Emulation of Finite State Automata with Recurrent Neural Networks Composed of Synfire Rings

Jérémie Cabessa
Joint work with Paolo Masulli

Department of Mathematical Economics
University Paris II, France

IJCNN 17, 18 May 2017, Anchorage

Introduction

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

TM: Turing complete

TM/poly(A): super-Turing

Here, we revisit the equivalence between Boolean RNNs and finite state automata from a more biological perspective.

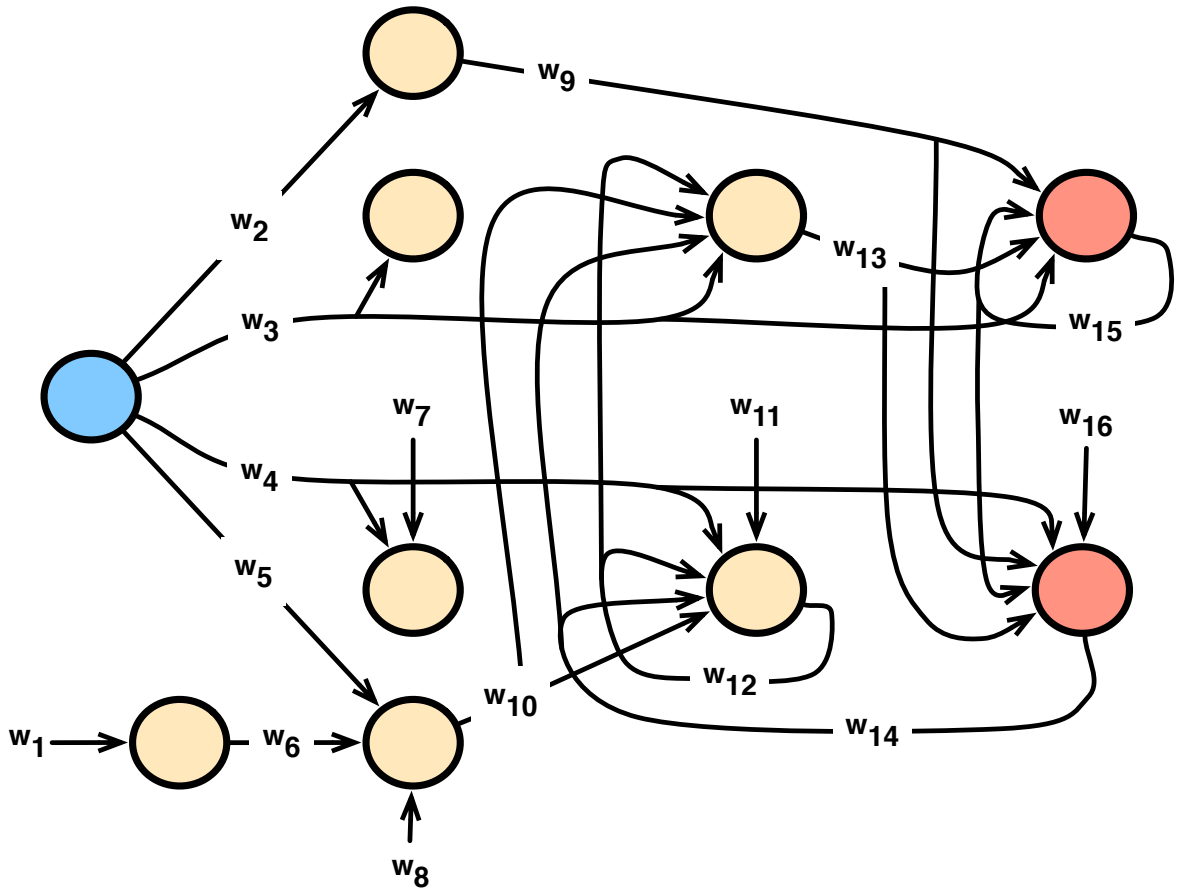
Introduction

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

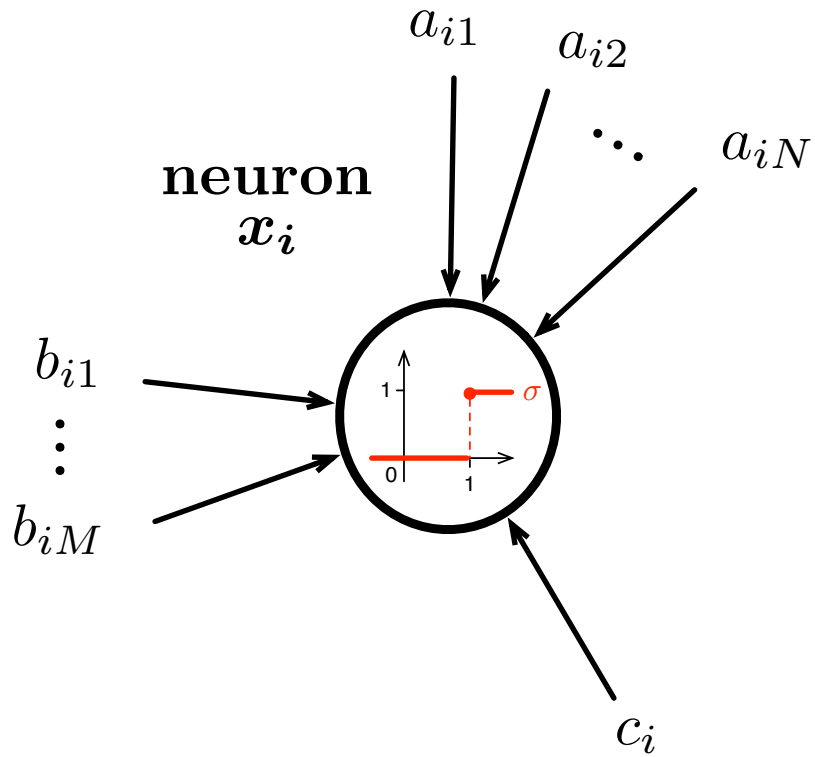
TM: Turing complete
TM/poly(A): super-Turing

Here, we revisit the equivalence between Boolean RNNs and finite state automata from a more biological perspective.

Recurrent Neural Network

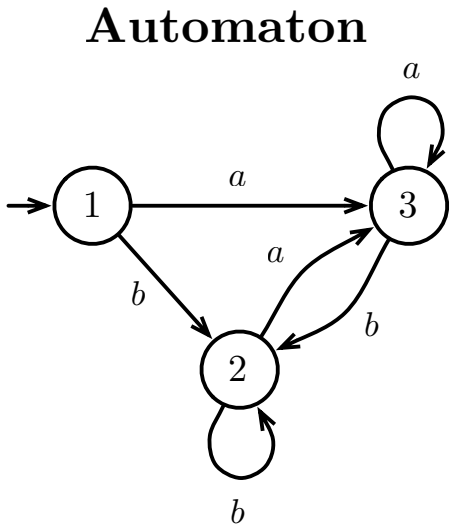


Boolean Recurrent Neural Network



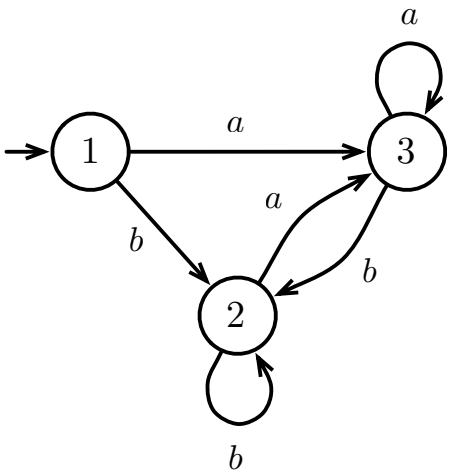
$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

