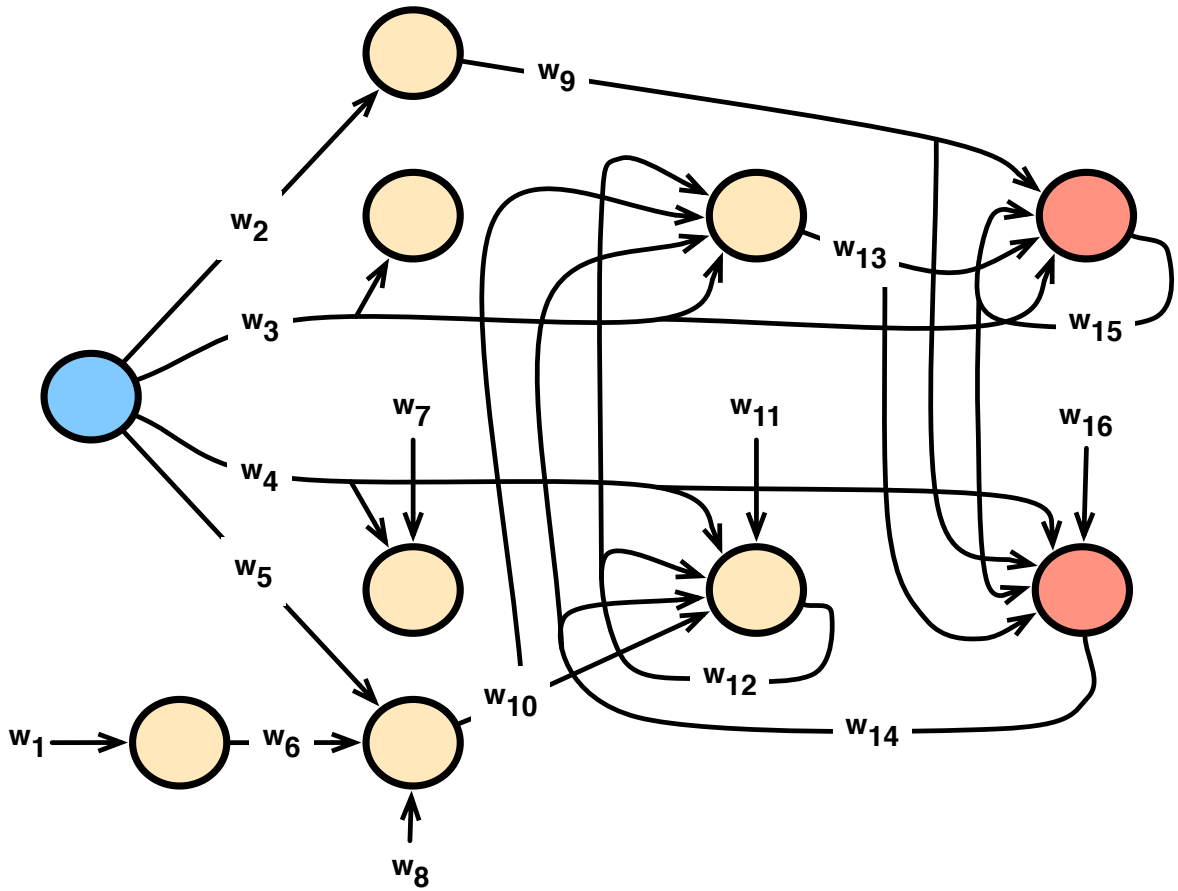
Turing Machine with Advice

A *Turing machine with advice* (TM/A) is a TM provided with an additional advice tape and advice function $\alpha : \mathbb{N} \longrightarrow \{0,1\}^*$.

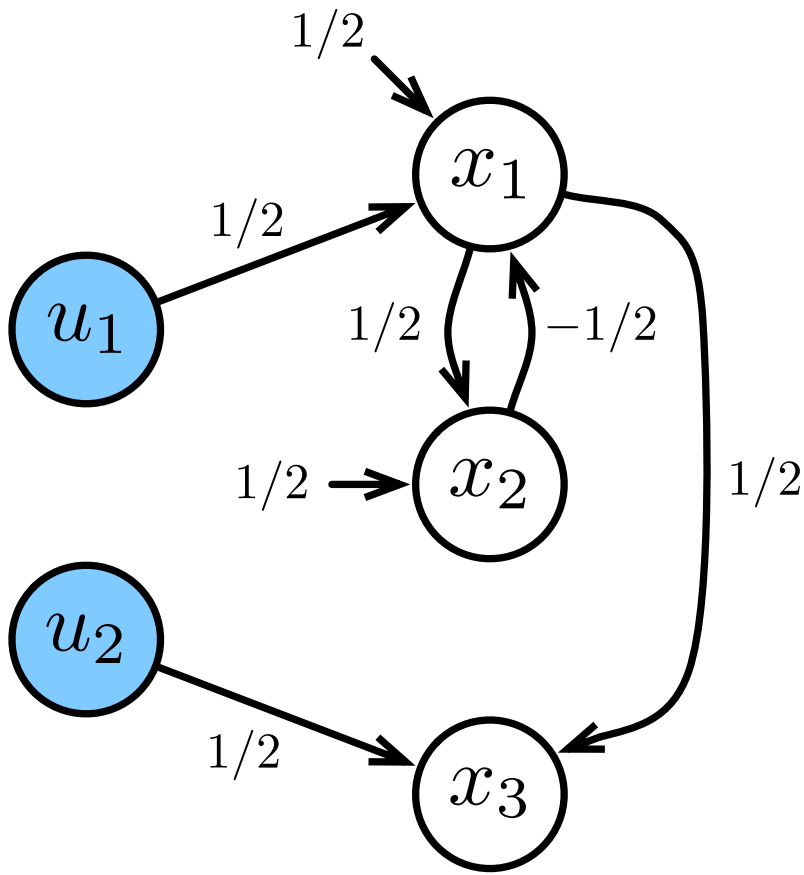


- TM/As are strictly more powerful than TMs: **P/poly** $\supsetneq$ **P**
  They are *super-Turing*...

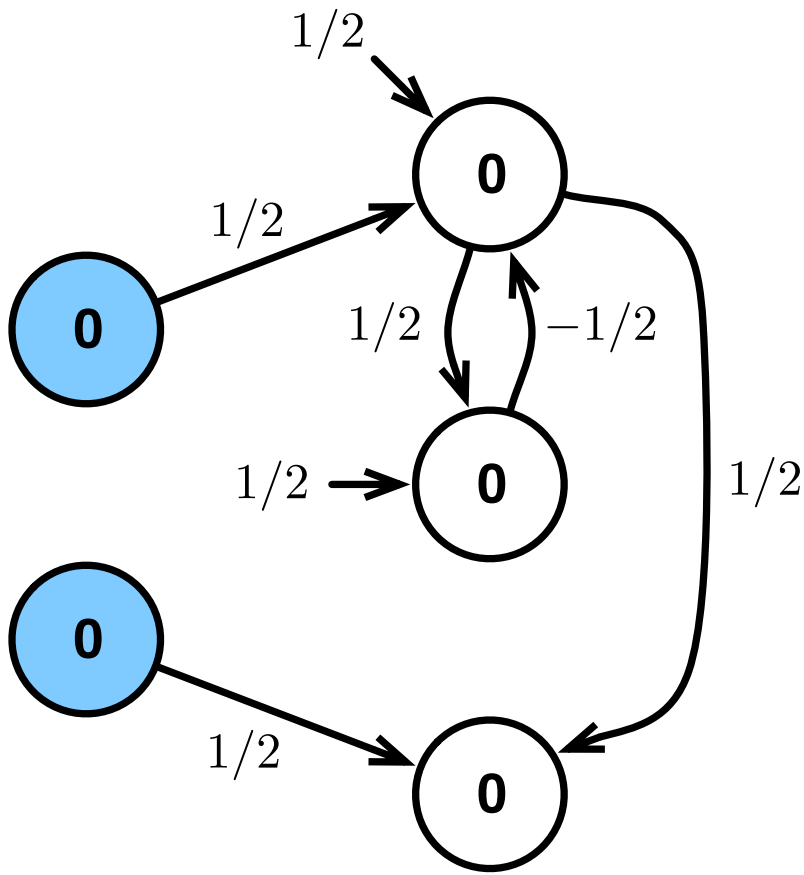
Recurrent Neural Network



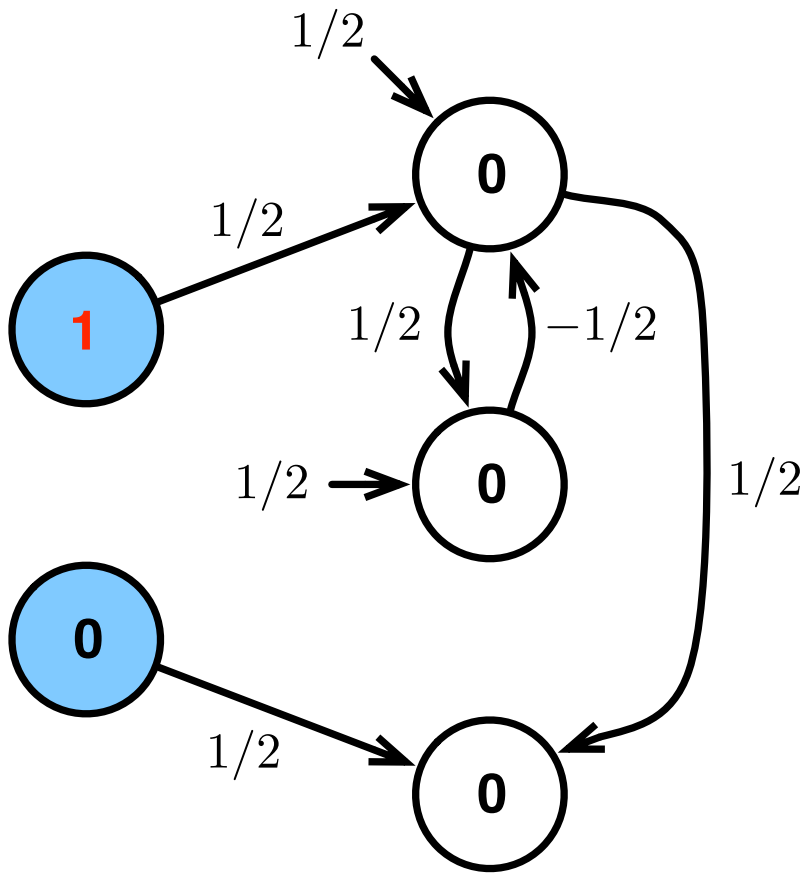
Boolean Recurrent Neural Networks



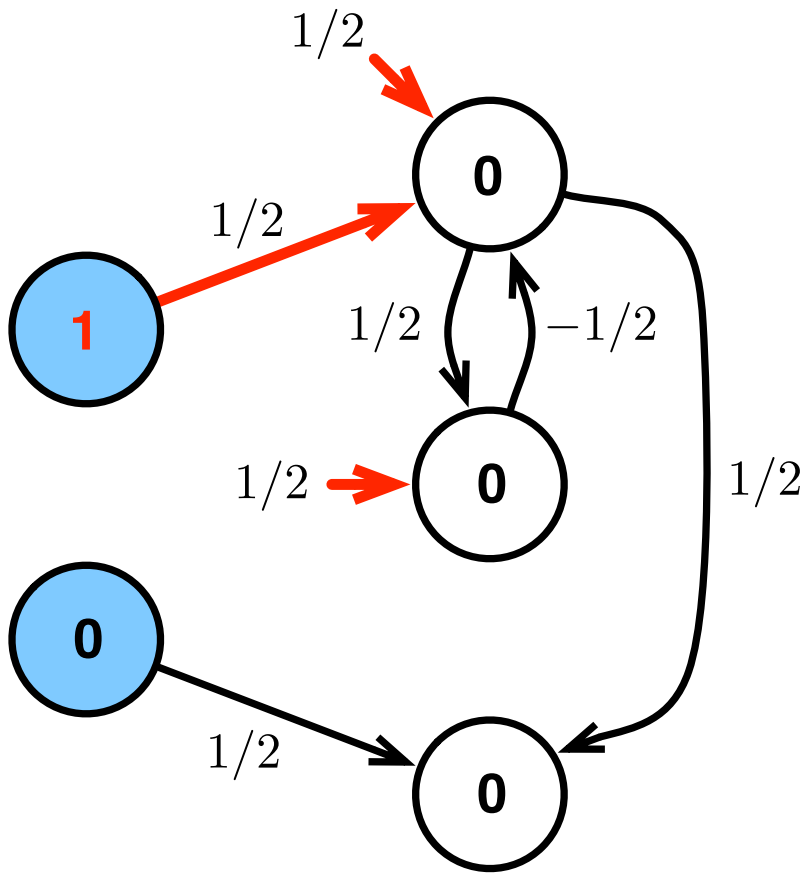
Boolean Recurrent Neural Networks



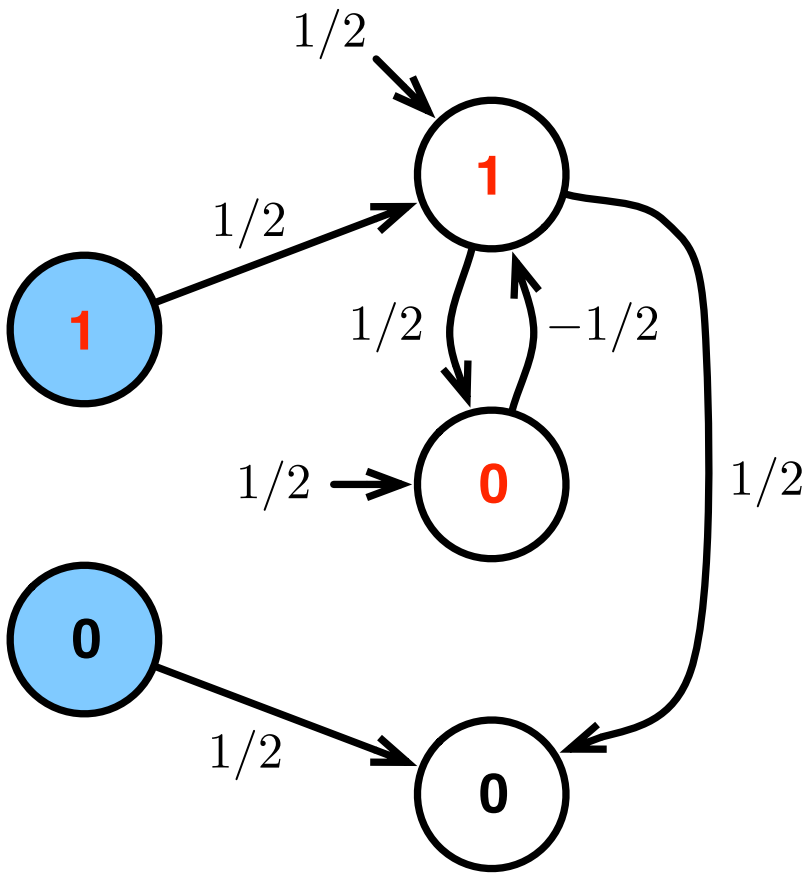
Boolean Recurrent Neural Networks



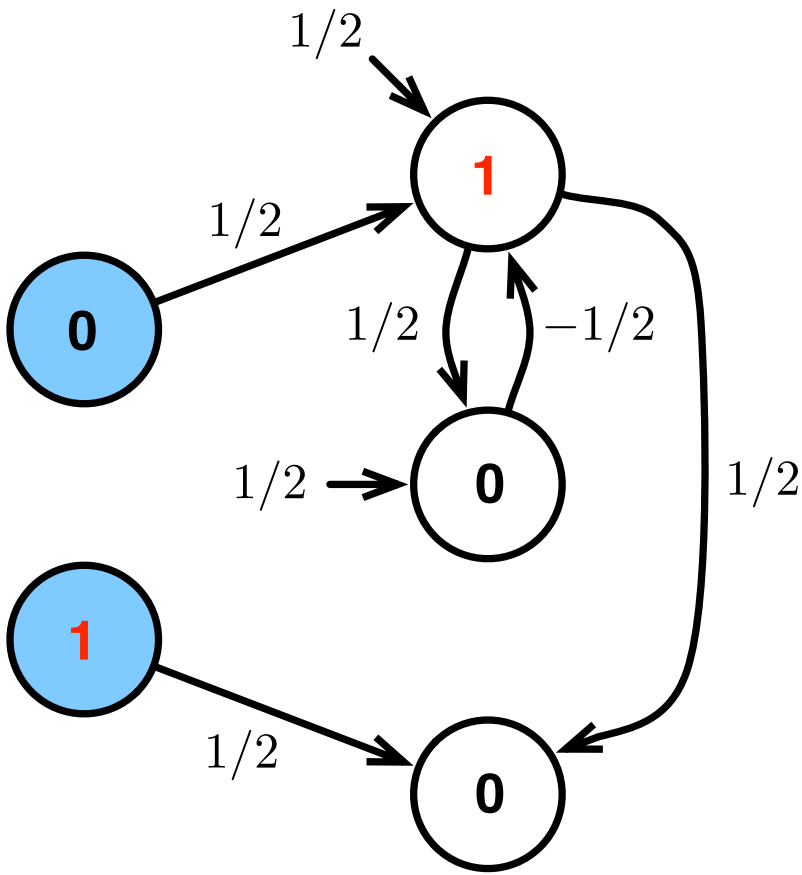
Boolean Recurrent Neural Networks



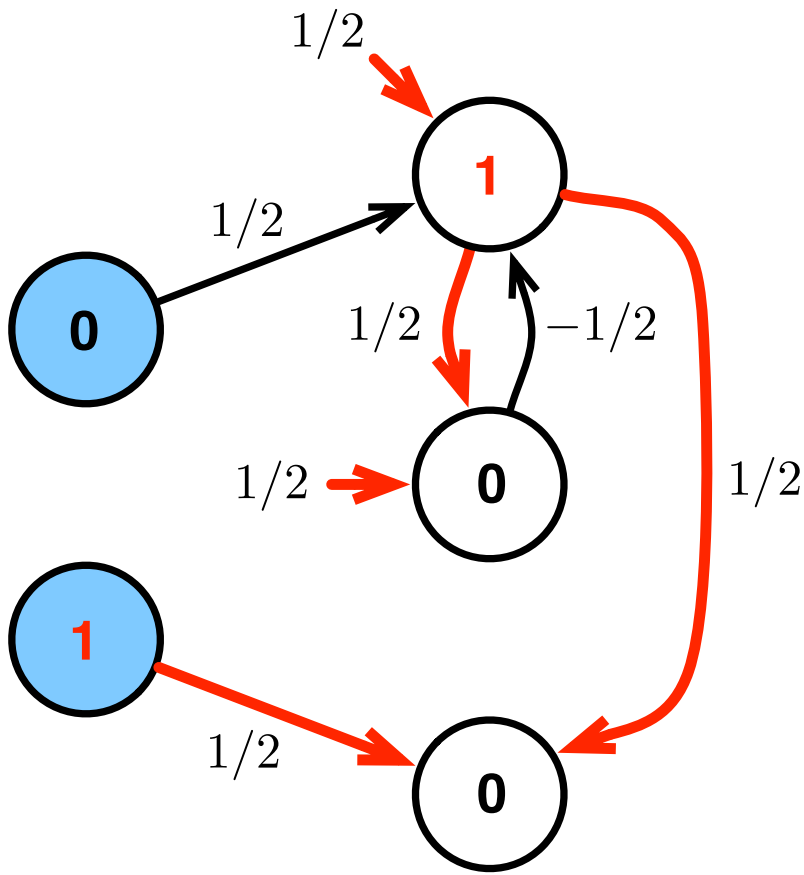
Boolean Recurrent Neural Networks



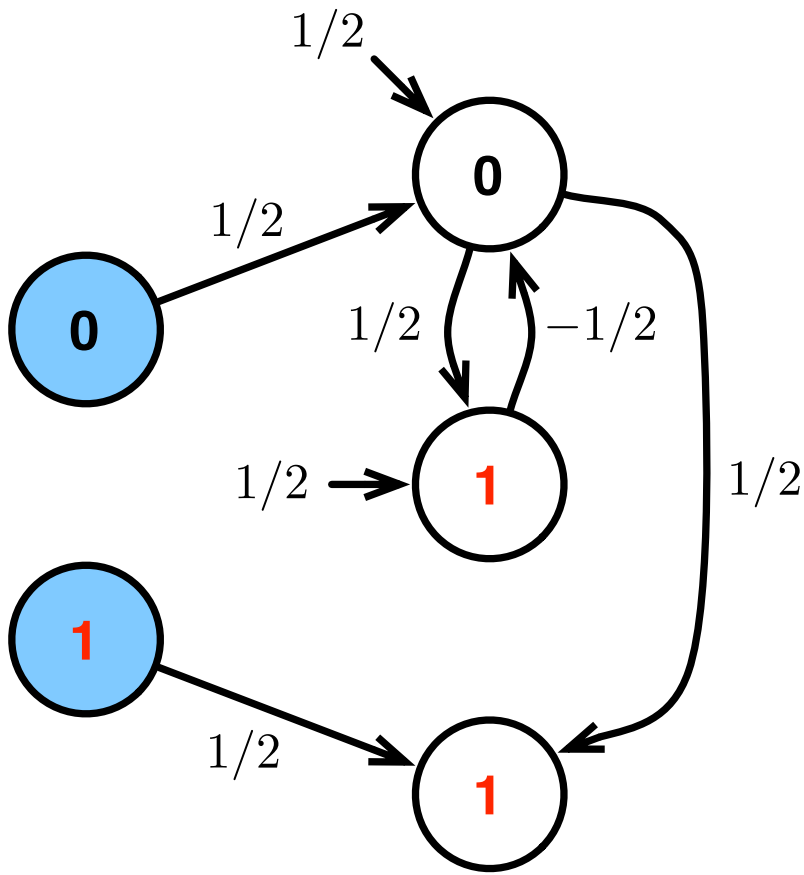
Boolean Recurrent Neural Networks



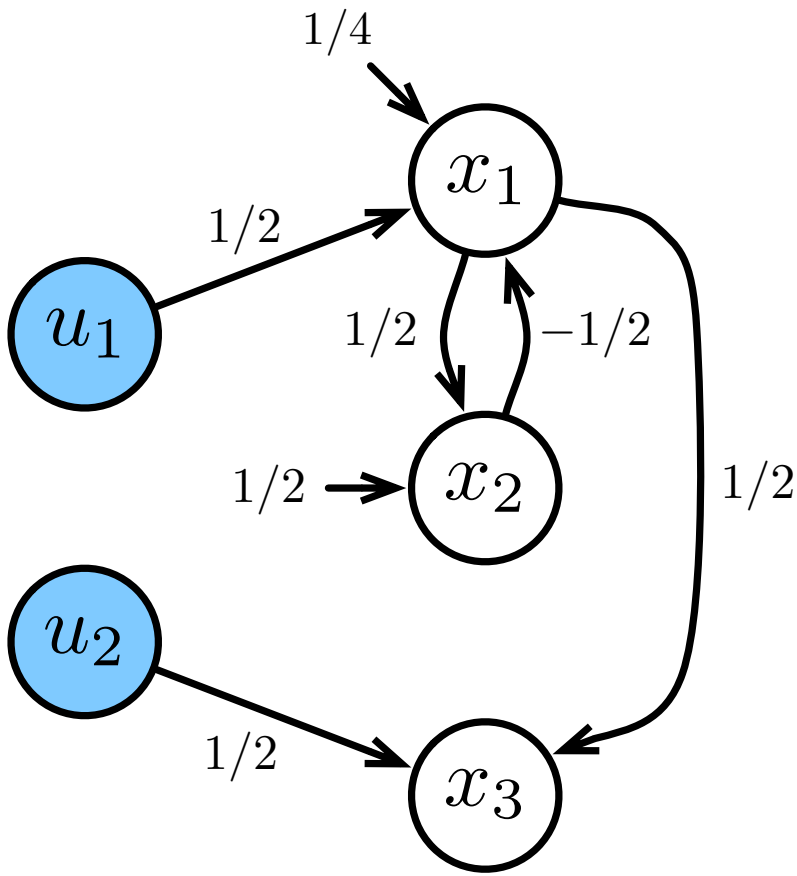
Boolean Recurrent Neural Networks



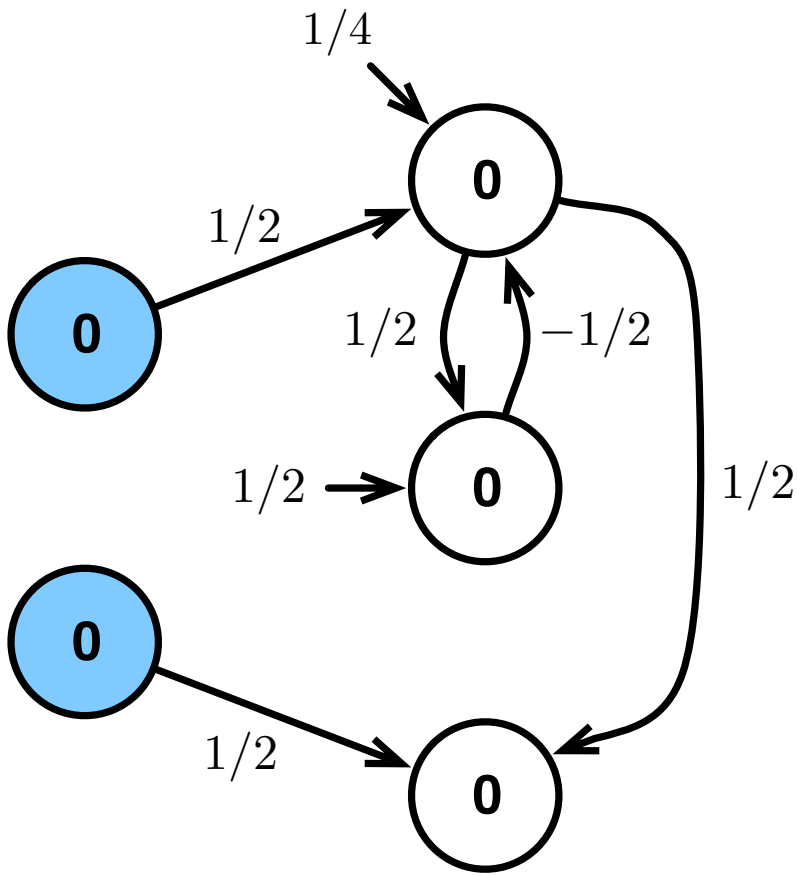
Boolean Recurrent Neural Networks



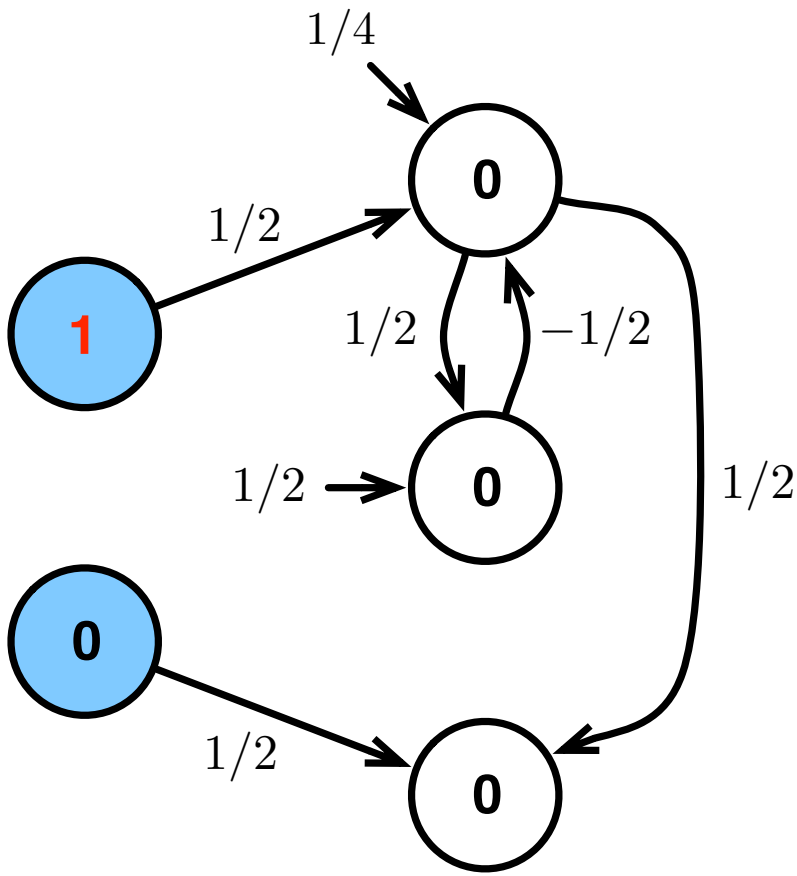
Sigmoid Recurrent Neural Networks



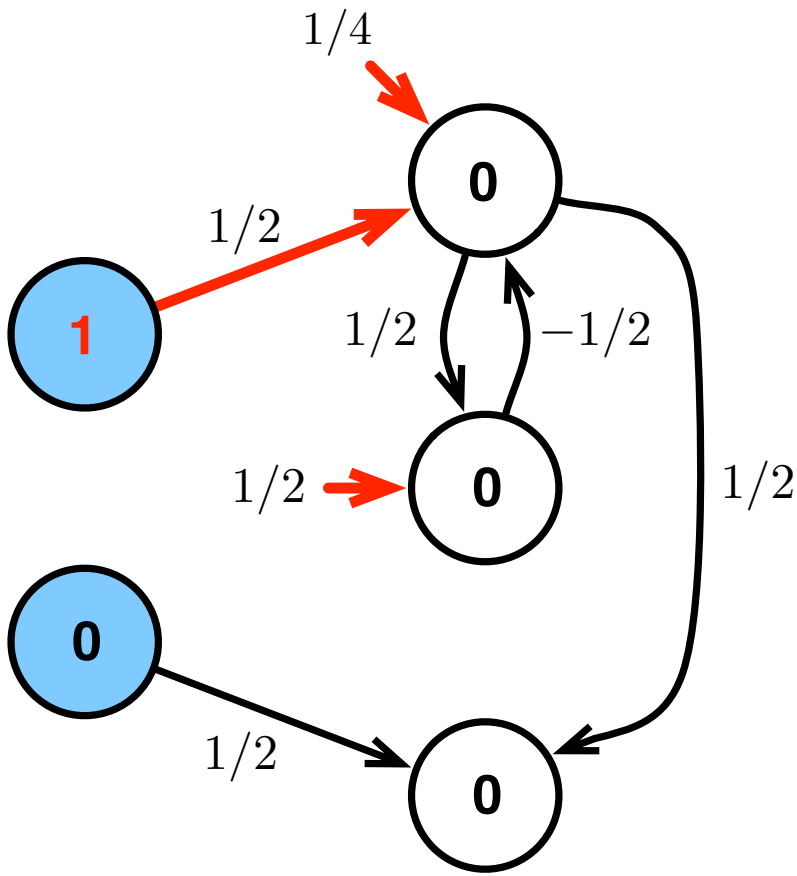
Sigmoid Recurrent Neural Networks



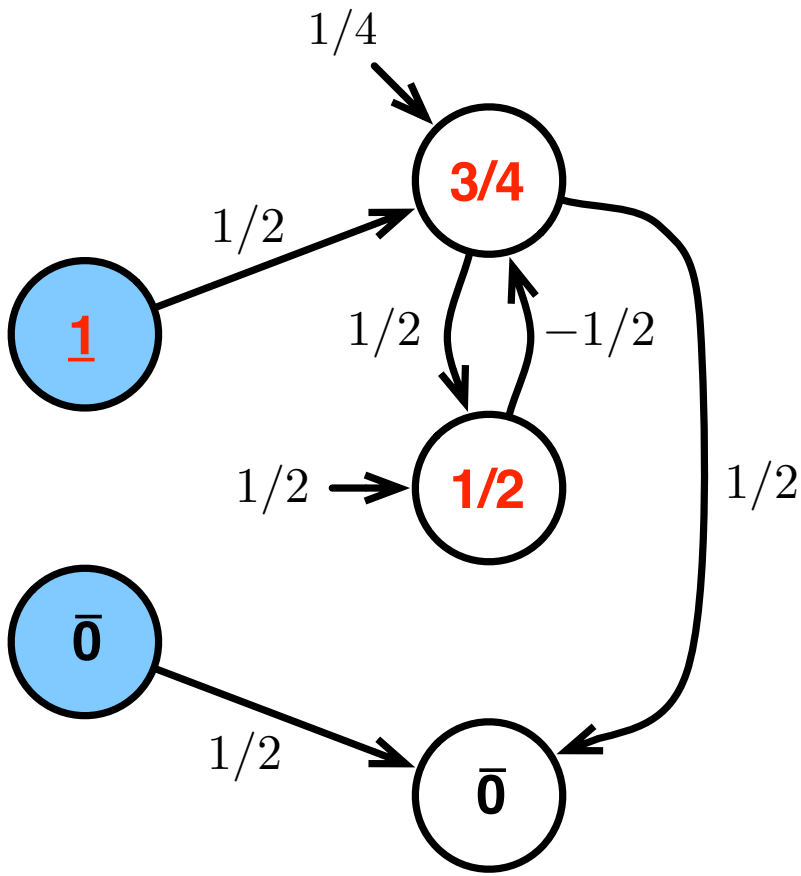
Sigmoid Recurrent Neural Networks



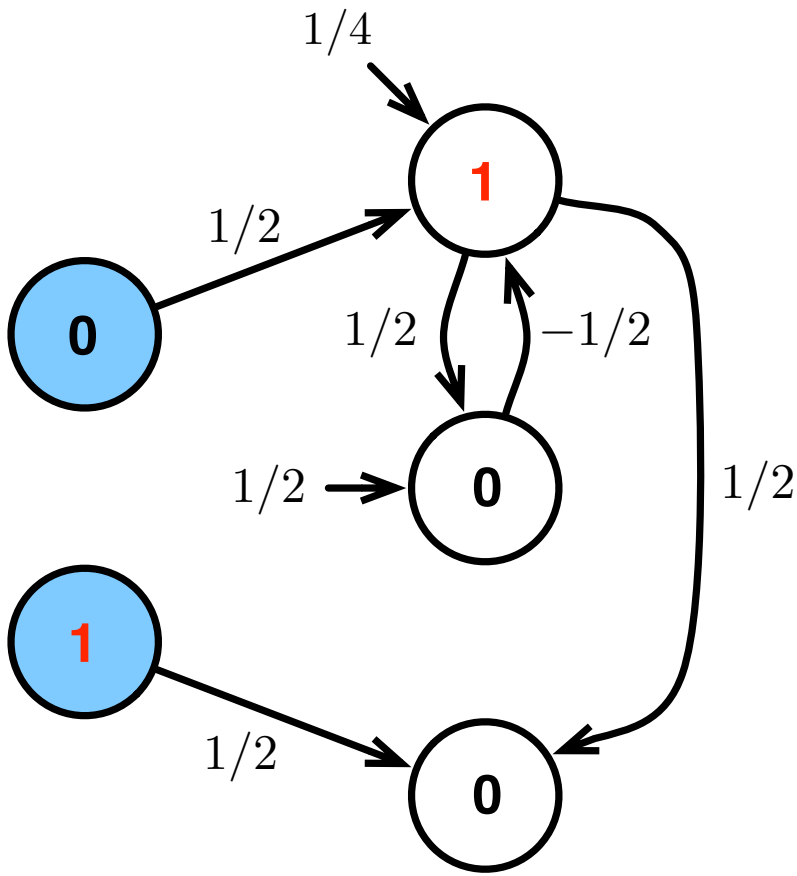
Sigmoid Recurrent Neural Networks



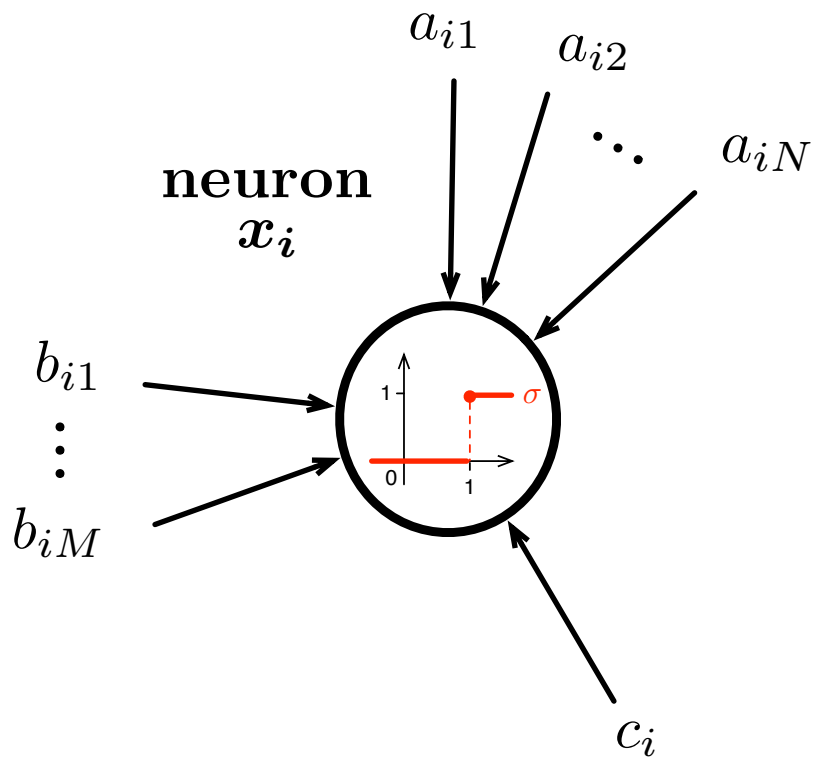
Sigmoid Recurrent Neural Networks



Sigmoid Recurrent Neural Networks

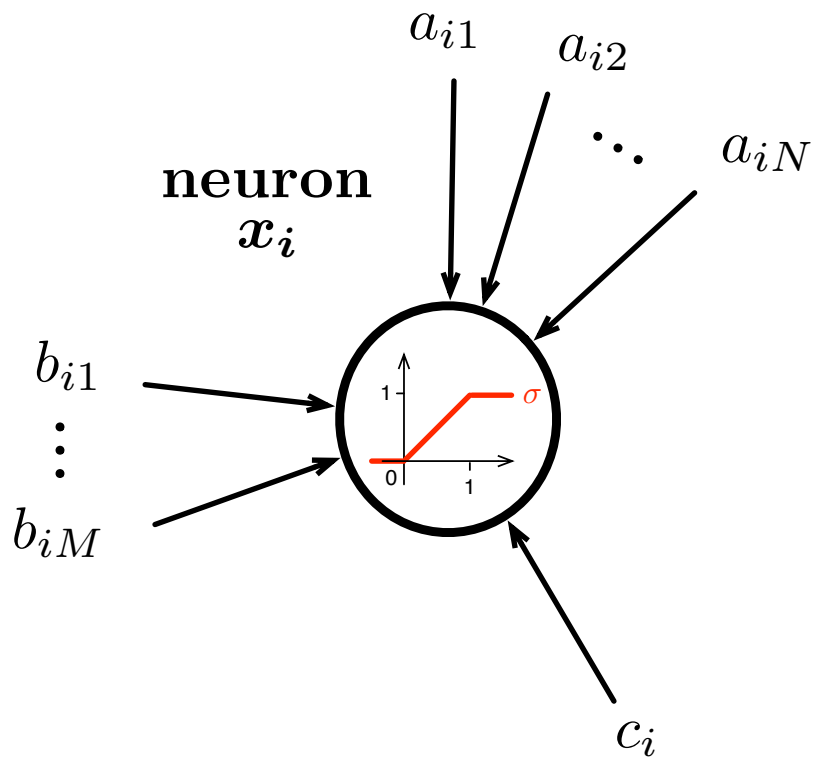


Boolean Recurrent Neural Networks



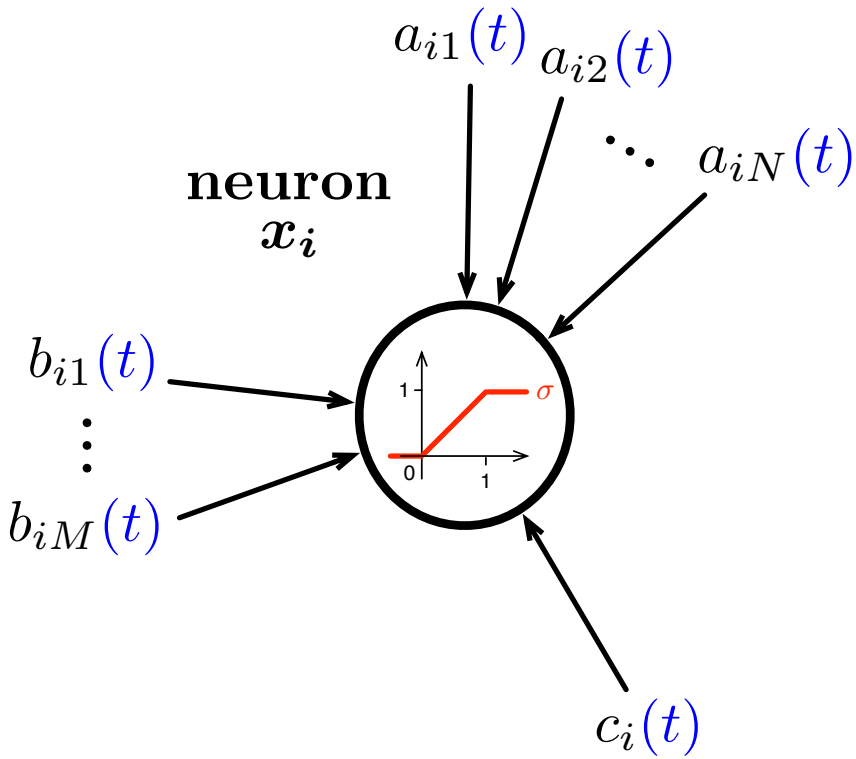
$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

Sigmoid Recurrent Neural Networks



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

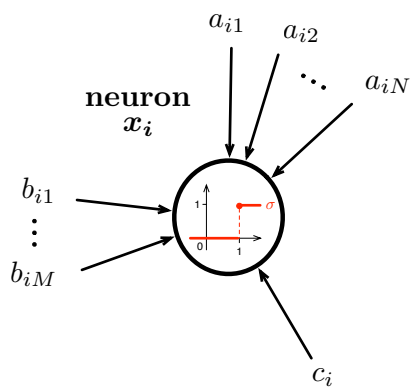
Sigmoid Recurrent Neural Networks



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

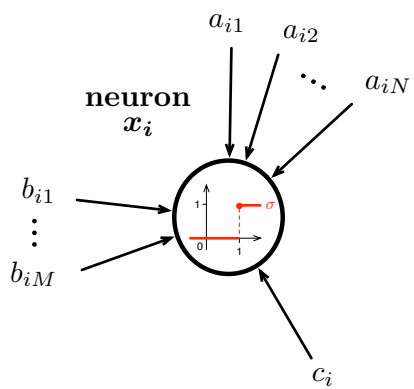
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



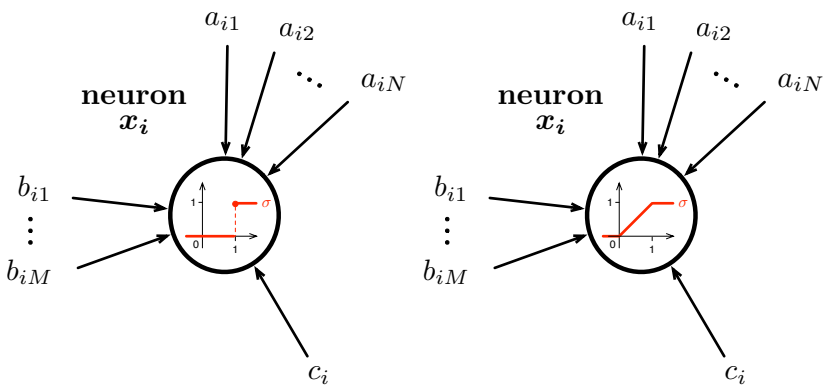
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



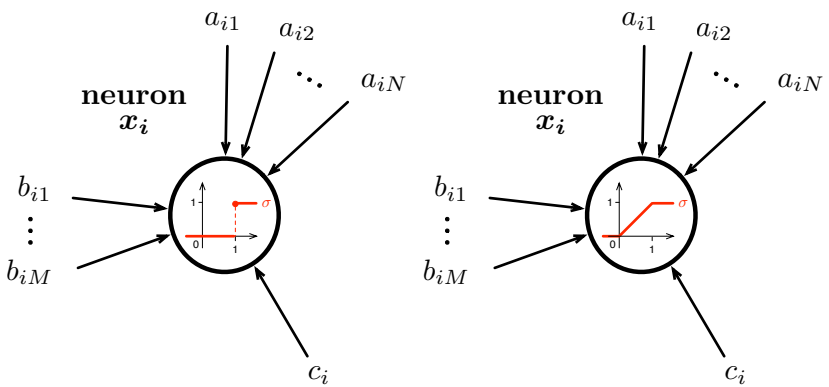
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



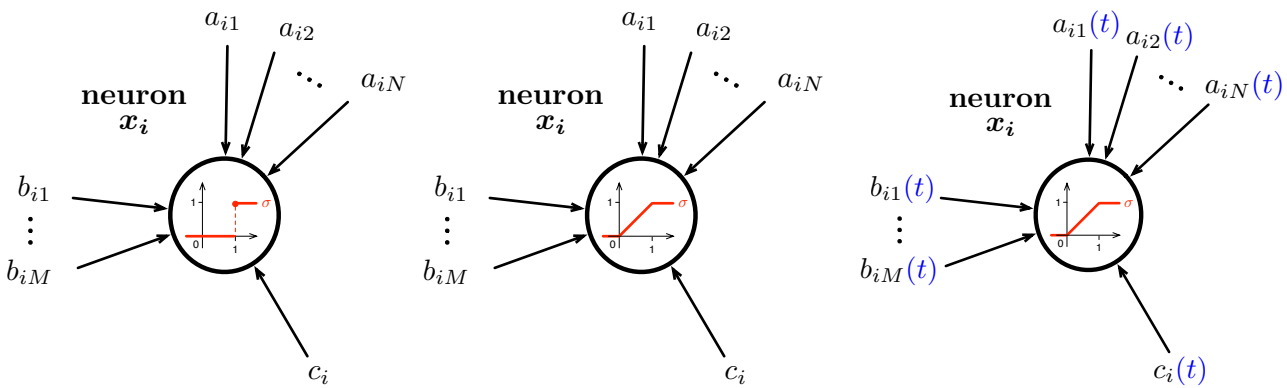
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



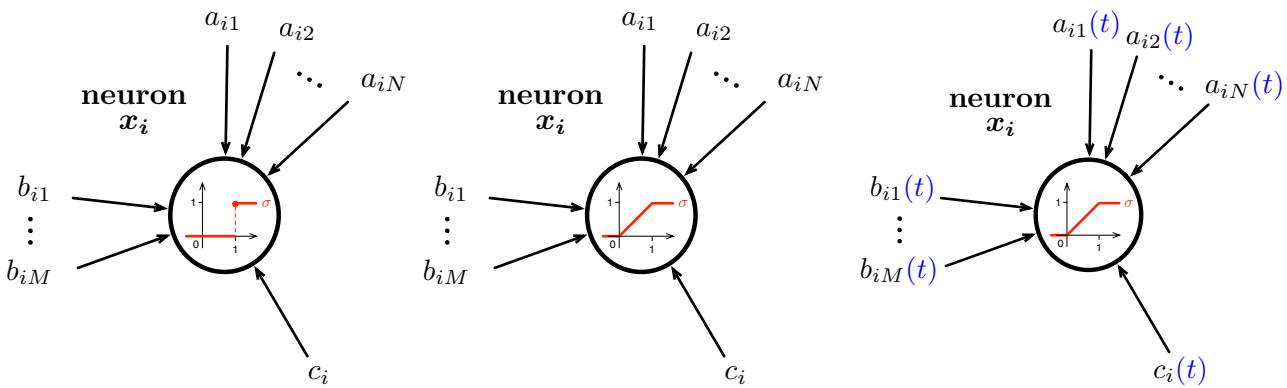
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



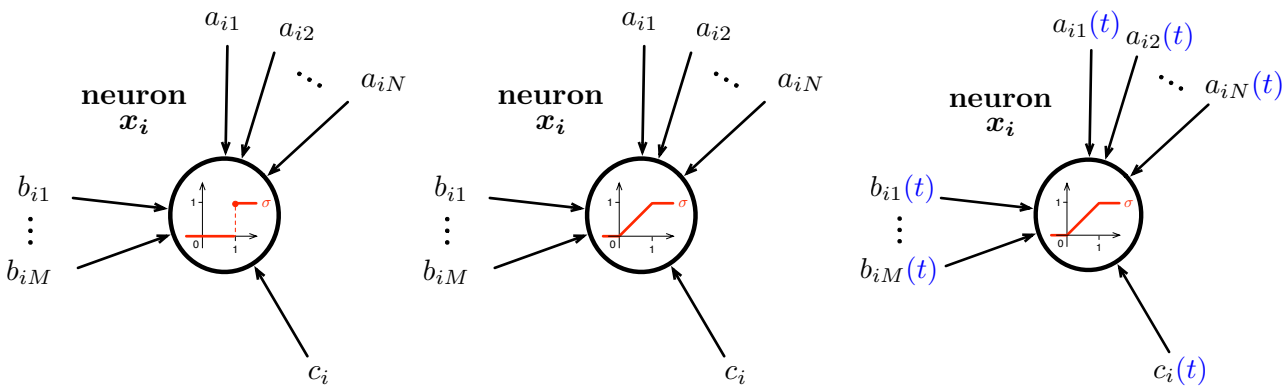
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



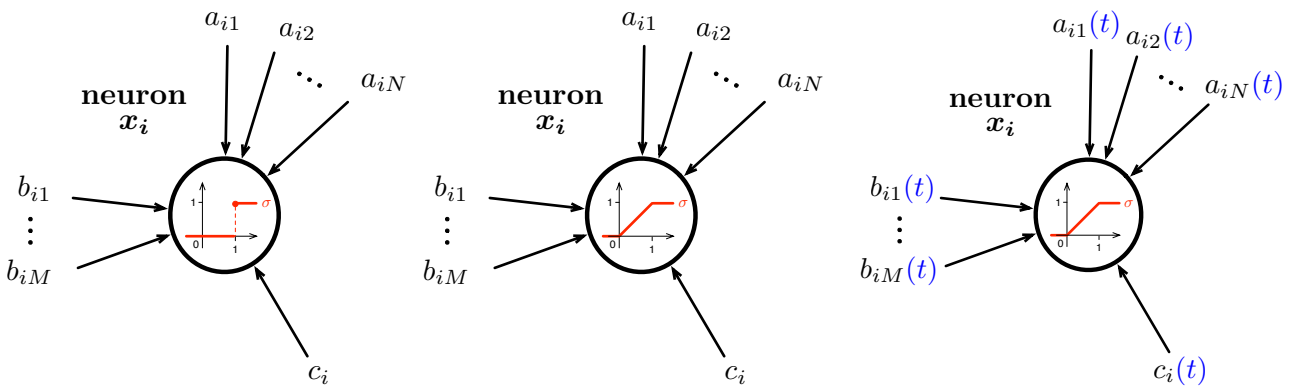
Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



Recurrent Neural Networks

1. Boolean rational RNNs: B-RNN[$\mathbb{Q}$]s
2. Boolean real RNNs: B-RNN[$\mathbb{R}$]s
3. Sigmoid static rational RNNs: St-RNN[$\mathbb{Q}$]s
4. Sigmoid static real RNNs: St-RNN[$\mathbb{R}$]s
5. Sigmoid bi-valued evolving rational RNNs: Ev$_2$-RNN[$\mathbb{Q}$]s
6. Sigmoid bi-valued evolving real RNNs: Ev$_2$-RNN[$\mathbb{R}$]s
7. Sigmoid general evolving rational RNNs: Ev-RNN[$\mathbb{Q}$]s
8. Sigmoid general evolving real N-RNNs: Ev-RNN[$\mathbb{R}$]s



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

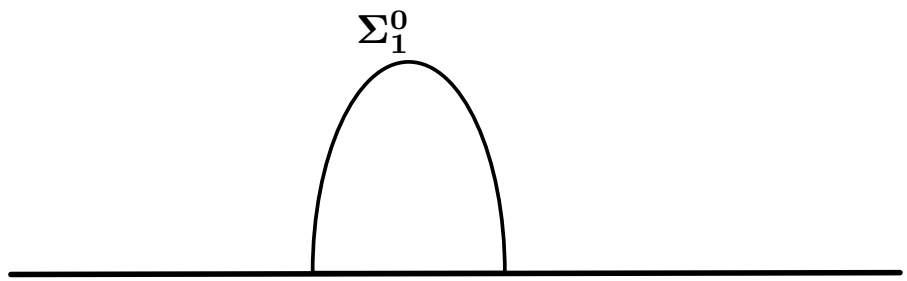
	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca &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

Results: classical computation

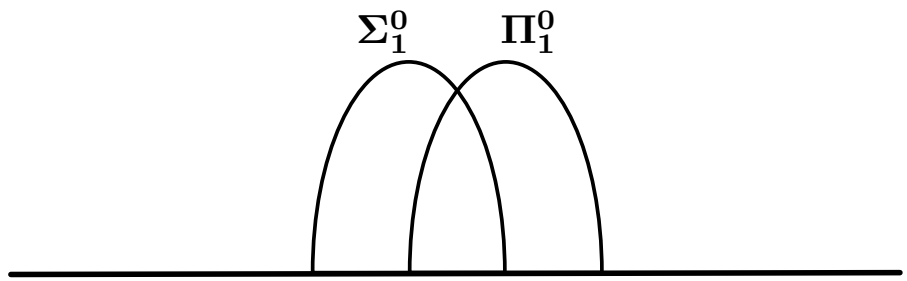
	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	FSA	TM	TM/poly(A)	TM/poly(A)
	REG	**P**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 95	Ca & Si 11,14	Ca & Si 11,14
$\mathbb{R}$	FSA	TM/poly(A)	TM/poly(A)	TM/poly(A)
	REG	**P/poly**	**P/poly**	**P/poly**
	Kl 56, Mi 67	Si & So 94	Ca &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

