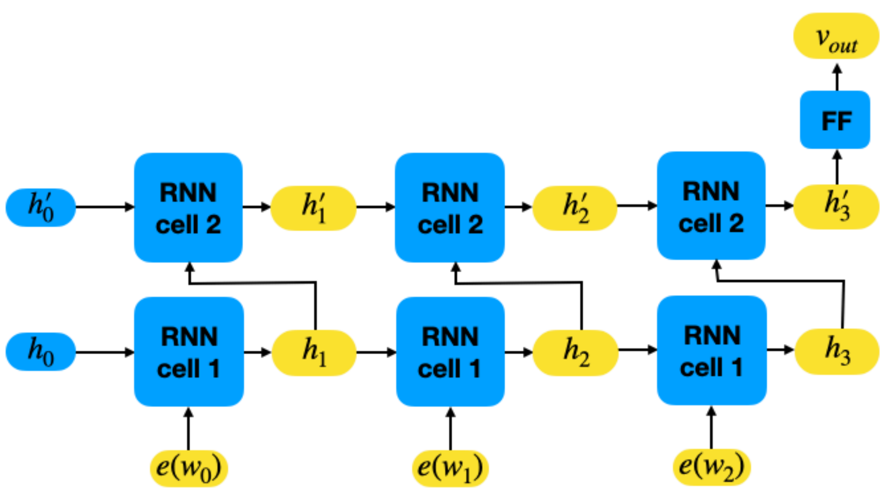
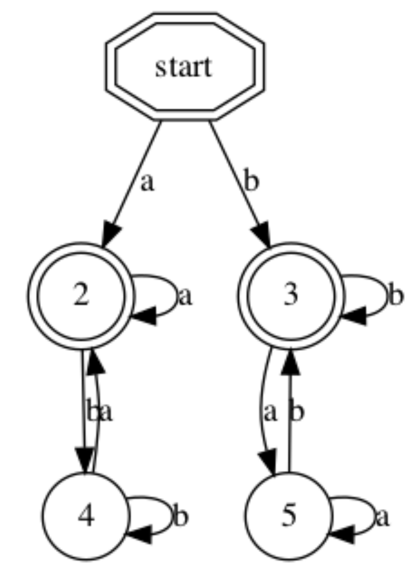
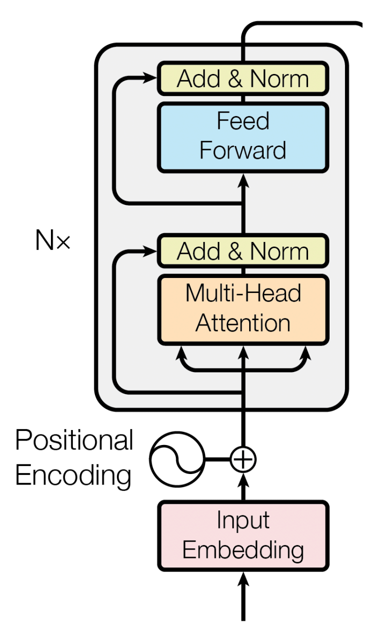
Formal Abstractions of Neural Sequence Models



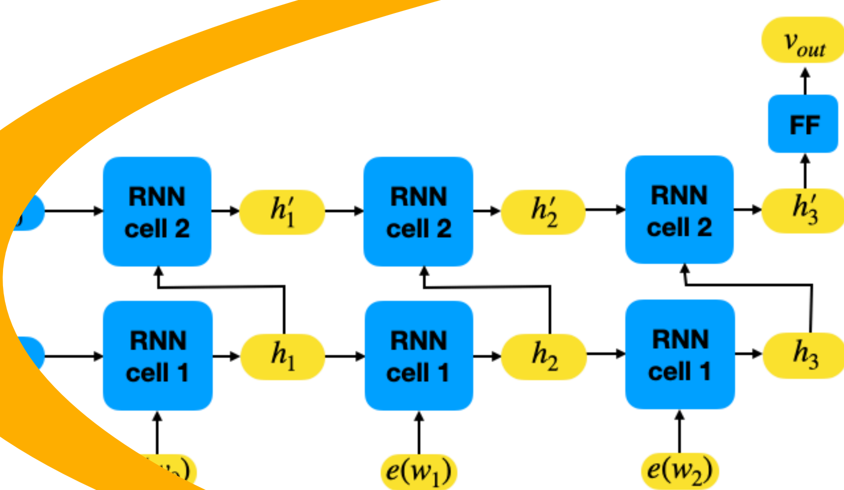


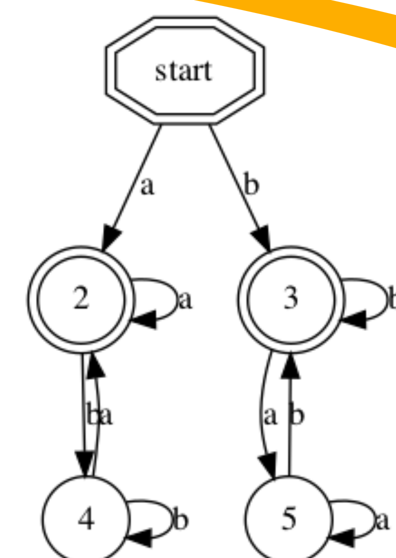


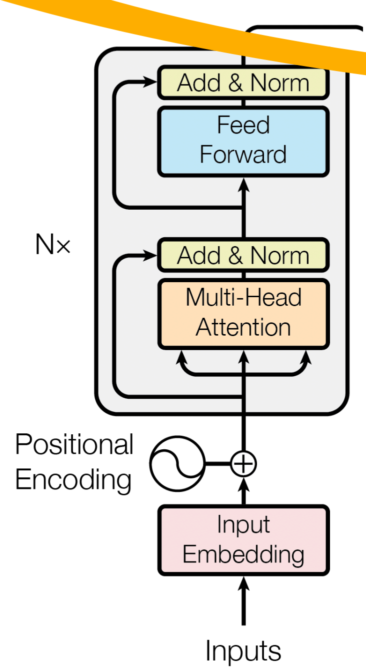
Code!?

Gail Weiss, ICGI 2021

Formal Abstractions of Neural Sequence Models



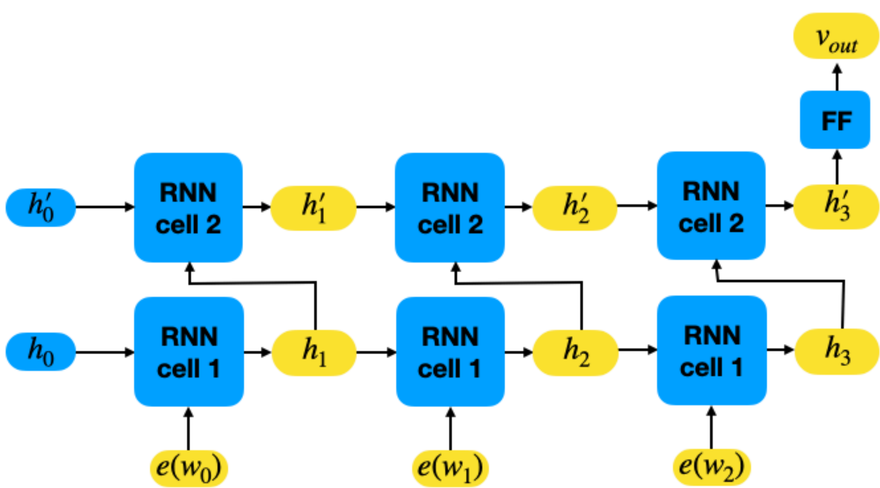


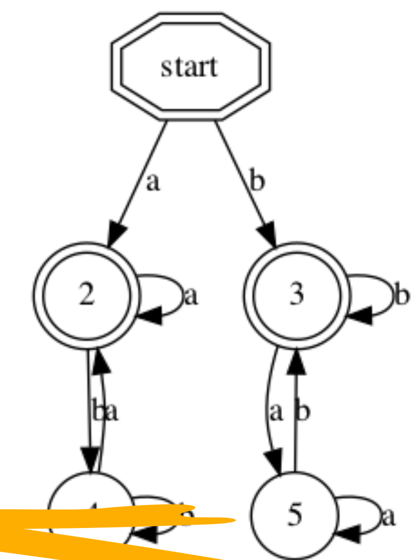


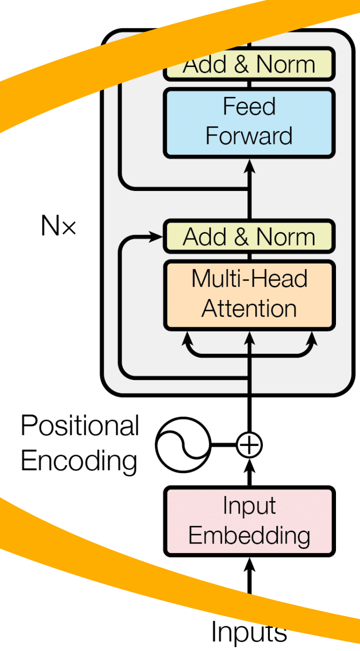
Code!?

Gail Weiss, ICGI 2021

Formal Abstractions of Neural Sequence Models







Code!?

Gail Weiss, ICGI 2021

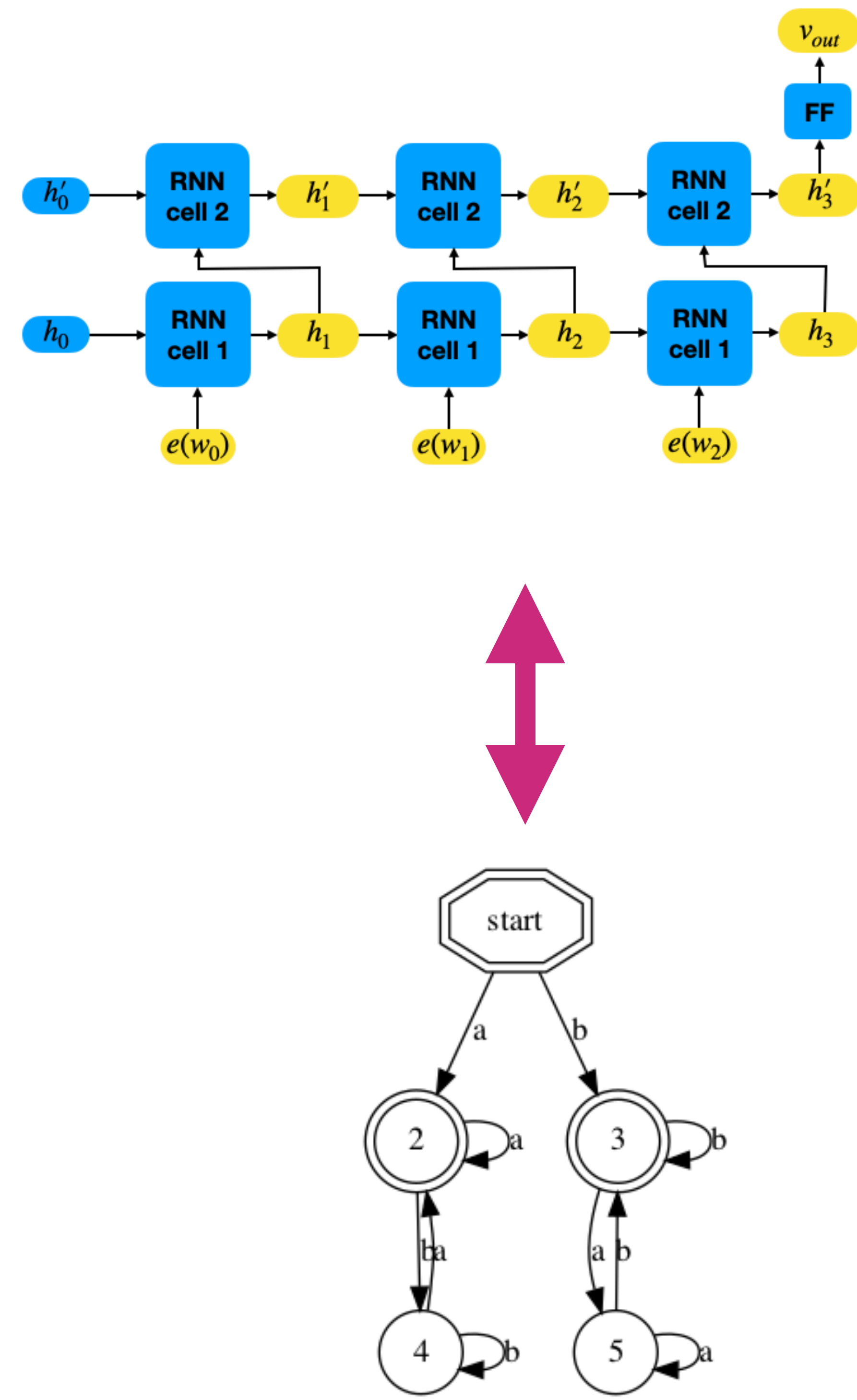
Overview

Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
    - DFAs
    - WFAs
    - More
- Analysis

Transformers

- Introduction
- A formal abstraction



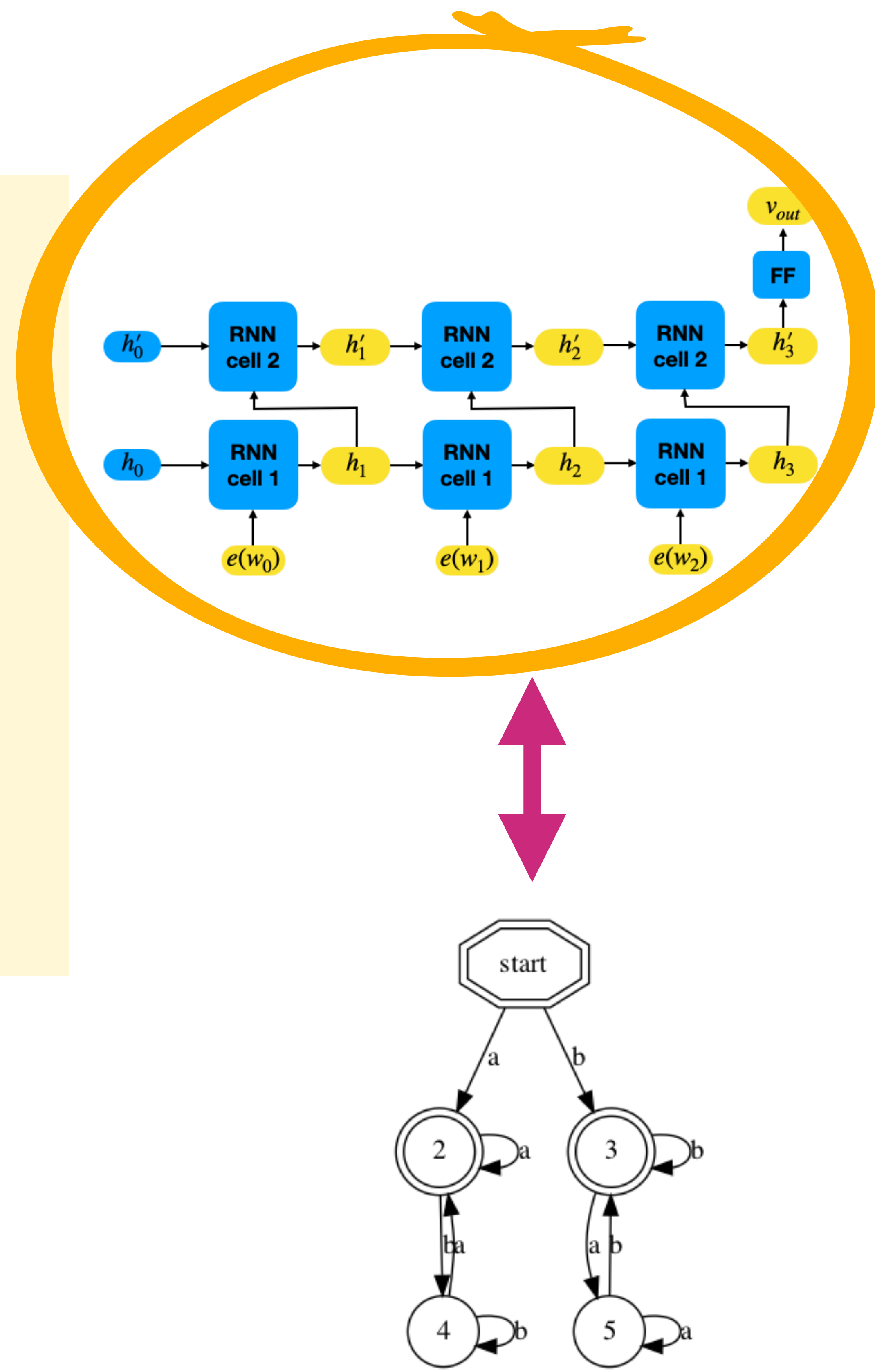
Overview

Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
    - DFAs
    - WFAs
    - More
- Analysis

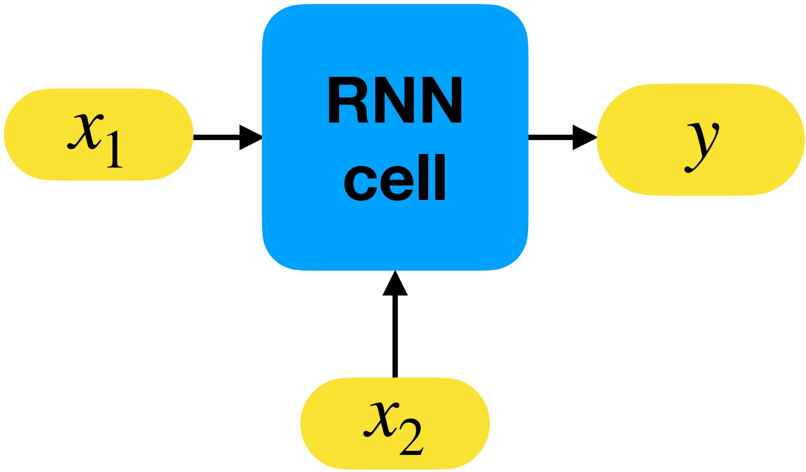
Transformers

- Introduction
- A formal abstraction



RNNs: Introduction

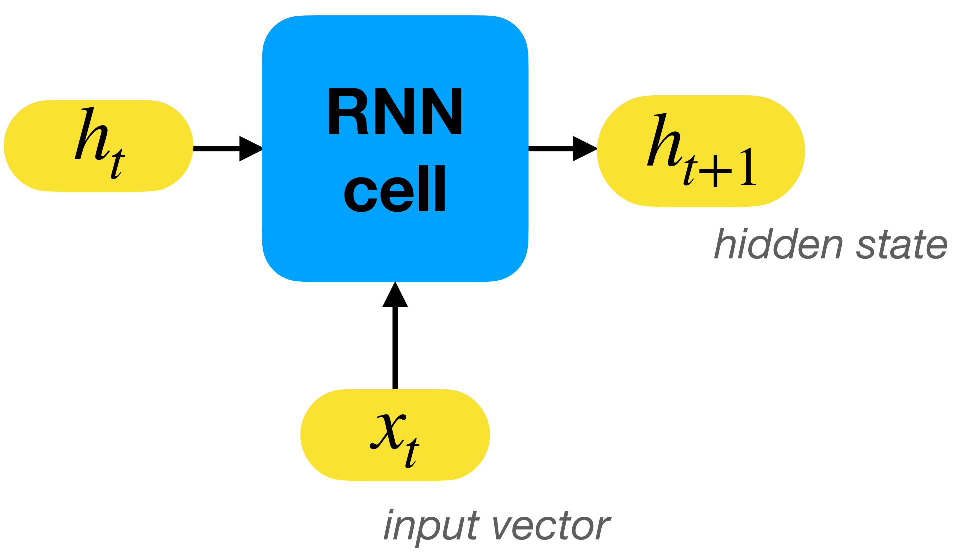
Finding Structure in Time
- Elman 1990



$x_1, y \in \mathbb{R}^{d_h} \qquad x_2 \in \mathbb{R}^{d_i}$

RNNs: Introduction

Finding Structure in Time
- Elman 1990



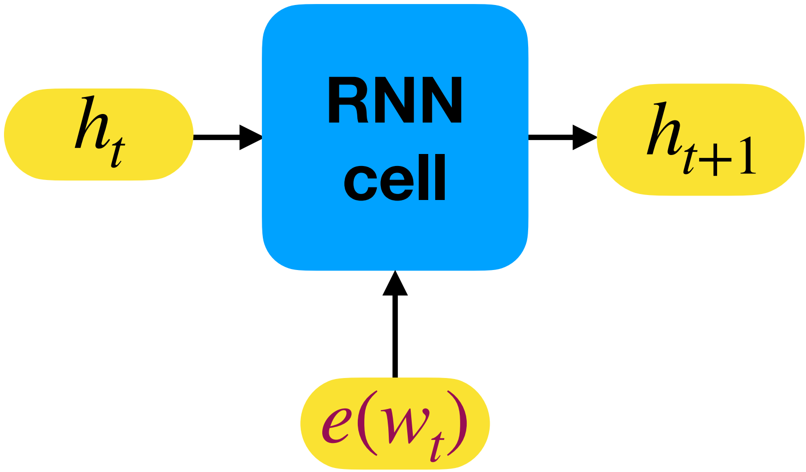
$$\forall t: \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$$

RNNs: Introduction

Finding Structure in Time
- Elman 1990

$$e : \Sigma \rightarrow \mathbb{R}^{d_i}$$

input embedding



$$x_t = e(w_t)$$

$$\forall t : \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$$

RNNs: Introduction

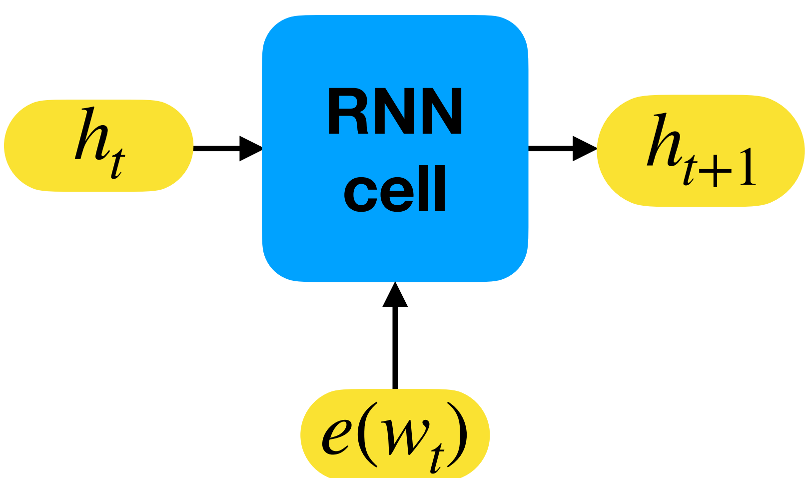
Finding Structure in Time
- Elman 1990

h_0

initial hidden state

$$e : \Sigma \rightarrow \mathbb{R}^{d_i}$$

input embedding



$$x_t = e(w_t)$$

$$\forall t : \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$$

RNNs: Introduction

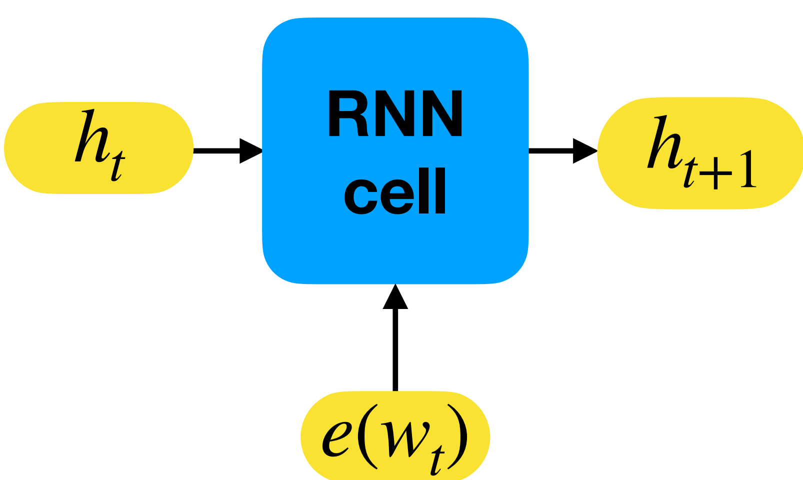
Finding Structure in Time
- Elman 1990

h_0

initial hidden state

$$e : \Sigma \rightarrow \mathbb{R}^{d_i}$$

input embedding



$$w = w_0 w_1 w_2 \in \Sigma^*$$

$$x_t = e(w_t)$$

$$\forall t : \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$$

RNNs: Introduction

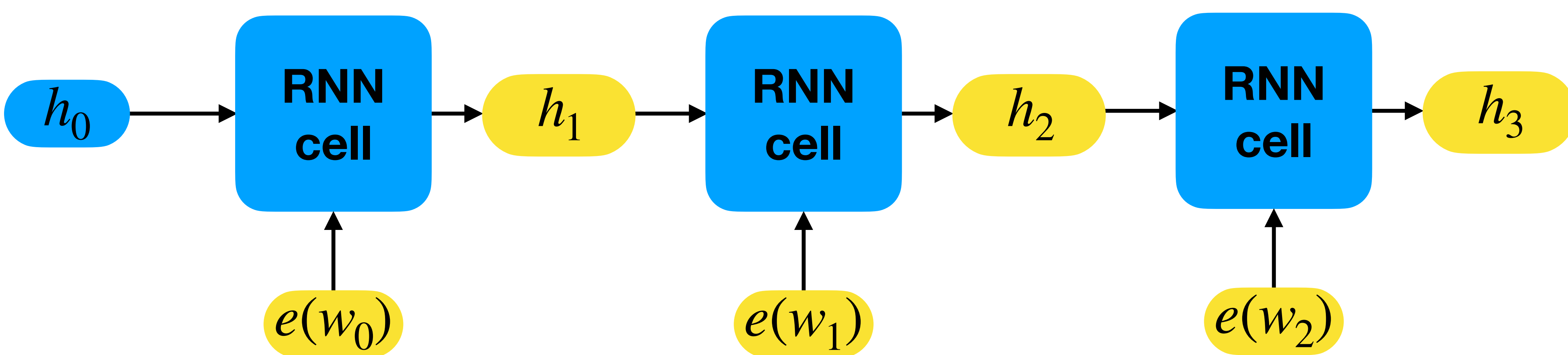
Finding Structure in Time
- Elman 1990

h_0

initial hidden state

$$e : \Sigma \to \mathbb{R}^{d_i}$$

input embedding



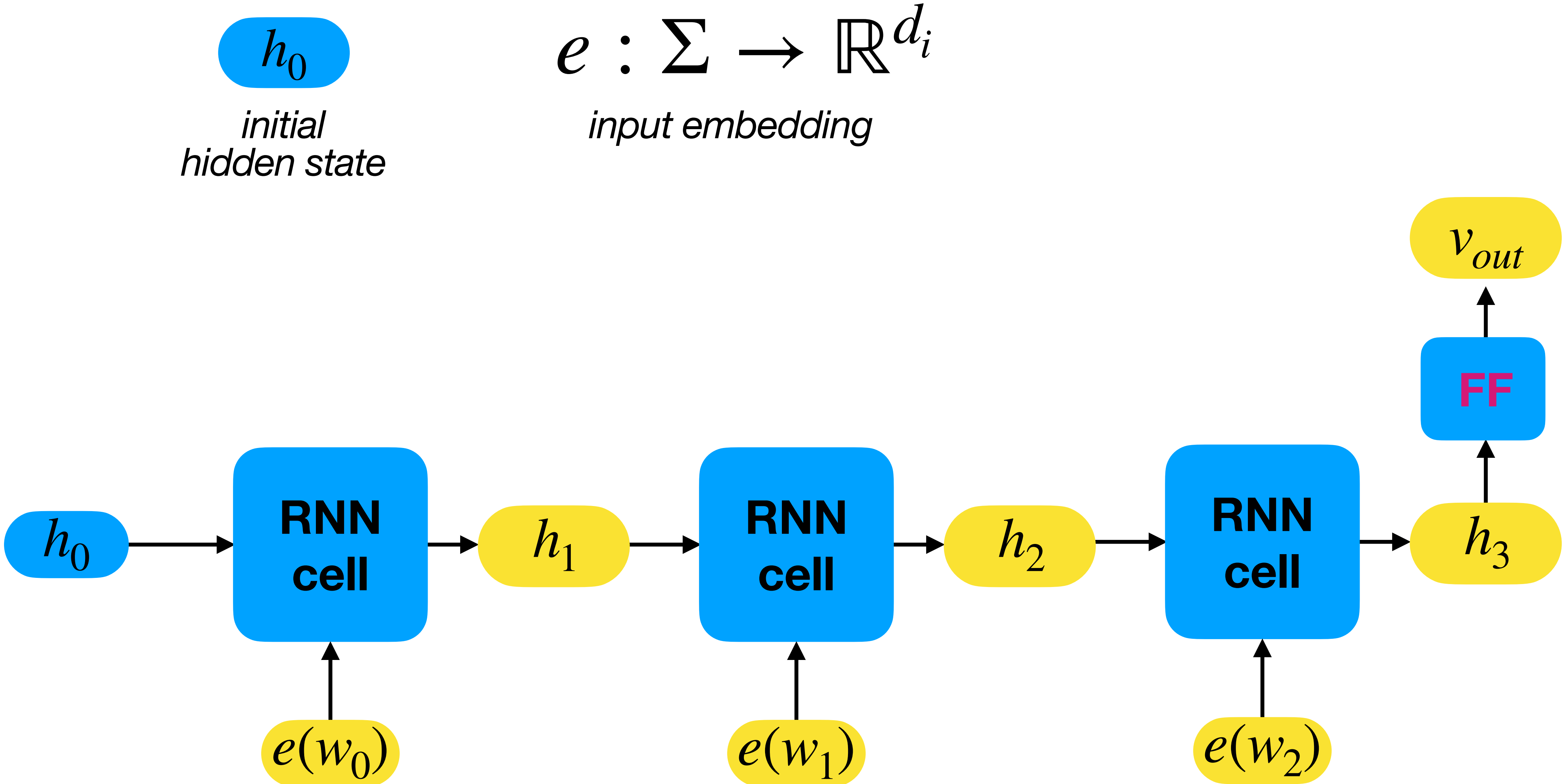
$$w = w_0 w_1 w_2 \in \Sigma^*$$

$$x_t = e(w_t)$$

$$\forall t : \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$$

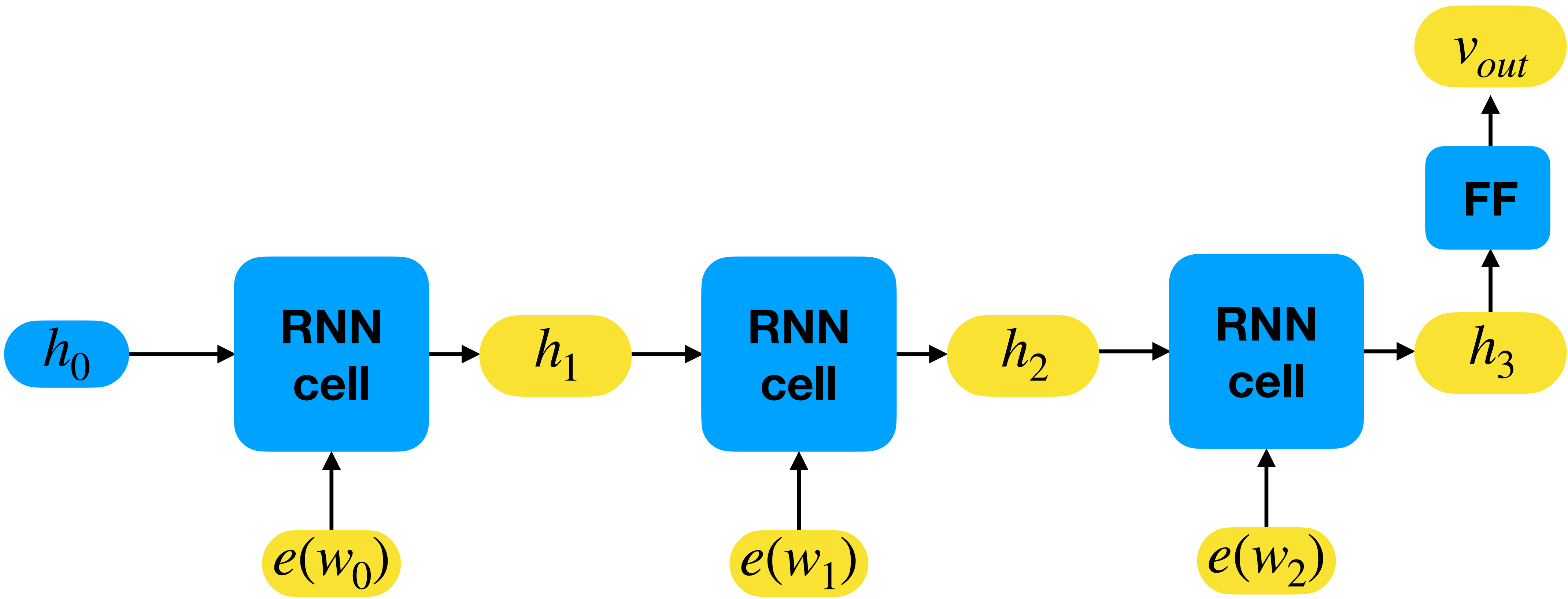
RNNs: Introduction

Finding Structure in Time
- Elman 1990



$w = w_0 w_1 w_2 \in \Sigma^*$ $\quad x_t = e(w_t)$ $\quad \forall t : \quad h_t \in \mathbb{R}^{d_h} \quad x_t \in \mathbb{R}^{d_i}$

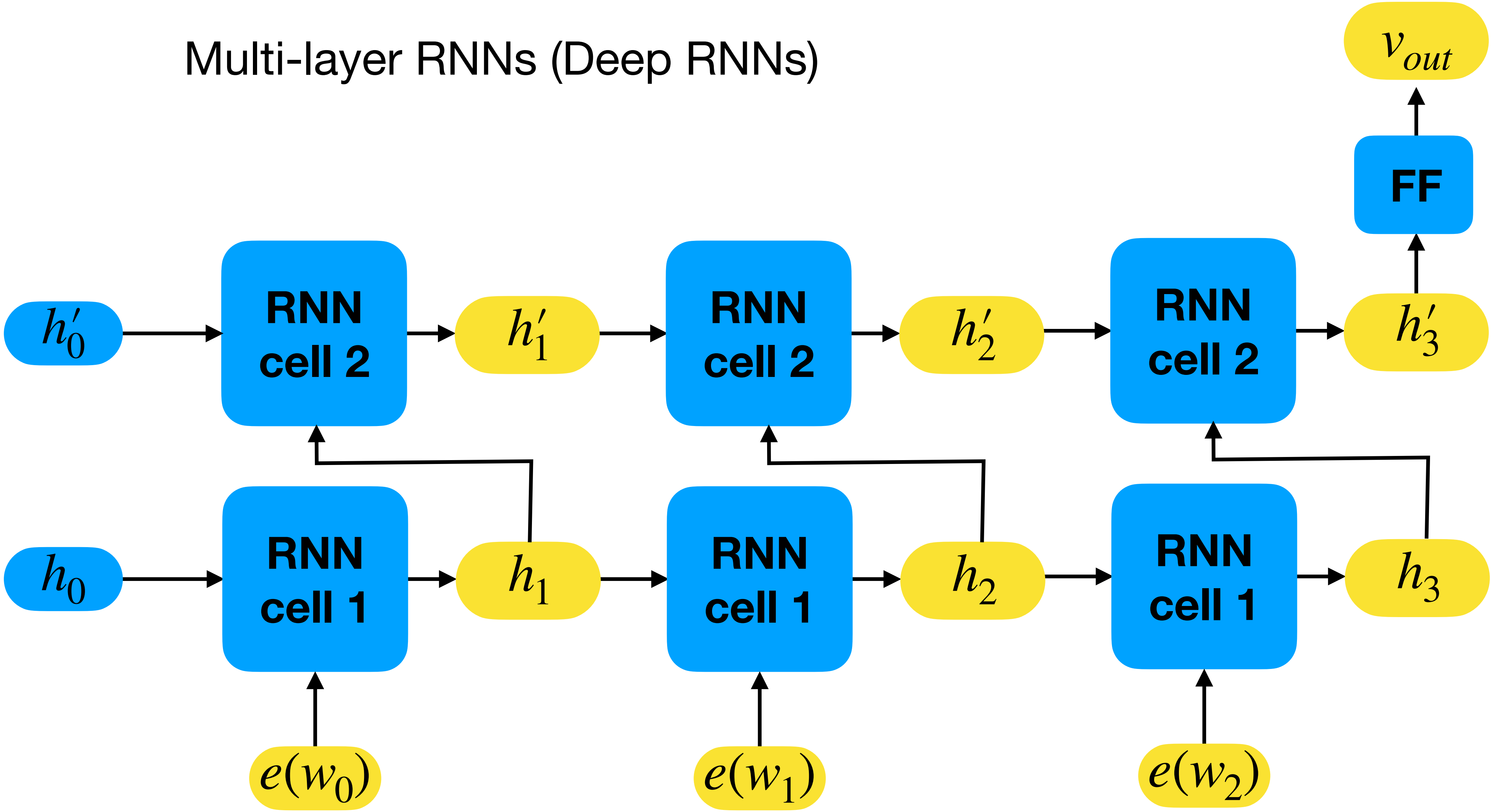
RNNs: Introduction



$$w = w_0 w_1 w_2 \in \Sigma^*$$

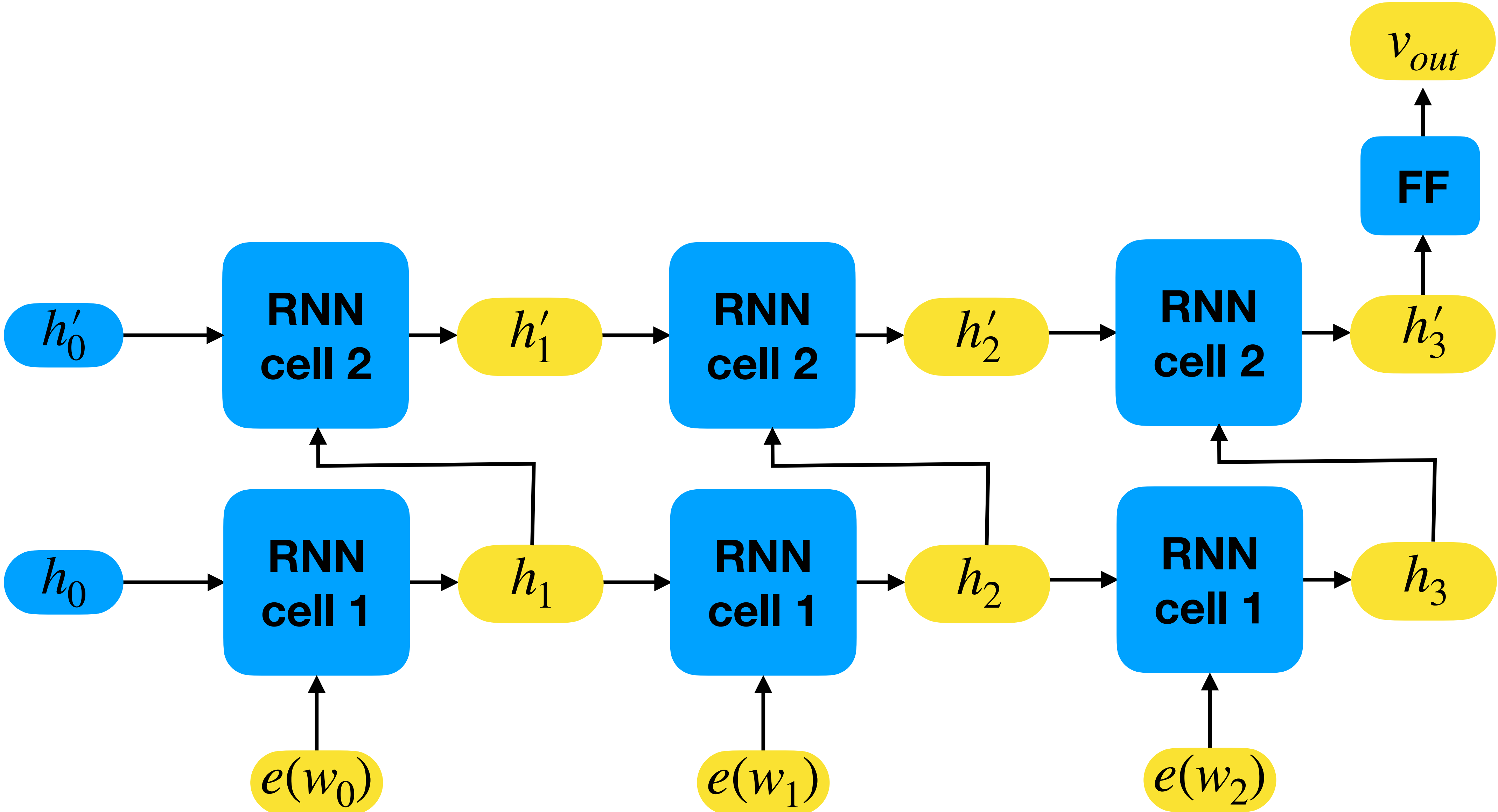
RNNs: Introduction

Multi-layer RNNs (Deep RNNs)



$$w = w_0 w_1 w_2 \in \Sigma^*$$

RNNs: Introduction



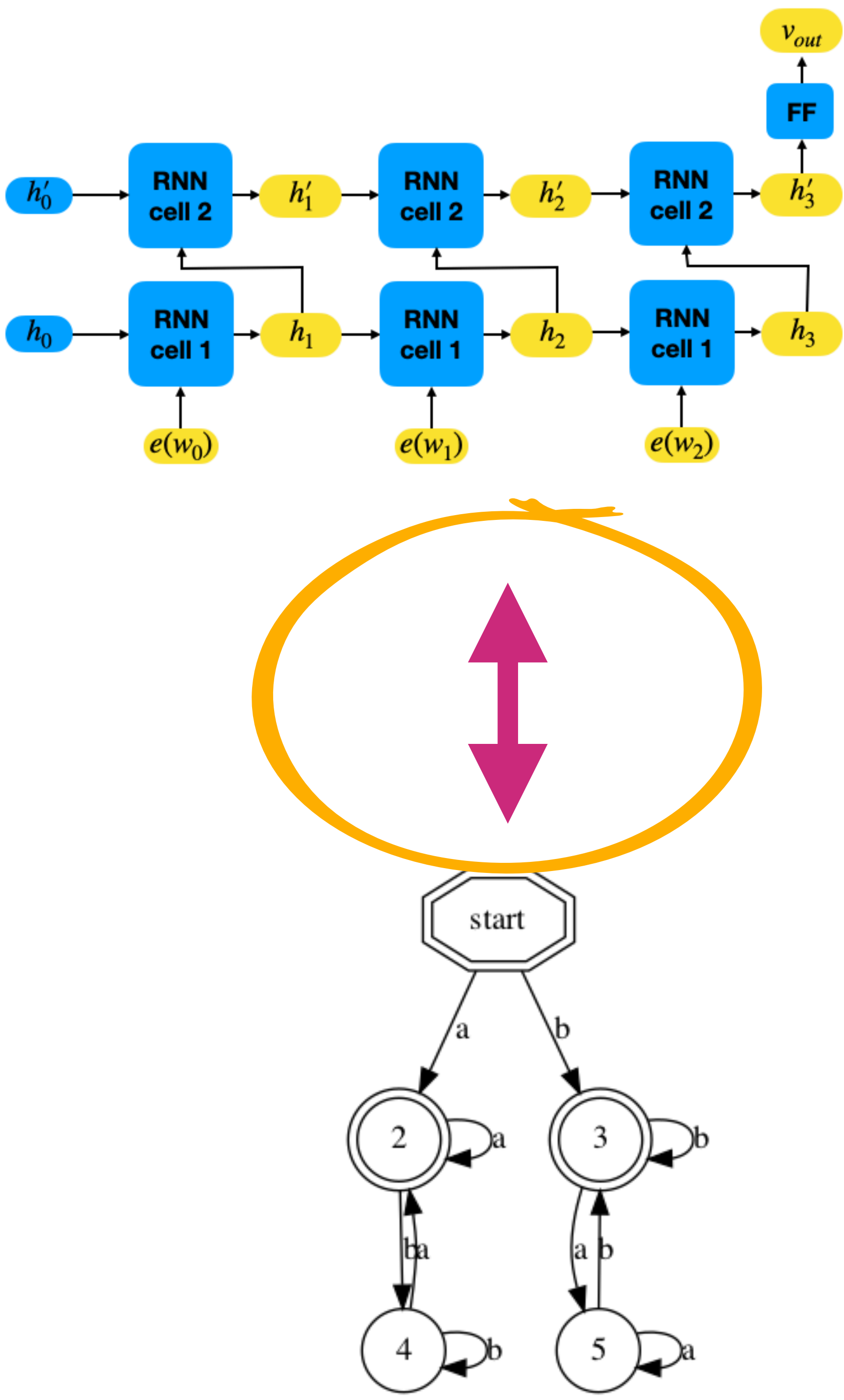
Overview

Recurrent Neural Networks (RNNs)

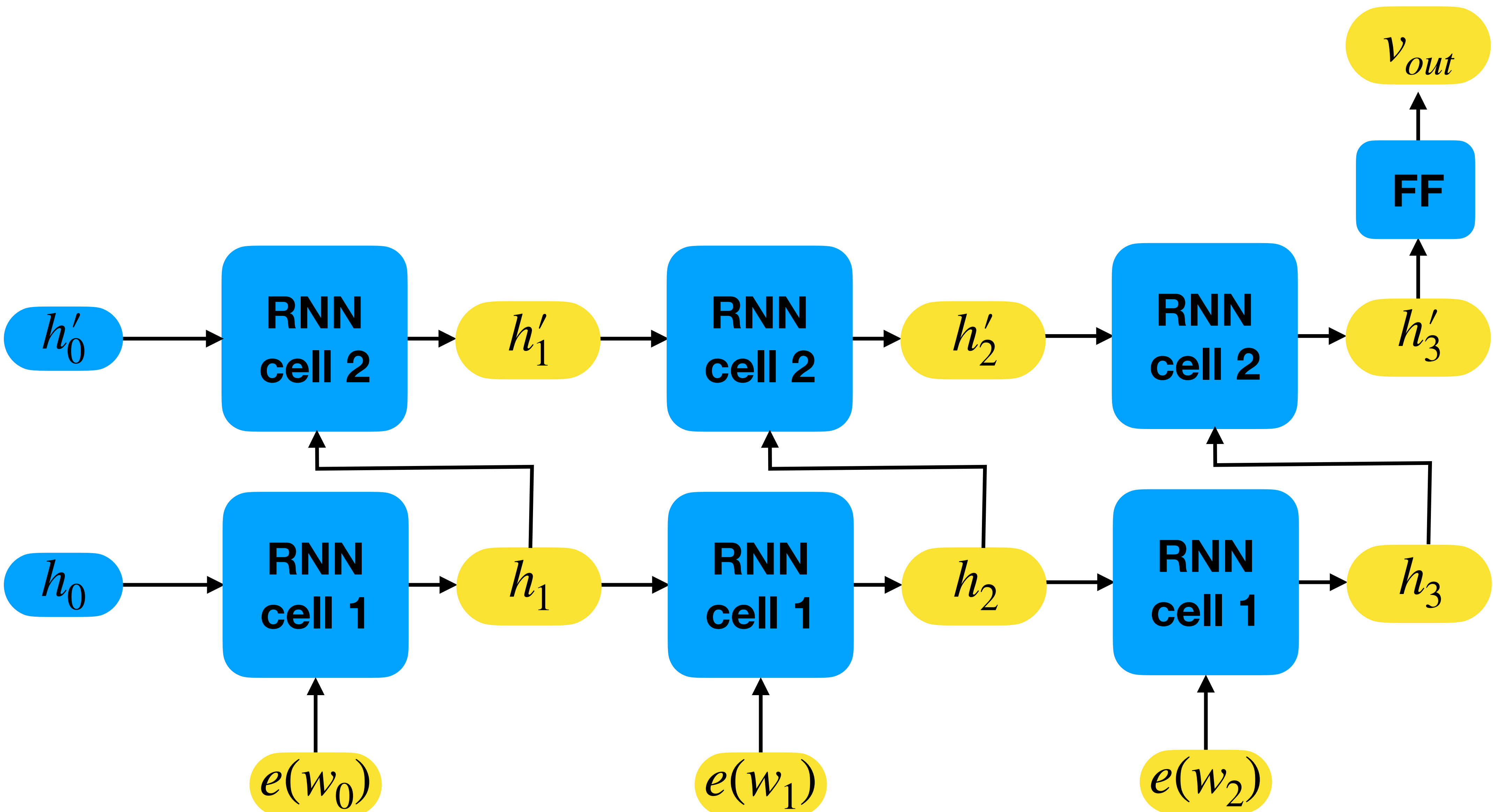
- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers

- Introduction
- A formal abstraction

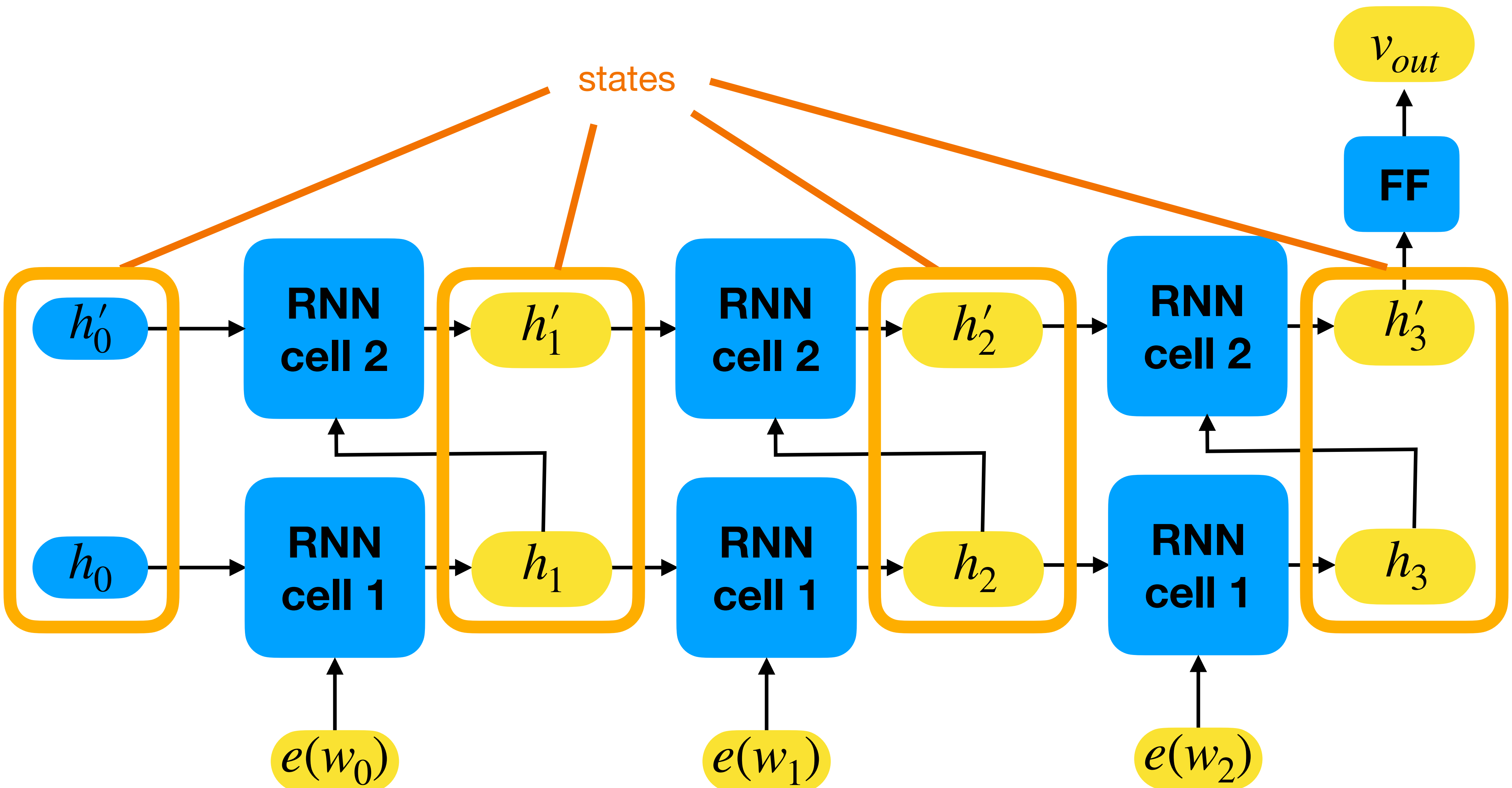


RNNs: Automata Relation



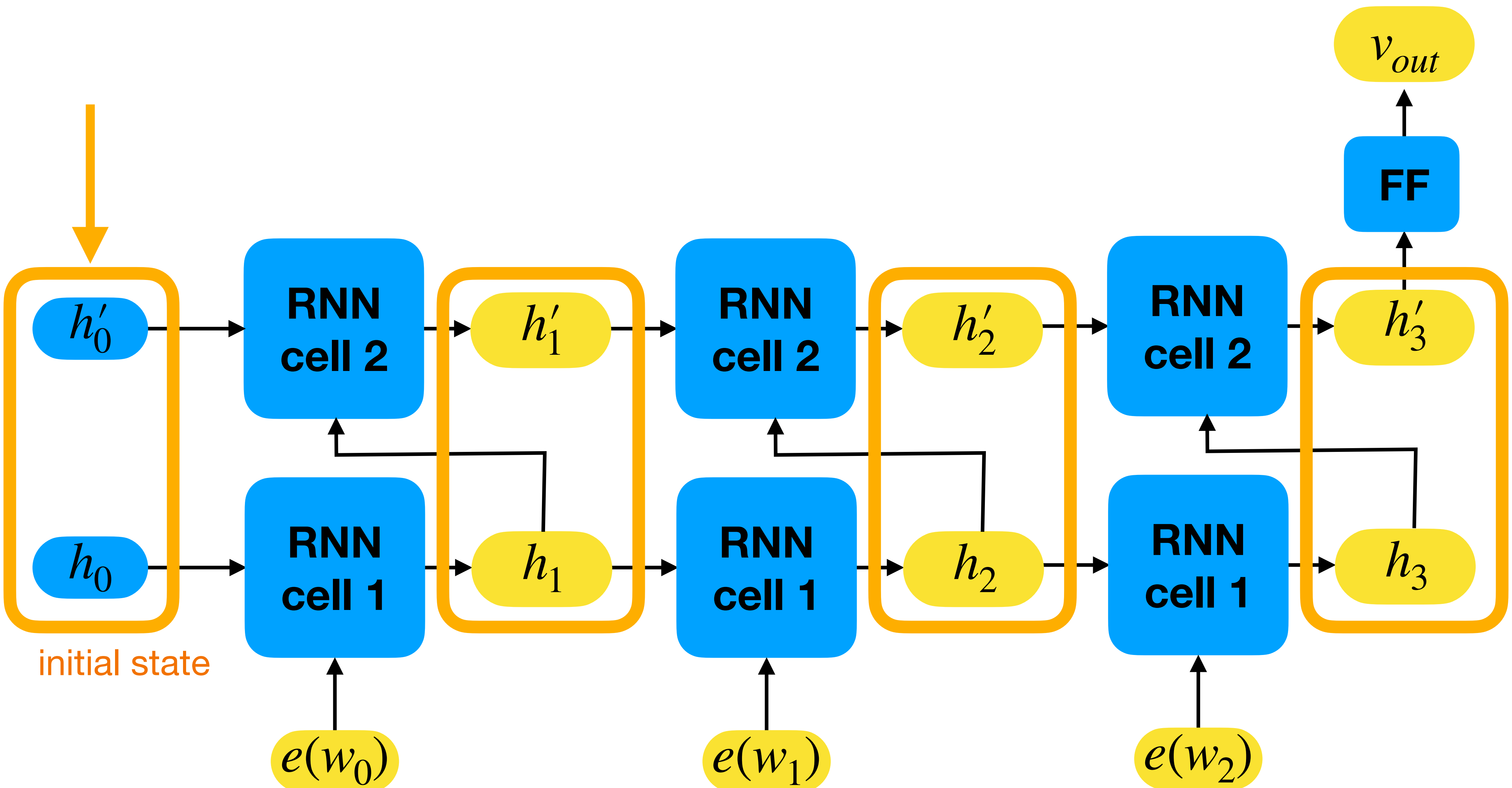
$$w = w_0w_1w_2 \in \Sigma^*$$

RNNs: Automata Relation



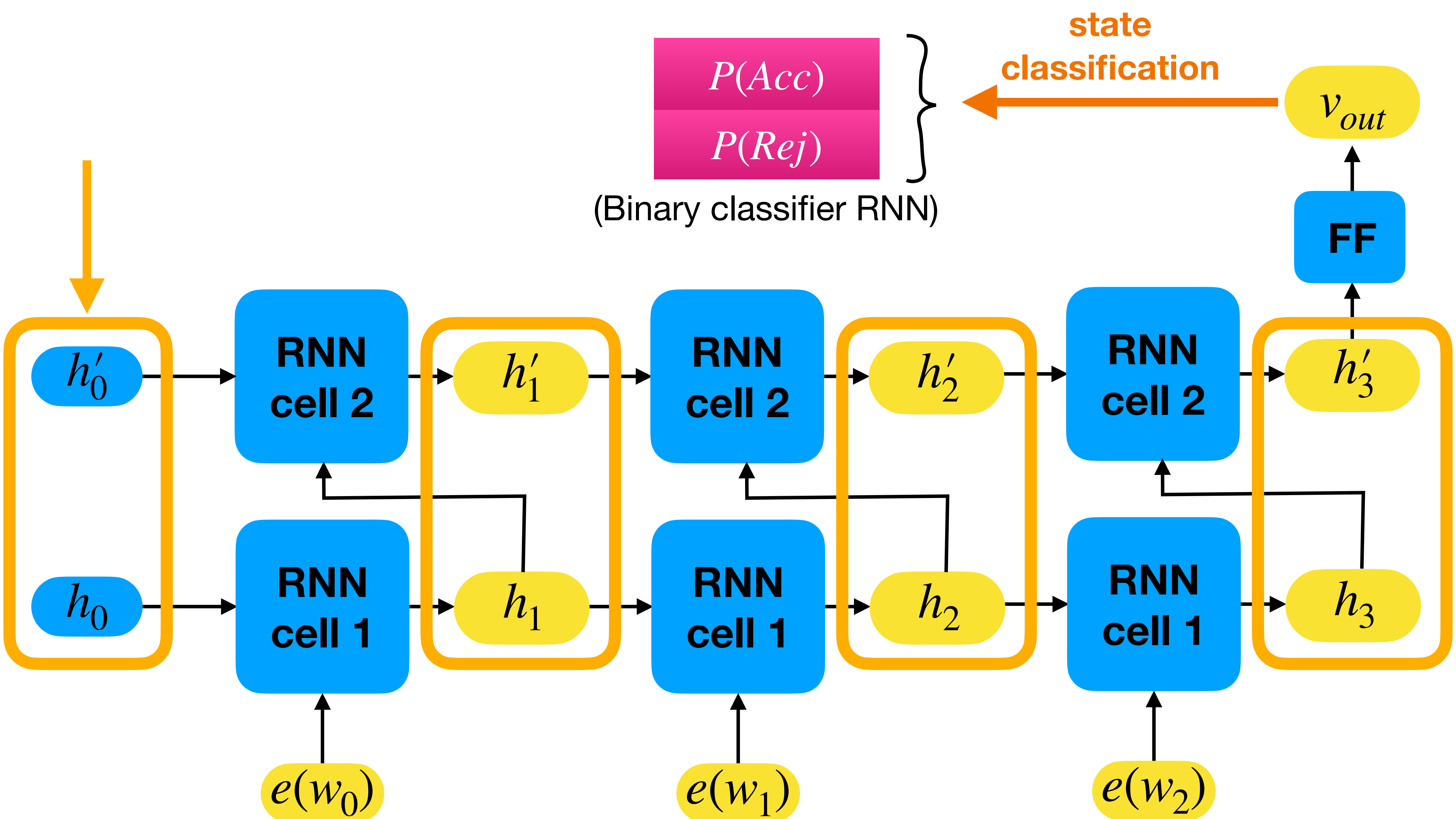
$$w = w_0 w_1 w_2 \in \Sigma^*$$

RNNs: Automata Relation



$$w = w_0 w_1 w_2 \in \Sigma^*$$

RNNs: Automata Relation



$$w = w_0 w_1 w_2 \in \Sigma^*$$

RNNs: Automata Relation

When learning a regular language, simple RNNs (Elman RNNs) cluster their states in manner that resembles an automaton for that language

Finite State Automata and Simple Recurrent Networks

- Cleeremans et al, 1989 (references older version of Elman 1990)

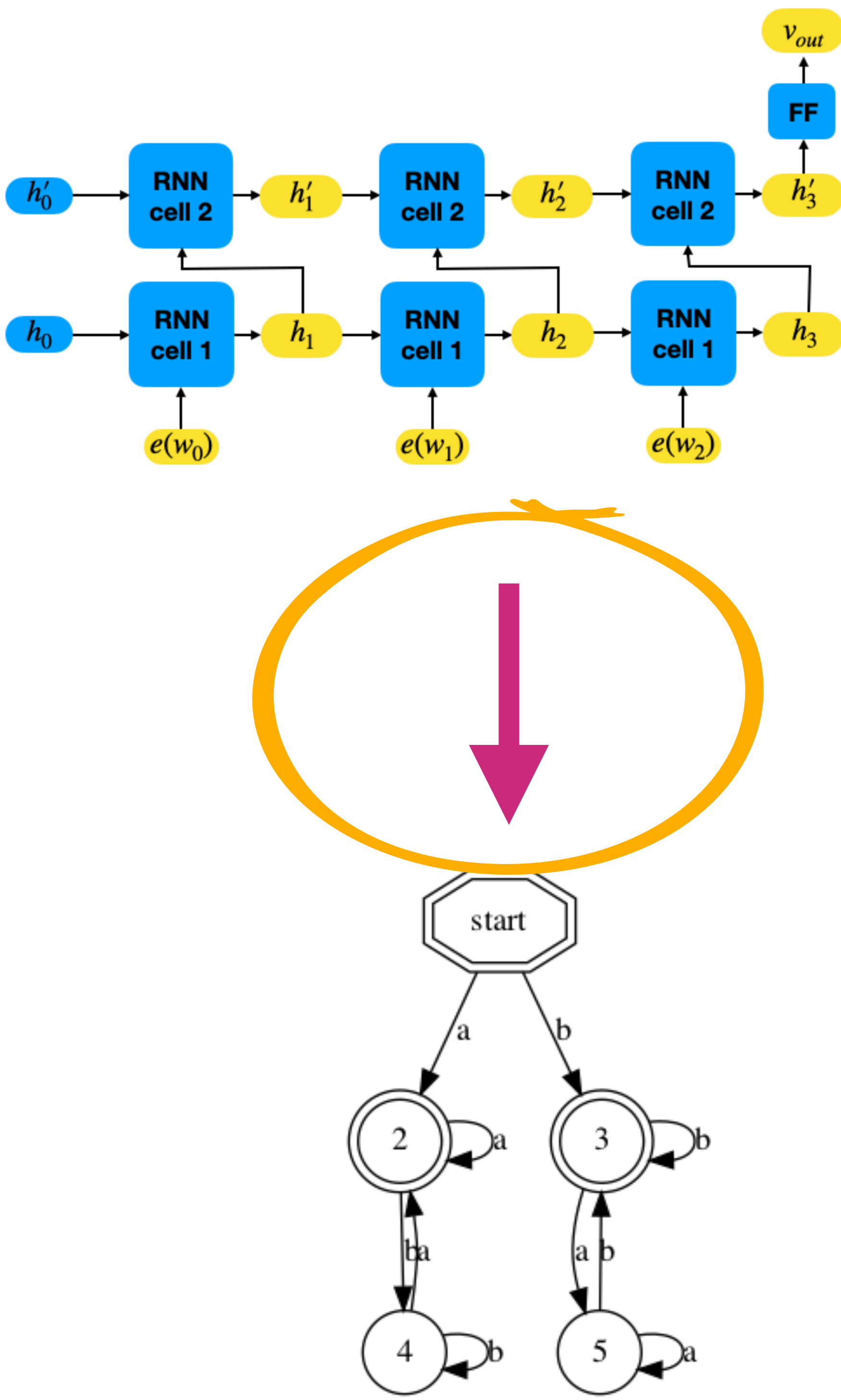
Overview

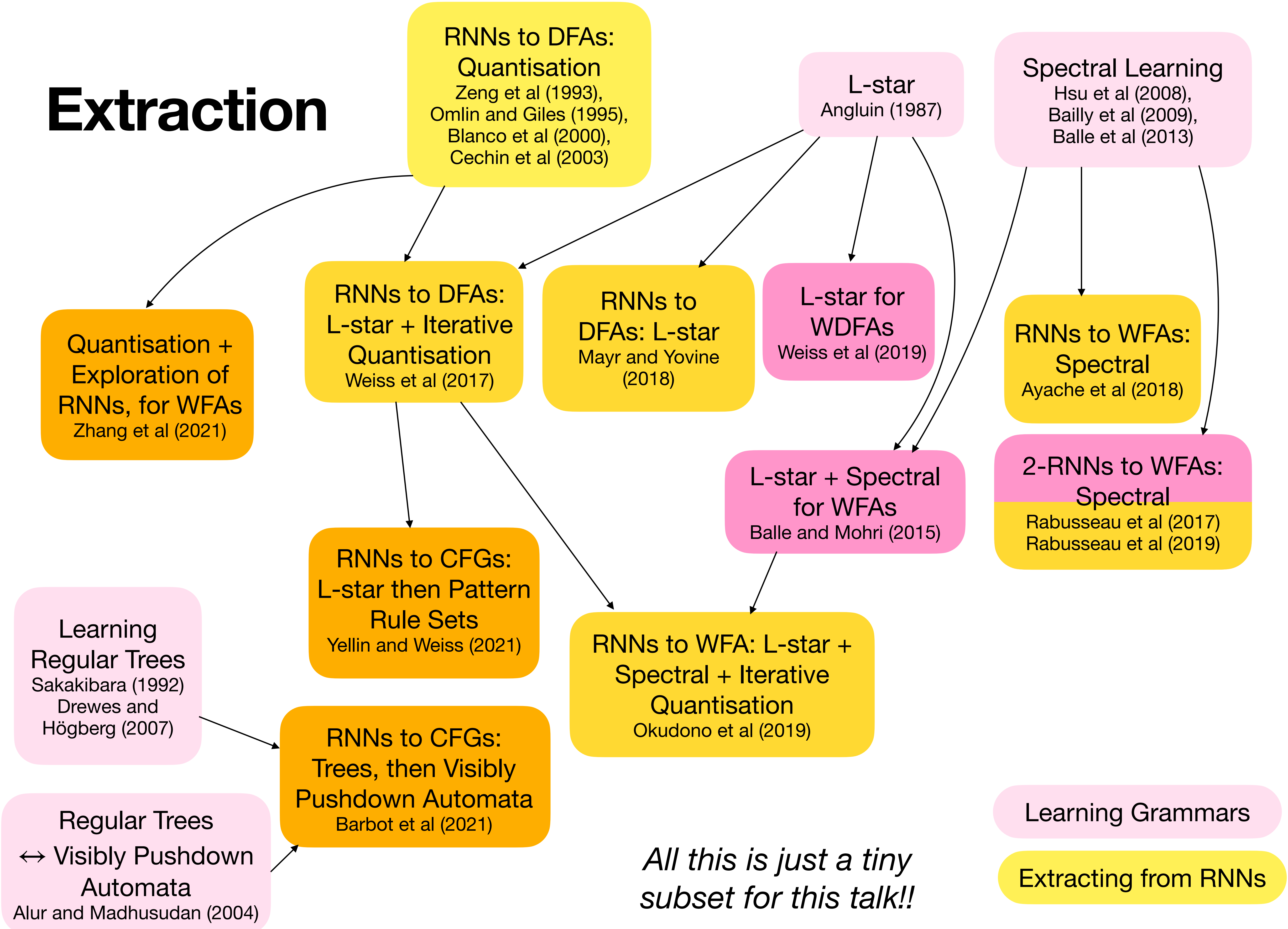
Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

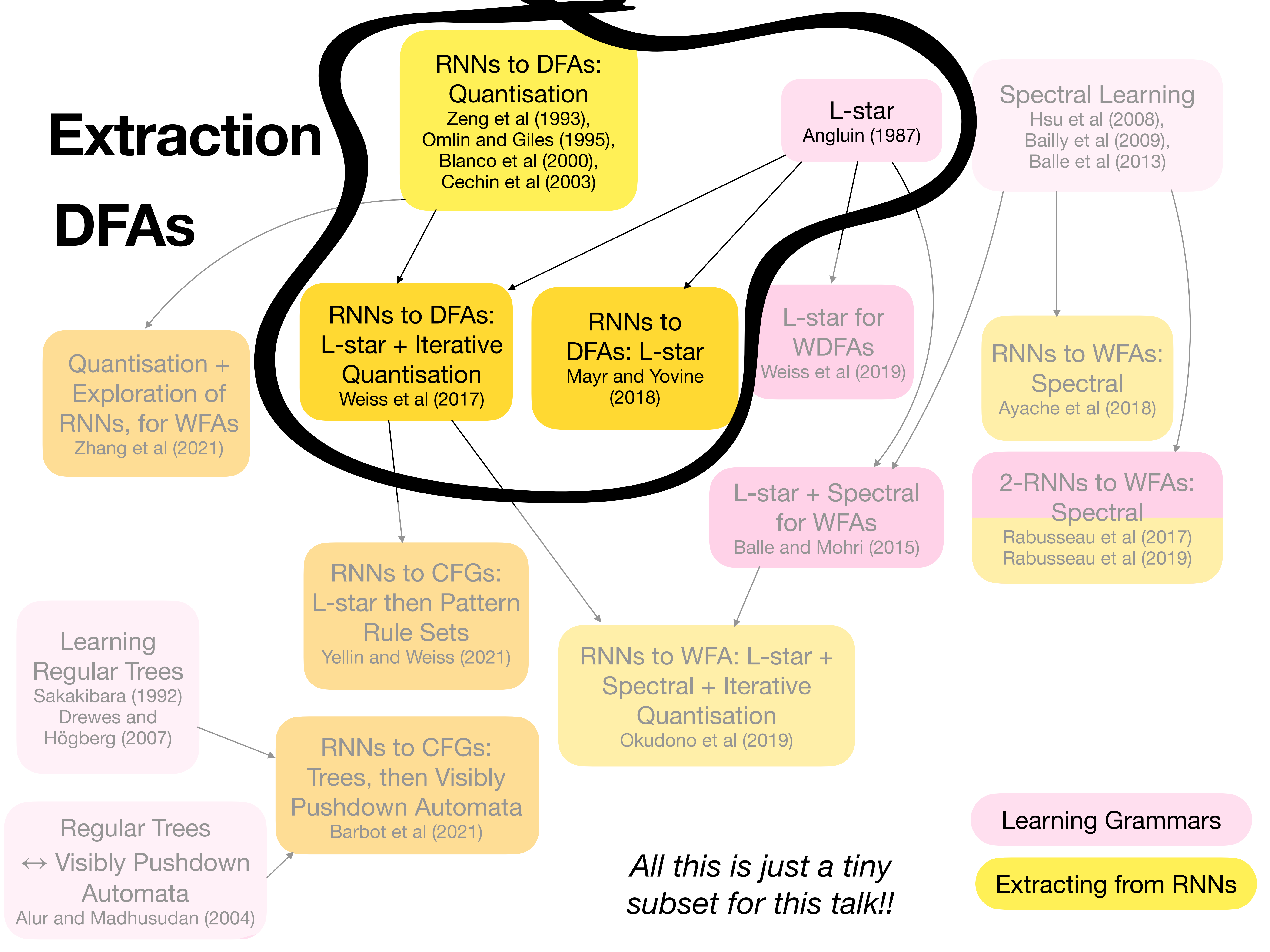
Transformers

- Introduction
- A formal abstraction

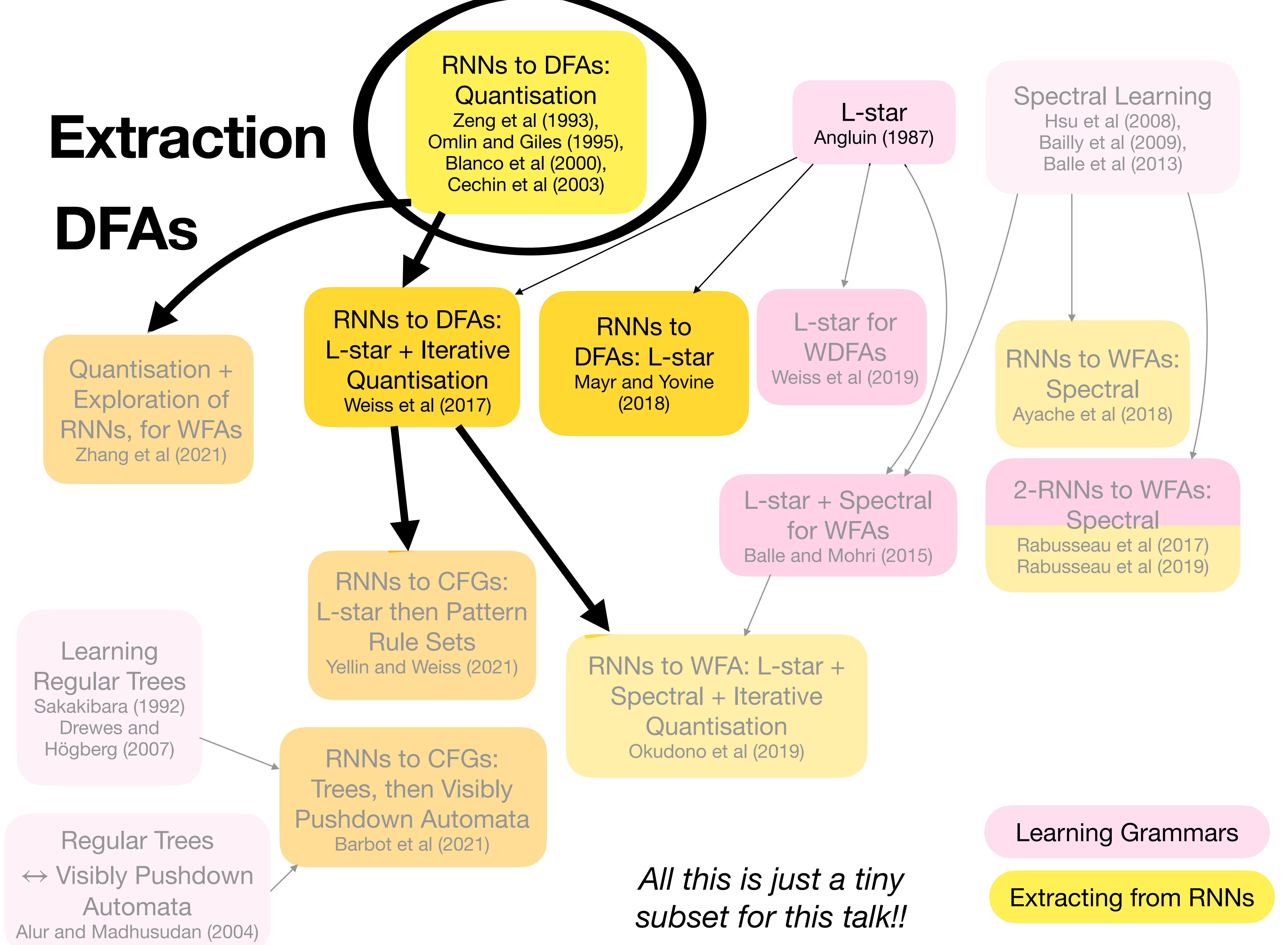




Extraction
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
All this is just a tiny subset for this talk!!
Learning Grammars
Extracting from RNNs


Extraction
DFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
All this is just a tiny subset for this talk!!
Learning Grammars
Extracting from RNNs

Extraction DFAs



All this is just a tiny subset for this talk!!

