Weighted Automata Extraction from Recurrent Neural Networks via Regression

(回帰による再帰型ニューラルネットワークからの
重み付きオートマトンの抽出)

Takamasa Okudono　Masaki Waga
Taro Sekiyama　Ichiro Hasuo

{tokudono,mwaga,hasuo,sekiyama}@nii.ac.jp
NII & SOKENDAI

To appear at AAAI 2020

Preprint: https://arxiv.org/abs/1904.02931

Recurrent neural networks (RNNs)

Neural networks expandable according to input sequences with variable lengths

😄 Great success in learning sequential data, especially thanks to LSTMs/GRUs!

Recurrent neural networks (RNNs)

Neural networks expandable according to input sequences with variable lengths

😄 Great success in learning sequential data, especially thanks to LSTMs/GRUs!

🤨 But there are drawbacks such that:

- Hard to interpret what they learnt
- Hard to verify their behavior
- Computationally costly (even for inference)

Surrogate of RNNs

Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Surrogate of RNNs

Apparent state transition

Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Surrogate of RNNs

Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Surrogate of RNNs

Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Less operations

Surrogate of RNNs

Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Less operations

Extracting
FAs
from
discrete-input RNNs

Surrogate of RNNs

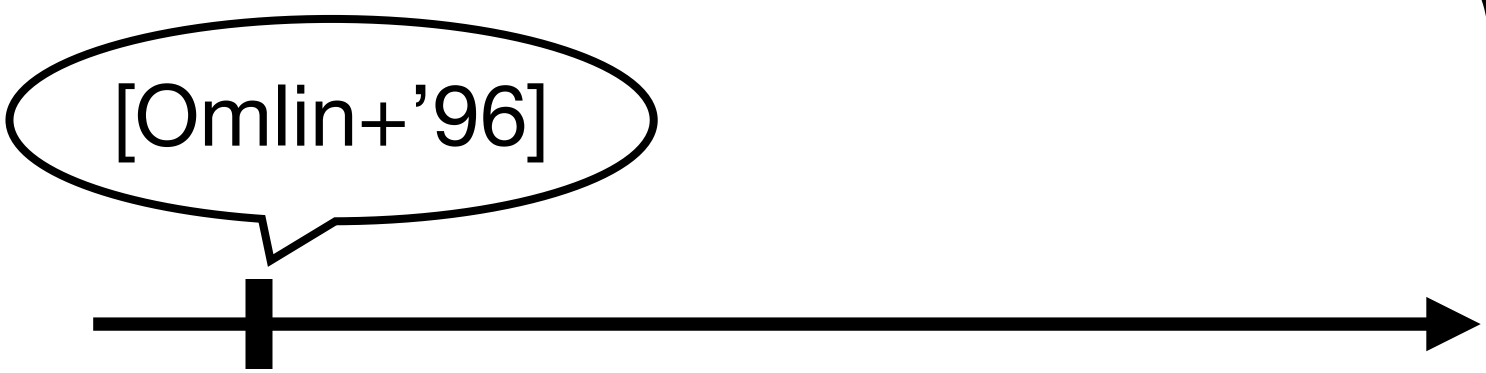
Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Less operations

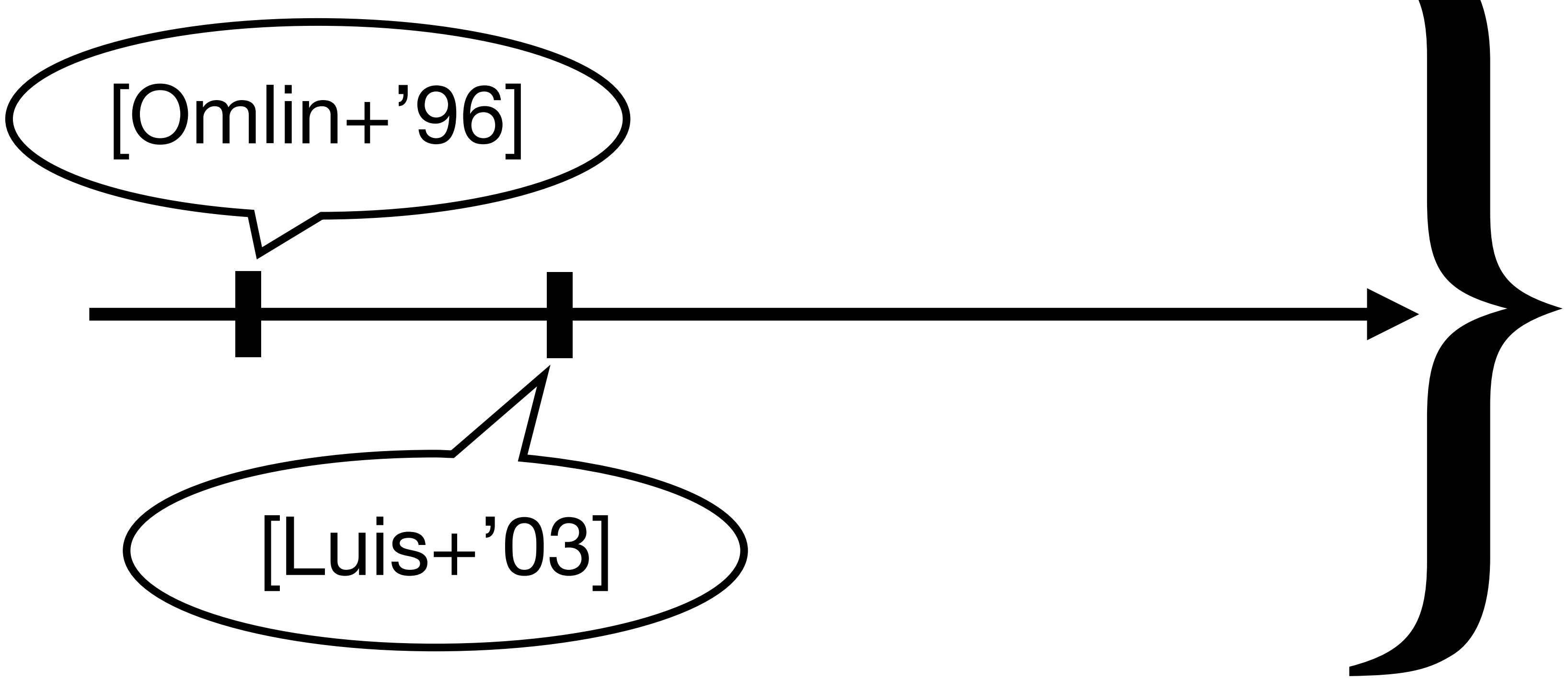


Extracting
FAs
from
discrete-input RNNs

Surrogate of RNNs

Finite automata (FAs)

- More interpretable — **Apparent state transition**
- More verifiable — **Many analysis algorithms**
- More efficient — **Less operations**



Extracting FAs from discrete-input RNNs

Surrogate of RNNs

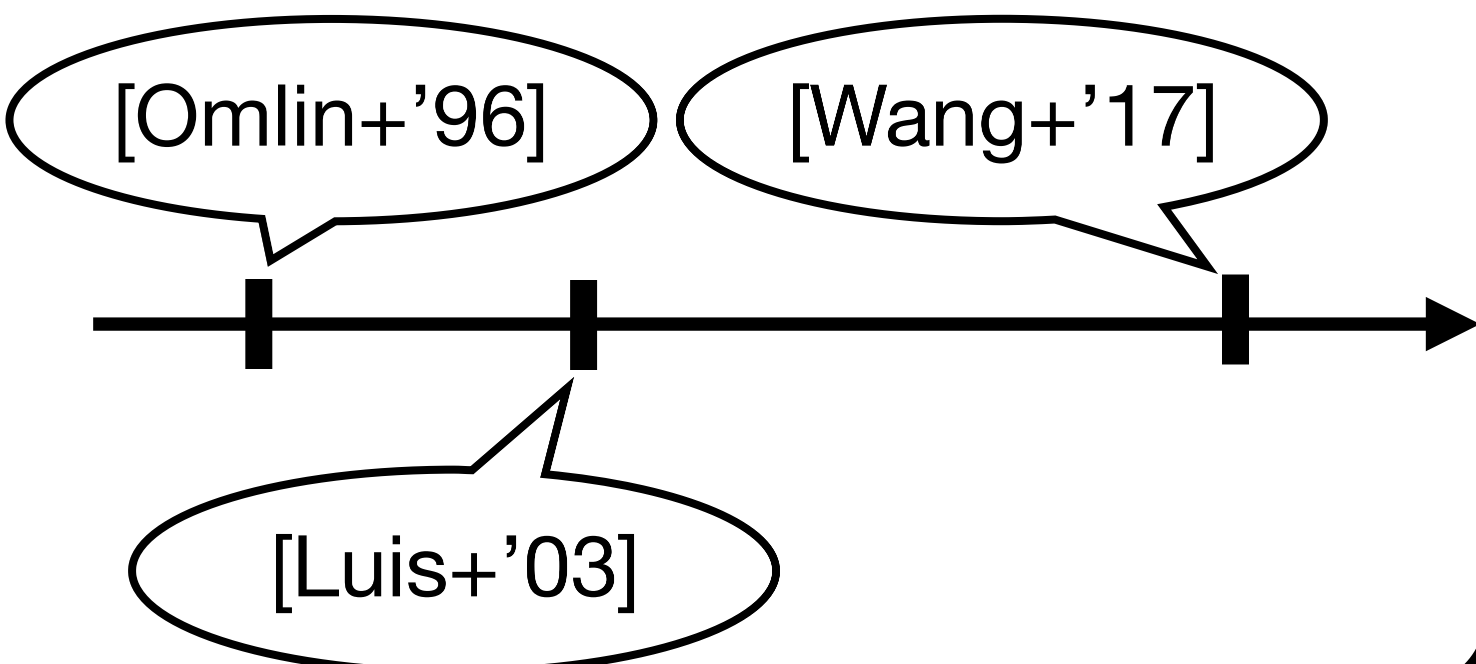
Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Less operations



Extracting FAs from discrete-input RNNs

Surrogate of RNNs

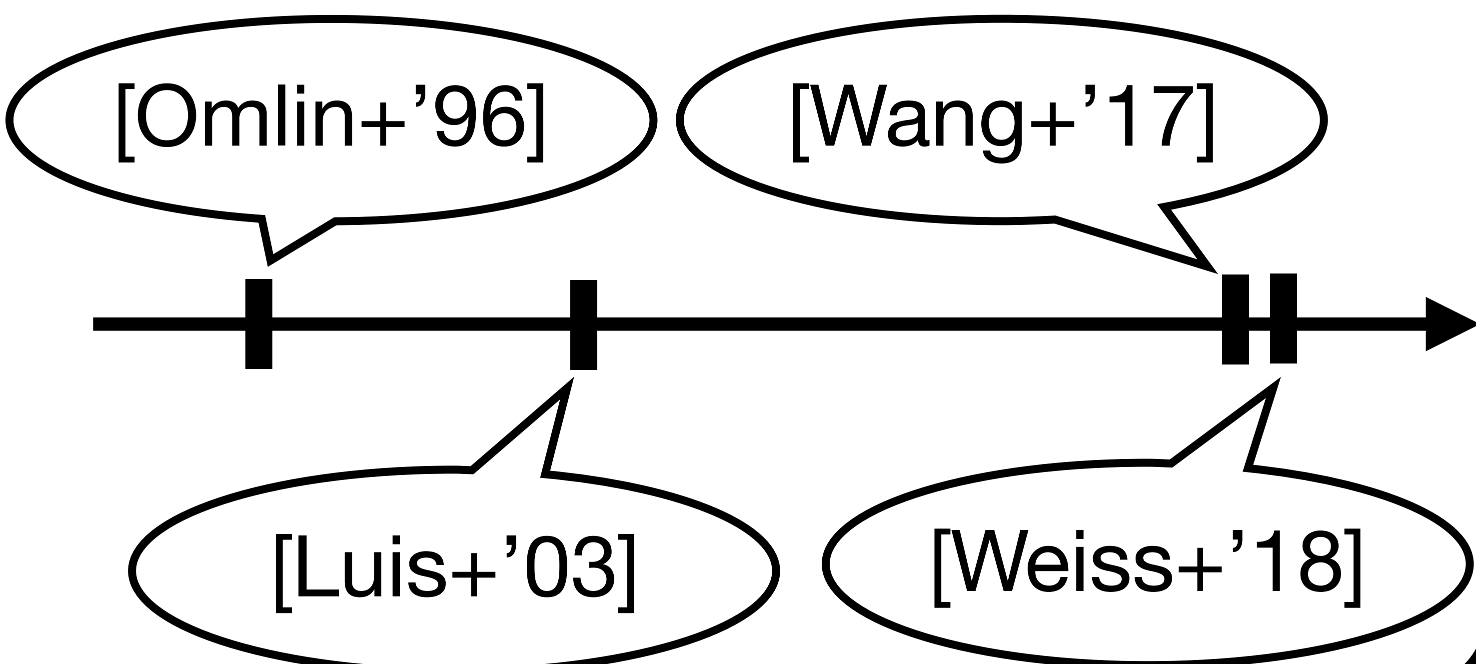
Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Less operations