From Automata to Boolean Neural Networks

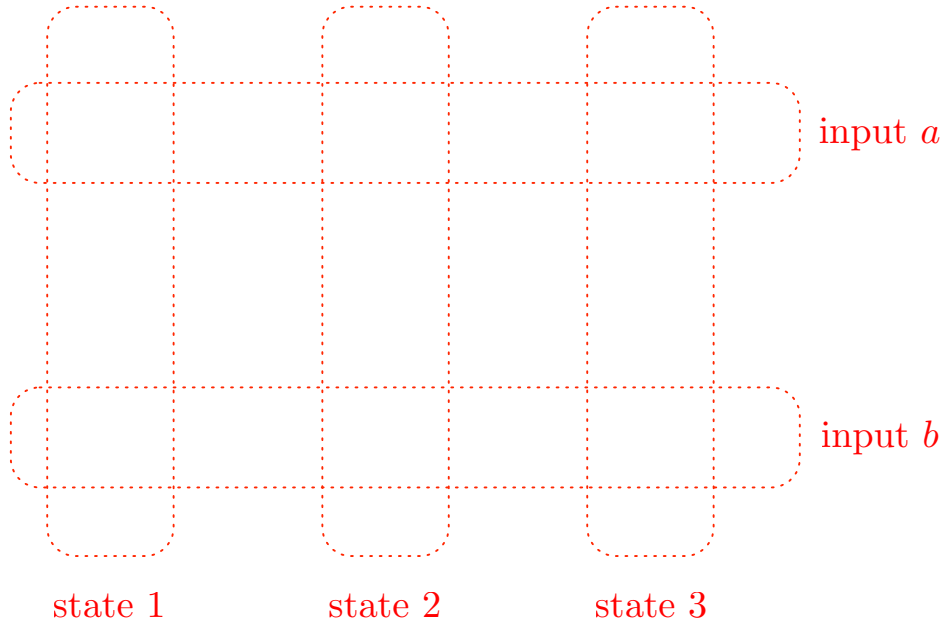


From Automata to Boolean Neural Networks

Automaton

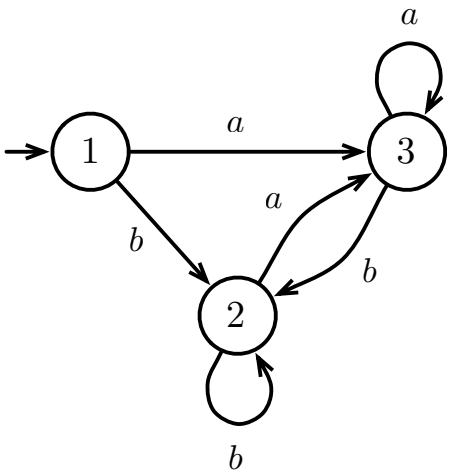


Boolean Neural Network

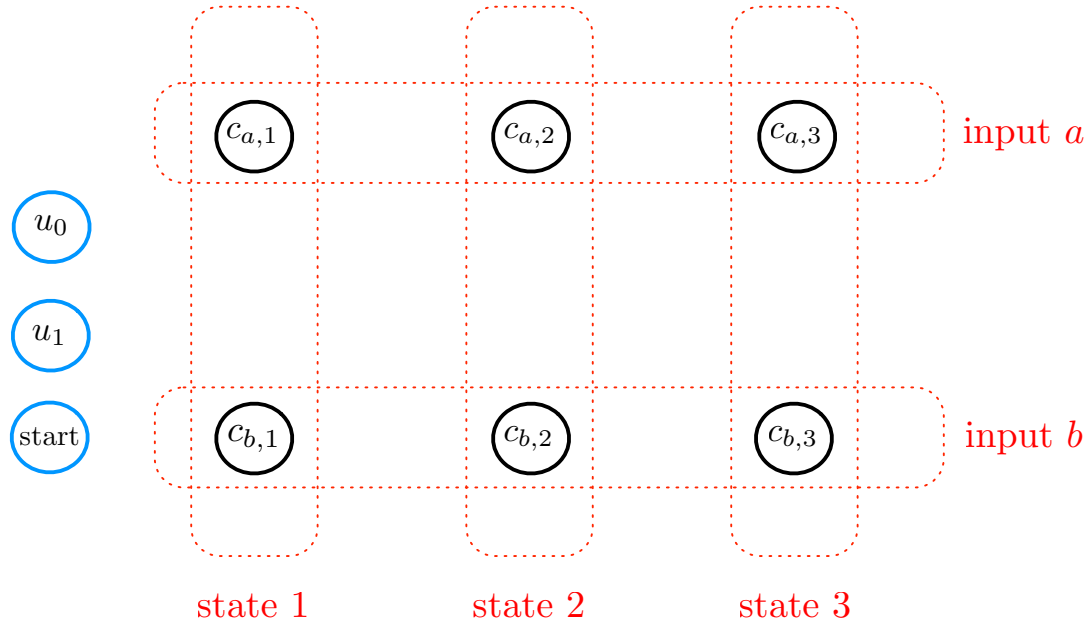


From Automata to Boolean Neural Networks

Automaton

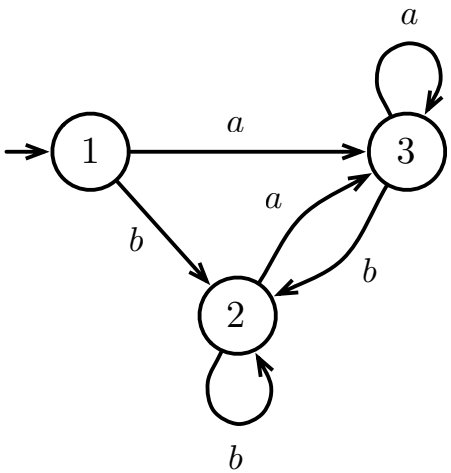


Boolean Neural Network

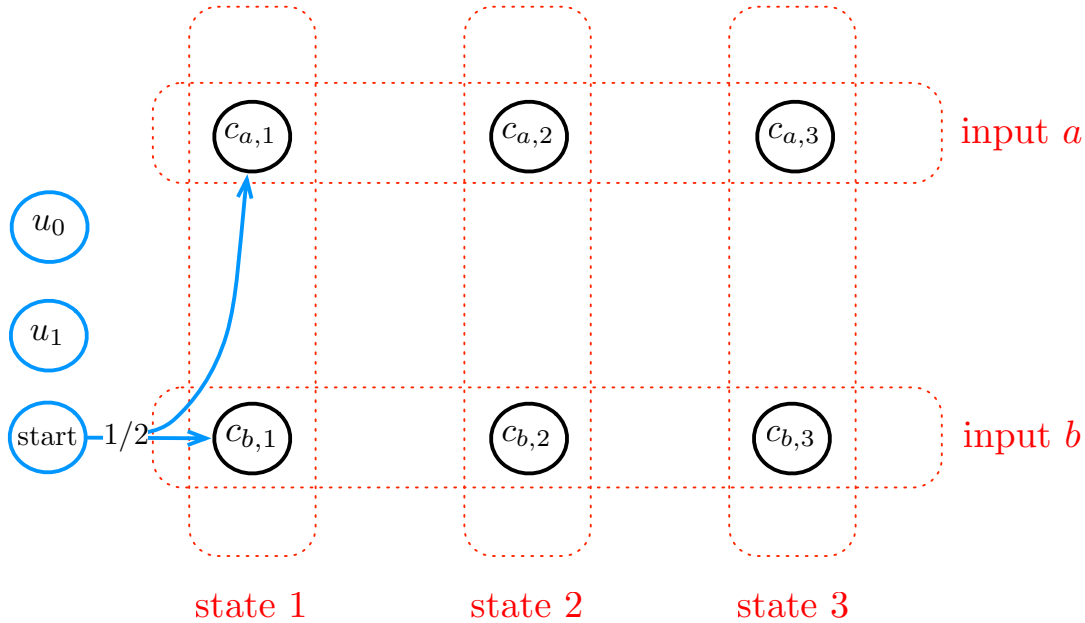


From Automata to Boolean Neural Networks

Automaton

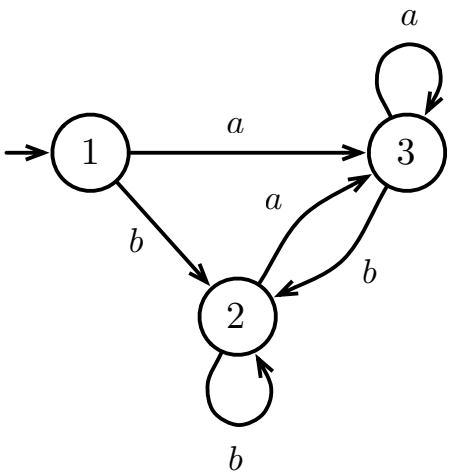


Boolean Neural Network

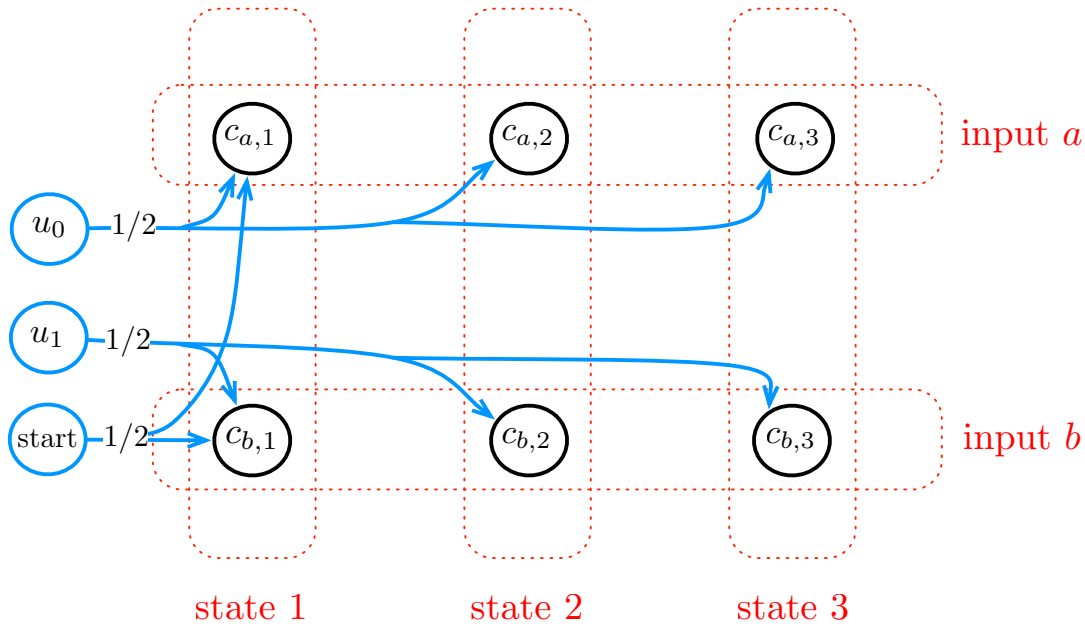


From Automata to Boolean Neural Networks

Automaton

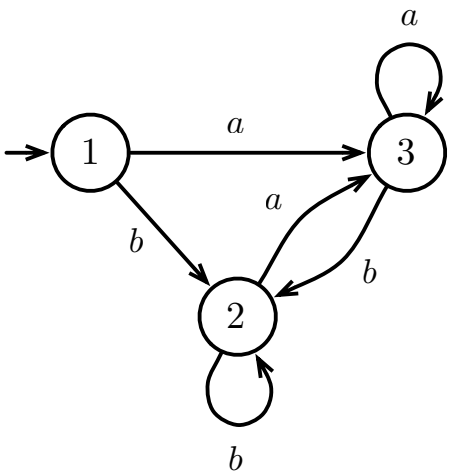


Boolean Neural Network

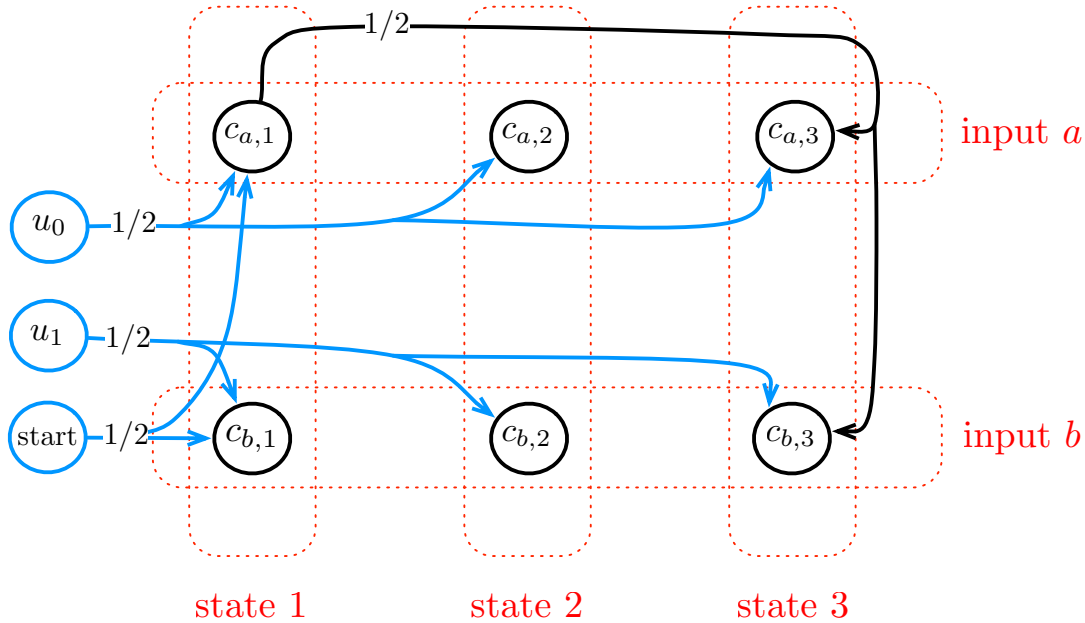


From Automata to Boolean Neural Networks

Automaton

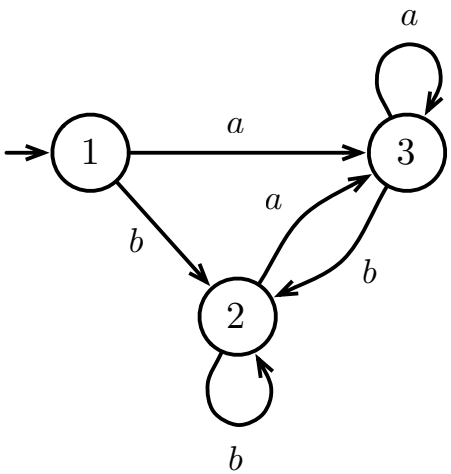


Boolean Neural Network

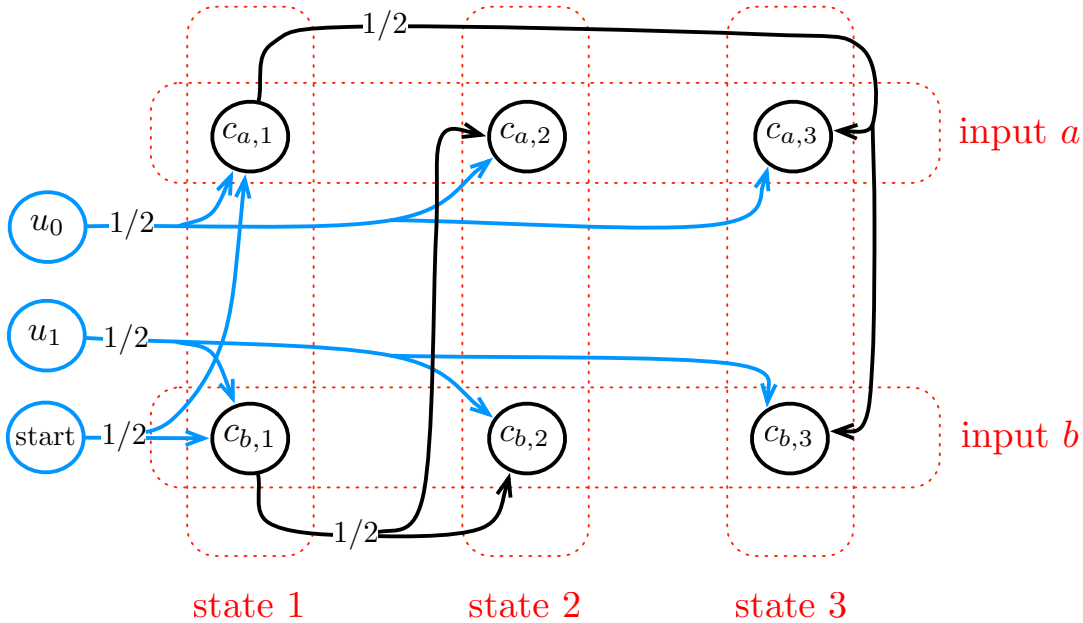


From Automata to Boolean Neural Networks

Automaton

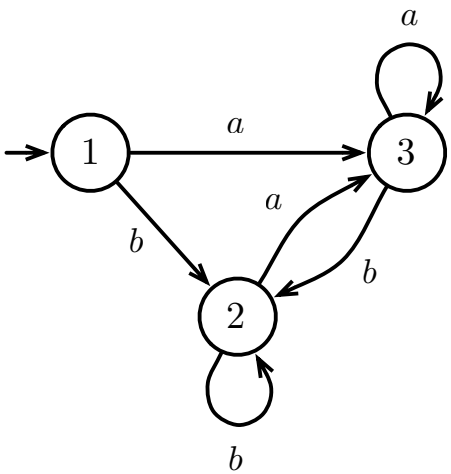


Boolean Neural Network

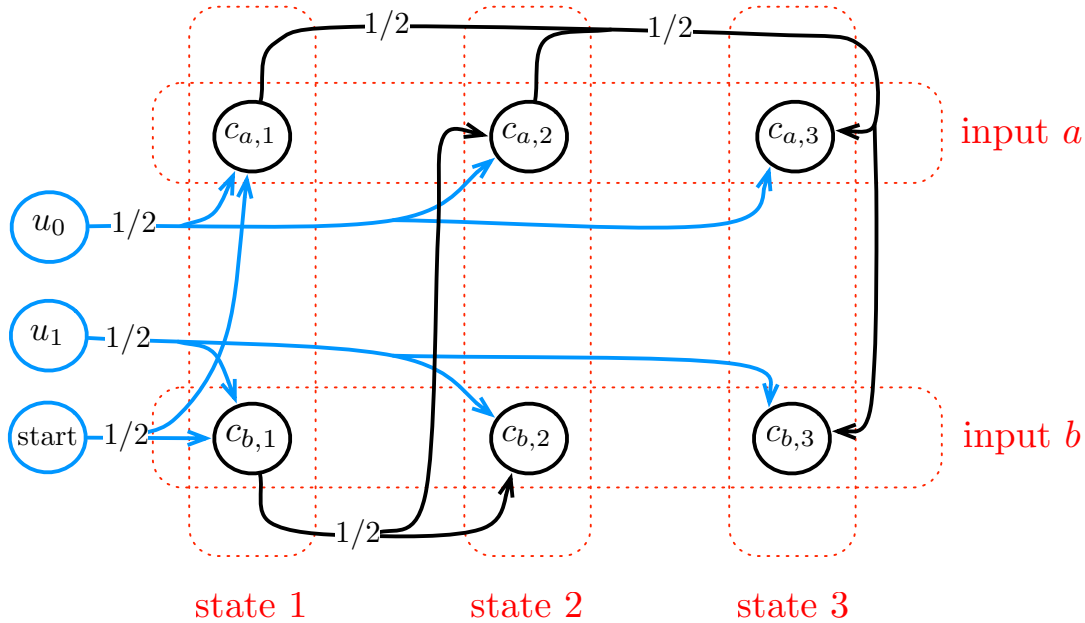


From Automata to Boolean Neural Networks

Automaton

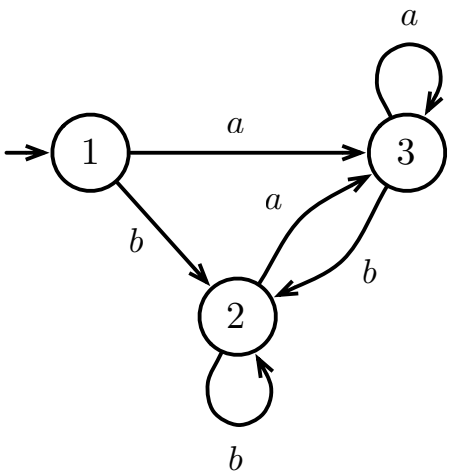


Boolean Neural Network

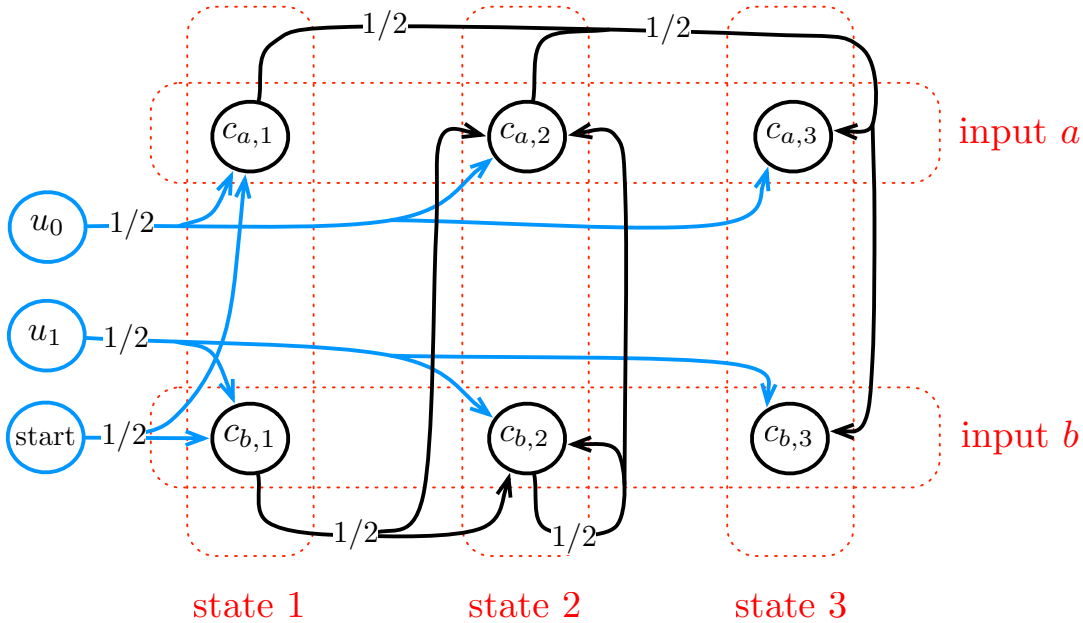


From Automata to Boolean Neural Networks

Automaton

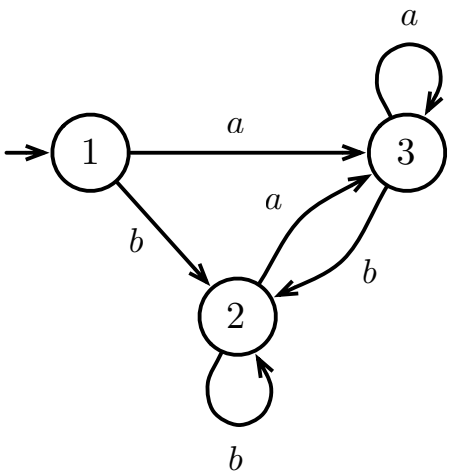


Boolean Neural Network

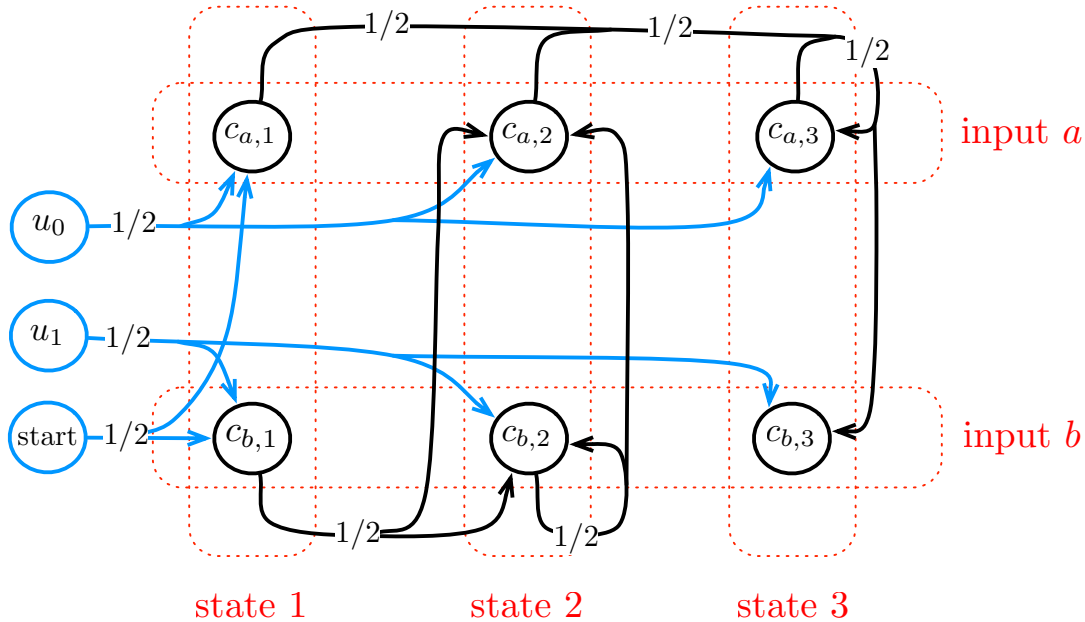


From Automata to Boolean Neural Networks

Automaton

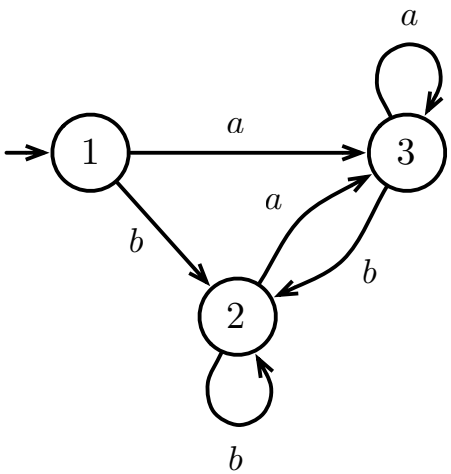


Boolean Neural Network

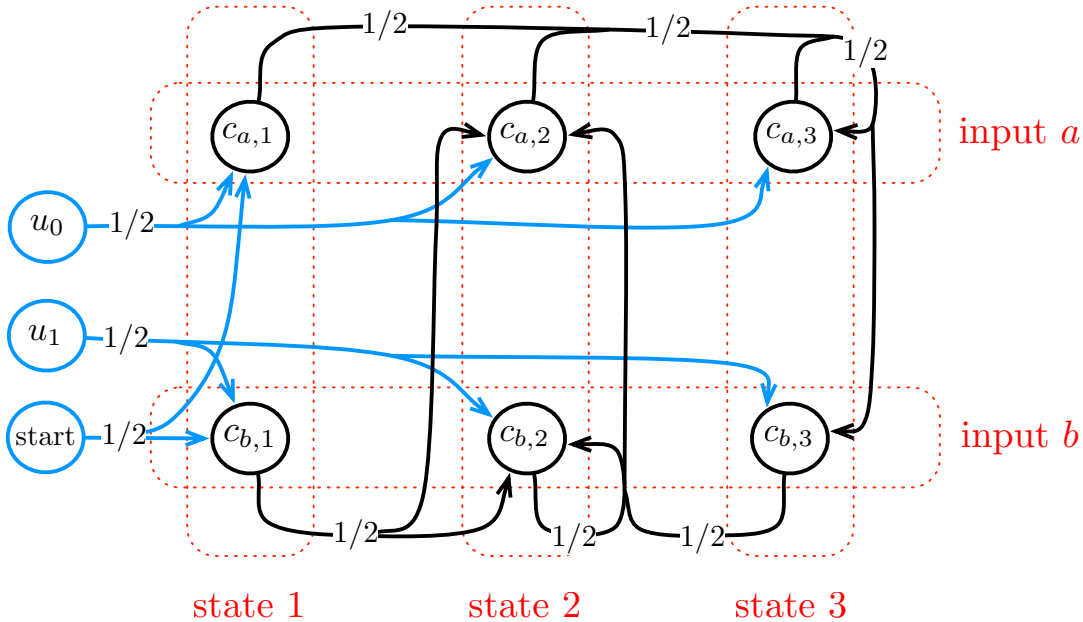


From Automata to Boolean Neural Networks

Automaton

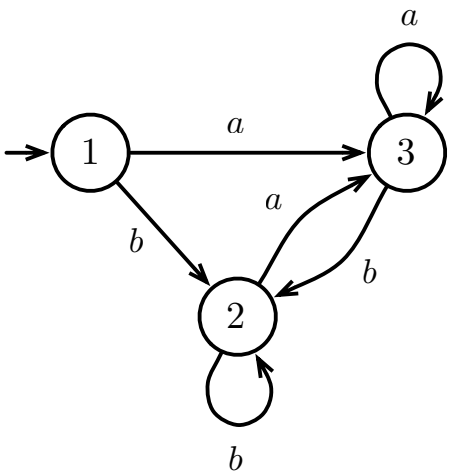


Boolean Neural Network

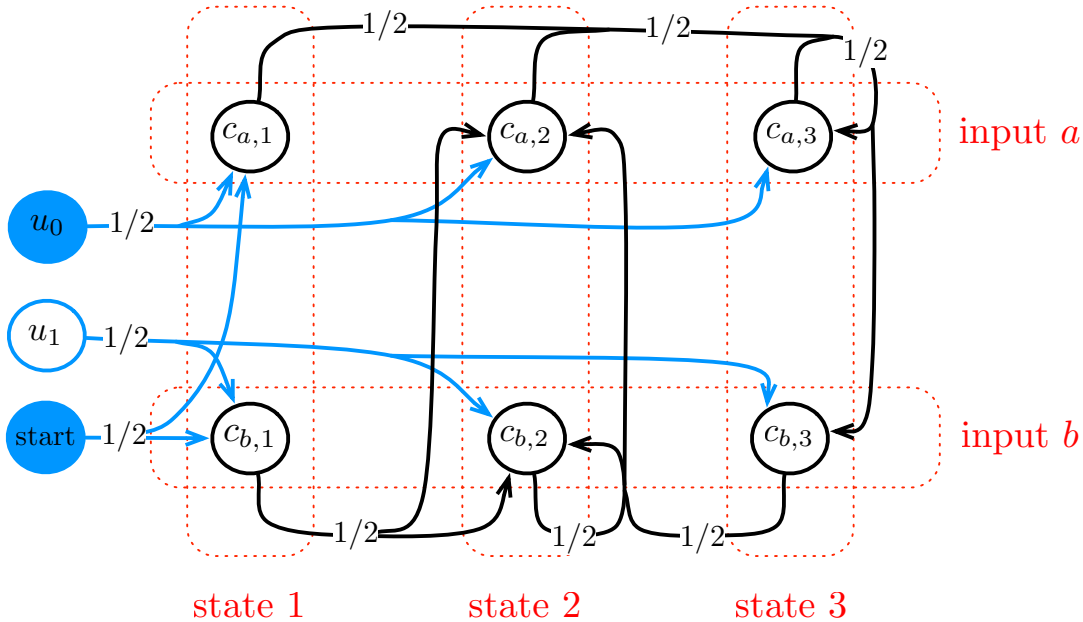


From Automata to Boolean Neural Networks

Automaton

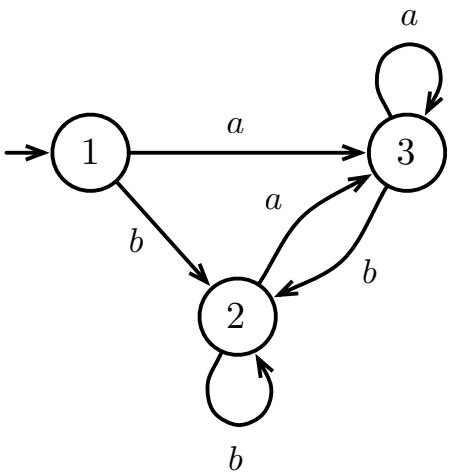


Boolean Neural Network

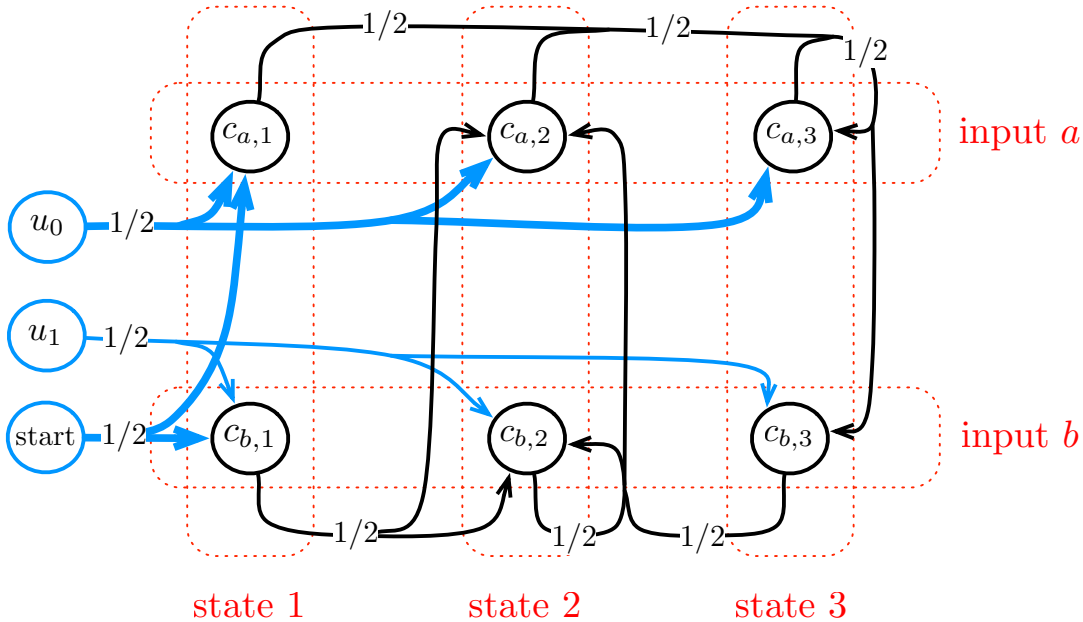


From Automata to Boolean Neural Networks

Automaton

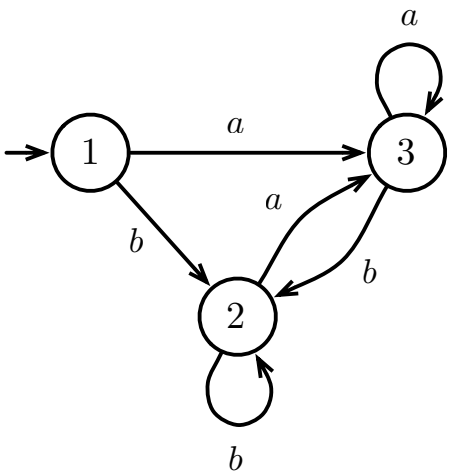


Boolean Neural Network

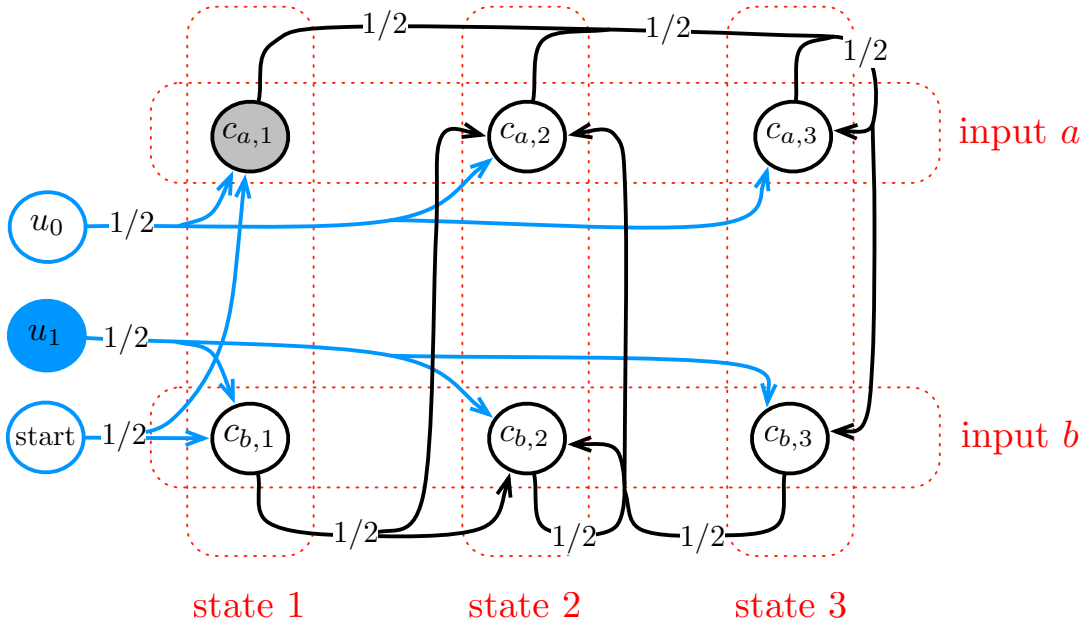


From Automata to Boolean Neural Networks

Automaton

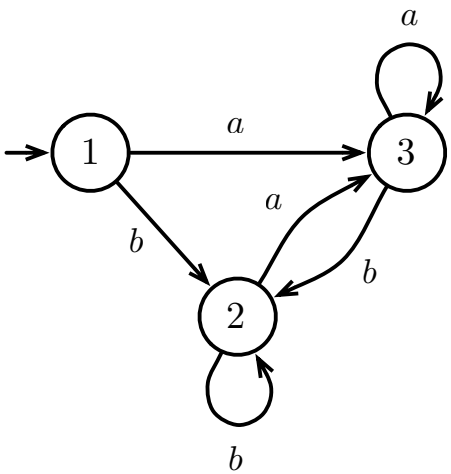


Boolean Neural Network

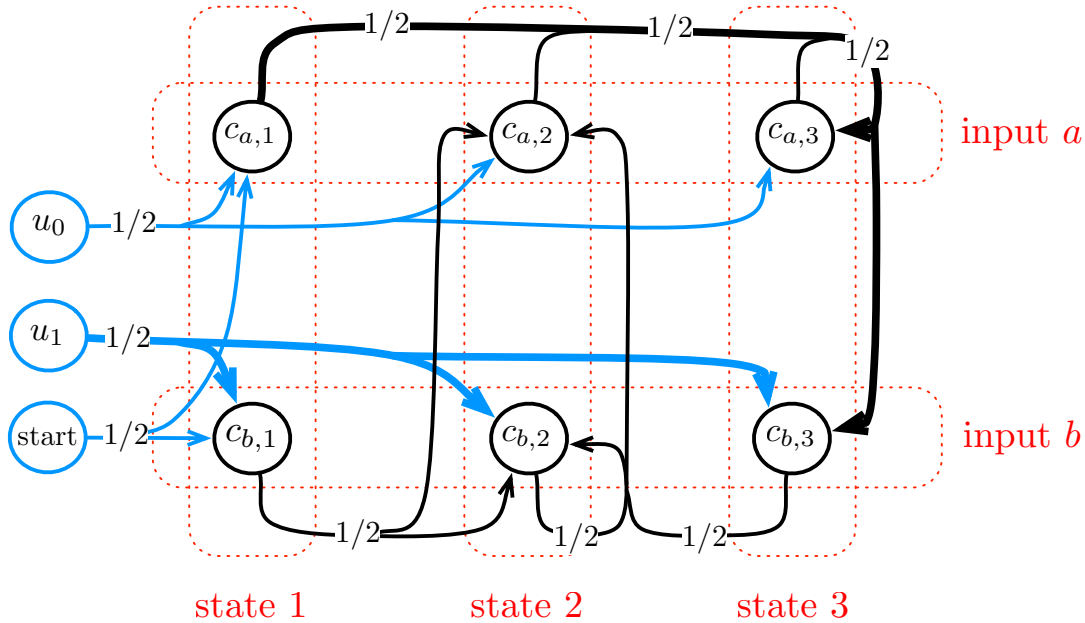


From Automata to Boolean Neural Networks

Automaton

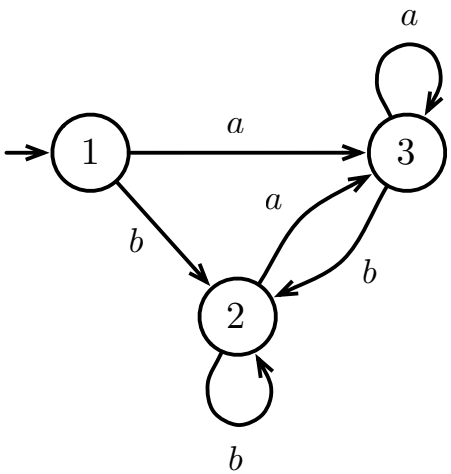


Boolean Neural Network

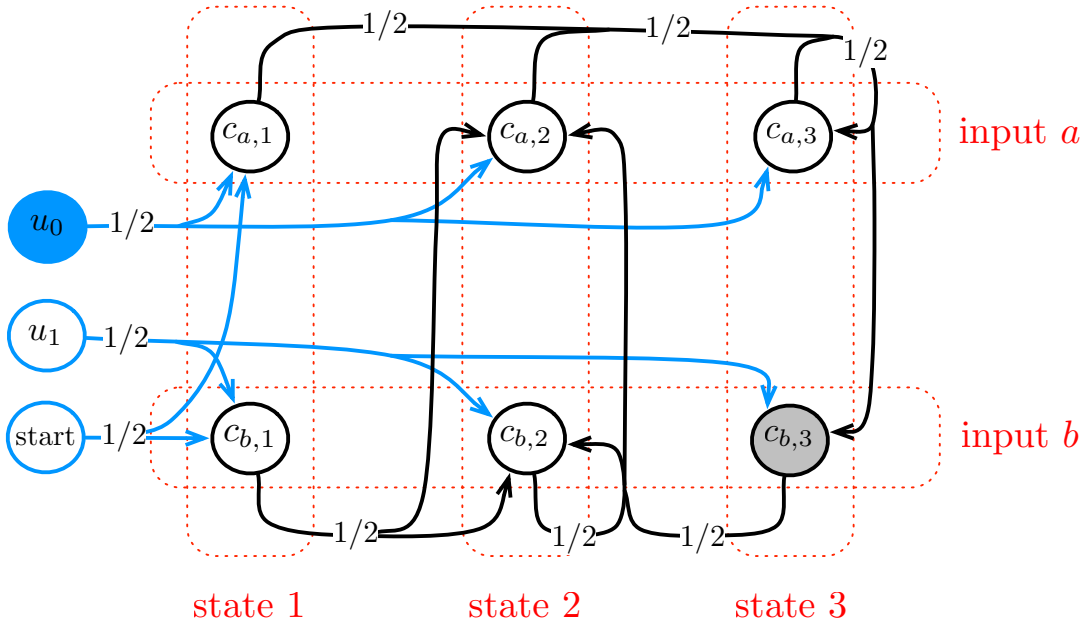


From Automata to Boolean Neural Networks

Automaton

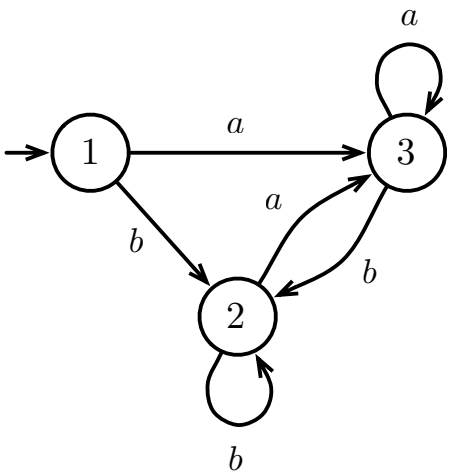


Boolean Neural Network

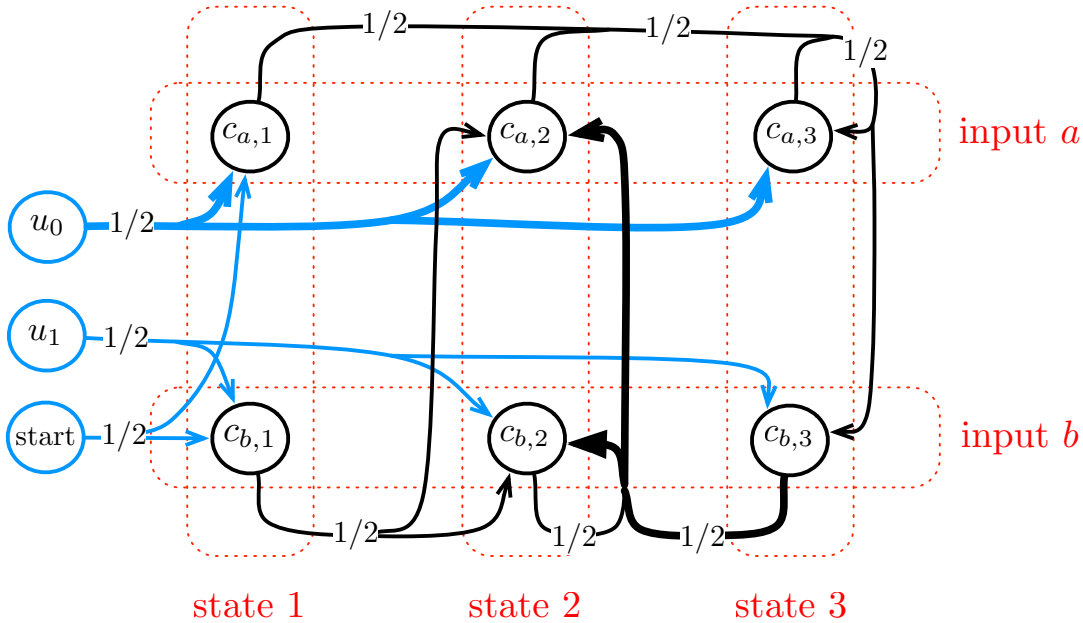


From Automata to Boolean Neural Networks

Automaton

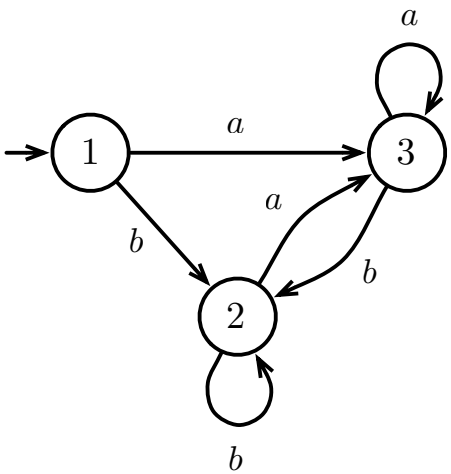


Boolean Neural Network

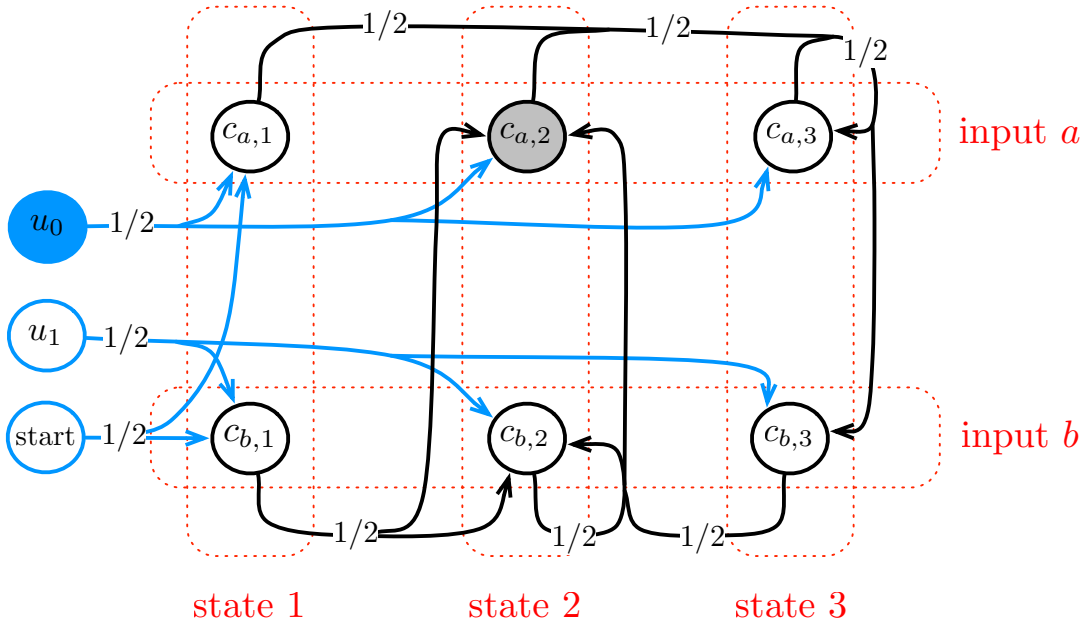


From Automata to Boolean Neural Networks

Automaton

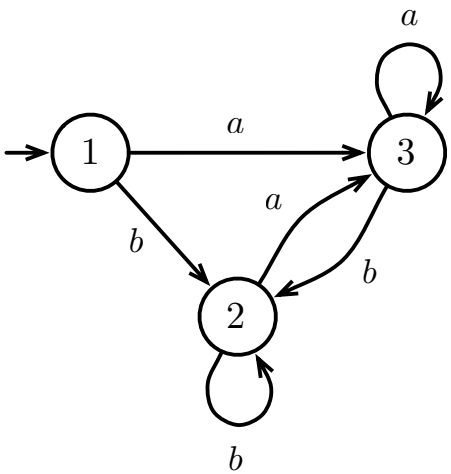


Boolean Neural Network

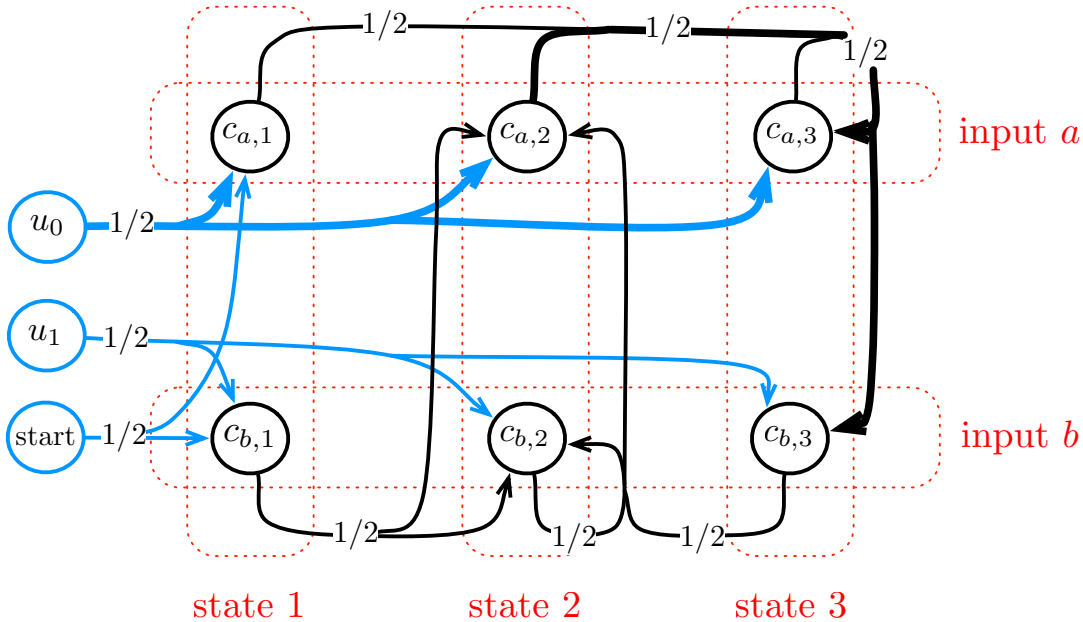


From Automata to Boolean Neural Networks

Automaton

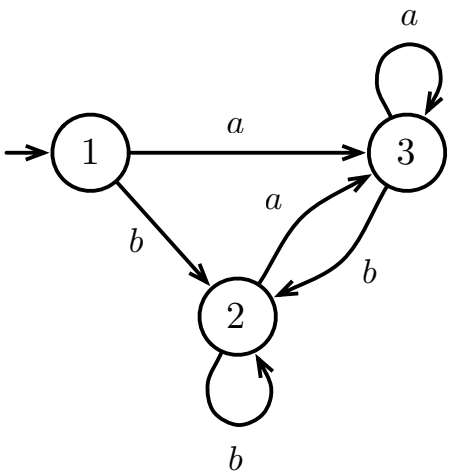


Boolean Neural Network

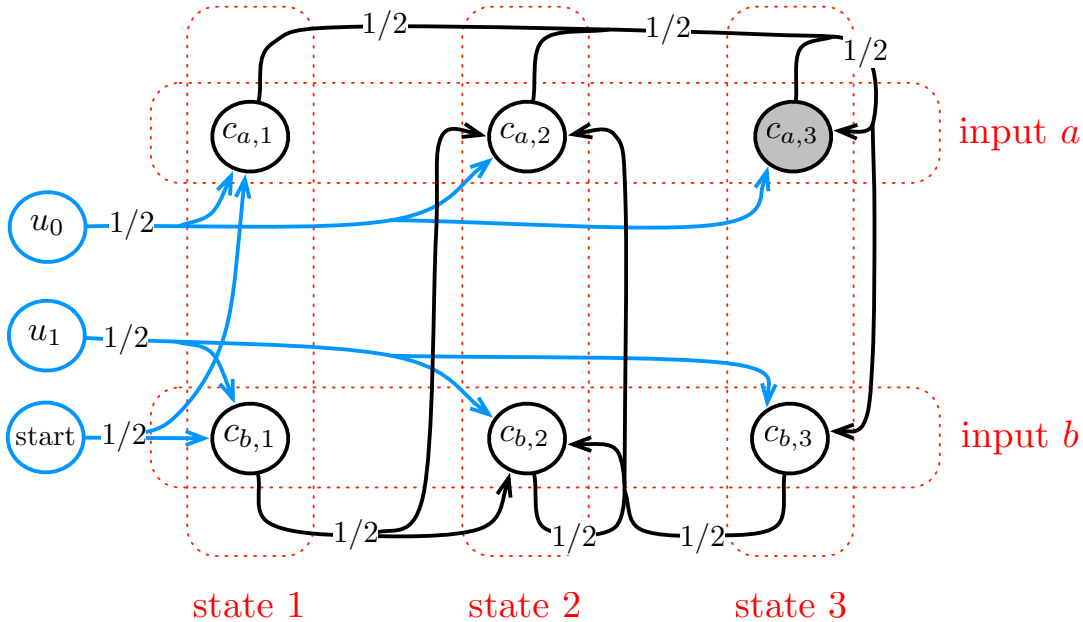


From Automata to Boolean Neural Networks

Automaton

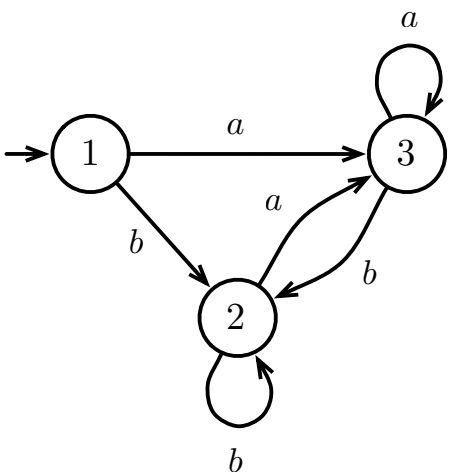


Boolean Neural Network

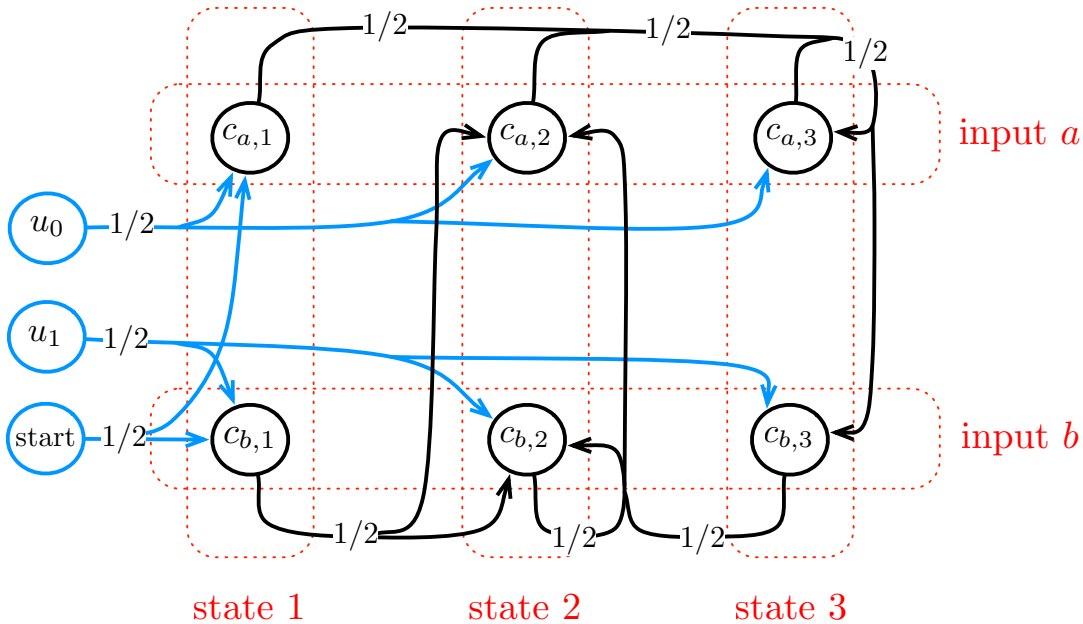


From Automata to Boolean Neural Networks

Automaton



Boolean Neural Network



$$1 \xrightarrow{a} 3 \xrightarrow{b} 2 \xrightarrow{a} 3 \xrightarrow{a} \dots \quad \_ \xrightarrow{(1,0)} c_{a,1} \xrightarrow{(0,1)} c_{b,3} \xrightarrow{(1,0)} c_{a,2} \xrightarrow{(1,0)} c_{a,3} \dots$$

Equivalence between Automata and Boolean Neural Networks

Theorem (Minsky 67)

"It is evident that each neural network of the kind we have been considering is a finite-state machine."

"[...] It is interesting and even surprising that there is a converse to this. Every finite-state machine is equivalent to, and can be "simulated" by, some neural net."

Equivalence between Automata and Boolean Neural Networks

Theorem (Minsky 67)

"It is evident that each neural network of the kind we have been considering is a finite-state machine."