RNNs: Extracting DFAs: Clustering

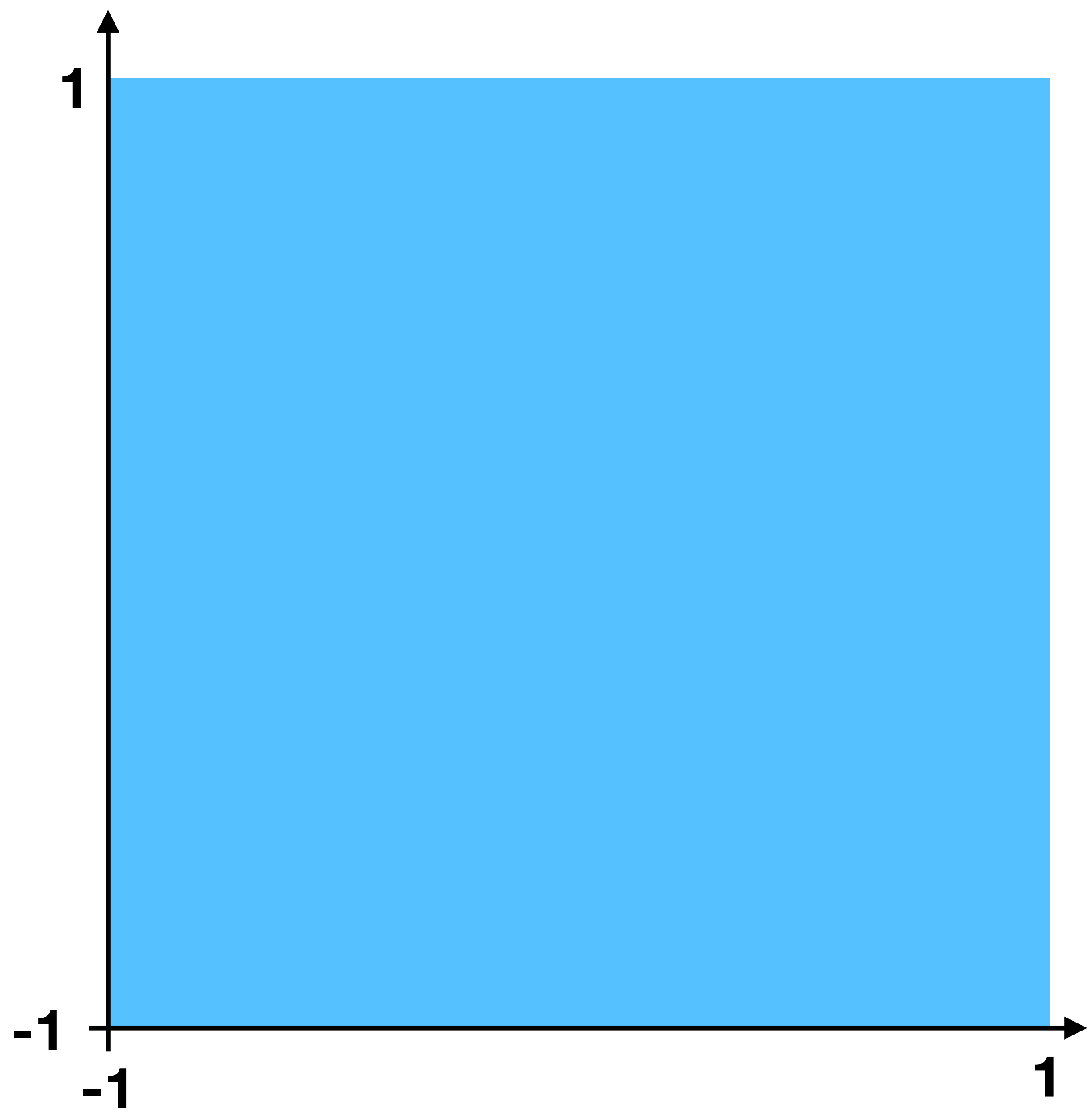
Omlin and Giles, 1996

Partition the RNN state space by dividing each dimension into q equal portions. Explore the partitions, marking transitions between them according to first-visited state in each partition

Extraction of Rules from Discrete-time Recurrent Neural Networks

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)

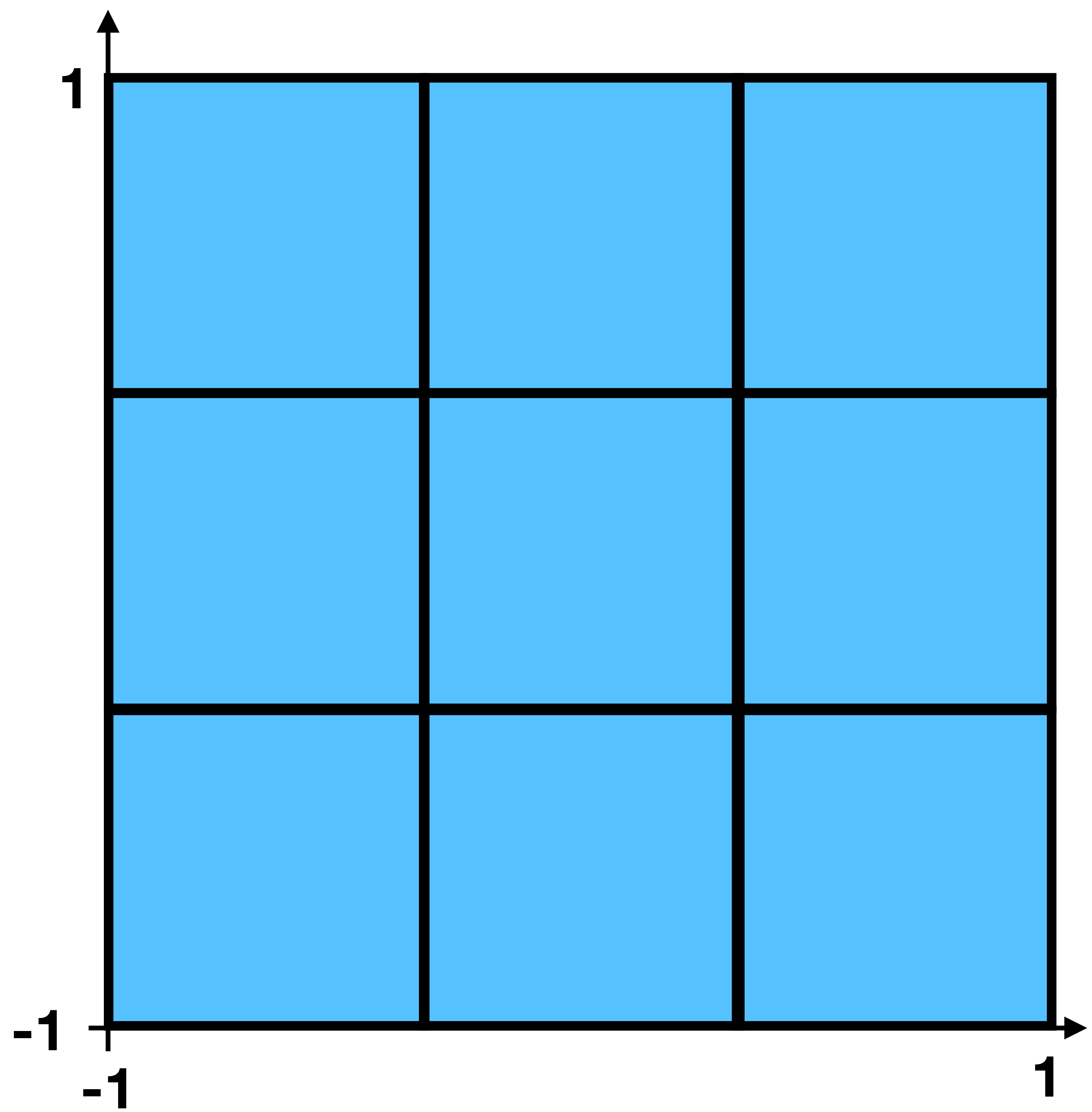


Input alphabet: {a,b}

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



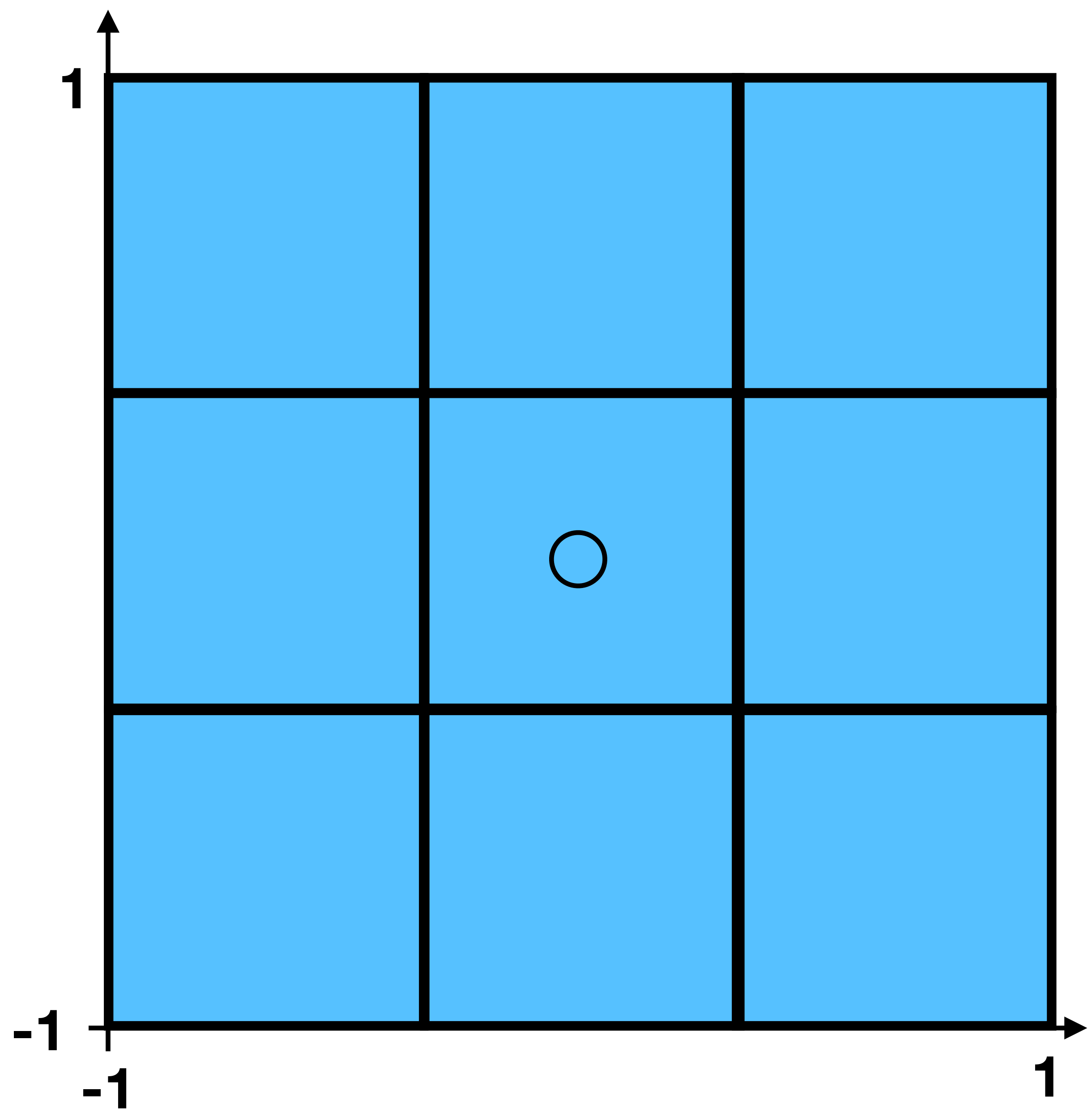
Input alphabet: {a,b}

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

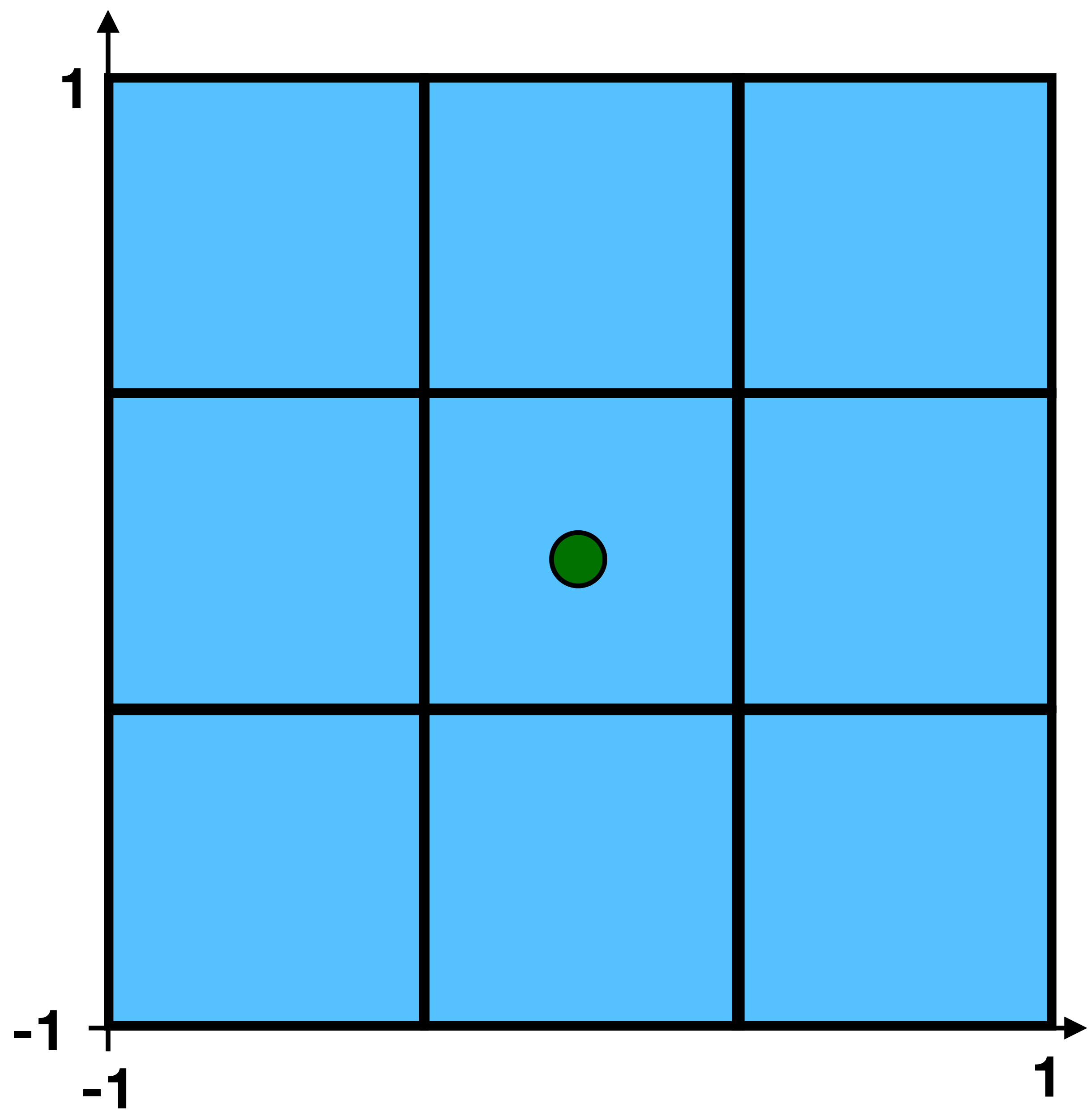
○ Initial state: (0,0)

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

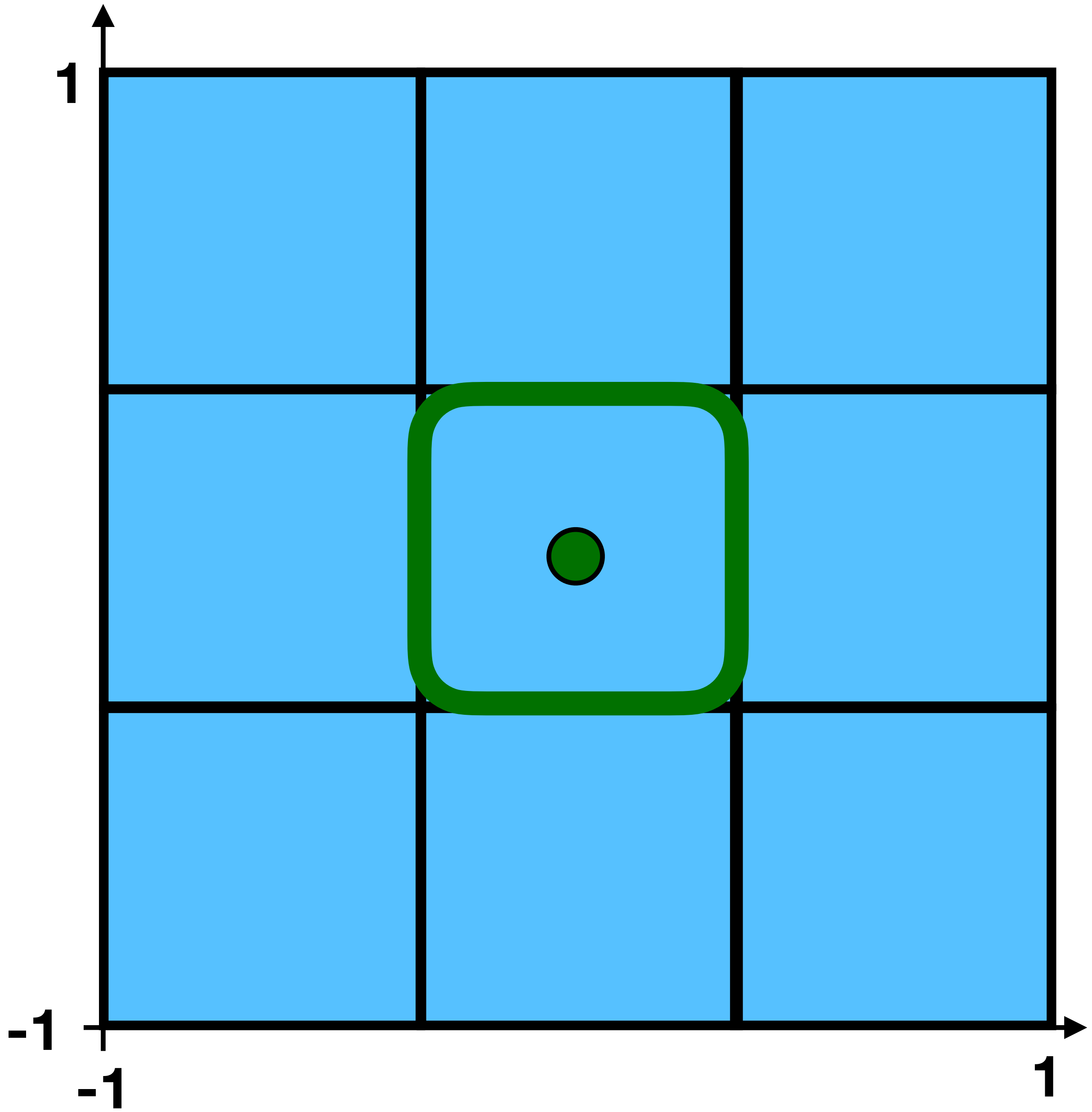
Rejecting state

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

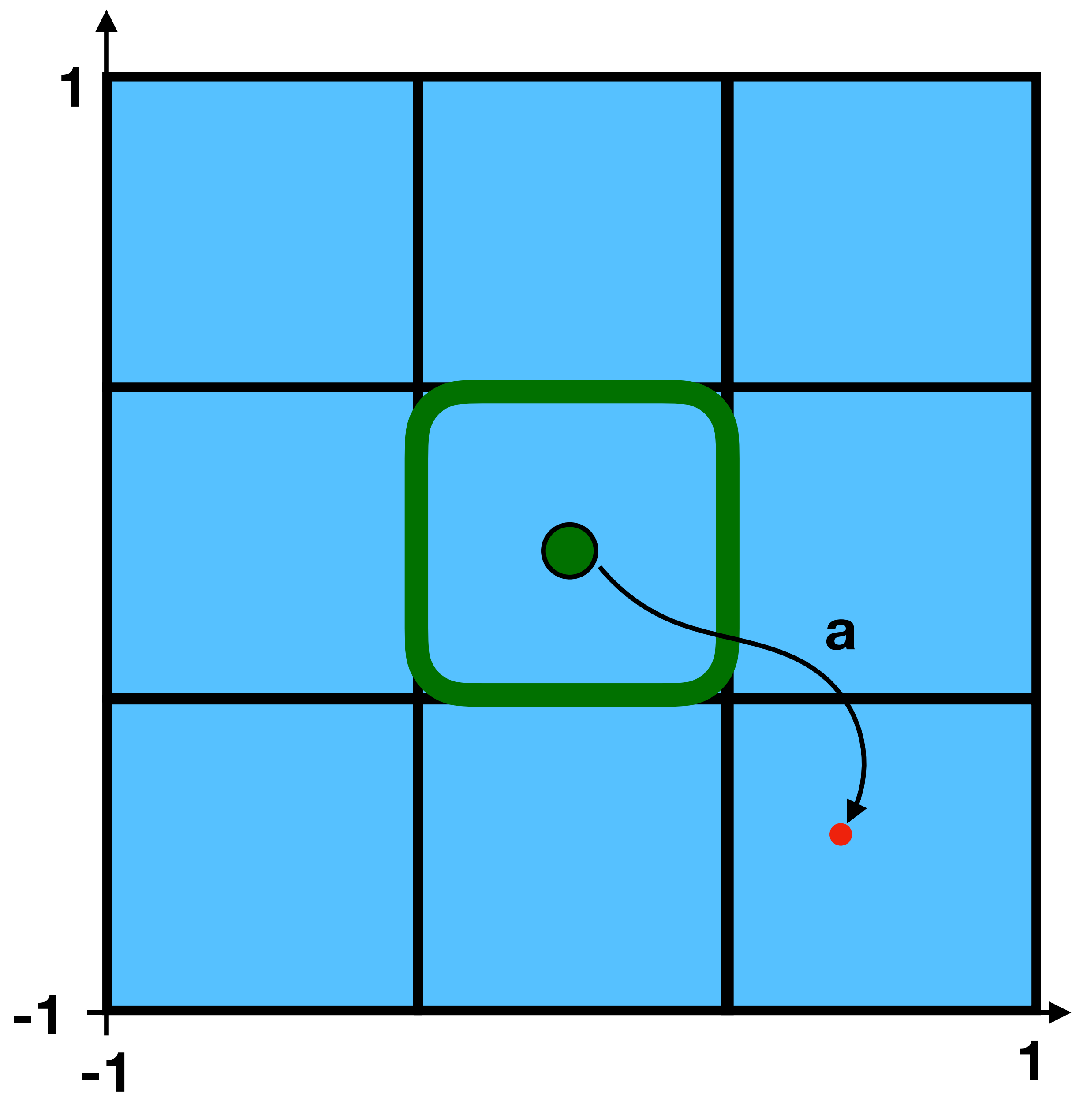
Rejecting state

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

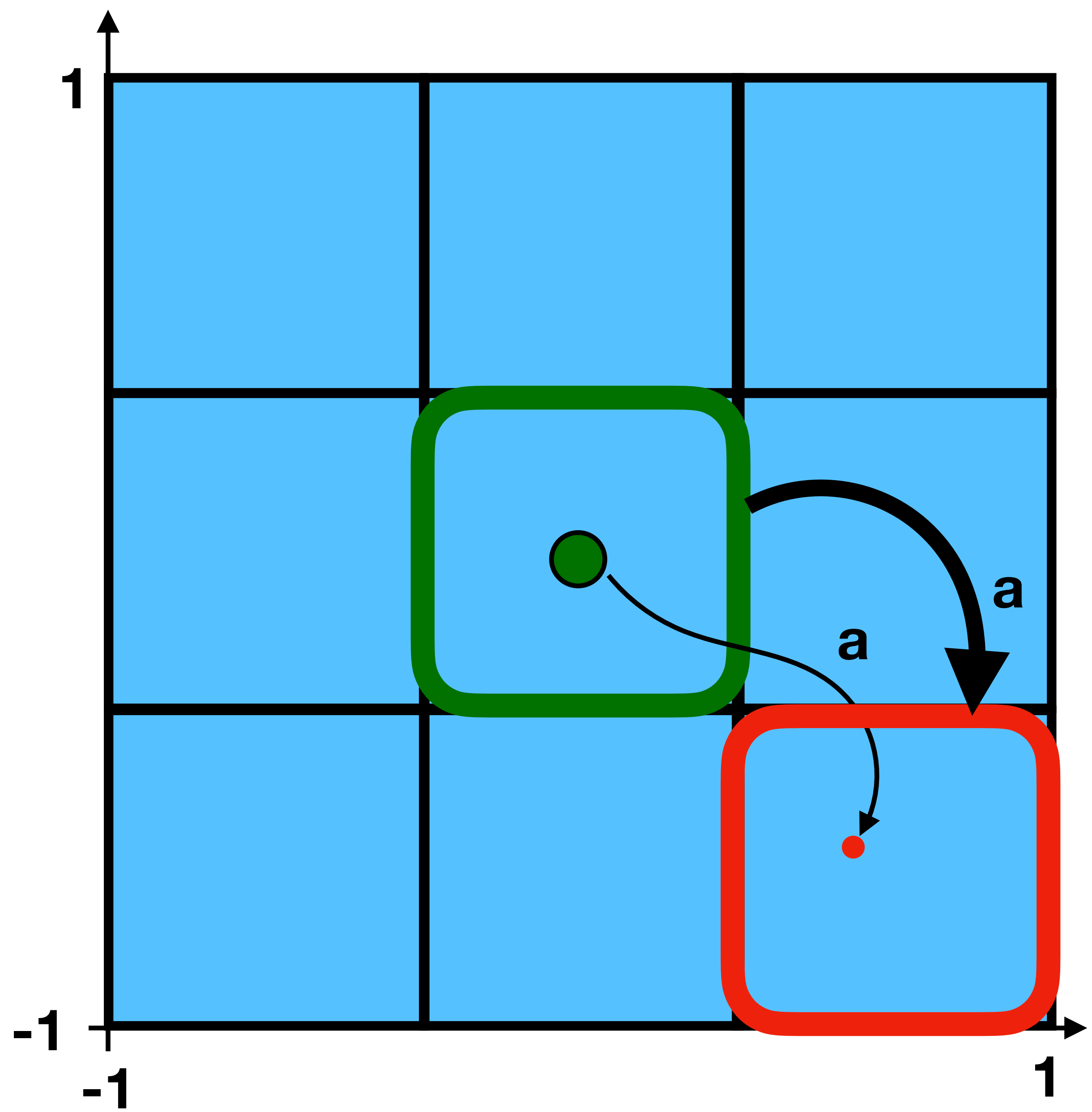
Rejecting state

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

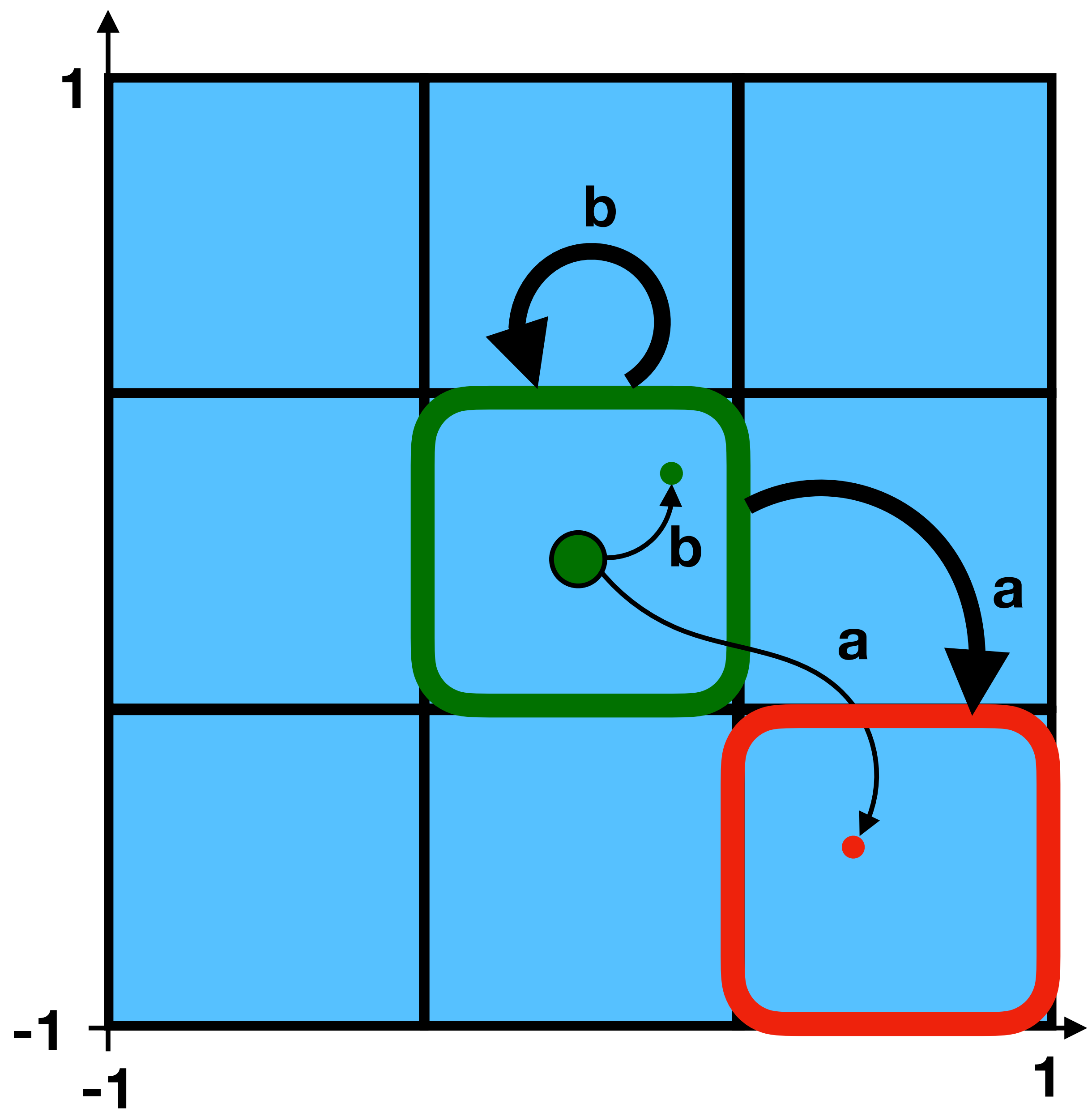
Rejecting state

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

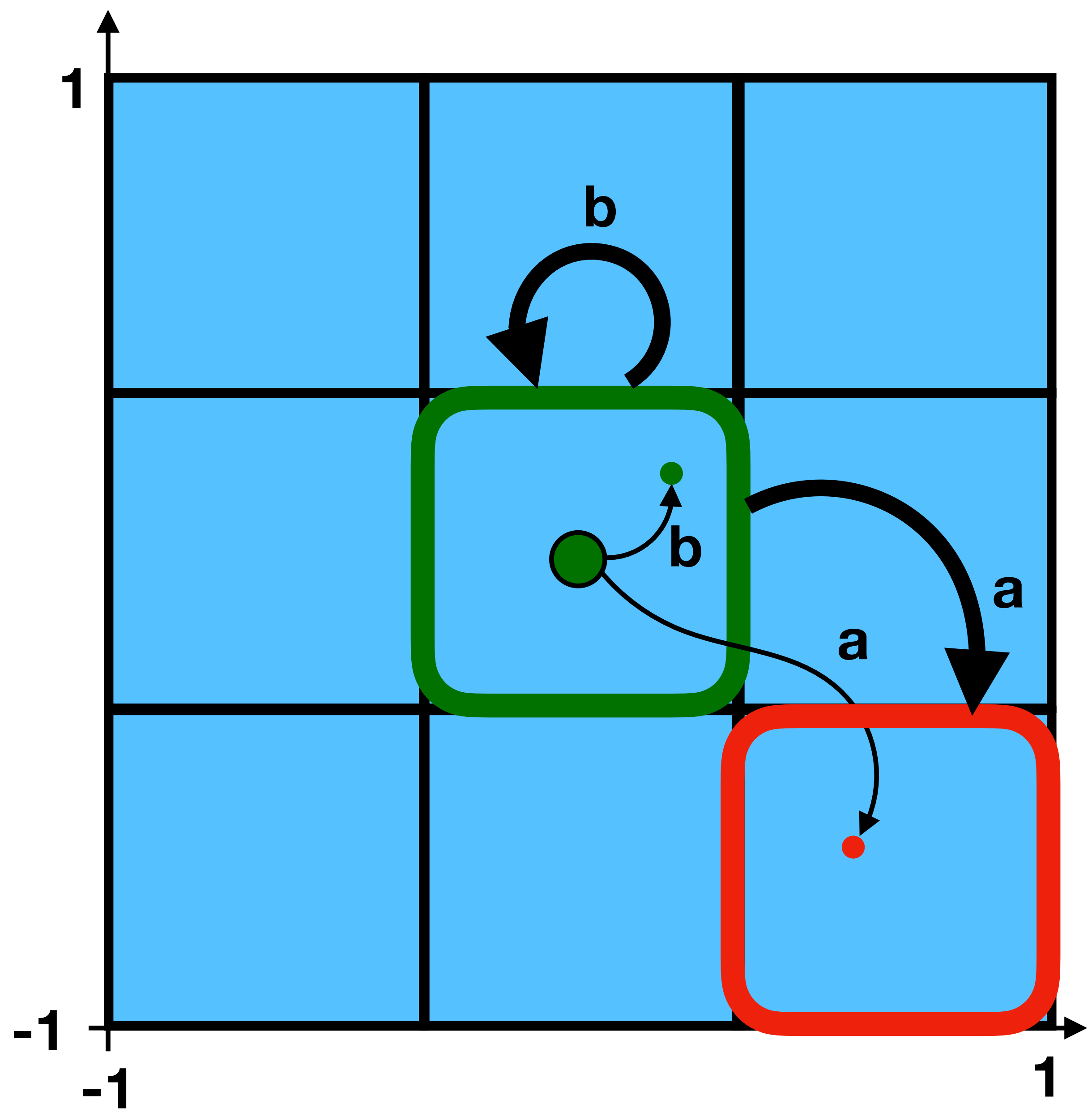
Rejecting state

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

Rejecting state

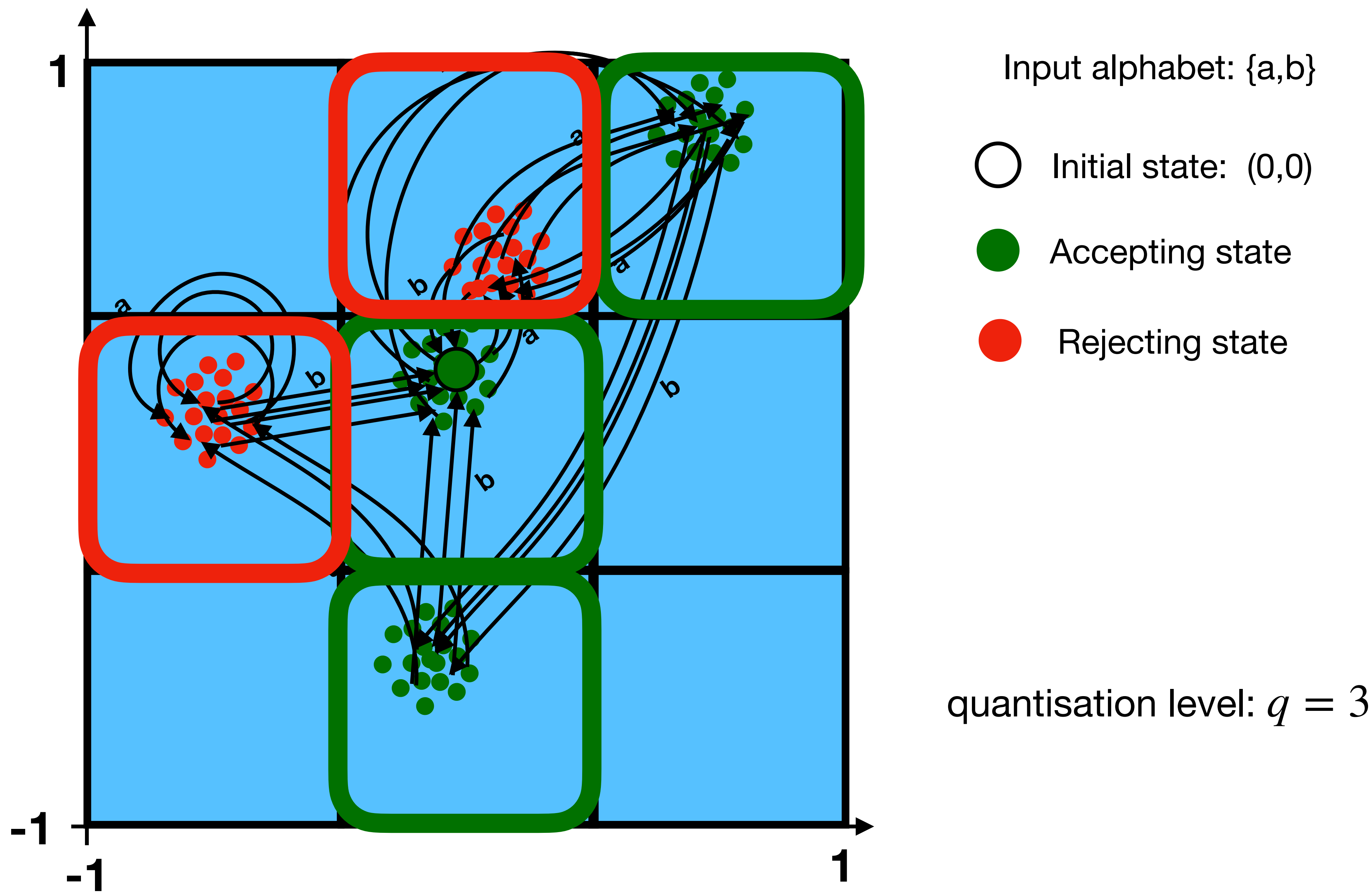
etc…

quantisation level: $q = 3$

Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

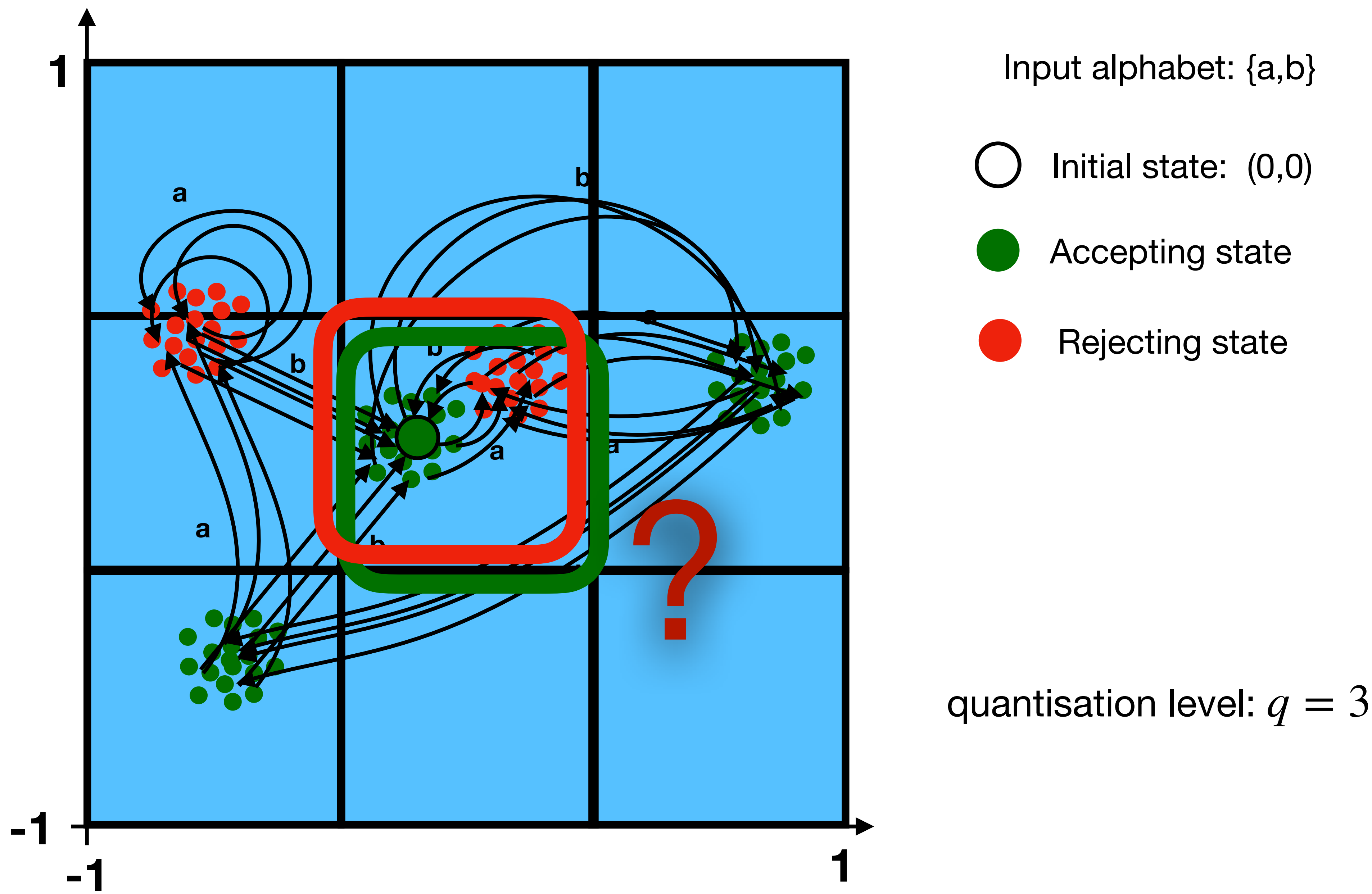
Quantisation, example (on RNN with total hidden state dimension 2)



Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

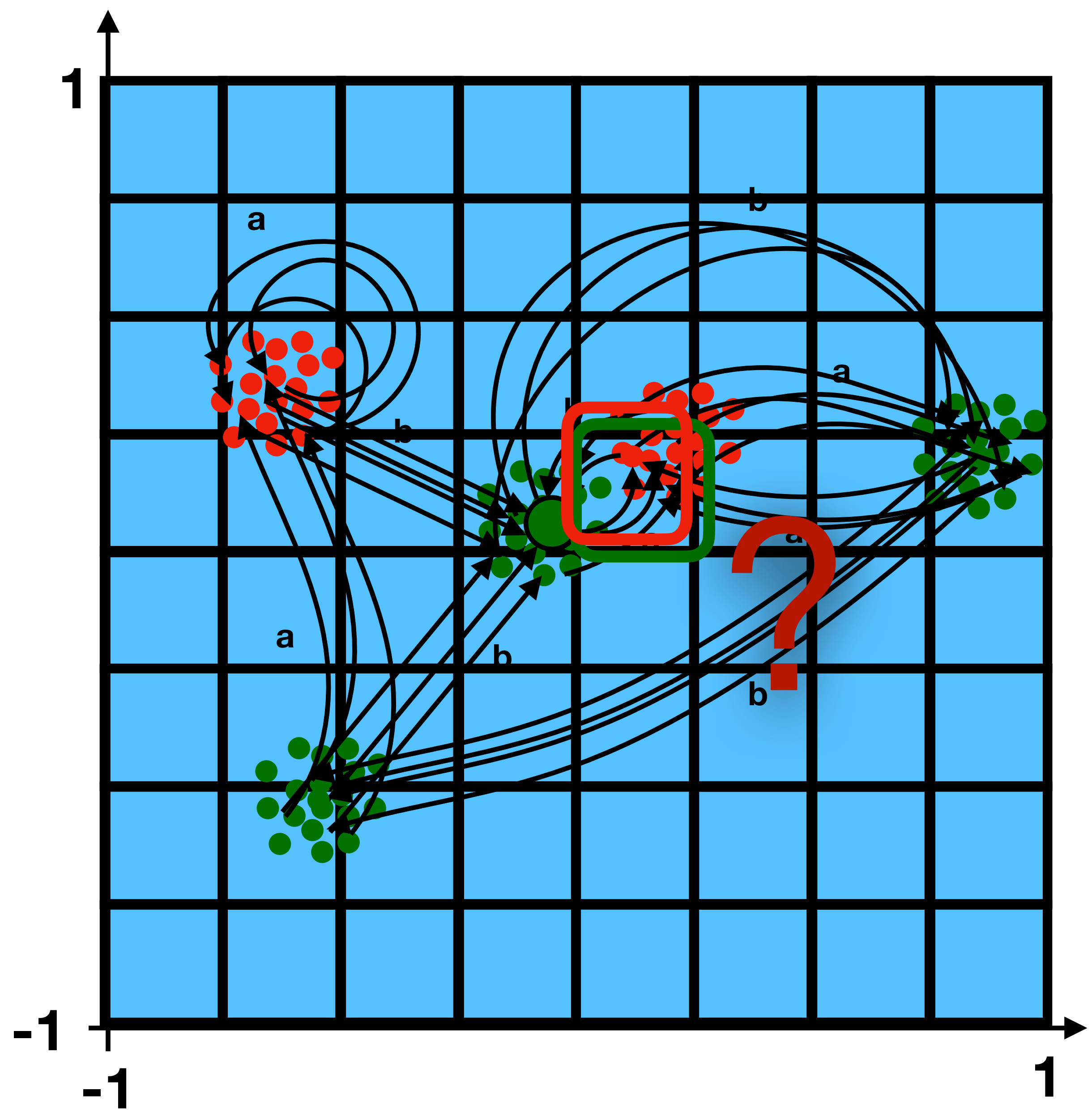
Quantisation, example (on RNN with total hidden state dimension 2)



Extraction of Rules from Discrete-time Recurrent Neural Networks
- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Quantisation, example (on RNN with total hidden state dimension 2)



Input alphabet: {a,b}

Initial state: (0,0)

Accepting state

Rejecting state

quantisation level: $q = 8$

Extraction of Rules from Discrete-time Recurrent Neural Networks

- Omlin and Giles, 1996

RNNs: Extracting DFAs: Clustering

Other approaches to clustering

Learning Finite State Machines With Self-clustering Recurrent Networks

Zeng et al, 1993

Extracting Rules from a (Fuzzy / Crisp) Recurrent Neural Network using a Self-Organizing Map

Blanco et al, 2000

State automata extraction from recurrent neural nets using k-means and fuzzy clustering

Cechin et al, 2003

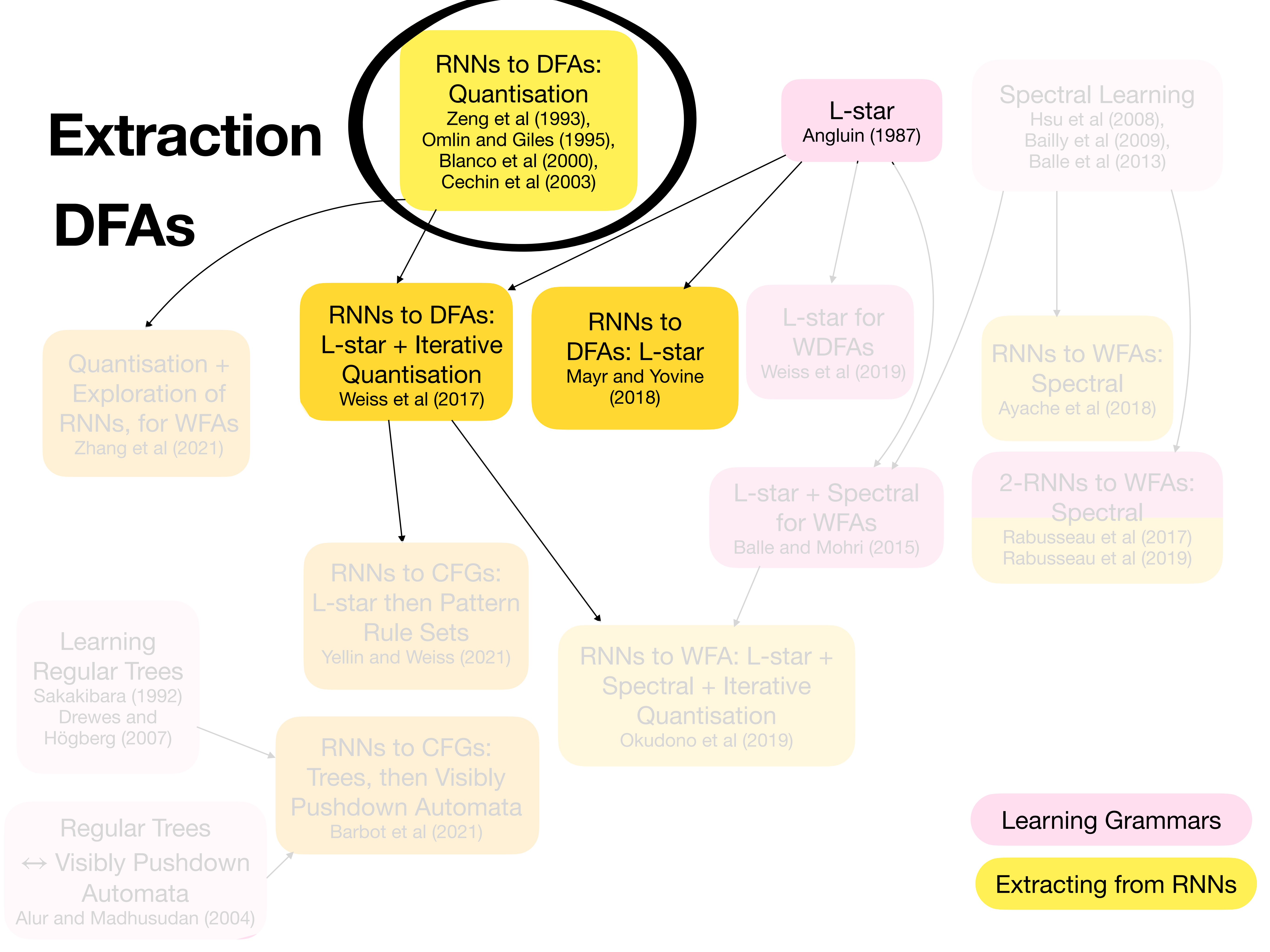
Surveys:

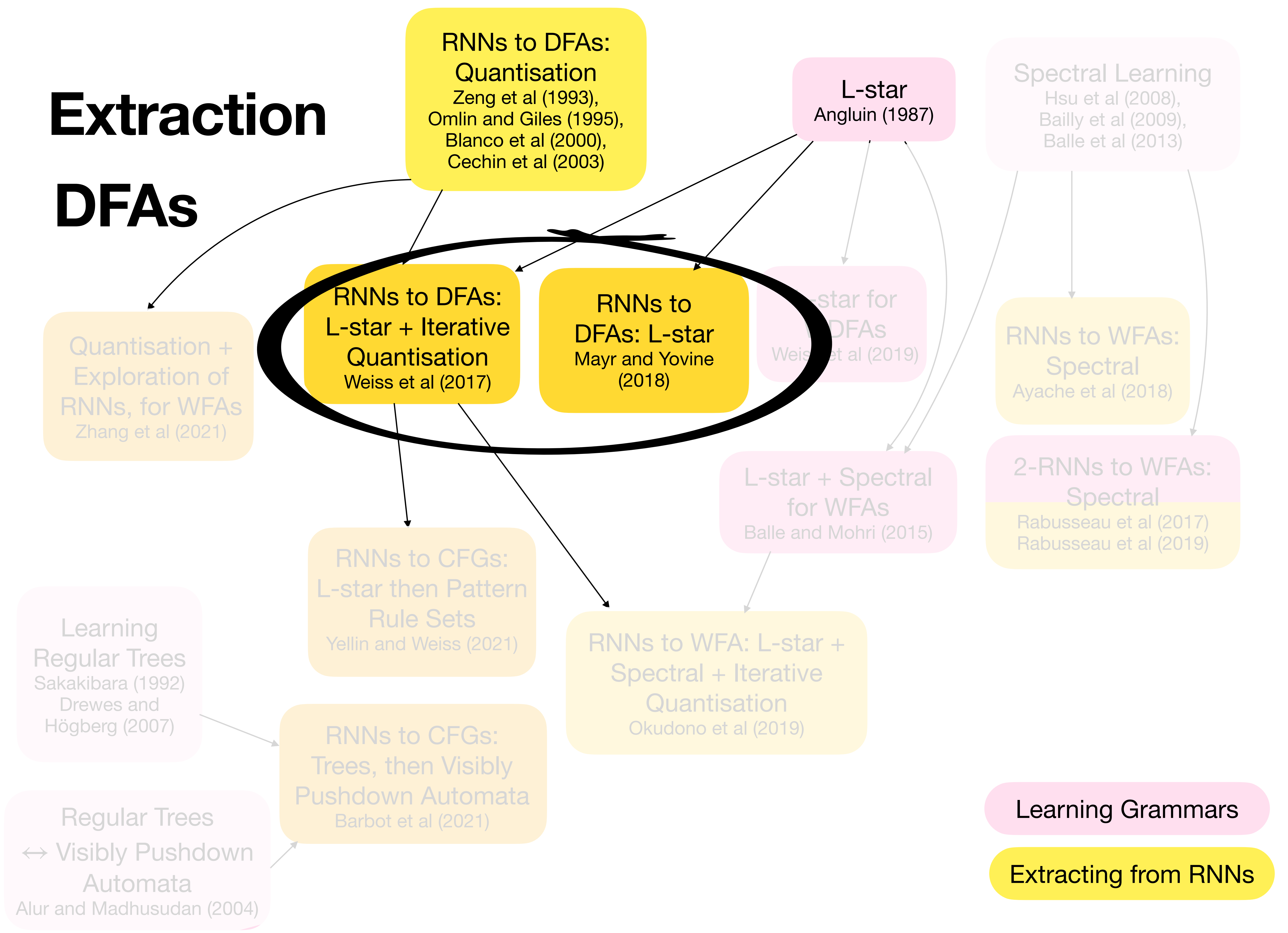
Rule Extraction from Recurrent Neural Networks: A Taxonomy and Review

Jacobsson, 2005

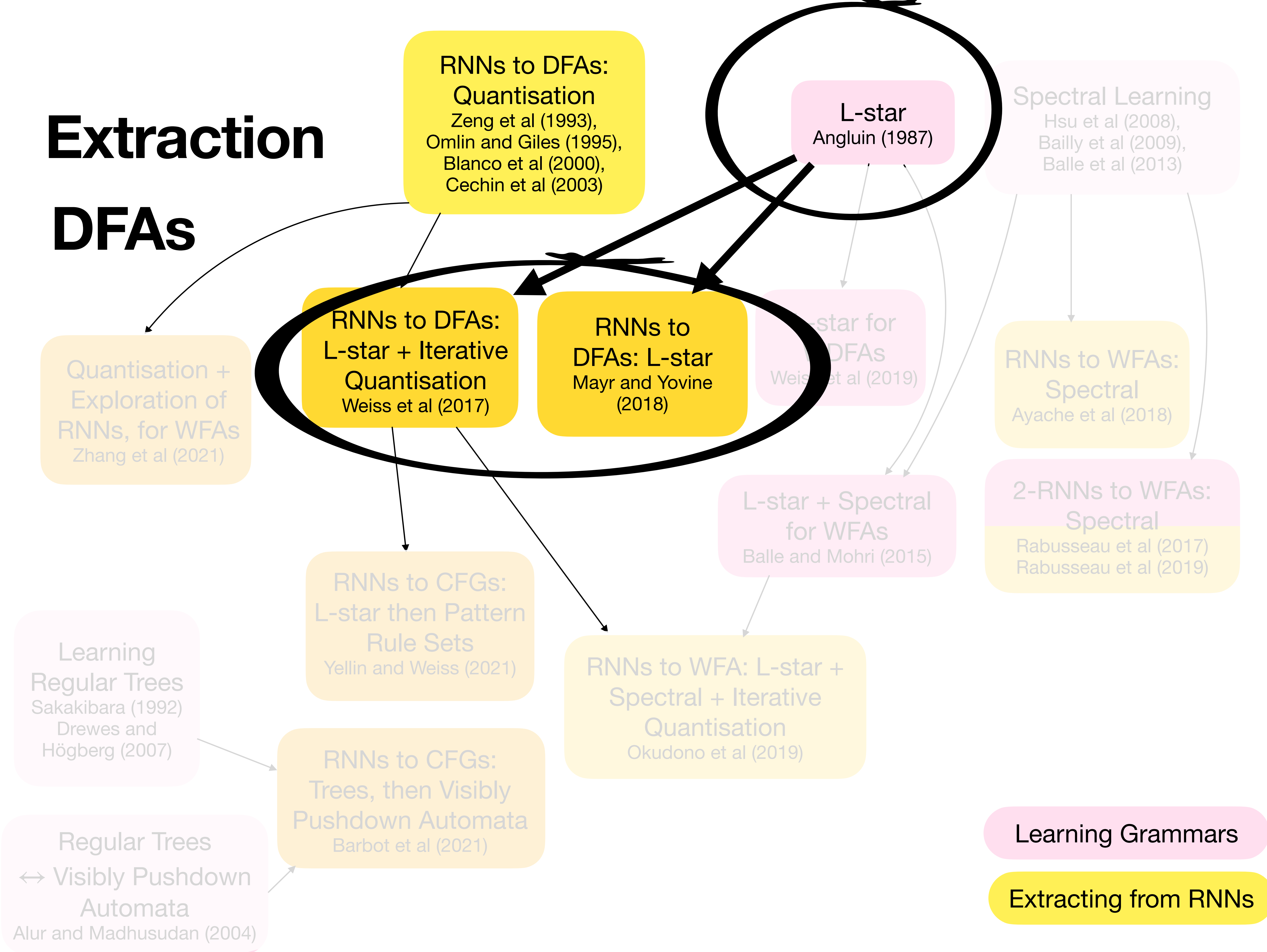
An Empirical Evaluation of Rule Extraction from Recurrent Neural Networks

Wang et al, 2017

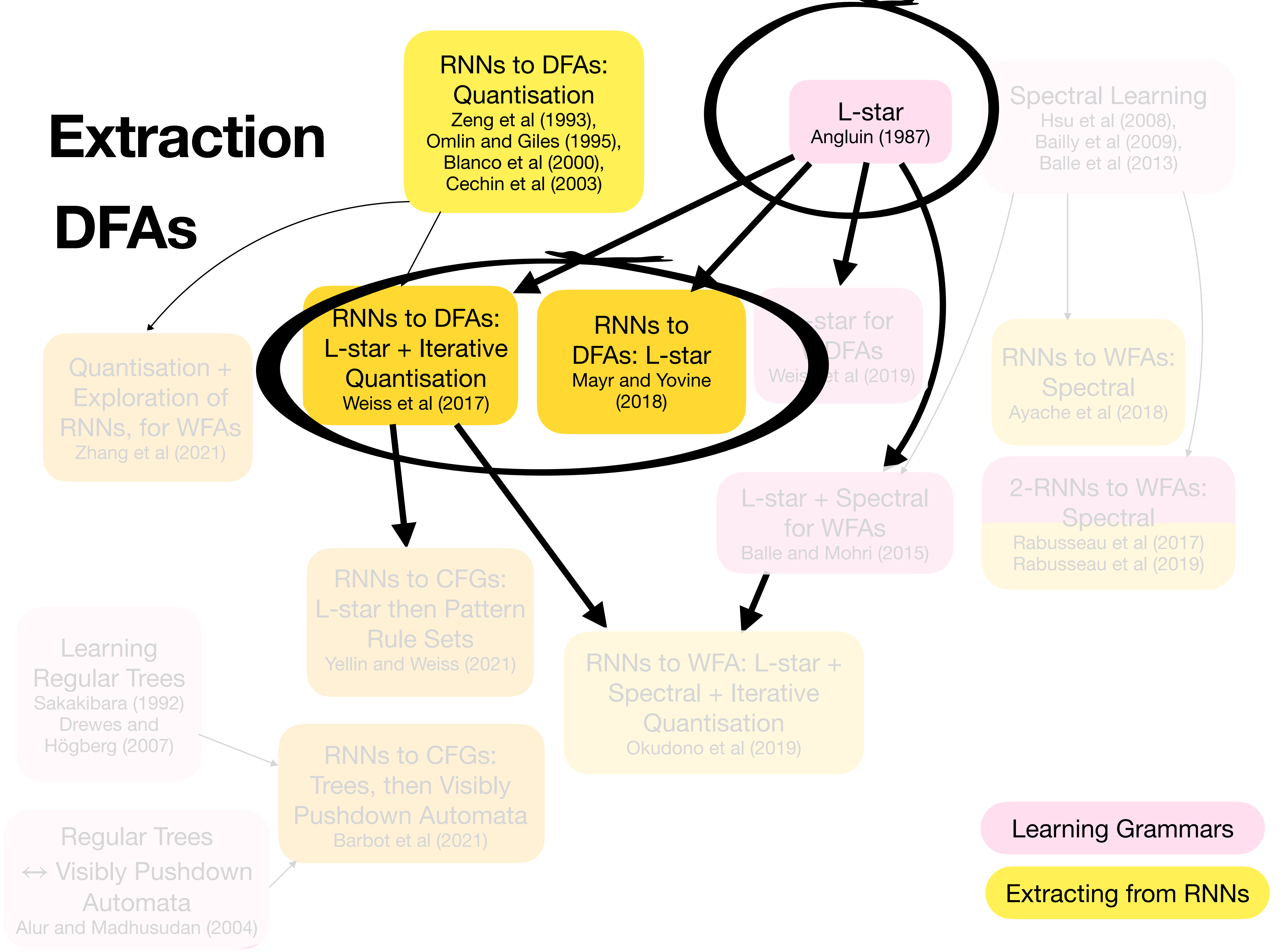

Extraction
DFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs


Extraction
DFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
star for
DFAs
Weis
et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
L-star + Spectral for WFAs
Balle and Mohri (2015)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

Extraction DFAs



Extraction DFAs



RNNs: Extracting DFAs: L-star

The L-star algorithm

$\varepsilon?$

$a?$

$b?$

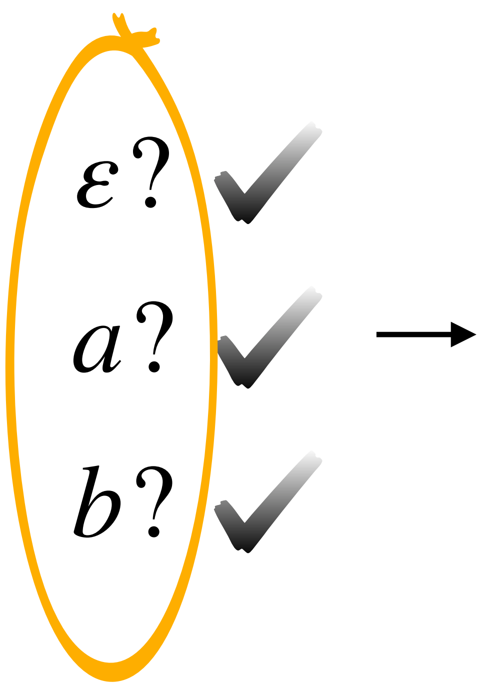
Membership Queries

Learning Regular Sets from Queries and Counterexamples

Angluin 1987

RNNs: Extracting DFAs: L-star

The L-star algorithm



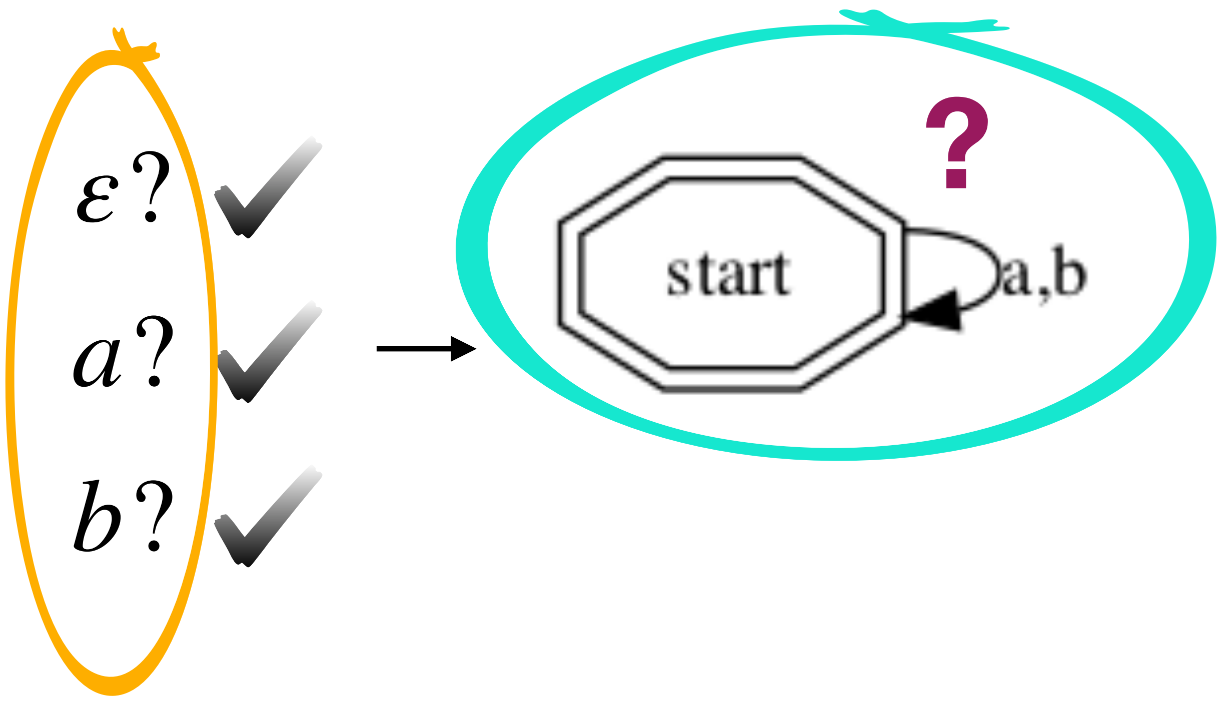
Membership Queries

Learning Regular Sets from Queries and Counterexamples

Angluin 1987

RNNs: Extracting DFAs: L-star

The L-star algorithm



Learning Regular Sets from Queries and Counterexamples

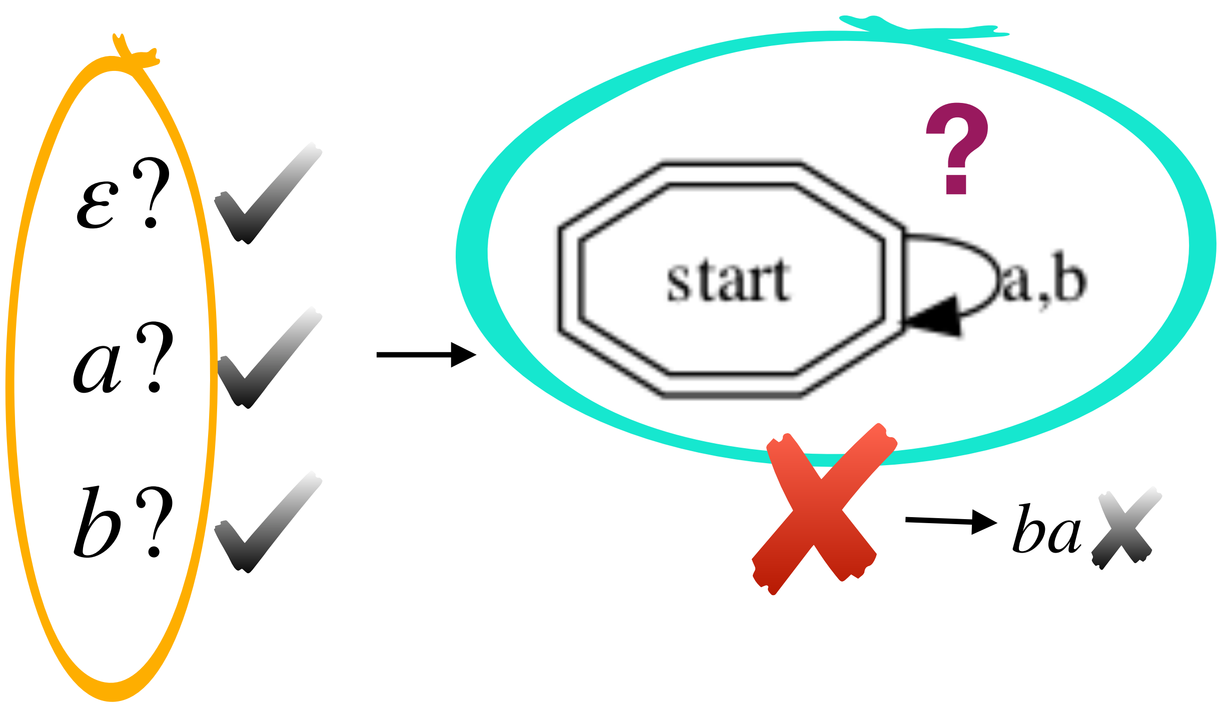
Angluin 1987

Membership Queries

Equivalence Queries

RNNs: Extracting DFAs: L-star

The L-star algorithm



Membership Queries

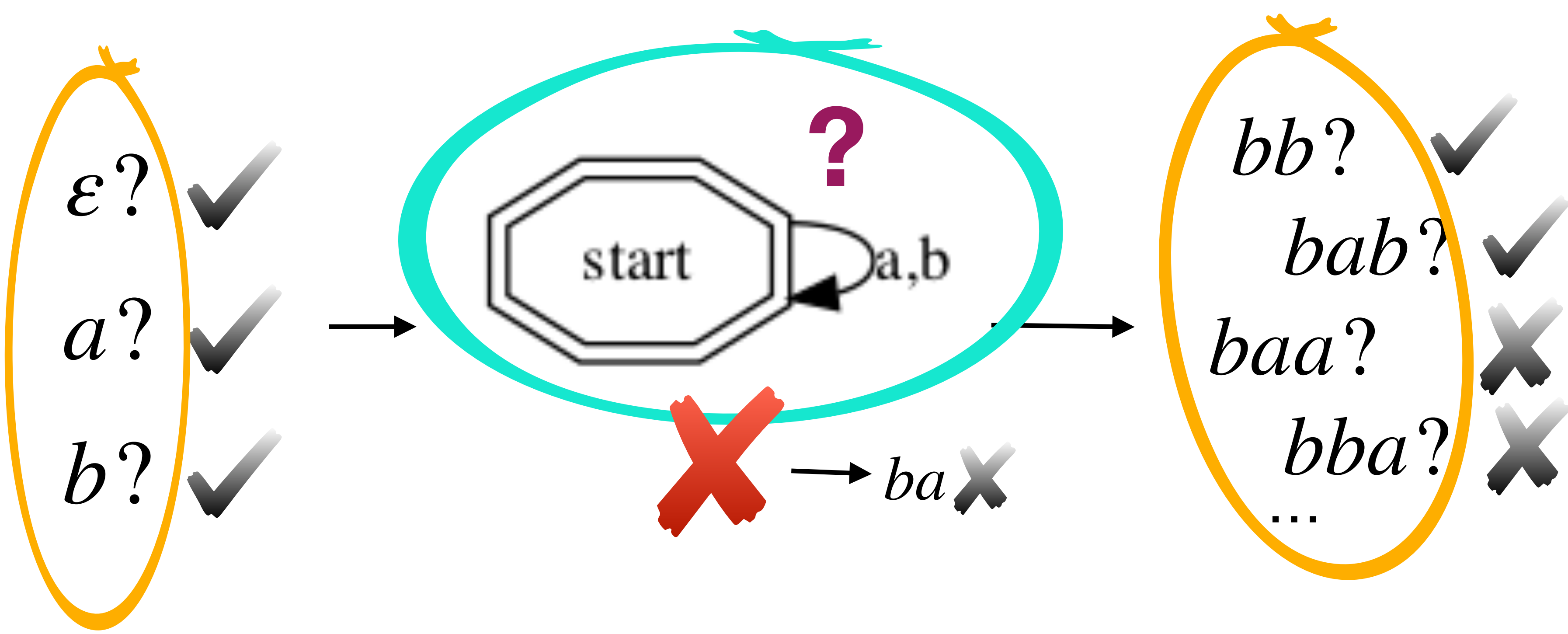
Equivalence Queries

Learning Regular Sets from Queries and Counterexamples

Angluin 1987

RNNs: Extracting DFAs: L-star

The L-star algorithm



Membership Queries

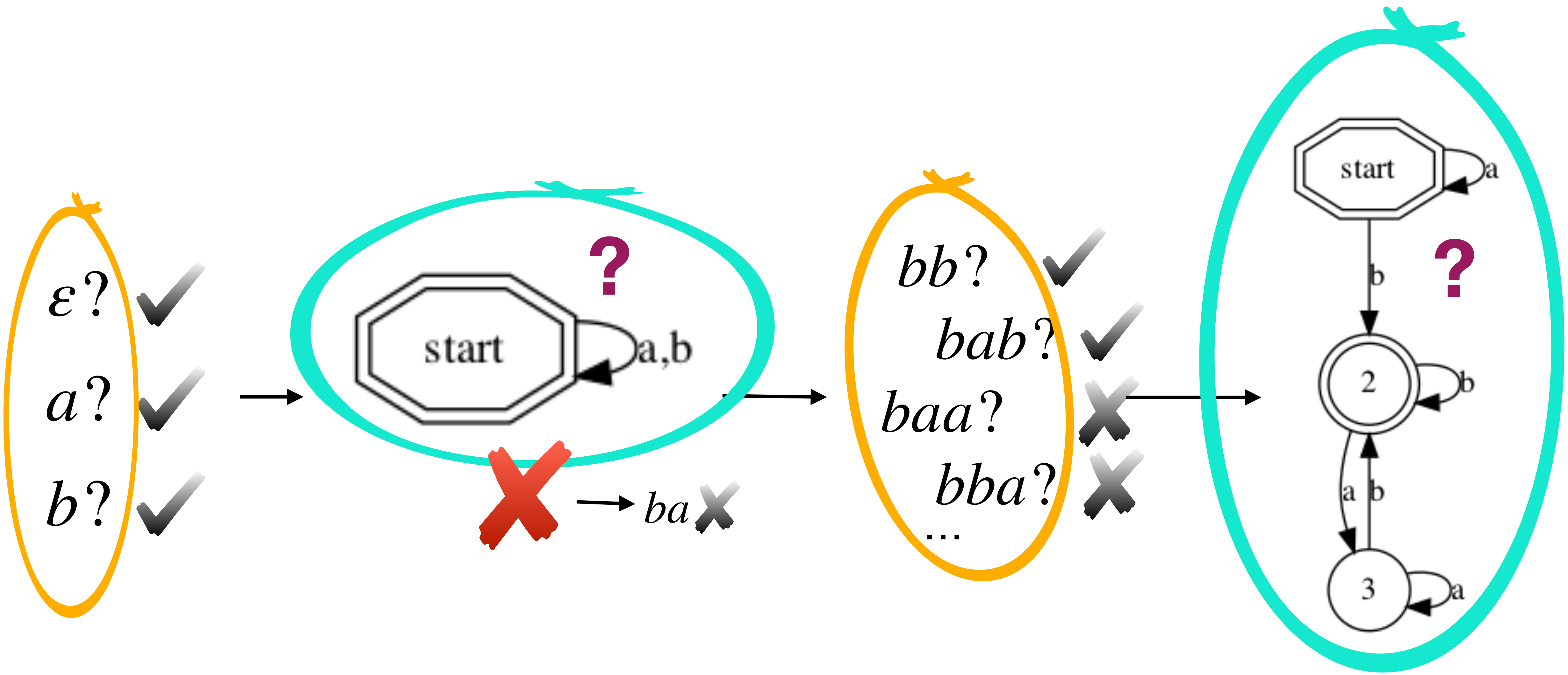
Equivalence Queries

Learning Regular Sets from Queries and Counterexamples

Angluin 1987

RNNs: Extracting DFAs: L-star

The L-star algorithm



Membership Queries

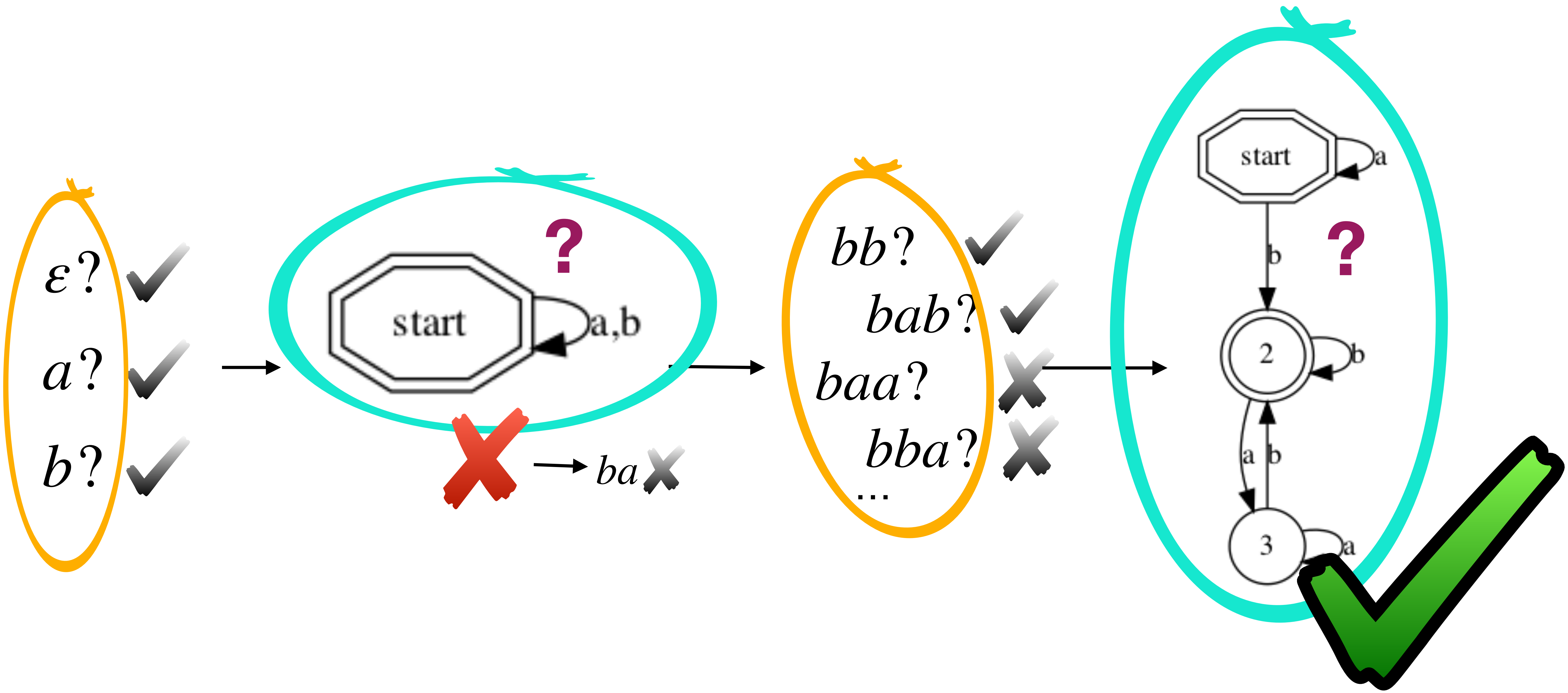
Equivalence Queries

Learning Regular Sets from
Queries and Counterexamples

Angluin 1987

RNNs: Extracting DFAs: L-star

The L-star algorithm



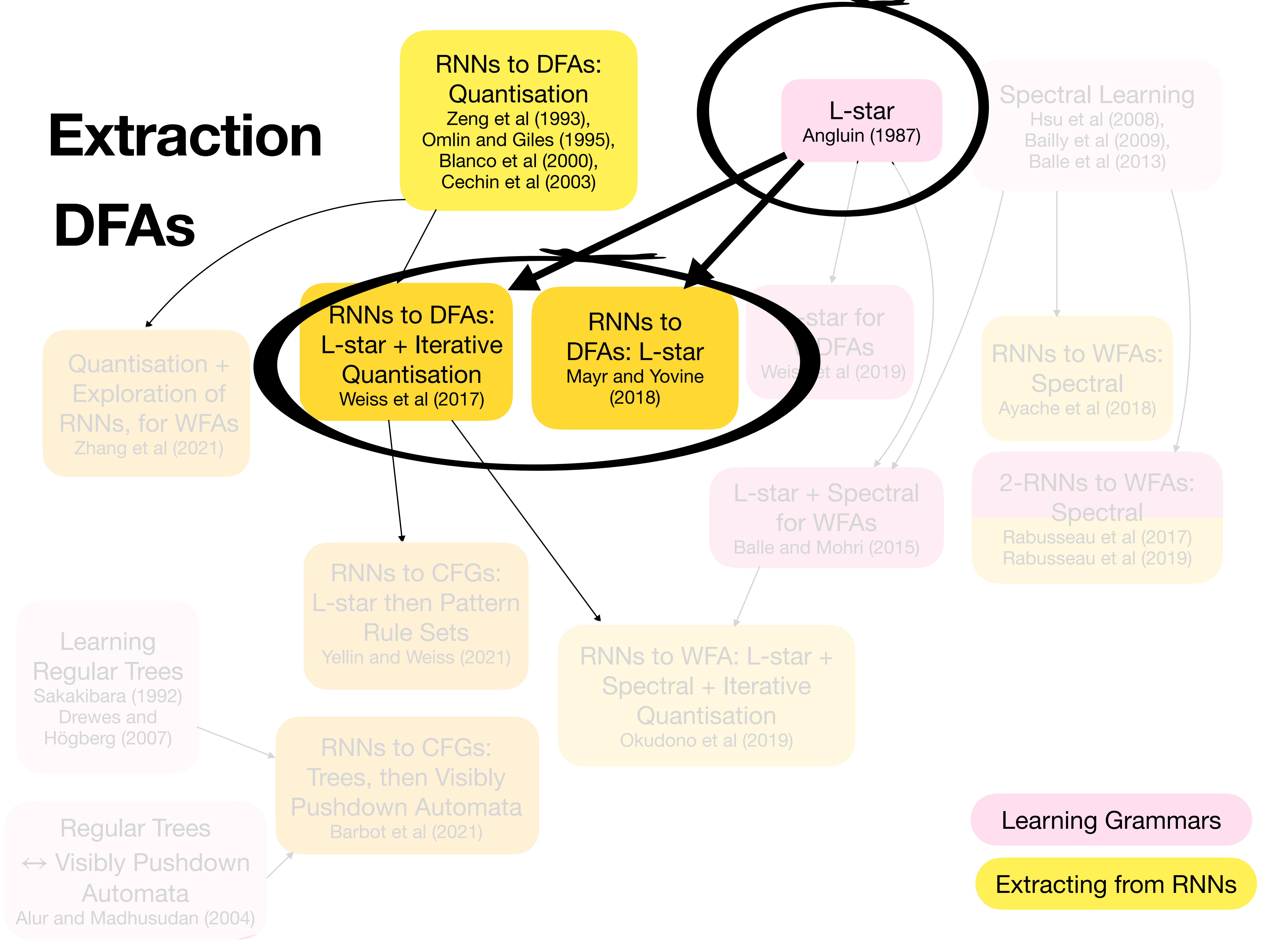
Learning Regular Sets from Queries and Counterexamples

Angluin 1987

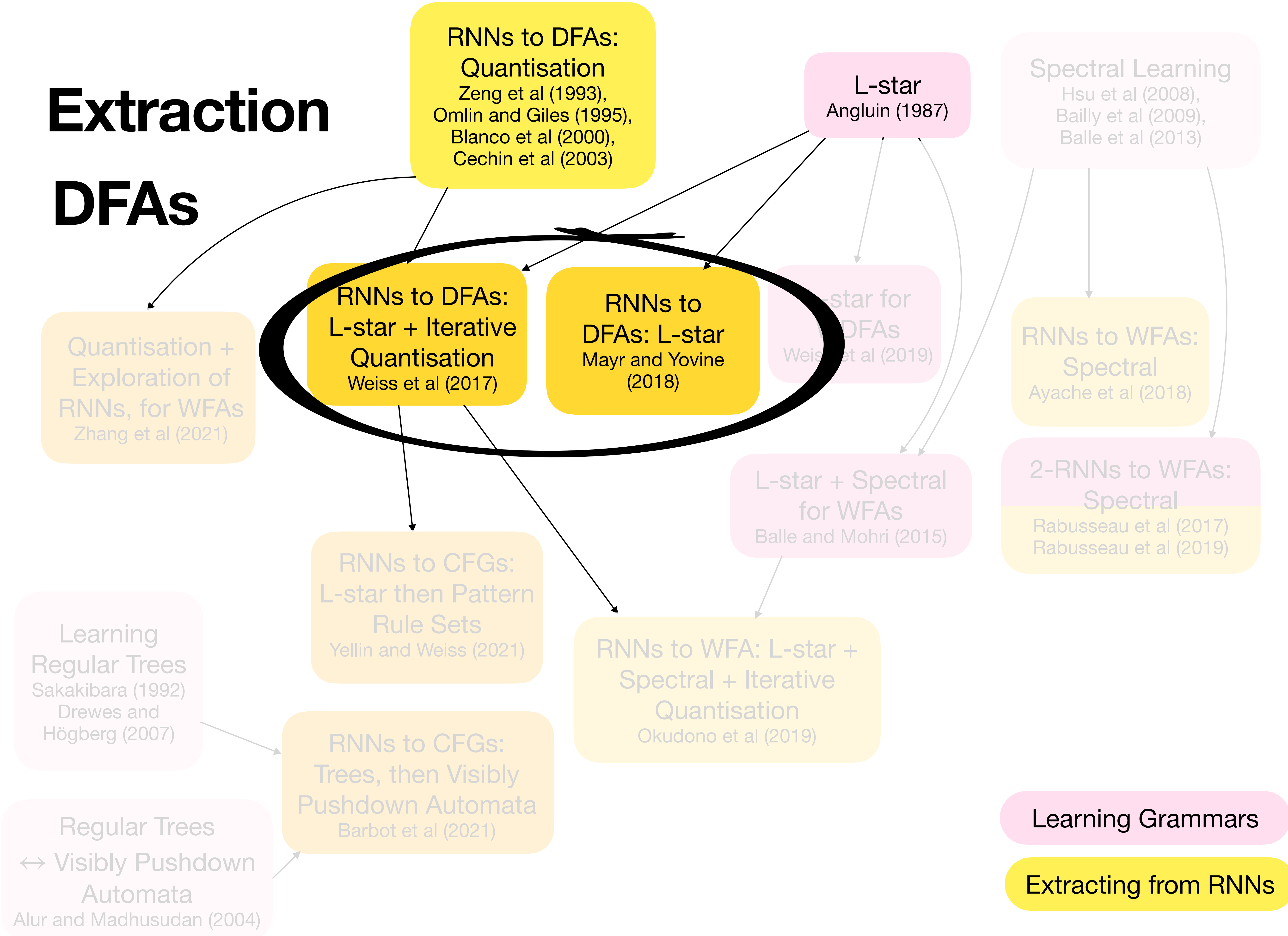
Membership Queries

Equivalence Queries

Extraction DFAs



Extraction DFAs



RNNs: Extracting DFAs: L-star

Apply L-star to an RNN, to learn a DFA representing/approximating it

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018

RNNs: Extracting DFAs: L-star

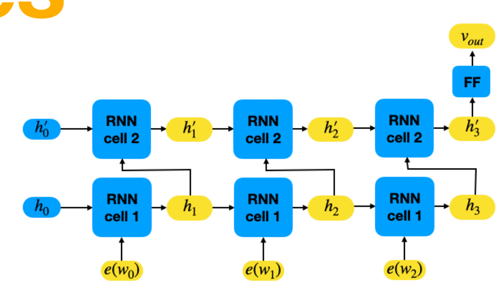
Apply L-star to an RNN, to learn a DFA representing/approximating it

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018

Membership Queries

bab?



✓ or ✗

RNNs: Extracting DFAs: L-star

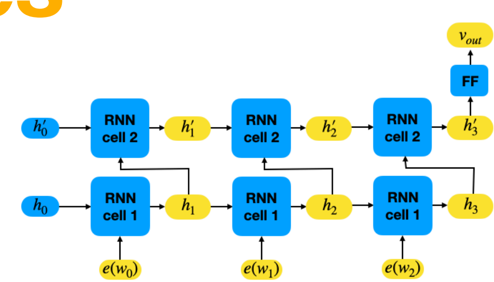
Apply L-star to an RNN, to learn a DFA representing/approximating it

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples
Weiss et al, 2017

Regular Inference on Artificial Neural Networks
Mayr and Yovine, 2018

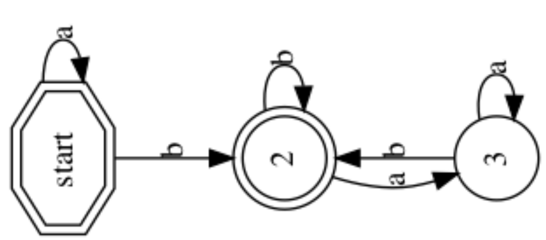
Membership Queries

bab? →

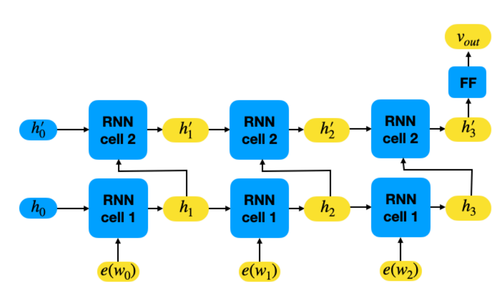


→ ✓ or ✗

Equivalence Queries

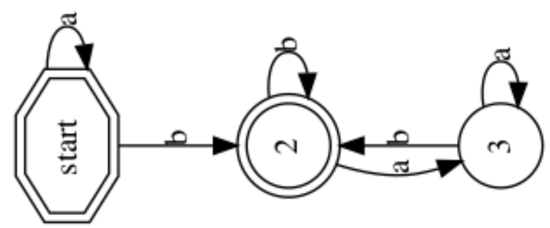


? ? ?

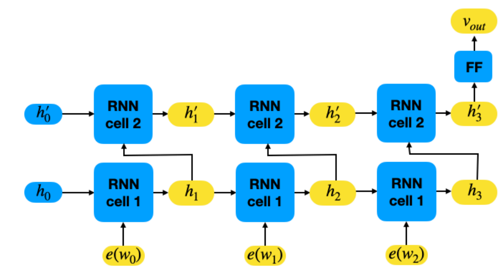


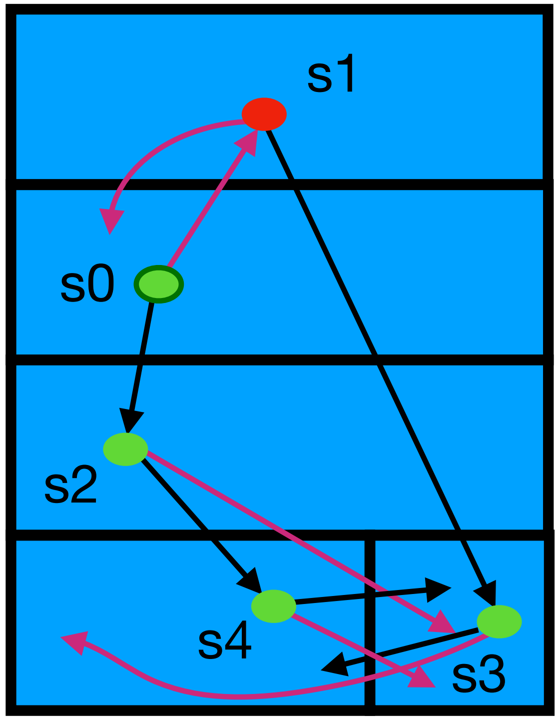
RNNs: Extracting DFAs: L-star

Equivalence Queries



???

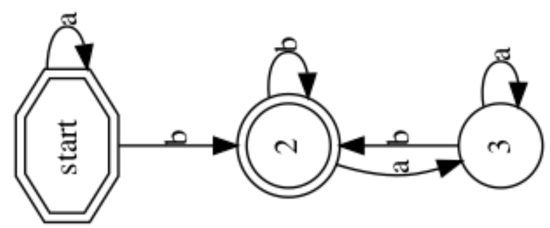




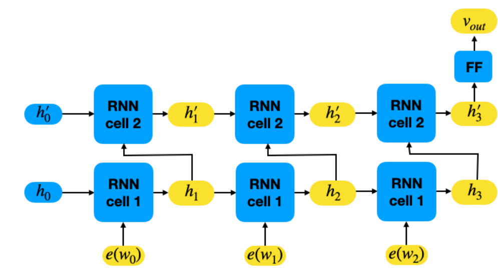
Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

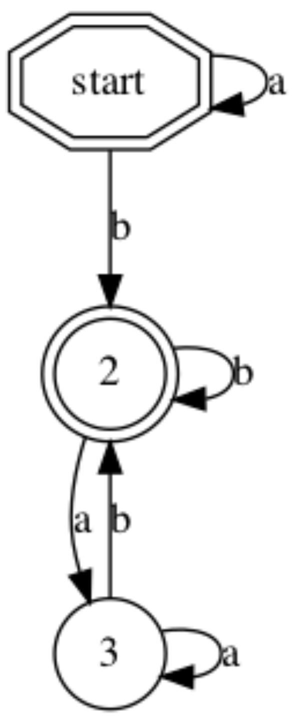
RNNs: Extracting DFAs: L-star

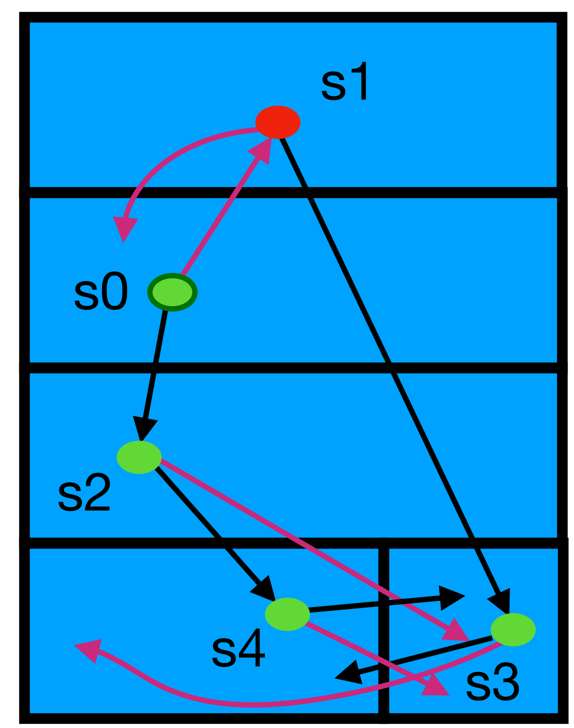
Equivalence Queries 

? ? ?