Infinite Computation

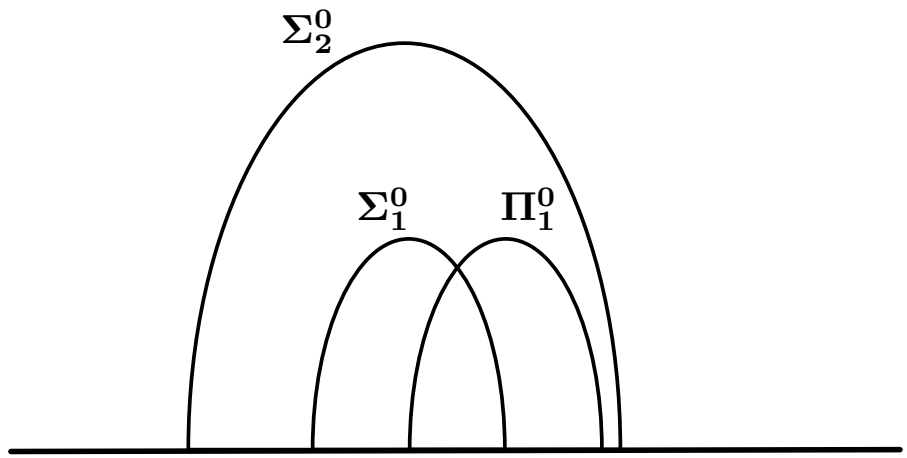
Infinite Computation



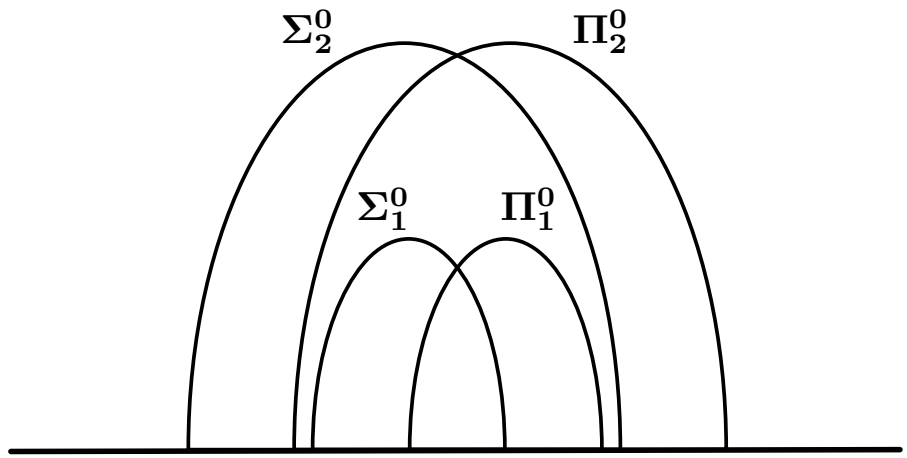
Infinite Computation



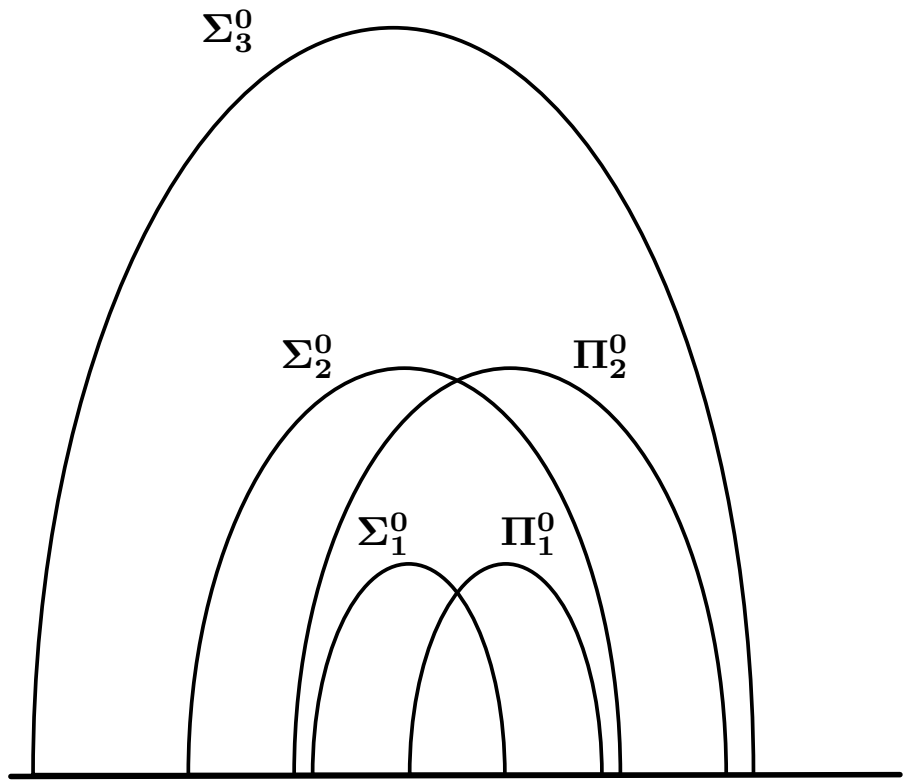
Infinite Computation



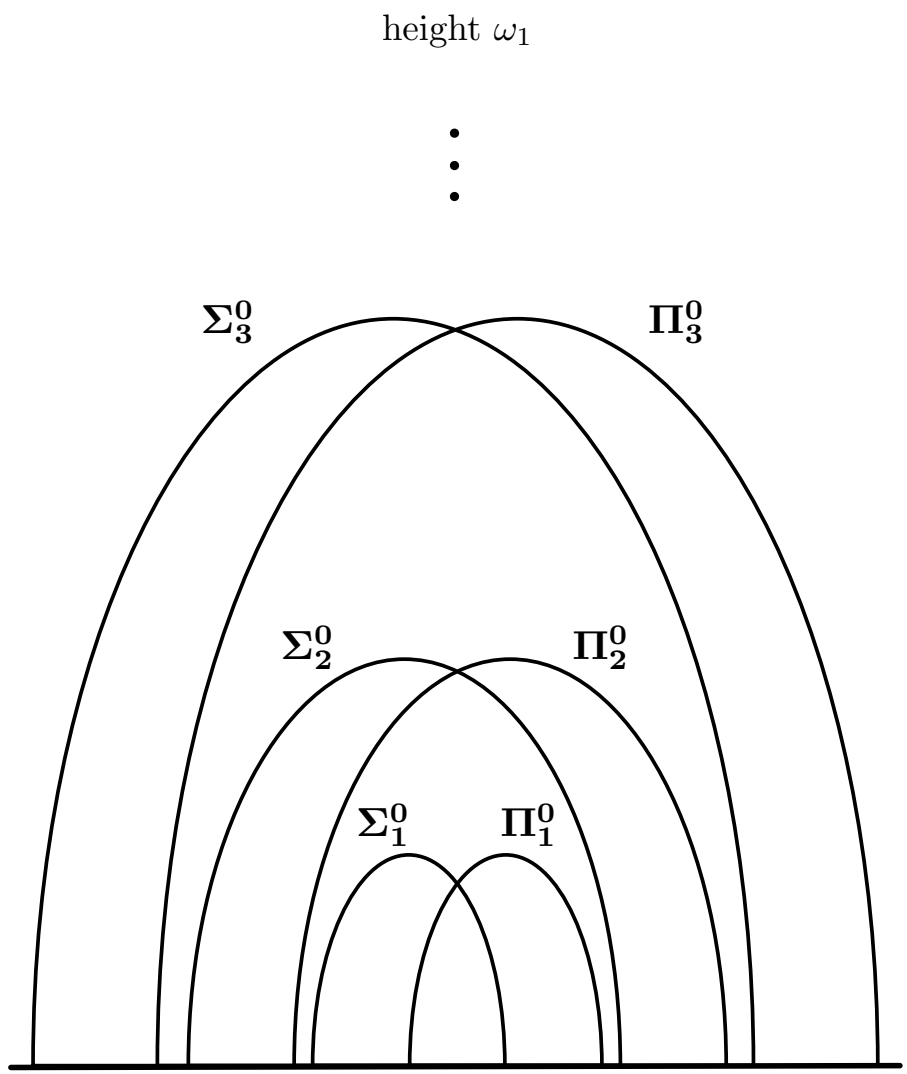
Infinite Computation



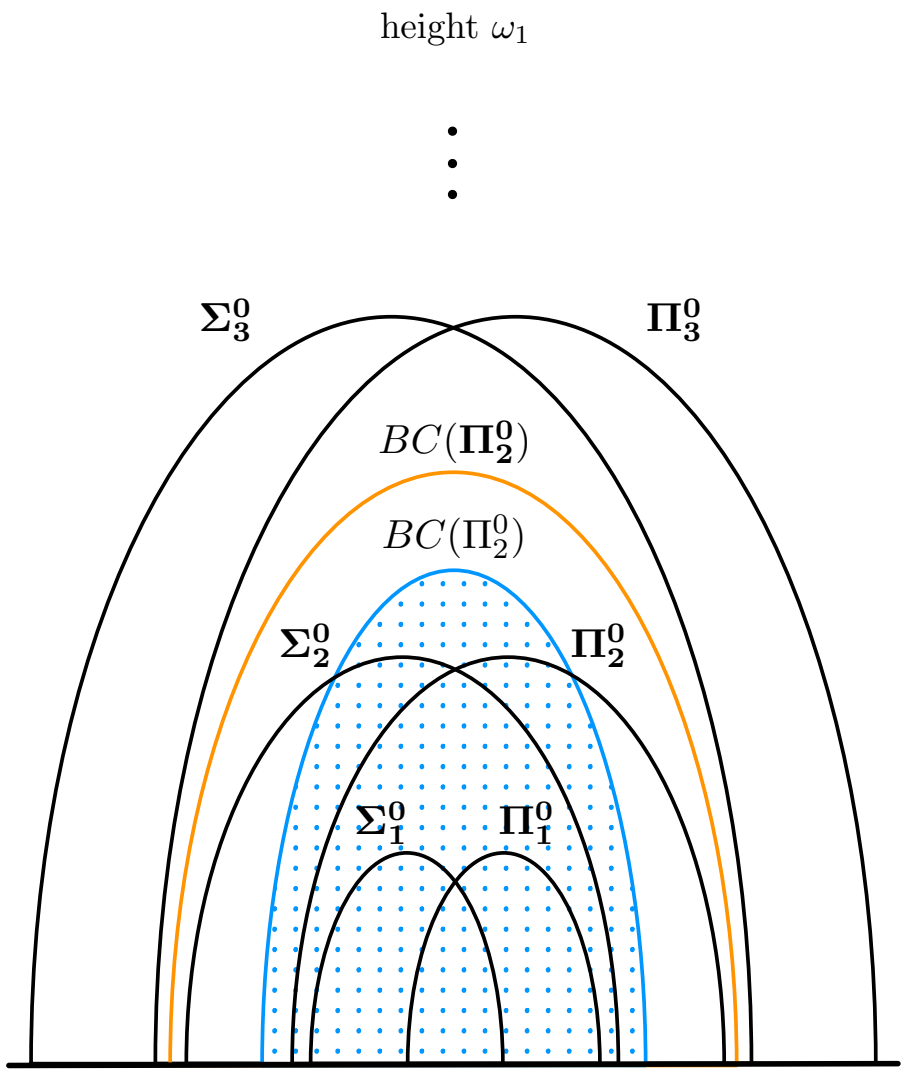
Infinite Computation



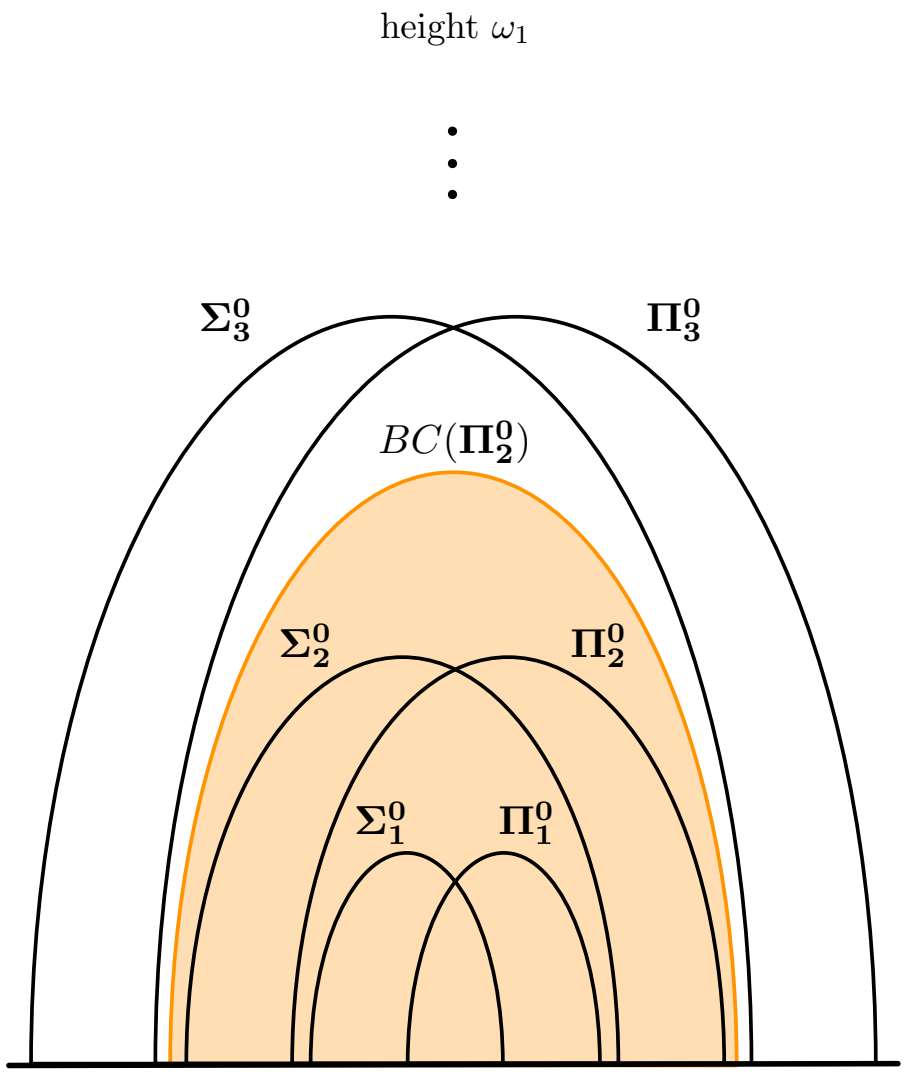
Infinite Computation



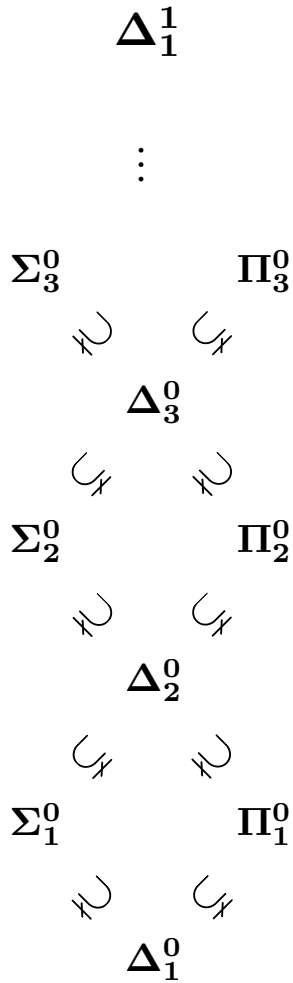
Infinite Computation



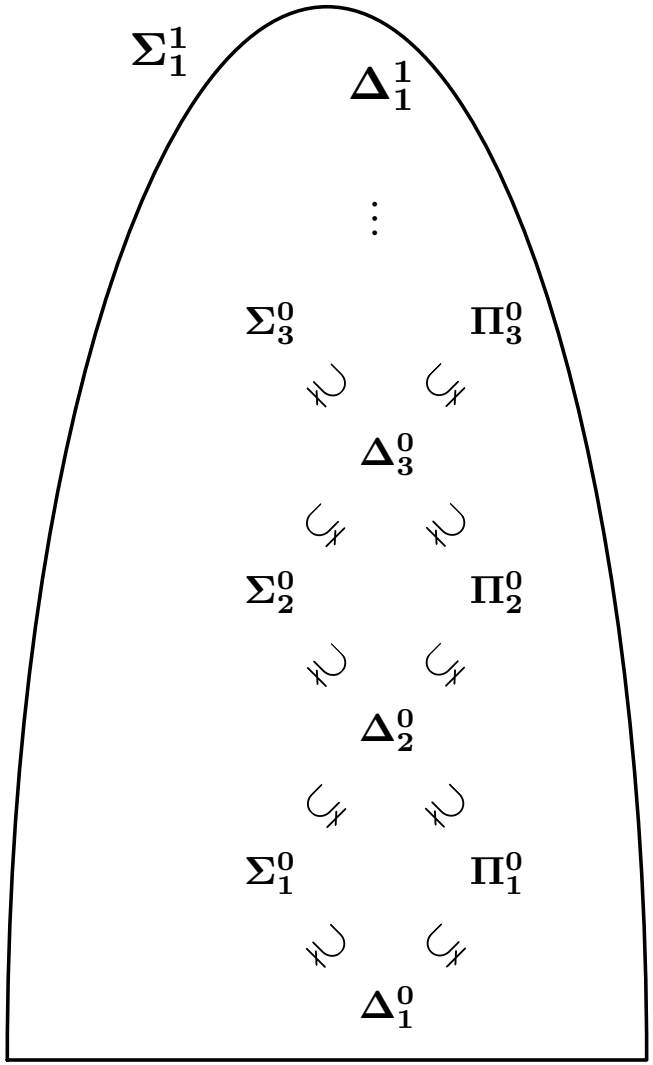
Infinite Computation



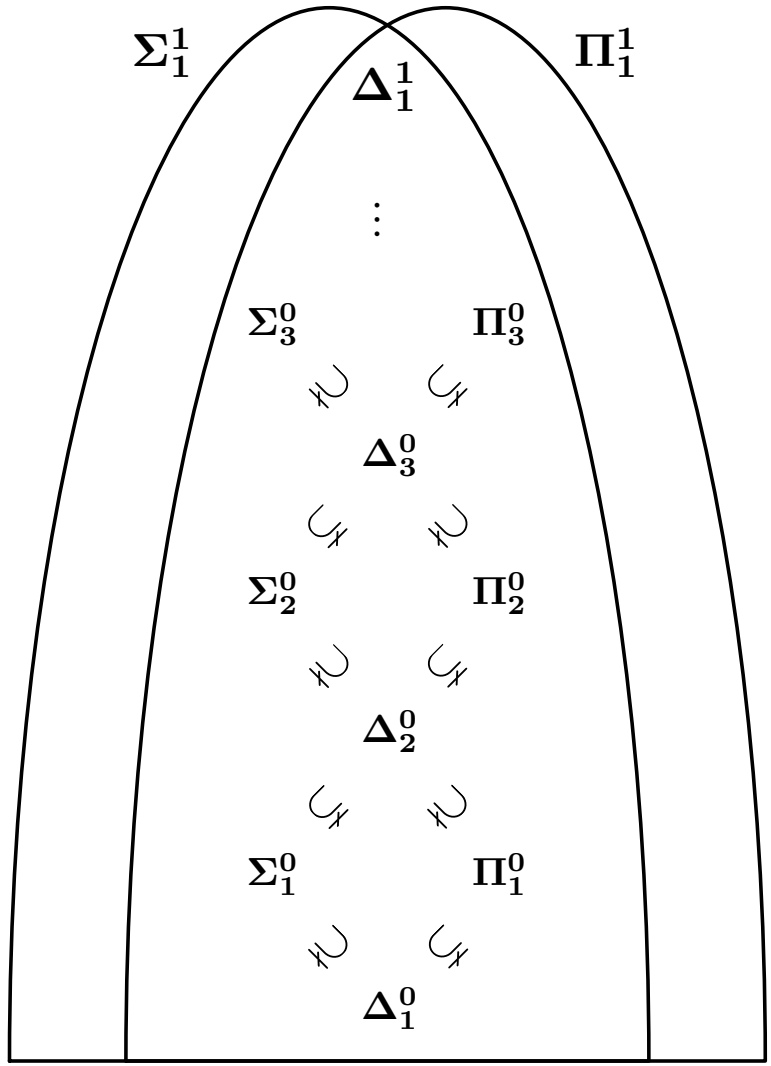
Infinite Computation



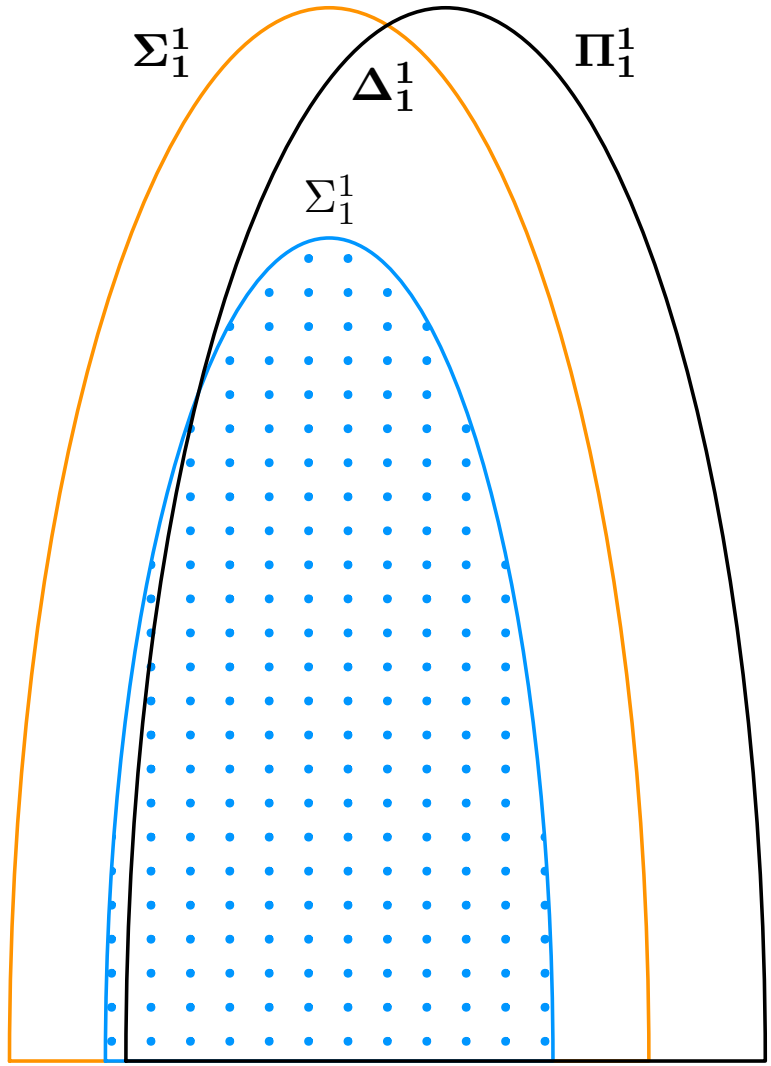
Infinite Computation



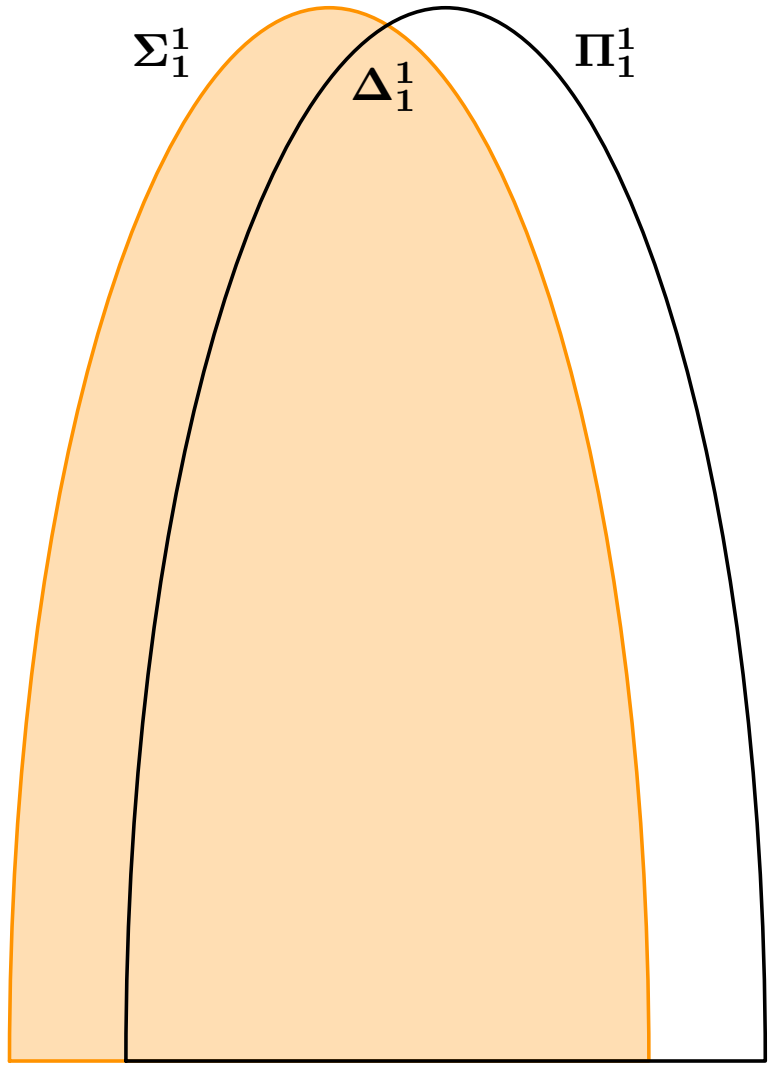
Infinite Computation



Infinite Computation



Infinite Computation



Results: infinite computation / det.

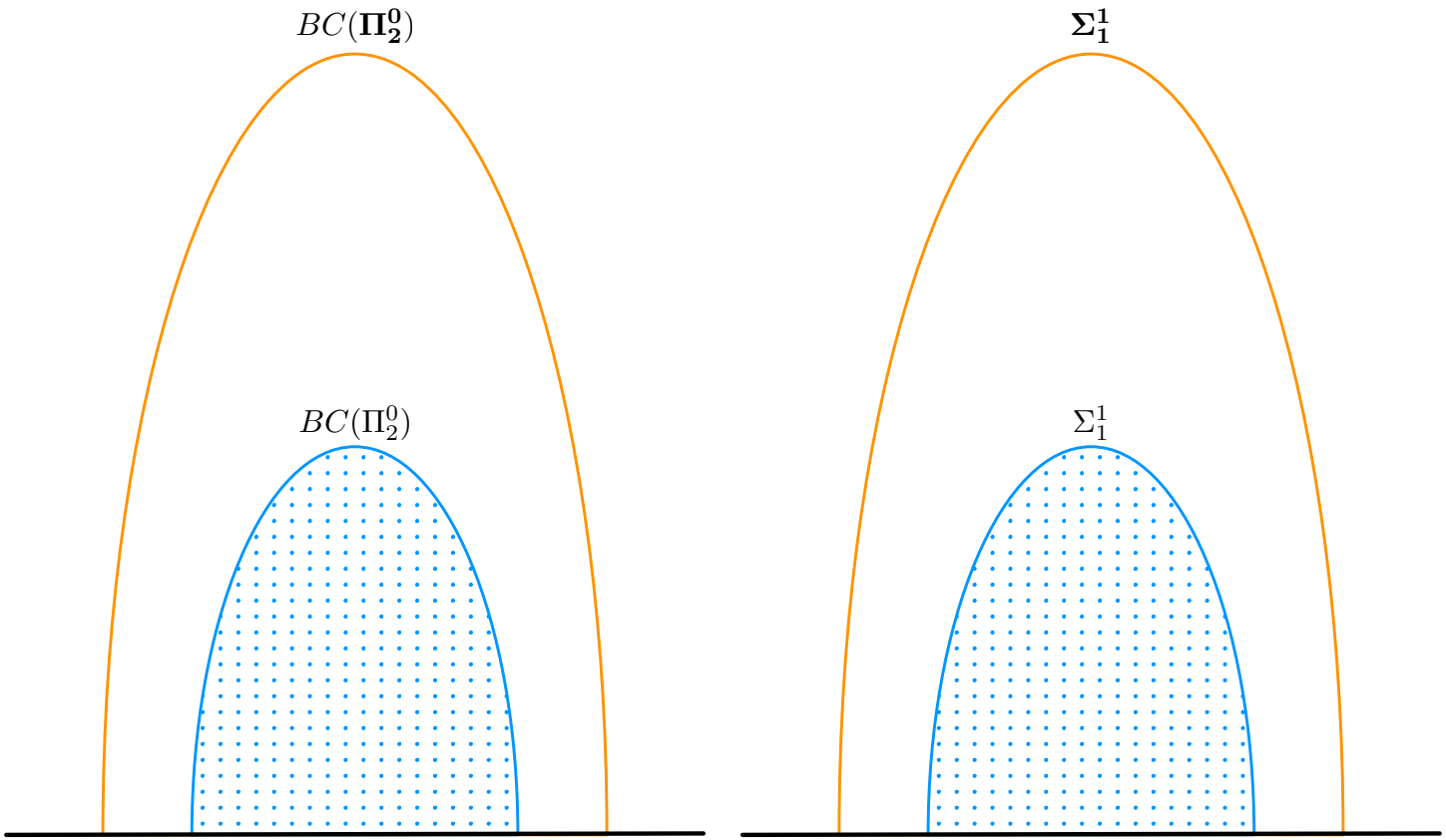
	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	Muller FSA $\in BC(\Pi_2^0)$ Ca & Vi 10,14	Muller TM $= BC(\Pi_2^0)$ Ca & Vi 15,16	super-Turing $= BC(\mathbf{\Pi_2^0})$ Ca & Vi 15,16	super-Turing $= BC(\mathbf{\Pi_2^0})$ Ca & Vi 15,16
$\mathbb{R}$	Muller FSA $\in BC(\Pi_2^0)$ Ca & Vi 10,14	super-Turing $= BC(\mathbf{\Pi_2^0})$ Ca & Vi 15,16	super-Turing $= BC(\mathbf{\Pi_2^0})$ Ca & Vi 15,16	super-Turing $= BC(\mathbf{\Pi_2^0})$ Ca & Vi 15,16

Results: infinite computation / nondet.

	Boolean	Sigmoid		
		Static	Bi-valued Evolving	Evolving
$\mathbb{Q}$	Muller FSA $\in \Sigma_1^1$ Ca & Vi 10,14	Muller TM $= \Sigma_1^1$ Ca & Vi 15,16	super-Turing $= \mathbf{\Sigma_1^1}$ Ca & Vi 15,16	super-Turing $= \mathbf{\Sigma_1^1}$ Ca & Vi 15,16
$\mathbb{R}$	Muller FSA $\in \Sigma_1^1$ Ca & Vi 10,14	super-Turing $= \mathbf{\Sigma_1^1}$ Ca & Vi 15,16	super-Turing $= \mathbf{\Sigma_1^1}$ Ca & Vi 15,16	super-Turing $= \mathbf{\Sigma_1^1}$ Ca & Vi 15,16

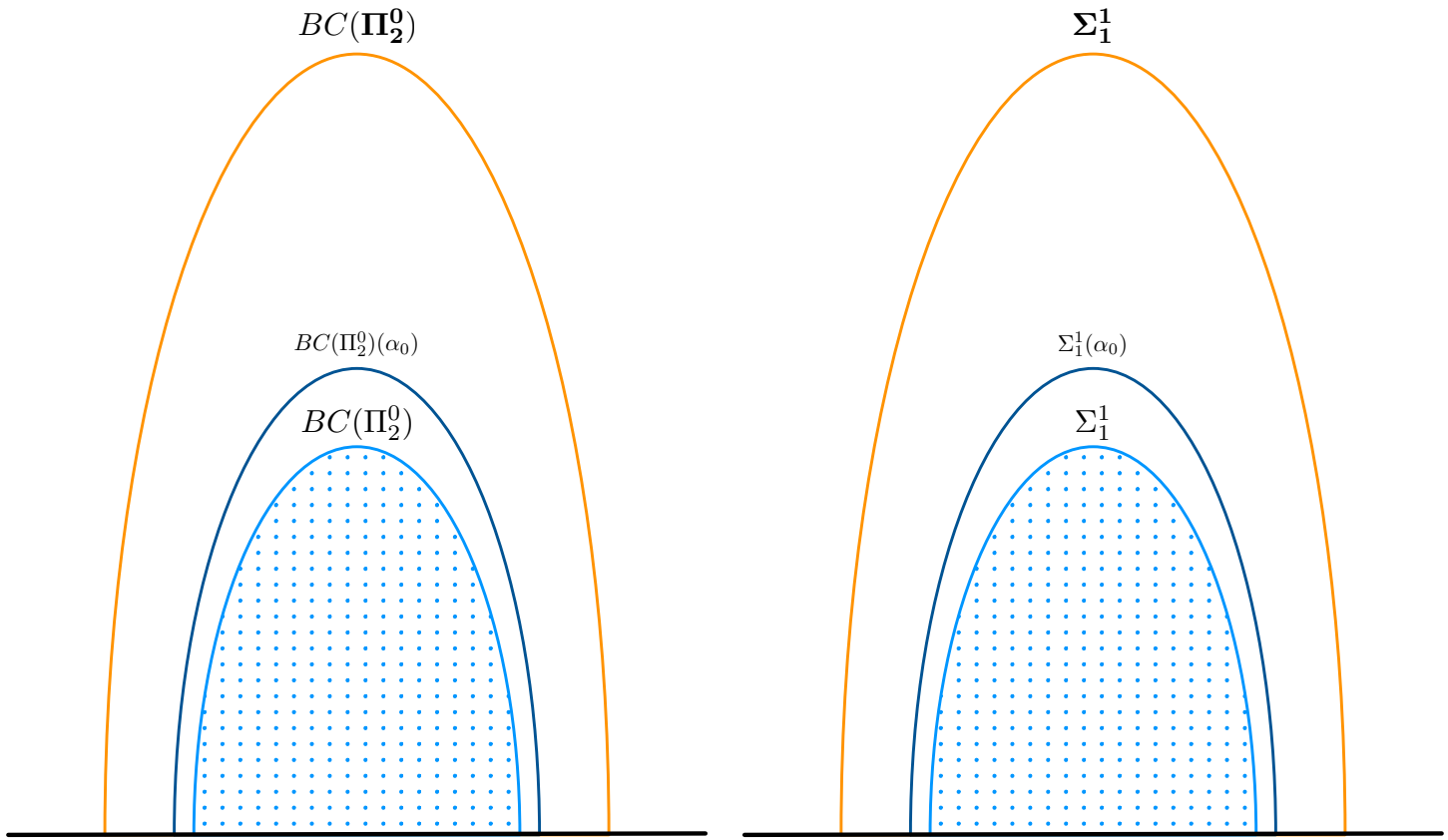
Hierarchy Theorems: stratification of the super-Turing "world"

- We can define infinitely many complexity classes between the Turing and super-Turing levels.



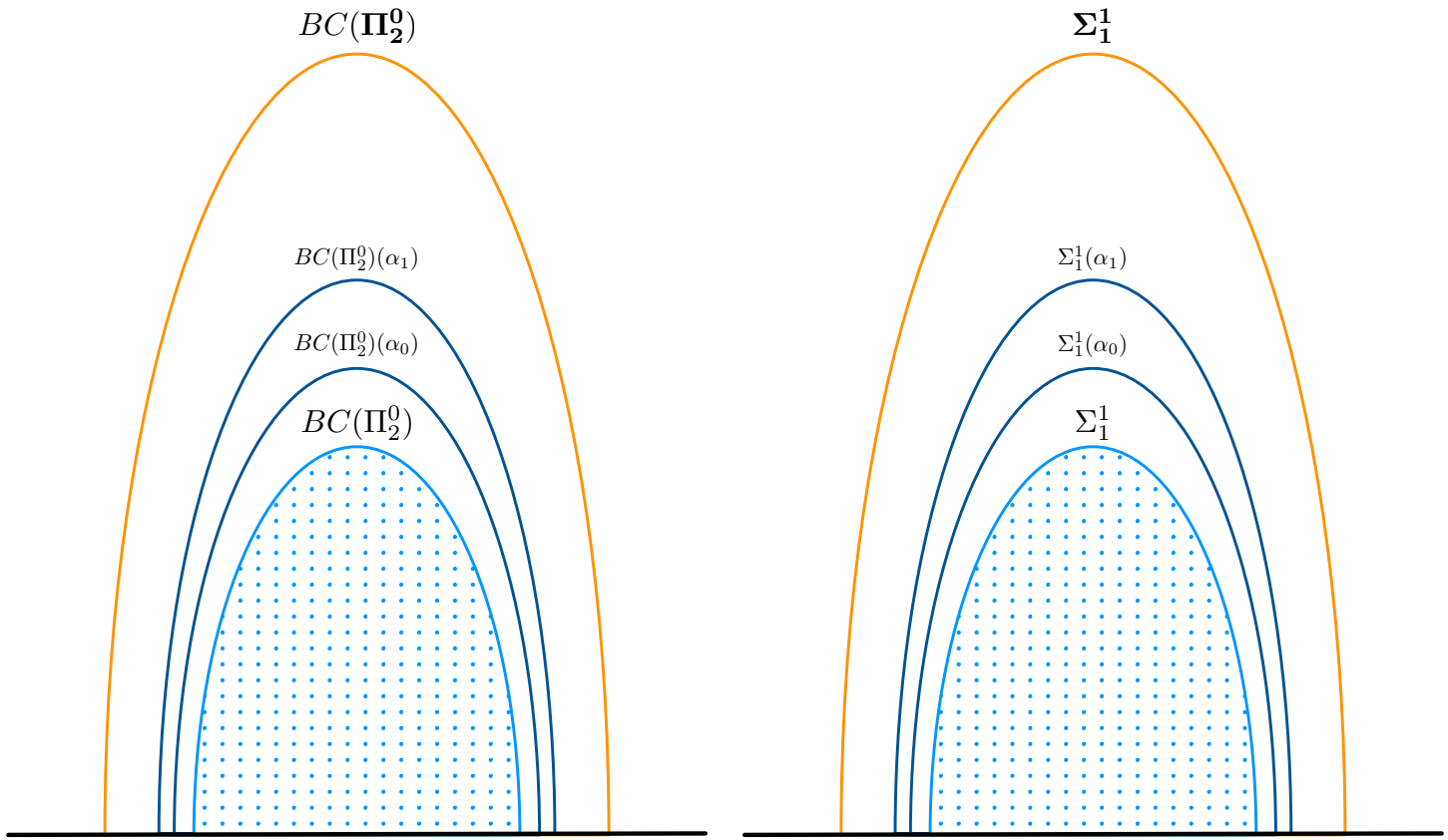
Hierarchy Theorems: stratification of the super-Turing "world"

- We can define infinitely many complexity classes between the Turing and super-Turing levels.



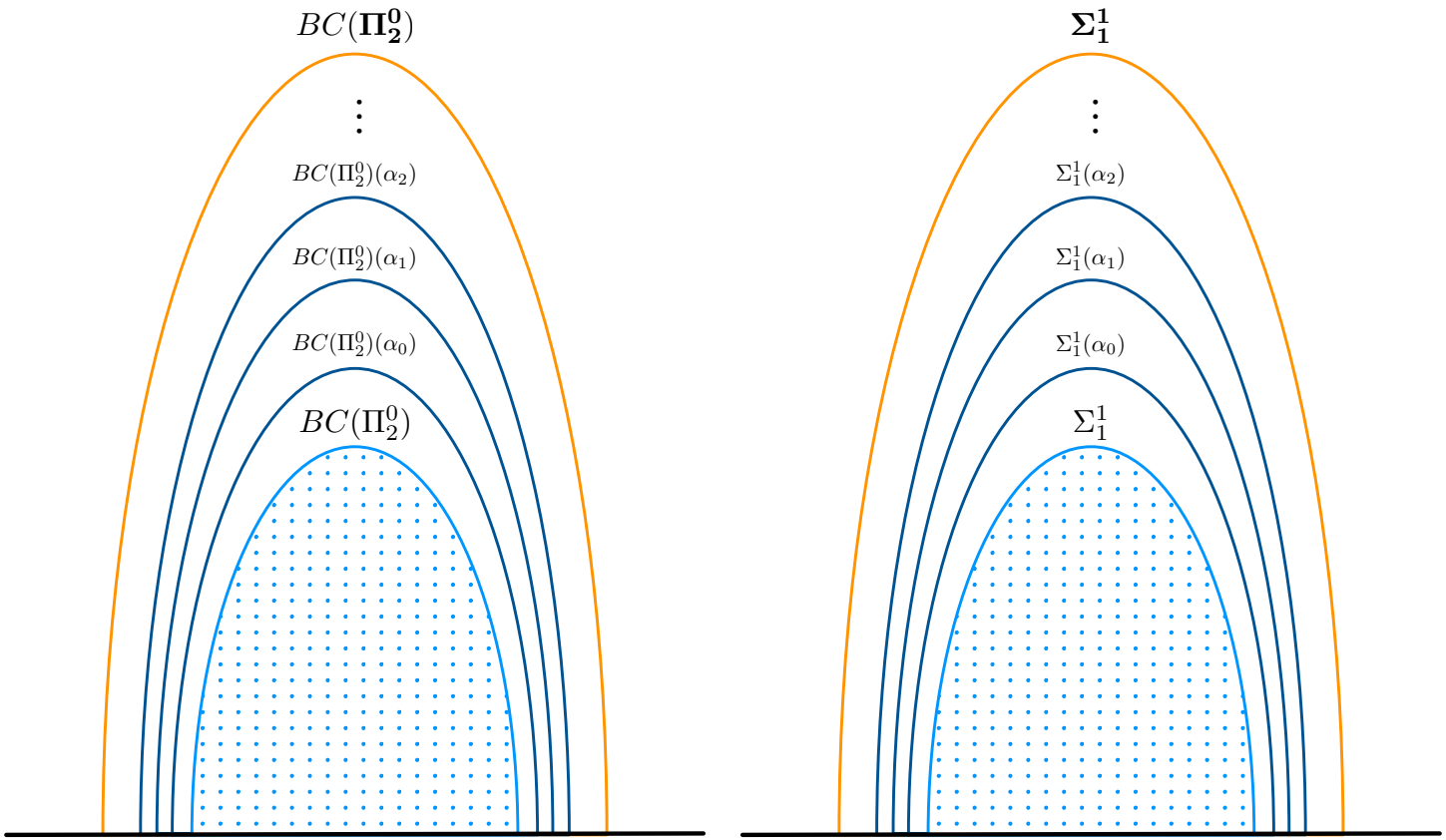
Hierarchy Theorems: stratification of the super-Turing "world"

- We can define infinitely many complexity classes between the Turing and super-Turing levels.



Hierarchy Theorems: stratification of the super-Turing "world"

- 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".
- We propose a novel paradigm for *abstract neural computation* that takes into account important biological features.

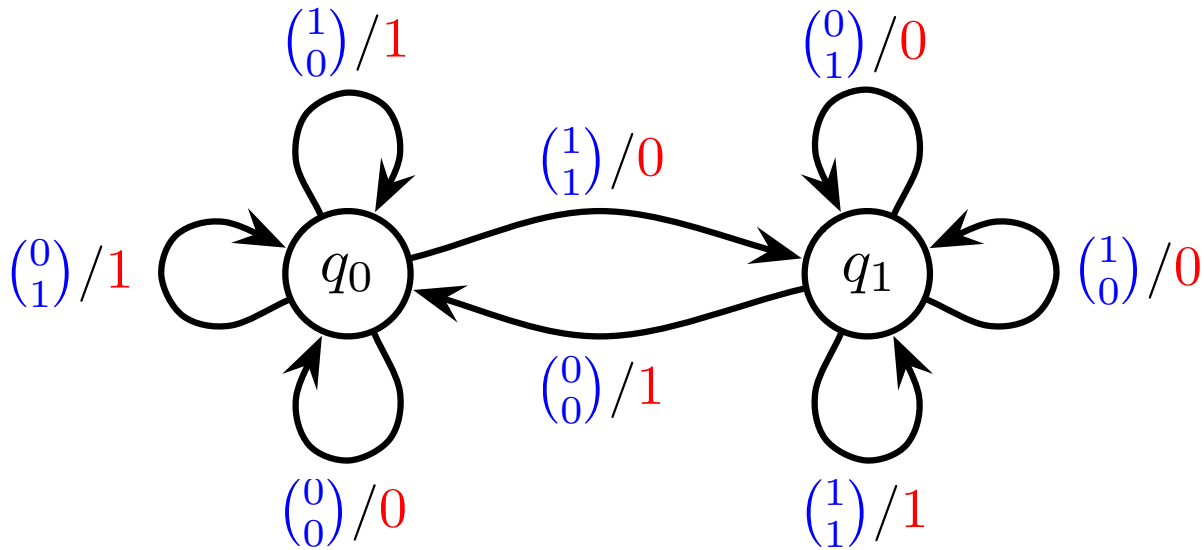
Intermediate Conclusions

- 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".
- We propose a novel paradigm for *abstract neural computation* that takes into account important biological features.

Intermediate Conclusions

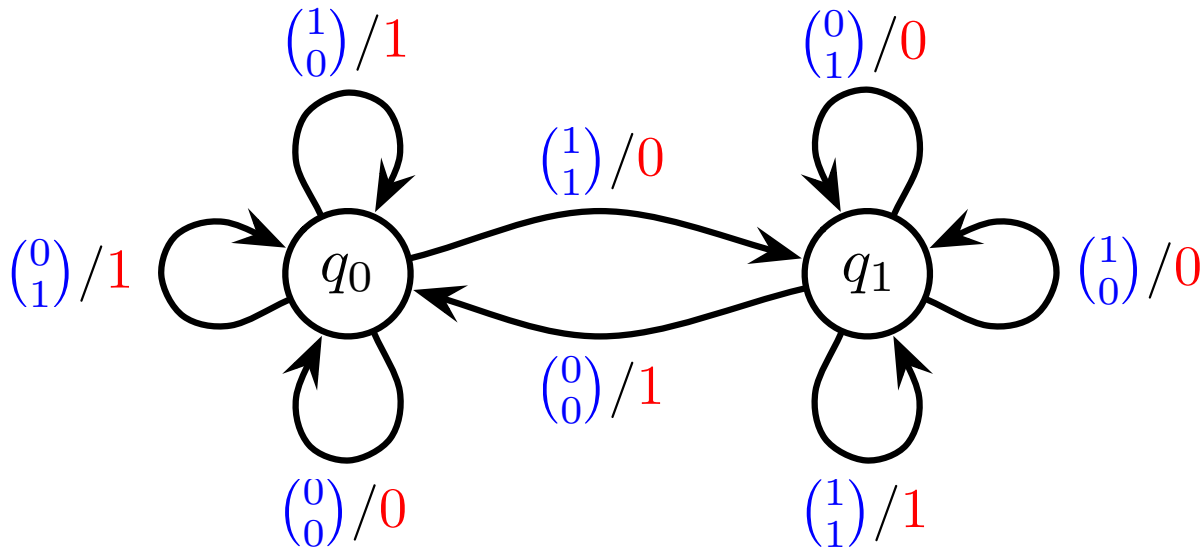
- Recurrent neural networks is a natural model for *oracle-based (super-Turing) computation*.
- In all these results, the simulation of finite state machines by recurrent neural networks is not "biologically plausible".
- We propose a novel paradigm for *abstract neural computation* that takes into account important biological features.

Binary Adder Automaton



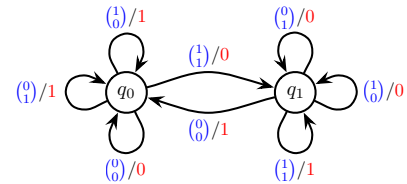
$$\begin{array}{cccccccc} & & 1 & 1^{1} & 0^{1} & 1 & 1 & 0^{1} & 1 \\ + & & & 1 & 1 & 1 & 0 & 0 & 1 \\ \hline & & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}$$

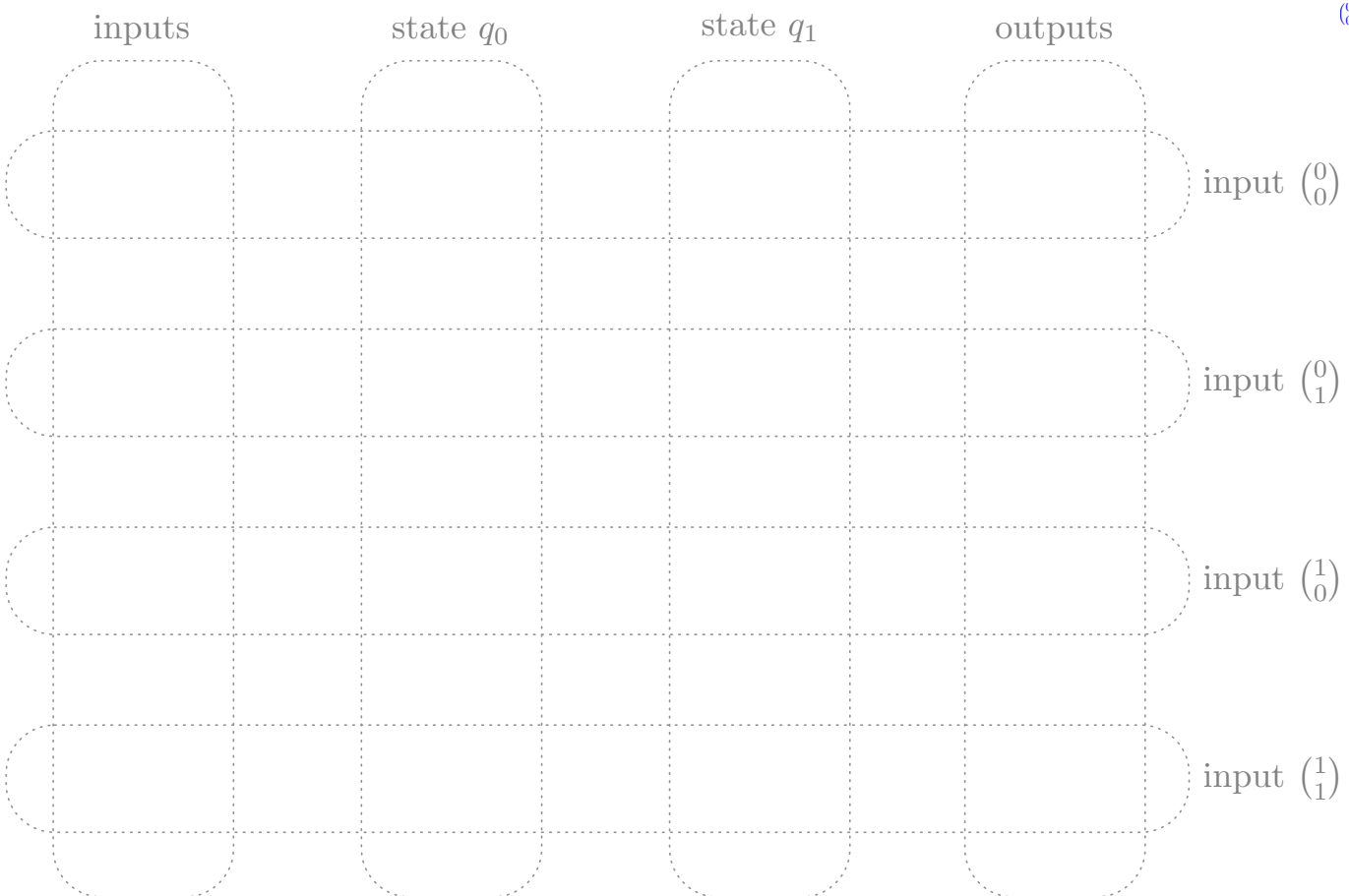
Binary Adder Automaton



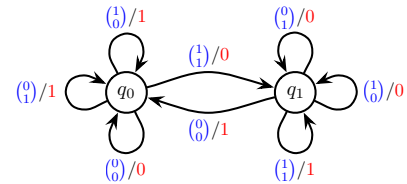
$$
\begin{array}{cccccccc}
 & {\scriptstyle 1} & 1^{\scriptstyle 1} & 0^{\scriptstyle 1} & 1 & 1 & 0^{\scriptstyle 1} & 1 \\
+ & & 1 & 1 & 1 & 0 & 0 & 1 \\
\hline
 & 1 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}
$$