"[...] It is interesting and even surprising that there is a converse to this. Every finite-state machine is equivalent to, and can be "simulated" by, some neural net."

Equivalence between Automata and Boolean Neural Networks

Theorem (Minsky 67)

"It is evident that each neural network of the kind we have been considering is a finite-state machine."

"[...] It is interesting and even surprising that there is a converse to this. Every finite-state machine is equivalent to, and can be "simulated" by, some neural net."

Drawbacks of the Construction

Two main drawbacks of this construction:

- Computational states of the automaton represented by single spiking configurations (or activation vectors) of the network.
  - ⋆ Computational states should rather be represented by *sustained activity of neural assemblies*.
- Network not robust to changing of weights or removal of connections.
  - ⋆ Network should be robust to such *synaptic noises*.

Drawbacks of the Construction

Two main drawbacks of this construction:

- Computational states of the automaton represented by single spiking configurations (or activation vectors) of the network.
- ⋆ Computational states should rather be represented by *sustained activity of neural assemblies*.
- Network not robust to changing of weights or removal of connections.
- ⋆ Network should be robust to such *synaptic noises*.

Drawbacks of the Construction

Two main drawbacks of this construction:

- Computational states of the automaton represented by single spiking configurations (or activation vectors) of the network.
- ⋆ Computational states should rather be represented by *sustained activity of neural assemblies*.
- Network not robust to changing of weights or removal of connections.
- ⋆ Network should be robust to such *synaptic noises*.

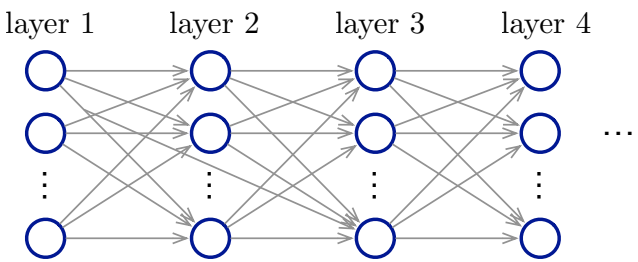
Drawbacks of the Construction

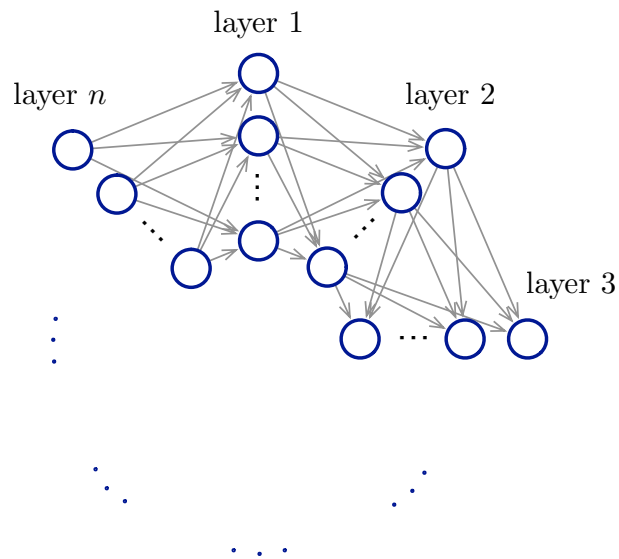
Two main drawbacks of this construction:

- Computational states of the automaton represented by single spiking configurations (or activation vectors) of the network.
- ⋆ Computational states should rather be represented by *sustained activity of neural assemblies*.
- Network not robust to changing of weights or removal of connections.
- ⋆ Network should be robust to such *synaptic noises*.

Synfire Chains and Synfire Rings

- *Synfire chains* are likely to be crucially involved in the processing and coding of information (Abeles 82).
- Spontaneous emergence of *synfire rings* (looping synfire chains) has been observed in self-organizing neural networks (Zheng & Triesch 14).
  - ⋆ *Computational states could be represented as sustained activities within synfire rings.*

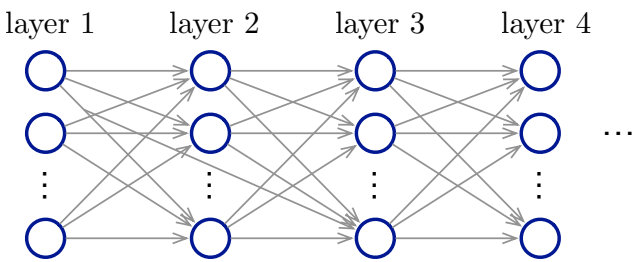


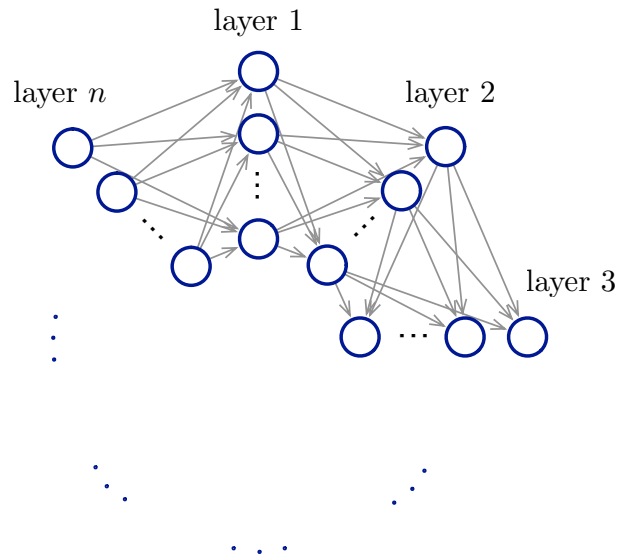


Synfire Chains and Synfire Rings

- *Synfire chains* are likely to be crucially involved in the processing and coding of information (Abeles 82).
- Spontaneous emergence of *synfire rings* (looping synfire chains) has been observed in self-organizing neural networks (Zheng & Triesch 14).

⋆ *Computational states could be represented as sustained activities within synfire rings.*

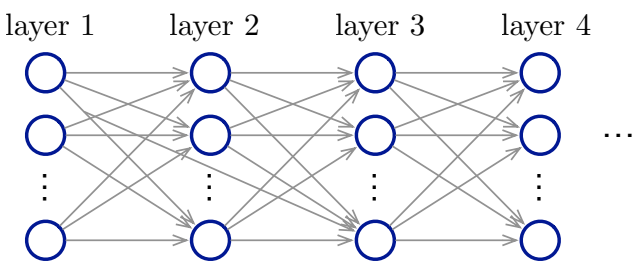


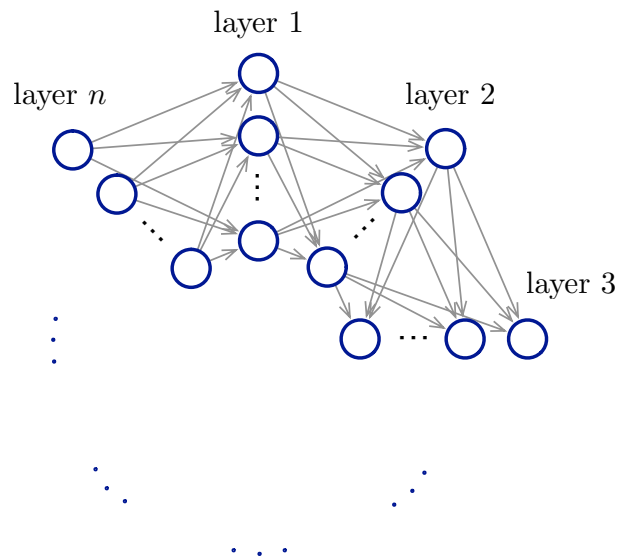


Synfire Chains and Synfire Rings

- *Synfire chains* are likely to be crucially involved in the processing and coding of information (Abeles 82).
- Spontaneous emergence of *synfire rings* (looping synfire chains) has been observed in self-organizing neural networks (Zheng & Triesch 14).

⋆ *Computational states could be represented as sustained activities within synfire rings.*

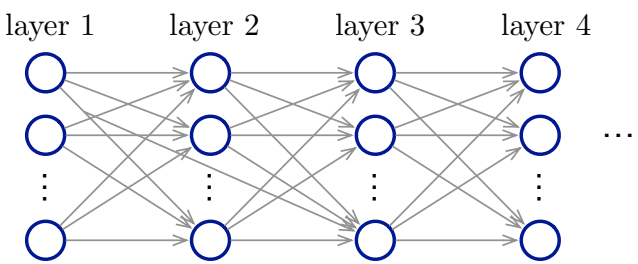


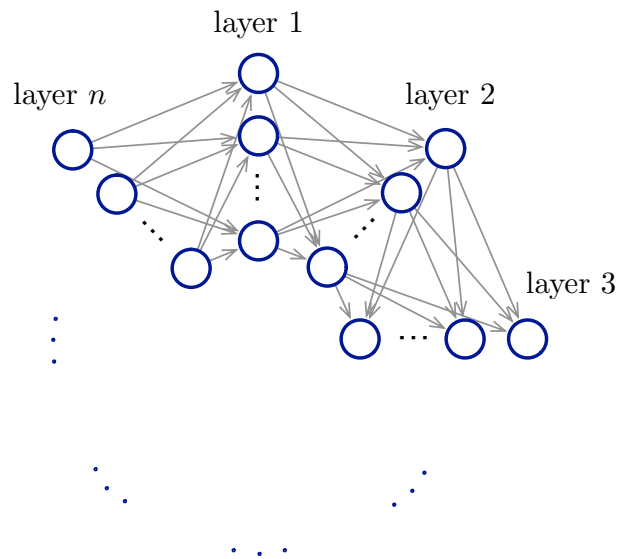


Synfire Chains and Synfire Rings

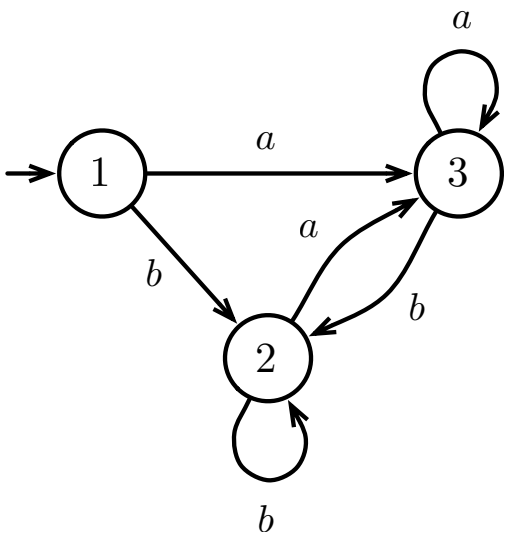
- *Synfire chains* are likely to be crucially involved in the processing and coding of information (Abeles 82).
- Spontaneous emergence of *synfire rings* (looping synfire chains) has been observed in self-organizing neural networks (Zheng & Triesch 14).