L* →



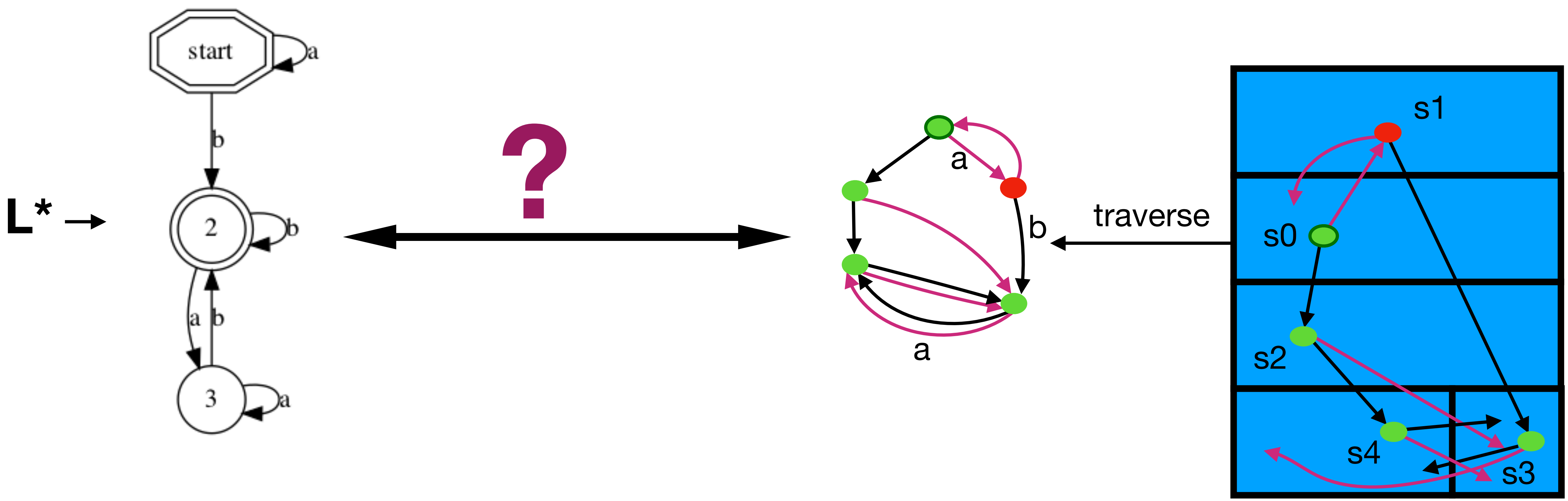


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

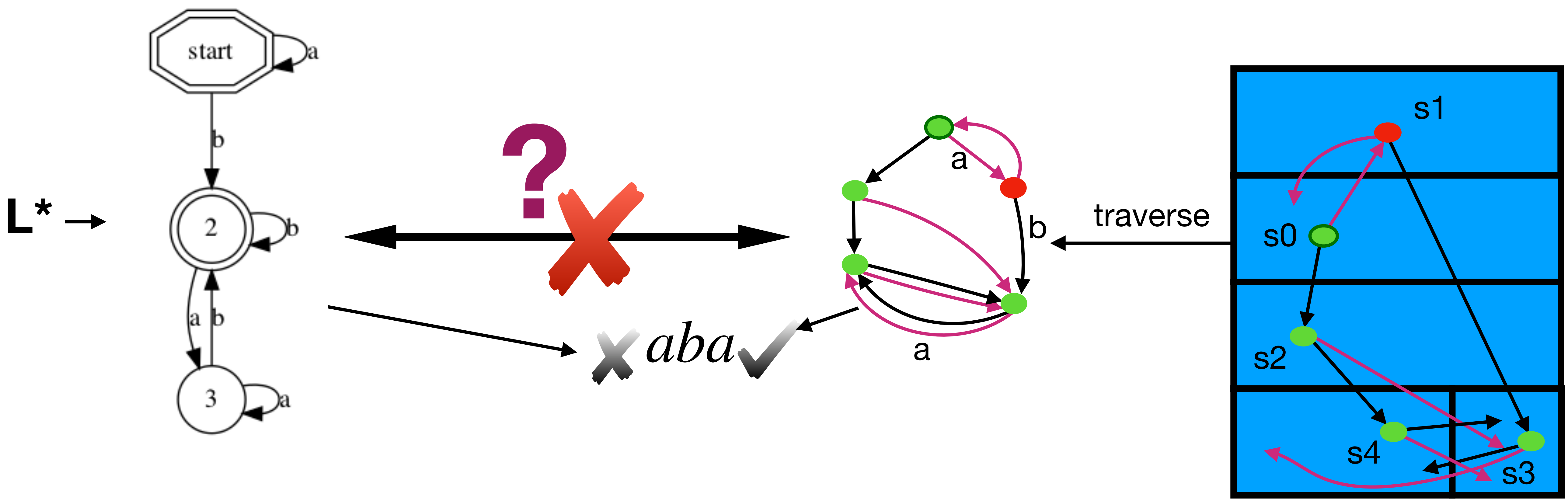


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

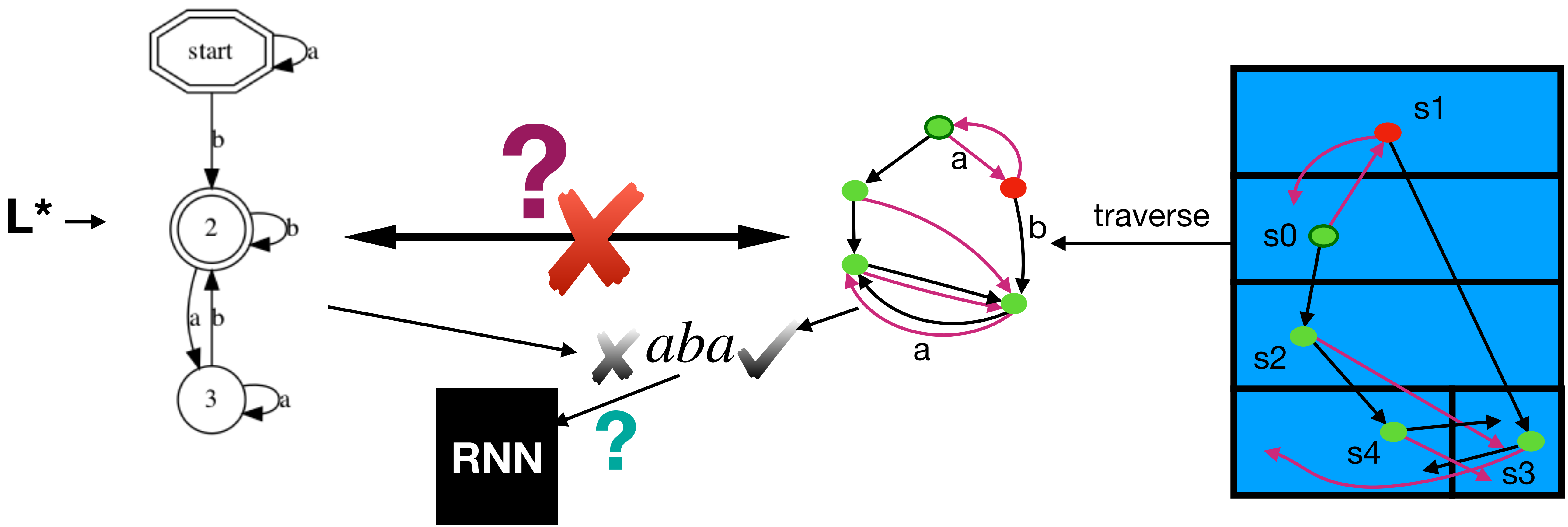


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

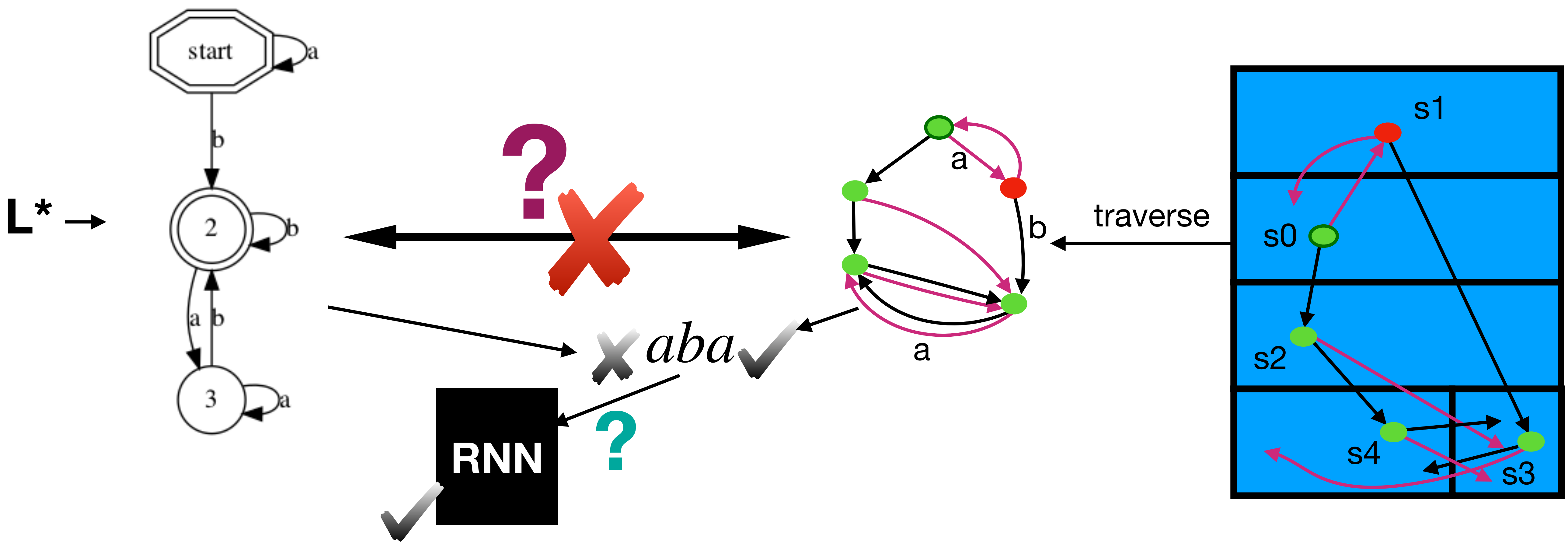


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

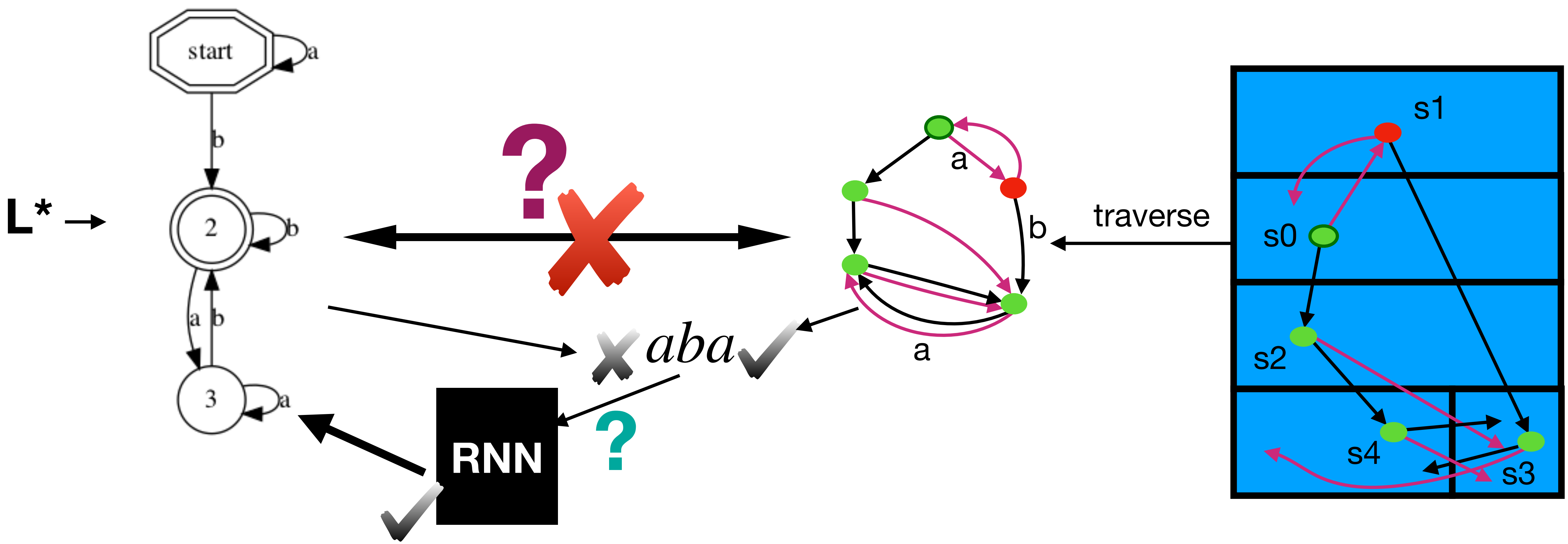


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

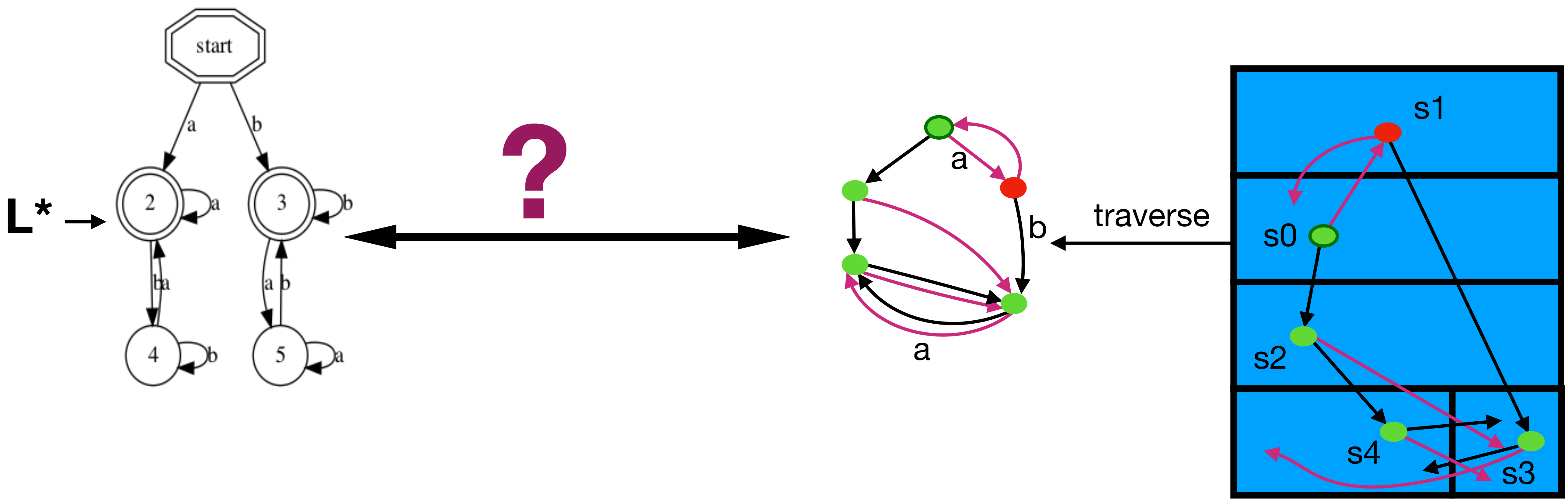


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

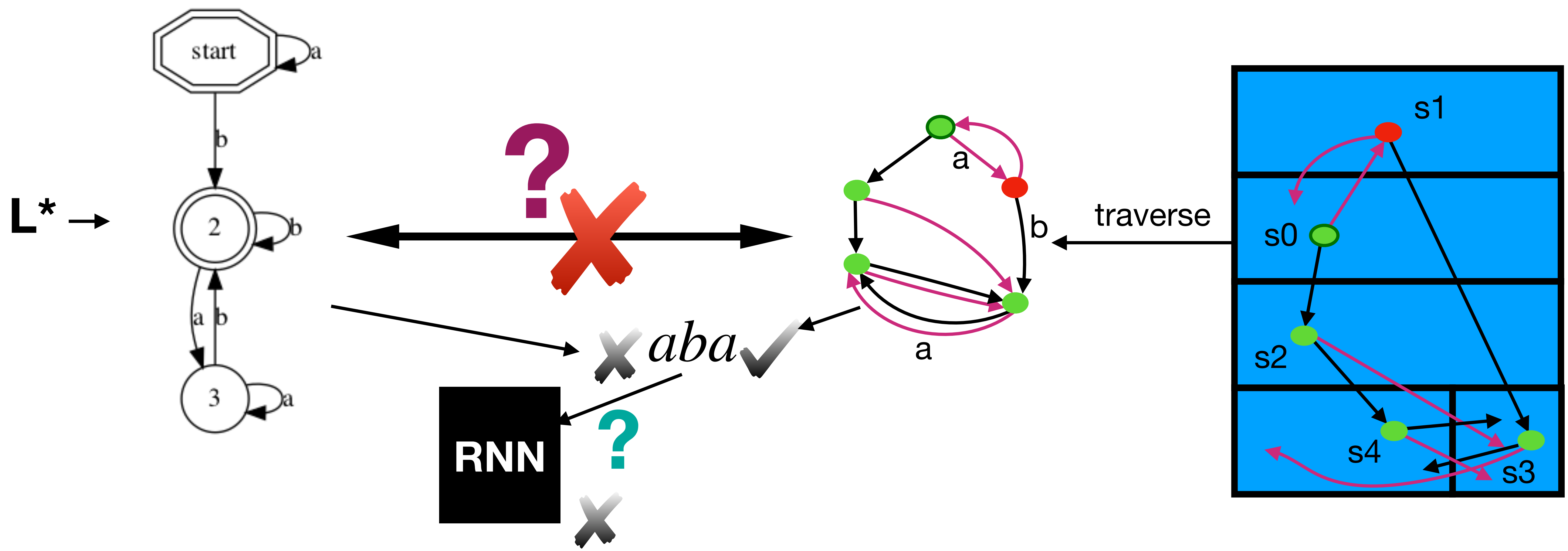


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

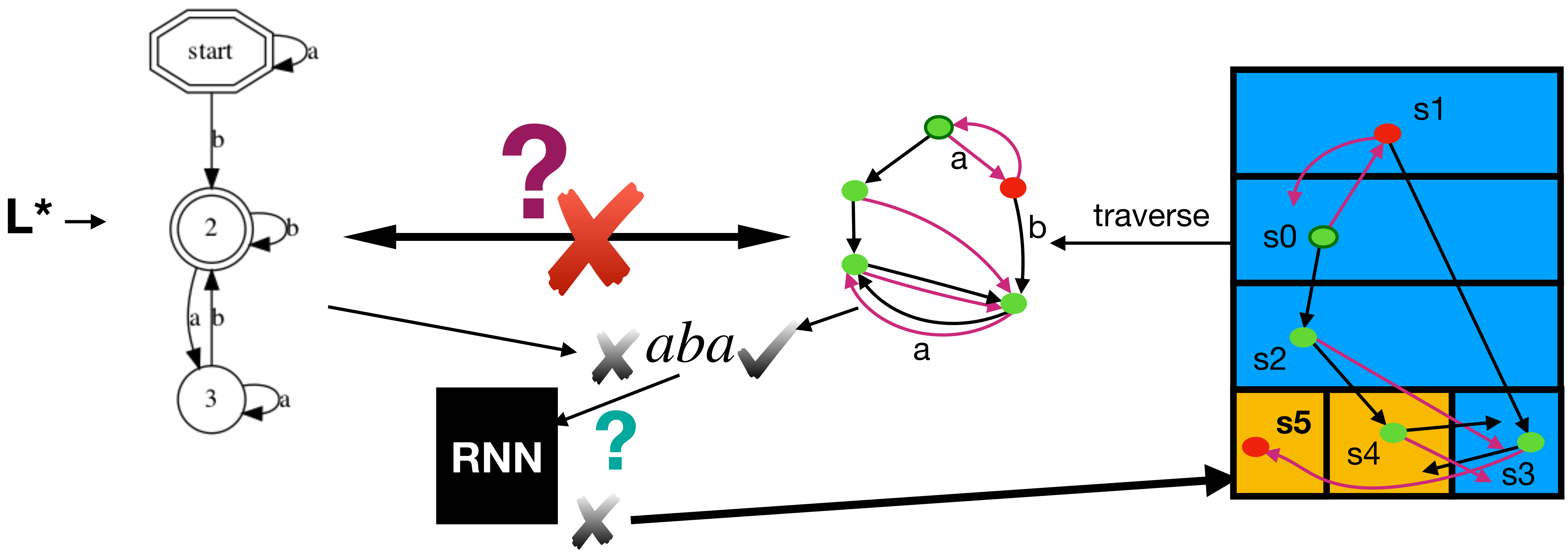


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

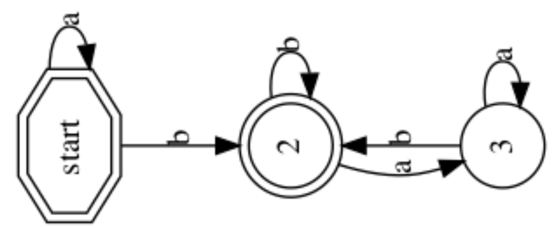


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

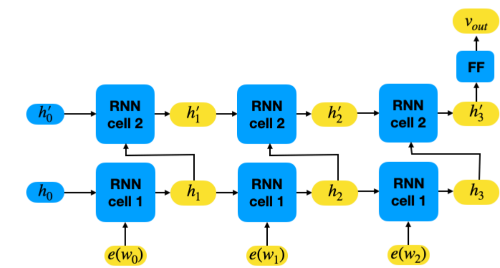
Weiss et al, 2017

RNNs: Extracting DFAs: L-star

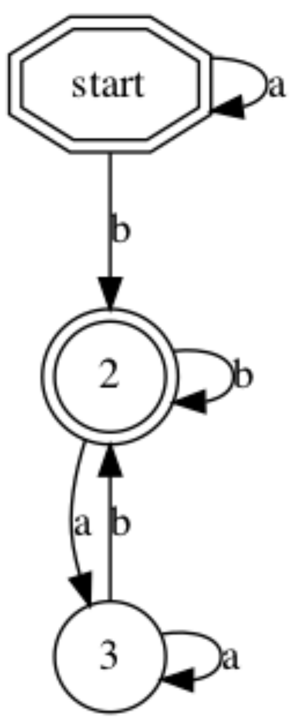
Equivalence Queries



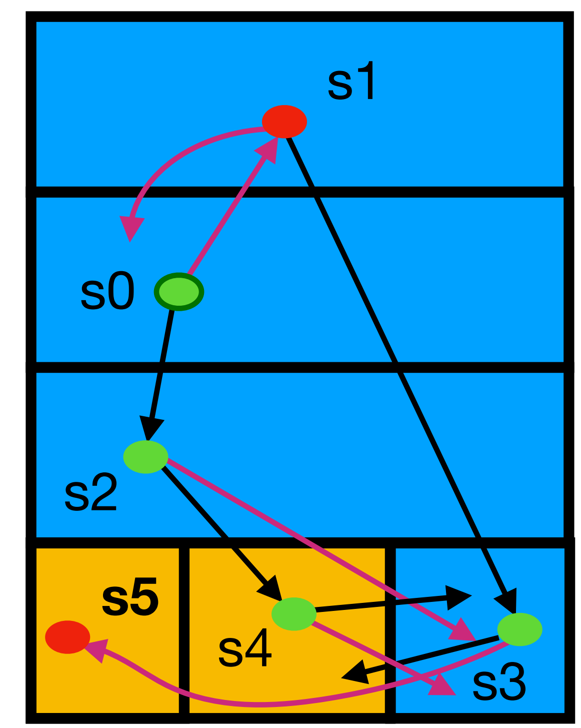
? ? ?



L* →



?

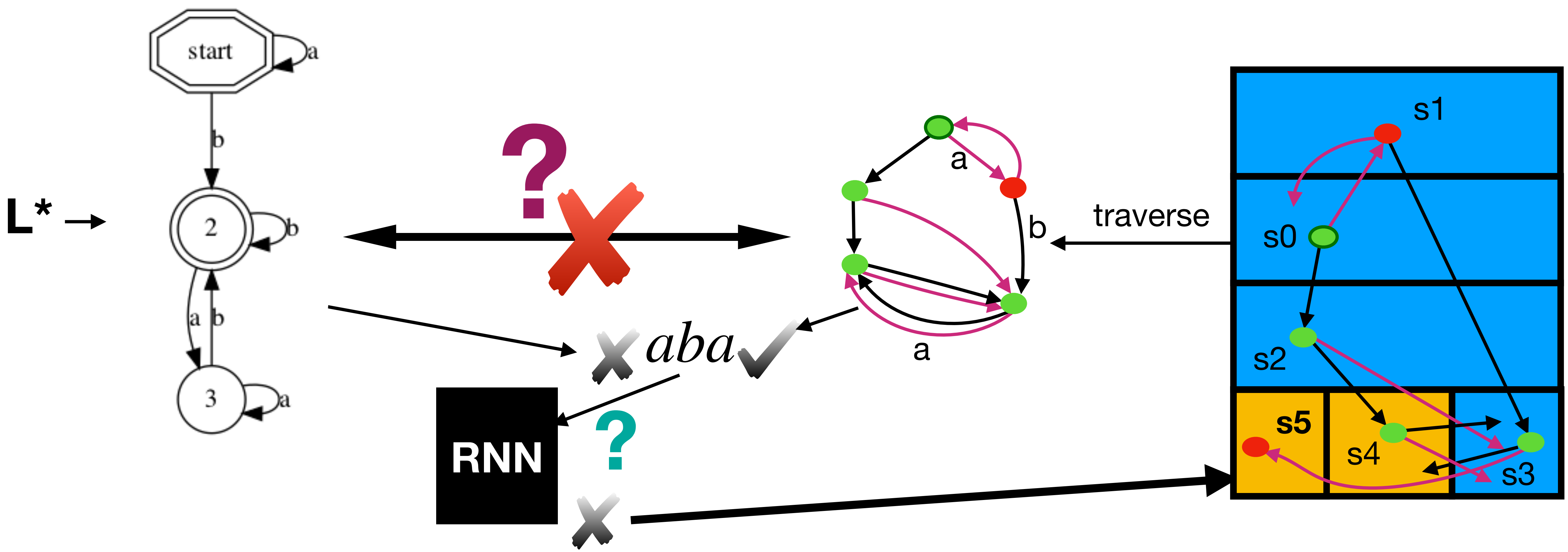


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

Weiss et al, 2017

RNNs: Extracting DFAs: L-star

Equivalence Queries

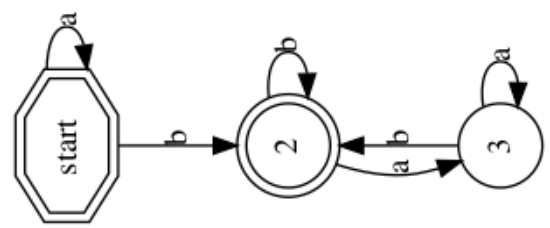


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

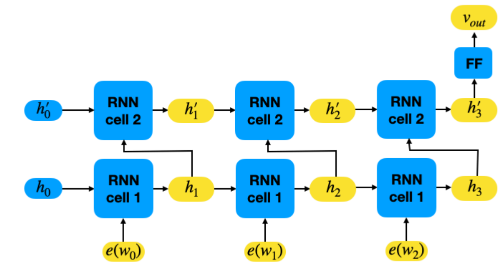
Weiss et al, 2017

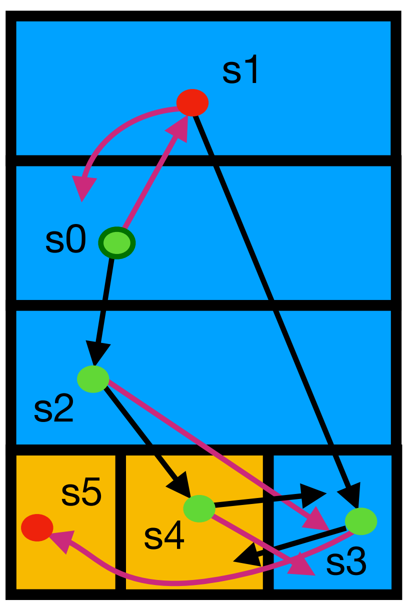
RNNs: Extracting DFAs: L-star

Equivalence Queries



? ? ?



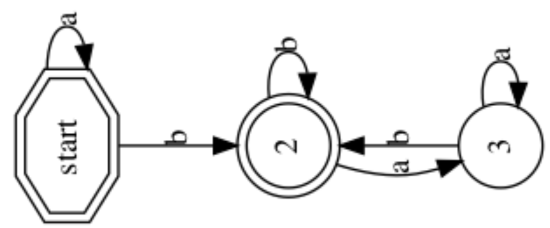


Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

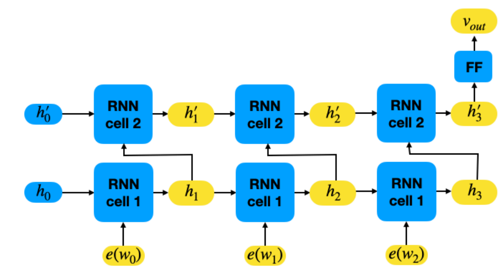
Weiss et al, 2017

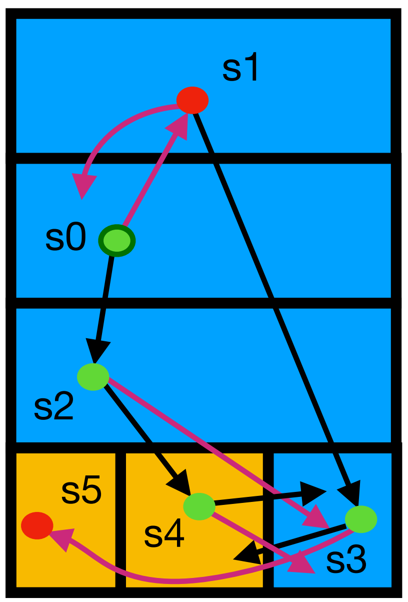
RNNs: Extracting DFAs: L-star

Equivalence Queries



? ? ?





Randomly Sample for Counterexamples

(Paper provides PAC analysis of this approach for equivalence queries)

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

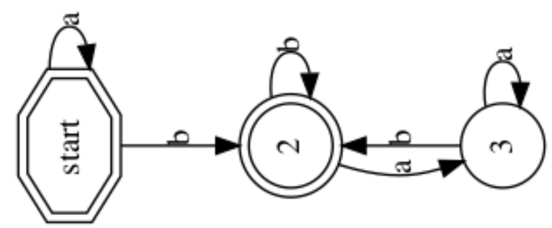
Weiss et al, 2017

Regular Inference on Artificial Neural Networks

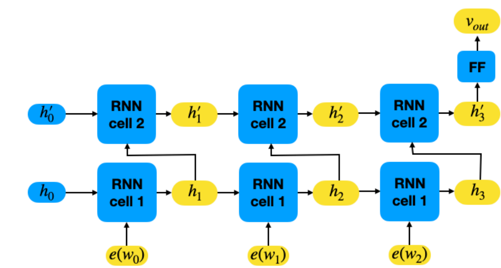
Mayr and Yovine, 2018

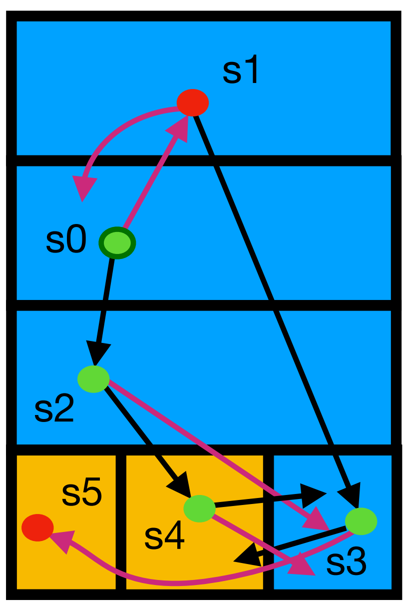
RNNs: Extracting DFAs: L-star

Equivalence Queries



? ? ?





Randomly Sample for Counterexamples

(Paper provides PAC analysis of this approach for equivalence queries)

Faster	**Slower**
Assumes white-box RNN	**Assumes black-box NN**
Complicated	**Simple**

Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples

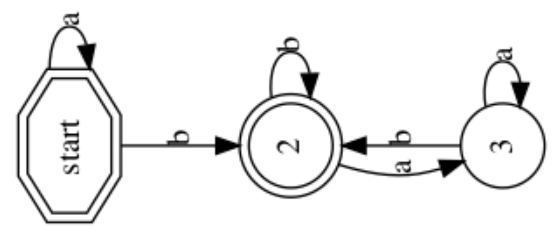
Weiss et al, 2017

Regular Inference on Artificial Neural Networks

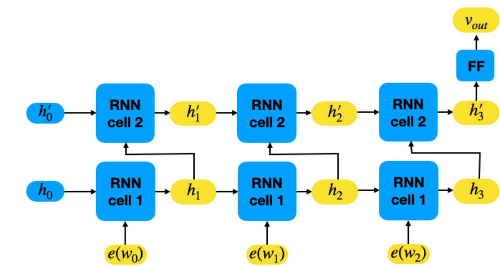
Mayr and Yovine, 2018

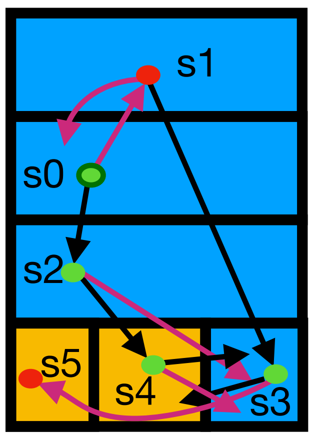
RNNs: Extracting DFAs: L-star

Equivalence Queries



? ? ?





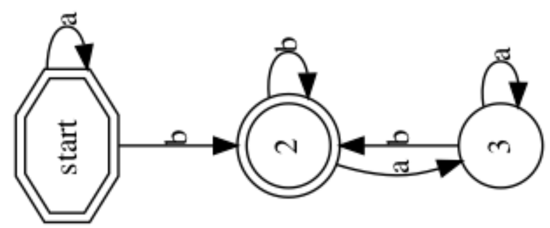
Randomly Sample for Counterexamples

Faster

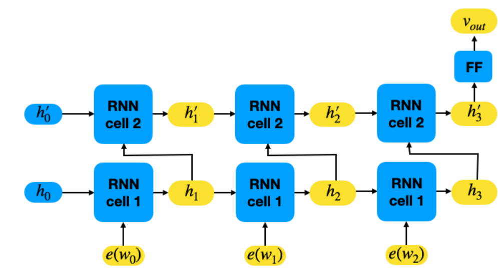
Slower

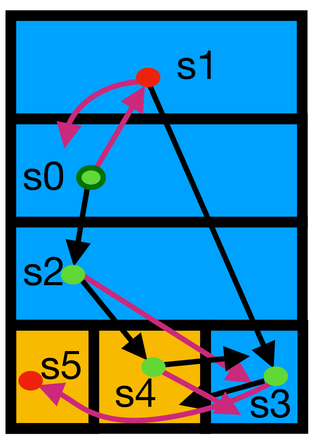
RNNs: Extracting DFAs: L-star

Equivalence Queries



? ? ?





Randomly Sample for Counterexamples

Faster

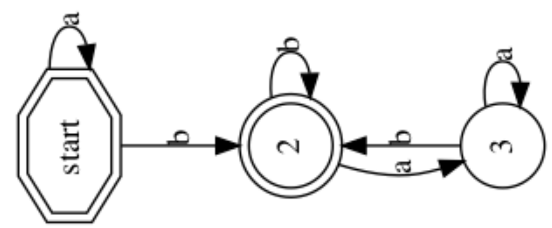
Slower

Learning Balanced Parentheses over $\Sigma = \{(,), a - z\}$

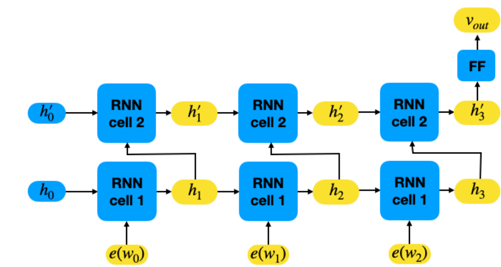
e.g. $()$, $()a()b$, $abc(()(a))$, etc

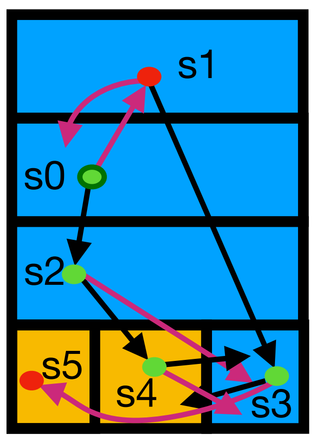
RNNs: Extracting DFAs: L-star

Equivalence Queries



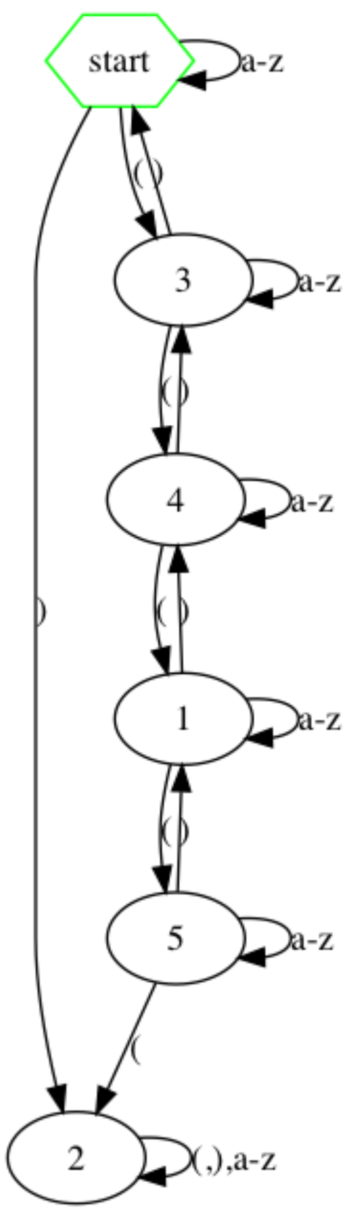
? ? ?





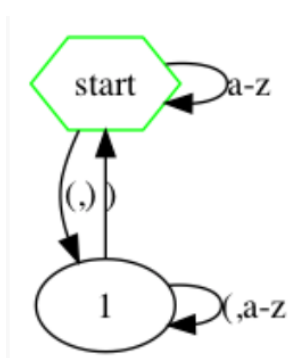
Randomly Sample for Counterexamples

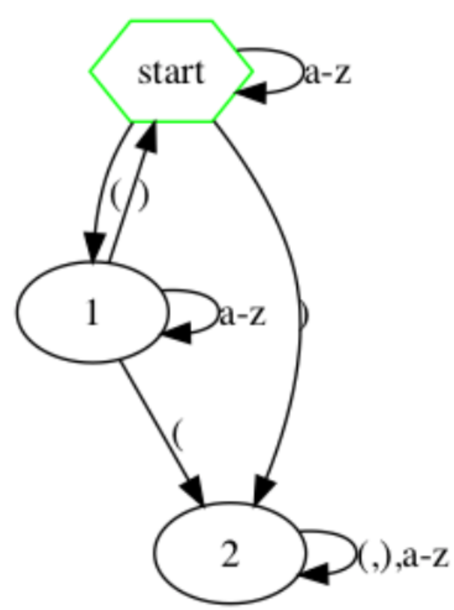
Faster **Slower**



Learning Balanced Parentheses over $\Sigma = \{(,),a-z\}$

e.g. (), $()a()b$, $abc(()(a))$, etc

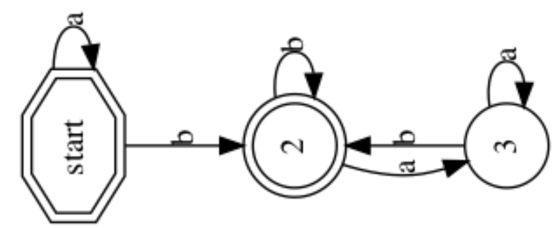




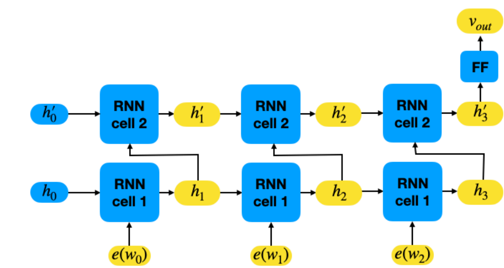
…

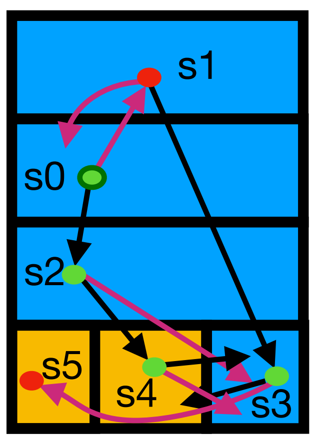
RNNs: Extracting DFAs: L-star

Equivalence Queries



? ? ?





Randomly Sample for Counterexamples

Faster **Slower**

Learning Balanced Parentheses over $\Sigma = \{(,), a - z\}$

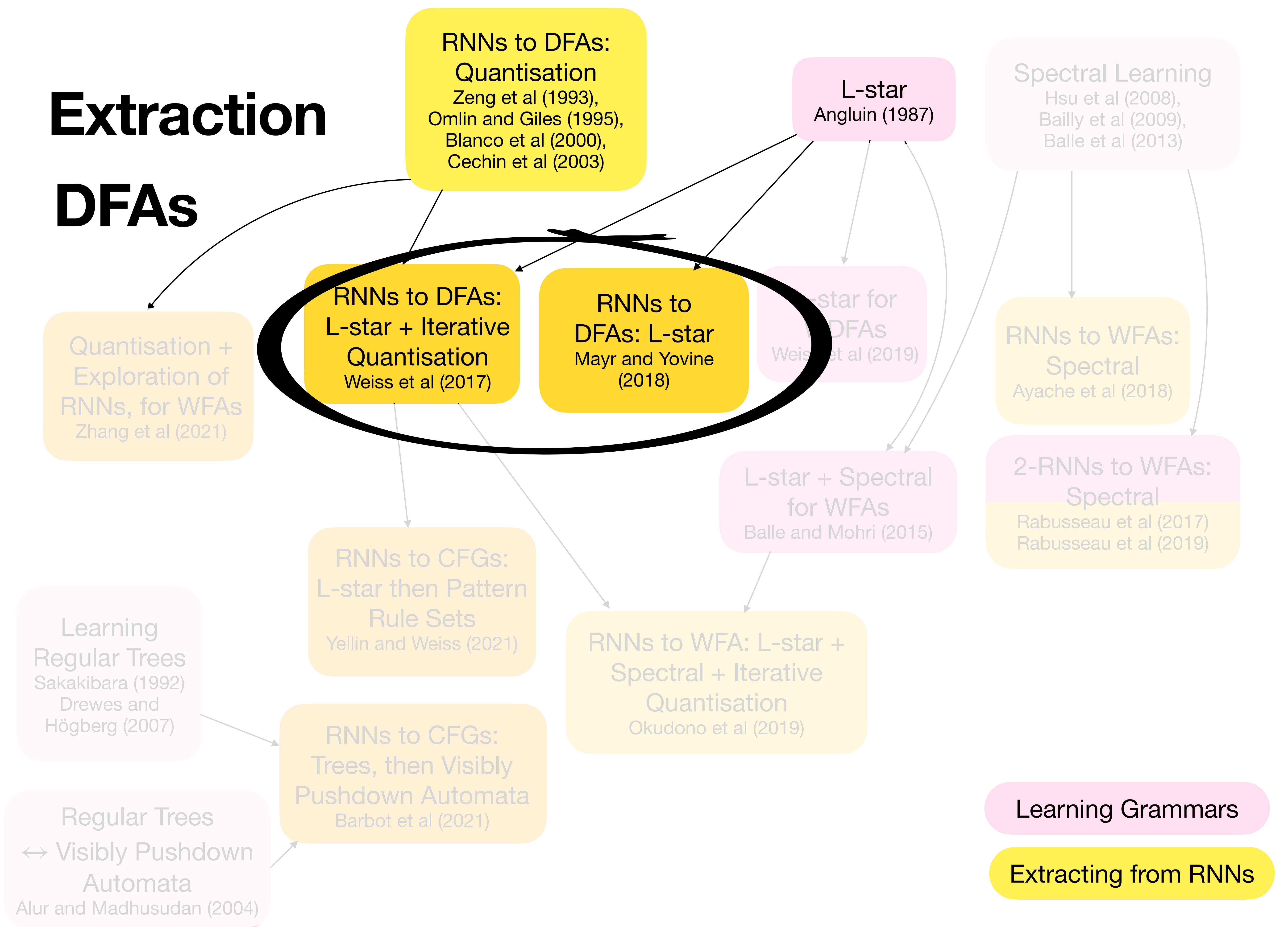
e.g. $()$, $()a()b$, $abc(()(a))$, etc

Random sampling counterexamples:

))	(1.5s)
tg(gu()uh)	(57.5s)
((wviw(iac)r)mrsnqqb)iew	(231.5s)

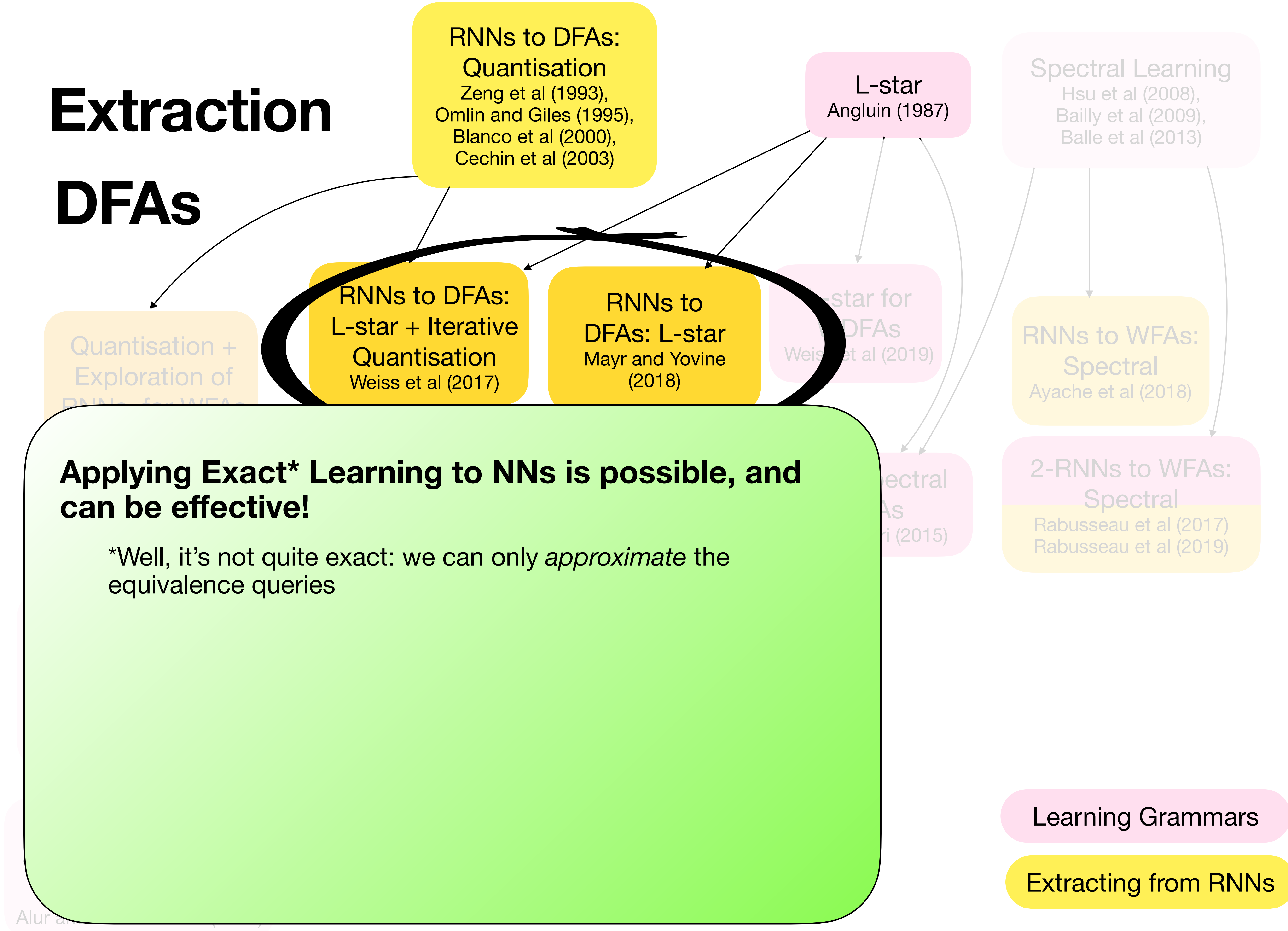
Abstraction based counterexamples:

))	(1.4s)
(())	(1.6s)
((()))	(3.1s)
(((())))	(3.1s)
((((()))))	(3.4s)
(((((())))))	(4.7s)
((((((()))))))	(6.3s)
(((((((())))))))	(9.2s)
((((((((()))))))))	(14.0s)



Extraction
DFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
star for
DFAs
Weis
et al (2019)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to WFAs: Spectral
Ayache et al (2018)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
L-star + Spectral for WFAs
Balle and Mohri (2015)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

Extraction DFAs

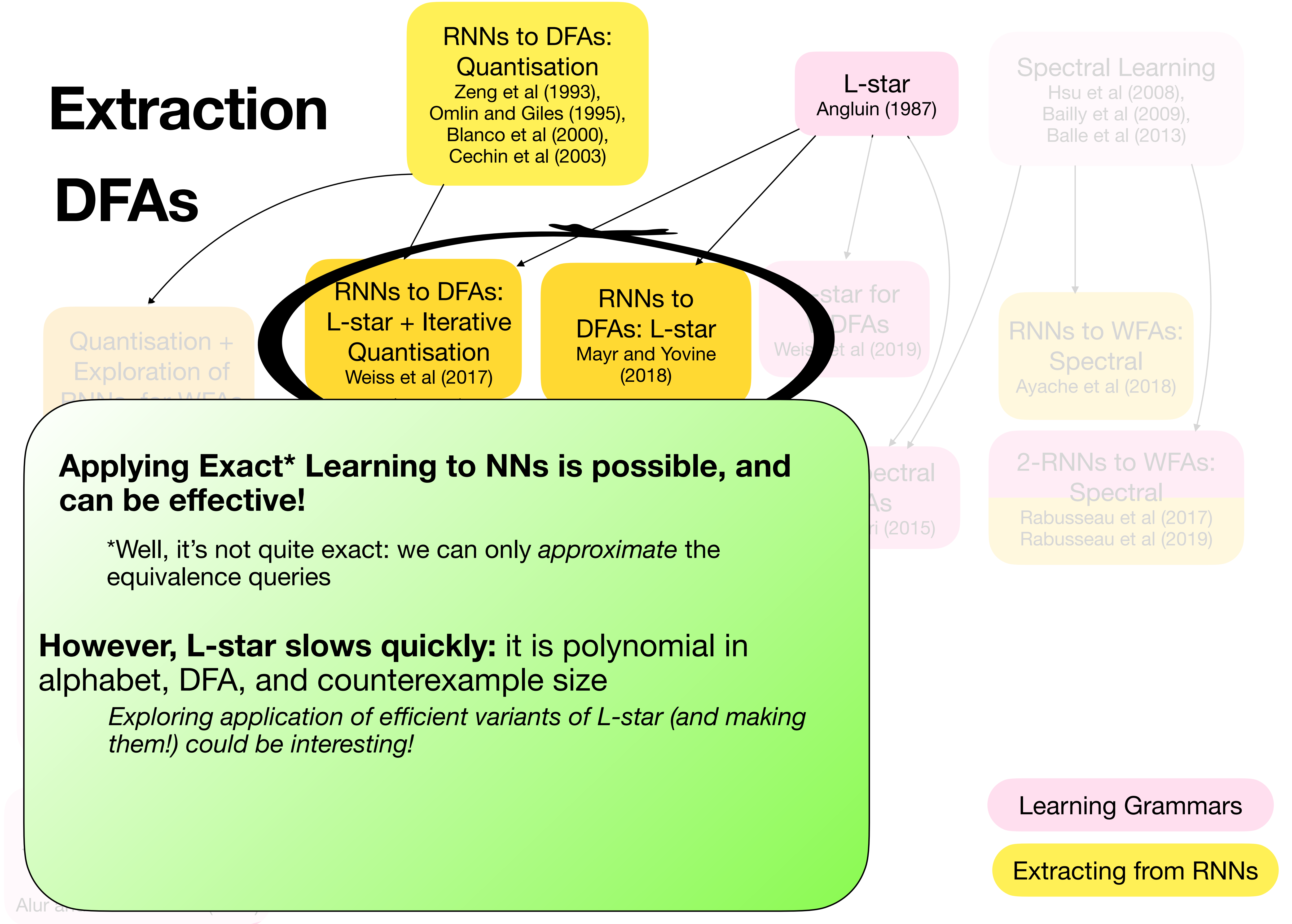


Applying Exact* Learning to NNs is possible, and can be effective!

*Well, it's not quite exact: we can only *approximate* the equivalence queries

Extraction

DFAs



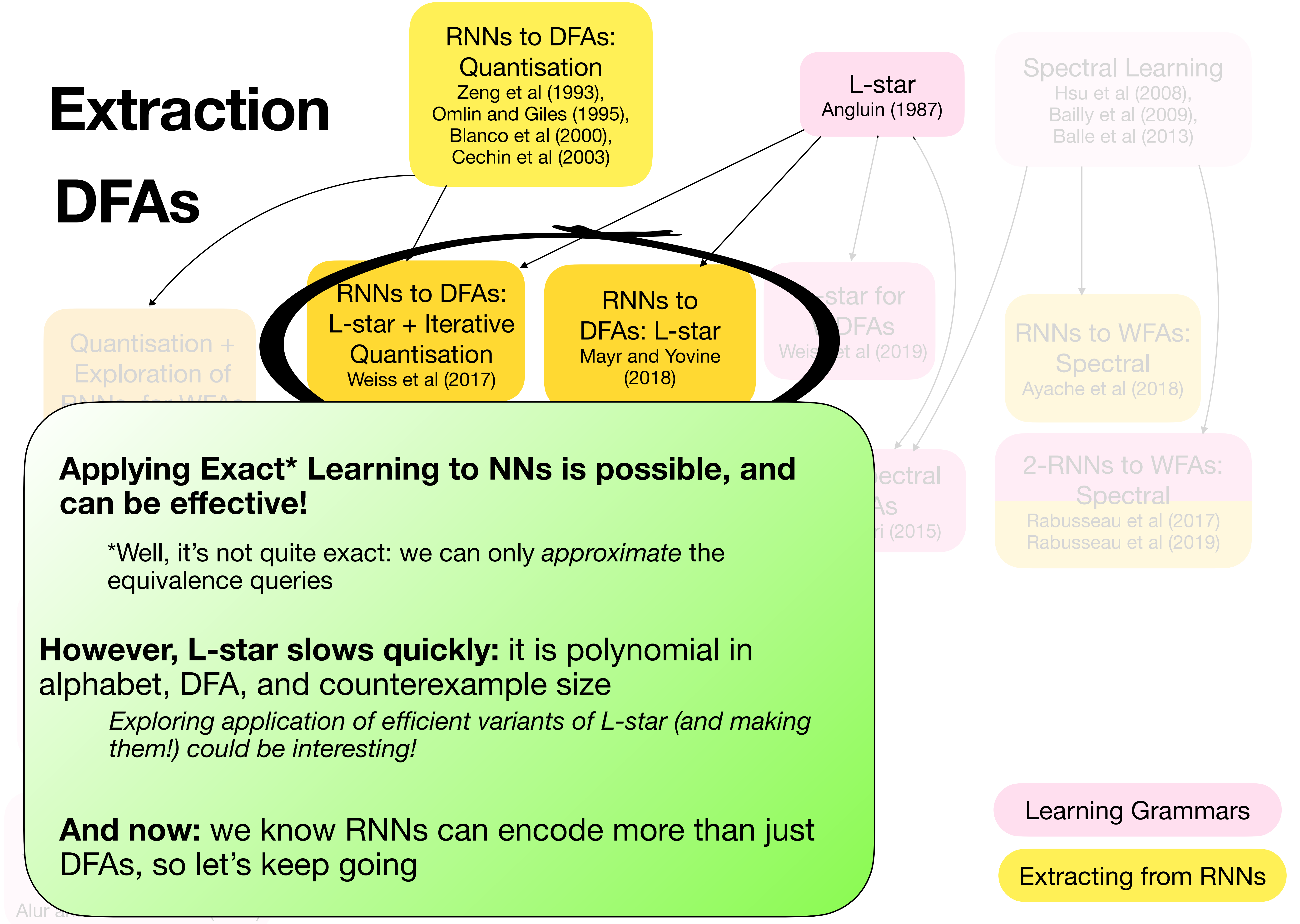
Applying Exact* Learning to NNs is possible, and can be effective!

*Well, it's not quite exact: we can only *approximate* the equivalence queries

However, L-star slows quickly: it is polynomial in alphabet, DFA, and counterexample size

Exploring application of efficient variants of L-star (and making them!) could be interesting!

Extraction DFAs



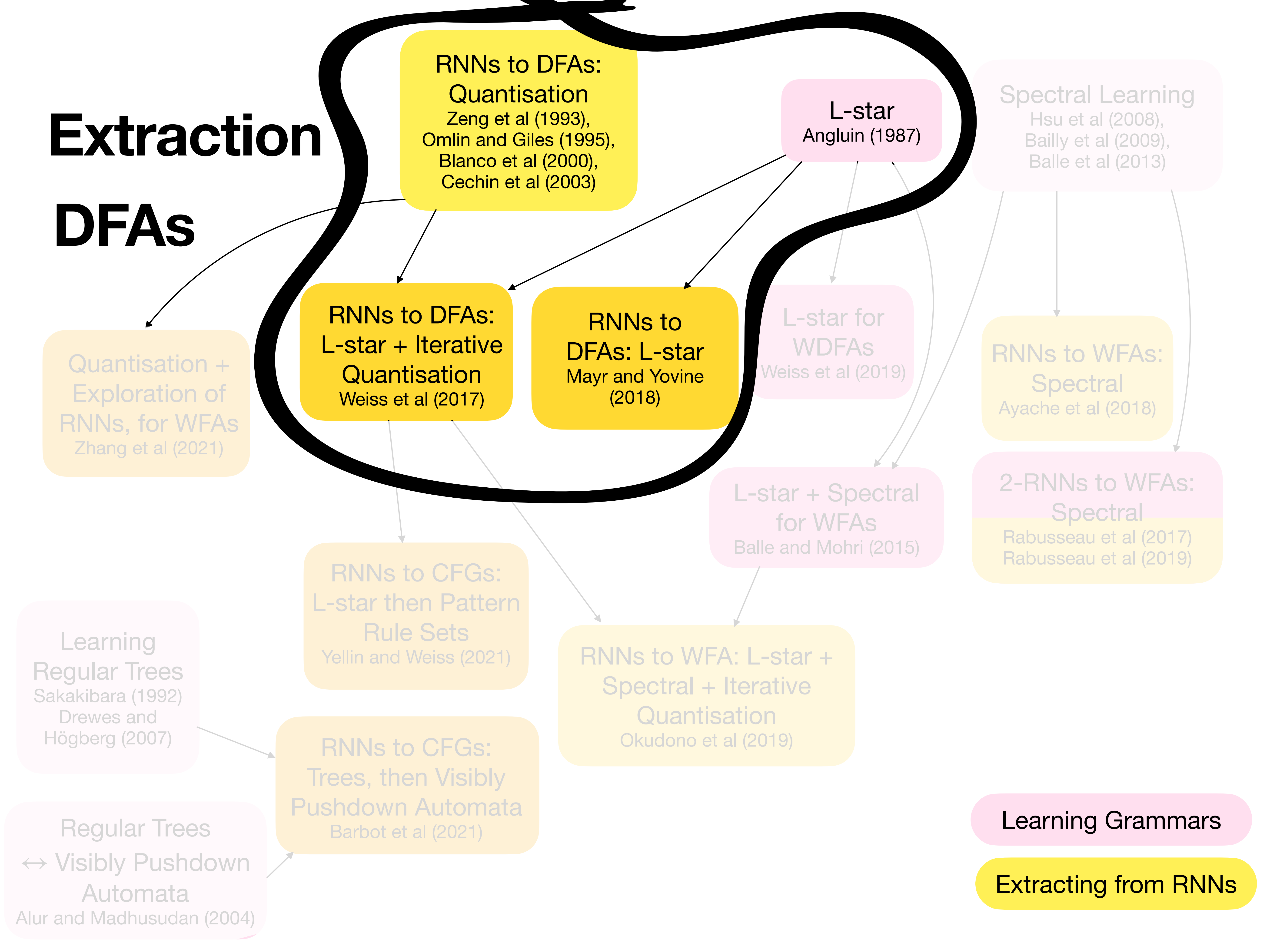
Applying Exact* Learning to NNs is possible, and can be effective!

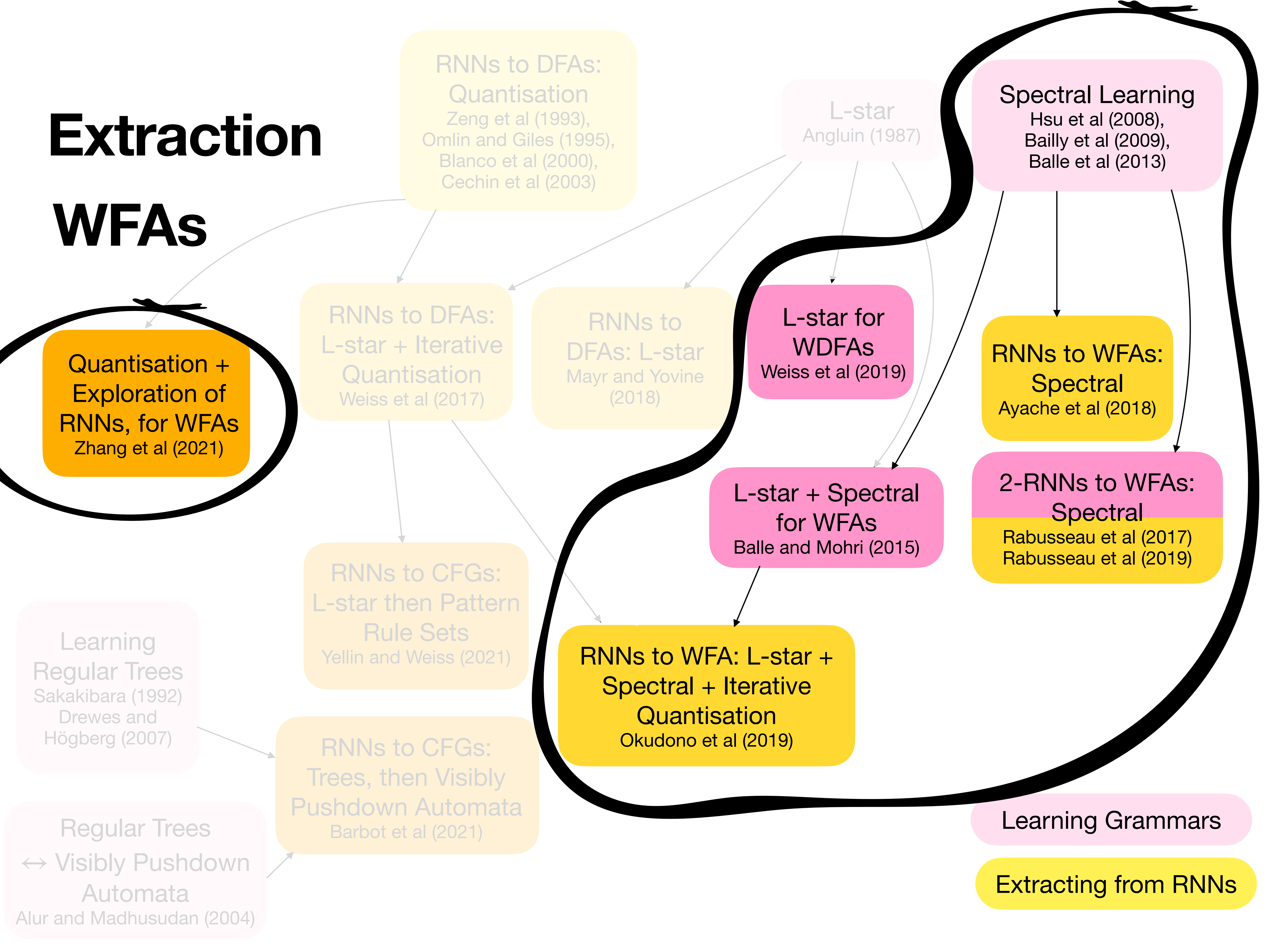
*Well, it's not quite exact: we can only *approximate* the equivalence queries

However, L-star slows quickly: it is polynomial in alphabet, DFA, and counterexample size

Exploring application of efficient variants of L-star (and making them!) could be interesting!

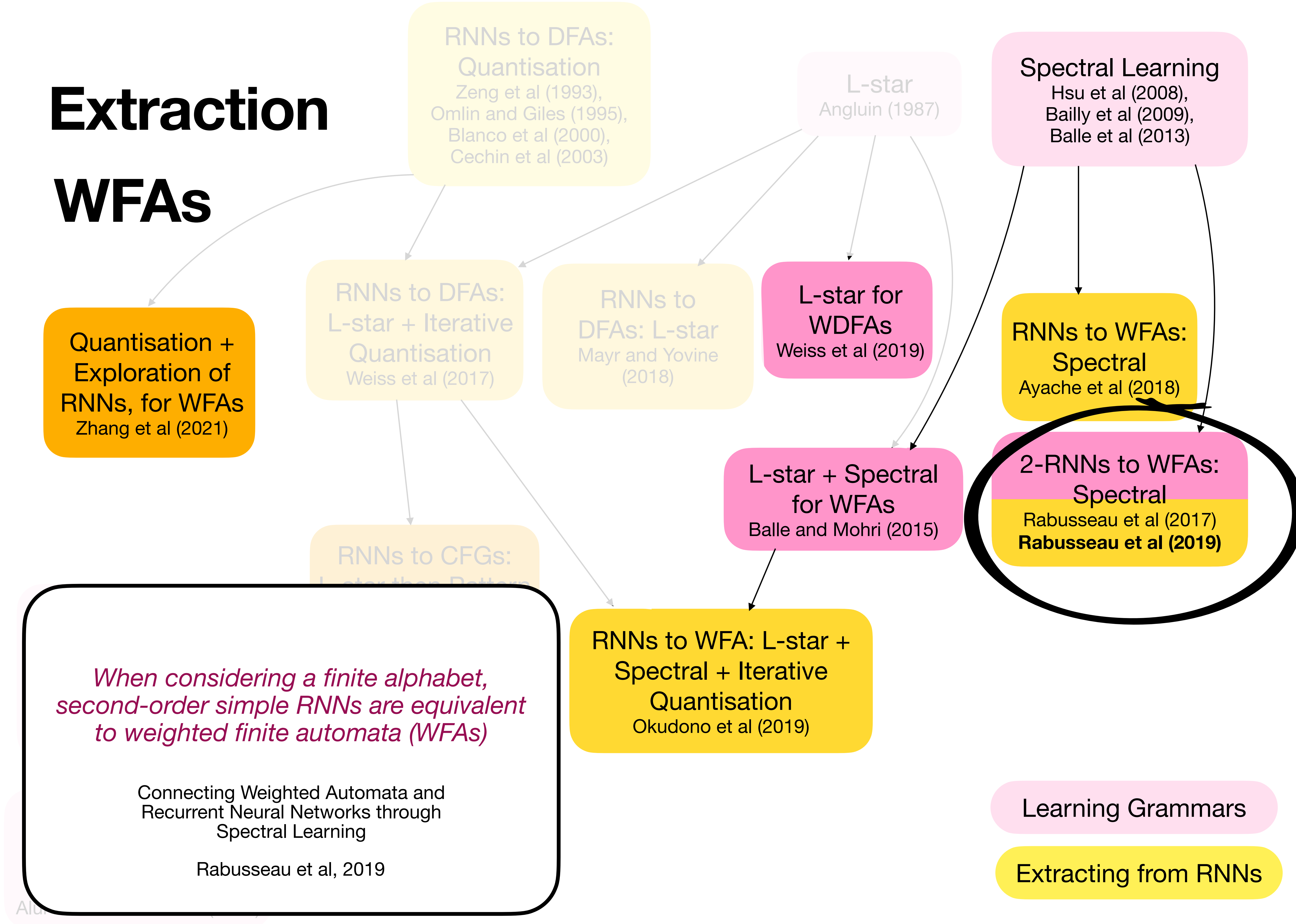
And now: we know RNNs can encode more than just DFAs, so let's keep going


Extraction
DFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs



Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
L-star + Spectral for WFAs
Balle and Mohri (2015)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

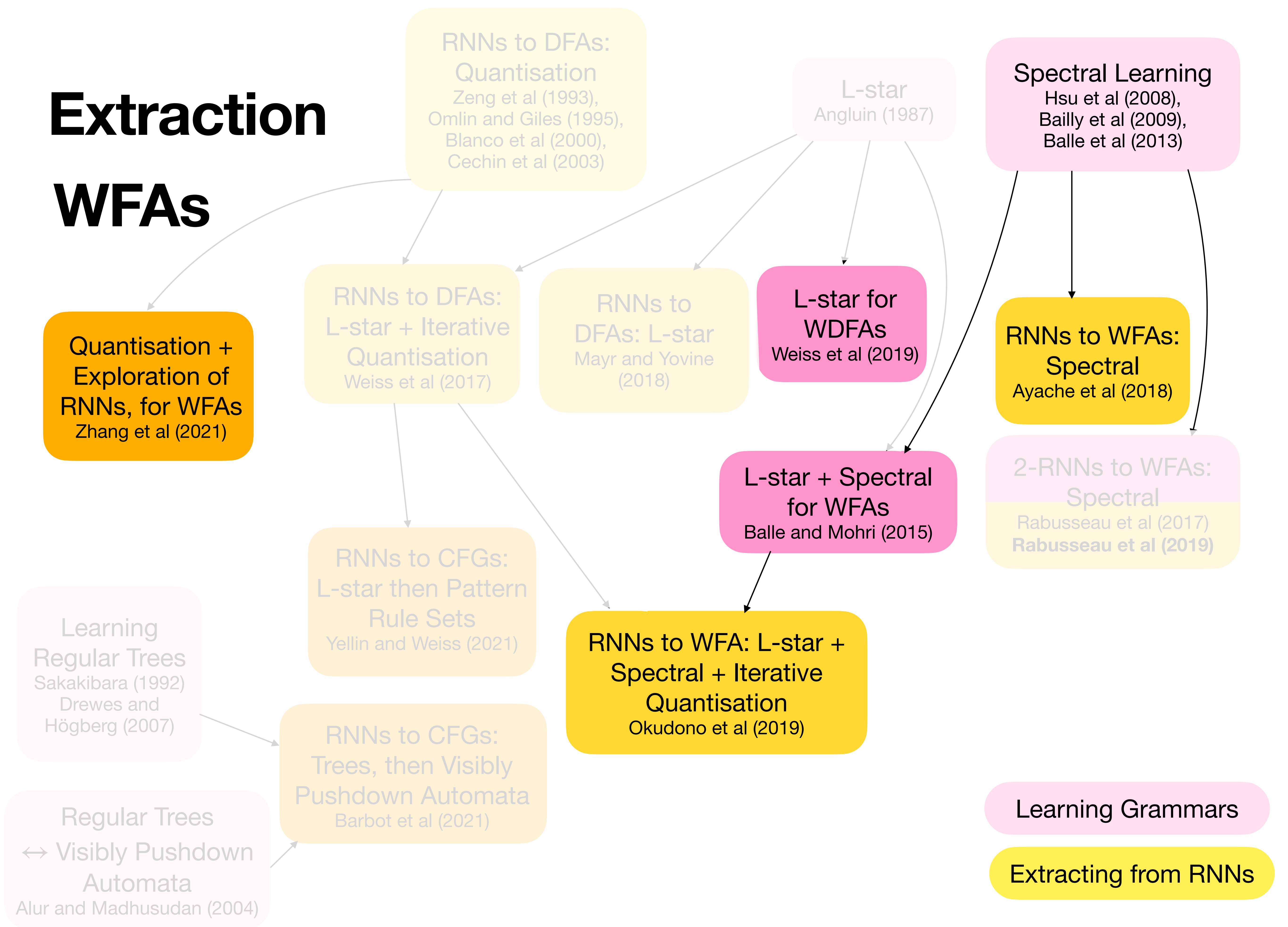
Extraction
WFAs



When considering a finite alphabet, second-order simple RNNs are equivalent to weighted finite automata (WFAs)

Connecting Weighted Automata and Recurrent Neural Networks through Spectral Learning

Rabusseau et al, 2019



Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

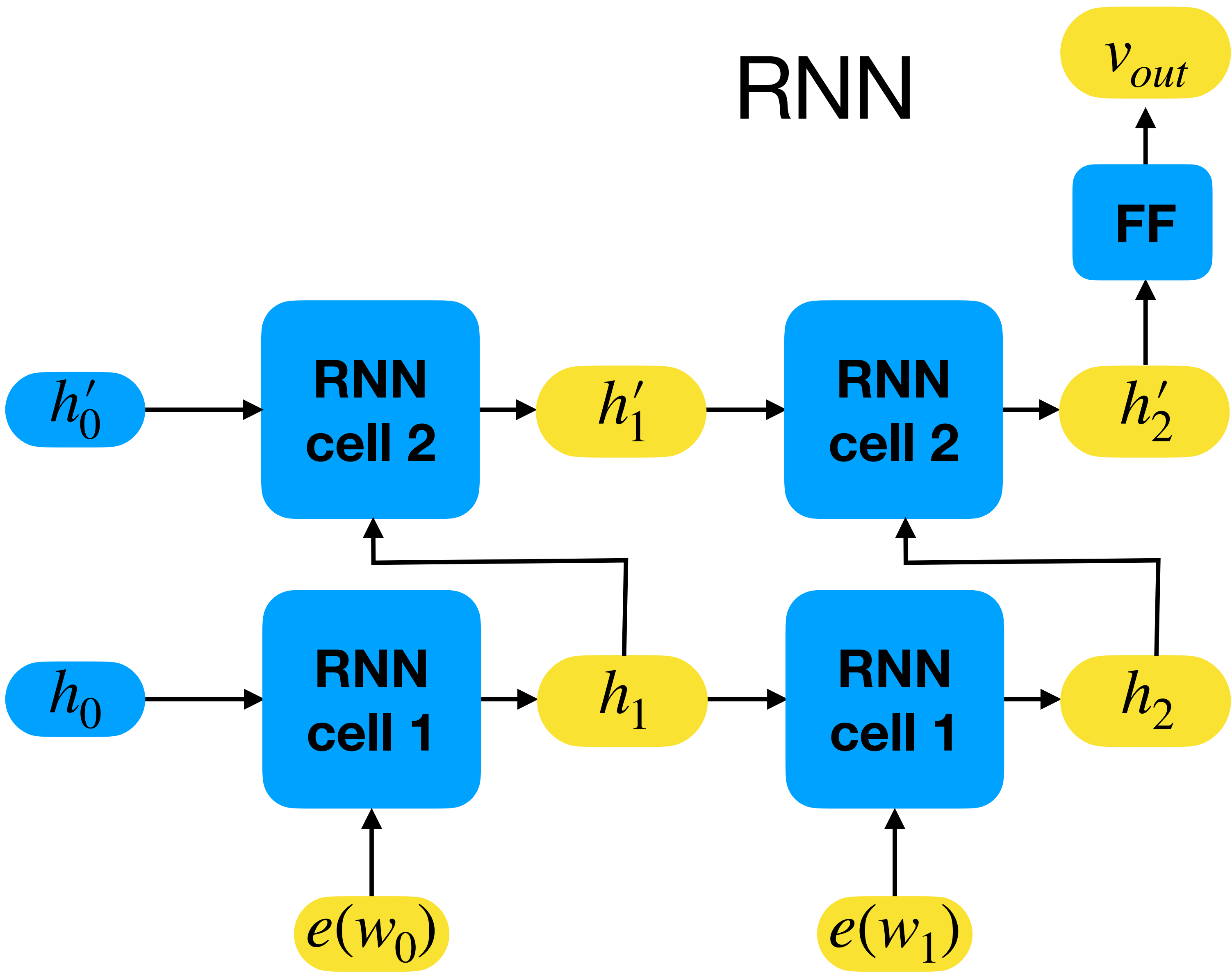
RNNs: Extracting WFAs: Background!

- Language-Model RNNs
- WFAs
  - Matrix Representation

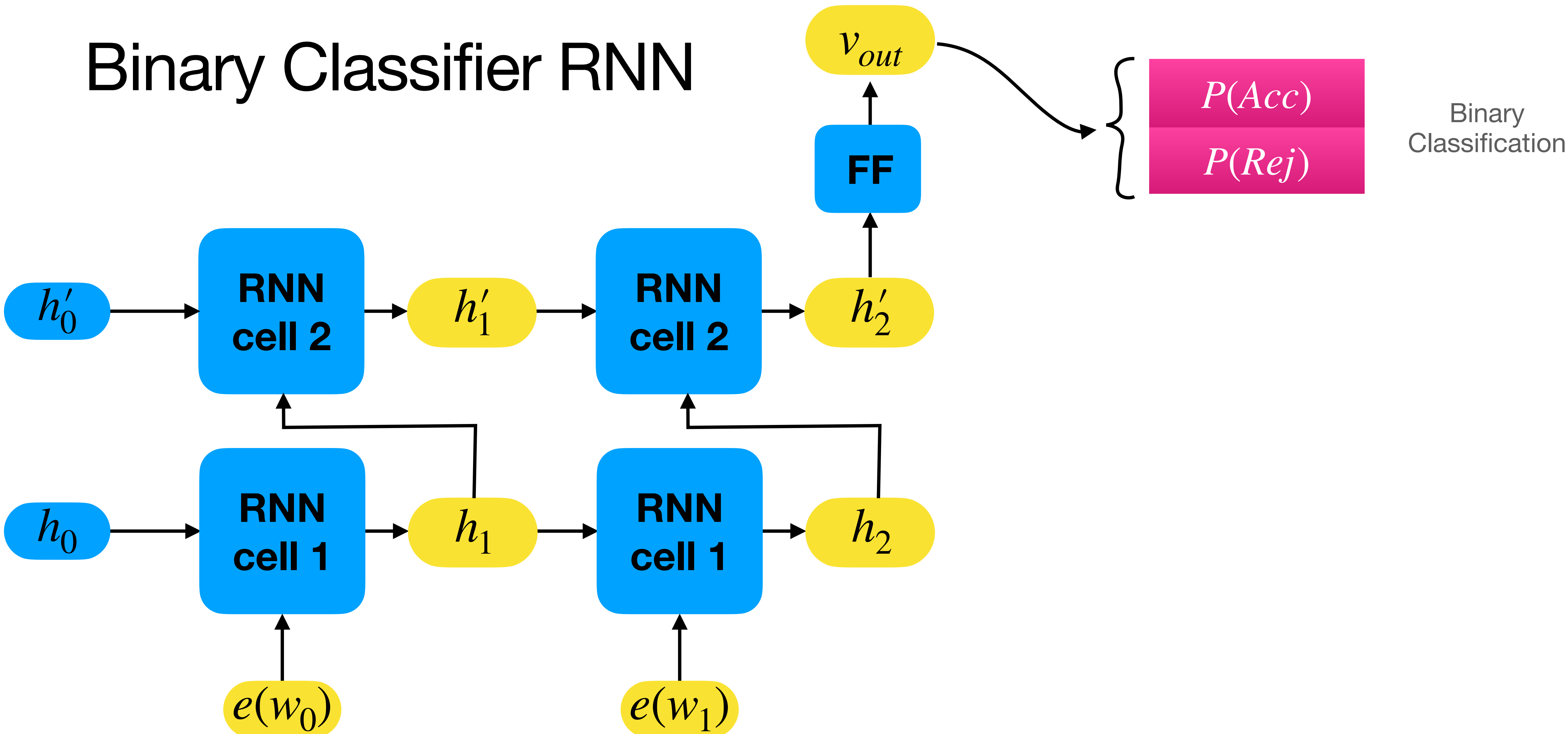
RNNs: Extracting WFAs: Background!

- Language-Model RNNs
- WFAs
  - Matrix Representation

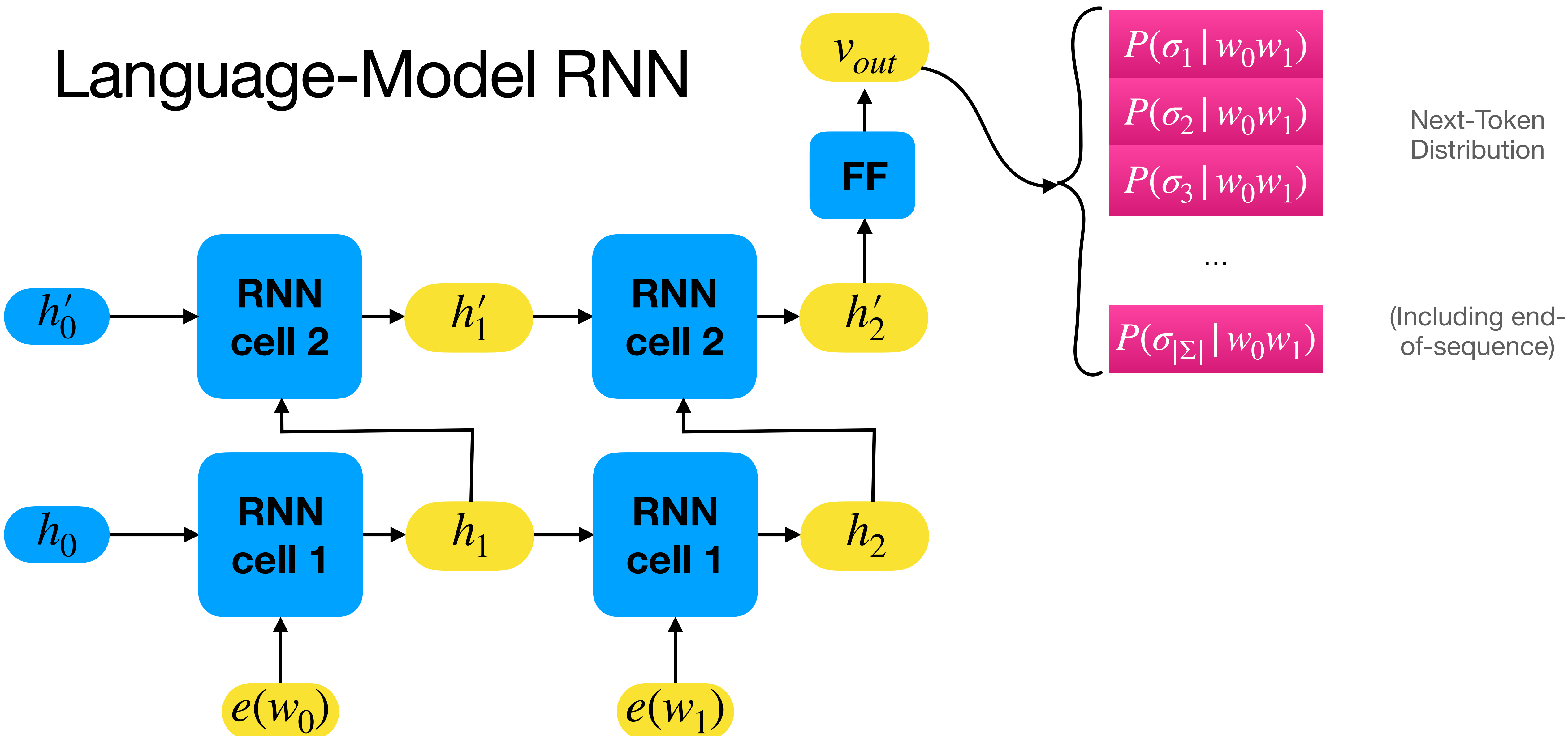
RNNs: Extracting WFAs: Background!



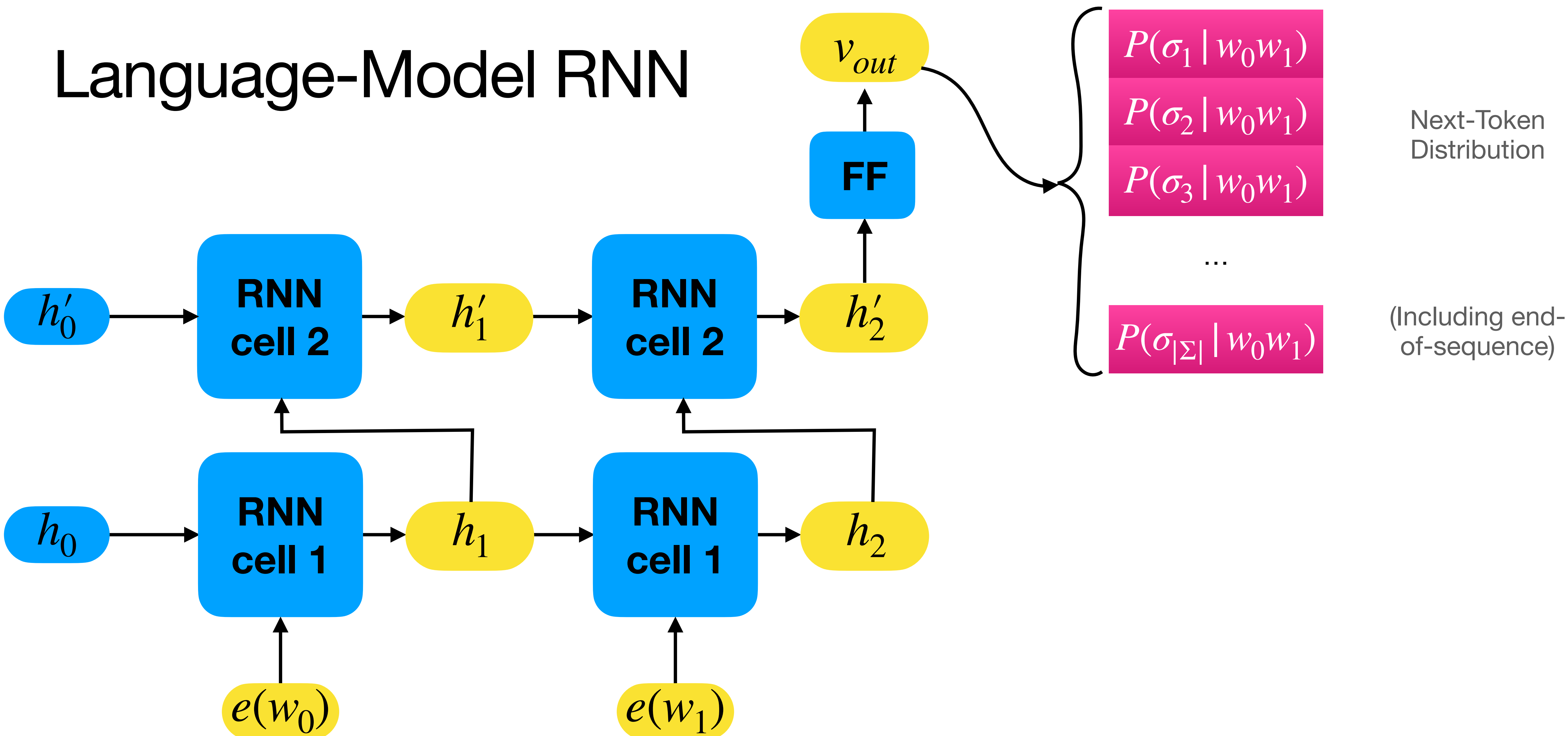
RNNs: Extracting WFAs: Background!



RNNs: Extracting WFAs: Background!



RNNs: Extracting WFAs: Background!

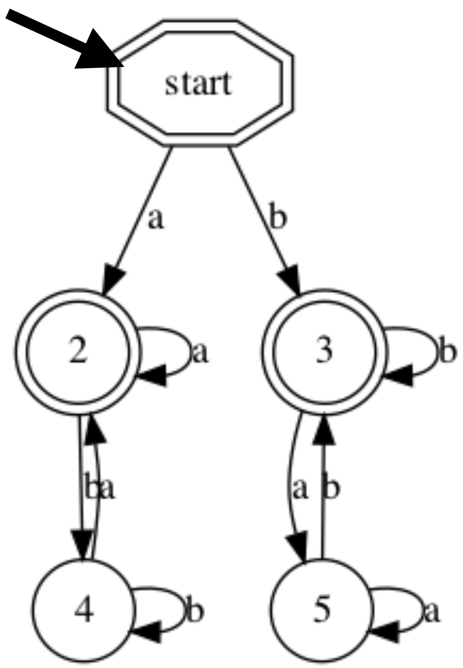


$$\text{RNN}(w_1 w_2) = P(w_1 | \varepsilon) \cdot P(w_2 | w_1) \cdot P(\text{EOS} | w_1 w_2)$$

RNNs: Extracting WFAs: Background!

- Language-Model RNNs
- WFAs
    - Matrix Representation

RNNs: Extracting WFAs: Background!



DFA

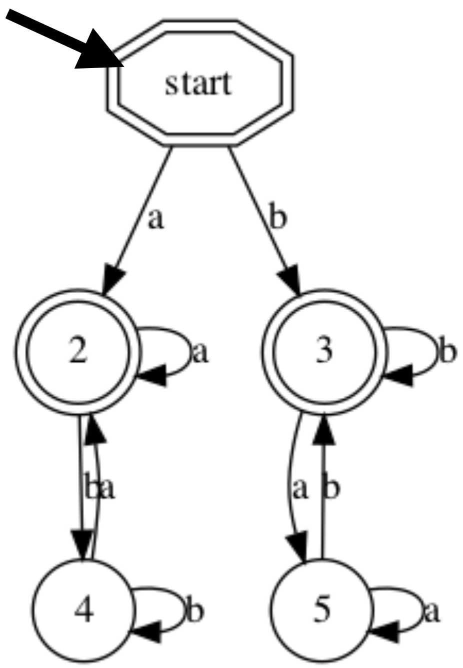
deterministic

$$A = \langle \Sigma, Q, q_0, F, \delta_Q \rangle$$

$$\delta_Q : Q \times \Sigma \rightarrow Q$$

$$A(w) = \begin{cases} \text{Acc} & \text{if } \hat{\delta}_Q(w) \in F \\ \text{Rej,} & \text{else} \end{cases}$$

RNNs: Extracting WFAs: Background!