Extracting FAs from discrete-input RNNs

Surrogate of RNNs

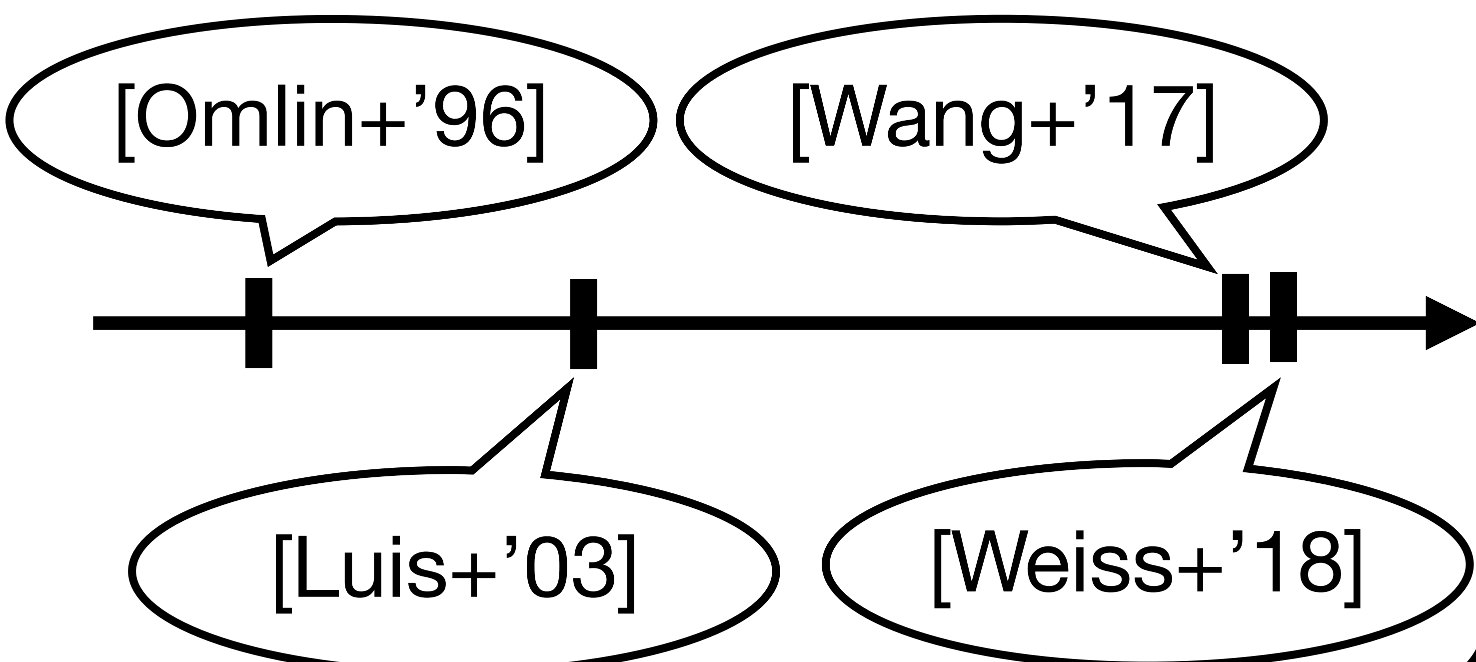
Finite automata (FAs)

- More interpretable
- More verifiable
- More efficient

Apparent state transition

Many analysis algorithms

Less operations



Extracting ***deterministic*** FAs from discrete-input, ***Boolean-output*** RNNs

Our work

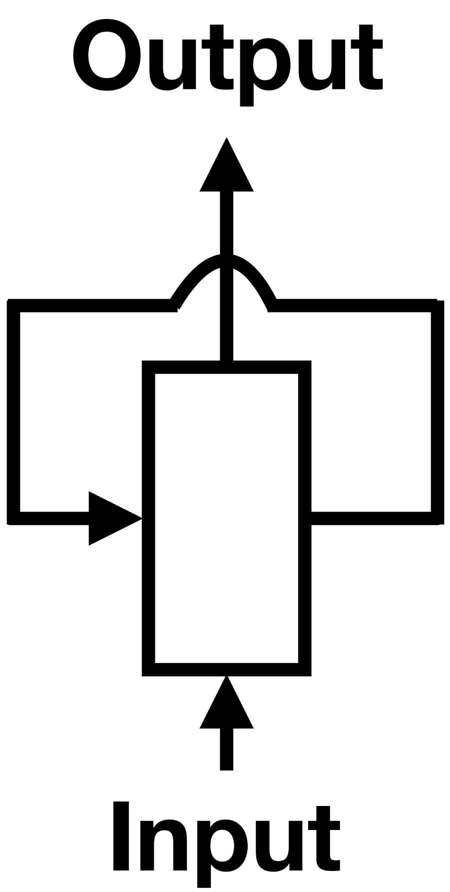
Extracting ***weighted*** **FAs *(WFAs)***
from **discrete-input, *real*-output RNNs**

- Based on a weighted extension of L^* [Angluin'87, Balle+'15]
  - **Technical contribution**:
    Answering equivalence queries b/w WFAs and RNNs by regression on their state spaces
- Experiments to answer:
  - How close the extracted WFAs are to the given RNNs
  - How applicable our method is to expressive RNNs
  - How efficient the extracted WFAs are

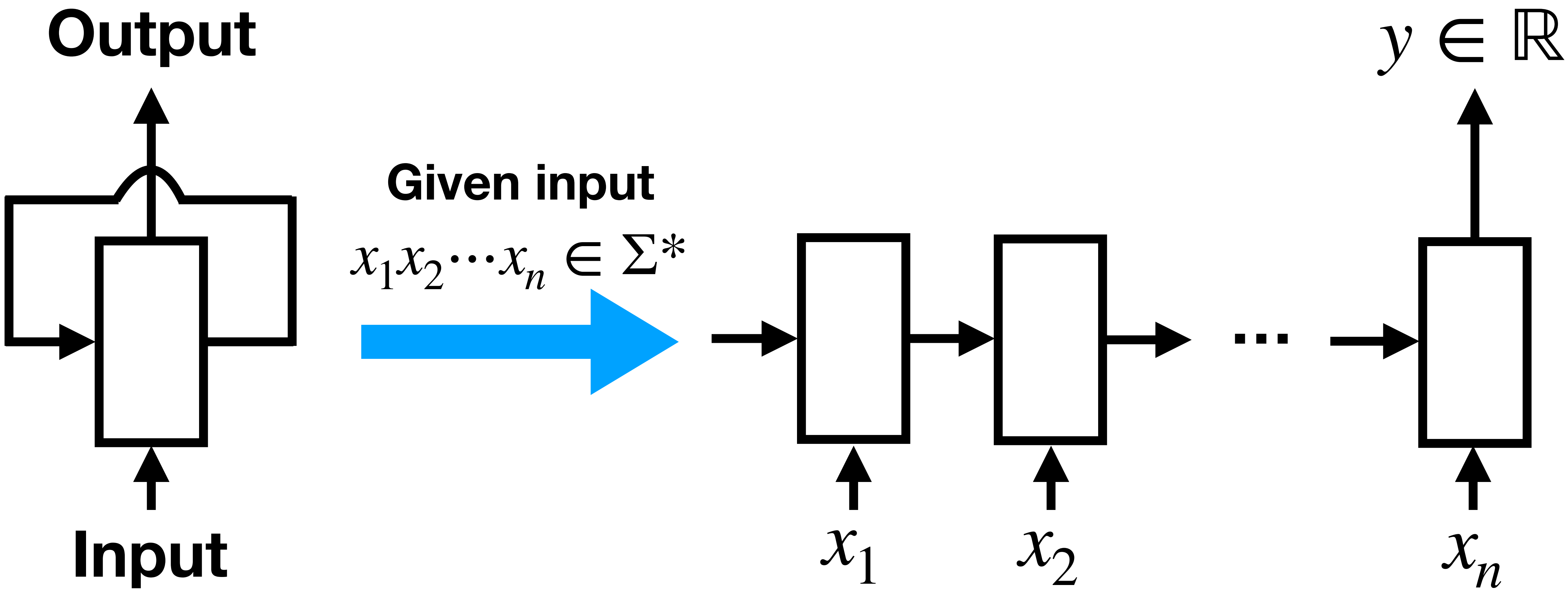
Outline

1. Introduction
2. Preliminaries
   A. Abstract definitions of RNNs and WFAs
   B. Weighted extension of Angluin's $L*$
3. Our work: extraction of WFAs from RNNs
4. Experiments

