Neural Computation with Spiking Neural Networks Composed of Synfire Rings

Jérémie Cabessa
Ginette Horcholle-Bossavit and Brigitte Quenet

Department of Mathematical Economics
University Paris II, France

ICANN 17, September 12, 2017, Alghero, Sardinia

Introduction

- In many biologically inspired neural models (like Hopfield networks or attractor networks in general), *information processing* is related to *attractor dynamics*.
- *Memories* are represented by *fixed point attractors* (in general). *Repetitive and oscillatory behaviors* are represented by *cyclic attractors*.
- A challenge in attractor networks is to determine how attractors (number, types, transitions between them) can be controlled.

Introduction

- In many biologically inspired neural models (like Hopfield networks or attractor networks in general), *information processing* is related to *attractor dynamics*.
- *Memories* are represented by *fixed point attractors* (in general). *Repetitive and oscillatory behaviors* are represented by *cyclic attractors*.
- A challenge in attractor networks is to determine how attractors (number, types, transitions between them) can be controlled.

Introduction

- In many biologically inspired neural models (like Hopfield networks or attractor networks in general), *information processing* is related to *attractor dynamics*.
- *Memories* are represented by *fixed point attractors* (in general). *Repetitive and oscillatory behaviors* are represented by *cyclic attractors*.
- A challenge in attractor networks is to determine how attractors (number, types, transitions between them) can be controlled.

Introduction

- Do biological neural networks have the ability to compute, in the precise sense of computation theory?
- Yes... We simulate *finite state automata* with *spiking neural networks* composed of synfire rings.
- *Computational states* are represented by *sustained activities of synfire rings*, i.e., by *cyclic attractors*.
- Transitions between such attractors is perfectly controlled, in an input-driven way.

Introduction

- Do biological neural networks have the ability to compute, in the precise sense of computation theory?
- Yes... We simulate *finite state automata* with *spiking neural networks* composed of synfire rings.
- *Computational states* are represented by *sustained activities of synfire rings*, i.e., by *cyclic attractors*.
- Transitions between such attractors is perfectly controlled, in an input-driven way.

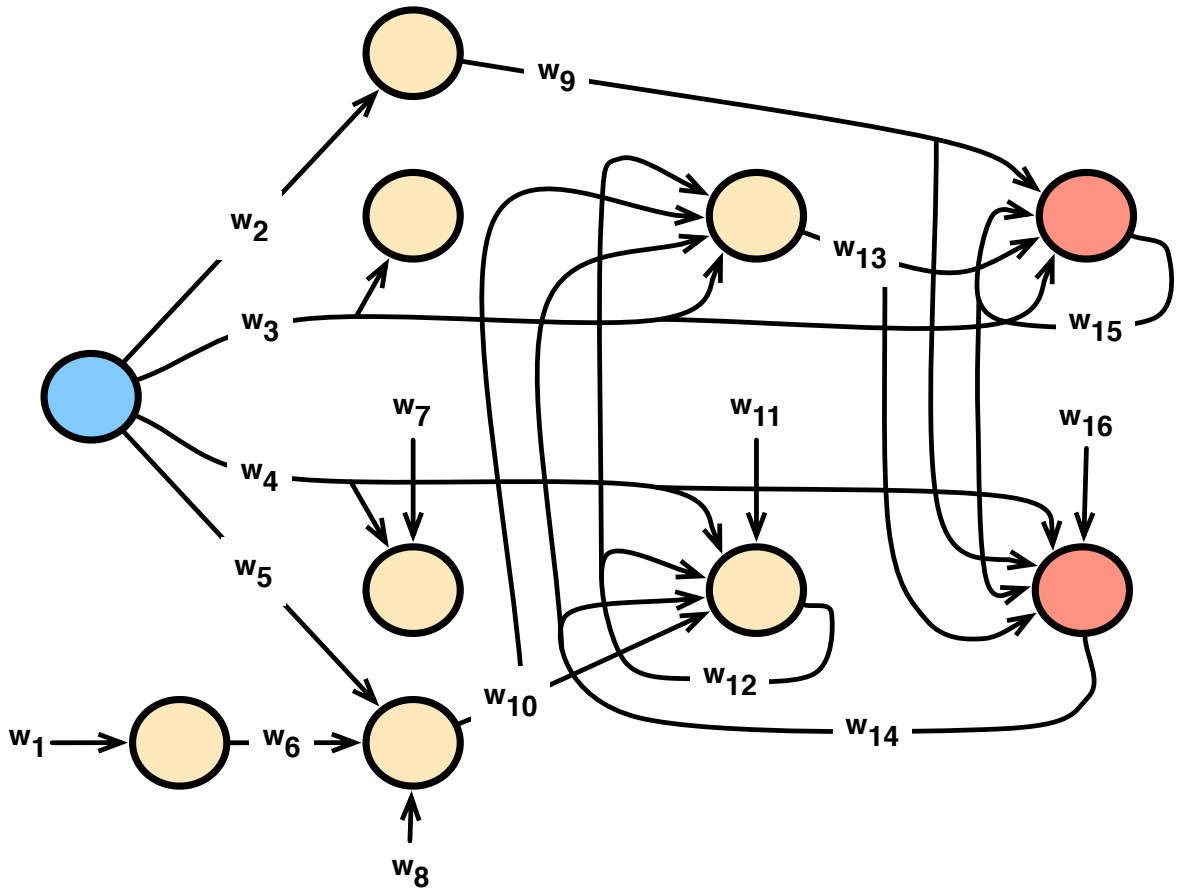
Introduction

- Do biological neural networks have the ability to compute, in the precise sense of computation theory?
- Yes... We simulate *finite state automata* with *spiking neural networks* composed of synfire rings.
- *Computational states* are represented by *sustained activities of synfire rings*, i.e., by *cyclic attractors*.
- Transitions between such attractors is perfectly controlled, in an input-driven way.

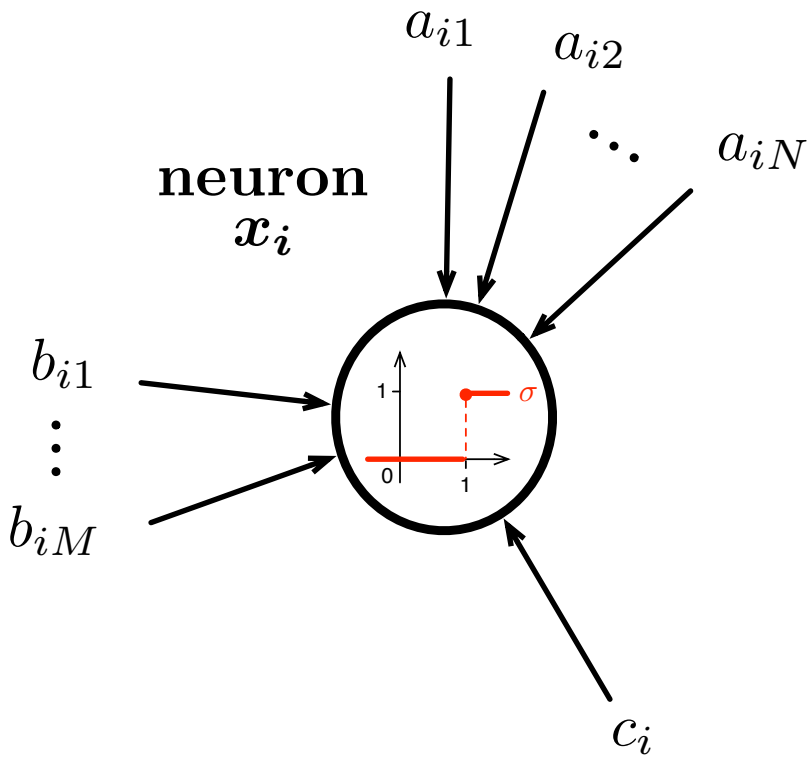
Introduction

- Do biological neural networks have the ability to compute, in the precise sense of computation theory?
- Yes... We simulate *finite state automata* with *spiking neural networks* composed of synfire rings.
- *Computational states* are represented by *sustained activities of synfire rings*, i.e., by *cyclic attractors*.
- Transitions between such attractors is perfectly controlled, in an input-driven way.

Recurrent Neural Network

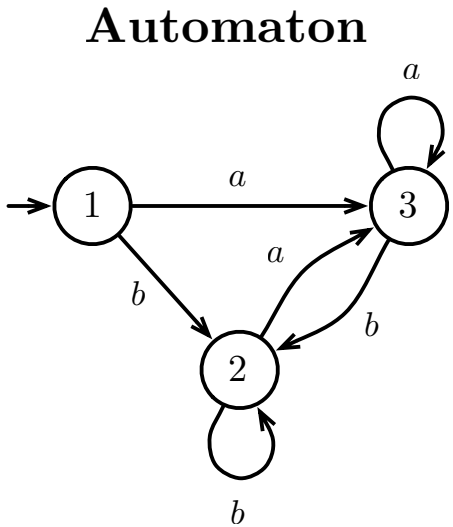


Boolean Recurrent Neural Network



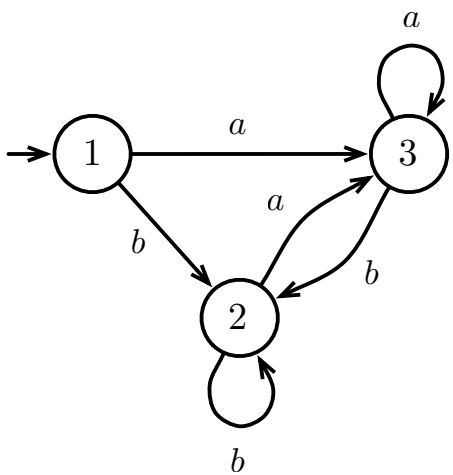
$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right)$$