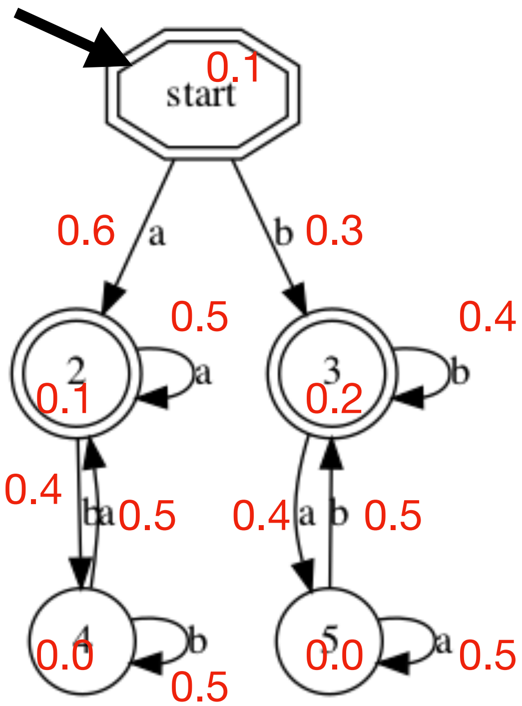
DFA

deterministic

$$A = \langle \Sigma, Q, q_0, F, \delta_Q \rangle$$

$$\delta_Q : Q \times \Sigma \to Q$$

$$A(w) = \begin{cases} \text{Acc} & \text{if } \hat{\delta}_Q(w) \in F \\ \text{Rej,} & \text{else} \end{cases}$$



WDFA

weighted deterministic

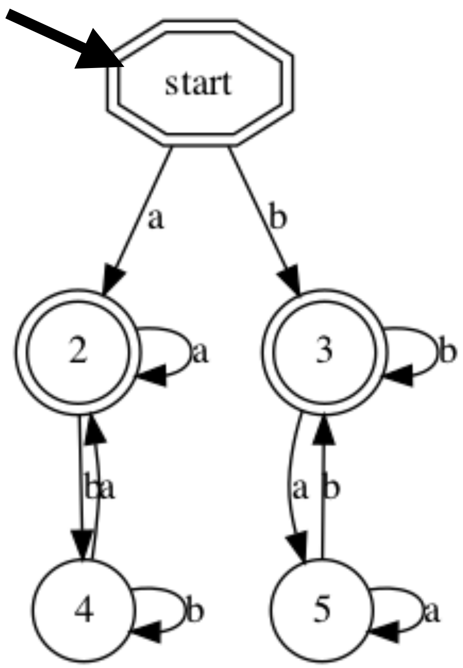
$$A = \langle \Sigma, Q, q_0, \delta_Q, \delta_W, \beta \rangle$$

$$\delta_W \colon Q \times \Sigma \to \mathbb{R}$$

$$\beta \colon Q \to \mathbb{R}$$

$$A(w) = \Big(\prod_{i \in [|w|]} \delta_W(\hat{\delta}_Q(w_{1:i-1}), w_i) \Big) \cdot \beta(\hat{\delta_Q(w)})$$

RNNs: Extracting WFAs: Background!



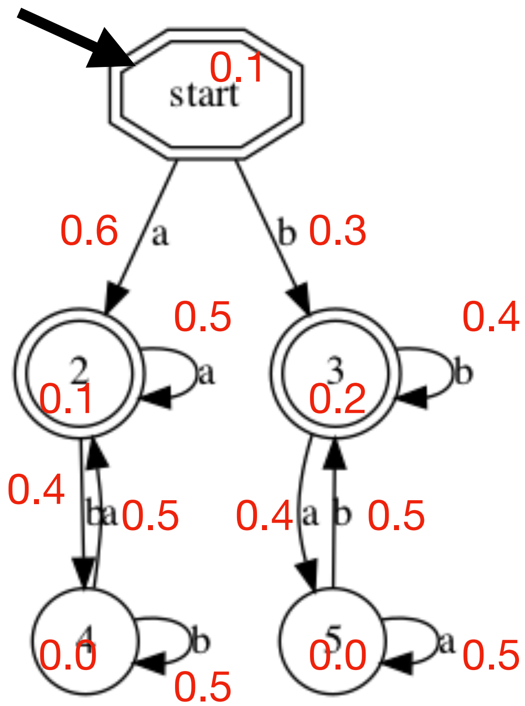
DFA

deterministic

$$A = \langle \Sigma, Q, q_0, F, \delta_Q \rangle$$

$$\delta_Q : Q \times \Sigma \to Q$$

$$A(w) = \begin{cases} \text{Acc} & \text{if } \hat{\delta}_Q(w) \in F \\ \text{Rej}, & \text{else} \end{cases}$$



WDFA

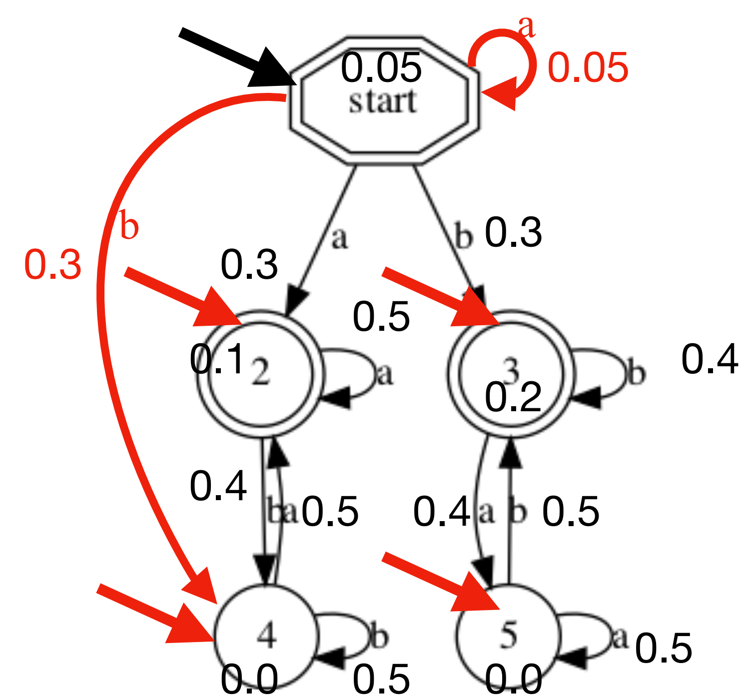
weighted deterministic

$$A = \langle \Sigma, Q, q_0, \delta_Q, \delta_W, \beta \rangle$$

$$\delta_W \colon Q \times \Sigma \to \mathbb{R}$$

$$\beta \colon Q \to \mathbb{R}$$

$$A(w) = \Big(\prod_{i \in [|w|]} \delta_W(\hat{\delta}_Q(w_{1:i-1}), w_i) \Big) \cdot \beta(\delta_Q \hat{(}w))$$



WFA

weighted

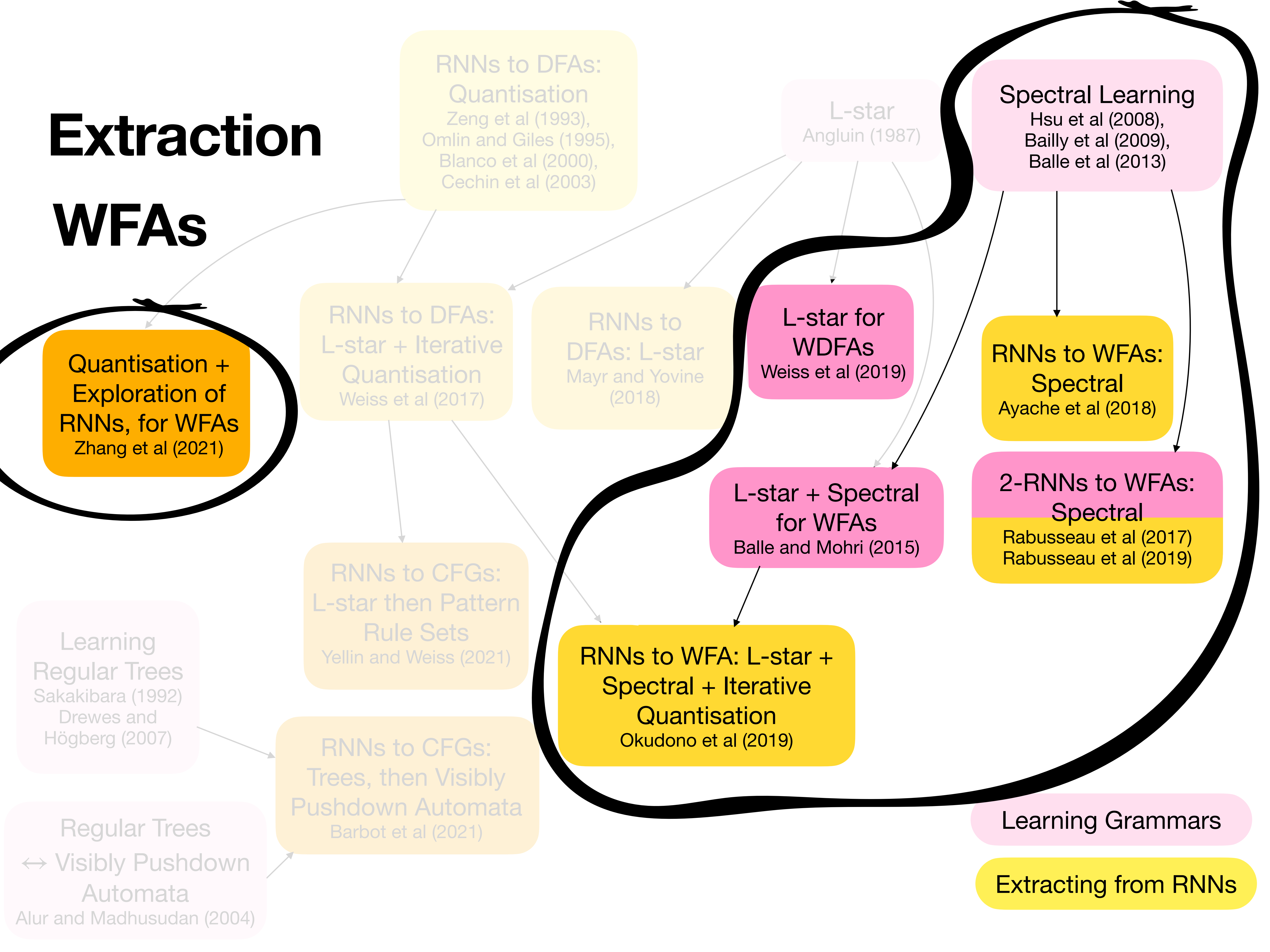
$$A = \langle \Sigma, Q, \alpha, \beta, \{W_\sigma\}_{\sigma \in \Sigma} \rangle$$

$$\alpha \colon Q \to \mathbb{R}$$

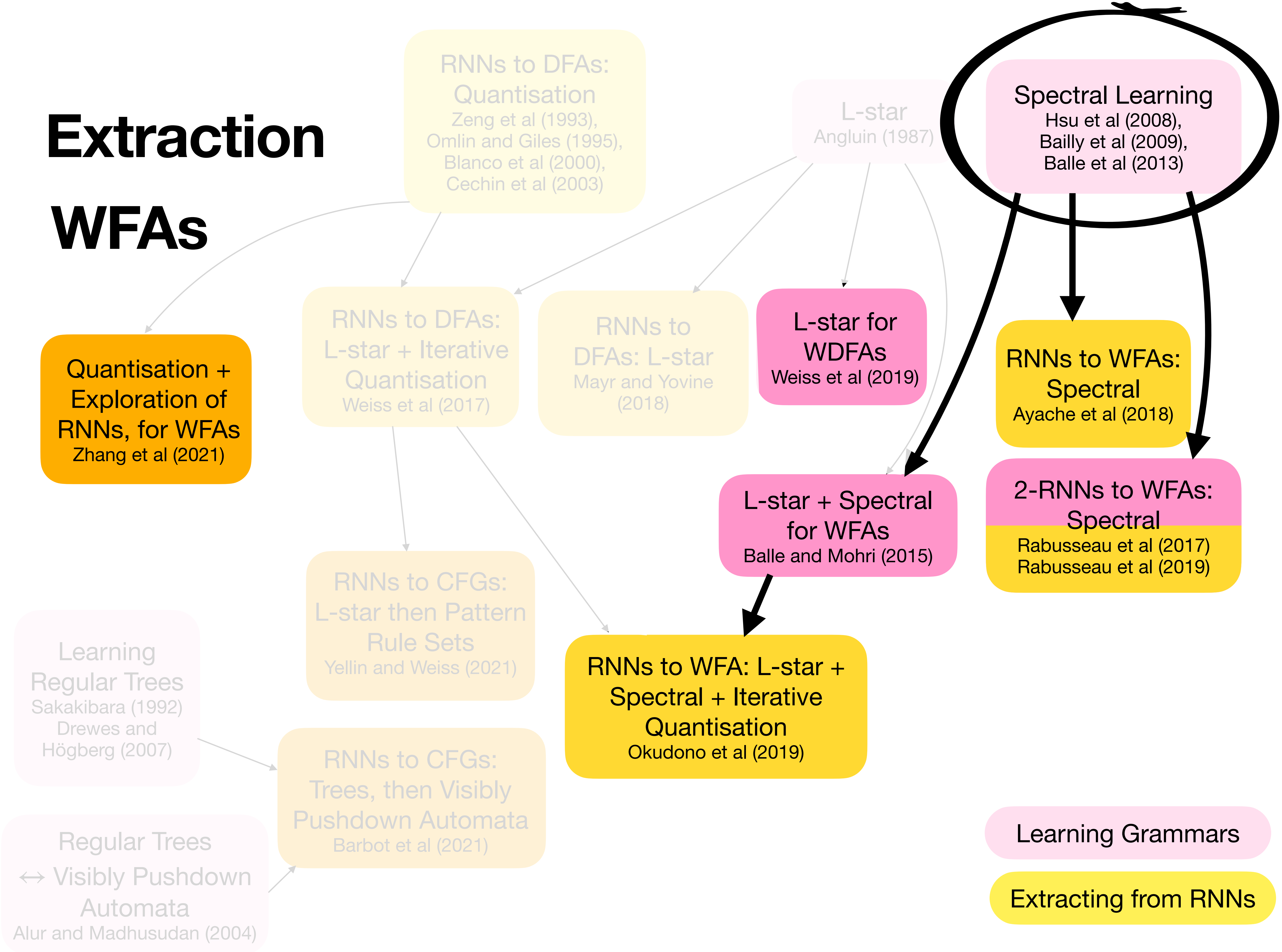
$$\beta \colon Q \to \mathbb{R}$$

$$W_\sigma \in \mathbb{R}^{Q \times Q}$$

$$A(w) = \alpha \cdot W_{w_1} \cdot W_{w_2} \cdot \ldots \cdot W_{w_{|w|}} \cdot \beta$$


Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

Extraction WFAs



RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

A spectral algorithm for learning hidden Markov models

Hsu et al, 2008

Grammatical inference as a principal component analysis problem

Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective

Balle et al, 2013

RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

$T = \langle \Sigma, Q, \alpha^G, \beta^G, \{W_\sigma^G\}_{\sigma \in \Sigma} \rangle$

(example on $\Sigma = \{a, b\}$)

A spectral algorithm for learning hidden Markov models

Hsu et al, 2008

Grammatical inference as a principal component analysis problem

Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective

Balle et al, 2013

RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

$T = \langle \Sigma, Q, \alpha^G, \beta^G, \{W_\sigma^G\}_{\sigma\in\Sigma} \rangle$

(example on $\Sigma = \{a, b\}$)

1. Make Hankel Sub-blocks

Hankel sub-block H

P \ S	ε	b	ab	$\cdots$	v
ε	$T(\varepsilon)$	$T(b)$	$T(ab)$		$T(v)$
a	$T(a)$	$T(ab)$	$T(aab)$		$T(a \cdot v)$
ab	$T(ab)$	$T(abb)$	$T(abab)$		$T(ab \cdot v)$
$\cdots$					
u	$T(u)$	$T(u \cdot b)$	$T(u \cdot ab)$		$T(u \cdot v)$

A spectral algorithm for learning hidden Markov models

Hsu et al, 2008

Grammatical inference as a principal component analysis problem

Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective

Balle et al, 2013

RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

$T = \langle \Sigma, Q, \alpha^G, \beta^G, \{W_\sigma^G\}_{\sigma \in \Sigma} \rangle$

(example on $\Sigma = \{a, b\}$)

1. Make Hankel Sub-blocks

Hankel sub-block H

P \ S	ε	b	ab	$\cdots$	v
ε	$T(\varepsilon)$	$T(b)$	$T(ab)$		$T(v)$
a	$T(a)$	$T(ab)$	$T(aab)$		$T(a \cdot v)$
ab	$T(ab)$	$T(abb)$	$T(abab)$		$T(ab \cdot v)$
$\cdots$					
u	$T(u)$	$T(u \cdot b)$	$T(u \cdot ab)$		$T(u \cdot v)$

Hankel sub-block H^b

P \ S	ε	b	ab	$\cdots$	v
ε					
a					
ab					
$\cdots$					
u					$T(u \cdot b \cdot v)$

Hankel sub-block H^a

P \ S	ε	b	ab	$\cdots$	v
ε					
a					
ab					
$\cdots$					
u					$T(u \cdot a \cdot v)$

A spectral algorithm for learning hidden Markov models

Hsu et al, 2008

Grammatical inference as a principal component analysis problem

Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective

Balle et al, 2013

RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

$T = \langle \Sigma, Q, \alpha^G, \beta^G, \{W^G_\sigma\}_{\sigma \in \Sigma} \rangle$

(example on $\Sigma = \{a, b\}$)

1. Make Hankel Sub-blocks

Hankel sub-block H

P \ S	ε	b	ab	$\cdots$	v
ε	$T(\varepsilon)$	$T(b)$	$T(ab)$		$T(v)$
a	$T(a)$	$T(ab)$	$T(aab)$		$T(a \cdot v)$
ab	$T(ab)$	$T(abb)$	$T(abab)$		$T(ab \cdot v)$
$\cdots$					
u	$T(u)$	$T(u \cdot b)$	$T(u \cdot ab)$		$T(u \cdot v)$

Hankel sub-block H^b

P \ S	ε	b	ab	$\cdots$	v
ε					
a					
ab					
$\cdots$					
u					$T(u \cdot b \cdot v)$

Hankel sub-block H^a

P \ S	ε	b	ab	$\cdots$	v
ε					
a					
ab					
$\cdots$					
u					$T(u \cdot a \cdot v)$

2. $U, d, V = \mathrm{SVD}(H)$

3. (Optional): Trim U, d, V to k largest singular values

4. $\alpha = H_{\varepsilon,:}V \;,\; \beta = (HV)^{\dagger} H_{:,\varepsilon} \;,$
$W_\sigma = (HV)^{\dagger} H^{\sigma} V$

5. $A = \langle \Sigma, [k], \alpha, \beta, \{W_\sigma\}_{\sigma \in \Sigma} \rangle$

A spectral algorithm for learning hidden Markov models

Hsu et al, 2008

Grammatical inference as a principal component analysis problem

Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective

Balle et al, 2013

RNNs: Extracting WFAs: Background!

Spectral Learning of WFAs

$T = \langle \Sigma, Q, \alpha^G, \beta^G, \{W^G_\sigma\}_{\sigma\in\Sigma} \rangle$

(example on $\Sigma = \{a, b\}$)

1. Make Hankel Sub-blocks

Hankel sub-block H

P \ S	ε	b	ab	$\cdots$	v
ε	$T(\varepsilon)$	$T(b)$	$T(ab)$		$T(v)$
a	$T(a)$	$T(ab)$	$T(aab)$		$T(a\cdot v)$
ab	$T(ab)$	$T(abb)$	$T(abab)$		$T(ab\cdot v)$
$\cdots$					
u	$T(u)$	$T(u\cdot b)$	$T(u\cdot ab)$		$T(u\cdot v)$

Hankel sub-block H^b

P \ S	ε	b	ab	$\cdots$	v
ε					
a					
ab					
$\cdots$					
u					$T(u\cdot b\cdot v)$

Hankel sub-block H^a

P \ S	ε	b	ab	$\cdots$	v
ε					
a					
ab					
$\cdots$					
u					$T(u\cdot a\cdot v)$

2. $U, d, V = \mathrm{SVD}(H)$

3. (Optional): Trim U, d, V to k largest singular values

4. $\alpha = H_{\varepsilon,:}V \;,\; \beta = (HV)^\dagger H_{:,\varepsilon} \;,$

$$W_\sigma = (HV)^\dagger H^\sigma V$$

5. $A = \langle \Sigma, [k], \alpha, \beta, \{W_\sigma\}_{\sigma\in\Sigma} \rangle$

A spectral algorithm for learning hidden Markov models

Hsu et al, 2008

Grammatical inference as a principal component analysis problem

Bailly et al, 2009

Spectral learning of weighted automata - A forward-backward perspective

Balle et al, 2013

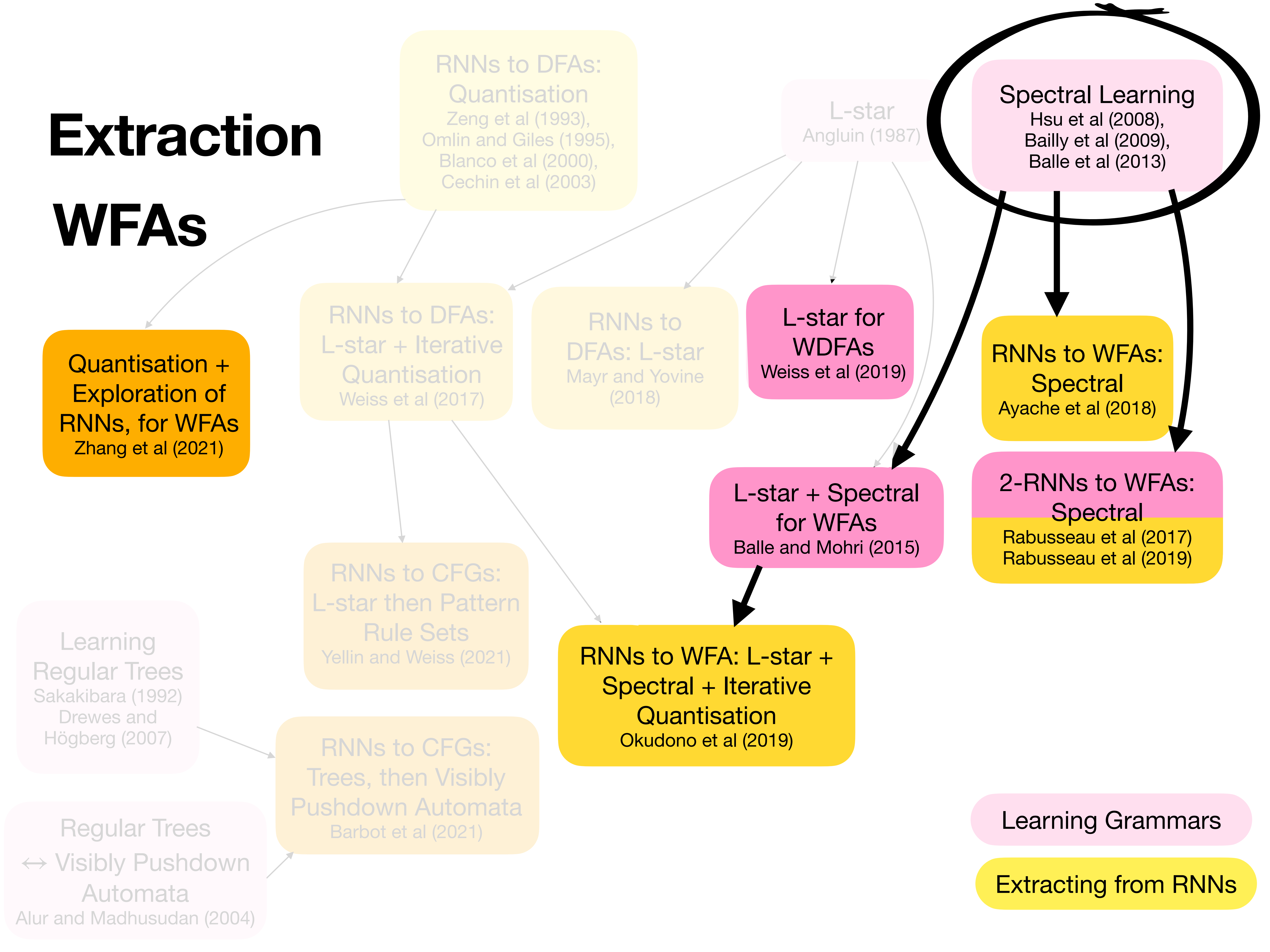
Learning Weighted Automata

Balle and Mohri, 2015

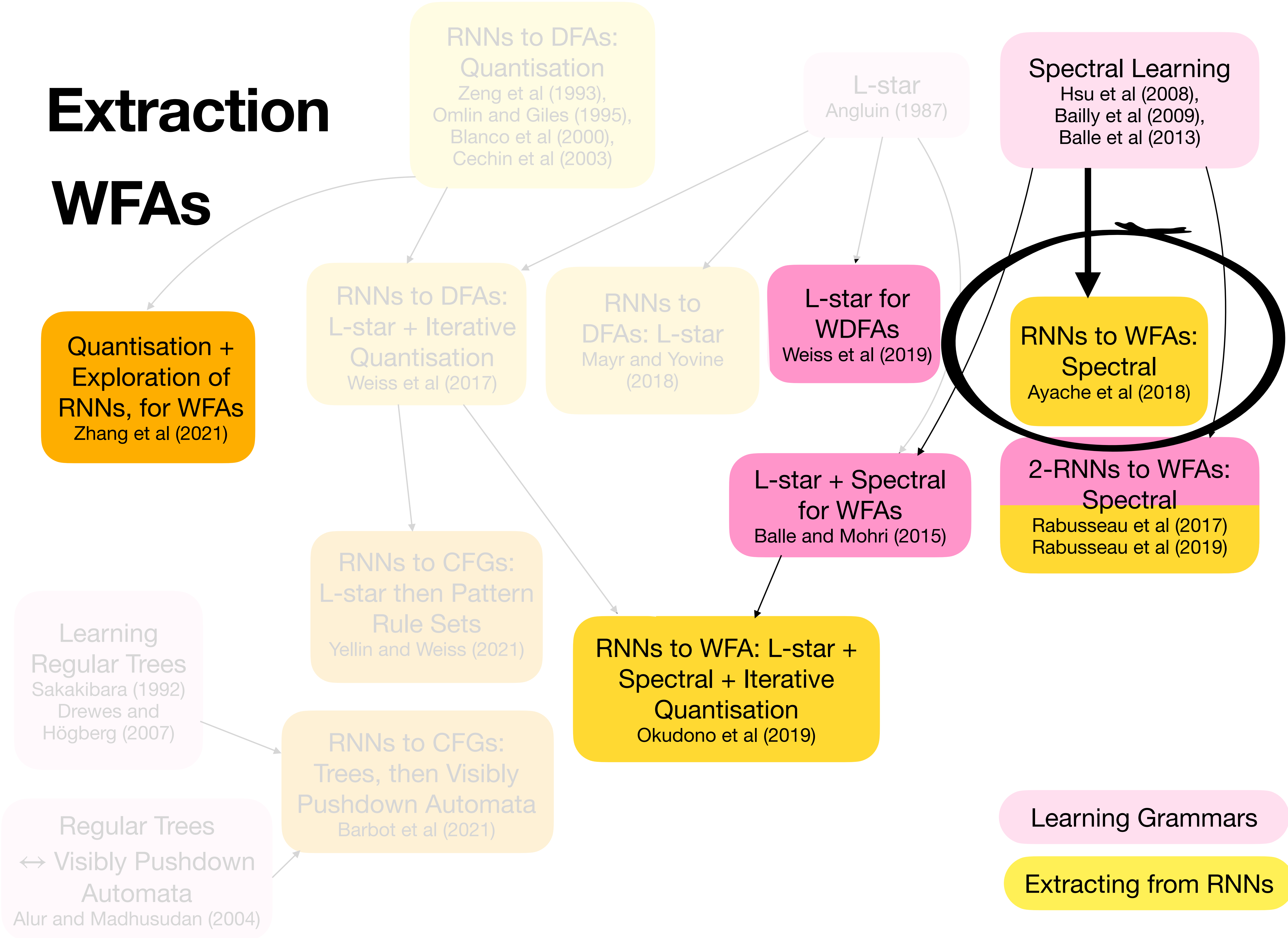
A Maximum Matching Algorithm for Basis Selection in Spectral Learning

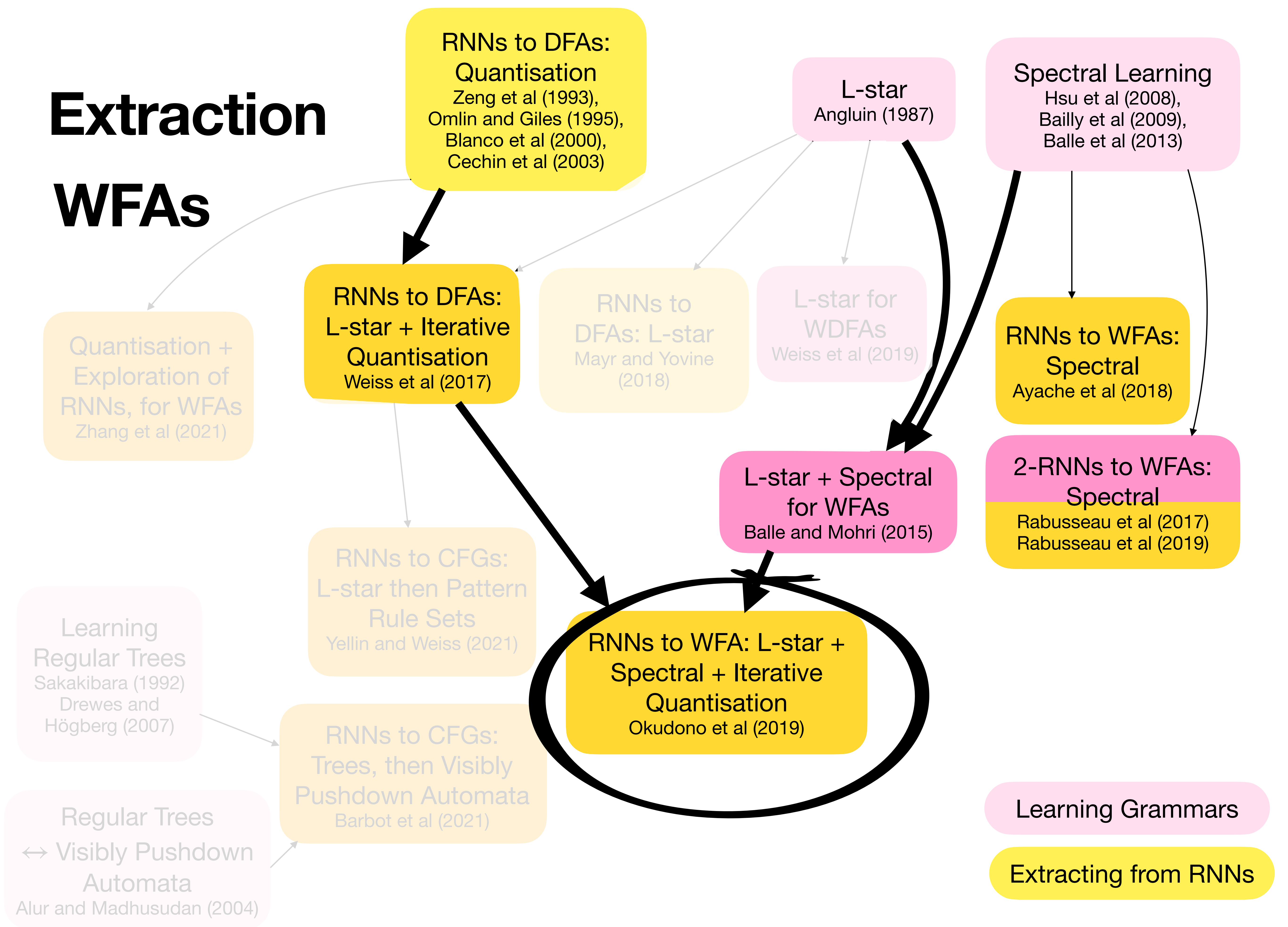
Quattoni et al, 2017

Extraction WFAs

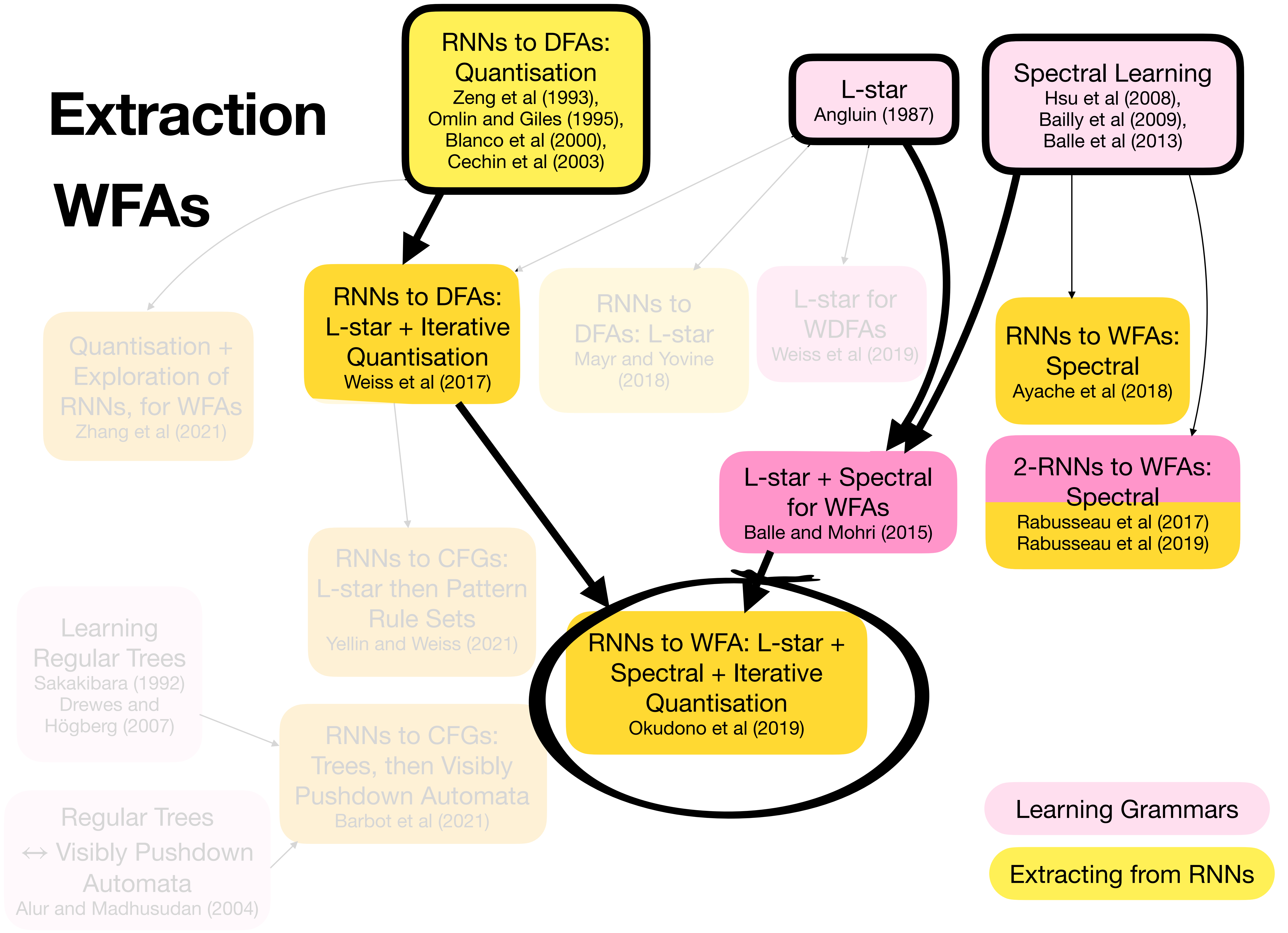


Extraction WFAs





Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs



Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to WFAs: Spectral
Ayache et al (2018)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
L-star + Spectral for WFAs
Balle and Mohri (2015)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

RNNs: Extracting WFAs: Spectral Methods

Explaining Black Boxes on Sequential Data Using Weighted Automata

Ayache et al, 2018

Black Box Model

Build Hankel basis (P,S) by sampling sequences according to black box's distribution

Try multiple sizes for final WFA (truncations k of SVD decomposition) and choose best result

Spectral Learning

Hsu et al (2008), Bailly et al (2009), Balle et al. (2013)

RNNs: Extracting WFAs: Spectral Methods

Explaining Black Boxes on Sequential Data Using Weighted Automata

Ayache et al, 2018

Black Box Model

Build Hankel basis (P,S) by sampling sequences according to black box's distribution

Try multiple sizes for final WFA (truncations k of SVD decomposition) and choose best result

Weighted Automata Extraction from Recurrent Neural Networks via Regression on State Spaces

Okudono et al, 2019

White Box Model (specifically RNN)

Build Hankel basis (P,S) according to queries from and counterexamples to active learning algorithm

Continue until reach equivalence

Spectral Learning

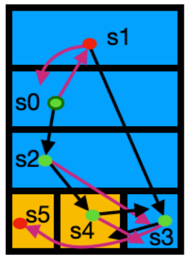
Hsu et al (2008), Bailly et al (2009), Balle et al. (2013)

⊕

Balle and Mohri, 2015

L-star

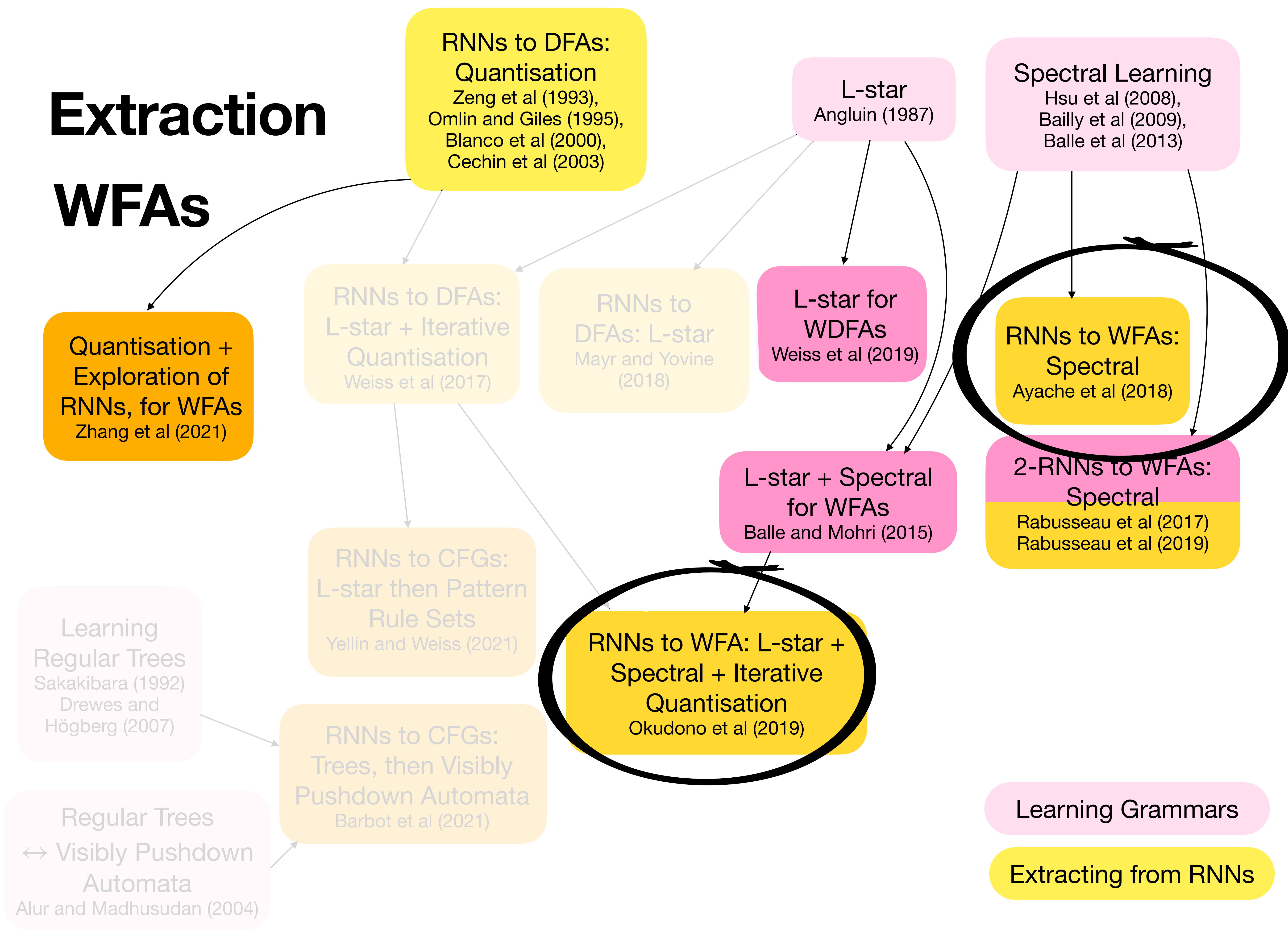
Angluin 1987



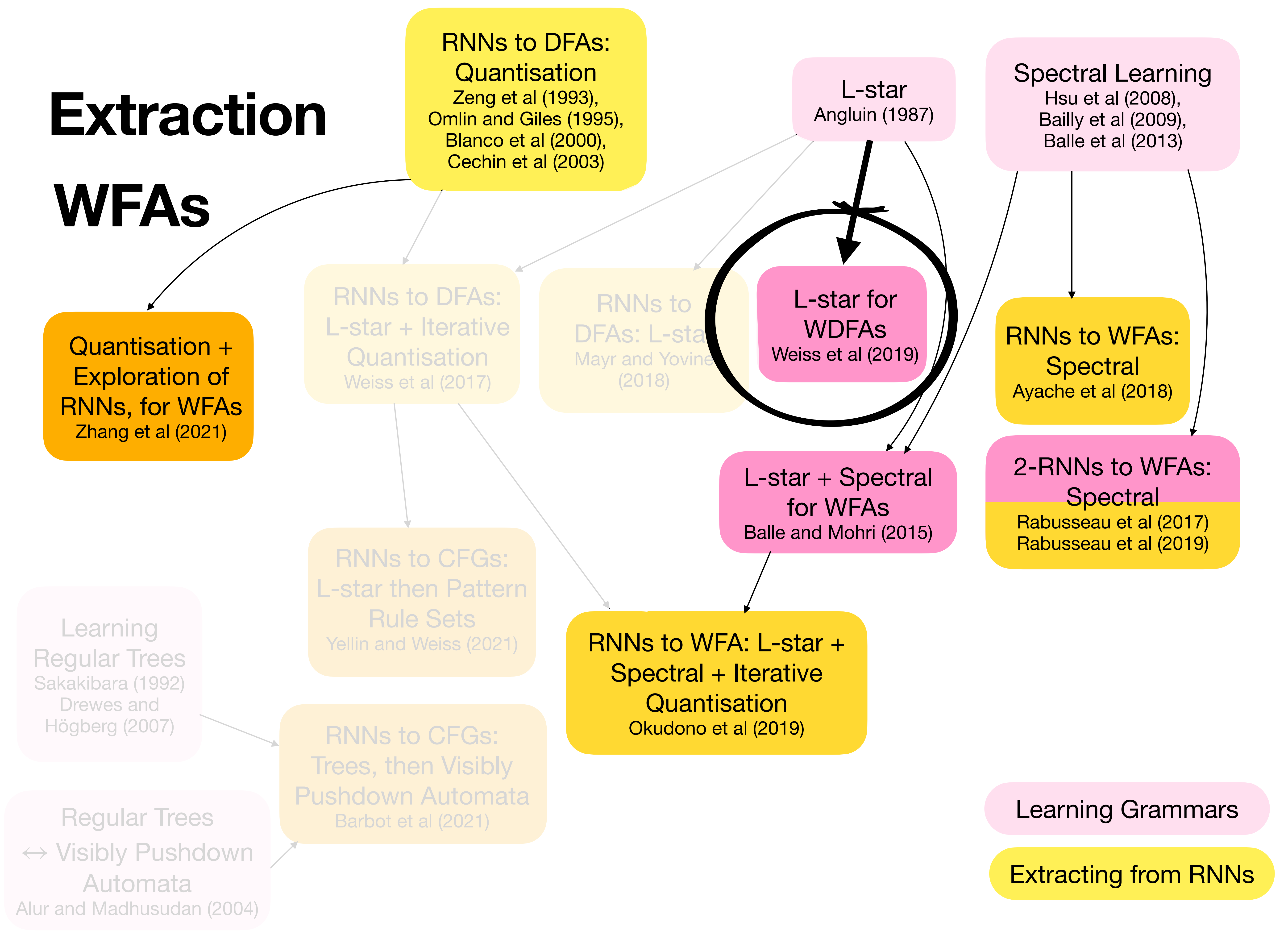
Weiss et al, 2017

Quantisation-Exploration

Omlin and Giles, 1996



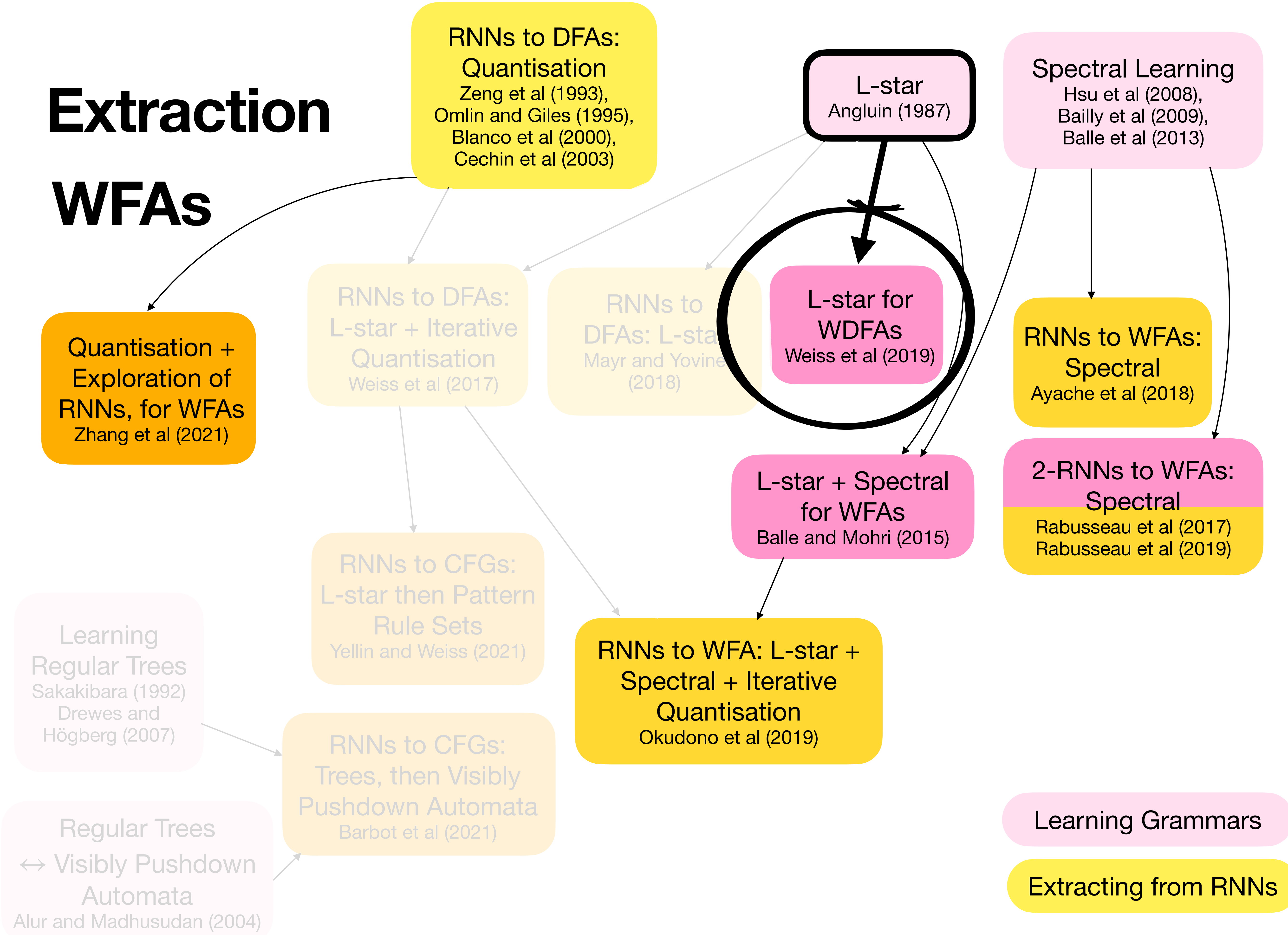
Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs



Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

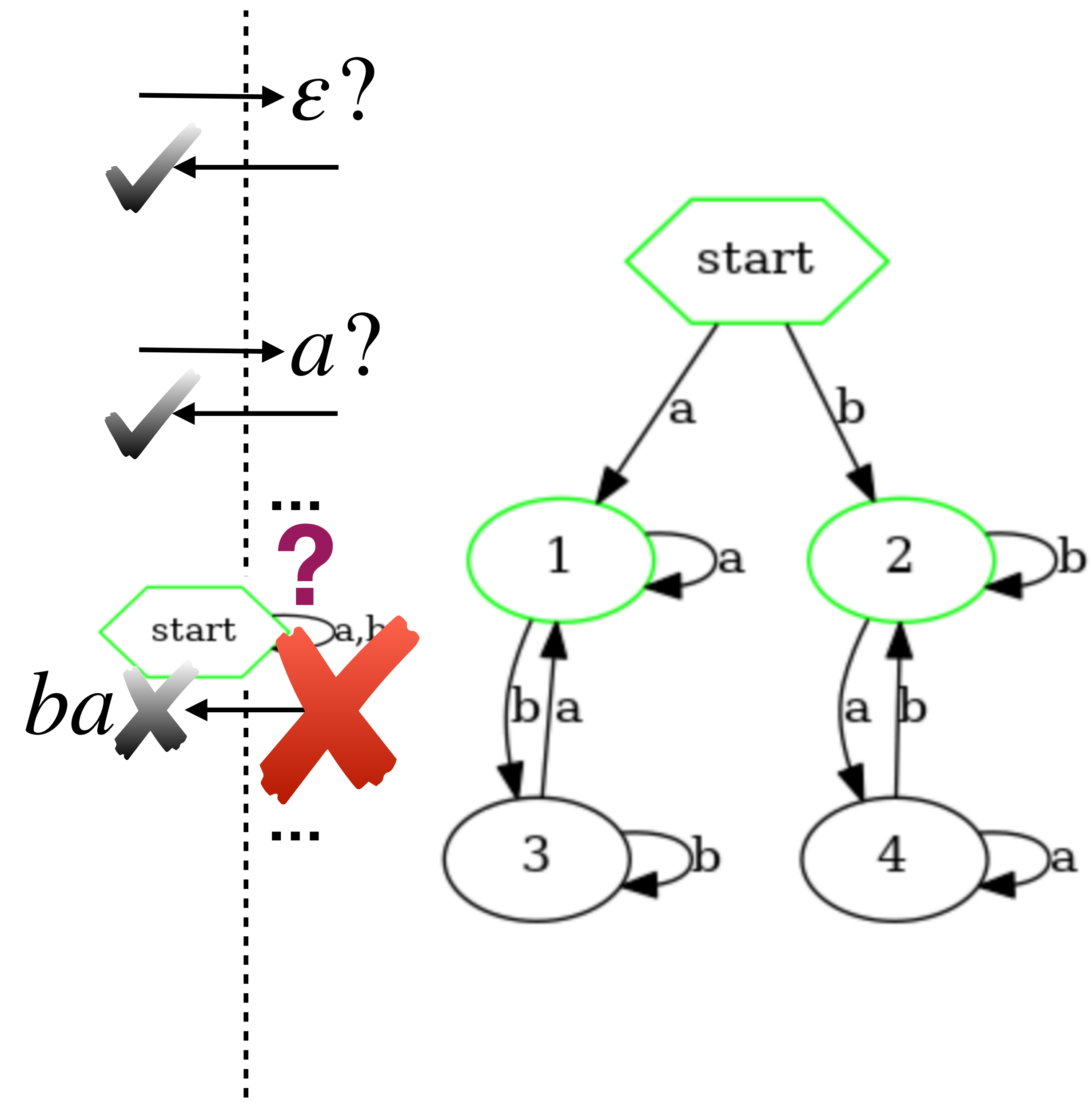
Extraction

WFAs



Background: L*

L*

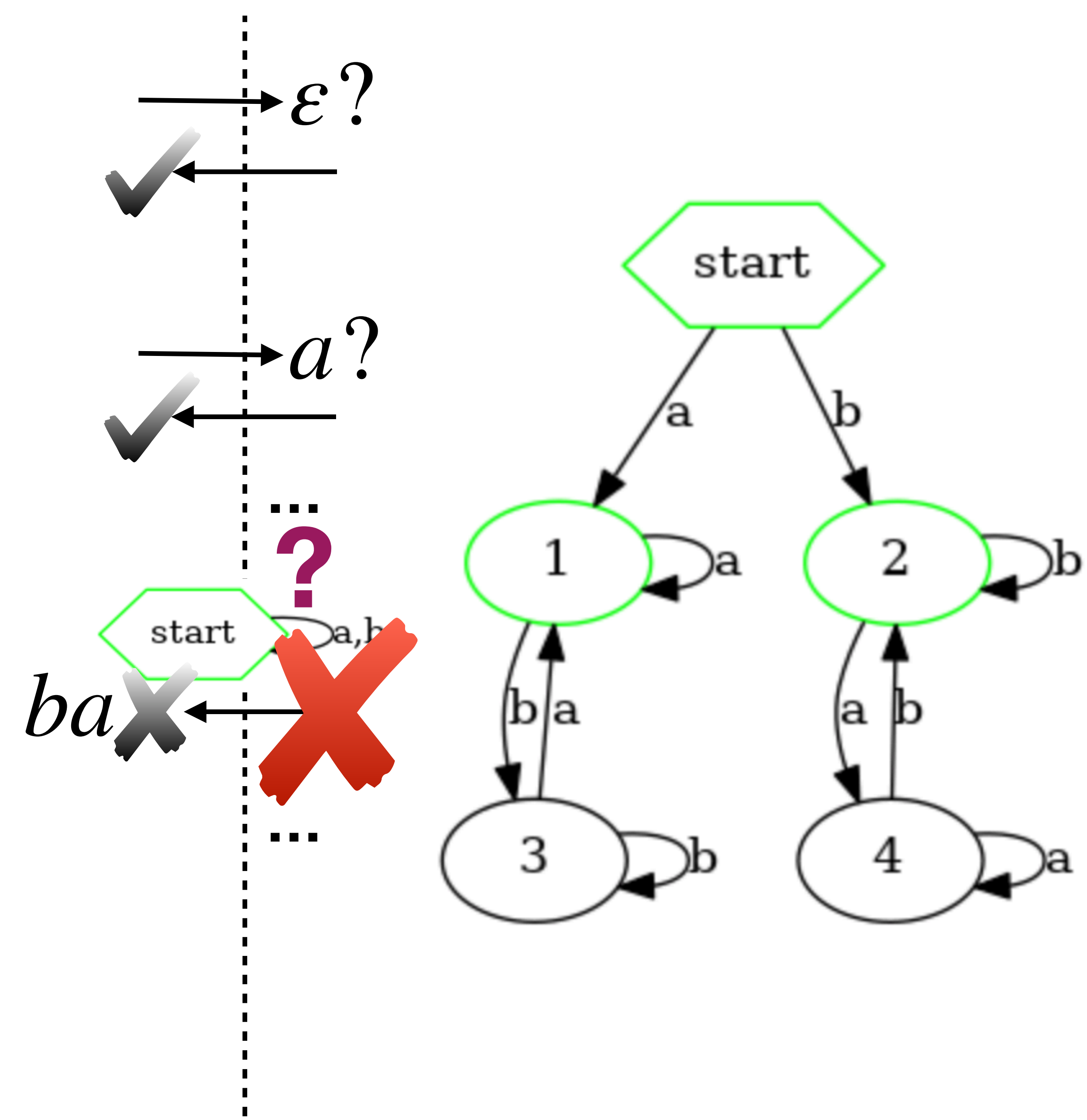


Membership Queries **Equivalence Queries** **Counter-Examples**

Background: L*

The Observation Table

P \ S	ε	a	ba	…
ε	1	1	0	
a	1	1	1	
b	1	0	0	
ba	0	0	0	
bb	1	0	0	
…				

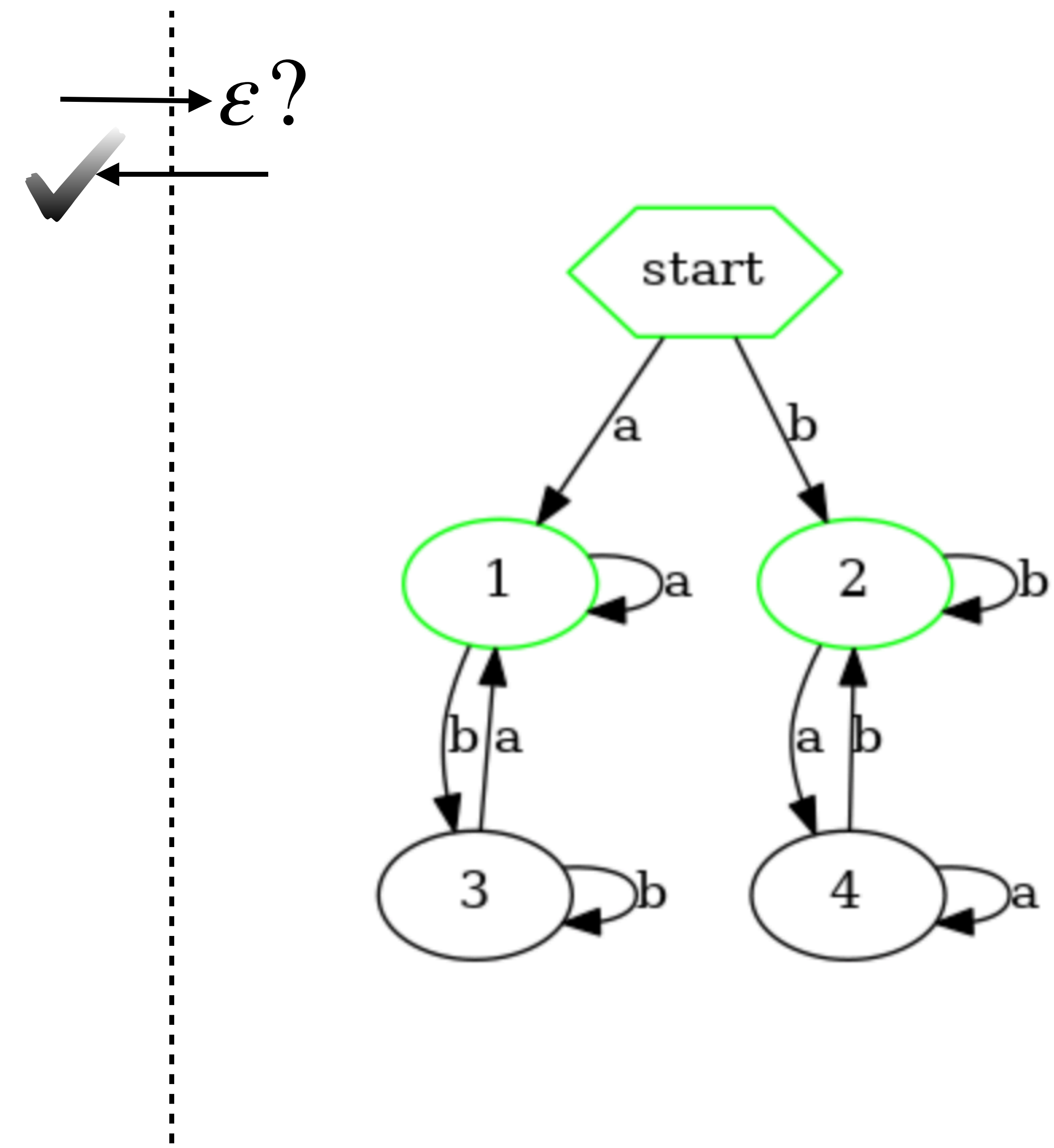


Membership Queries **Equivalence Queries** **Counter-Examples**

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0



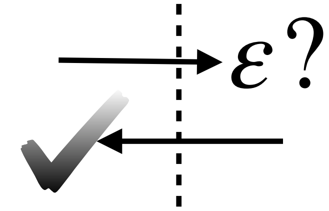
Background: L*

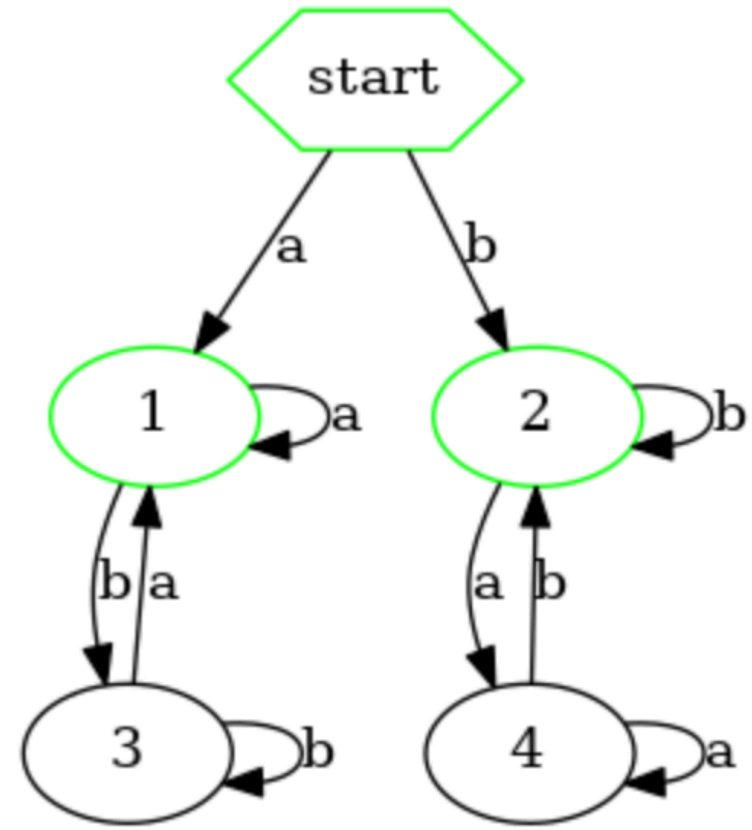
The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0		
bb	1	0	0

Closedness

Consistency





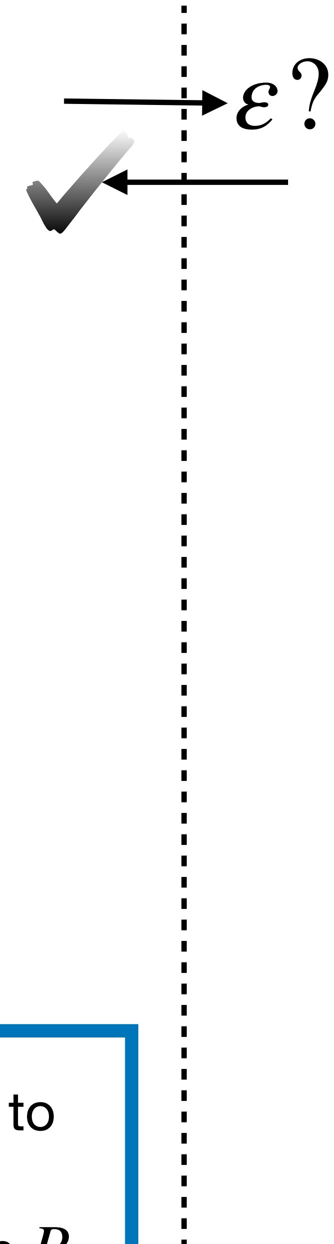
Background: L*

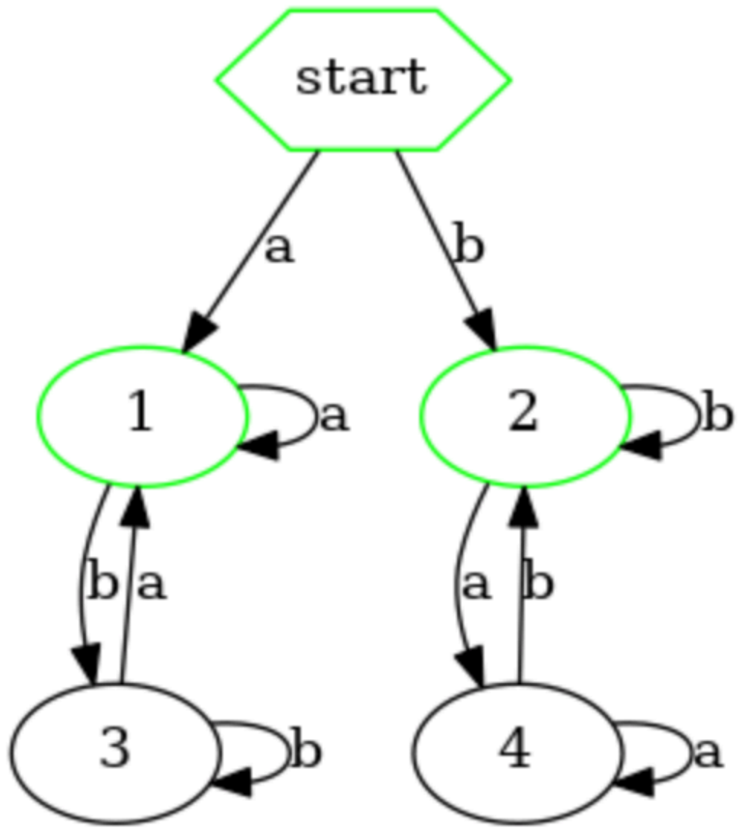
The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	[illegible]	[illegible]
bb	1	[illegible]	[illegible]

Closedness

For all $p \in P$ and $\sigma \in \Sigma$, if we were to add $p \cdot \sigma$ to P, its row would be identical to that of some p' already in P





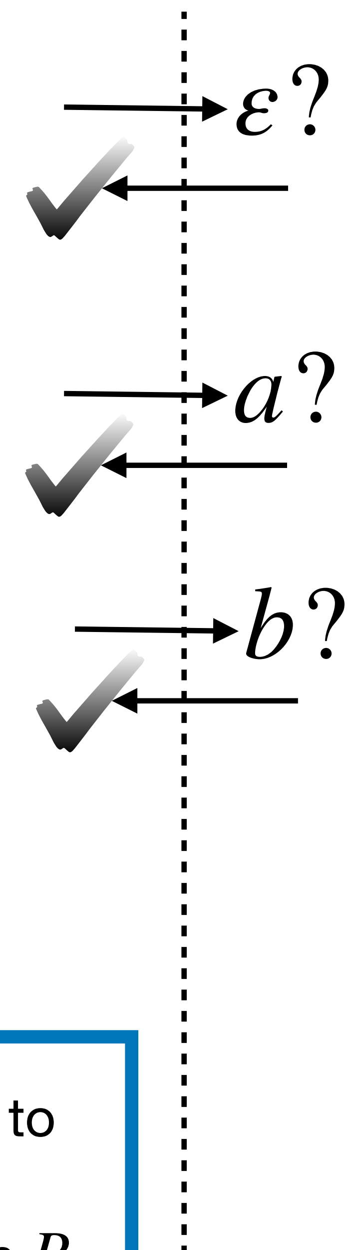
Background: L*

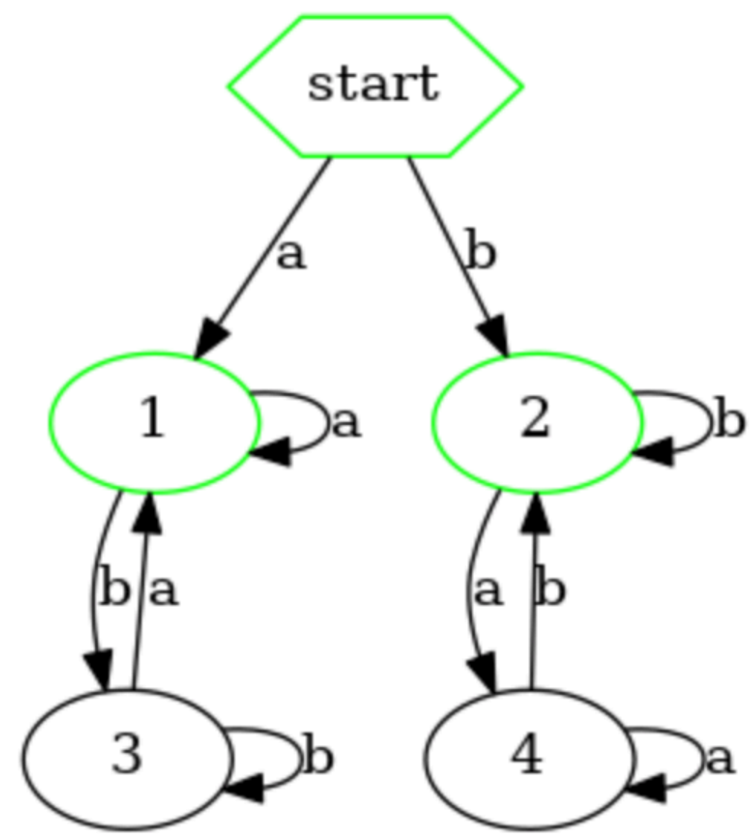
The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	[illegible]
bb	1	[illegible]	[illegible]

Closedness

For all $p \in P$ and $\sigma \in \Sigma$, if we were to add $p \cdot \sigma$ to P, its row would be identical to that of some p' already in P





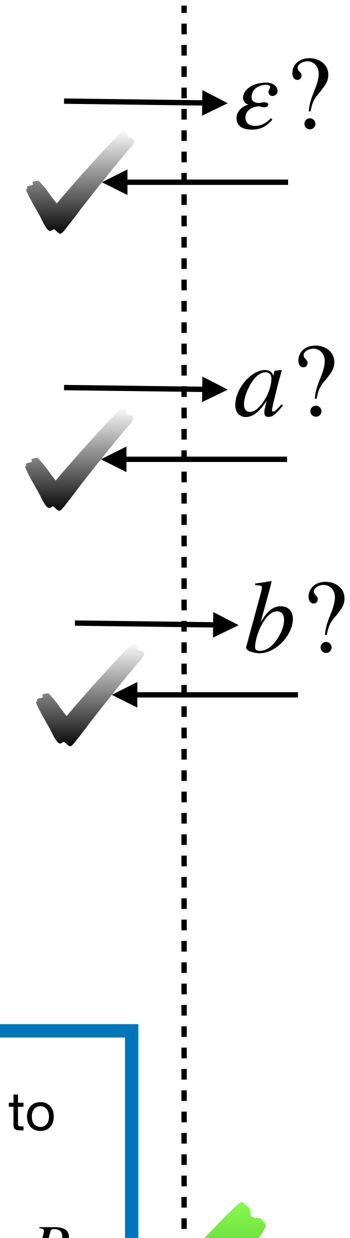
Background: L*

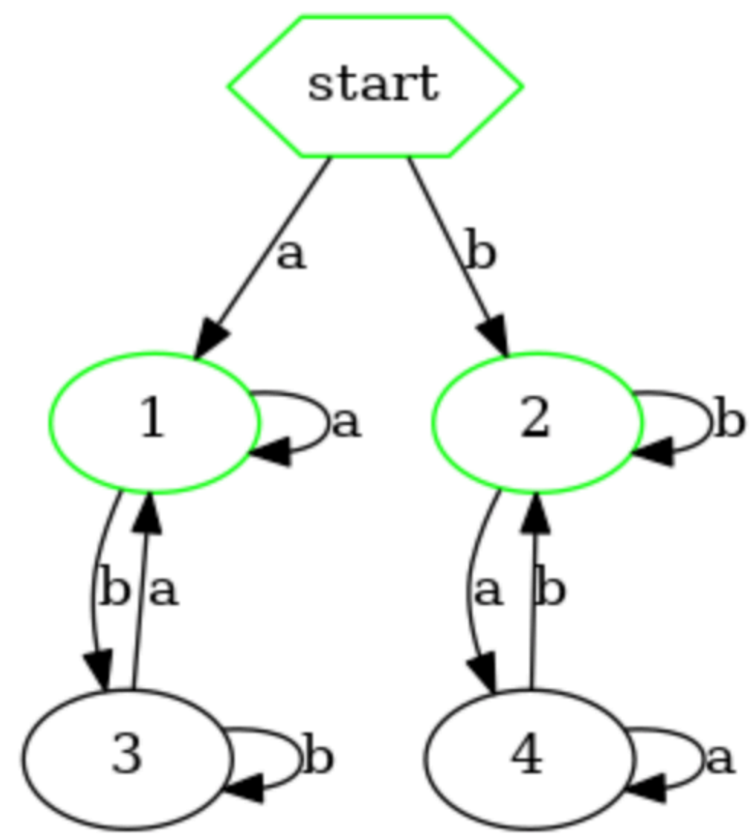
The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0		
bb	1		

Closedness

For all $p \in P$ and $\sigma \in \Sigma$, if we were to add $p \cdot \sigma$ to P, its row would be identical to that of some p' already in P

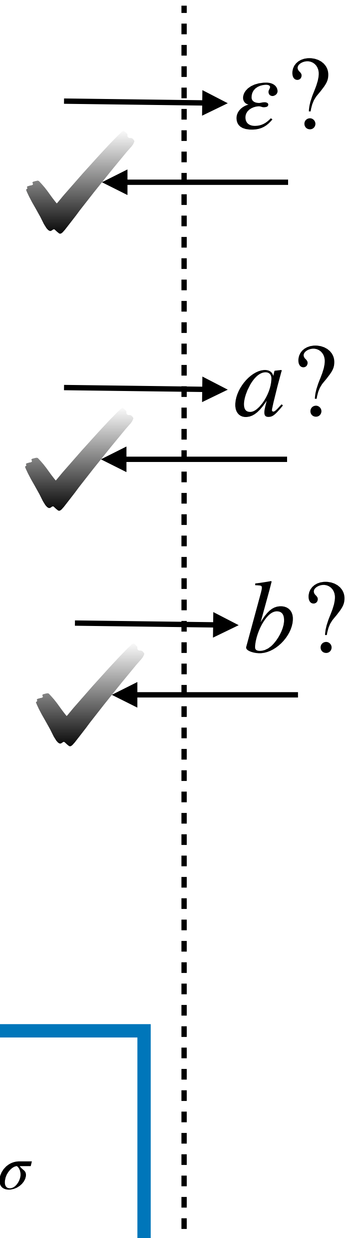


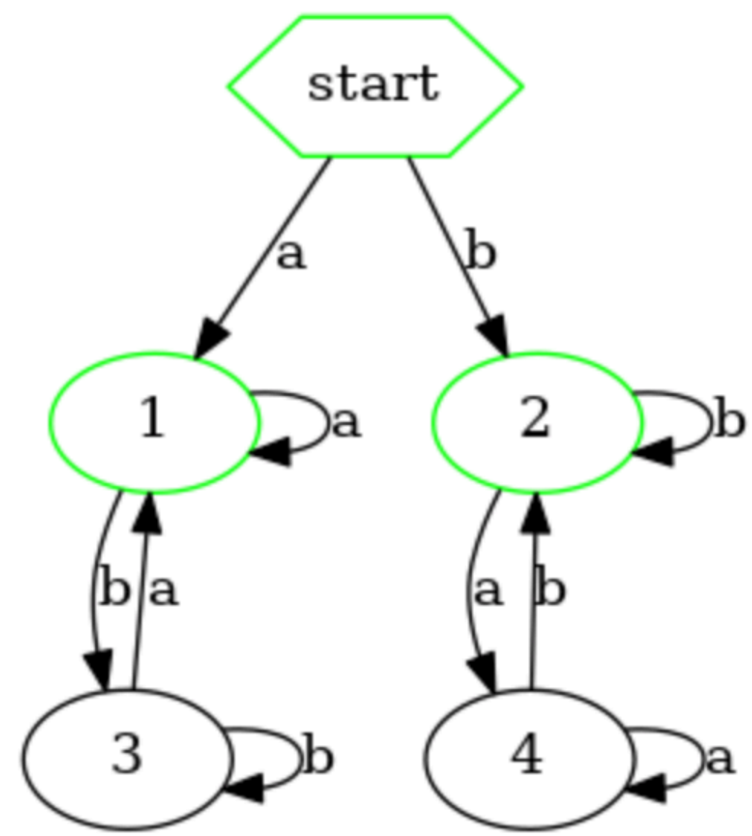


Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba			
bb			





Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

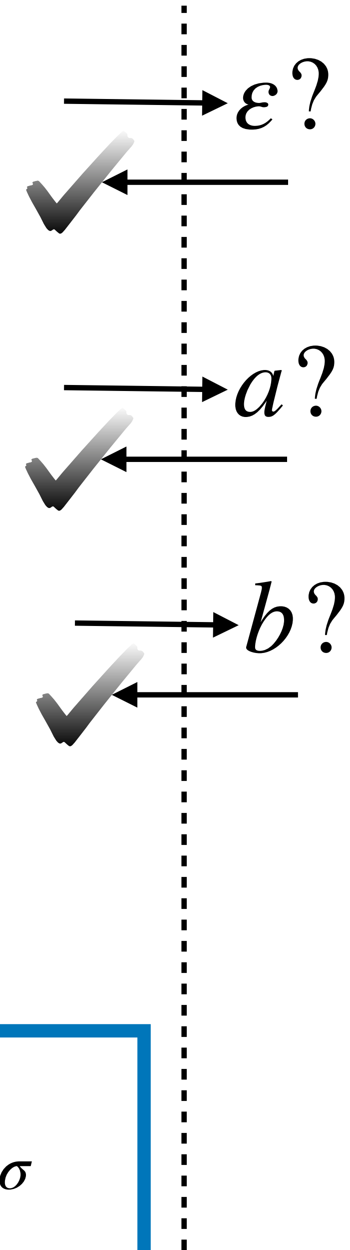
Background: L*

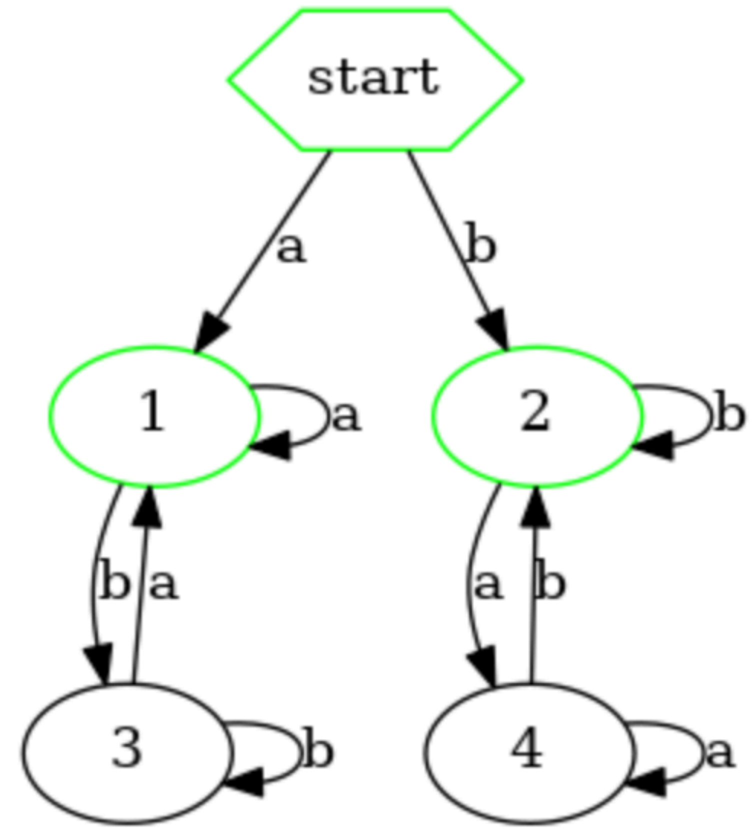
The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	[illegible]	[illegible]	[illegible]
bb	[illegible]	[illegible]	[illegible]

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other



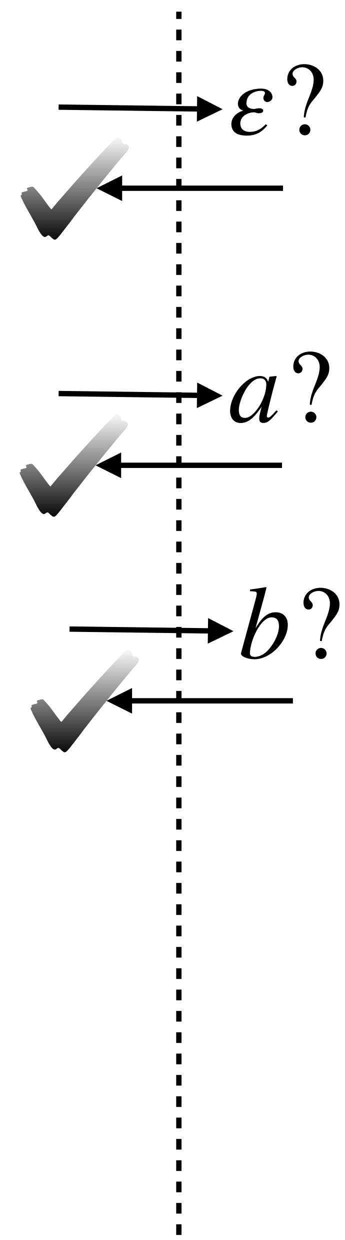


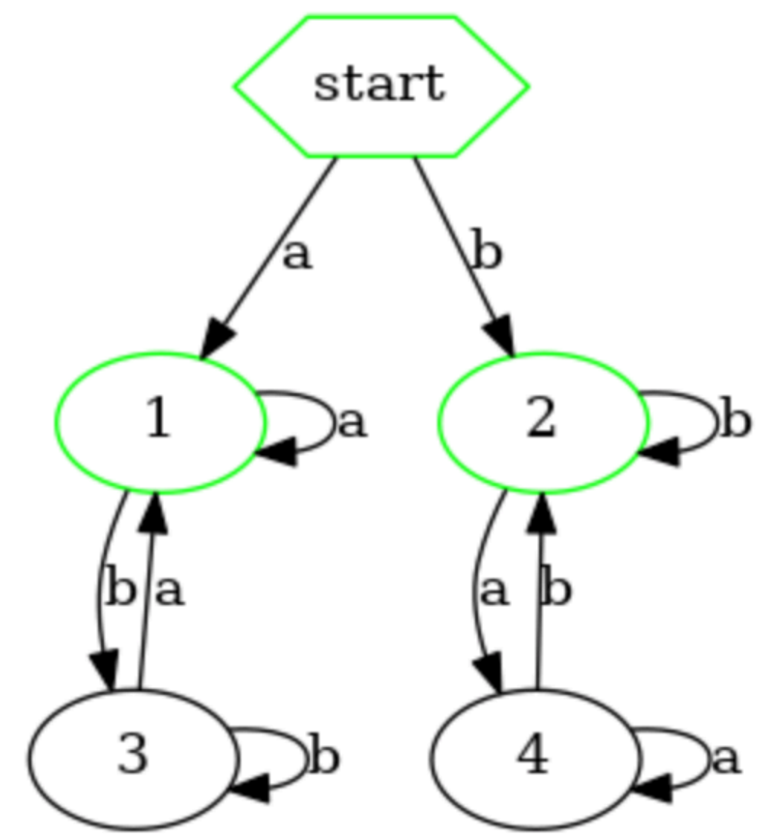
Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

Equivalence Query



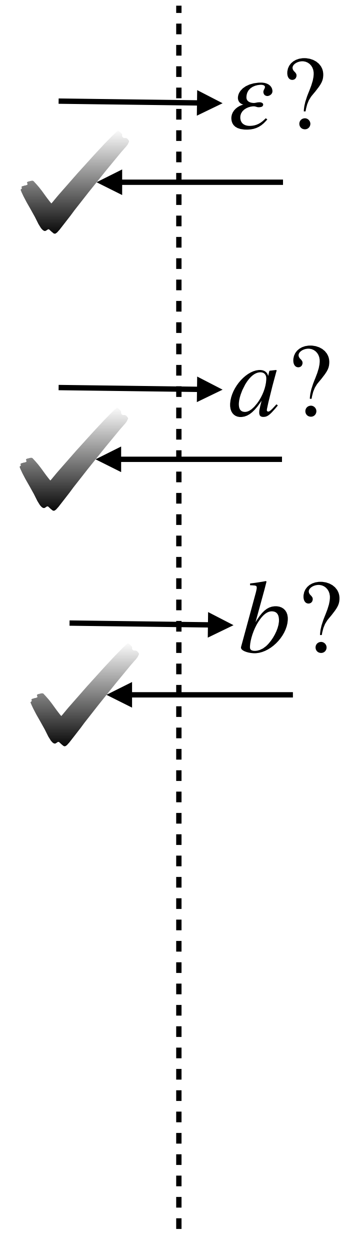


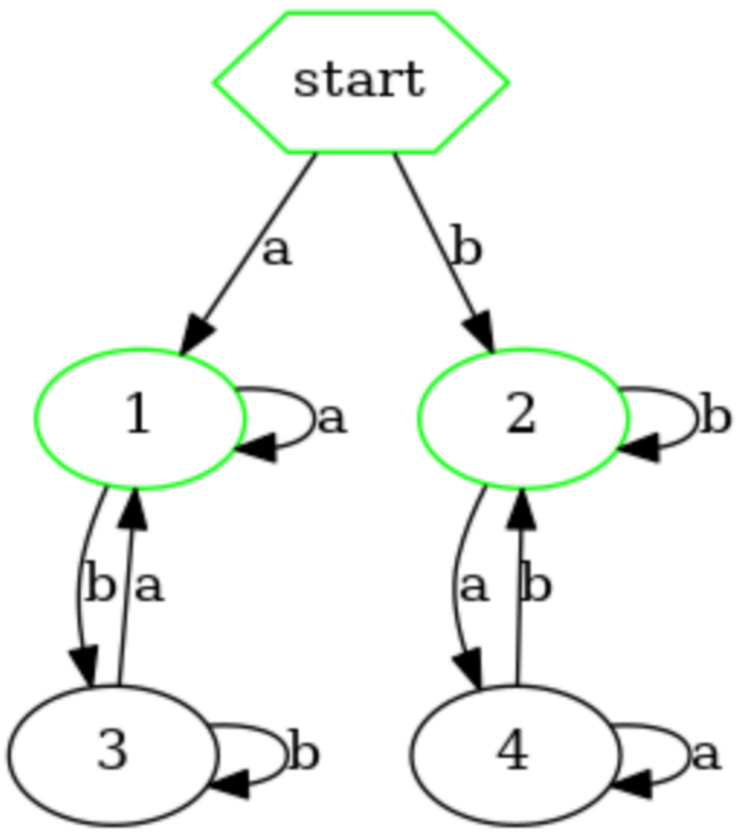
Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0		
bb	1	[illegible]	[illegible]