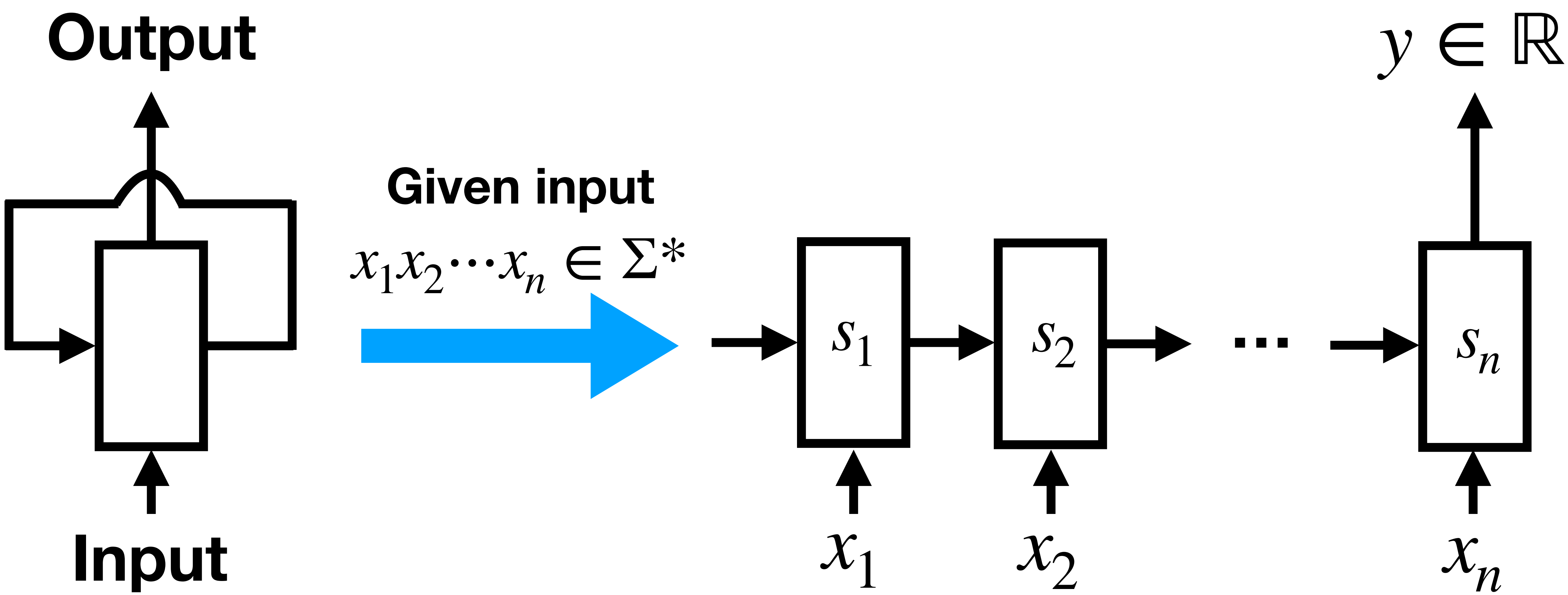
RNNs



RNNs

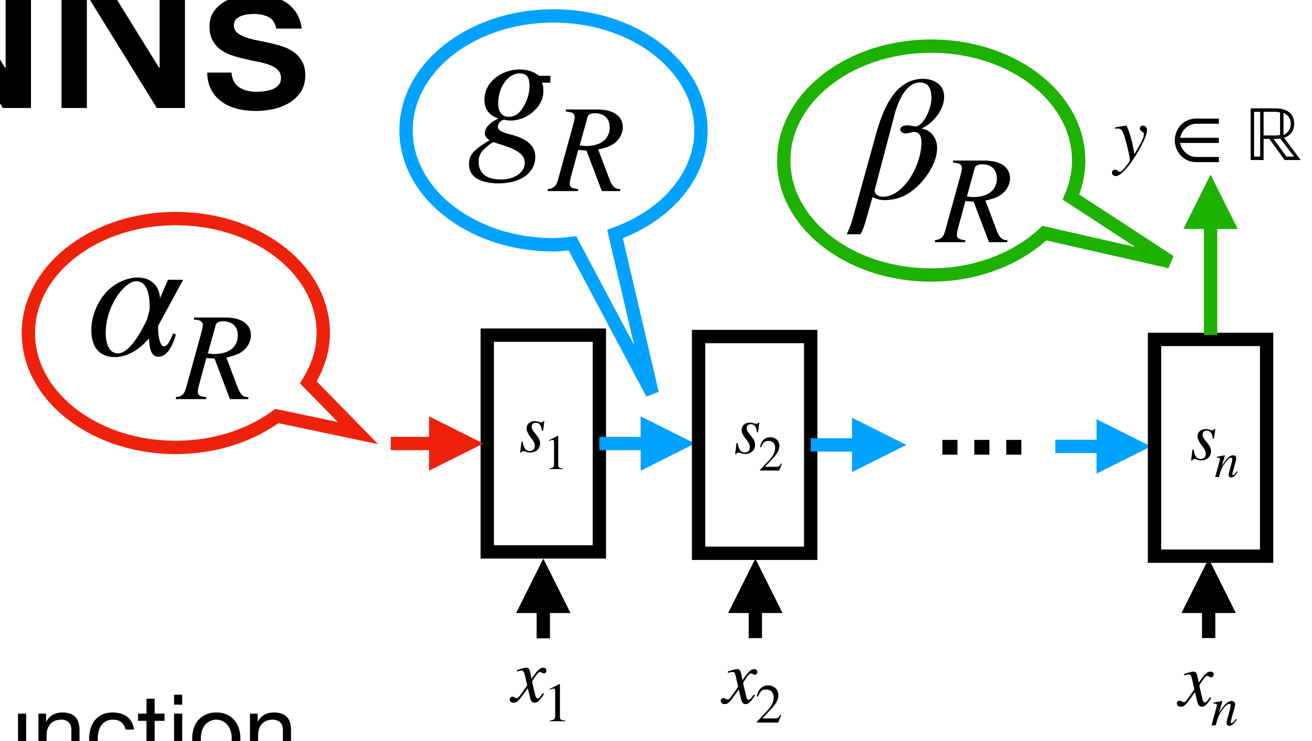

Output
Input
Given input
$x_1 x_2 \cdots x_n \in \Sigma^*$
$y \in \mathbb{R}$
x_1
x_2
x_n

RNNs



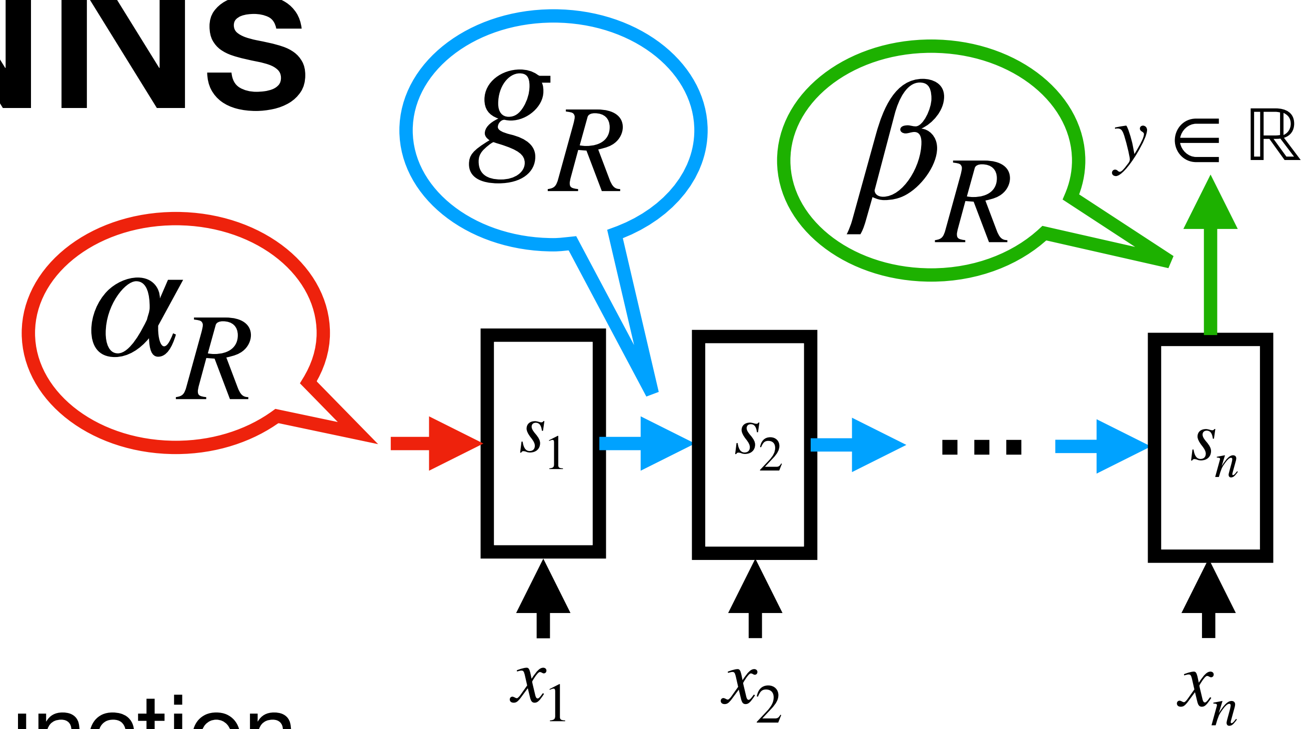
- Can be viewed as state-passing machines
  - States $s_1, s_2, \cdots, s_n \in \mathbb{R}^d$

RNNs



- RNN R comprises
  - $\alpha_R \in \mathbb{R}^d$: initial state
  - $\beta_R \in \mathbb{R}^d \to \mathbb{R}$: final function
  - $g_R \in \mathbb{R}^d \times \Sigma \to \mathbb{R}^d$: state transition function

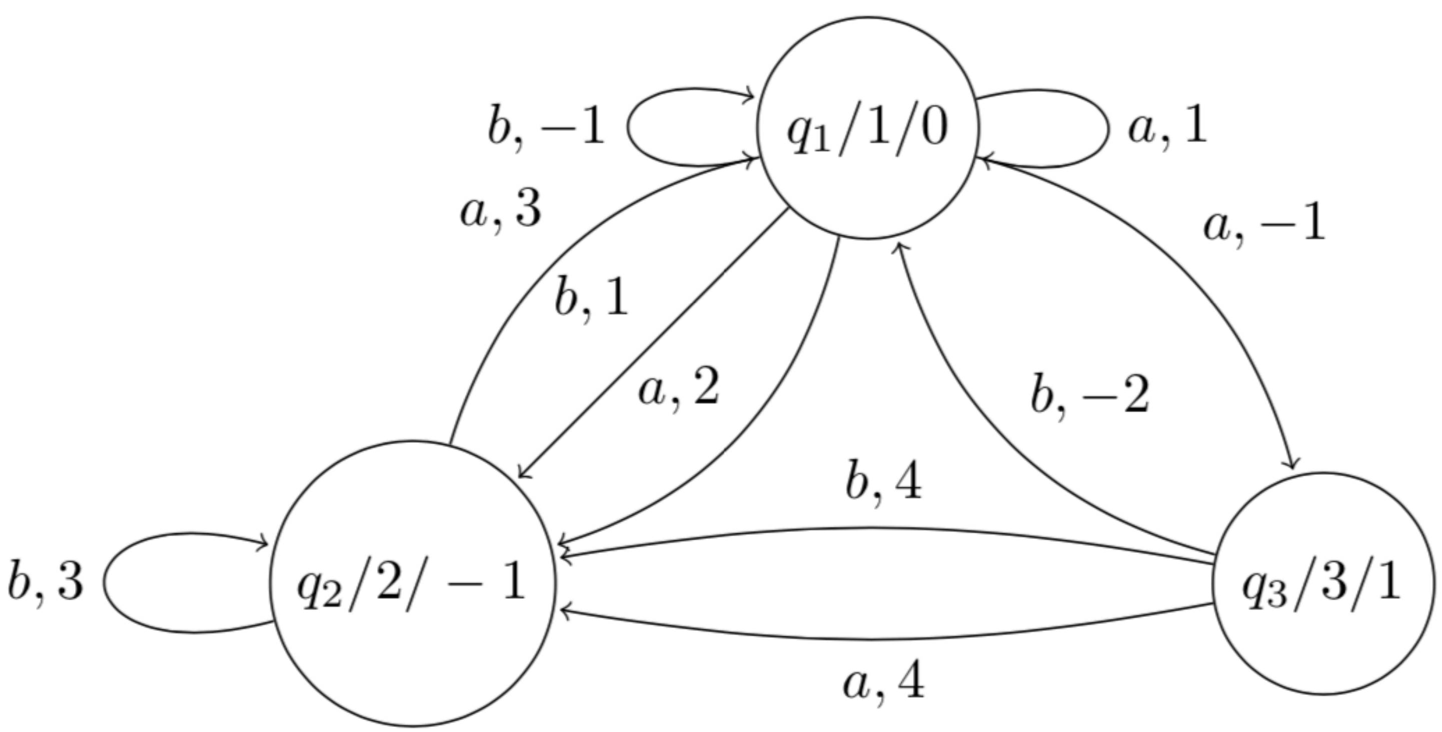
RNNs



- RNN R comprises
  - $\alpha_R \in \mathbb{R}^d$: initial state
  - $\beta_R \in \mathbb{R}^d \to \mathbb{R}$: final function
  - $g_R \in \mathbb{R}^d \times \Sigma \to \mathbb{R}^d$: state transition function
- Derived functions
  - $\delta_R(x_1 x_2 \cdots x_n) = g_R(\cdots, g_R(g_R(\alpha_R, x_1), x_2), \cdots, x_n)$
    $= s_n \in \mathbb{R}^d$
  - $f_R(x_1 x_2 \cdots x_n) = \beta_R(\delta_R(x_1 x_2 \cdots x_n)) = y \in \mathbb{R}$