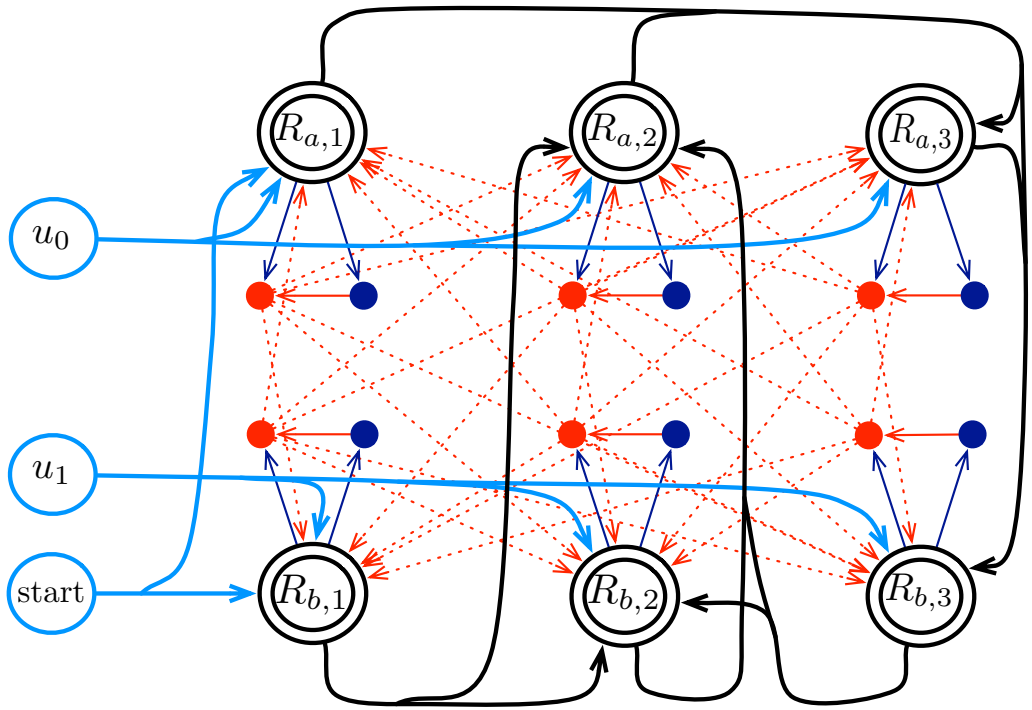
⋆ *Computational states could be represented as sustained activities within synfire rings.*