Equivalence Query



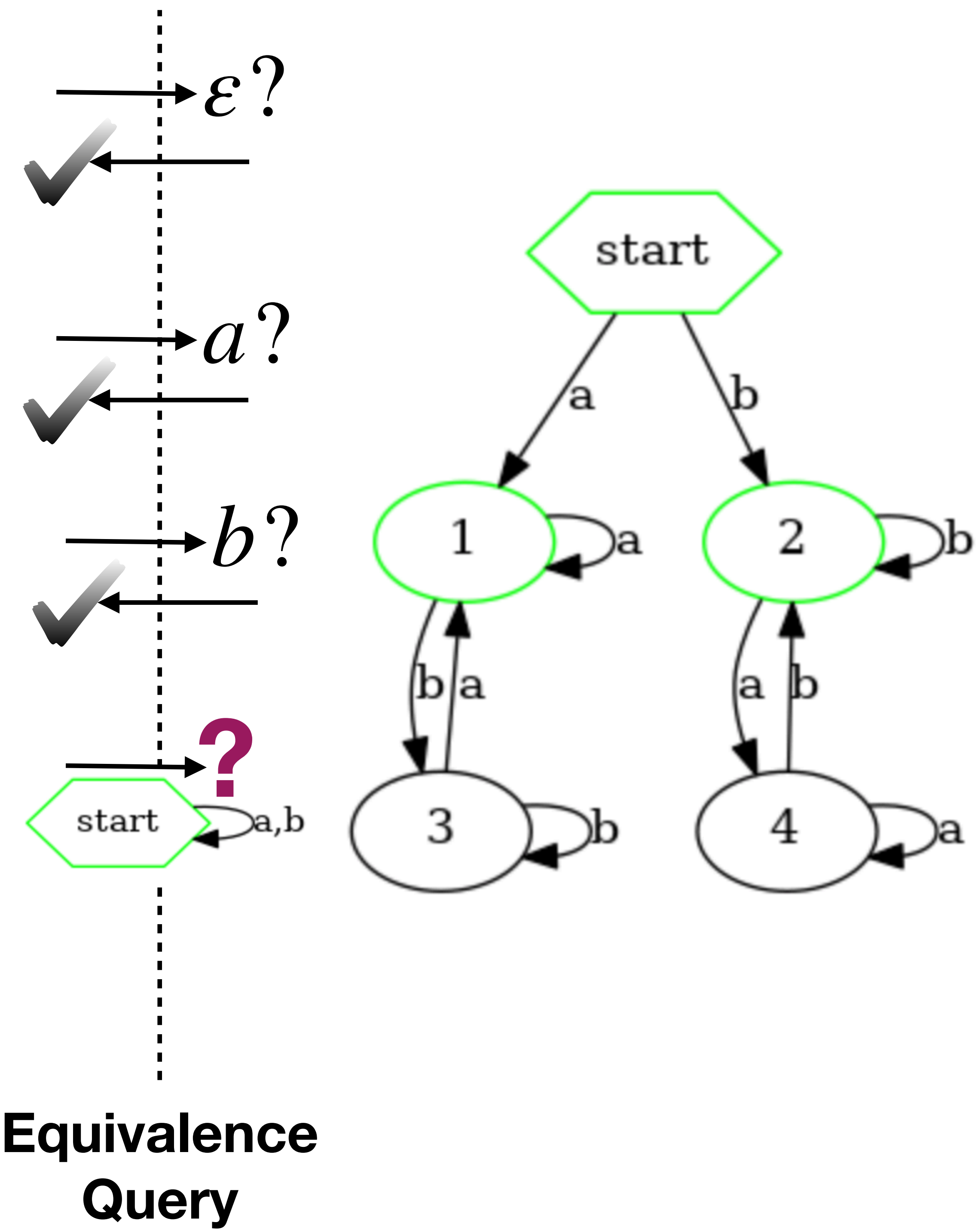


Each group of identical rows describes a single state

Background: L*

The Observation Table

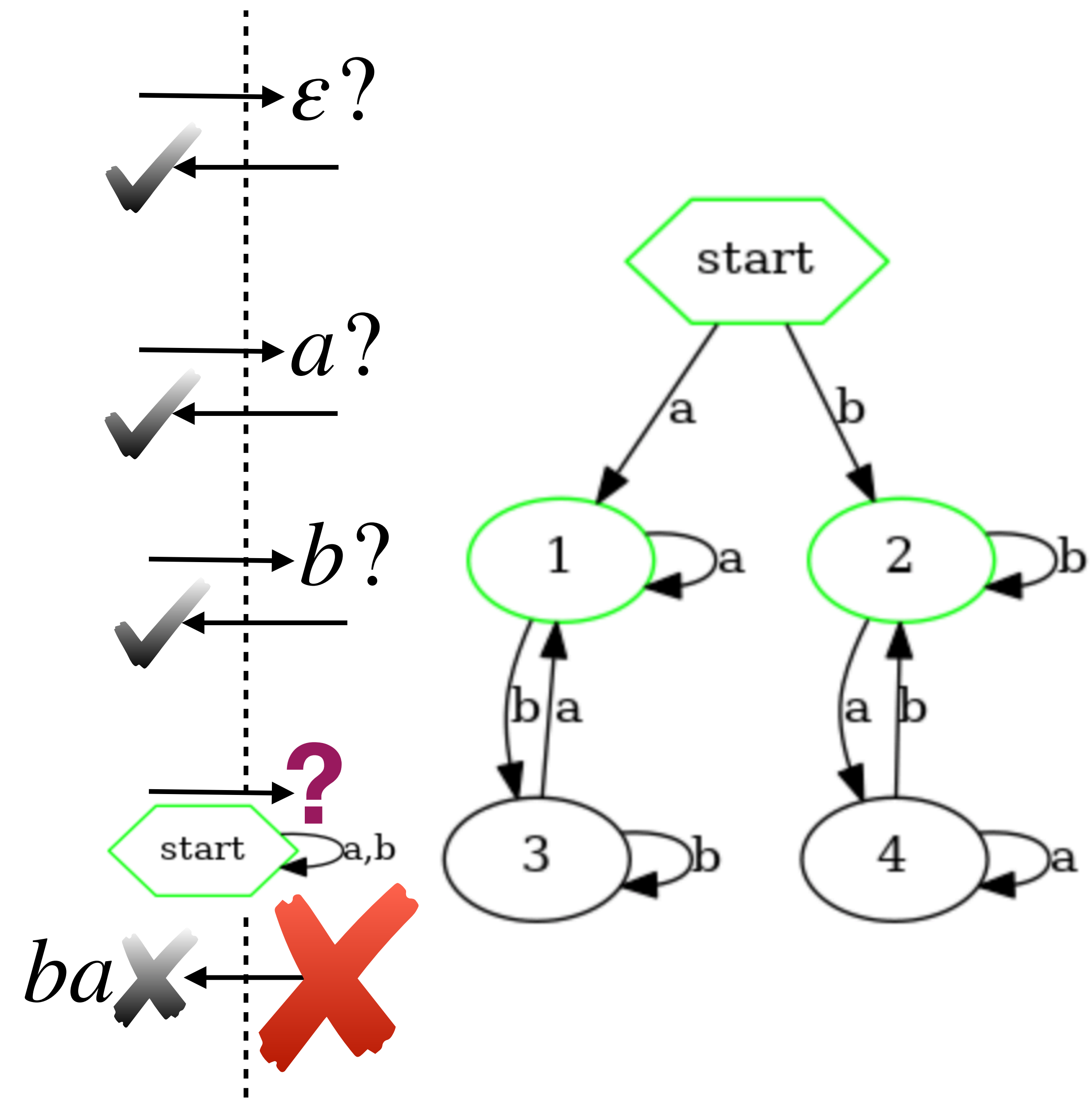
P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0



Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

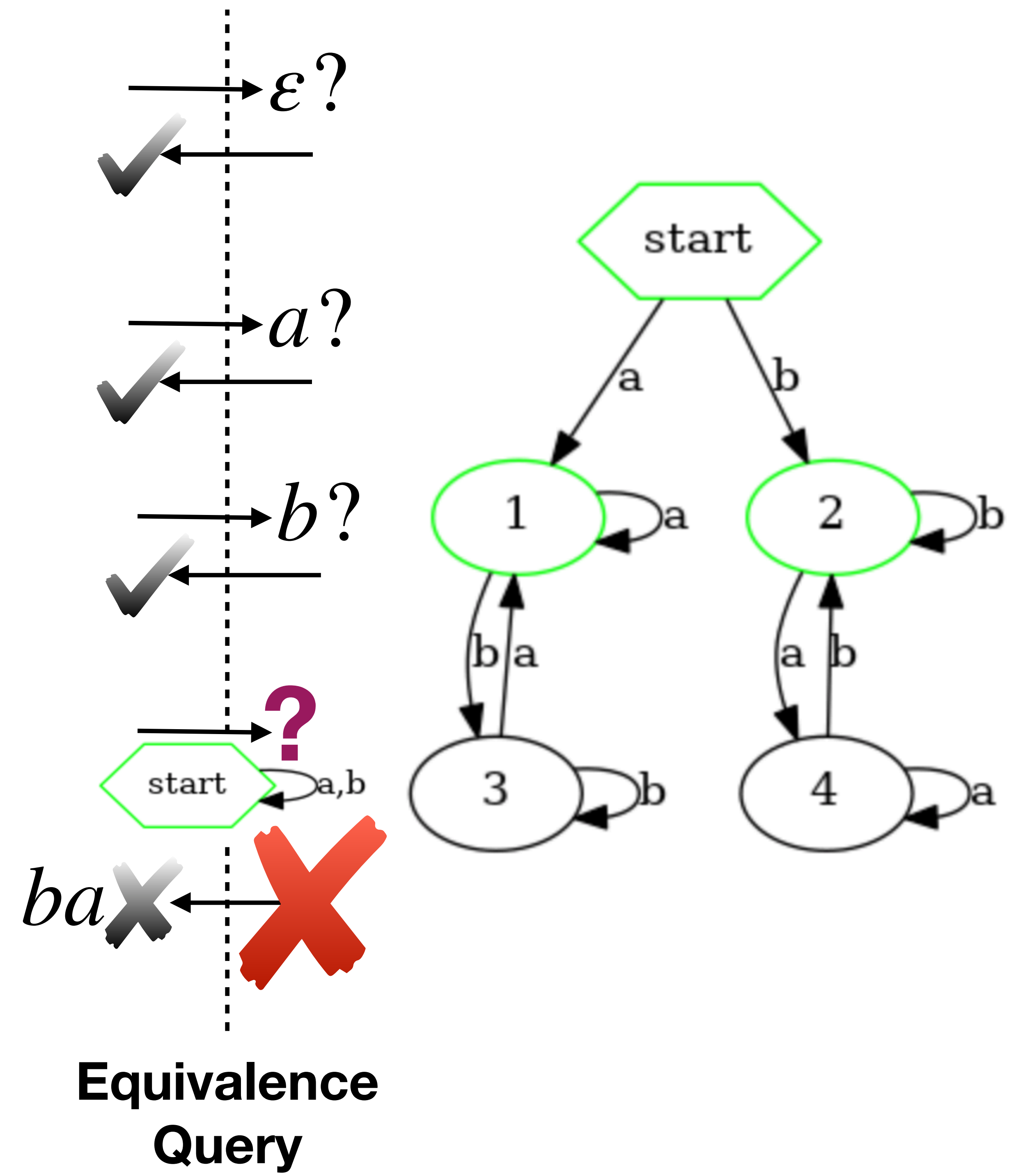


Equivalence Query

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

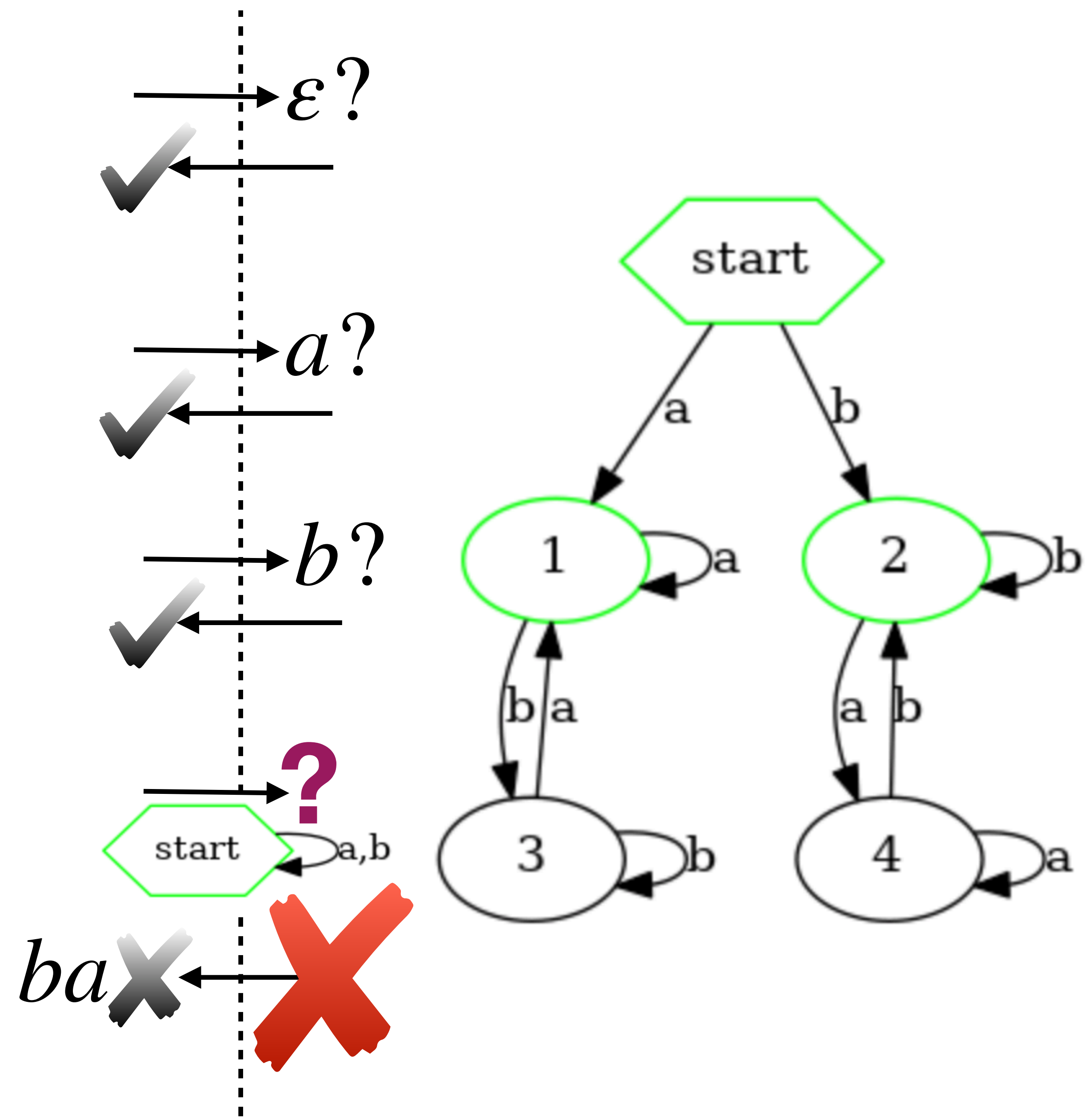


(this is simplified: it also adds to S)

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0



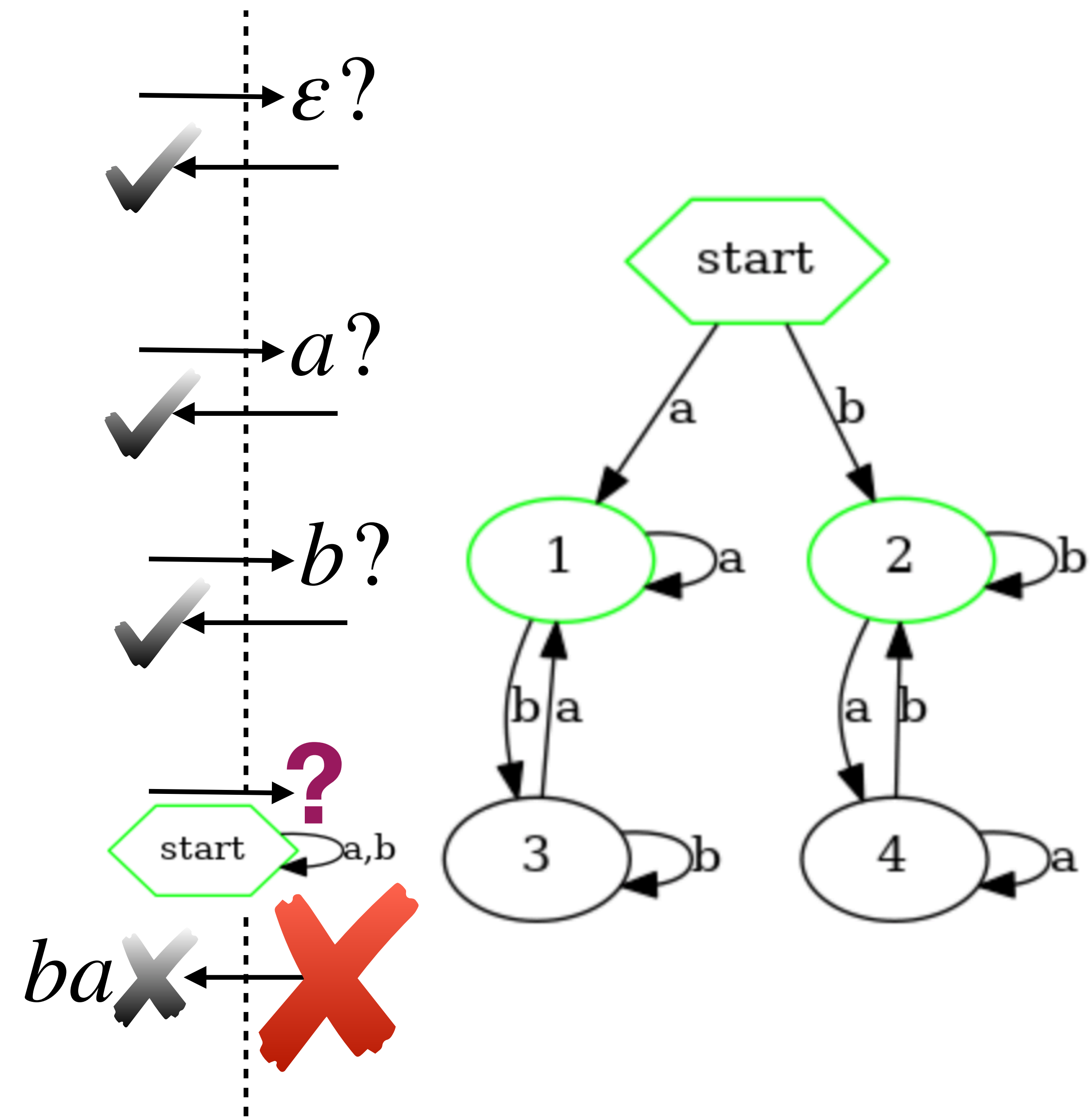
Equivalence Query

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

Closedness



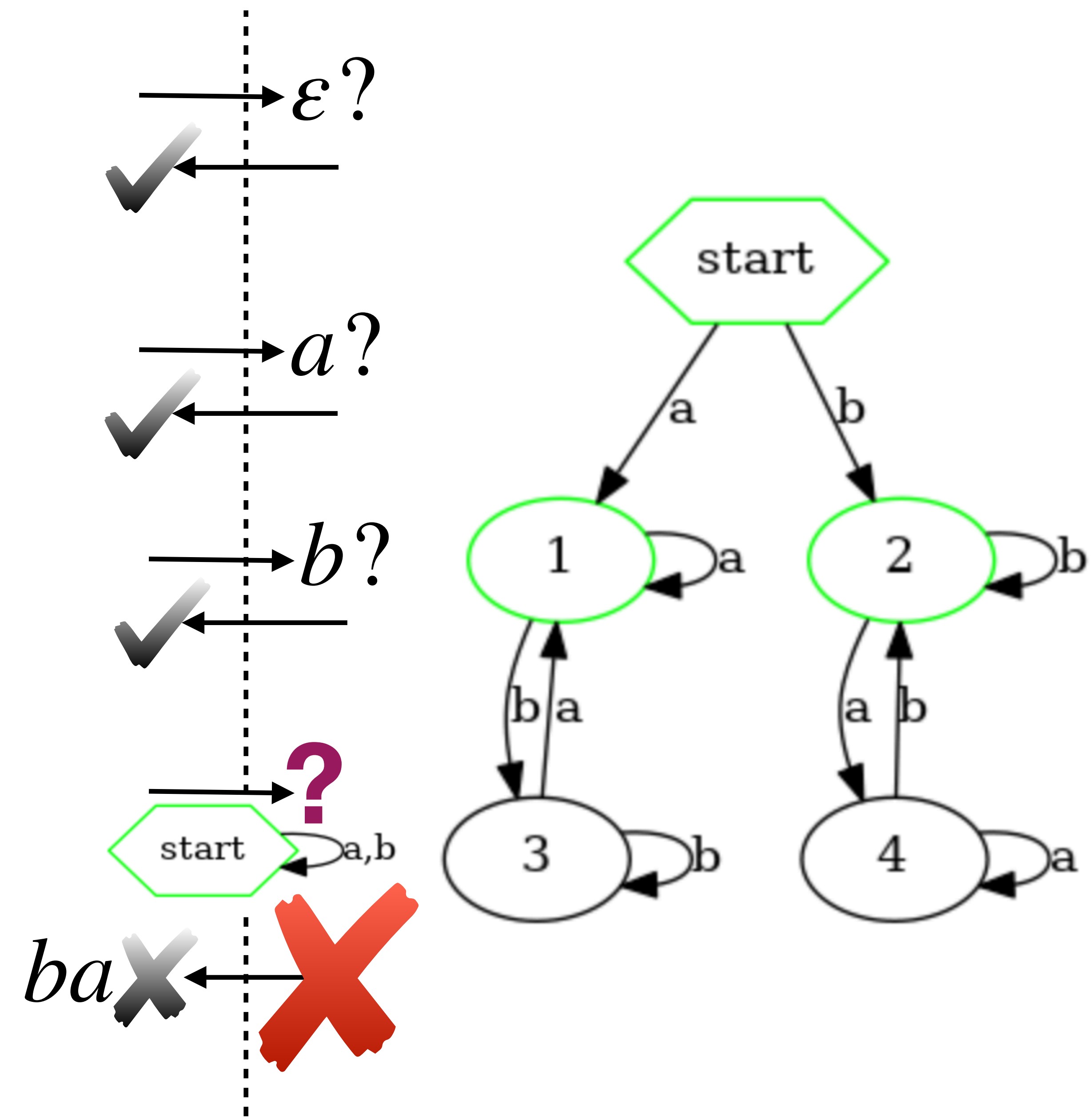
Equivalence Query

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

Consistency?

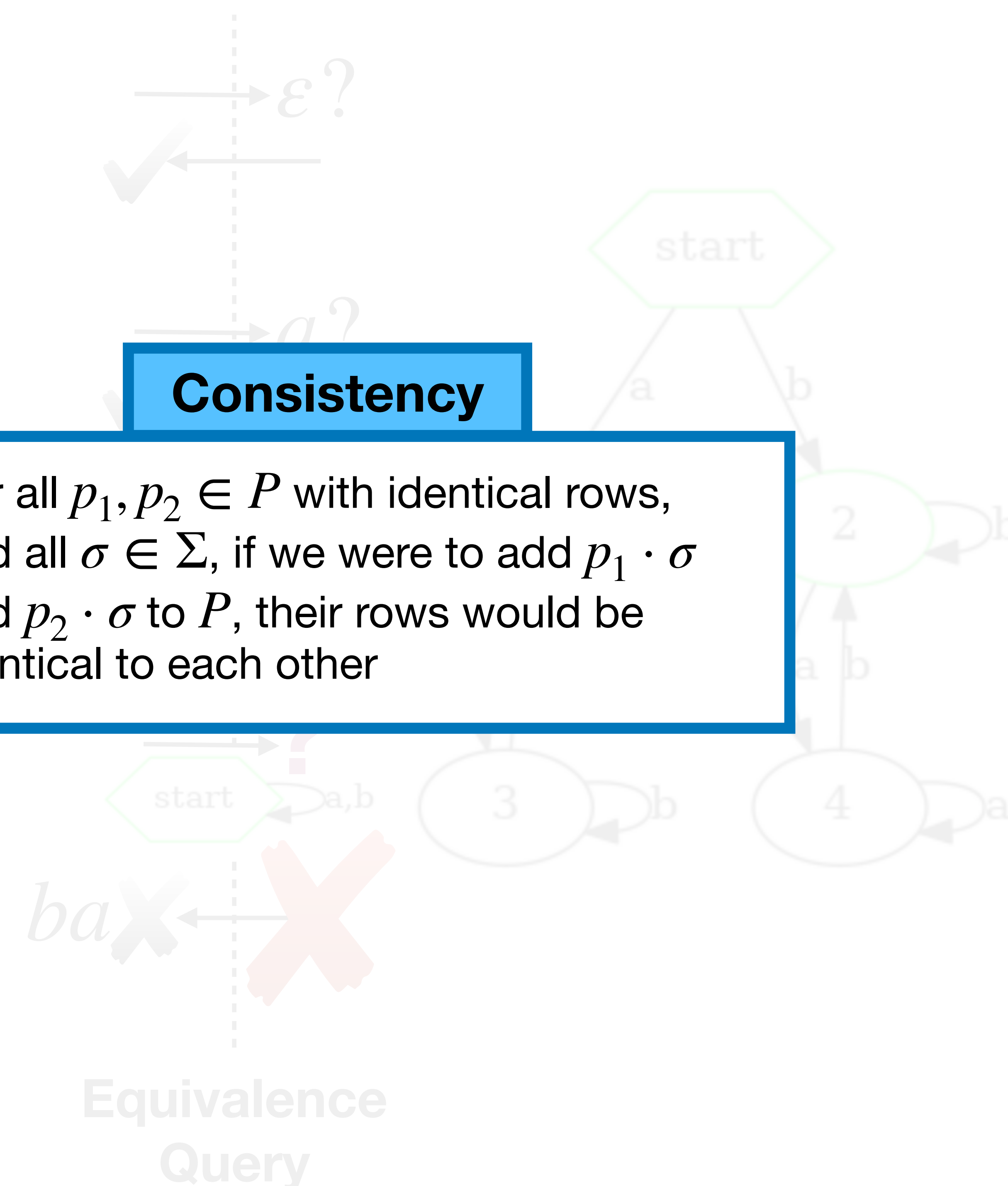


Equivalence Query

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0



Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

Agree

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

Background: L*

The Observation Table

P \ S	ε
ε	**1**
a	**1**
b	**1**
ba	**0**

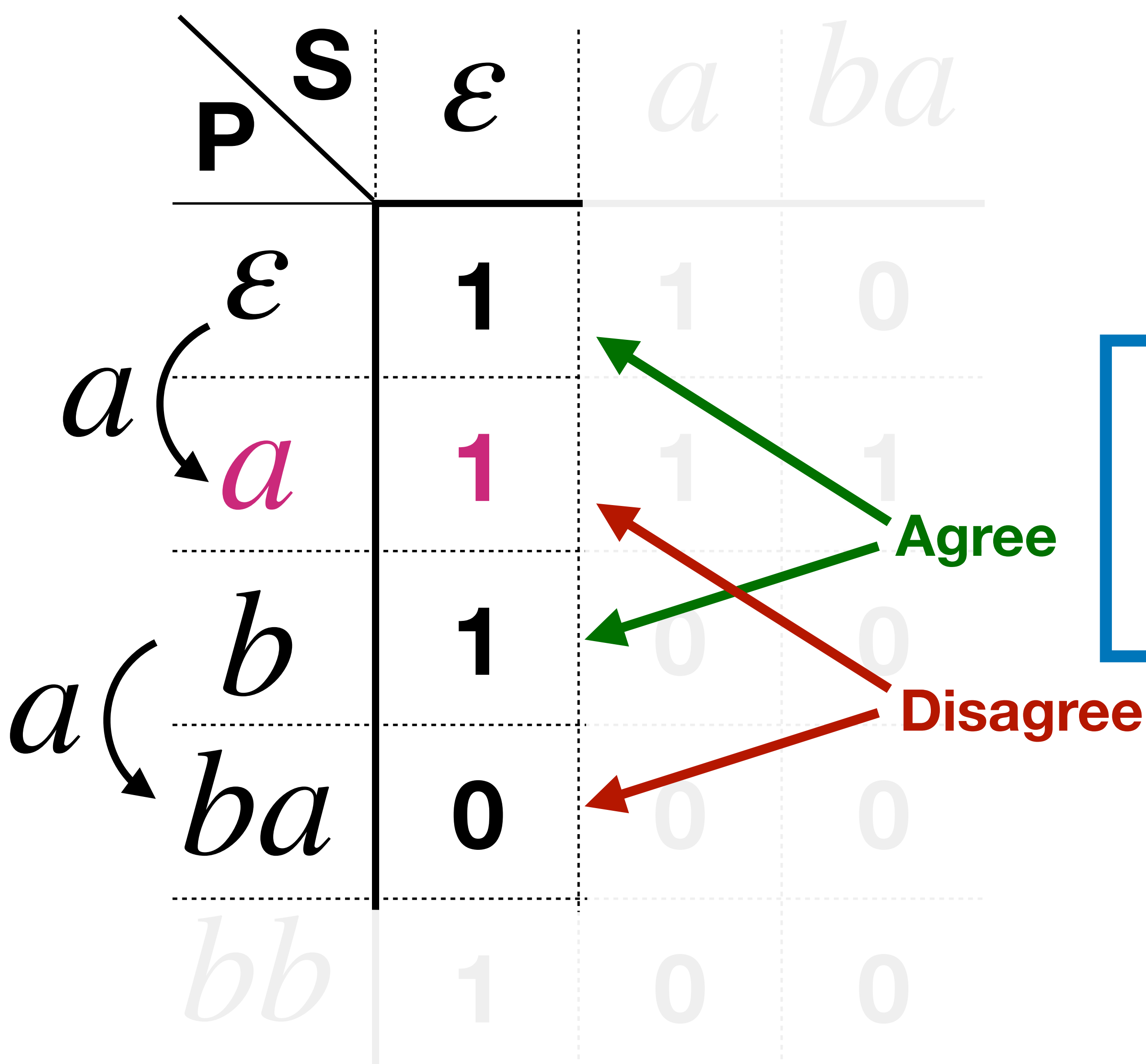
Agree

Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

Background: L*

The Observation Table

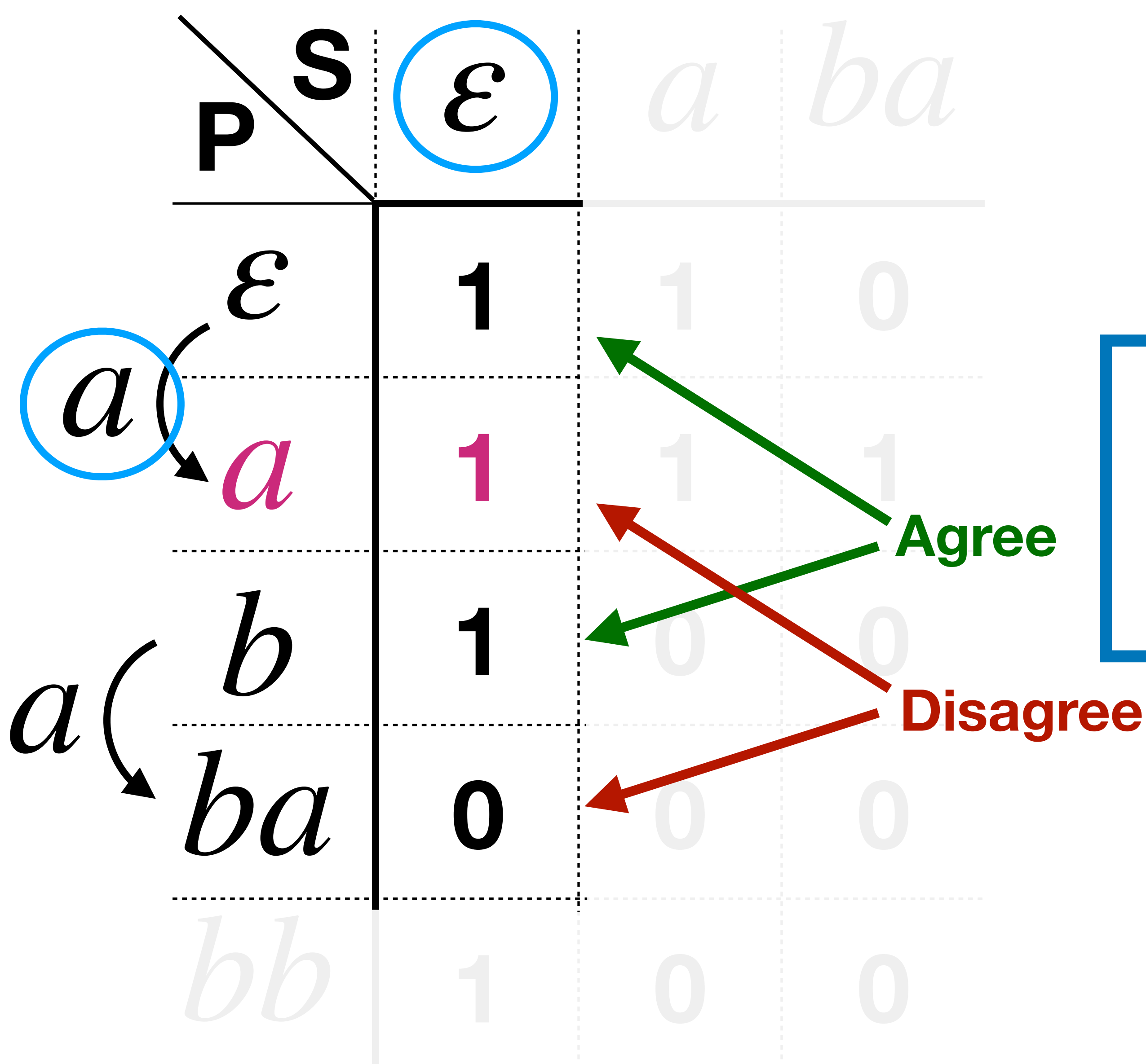


Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

Background: L*

The Observation Table



Consistency

For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

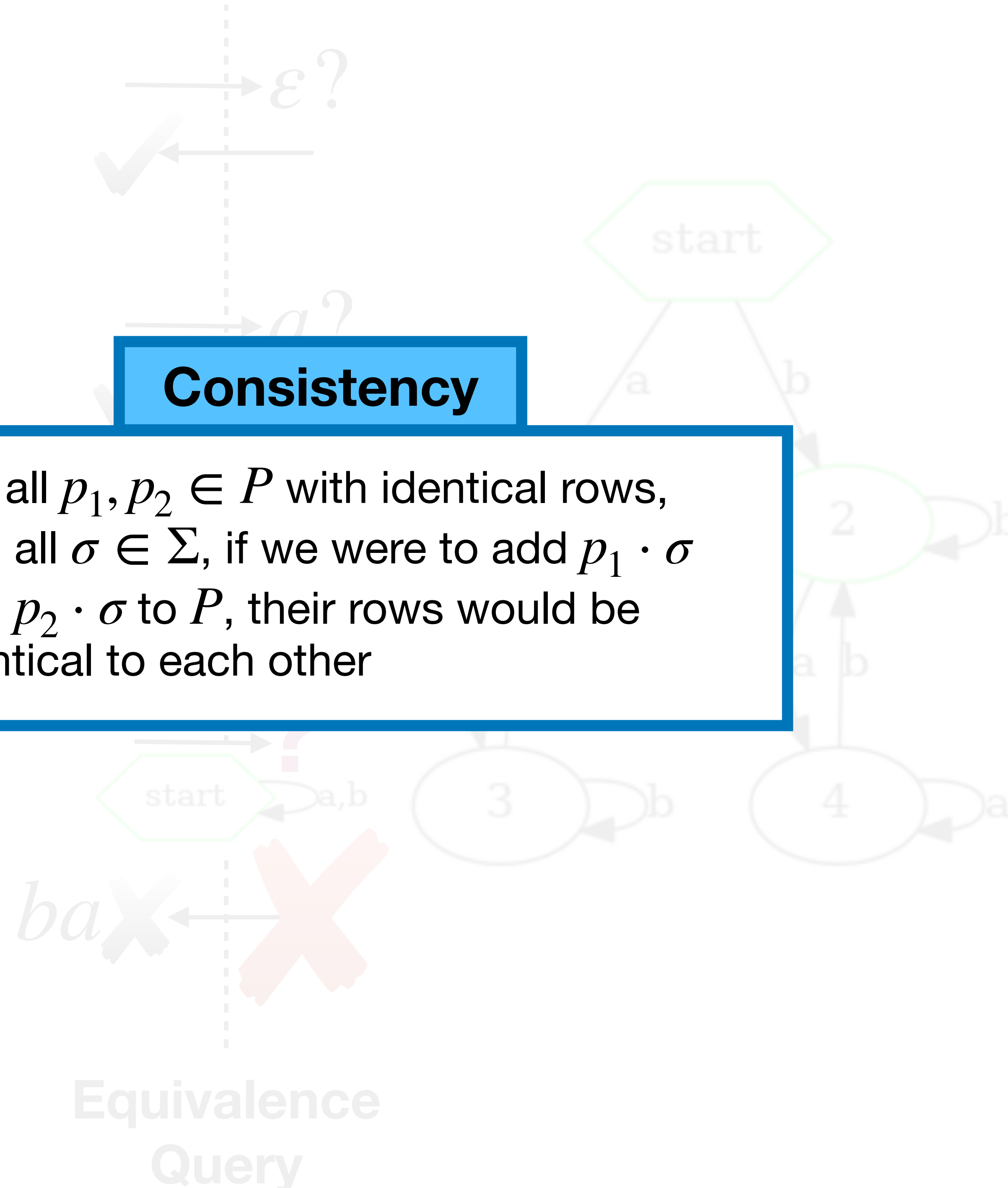
Background: L*

The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

Consistency

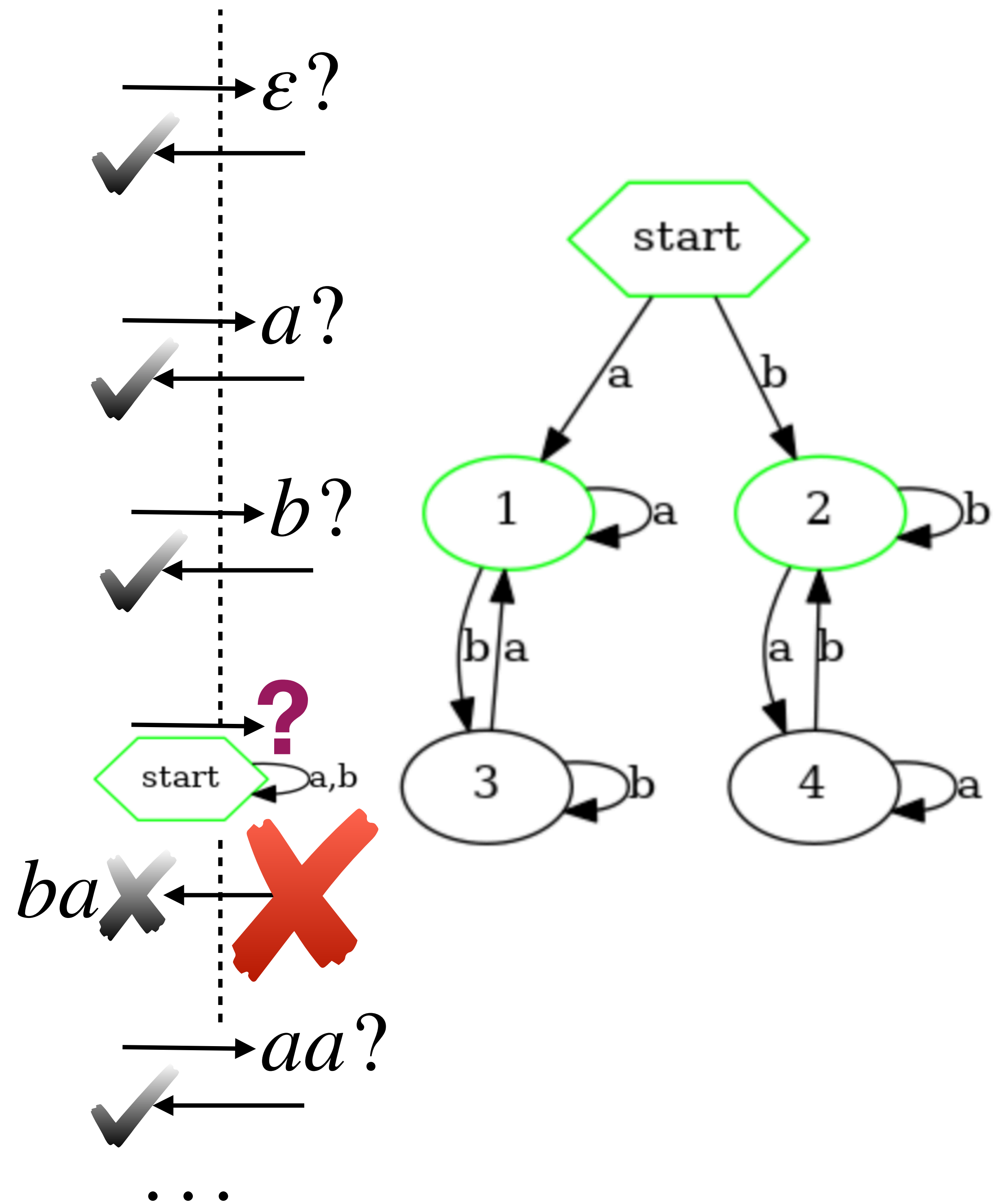
For all $p_1, p_2 \in P$ with identical rows, and all $\sigma \in \Sigma$, if we were to add $p_1 \cdot \sigma$ and $p_2 \cdot \sigma$ to P, their rows would be identical to each other

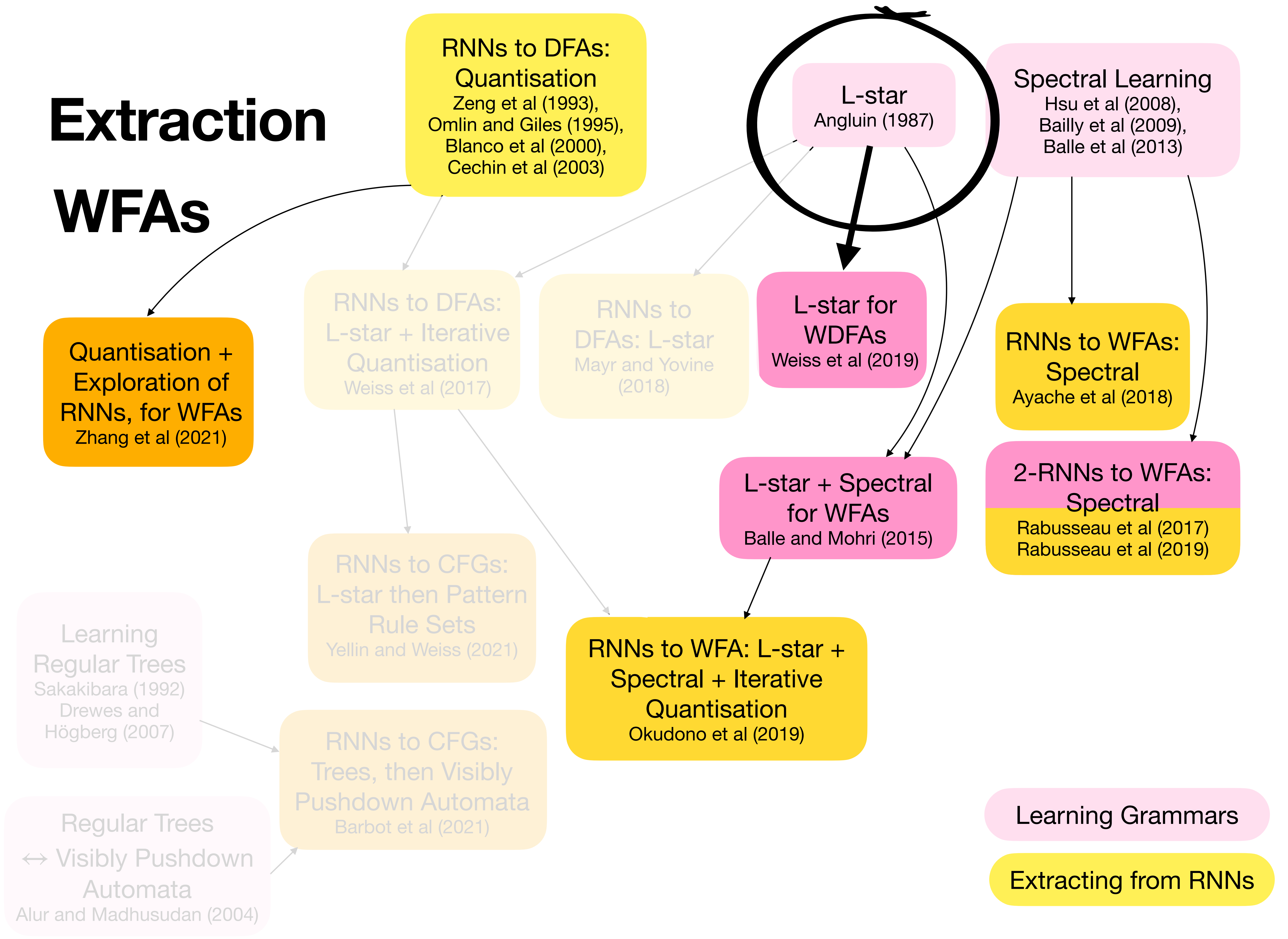


Background: L*

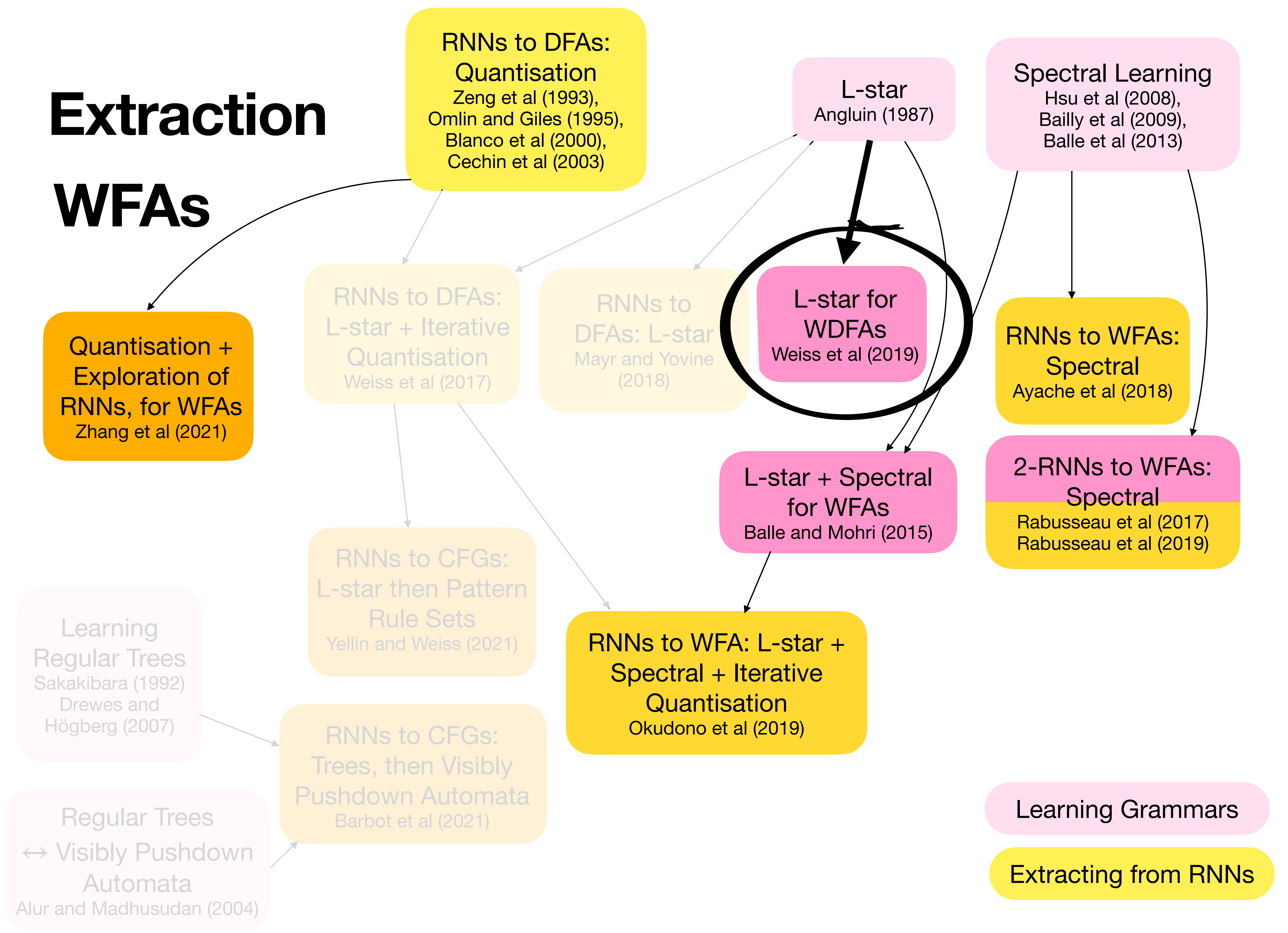
The Observation Table

P \ S	ε	a	ba
ε	1	1	0
a	1	1	1
b	1	0	0
ba	0	0	0
bb	1	0	0

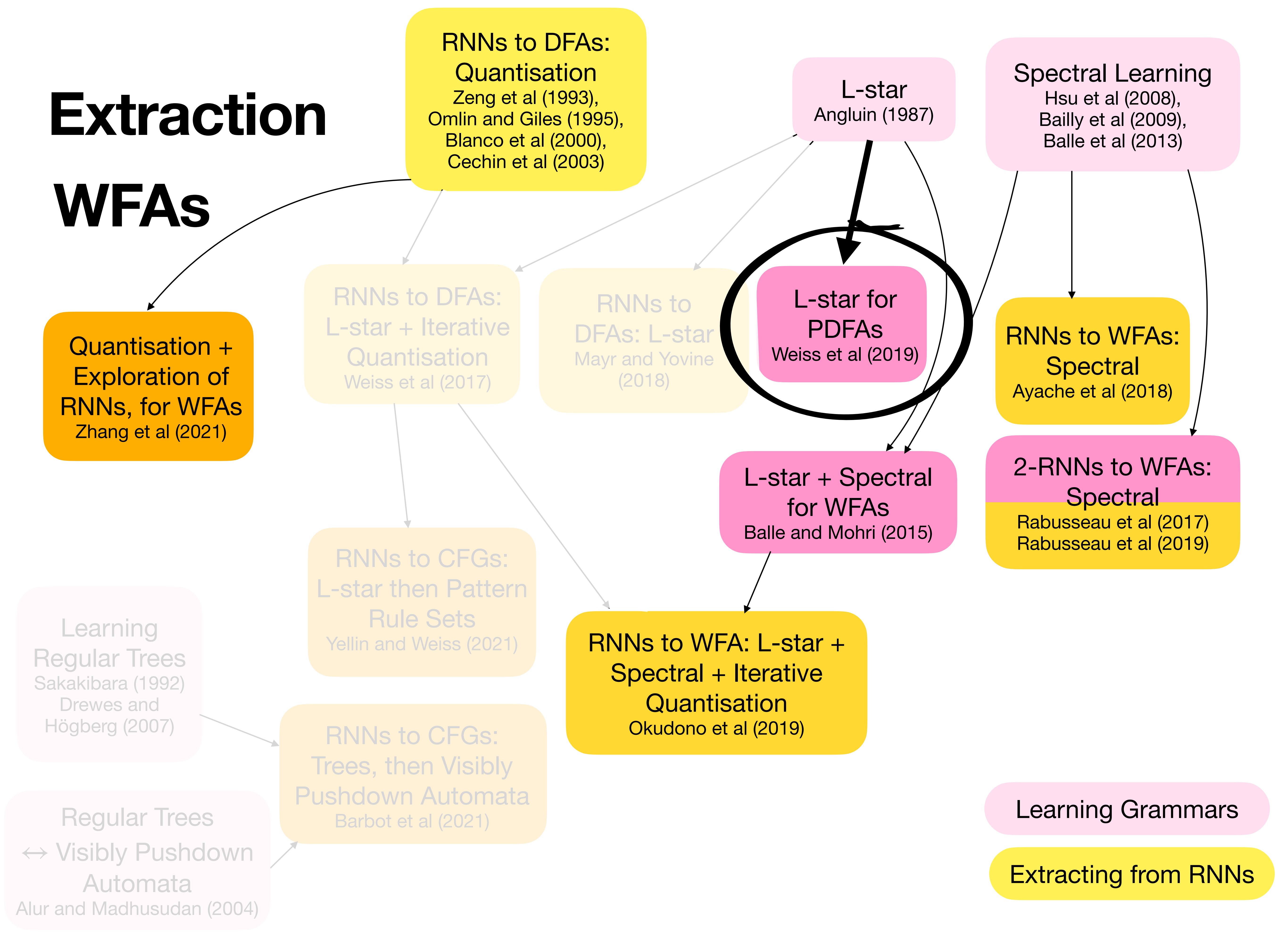




Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs



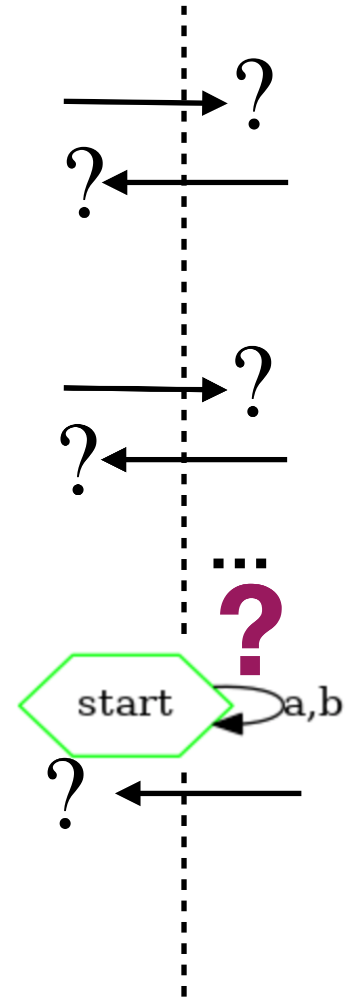
Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs


Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for PDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

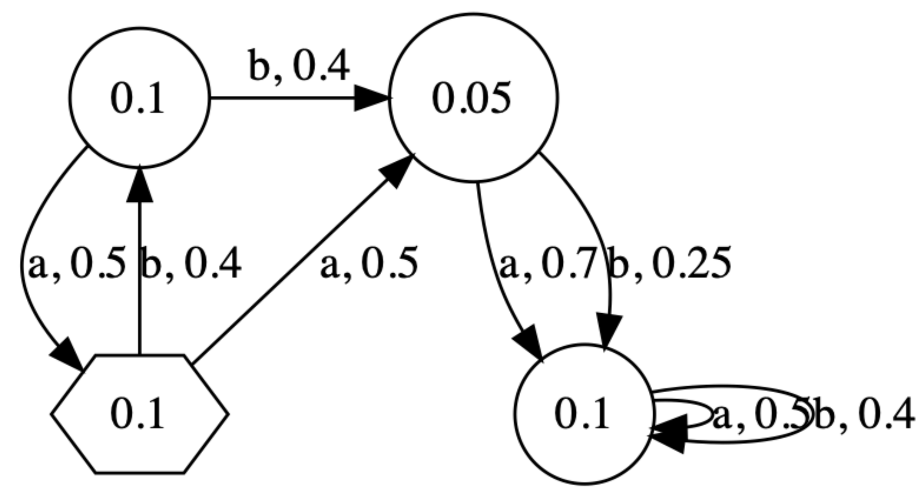
Adapting L*

The Observation Table

P \ S	ε	a	ba
ε	?	?	?
a	?	?	?
b	?	?	?
ba	?	?	?
bb	?	?	?



RNN, trained on



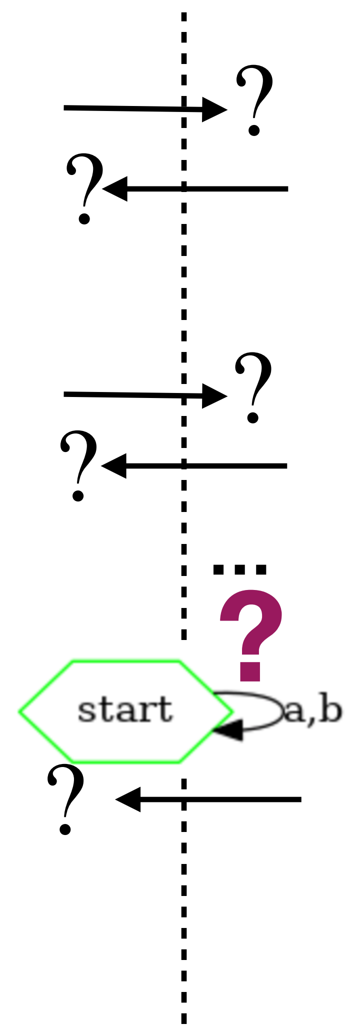
there may be some noise...

What shall we put in the table?

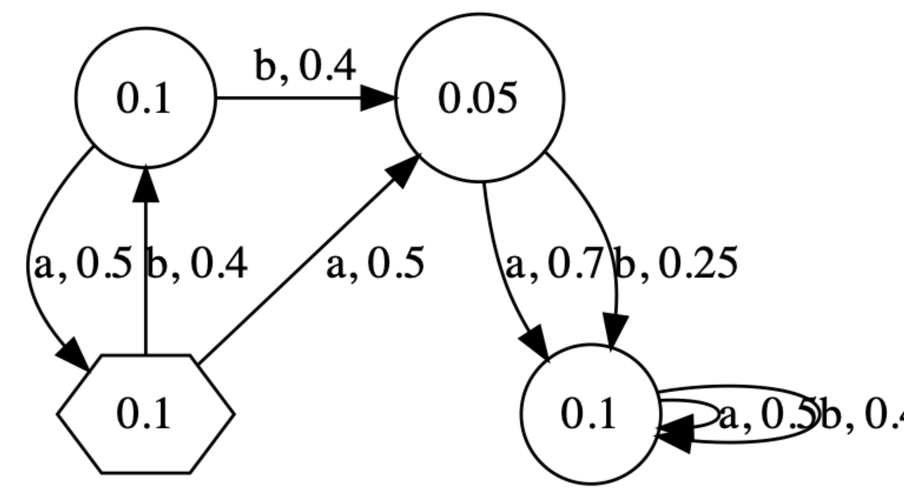
Adapting L*

The Observation Table

P \ S	ε	a	ba
ε	?	?	?
a	?	?	?
b	?	?	?
ba	?	?	?
bb	?	?	?



RNN, trained on



there may be some noise…

What shall we put in the table?

Direct approach: Full sequence weight

Flaw: Will quickly degrade

Intuition: Conditional probabilities

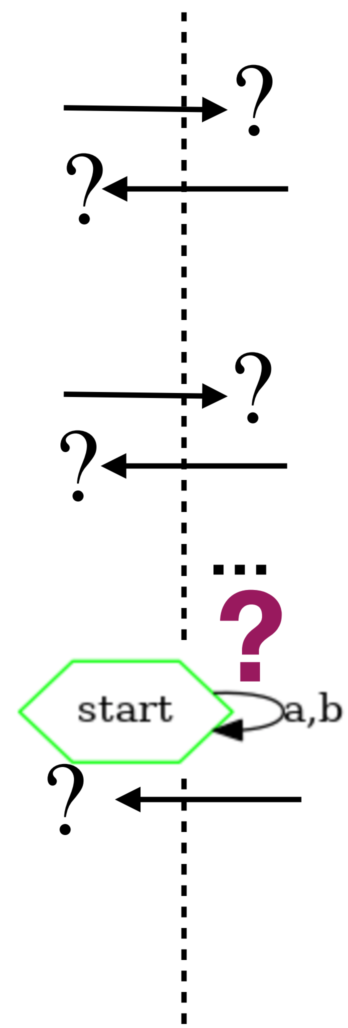
Flaw: Also degrade as S grows

Fix: Last token probabilities

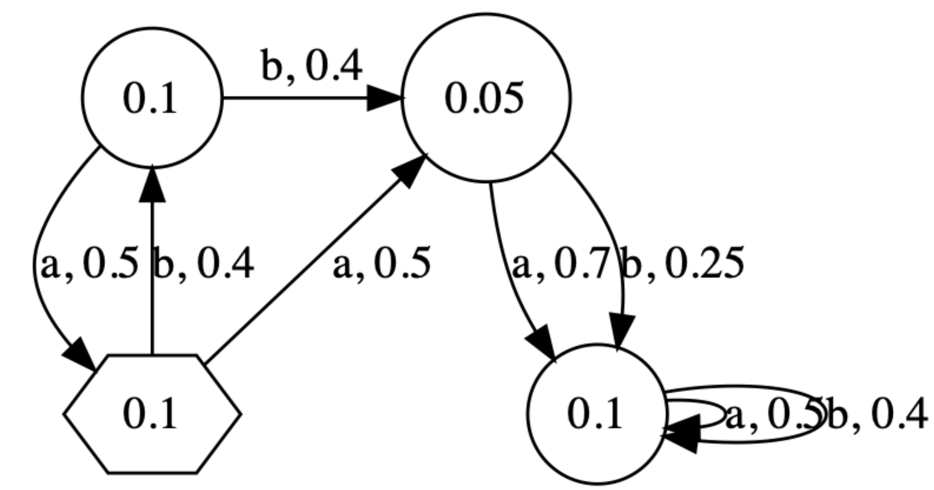
Adapting L*

The Observation Table

P \ S	ε	a	ba
ε	?	?	?
a	?	?	?
b	?	?	?
ba	?	?	?
bb	?	?	?



RNN, trained on



there may be some noise…

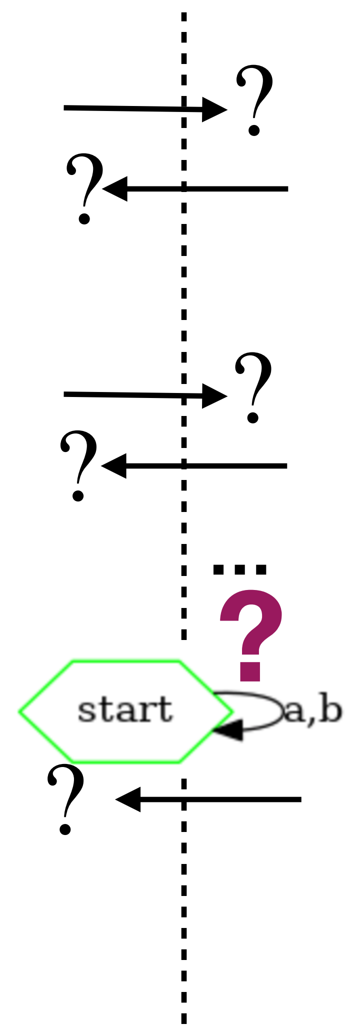
What shall we put in the table?

Final Choice: Last Token Probabilities

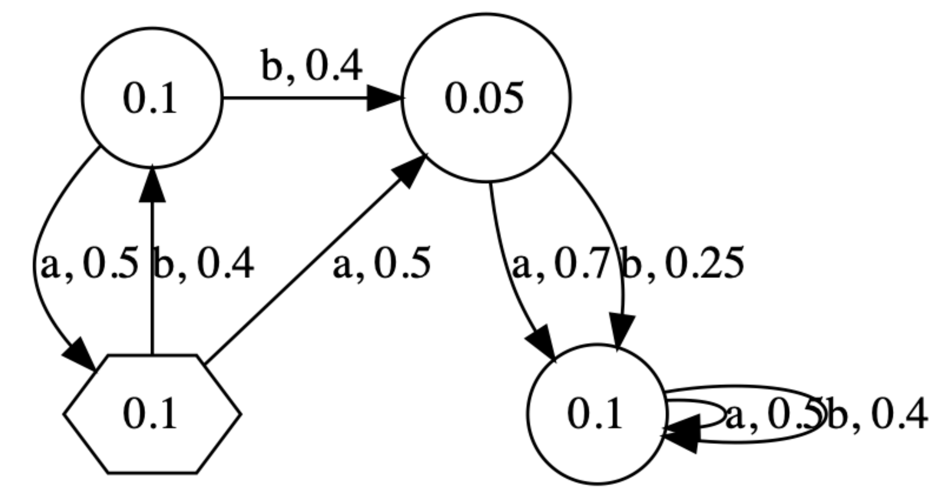
Adapting L*

The Observation Table

P \ S	ε	a	ba
ε	?	0.5	0.4
a	?	0.7	0.5
b	?	0.5	0.7
ba	?	0.5	0.5
bb	?	0.7	0.5



RNN, trained on



there may be some noise…

What shall we put in the table?

Final Choice: Last Token Probabilities

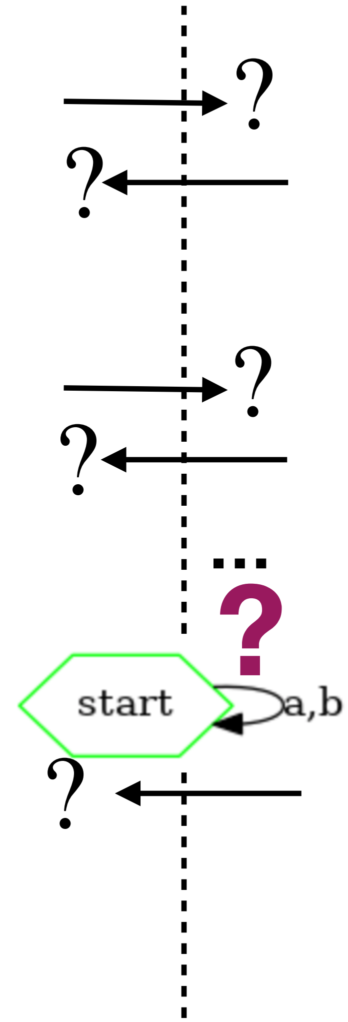
Realisation:

empty suffix doesn't mean anything anymore…

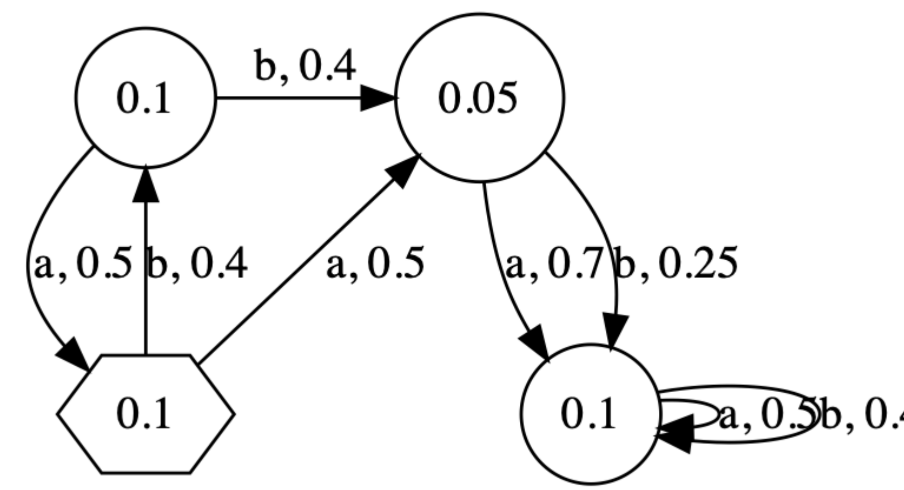
Adapting L*

The Observation Table

P \ S	$	a	ba
ε	0.1	0.5	0.4
a	0.05	0.7	0.5
b	0.1	0.5	0.7
ba	0.1	0.5	0.5
bb	0.05	0.7	0.5



RNN, trained on



there may be some noise…

What shall we put in the table?

Final Choice: Last Token Probabilities

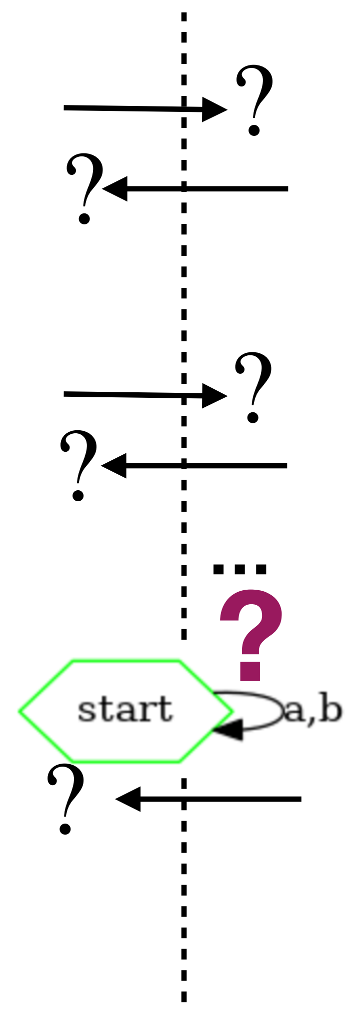
Realisation:

empty suffix doesn’t mean anything anymore…
but end-of-sequence does

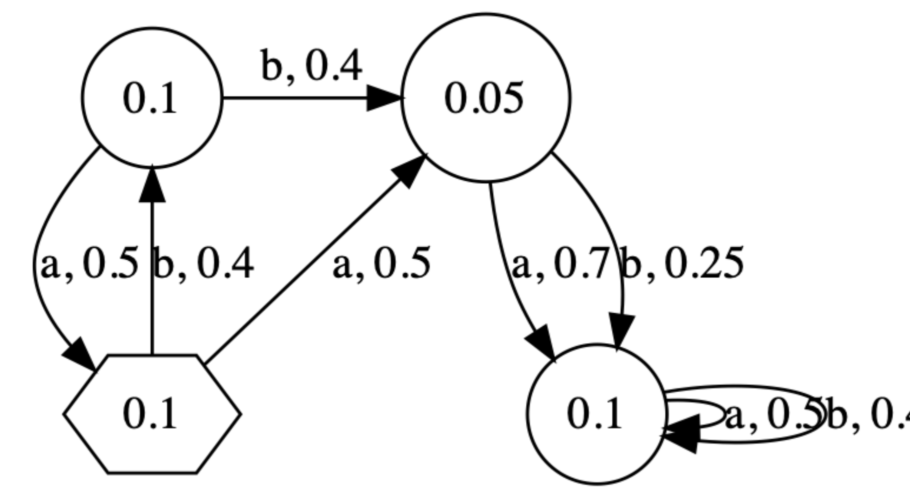
Adapting L*

The Observation Table

P \ S	$	a	ba
ε	0.1	0.5	0.4
a	0.05	0.7	0.5
b	0.1	0.5	0.7
ba	0.1	0.5	0.5
bb	0.05	0.7	0.5



RNN, trained on



there may be some noise...

What shall we put in the table?

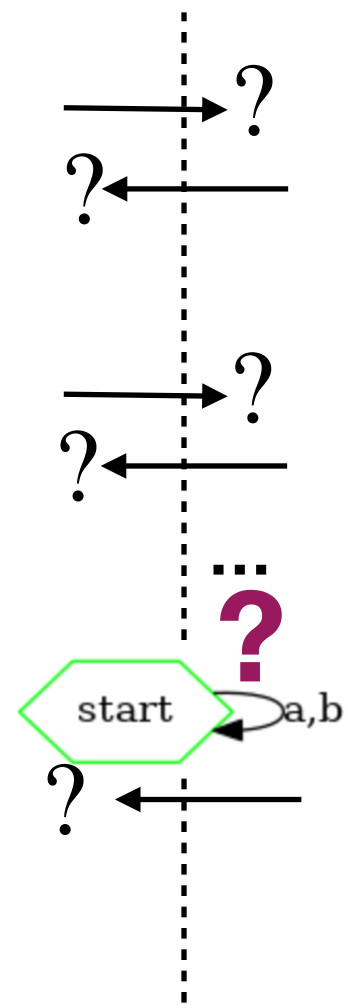
Final Choice: Last Token Probabilities

Okay, we have our adaptation. Let's go!?

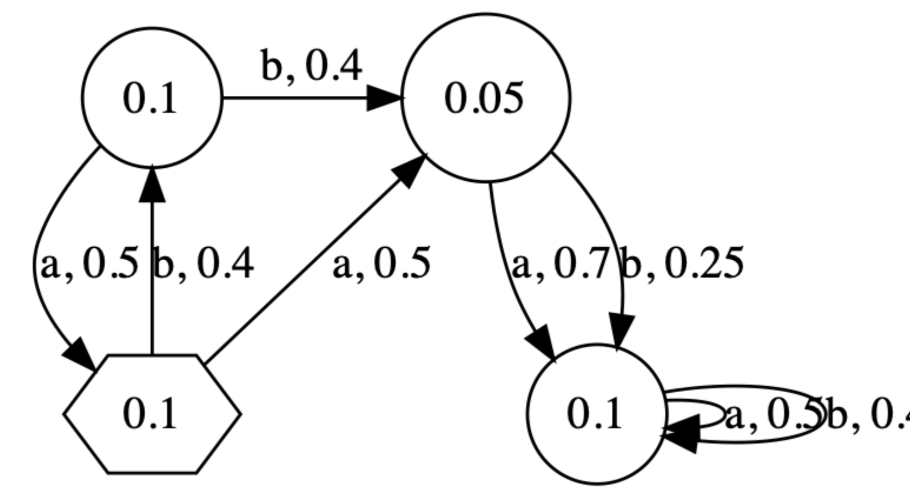
Adapting L*

The Observation Table

P \ S	$	a	ba
ε	0.1	0.5	0.4
a	0.05	0.7	0.5
b	0.1	0.5	0.7
ba	0.1	0.5	0.5
bb	0.05	0.7	0.5



RNN, trained on



there may be some noise…

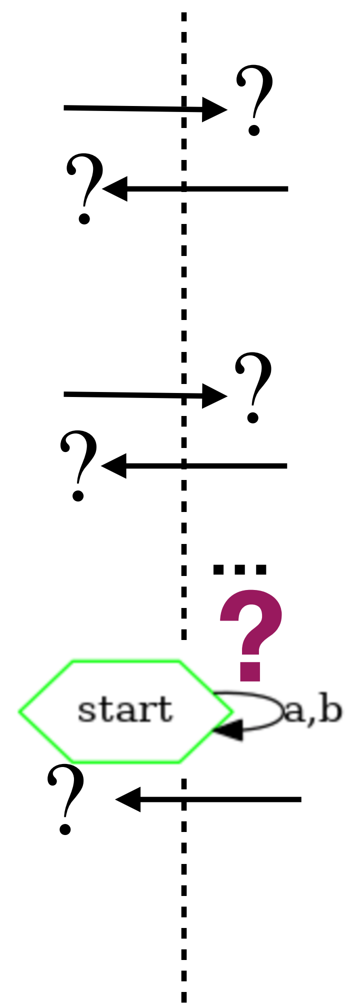
What shall we put in the table?

Final Choice: Last Token Probabilities

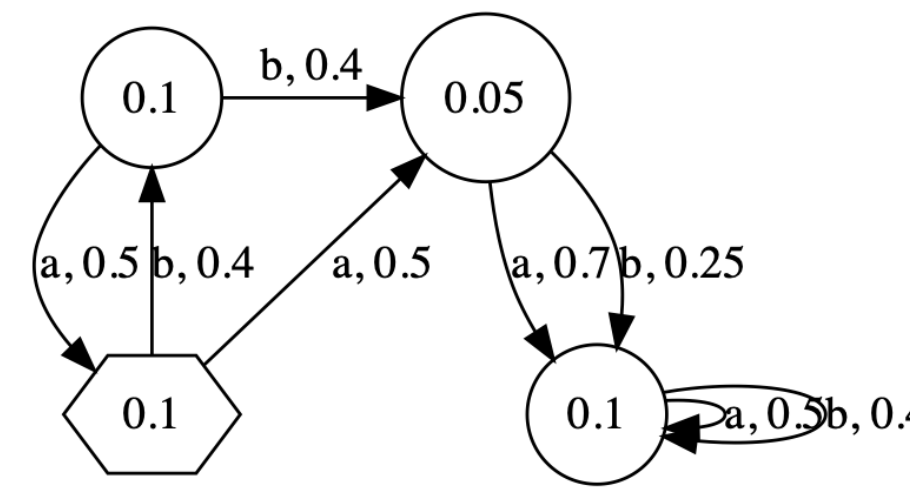
Adapting L*

The Observation Table

P \ S	$	a	ba
ε	0.1	0.5	0.4
a	0.05	0.7	0.5
b	0.1	0.5	0.7
ba	0.1	0.5	0.5
bb	0.05	0.7	0.5



RNN, trained on



there may be some noise…

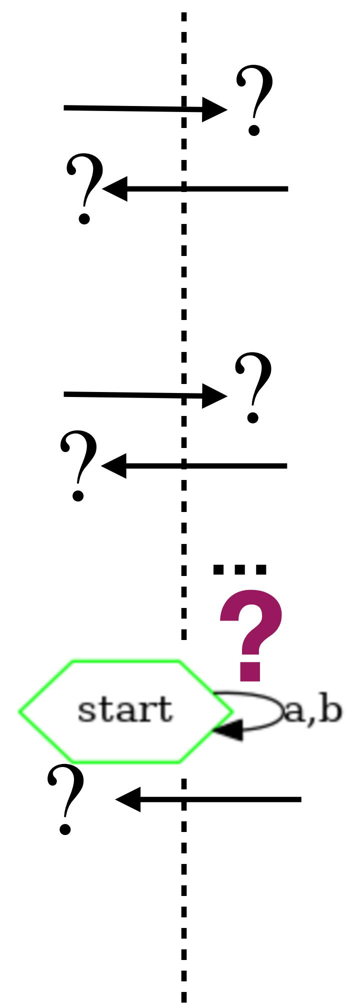
What shall we put in the table?

Final Choice: Last Token Probabilities
Nice Realisation: Can use additive tolerance

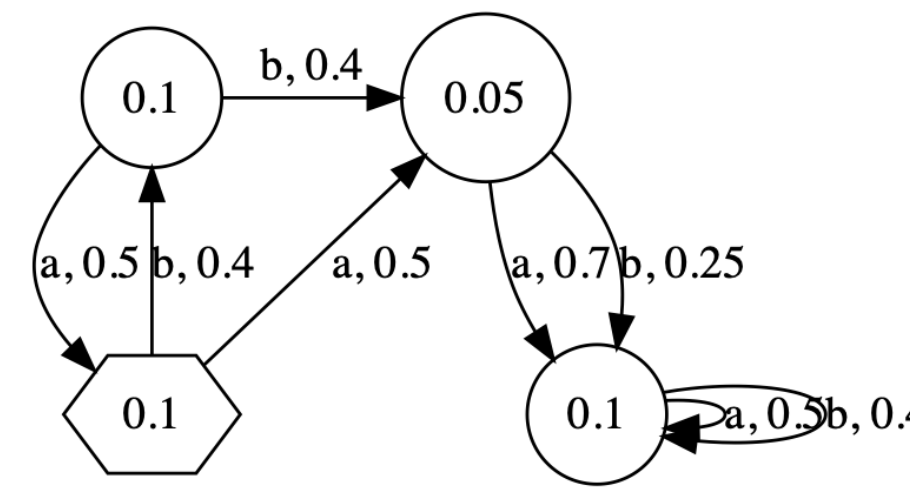
Adapting L*

The Observation Table

P \ S	$	a	ba
ε	0.1	0.5	0.4
a	0.05	0.7	0.5
b	0.1	0.5	0.7
ba	0.1	0.5	0.5
bb	0.05	0.7	0.5



RNN, trained on



there may be some noise…

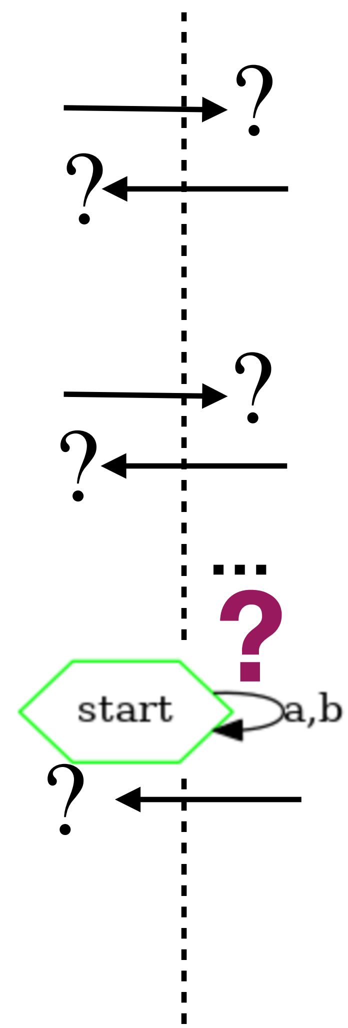
What shall we put in the table?

Final Choice: Last Token Probabilities
Nice Realisation: Can use additive tolerance
Challenge: Non-transitivity of tolerance

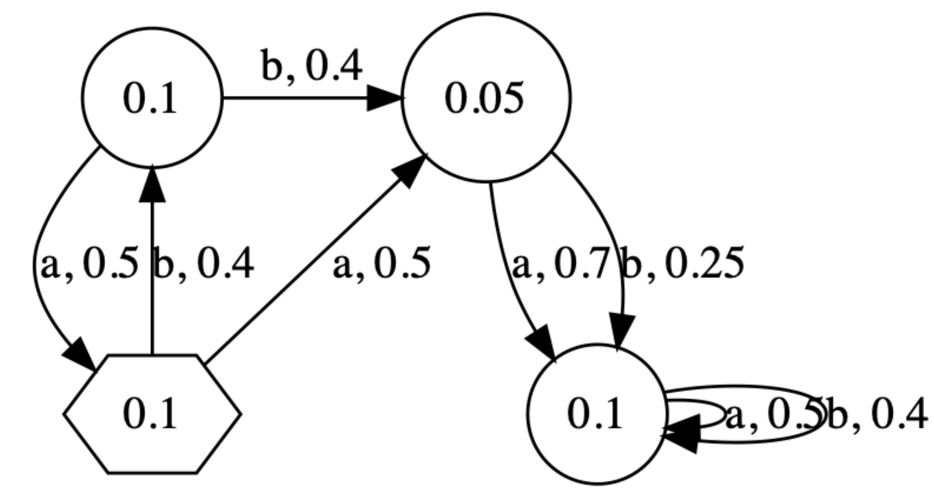
Adapting L*

The Observation Table

P \ S	$	a	ba
ε	0.1	0.5	0.4
a	0.05	0.7	0.5
b	0.1	0.5	0.7
ba	0.1	0.5	0.5
bb	0.05	0.7	0.5



RNN, trained on



there may be some noise…

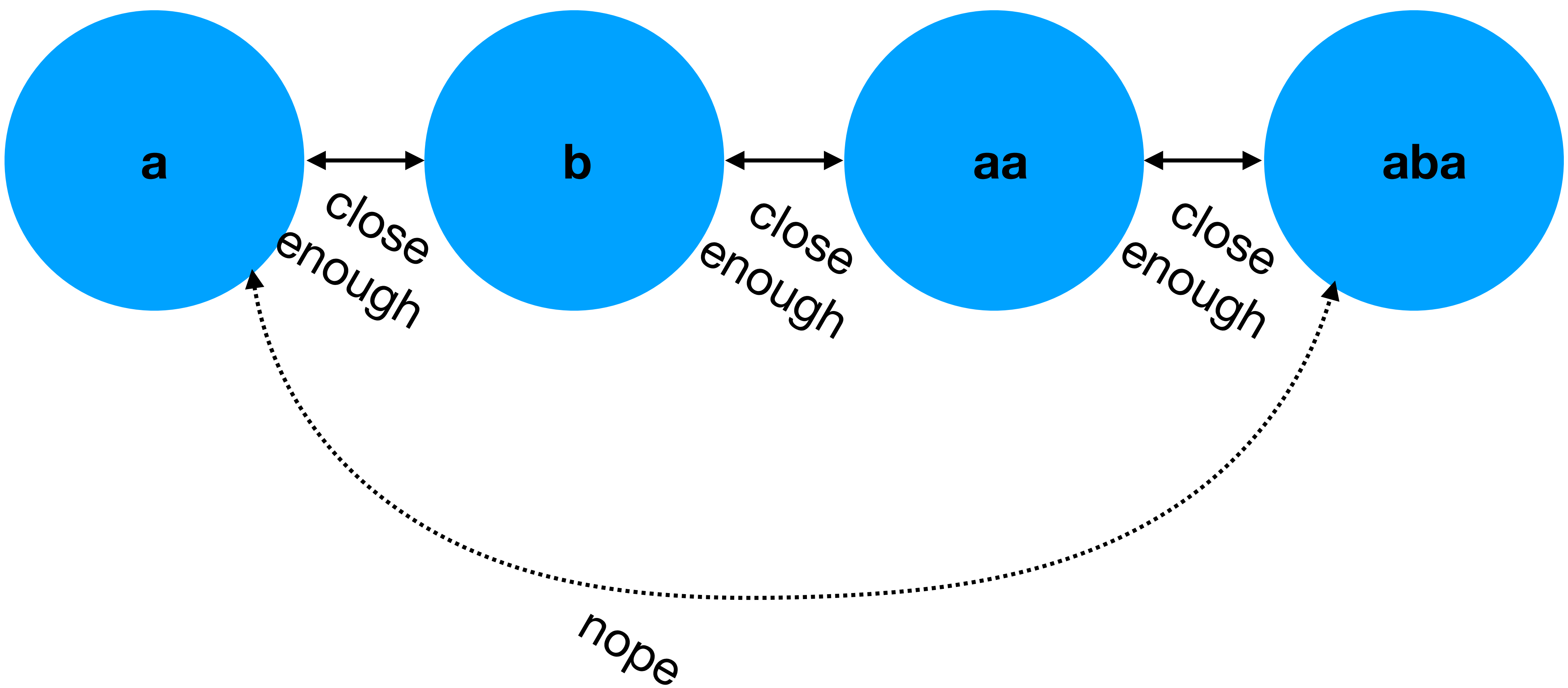
What shall we put in the table?