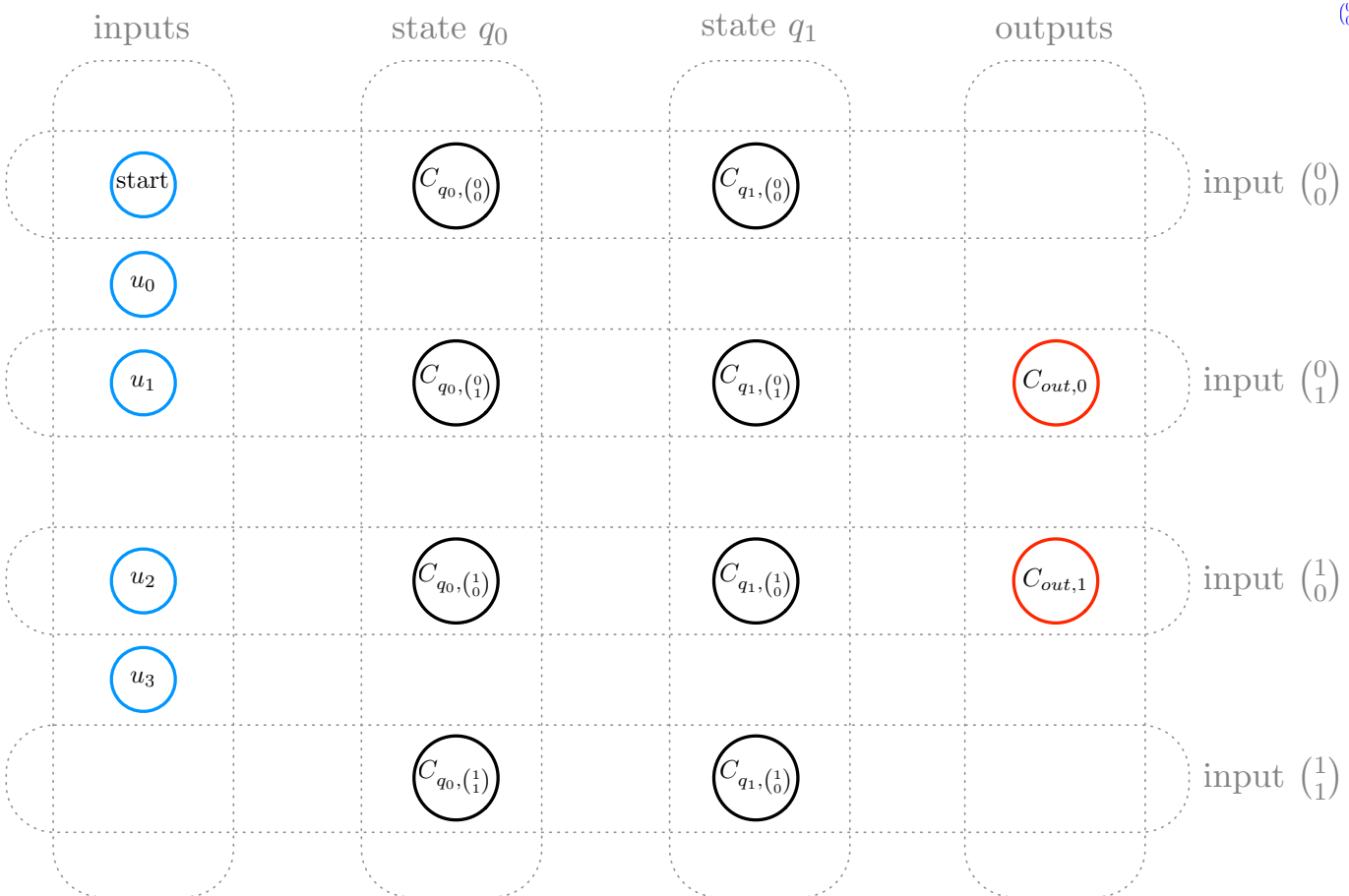
Binary Adder Boolean Neural Network



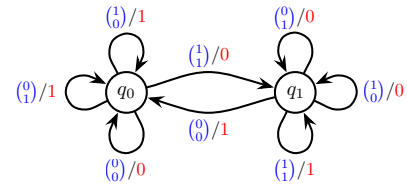


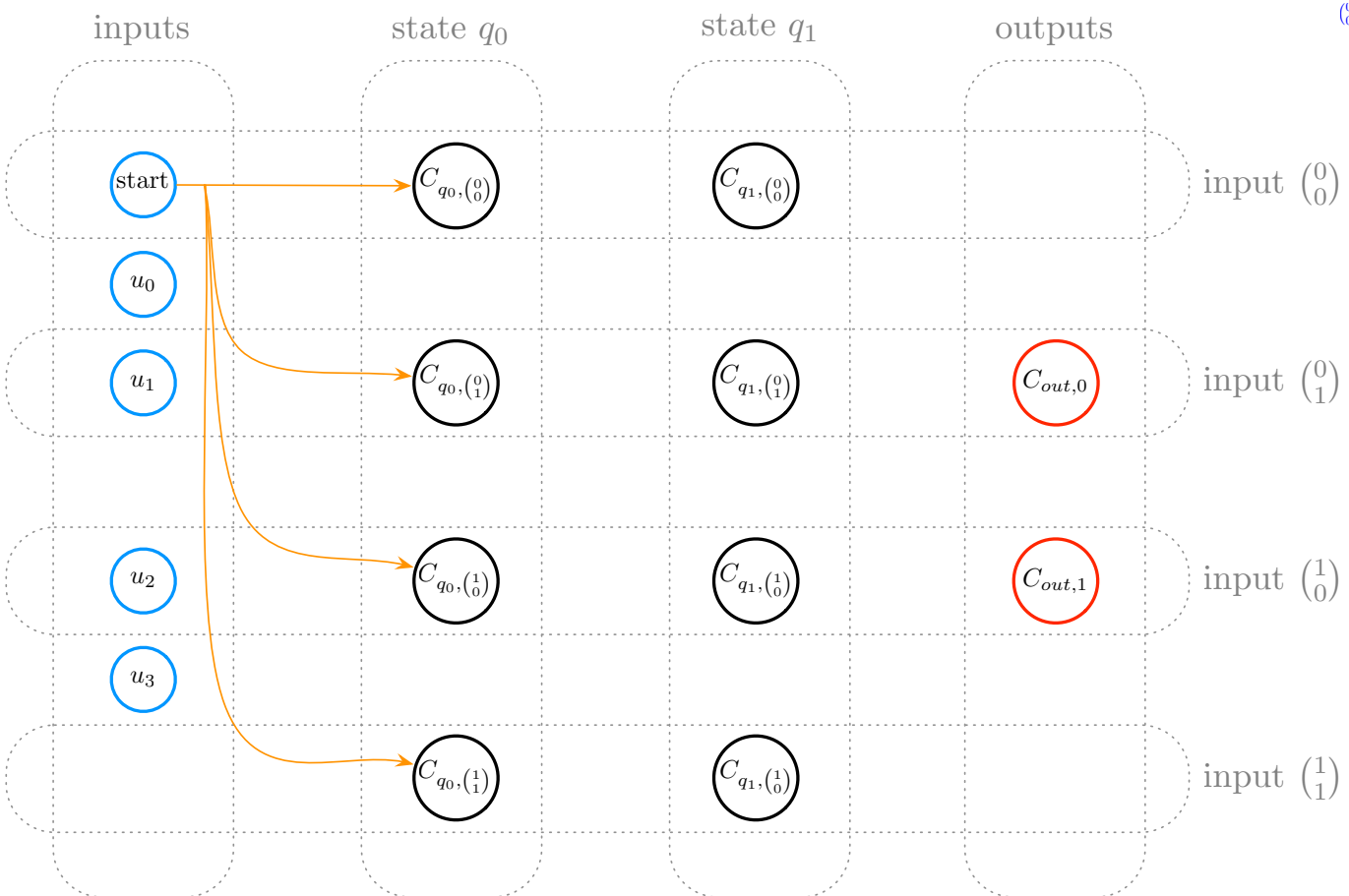
Binary Adder Boolean Neural Network



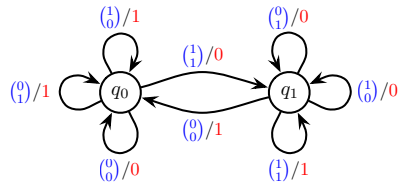


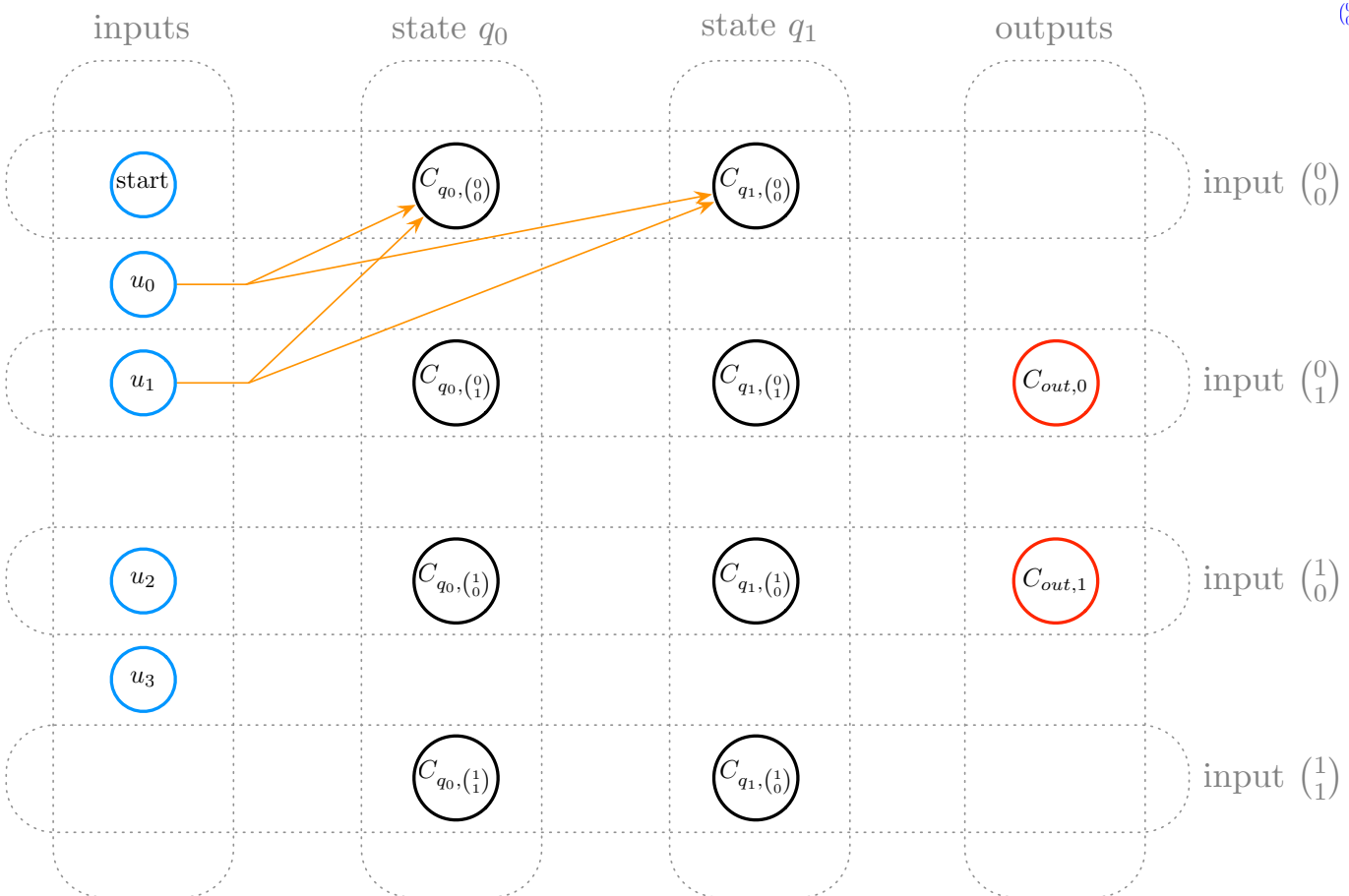
Binary Adder Boolean Neural Network



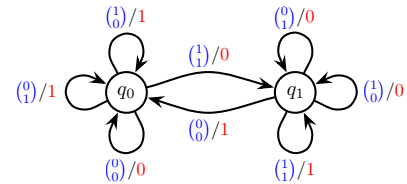


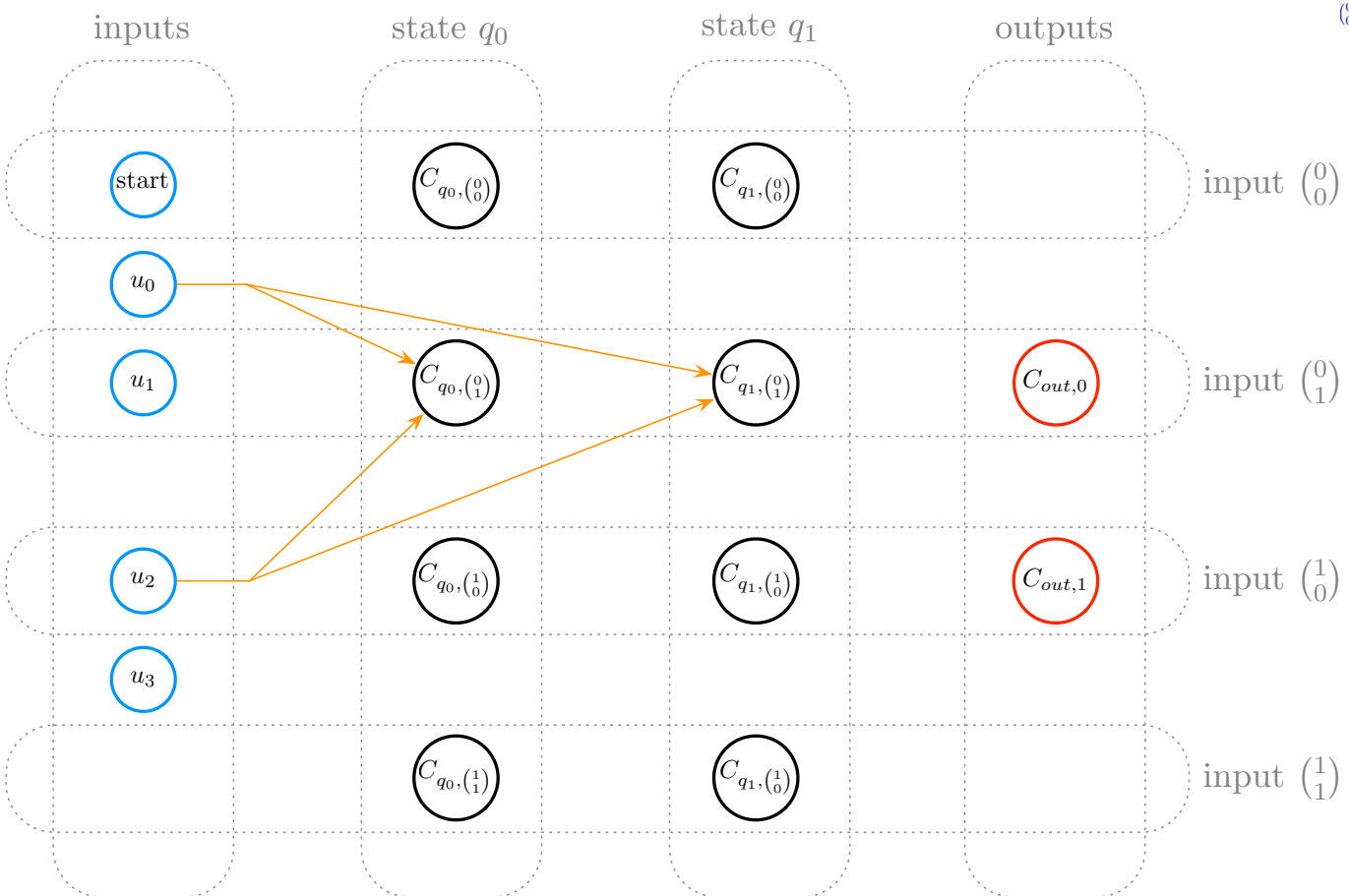
Binary Adder Boolean Neural Network



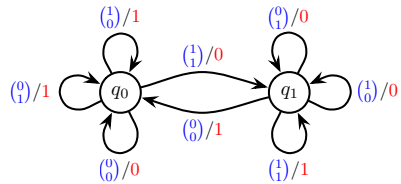


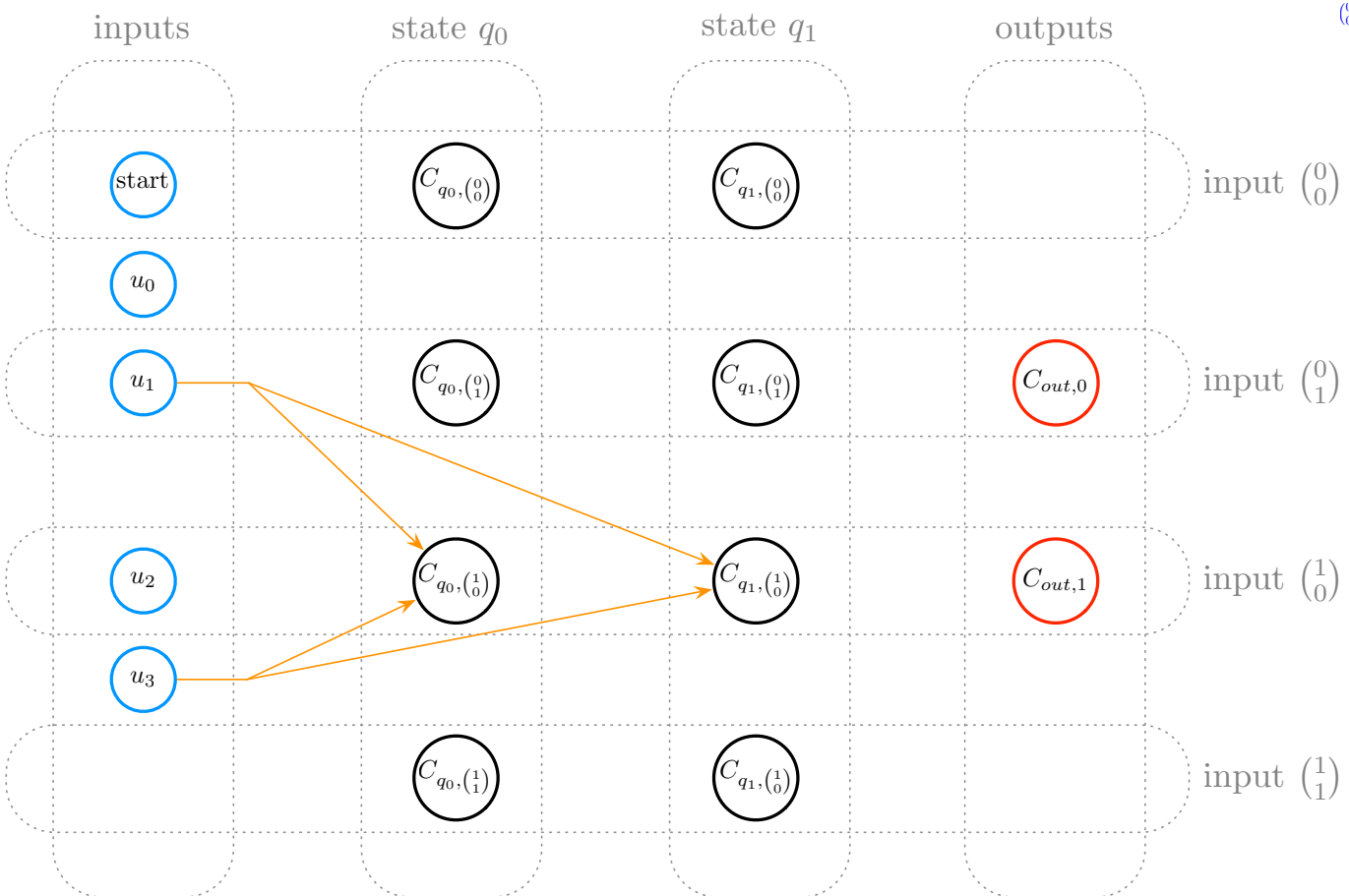
Binary Adder Boolean Neural Network



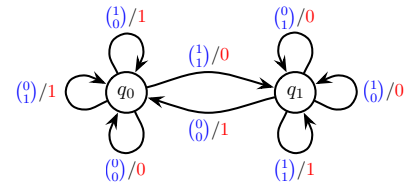


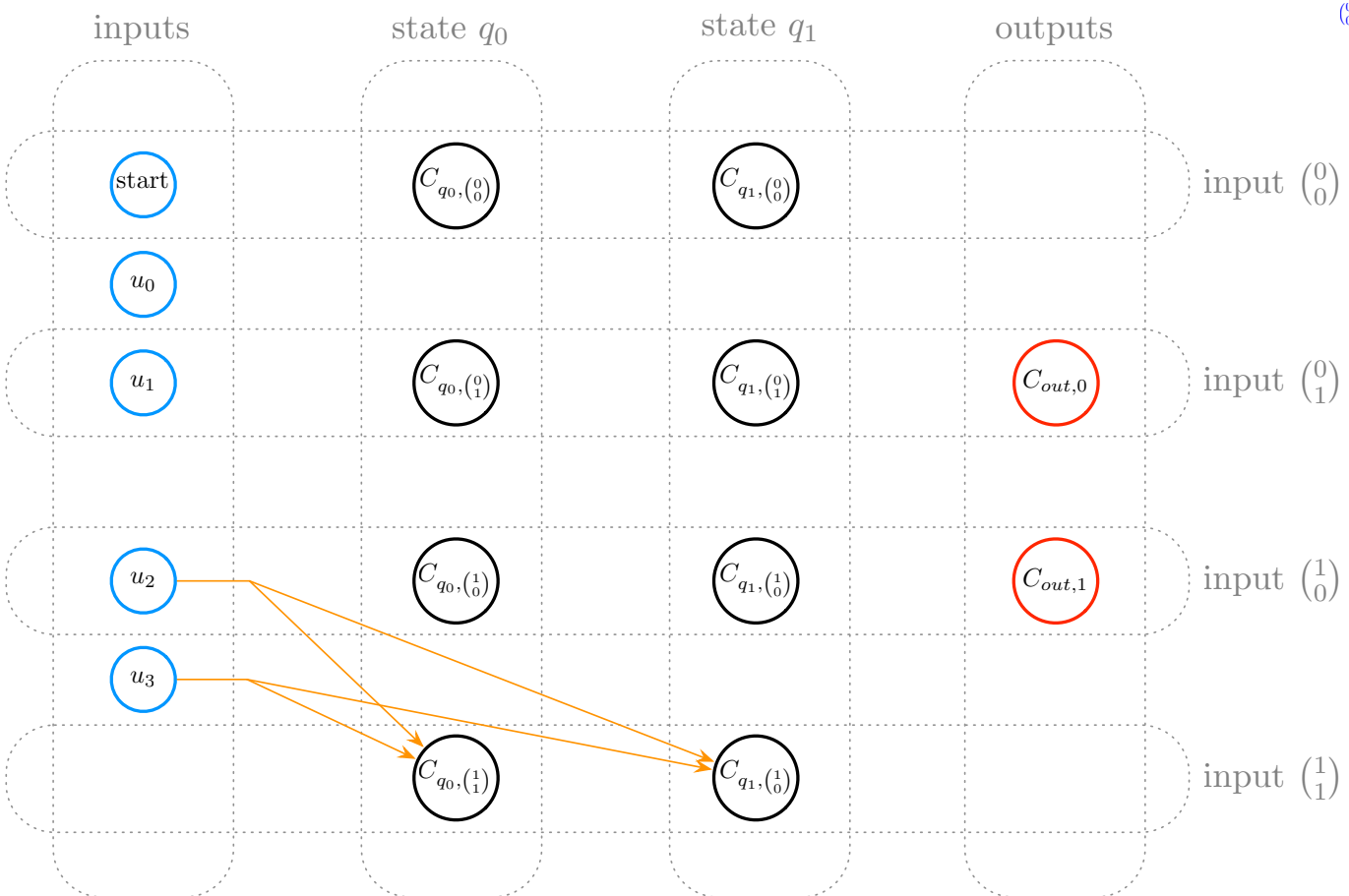
Binary Adder Boolean Neural Network



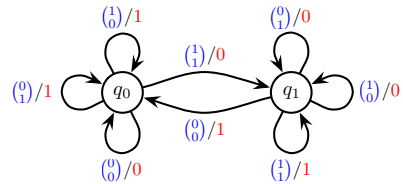


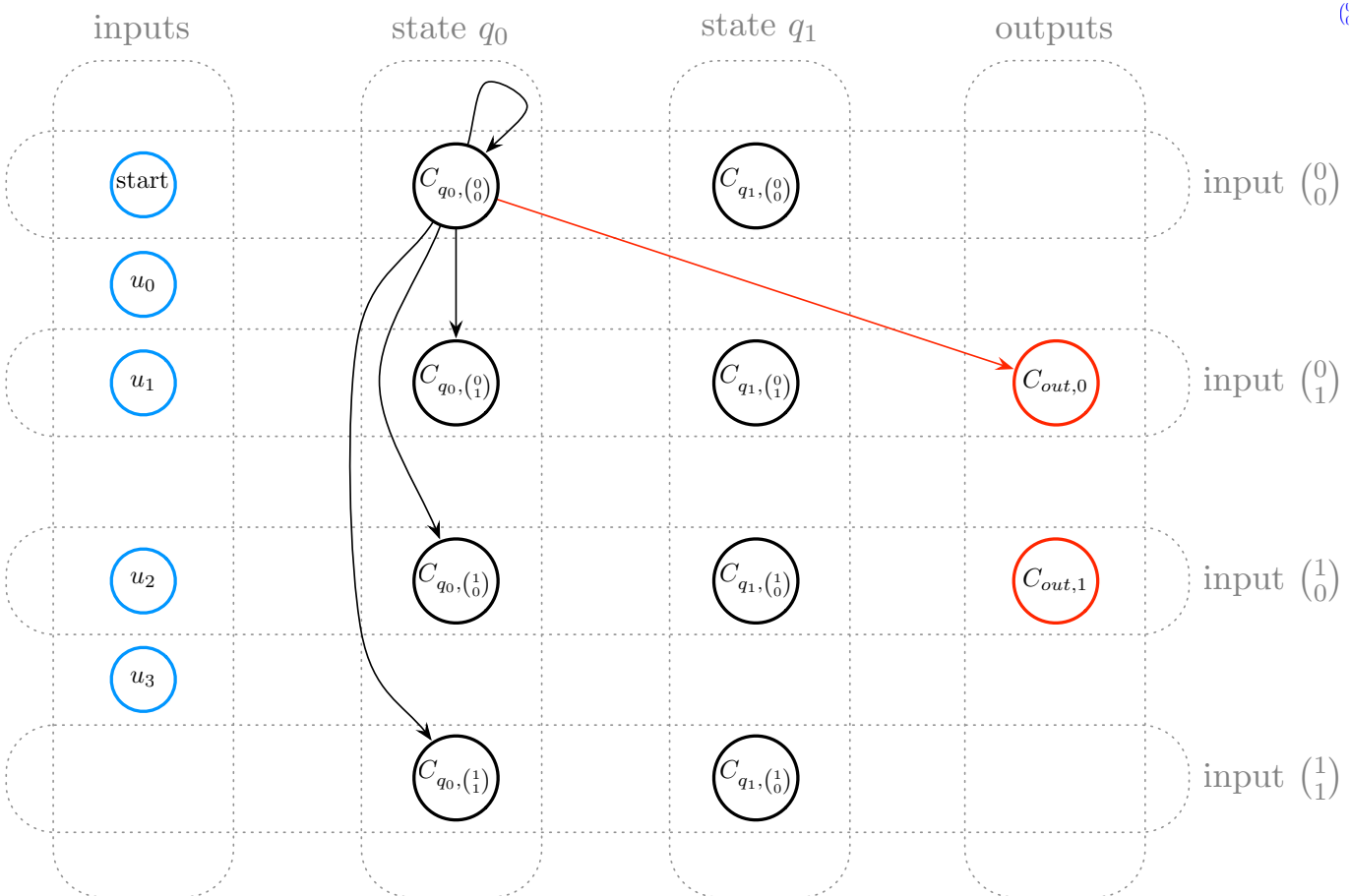
Binary Adder Boolean Neural Network



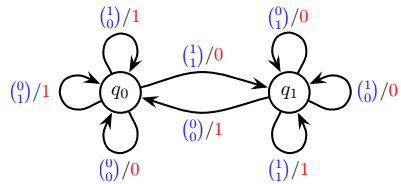


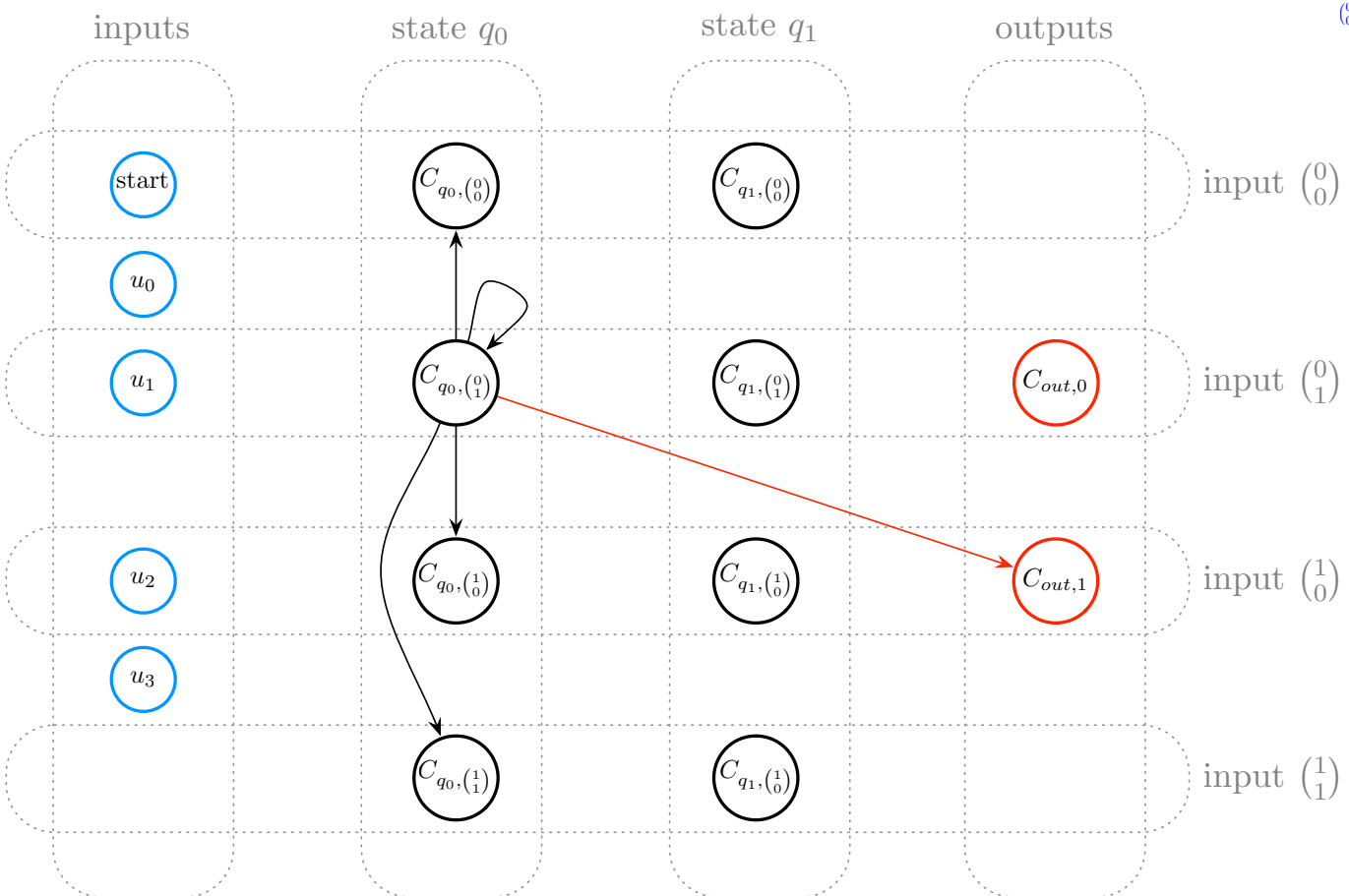
Binary Adder Boolean Neural Network



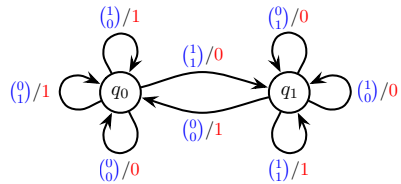


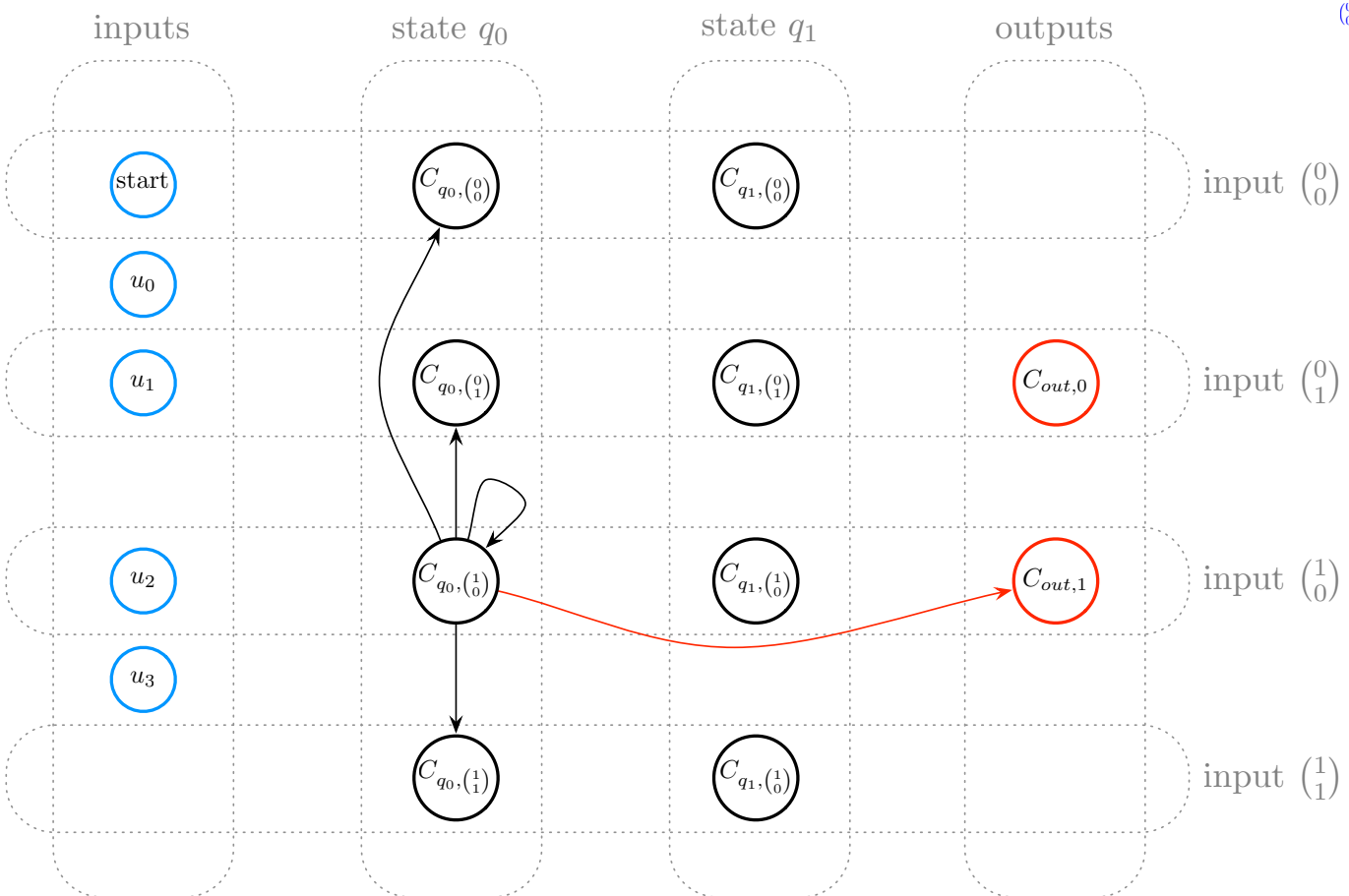
Binary Adder Boolean Neural Network



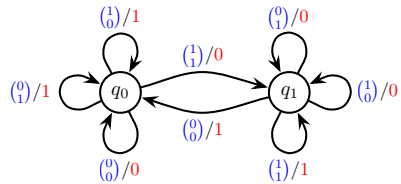


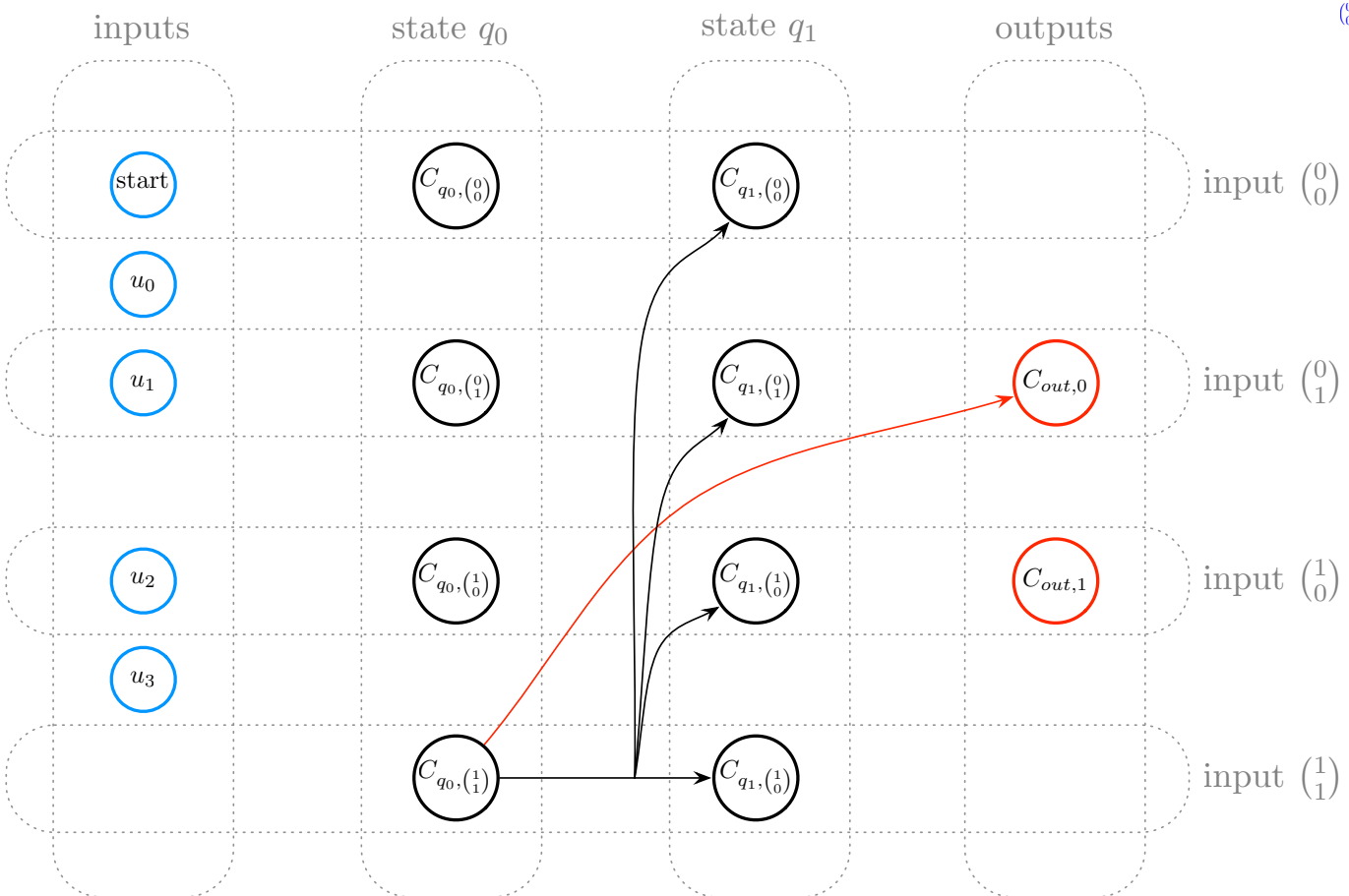
Binary Adder Boolean Neural Network



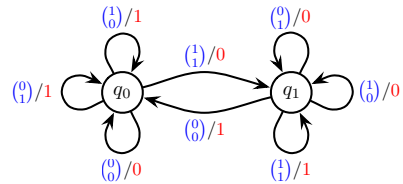


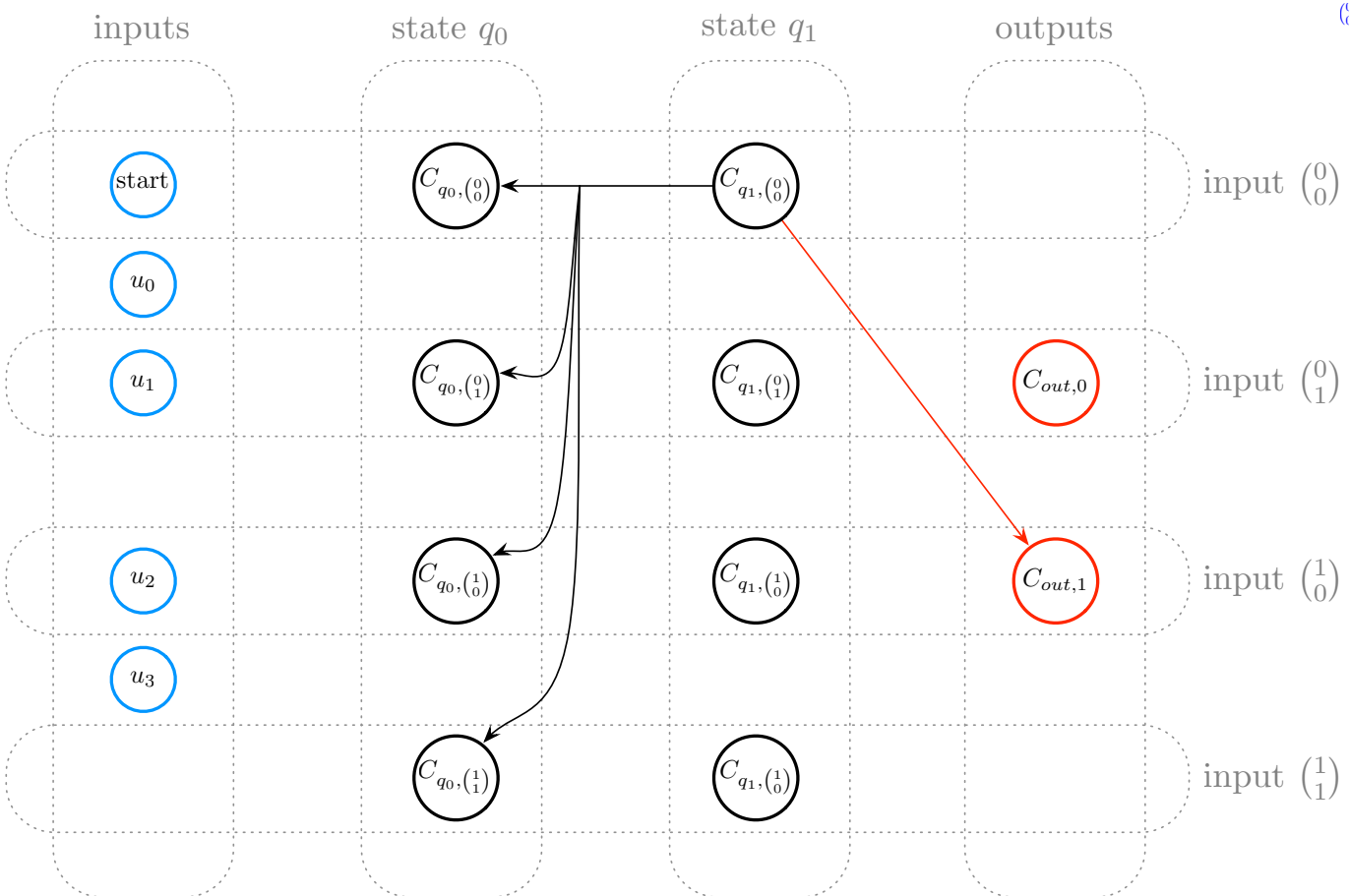
Binary Adder Boolean Neural Network



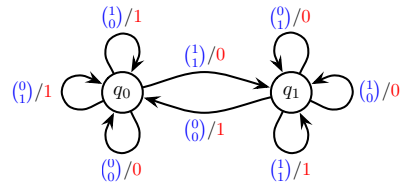


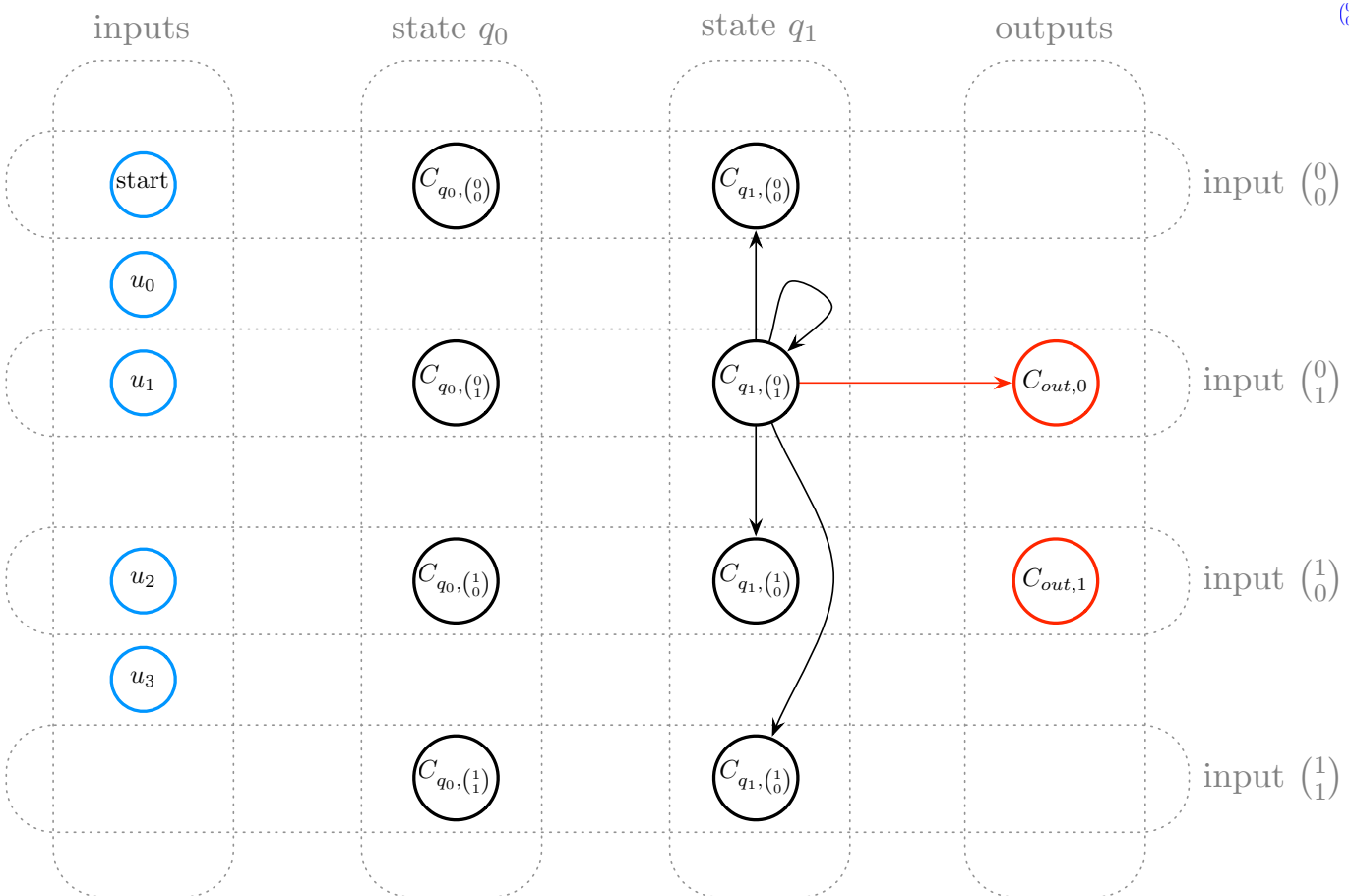
Binary Adder Boolean Neural Network



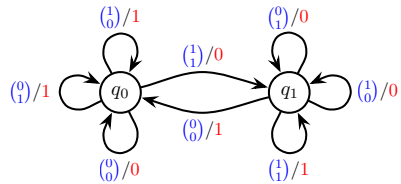


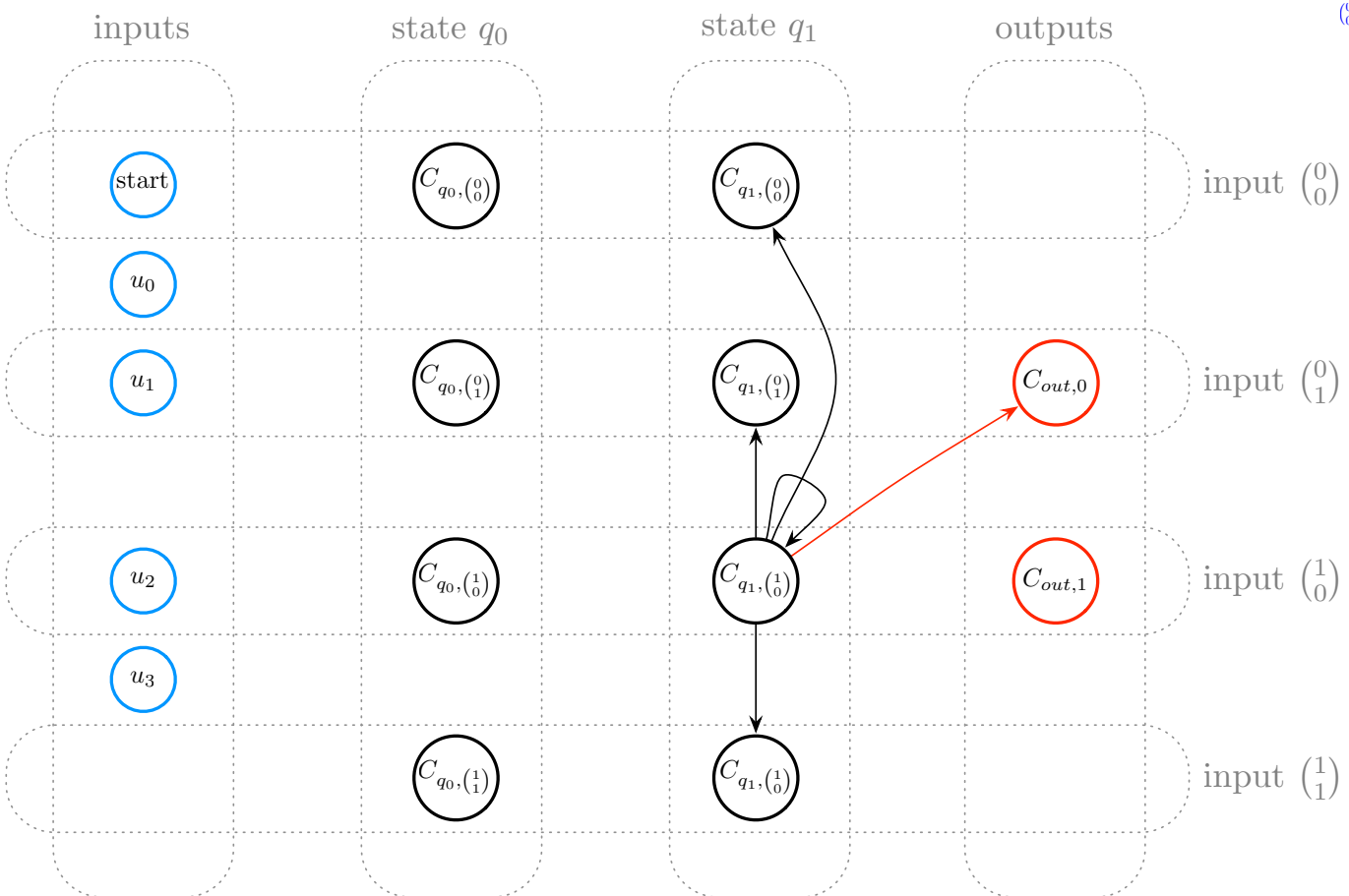
Binary Adder Boolean Neural Network



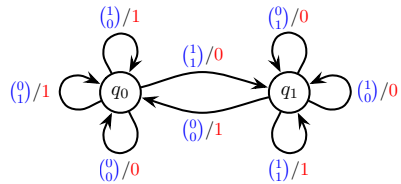


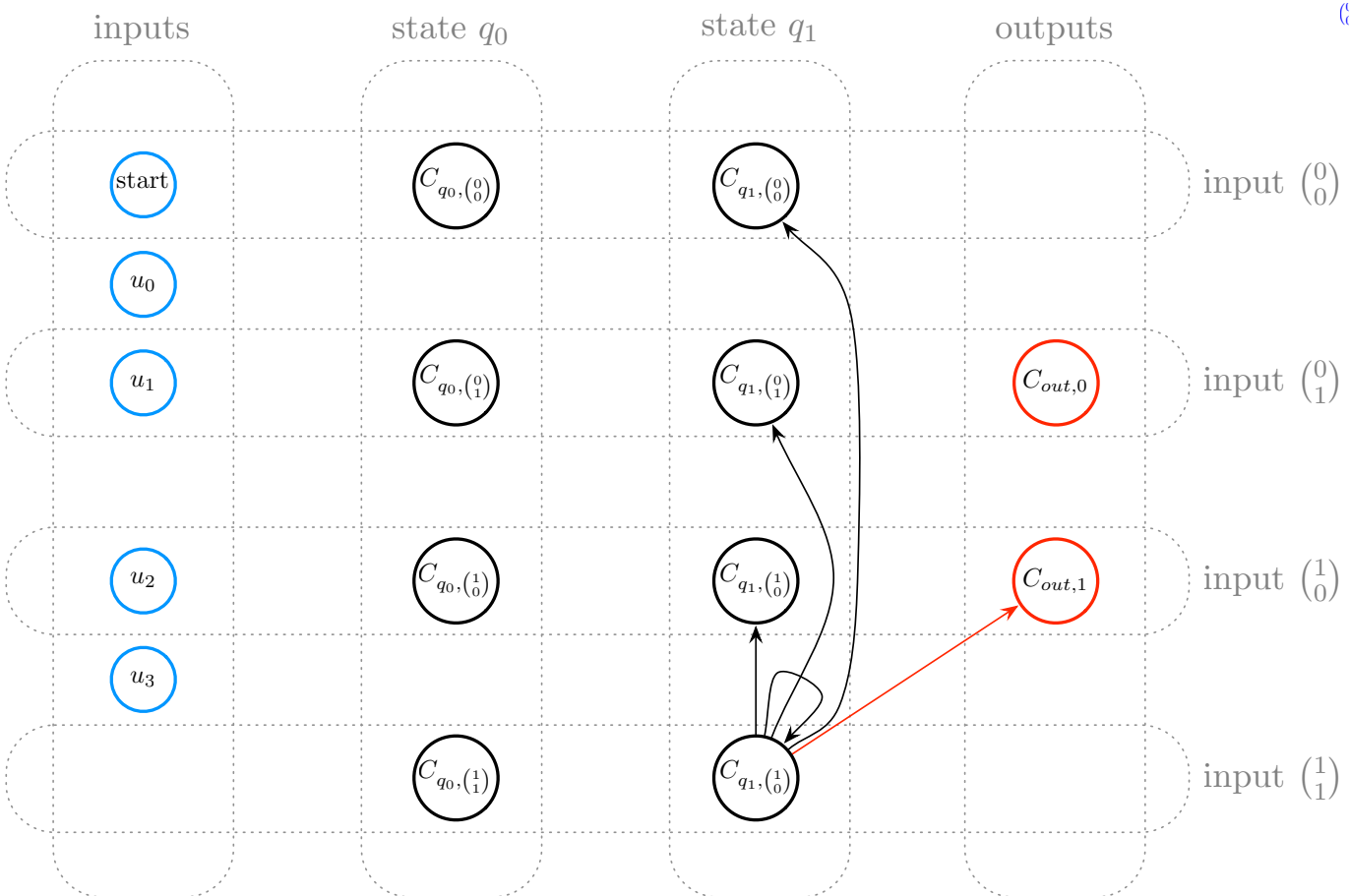
Binary Adder Boolean Neural Network



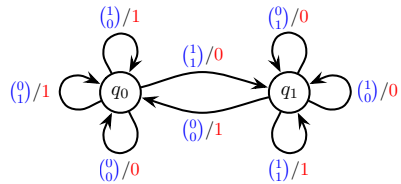


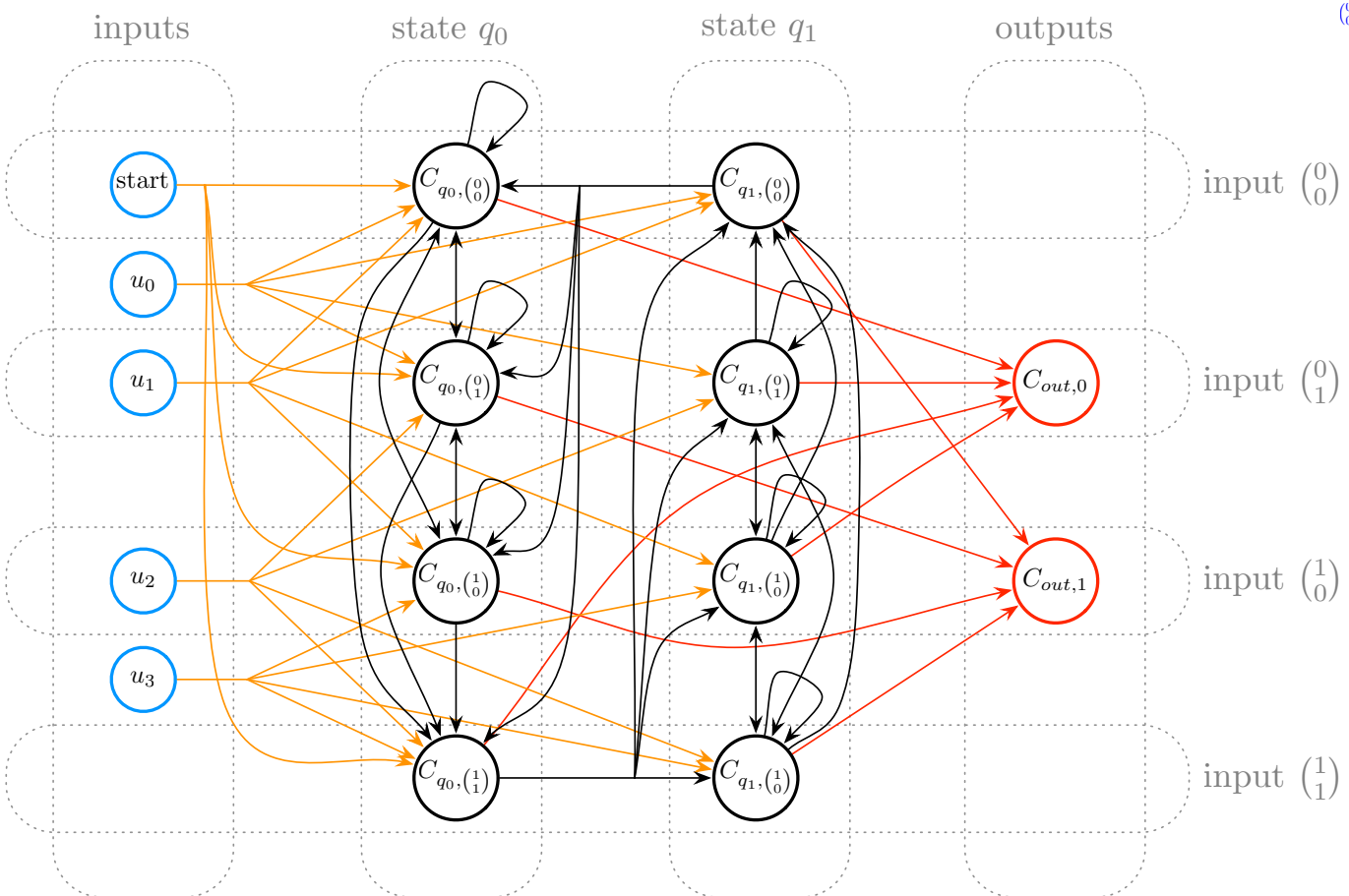
Binary Adder Boolean Neural Network



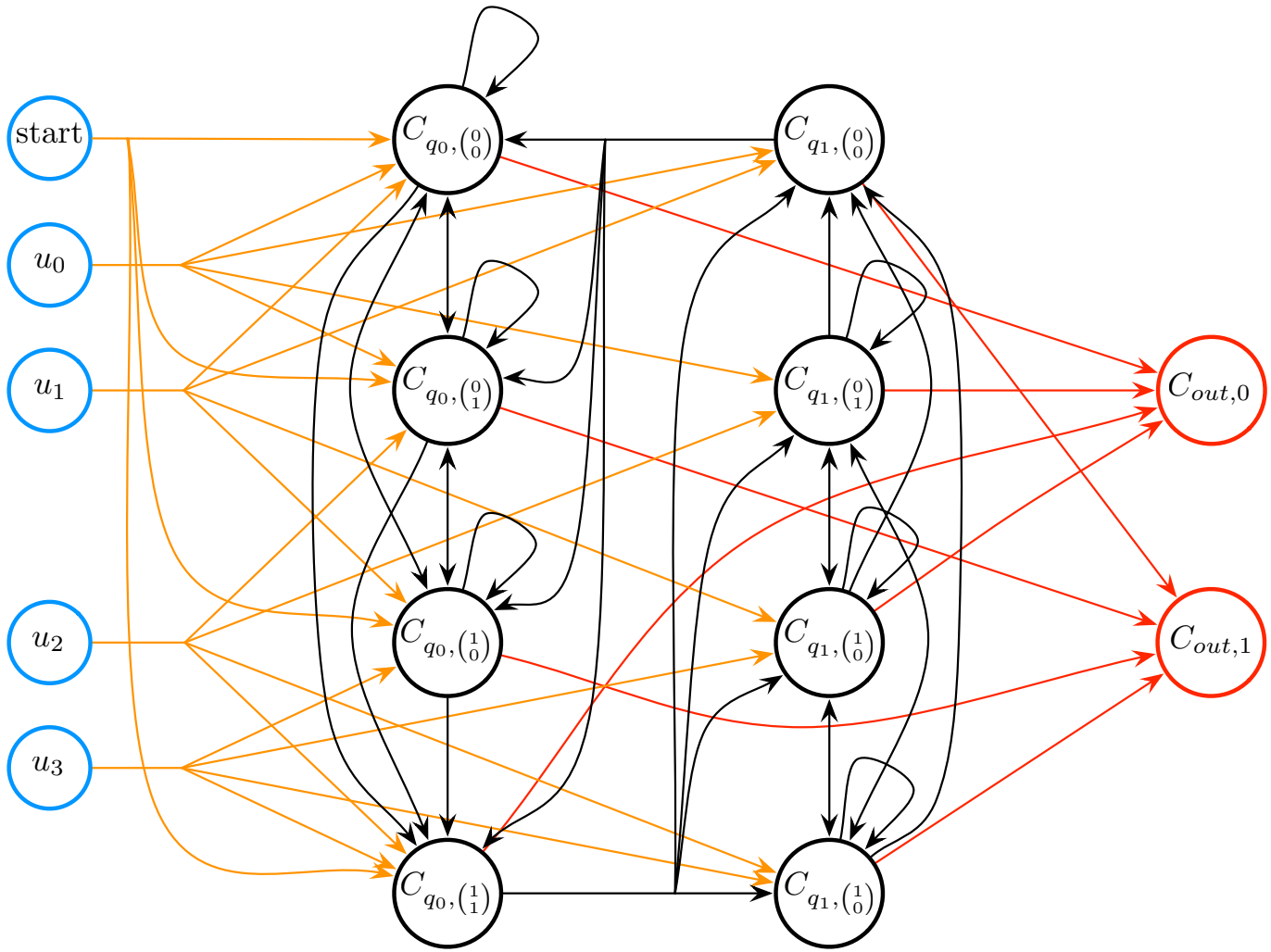


Binary Adder Boolean Neural Network

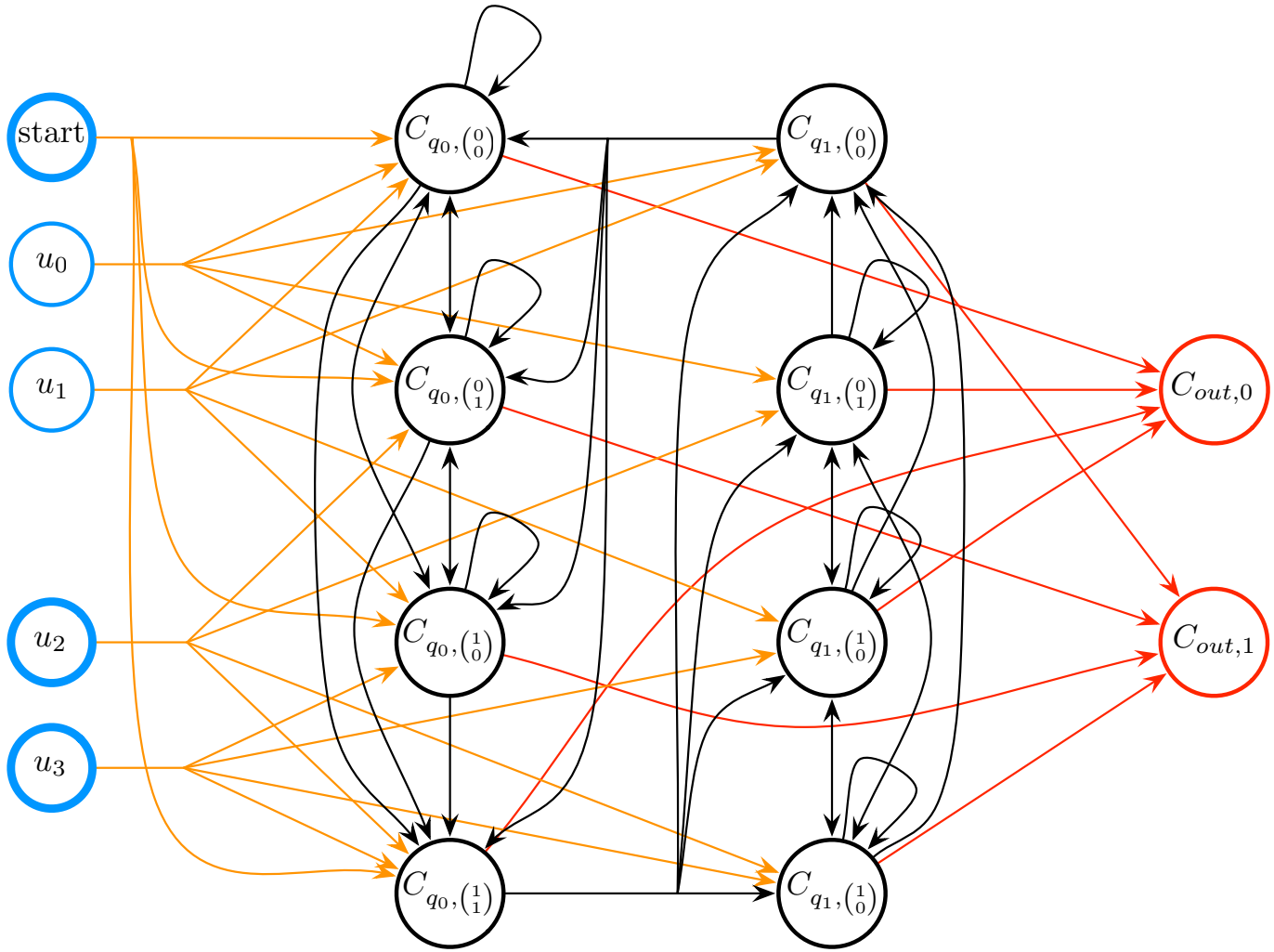




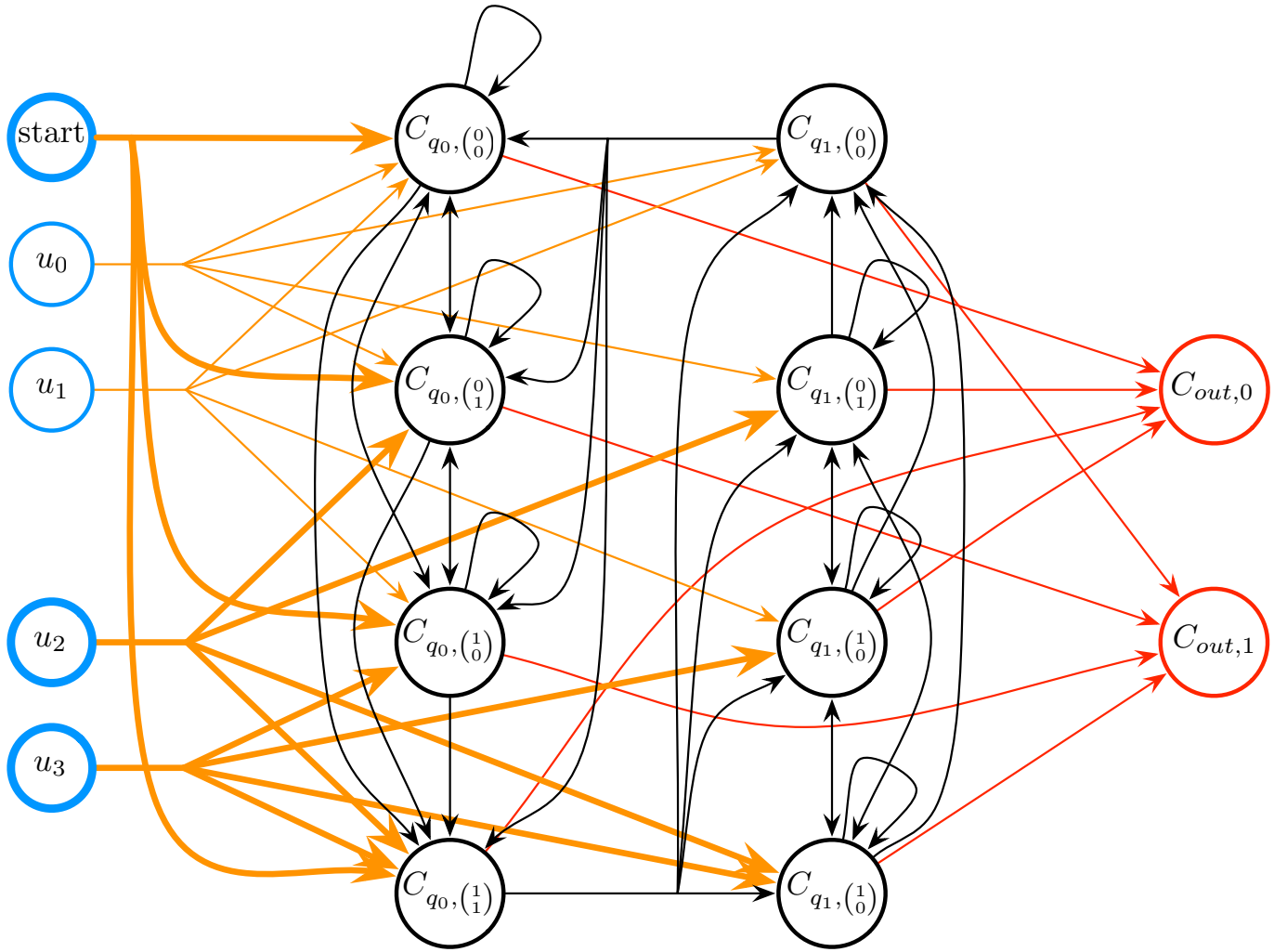
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

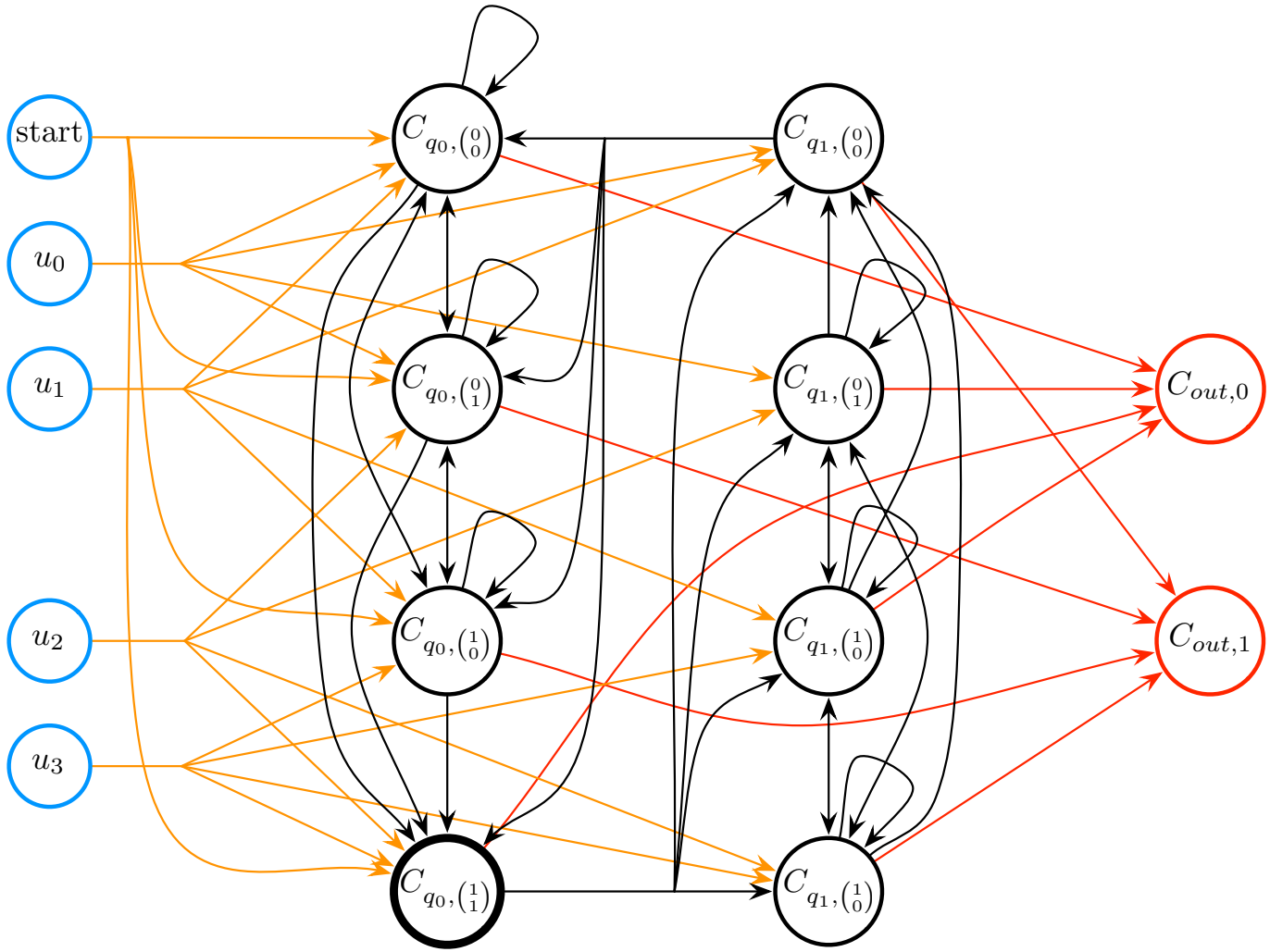
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

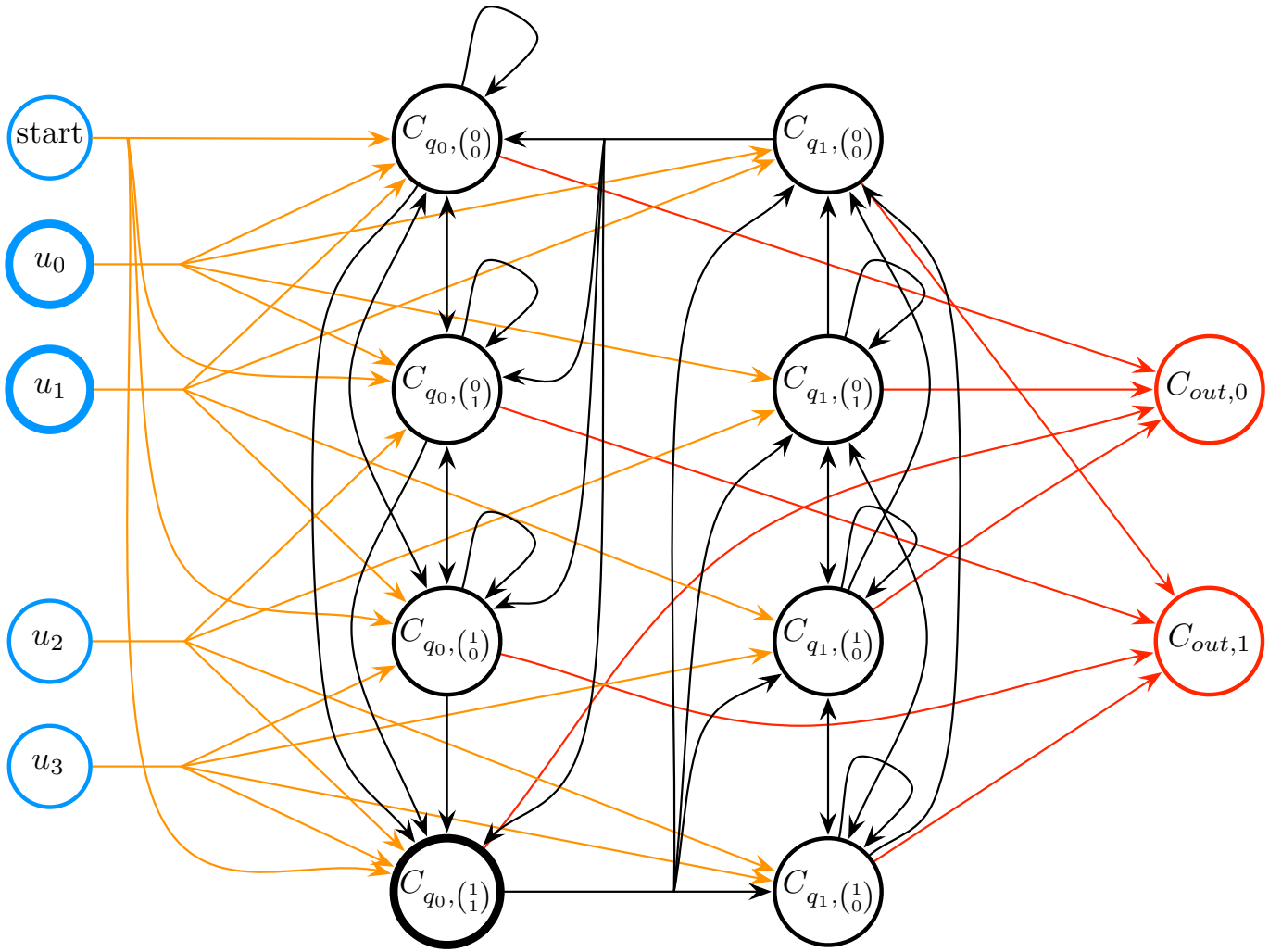
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