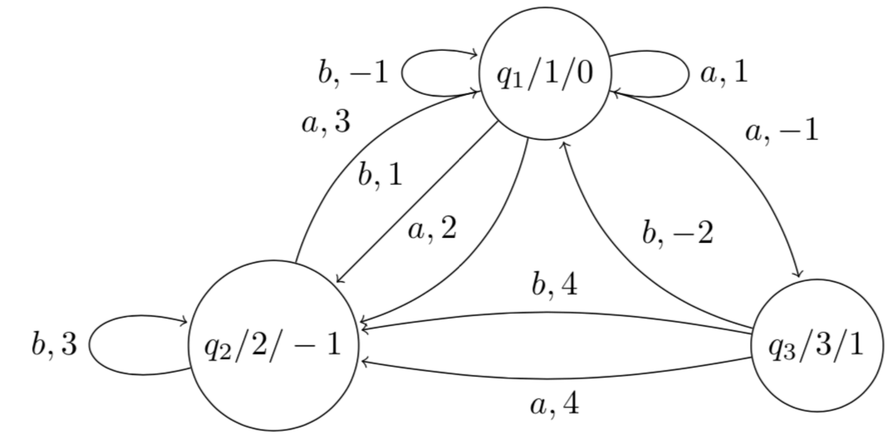
WFAs

- Finite automata s.t.
  - Each state has an initial and final weight
  - Each transition is weighted



- State label “q/m/n” means initial and final weights at q are m and n, respectively
- Transition label “a,n” means transition by letter "a" is weighted by n

WFAs



- WFA A with n states comprises
  - $\alpha_A \in \mathbb{R}^n$: initial vector (a collection of initial weights)
  - $\beta_A \in \mathbb{R}^n$: final vector (a collection of final weights)
  - $g_A \in \Sigma \to \mathbb{R}^{n \times n}$: transition function
    - $g_A(a)[i, j]$ is the weight of the transition by a from state i to j
- Derived functions
  - $\delta_A(x_1 x_2 \cdots x_n) = \alpha_A g_A(x_1) g_A(x_2) \cdots g_A(x_n) \in \mathbb{R}^n$
  - $f_A(x_1 x_2 \cdots x_n) = \delta_A(x_1 x_2 \cdots x_n) \cdot \beta_A \in \mathbb{R}$

Outline

1. Introduction
2. **Preliminaries**
    1. Abstract definitions of RNNs and WFAs
    2. **Weighted extension of Angluin's L^***
3. Our method: extraction of WFAs from RNNs
4. Experiments

$L^*(m, e)$ algorithm for WFAs [Balle+'15]

Learning a WFA from a black-box B by queries

- Input: answerers to two kinds of queries
  - $m \in \Sigma^* \to \mathbb{R}$ for ***membership queries***

    "What value does B output for input $w \in \Sigma^*$?"

  - $e \in \{\text{WFAs}\} \to \{\text{Eq}\} \uplus \Sigma^*$ for ***equivalence queries***

    "Is a WFA being constructed equivalent to B?"

    - Returns counterexample $w \in \Sigma^*$ if not
- Output
  - A (minimum) WFA

Property of $L^*(m, e)$

For a WFA A,

$$L^*(f_A, e_A) = A_0 \text{ s.t. } f_A = f_{A_0}$$

where $e_A(A') = \begin{cases} \text{Eq} & (\text{if } f_A = f_{A'}) \\ w & (\text{if } f_A(w) \neq f_{A'}(w)) \end{cases}$

Outline

1. Introduction
2. Preliminaries
3. **Our work: extraction of WFAs from RNNs**
4. Experiments

Our work

- Using L^* to obtain a WFA close to an RNN

Property of L^*

For a WFA A,

$$L^*(f_A, e_A) = A_0 \text{ s.t. } f_A = f_{A_0}$$

where $e_A(A') = \begin{cases} \text{Eq} & (\text{if } f_A = f_{A'}) \\ w & (\text{if } f_A(w) \neq f_{A'}(w)) \end{cases}$

Our work

- Using L^* to obtain a WFA close to an RNN

Our expectation

For an **RNN** $\mathbf{R}$,

$$L^*(f_{\mathbf{R}}, e_{\mathbf{R}}) = A_0 \text{ s.t. } f_{\mathbf{R}} \simeq f_{A_0}$$

where $e_{\mathbf{R}}(A') = \begin{cases} \text{Eq} & (\text{if } f_{\mathbf{R}} \simeq f_{A'}) \\ w & (\text{if } f_{\mathbf{R}}(w) \not\simeq f_{A'}(w)) \end{cases}$

Our work

- Using L^* to obtain a WFA close to an RNN

Our expectation

For an **RNN** $\mathbf{R}$,

$$L^*(f_{\mathbf{R}}, e_{\mathbf{R}}) = A_0 \text{ s.t. } f_{\mathbf{R}} \simeq f_{A_0}$$

where $$e_{\mathbf{R}}(A') = \begin{cases} \text{Eq} & (\text{if } f_{\mathbf{R}} \simeq f_{A'}) \\ w & (\text{if } f_{\mathbf{R}}(w) \not\simeq f_{A'}(w)) \end{cases}$$

- **Challenge**: how do we implement $e_R(A)$ that answers if $f_R \simeq f_A$ and returns an evidence if not

Strategy for $e_R(A)$

Goal: implement $e_R(A) = \begin{cases} \text{Eq} & (\text{if } f_R \simeq f_A) \\ w & (\text{if } f_R(w) \not\simeq f_A(w)) \end{cases}$

1. Trying to find counterexample w s.t. $f_R(w) \neq f_A(w)$

2(a). Returning w if found

2(b). Returning “Eq” if no counterexample is found even by sufficient search

Strategy for $e_R(A)$

Goal: implement $e_R(A) = \begin{cases} \text{Eq} & (\text{if } f_R \simeq f_A) \\ w & (\text{if } f_R(w) \not\simeq f_A(w)) \end{cases}$

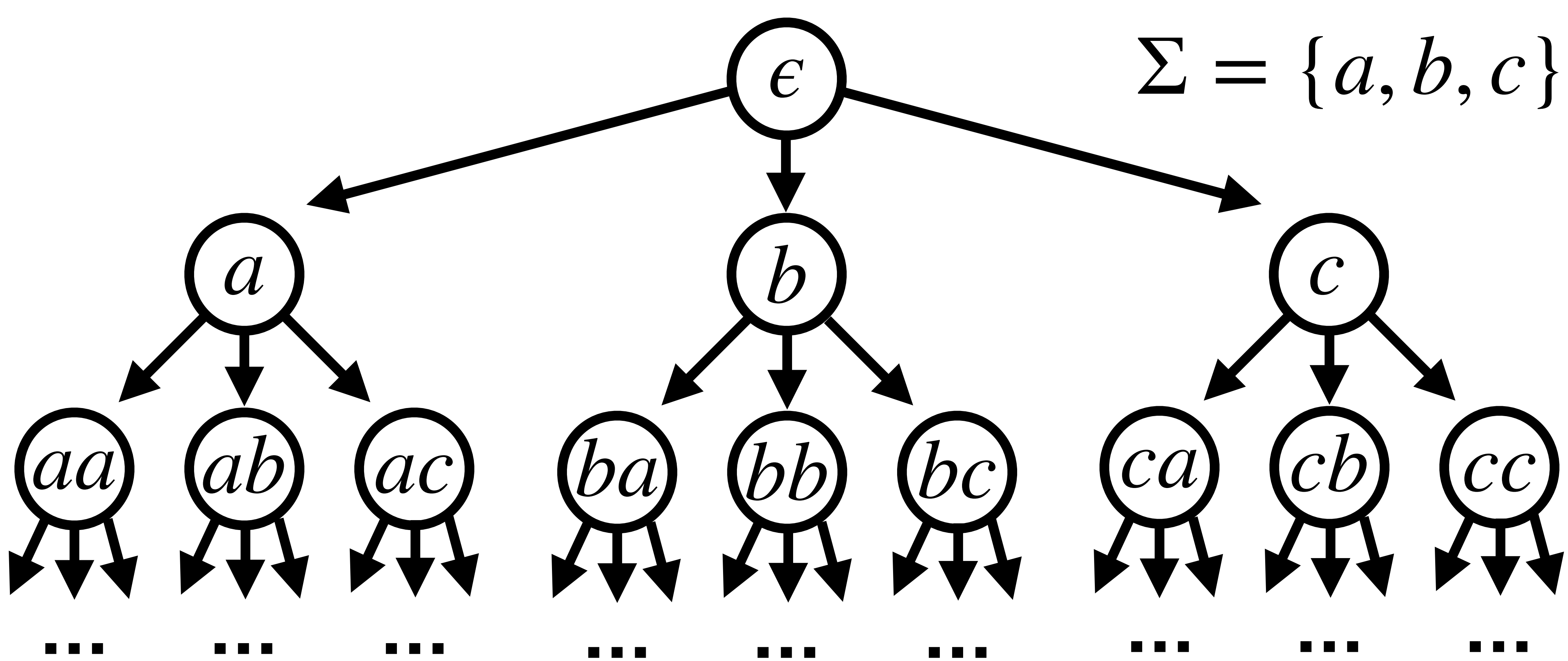
1. Trying to find counterexample w s.t. $f_R(w) \neq f_A(w)$