Final Choice: Last Token Probabilities
Nice Realisation: Can use additive tolerance
Challenge: Non-transitivity of tolerance

Adapting L*

Dealing with the Additive Tolerance

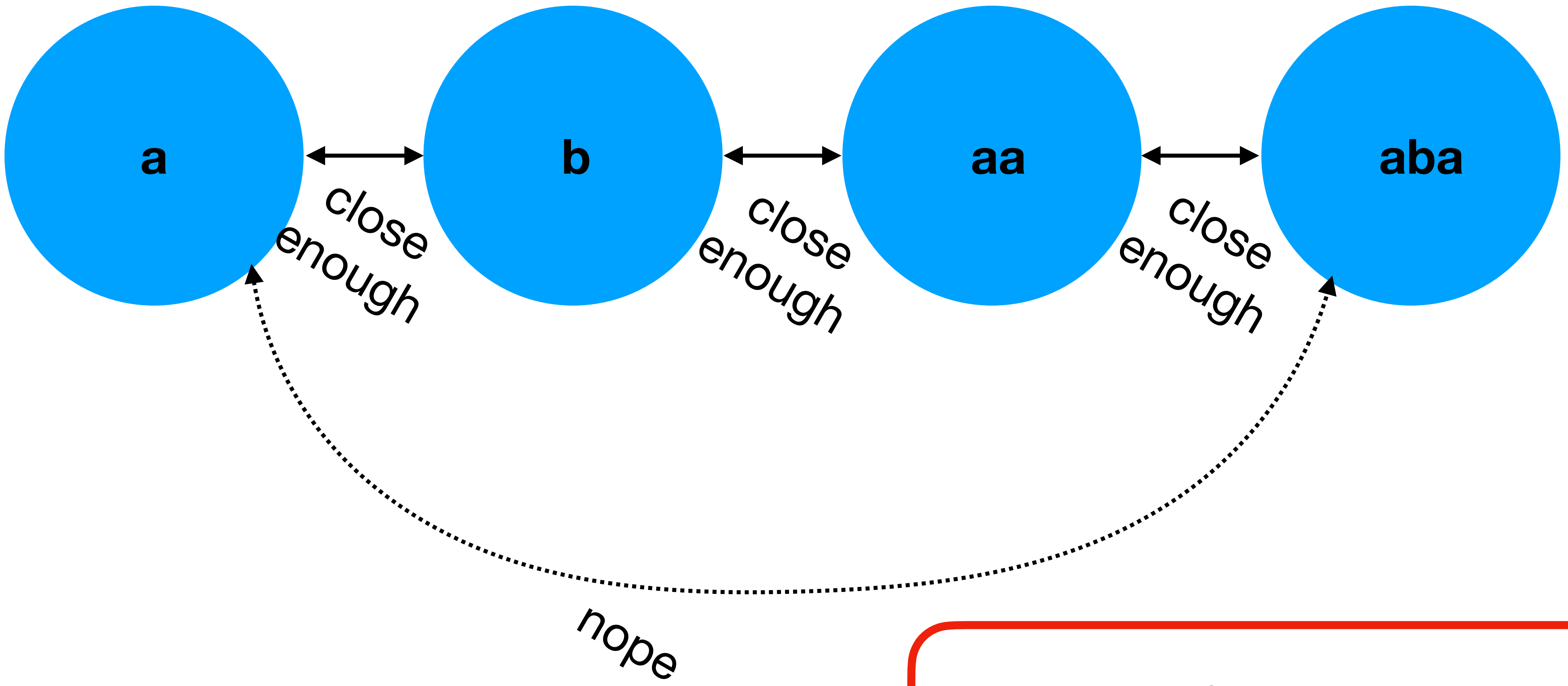
In particular: dealing with 'chains' of similar prefixes



Adapting L*

Dealing with the Additive Tolerance

In particular: dealing with ‘chains’ of similar prefixes



How can we fix the “**closedness**” and “**consistency**” definitions, to avoid mistaken groupings?

Adapting L*

Dealing with the Additive Tolerance

In particular: dealing with ‘chains’ of similar prefixes

Immediate realisation: Attempting to fix definitions for table is painful

Adapting L*

Dealing with the Additive Tolerance

In particular: dealing with ‘chains’ of similar prefixes

Immediate realisation: Attempting to fix definitions for table is painful

Solution:

Fill table optimistically, and fix problems post-hoc

Adapting L*

Optimistic Table and Post-Hoc Fixes

Adapting L*

Optimistic Table and Post-Hoc Fixes

1. **Check closedness as normal**,
   just with the additive tolerance

2. **Check consistency as normal**,
   just with the additive tolerance

3. **Make hypothesis with caution!**

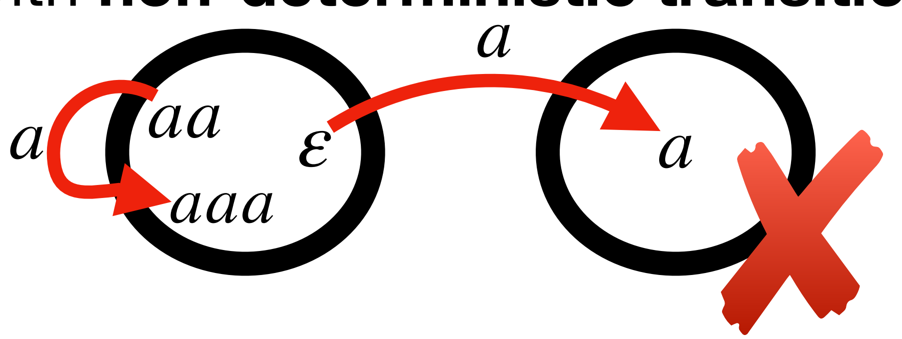
Adapting L*

Optimistic Table and Post-Hoc Fixes

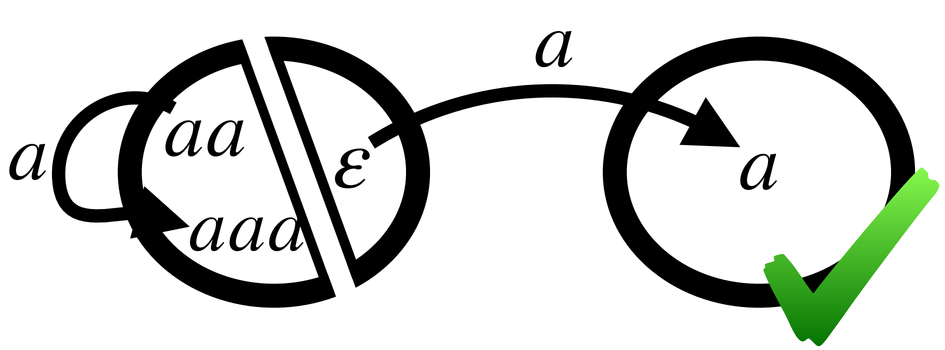
3. Make hypothesis with caution!

Potential Problems:

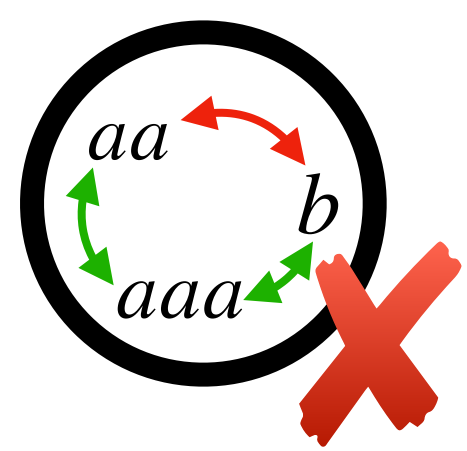
1. Clustering of prefixes causes states with **non-deterministic transitions**



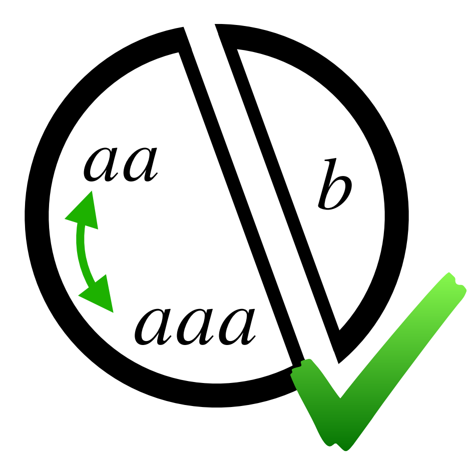
post hoc fix: refine



2. Clustering of prefixes creates states with prefixes **beyond threshold** of each other



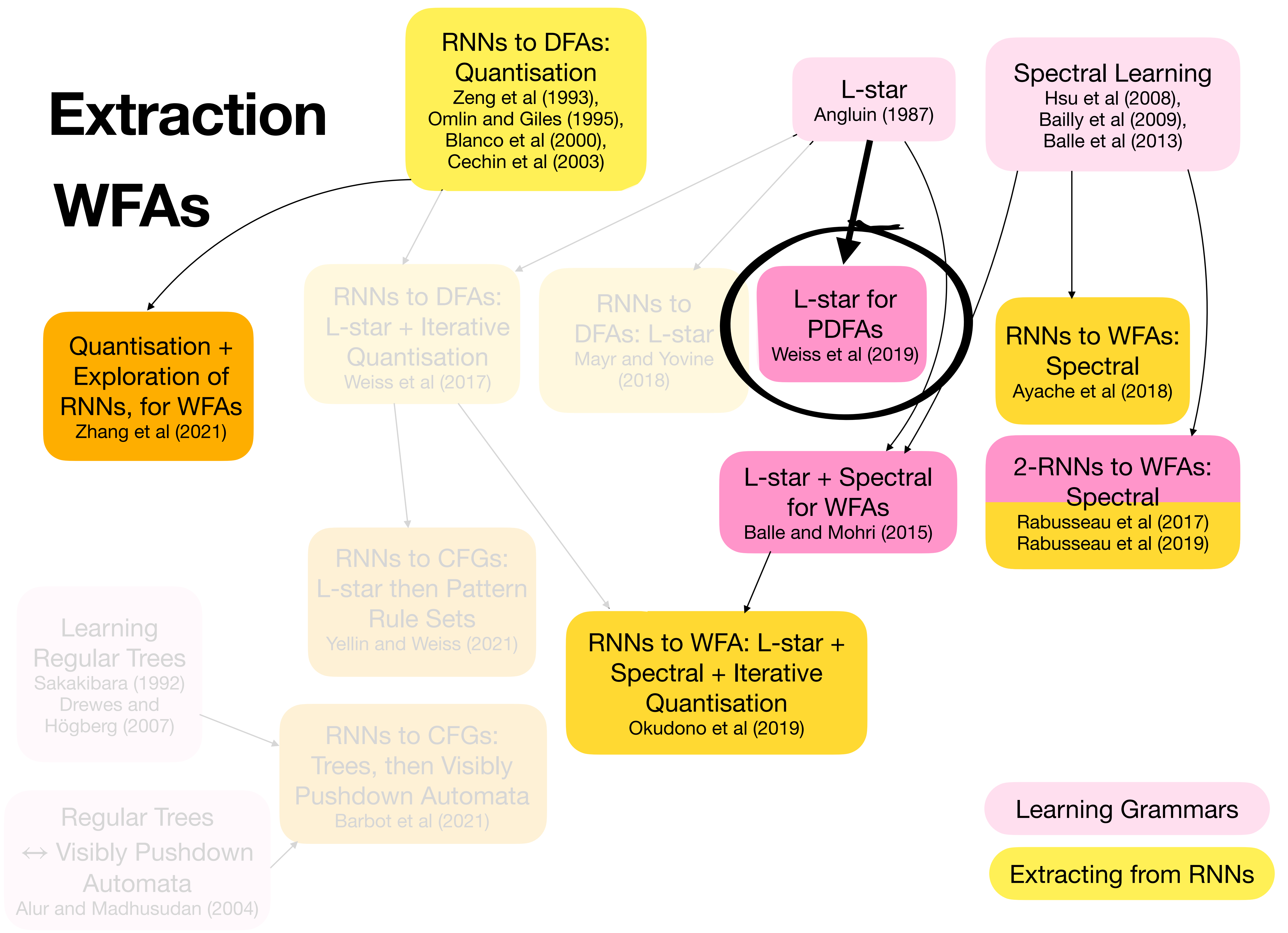
post hoc fix: refine

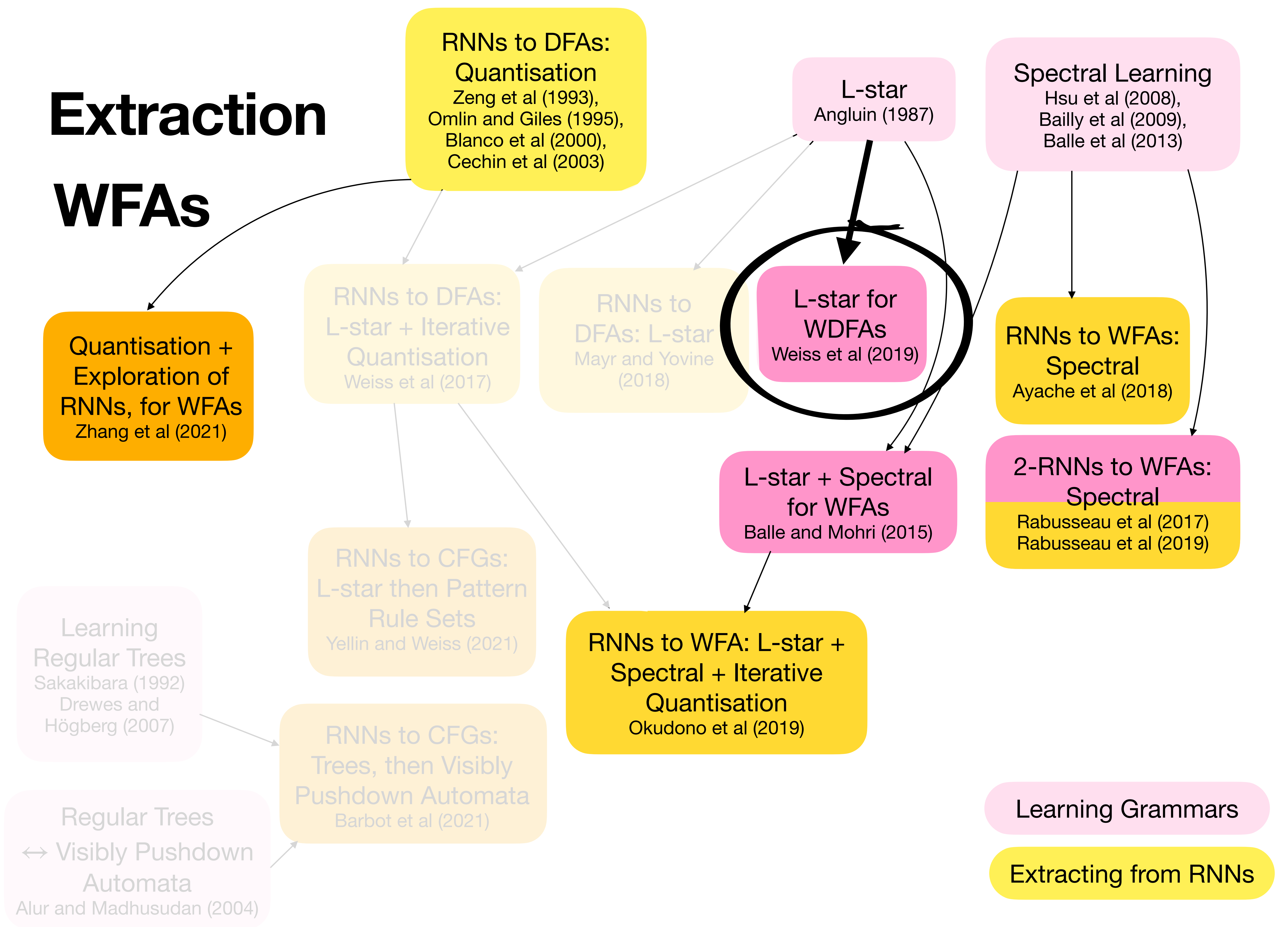


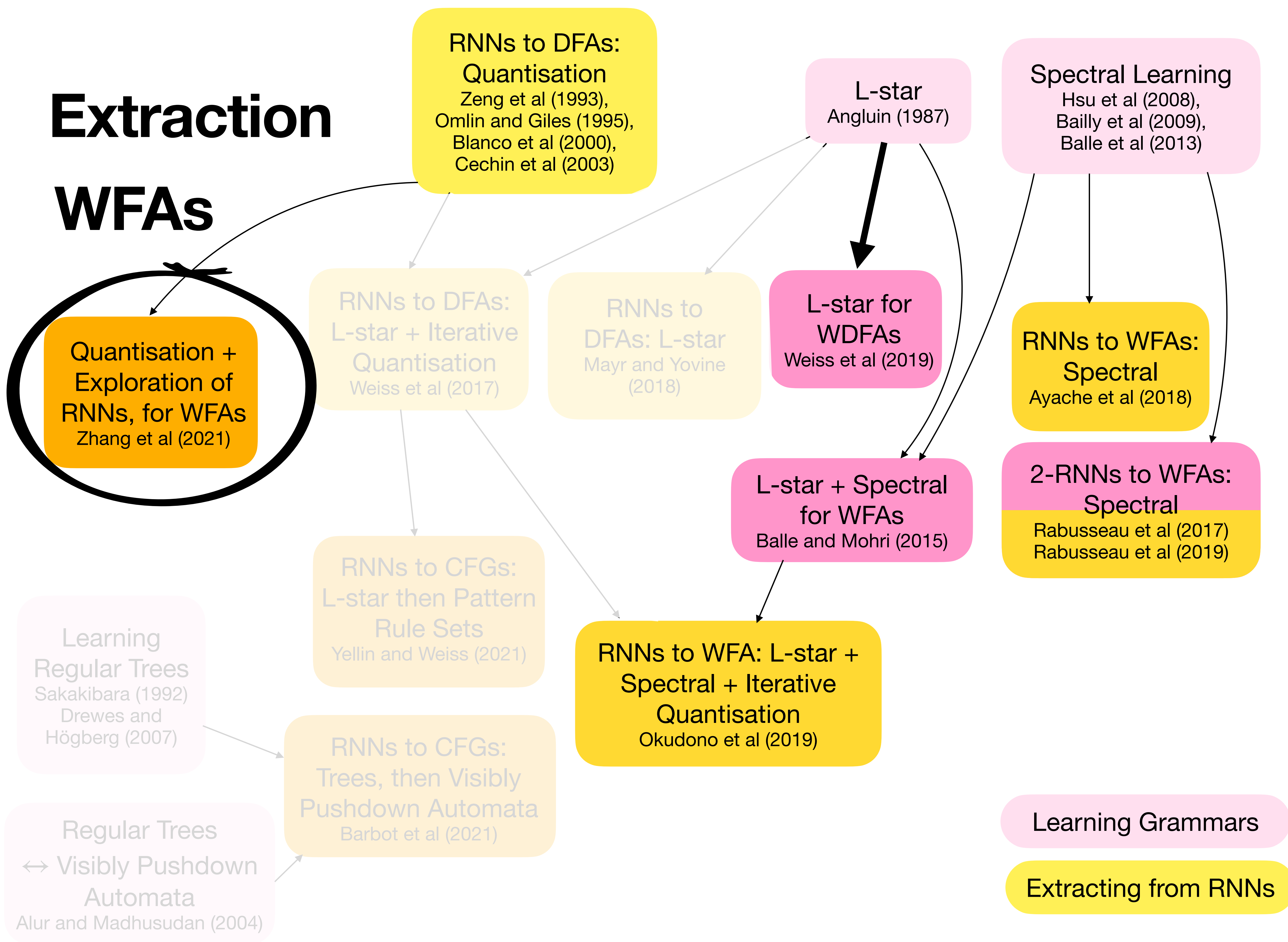
Anytime Stopping

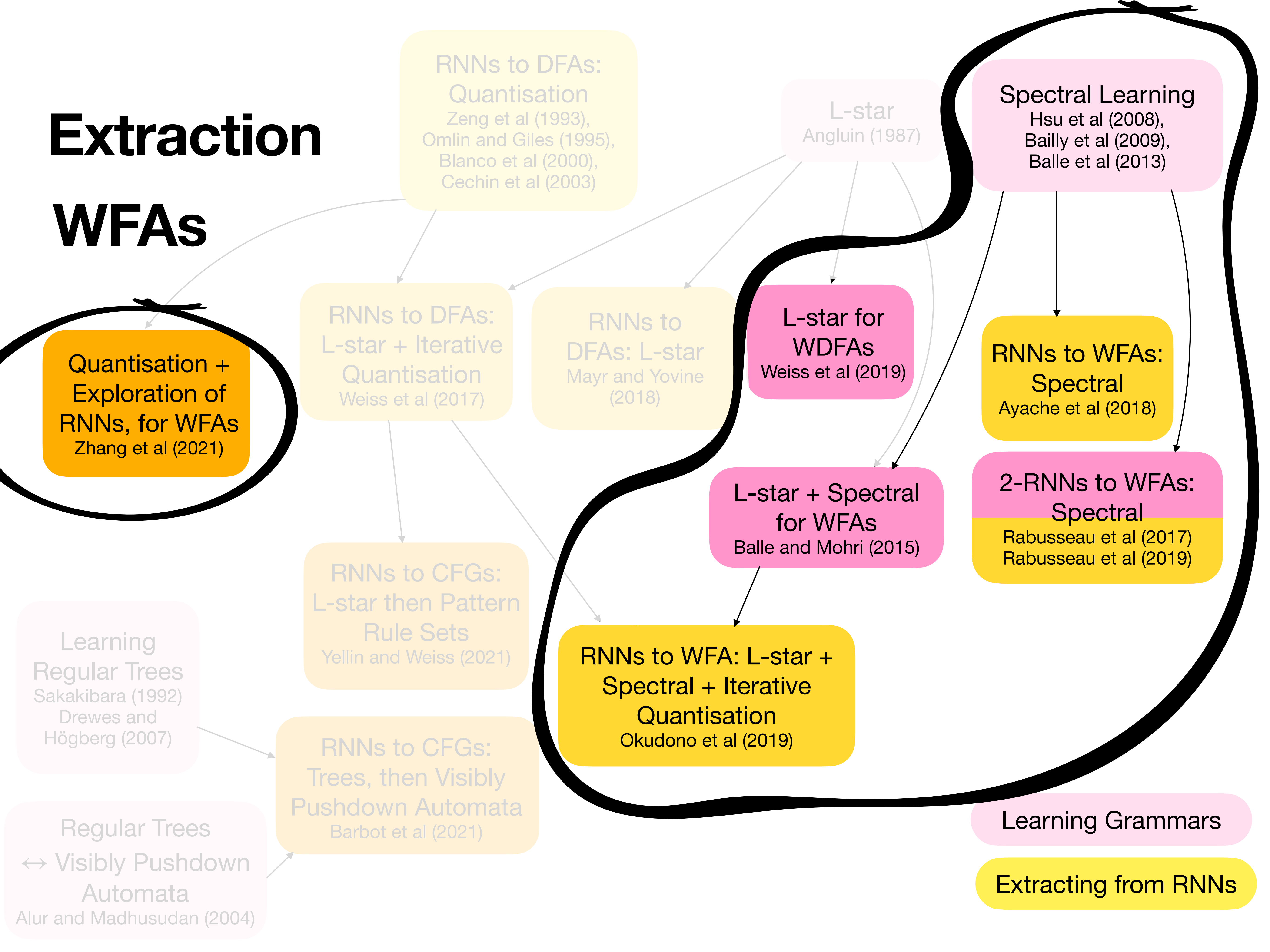
This algorithm is unlikely to complete on real-world tasks. Thus, we allow anytime stopping:

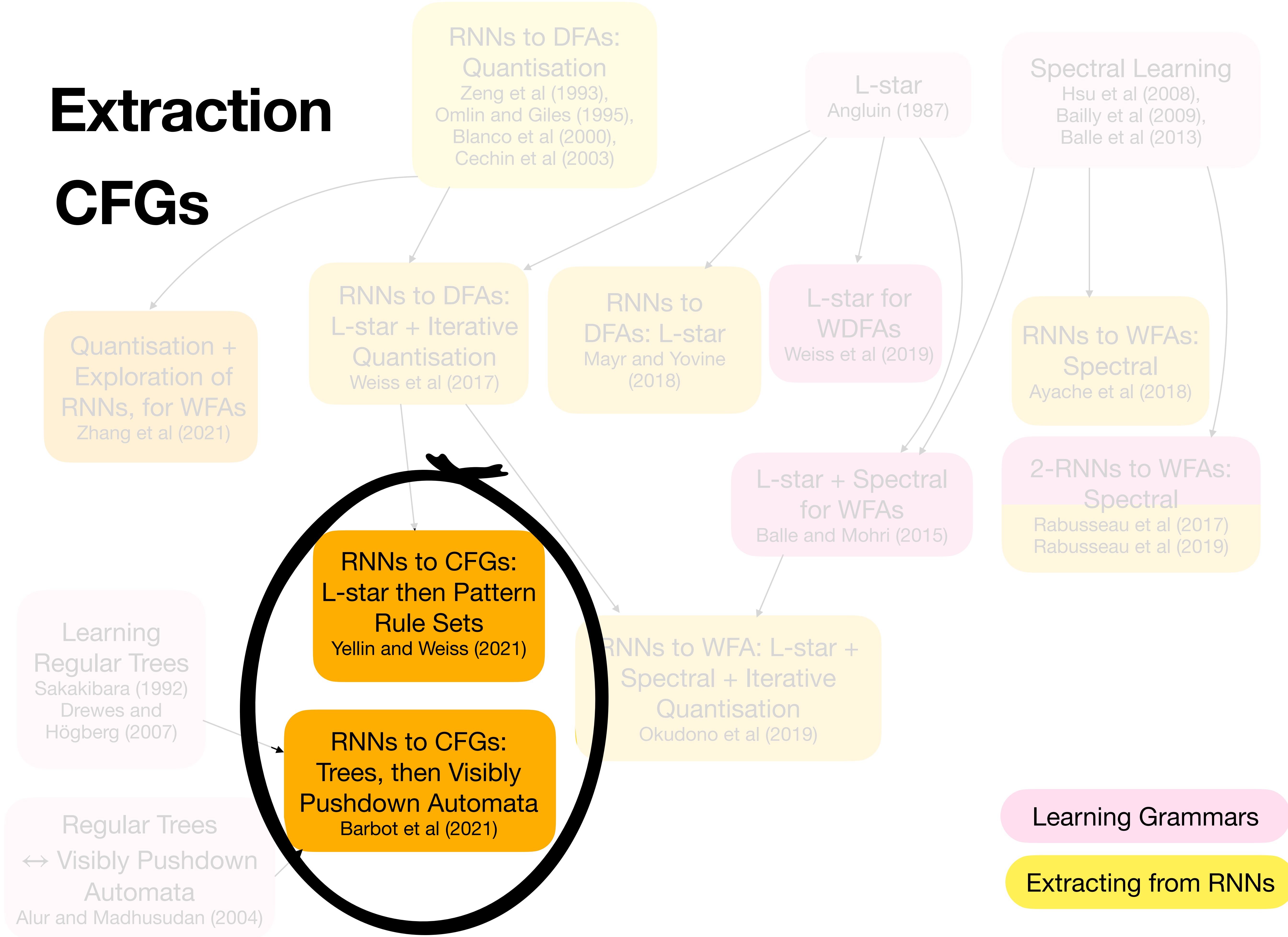
- Prioritise high-weight prefixes
- Avoid very low-weight separating suffixes
- On stop, map remaining prefixes to best match
  - This is actually quite slow, and might not be very beneficial *(needs to be tested!)*


Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for PDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs


Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

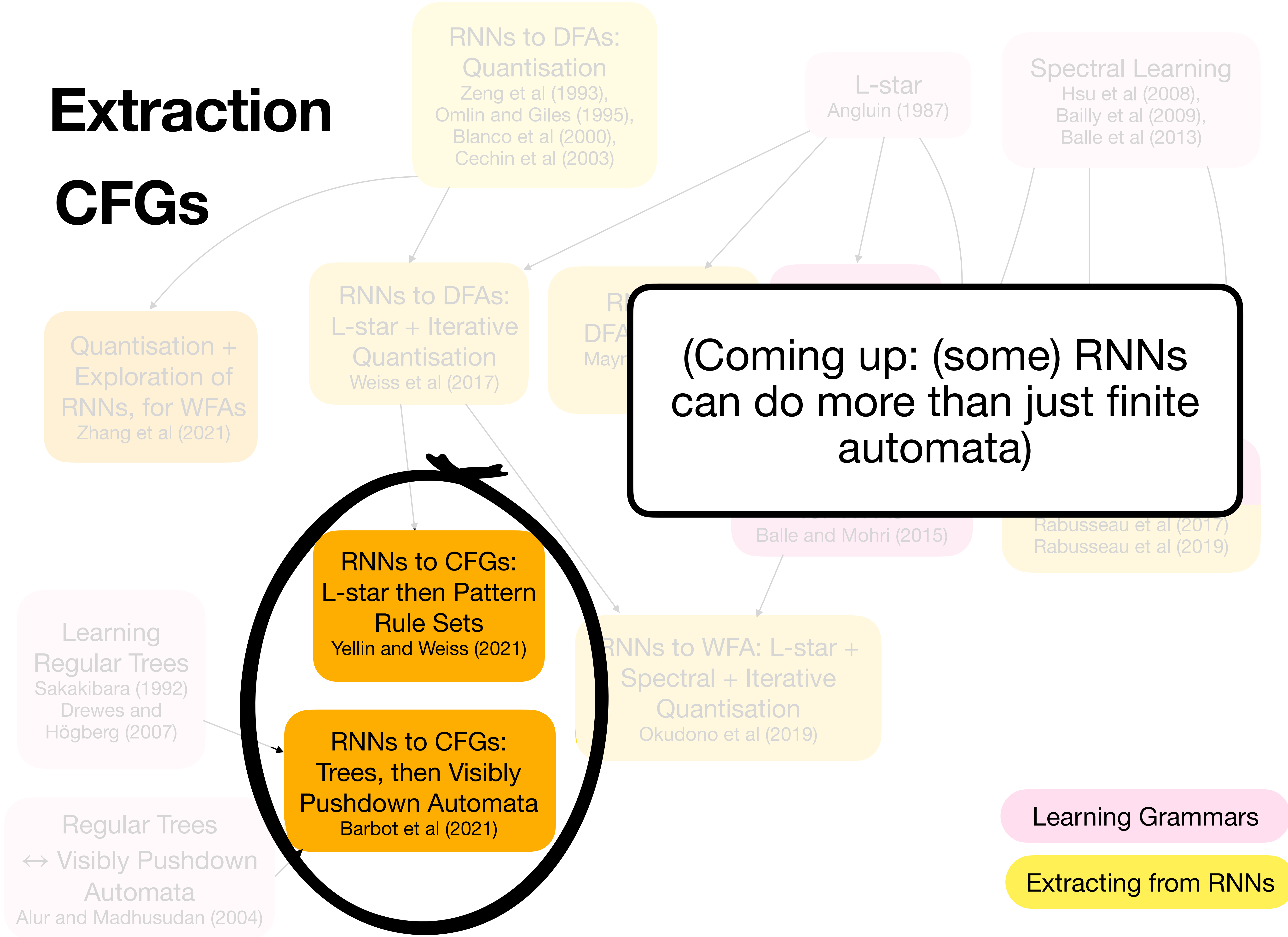

Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

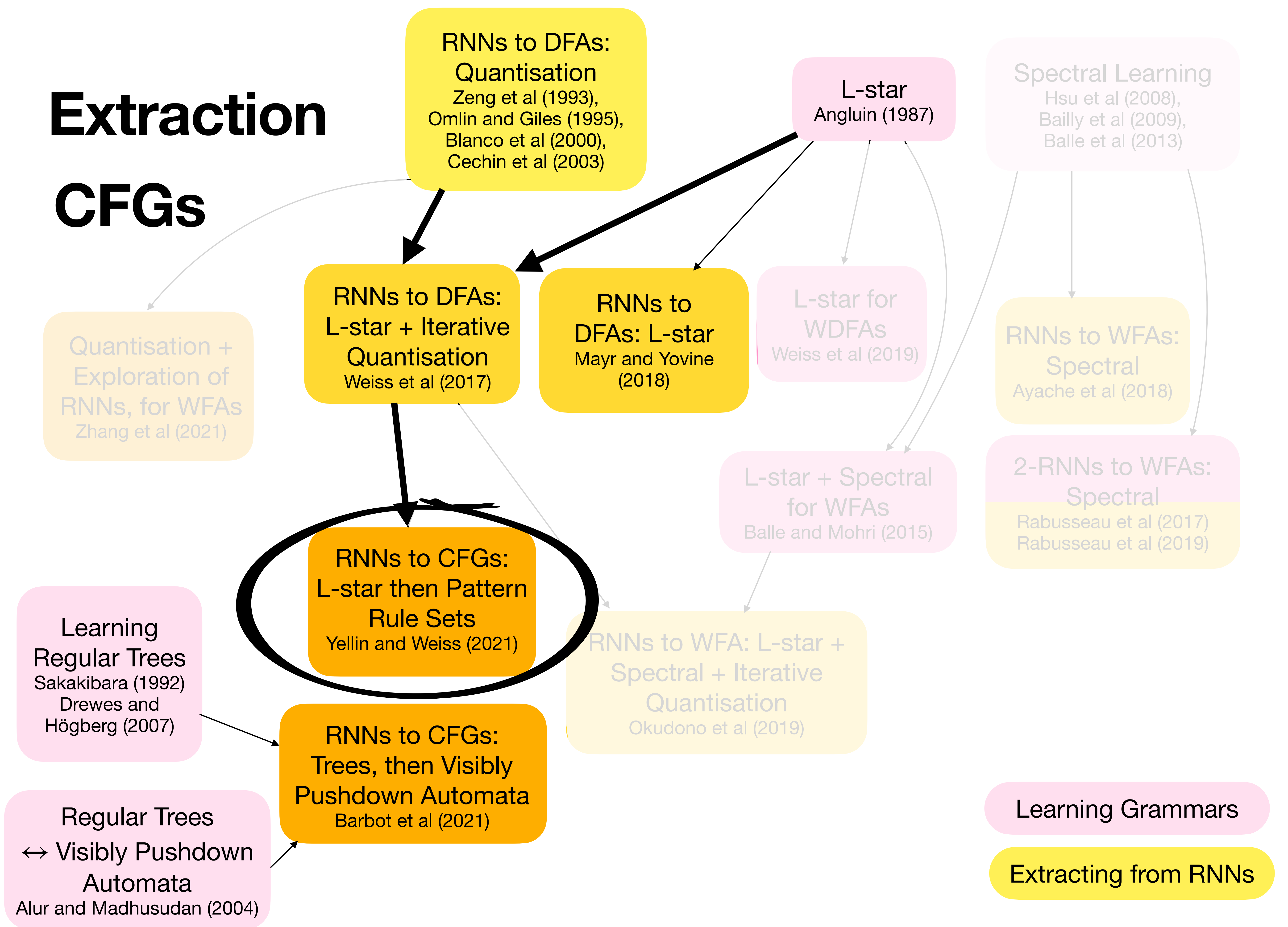

Extraction
WFAs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs


Extraction
CFGs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

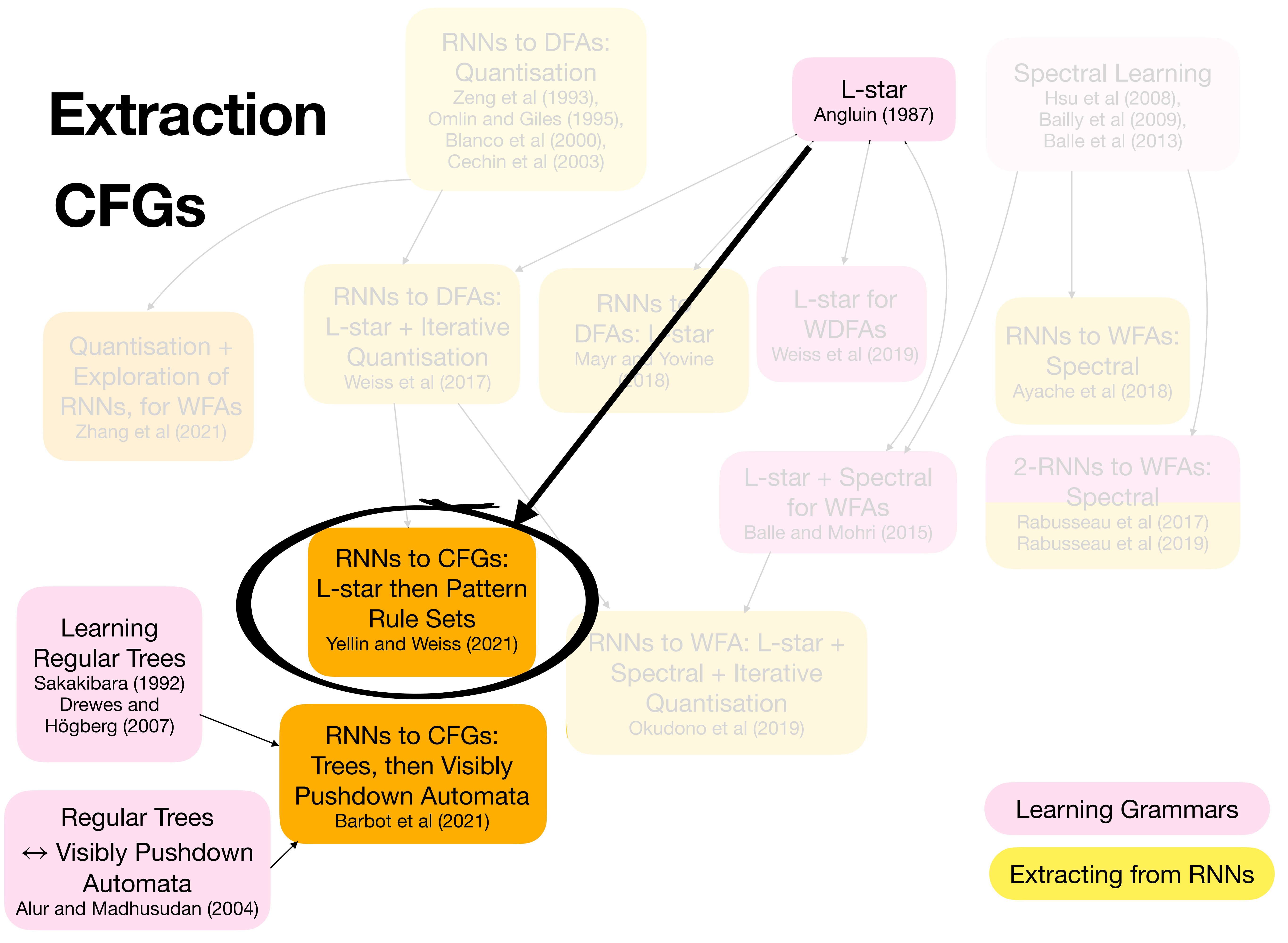
Extraction

CFGs





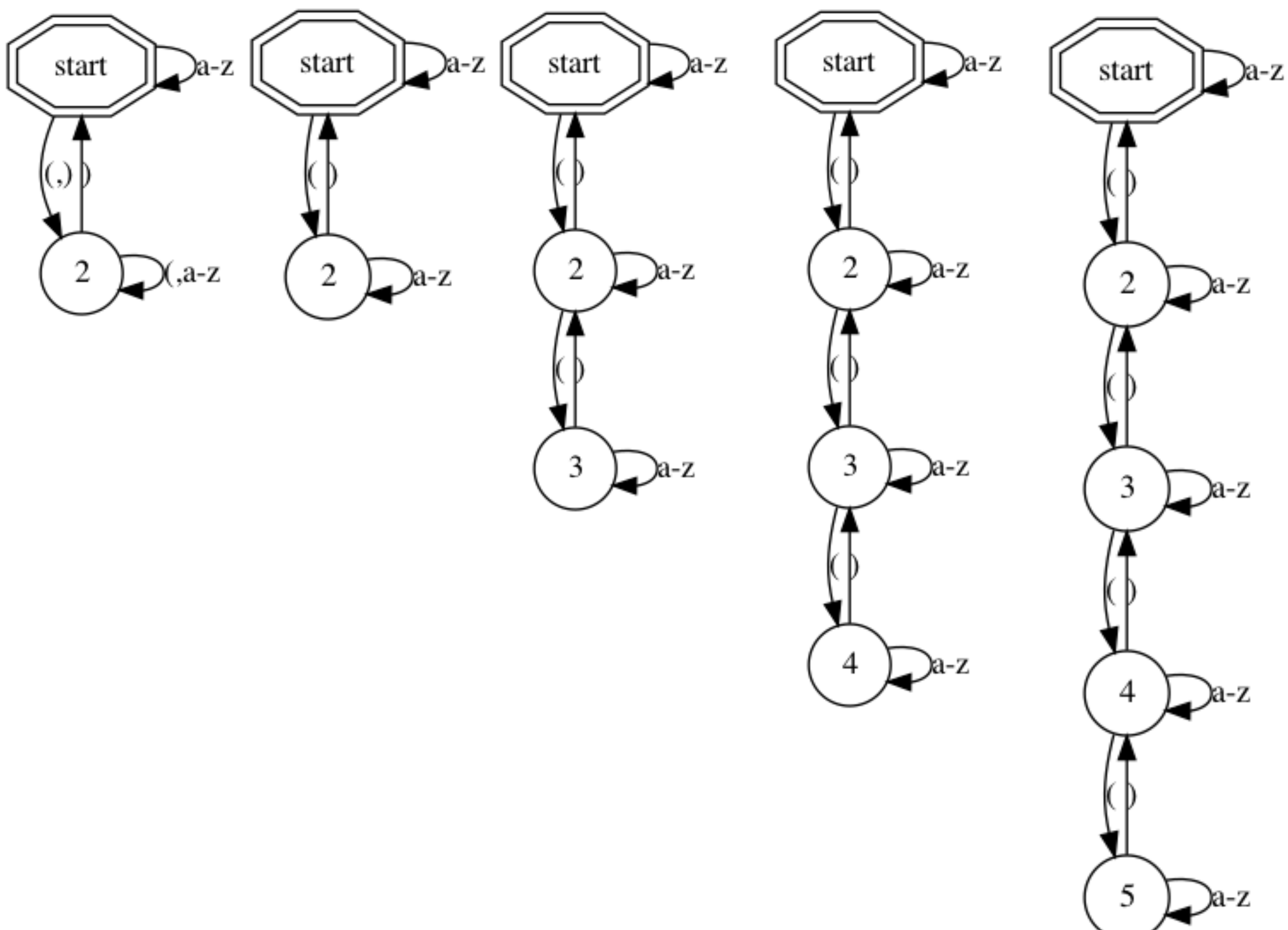
Extraction
CFGs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs


Extraction
CFGs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)

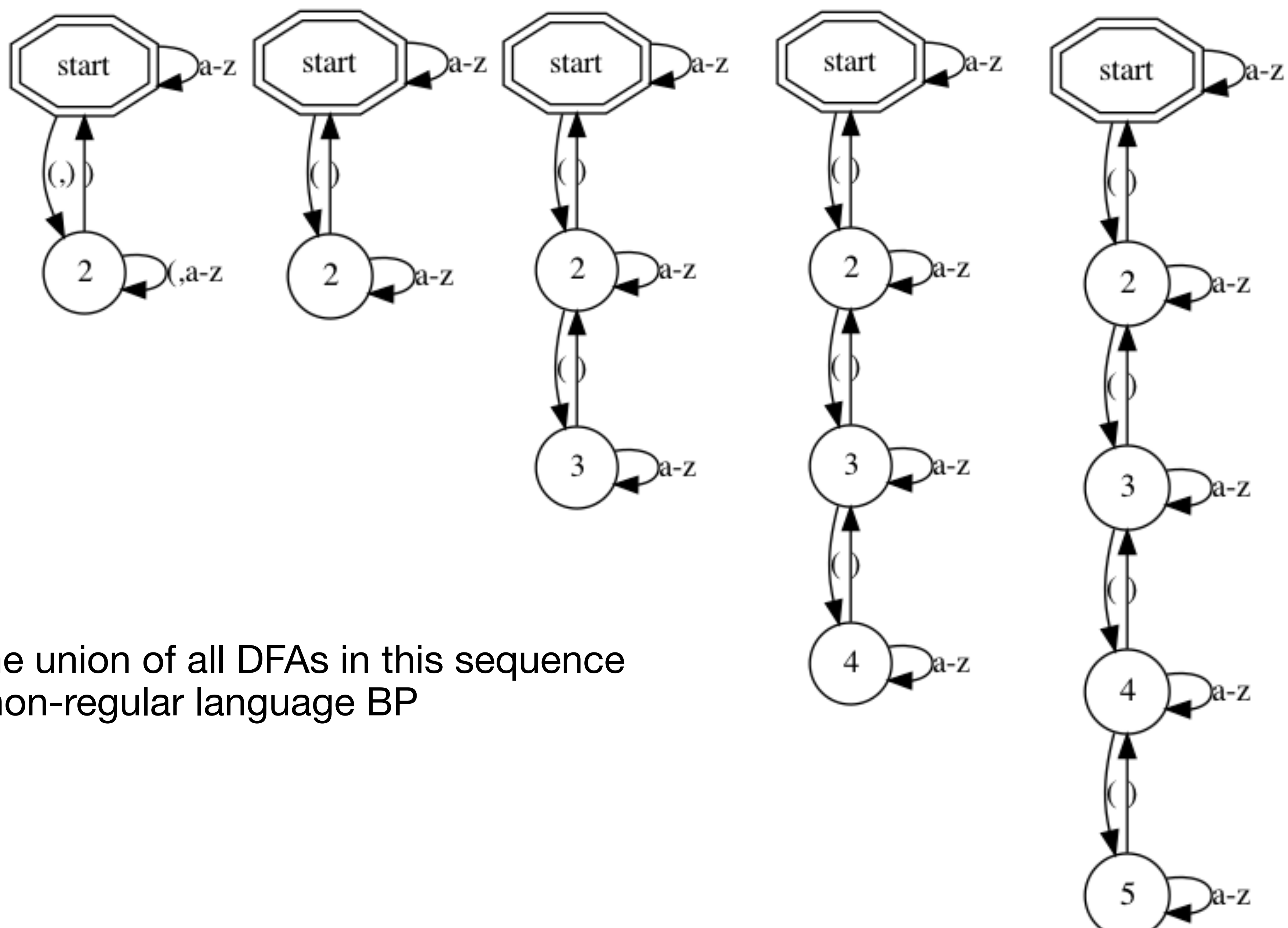


etc…

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)



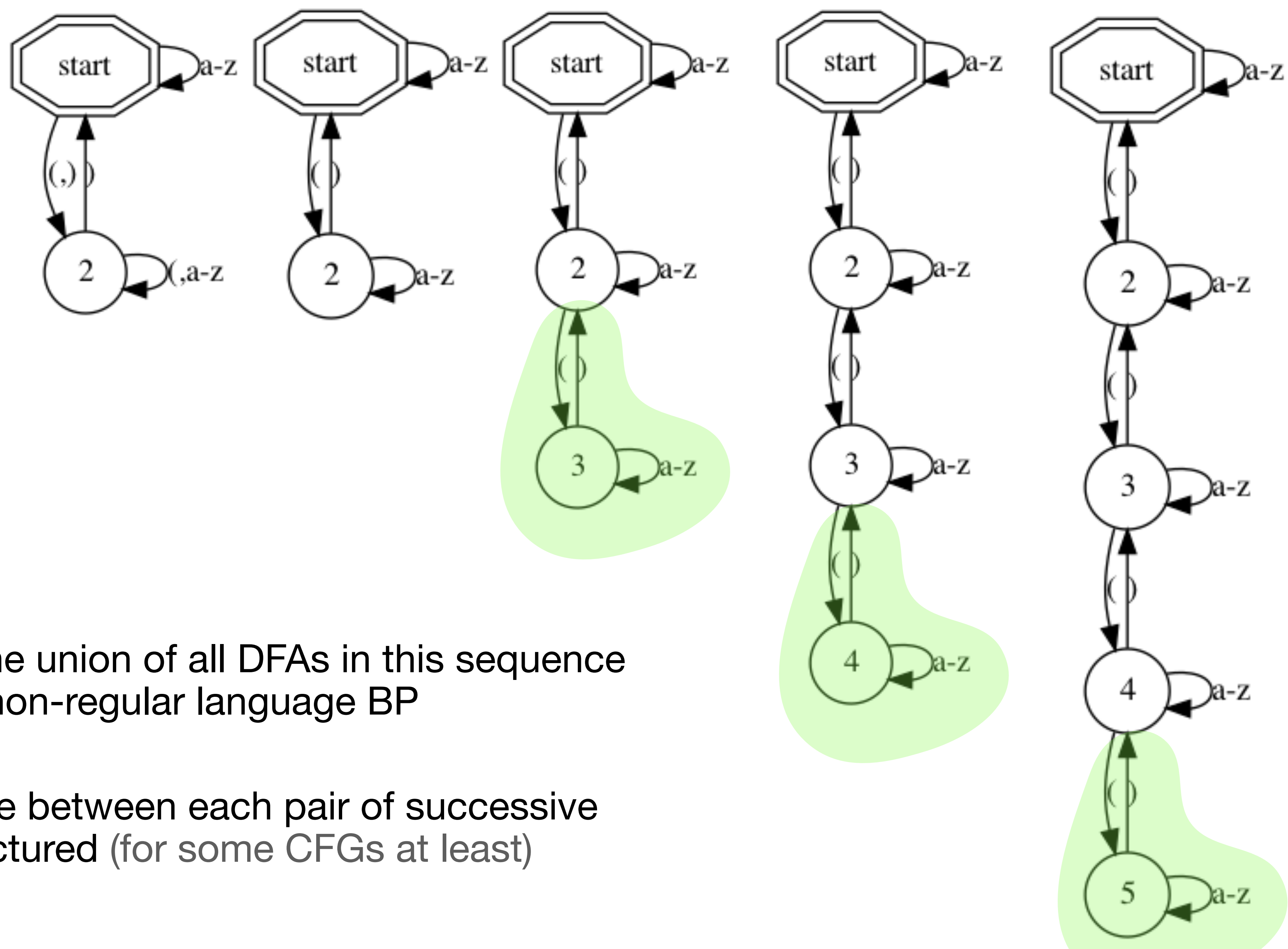
etc…

1. In the limit, the union of all DFAs in this sequence accepts the non-regular language BP

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)



etc…

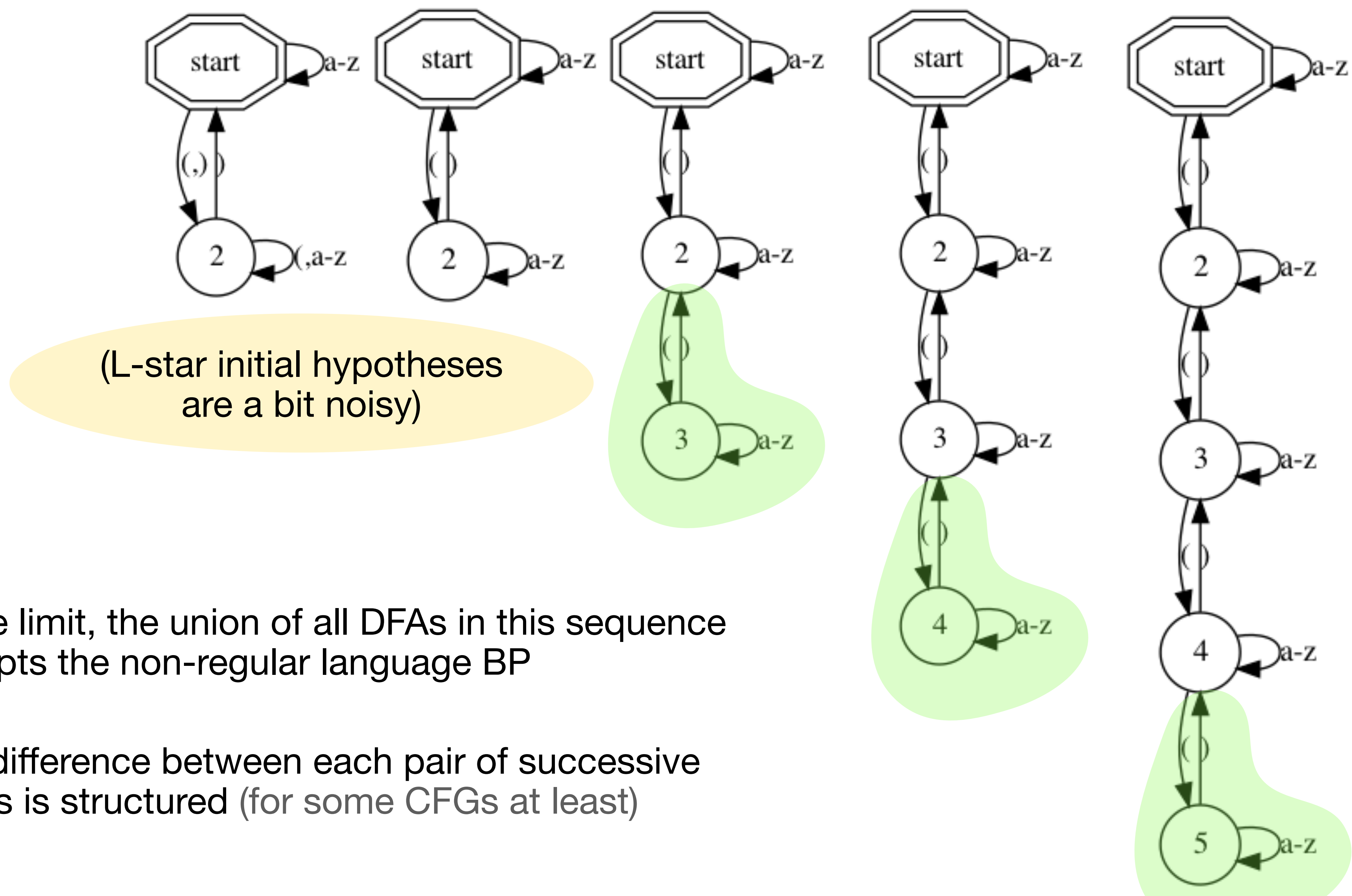
1. In the limit, the union of all DFAs in this sequence accepts the non-regular language BP

2. The difference between each pair of successive RNNs is structured (for some CFGs at least)

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Observation: L-star learning a CFG seems to have structured increases (example on balanced parentheses)



etc…

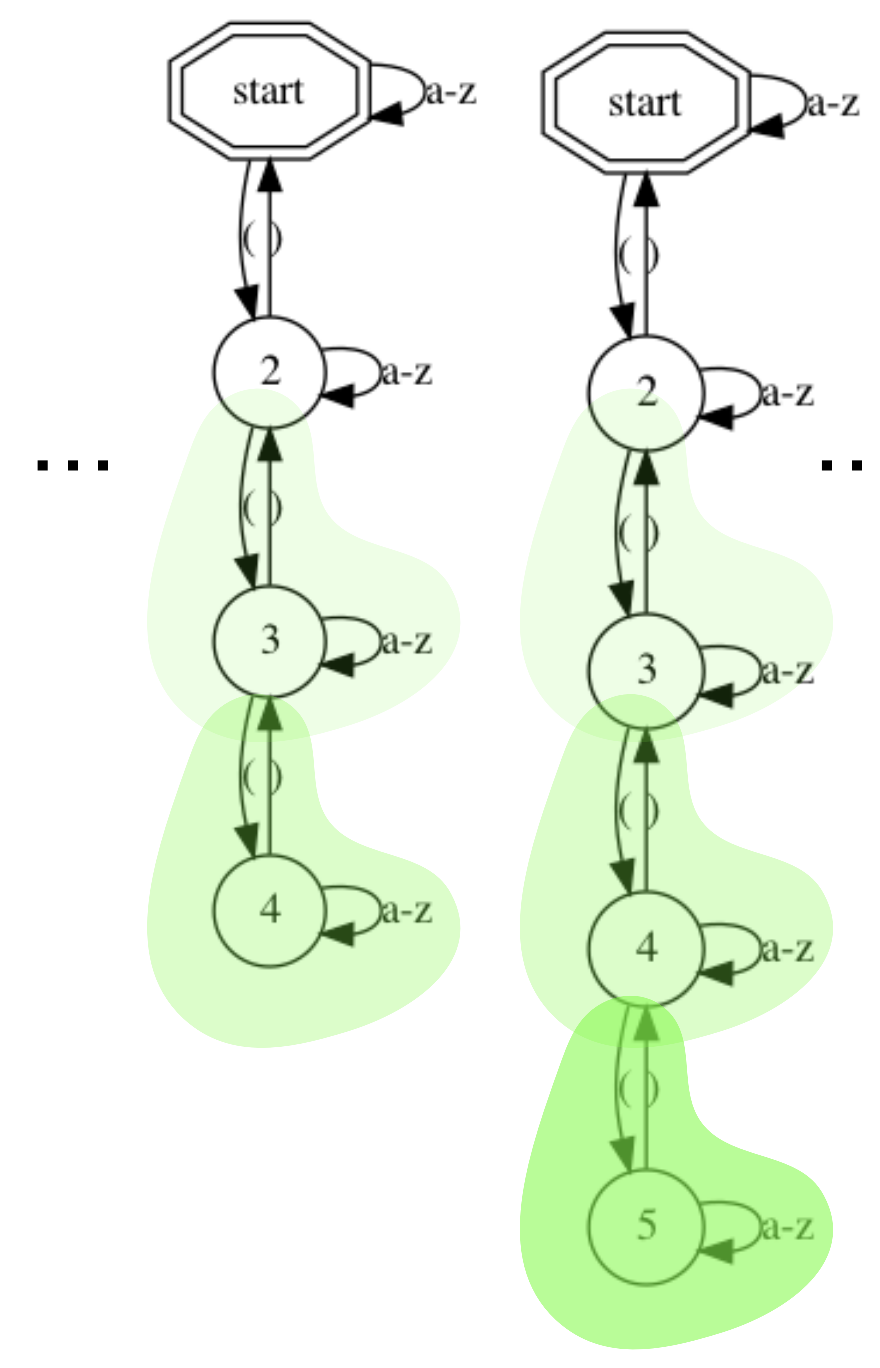
1. In the limit, the union of all DFAs in this sequence accepts the non-regular language BP

2. The difference between each pair of successive RNNs is structured (for some CFGs at least)

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Patterns

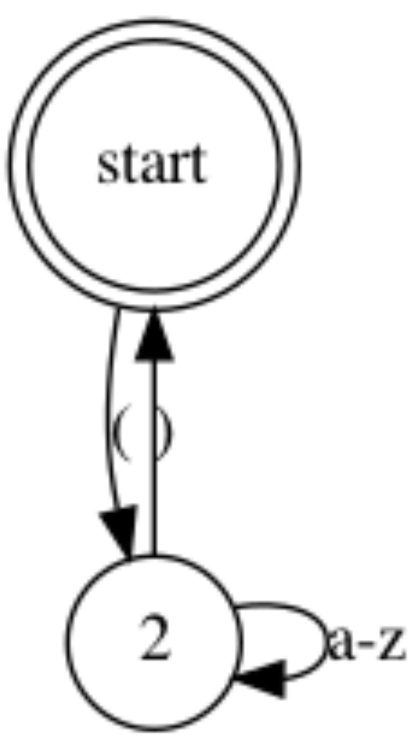


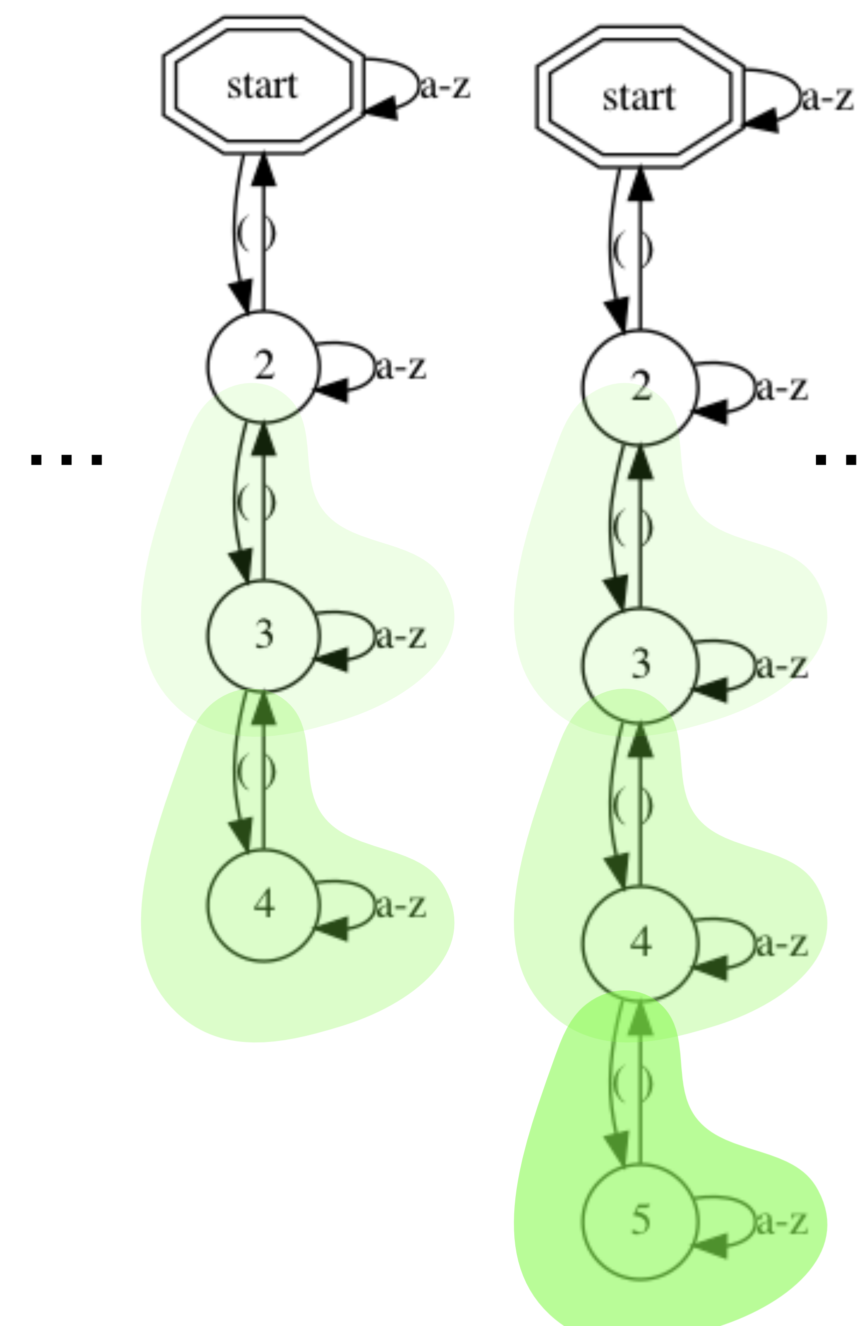
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Patterns

- Structure



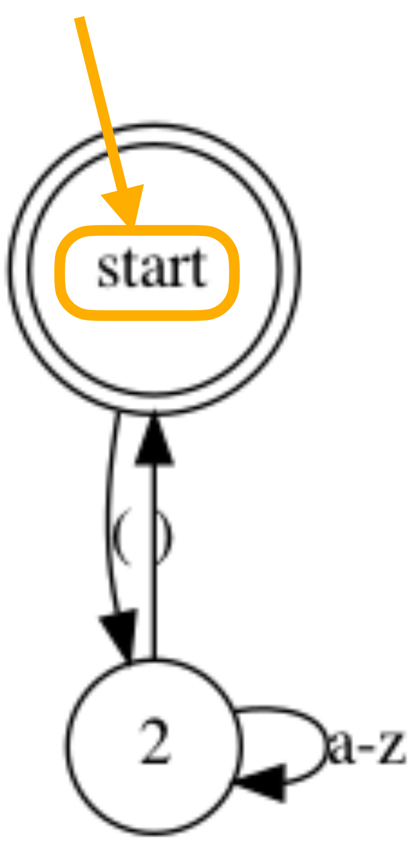


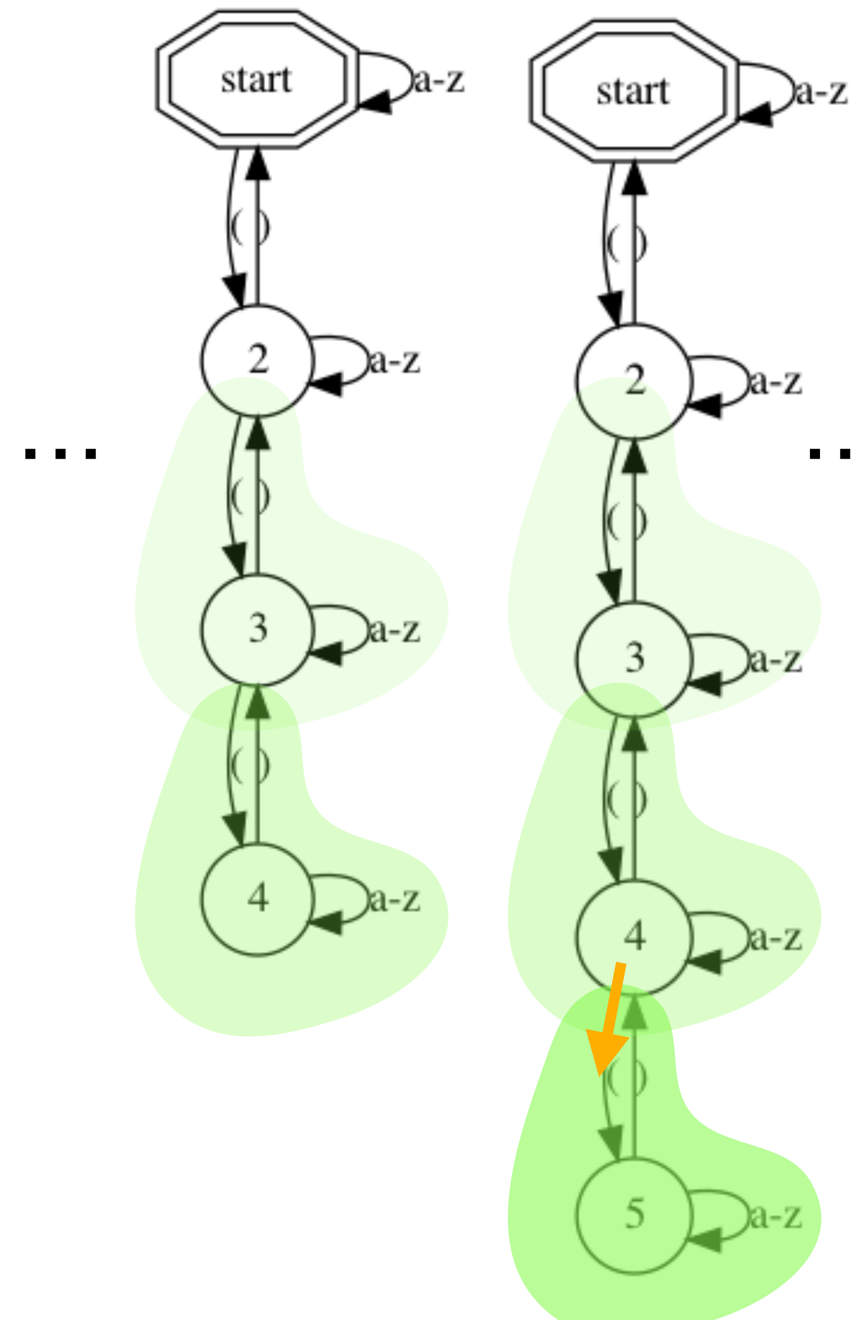
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Patterns

- Structure
  - Entry



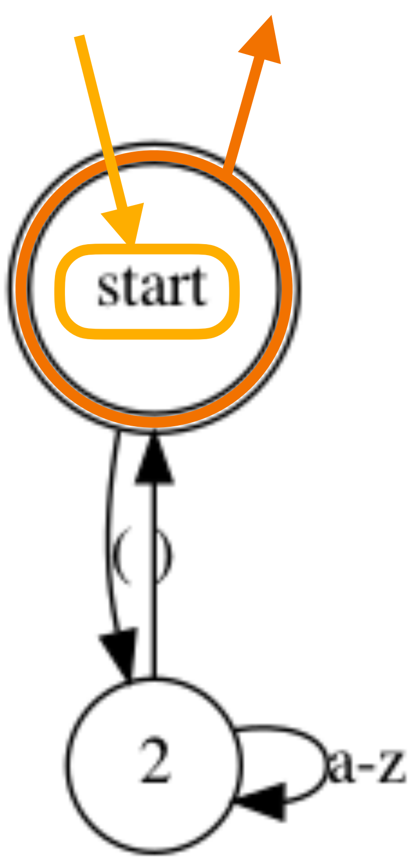


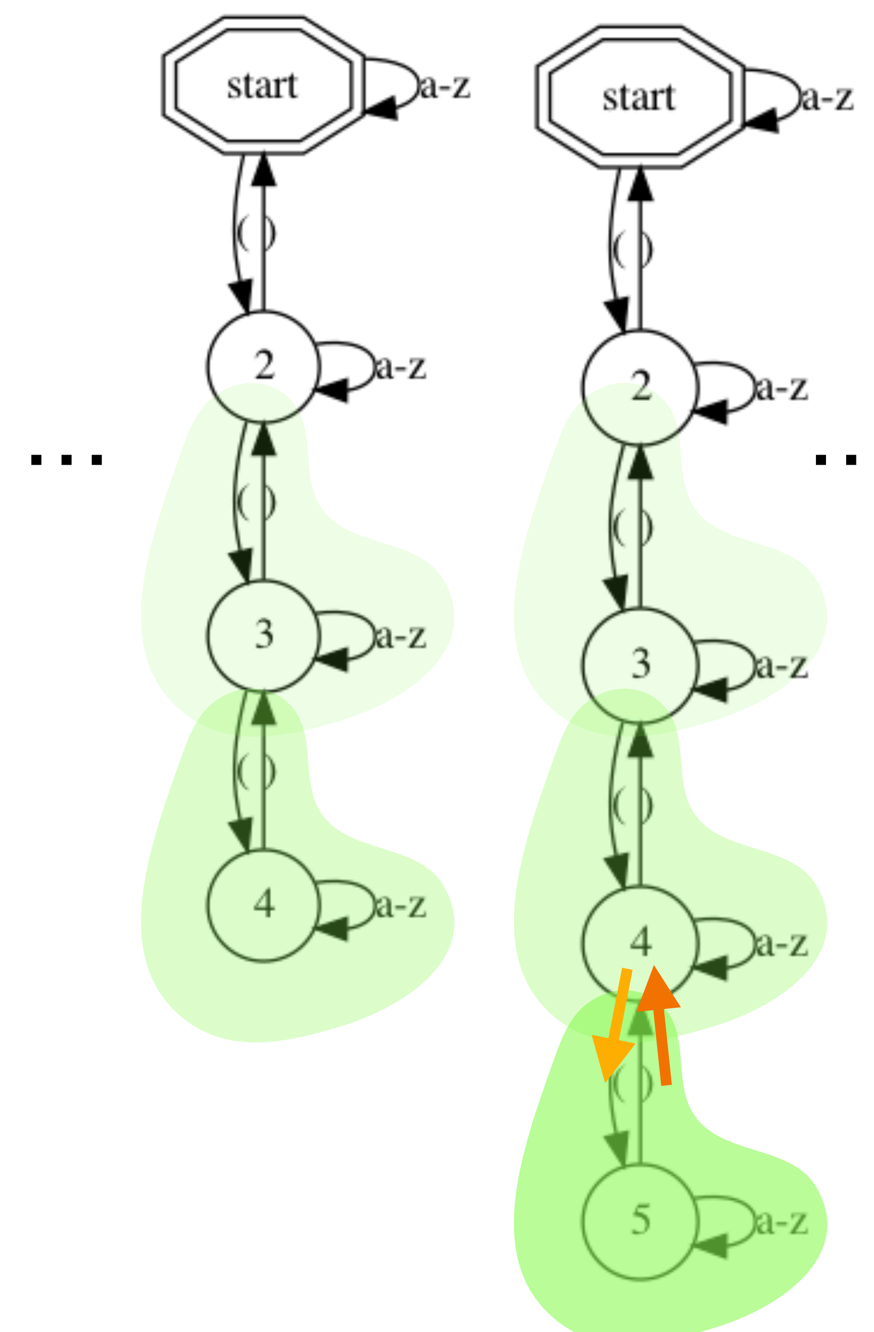
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Patterns

- Structure
  - Entry
  - Exit



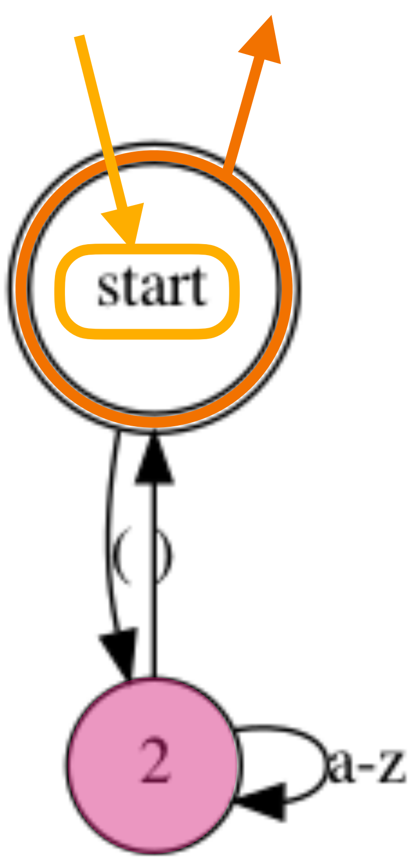


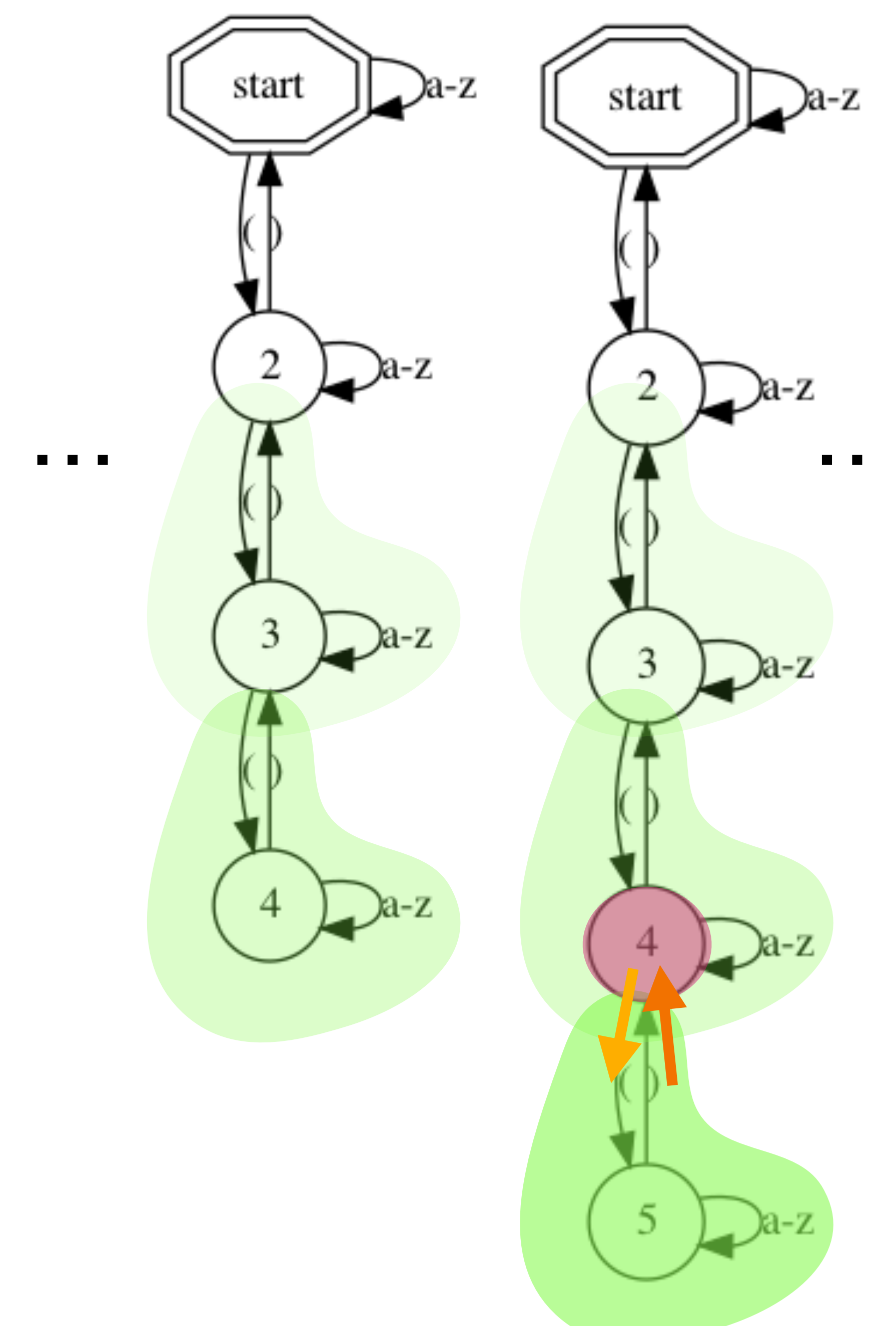
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Patterns

- Structure
  - Entry
  - Exit
- Connection Point(s)



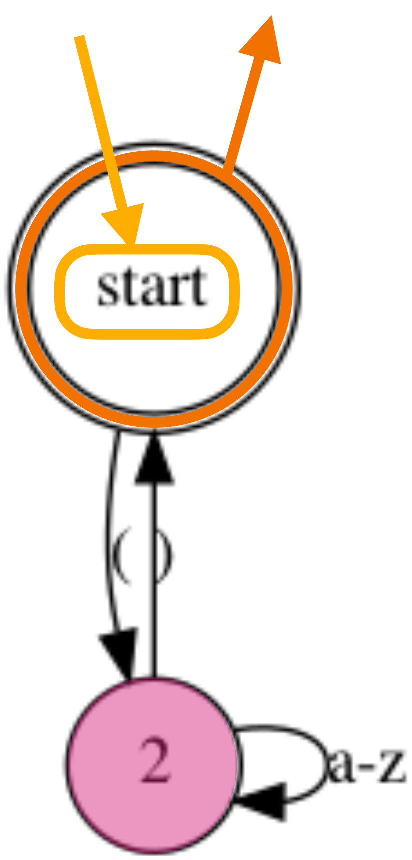


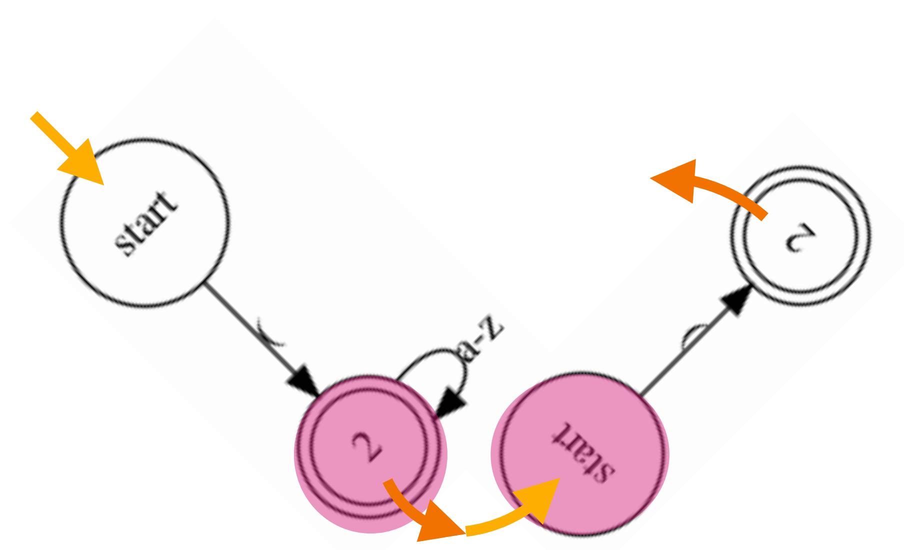
RNNs: Extraction: CFGs: **Pattern Rule Sets**

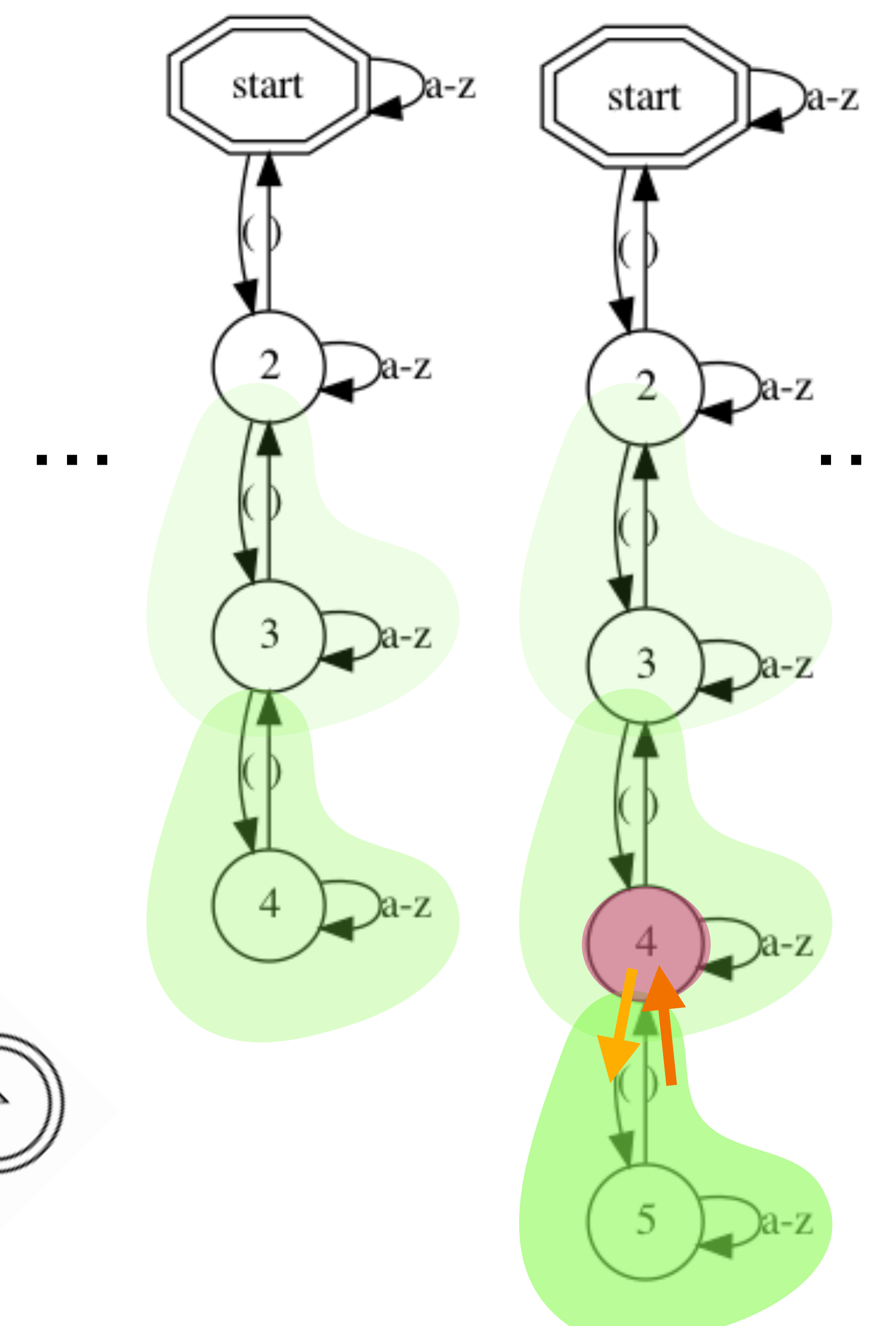
Yellin and Weiss (2021)

Patterns

- Structure
  - Entry
  - Exit
- Connection Point(s)
- Composable
  - Connection points are on compositions





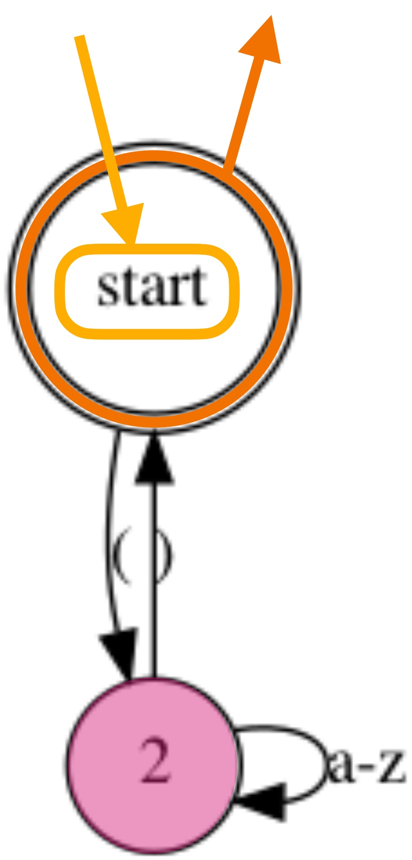


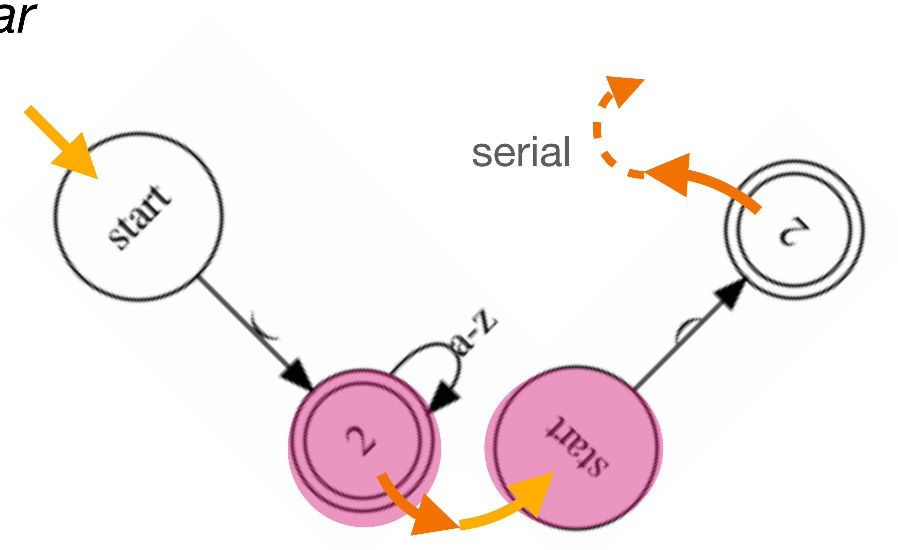
RNNs: Extraction: CFGs: **Pattern Rule Sets**