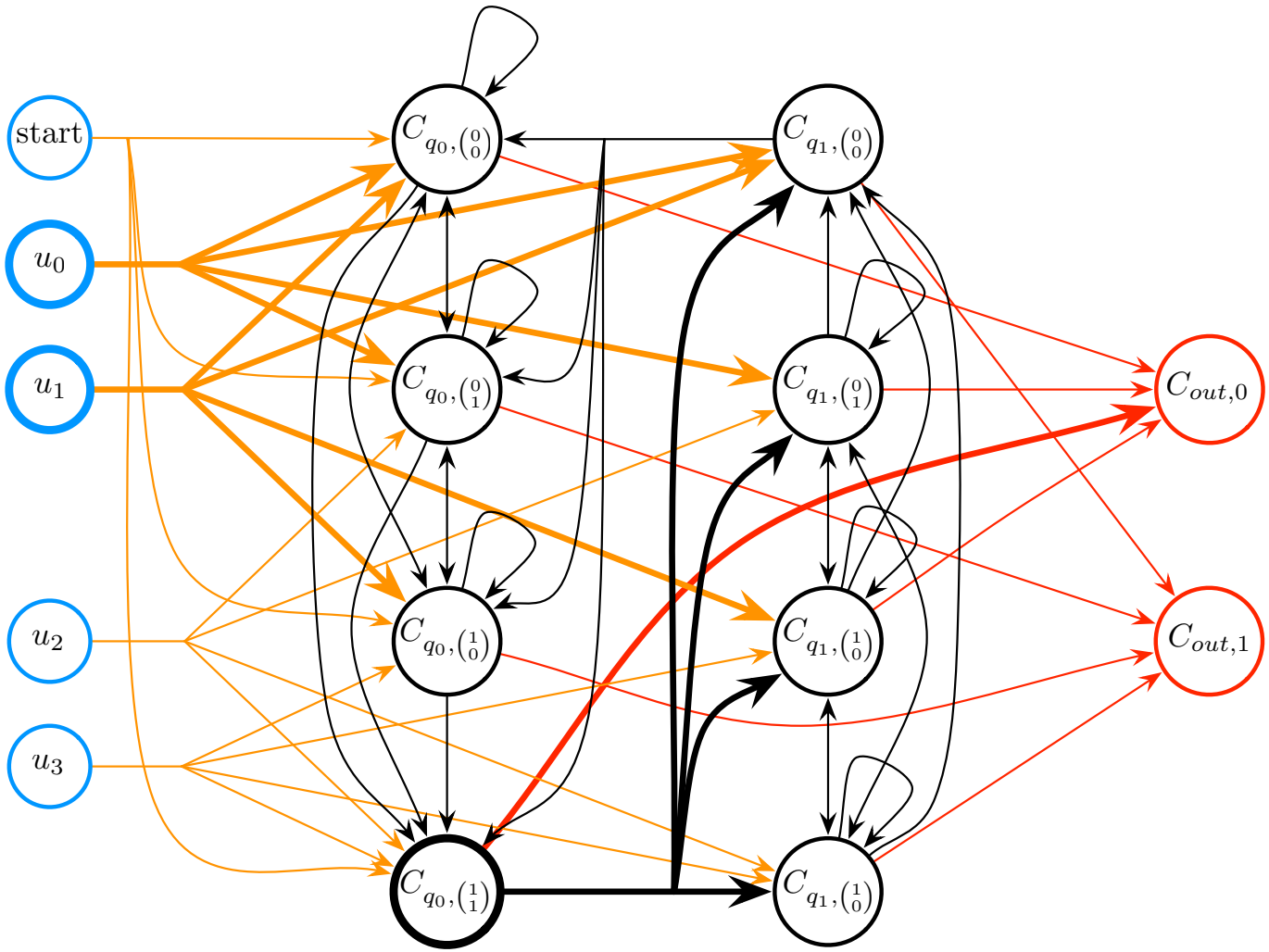
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

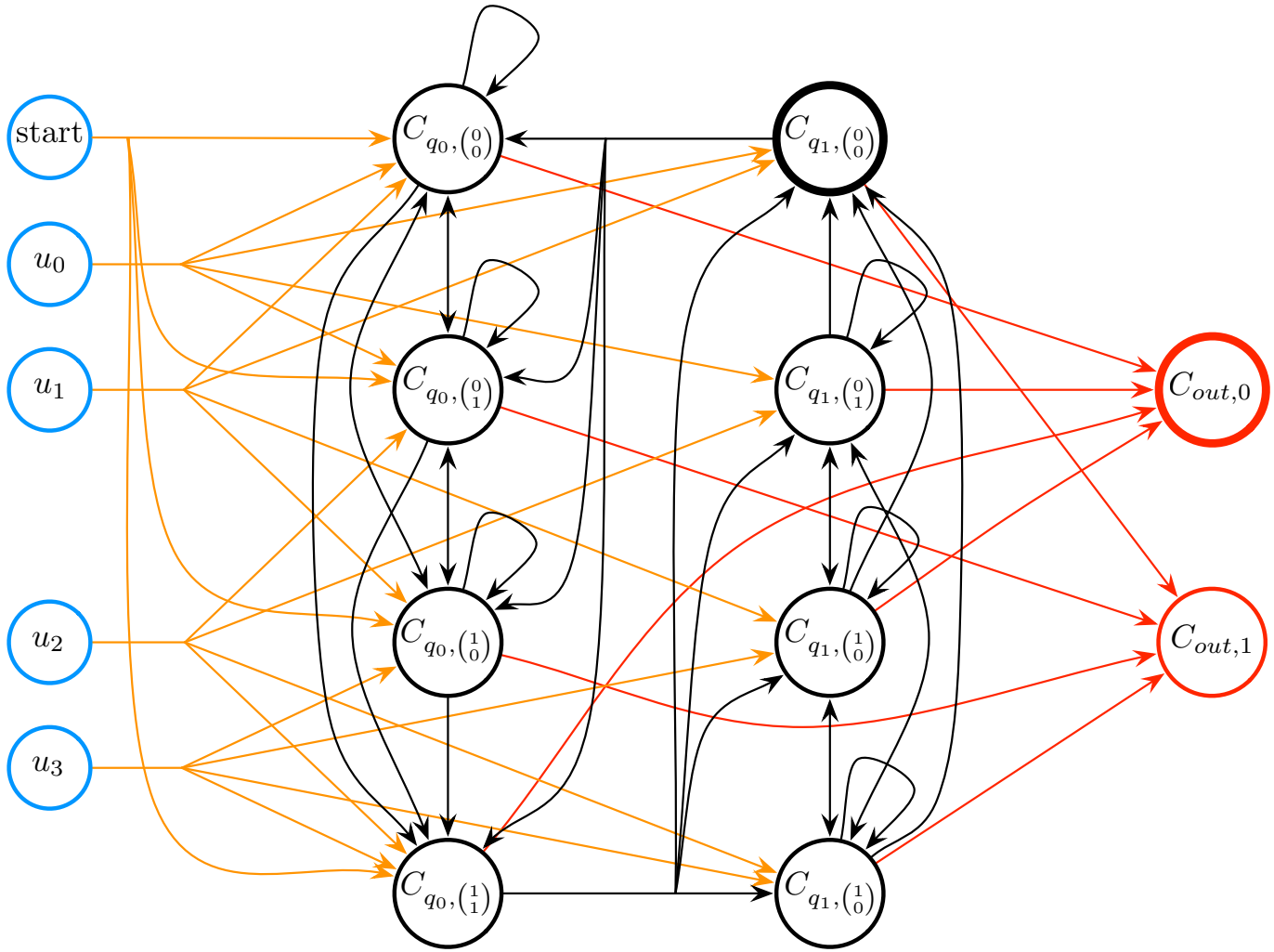
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

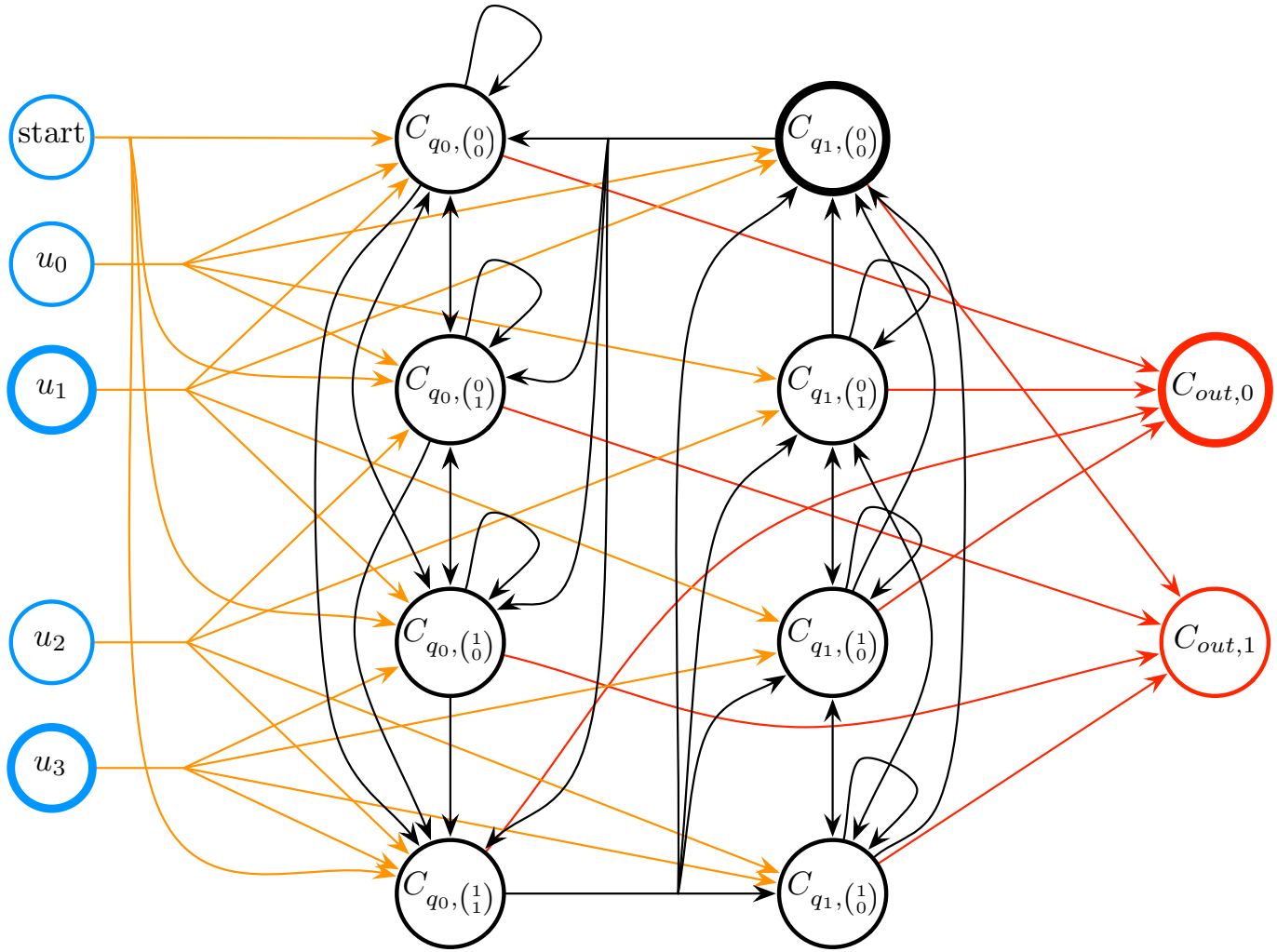
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

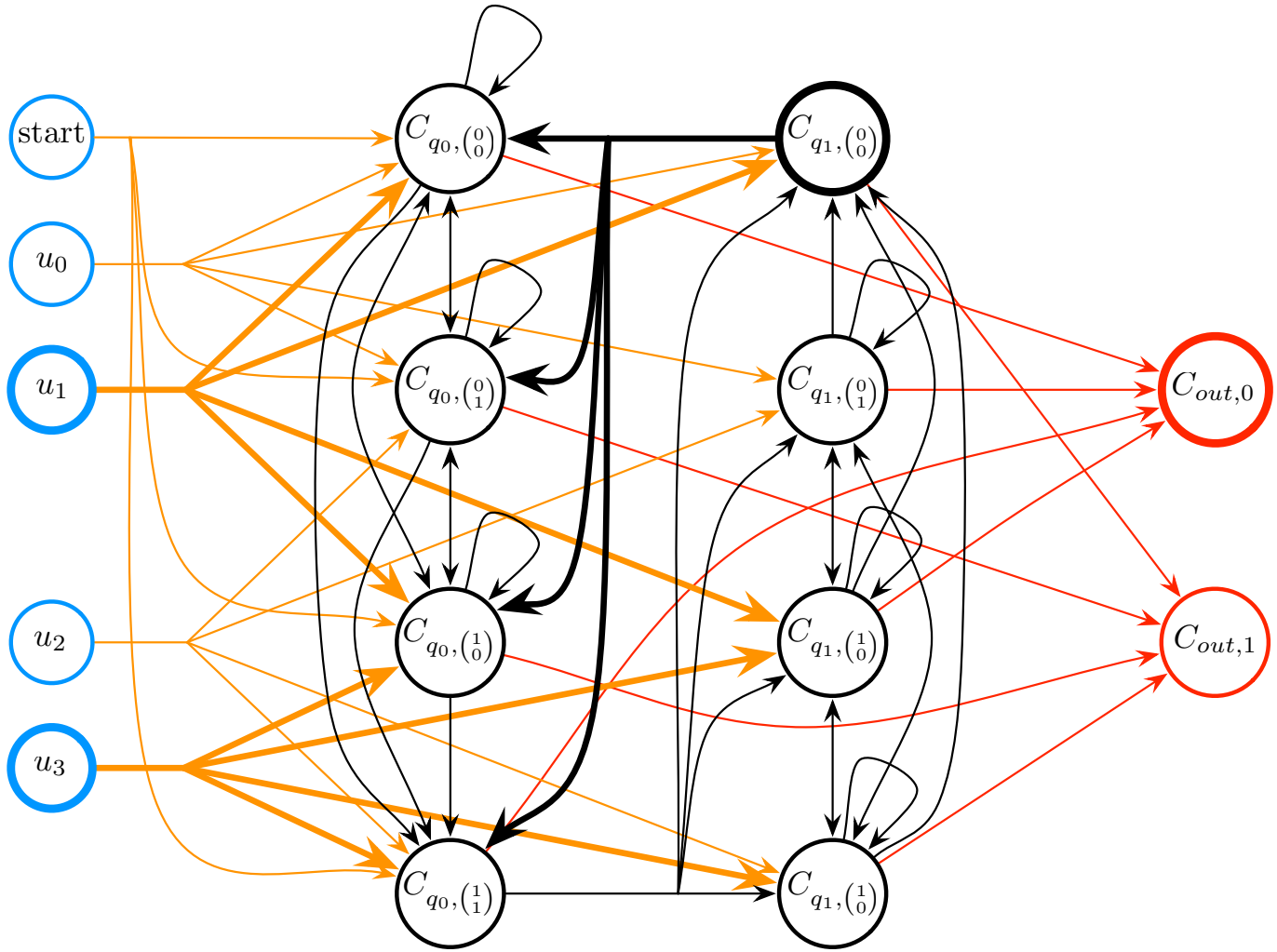
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

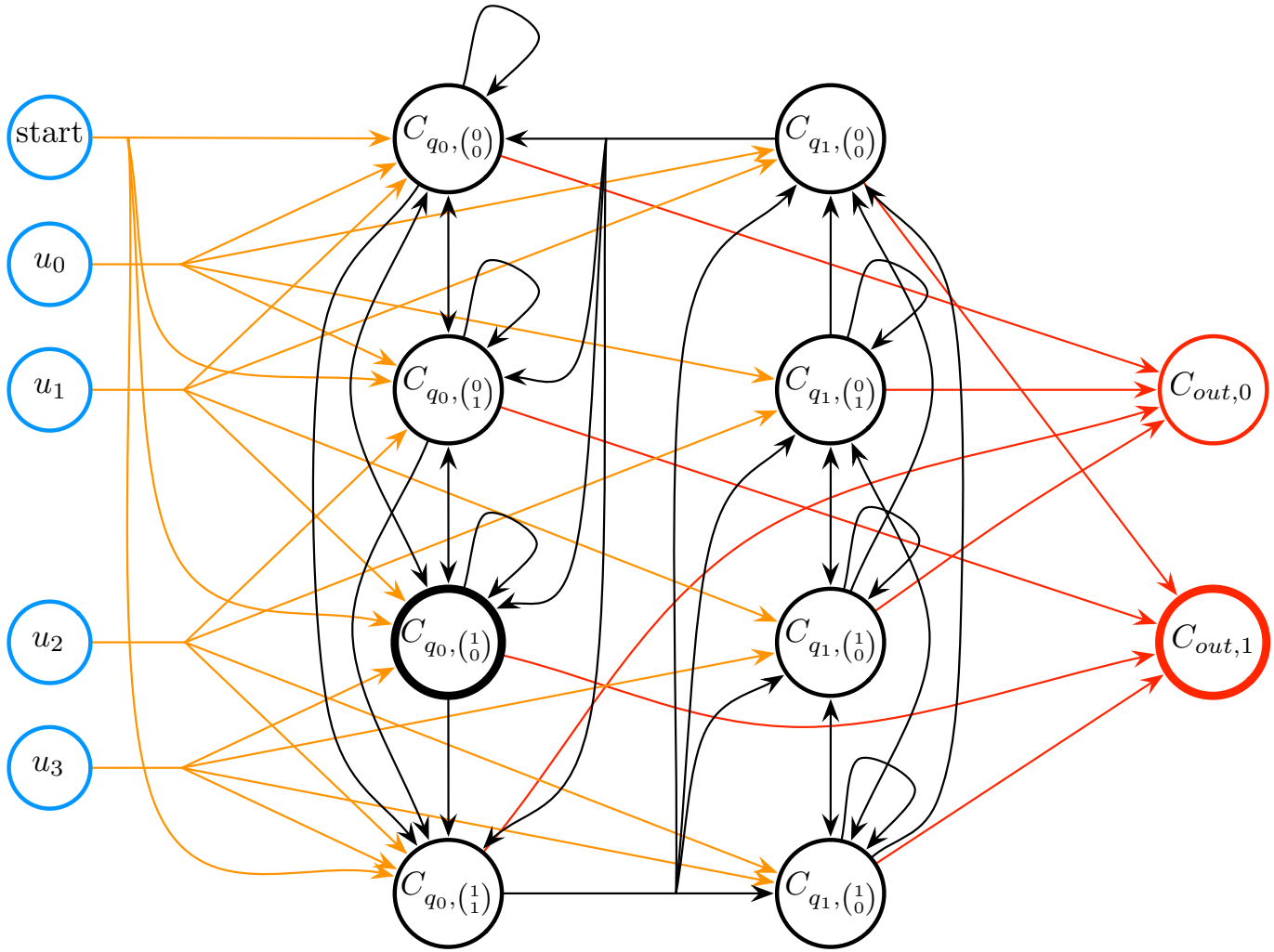
Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

Simulation


start
u_0
u_1
u_2
u_3
$C_{q_0,\binom{0}{0}}$
$C_{q_0,\binom{0}{1}}$
$C_{q_0,\binom{1}{0}}$
$C_{q_0,\binom{1}{1}}$
$C_{q_1,\binom{0}{0}}$
$C_{q_1,\binom{0}{1}}$
$C_{q_1,\binom{1}{0}}$
$C_{q_1,\binom{1}{0}}$
$C_{out,0}$
$C_{out,1}$

Simulation

time	0	1	2	3	4	5	6	7	8
states	q_0	q_1	q_0	q_0	q_1	q_1	q_1	q_0	–
inputs	$\binom{1}{1}$	$\binom{0}{0}$	$\binom{1}{0}$	$\binom{1}{1}$	$\binom{0}{1}$	$\binom{1}{1}$	$\binom{0}{0}$	–	–
outputs	0	1	1	0	0	1	1	–	–
$start$	1	0	0	0	0	0	0	0	0
u_0	0	1	0	0	1	0	1	0	0
u_1	0	1	1	0	0	0	1	0	0
u_2	1	0	0	1	1	1	0	0	0
u_3	1	0	1	1	0	1	0	0	0
$C_{s,i}$	–	$q_0, \binom{1}{1}$	$q_1, \binom{0}{0}$	$q_0, \binom{1}{0}$	$q_0, \binom{1}{1}$	$q_1, \binom{0}{1}$	$q_1, \binom{1}{1}$	$q_1, \binom{0}{0}$	–
$C_{out,0}$	0	0	1	0	0	1	1	0	0
$C_{out,1}$	0	0	0	1	1	0	0	1	1

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.
  - ⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.
- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.
  - ⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.
- ⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.
- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.
- ⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Drawbacks of the Construction

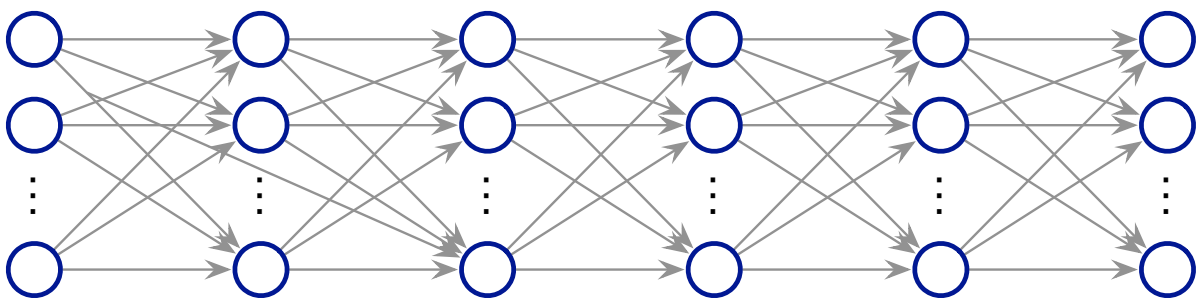
- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.
- ⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.
- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.
- ⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

Drawbacks of the Construction

- Computational states of the automaton are represented as Boolean states, i.e., spiking configurations of the network.
- ⋆ Computational states should rather be represented by *sustained activities of neural assemblies*, e.g., by *cyclic attractors*.
- Network is not robust to cell death, synaptic plasticity, architectural plasticity in general.
- ⋆ Network should be robust to *architectural plasticity* and *synaptic noises*.

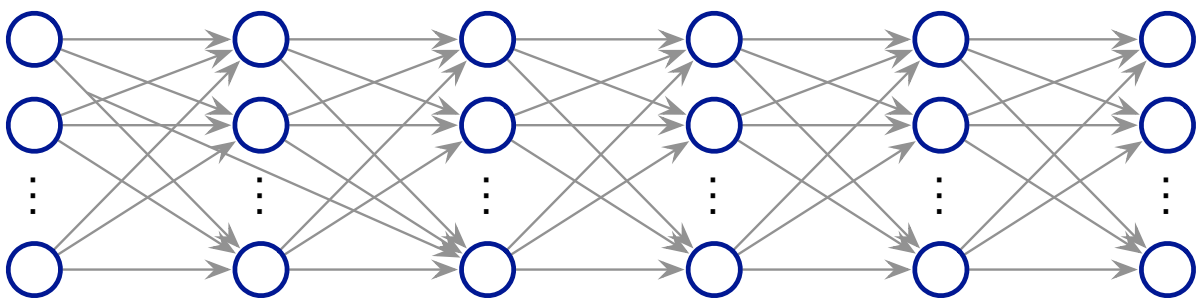
Synfire Chains

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



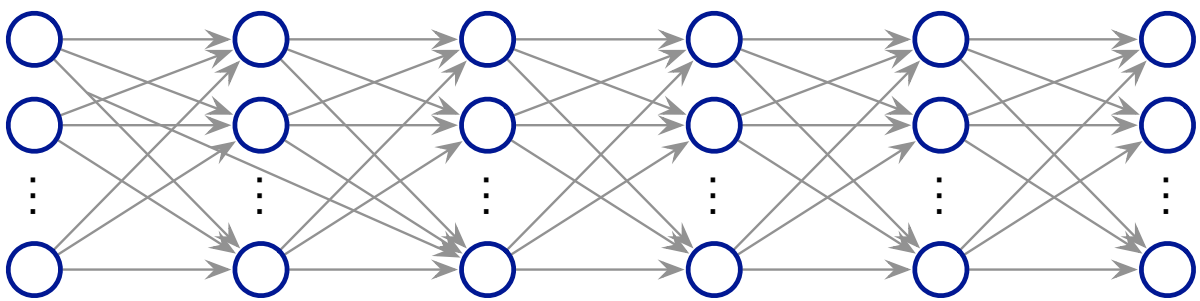
Synfire Chains

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



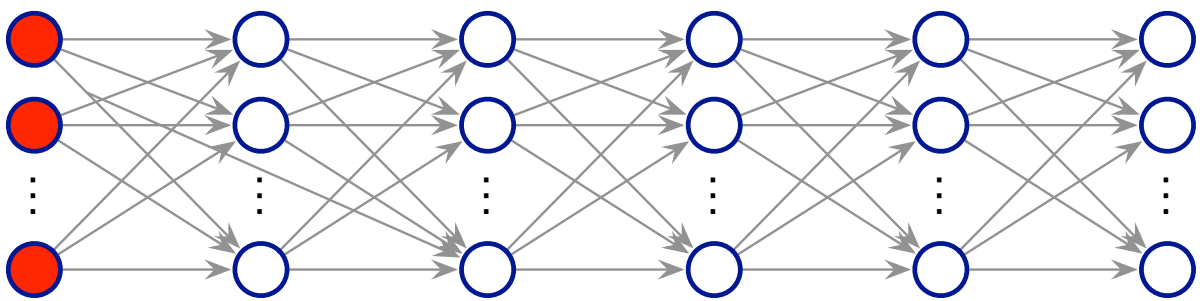
Synfire Chains

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



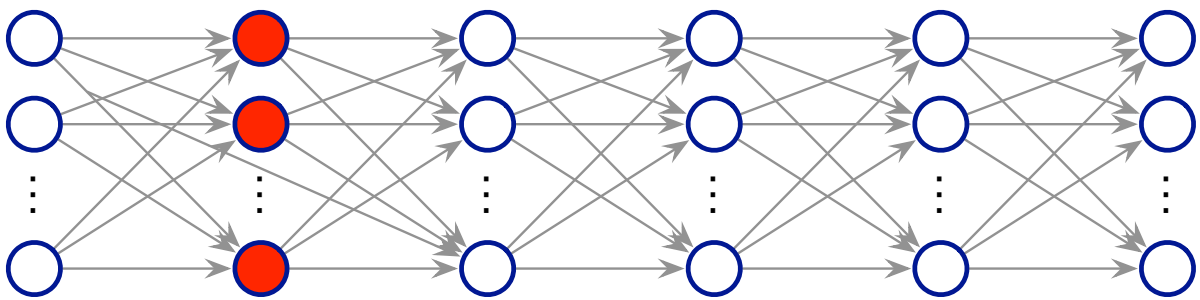
Synfire Chains

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



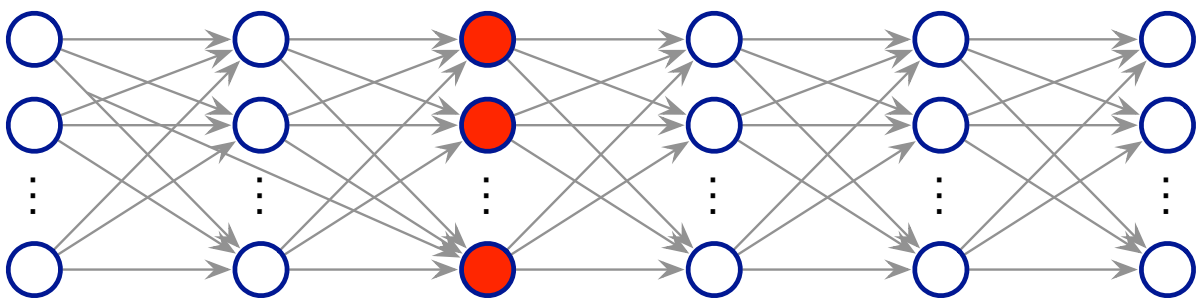
Synfire Chains

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



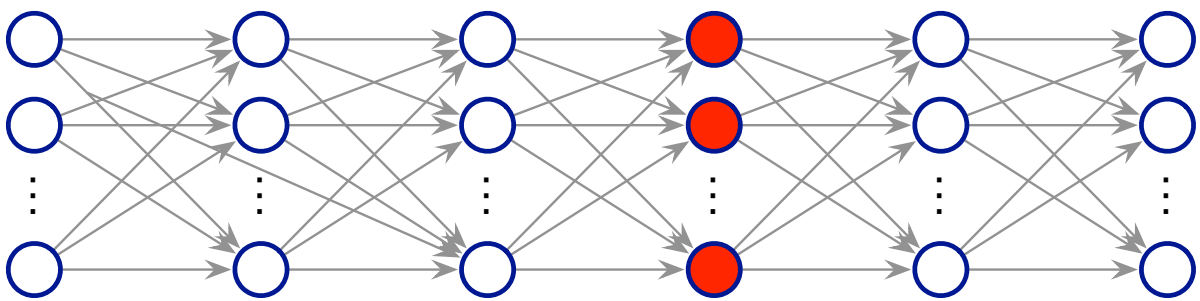
Synfire Chains

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



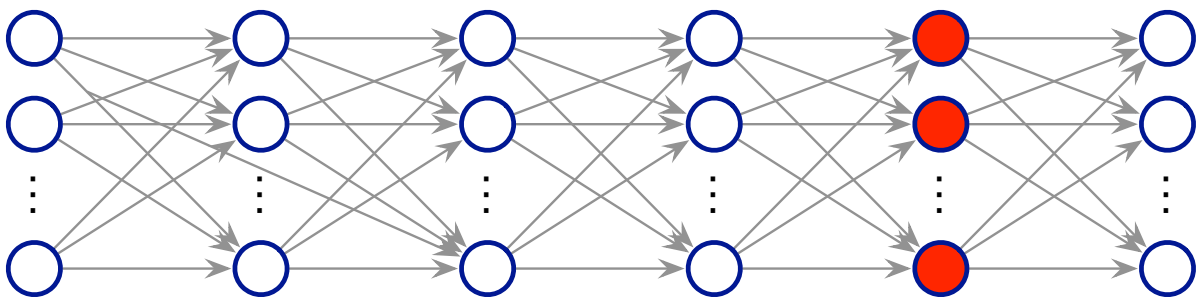
Synfire Chains

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



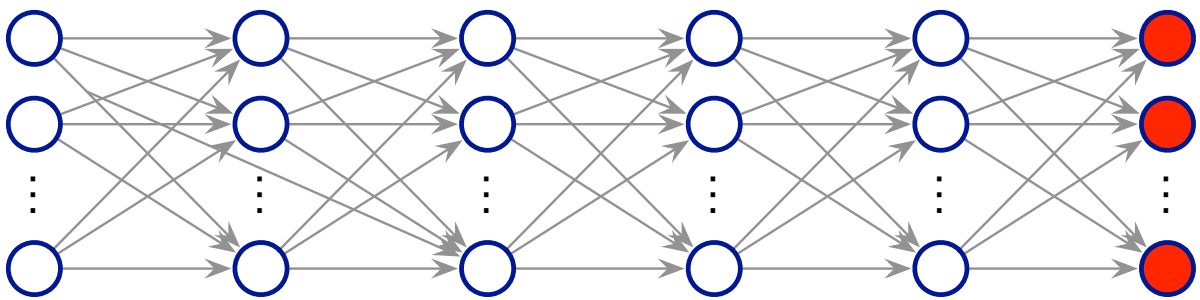
Synfire Chains

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



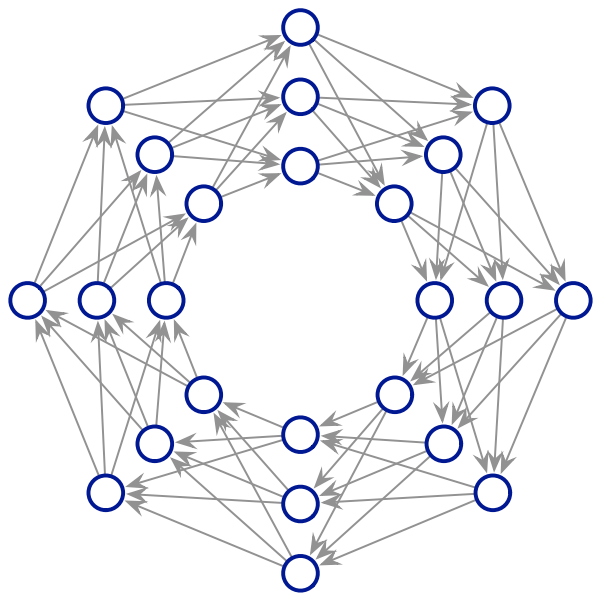
Synfire Chains

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



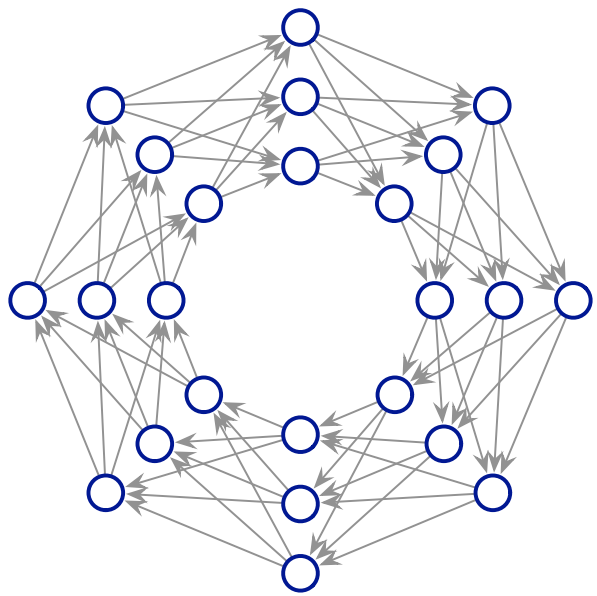
Synfire Rings

- *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*.



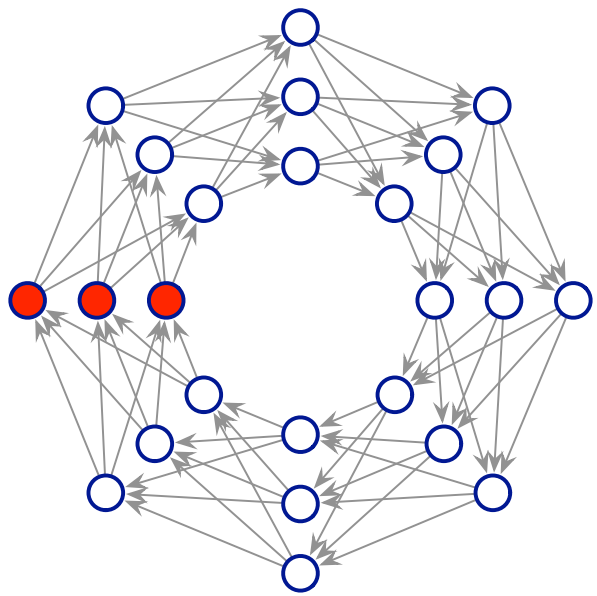
Synfire Rings

- *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*.



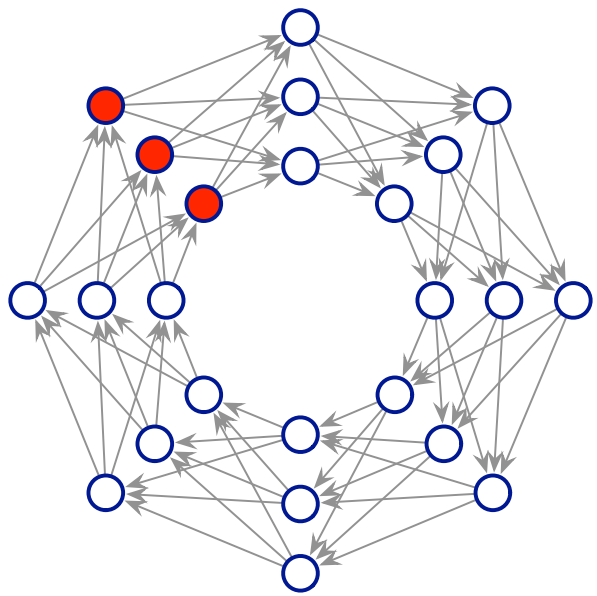
Synfire Rings

- *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*.



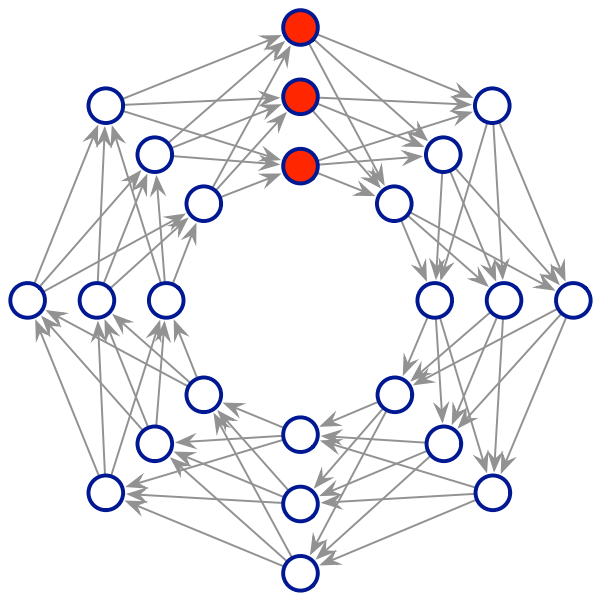
Synfire Rings

- *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*.



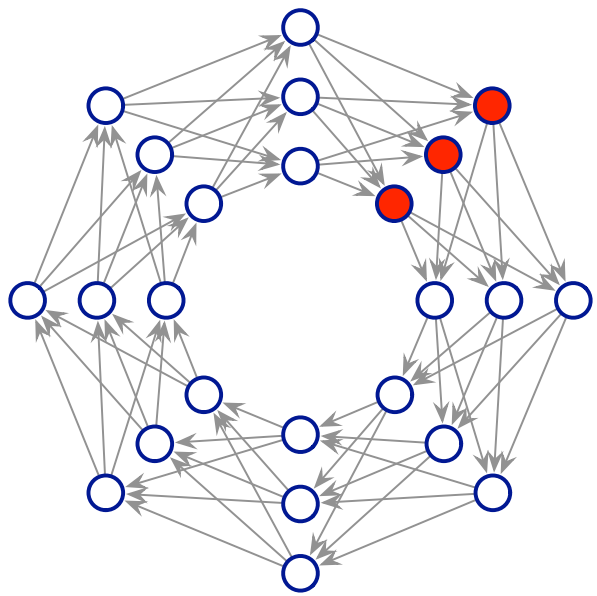
Synfire Rings

- *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*.



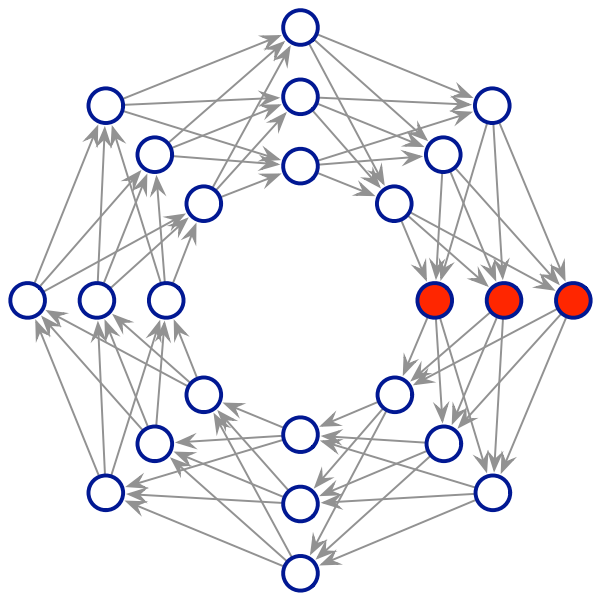
Synfire Rings

- *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*.



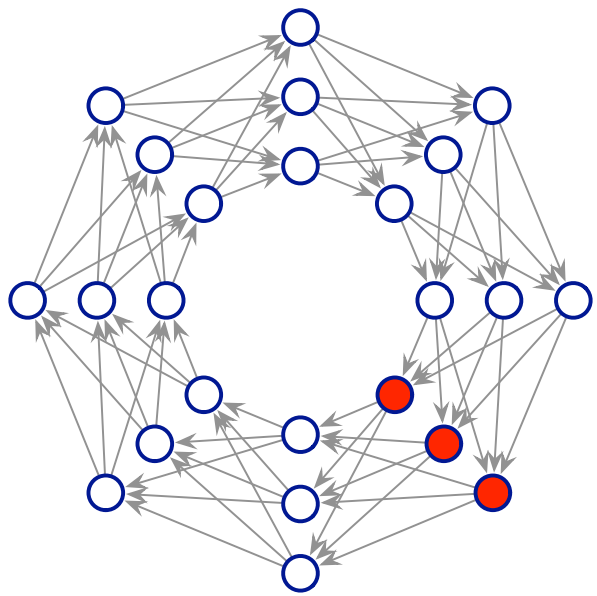
Synfire Rings

- *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*.



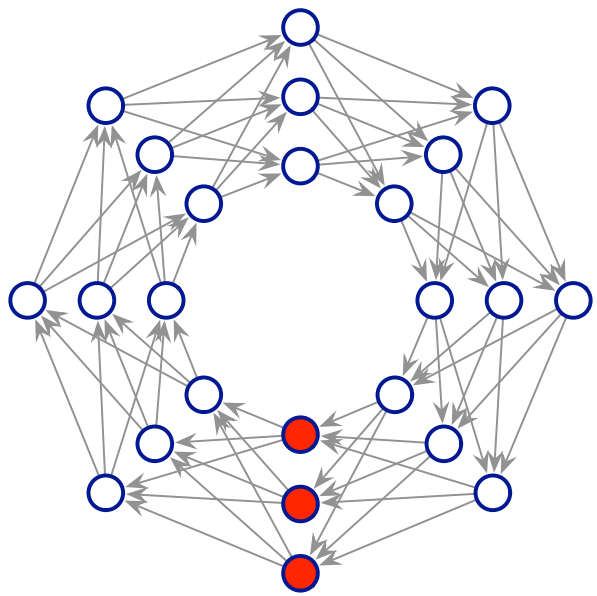
Synfire Rings

- *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*.



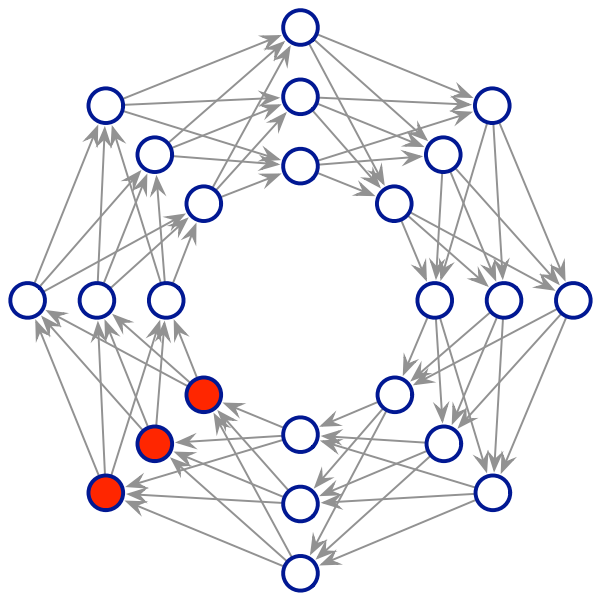
Synfire Rings

- *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*.



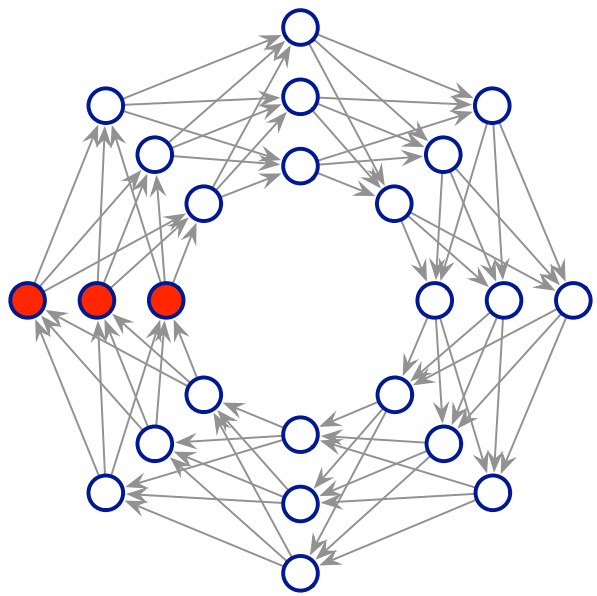
Synfire Rings

- *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*.



Neural Computation with Synfire Rings

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

Neural Computation with Synfire Rings

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

Neural Computation with Synfire Rings

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

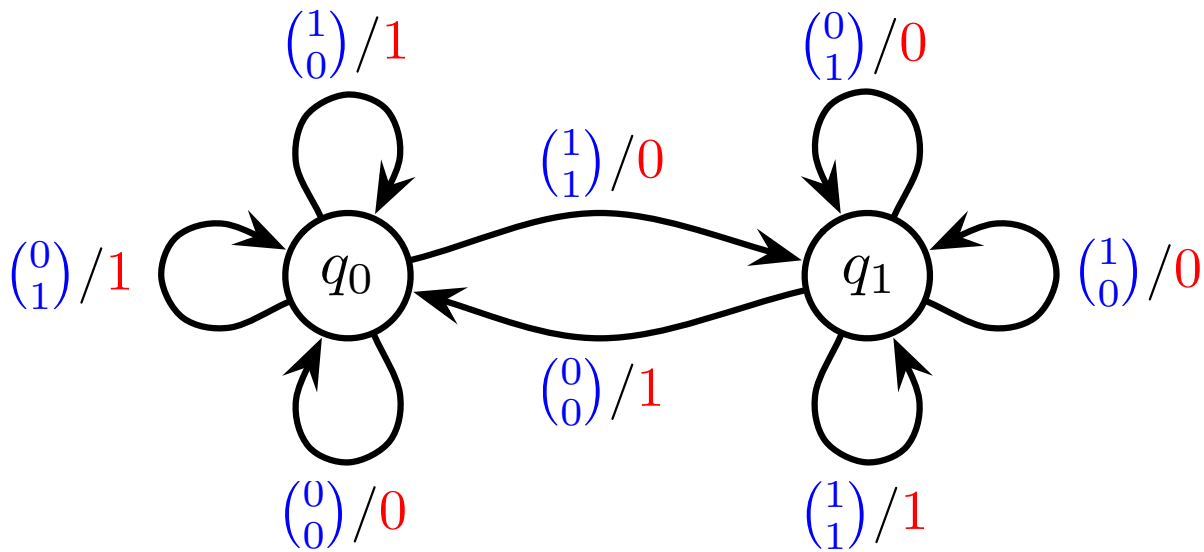
Neural Computation with Synfire Rings

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

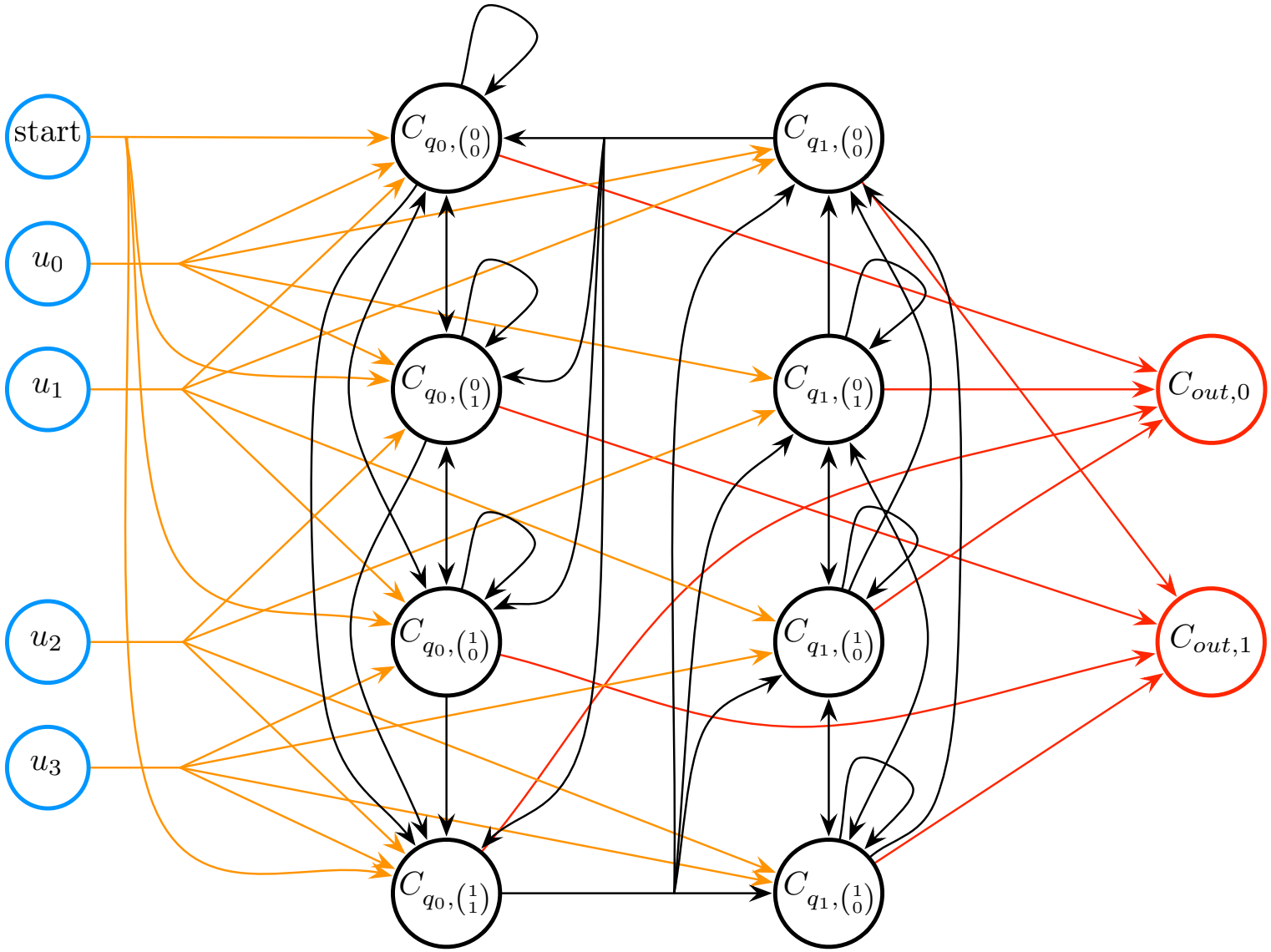
Neural Computation with Synfire Rings

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

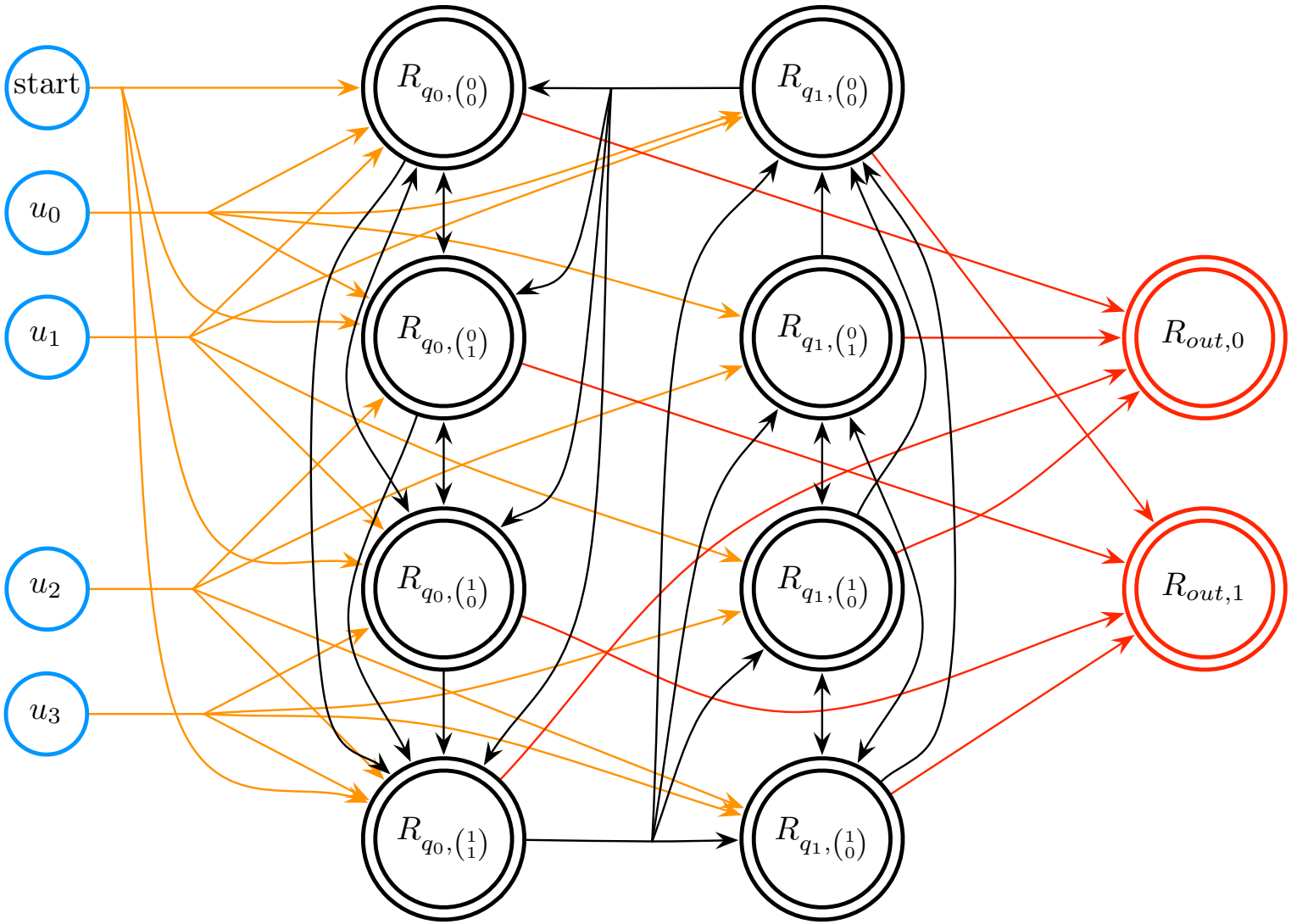
General Construction



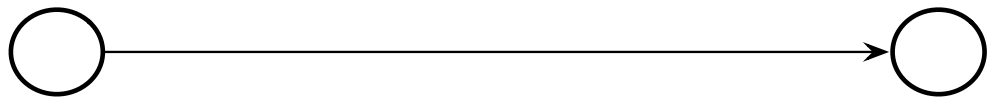
General Construction



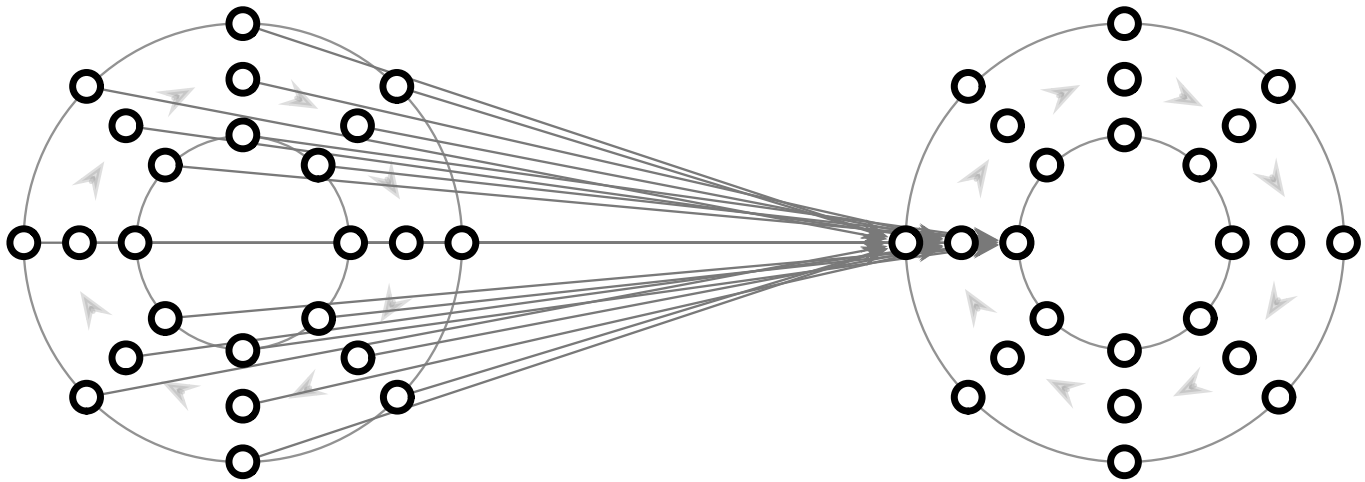
General Construction



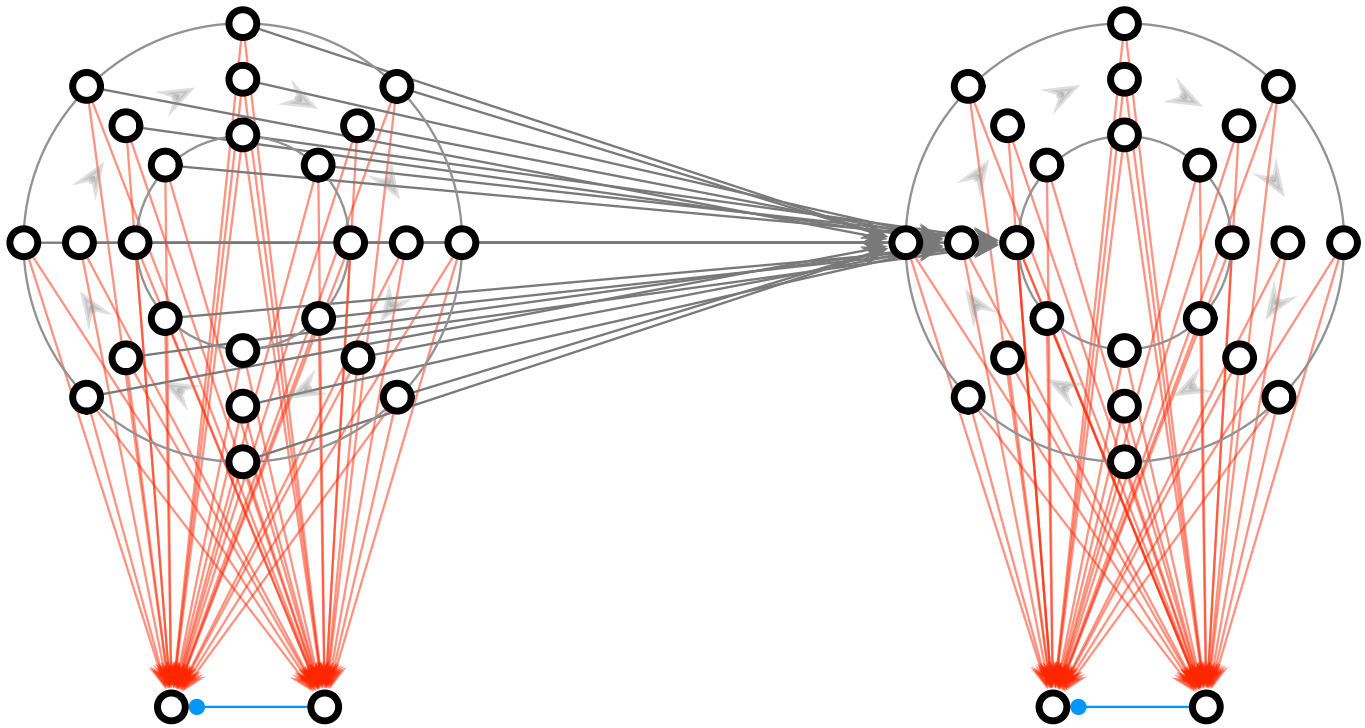
Fibres of connections & Inhibitory System