2(a). Returning w if found

2(b). Returning "Eq" if no counterexample is found even by sufficient search

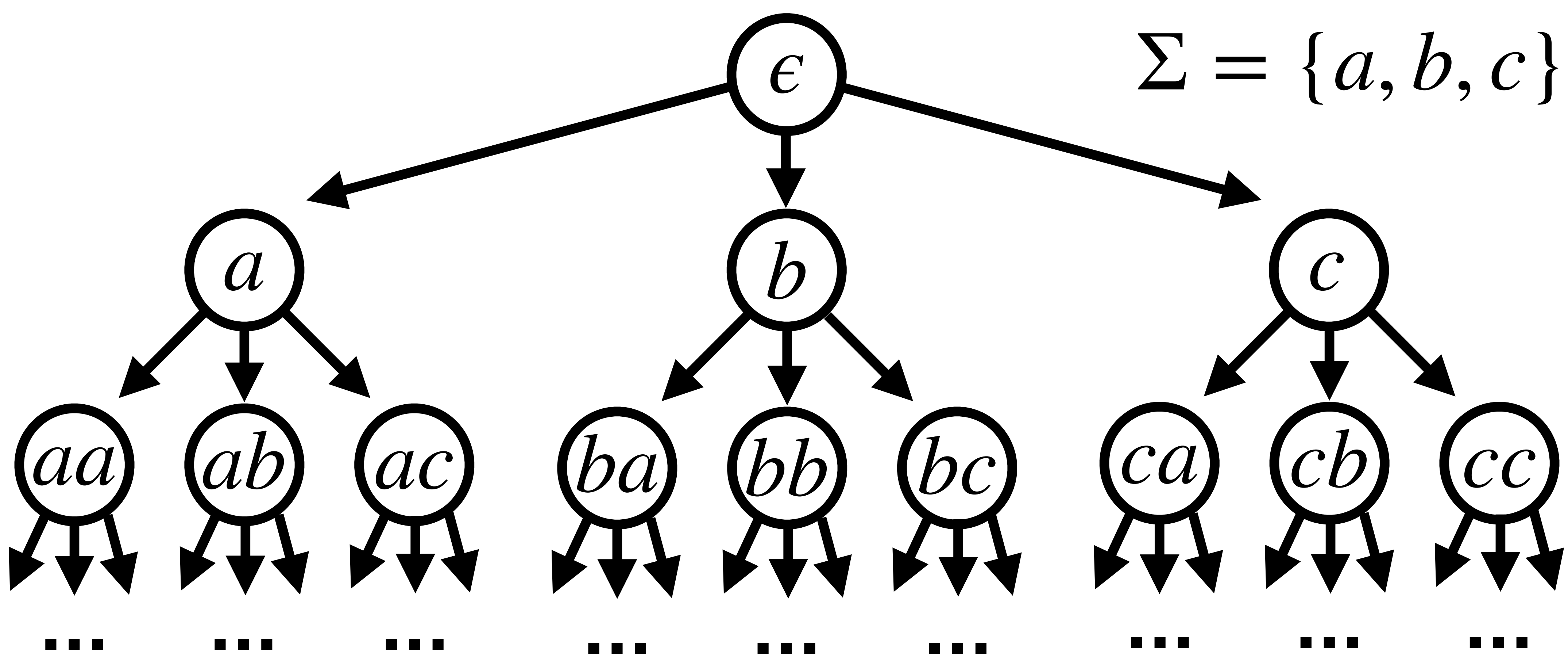
Tree traversal for finding counterexamples

- Finding a node w s.t. $f_R(w) \neq f_A(w)$



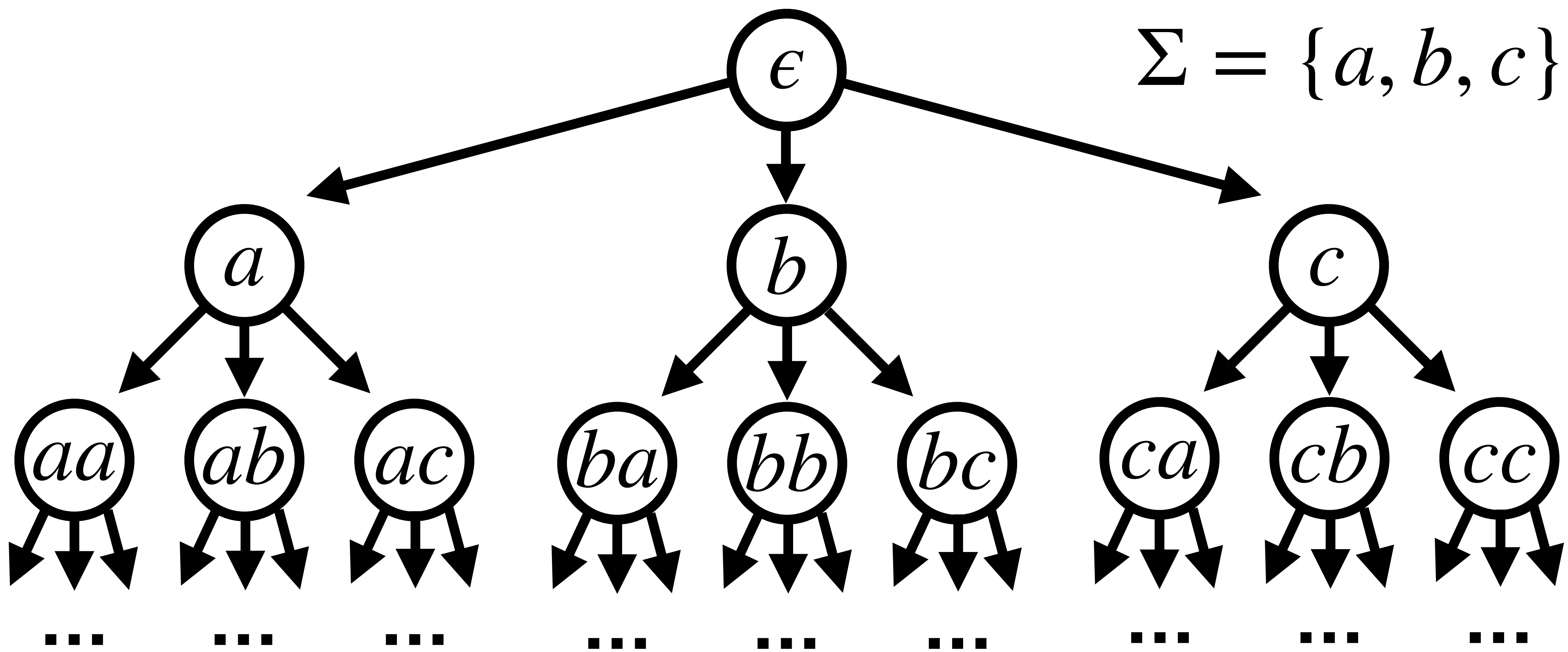
Tree traversal for finding counterexamples

- Finding a node w s.t. $f_R(w) \neq f_A(w)$
- **Question**: What order of traversal can find a counterexample effectively?



Our approach: guiding the traversal by
a relationship between WFA A and RNN R

- Finding a node w s.t. $f_R(w) \neq f_A(w)$
- **Question**: What order of traversal can find a counterexample effectively?



Relationship b/t RNN R and WFA A

Working hypothesis

$$f_R \simeq f_A$$

Relationship b/t RNN R and WFA A

$= \beta_R \circ \delta_R$

Working hypothesis

$f_R \simeq f_A$

… R and WFA A

$= \beta_R \circ \delta_R$

$= w \mapsto \delta_A(w) \cdot \beta_A$

Working hypothesis

$f_R \simeq f_A$

R and WFA A

$= \beta_R \circ \delta_R$

$= w \mapsto \delta_A(w) \cdot \beta_A$

Working hypothesis

$f_R \simeq f_A \iff$ Internal states obtained by δ_R and δ_A are related

… R and WFA A

$= \beta_R \circ \delta_R$

$= w \mapsto \delta_A(w) \cdot \beta_A$

Working hypothesis

$$
f_R \simeq f_A \quad \Leftrightarrow \quad
\begin{tikzcd}
 & \mathbb{R}^d \arrow[dd, dashed, no head, "?"] \\
\Sigma^* \arrow[ur, "\delta_R"] \arrow[dr, "\delta_A"'] & \\
 & \mathbb{R}^n
\end{tikzcd}
$$

… R and WFA A

$= \beta_R \circ \delta_R$

$= w \mapsto \delta_A(w) \cdot \beta_A$

Working hypothesis

$f_R \simeq f_A \quad \Leftrightarrow$

$$\begin{tikzcd} & \mathbb{R}^d \arrow[dd, "p"] \\ \Sigma^* \arrow[ur, "\delta_R"] \arrow[dr, "\delta_A"'] & \\ & \mathbb{R}^n \end{tikzcd}$$

for some $p \in \mathbb{R}^d \to \mathbb{R}^n$

R and WFA A

$= \beta_R \circ \delta_R$

$= w \mapsto \delta_A(w) \cdot \beta_A$

Working hypothesis

$$f_R \simeq f_A \quad \Leftrightarrow \quad \begin{tikzcd} & \mathbb{R}^d \arrow[dd, "p"] \\ \Sigma^* \arrow[ur, "\delta_R"] \arrow[dr, "\delta_A"'] & \\ & \mathbb{R}^n \end{tikzcd}$$

for some

$p \in \mathbb{R}^d \to \mathbb{R}^n$

- Utilizing p to find counterexamples effectively
- Approximating p by ***regression on state spaces***

Traversal with p

$$
\begin{tikzcd}
 & \mathbb{R}^d \arrow[dd, "p"] \\
\Sigma^* \arrow[ur, "\delta_R"] \arrow[dr, "\delta_A"'] & \\
 & \mathbb{R}^n
\end{tikzcd}
$$

- **Idea:**

$$f_R(w) \not\approx f_A(w) \Leftrightarrow \quad p(\delta_R(w)) \not\approx \delta_A(w)$$

Traversal with p

$$\begin{tikzcd} & \mathbb{R}^d \arrow[dd, "p"] \\ \Sigma^* \arrow[ur, "\delta_R"] \arrow[dr, "\delta_A"'] & \\ & \mathbb{R}^n \end{tikzcd}$$

- **Idea:**

$$f_R(w) \not\approx f_A(w) \Leftrightarrow \quad p(\delta_R(w)) \not\approx \delta_A(w)$$

$\Leftarrow$

$$D_w = \|p_0(\delta_R(w)) - \delta_A(w)\| \gg 0$$

if ***approximation*** p_0 is sufficiently close to p

Traversal with p

$$
\begin{tikzcd}
 & \mathbb{R}^d \arrow[dd, "p"] \\
\Sigma^* \arrow[ur, "\delta_R"] \arrow[dr, "\delta_A"'] & \\
 & \mathbb{R}^n
\end{tikzcd}
$$

- **Idea:**

$$f_R(w) \not\simeq f_A(w) \iff p(\delta_R(w)) \not\simeq \delta_A(w)$$

$\Leftarrow$

$D_w = \|p_0(\delta_R(w)) - \delta_A(w)\| \gg 0$

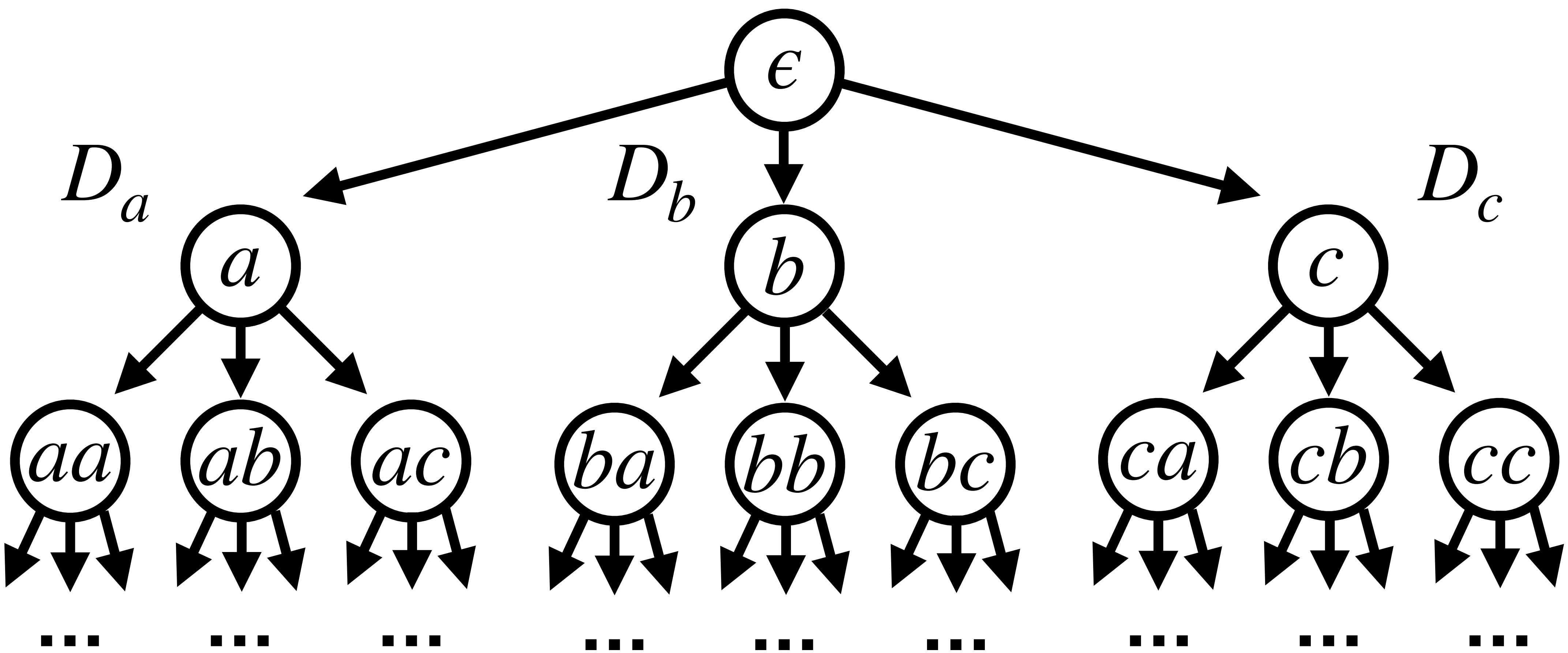
if ***approximation*** p_0 is sufficiently close to p