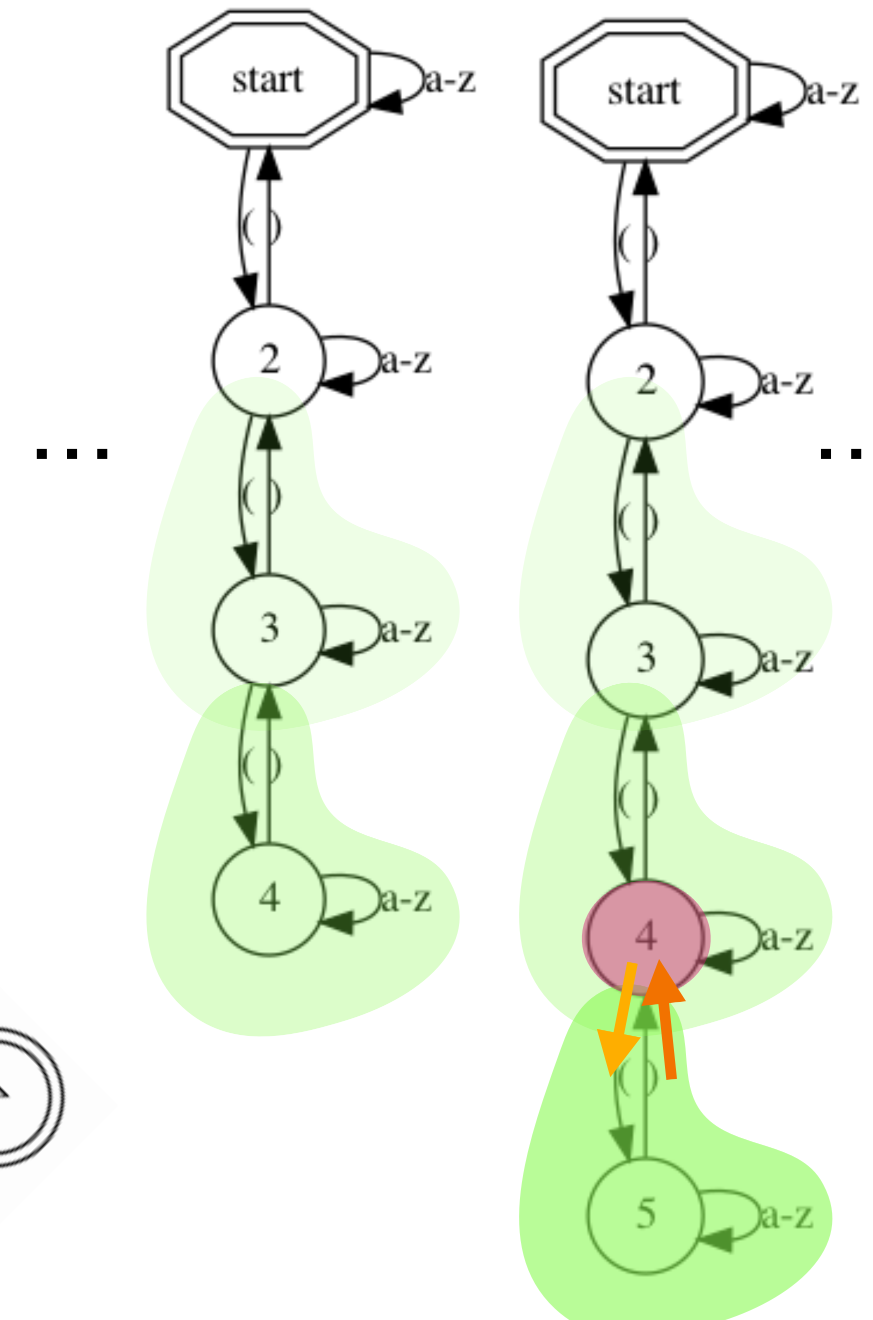
Yellin and Weiss (2021)

Patterns

- Structure
  - Entry
  - Exit
- Connection Point(s)
- Composable
  - Connection points are on compositions
  - Composition can be *serial* or *circular*

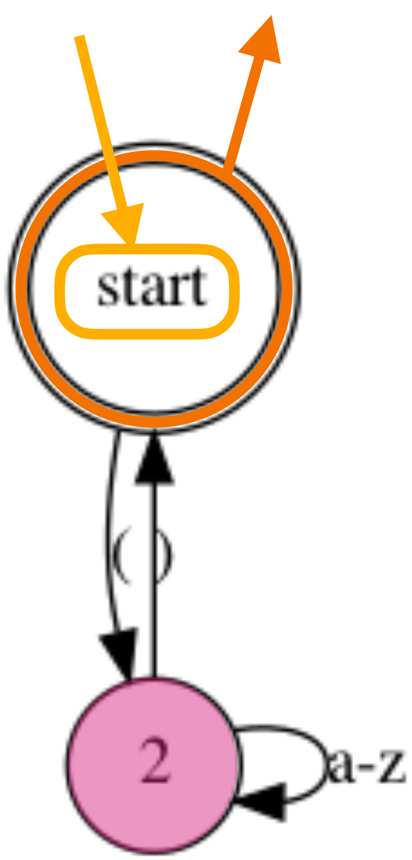






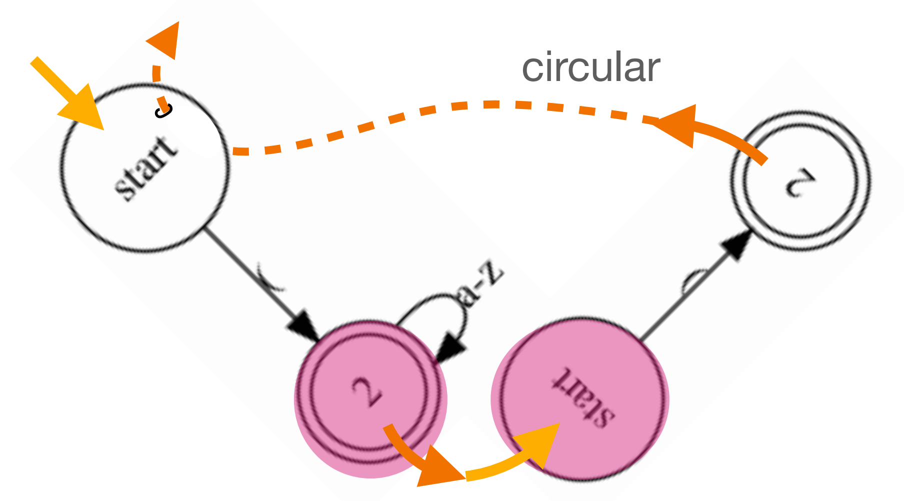
RNNs: Extraction: CFGs: **Pattern Rule Sets**

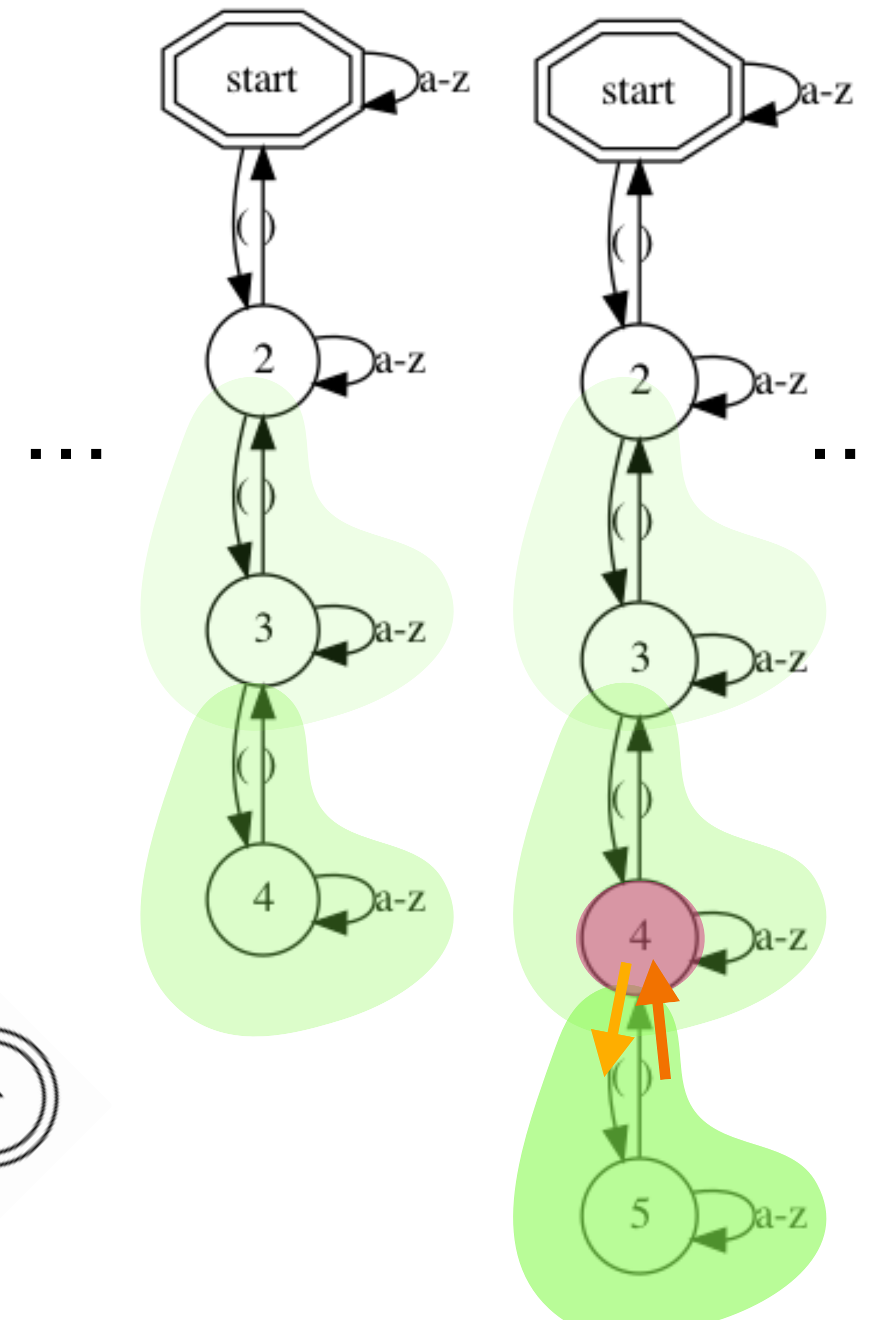
Yellin and Weiss (2021)



Patterns

- Structure
  - Entry
  - Exit
- Connection Point(s)
- Composable
  - Connection points are on compositions
  - Composition can be *serial* or *circular*





RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. **The first DFA**: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2

RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. **The first DFA**: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2

(When we get to extraction:
this might not be the same
first DFA that L-star suggests)

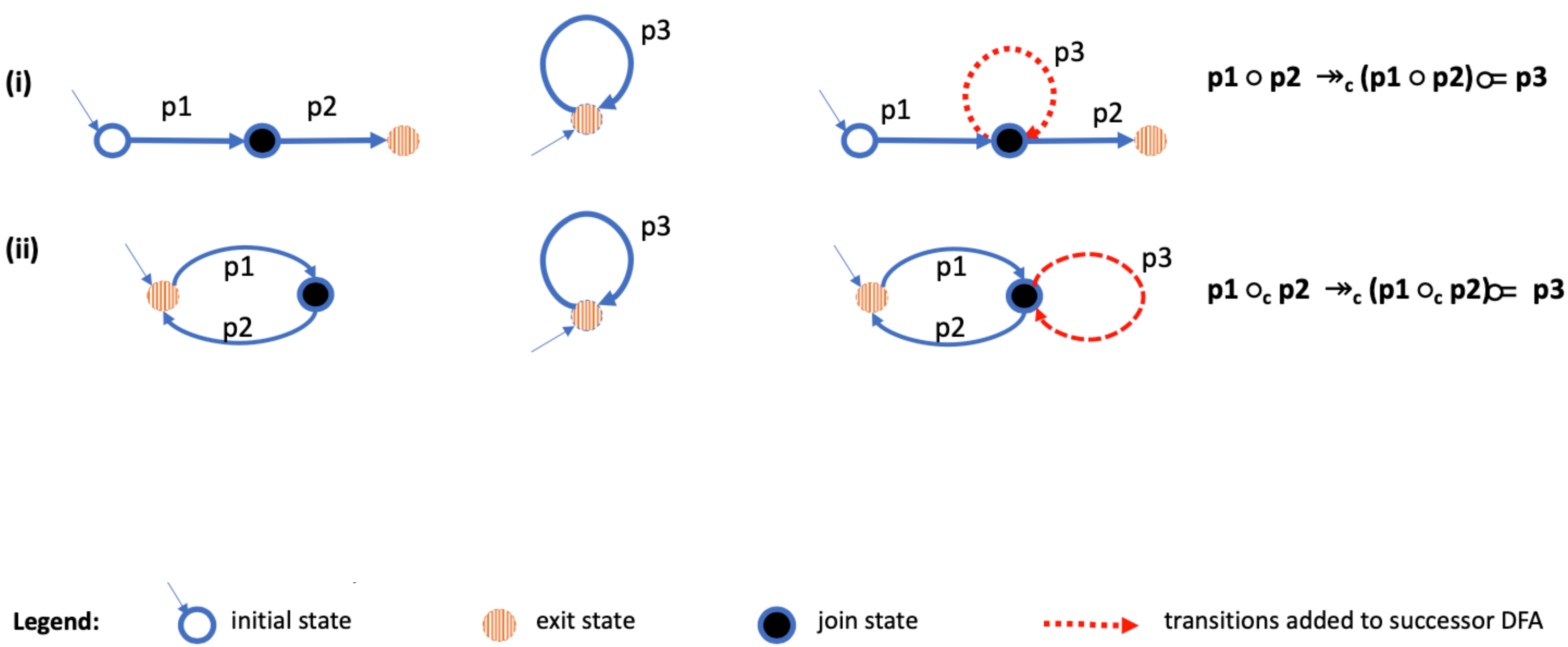
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. **Insert circular pattern** p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2



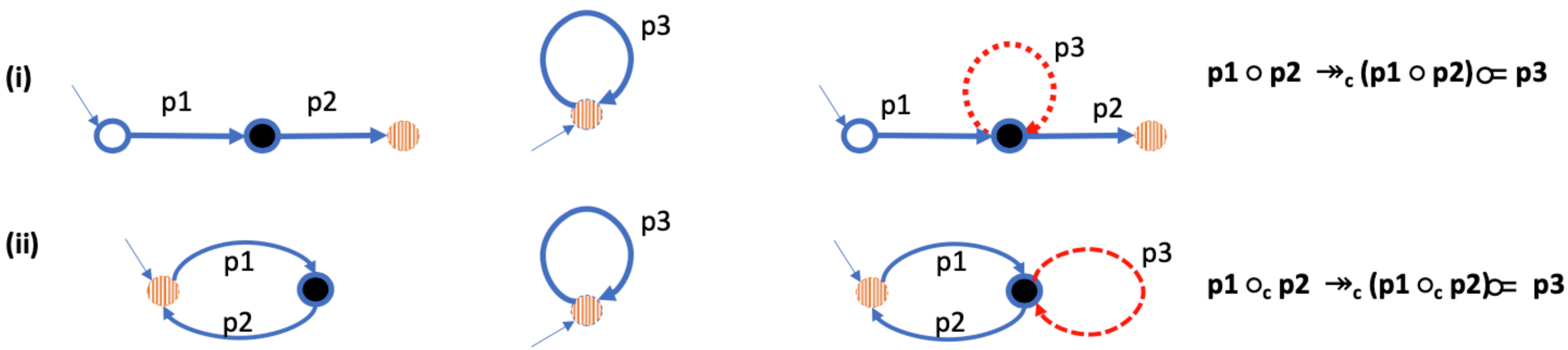
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. **Insert circular pattern** p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2



What happens when adding another (different) circular pattern to the same state?

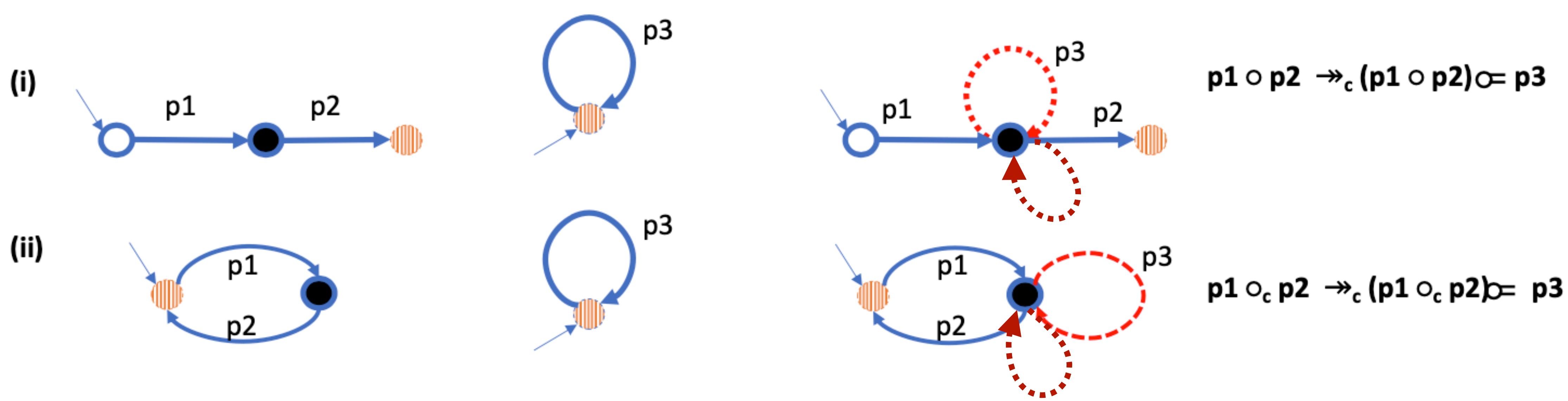
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. **Insert circular pattern** p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2



What happens when adding another (different) circular pattern to the same state?

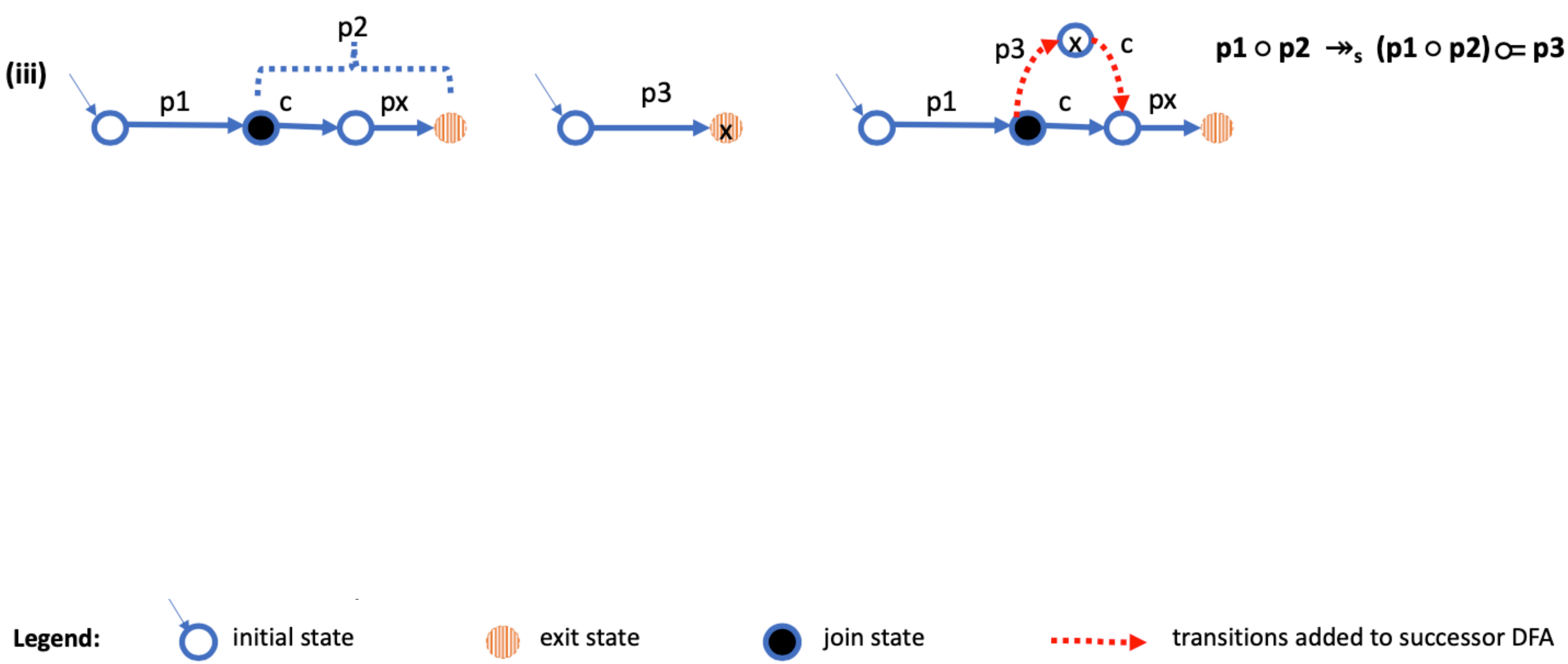
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. **Insert serial pattern** p_1 on join state of serial pattern p_2



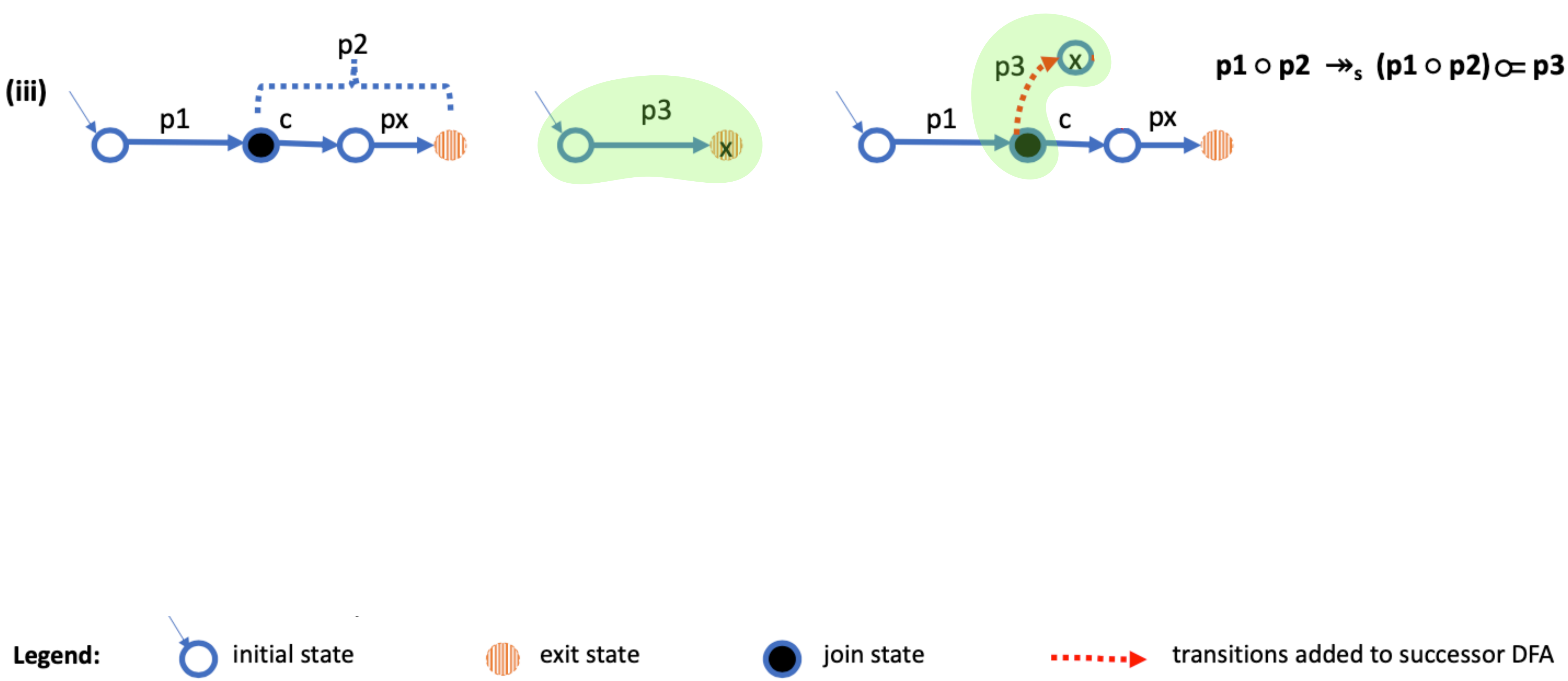
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. **Insert serial pattern** p_1 on join state of serial pattern p_2



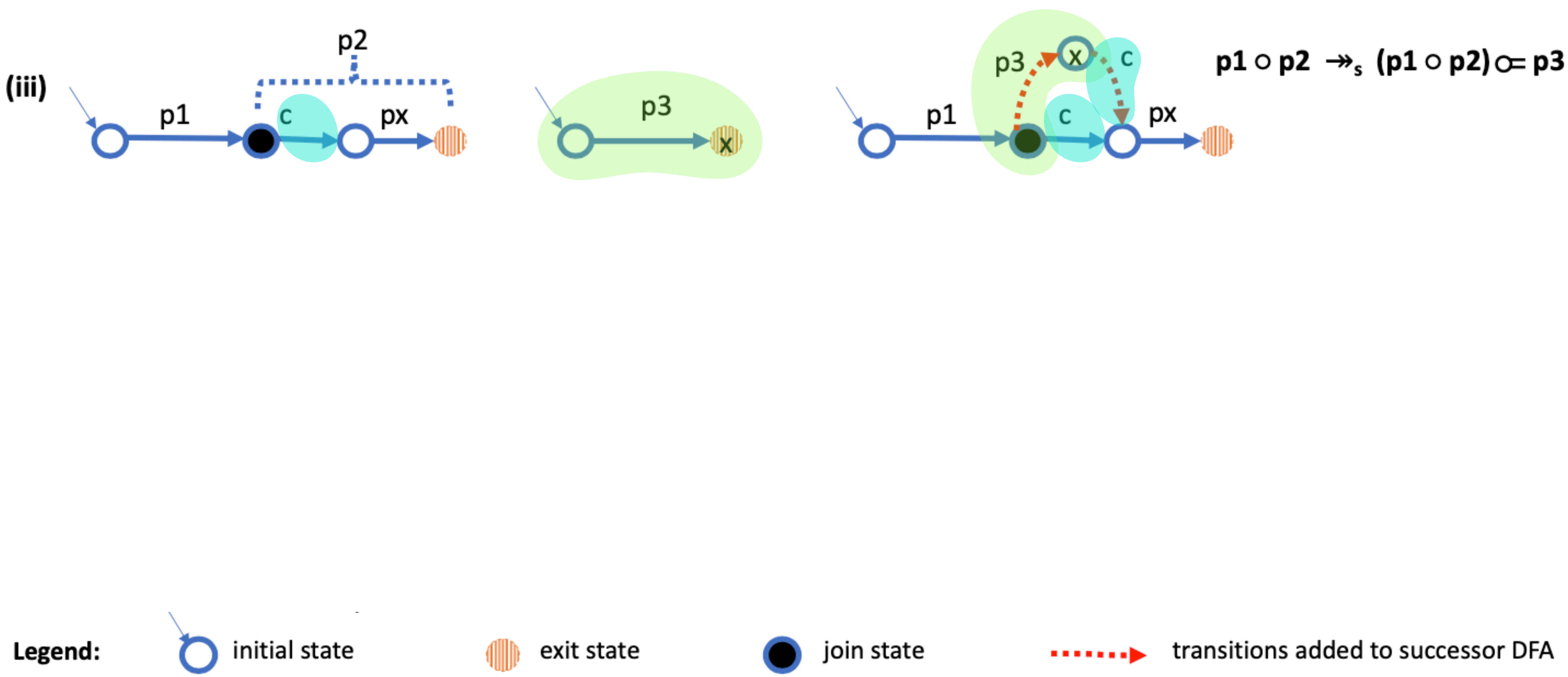
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. **Insert serial pattern** p_1 on join state of serial pattern p_2



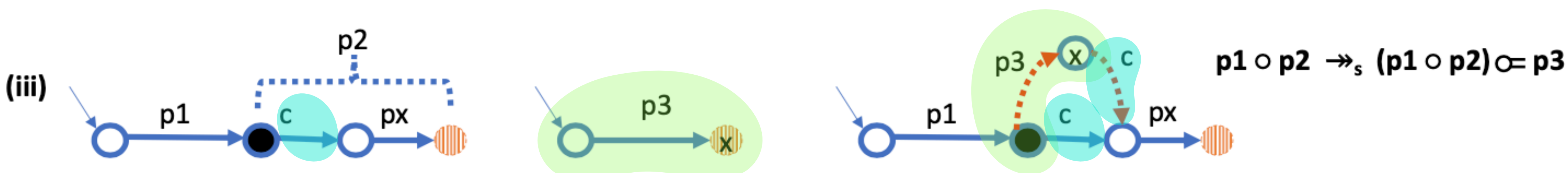
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. **Insert serial pattern** p_1 on join state of serial pattern p_2



What happens when adding another (different) serial pattern to the same state?

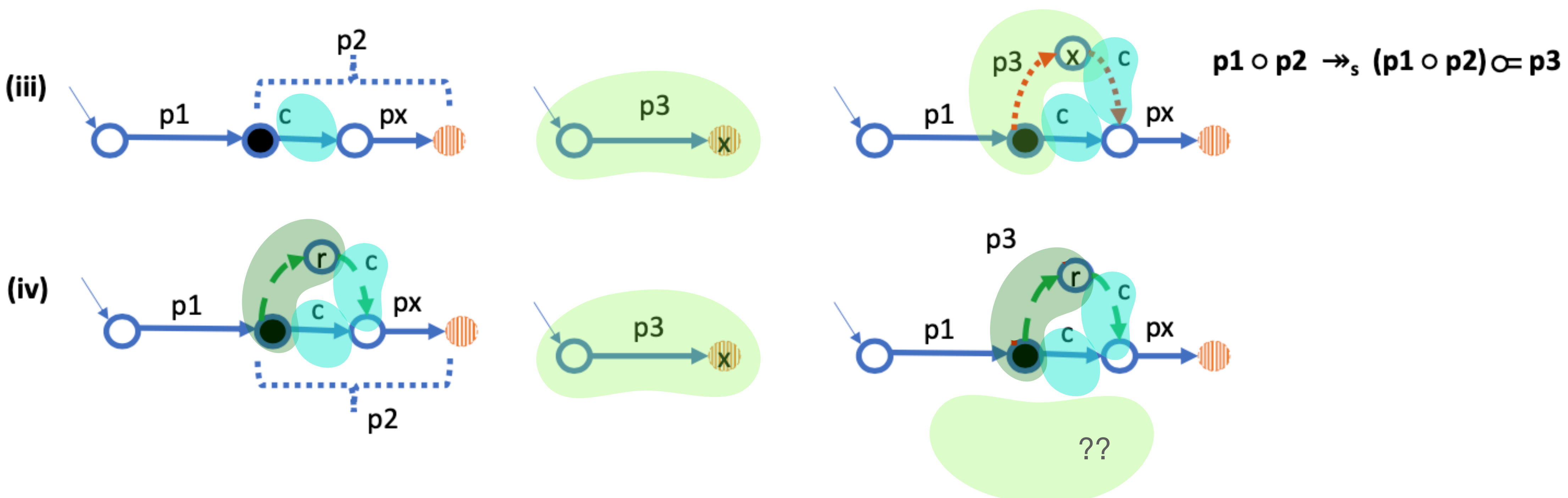
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. **Insert serial pattern** p_1 on join state of serial pattern p_2



What happens when adding another (different) serial pattern to the same state?

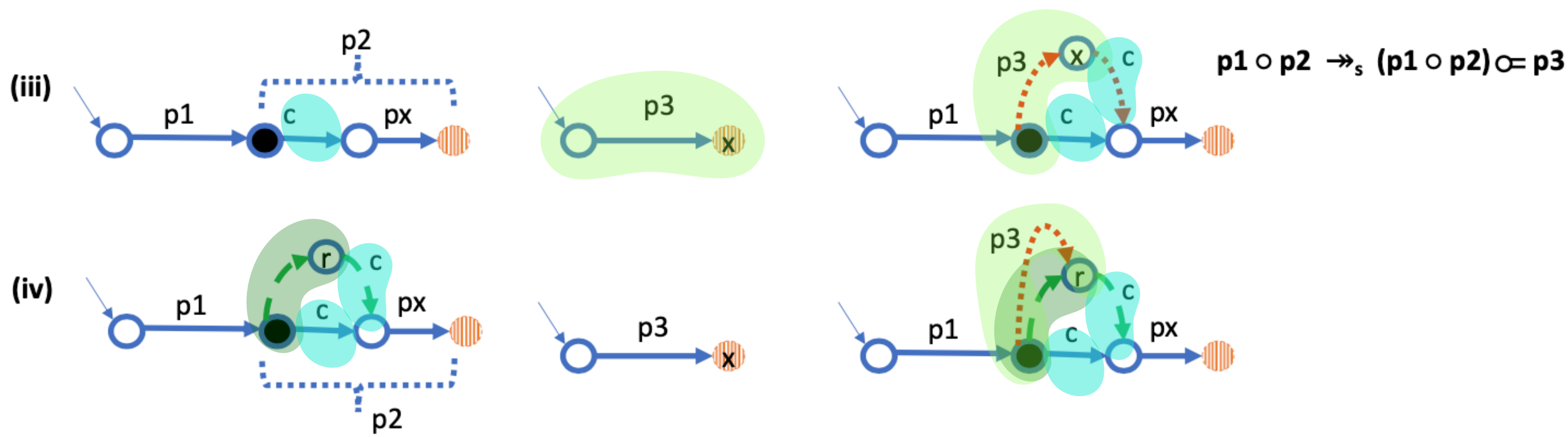
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. **Insert serial pattern** p_1 on join state of serial pattern p_2



What happens when adding another (different) serial pattern to the same state?

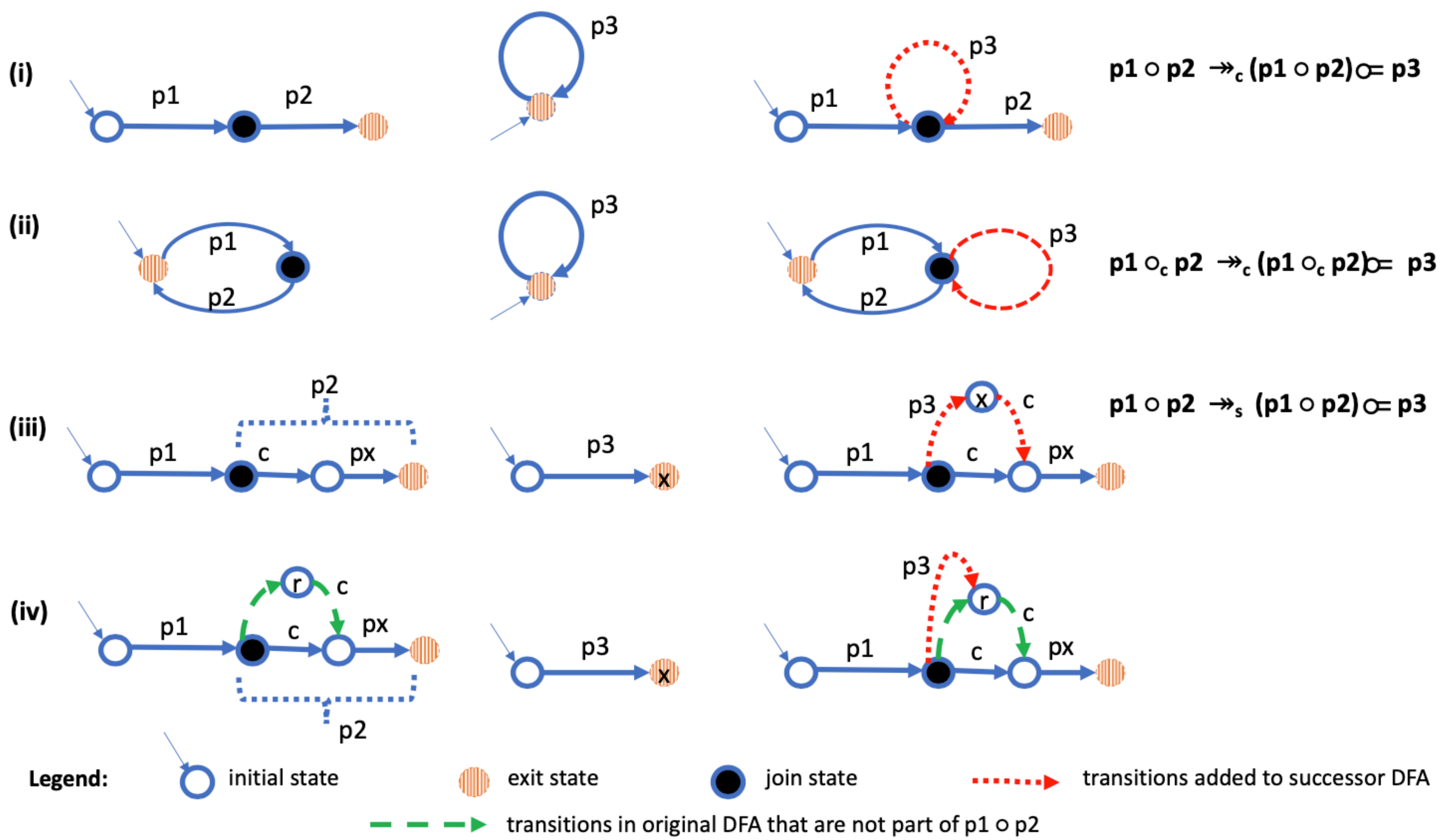
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Rules

Rules describe how specific patterns initiate and expand the DFAs. There are three types:

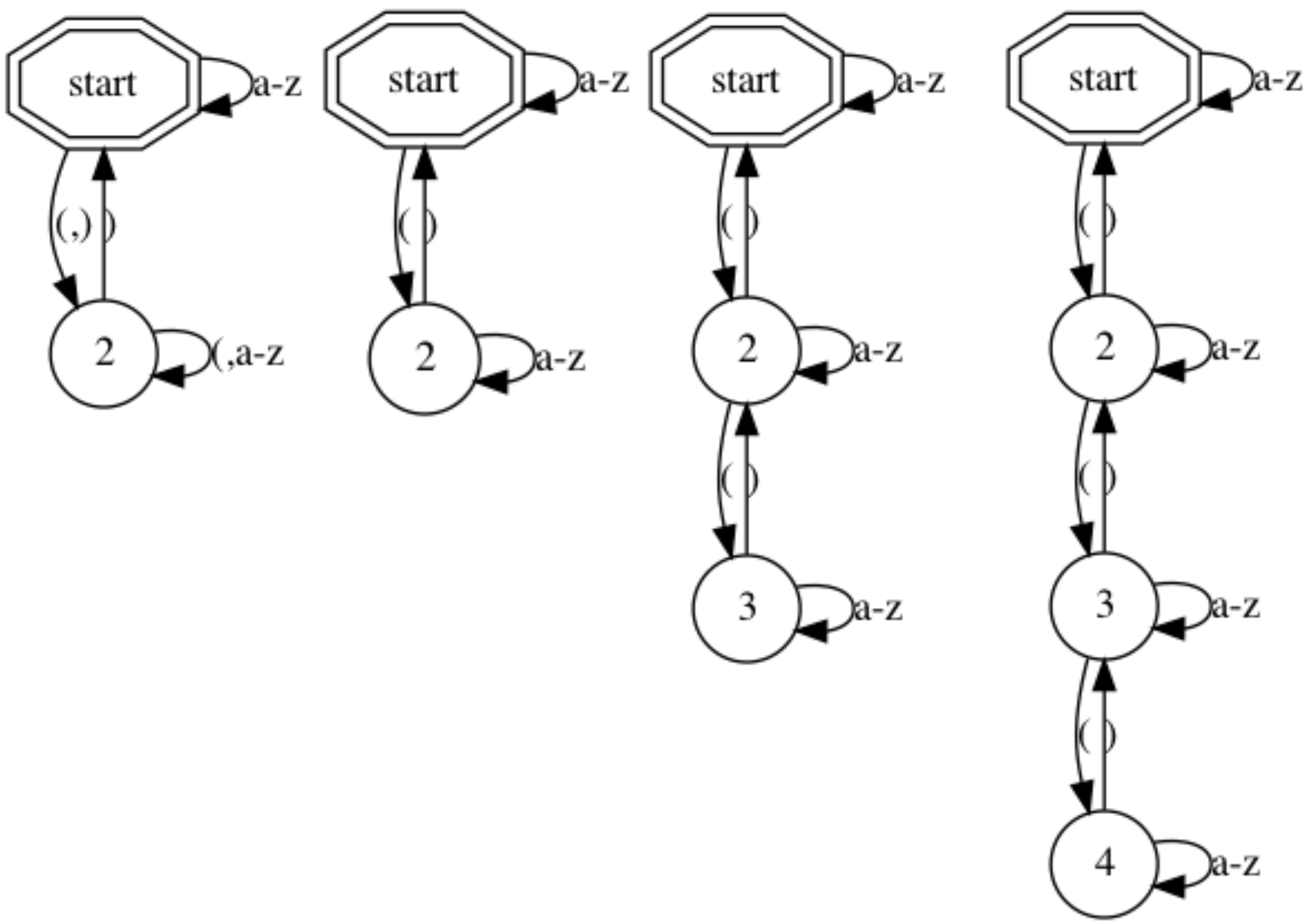
1. The first DFA: an initial pattern p_I . 2. Insert circular pattern p_1 on join state of p_2 . 3. Insert serial pattern p_1 on join state of serial pattern p_2



RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

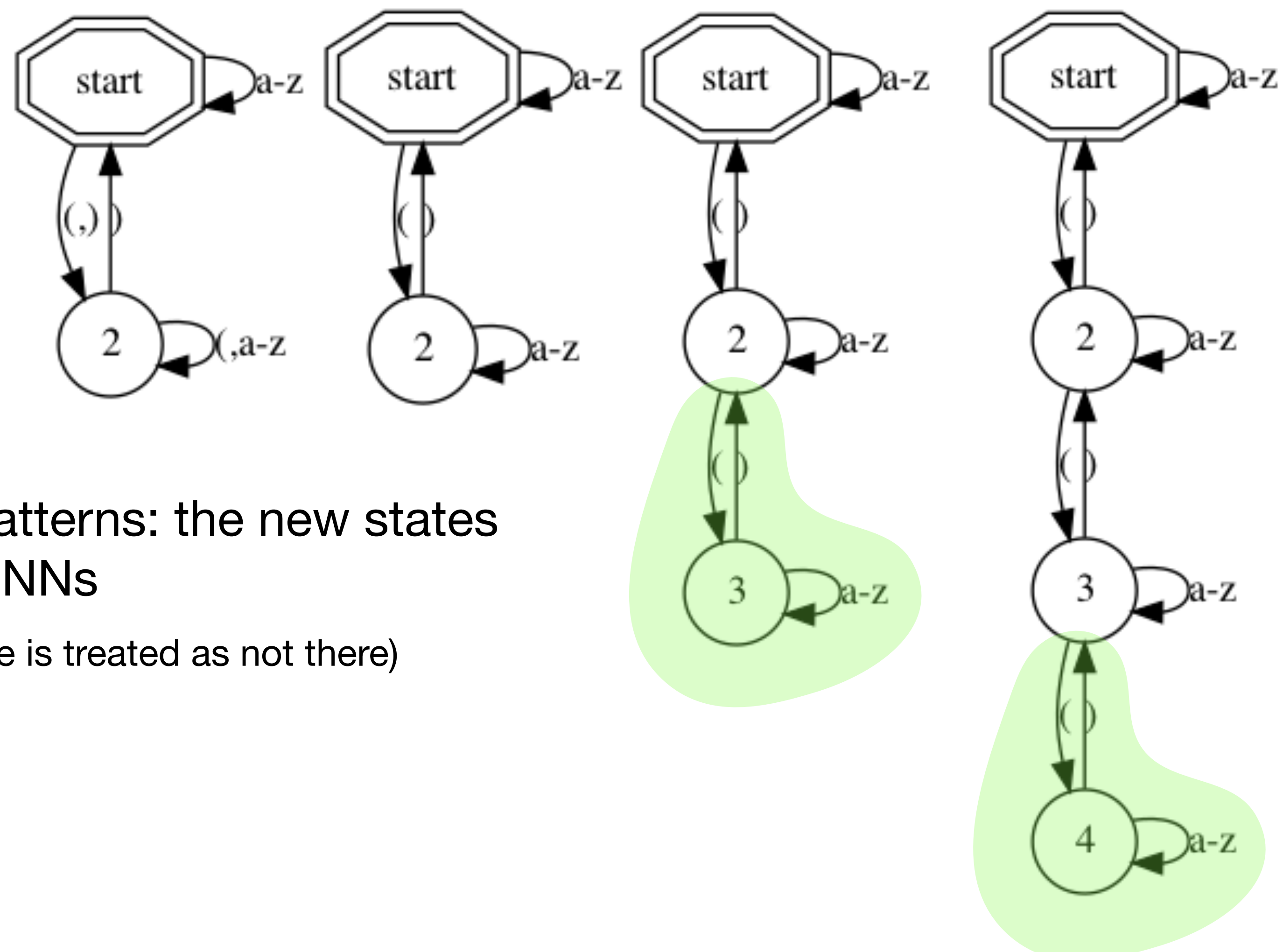
Recovering a Pattern Rule Set



RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Recovering a Pattern Rule Set

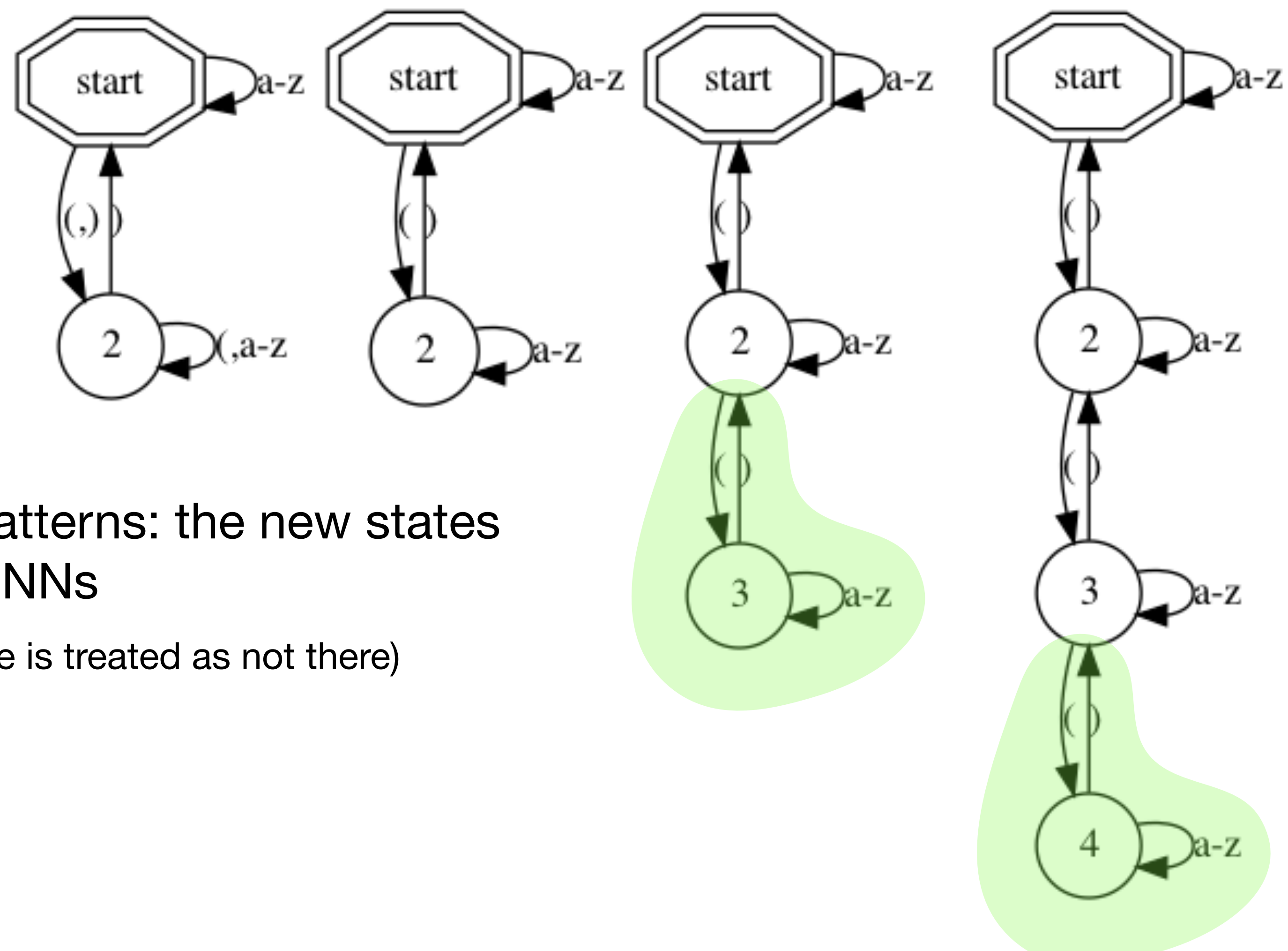


1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)

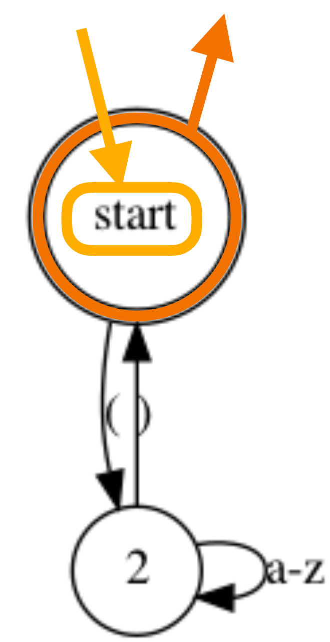
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Recovering a Pattern Rule Set



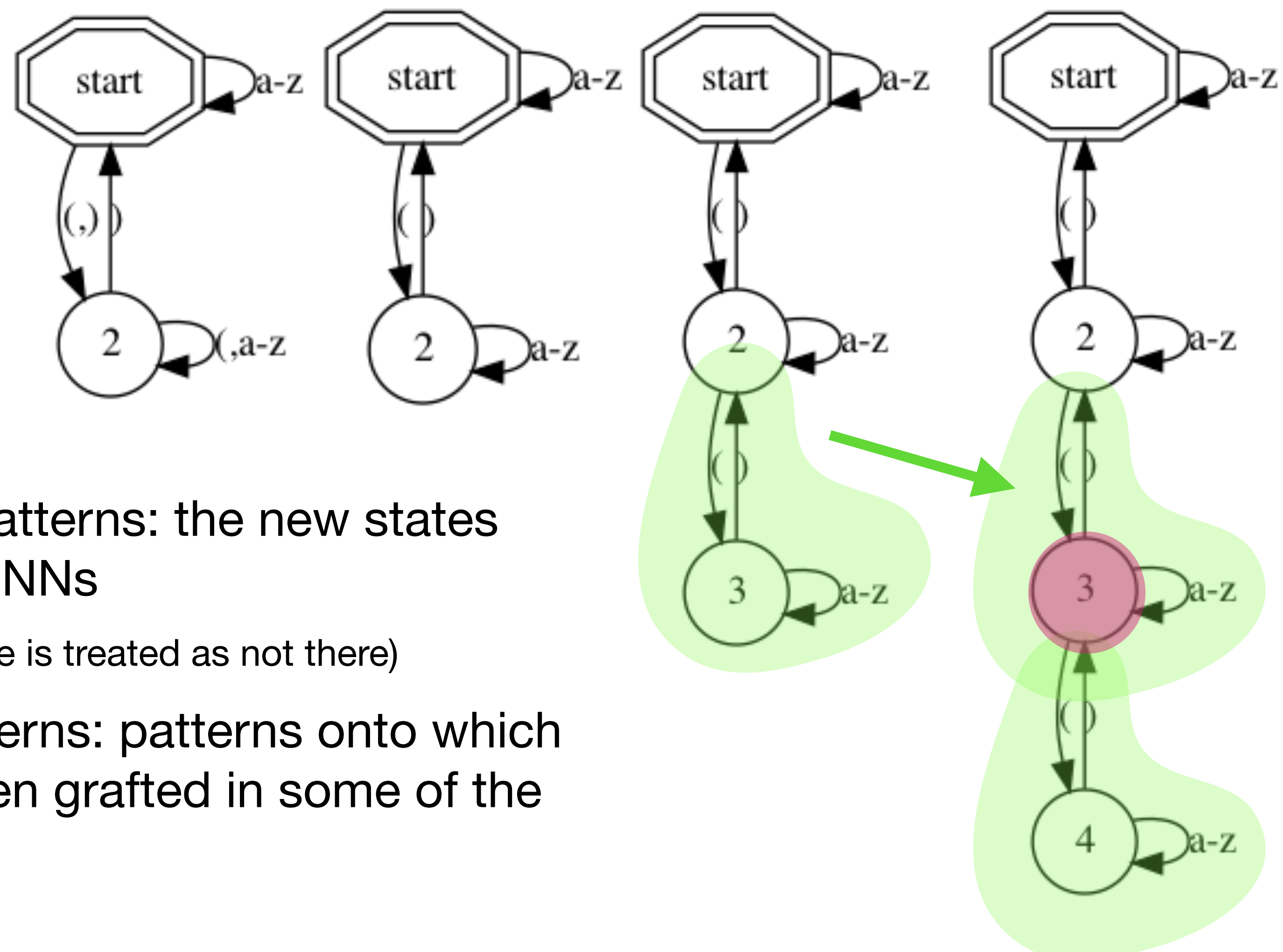
1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)



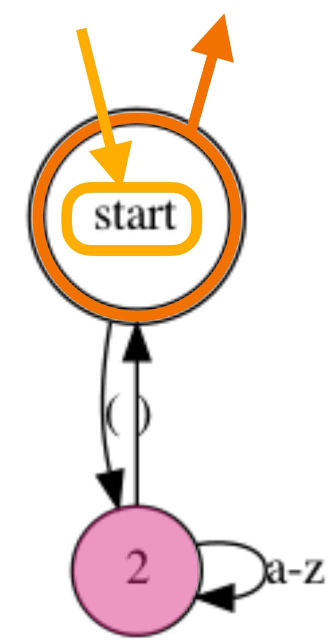
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Recovering a Pattern Rule Set



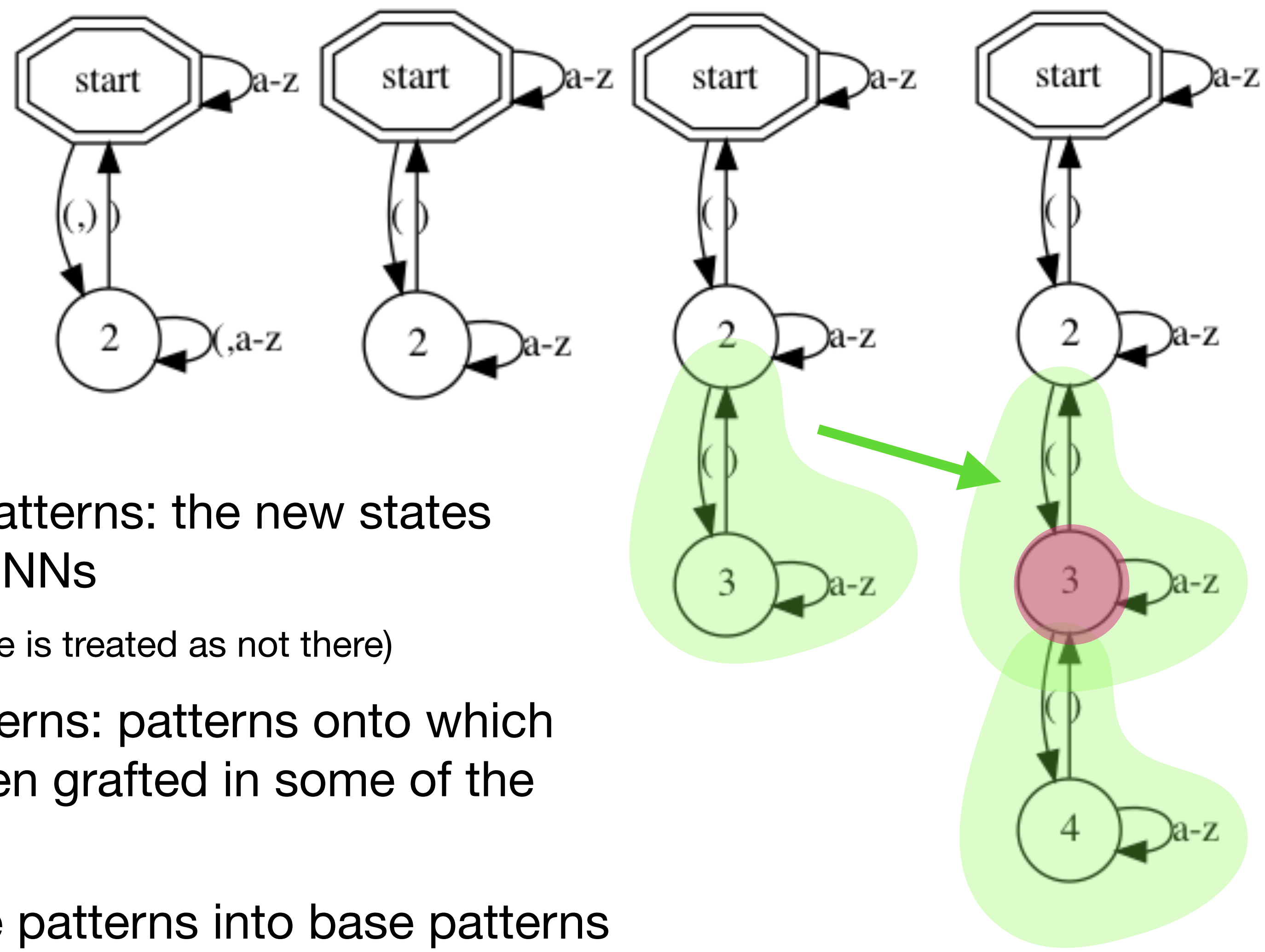
1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions



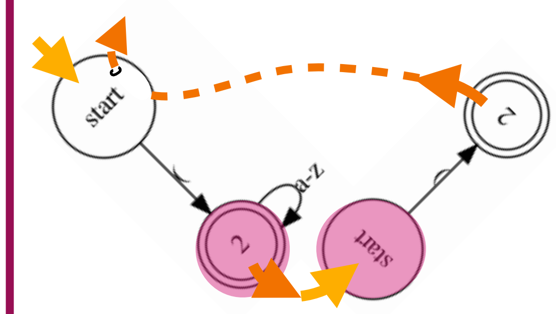
RNNs: Extraction: CFGs: **Pattern Rule Sets**

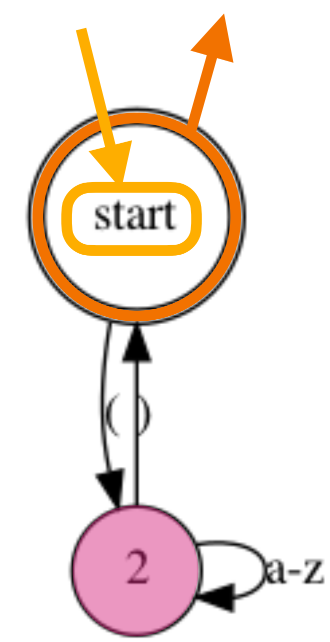
Yellin and Weiss (2021)

Recovering a Pattern Rule Set



1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
    1. Split composite patterns into base patterns according to observed join state.

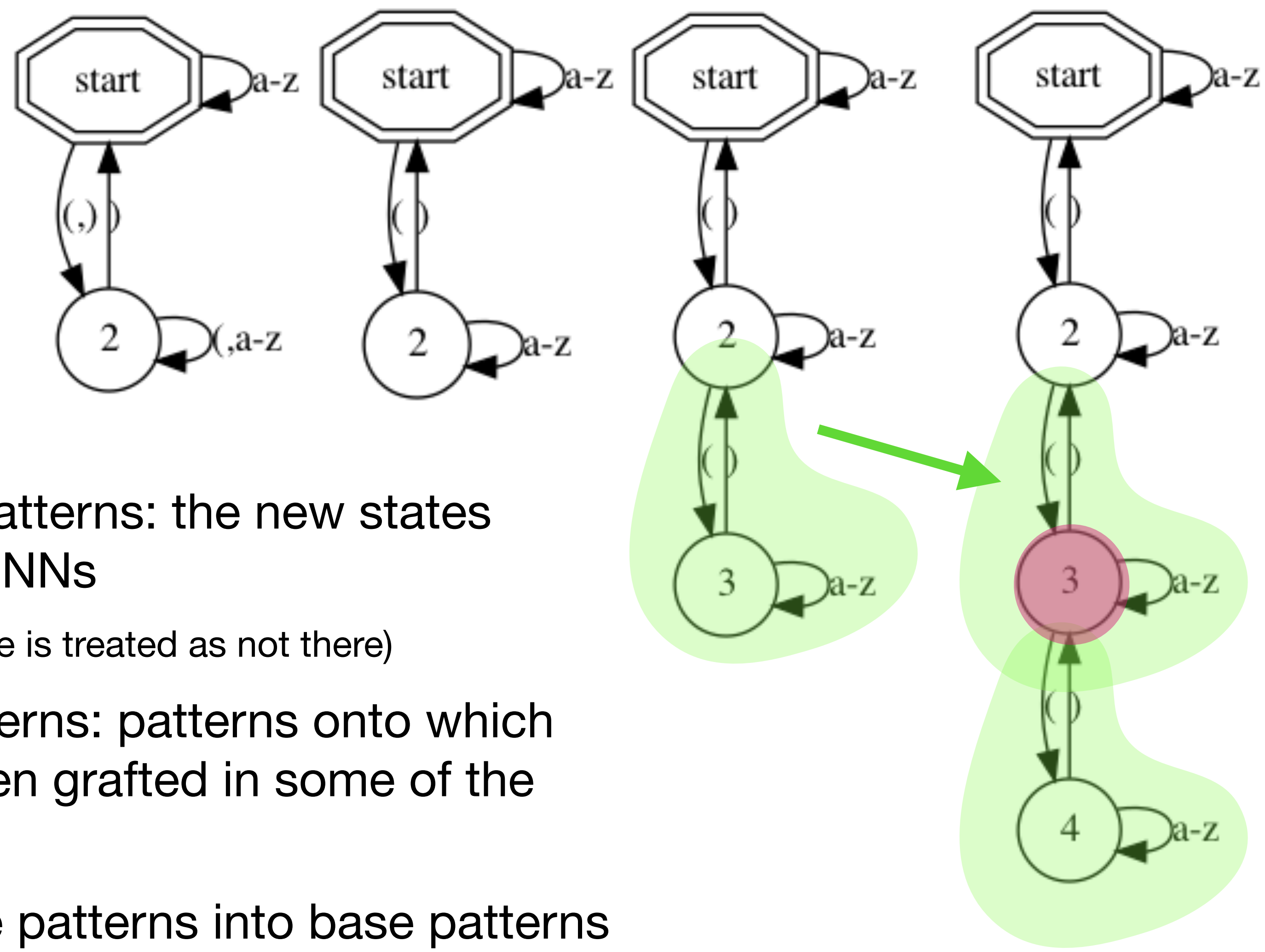




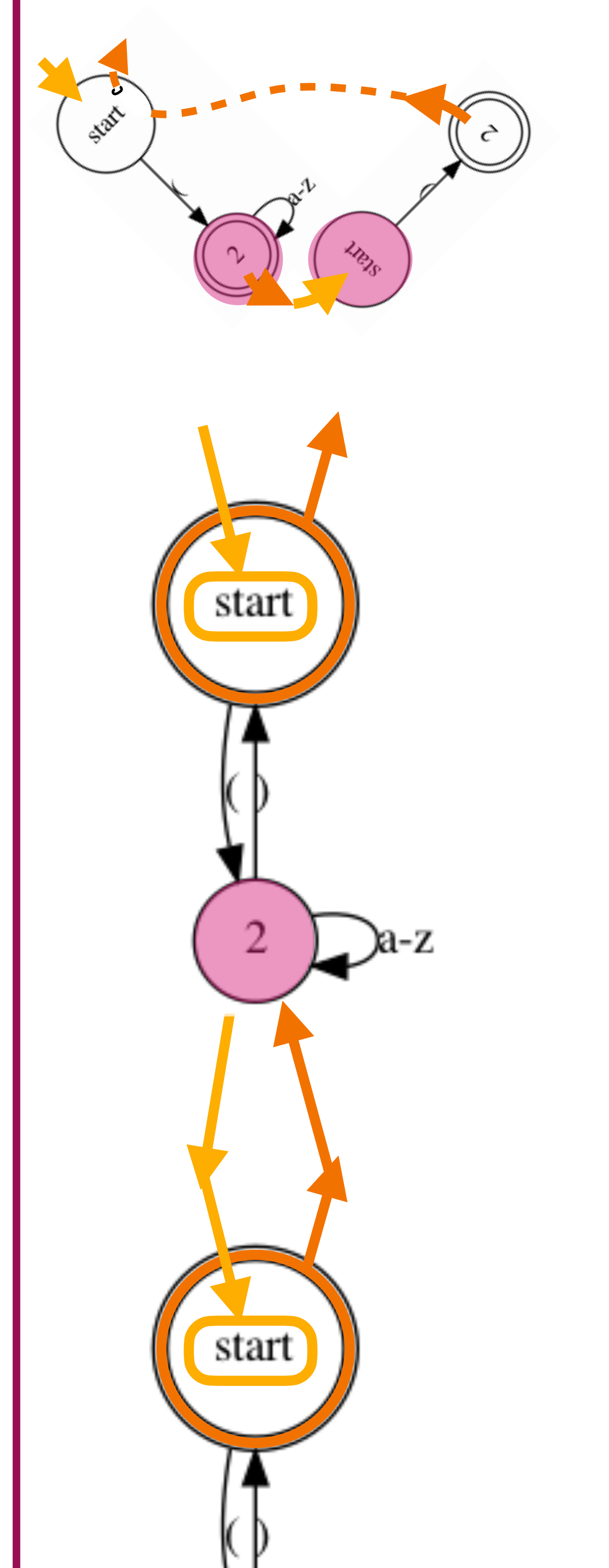
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Recovering a Pattern Rule Set



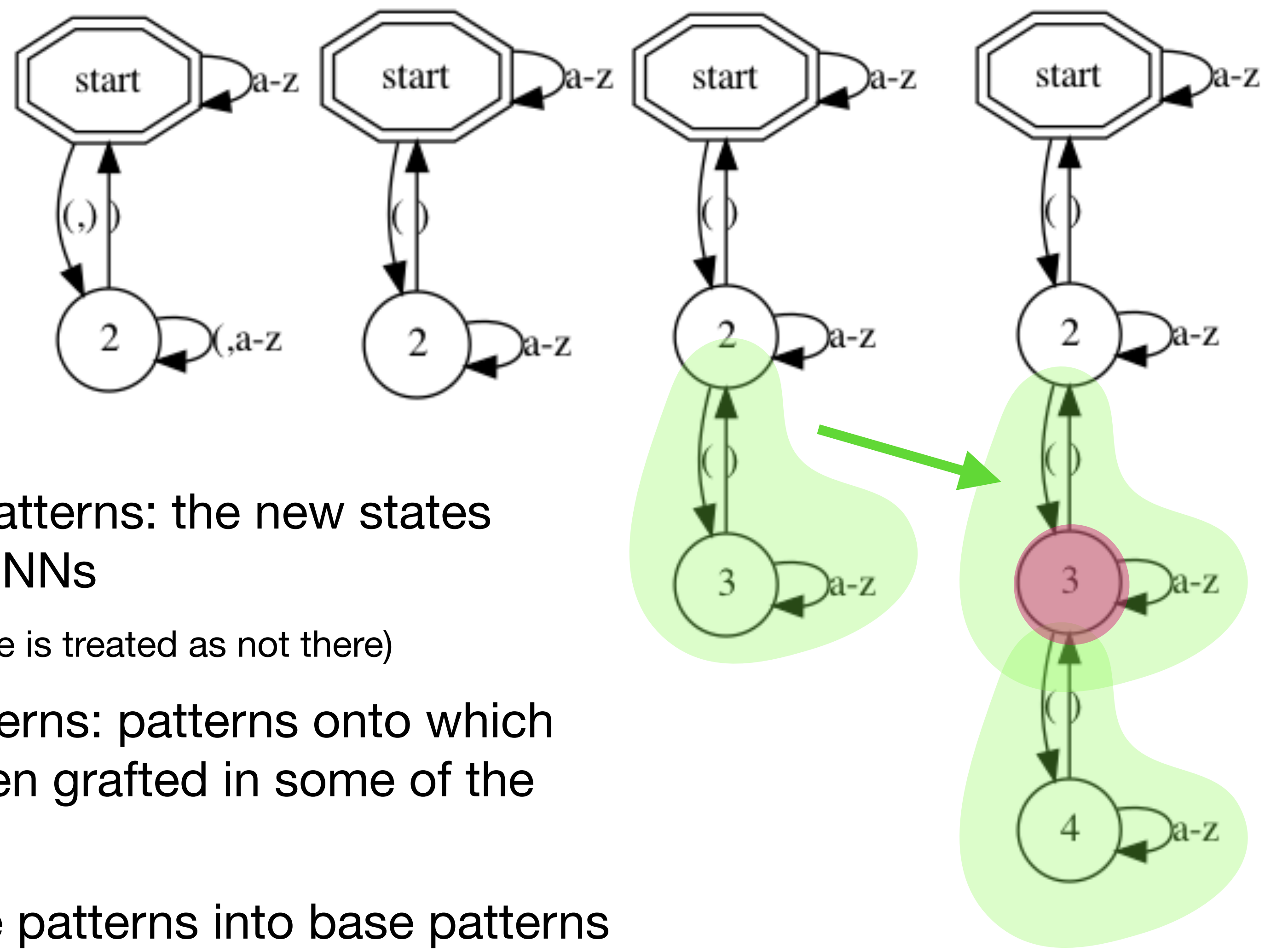
1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
    1. Split composite patterns into base patterns according to observed join state.
    2. Record which pattern was grafted on - i.e., which rule was used.



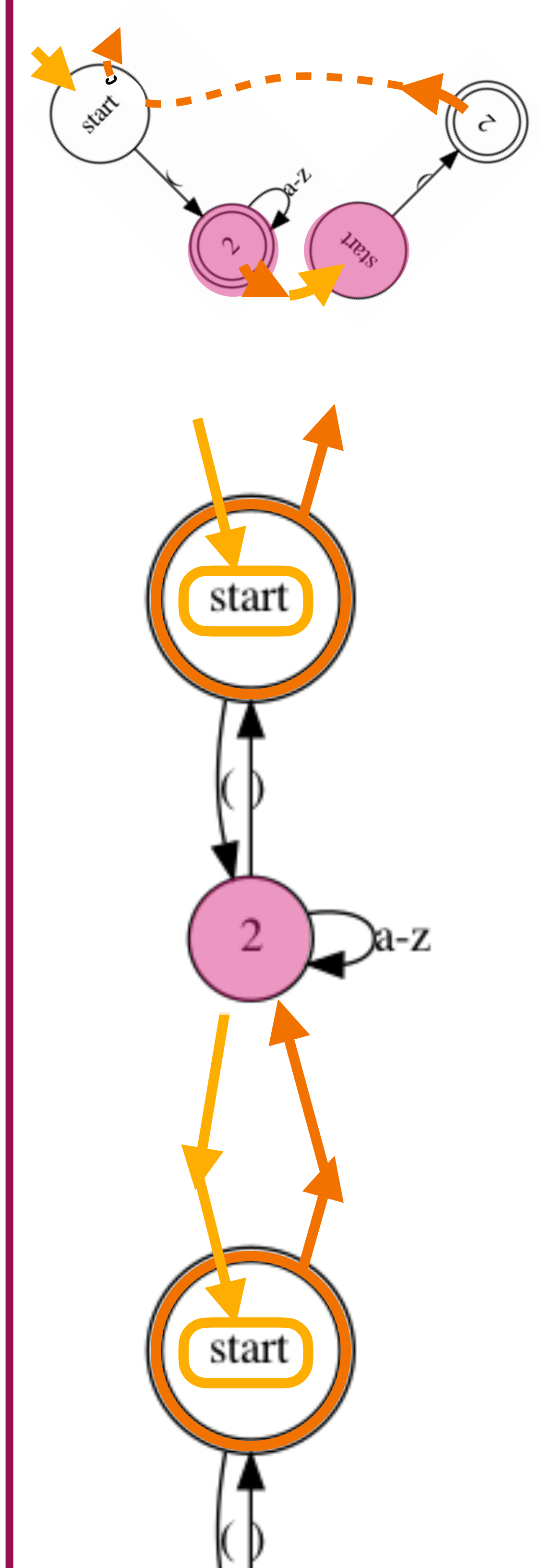
RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

Recovering a Pattern Rule Set



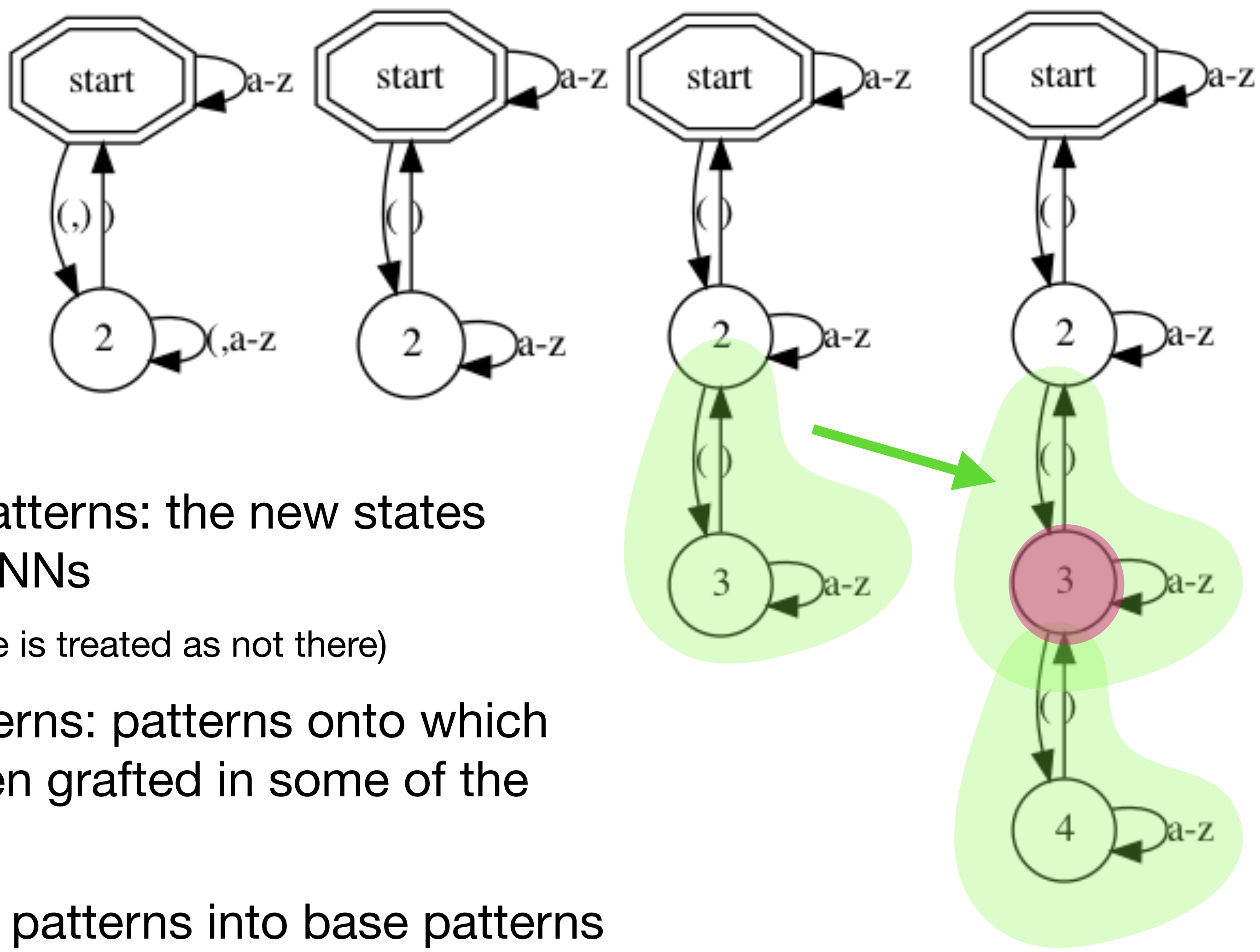
1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
    1. Split composite patterns into base patterns according to observed join state.
    2. Record which pattern was grafted on - i.e., which rule was used.
3. To handle noise: have threshold, and only keep patterns and rules that have frequency above that threshold

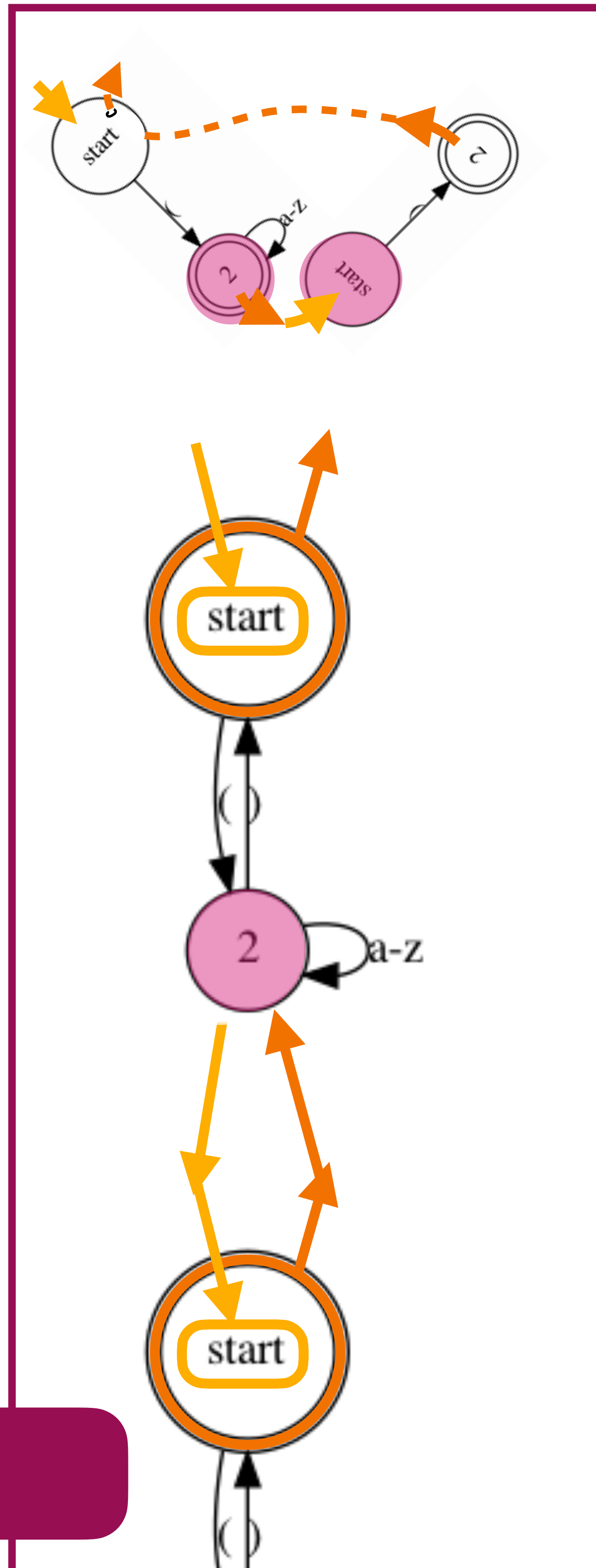


RNNs: Extraction: CFGs: **Pattern Rule Sets**

Yellin and Weiss (2021)

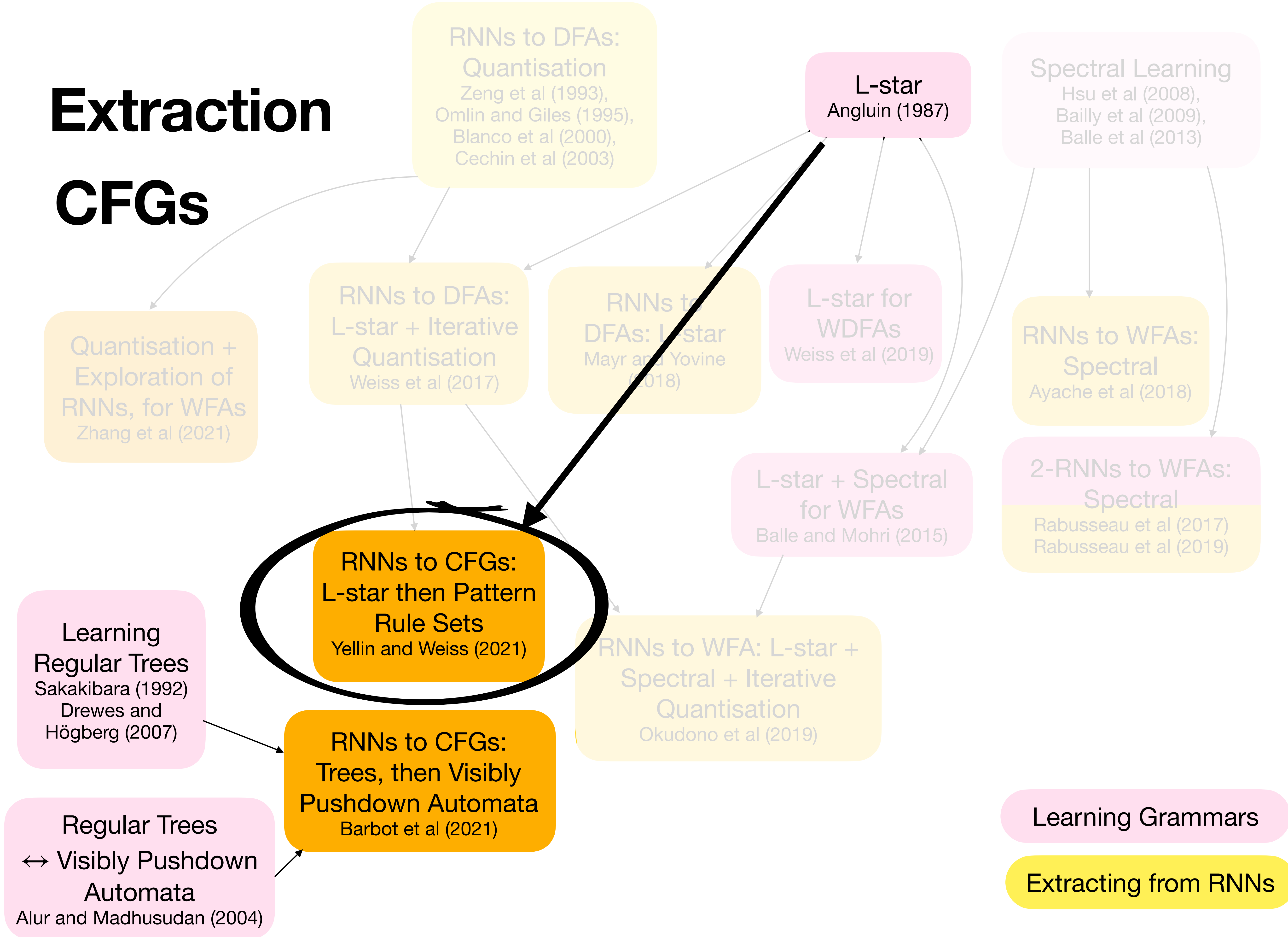
Recovering a Pattern Rule Set





1. Identify (some of) the patterns: the new states between consecutive RNNs
    1. (Note that reject state is treated as not there)
2. Identify composite patterns: patterns onto which other patterns have been grafted in some of the expansions
    1. Split composite patterns into base patterns according to observed join state.
    2. Record which pattern was grafted on - i.e., which rule was used.
3. To handle noise: have threshold, and only keep patterns and rules that have frequency above that threshold

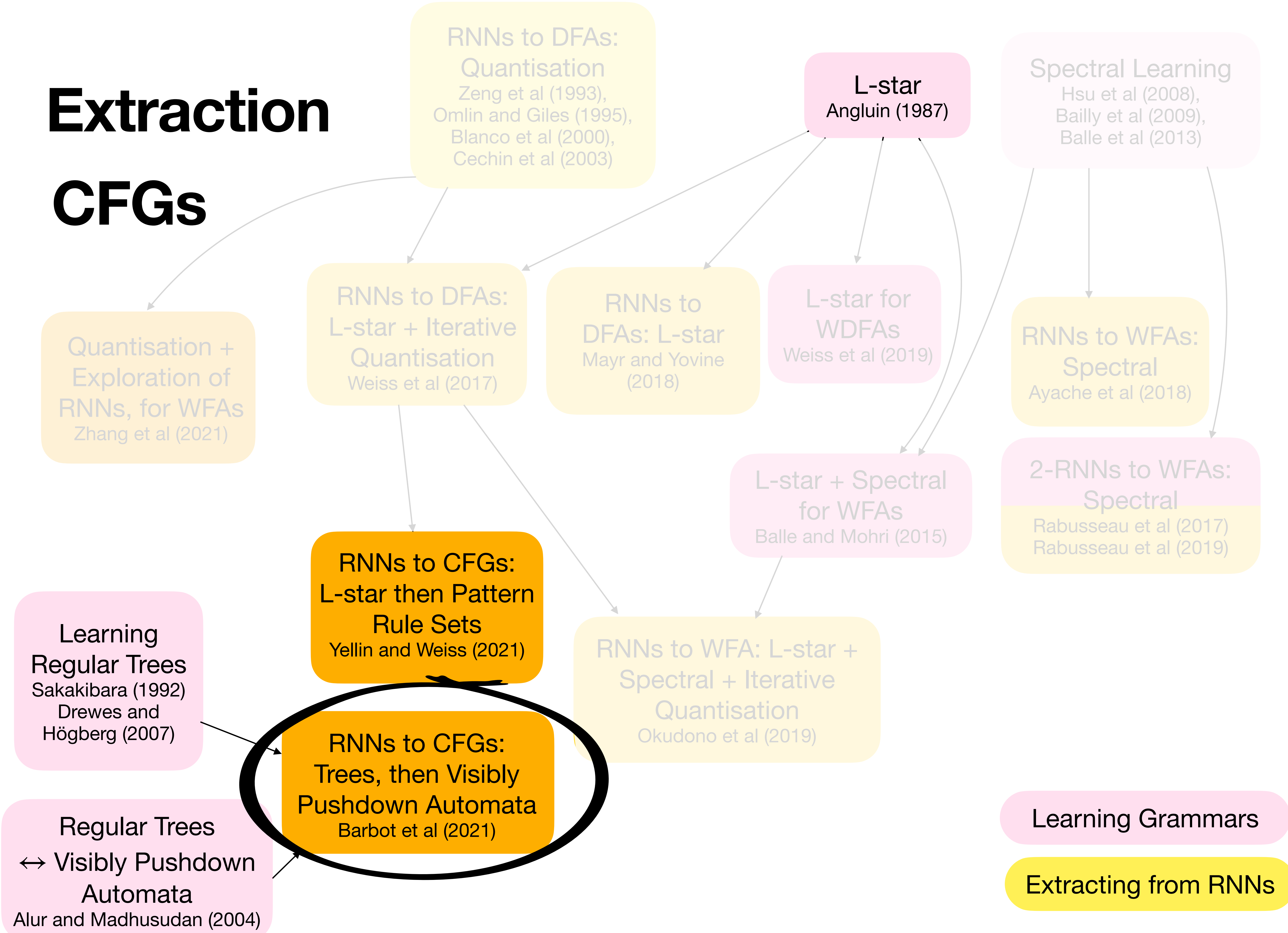
4. Convert PRS to CFG!