From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

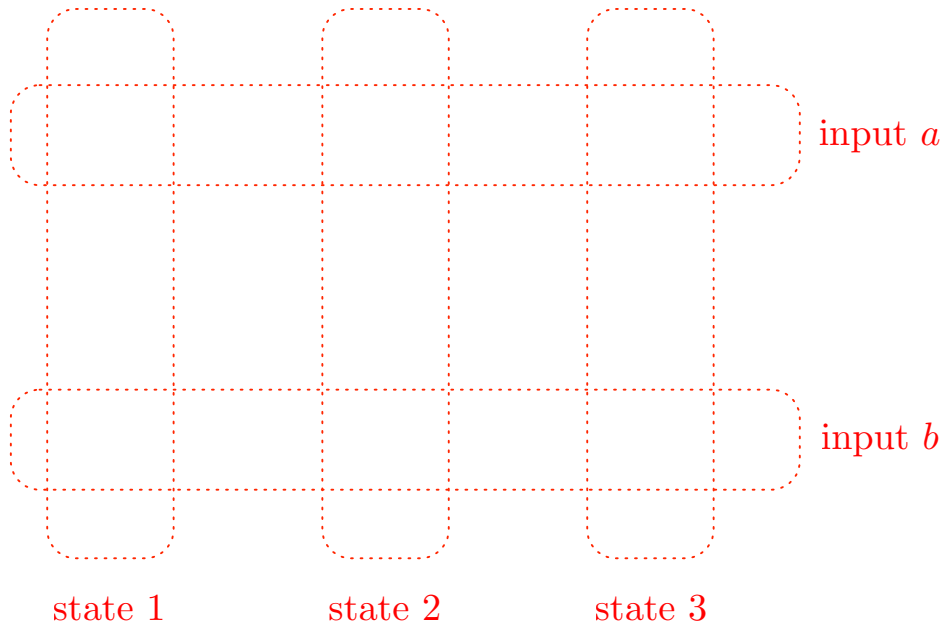


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

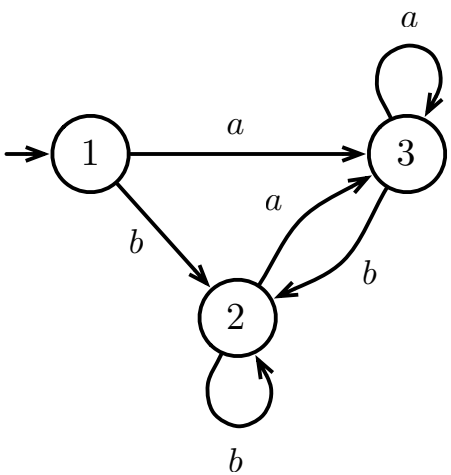


Boolean Neural Network

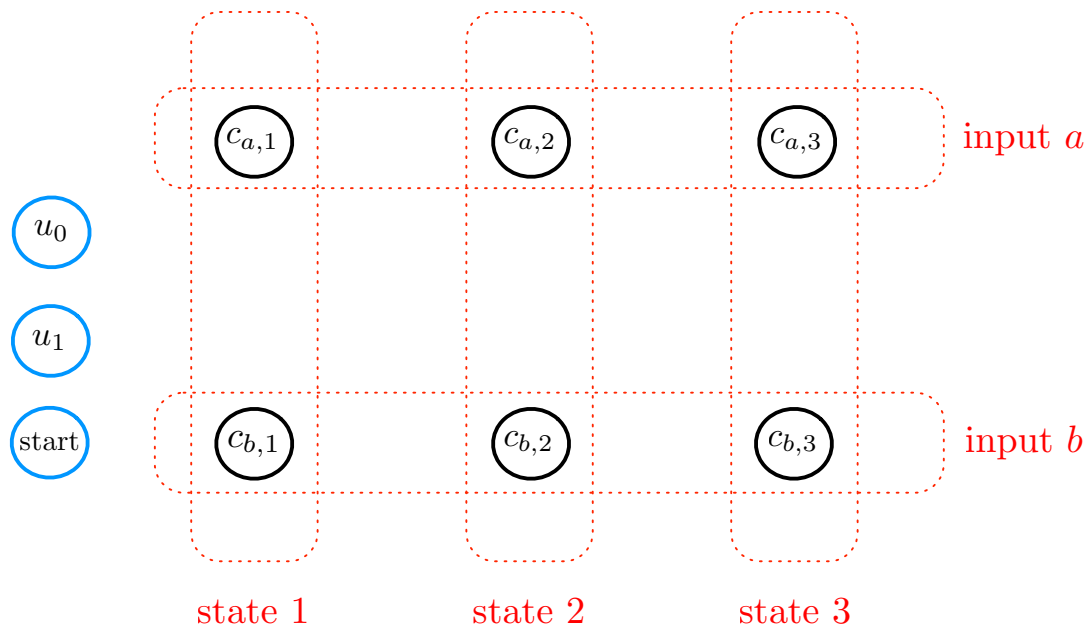


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

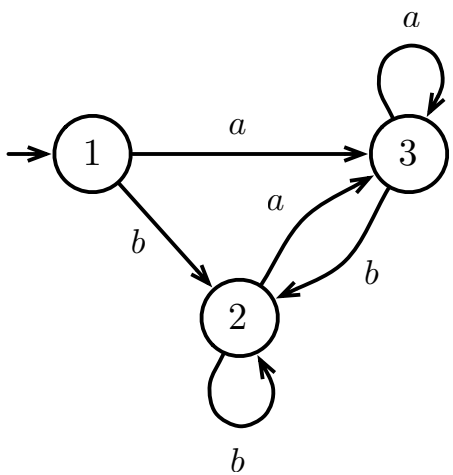


Boolean Neural Network

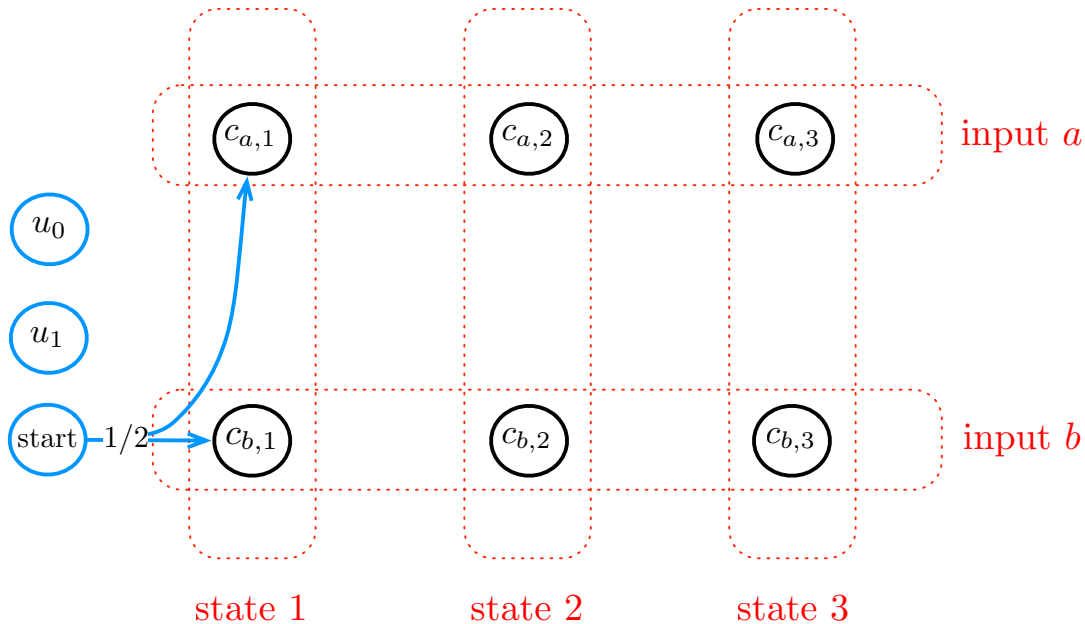


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

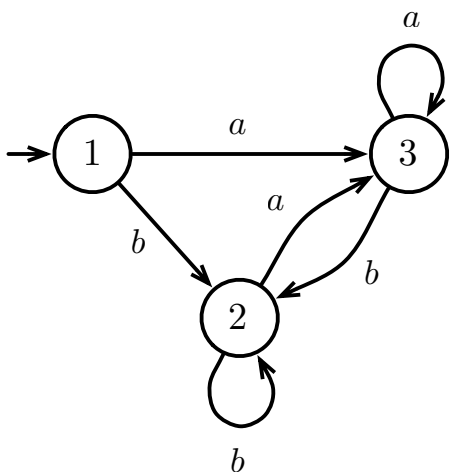


Boolean Neural Network

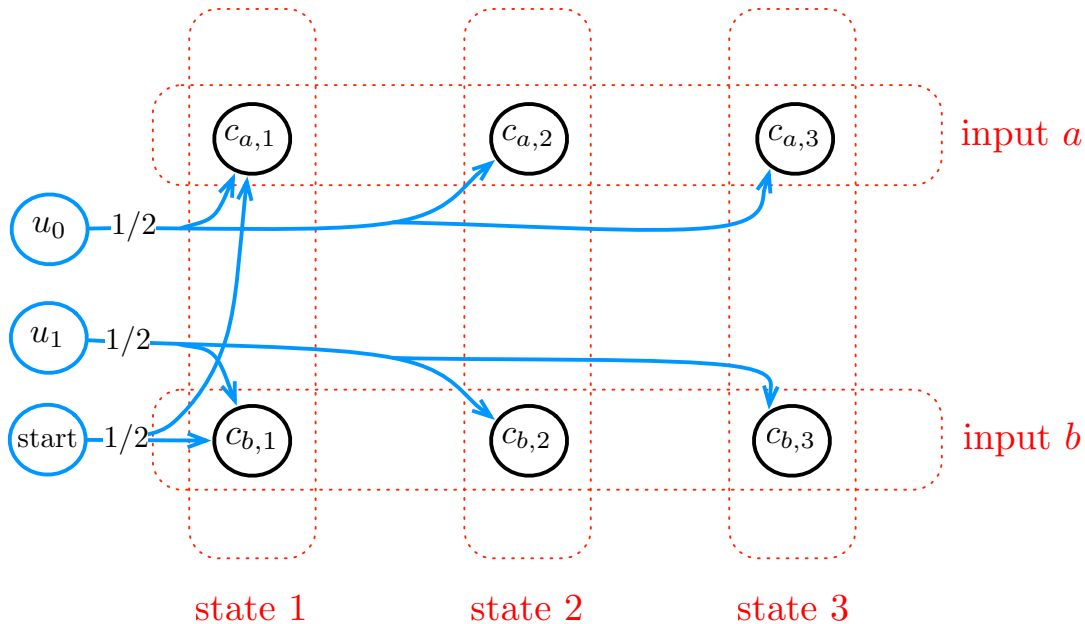


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

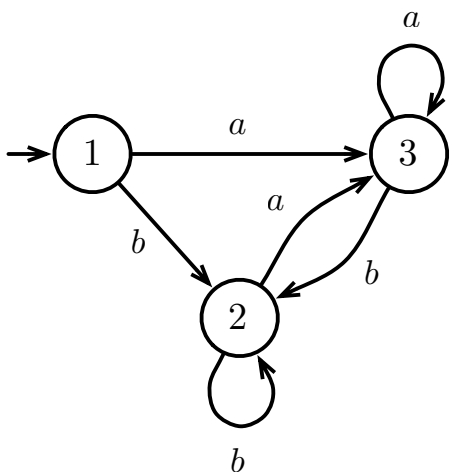


Boolean Neural Network

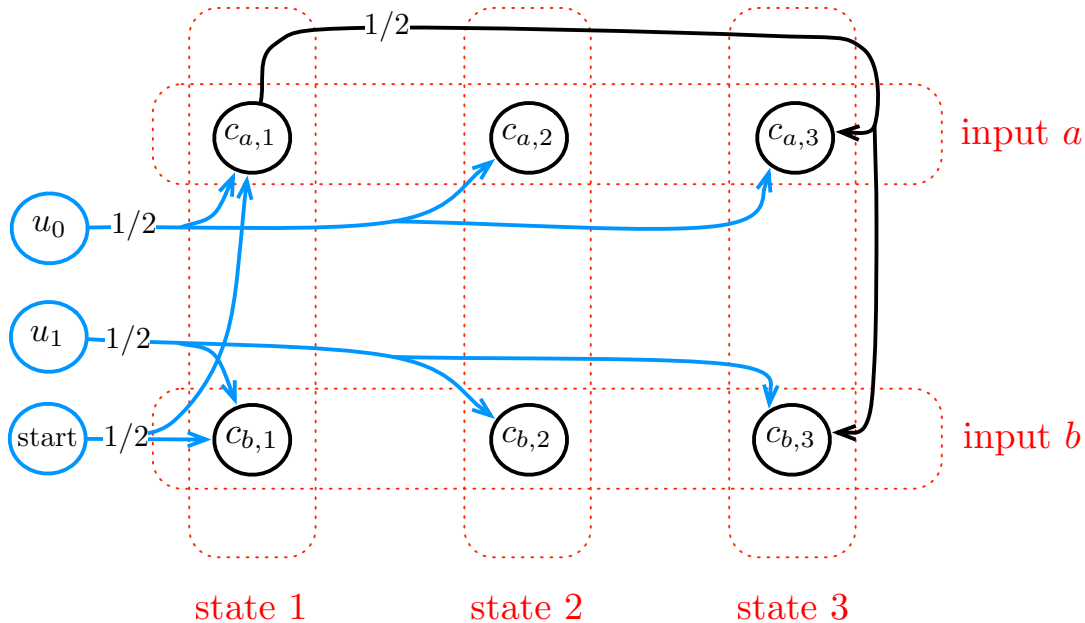


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

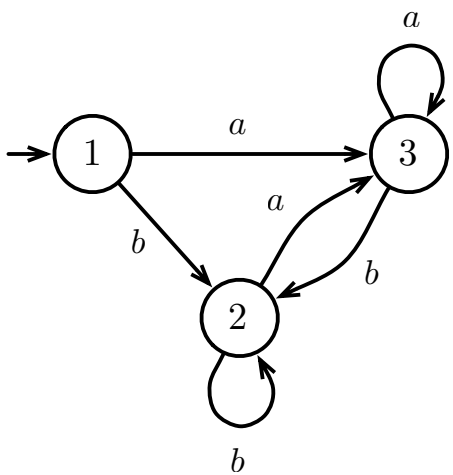


Boolean Neural Network

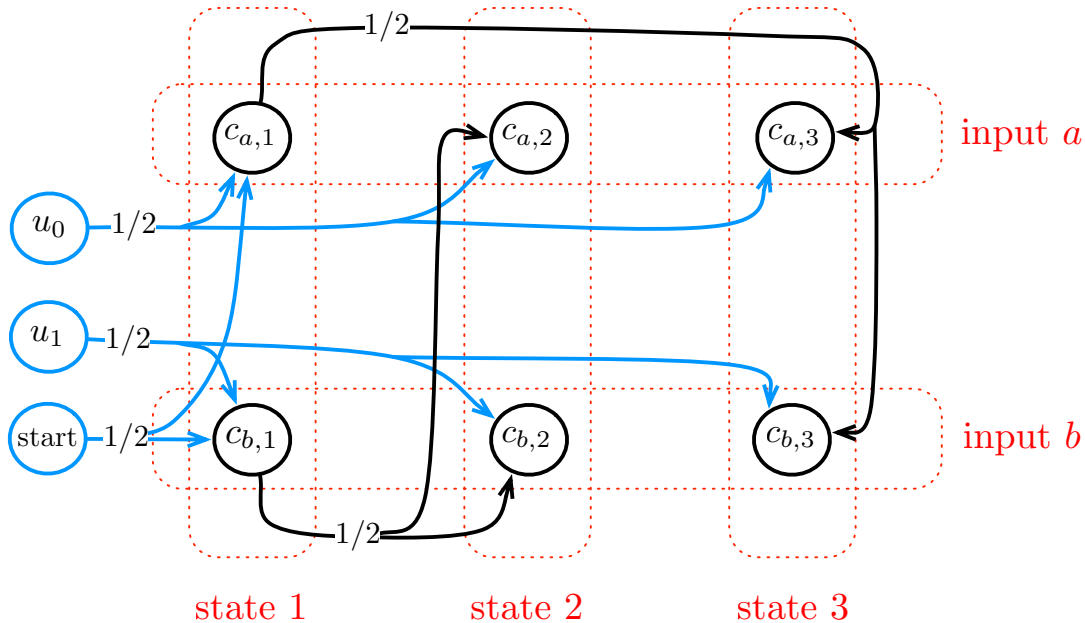


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

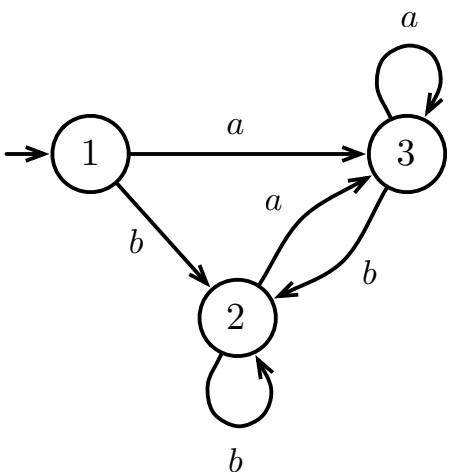


Boolean Neural Network

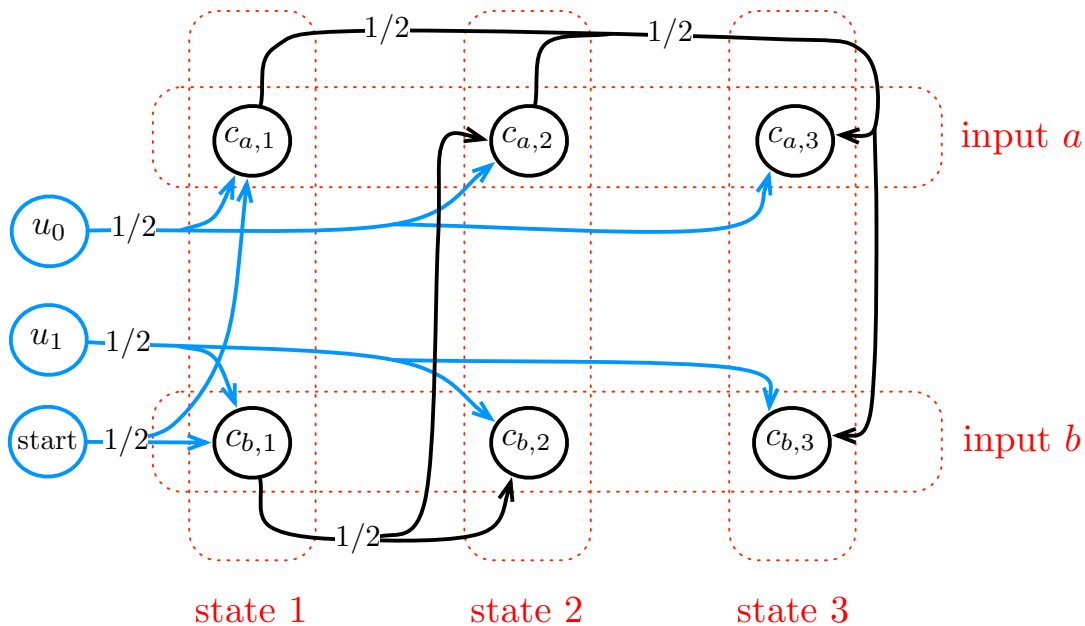


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

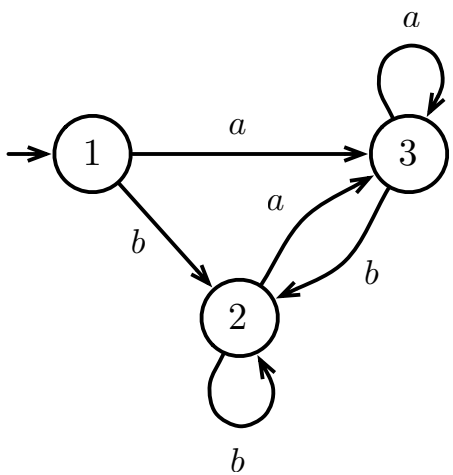


Boolean Neural Network

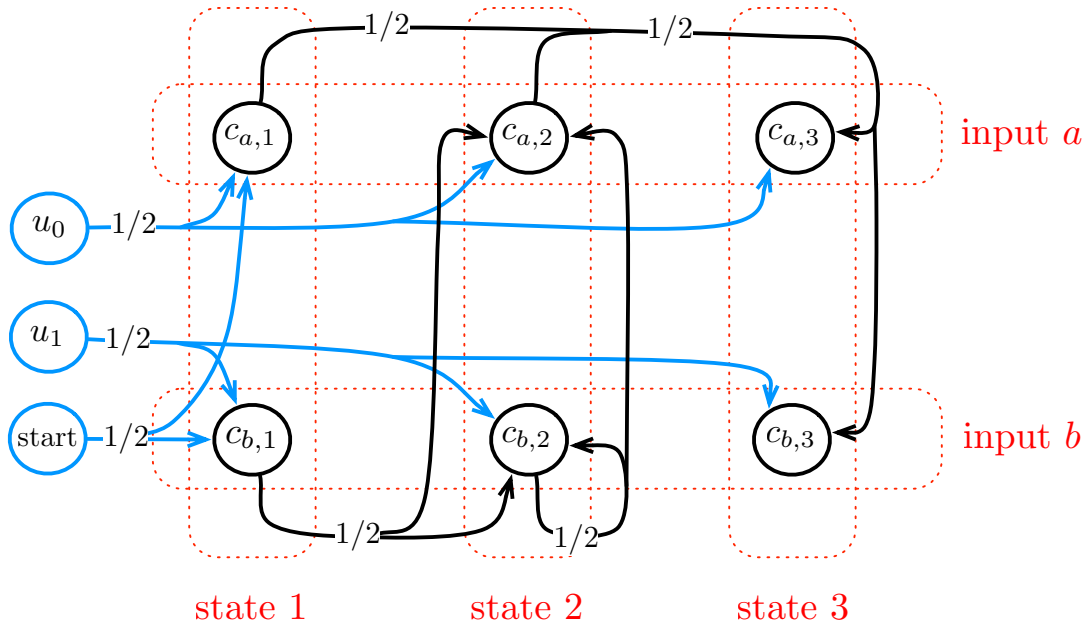


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

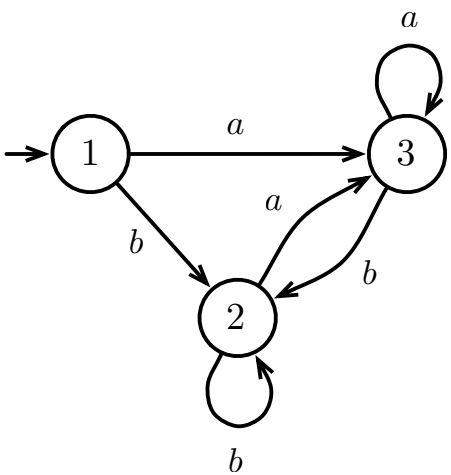


Boolean Neural Network

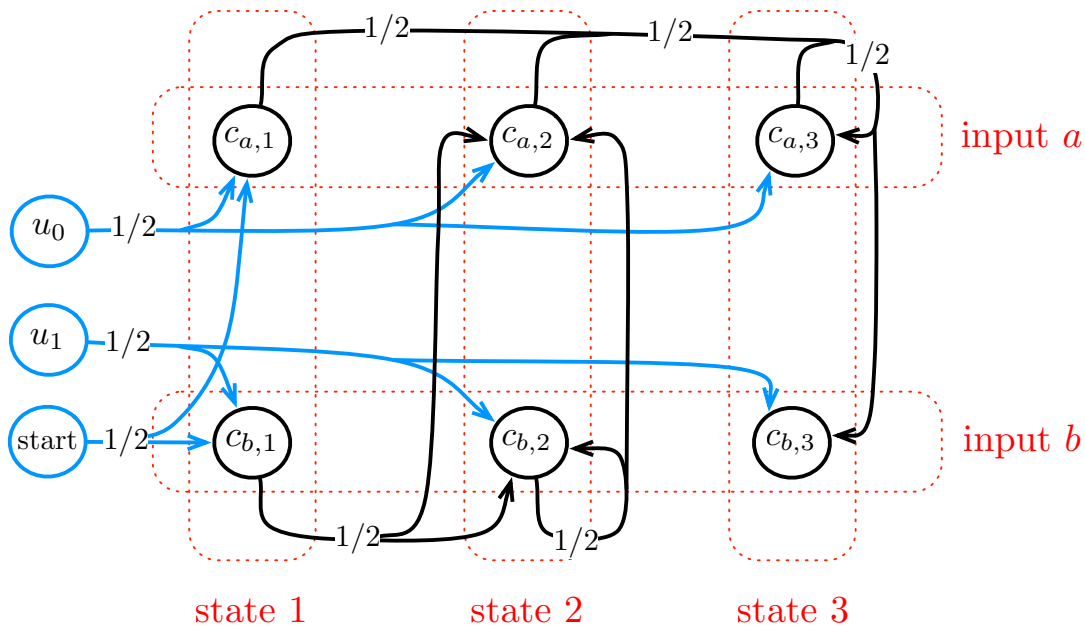


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

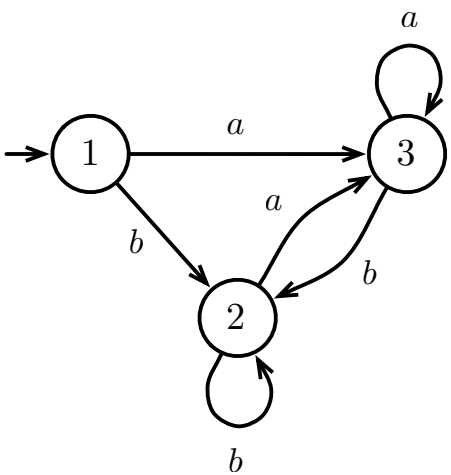


Boolean Neural Network

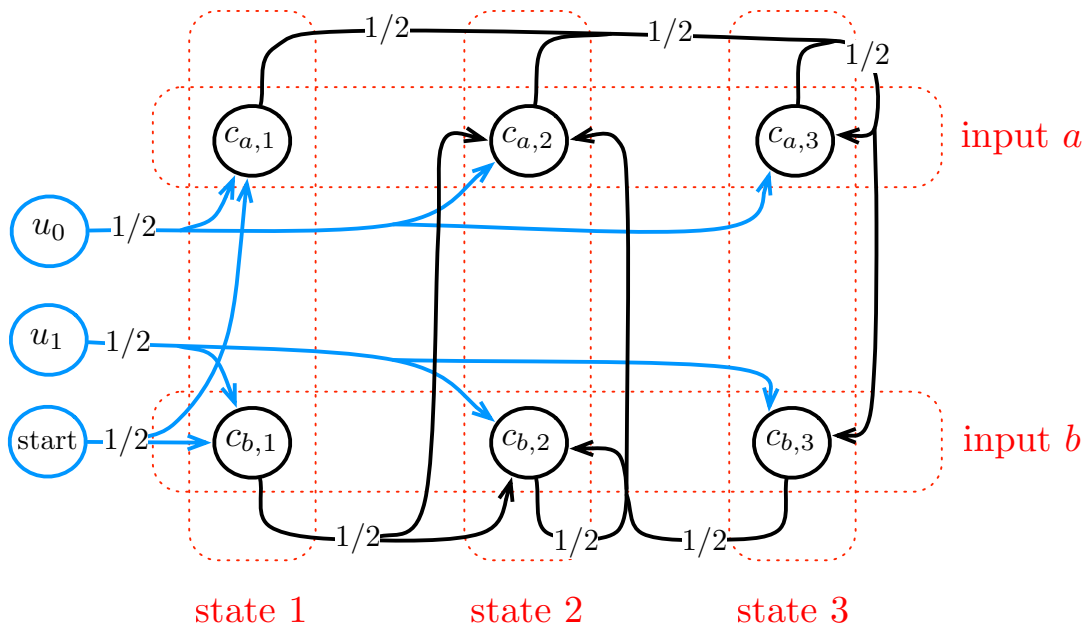


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

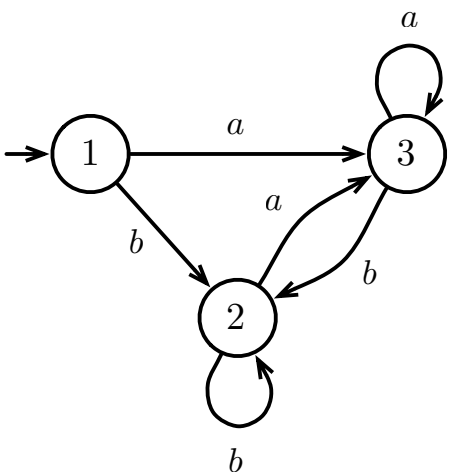


Boolean Neural Network

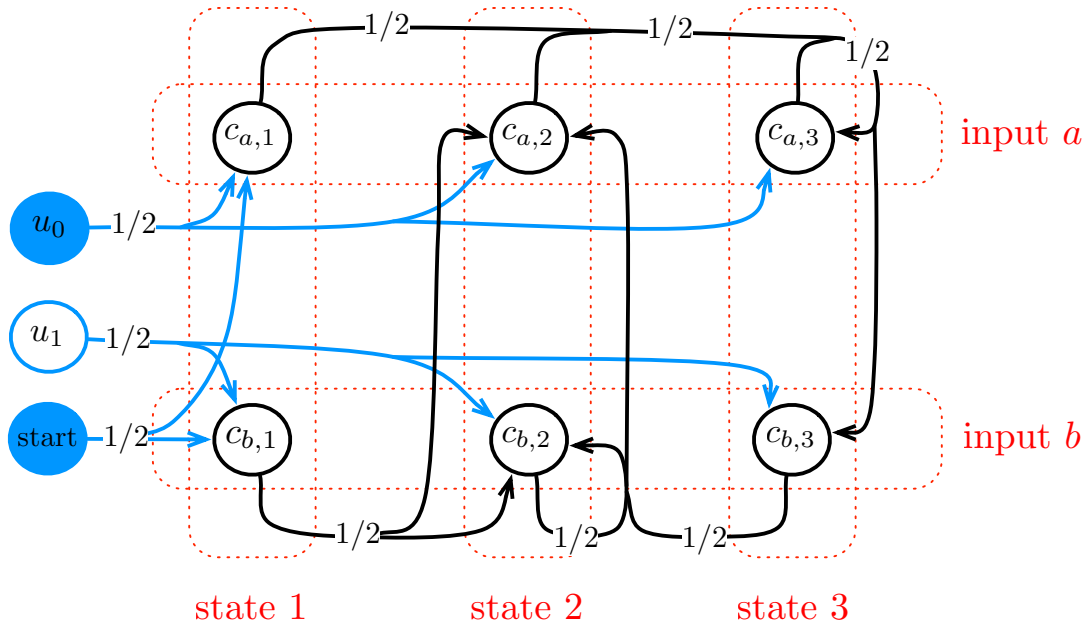


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

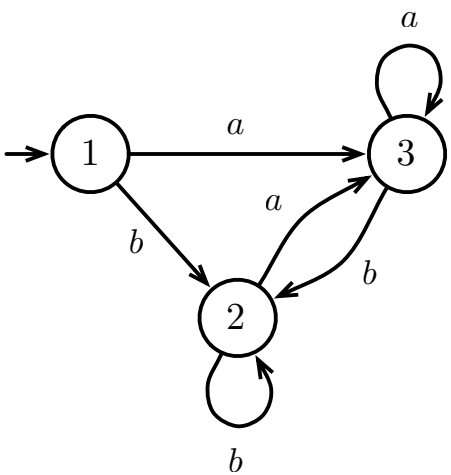


Boolean Neural Network

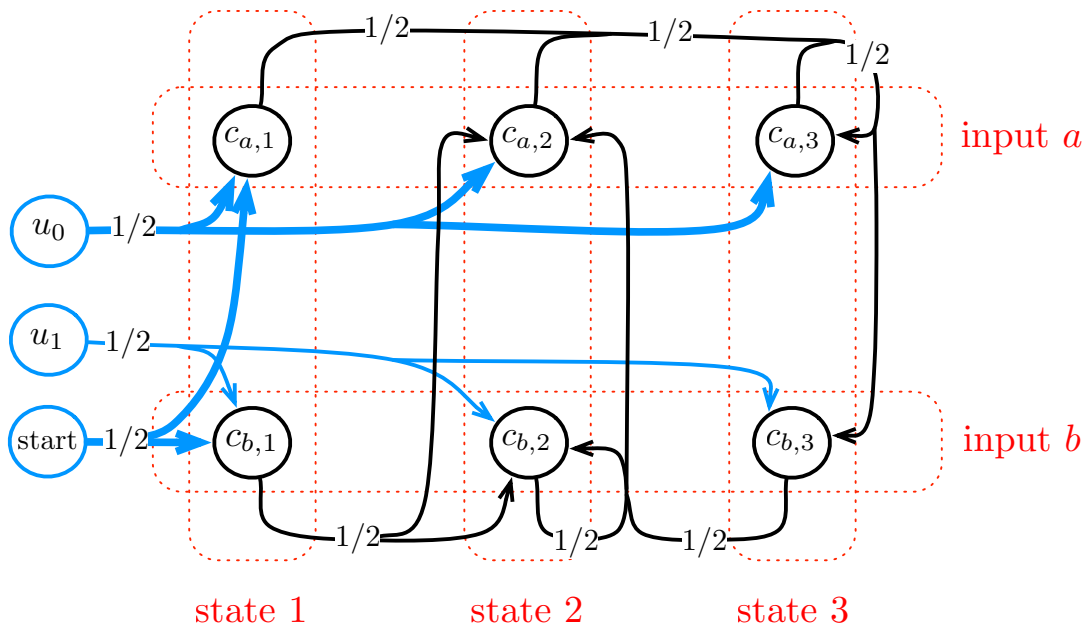


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

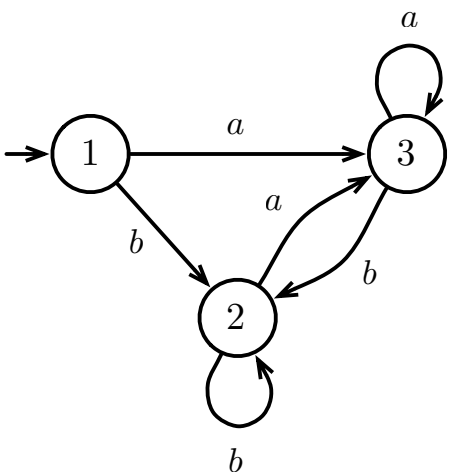


Boolean Neural Network

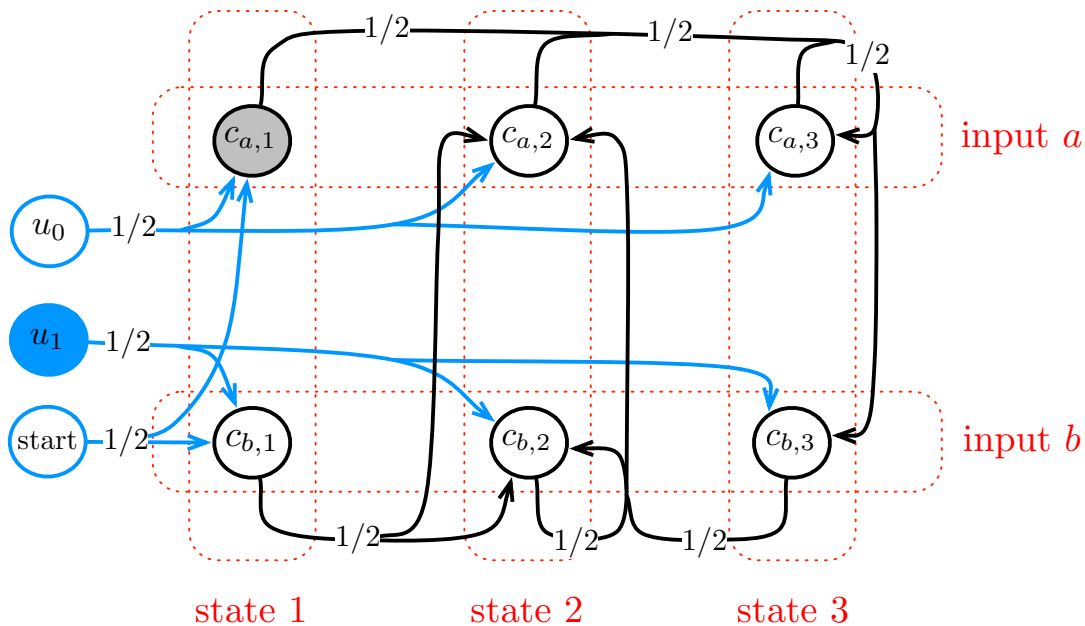


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

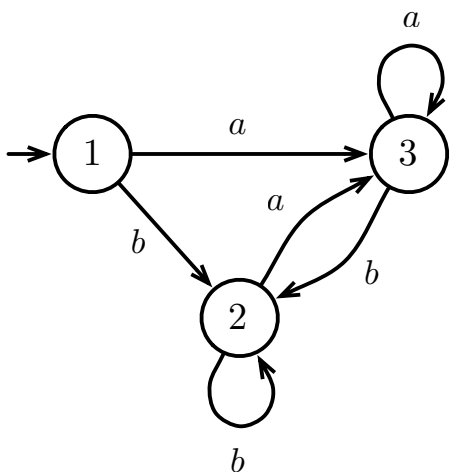


Boolean Neural Network

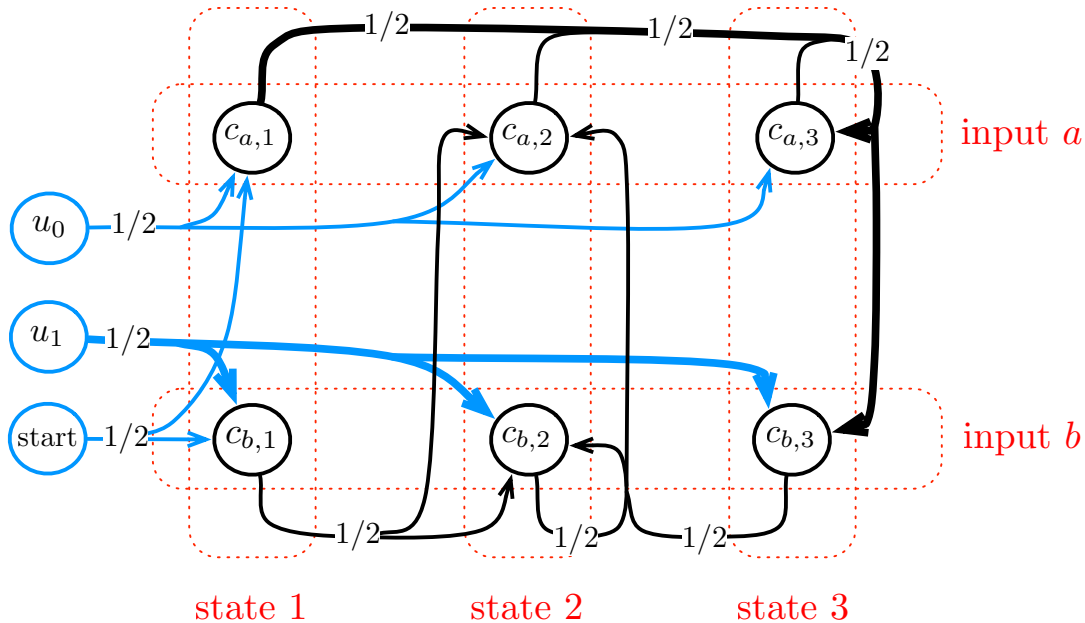


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

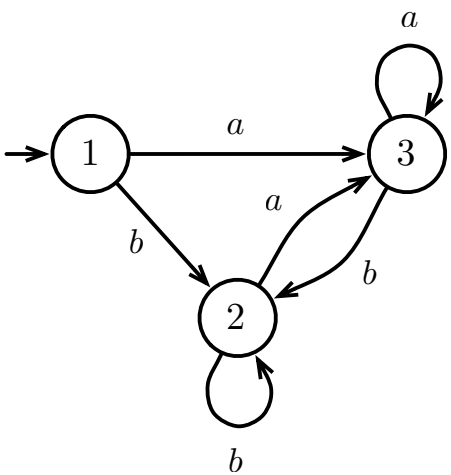


Boolean Neural Network

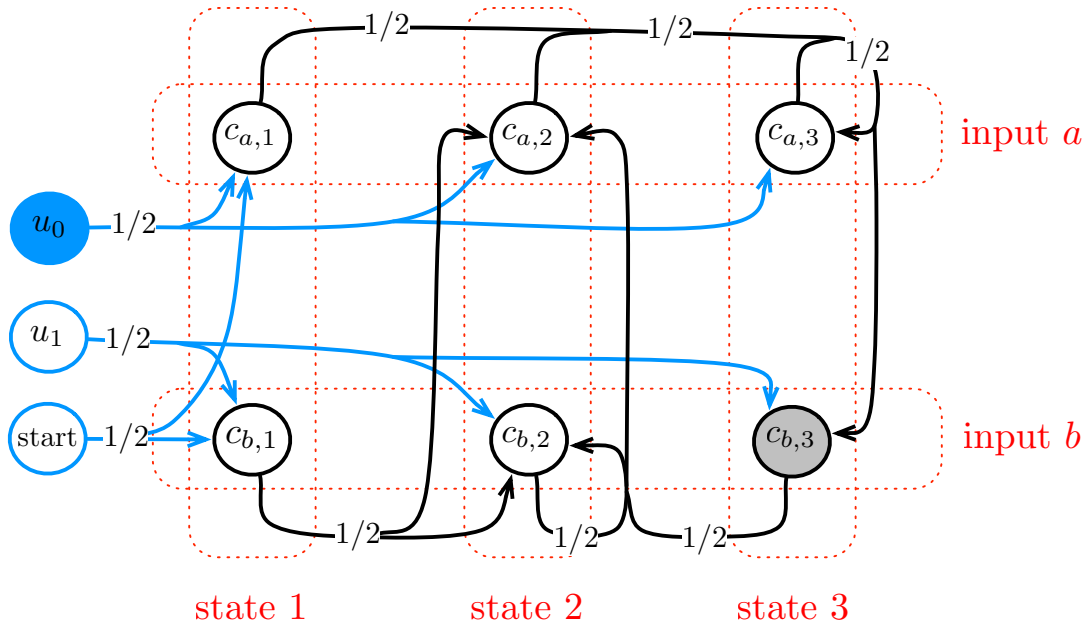


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

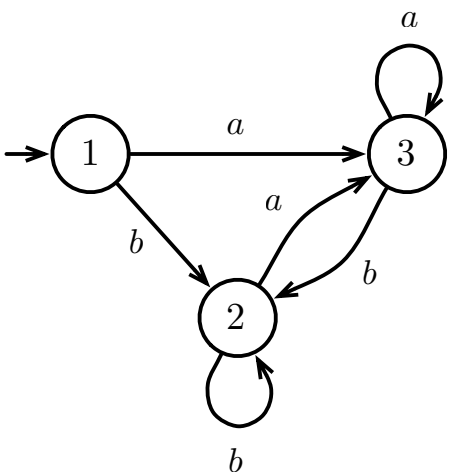


Boolean Neural Network

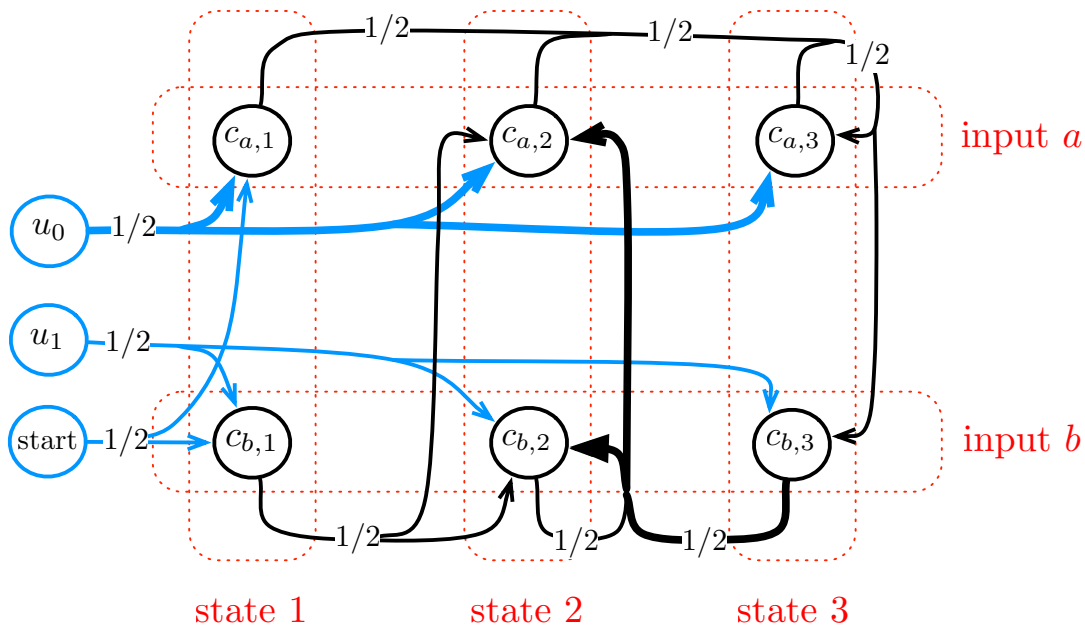


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

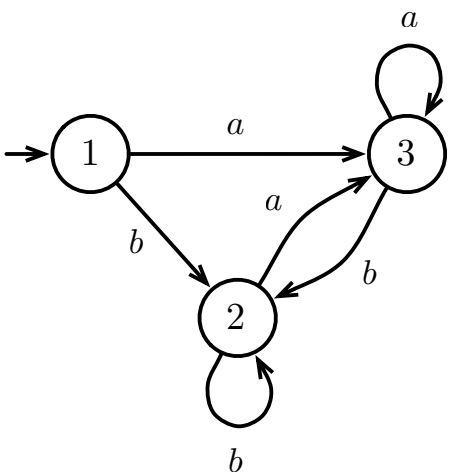


Boolean Neural Network

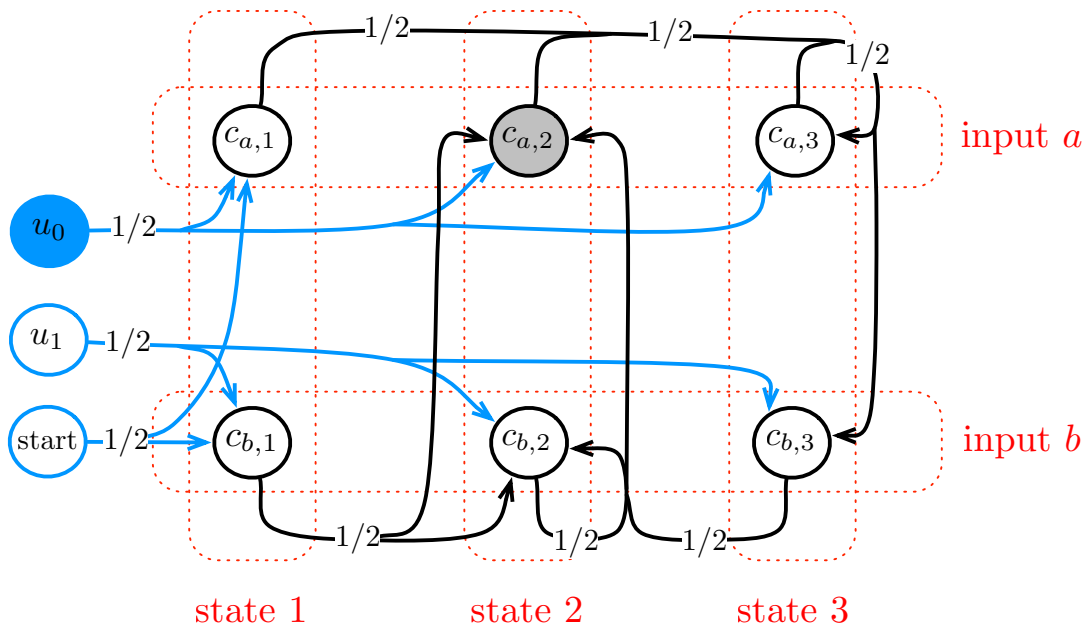