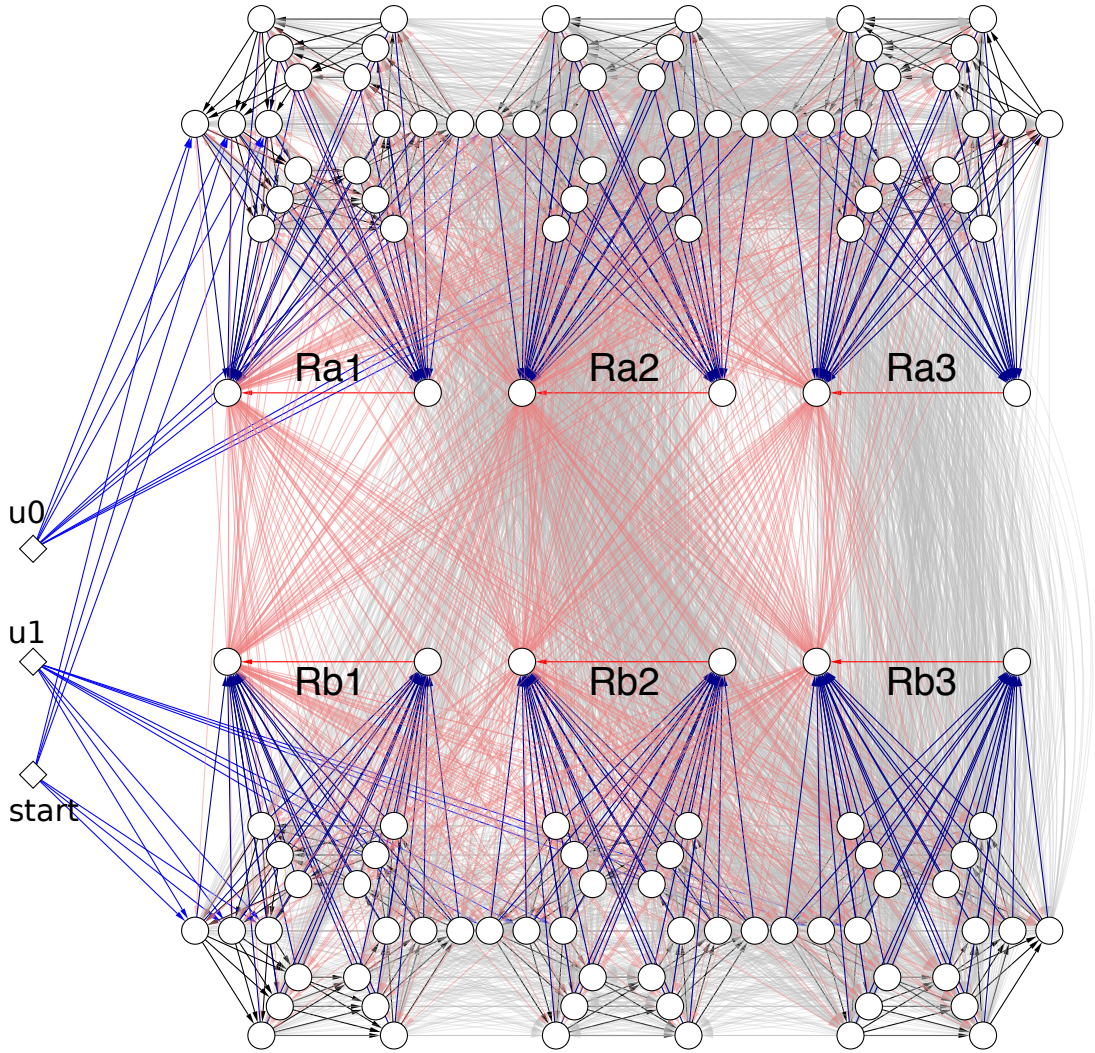


From Automata to Boolean Neural Networks with Synfire Rings

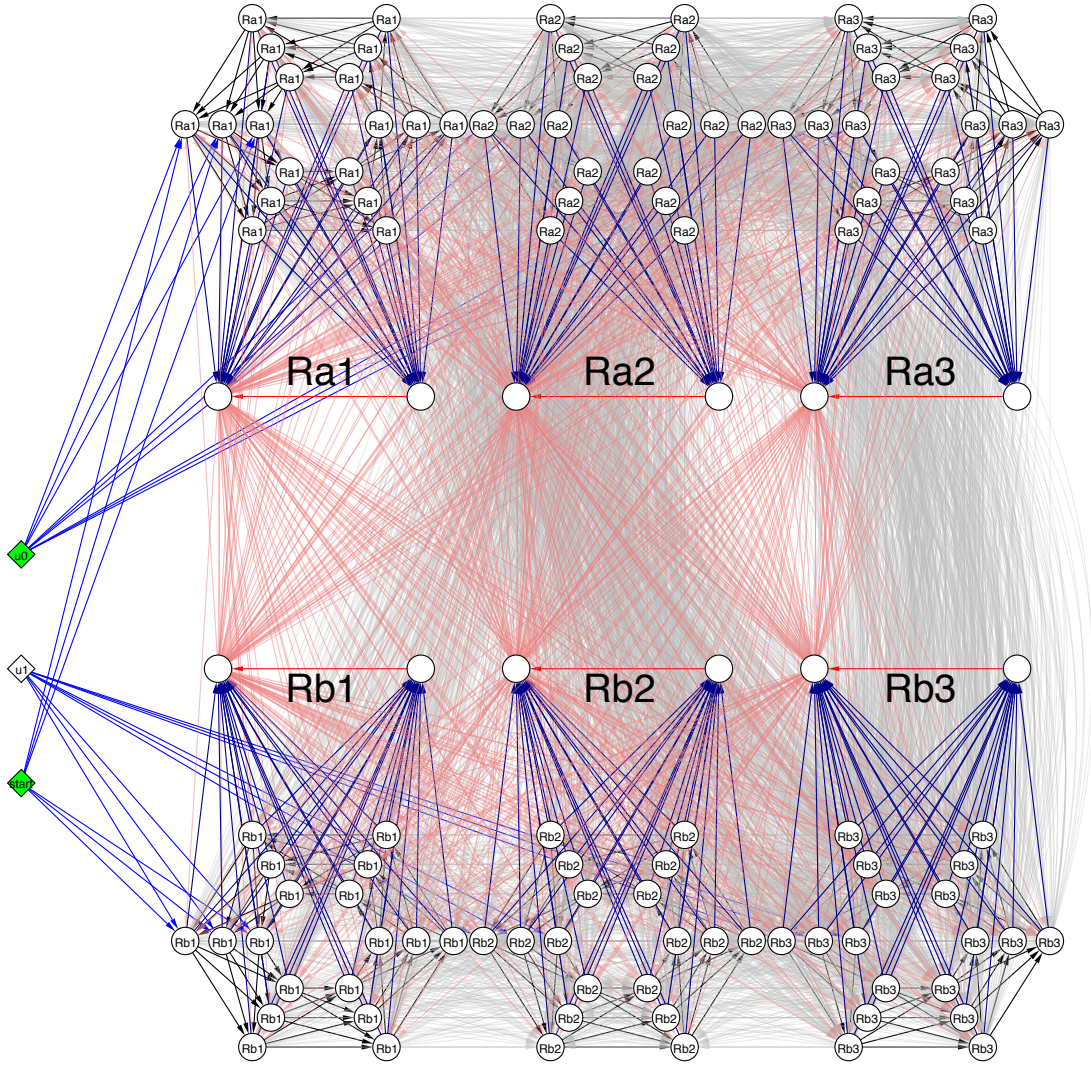




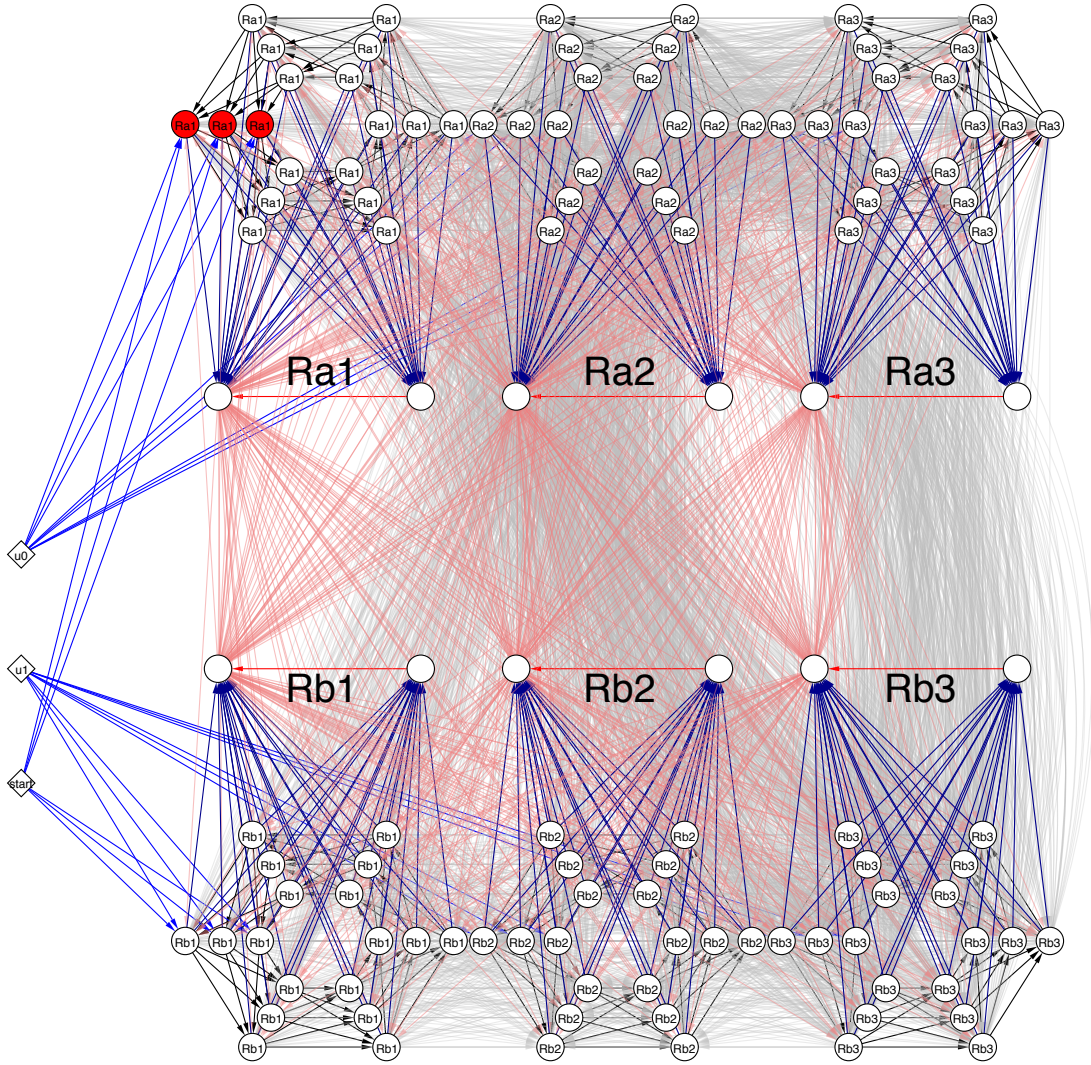
Simulation



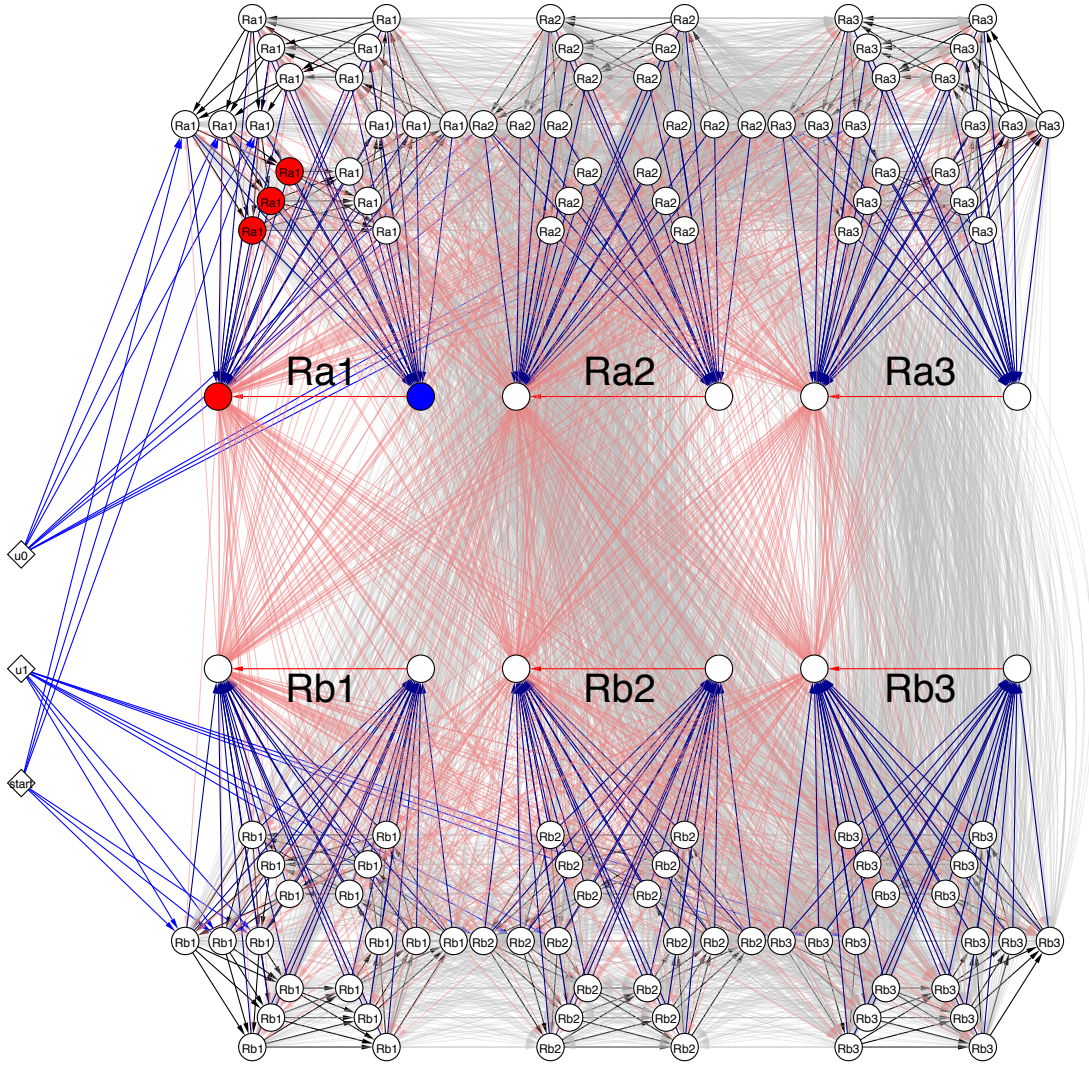
Simulation



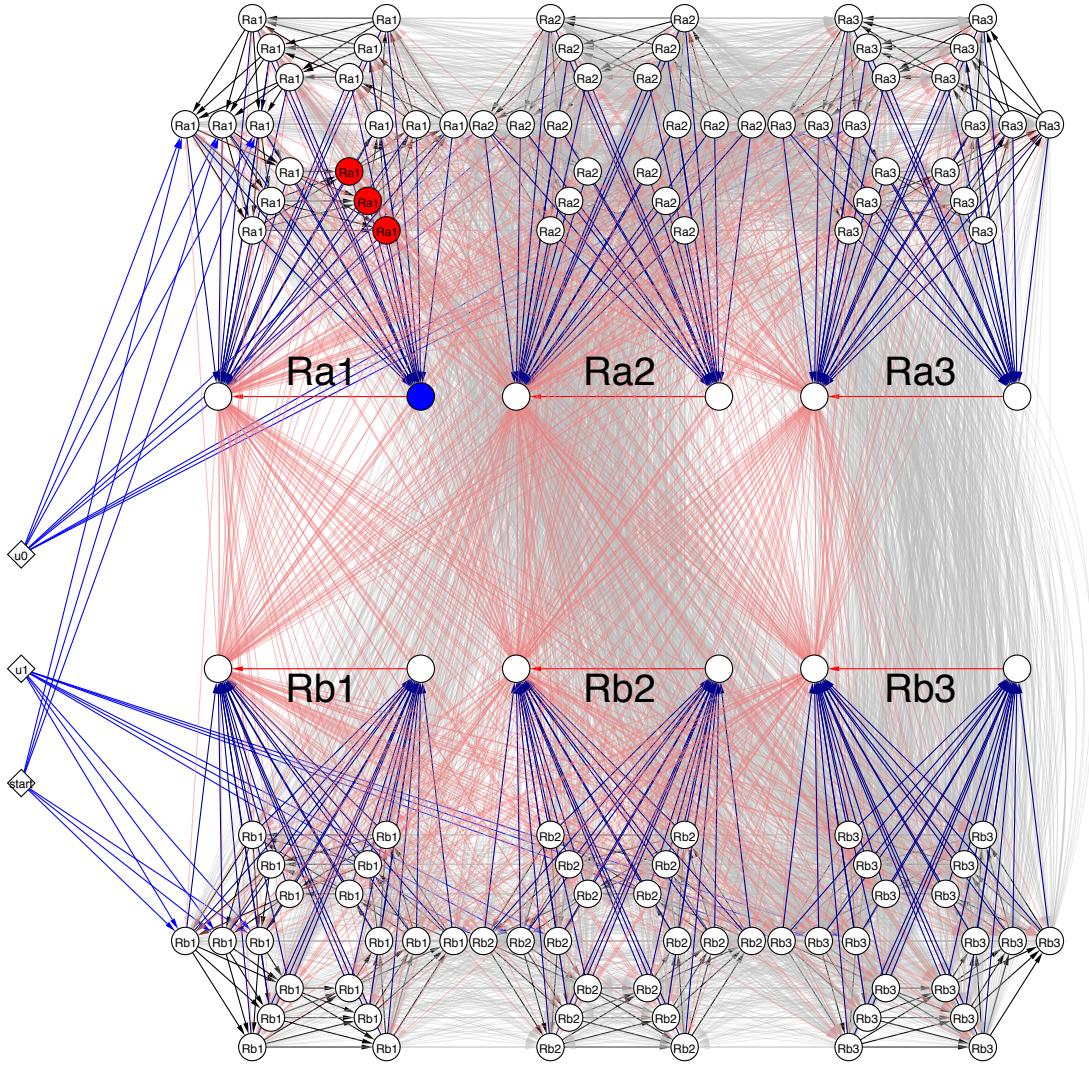
Simulation



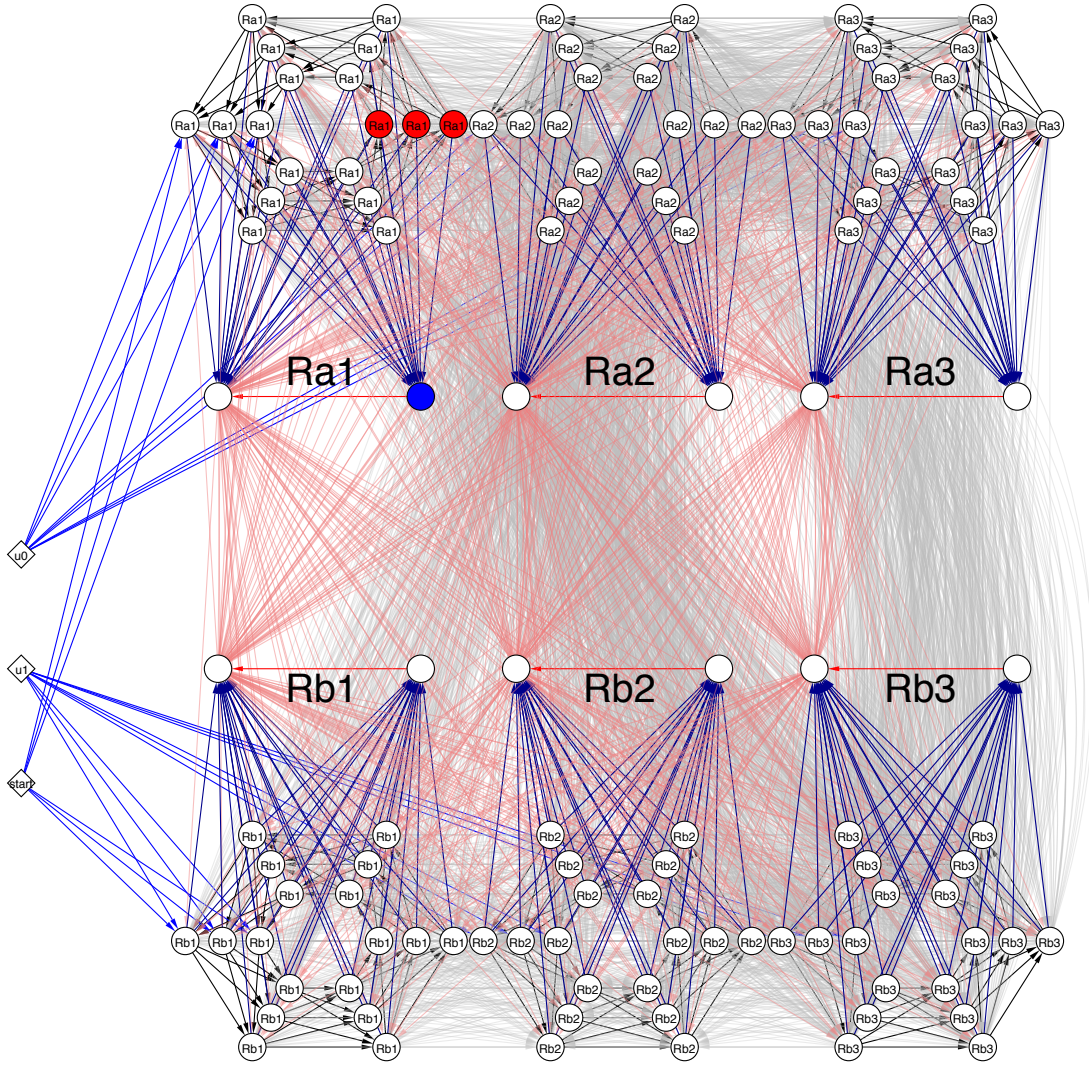
Simulation



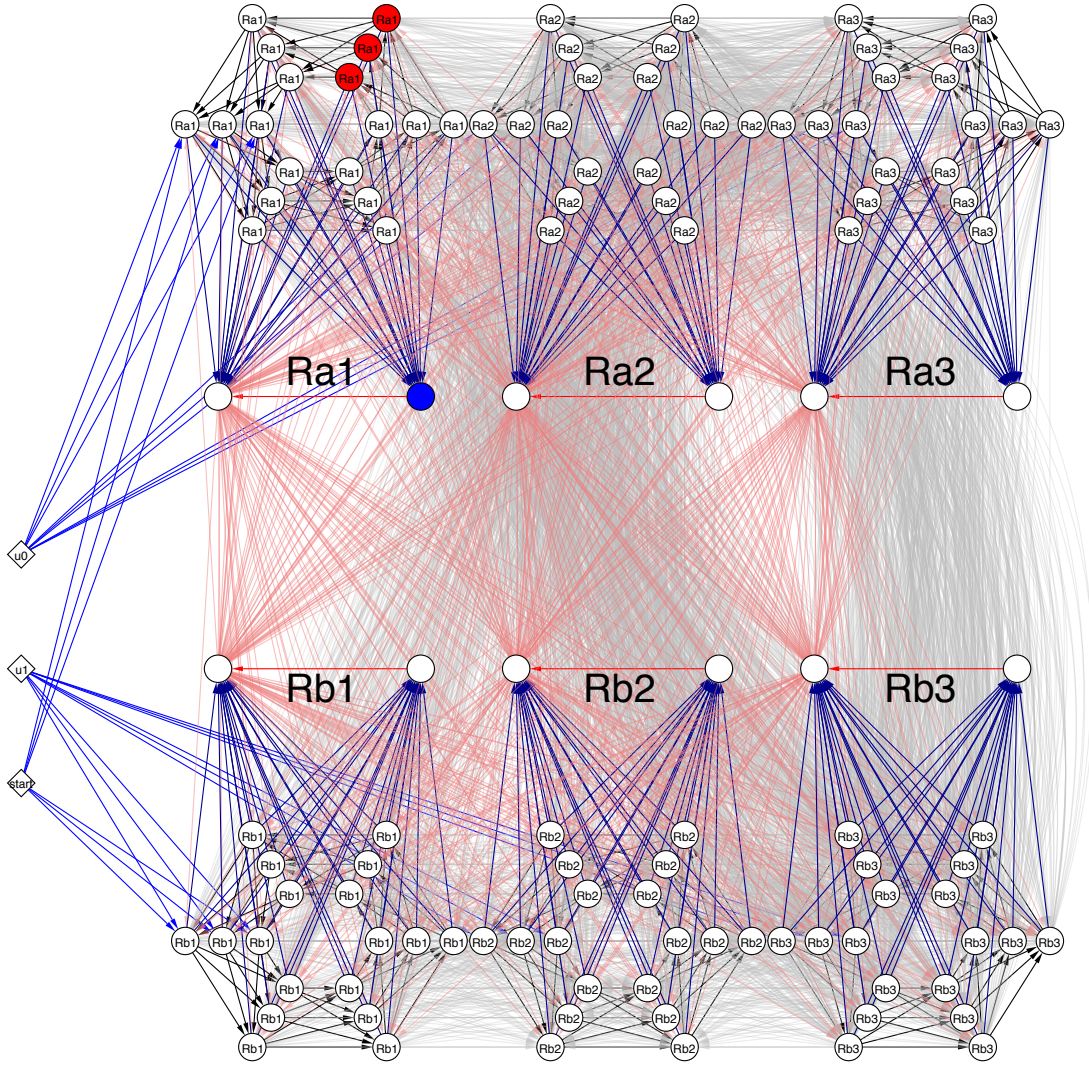
Simulation



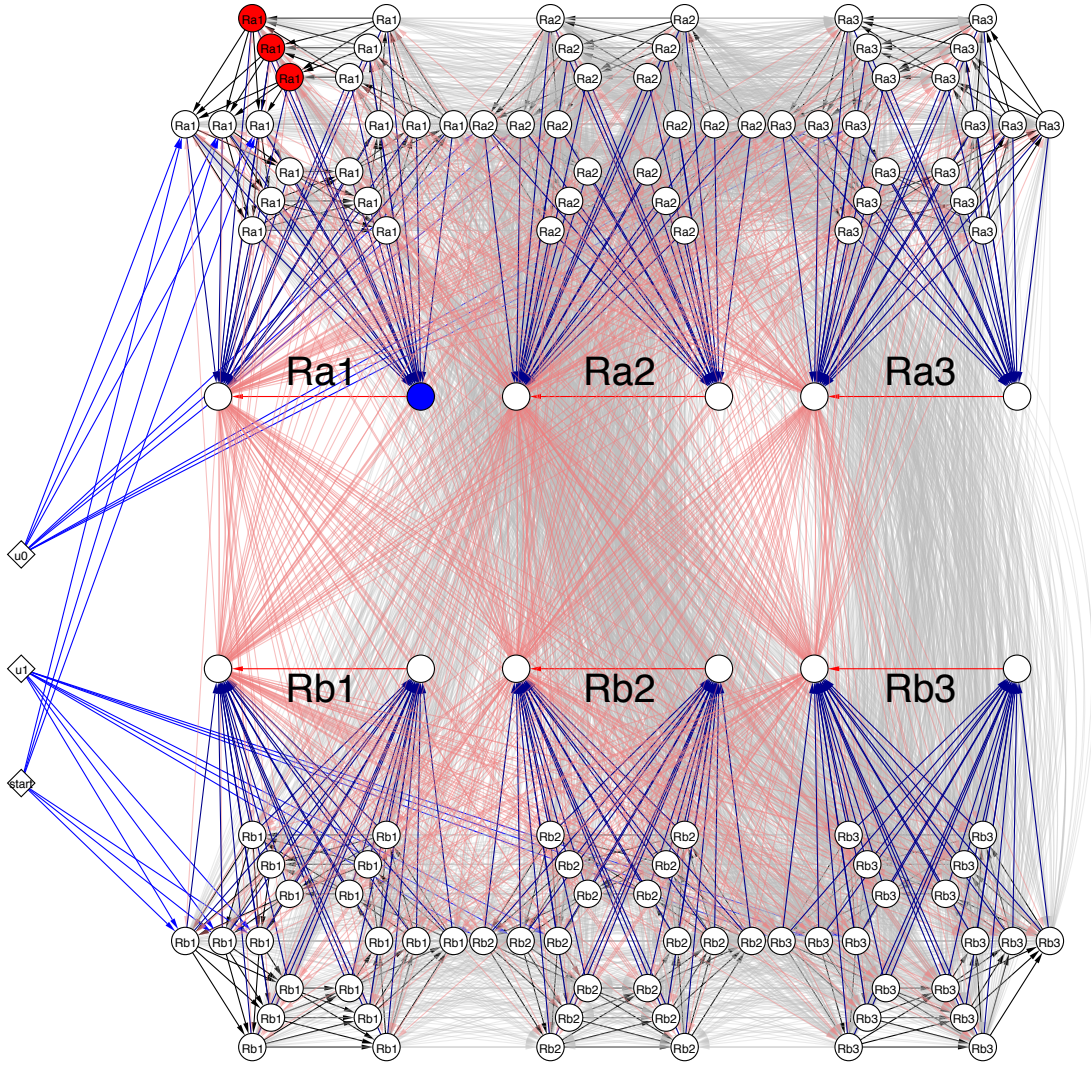
Simulation



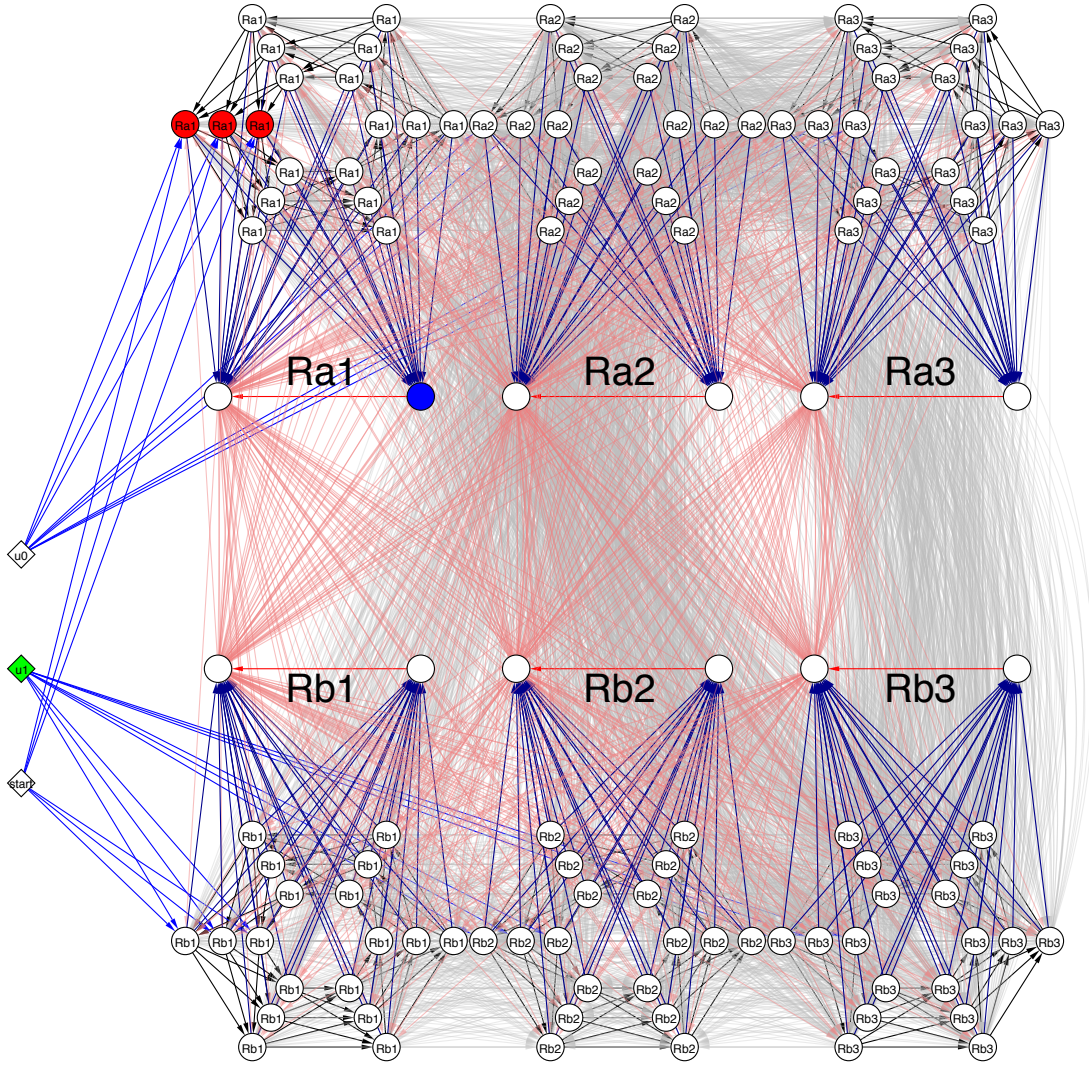
Simulation



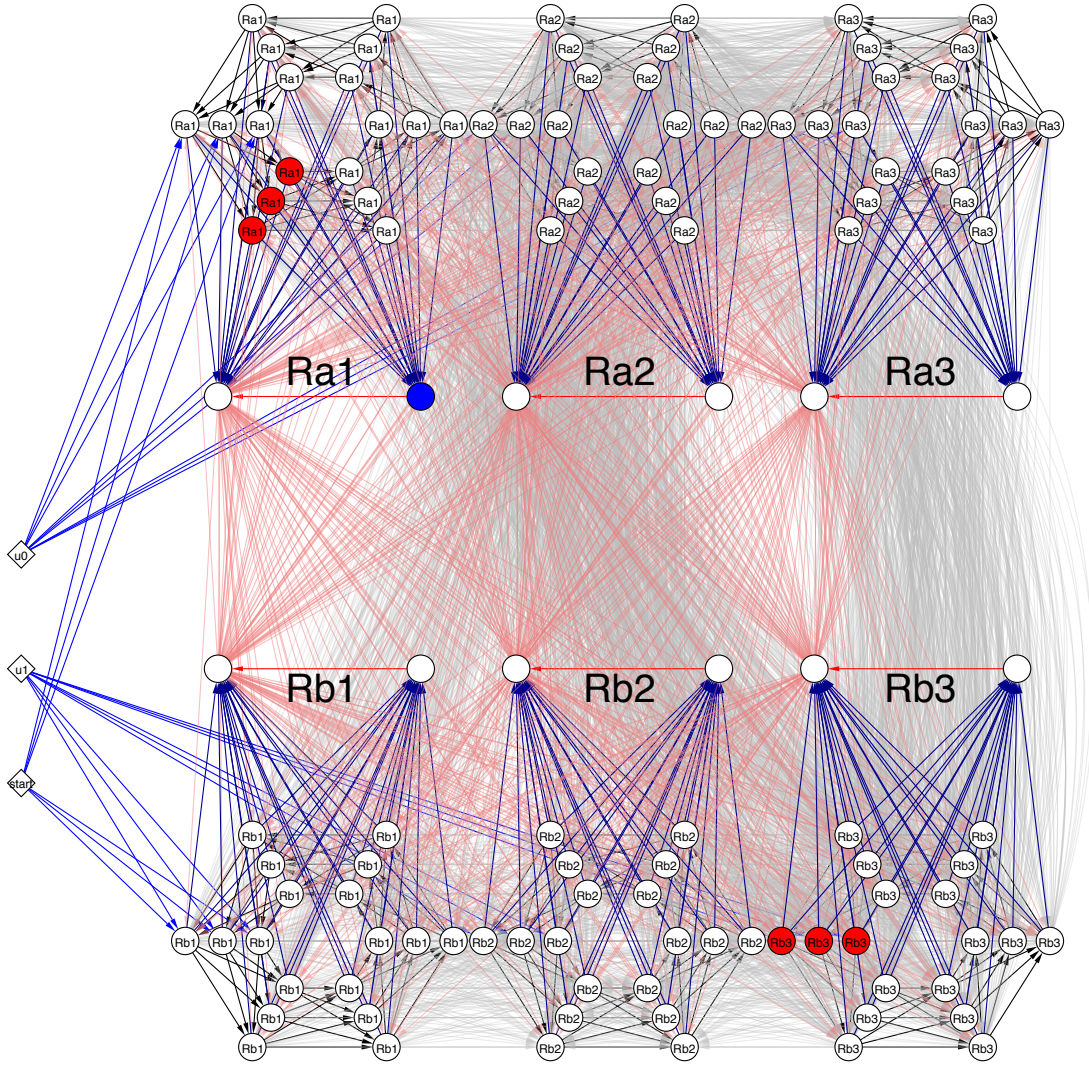
Simulation



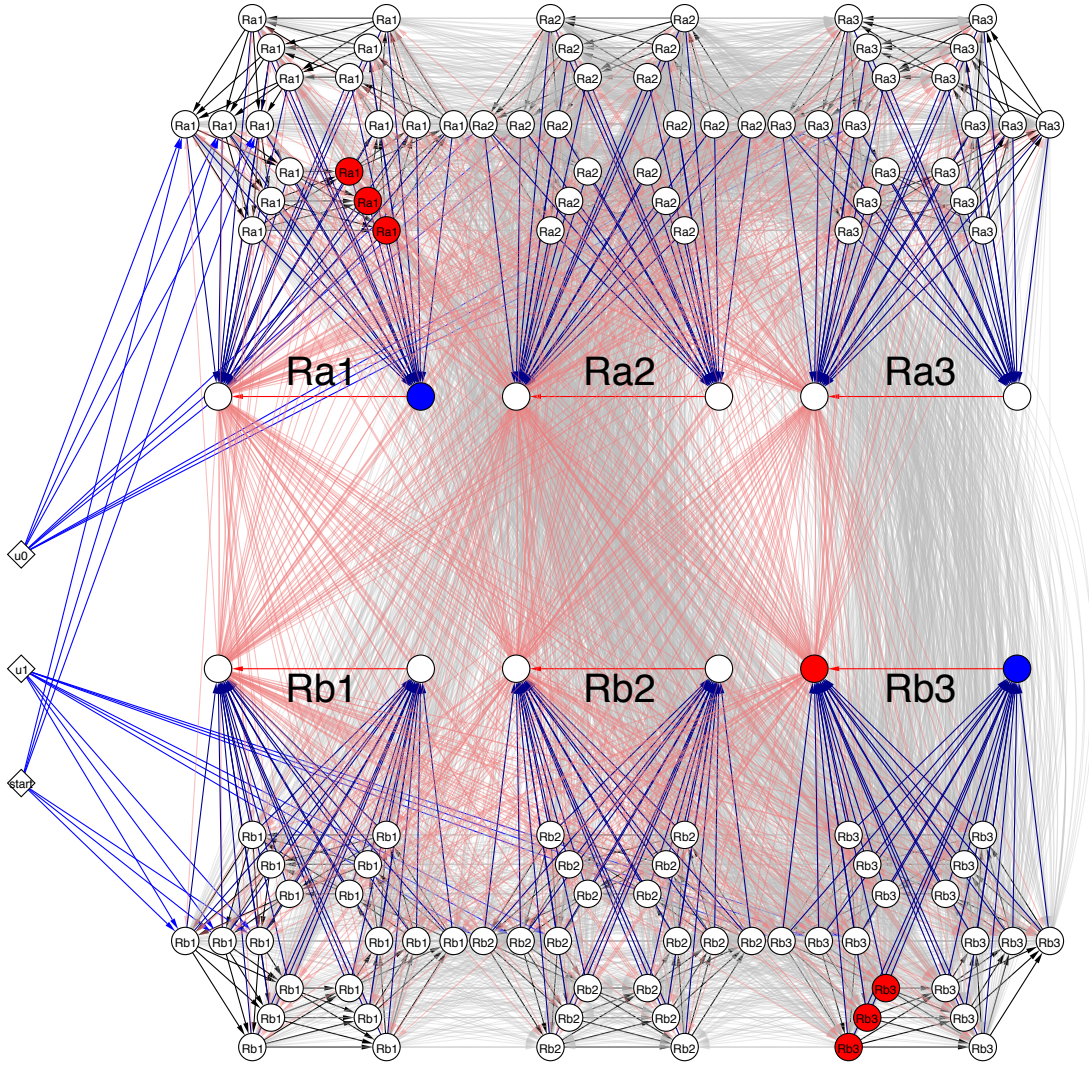
Simulation



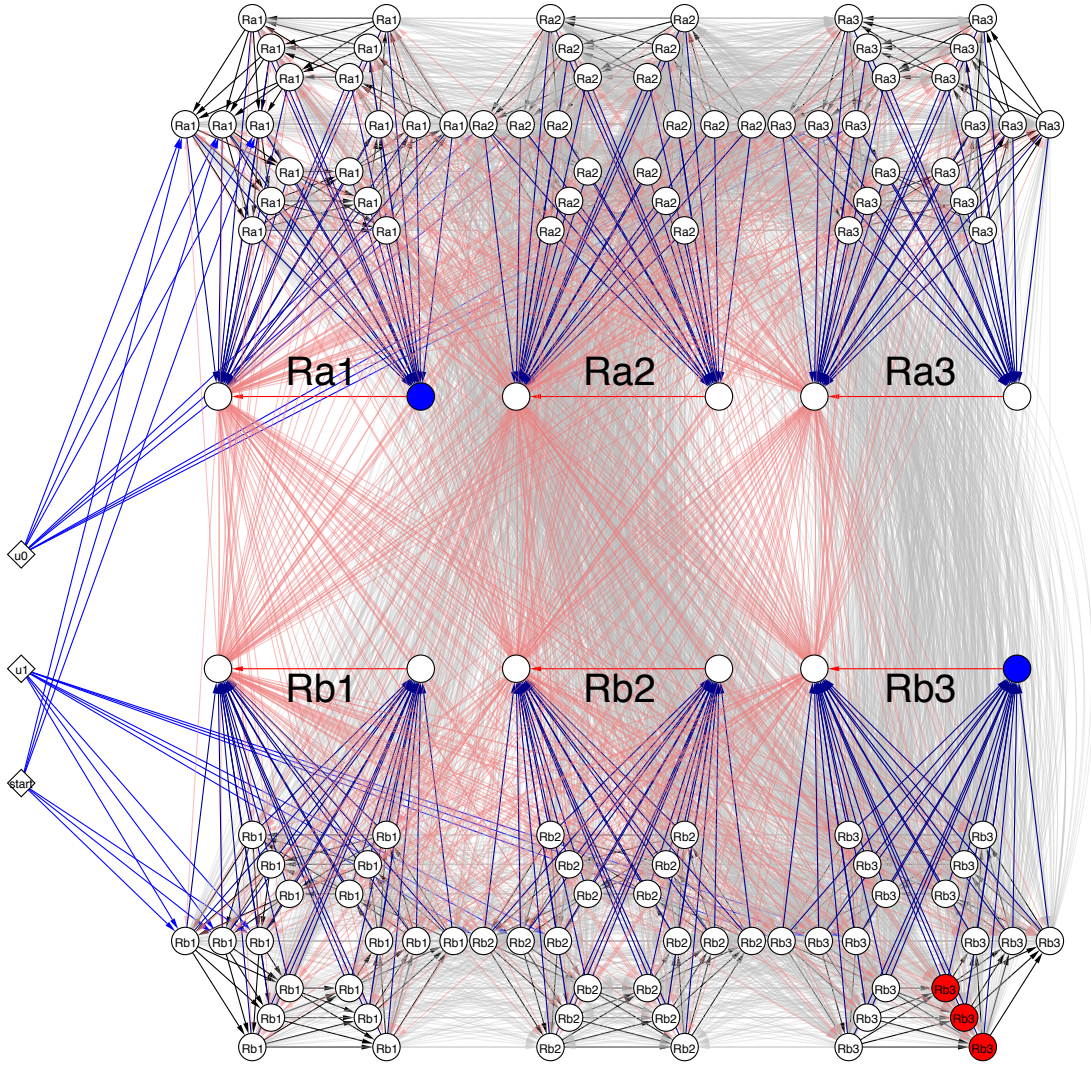
Simulation



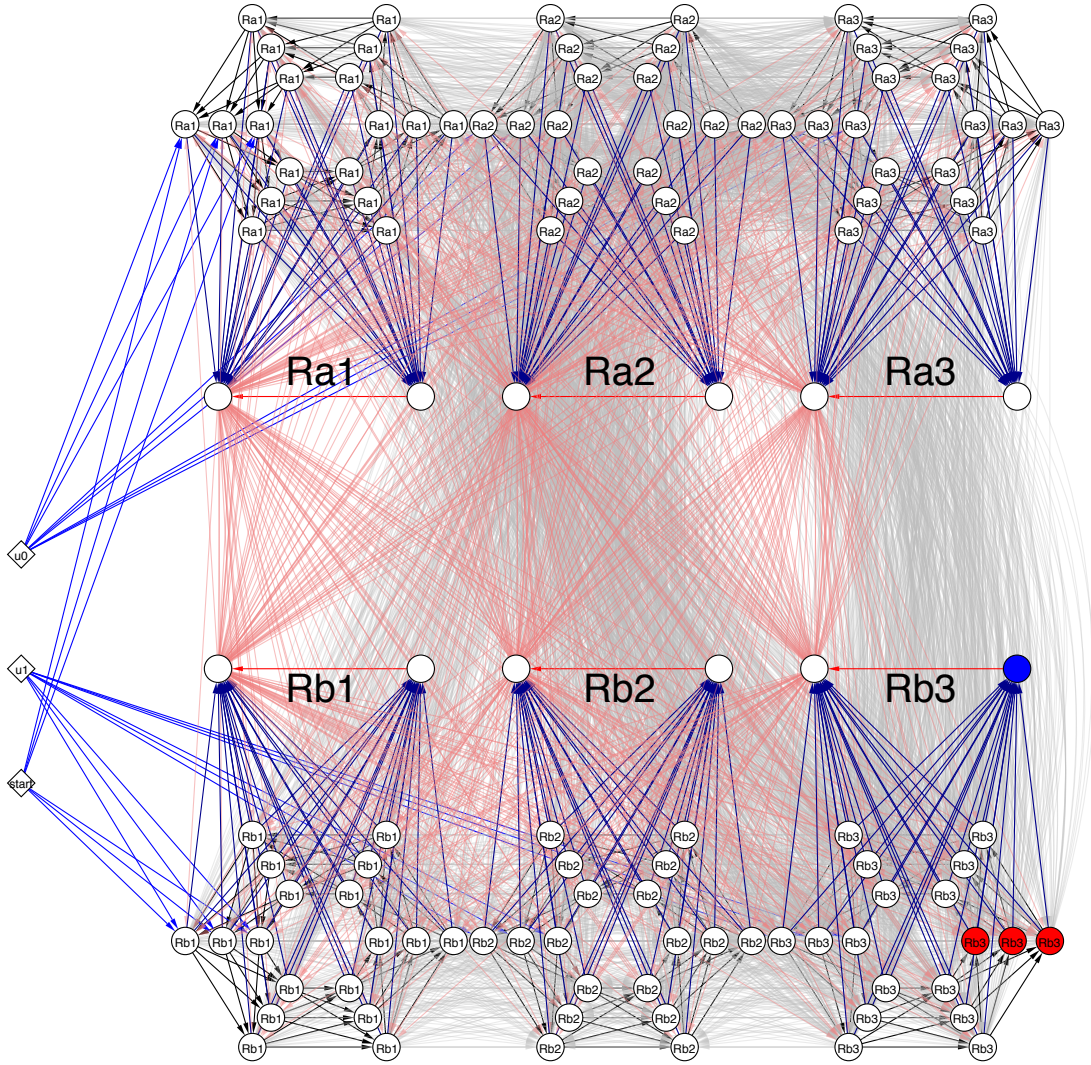
Simulation



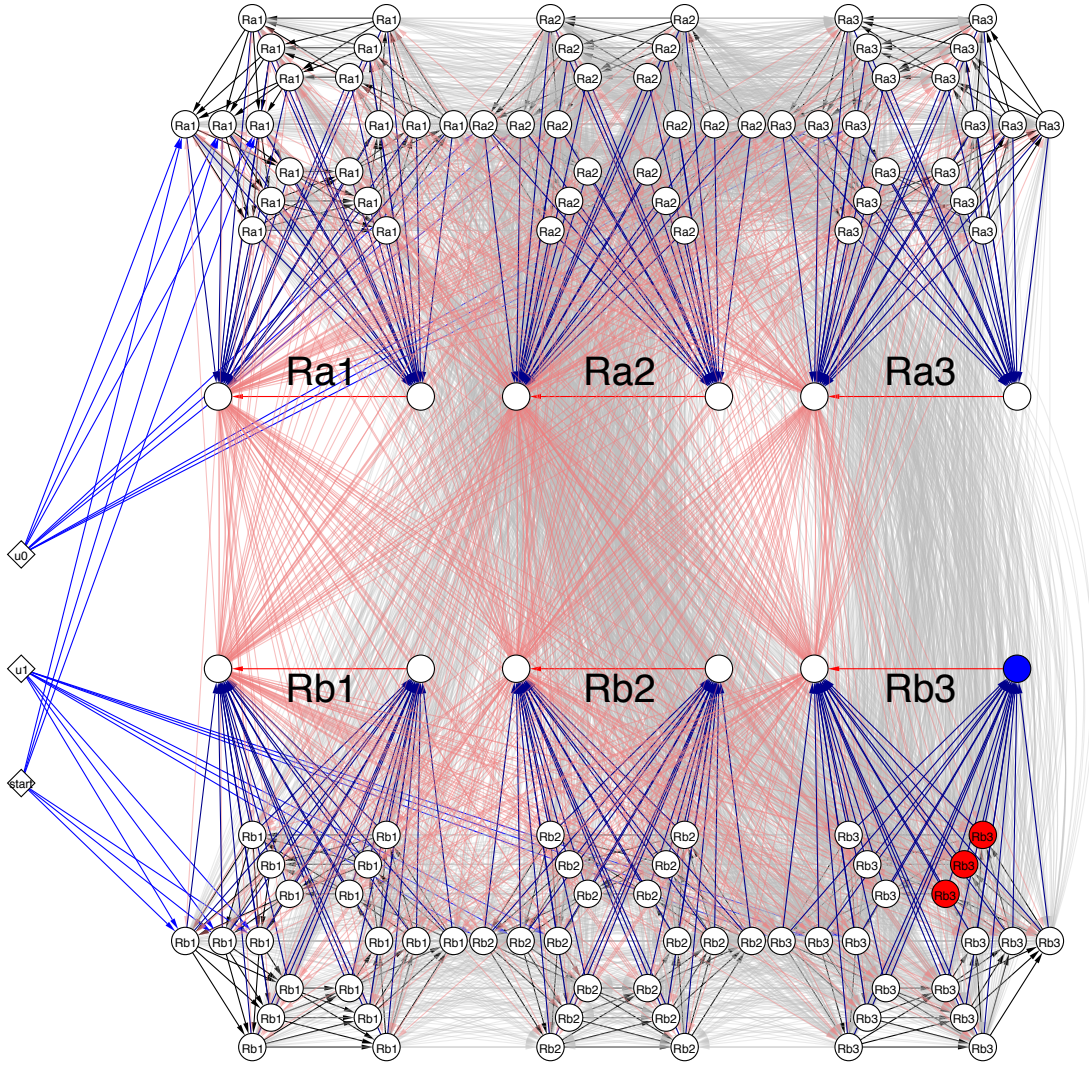
Simulation



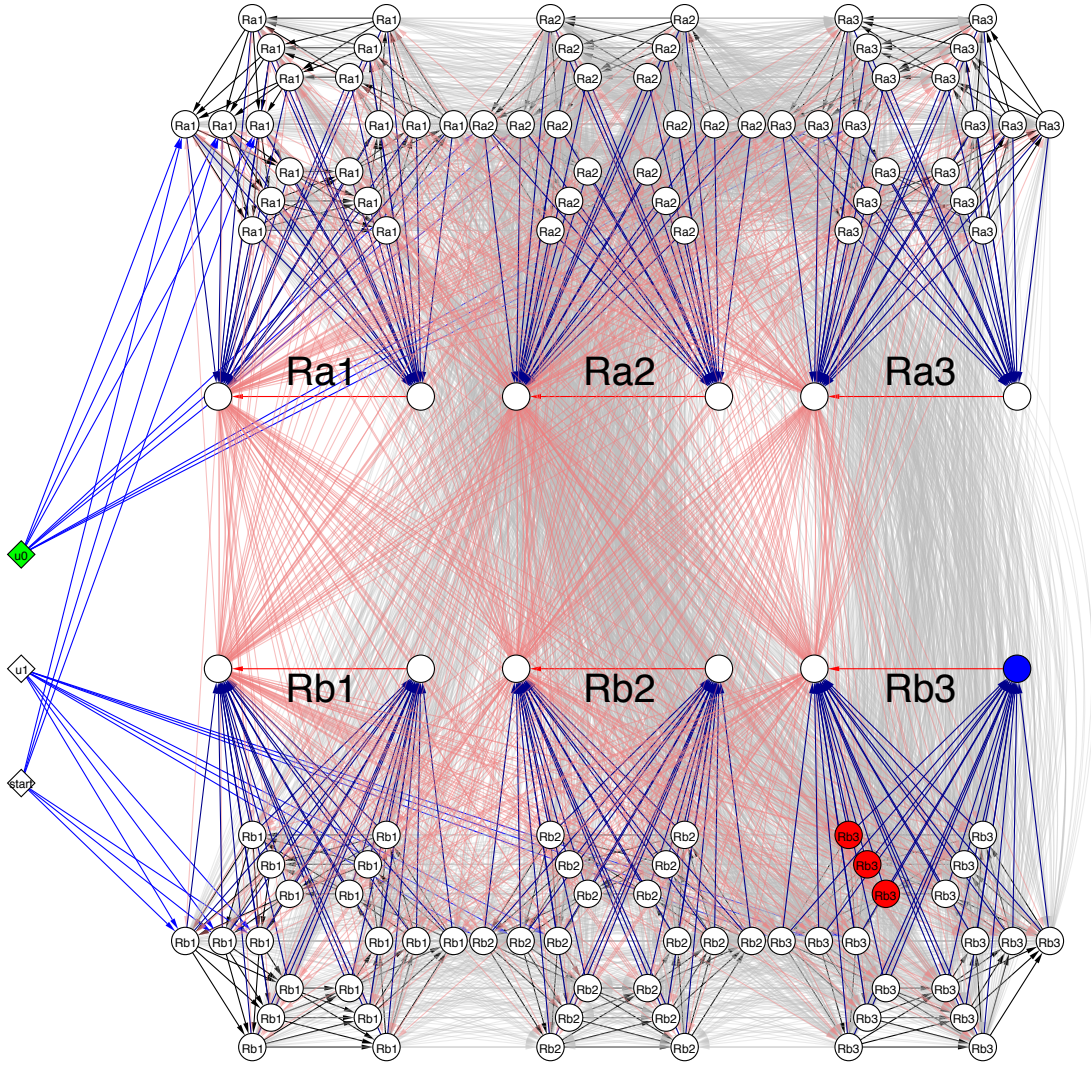
Simulation



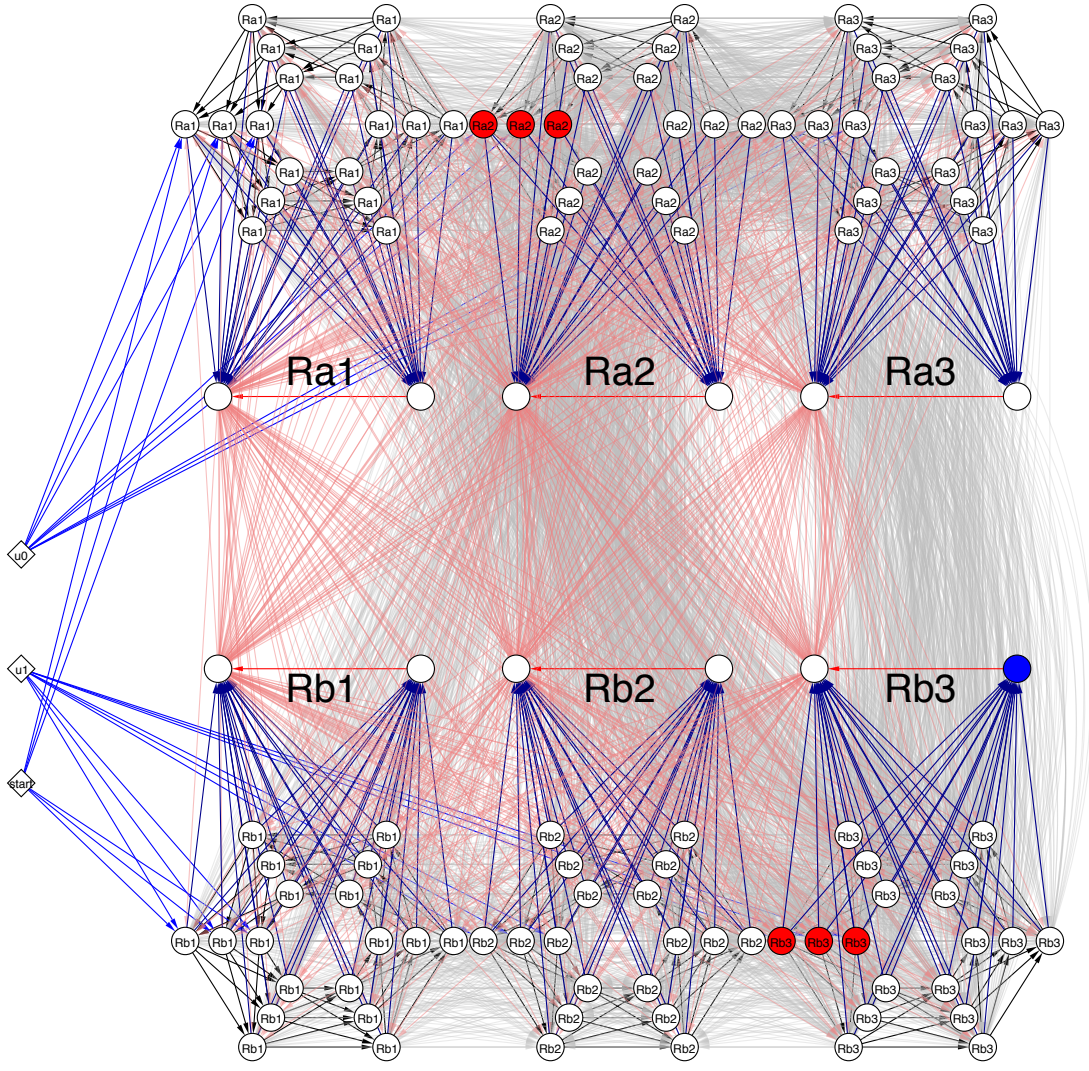
Simulation



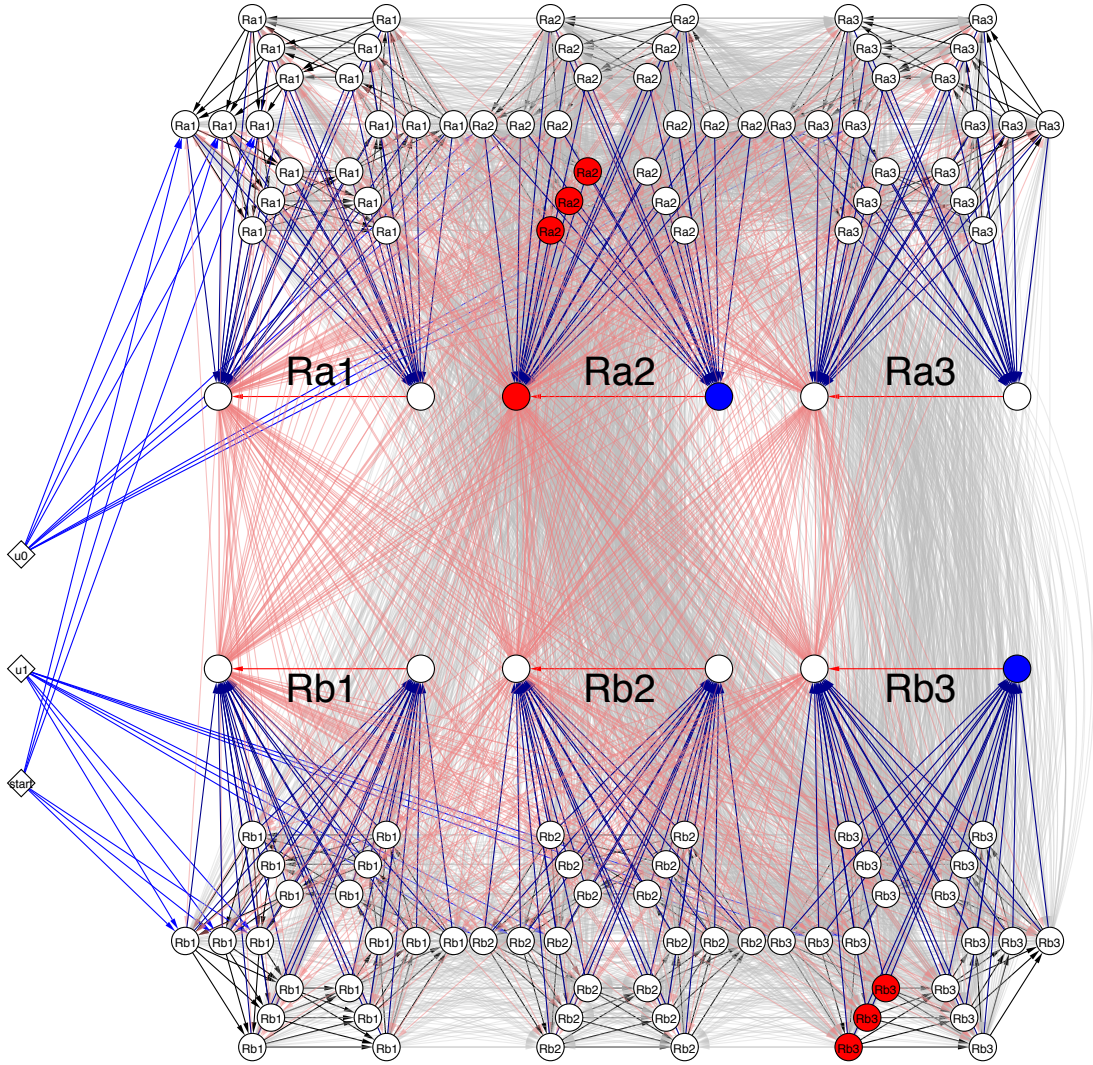
Simulation



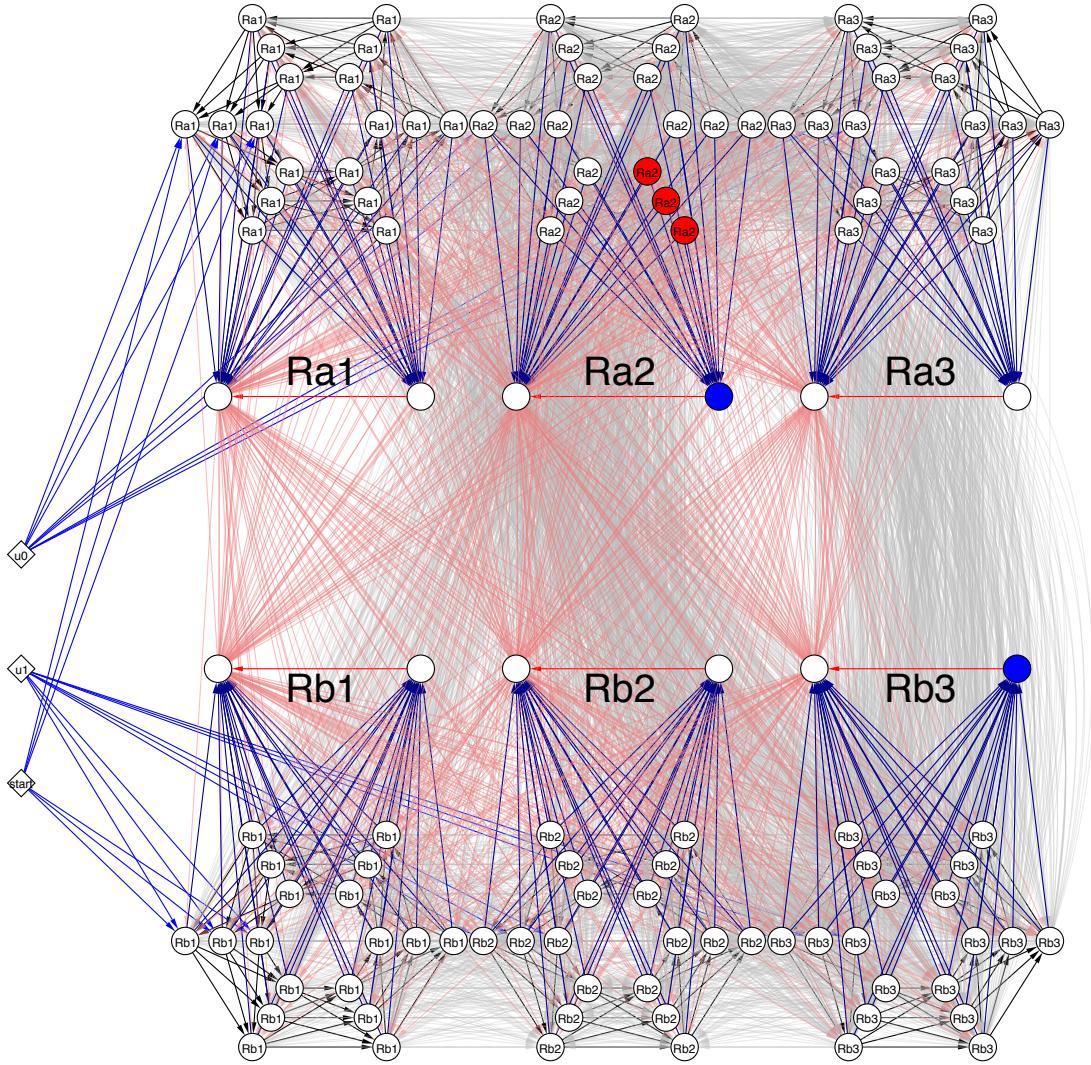
Simulation



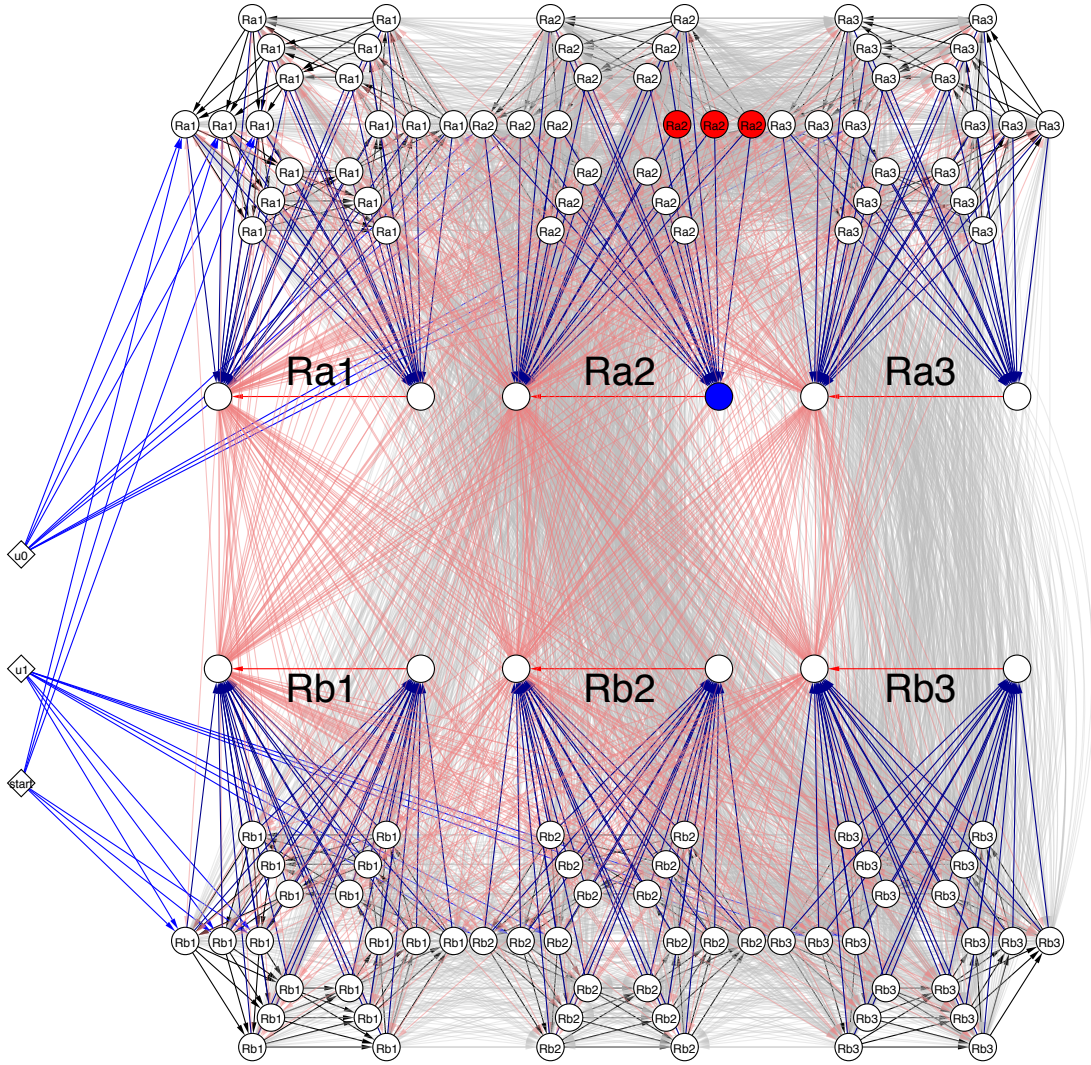
Simulation



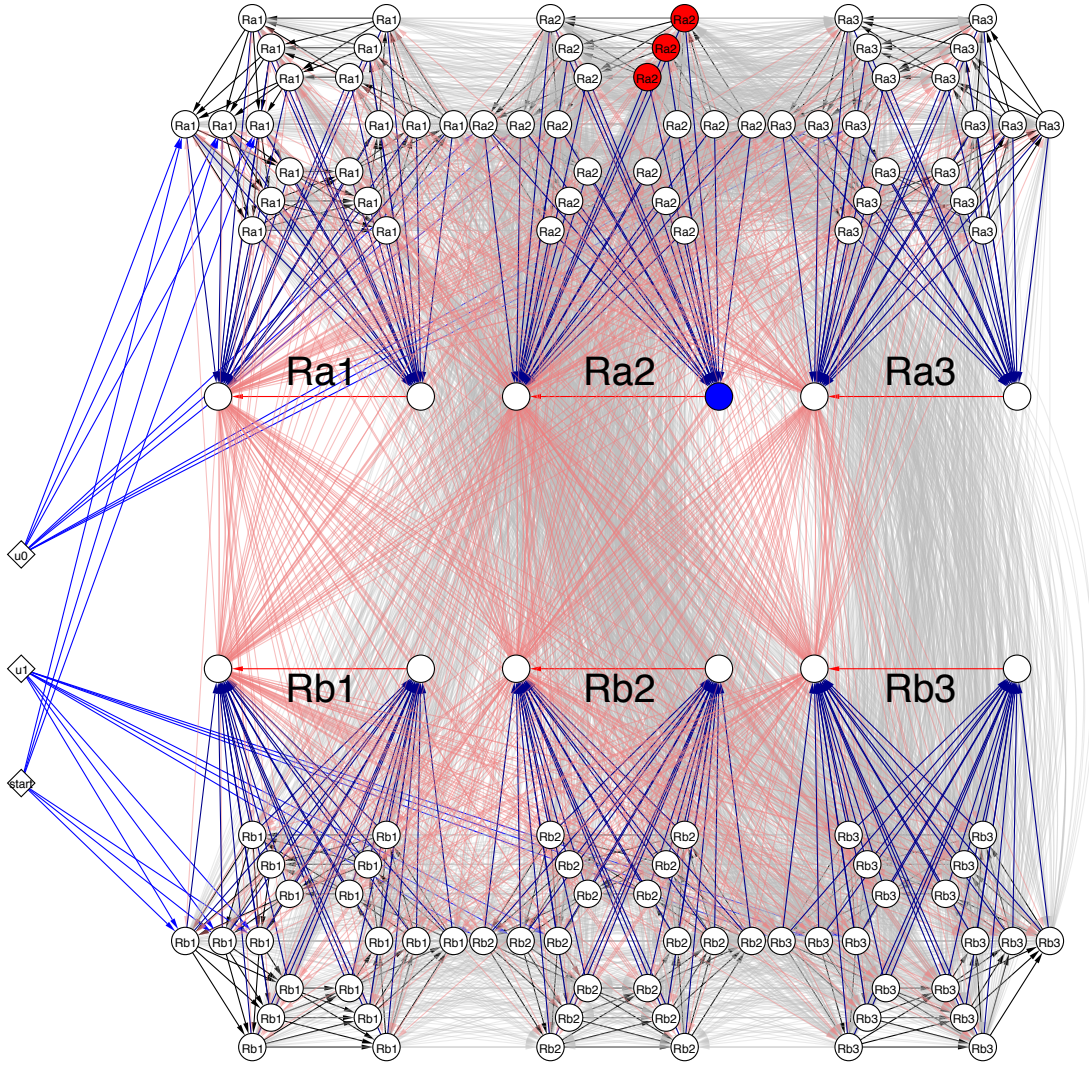
Simulation



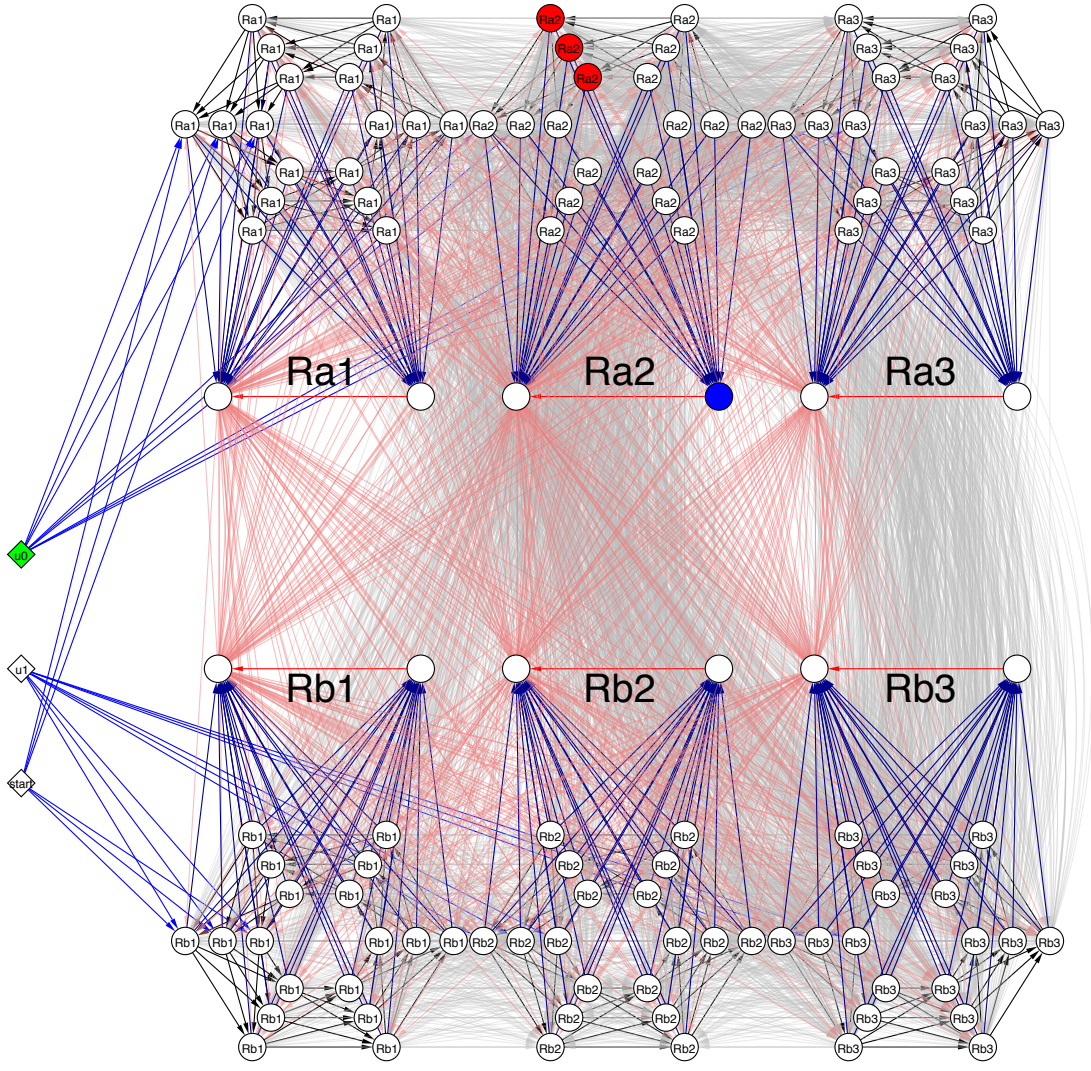
Simulation



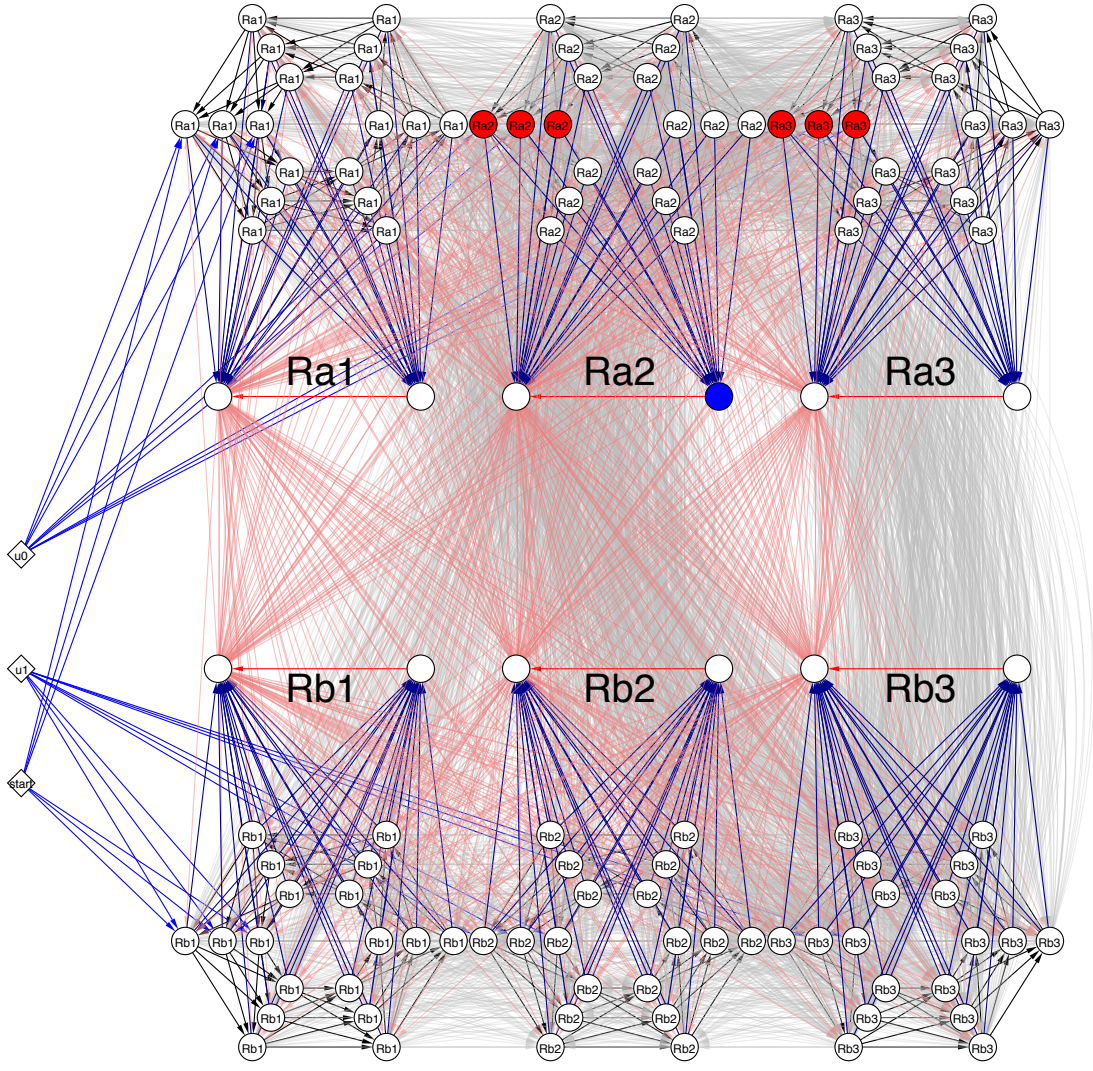
Simulation



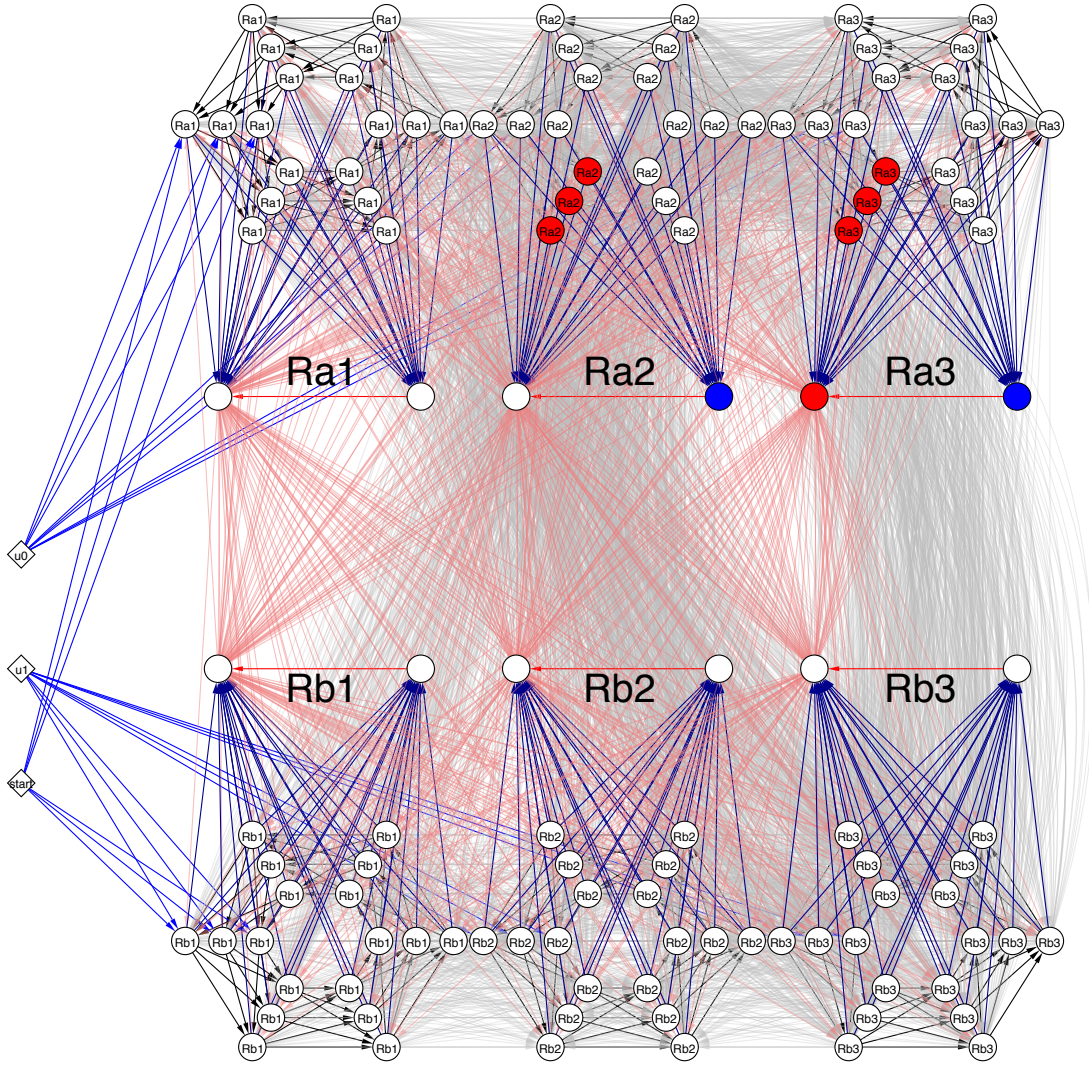
Simulation



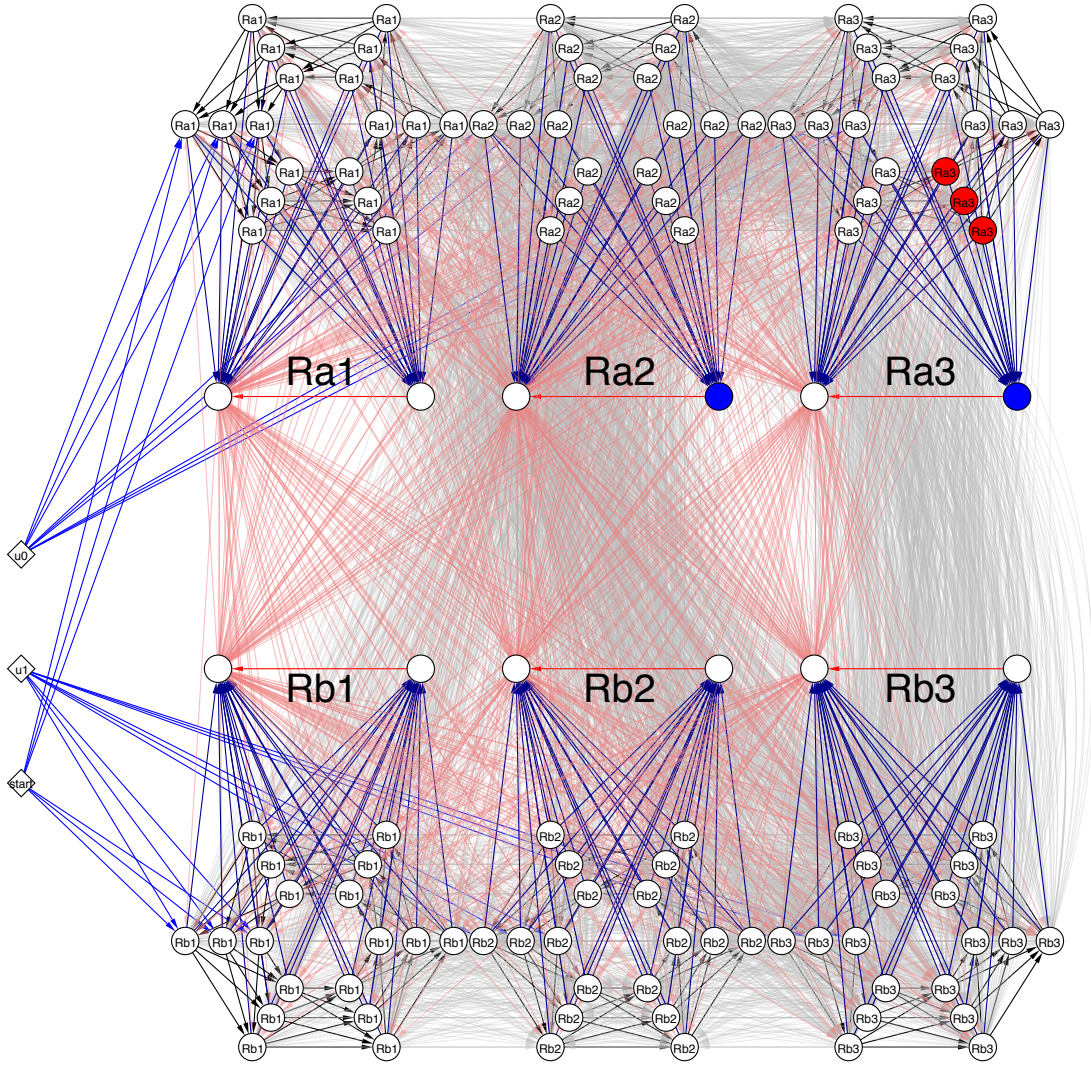
Simulation



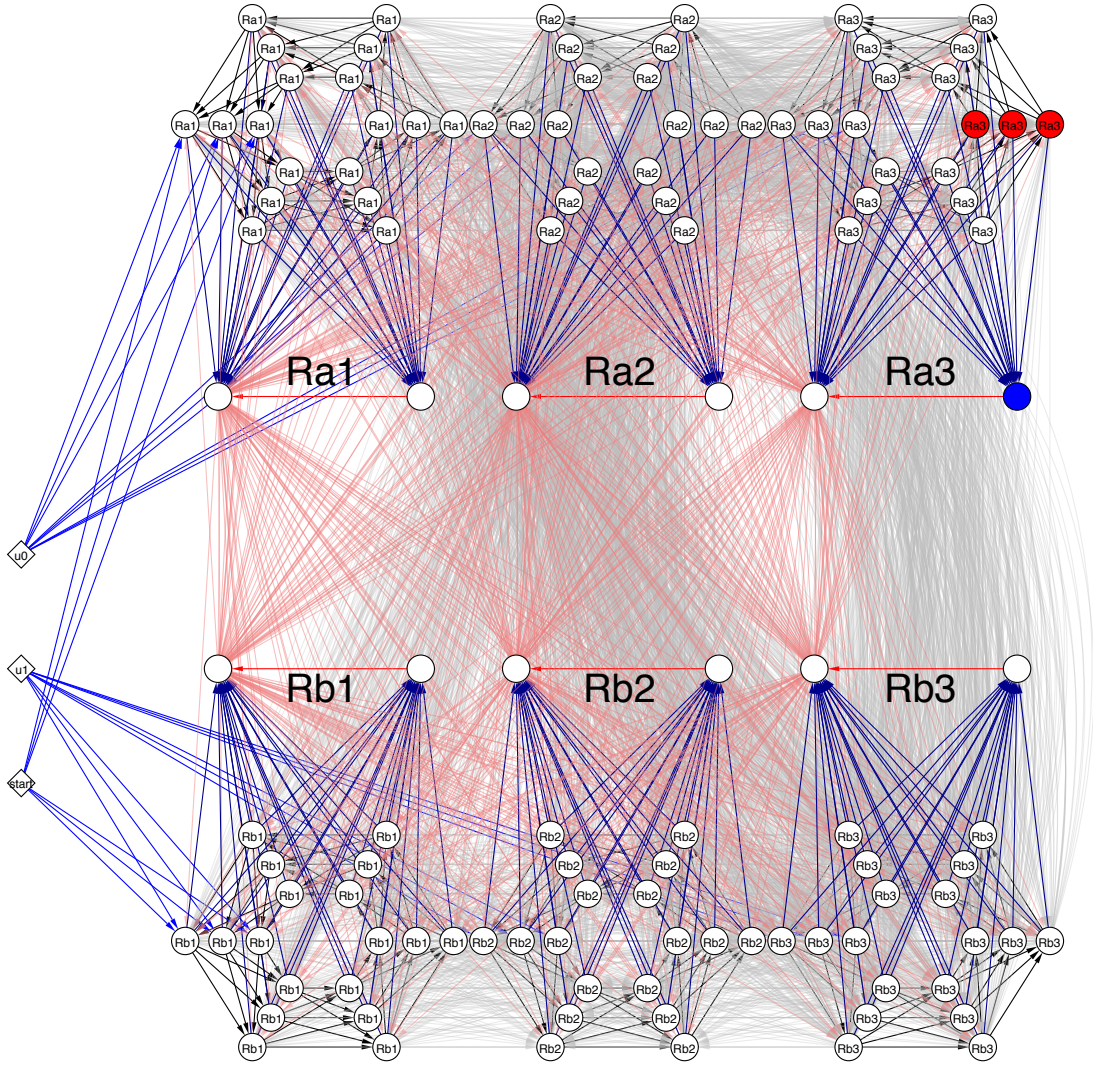
Simulation



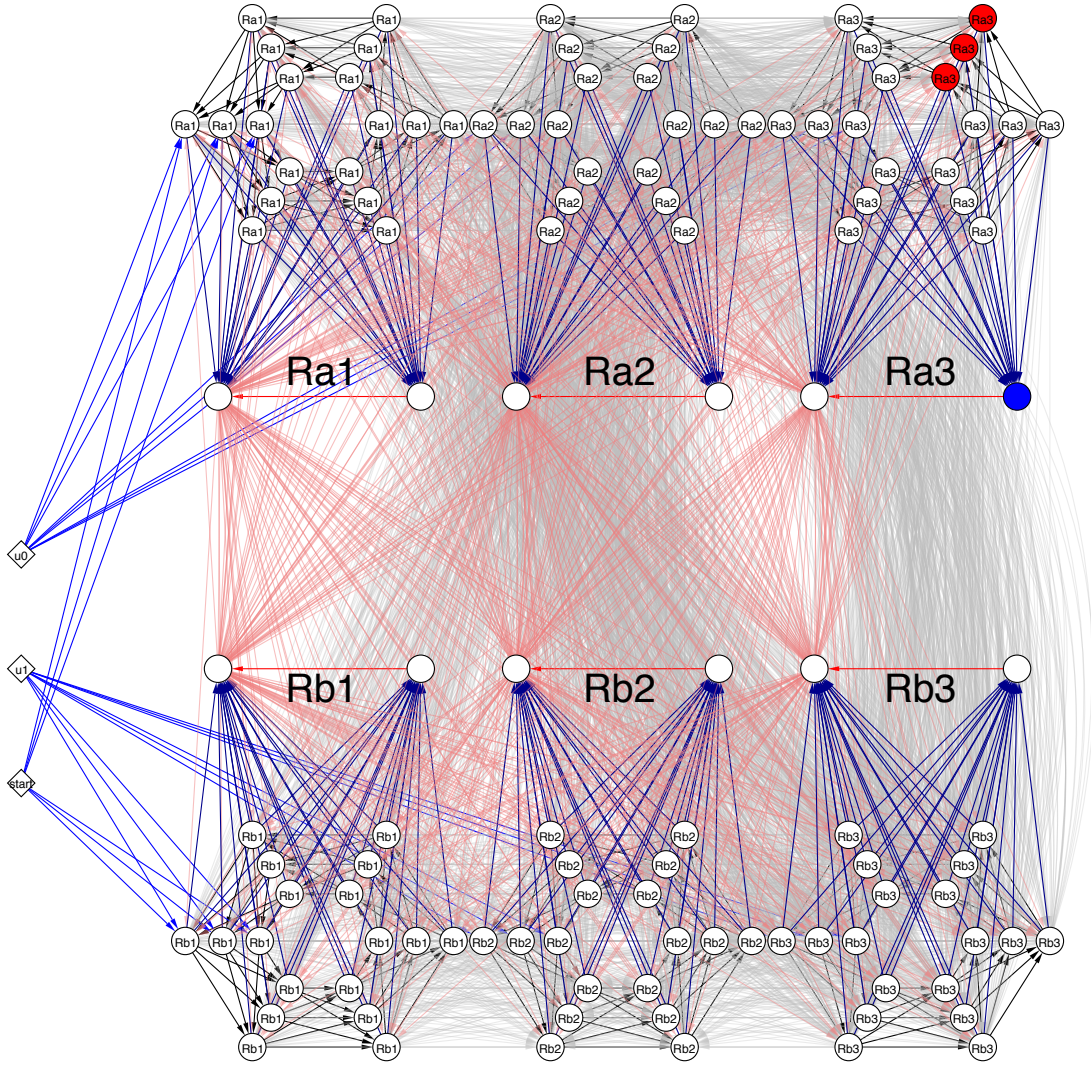
Simulation