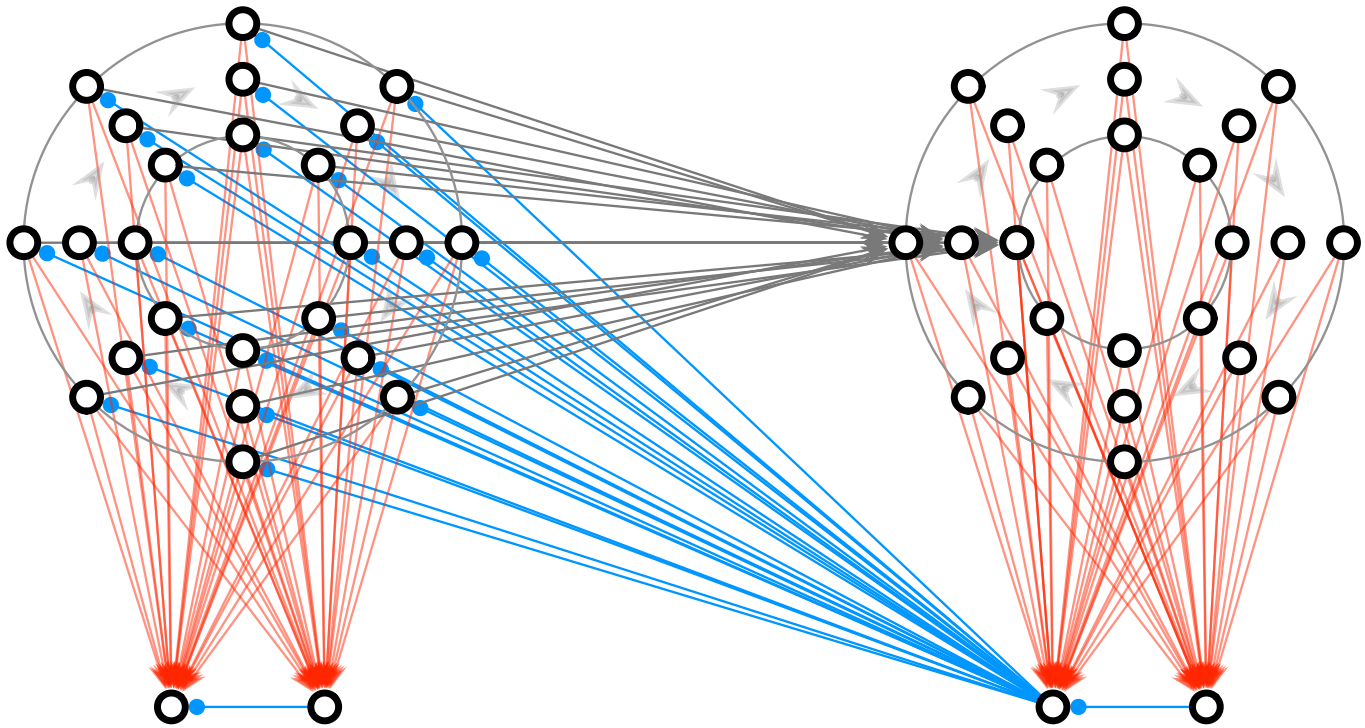
Fibres of connections & Inhibitory System



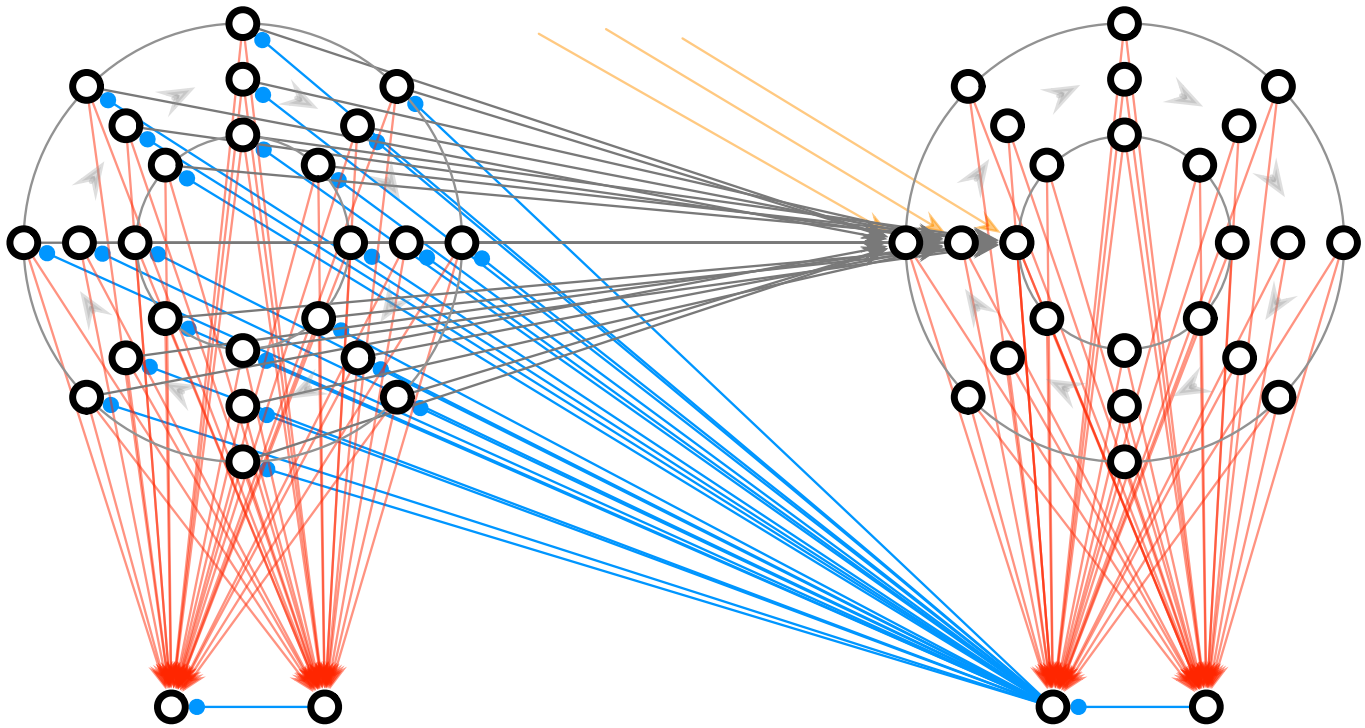
Fibres of connections & Inhibitory System



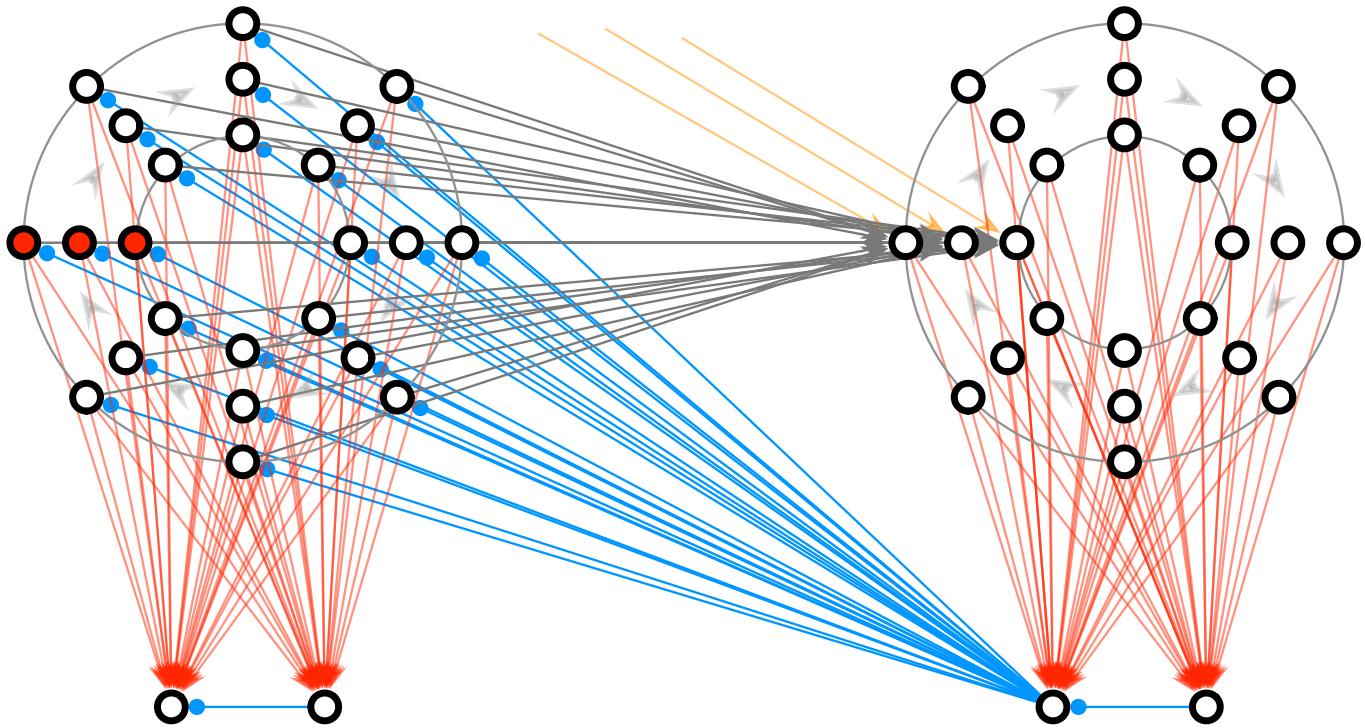
Fibres of connections & Inhibitory System



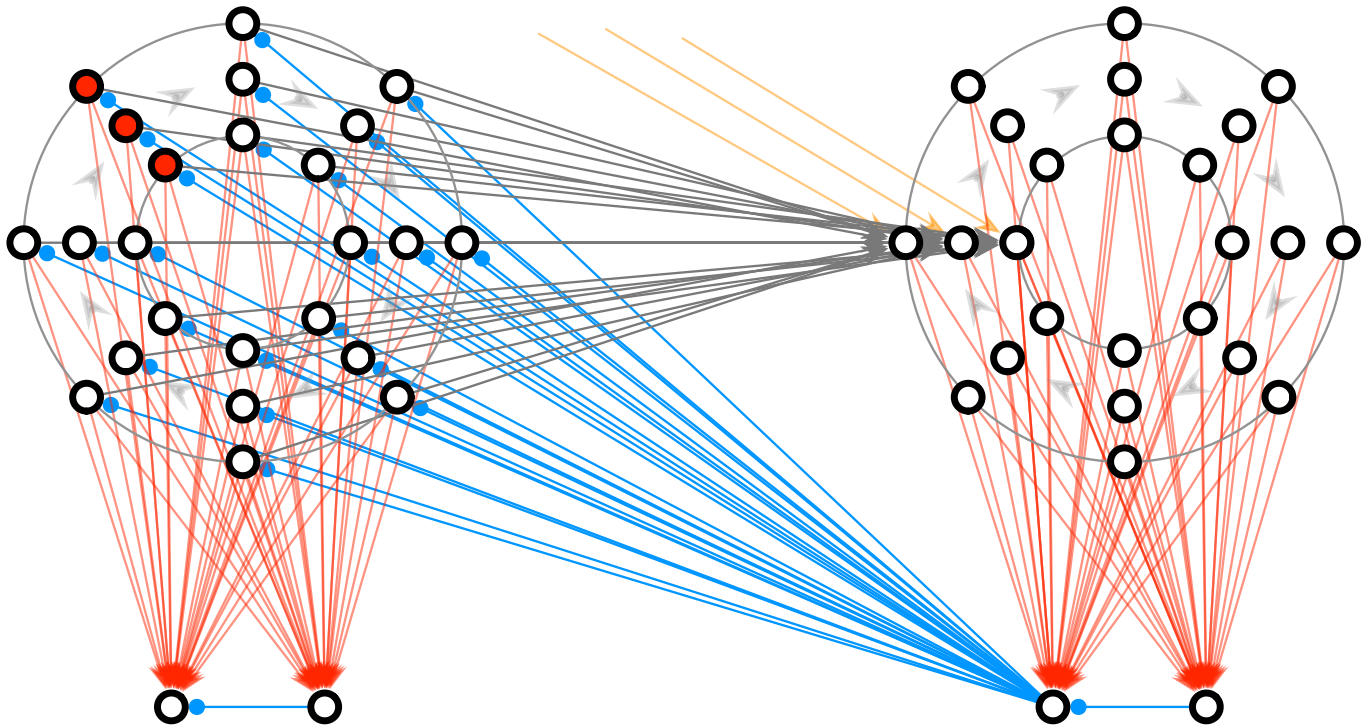
Fibres of connections & Inhibitory System



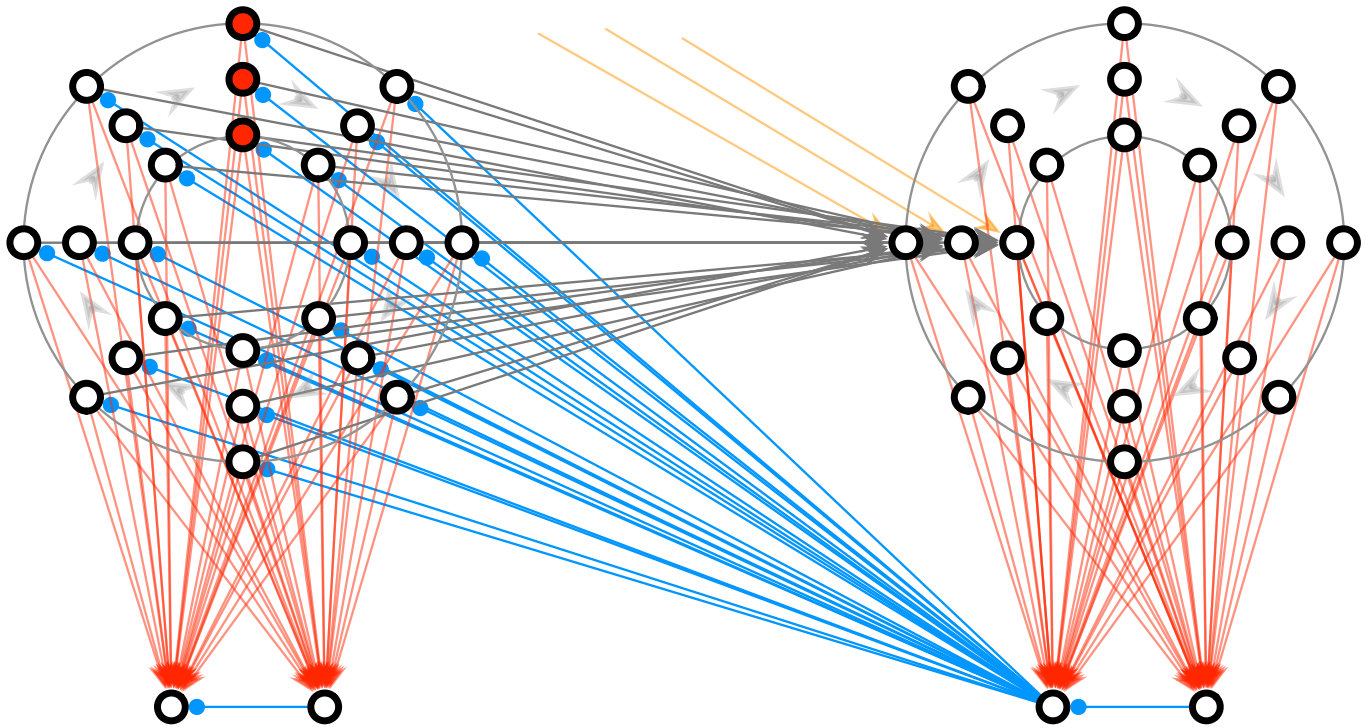
Fibres of connections & Inhibitory System



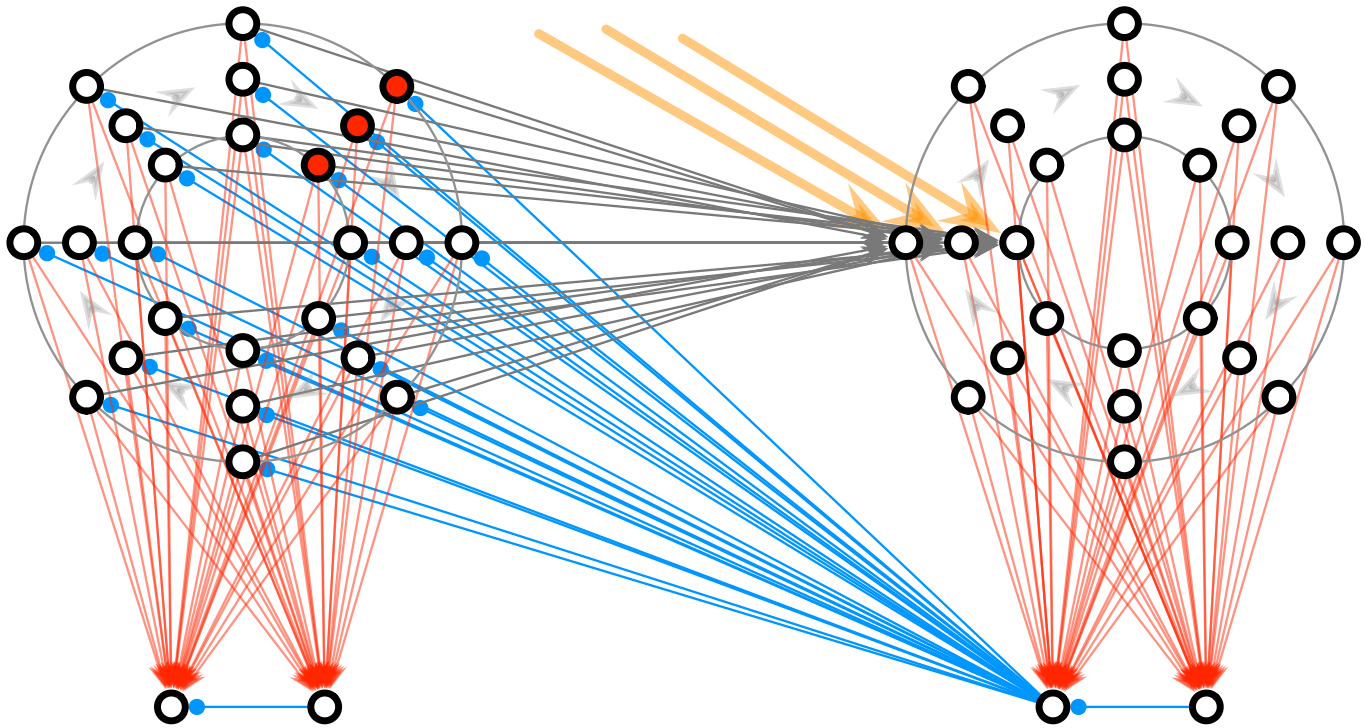
Fibres of connections & Inhibitory System



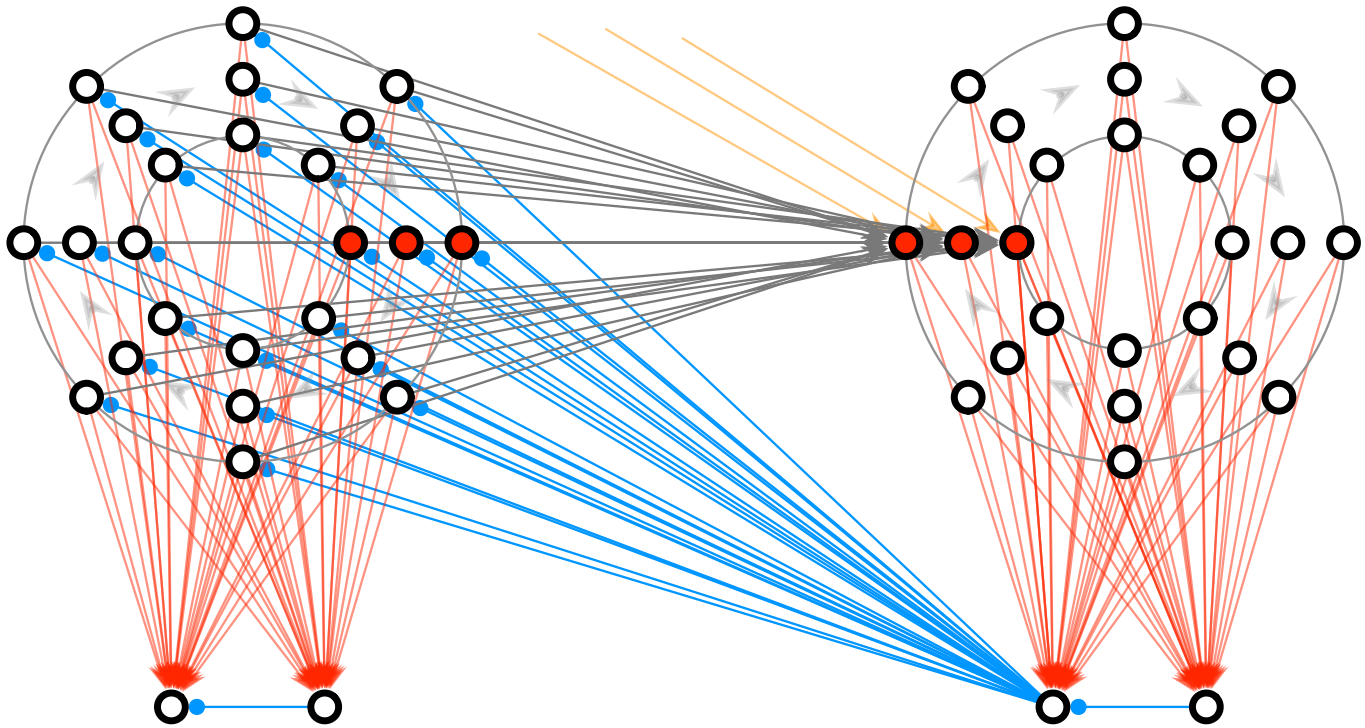
Fibres of connections & Inhibitory System



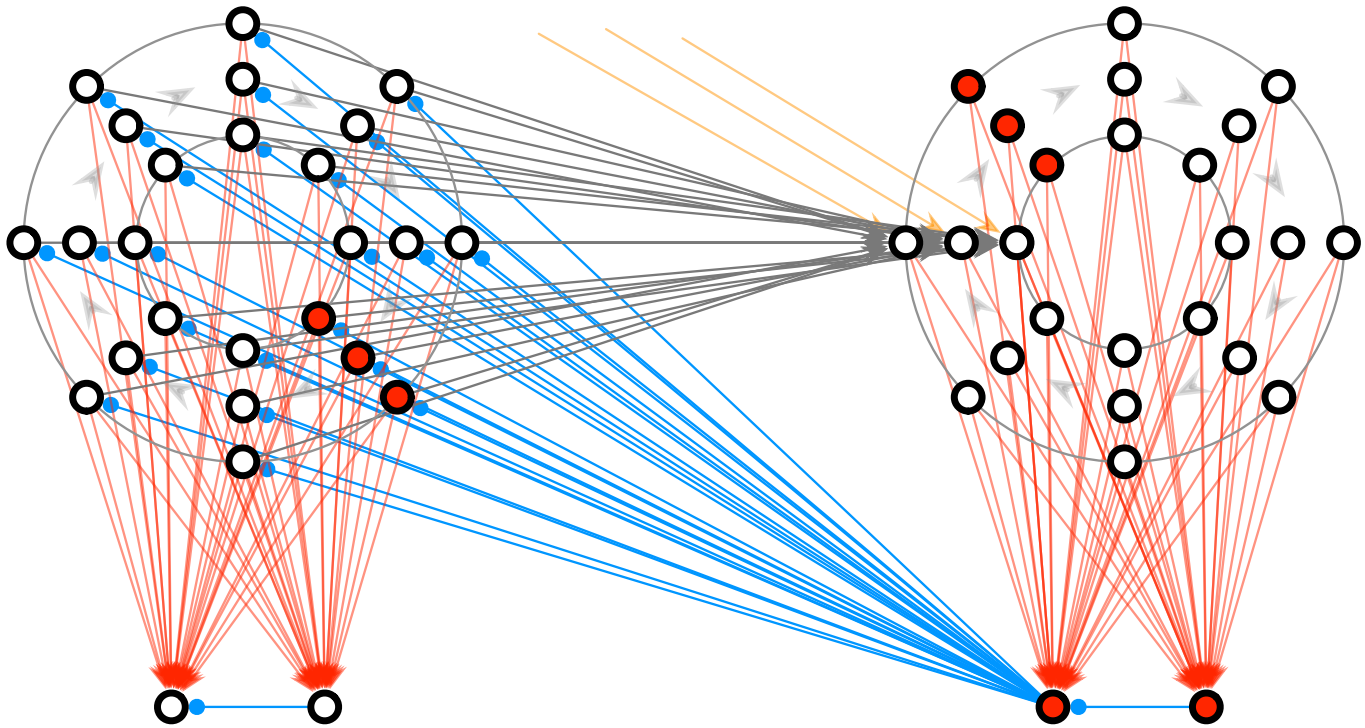
Fibres of connections & Inhibitory System



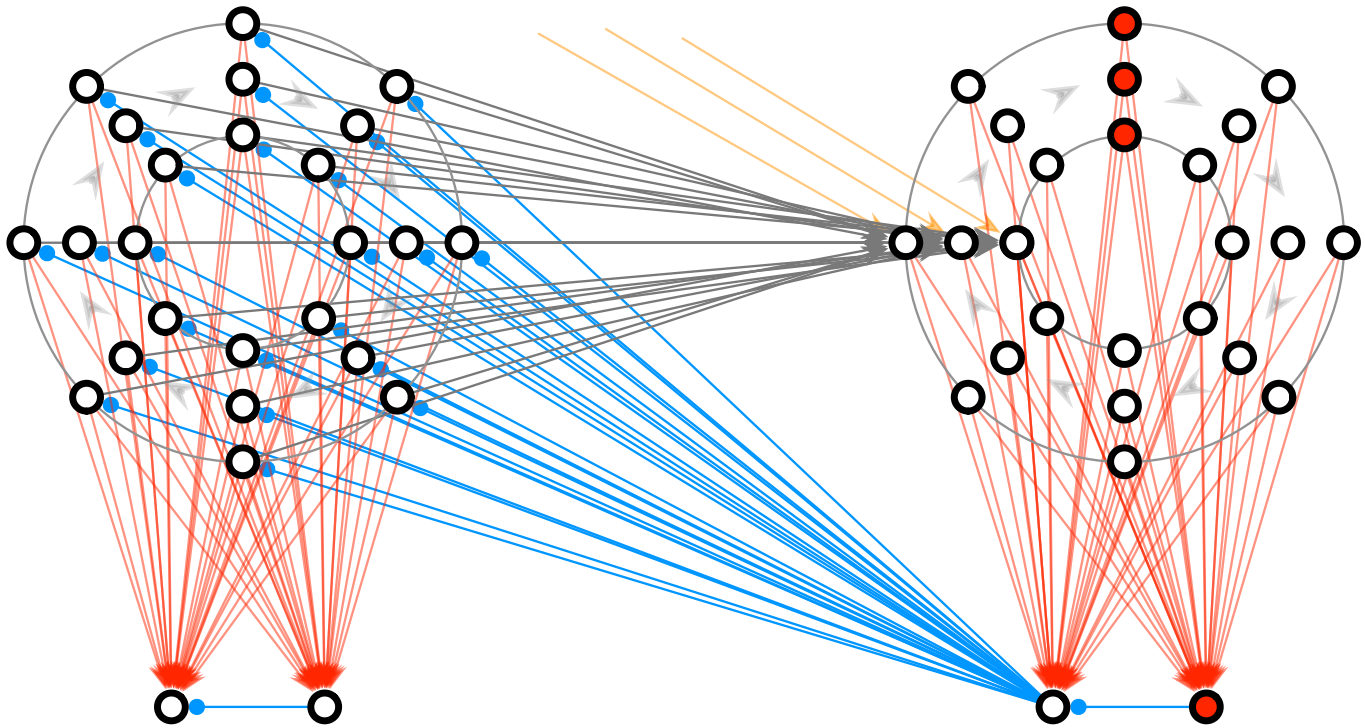
Fibres of connections & Inhibitory System



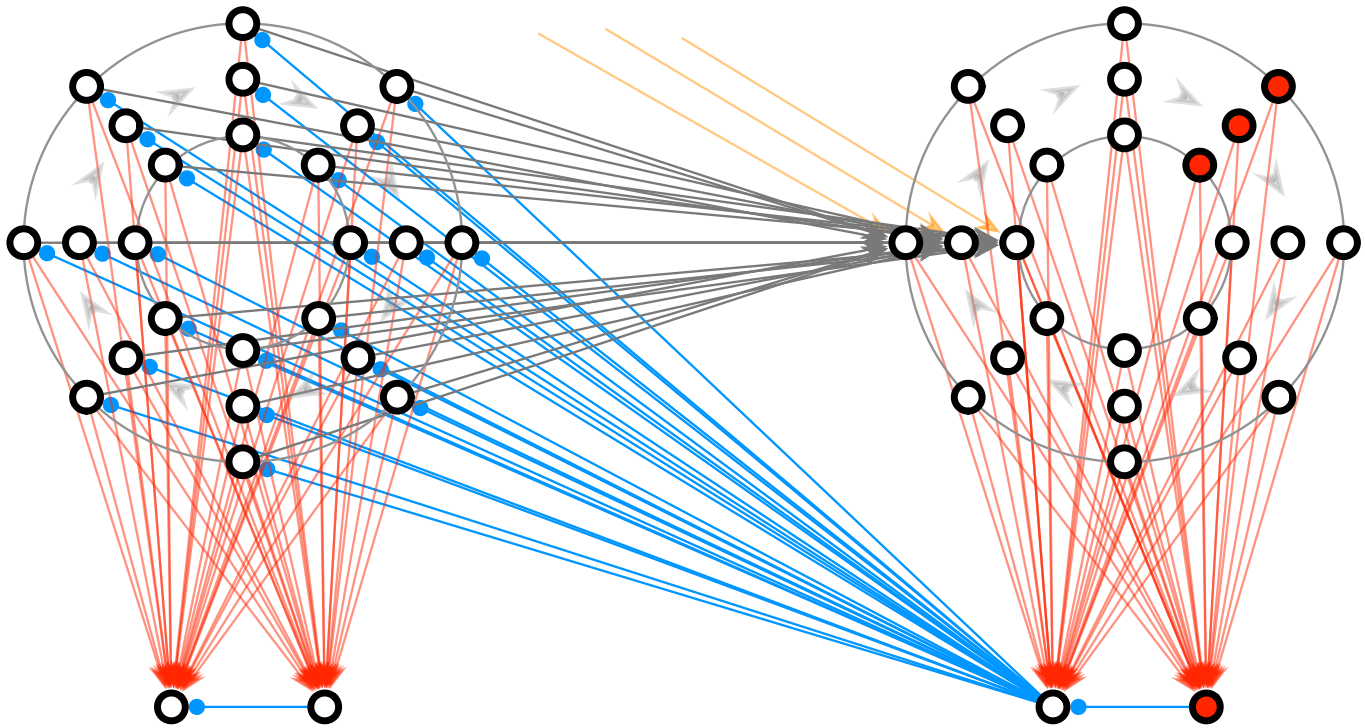
Fibres of connections & Inhibitory System



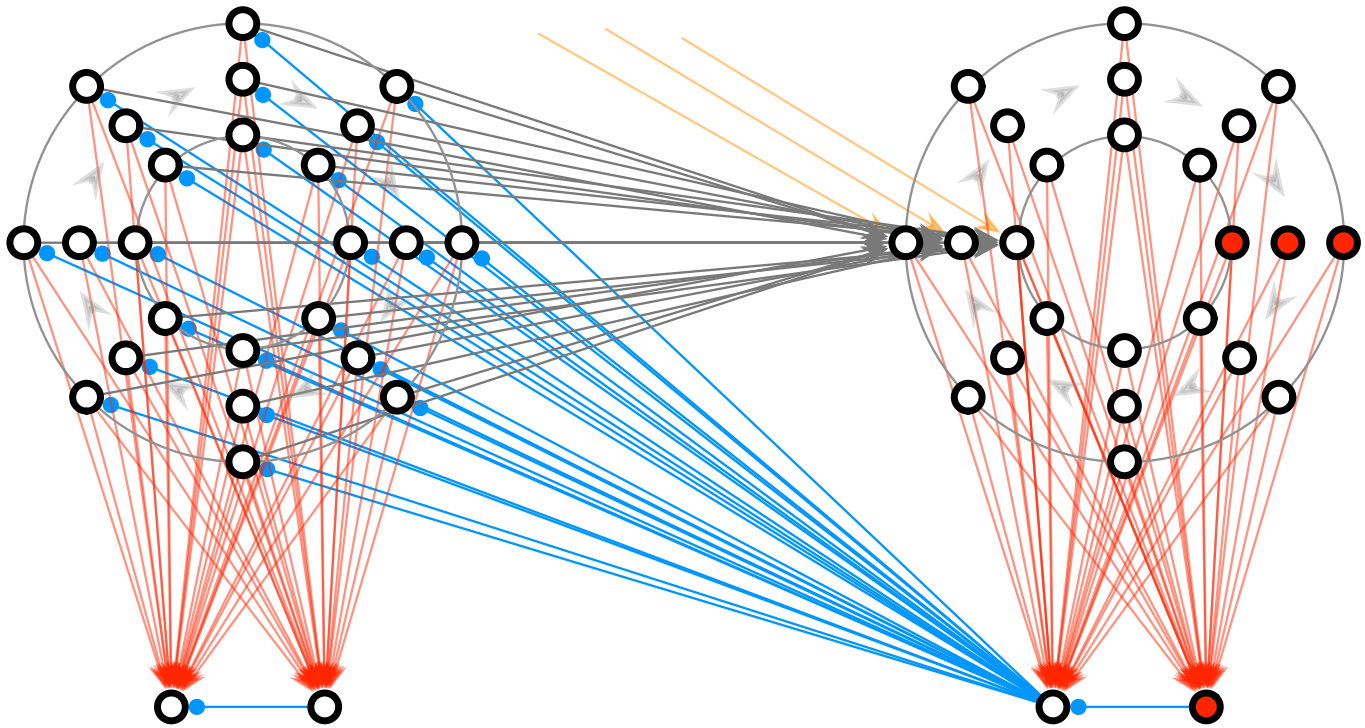
Fibres of connections & Inhibitory System



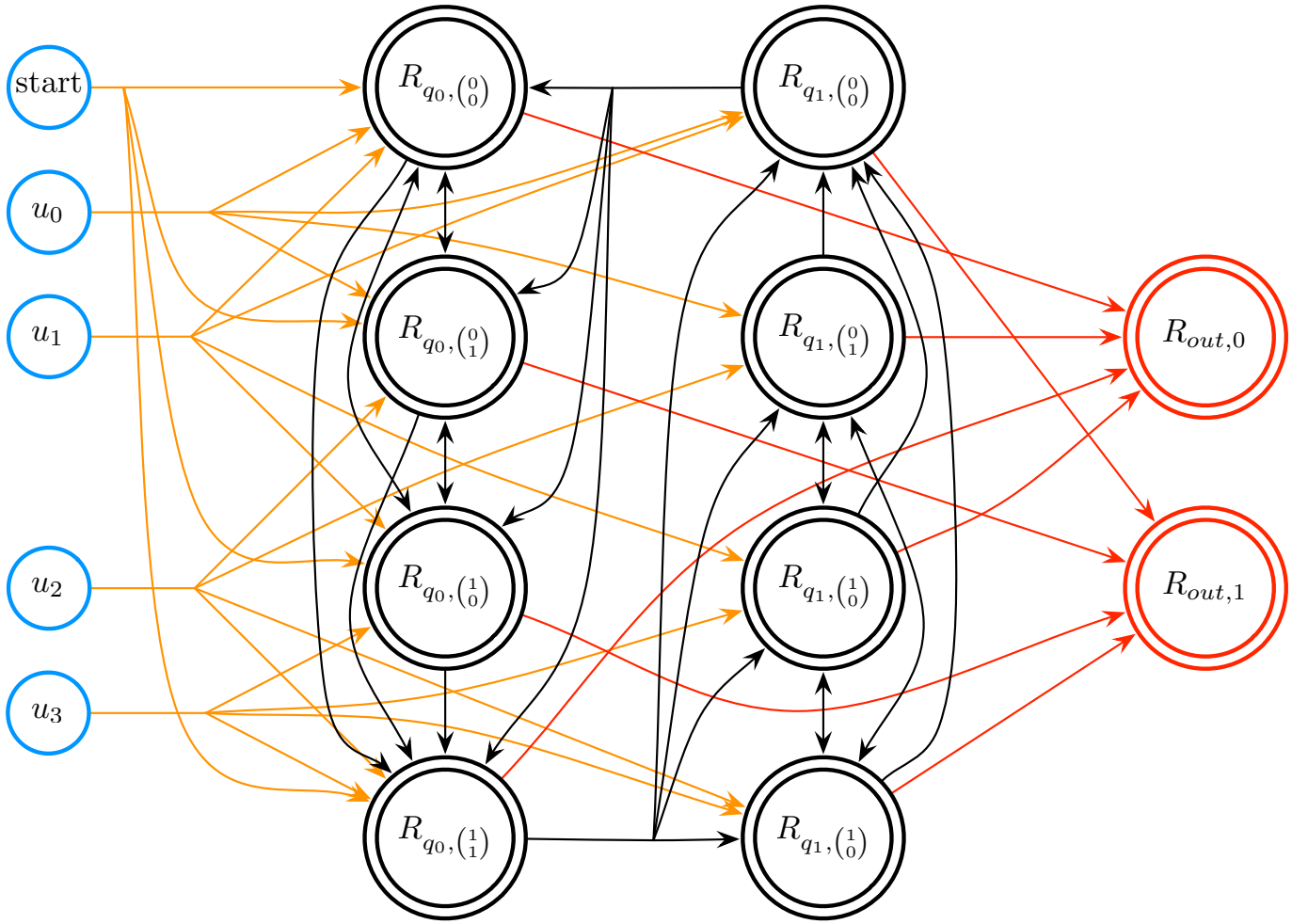
Fibres of connections & Inhibitory System



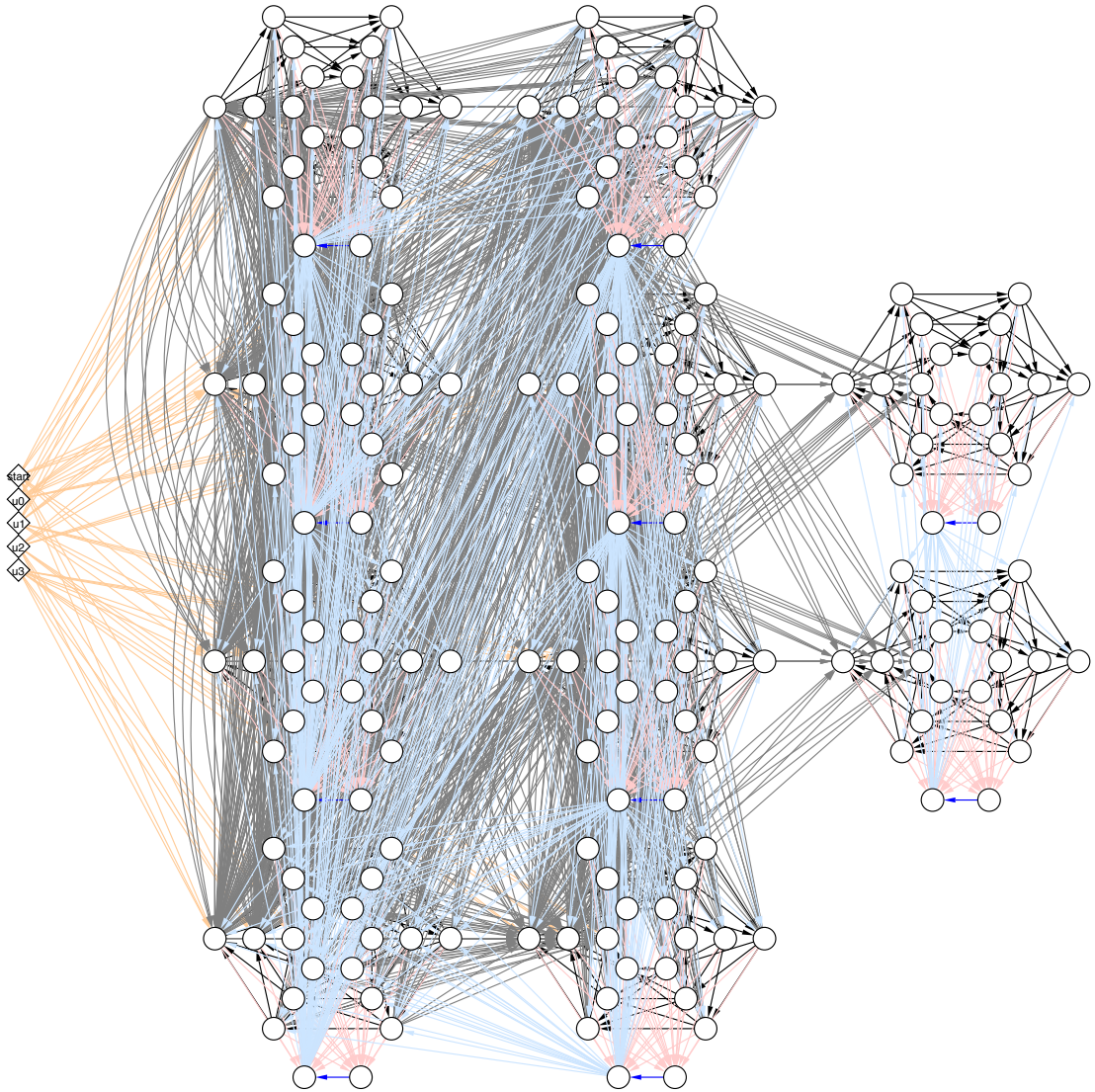
Fibres of connections & Inhibitory System



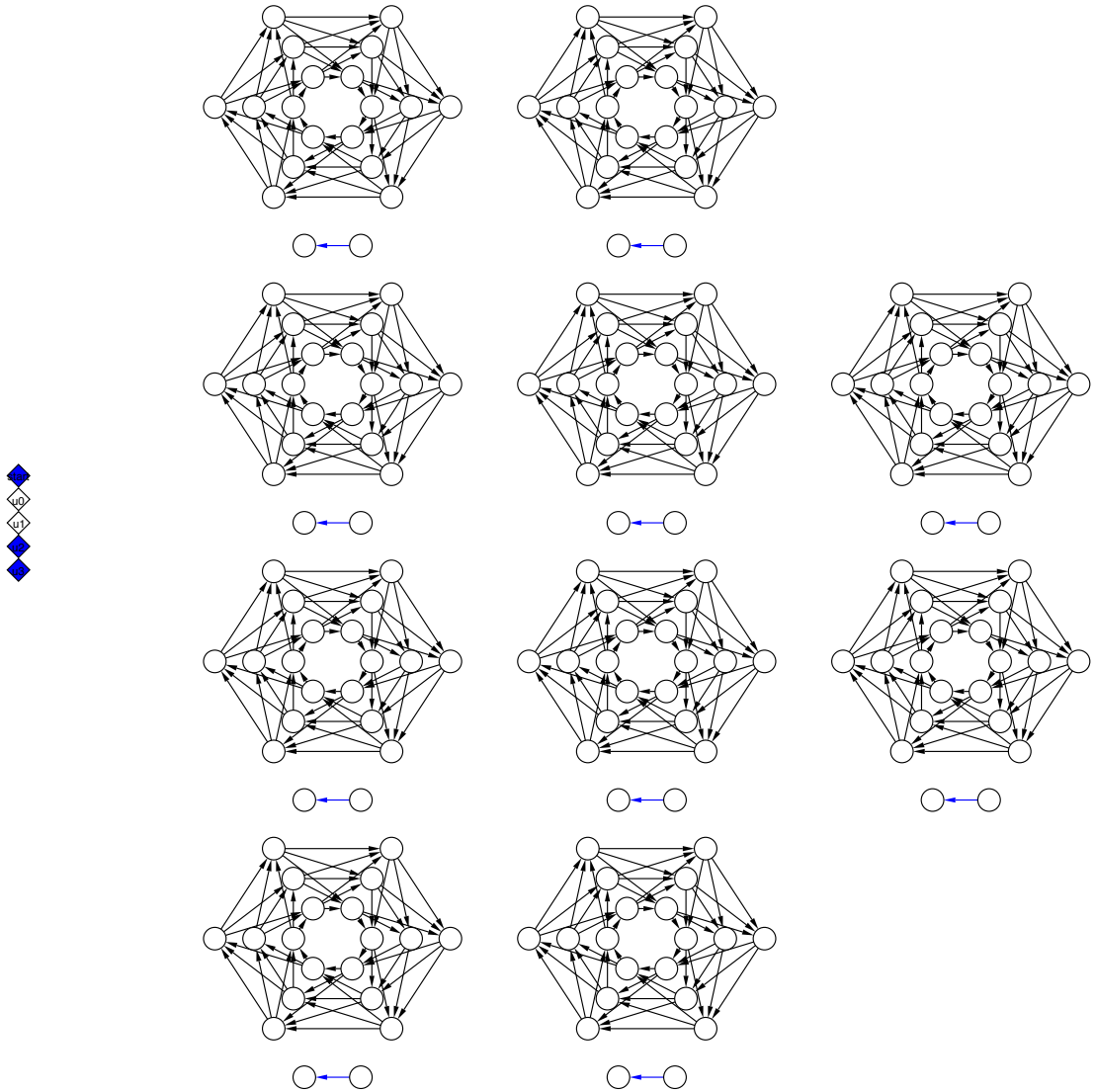
Simulation


start
u_0
u_1
u_2
u_3
$R_{q_0,\binom{0}{0}}$
$R_{q_0,\binom{0}{1}}$
$R_{q_0,\binom{1}{0}}$
$R_{q_0,\binom{1}{1}}$
$R_{q_1,\binom{0}{0}}$
$R_{q_1,\binom{0}{1}}$
$R_{q_1,\binom{1}{0}}$
$R_{q_1,\binom{1}{0}}$
$R_{out,0}$
$R_{out,1}$