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

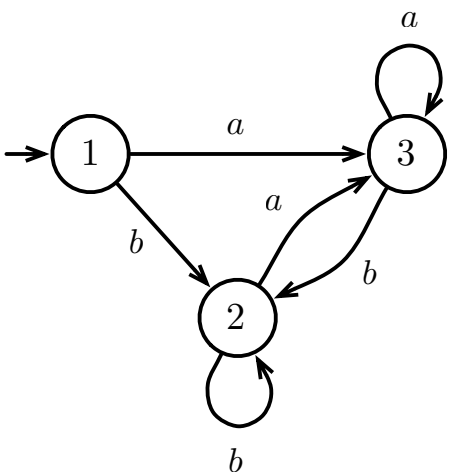


Boolean Neural Network

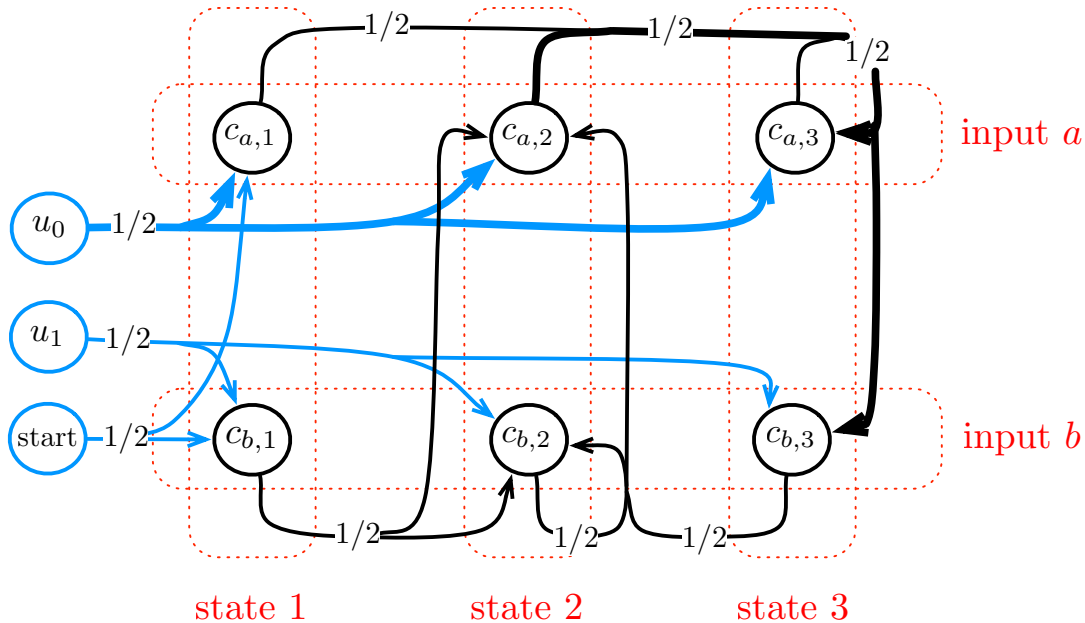


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

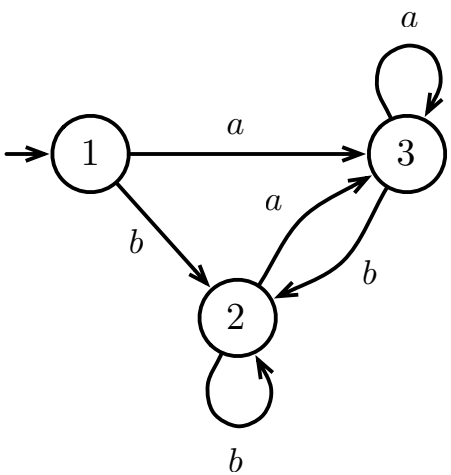


Boolean Neural Network

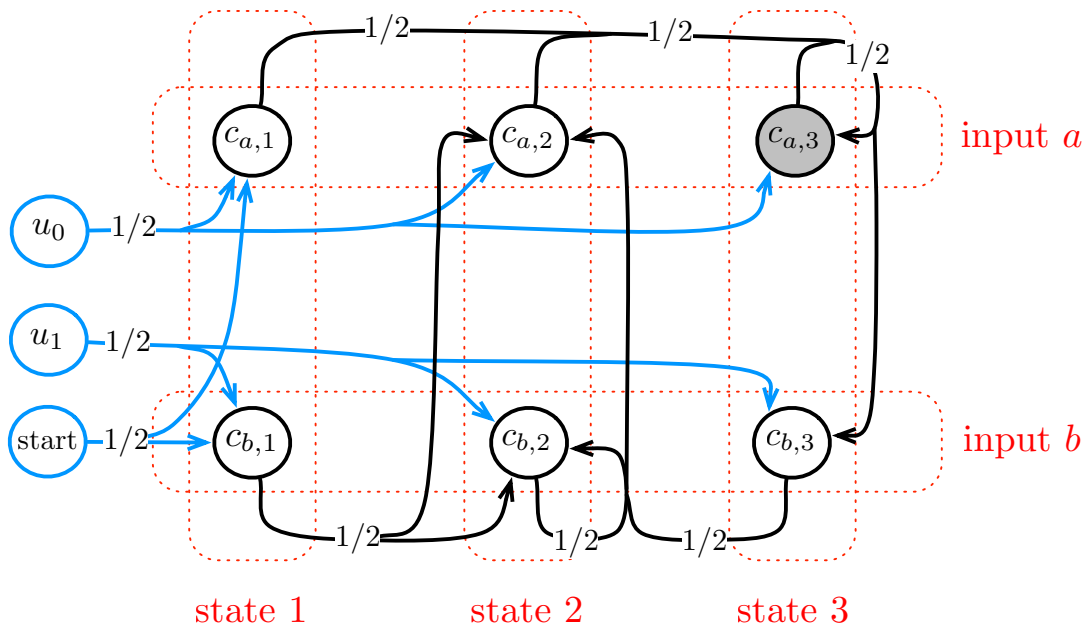


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton

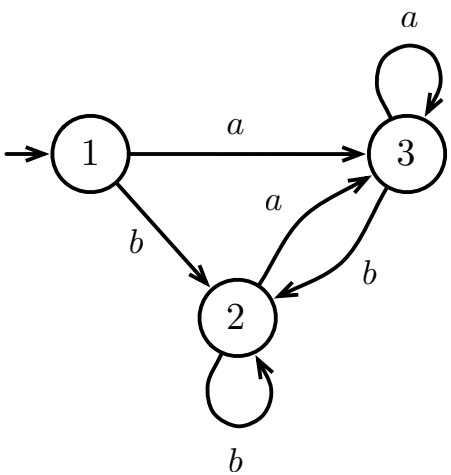


Boolean Neural Network

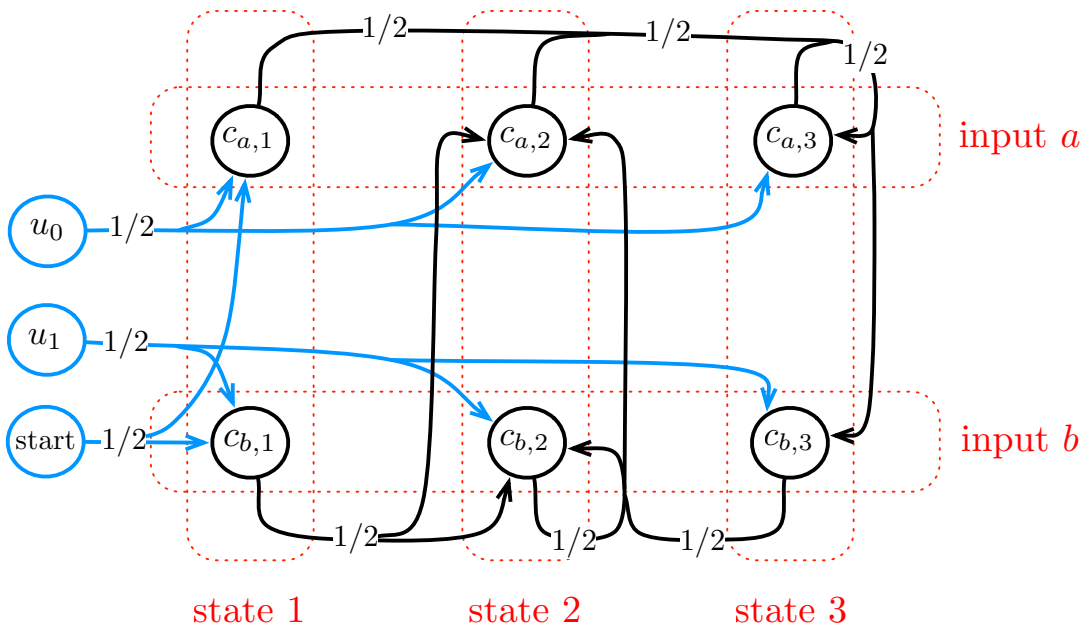


From Automata to Boolean Neural Networks Minsky's Construction (not optimal)

Automaton



Boolean Neural Network



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

Drawbacks of the Construction

Two main drawbacks of this construction:

- Computational states of the automaton are represented by single spiking configurations (or activation vectors) 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 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 are represented by single spiking configurations (or activation vectors) 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 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 are represented by single spiking configurations (or activation vectors) 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 changing of weights or removal of connections.
  - ⋆ Network should be robust to such *synaptic noises*.

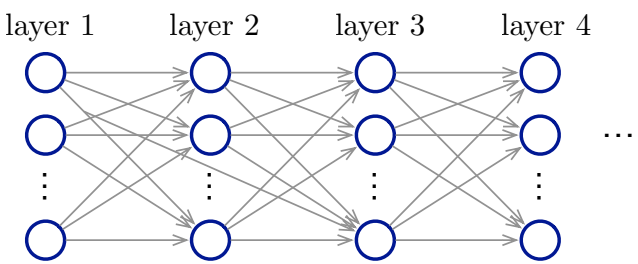
Drawbacks of the Construction

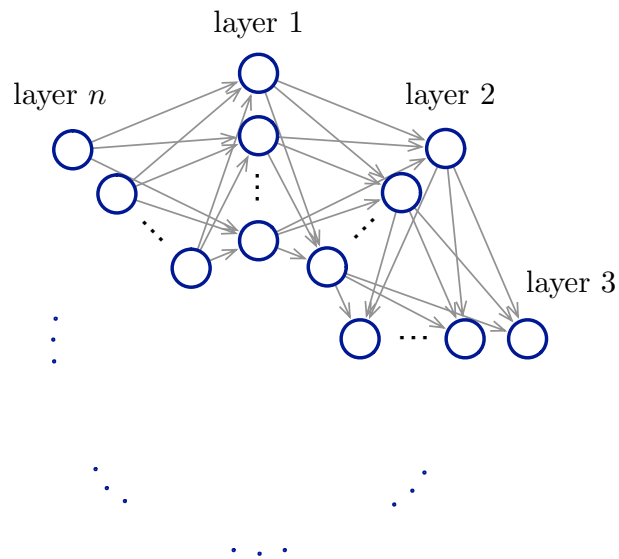
Two main drawbacks of this construction:

- Computational states of the automaton are represented by single spiking configurations (or activation vectors) 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 changing of weights or removal of connections.
  - ⋆ Network should be robust to such *synaptic noises*.

Synfire Chains and Synfire Rings

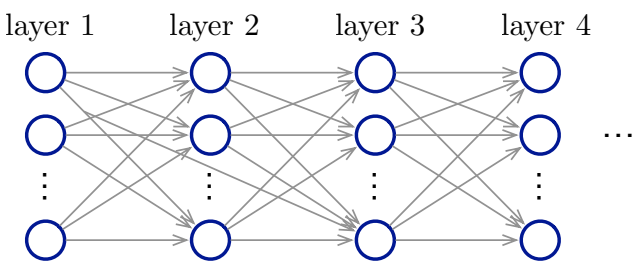
- *Synfire chains* are likely to be crucially involved in the processing and coding of information (Abeles 82).
- Spontaneous emergence of *synfire rings* (looping synfire chains) has been observed in self-organizing neural networks (Zheng & Triesch 14).
  - ⋆ *Computational states could be represented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

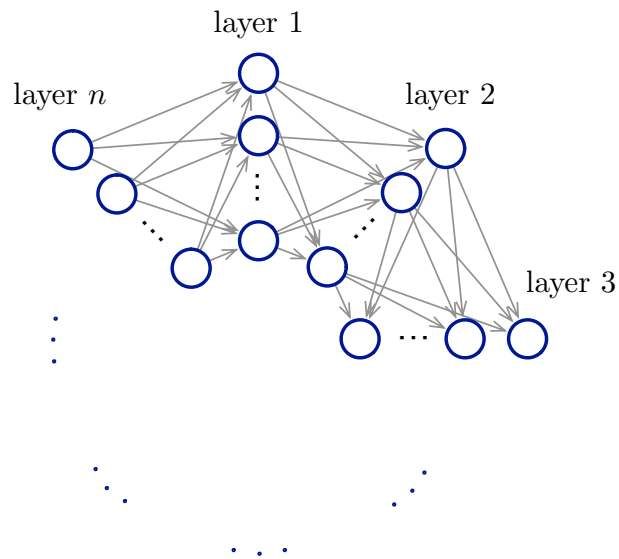




Synfire Chains and Synfire Rings

- *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.*

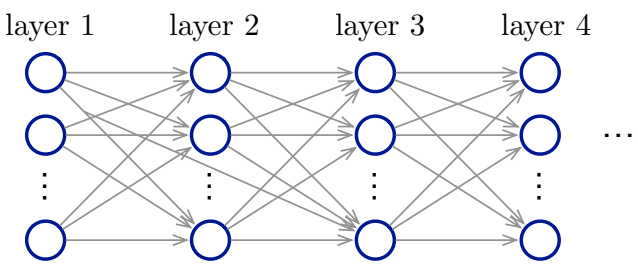


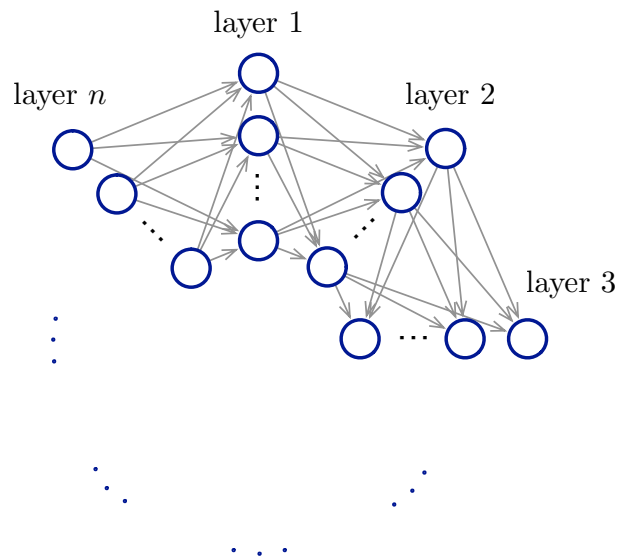


Synfire Chains and Synfire Rings

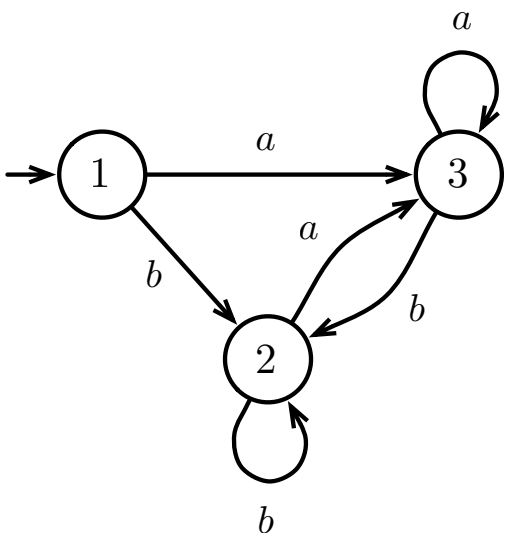
- *Synfire chains* are likely to be crucially involved in the processing and coding of information (Abeles 82).
- Spontaneous emergence of *synfire rings* (looping synfire chains) has been observed in self-organizing neural networks (Zheng & Triesch 14).