- **Traversal strategy:**

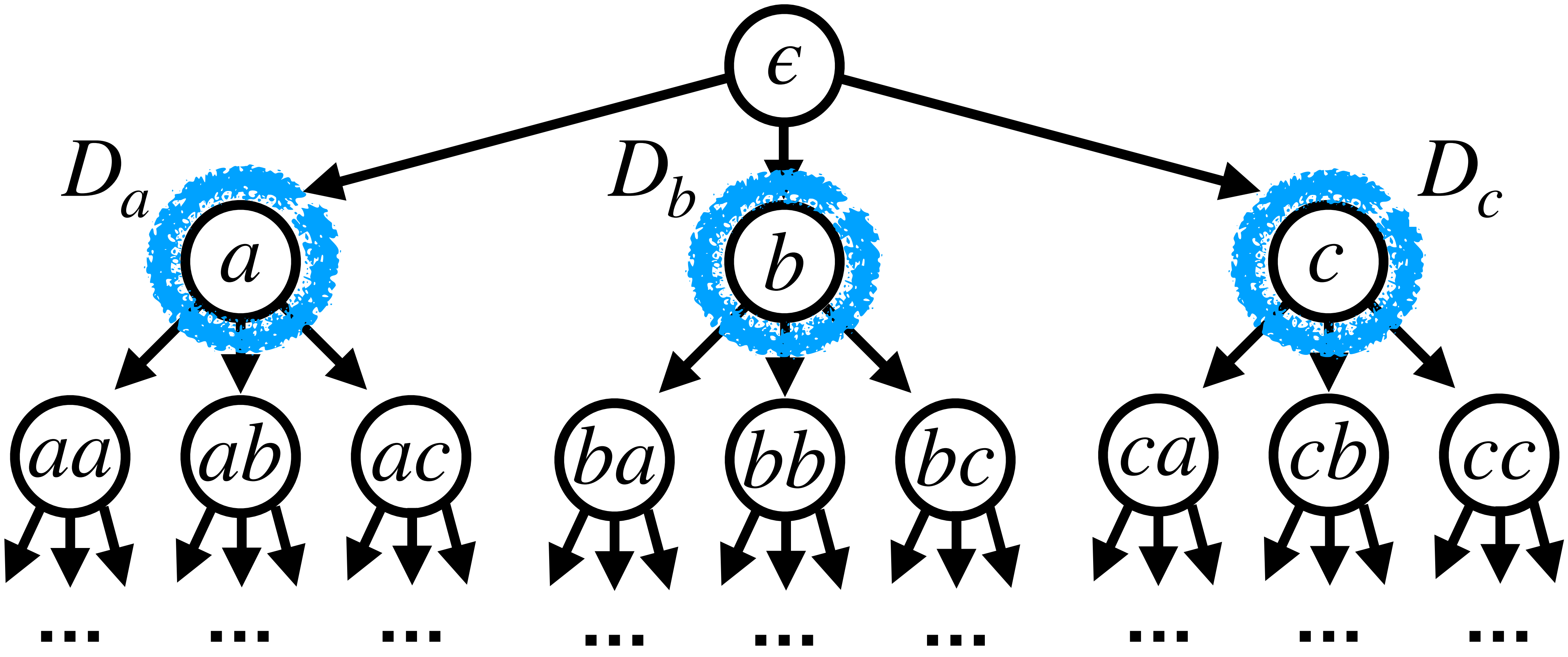
  nodes w with ***larger*** D_w have ***higher*** priorities

  - If D_w is large but $f_R(w) \simeq f_A(w)$, p_0 is refined on $(\delta_R(w), \delta_A(w))$