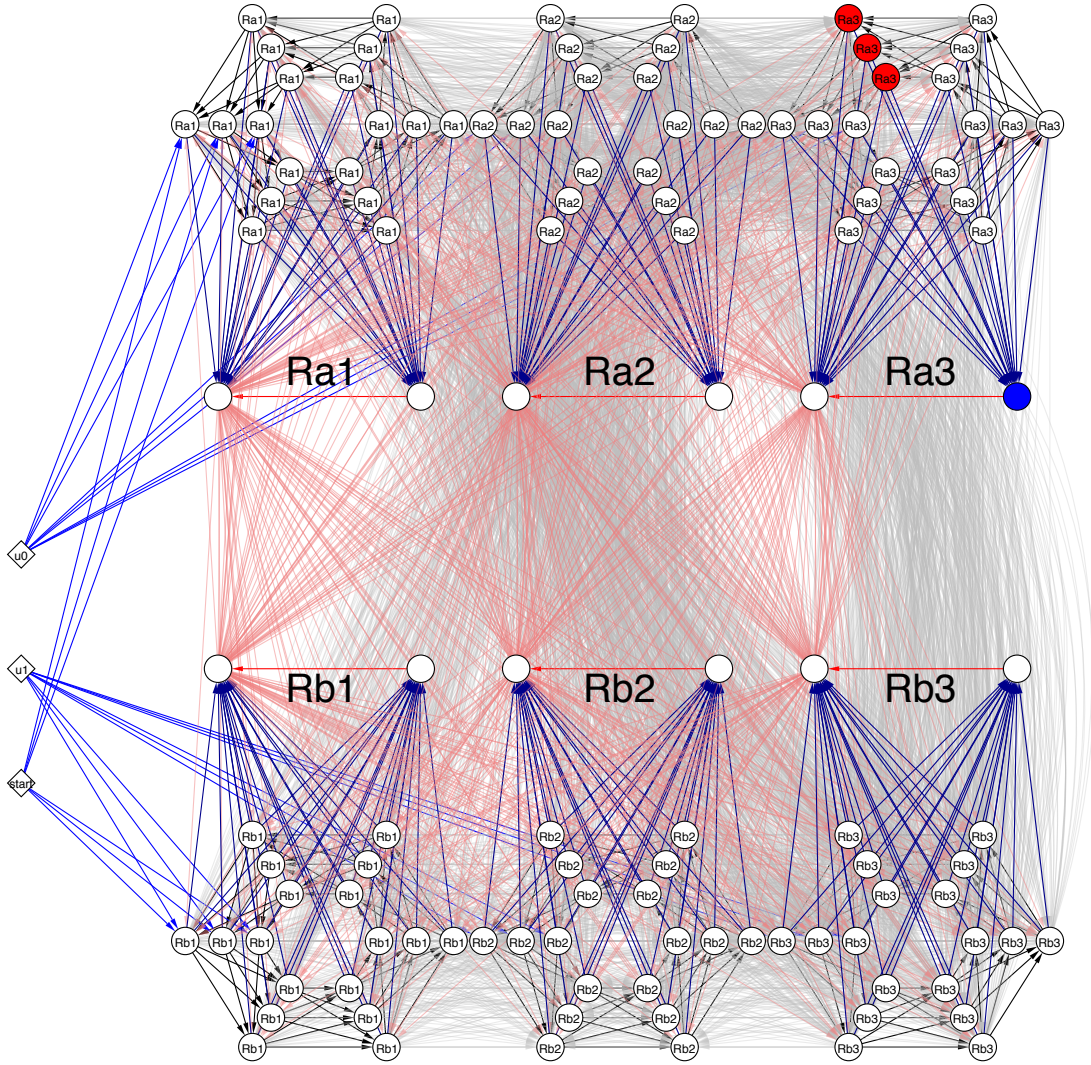
Simulation



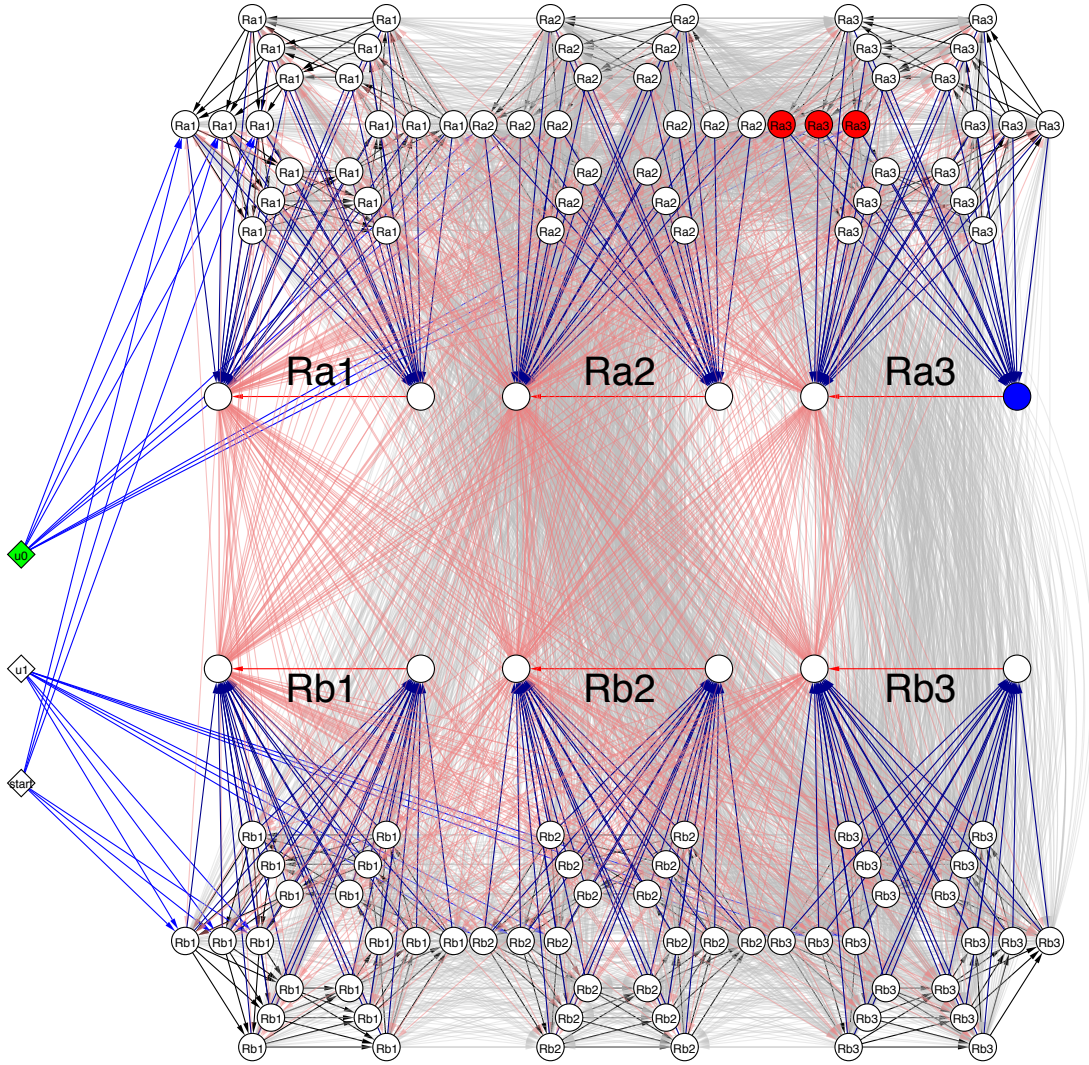
Simulation



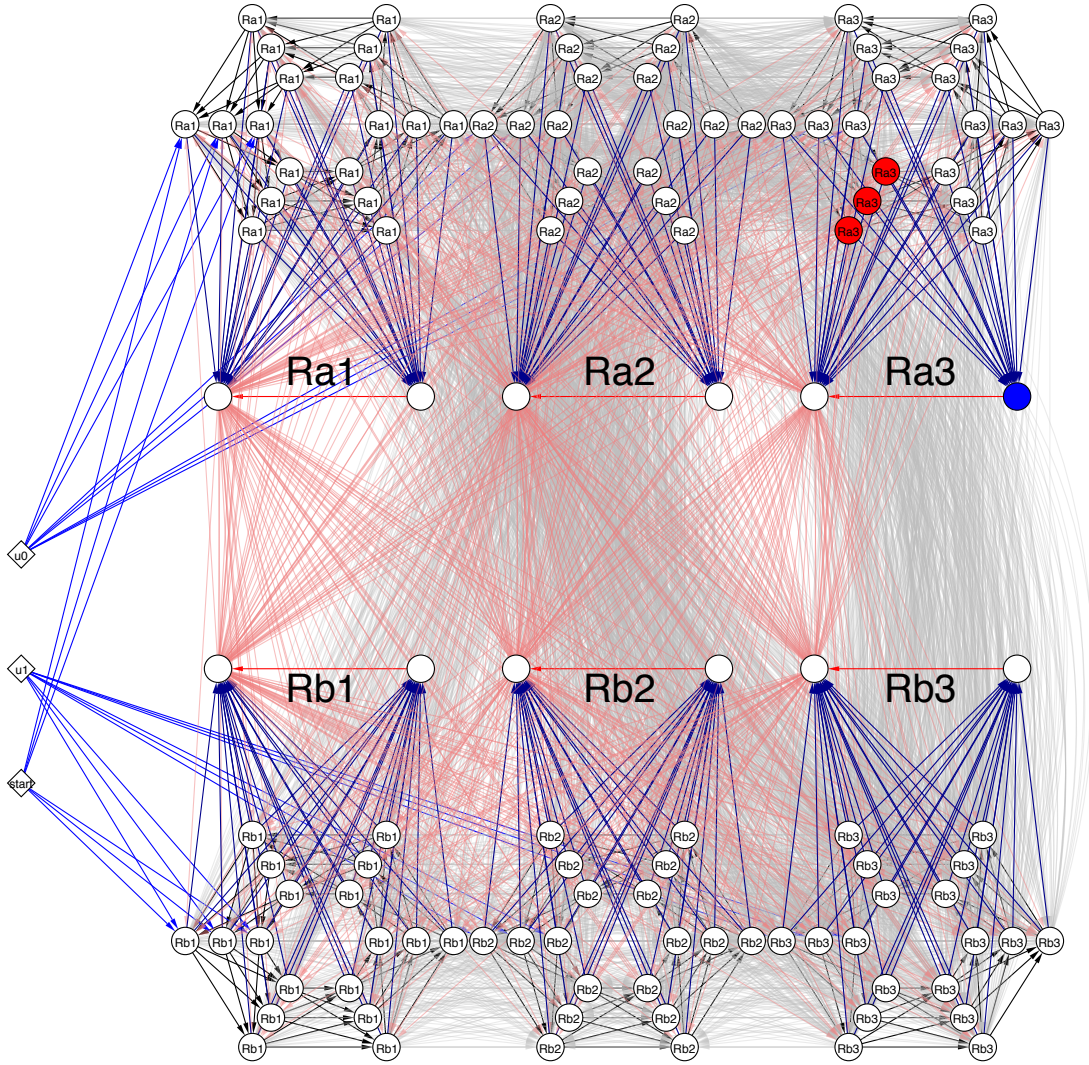
Simulation



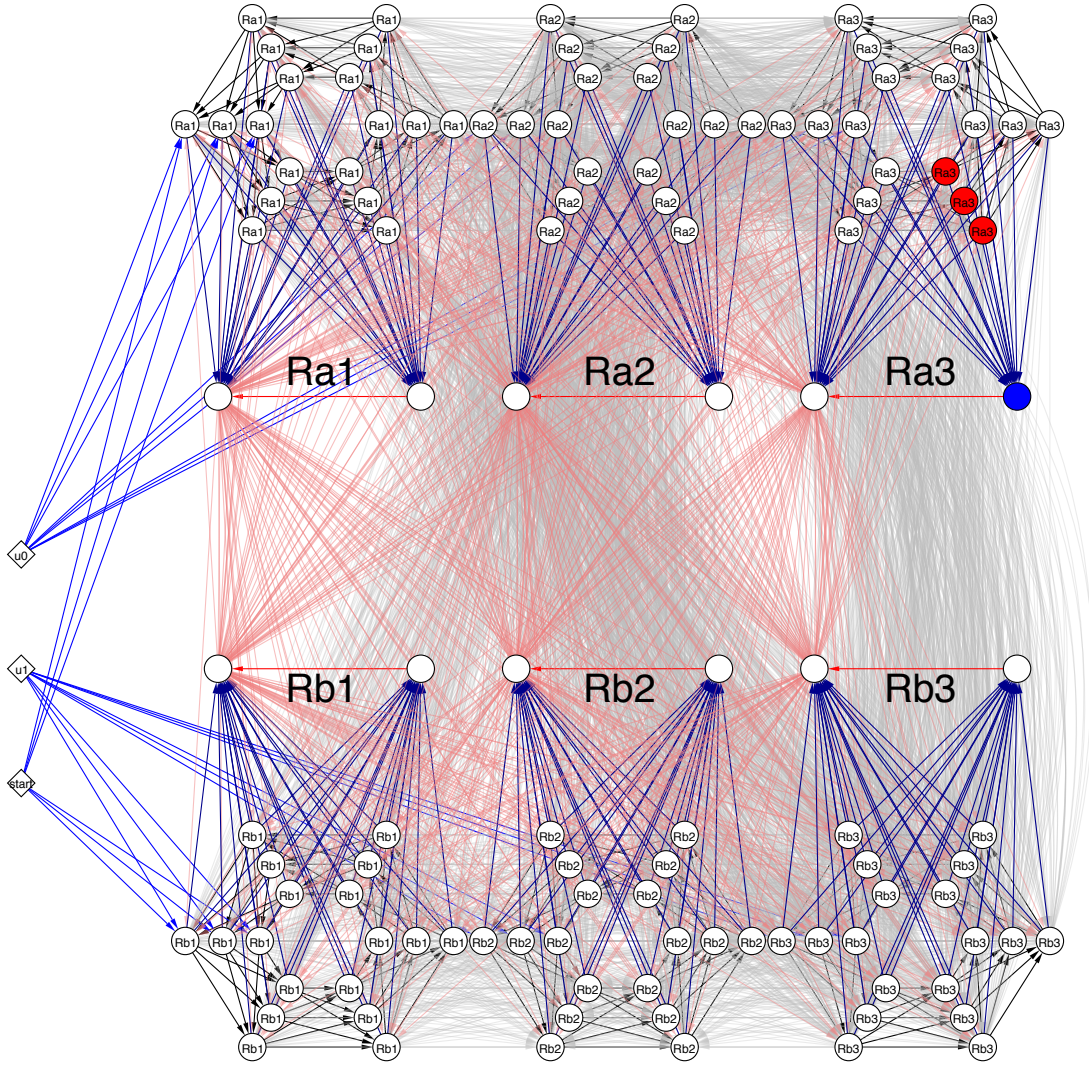
Simulation



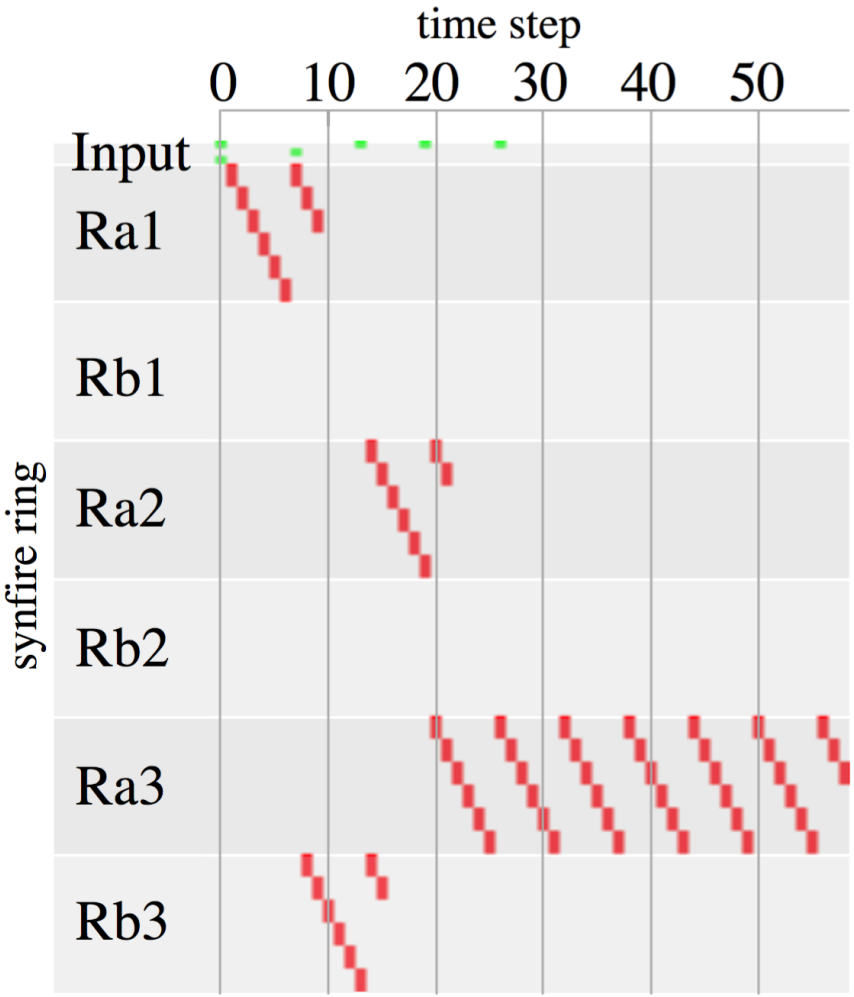
Simulation



Simulation



Simulation


time step
0
10
20
30
40
50
Input
Ra1
Rb1
Ra2
Rb2
Ra3
Rb3
synfire ring

Robustness

Table: Robustness when inter-ring connections are removed.

Connection removal coeff.	Avg number of steps	Performance
0.50	12.00	100.0%
0.55	12.00	100.0%
0.60	12.00	100.0%
0.65	11.45	95.4%
0.70	11.20	93.3%
0.75	10.30	85.8%
0.80	7.58	63.1%
0.85	6.85	57.1%
0.90	4.17	34.8%
0.95	3.75	31.2%

Robustness

Table: Robustness when intra-ring connections are removed.

Connection removal coeff.	Avg number of steps	Performance
0.00	12.00	100.0%
0.05	12.00	100.0%
0.10	12.00	100.0%
0.15	12.00	100.0%
0.20	12.00	100.0%
0.25	11.78	98.1%
0.30	11.72	97.7%
0.35	10.35	86.2%
0.40	10.05	83.8%
0.45	7.58	63.1%
0.50	6.70	55.8%
0.55	4.62	38.5%
0.60	3.05	25.4%
0.65	2.15	17.9%
0.70	1.57	13.1%
0.75	1.40	11.7%
0.80	1.12	9.4%
0.85	1.10	9.2%
0.90	1.02	8.5%
0.95	1.00	8.3%

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.

Izhikevich Spiking Neural Networks

We extend the construction to the case of Izhikevich Spiking RNNs, i.e., where each neuron's dynamics is governed by:

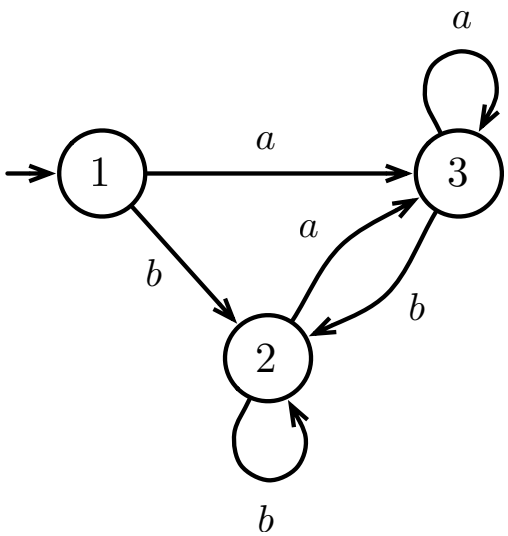
$$\begin{cases} v' &= 0.04v^2 + 5v + 140 - u + I \\ u' &= a(bv - u) \end{cases}$$