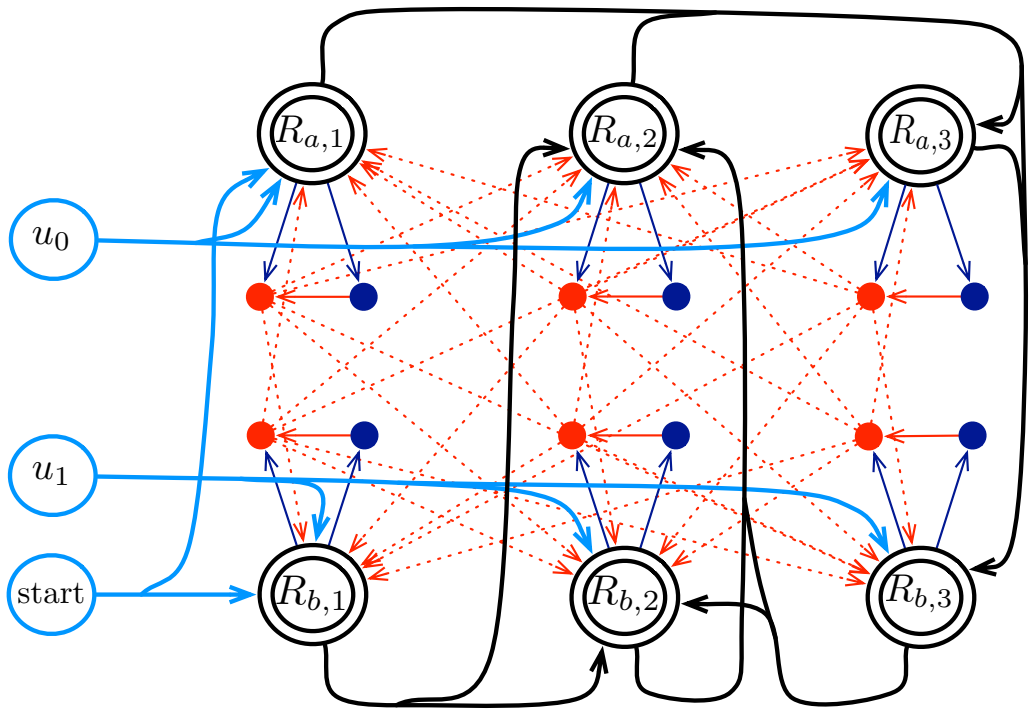
⋆ *Computational states could be represented as sustained activities within synfire rings.*