Running example

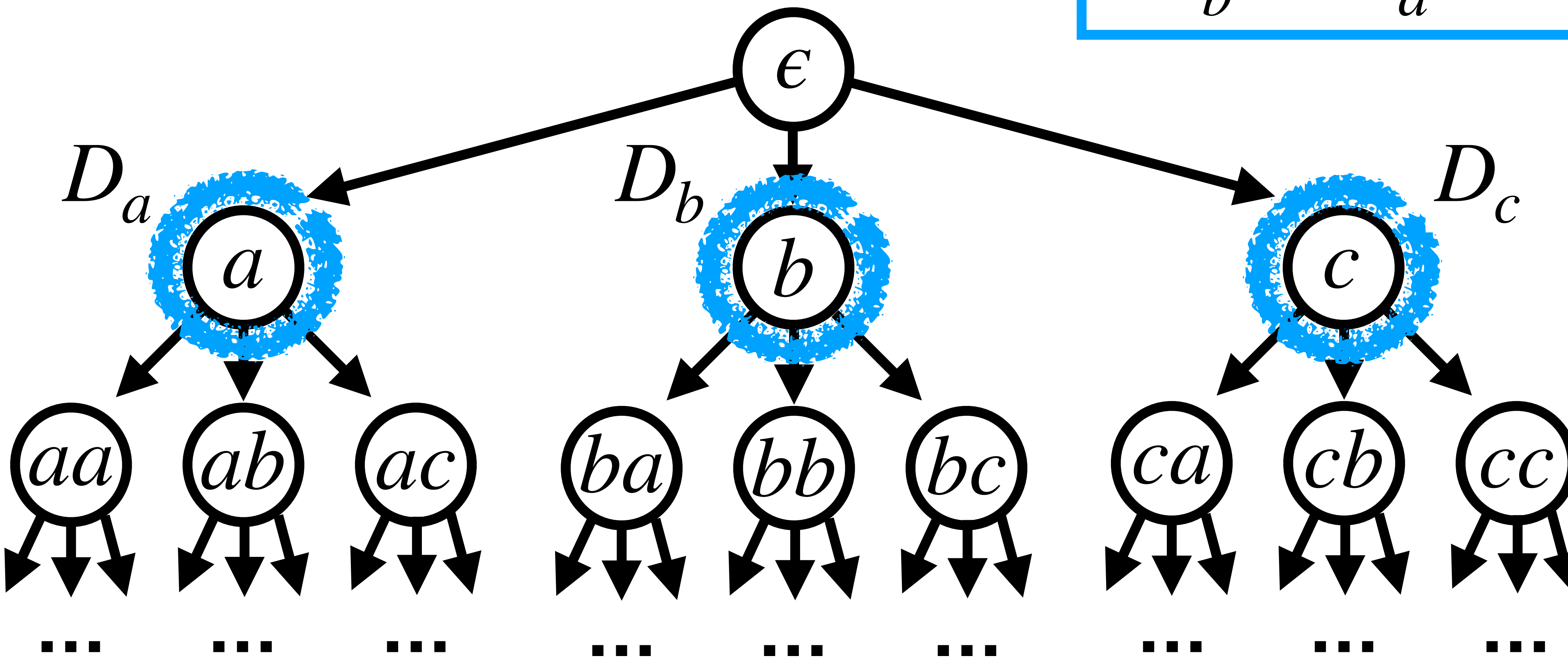


Running example



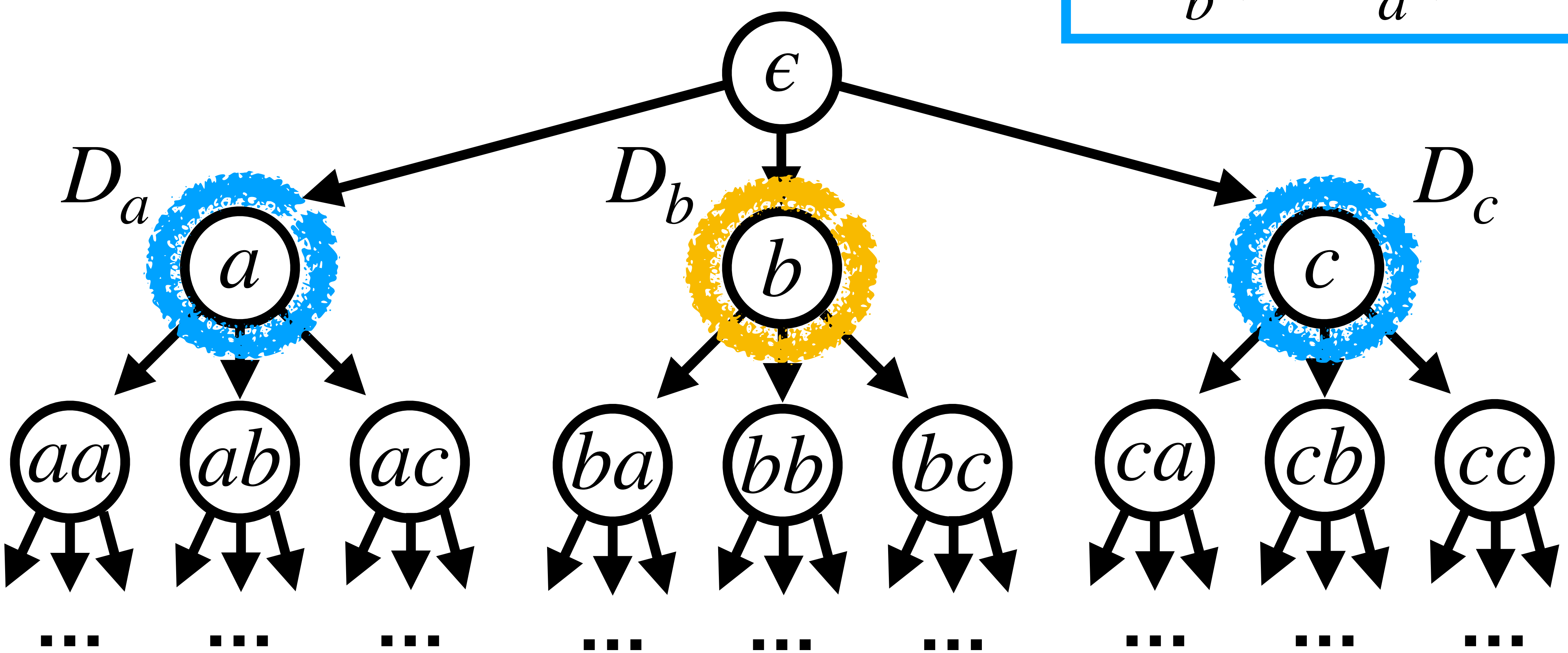
Running example

$$D_b > D_a > D_c$$

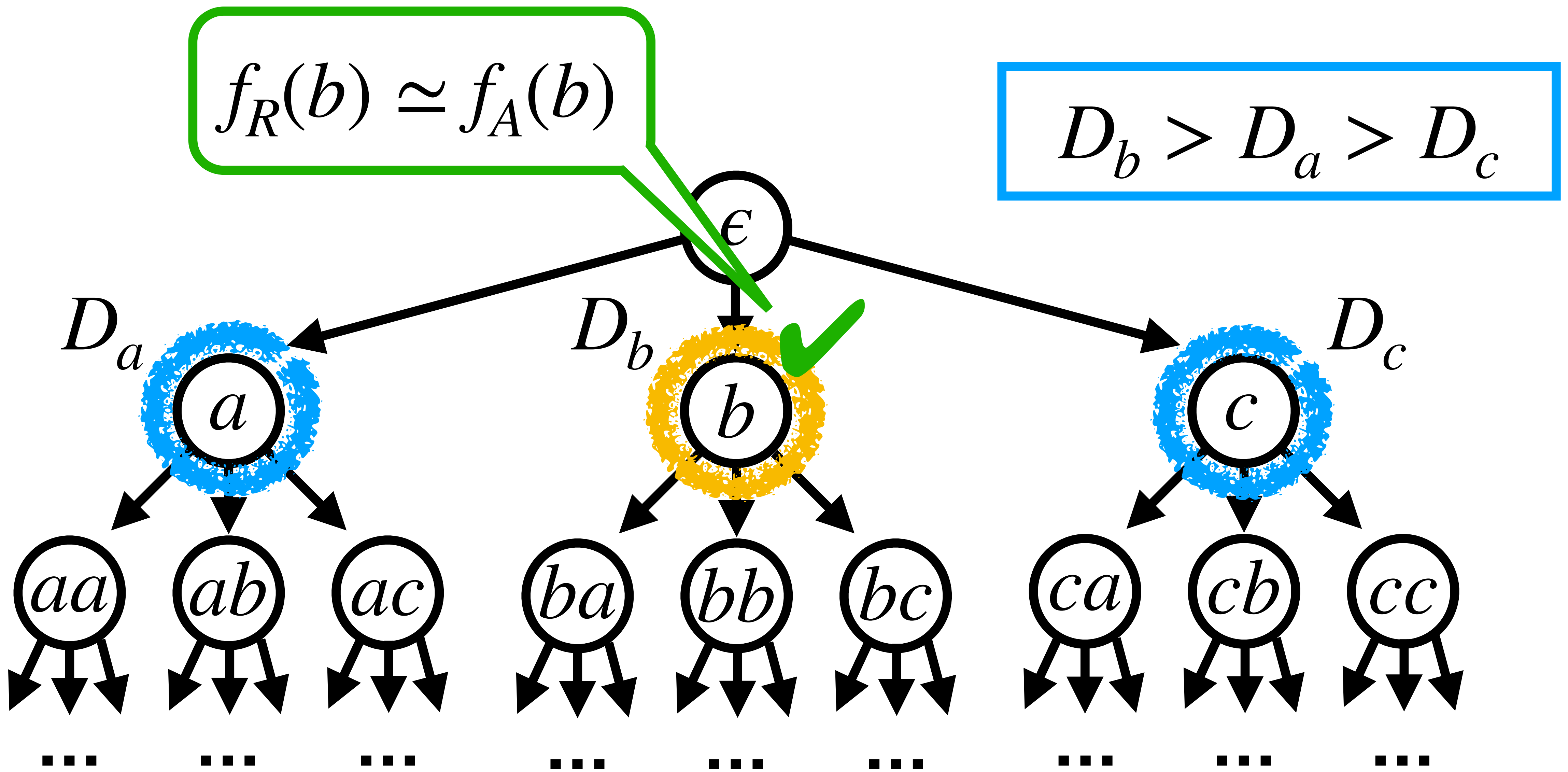


Running example

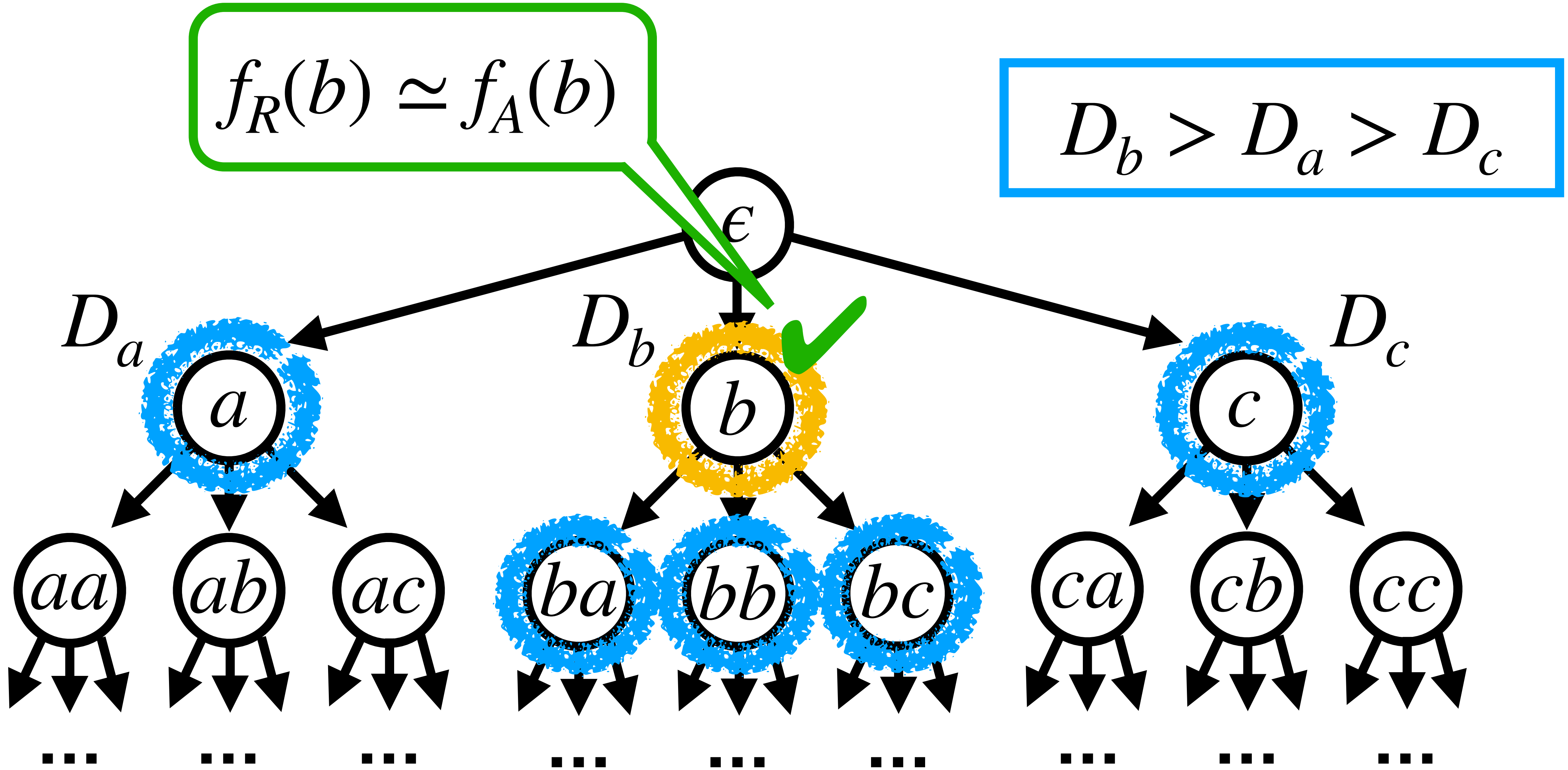