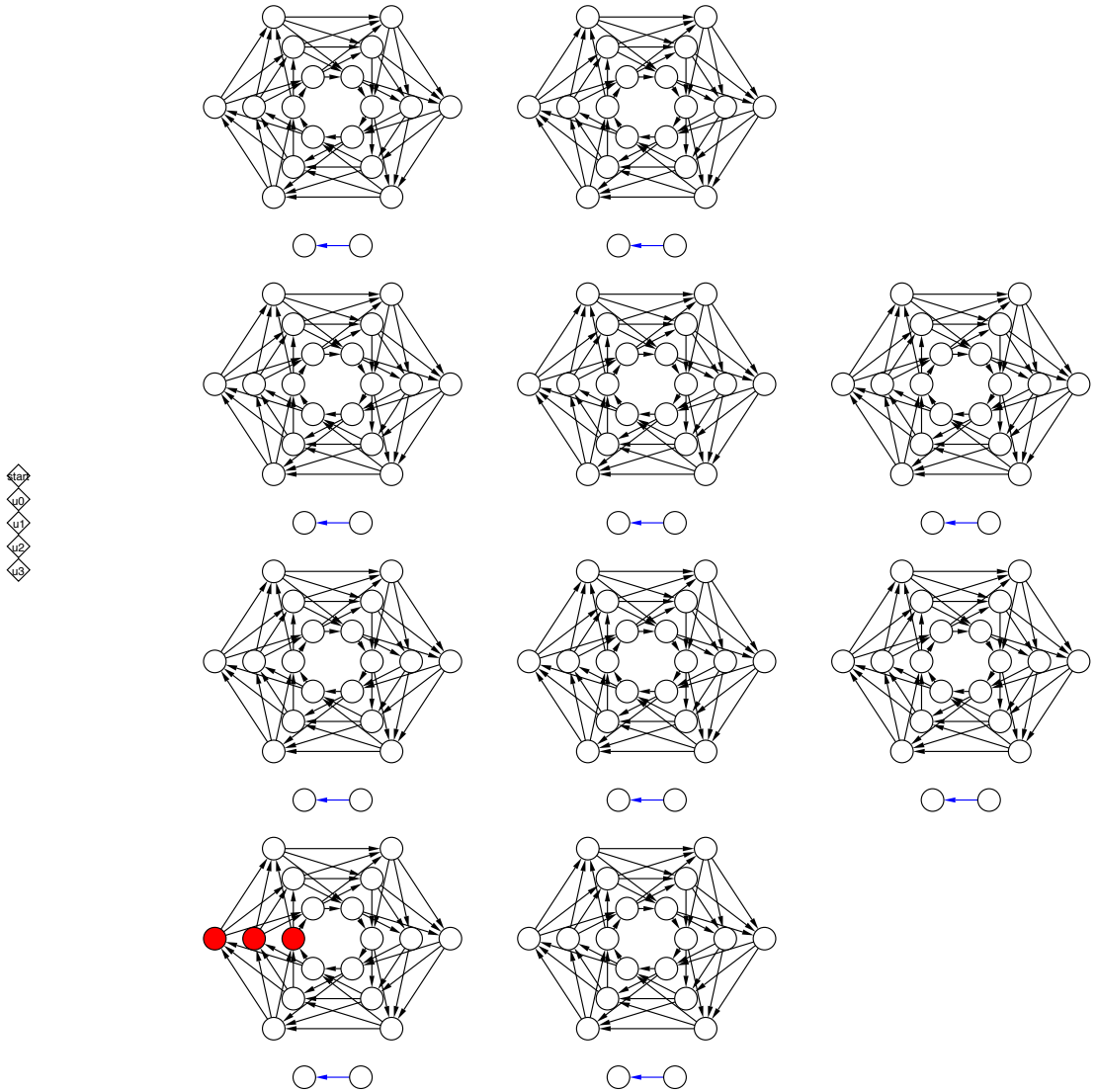
Simulation



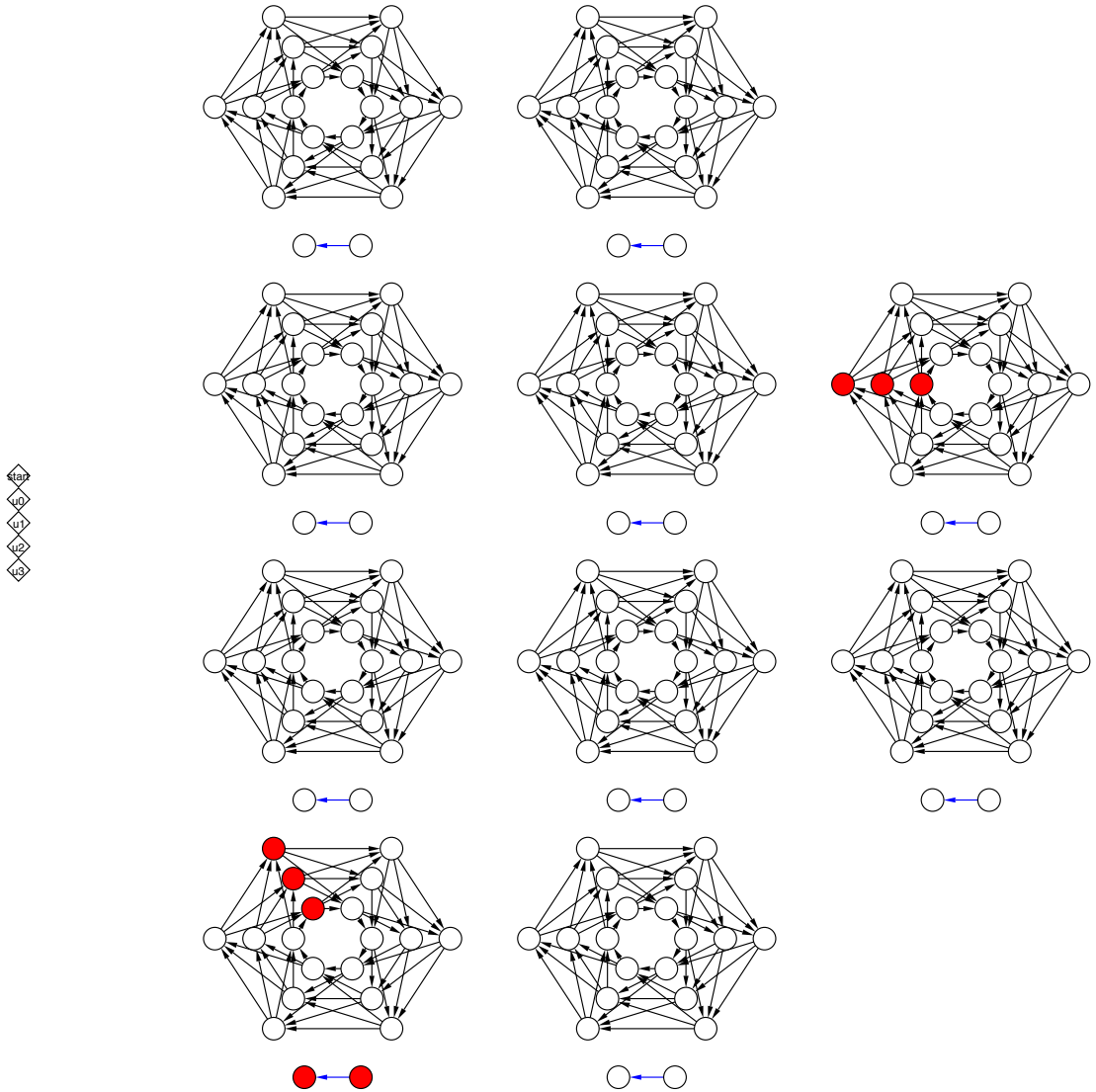
Simulation



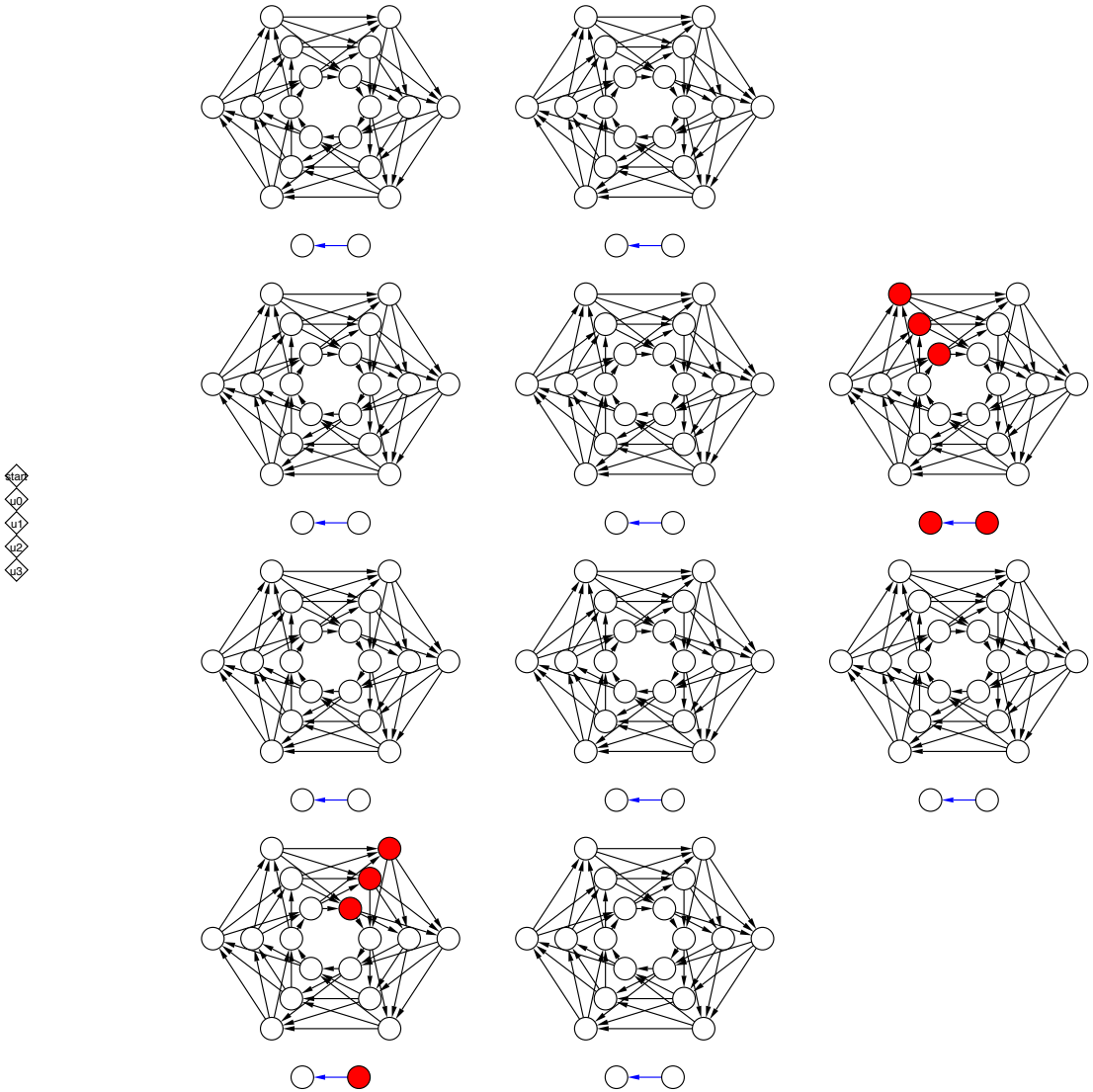
Simulation



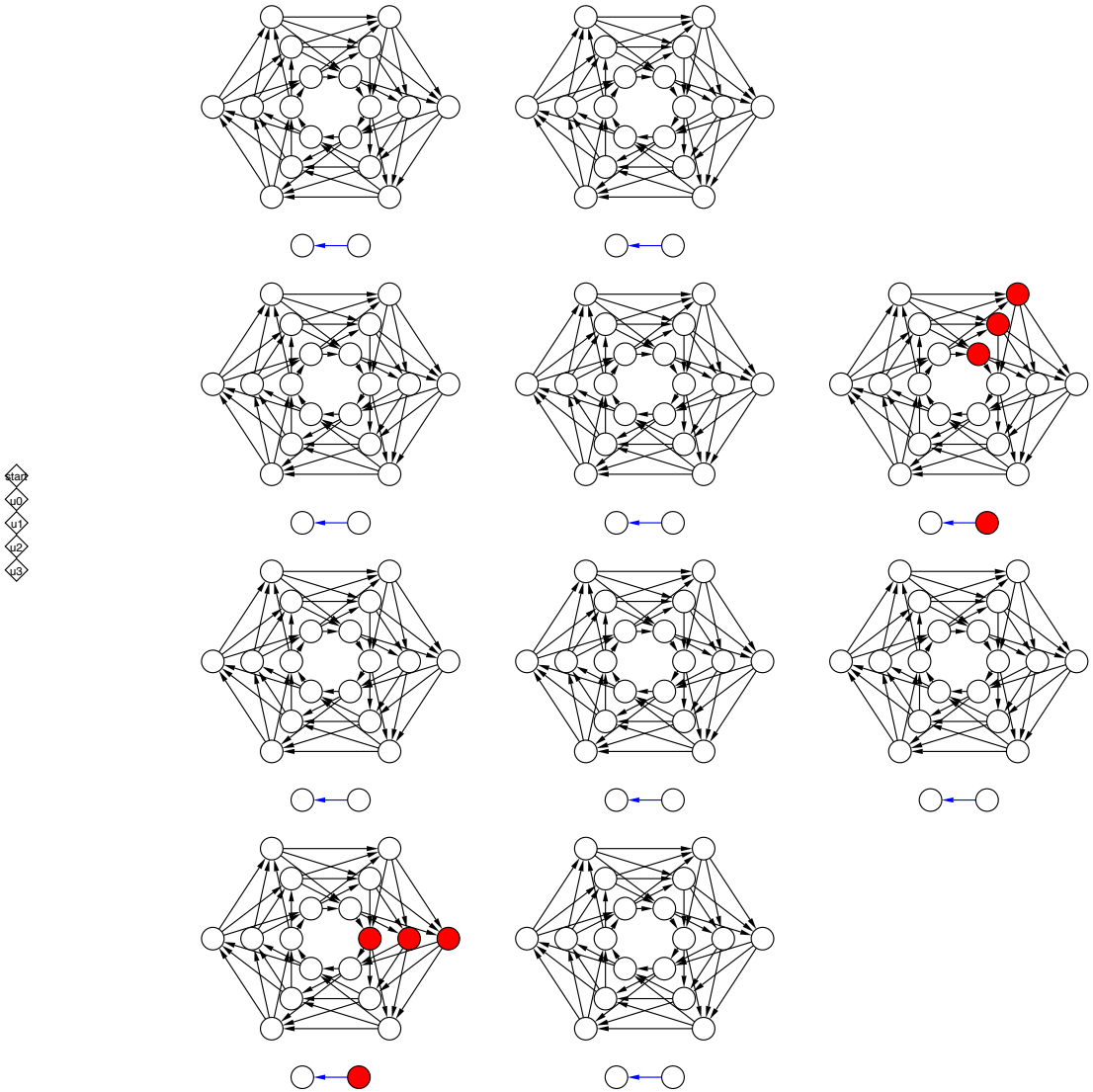
Simulation



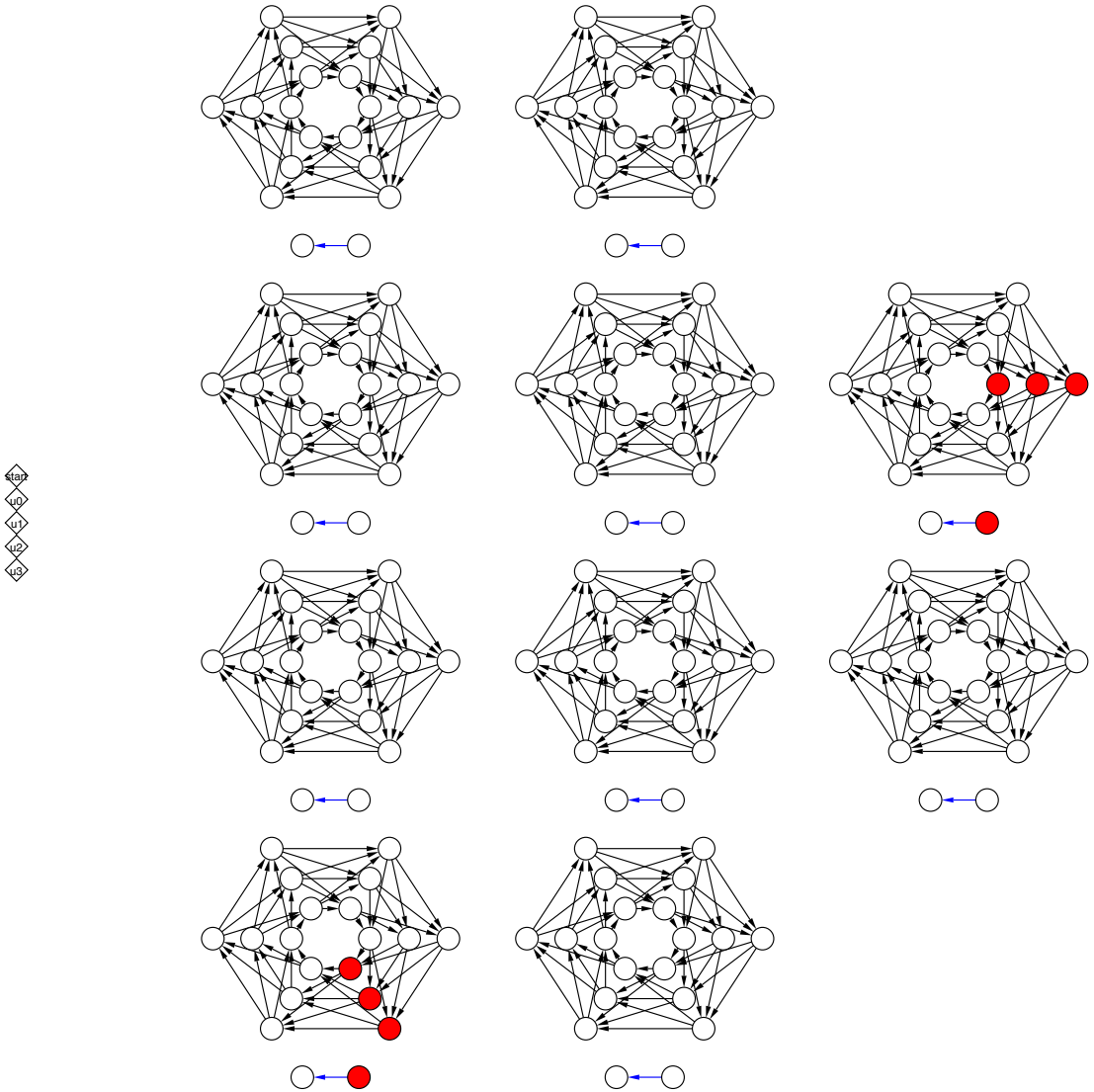
Simulation



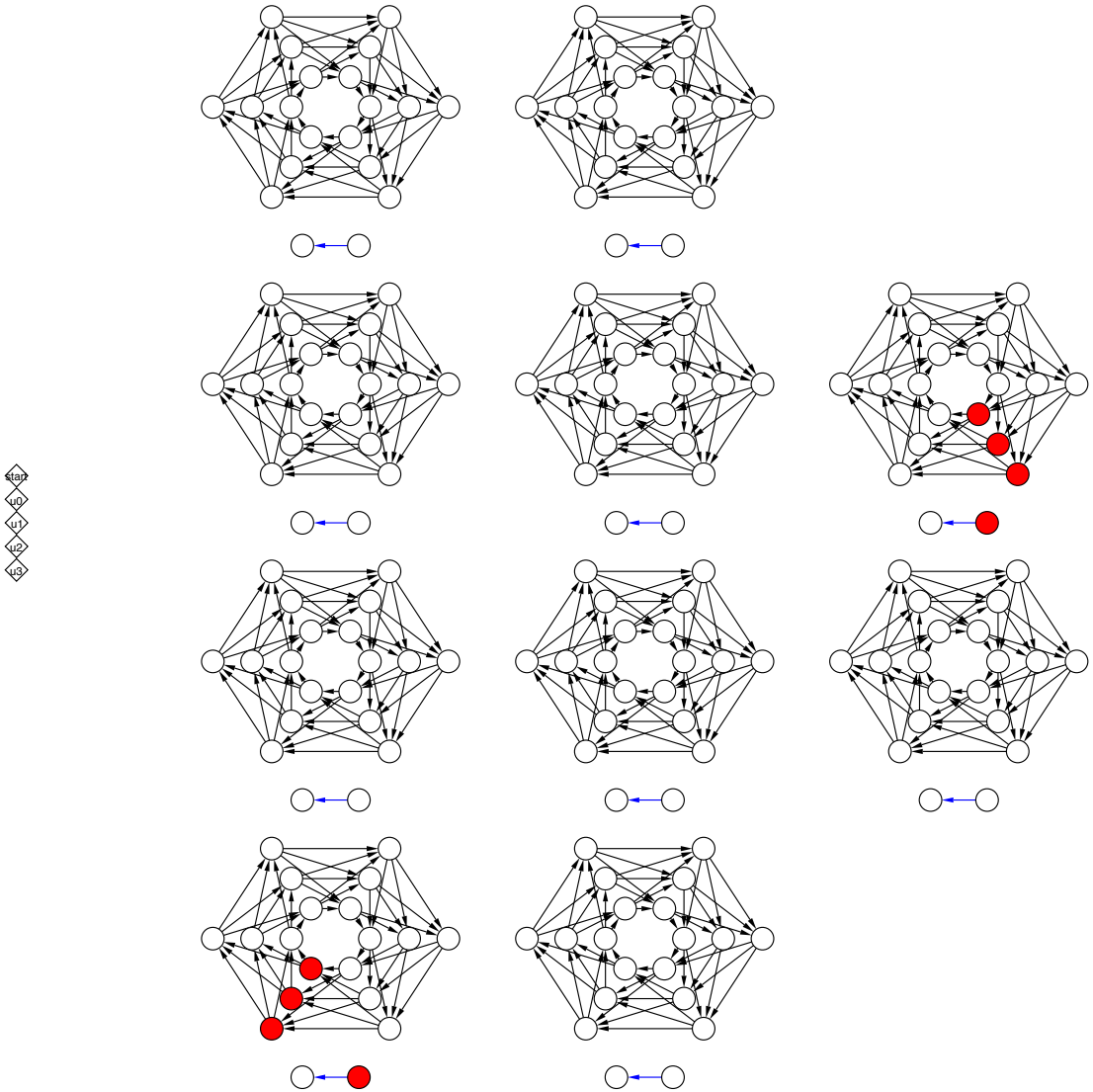
Simulation



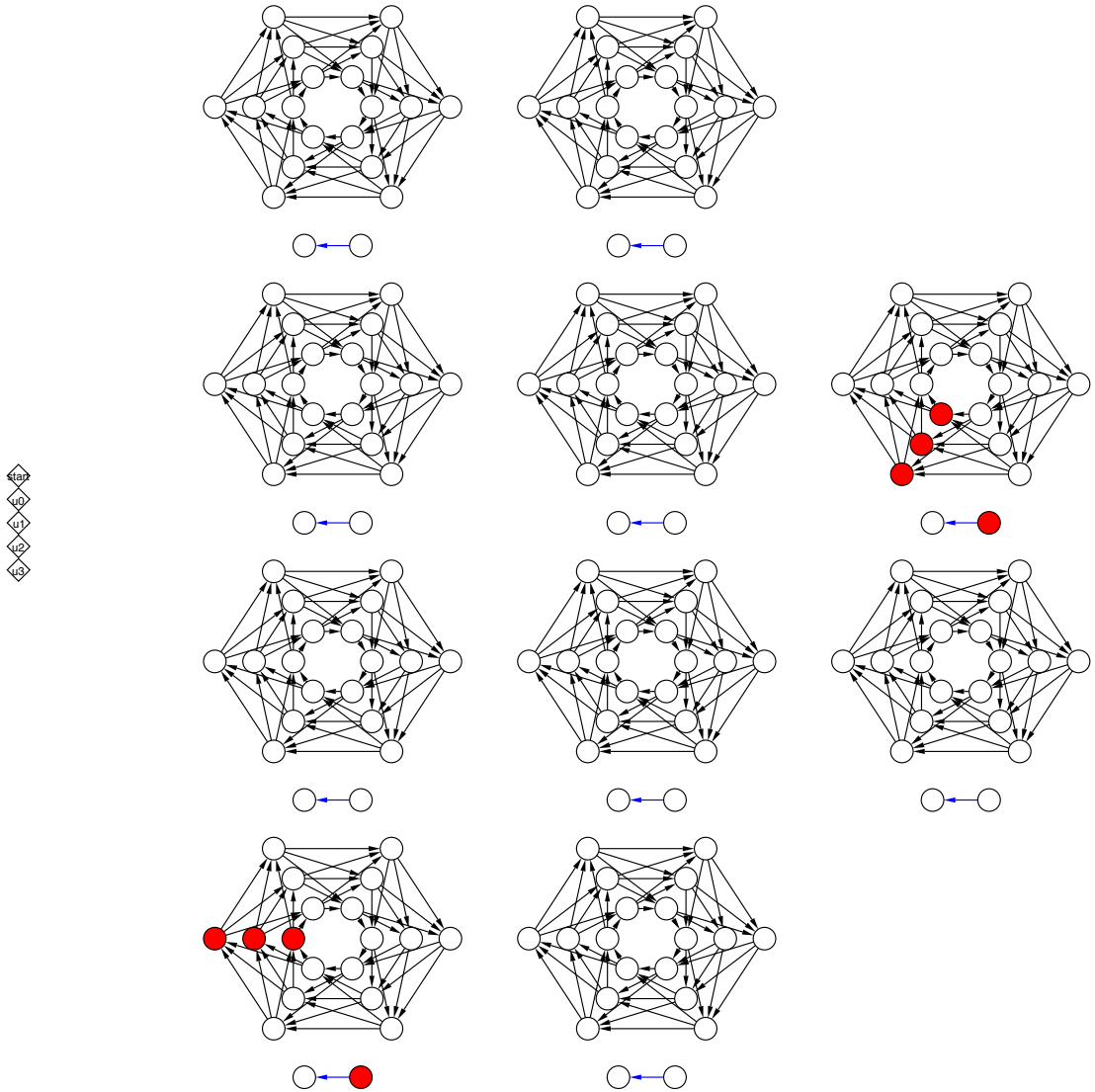
Simulation



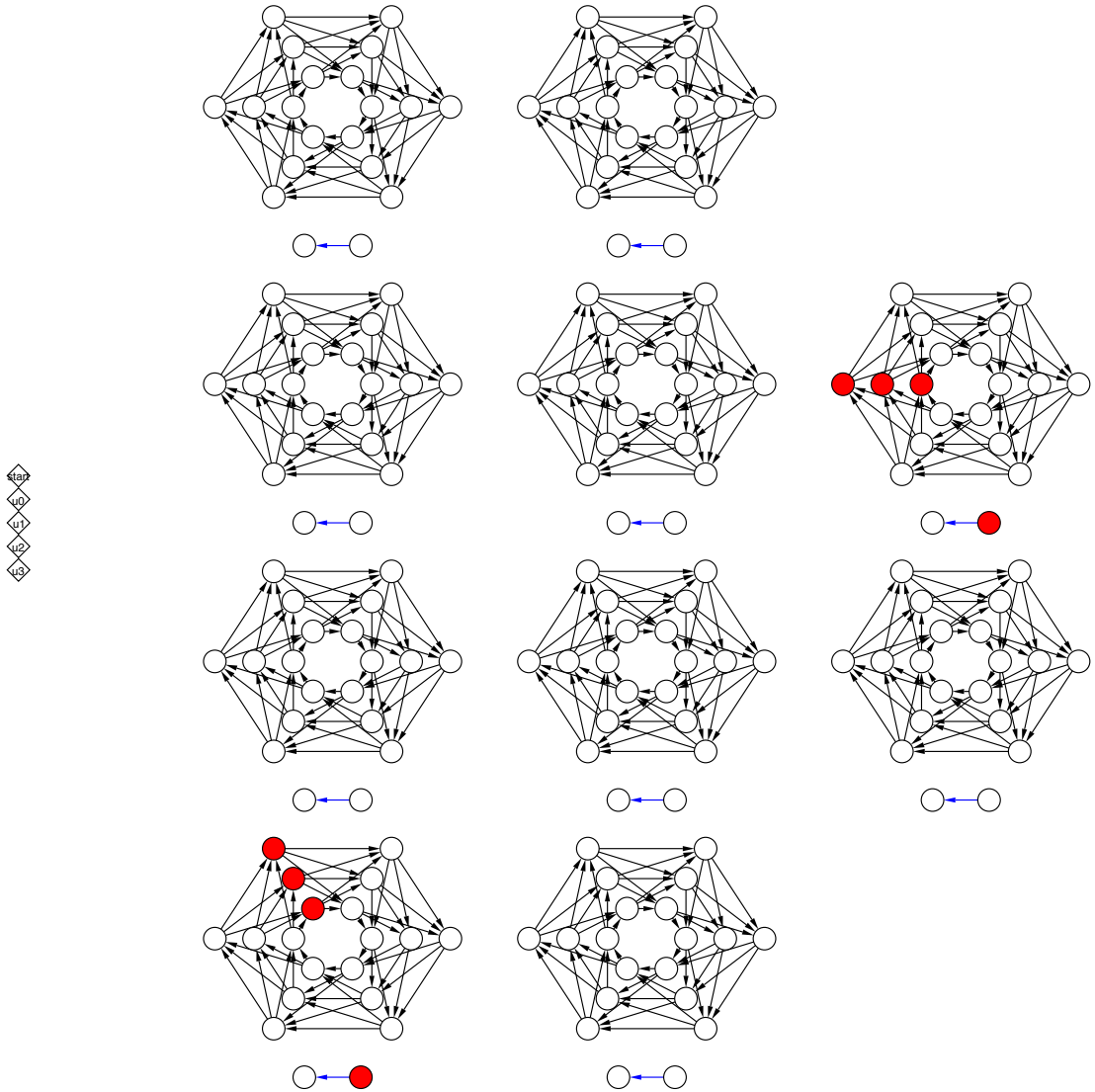
Simulation



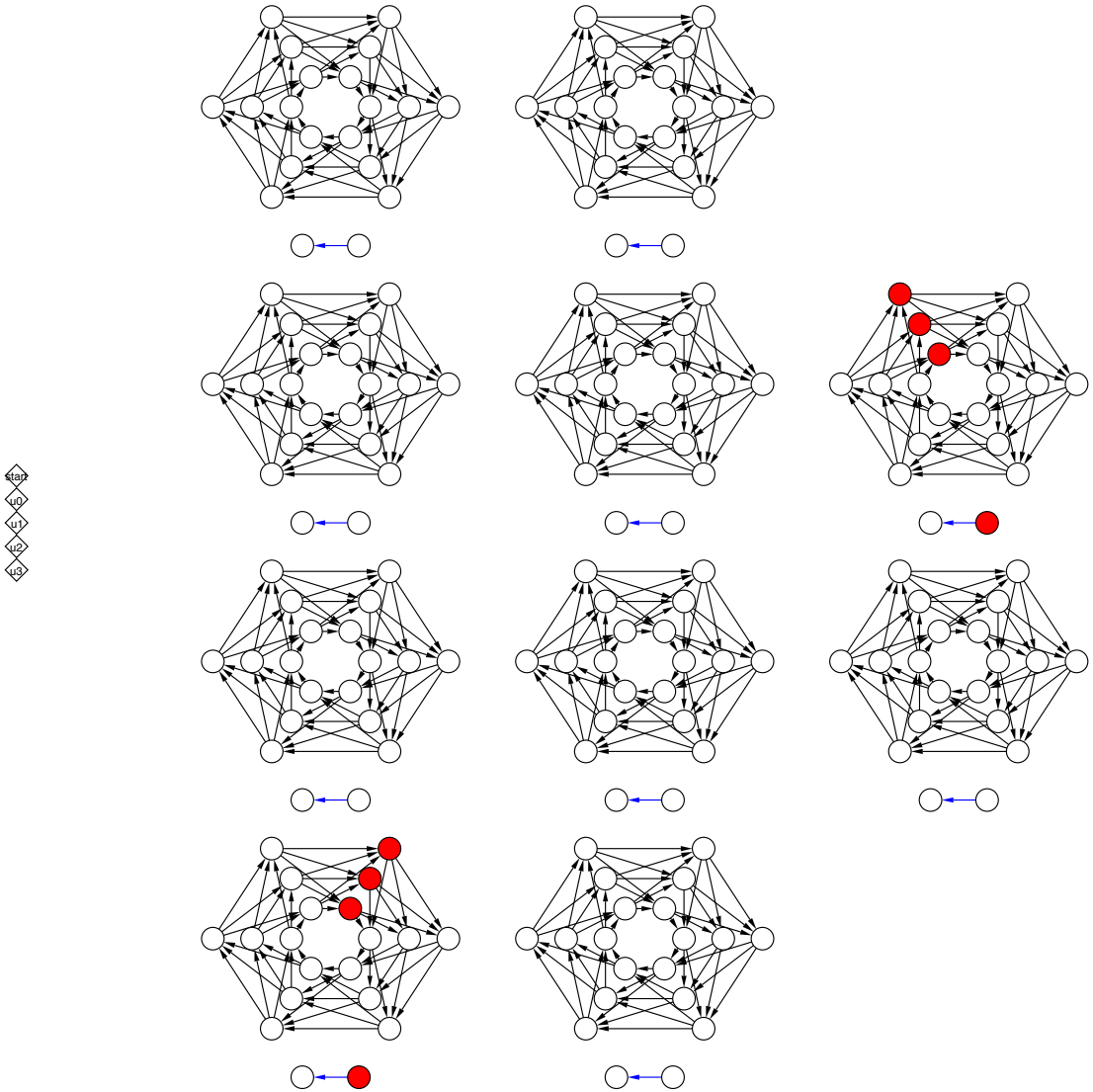
Simulation



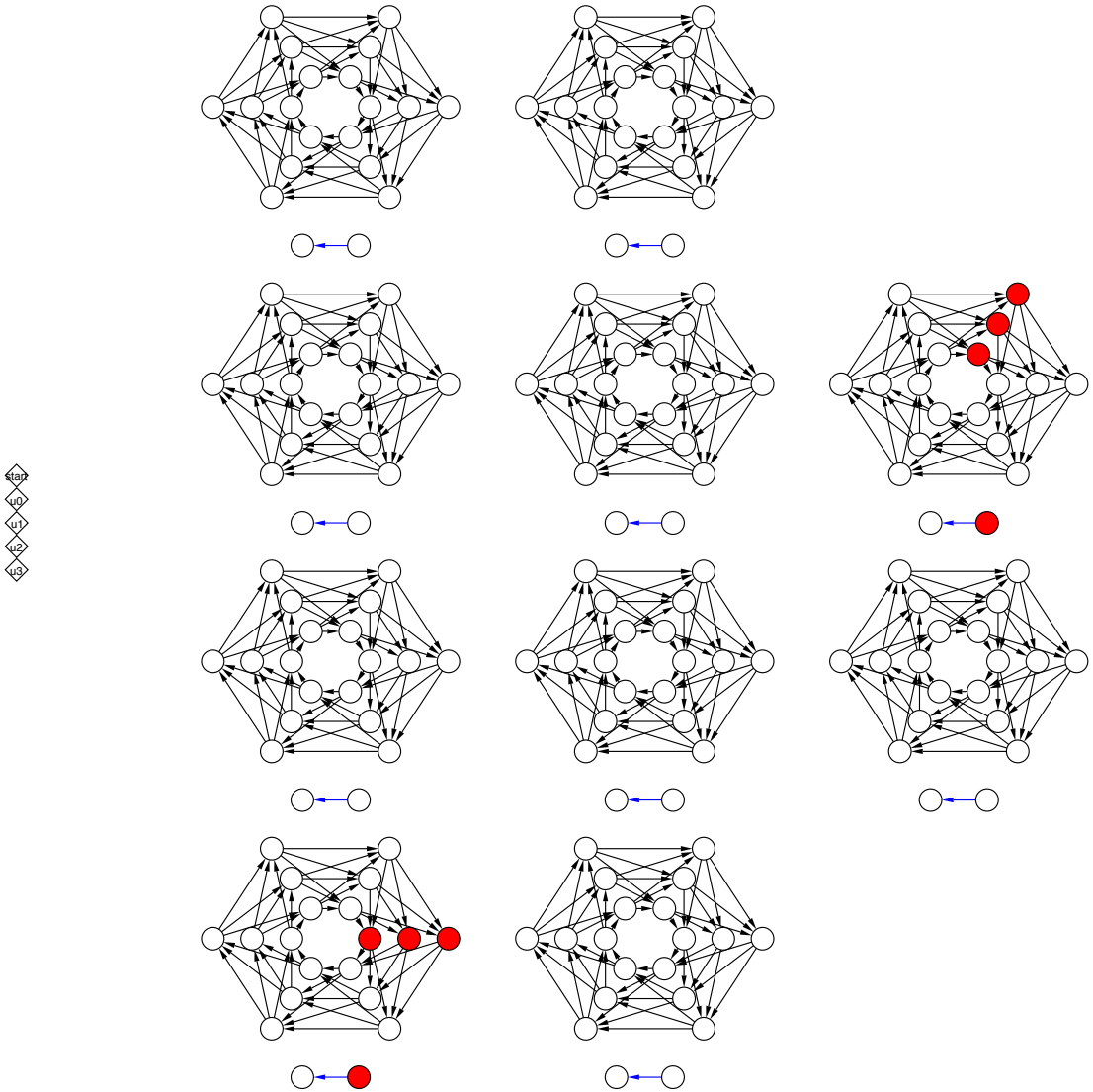
Simulation



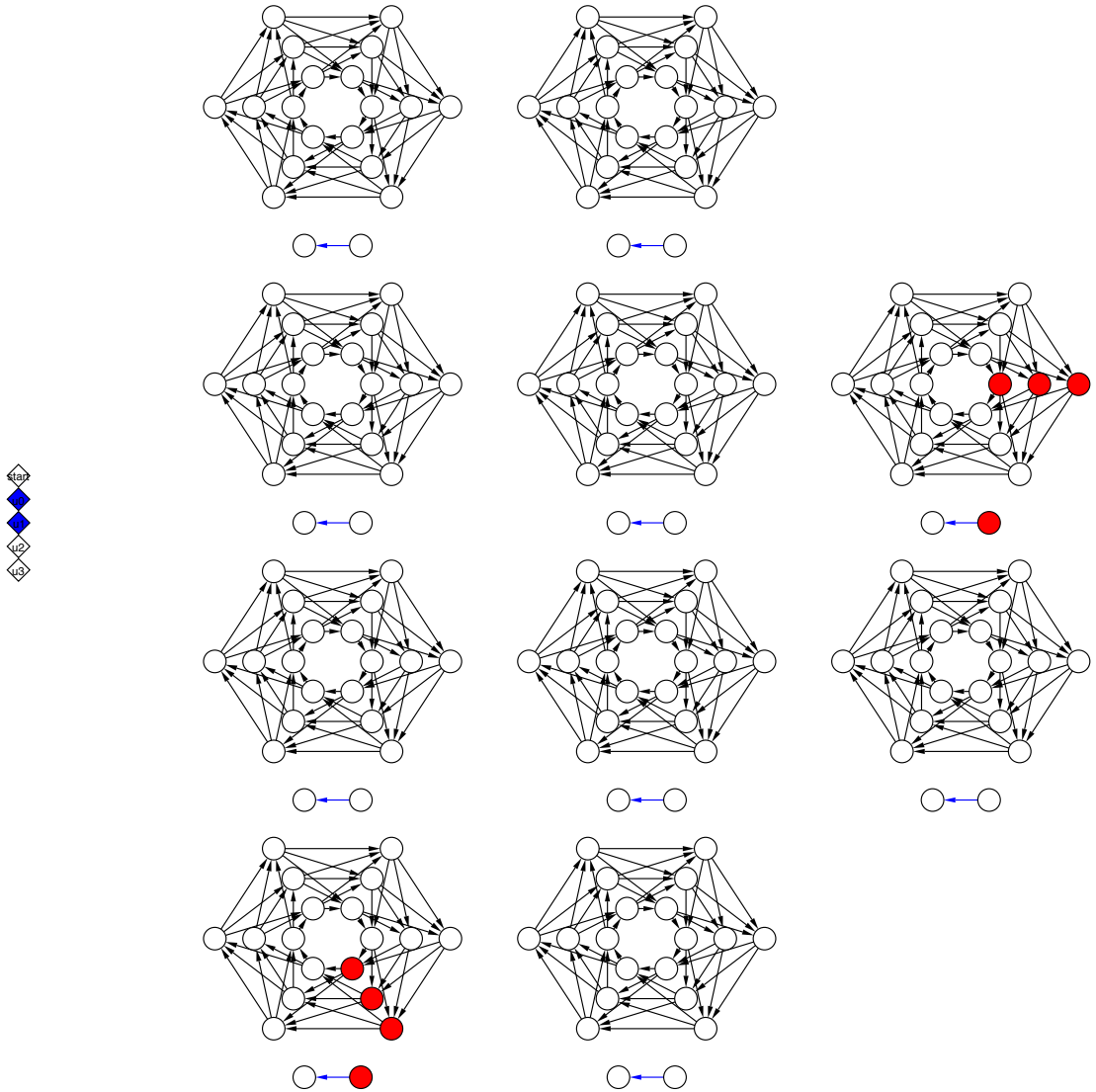
Simulation



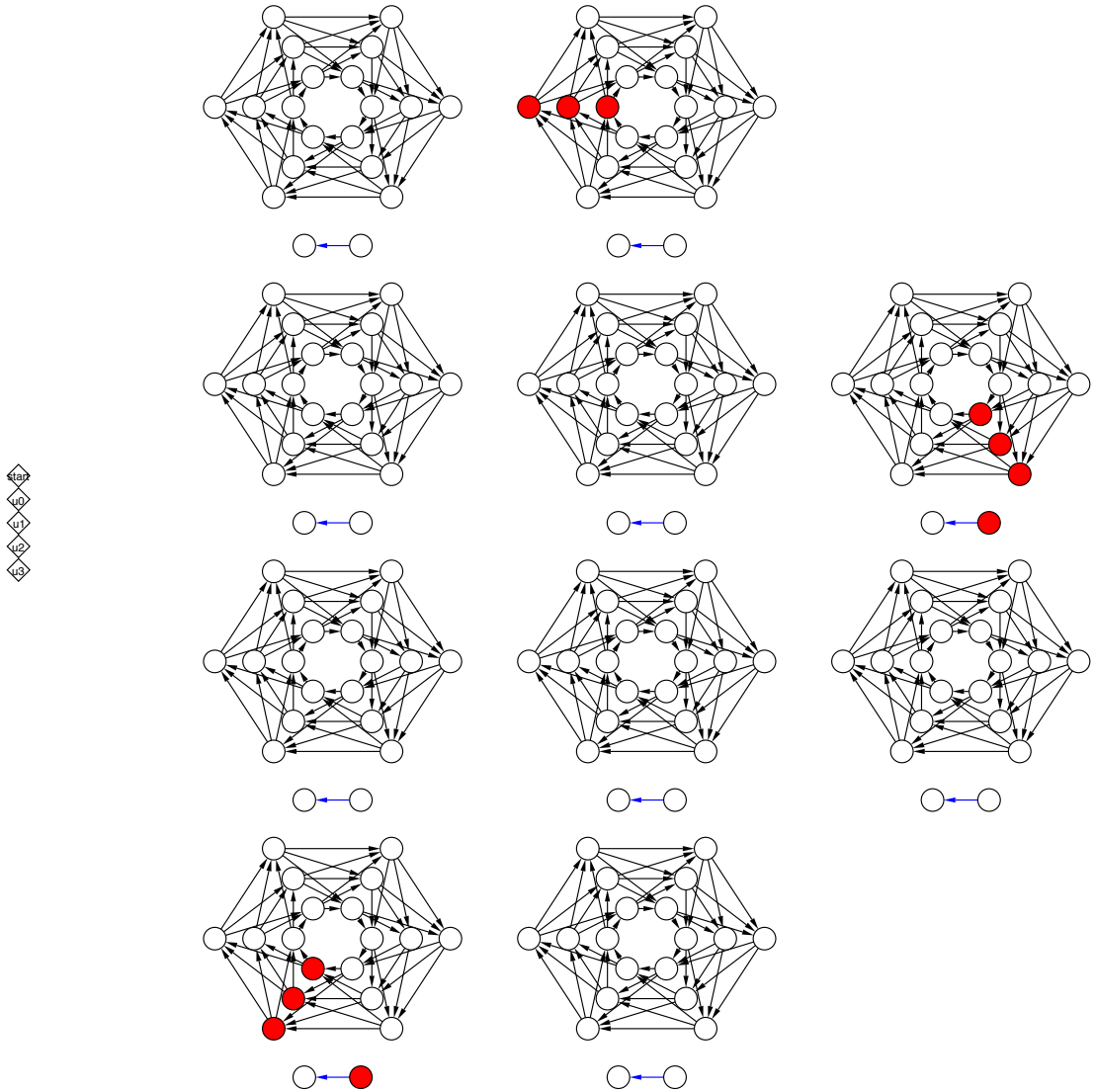
Simulation



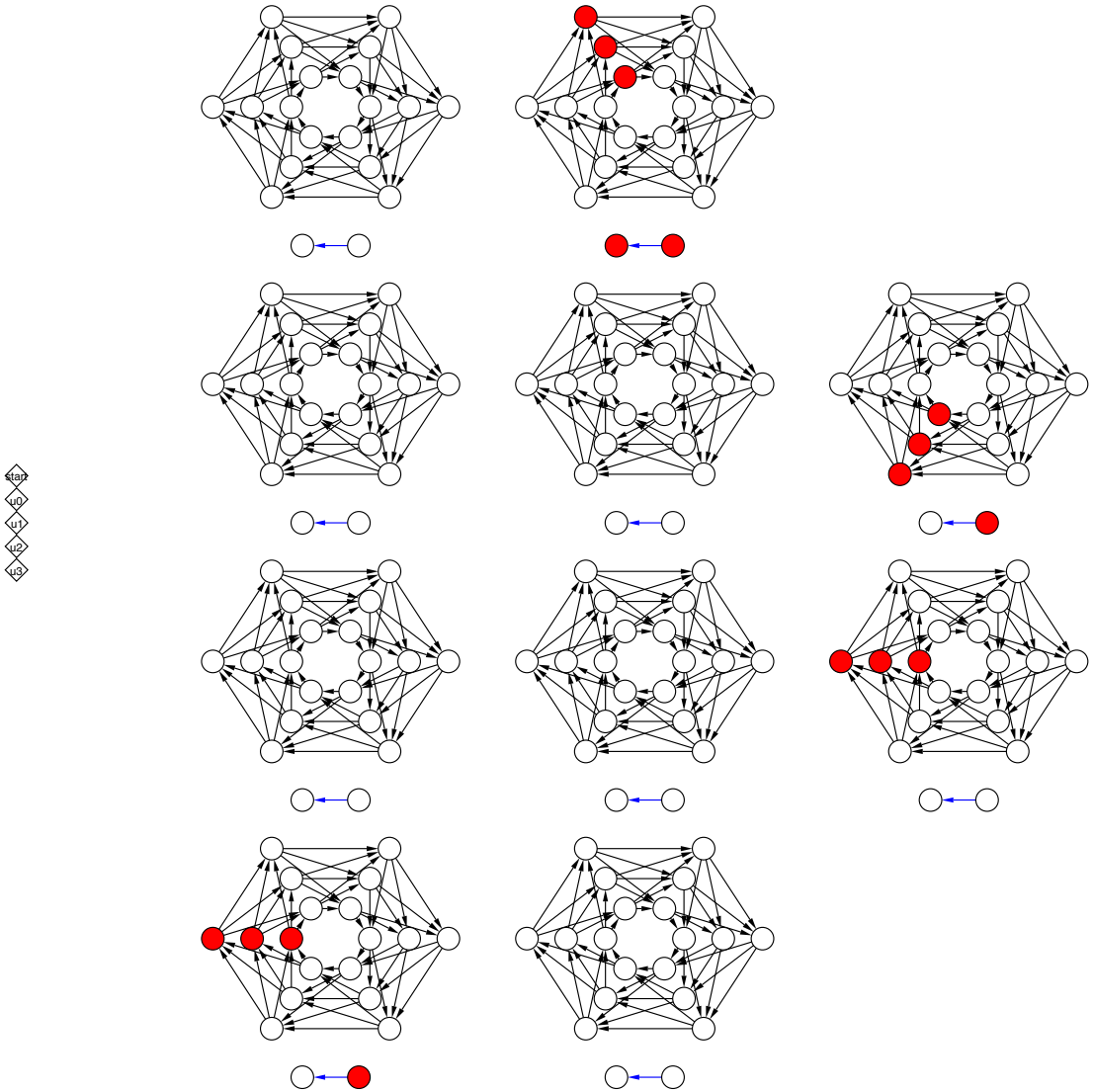
Simulation



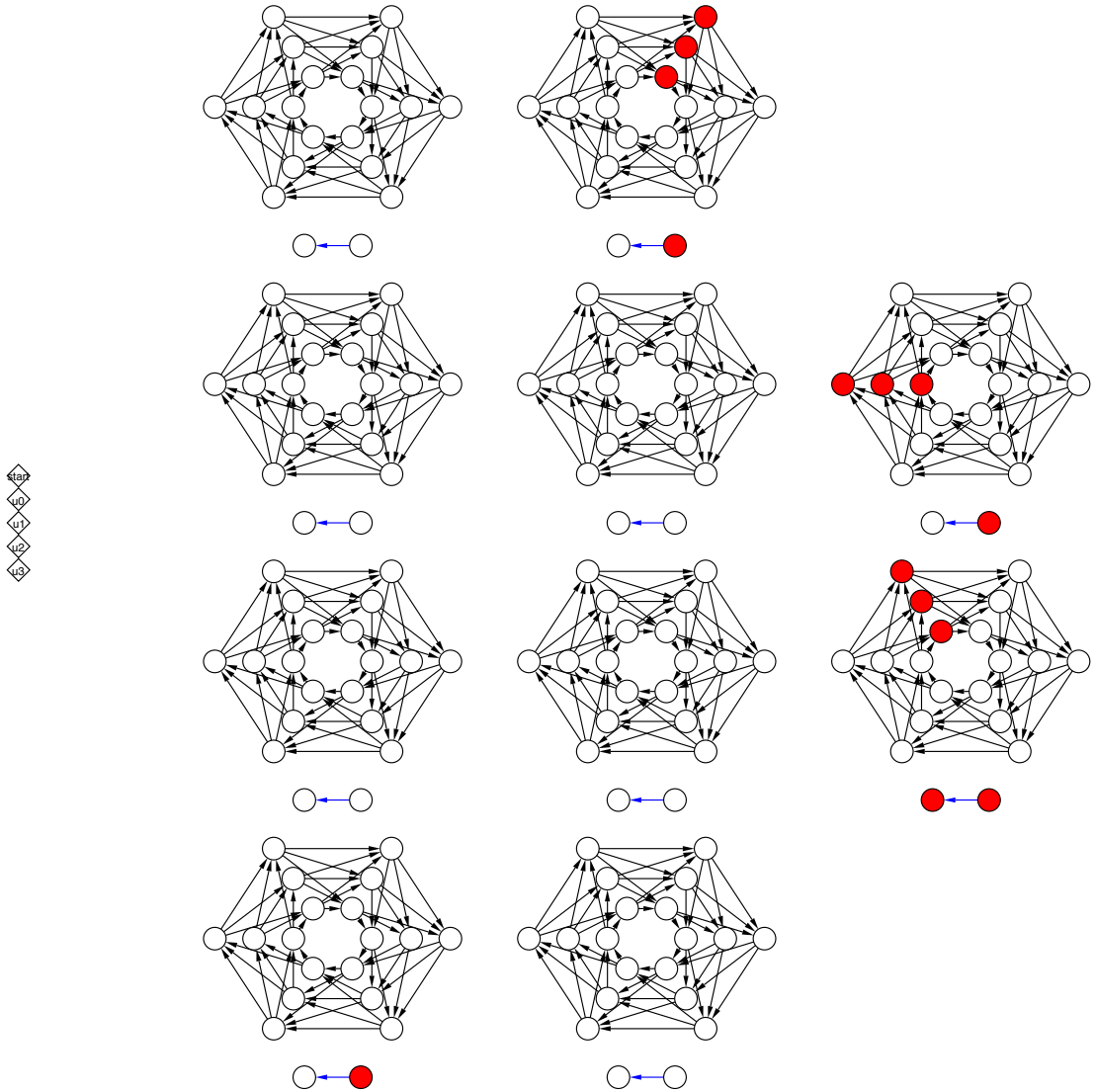
Simulation



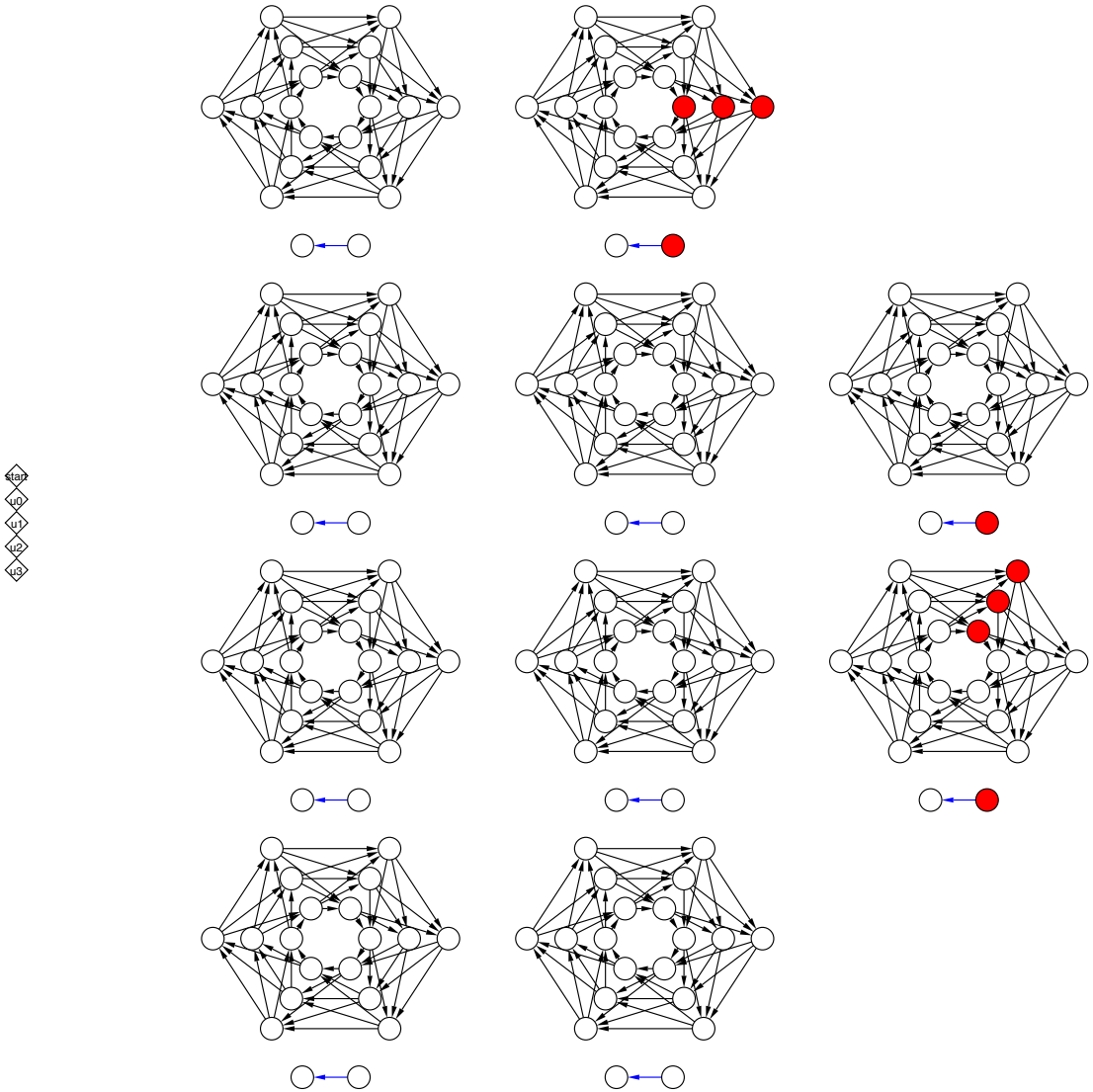
Simulation



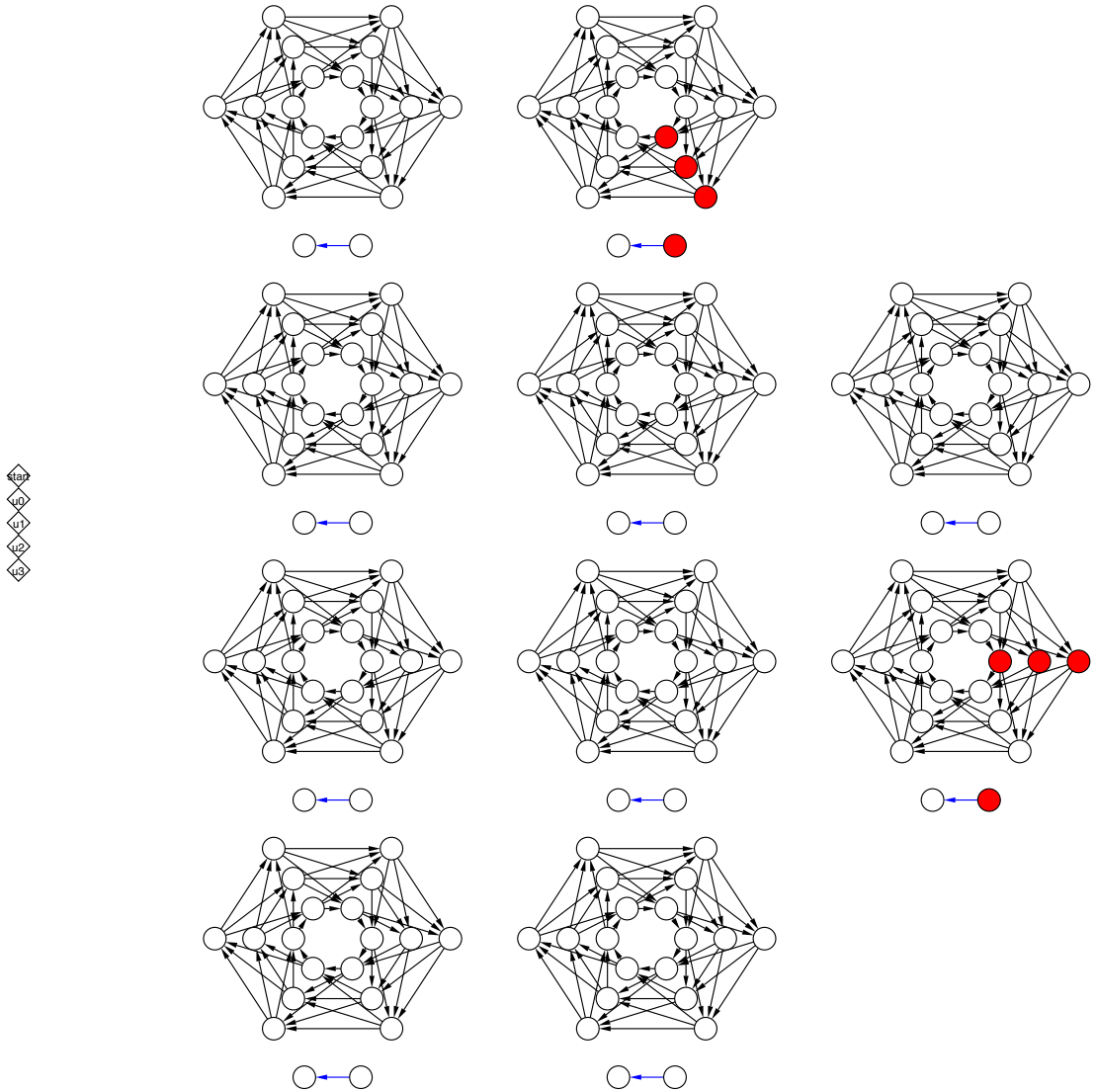
Simulation



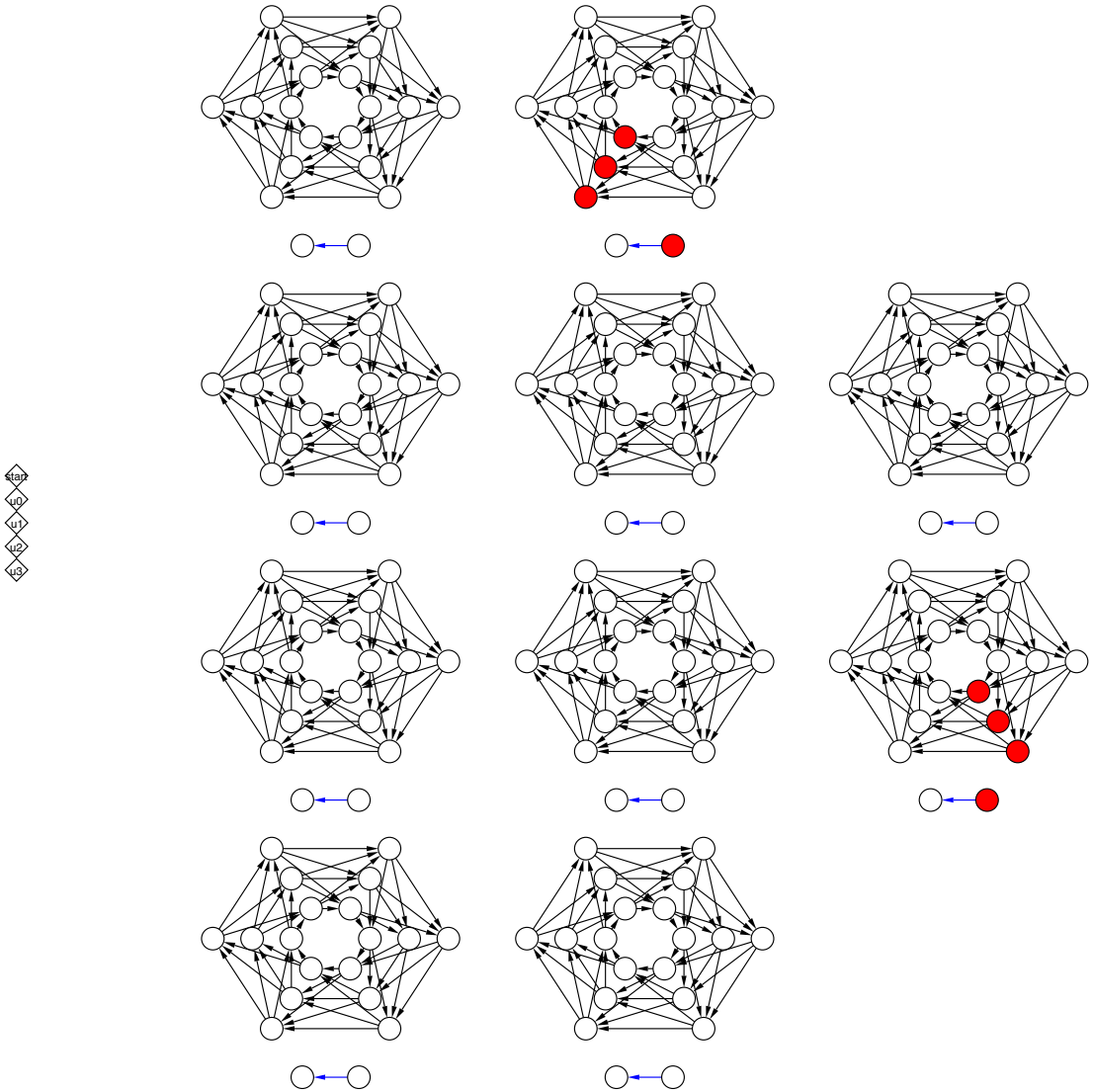
Simulation



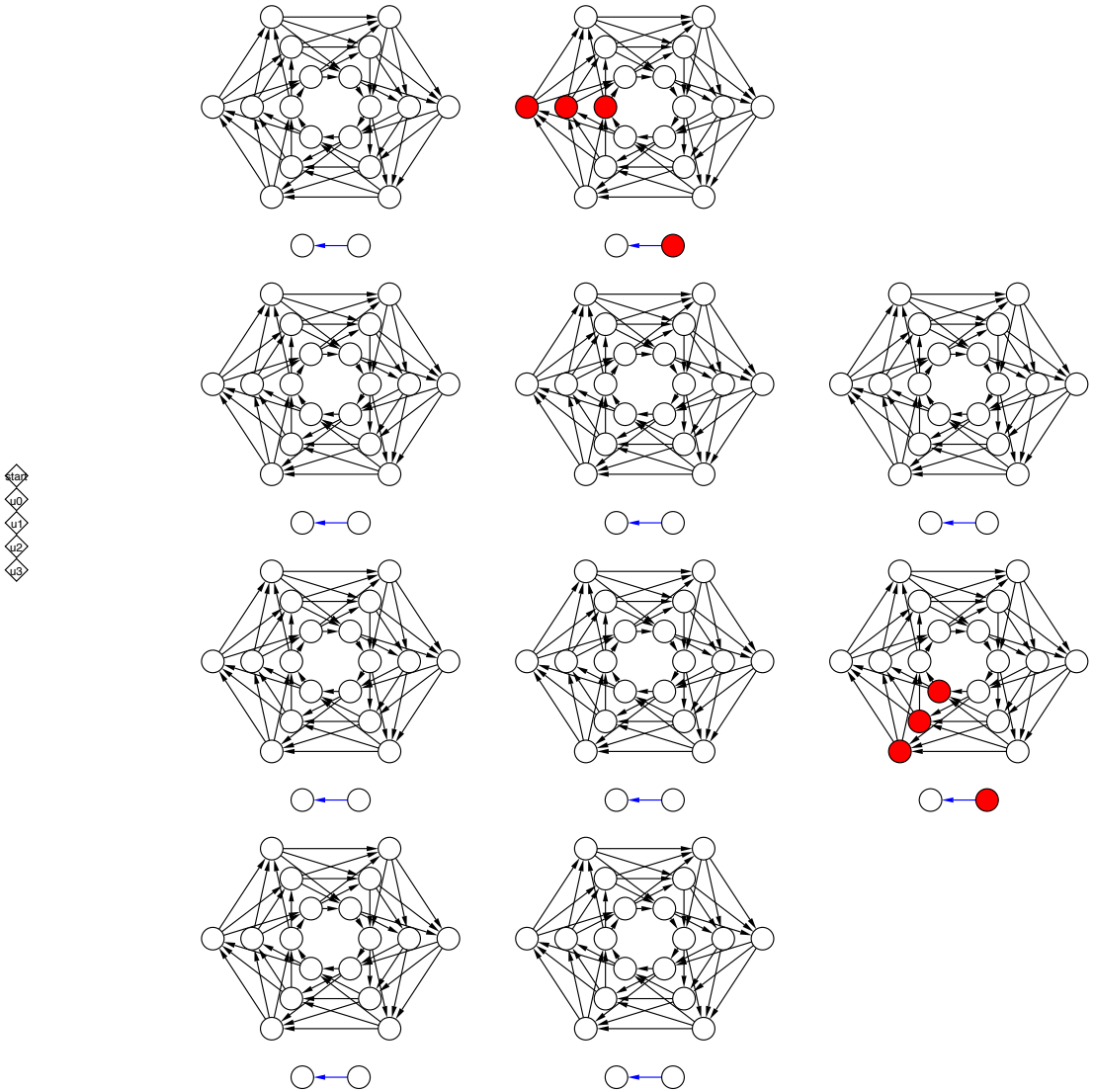
Simulation



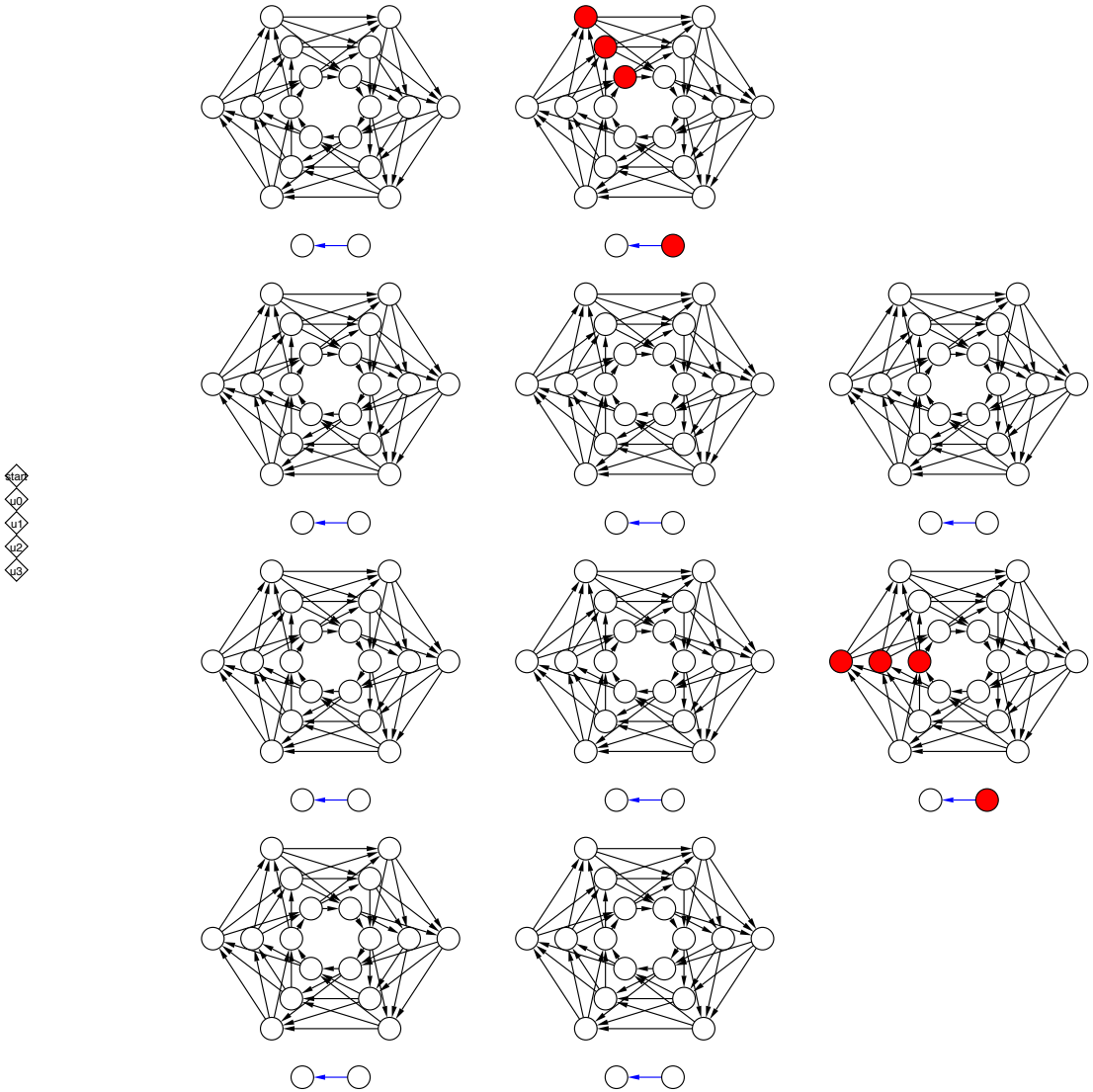
Simulation



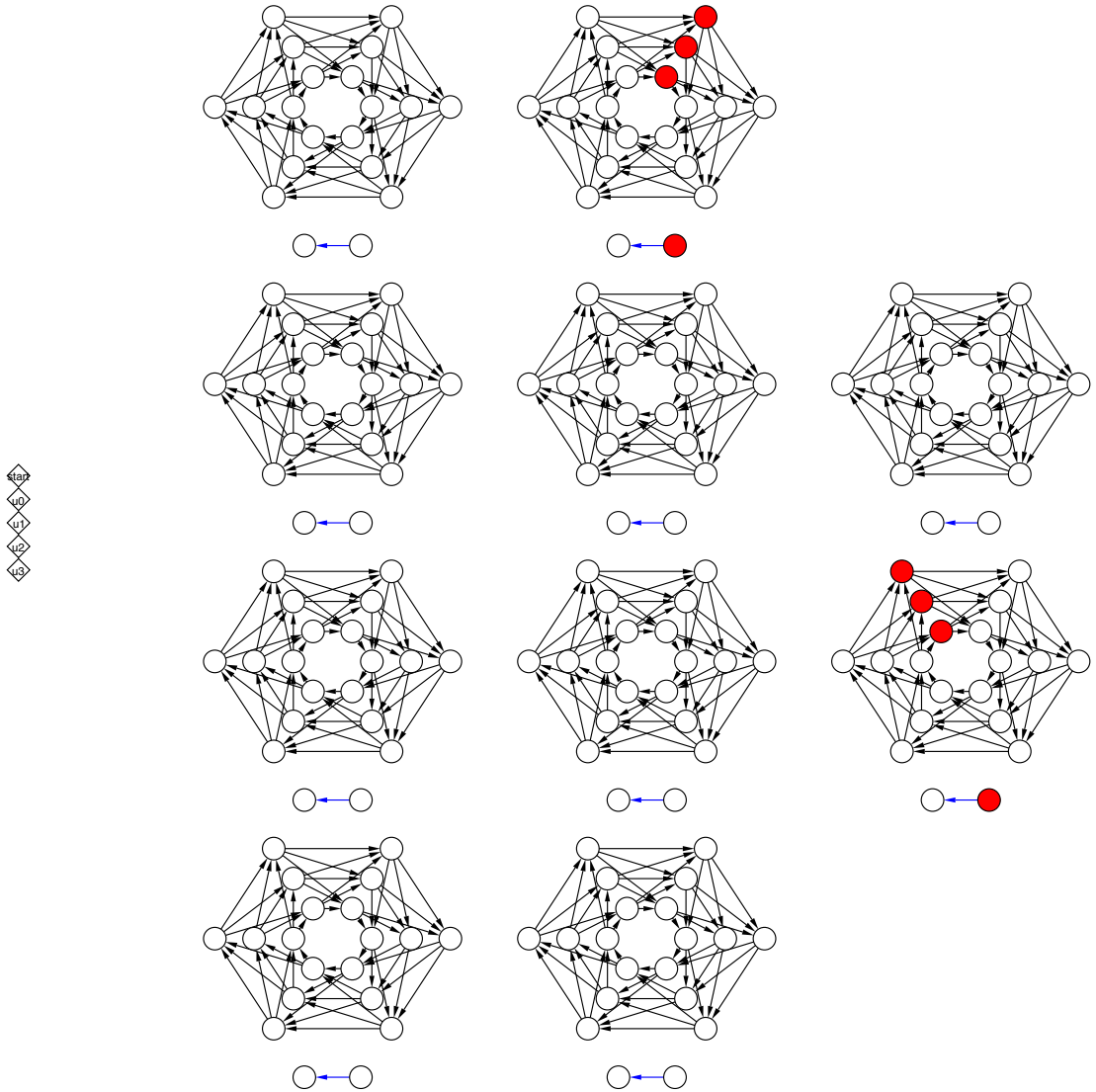
Simulation



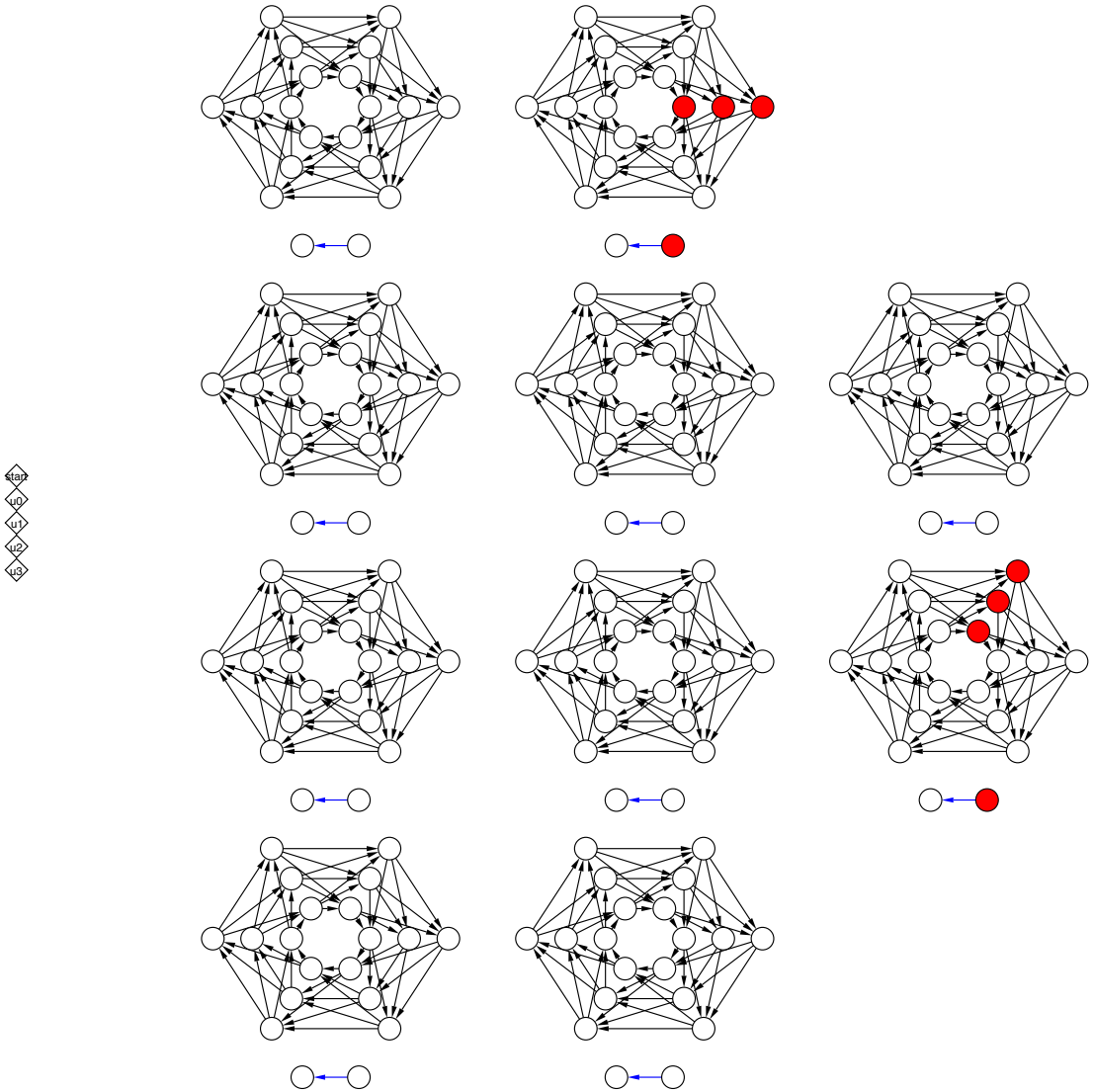
Simulation



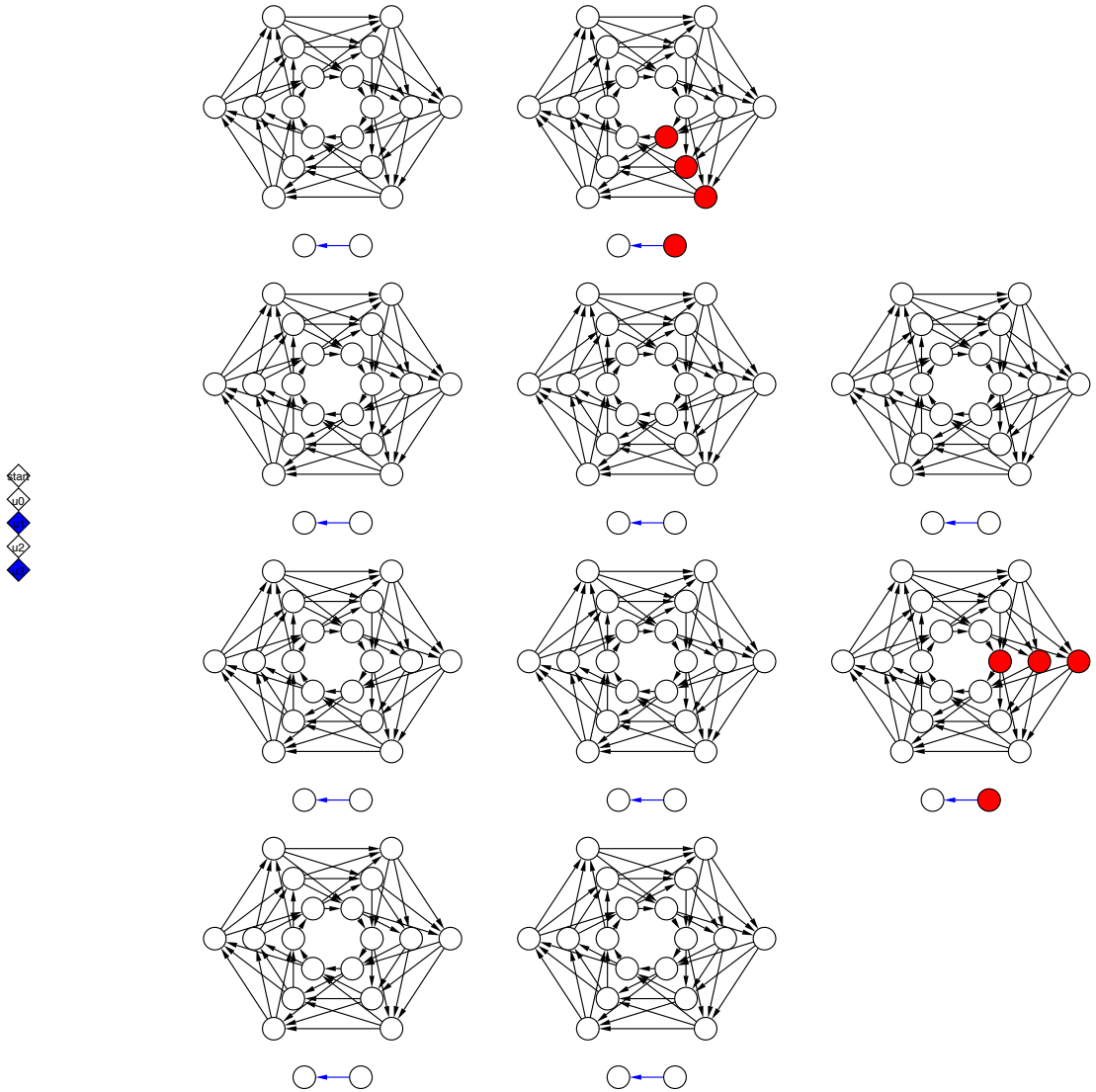
Simulation



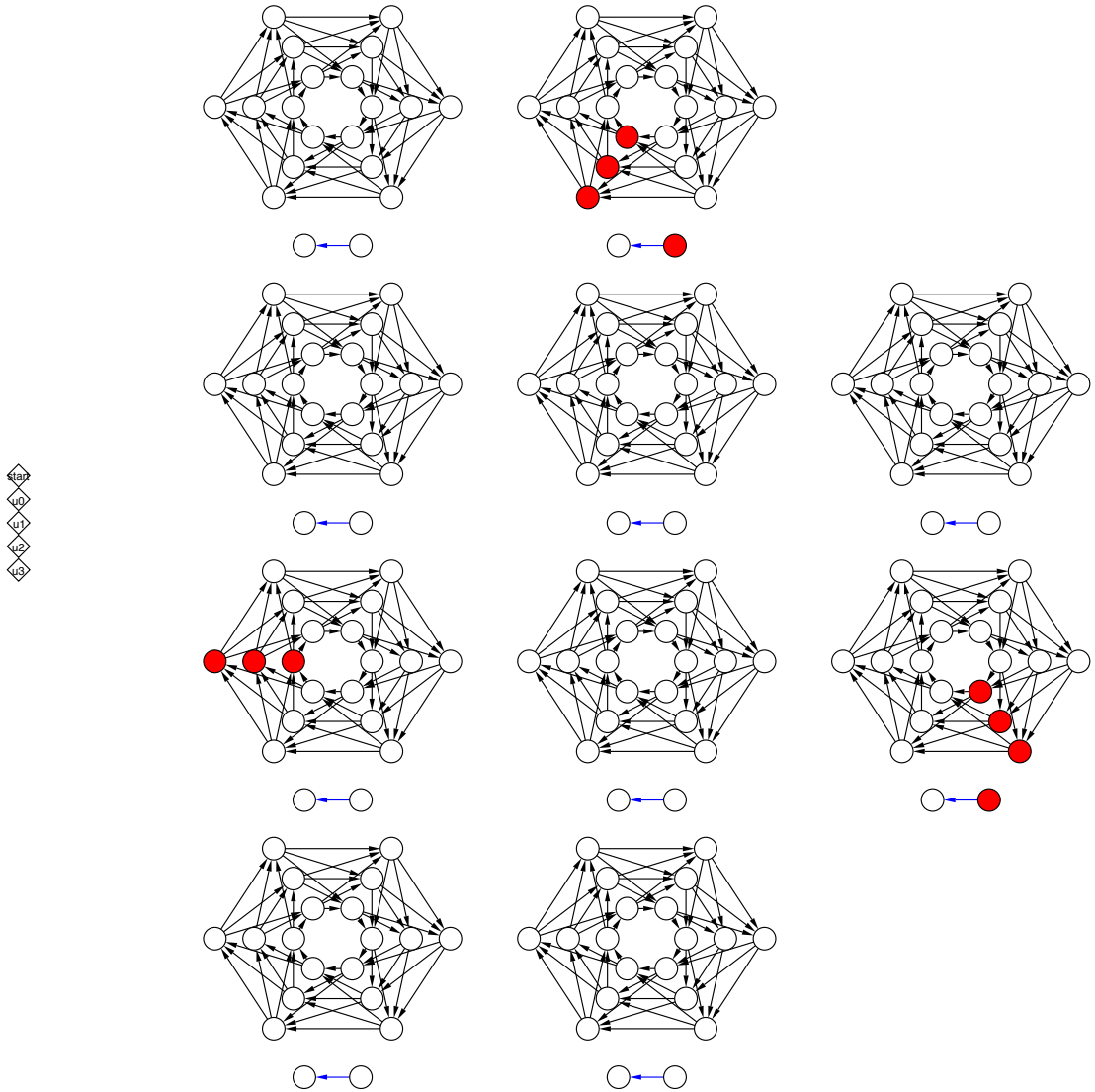
Simulation