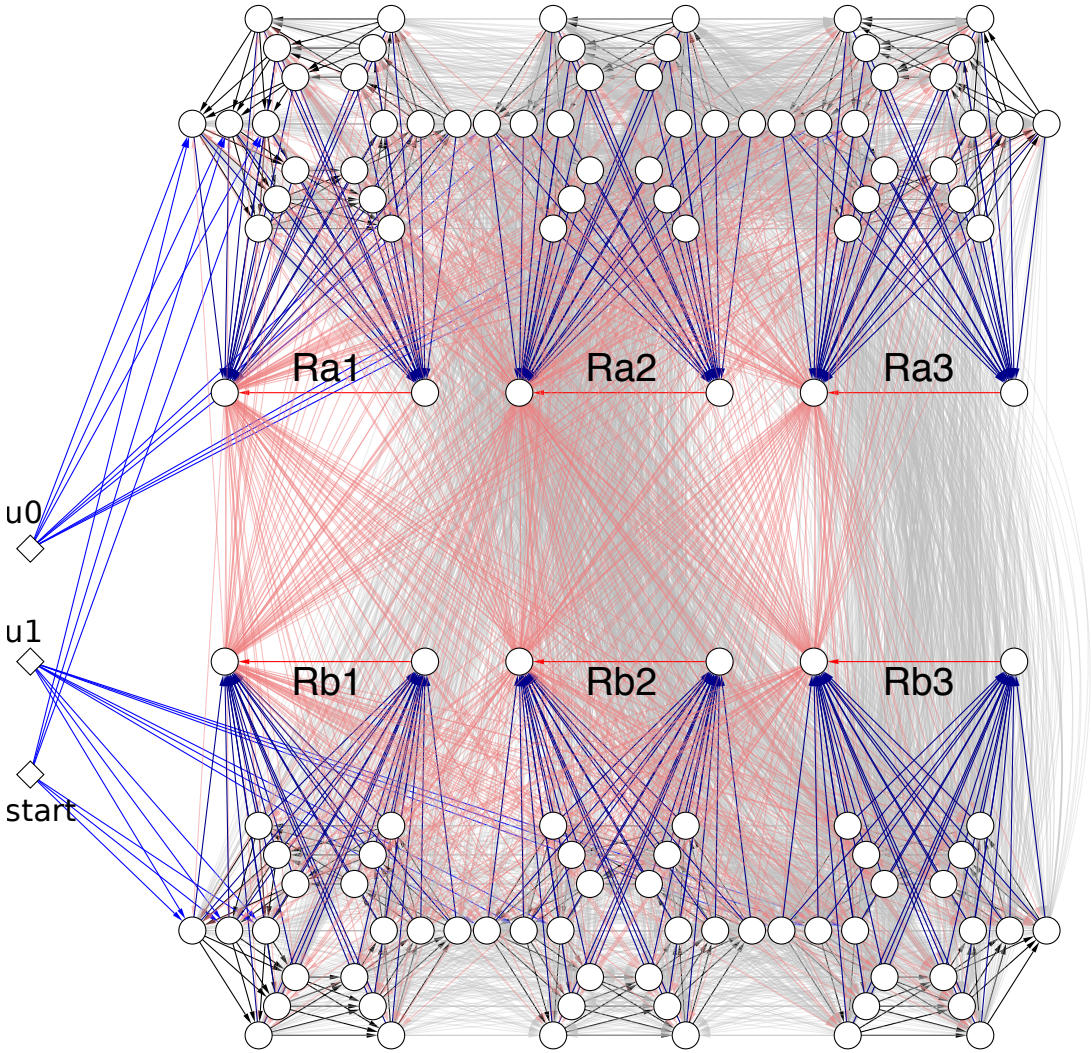


From Automata to Boolean Neural Networks with Synfire Rings

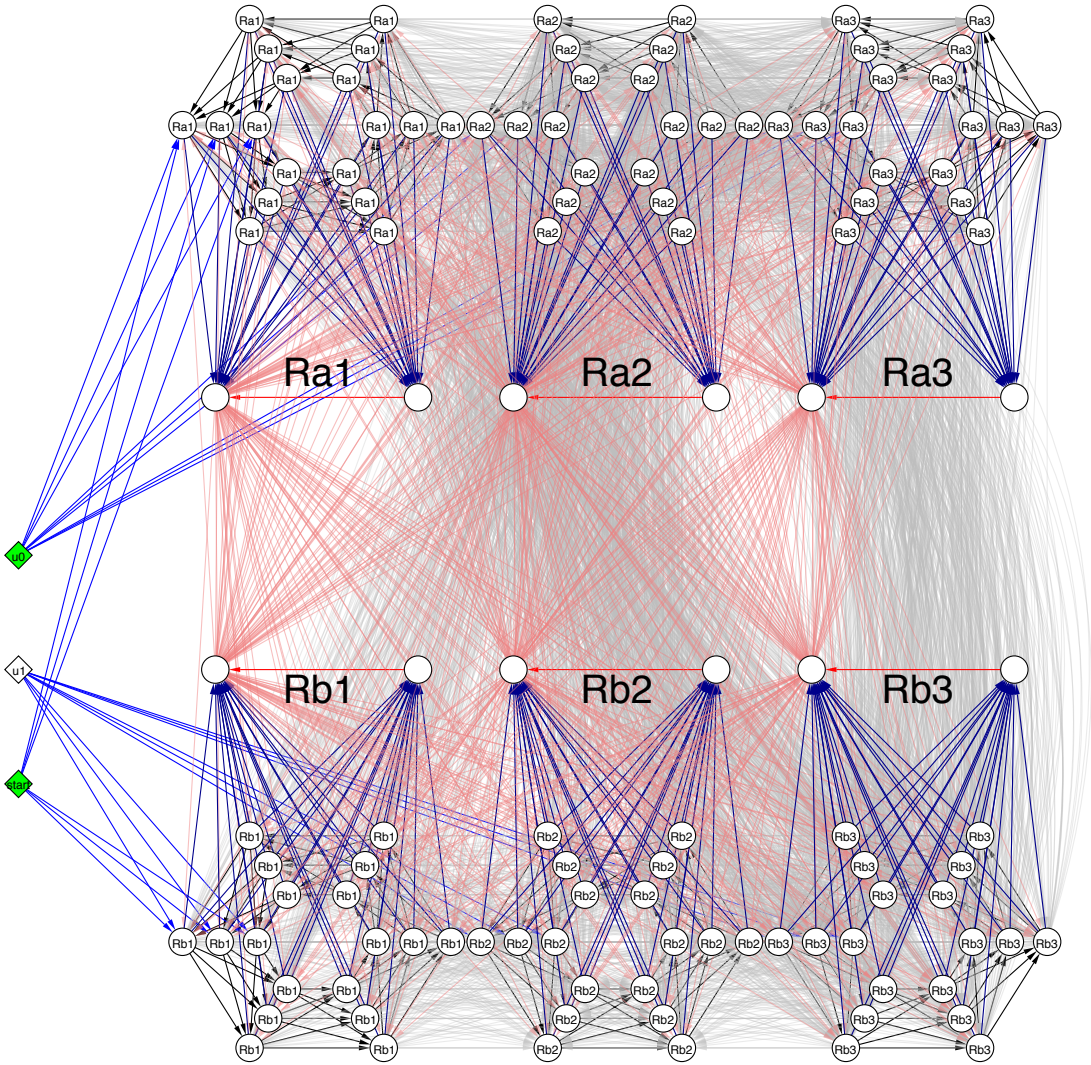




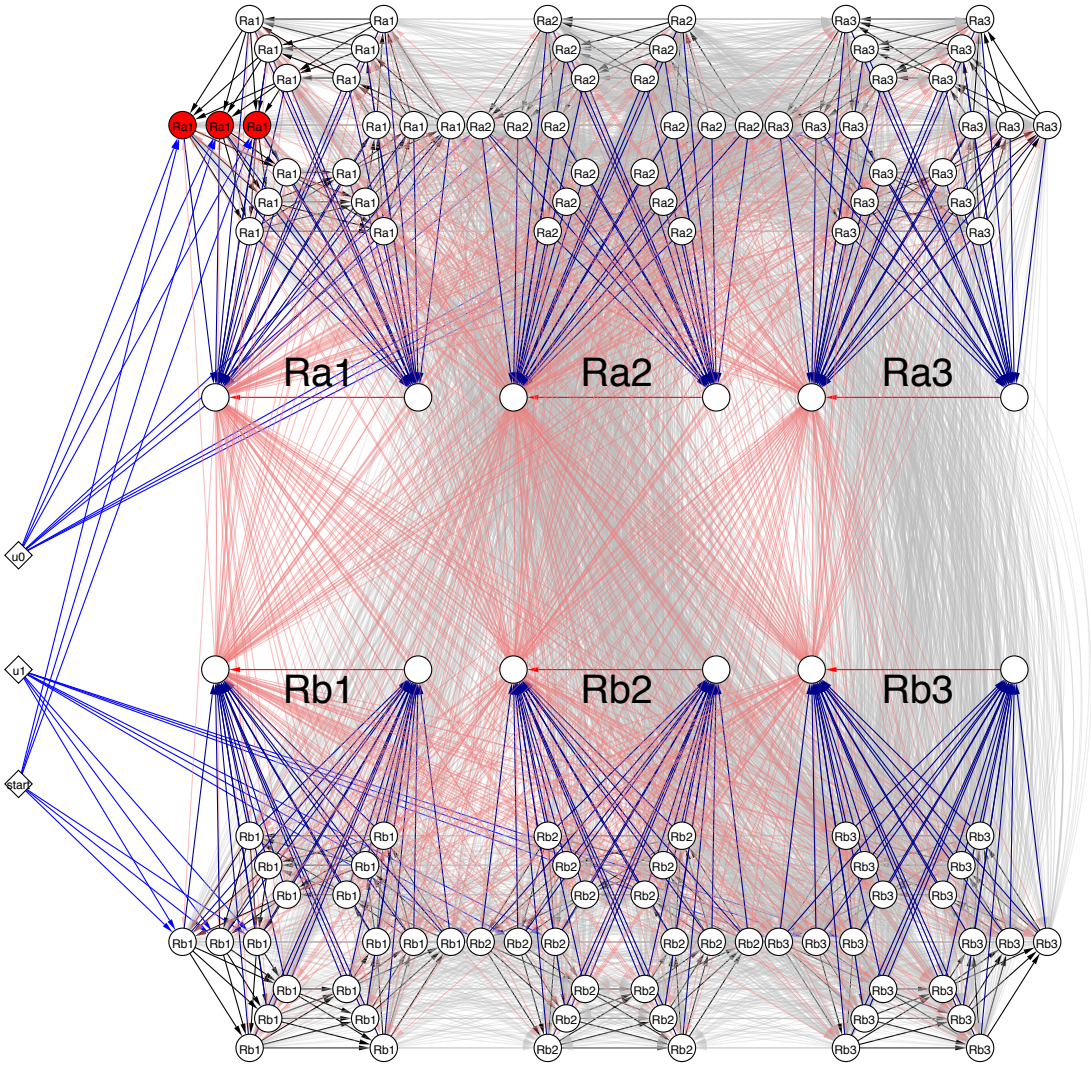
Simulation



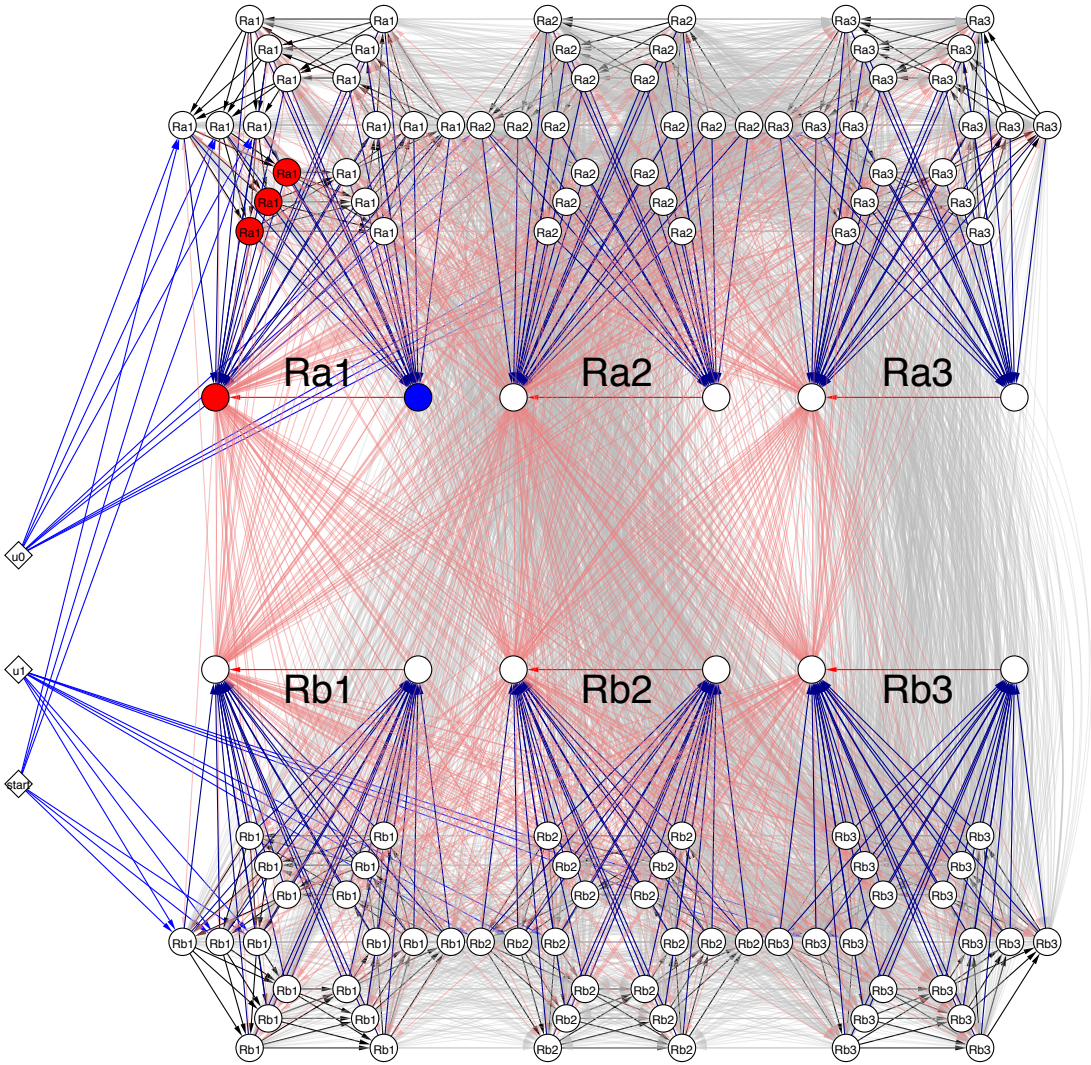
Simulation



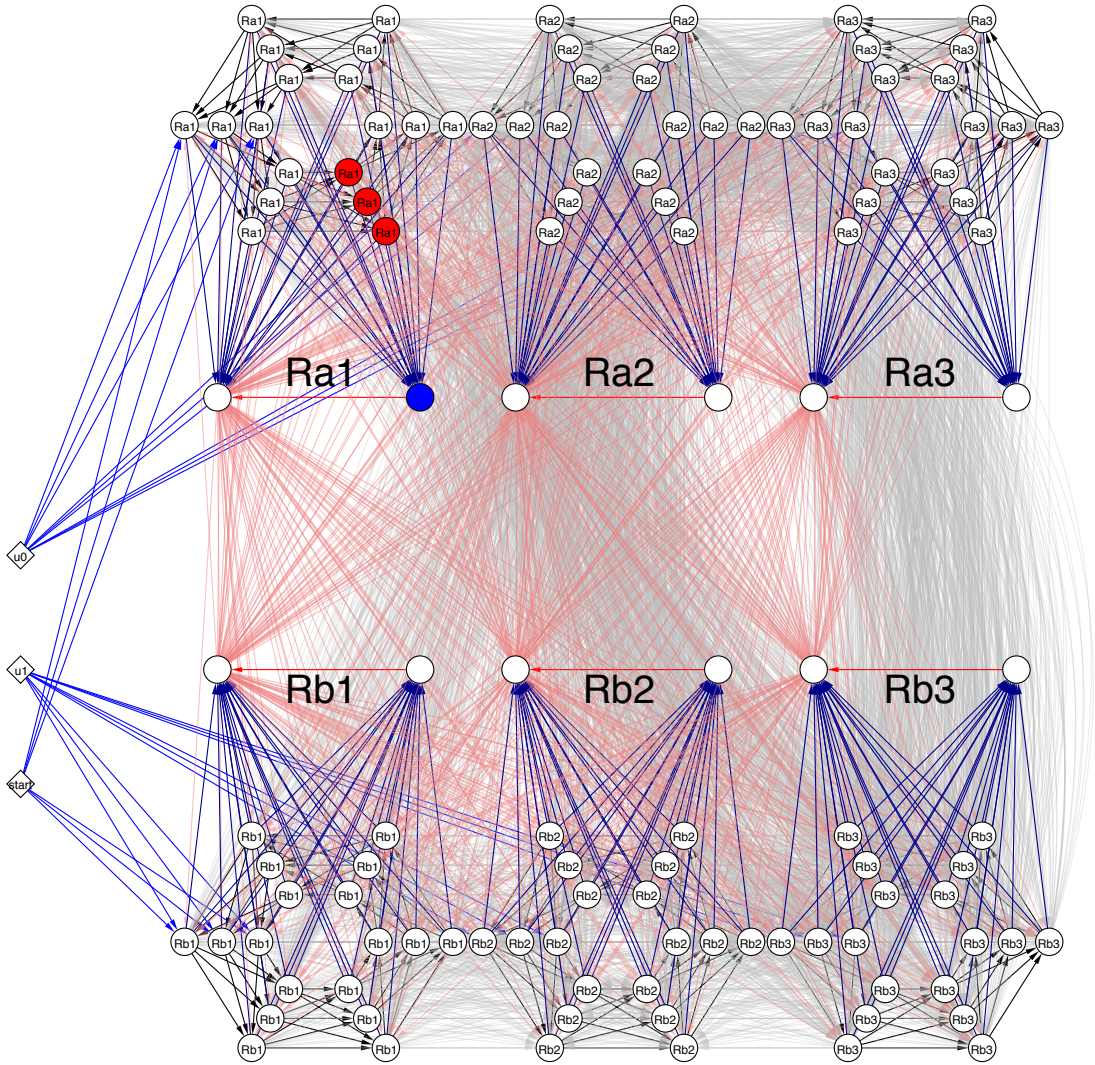
Simulation



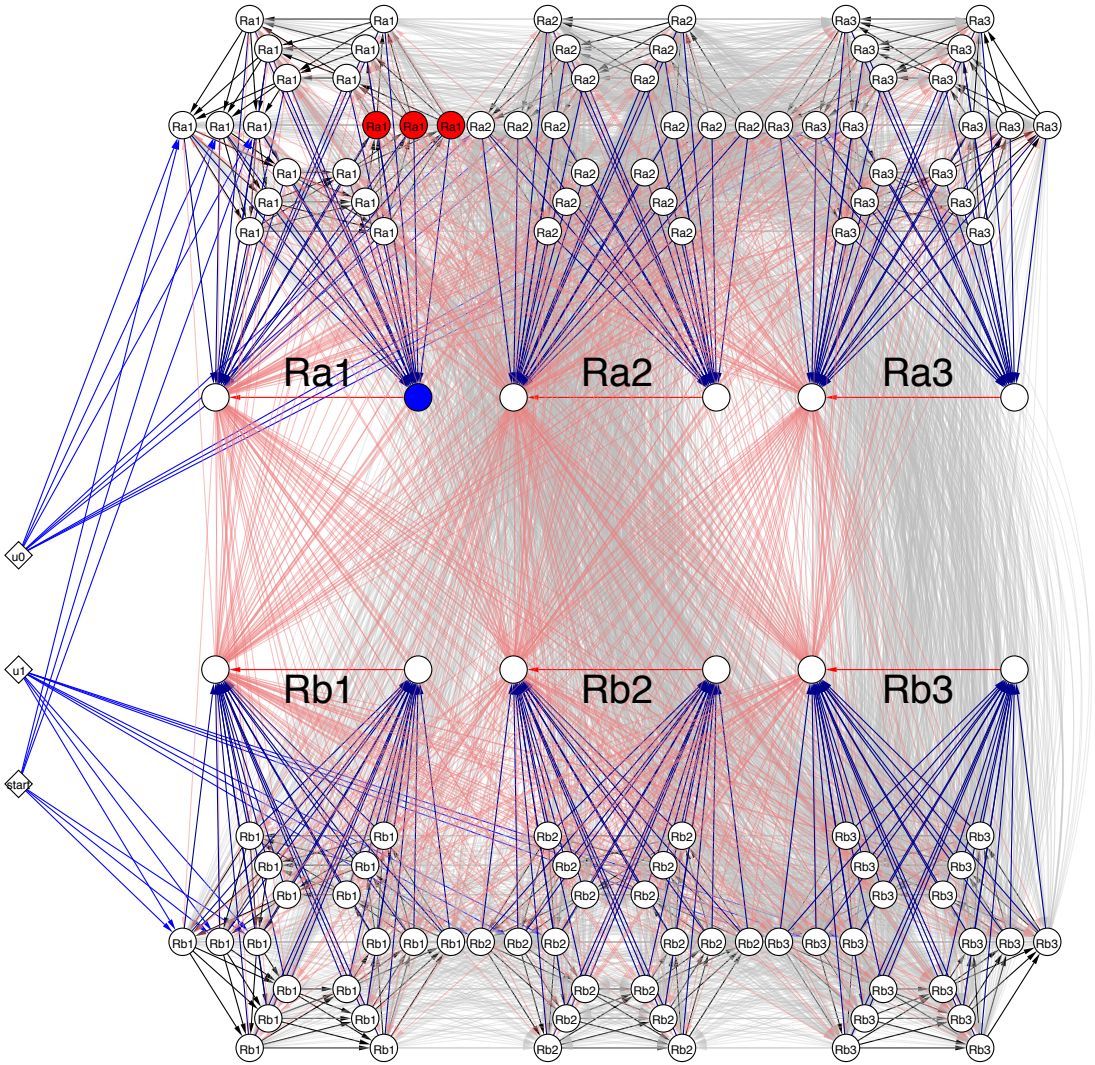
Simulation



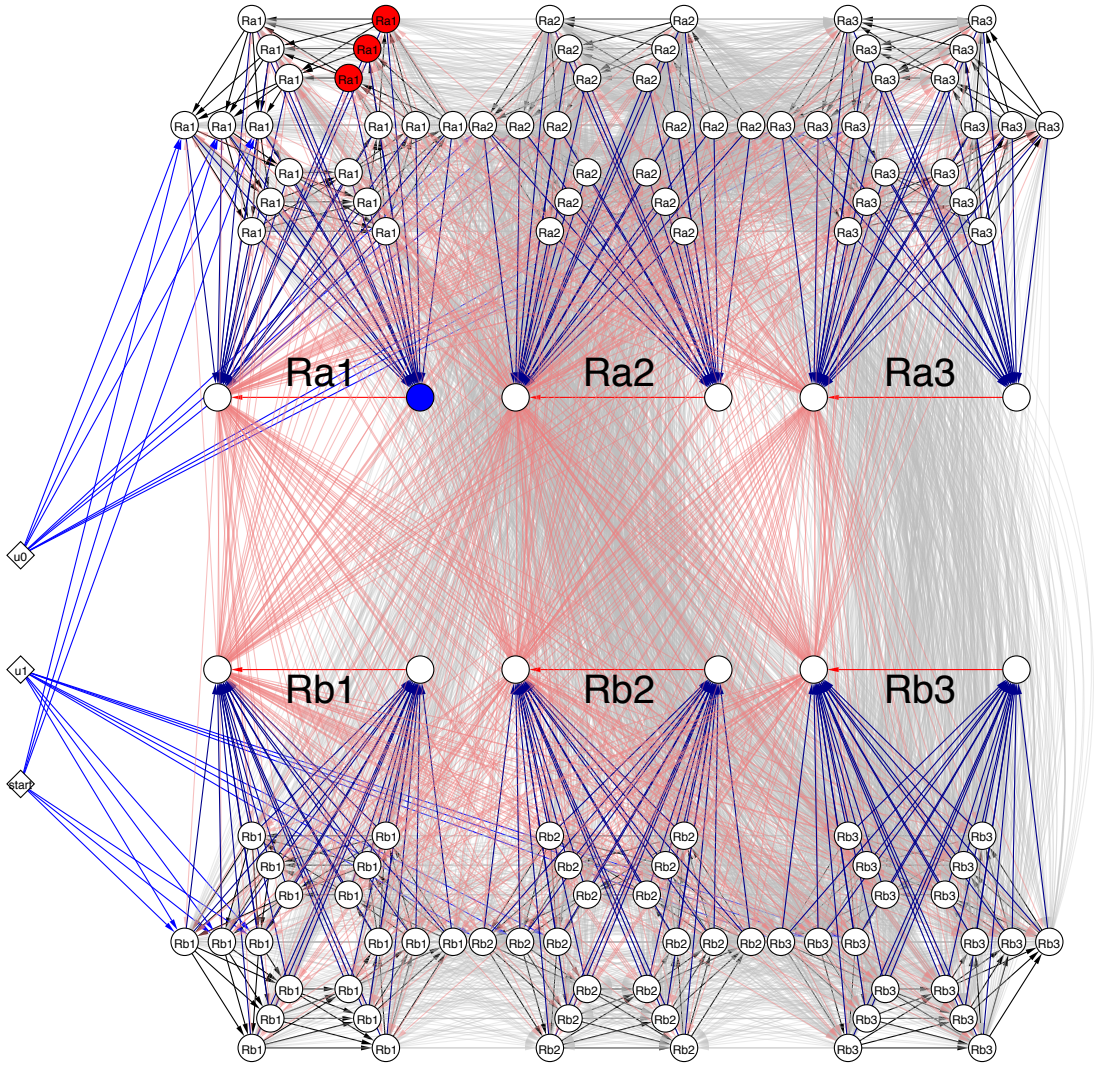
Simulation



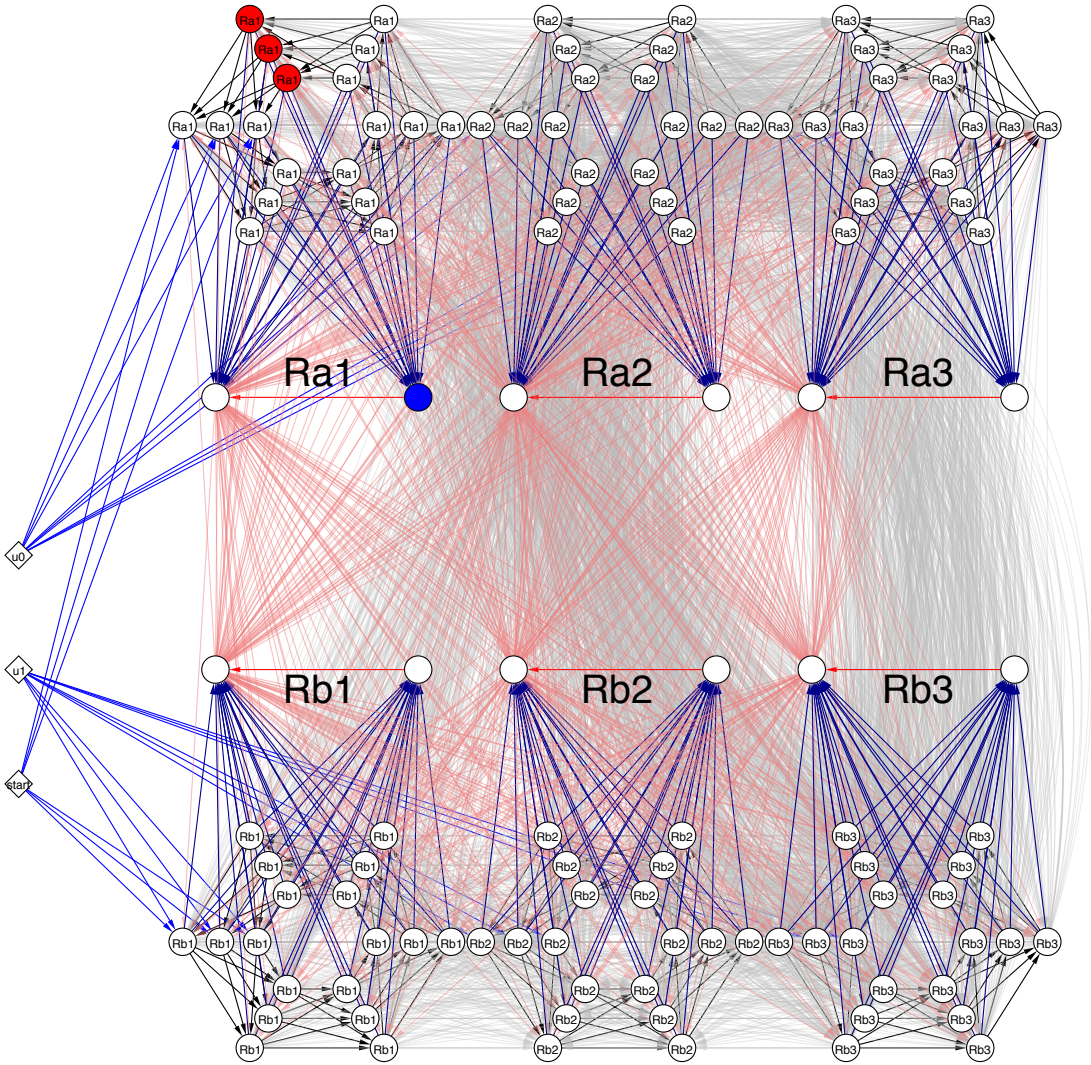
Simulation



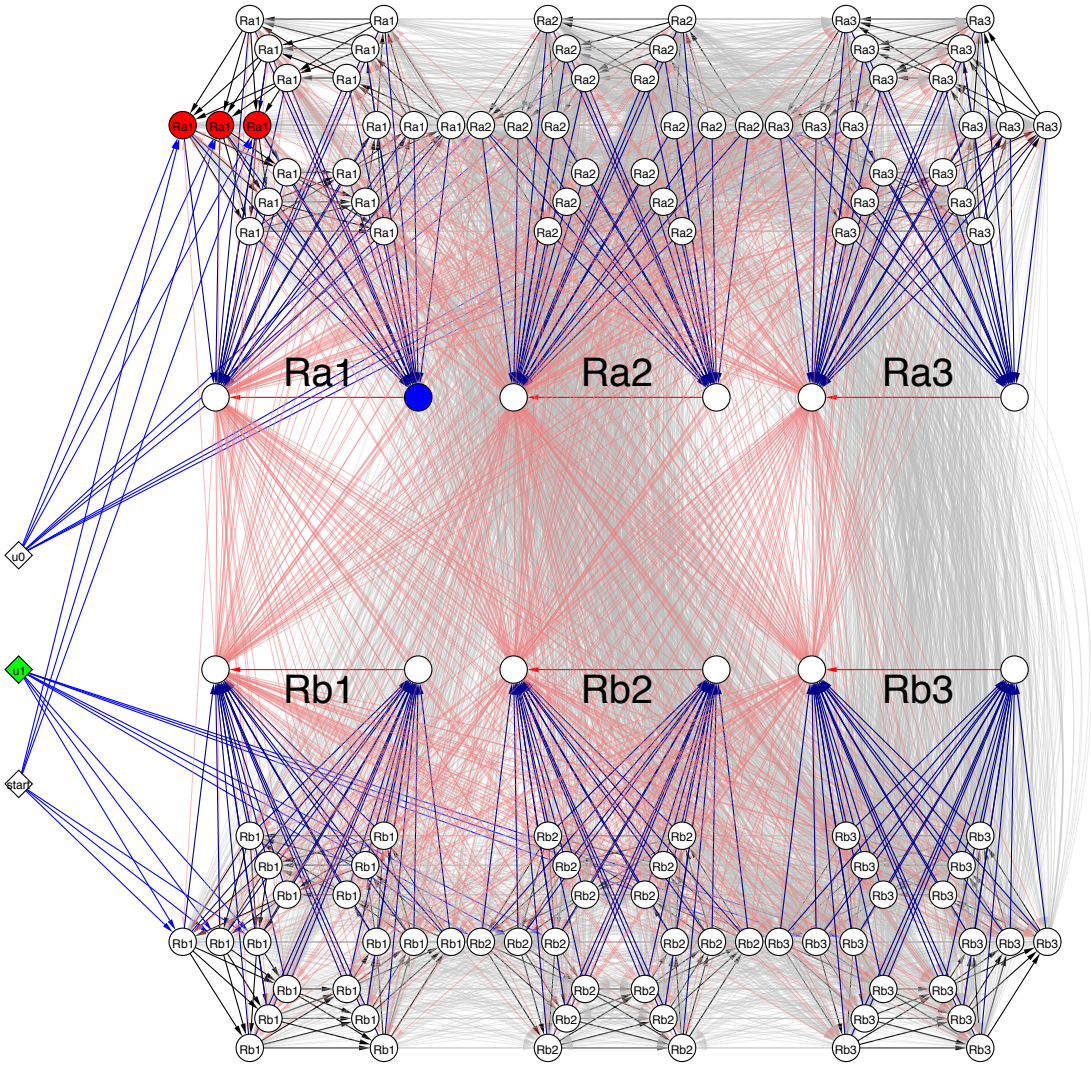
Simulation



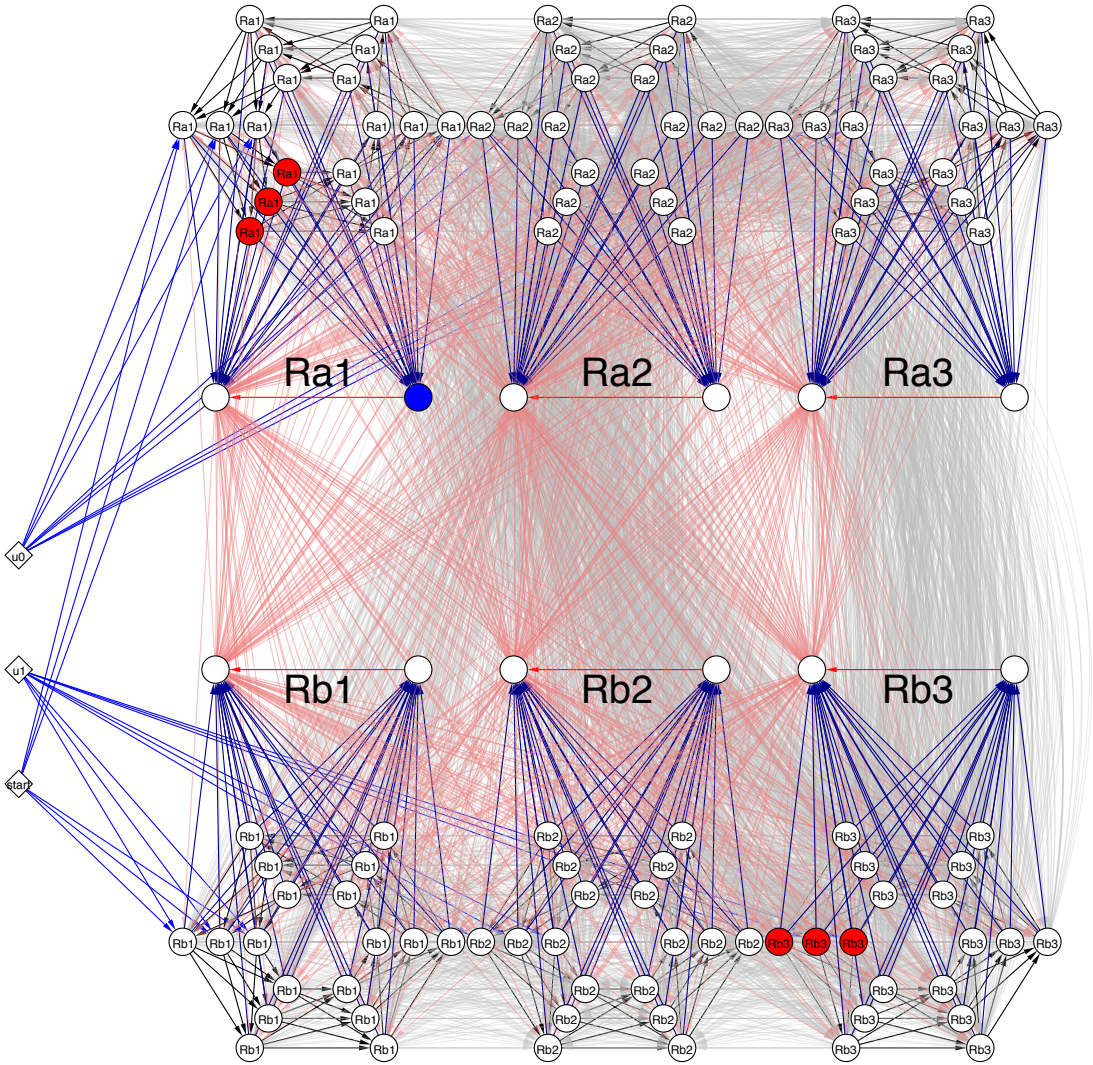
Simulation



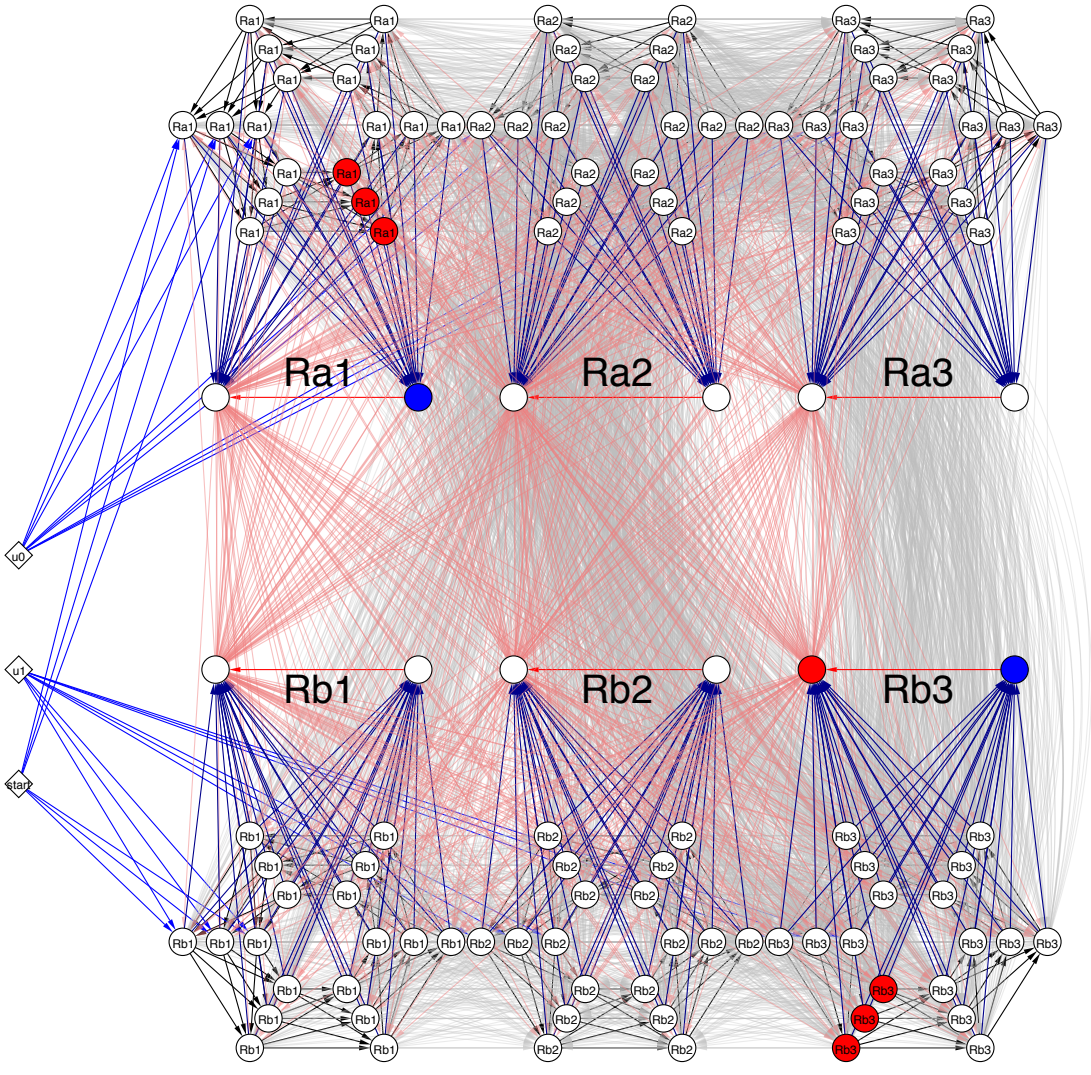