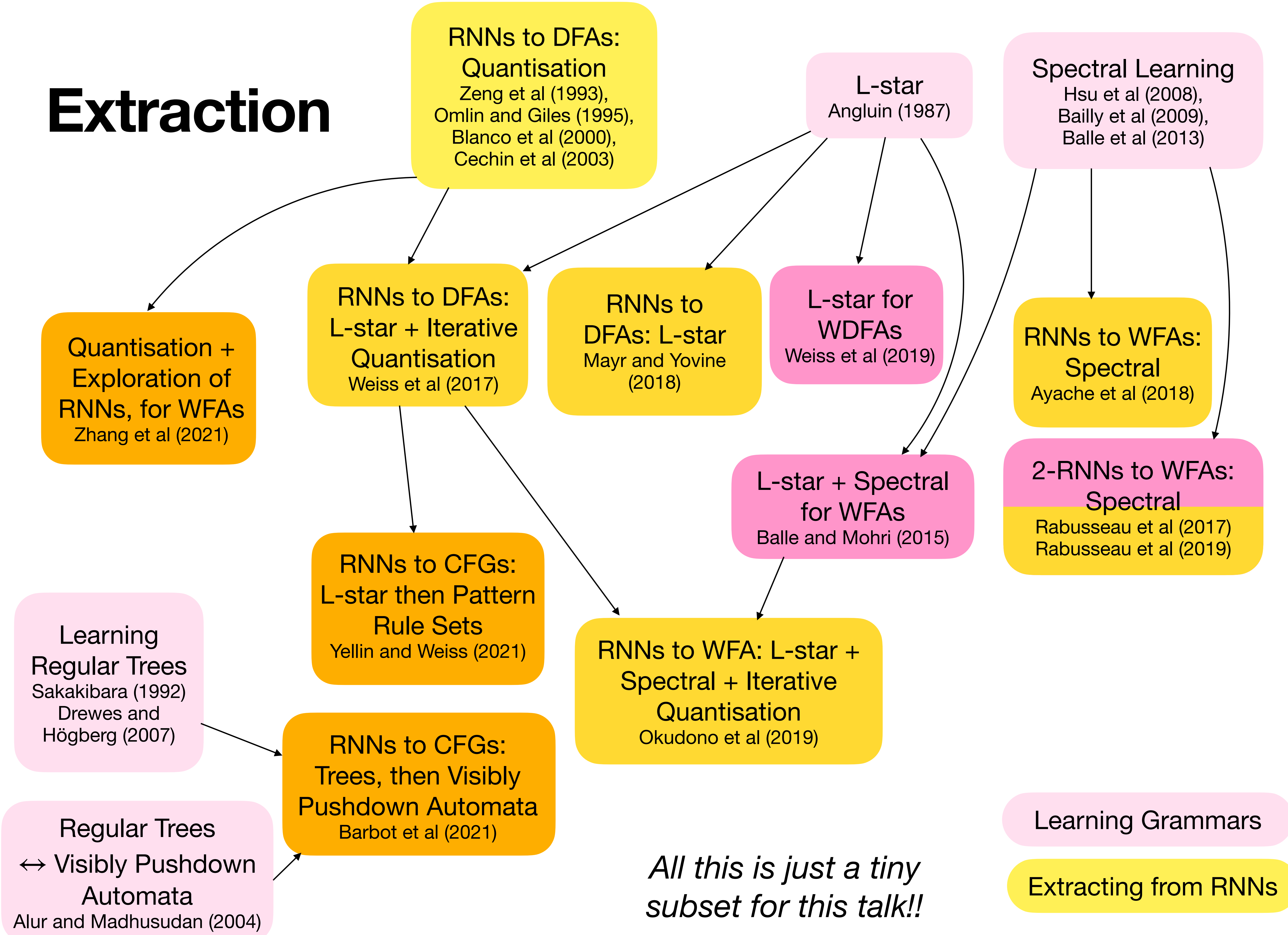
Extraction
CFGs
RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
Learning Grammars
Extracting from RNNs

Extraction

CFGs



Extraction


RNNs to DFAs: Quantisation
Zeng et al (1993), Omlin and Giles (1995), Blanco et al (2000), Cechin et al (2003)
L-star
Angluin (1987)
Spectral Learning
Hsu et al (2008), Bailly et al (2009), Balle et al (2013)
Quantisation + Exploration of RNNs, for WFAs
Zhang et al (2021)
RNNs to DFAs: L-star + Iterative Quantisation
Weiss et al (2017)
RNNs to DFAs: L-star
Mayr and Yovine (2018)
L-star for WDFAs
Weiss et al (2019)
RNNs to WFAs: Spectral
Ayache et al (2018)
L-star + Spectral for WFAs
Balle and Mohri (2015)
2-RNNs to WFAs: Spectral
Rabusseau et al (2017)
Rabusseau et al (2019)
RNNs to CFGs: L-star then Pattern Rule Sets
Yellin and Weiss (2021)
Learning Regular Trees
Sakakibara (1992)
Drewes and Högberg (2007)
RNNs to WFA: L-star + Spectral + Iterative Quantisation
Okudono et al (2019)
RNNs to CFGs: Trees, then Visibly Pushdown Automata
Barbot et al (2021)
Regular Trees ↔ Visibly Pushdown Automata
Alur and Madhusudan (2004)
All this is just a tiny subset for this talk!!
Learning Grammars
Extracting from RNNs

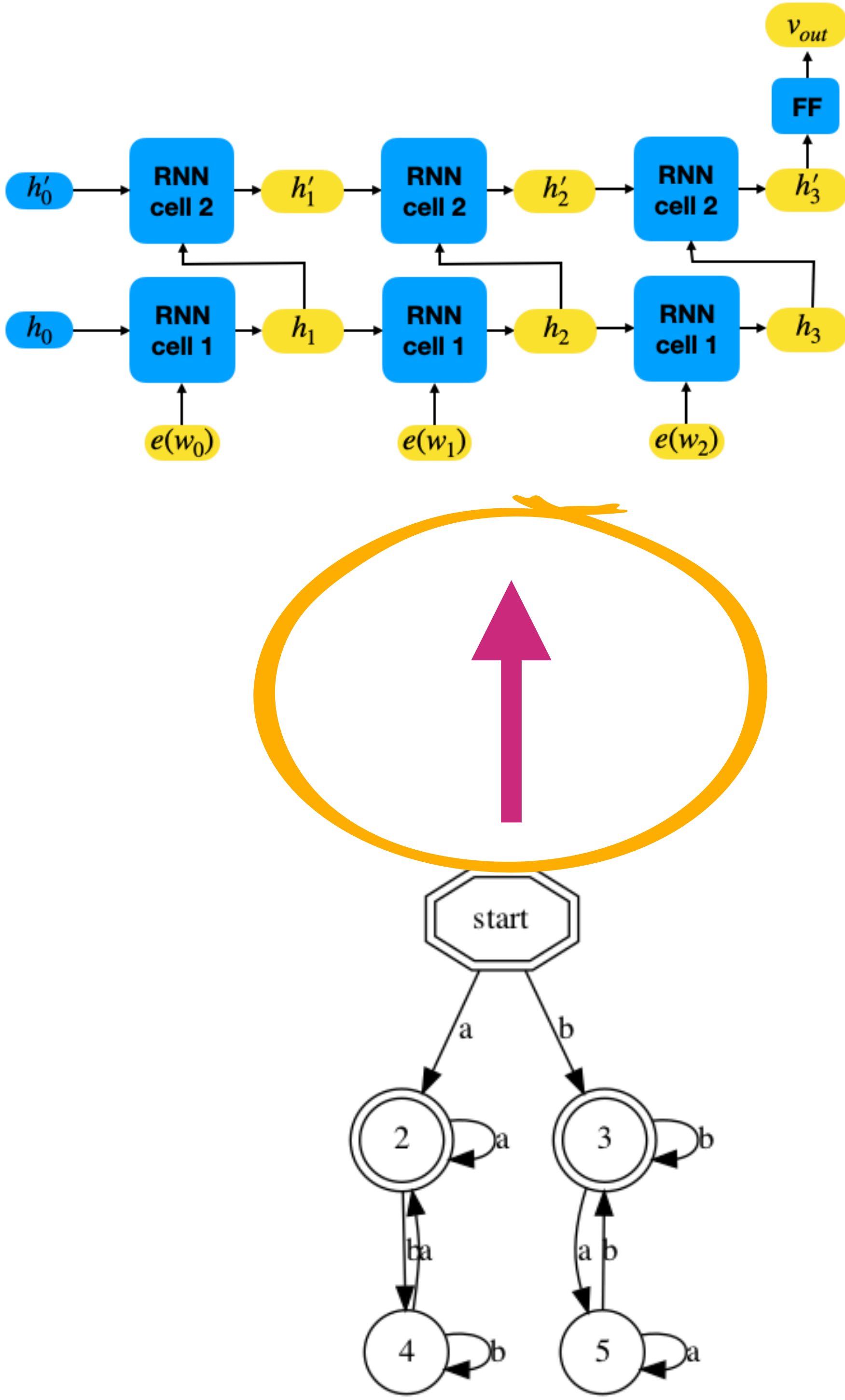
Overview

Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers

- Introduction
- A formal abstraction



RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

RNNs are like DFAs
Cleeremans et al 1989

RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

RNNs are like DFAs
Cleeremans et al 1989

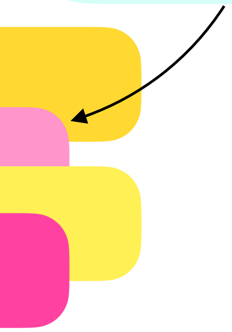
RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

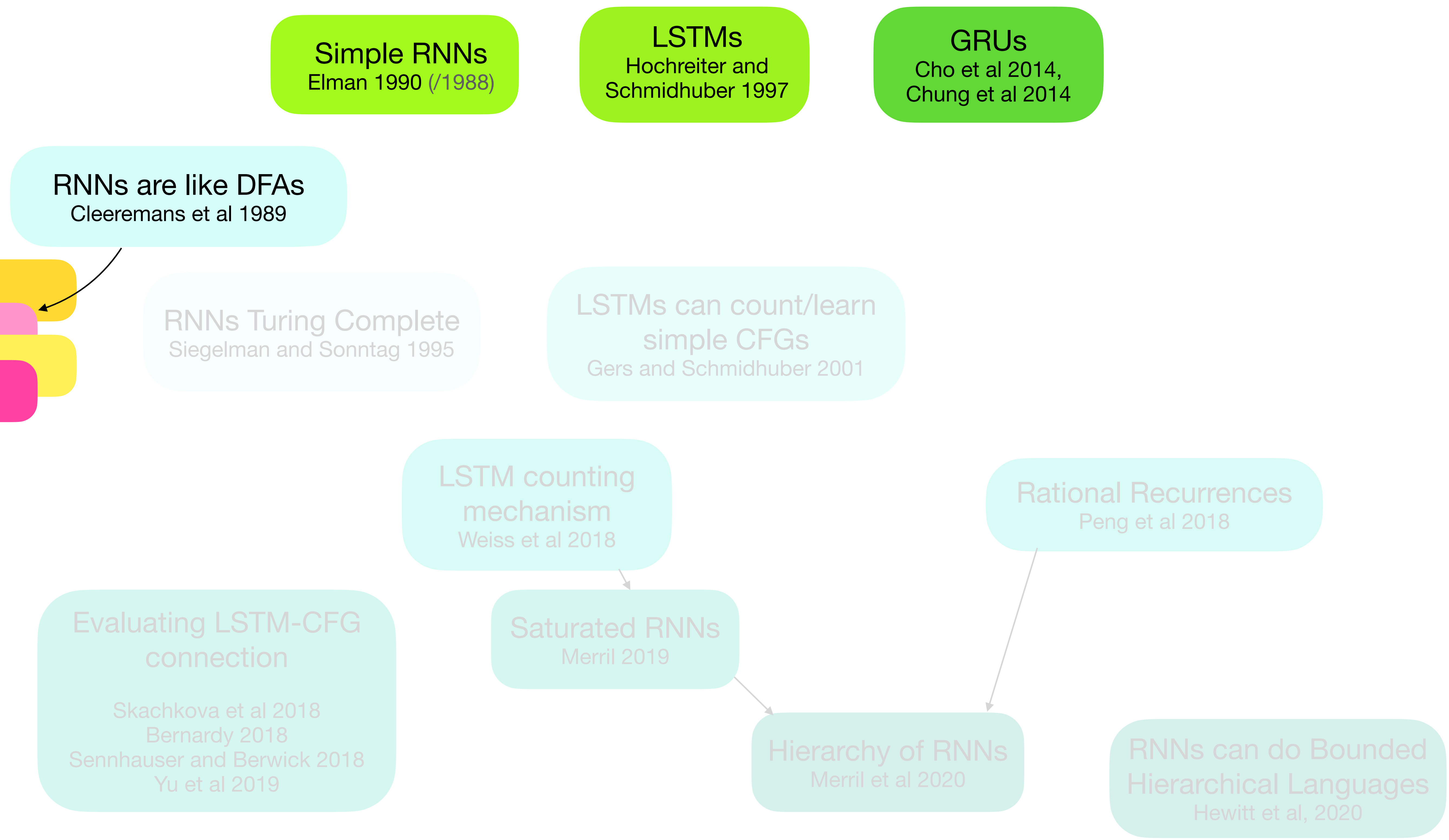
LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014, Chung et al 2014

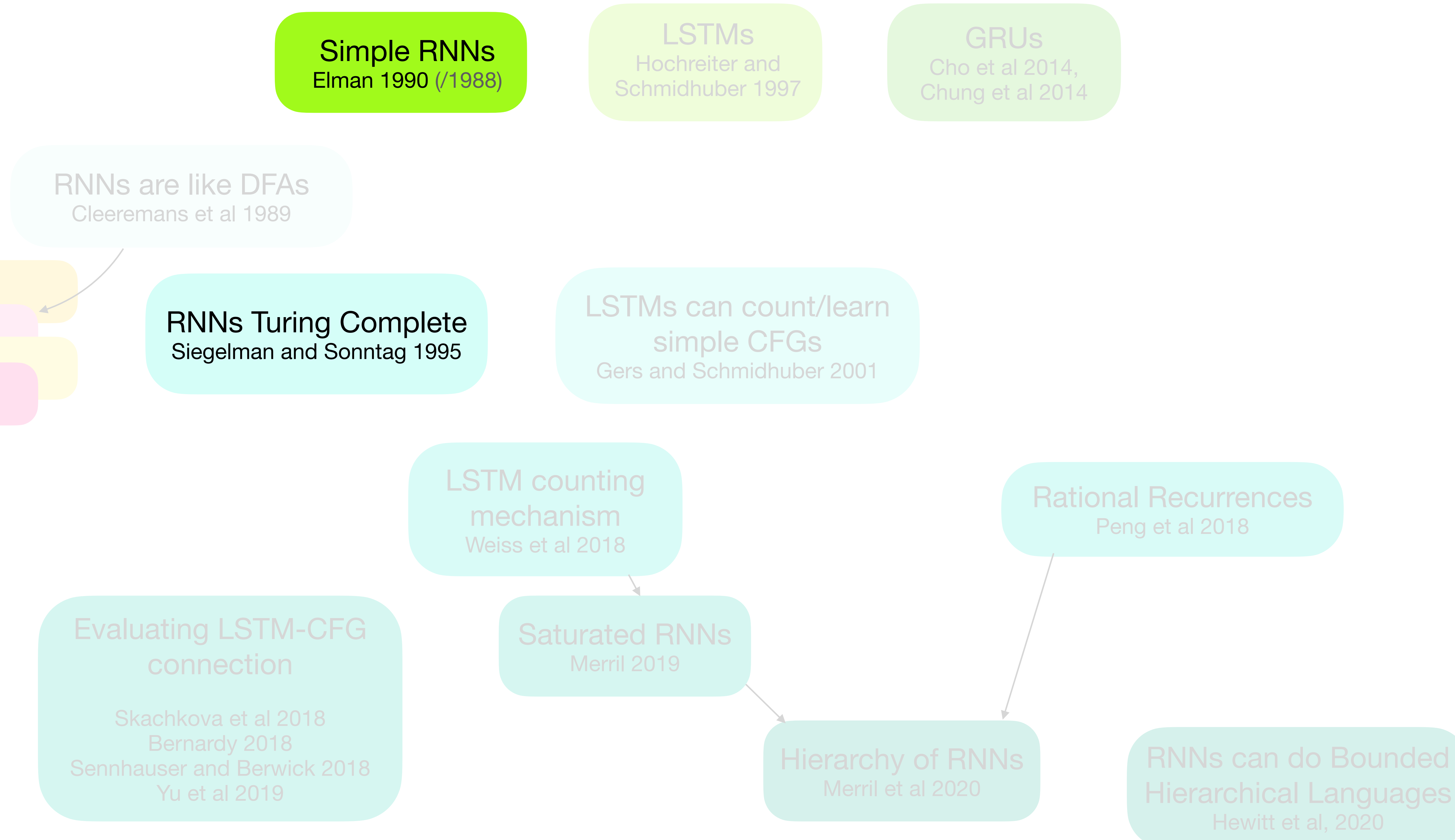
RNNs are like DFAs
Cleeremans et al 1989



RNNs: Expressive Power



RNNs: Expressive Power



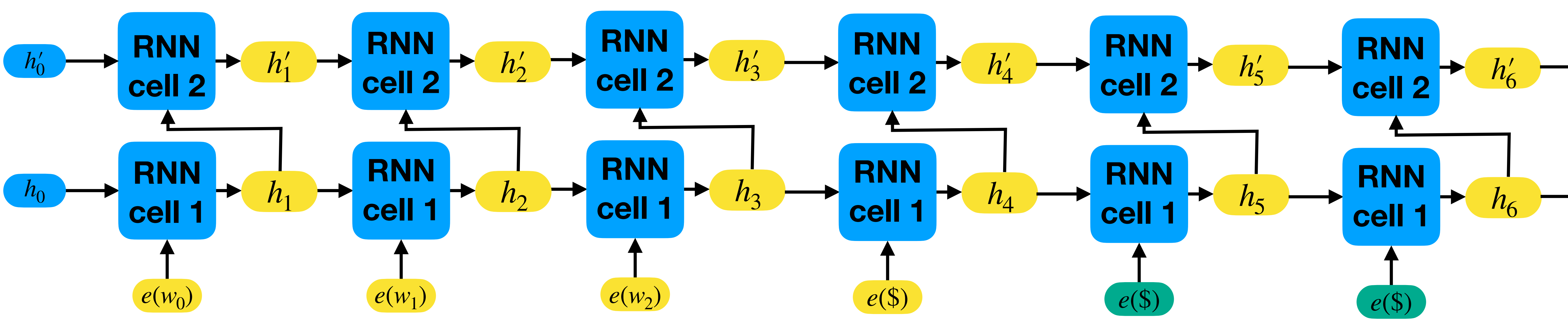
RNNs: Expressive Power: Theory

RNNs are Turing Complete:

Given infinite precision, RNNs can emulate pushing and popping to/from stacks in their hidden state. Thus, given also infinite time, they can simulate any Turing Machine

On the computational power of Neural Nets

Siegelmann and Sonntag (1995)



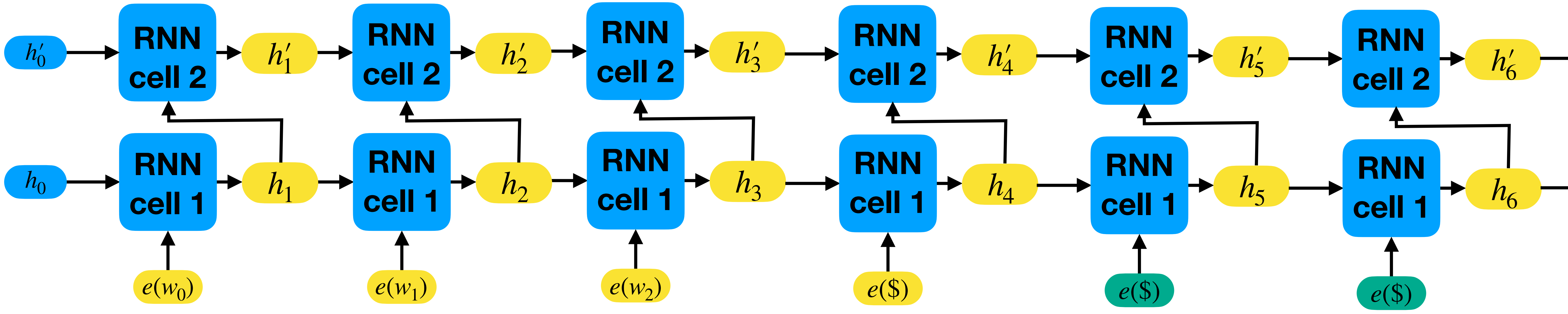
RNNs: Expressive Power: Theory

RNNs are Turing Complete:

Given infinite precision, RNNs can emulate pushing and popping to/from stacks in their hidden state. Thus, given also infinite time, they can simulate any Turing Machine

On the computational power of Neural Nets

Siegelmann and Sonntag (1995)



RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014,
Chung et al 2014

RNNs are like DFAs
Cleeremans et al 1989

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection

Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

Saturated RNNs
Merril 2019

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

RNNs: Expressive Power

Simple RNNs
Elman 1990 (/1988)

LSTMs
Hochreiter and Schmidhuber 1997

GRUs
Cho et al 2014,
Chung et al 2014

RNNs are like DFAs
Cleeremans et al 1989

RNNs Turing Complete
Siegelman and Sonntag 1995

LSTMs can count/learn simple CFGs
Gers and Schmidhuber 2001

LSTM counting mechanism
Weiss et al 2018

Rational Recurrences
Peng et al 2018

Evaluating LSTM-CFG connection
Skachkova et al 2018
Bernardy 2018
Sennhauser and Berwick 2018
Yu et al 2019

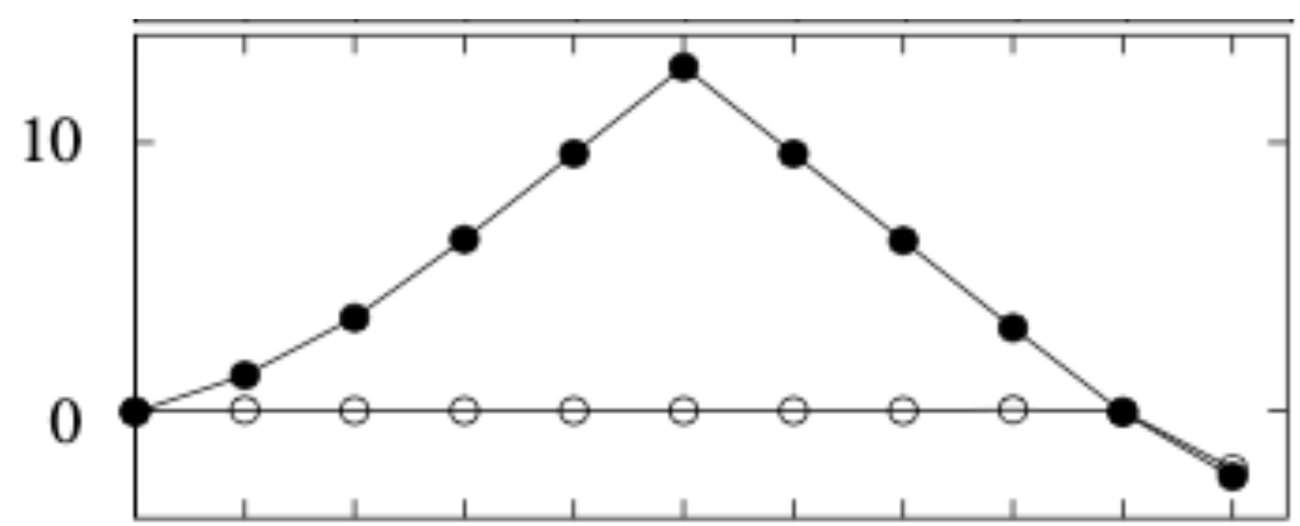
Saturated RNNs
Merril 2019

Hierarchy of RNNs
Merril et al 2020

RNNs can do Bounded Hierarchical Languages
Hewitt et al, 2020

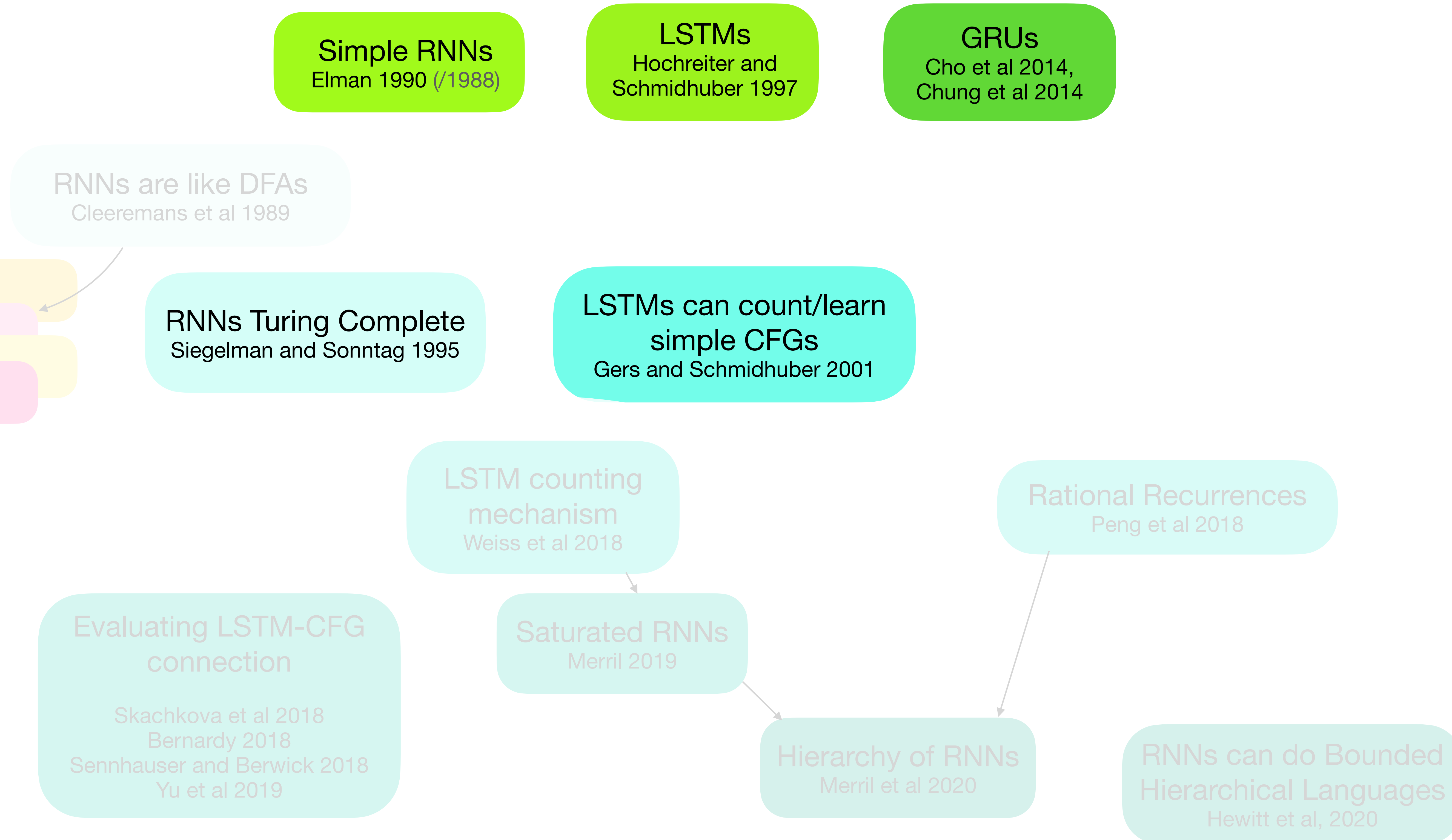
RNNs: Expressive Power: Practice

LSTMs can count

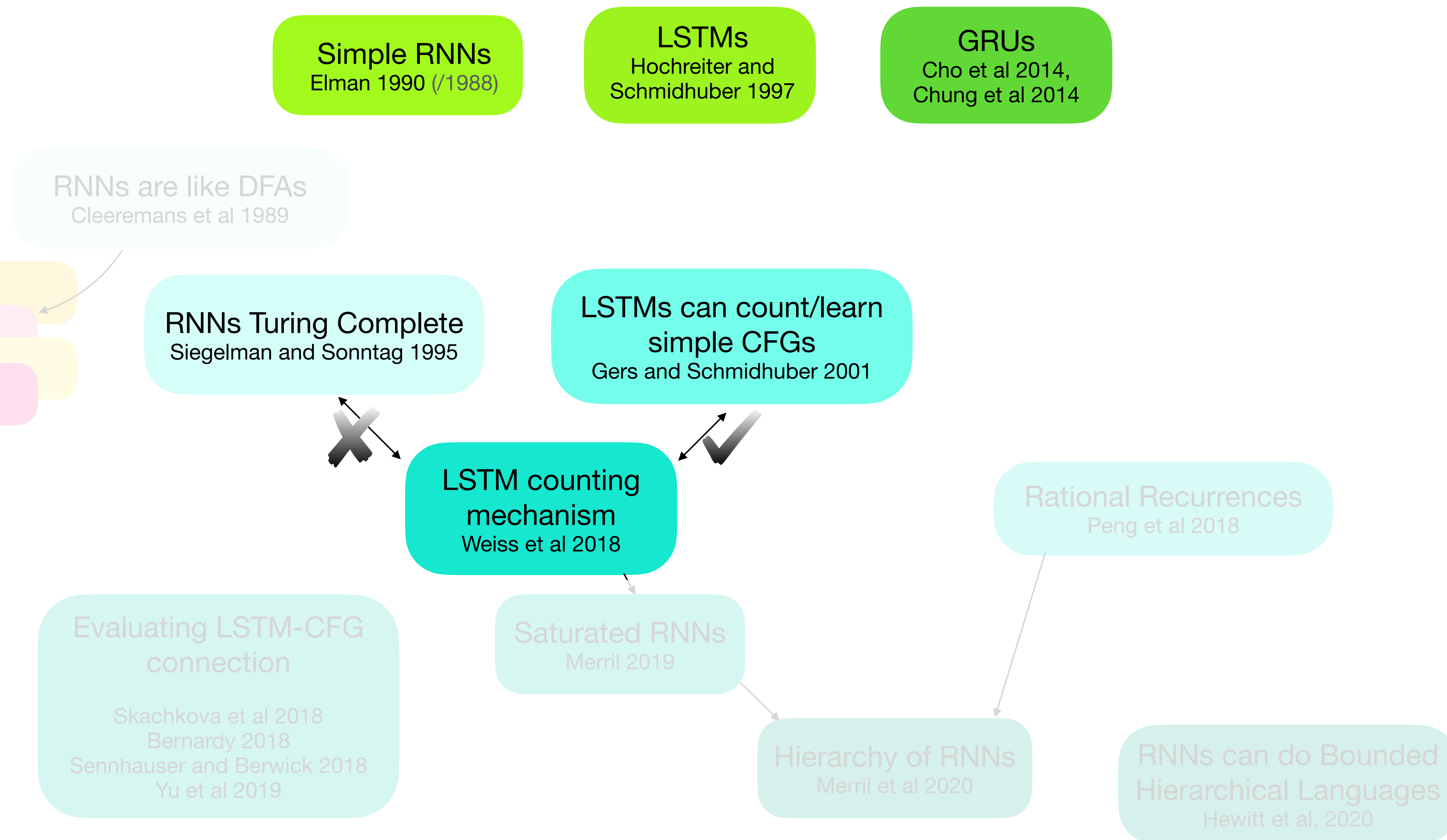


LSTM recurrent networks learn simple context-free and context-sensitive languages
Gers and Schmidhuber, 2001

RNNs: Expressive Power

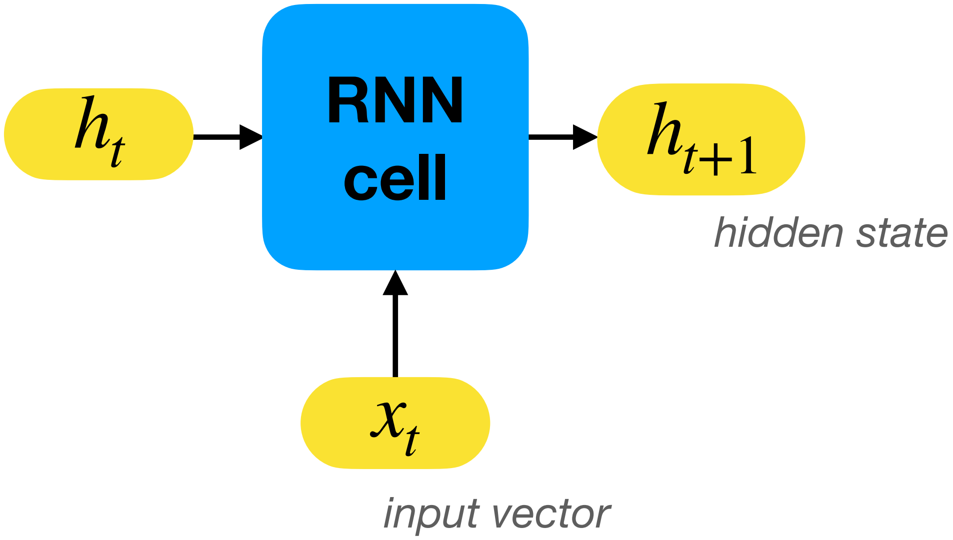


RNNs: Expressive Power



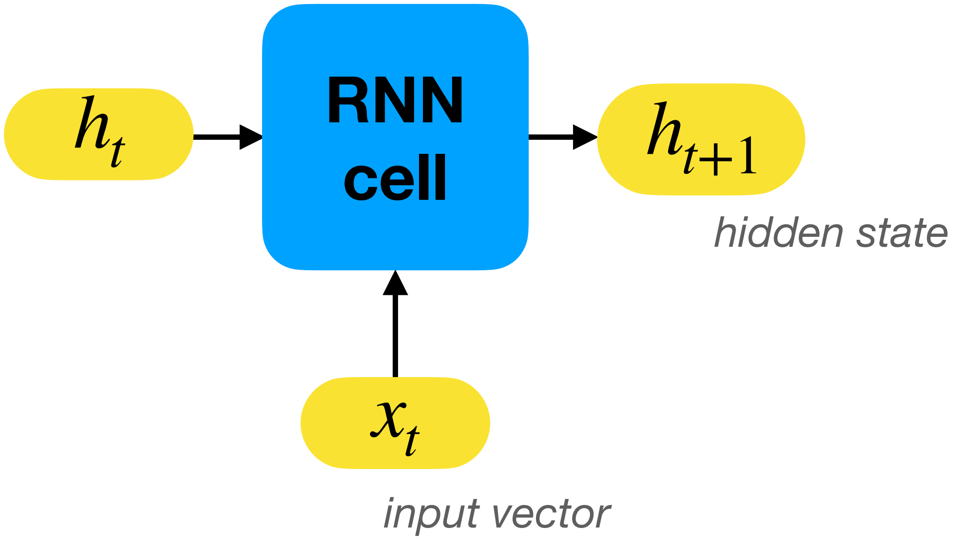
LSTMs: Counting Mechanism

Simple RNN $h_{t+1} = \tanh(W^h h_t + W^x x_t + b)$ Elman (1990)



LSTMs: Counting Mechanism

Simple RNN $h_{t+1} = \tanh(W^h h_t + W^x x_t + b)$ Elman (1990)



LSTMs: Counting Mechanism

Simple RNN $$h_{t+1} = \tanh(W^h h_t + W^x x_t + b)$$ Elman (1990)

GRU

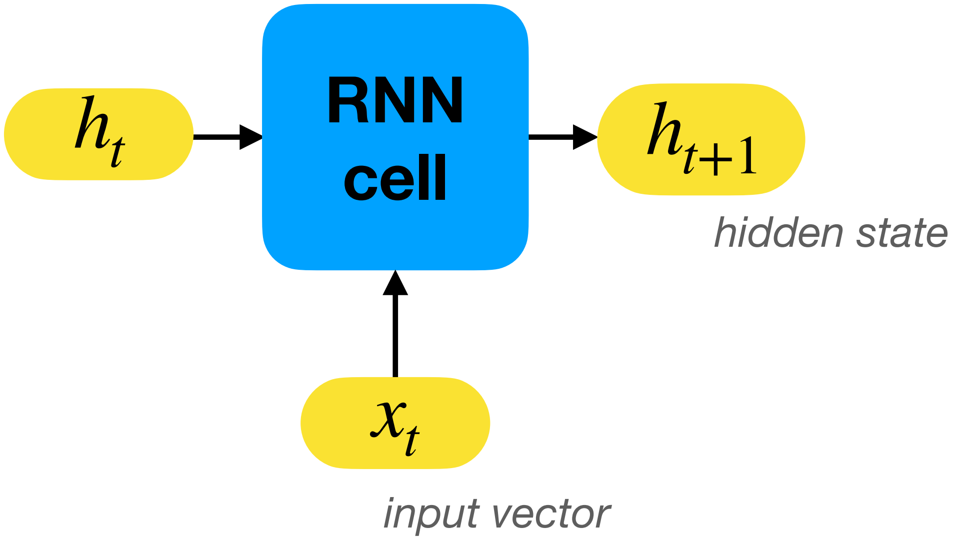
$$z_t = \sigma(W^z x_t + U^z h_{t-1} + b^z)$$
$$r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r)$$
$$\tilde{h}_t = \tanh(W^h x_t + U^h (r_t \circ h_{t-1}) + b^h)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

Cho et al (2014), Chung et al (2014)

LSTM

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$
$$\tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1} + b^c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ g(c_t)$$

Hochreiter and Schmidhuber (1997)



LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$$z_t = \sigma(W^z x_t + U^z h_{t-1} + b^z)$$

$$r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r)$$

$$\tilde{h}_t = \tanh(W^h x_t + U^h (r_t \circ h_{t-1}) + b^h)$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

LSTM

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$\tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1} + b^c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ g(c_t)$$

gates

candidate vectors

update functions

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$$z_t \in (0,1)$$
$$r_t \in (0,1)$$
$$\tilde{h}_t = \tanh(W^h x_t + U^h(r_t \circ h_{t-1}) + b^h)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

LSTM

$$f_t \in (0,1)$$
$$i_t \in (0,1)$$
$$o_t \in (0,1)$$
$$\tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1} + b^c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ g(c_t)$$

gates

candidate vectors

update functions

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$$z_t \in (0,1)$$
$$r_t \in (0,1)$$
$$\tilde{h}_t \in (-1,1)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

LSTM

$$f_t \in (0,1)$$
$$i_t \in (0,1)$$
$$o_t \in (0,1)$$
$$\tilde{c}_t \in (-1,1)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ g(c_t)$$

gates

candidate vectors

update functions

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU	LSTM
$z_t \in (0,1)$	$f_t \in (0,1)$
$r_t \in (0,1)$	$i_t \in (0,1)$
	$o_t \in (0,1)$
$\tilde{h}_t \in (-1,1)$	$\tilde{c}_t \in (-1,1)$
$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$	$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$
	$h_t = o_t \circ g(c_t)$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU	**LSTM**
$z_t \in (0,1)$	$f_t \in (0,1)$
$r_t \in (0,1)$	$i_t \in (0,1)$
	$o_t \in (0,1)$
$\tilde{h}_t \in (-1,1)$	$\tilde{c}_t \in (-1,1)$
$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$	$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$
	$h_t = o_t \circ g(c_t)$

Interpolation

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1 - z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \in (0,1)$

$i_t \in (0,1)$

$o_t \in (0,1)$

$\tilde{c}_t \in (-1,1)$

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

$h_t = o_t \circ g(c_t)$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \in (0,1)$

$i_t \in (0,1)$

$o_t \in (0,1)$

$\tilde{c}_t \in (-1,1)$

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

$h_t = o_t \circ g(c_t)$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \in (0,1)$

$i_t \in (0,1)$

$o_t \in (0,1)$

$\tilde{c}_t \in (-1,1)$

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

$h_t = o_t \circ g(c_t)$

Addition

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1 - z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \approx 1$

$i_t \approx 1$

$o_t \in (0,1)$

$\tilde{c}_t \in (-1,1)$

$c_t \approx c_{t-1} + \tilde{c}_t$

$h_t = o_t \circ g(c_t)$

Addition

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \approx 1$

$i_t \approx 1$

$o_t \in (0,1)$

$\tilde{c}_t \approx 1$

$c_t \approx c_{t-1} + 1$

$h_t = o_t \circ g(c_t)$

Increase by 1

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \approx 1$

$i_t \approx 1$

$o_t \in (0,1)$

$\tilde{c}_t \approx -1$

$c_t \approx c_{t-1} - 1$

$h_t = o_t \circ g(c_t)$

Decrease by 1

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \approx 1$

$i_t \approx 0$

$o_t \in (0,1)$

$\tilde{c}_t \in (-1,1)$

$c_t \approx c_{t-1}$

$h_t = o_t \circ g(c_t)$

Do Nothing

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$ ...

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1-z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \approx 0$

$i_t \approx 0$

$o_t \in (0,1)$

$\tilde{c}_t \in (-1,1)$

$c_t \approx 0$

$h_t = o_t \circ g(c_t)$

Reset

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

GRU

$z_t \in (0,1)$

$r_t \in$

Bounded!

$\tilde{h}_t \in (-1,1)$

$h_t = z_t \circ h_{t-1} + (1 - z) \circ \tilde{h}_t$

Interpolation

LSTM

$f_t \approx 0$

$i_t \approx 0$

Can Count!

$o_t \in$

$\tilde{c}_t \in (-1,1)$

$c_t \approx 0$

$h_t = o_t \circ g(c_t)$

Reset

$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$

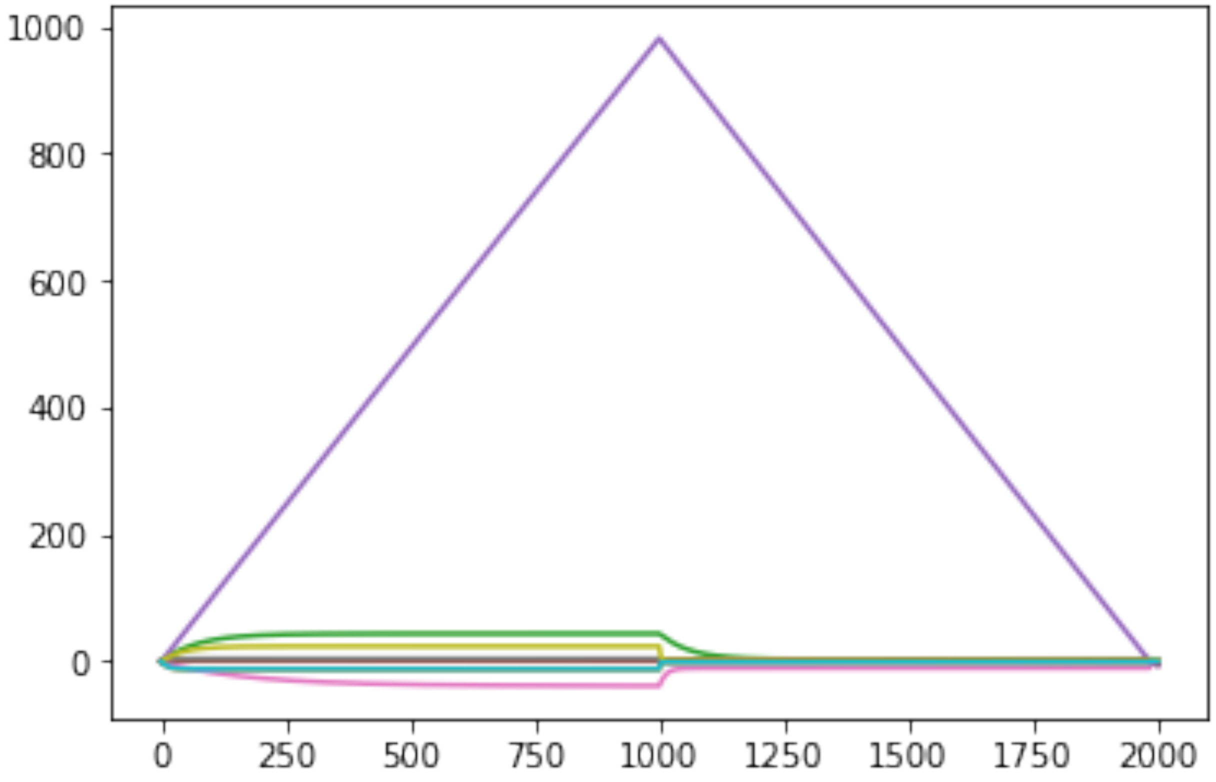
LSTMs: Counting Mechanism

LSTMs can count (and GRUs cannot)

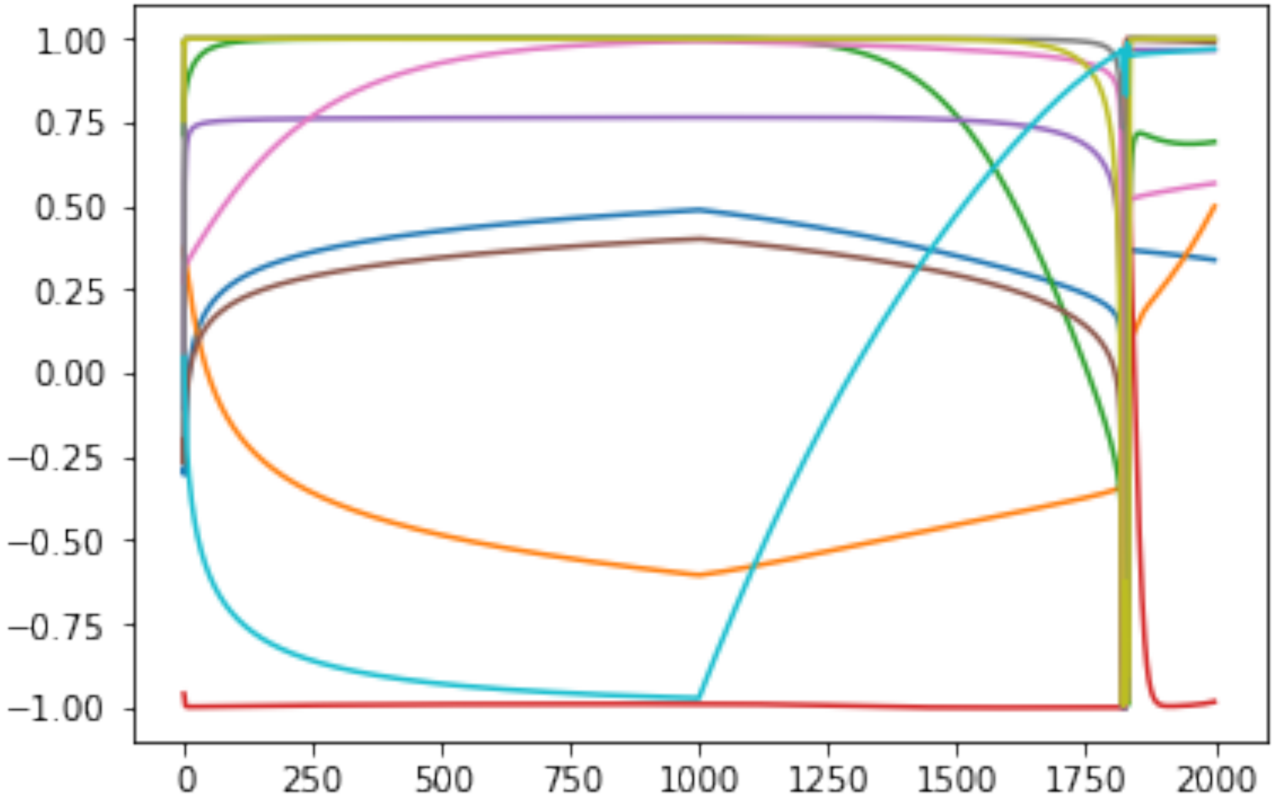
Trained $a^n b^n$**,** (on positive examples up to length 100)

Activations on $a^{1000} b^{1000}$ **:**

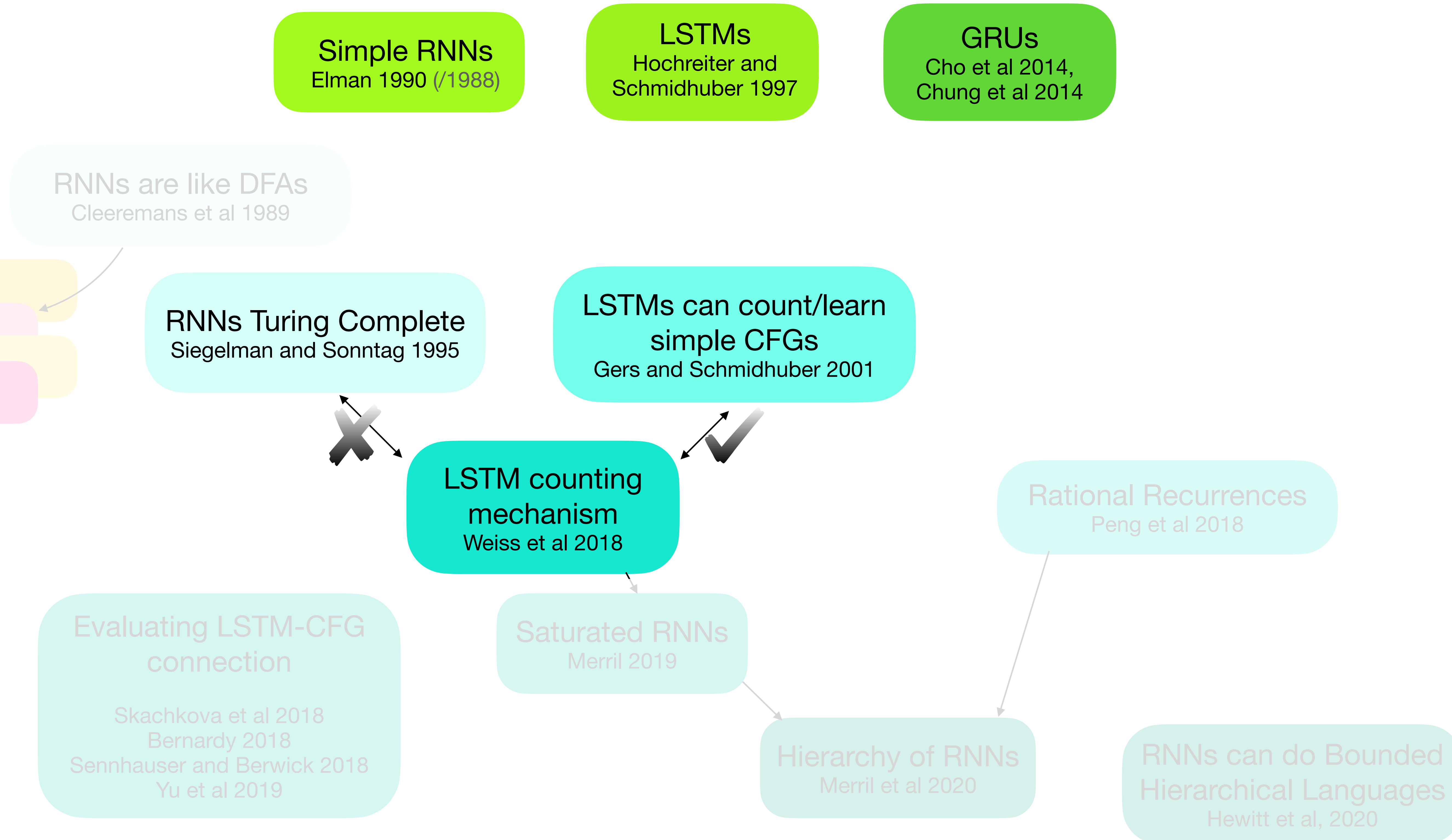
LSTM



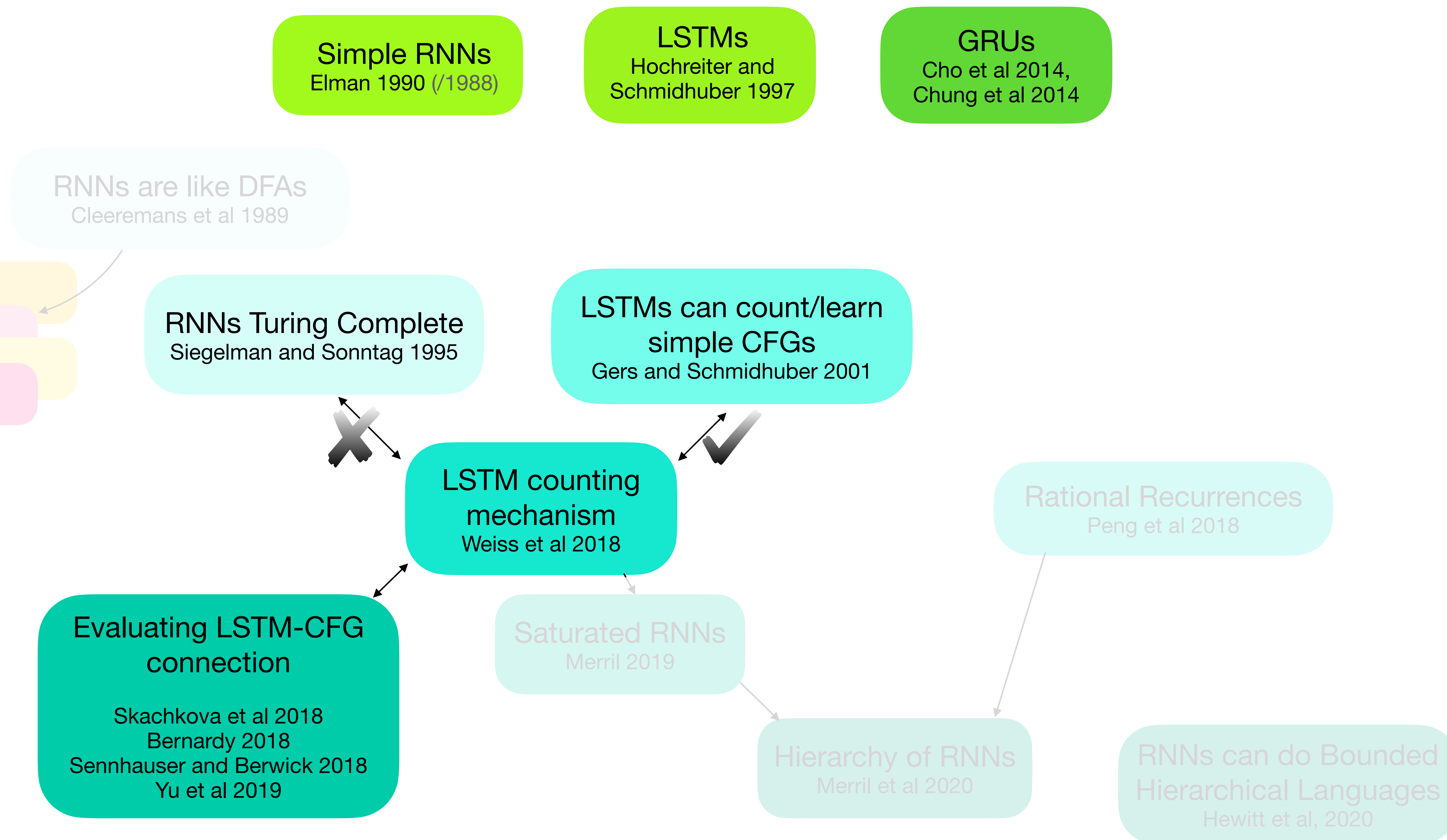
GRU



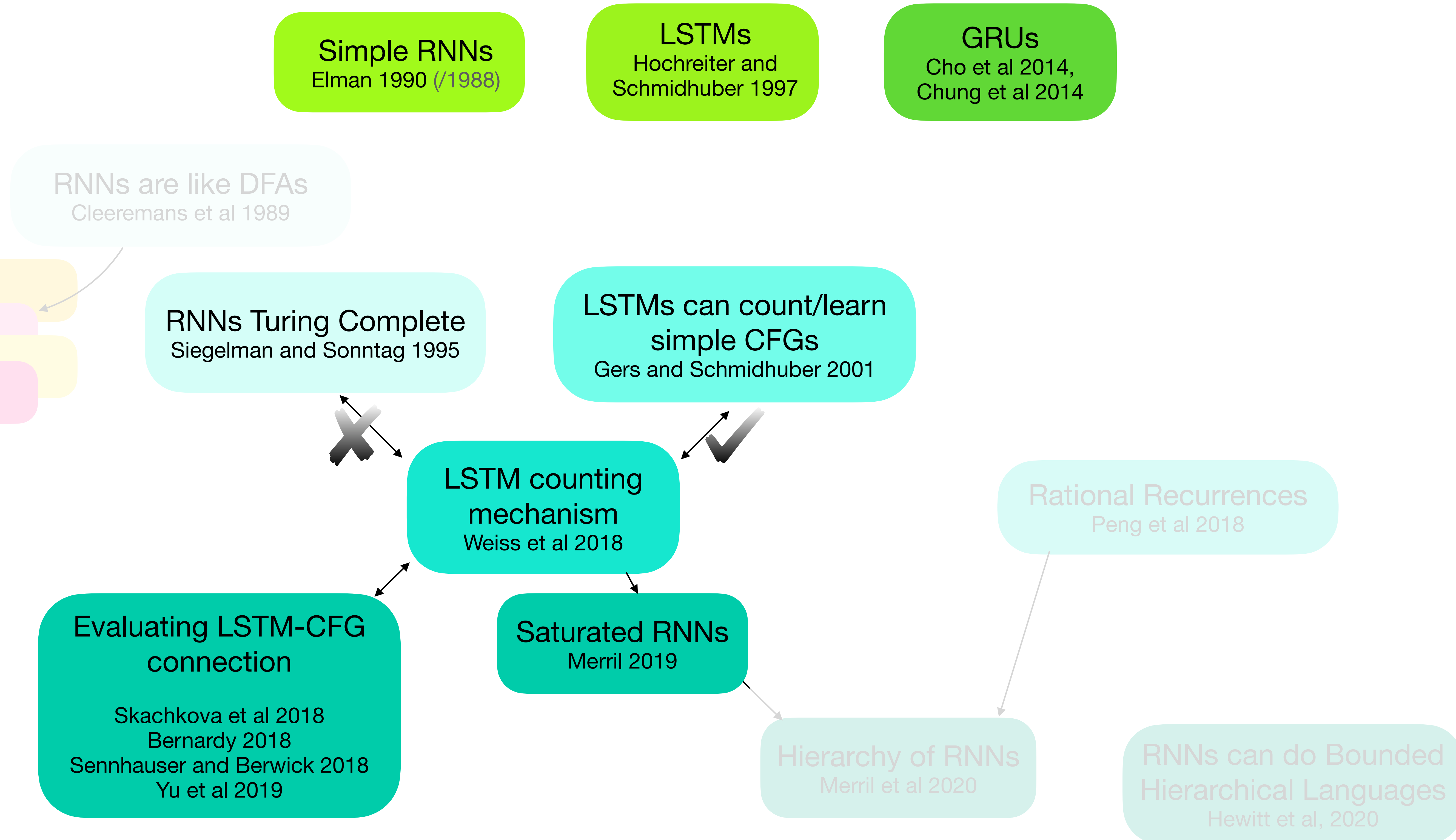
RNNs: Expressive Power



RNNs: Expressive Power



RNNs: Expressive Power



Saturated RNNs

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$f_t \in (0,1)$$
$$i_t \in (0,1)$$
$$o_t \in (0,1)$$

Saturated RNNs

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$f_t \in (0,1)$$
$$i_t \in (0,1)$$
$$o_t \in (0,1)$$

$$\sigma : \mathbb{R} \to (0,1)$$
$$\tanh : \mathbb{R} \to (-1,1)$$

Saturated RNNs

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$f_t \approx 1$$
$$i_t \approx 0$$
$$o_t \in (0,1)$$

$$\sigma : \mathbb{R} \to (0,1)$$
$$\tanh : \mathbb{R} \to (-1,1)$$

Saturated RNNs

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$f_t \approx 1$$
$$i_t \approx 0$$
$$o_t \in (0,1)$$

?

$$\sigma : \mathbb{R} \rightarrow (0,1)$$
$$\tanh : \mathbb{R} \rightarrow (-1,1)$$

Saturated RNNs

Sequential Neural Networks as Automata - Merrill (2019)

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$f_t \approx 1$$
$$i_t \approx 0$$
$$o_t \in (0,1)$$

?

$$\sigma : \mathbb{R} \to (0,1)$$
$$\tanh : \mathbb{R} \to (-1,1)$$

Saturated RNNs

Sequential Neural Networks as Automata - Merrill (2019)

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f)$$
$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i)$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o)$$

$$f_t \approx 1$$
$$i_t \approx 0$$
$$o_t \in (0,1)$$

?

$$\sigma : \mathbb{R} \rightarrow (0,1)$$
$$\tanh : \mathbb{R} \rightarrow (-1,1)$$

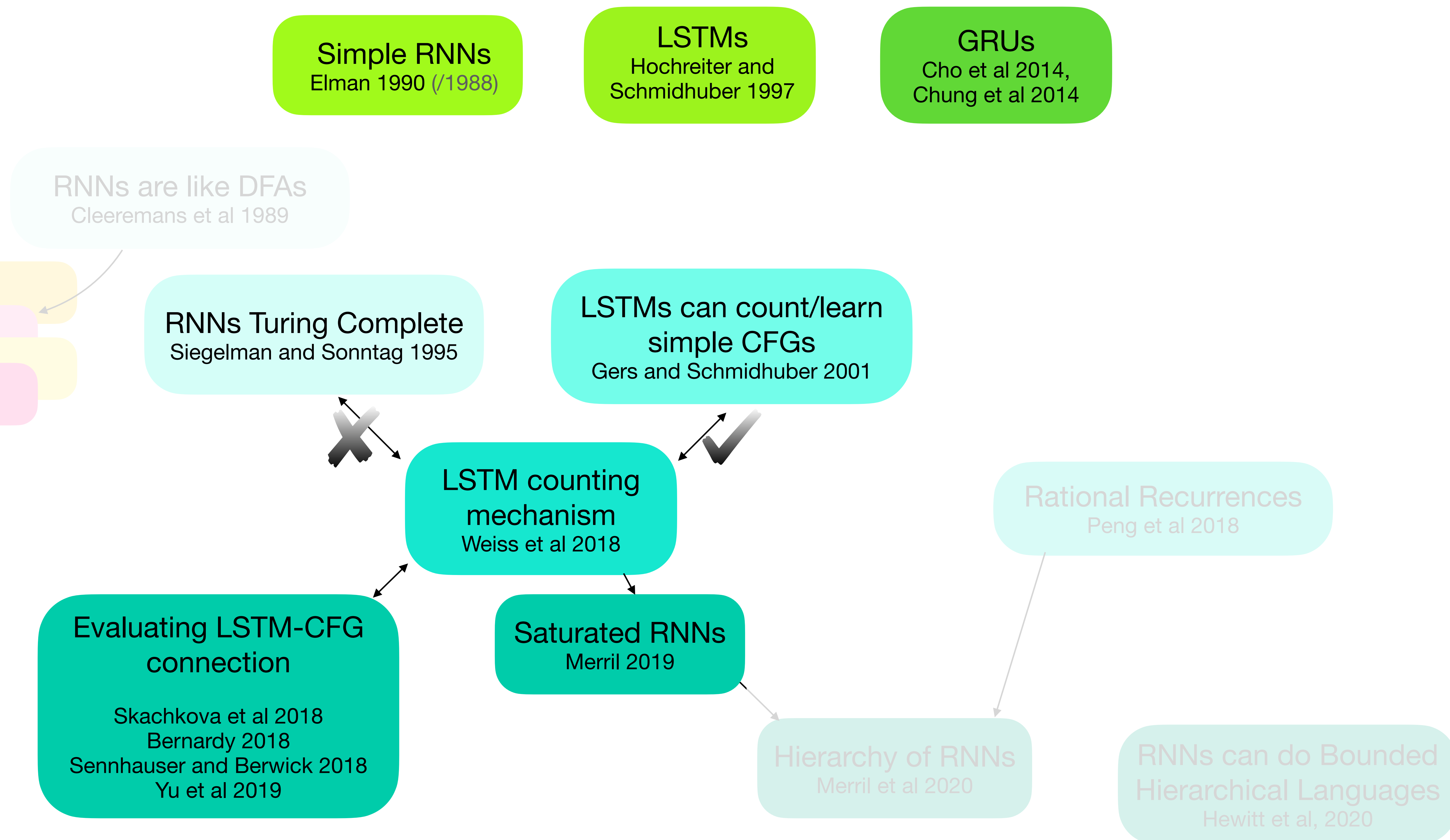
$$\theta$$

RNN is a parameterised function, $R(w : \theta)$

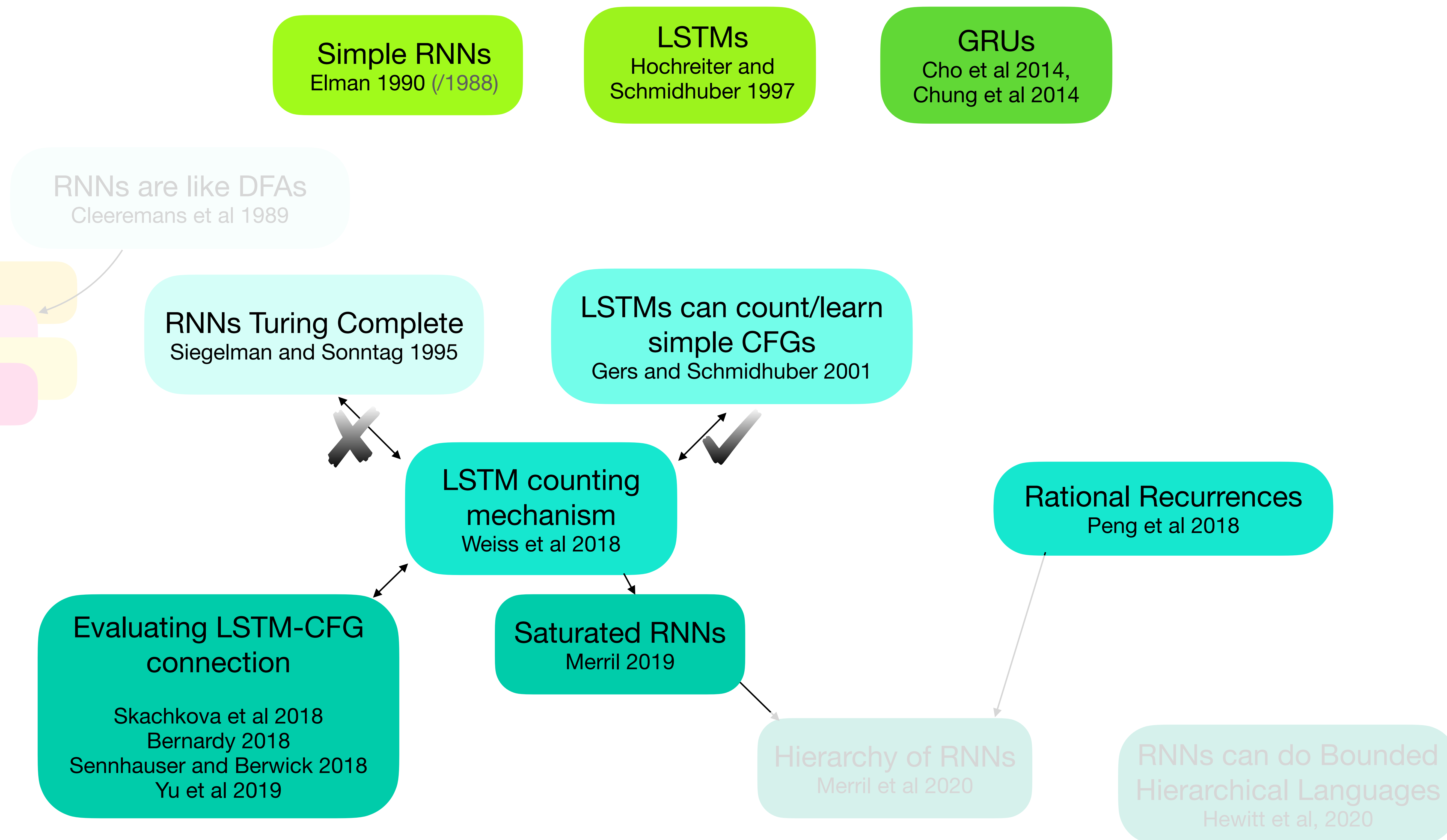
As θ “increases”, inputs to activations increase, saturating them

Saturated RNN: $sat\text{-}R(w : \theta) = \lim_{N \rightarrow \infty} R(w : N\theta)$

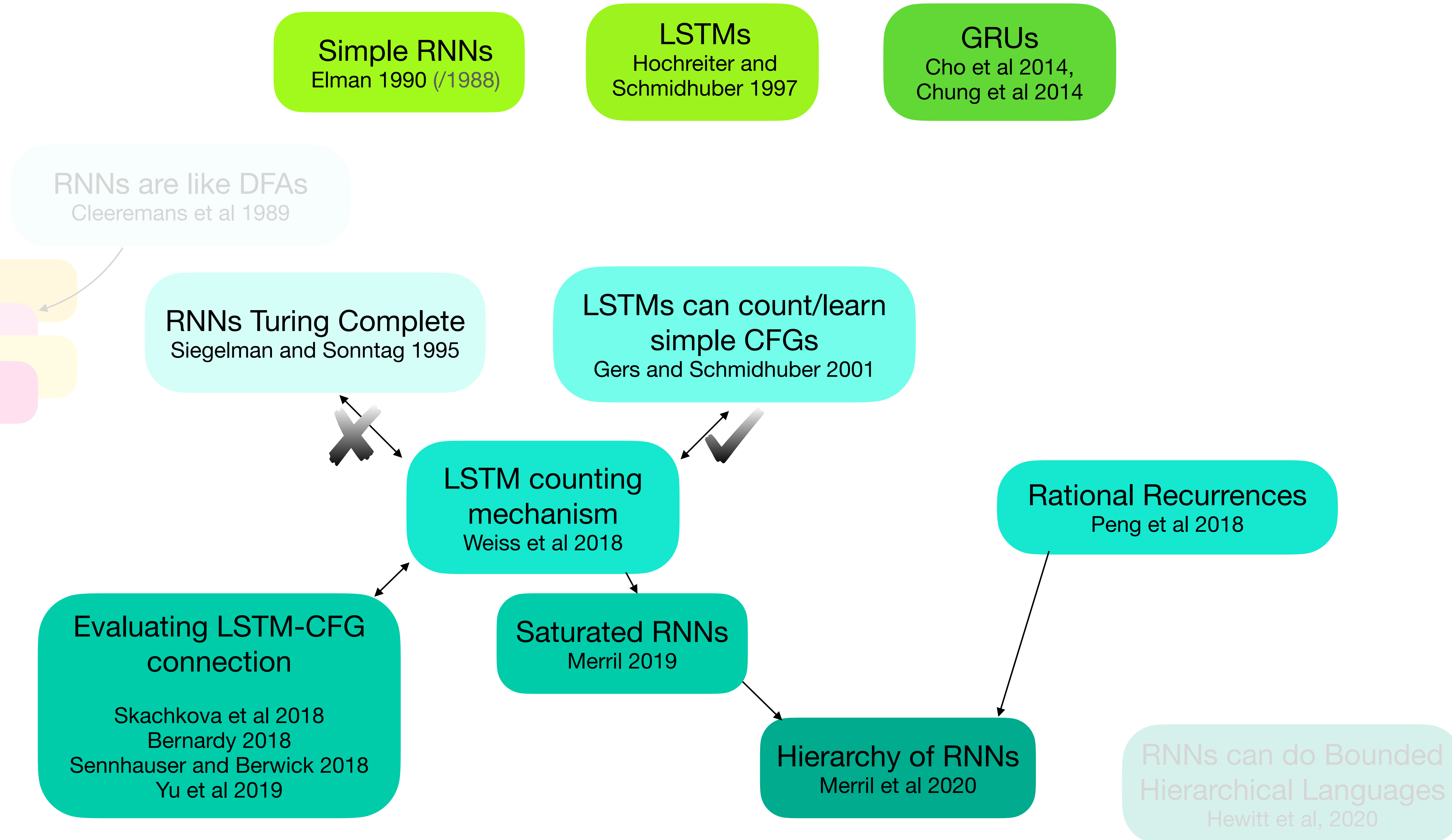
RNNs: Expressive Power



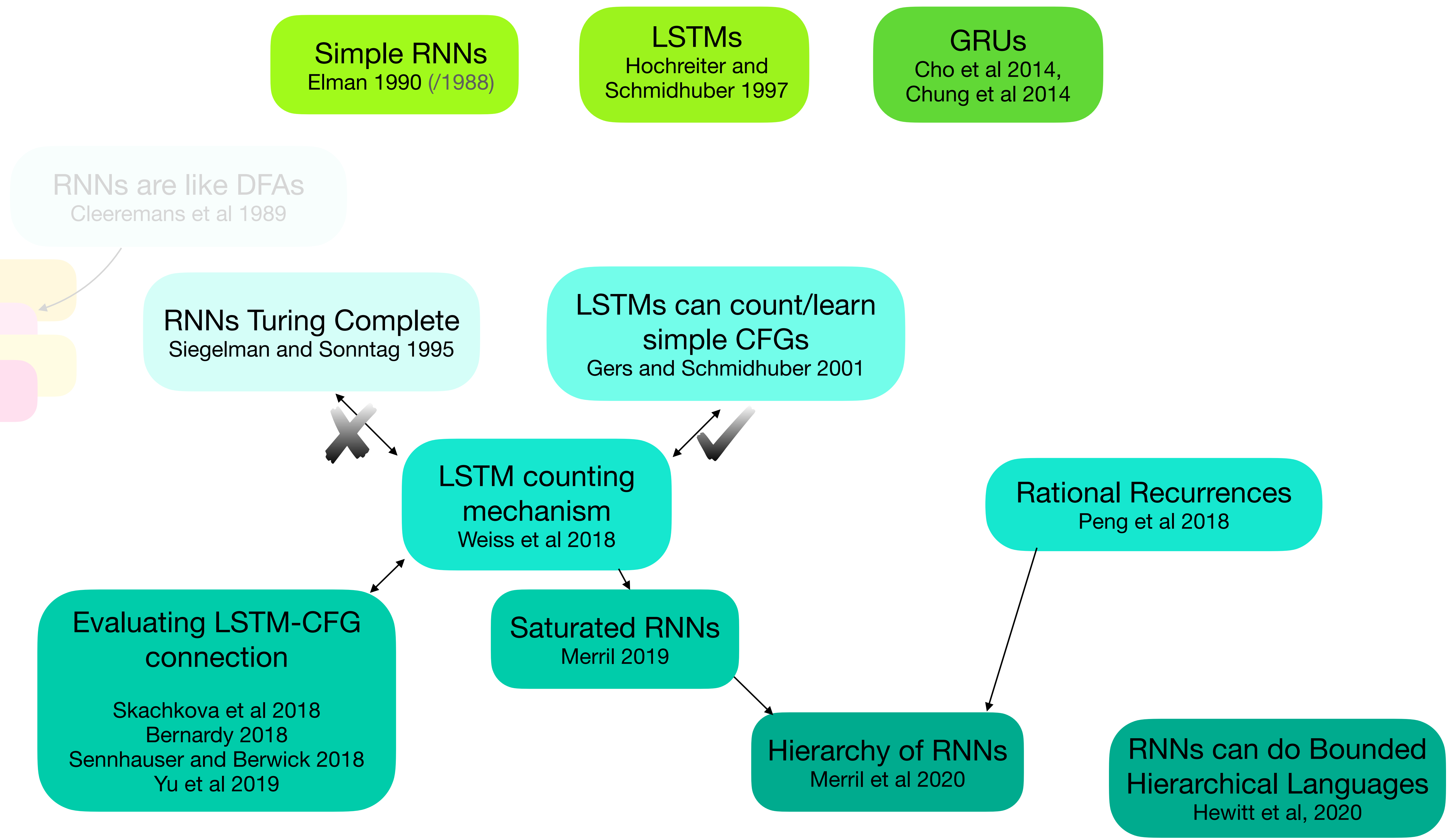
RNNs: Expressive Power



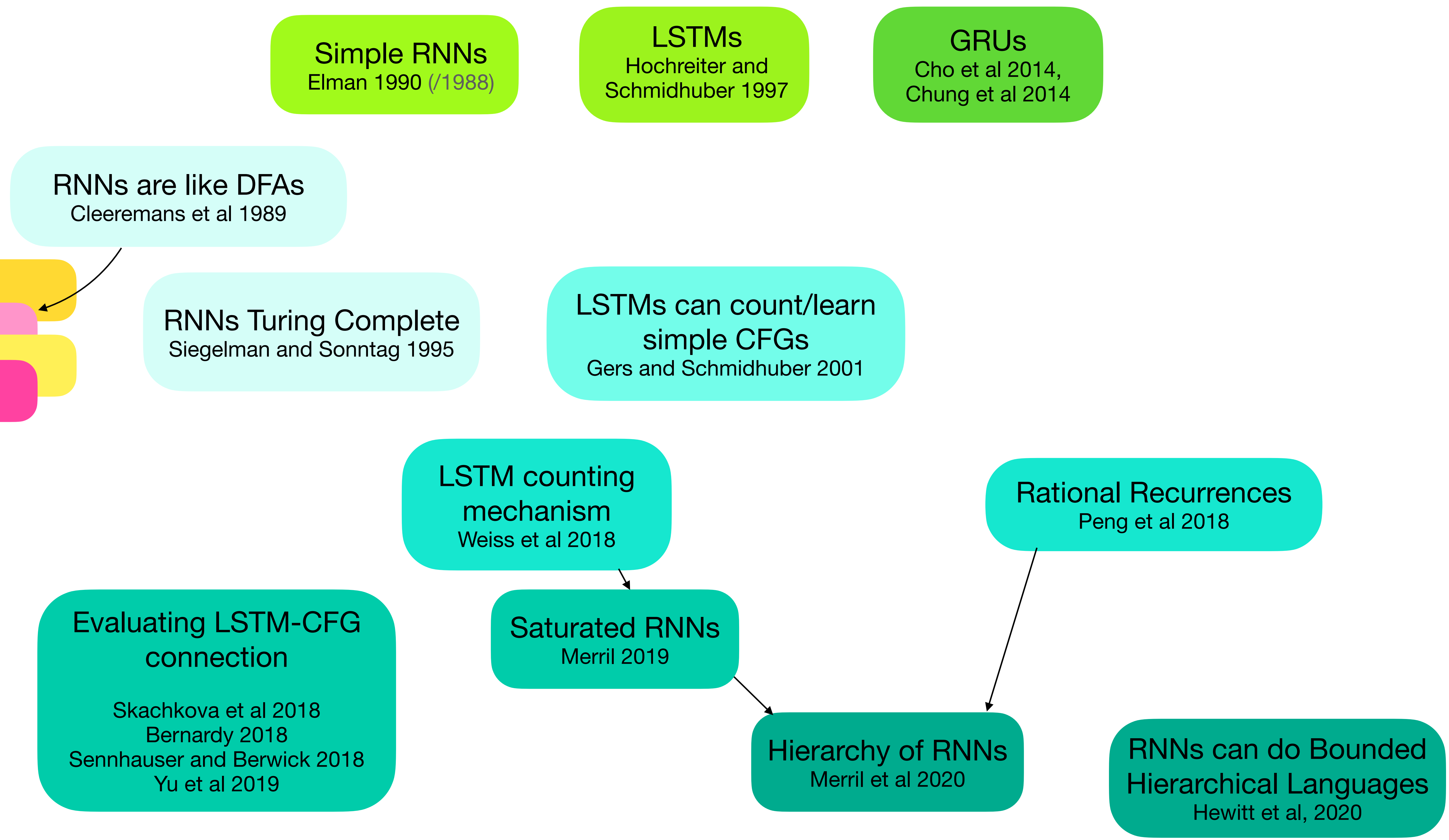
RNNs: Expressive Power



RNNs: Expressive Power



RNNs: Expressive Power



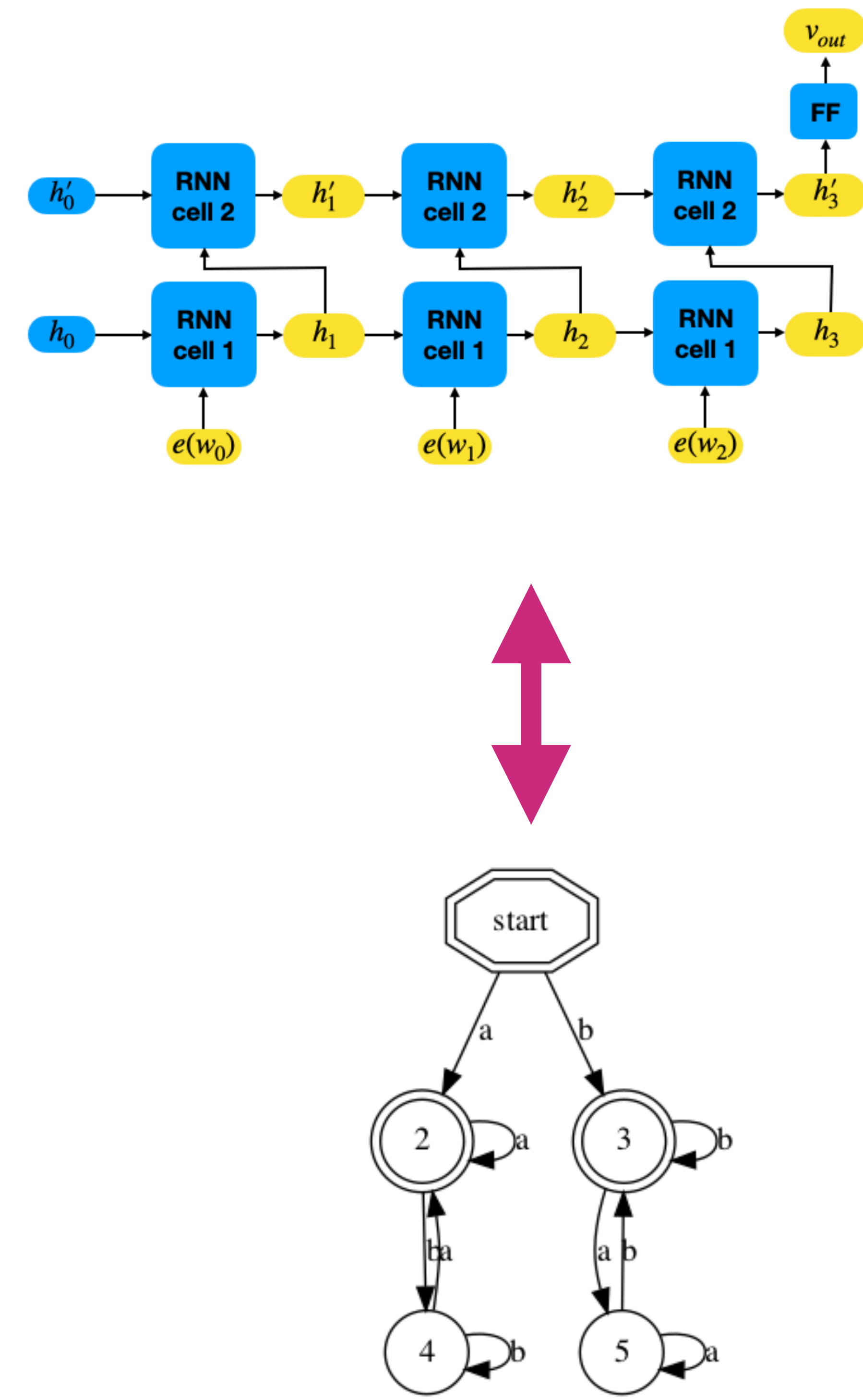
Overview

Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
    - DFAs
    - WFAs
    - More
- Analysis

Transformers

- Introduction
- A formal abstraction



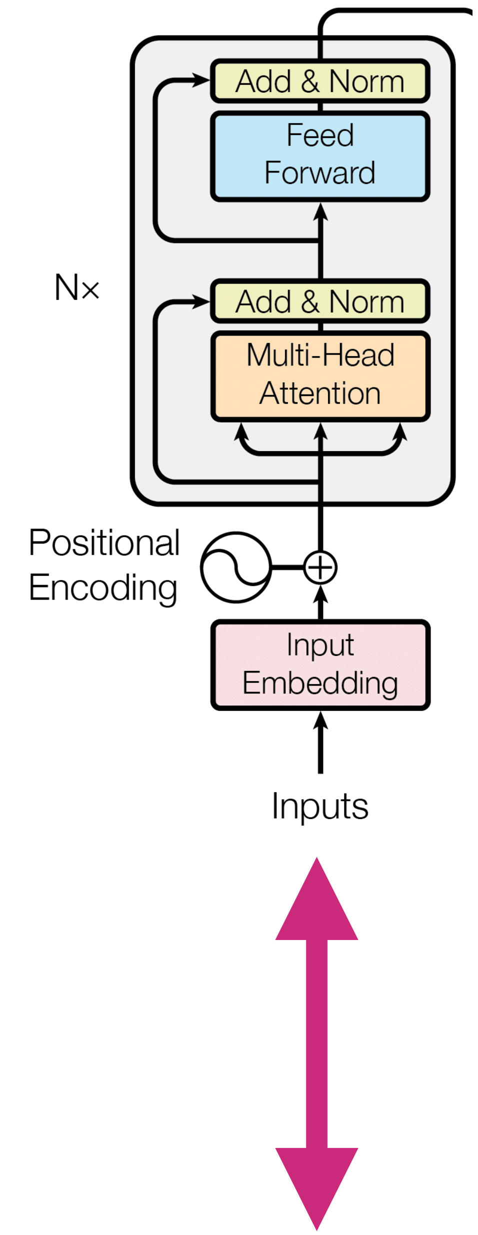
Overview

Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformers

- Introduction
- A formal abstraction



Code!?

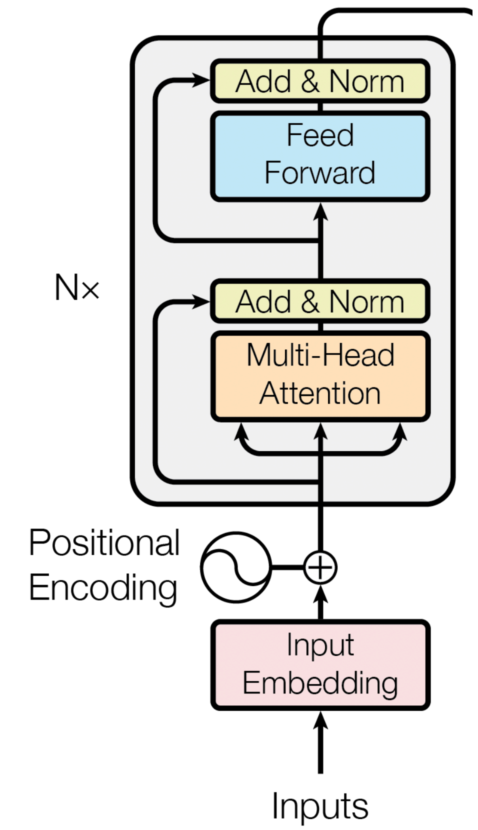
Overview

Recurrent Neural Networks (RNNs)

- Introduction
- RNN-Automata relation
- Extraction
  - DFAs
  - WFAs
  - More
- Analysis

Transformer Encoders

- Introduction
- A formal abstraction



Didn't make it! :(

But my website has links to talks on "Thinking Like Transformers", the work I wanted to introduce here:

https://sgailw.cswp.cs.technion.ac.il/publications/

The 1 hour talk includes an introduction on transformers, while the 5 minute talk assumes familiarity.