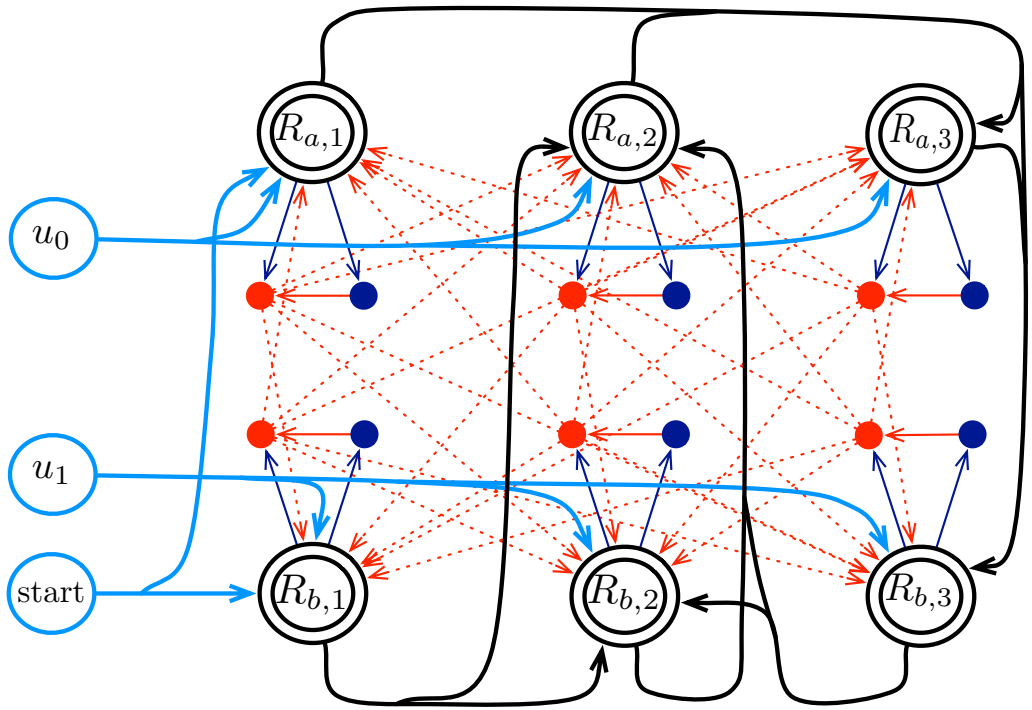
with the auxiliary after-spike reseting:

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

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

From Automata to Spiking Neural Networks with Synfire Rings





Simulation: without synaptic noise

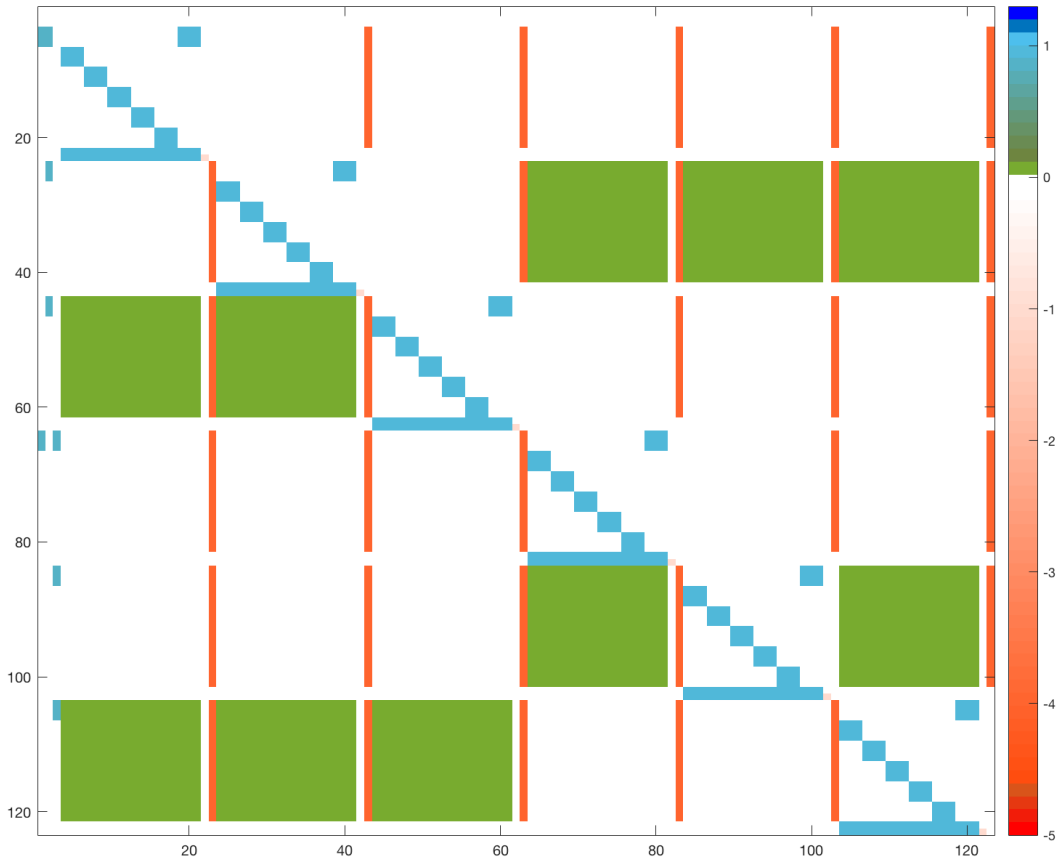


Figure: Connectivity matrix of the network

Simulation: without synaptic noise

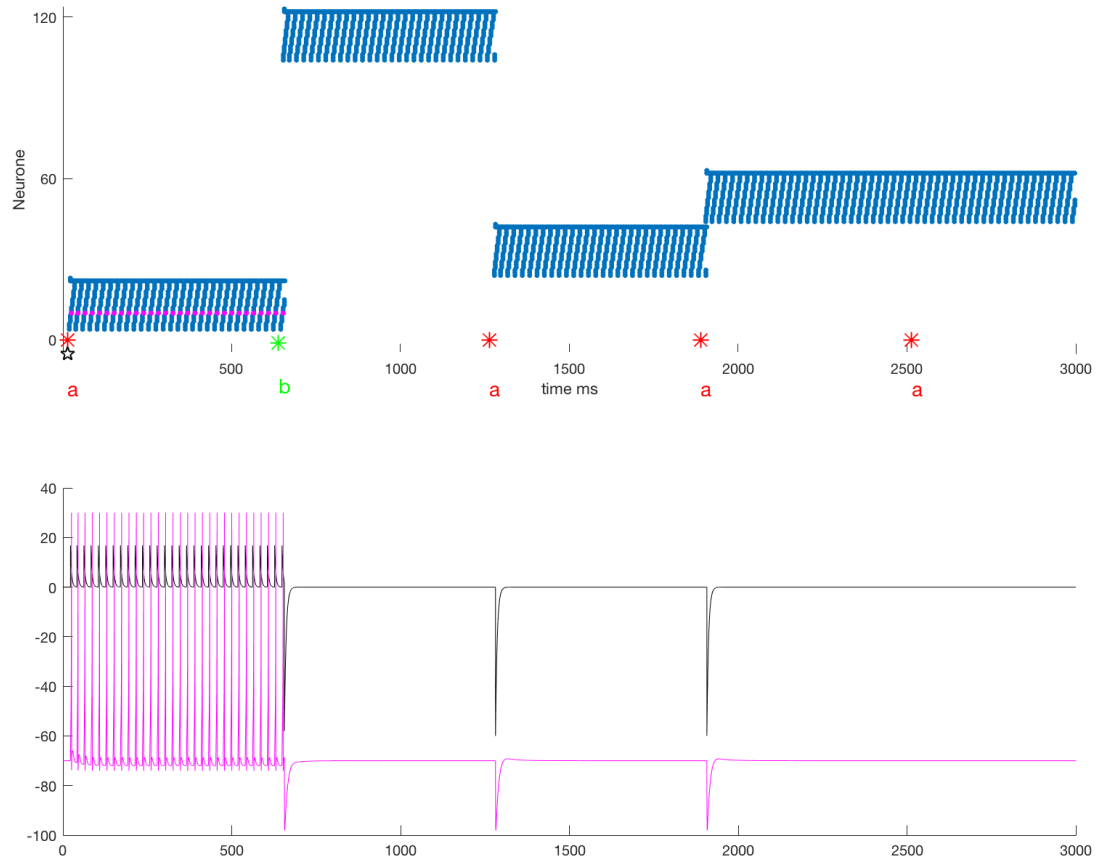


Figure: Raster plot of the simulation and activity of one spiking neuron

Simulation: without synaptic noise

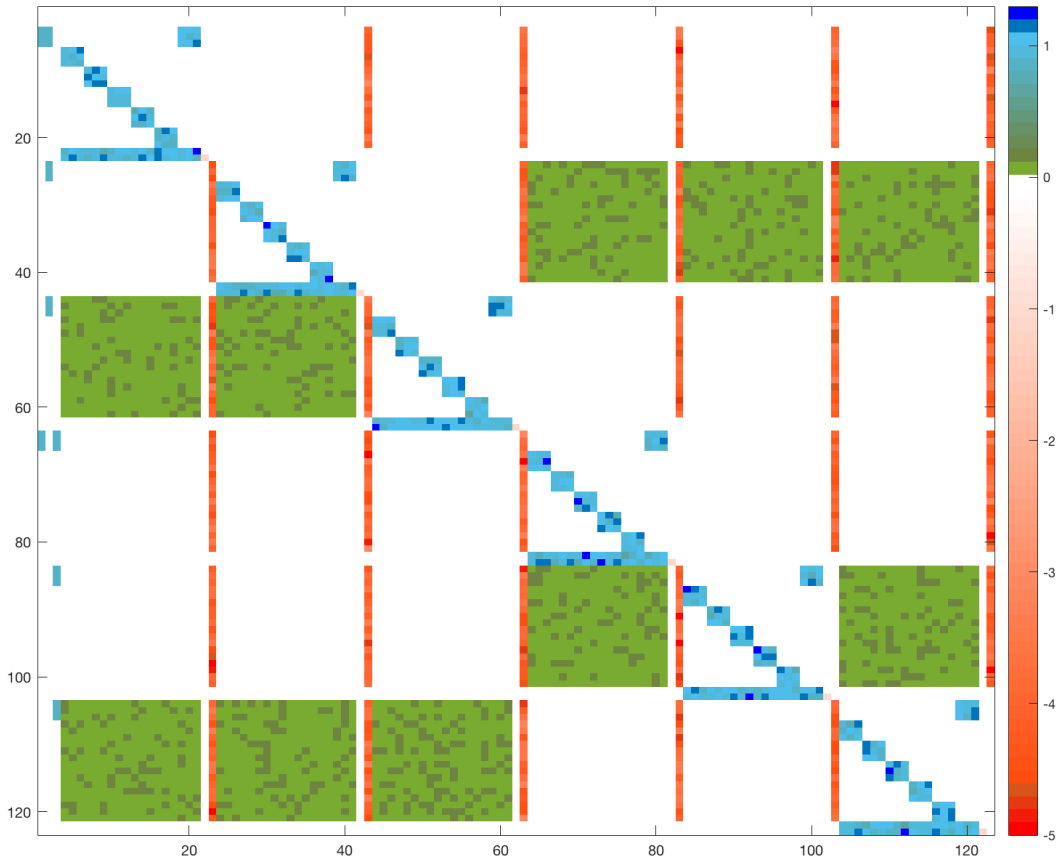
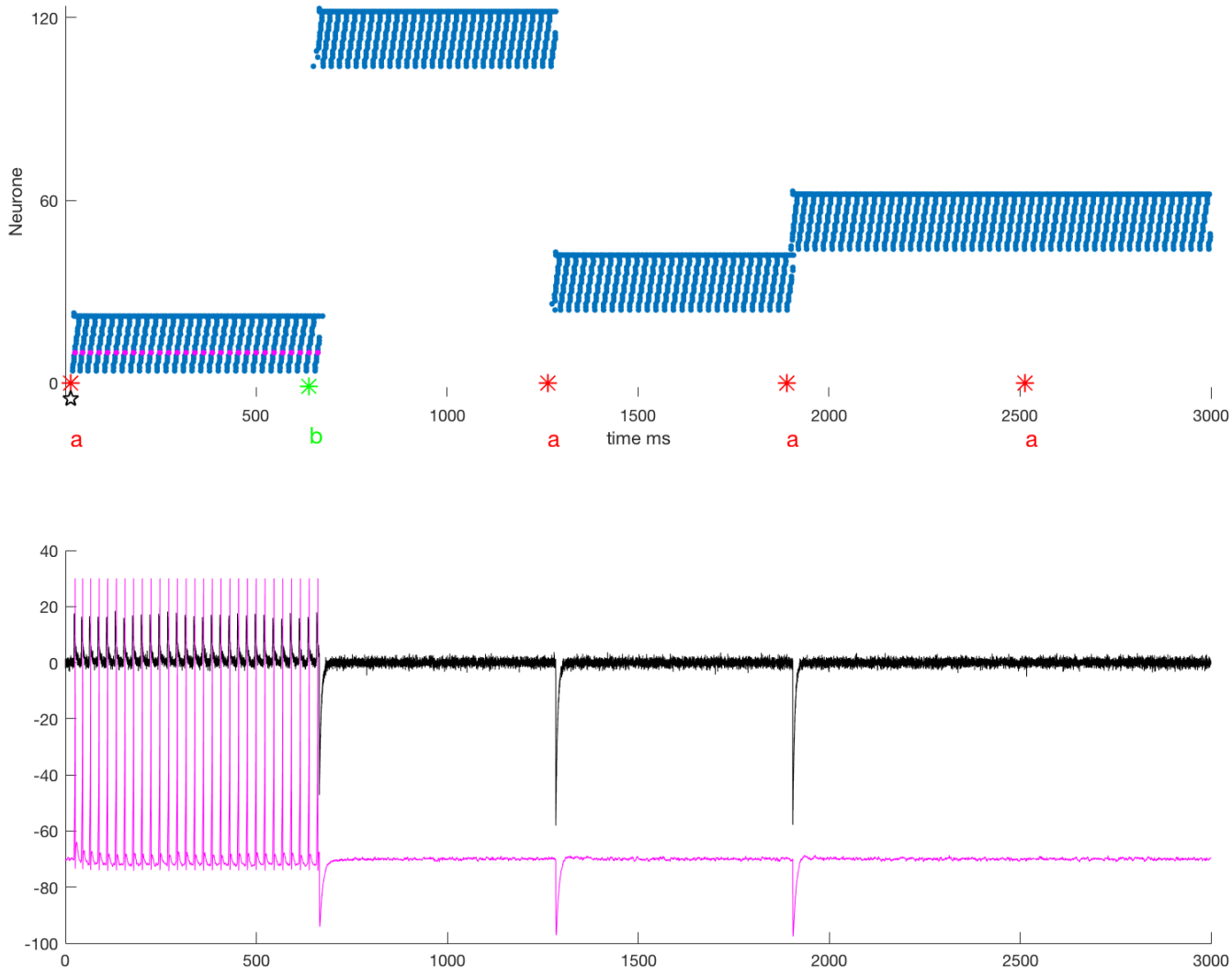


Figure: Connectivity matrix of the network

Simulation: with synaptic noise



Automata & Spiking RNNs with Synfire Rings

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

Theorem

Any finite state automaton can be simulated by some (noisy) Izhikevich spiking neural network composed of synfire rings.

Conclusions

- We showed that any finite state automaton can be simulated by some spiking RNN composed of synfire rings.
- The computational states are represented as sustained activities of neural assemblies rather than by static states (or activation vectors) of the network.
- We intend to show that abstract models of computation – rather than single specific problems – can be implemented in bio-inspired neural networks.
- Future work along these lines: towards Turing complete computation.

Conclusions

- We showed that any finite state automaton can be simulated by some spiking RNN composed of synfire rings.
- The computational states are represented as sustained activities of neural assemblies rather than by static states (or activation vectors) of the network.
- We intend to show that abstract models of computation – rather than single specific problems – can be implemented in bio-inspired neural networks.
- Future work along these lines: towards Turing complete computation.

Conclusions

- We showed that any finite state automaton can be simulated by some spiking RNN composed of synfire rings.
- The computational states are represented as sustained activities of neural assemblies rather than by static states (or activation vectors) of the network.
- We intend to show that abstract models of computation – rather than single specific problems – can be implemented in bio-inspired neural networks.
- Future work along these lines: towards Turing complete computation.

Conclusions

- We showed that any finite state automaton can be simulated by some spiking RNN composed of synfire rings.
- The computational states are represented as sustained activities of neural assemblies rather than by static states (or activation vectors) of the network.
- We intend to show that abstract models of computation – rather than single specific problems – can be implemented in bio-inspired neural networks.
- Future work along these lines: towards Turing complete computation.