Simulation



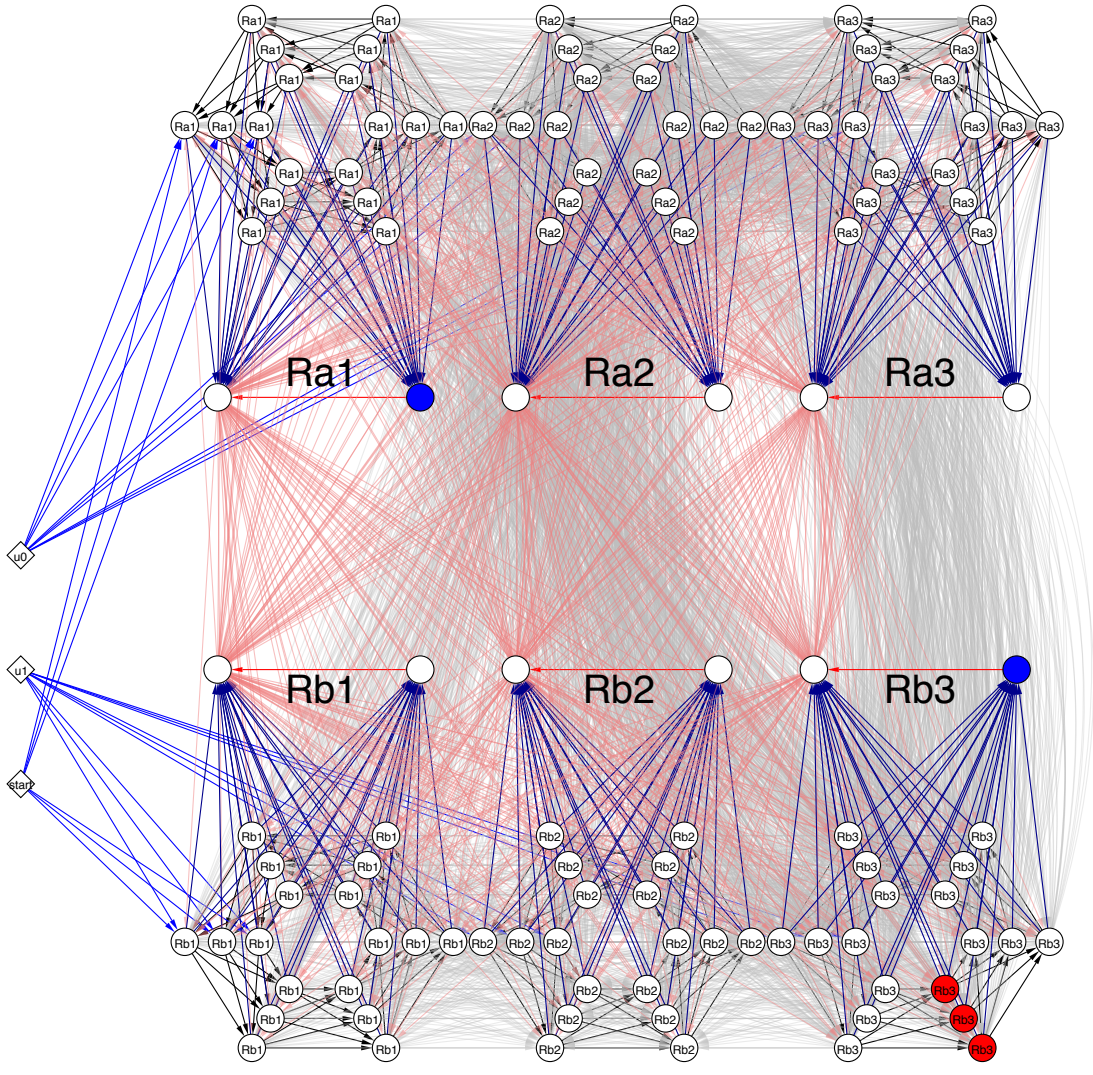
Simulation



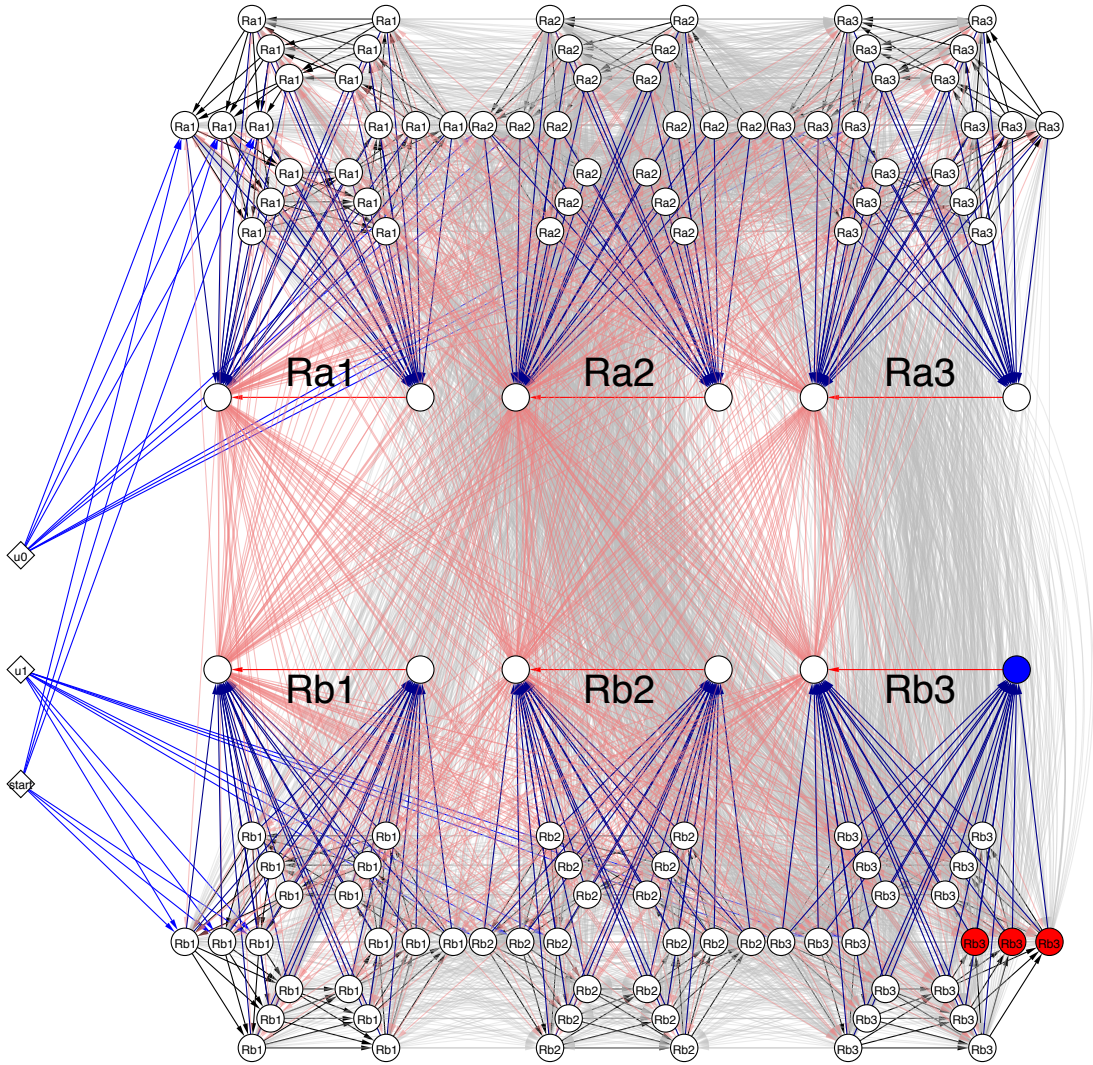
Simulation



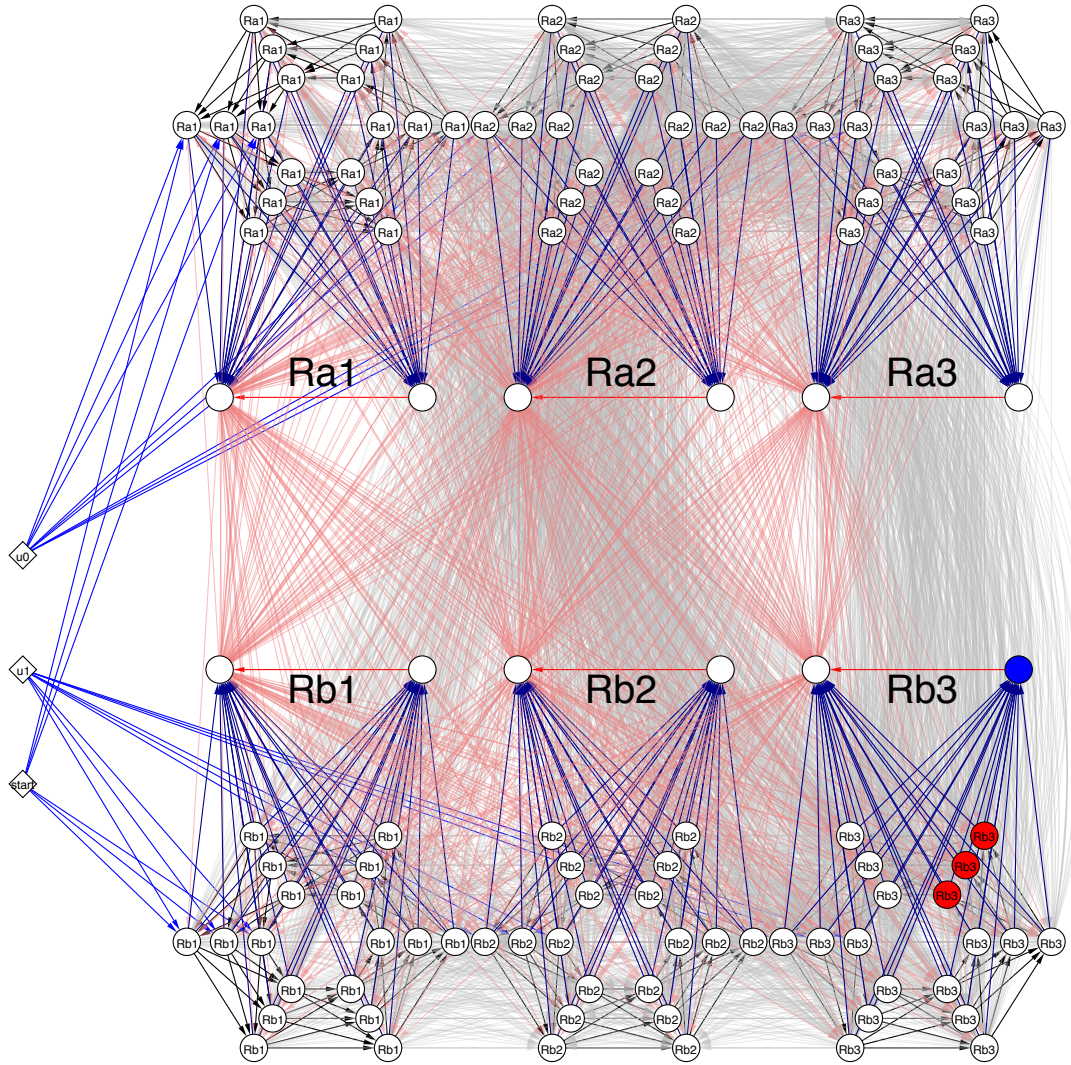
Simulation



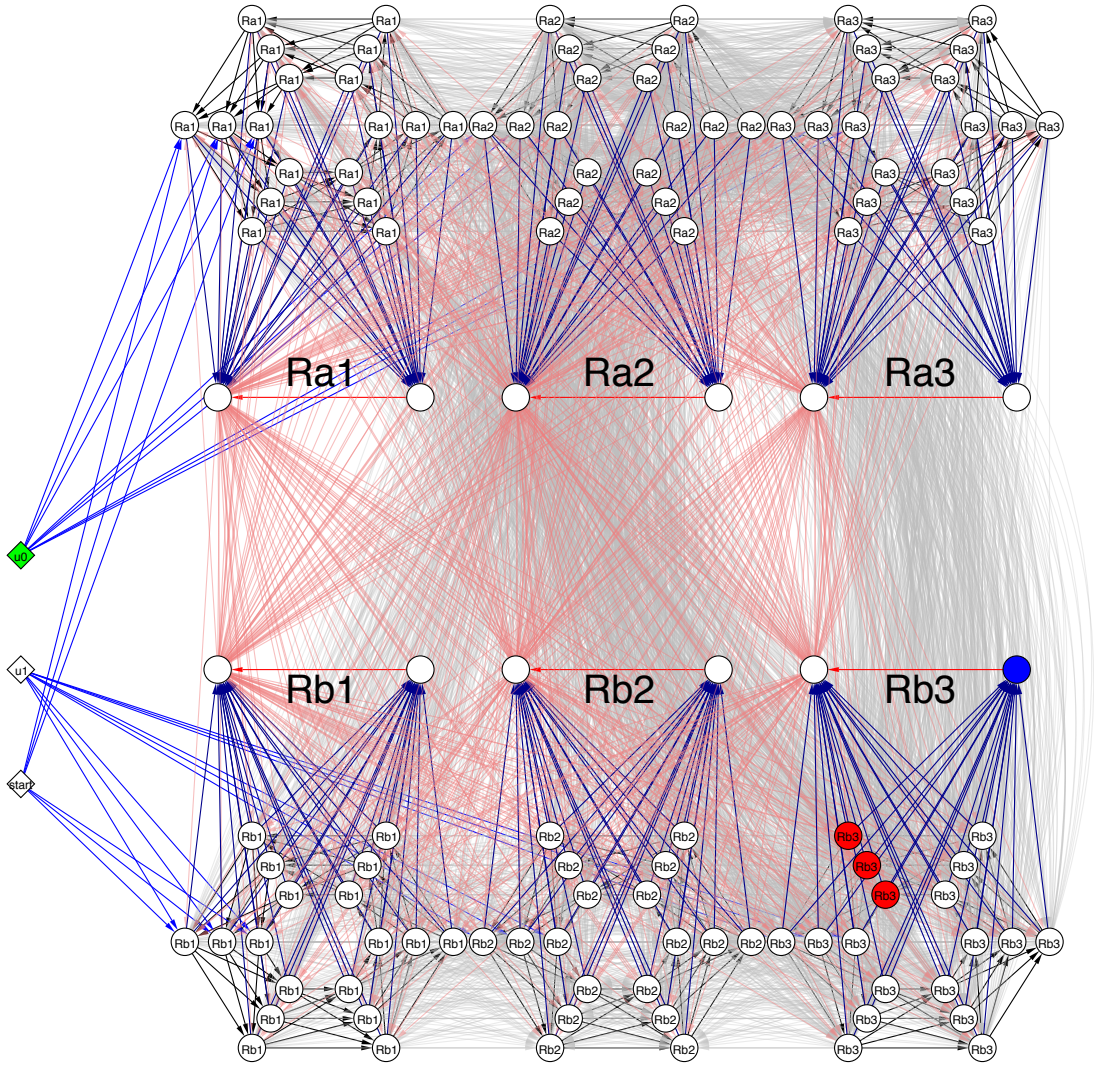
Simulation



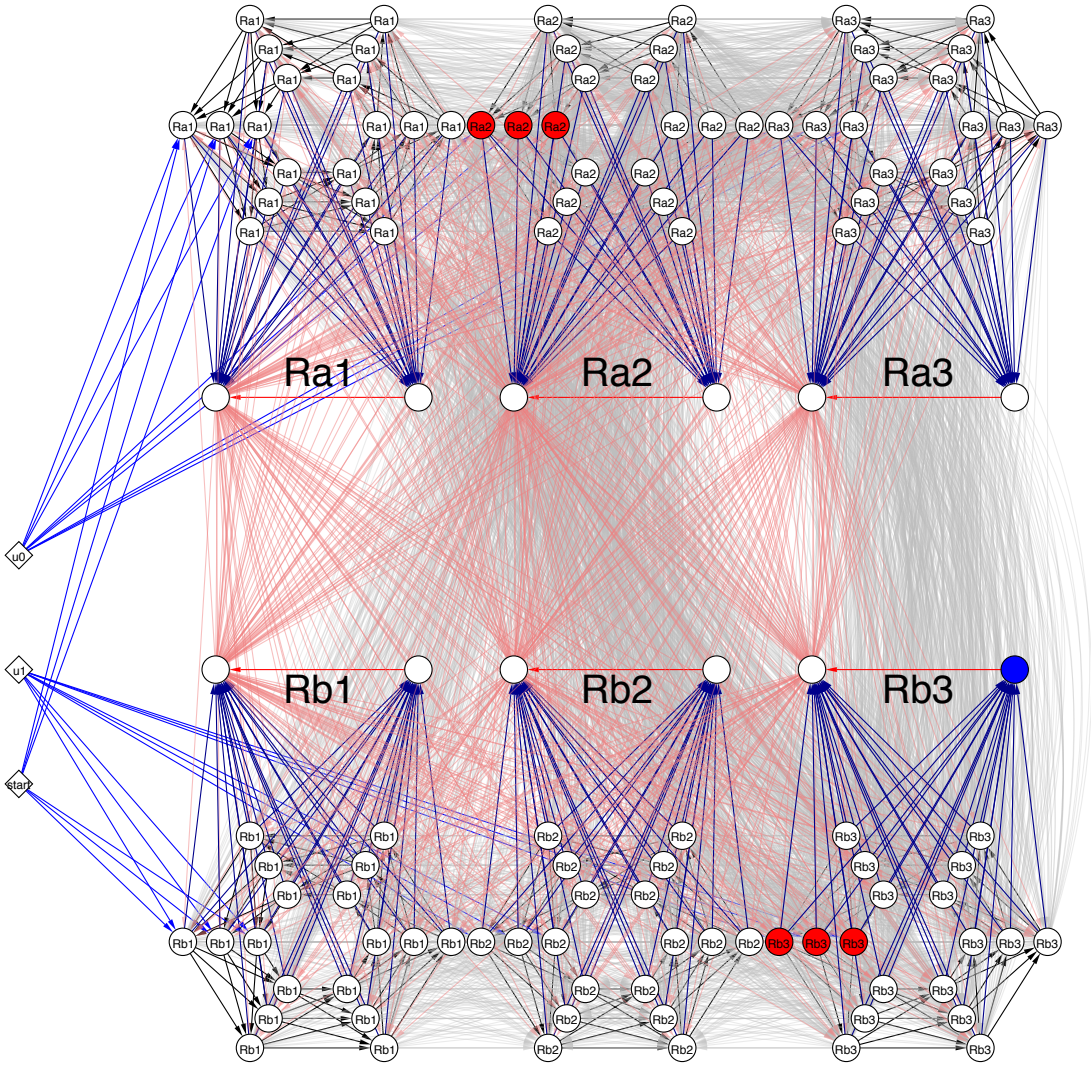
Simulation



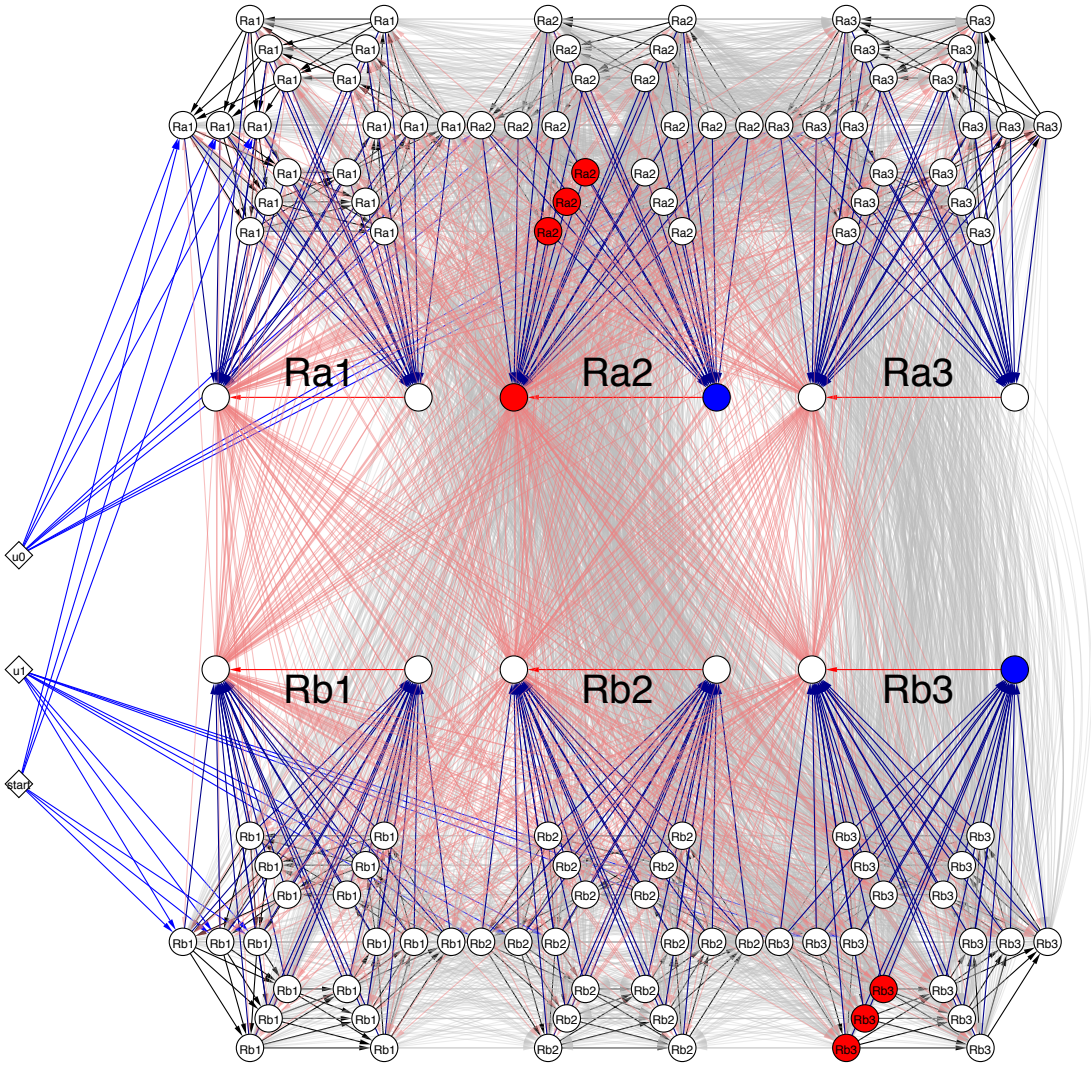
Simulation



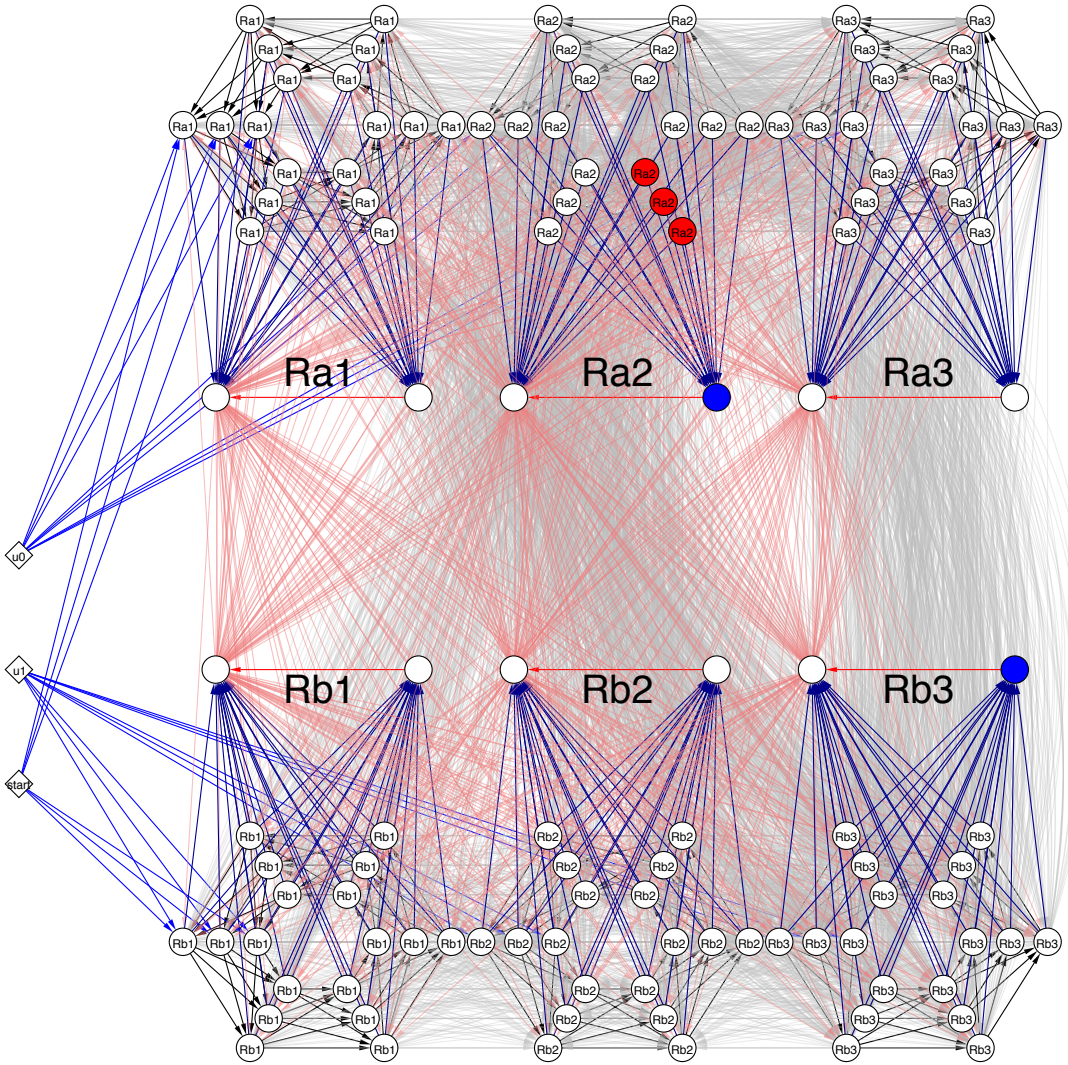
Simulation



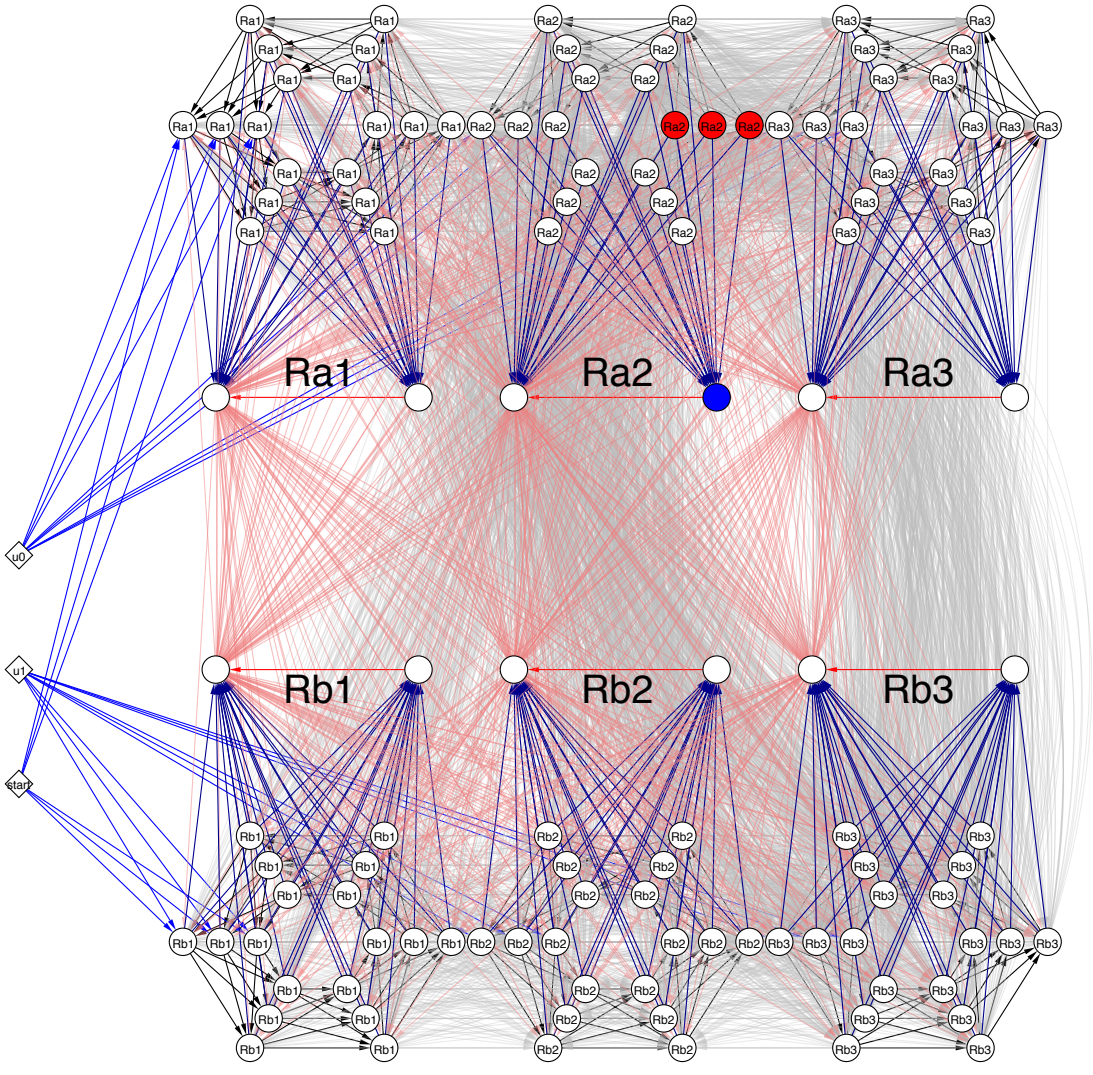
Simulation



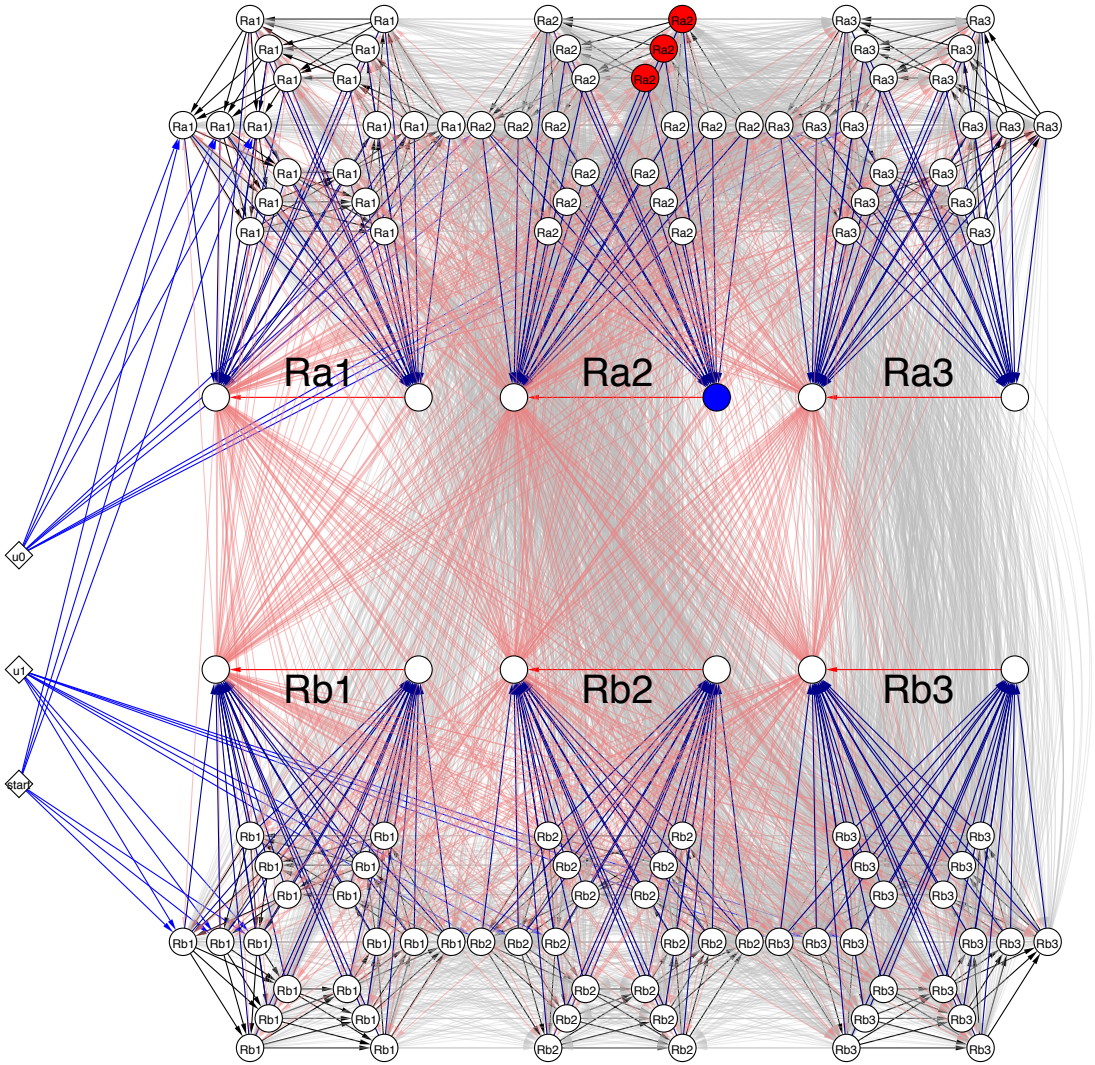
Simulation



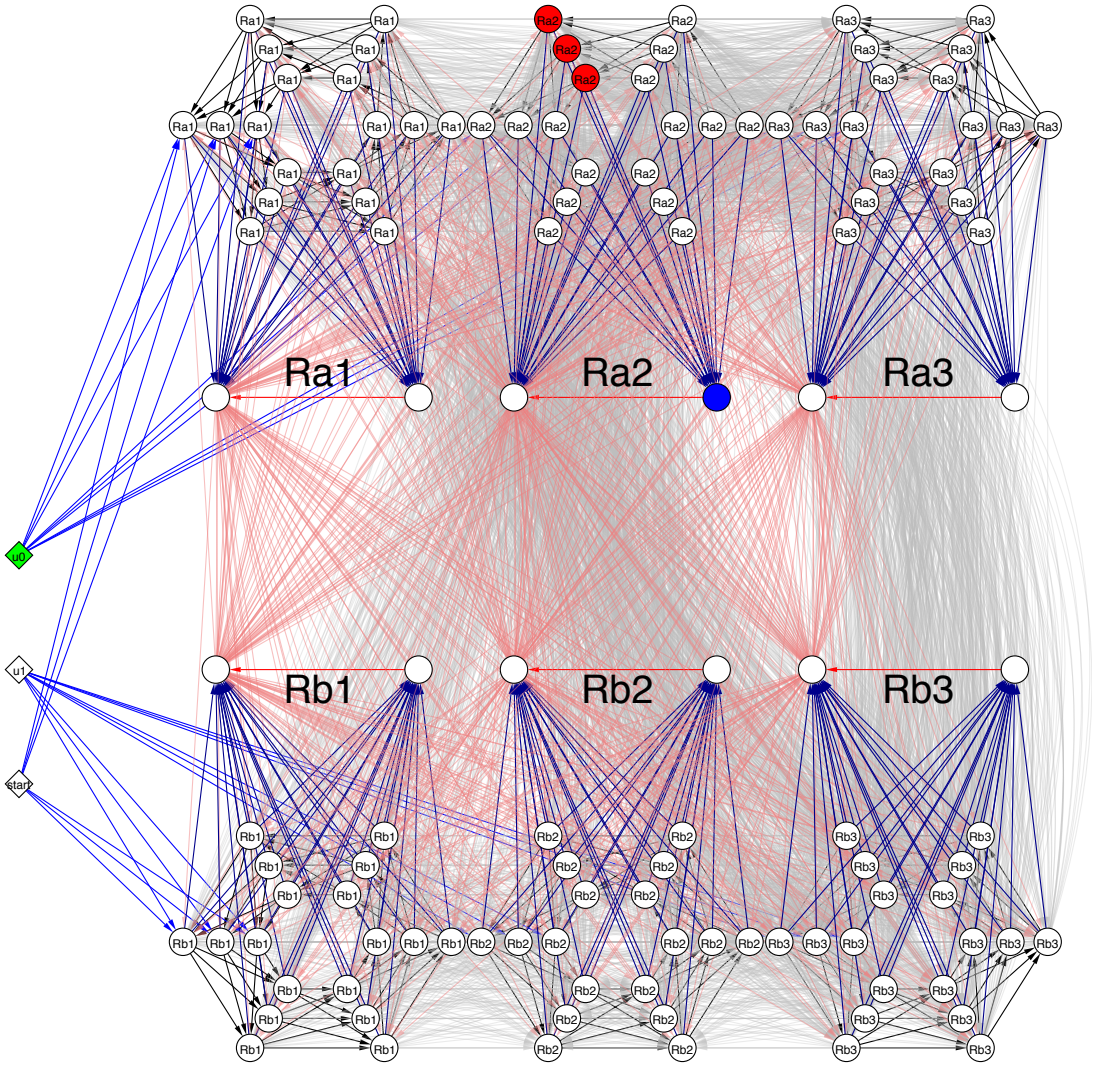
Simulation



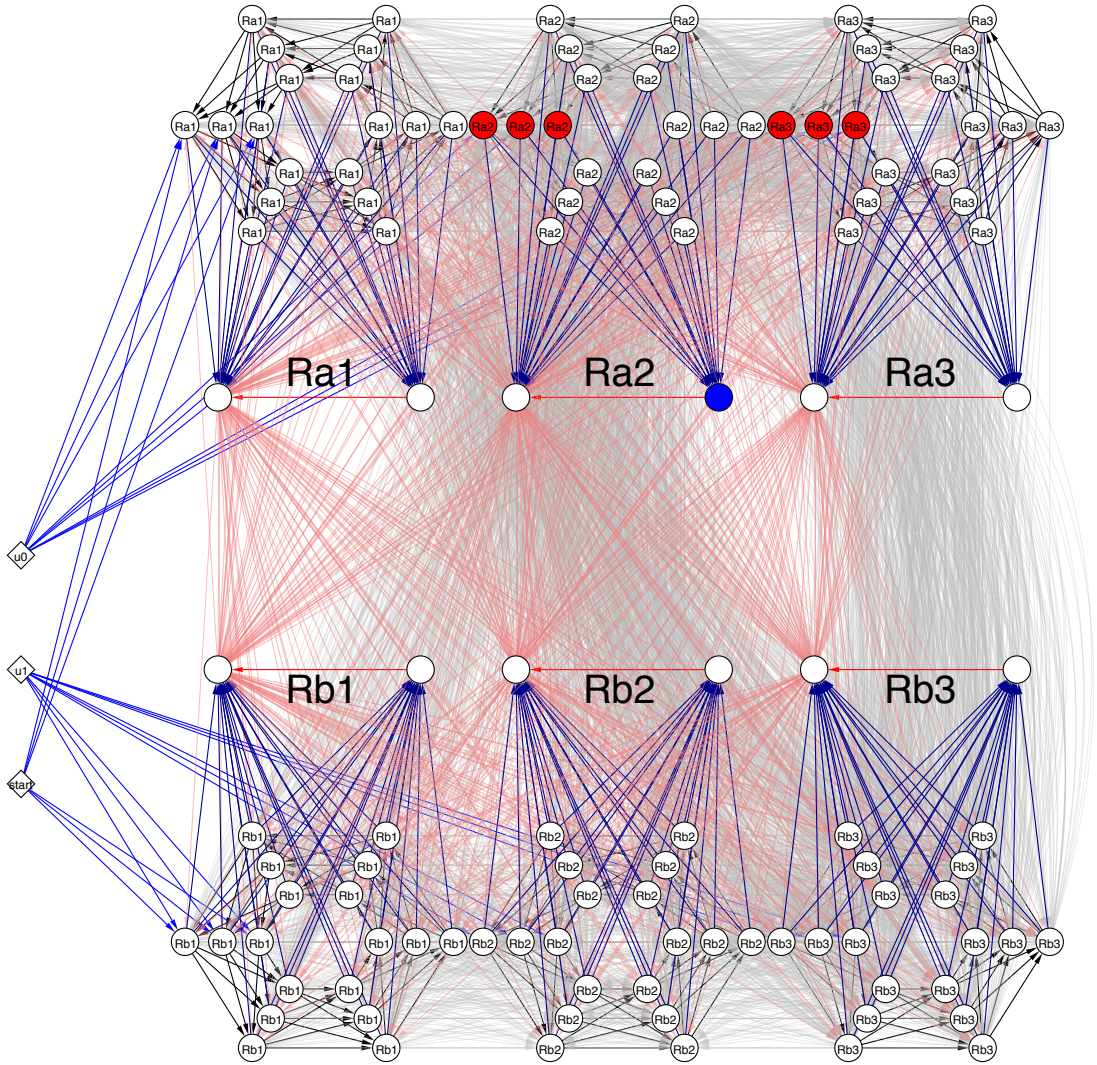
Simulation



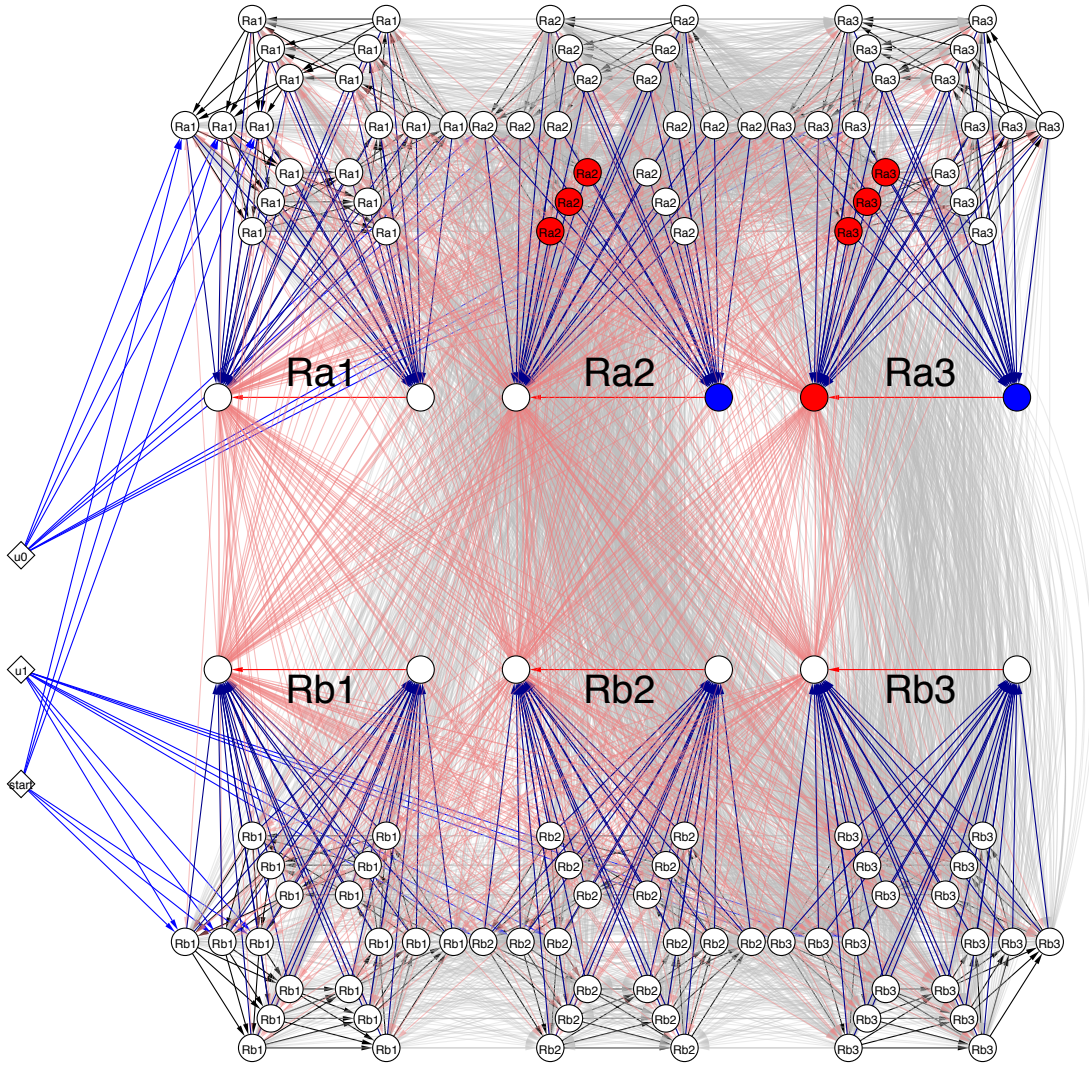