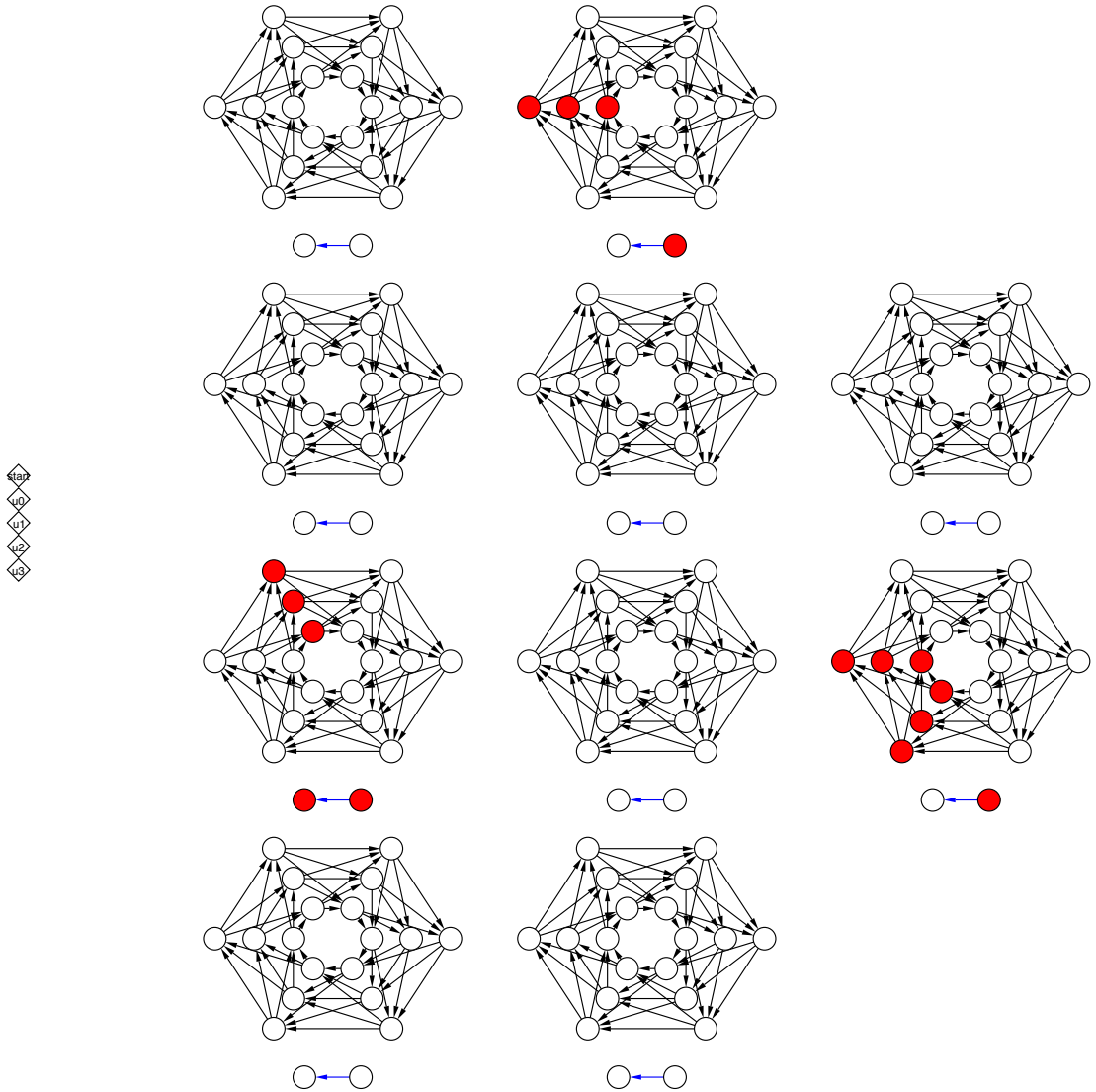
Simulation



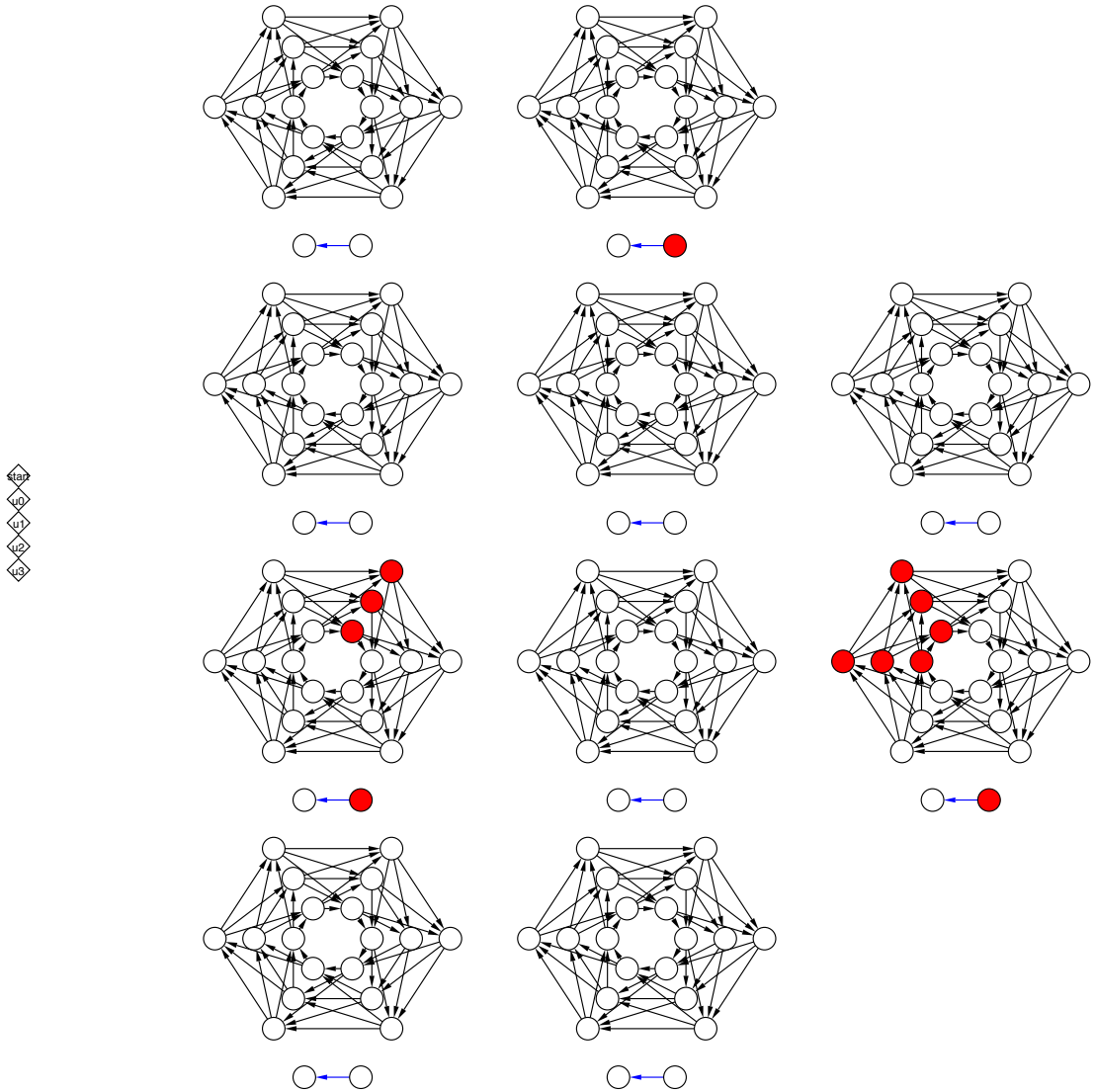
Simulation



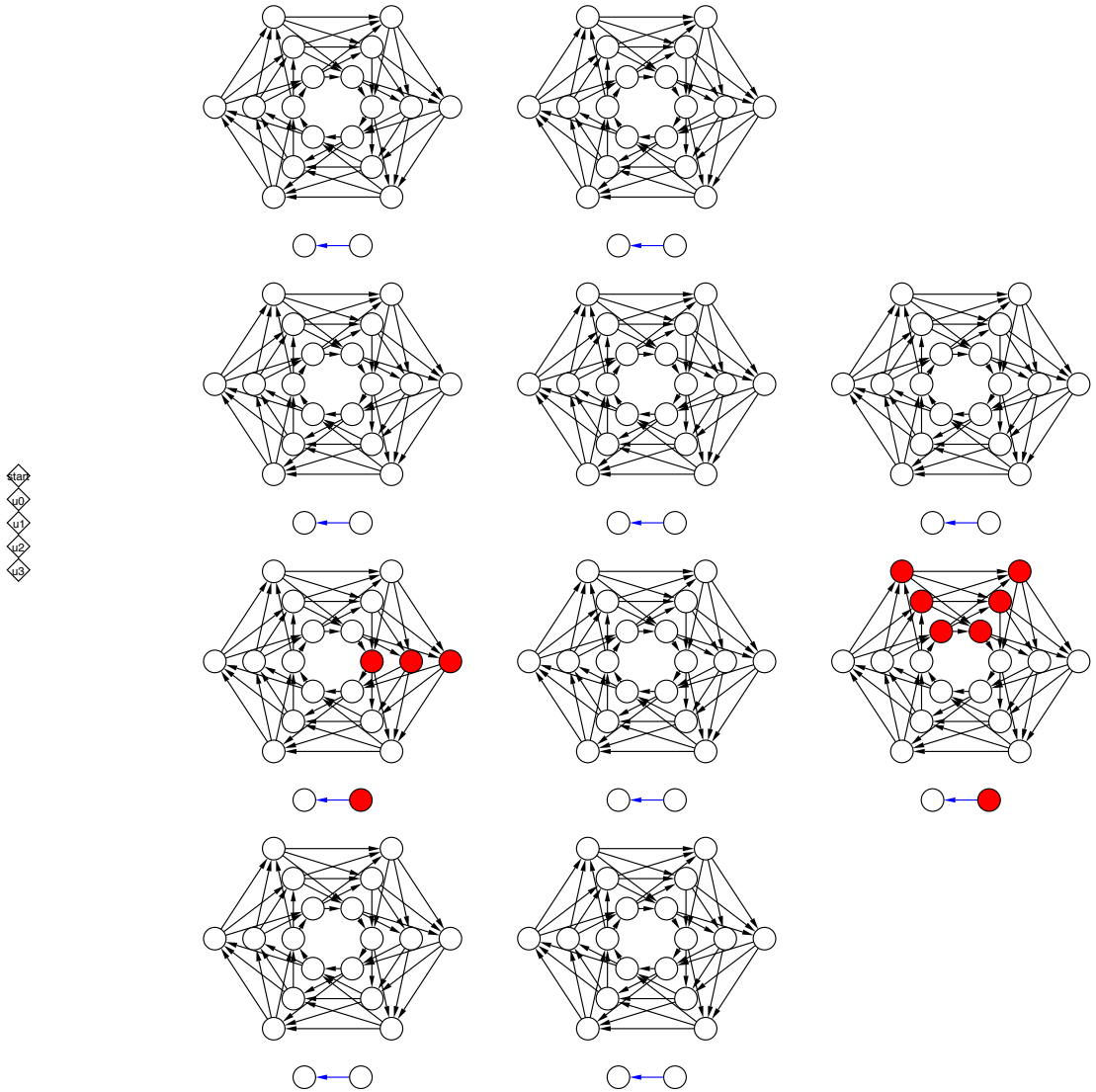
Simulation



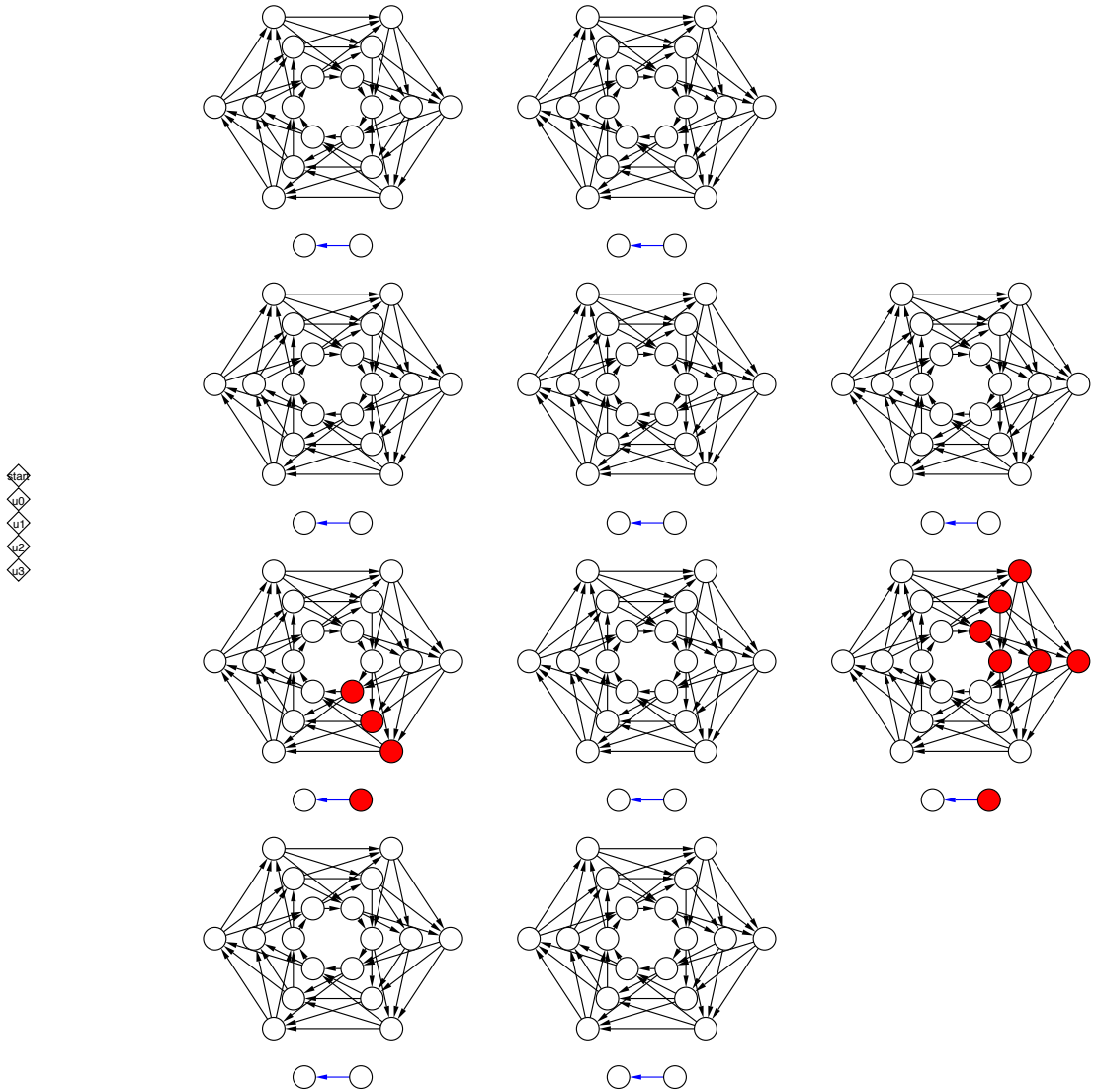
Simulation



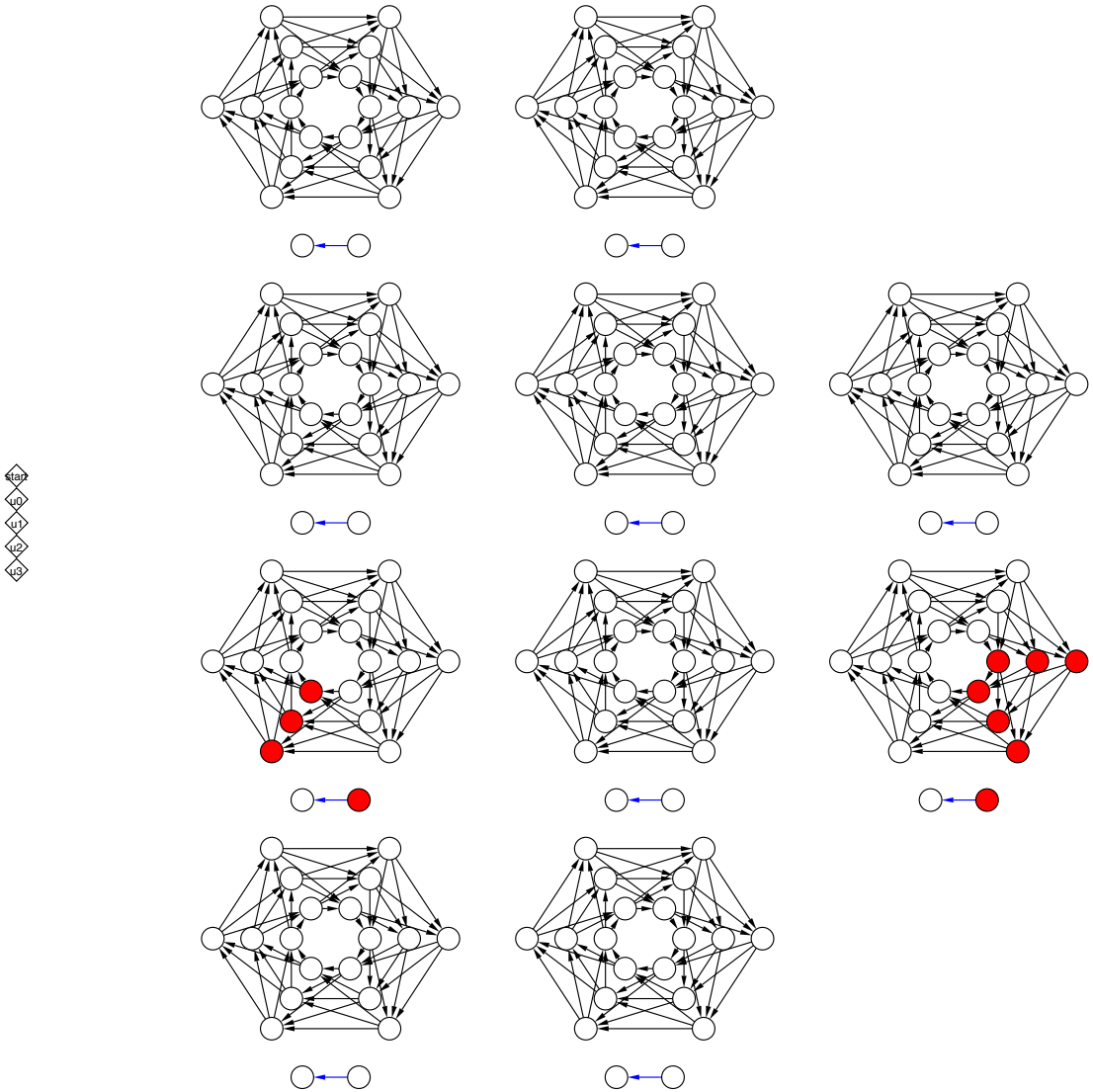
Simulation



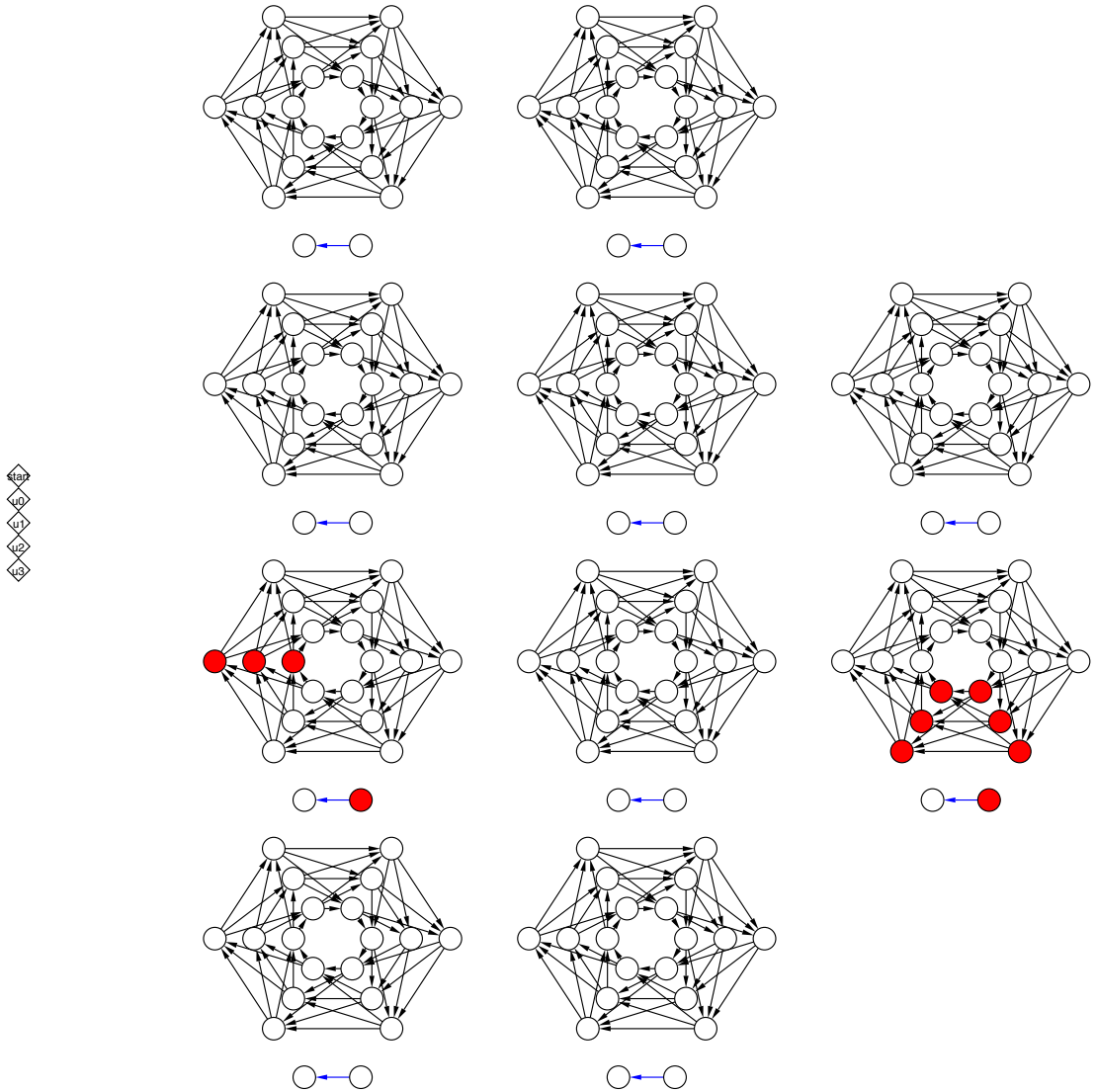
Simulation



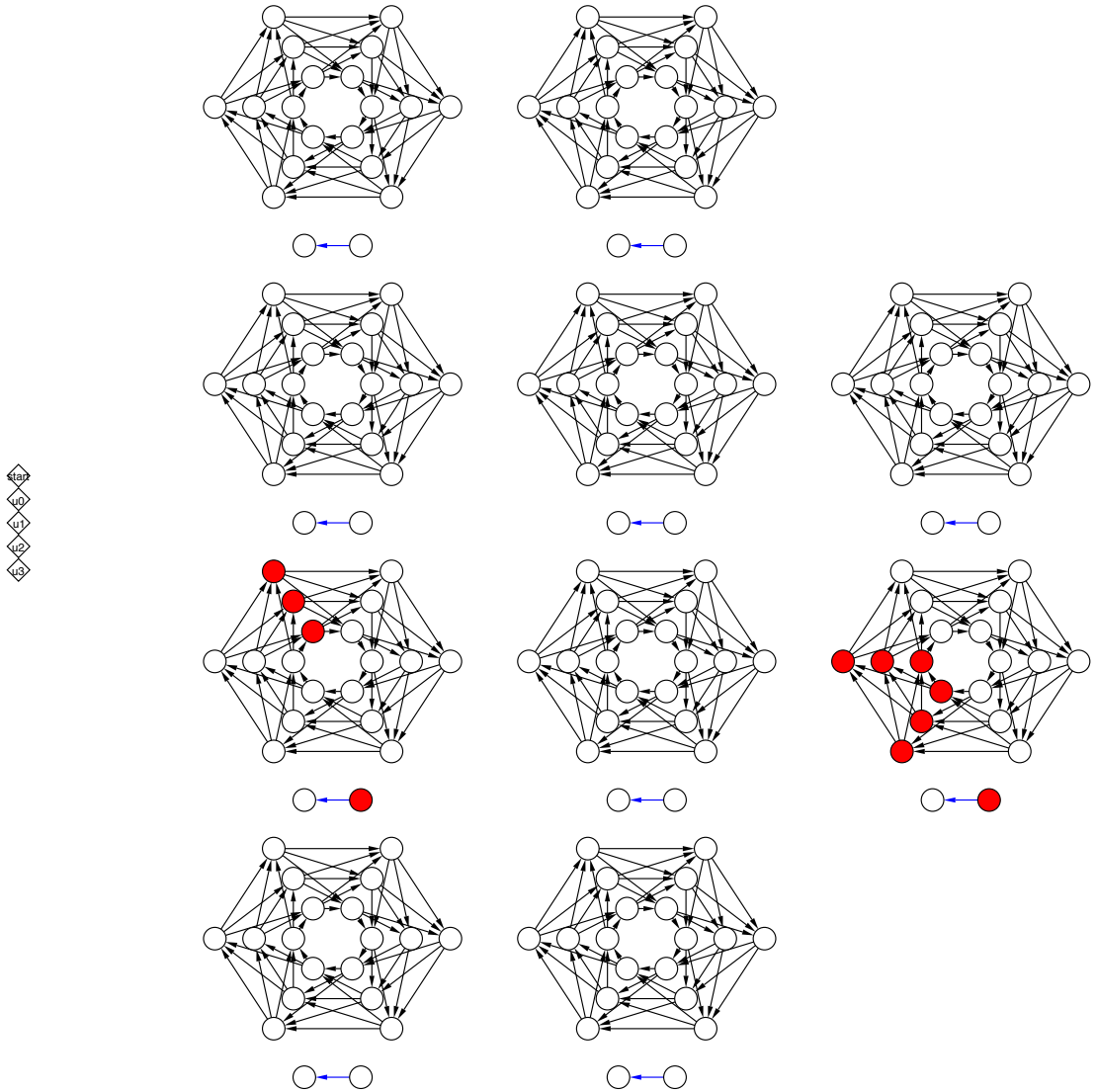
Simulation



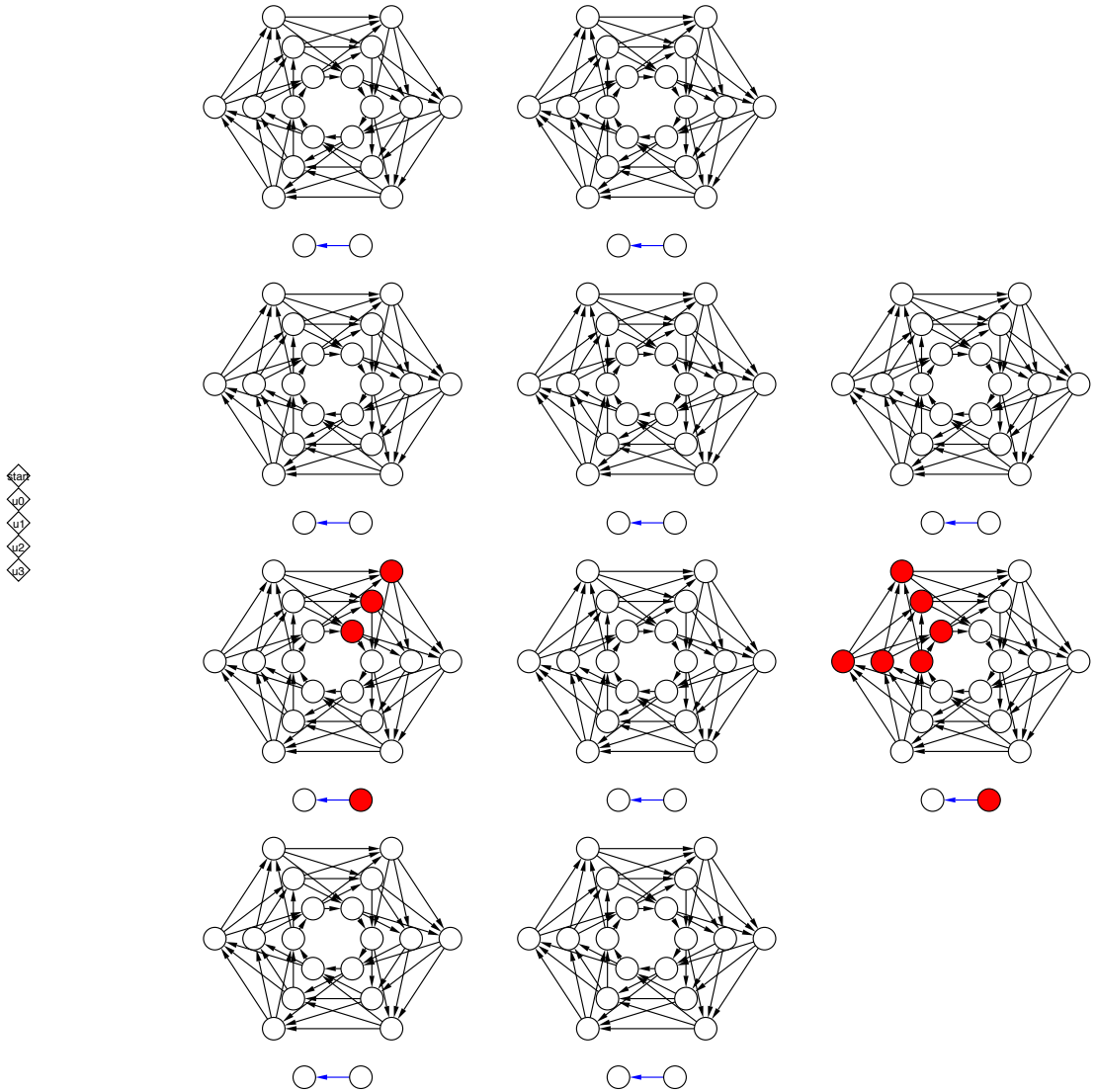
Simulation



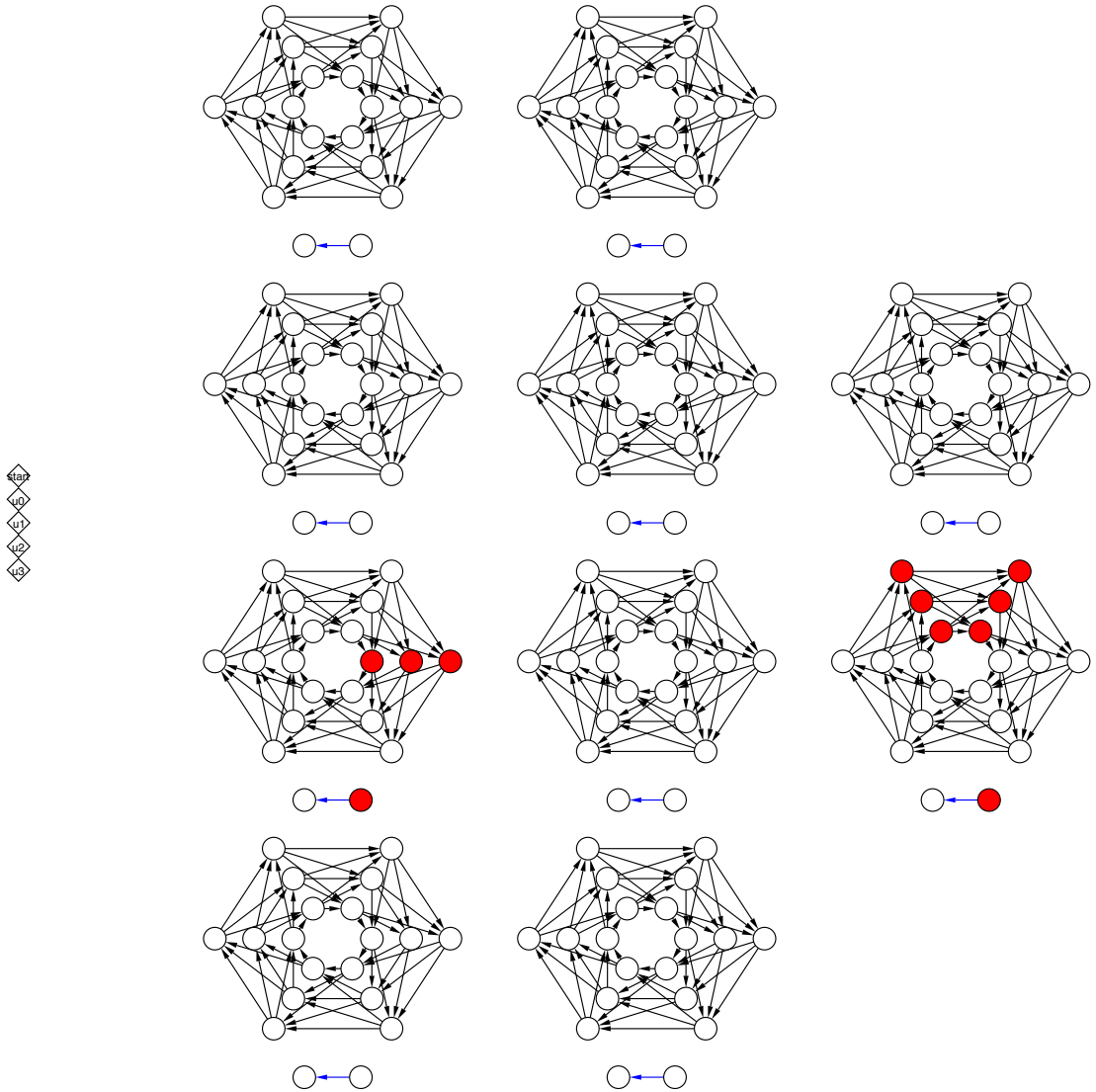
Simulation



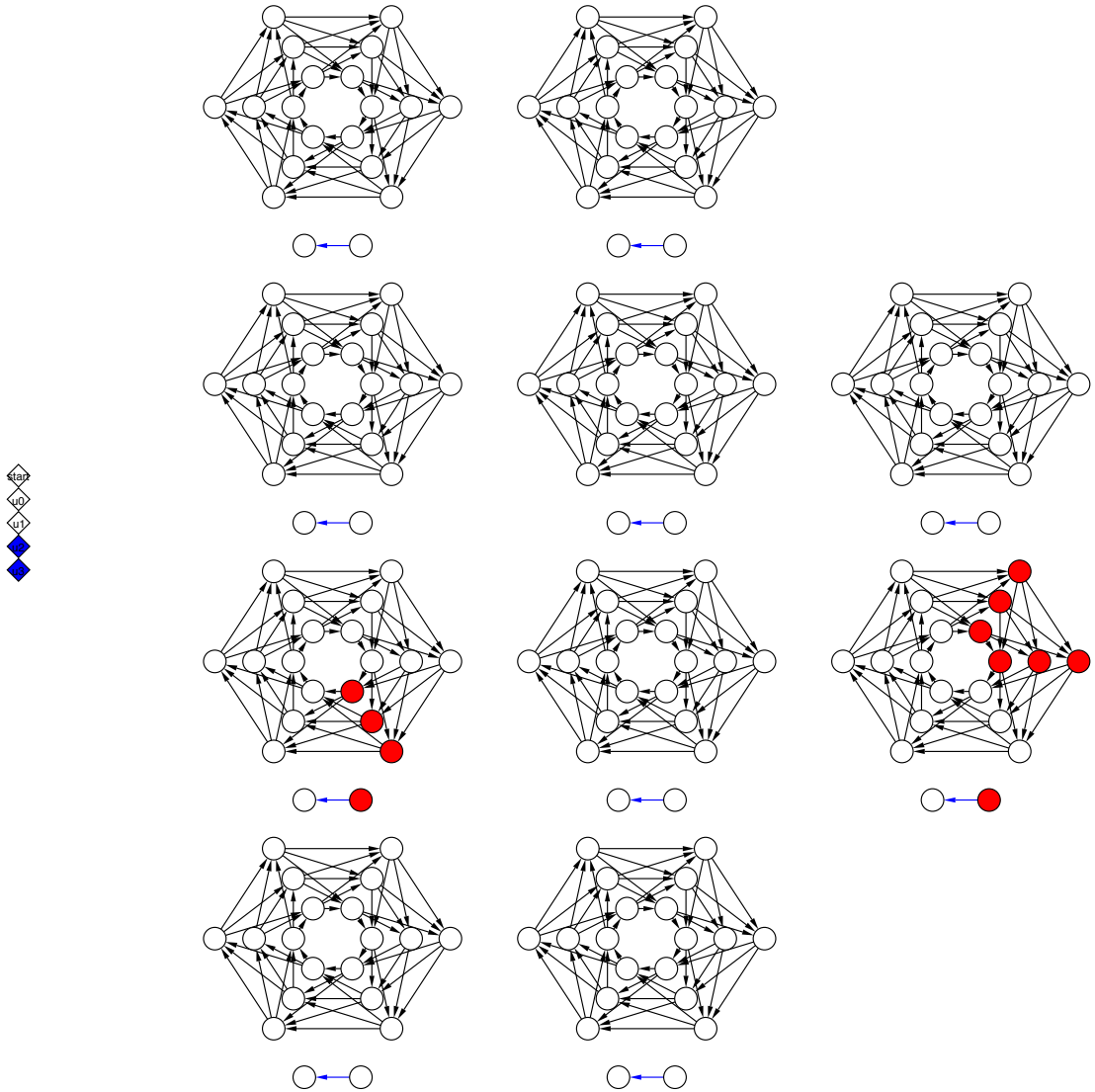
Simulation



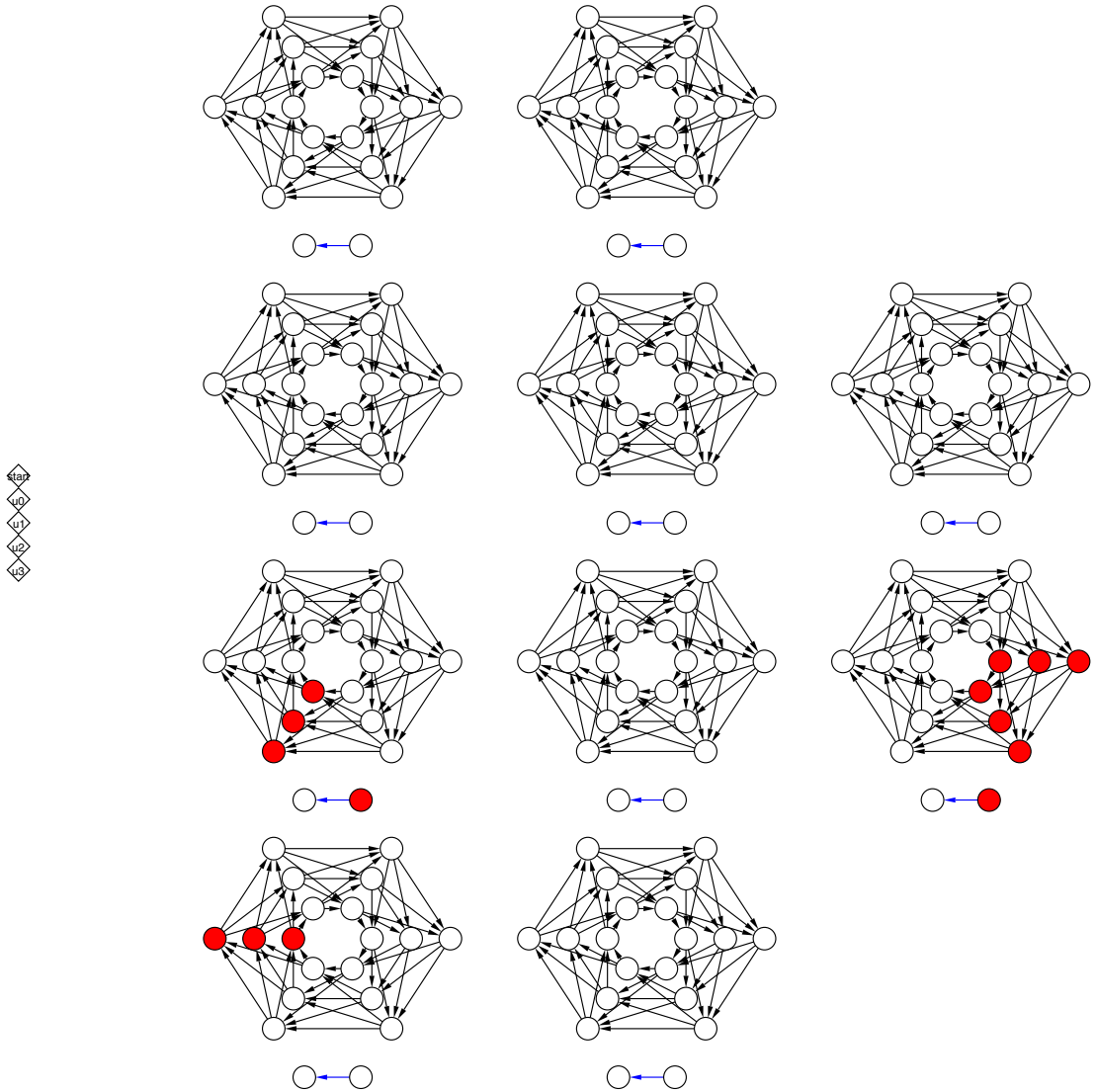
Simulation



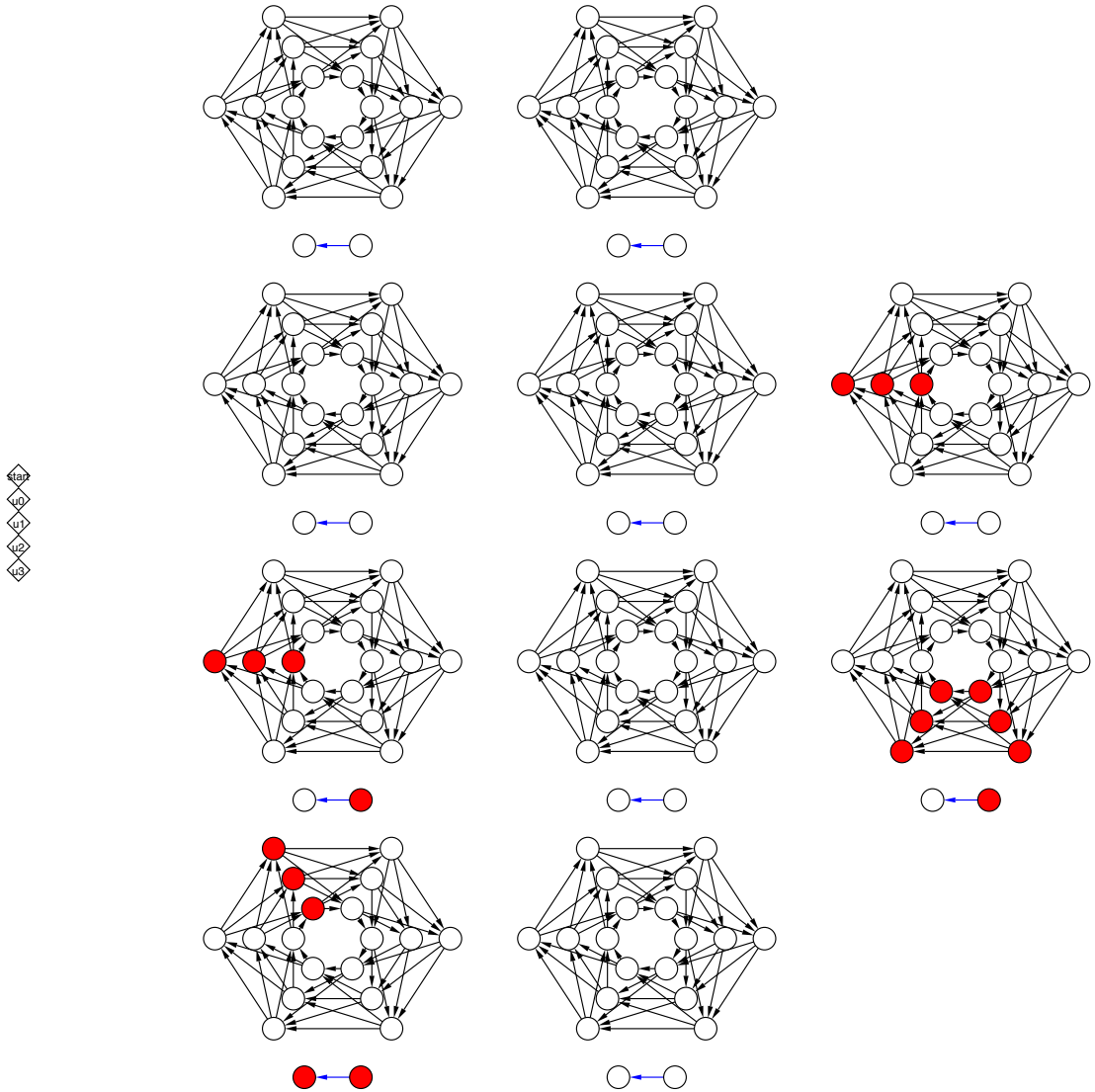
Simulation



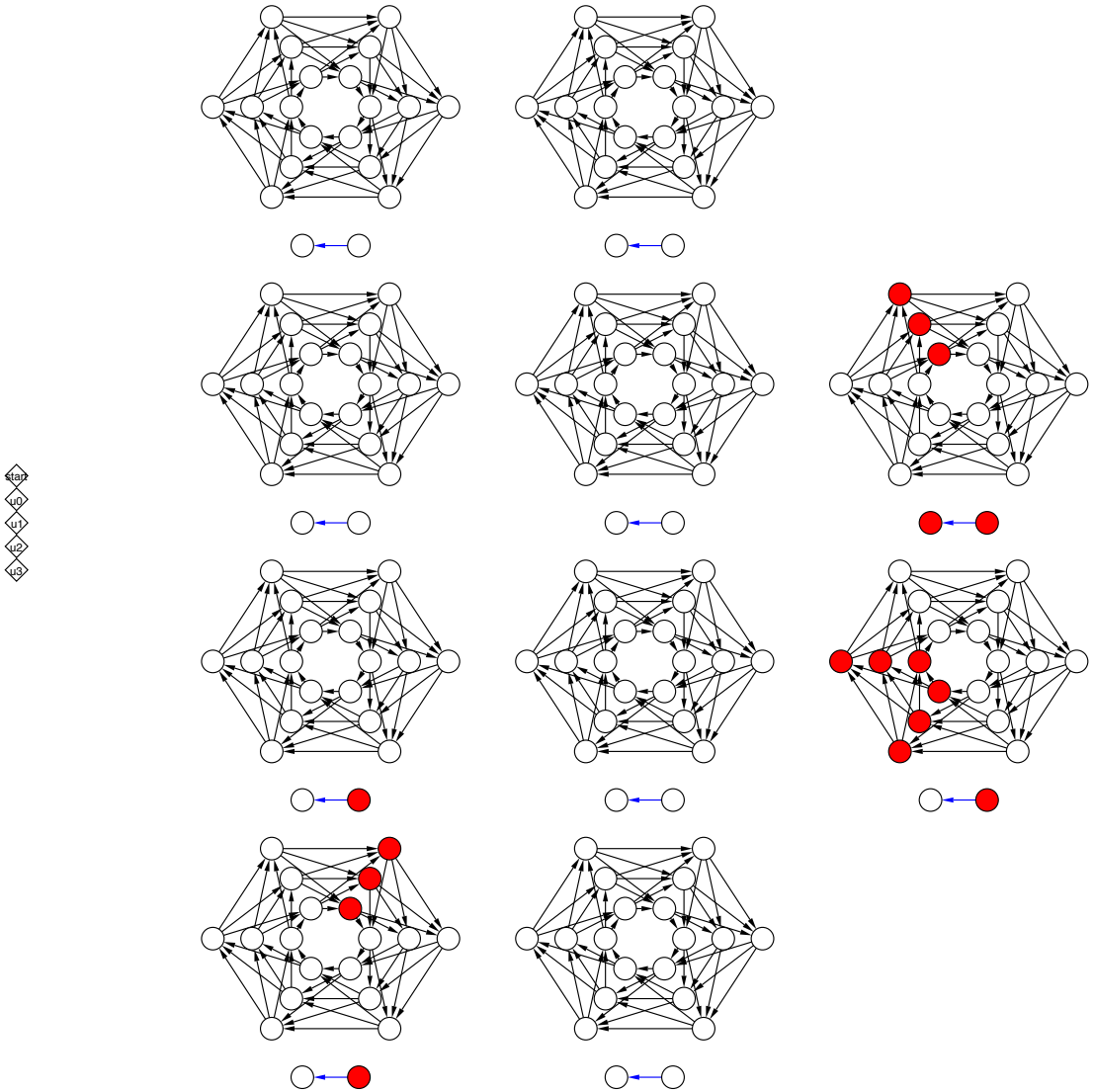
Simulation



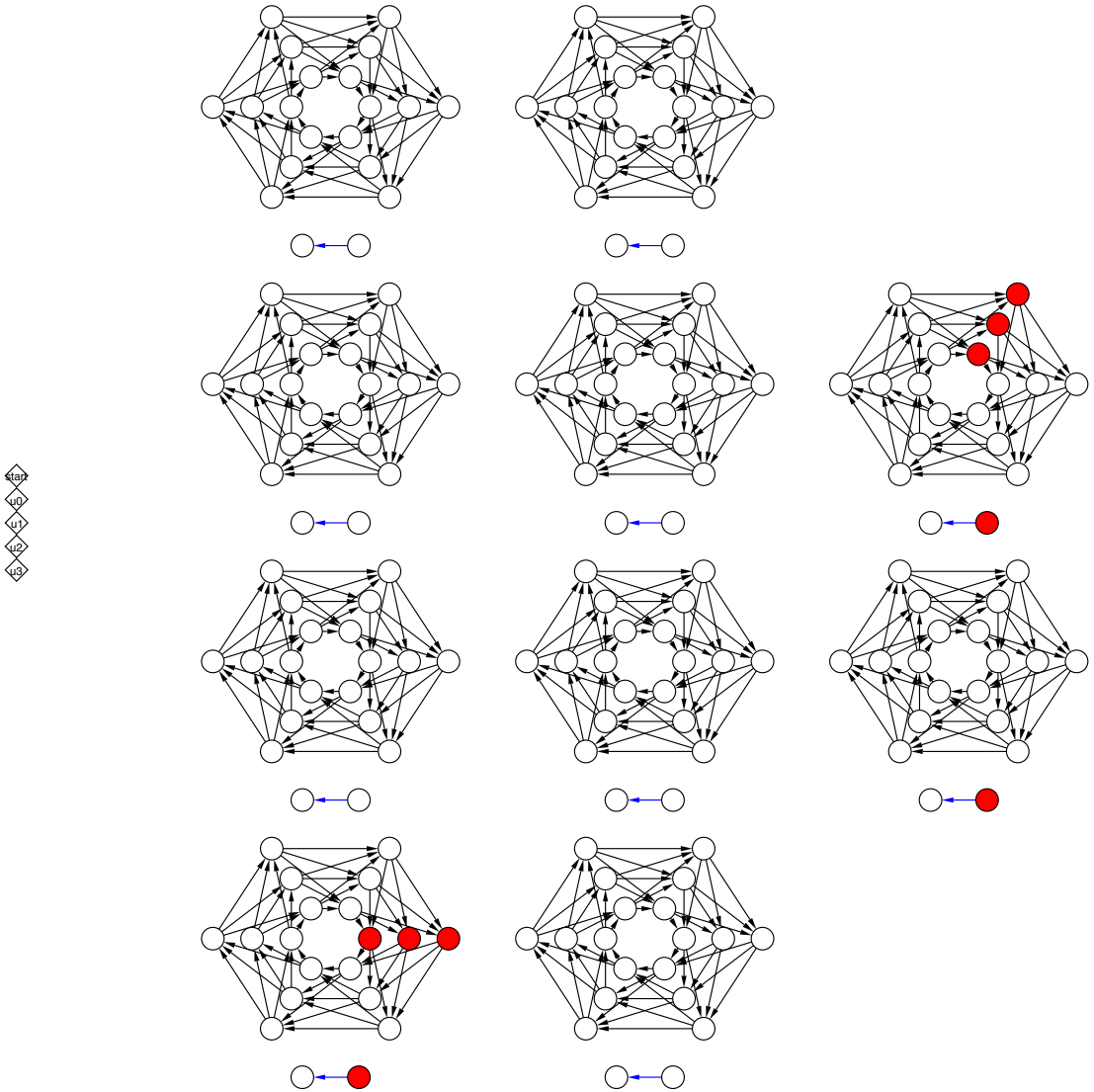
Simulation



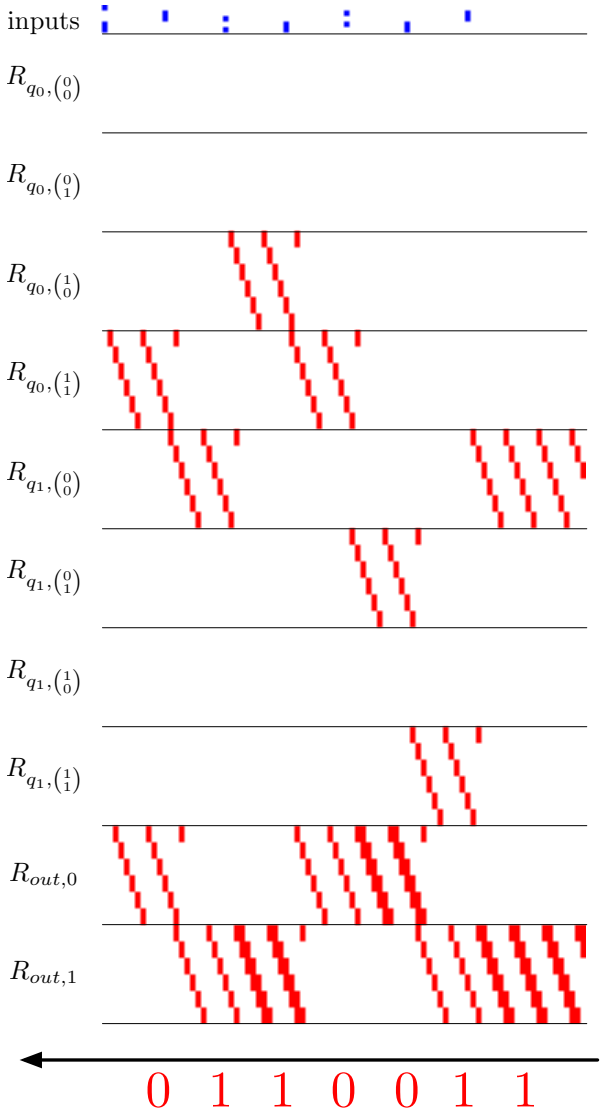
Simulation



Simulation



Simulation


inputs
$R_{q_0,\binom{0}{0}}$
$R_{q_0,\binom{0}{1}}$
$R_{q_0,\binom{1}{0}}$
$R_{q_0,\binom{1}{1}}$
$R_{q_1,\binom{0}{0}}$
$R_{q_1,\binom{0}{1}}$
$R_{q_1,\binom{1}{0}}$
$R_{q_1,\binom{1}{1}}$
$R_{out,0}$
$R_{out,1}$
0 1 1 0 0 1 1

Automata & Boolean RNNs with Synfire Rings

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

Theorem

Any finite state automaton can be simulated by some Boolean neural network composed of synfire rings.