Simulation



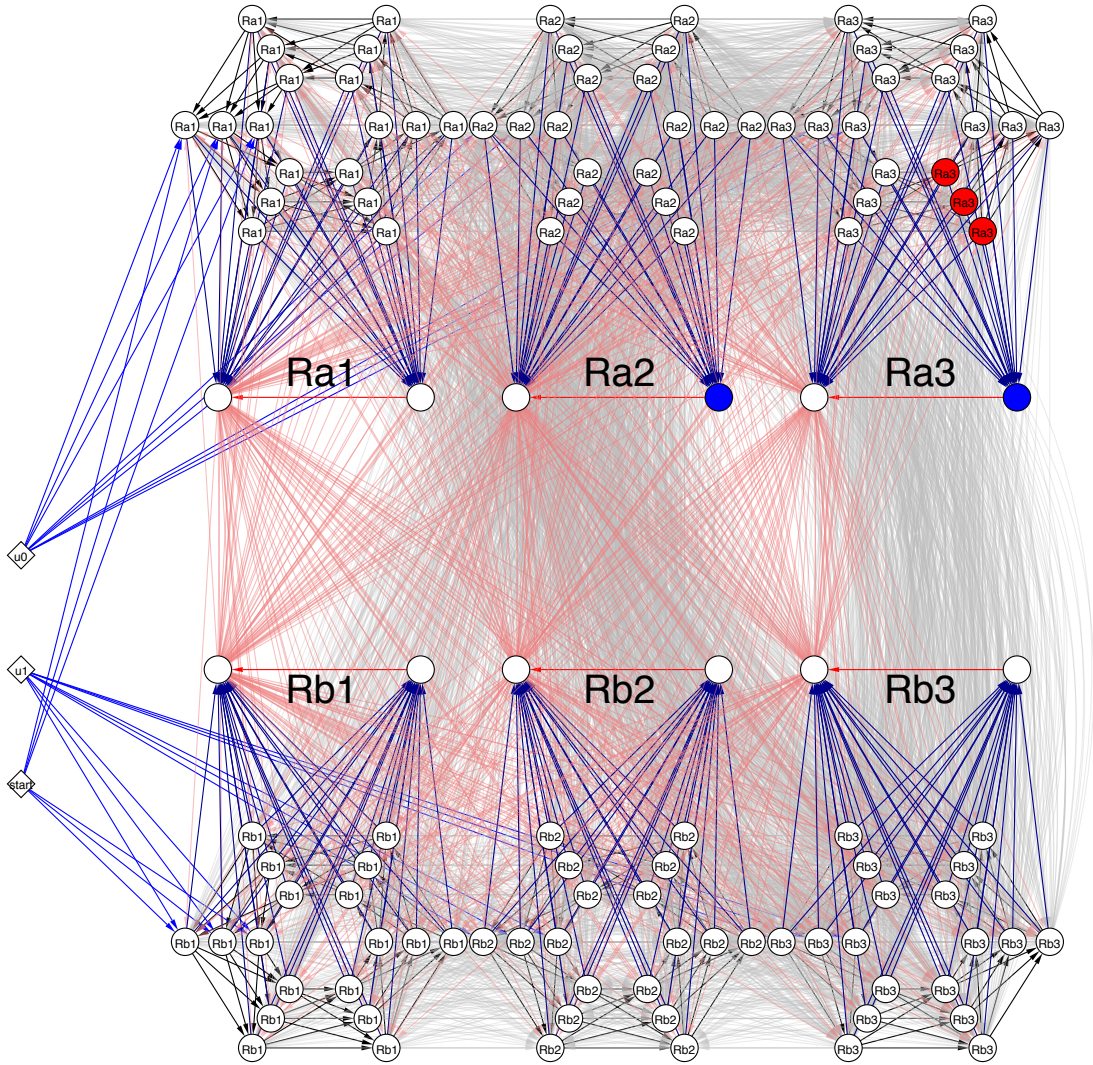
Simulation



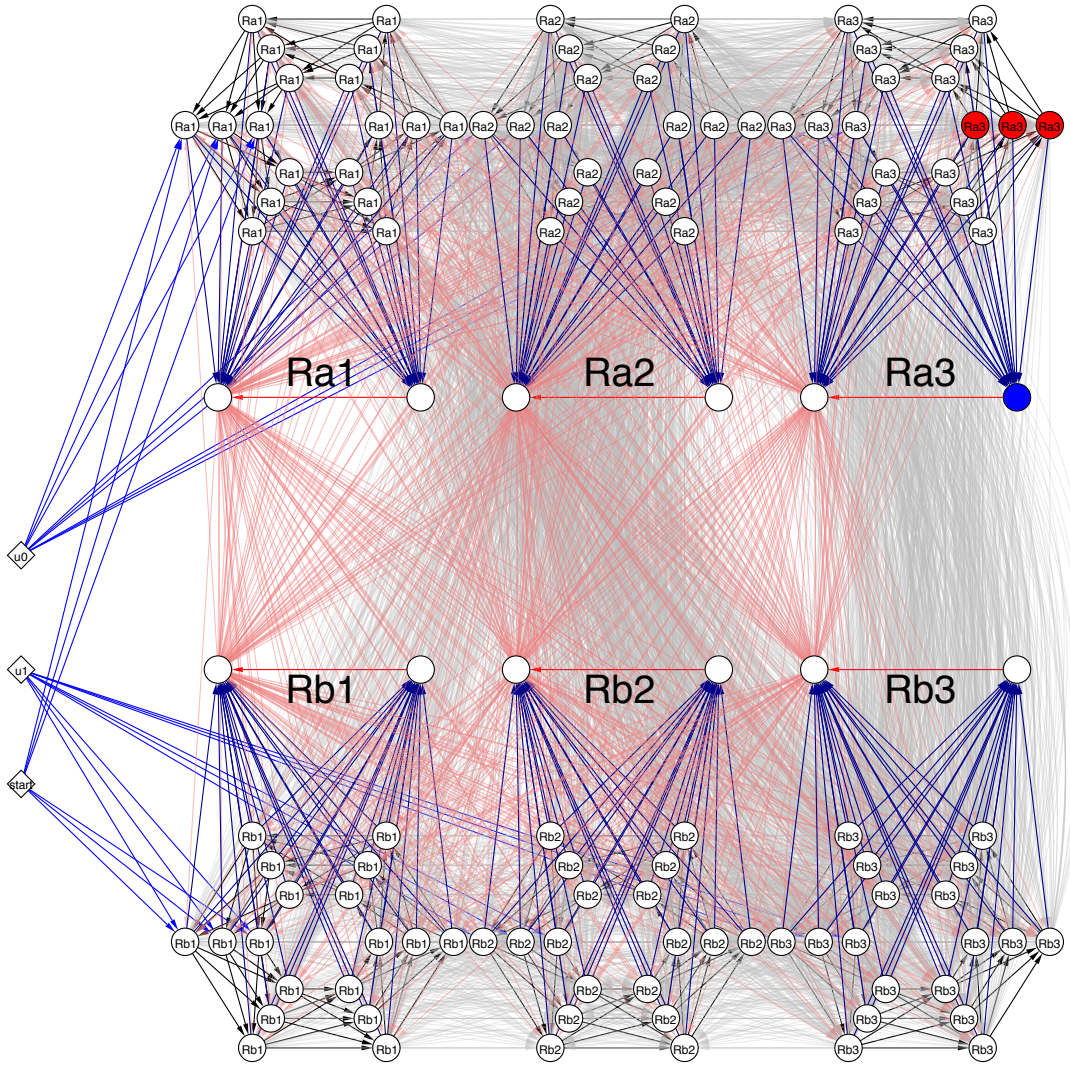
Simulation



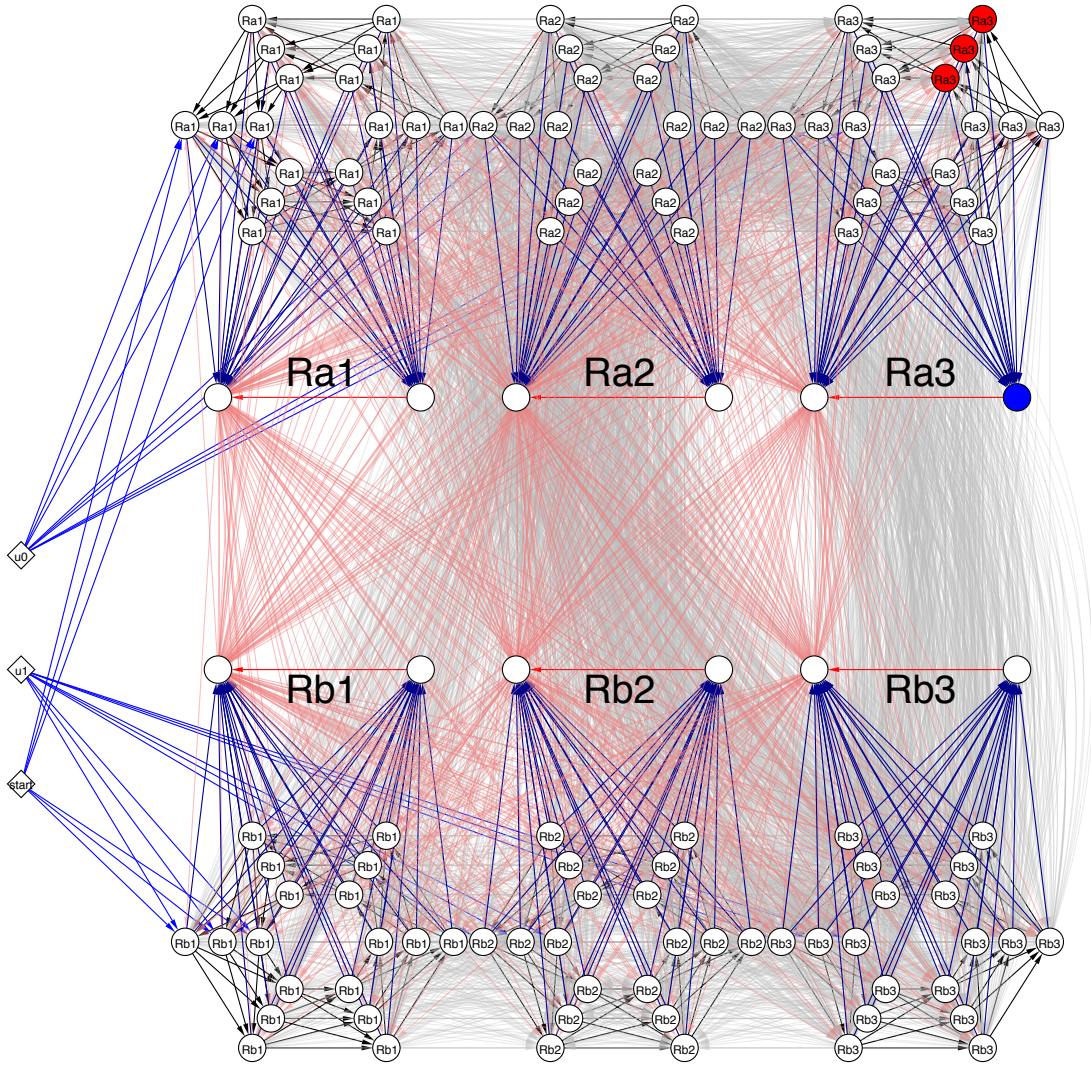
Simulation



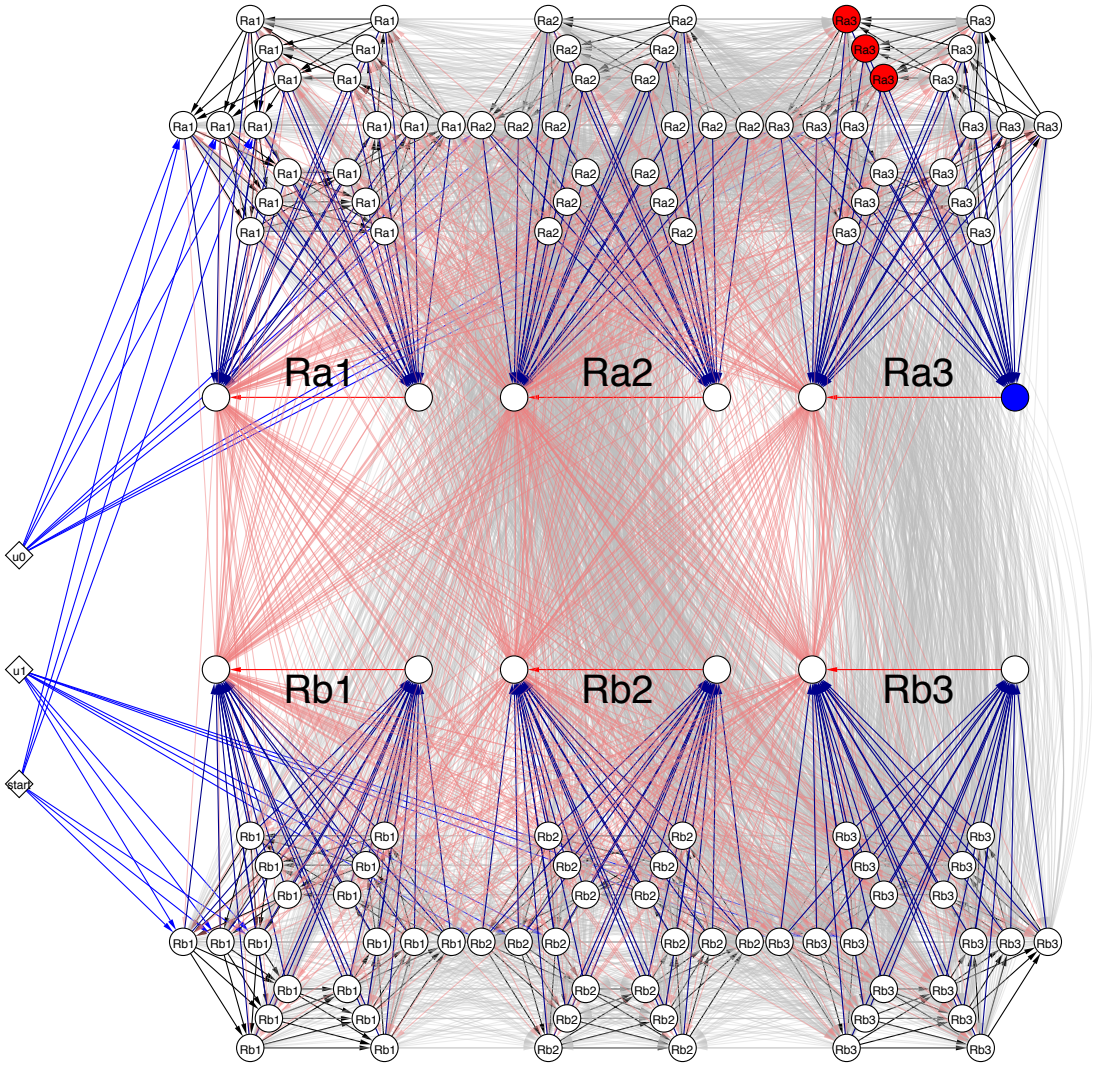
Simulation



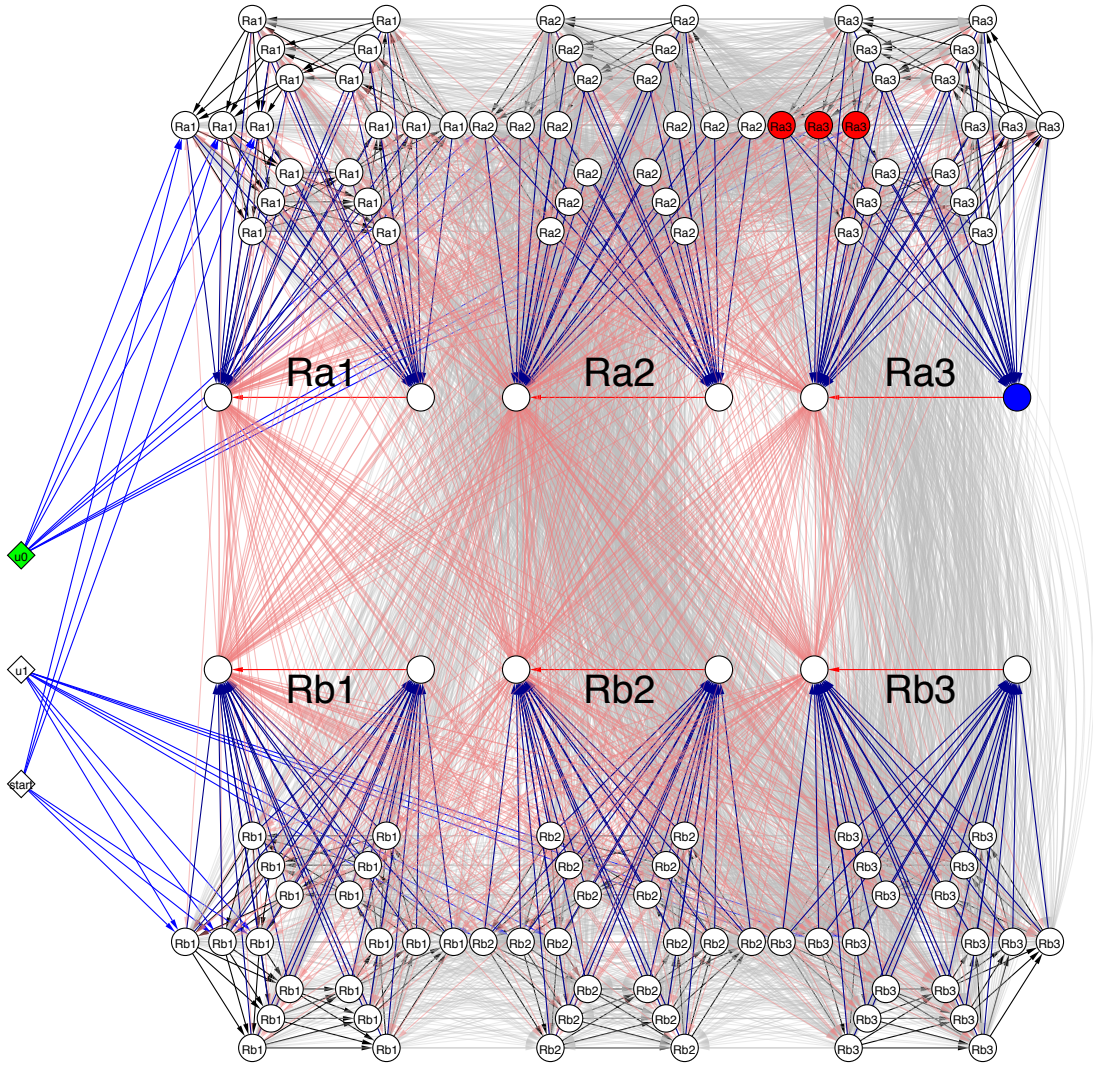
Simulation



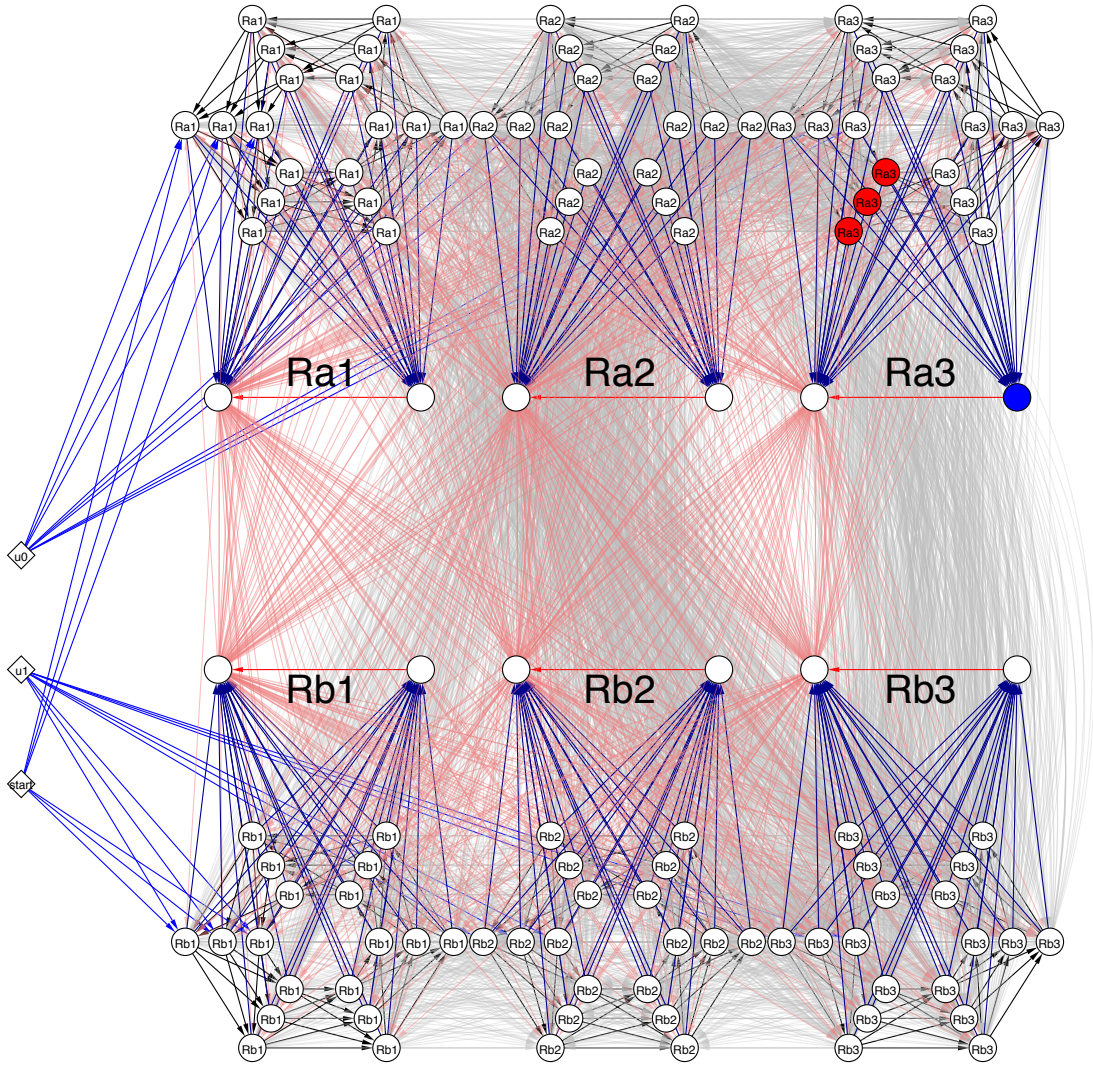
Simulation



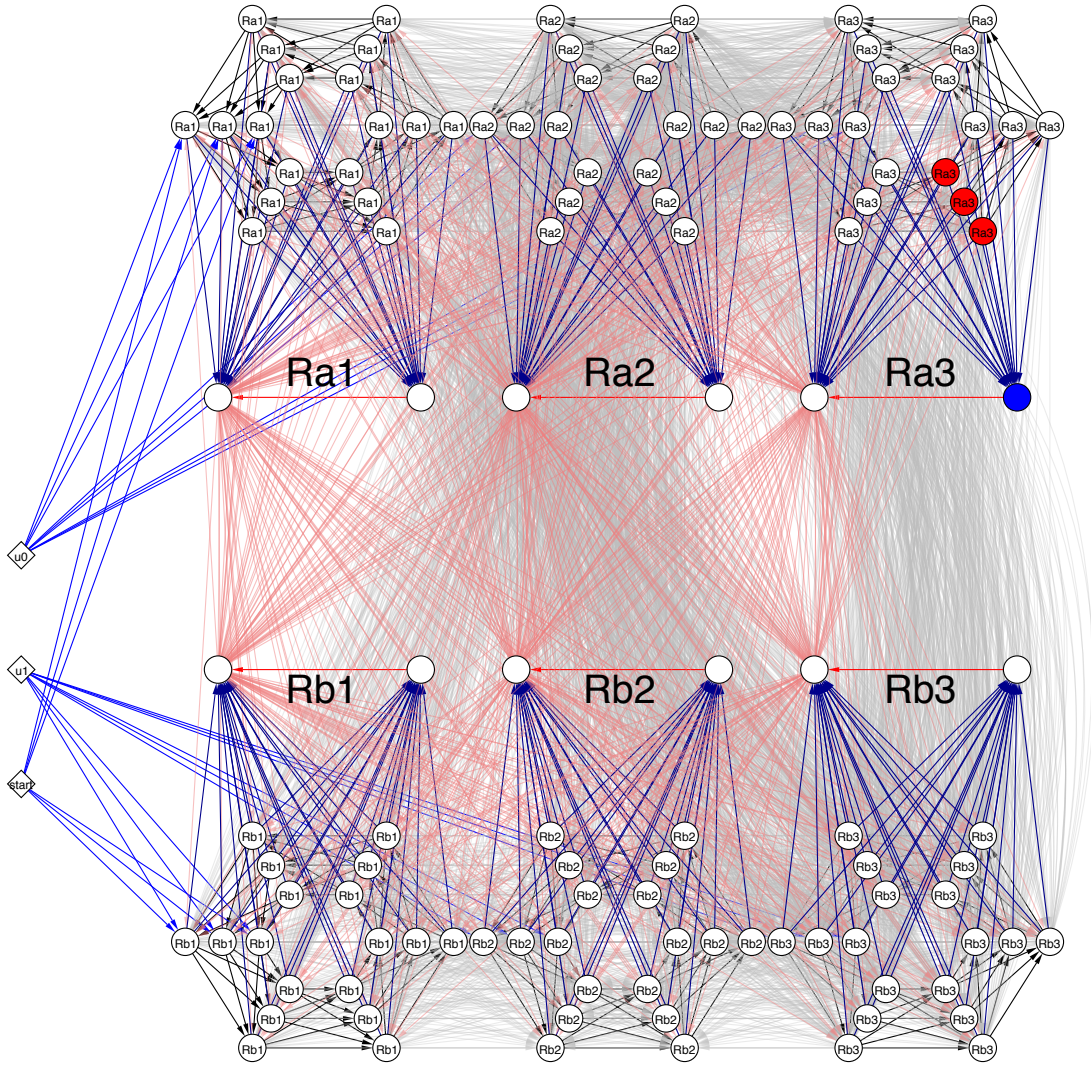
Simulation



Simulation



Simulation



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:

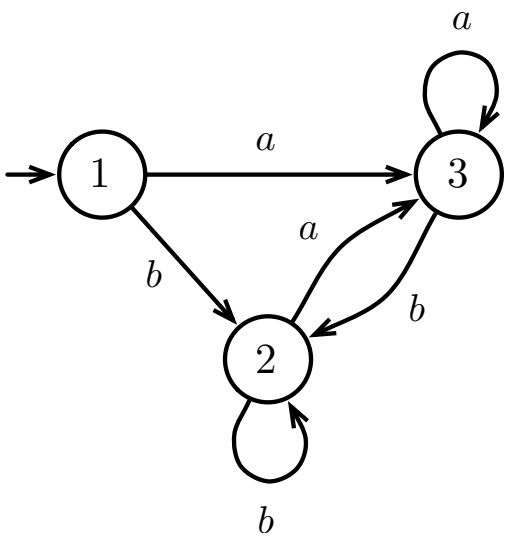
$$\begin{cases} v' &= 0.04v^2 + 5v + 140 - u + I \\ u' &= a(bv - u) \end{cases}$$

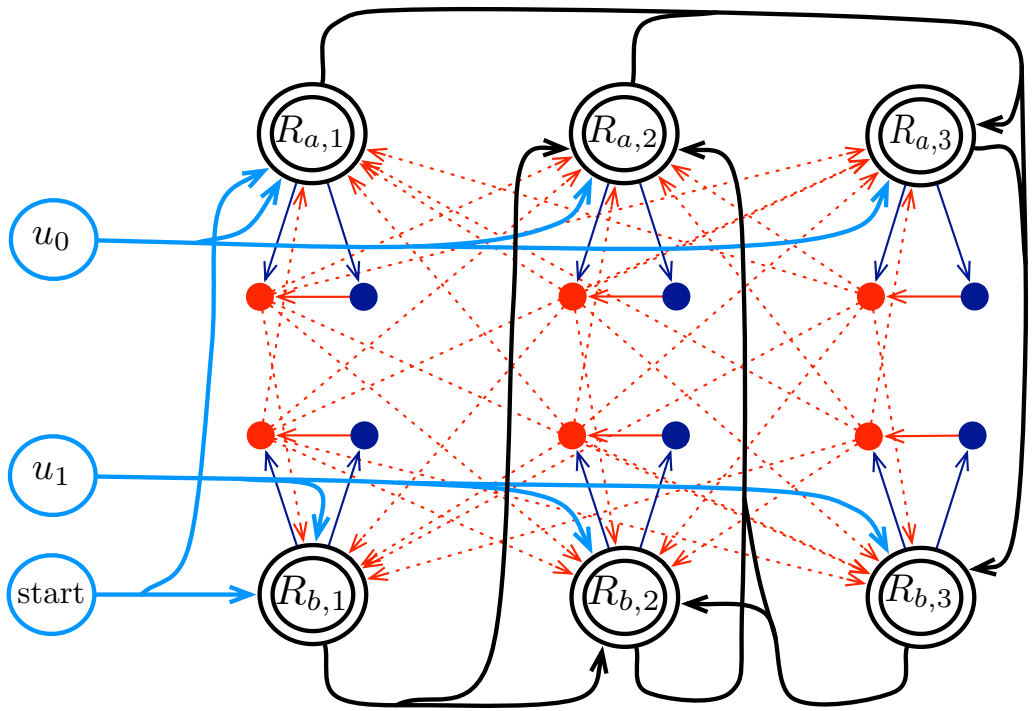
with the auxiliary after-spike reseting:

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

- 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

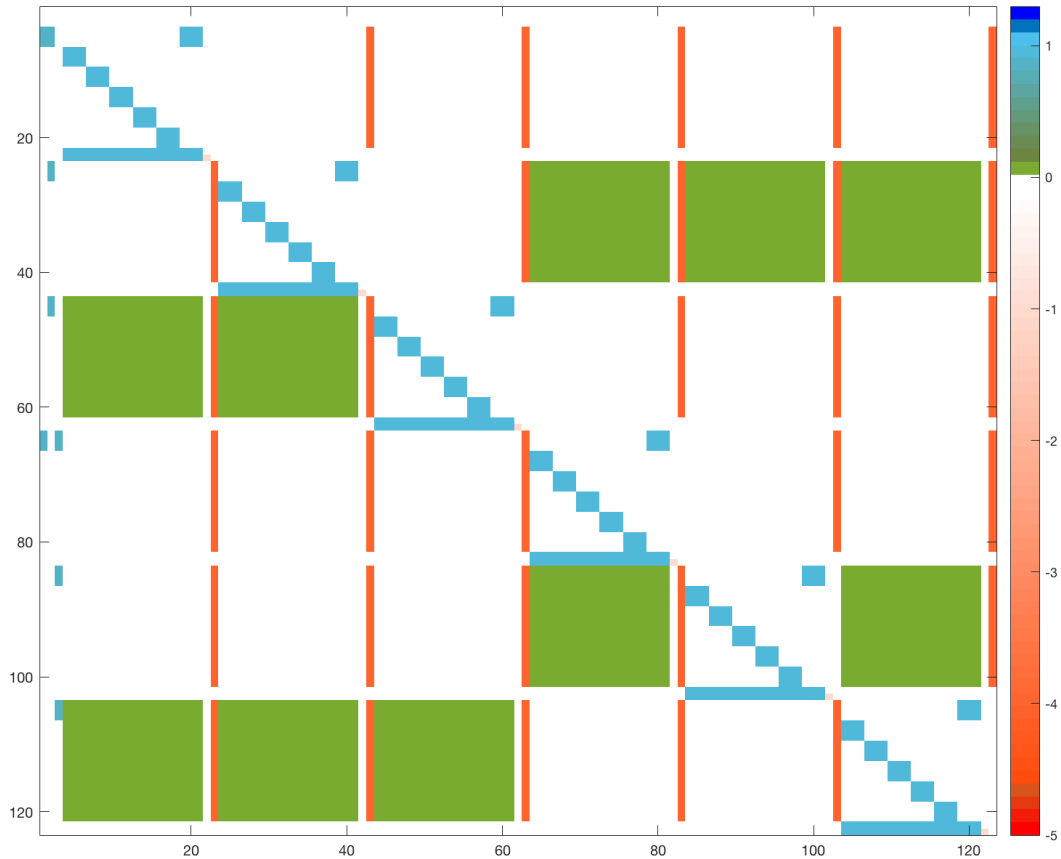


Figure: Connectivity matrix of the network

Simulation: without synaptic noise

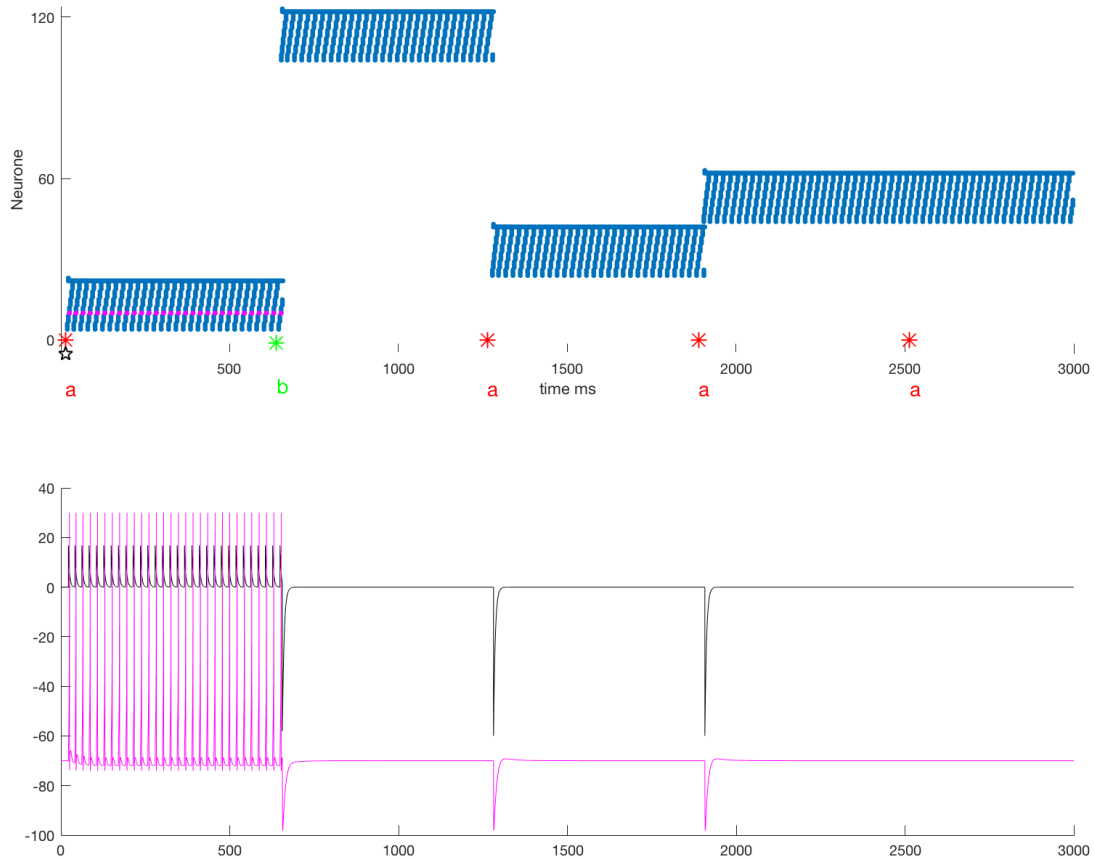


Figure: Raster plot of the simulation and activity of one spiking neuron

Simulation: with synaptic noise

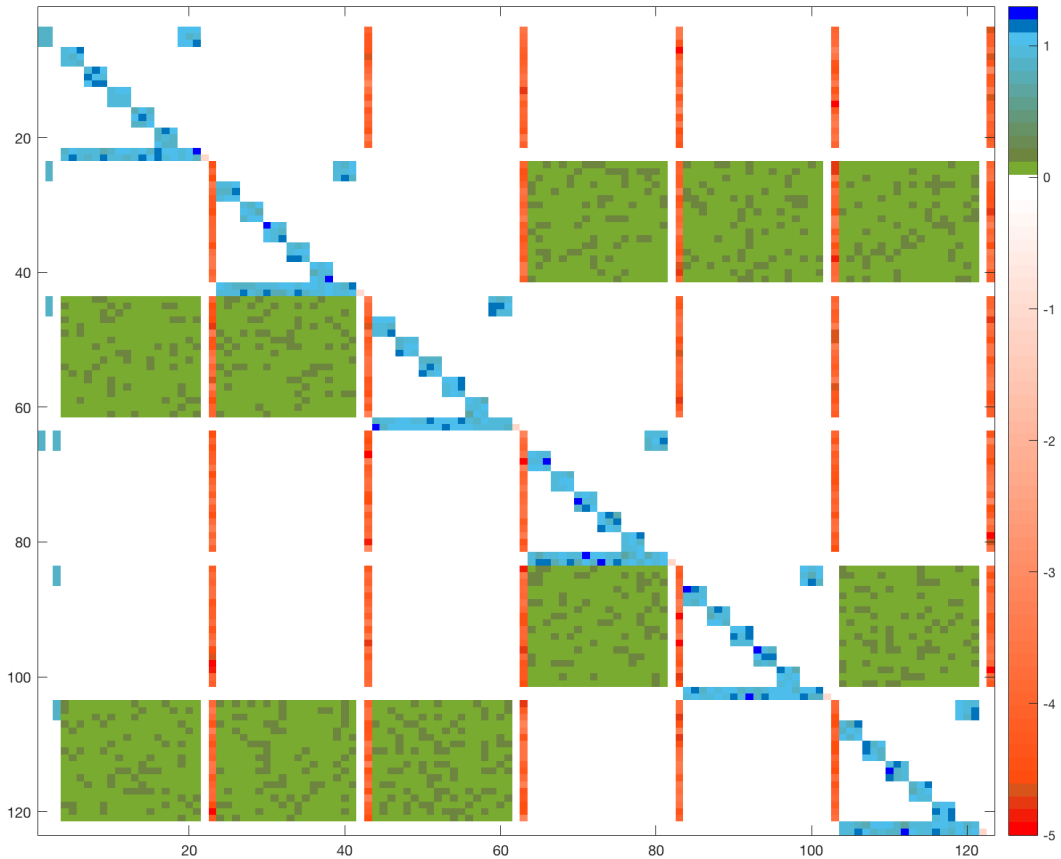
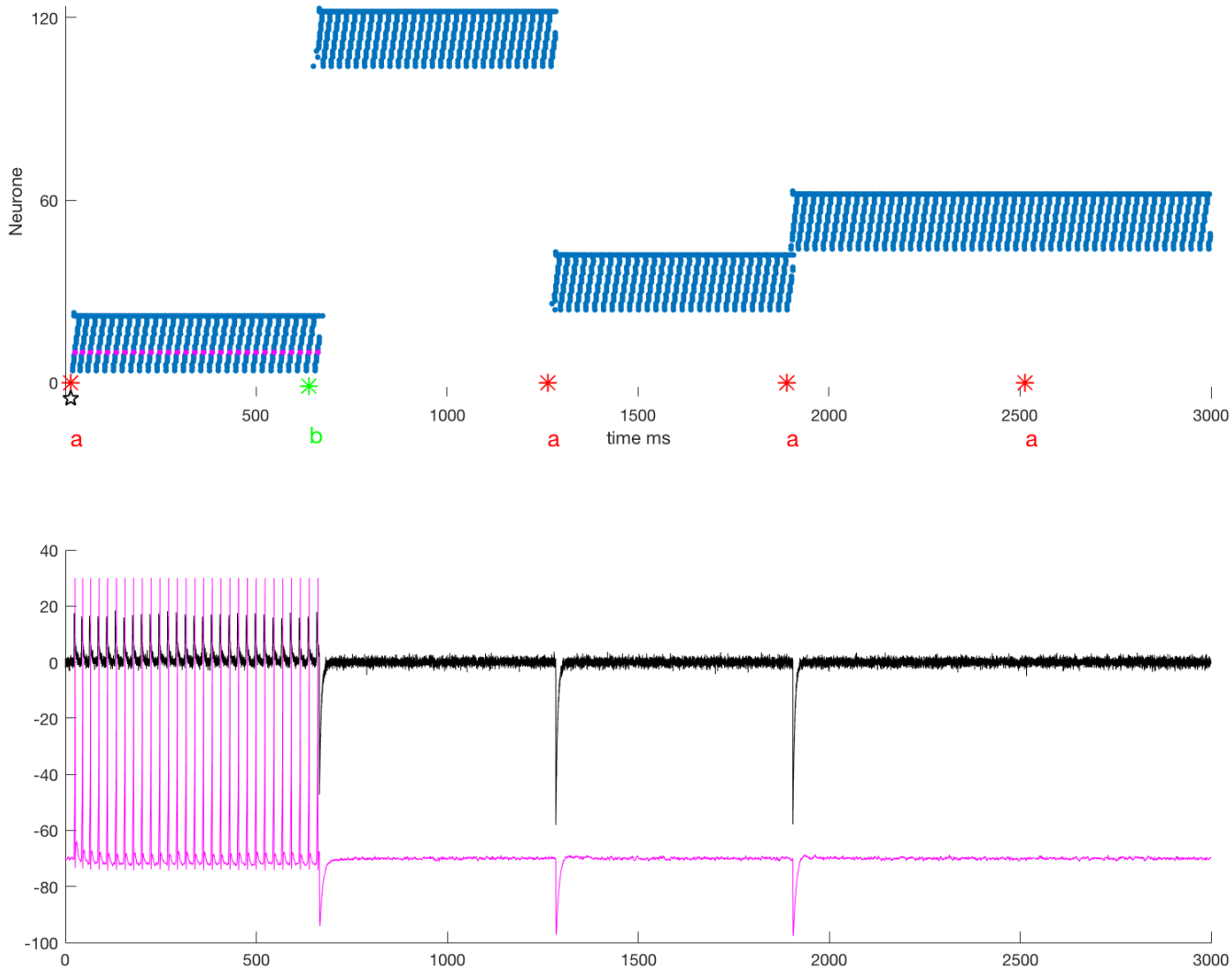


Figure: Connectivity matrix of the network

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.