Hodgkin-Huxley Neurons (software demo)

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

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

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

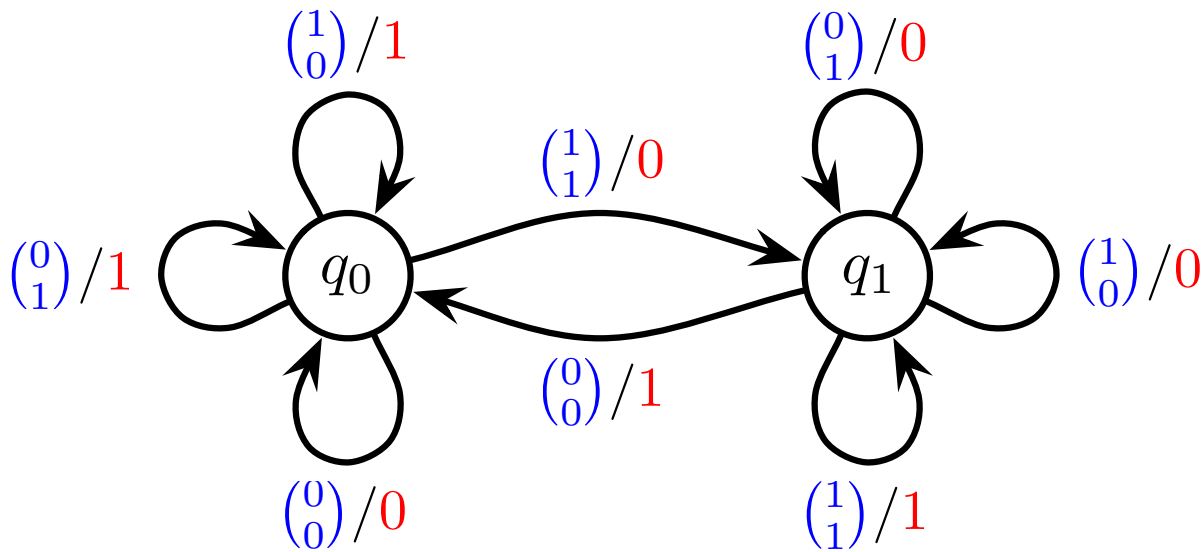
$$\frac{dn}{dt} = \alpha_n(V_m)(1 - n) - \beta_n(V_m)n$$

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

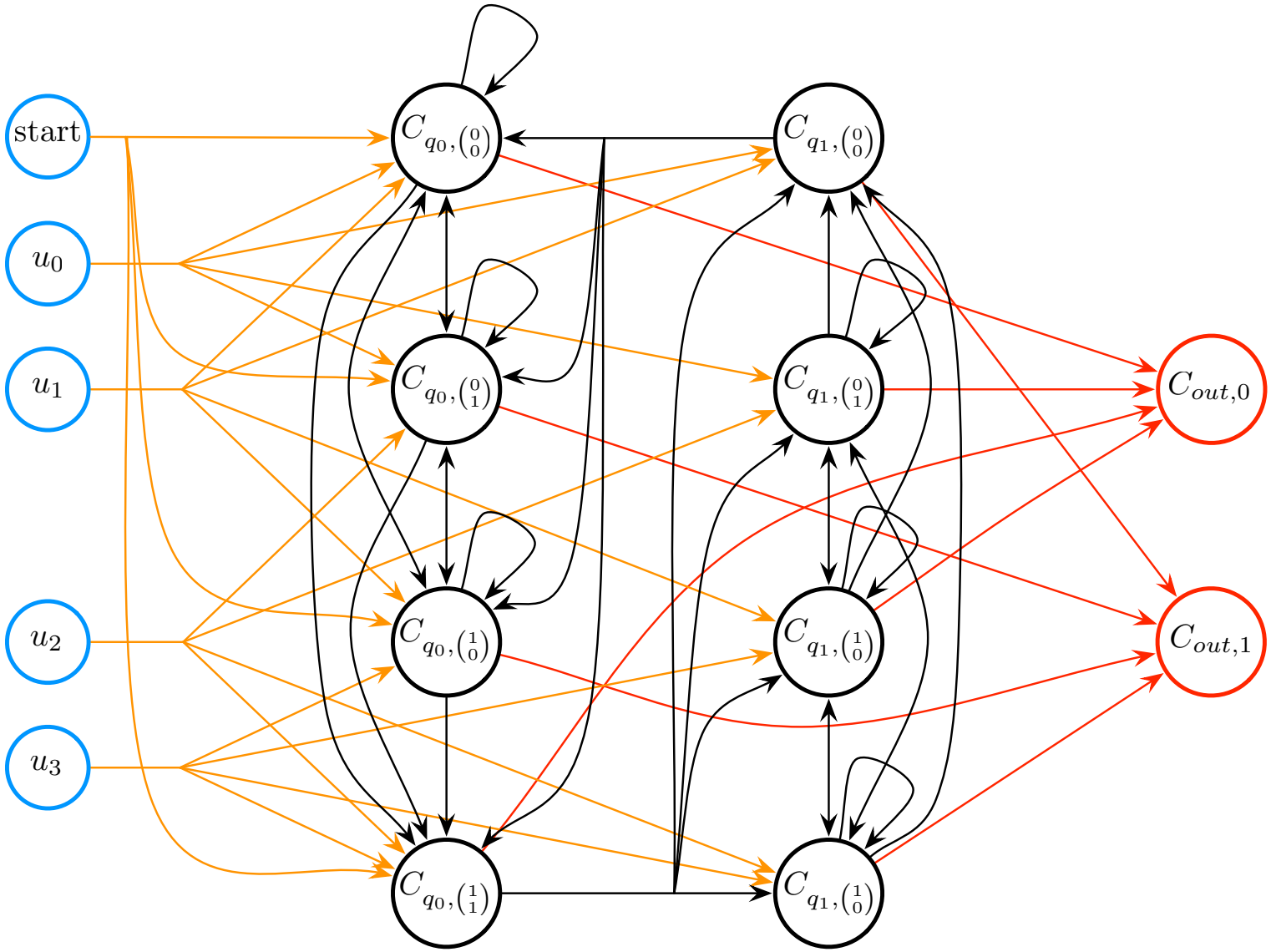
$$\frac{dh}{dt} = \alpha_h(V_m)(1 - h) - \beta_h(V_m)h$$

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

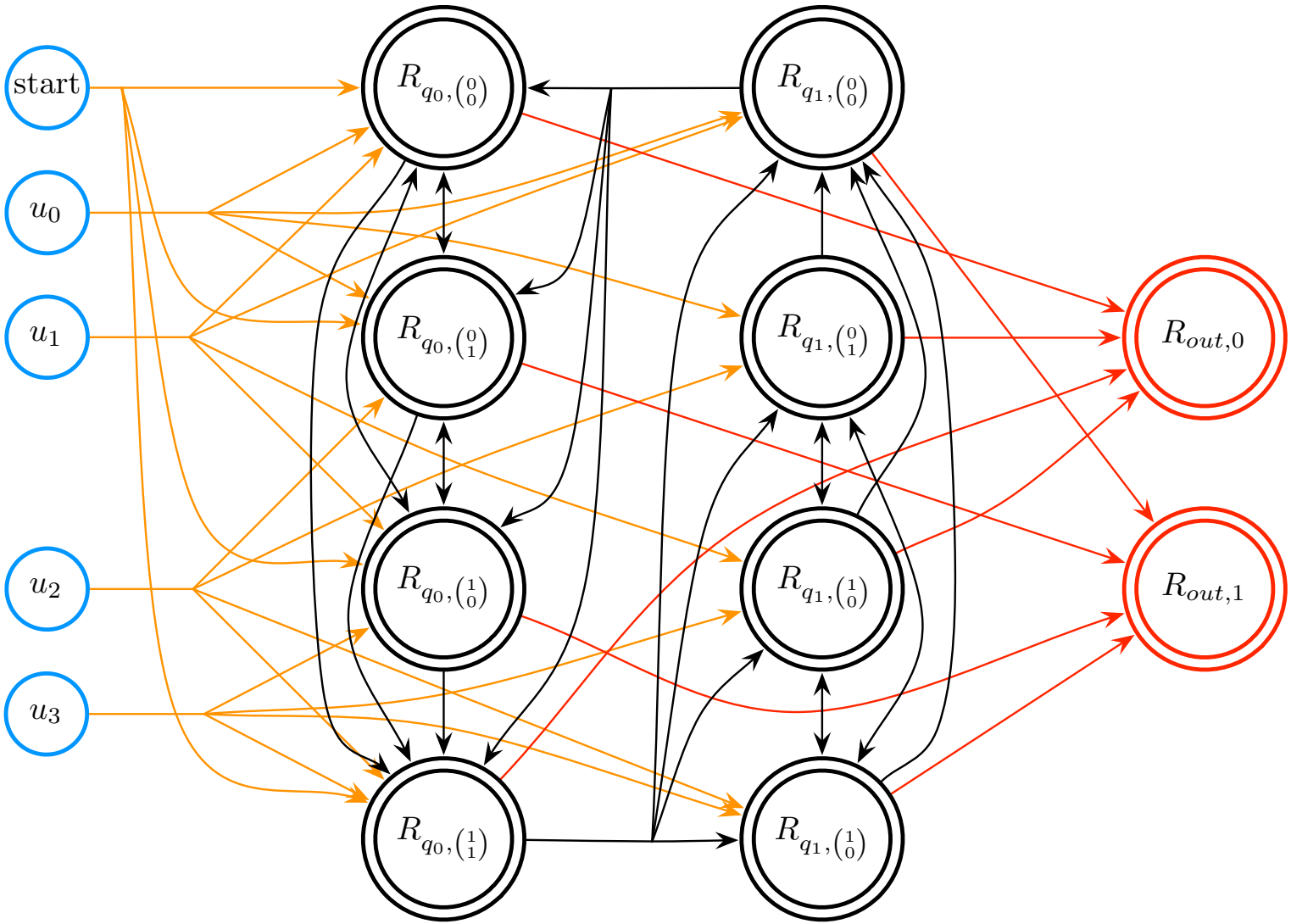
General Construction



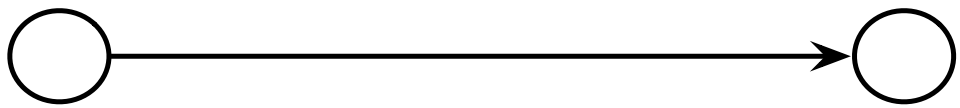
General Construction



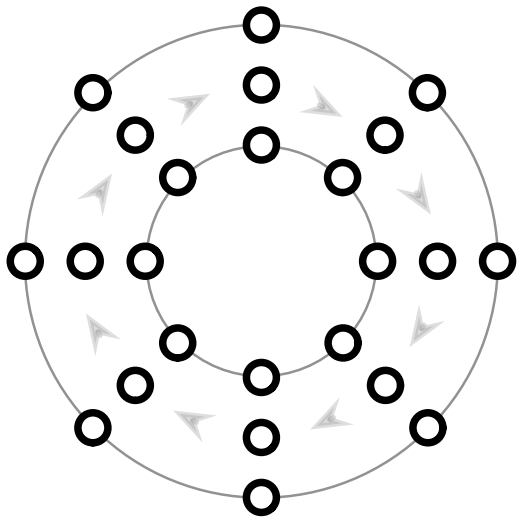
General Construction

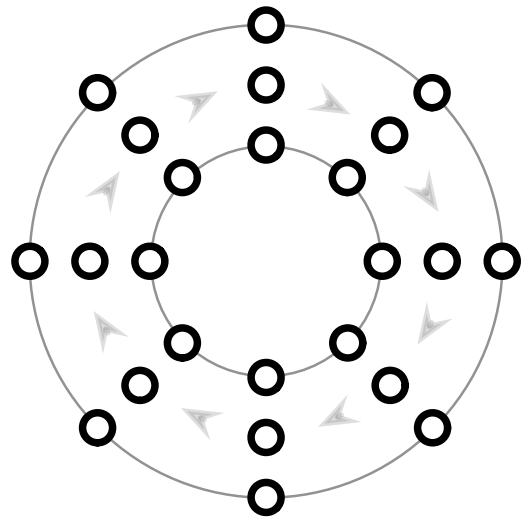


Fibres of connections & Inhibitory System

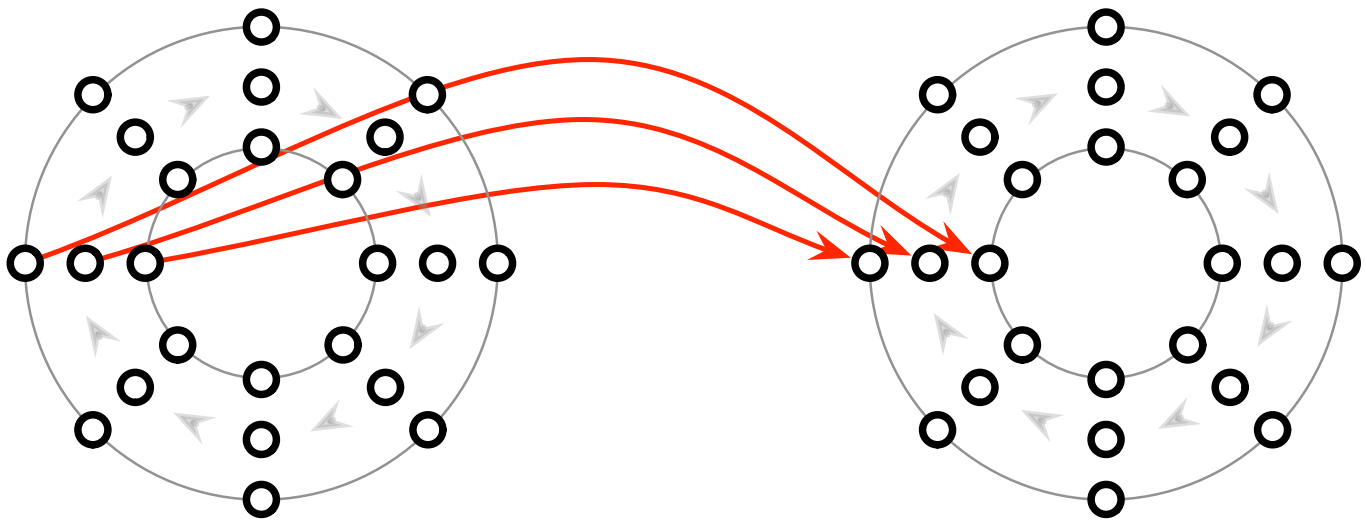


Fibres of connections & Inhibitory System

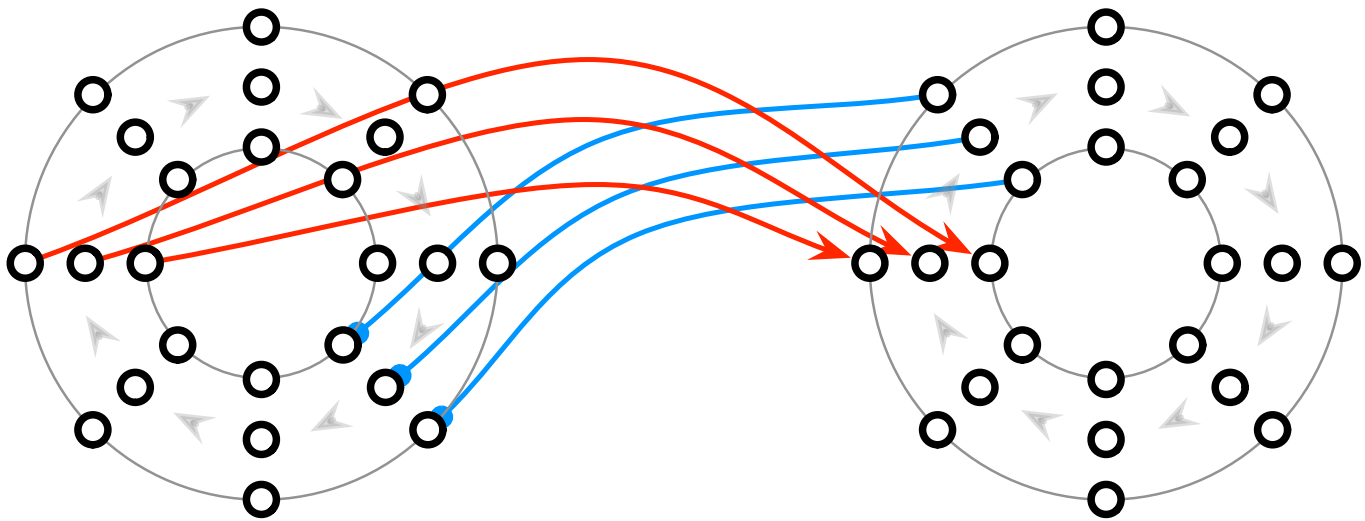




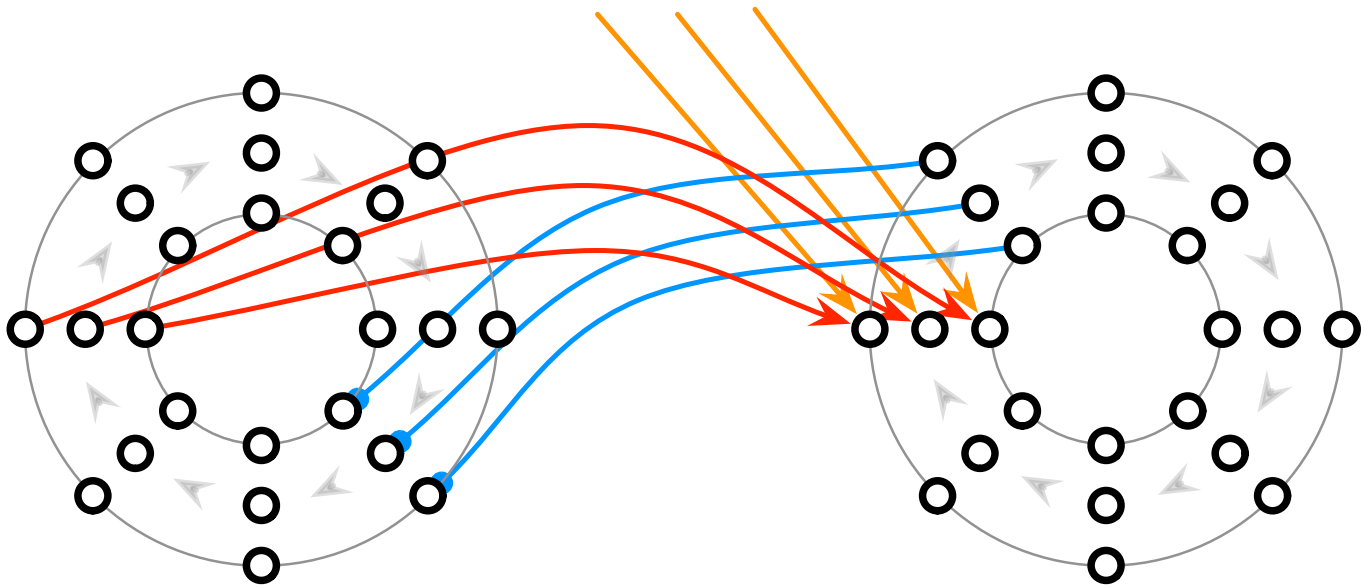
Fibres of connections & Inhibitory System



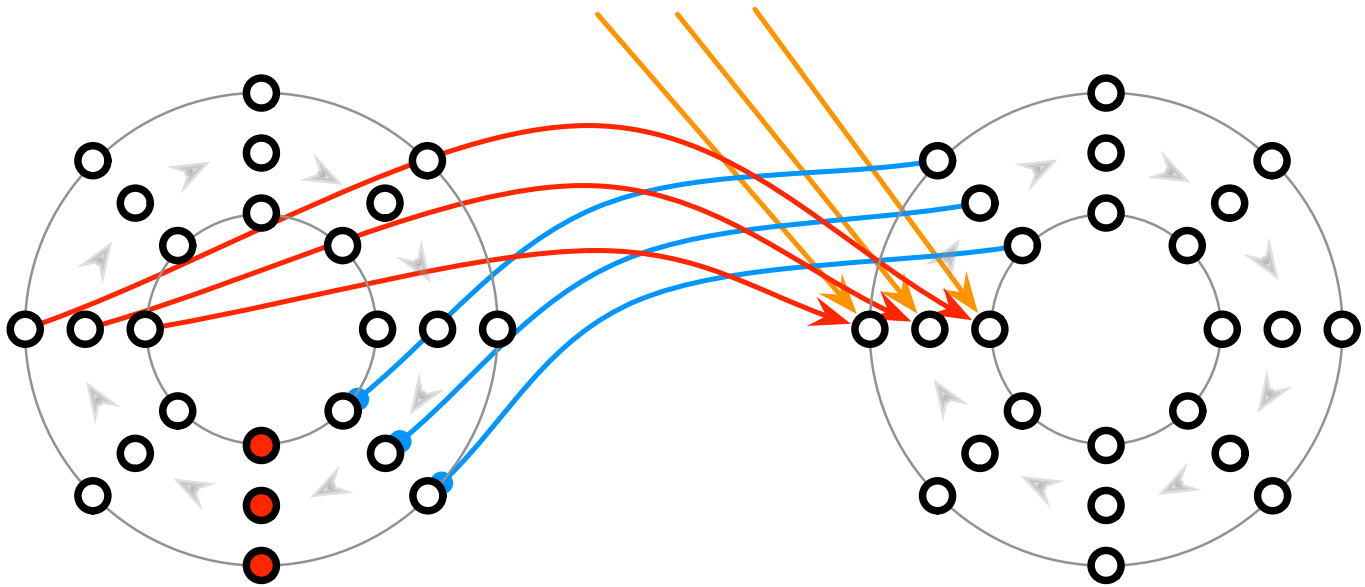
Fibres of connections & Inhibitory System



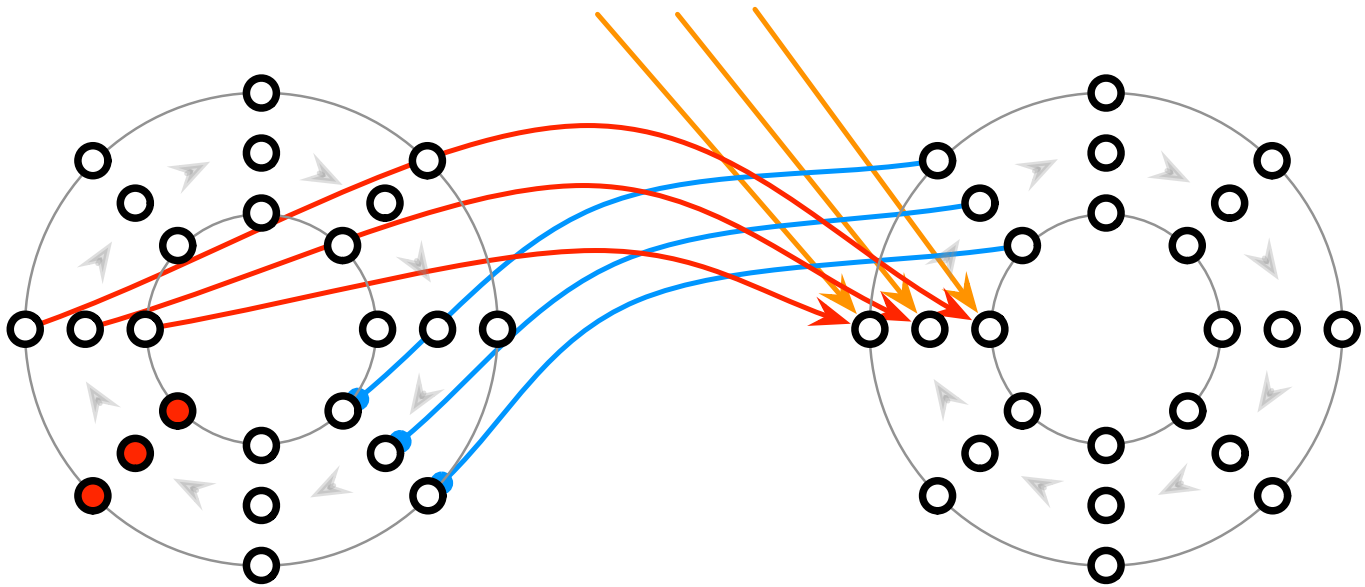
Fibres of connections & Inhibitory System



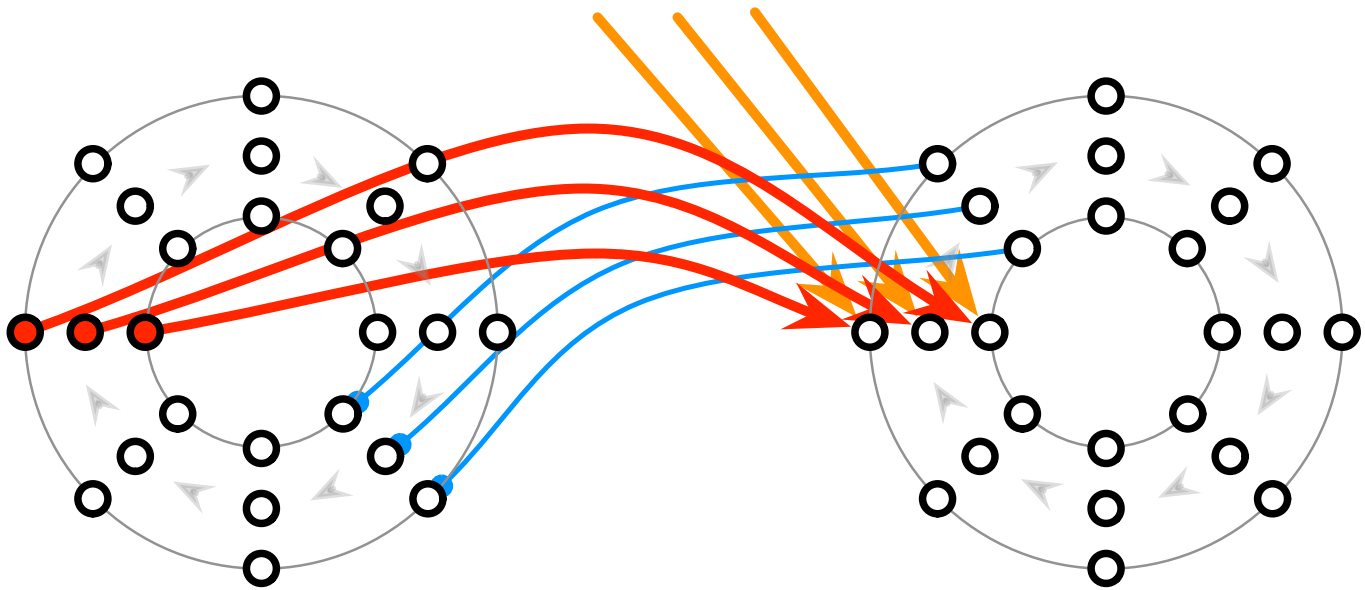
Fibres of connections & Inhibitory System



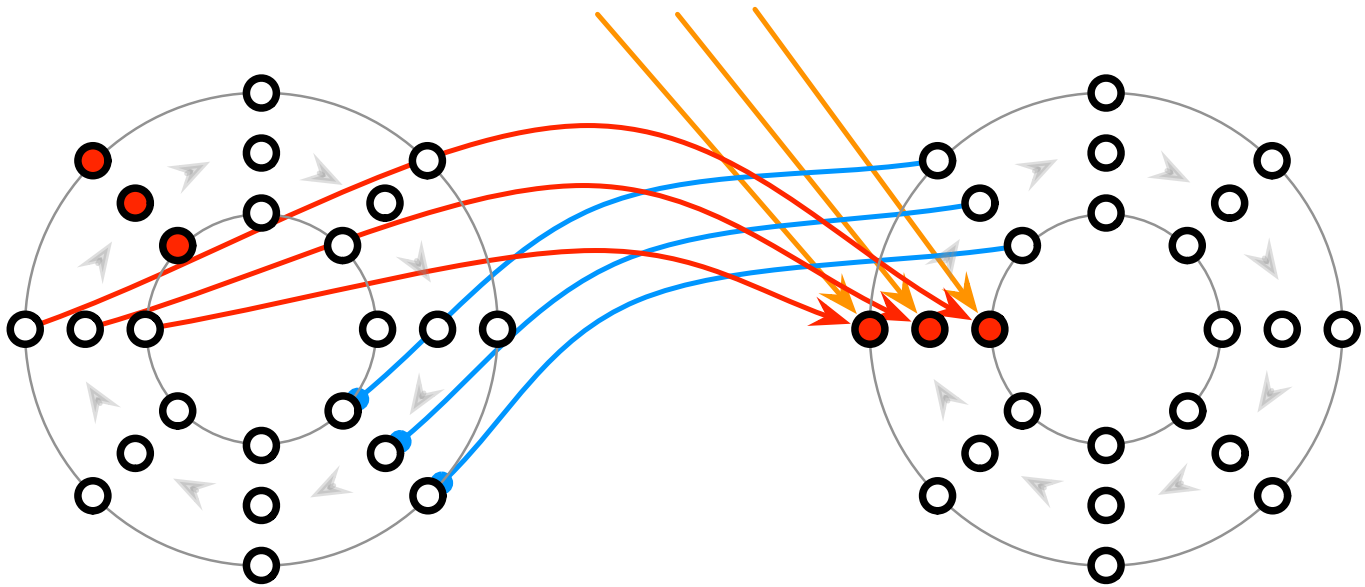
Fibres of connections & Inhibitory System



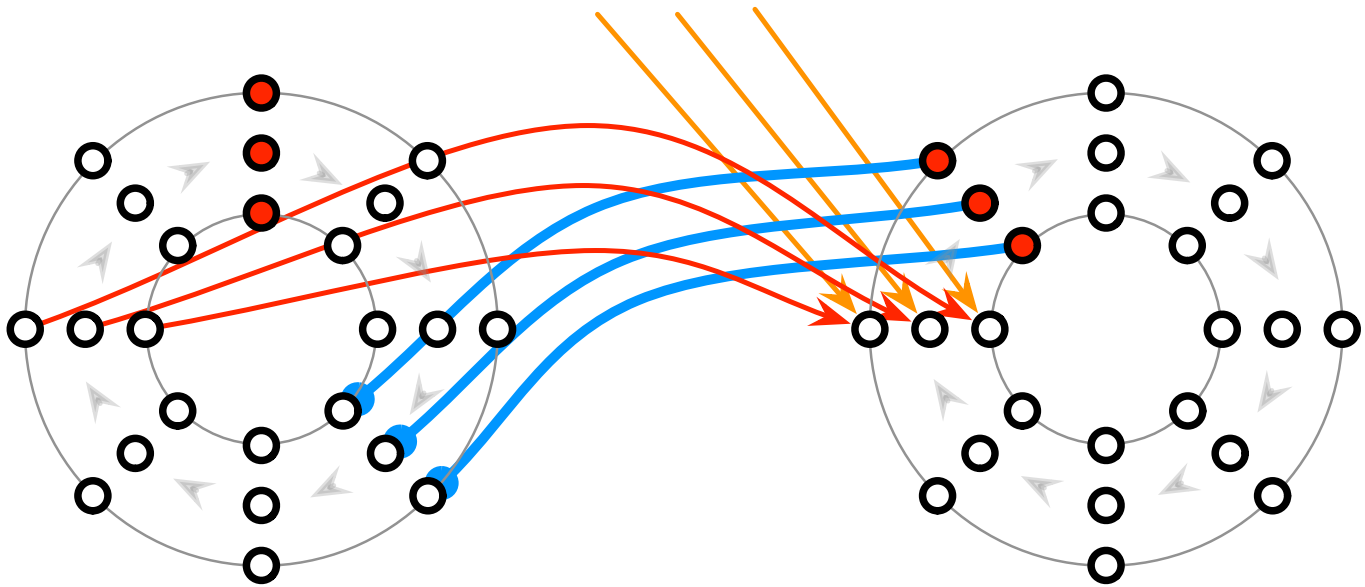
Fibres of connections & Inhibitory System



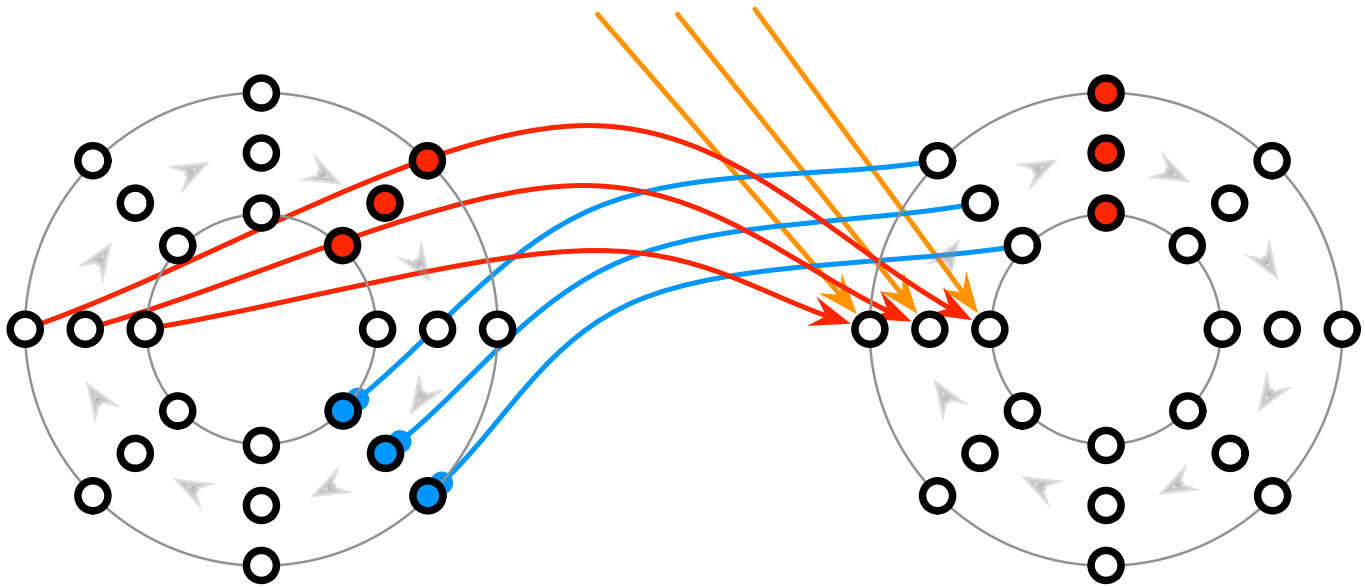
Fibres of connections & Inhibitory System



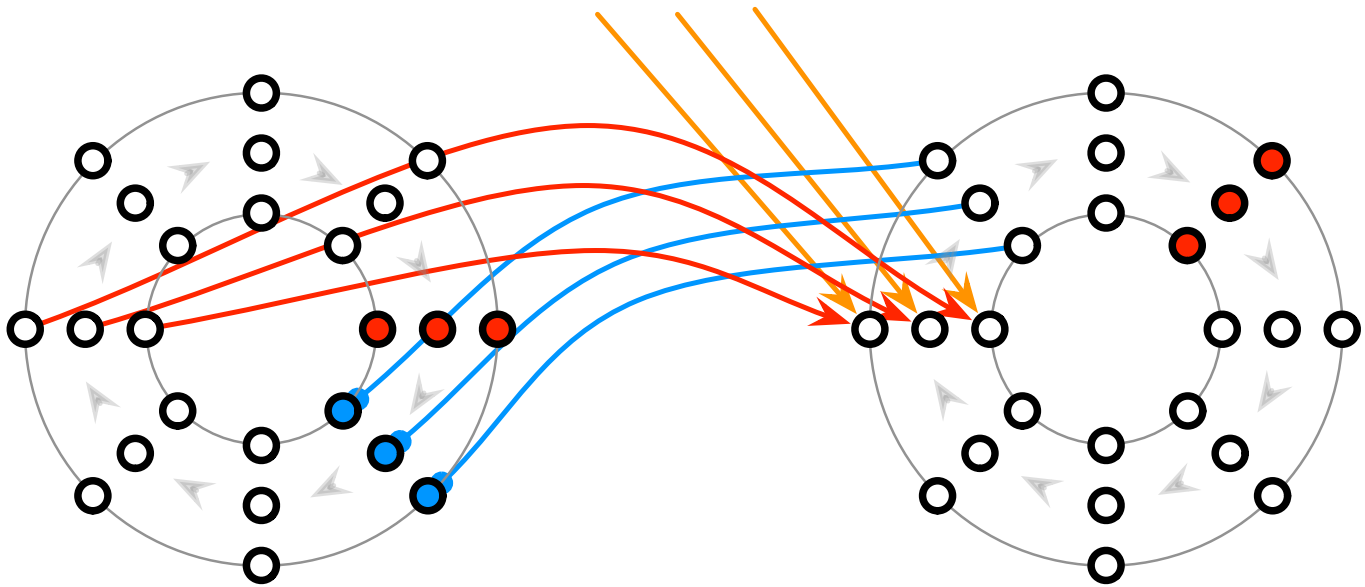
Fibres of connections & Inhibitory System



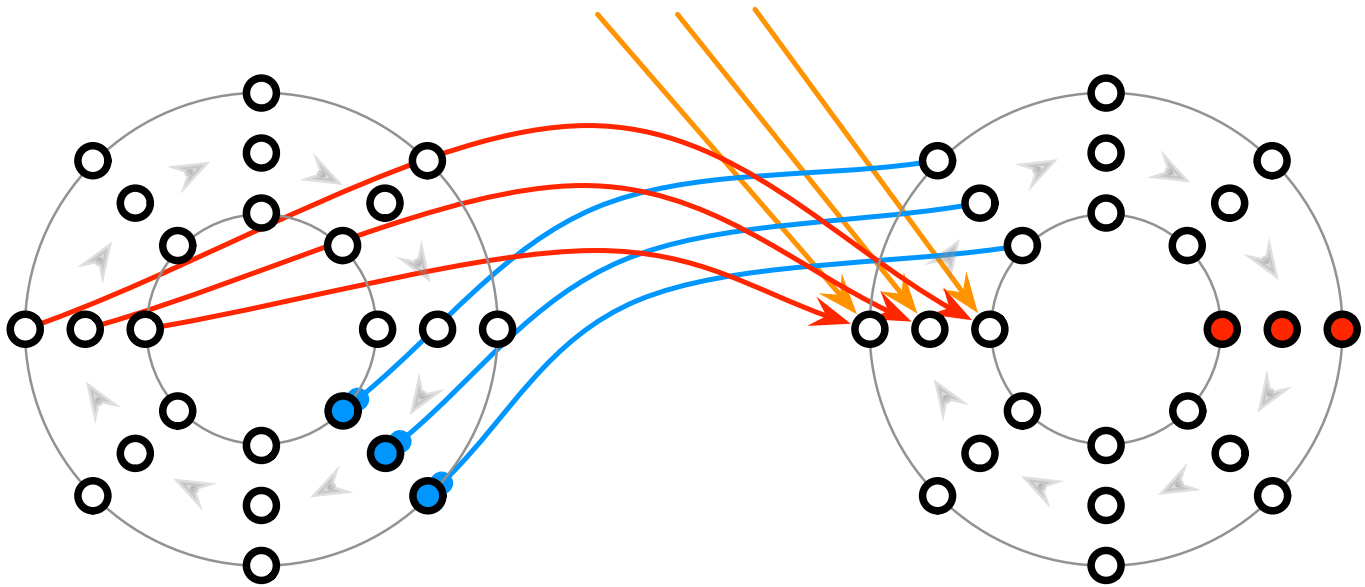
Fibres of connections & Inhibitory System



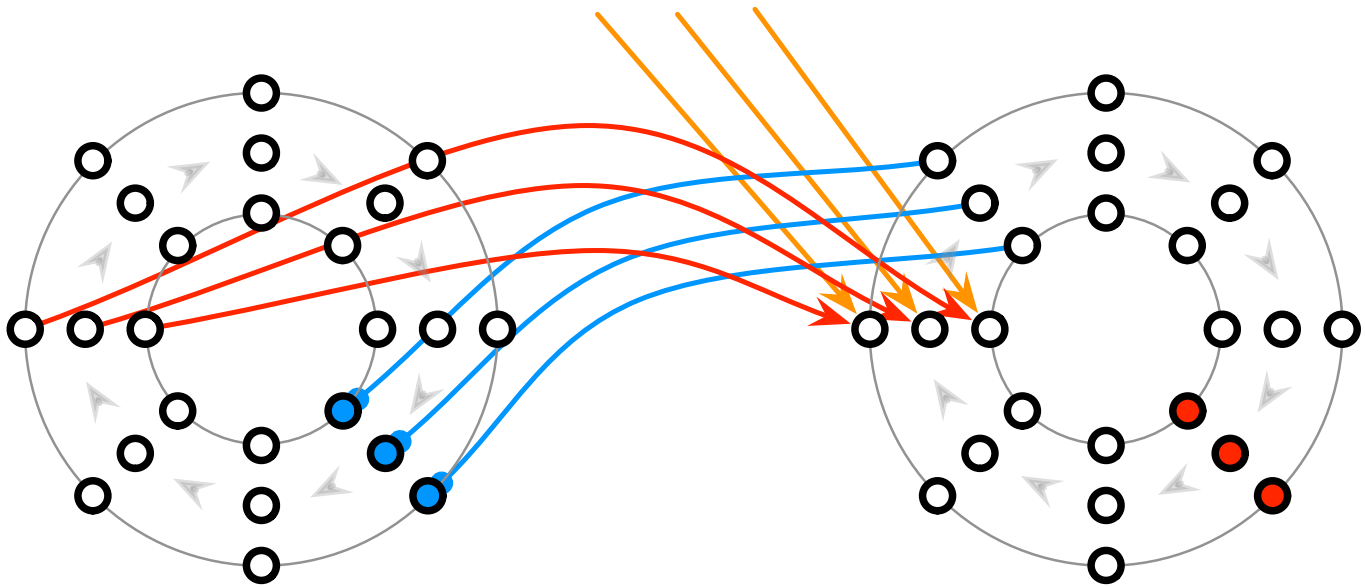
Fibres of connections & Inhibitory System



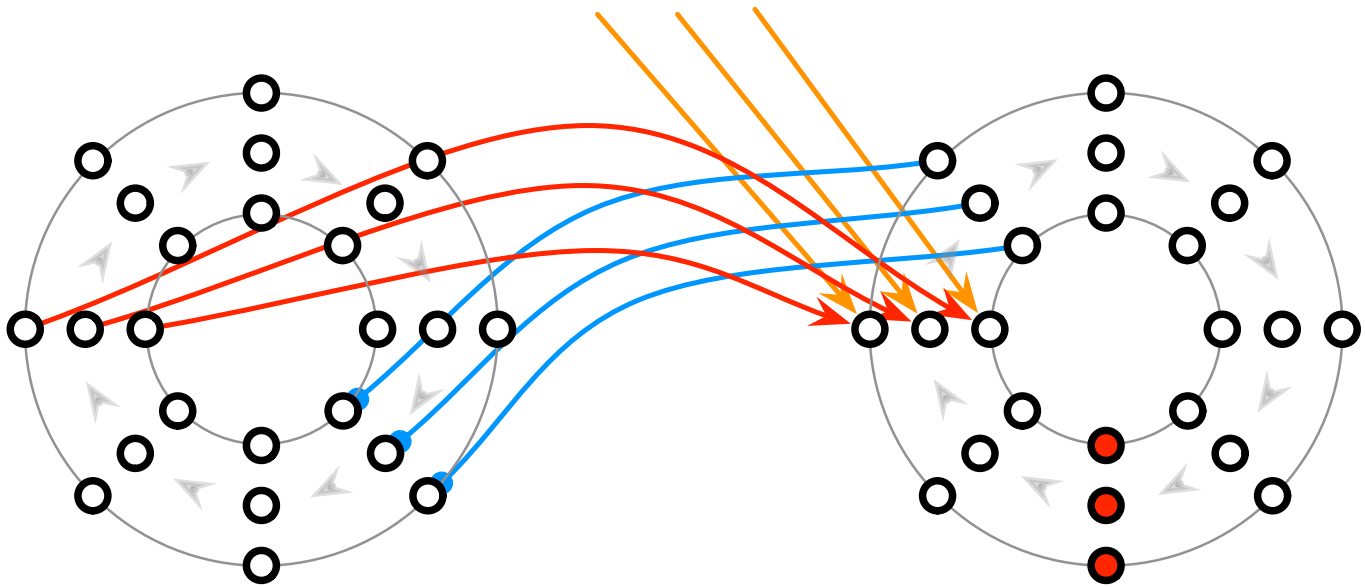
Fibres of connections & Inhibitory System



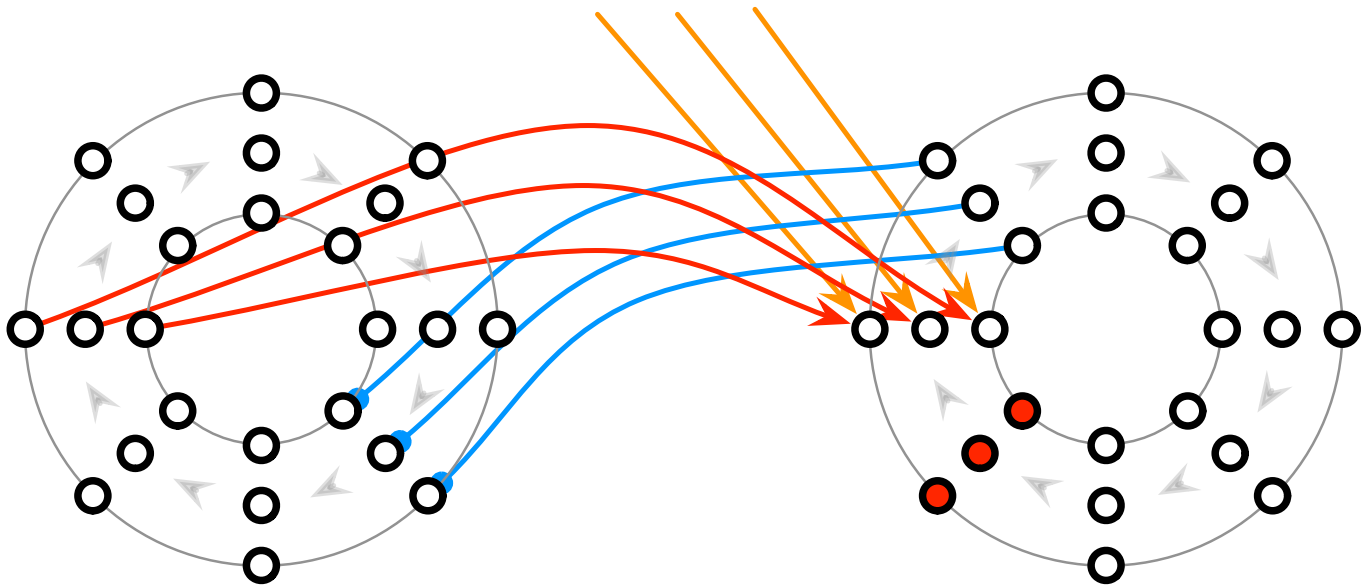
Fibres of connections & Inhibitory System



Fibres of connections & Inhibitory System



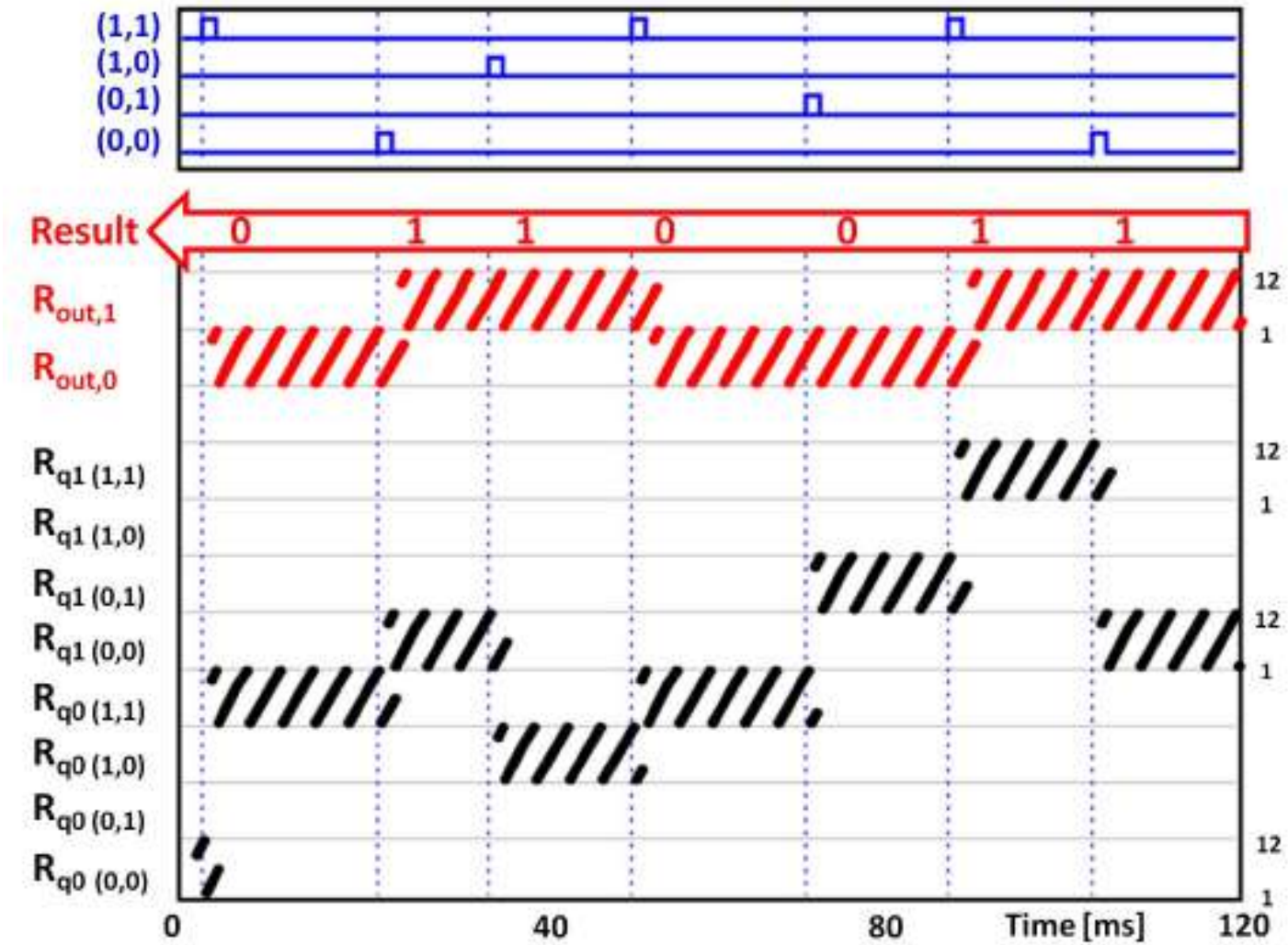
Fibres of connections & Inhibitory System



Simulation

Play movie...

Simulation


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

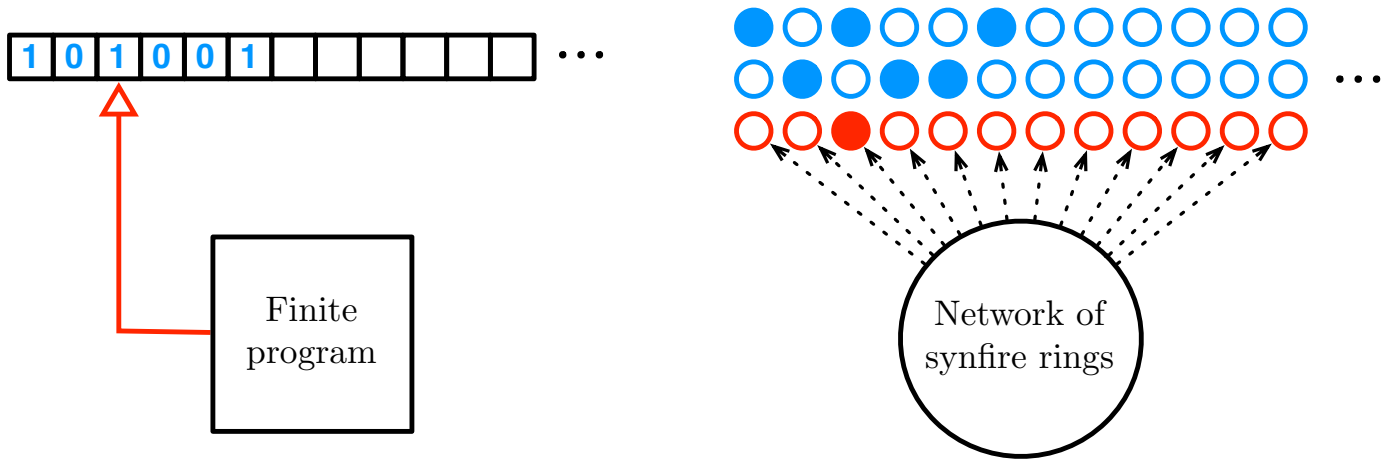
Automata & Hodgkin-Huxley RNNs with Synfire Rings

Theorem

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

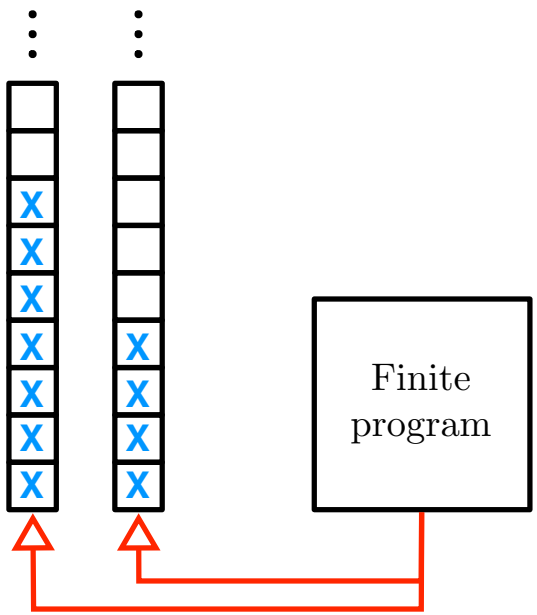
Future Work: Towards Turing Completeness

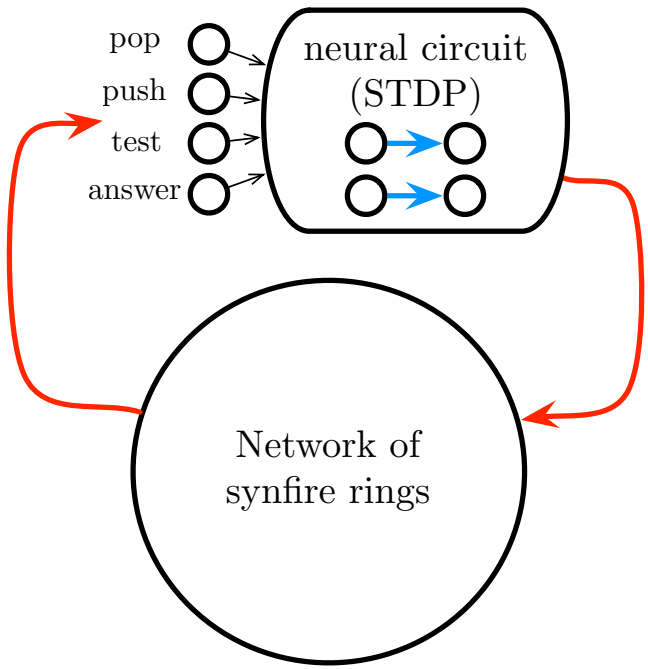
We intend to simulate Turing machines with recurrent neural networks composed of synfire rings.



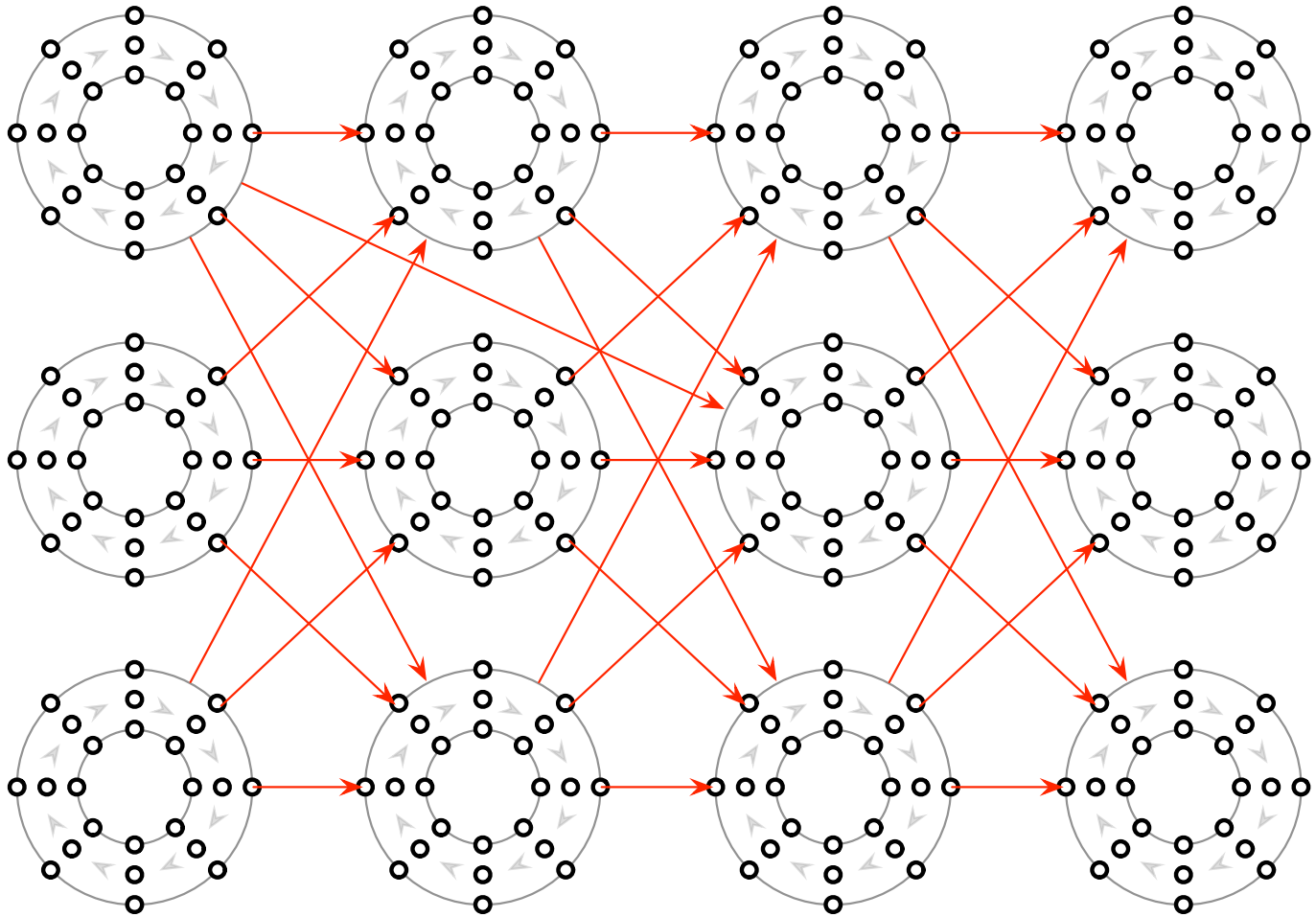
Future Work: Towards Turing Completeness

We intend to simulate Turing machines with recurrent neural networks composed of synfire rings.





Future Work: Learning within the Synfire Ring Architecture



Conclusions

- Recurrent neural networks represent a natural models for oracle-based computation, beyond the Turing limits.
- We introduced a new paradigm of neural computation based on the concept of synfire rings.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of neural computers. By growing cultures of neurons according to the synfire ring architecture, one could in principle realize neural computers.

Conclusions

- Recurrent neural networks represent a natural models for oracle-based computation, beyond the Turing limits.
- We introduced a new paradigm of neural computation based on the concept of synfire rings.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of neural computers. By growing cultures of neurons according to the synfire ring architecture, one could in principle realize neural computers.

Conclusions

- Recurrent neural networks represent a natural models for oracle-based computation, beyond the Turing limits.
- We introduced a new paradigm of neural computation based on the concept of synfire rings.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of neural computers. By growing cultures of neurons according to the synfire ring architecture, one could in principle realize neural computers.

Conclusions

- Recurrent neural networks represent a natural models for oracle-based computation, beyond the Turing limits.
- We introduced a new paradigm of neural computation based on the concept of synfire rings.
- We intend to study the issue of *learning* within the synfire ring architecture.
- Utopia: achieve the realization of neural computers. By growing cultures of neurons according to the synfire ring architecture, one could in principle realize neural computers.