$$D_b > D_a > D_c$$



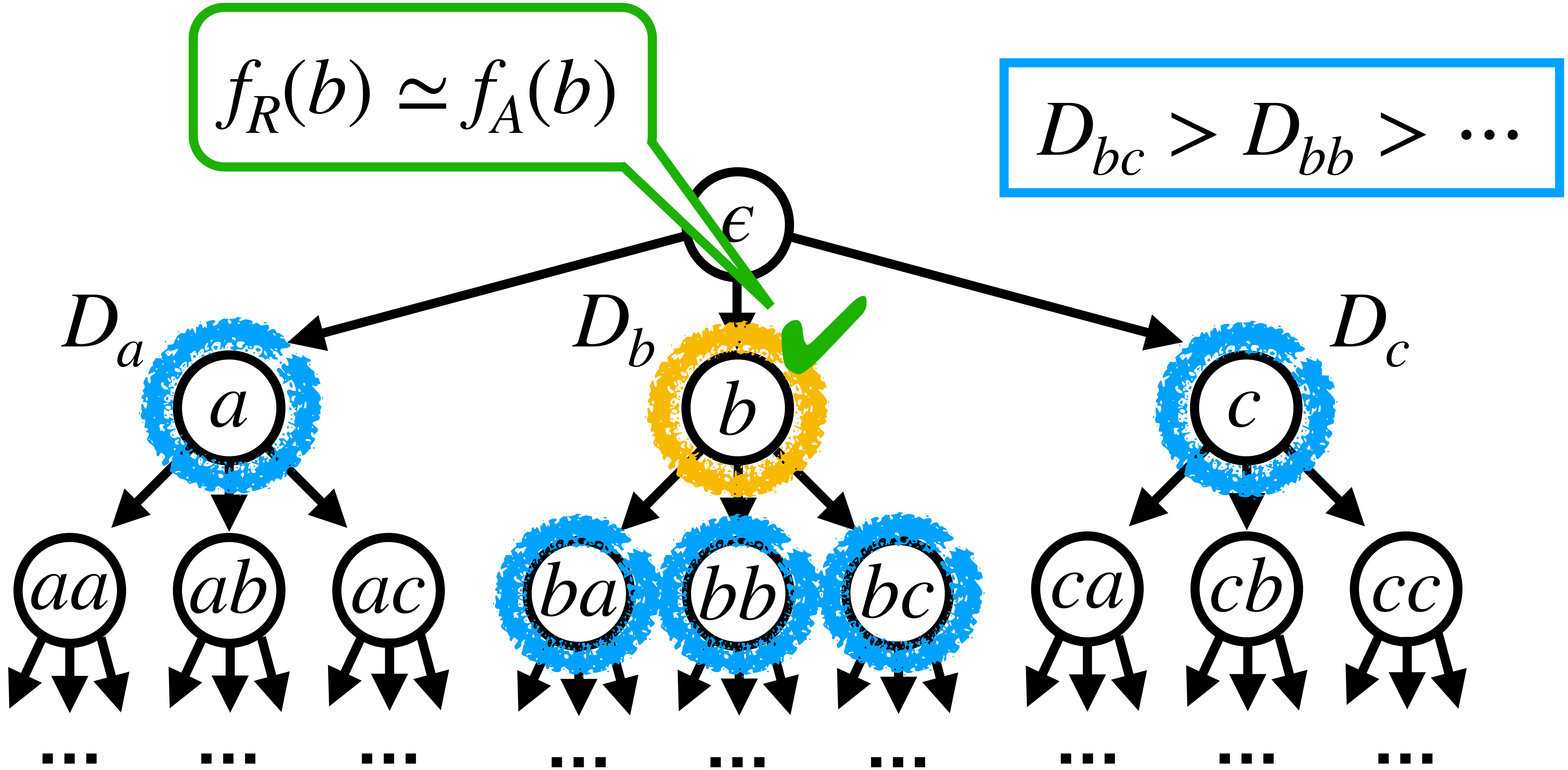
Running example



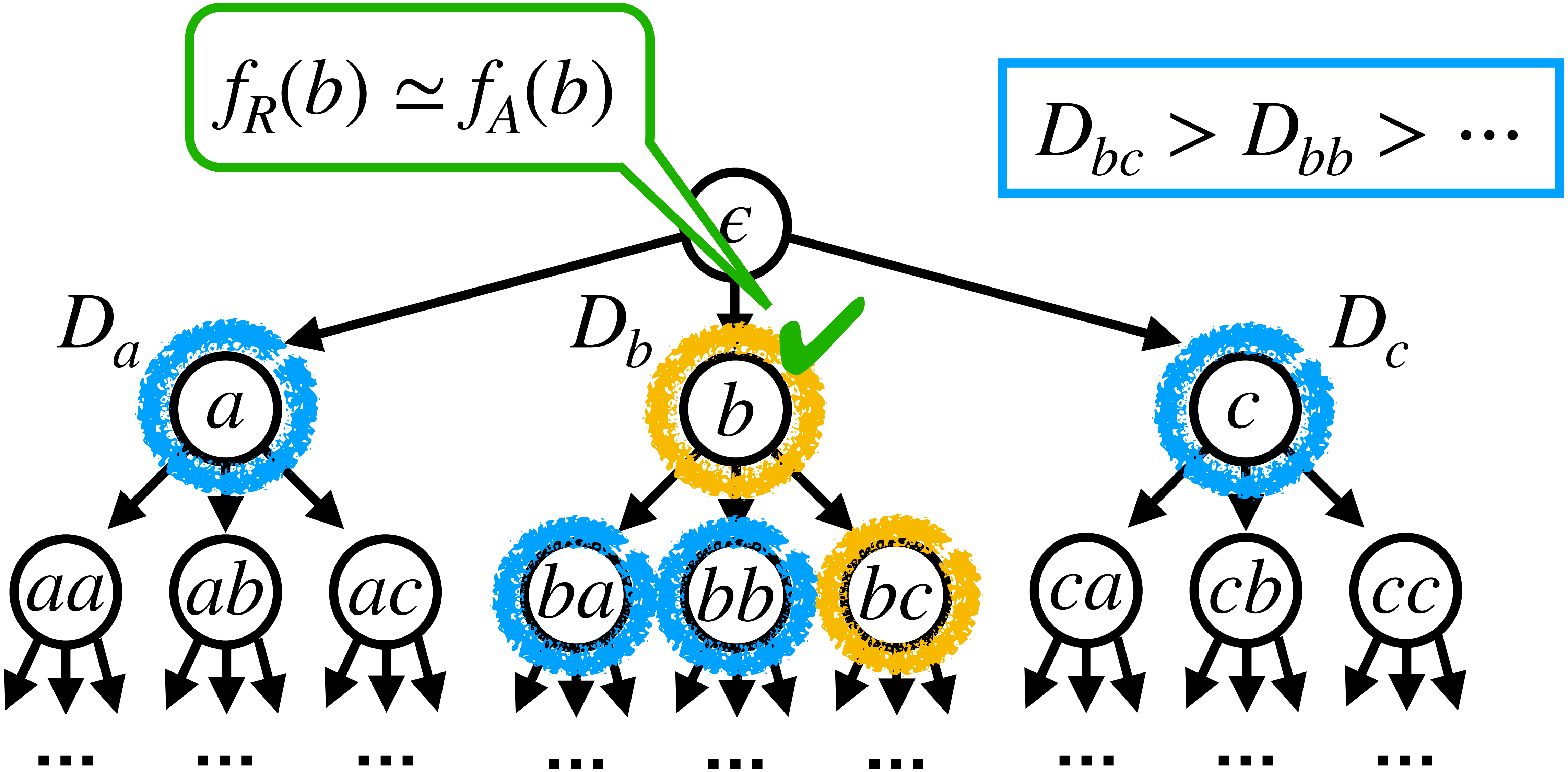
Running example



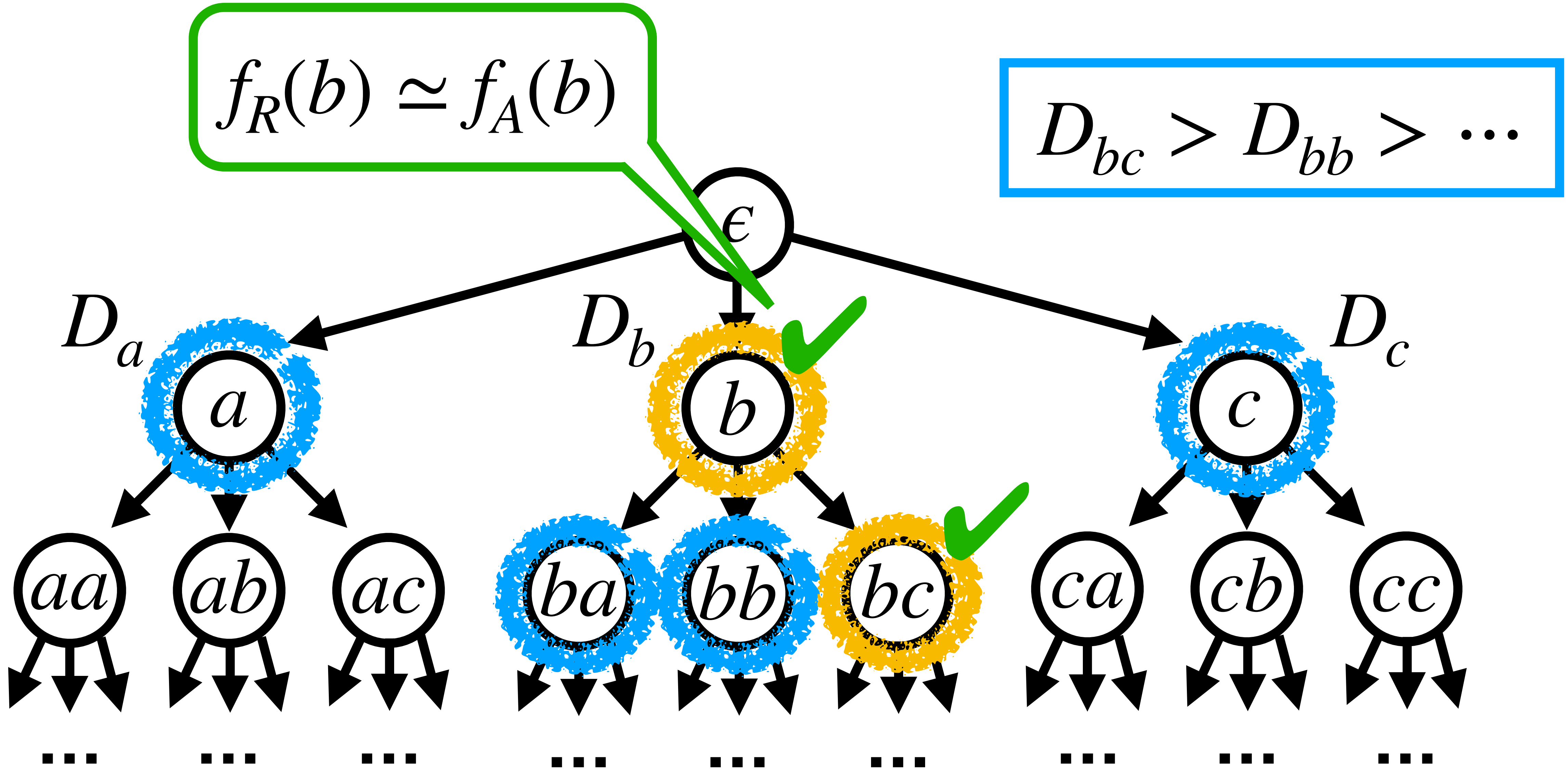
Running example



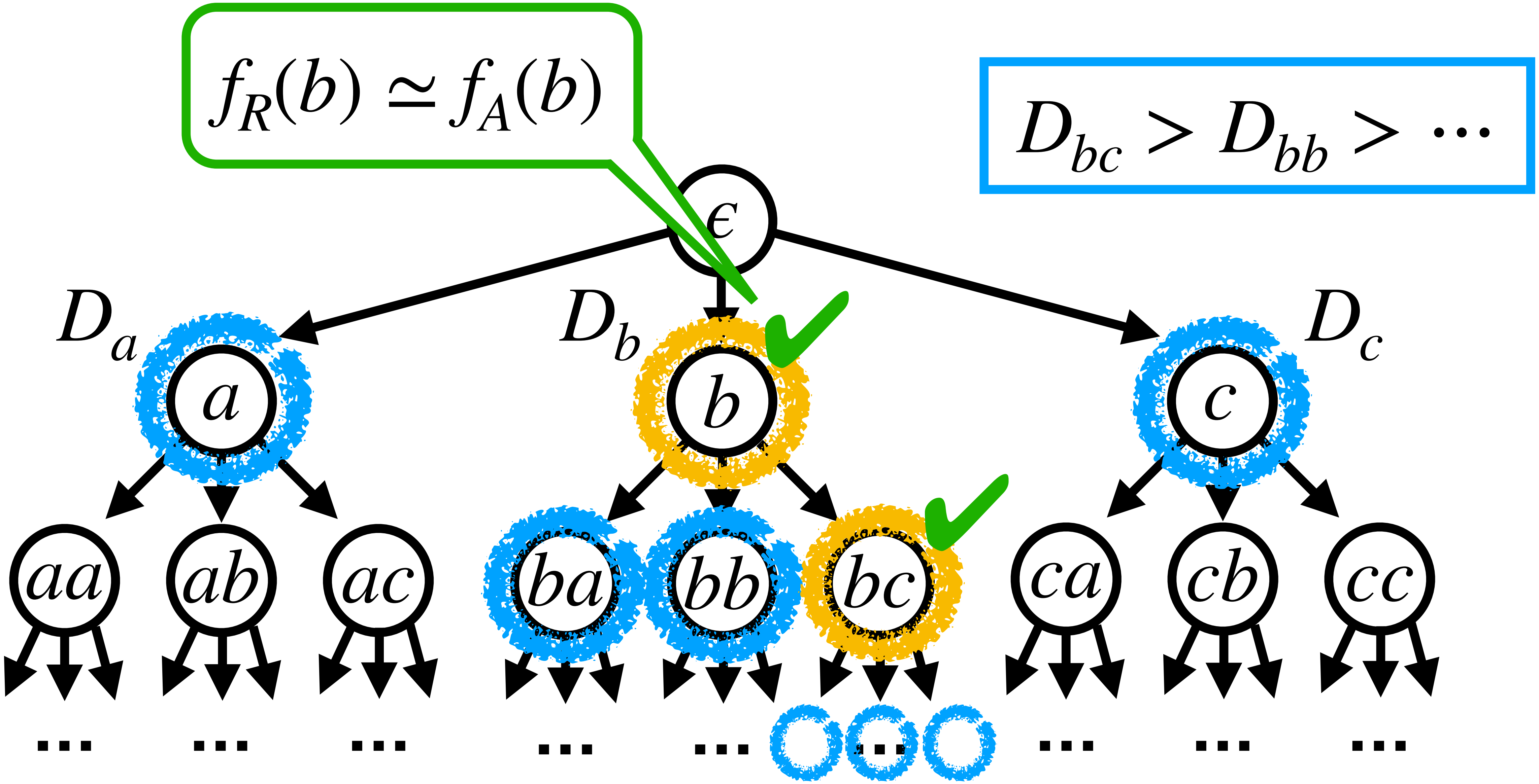
Running example



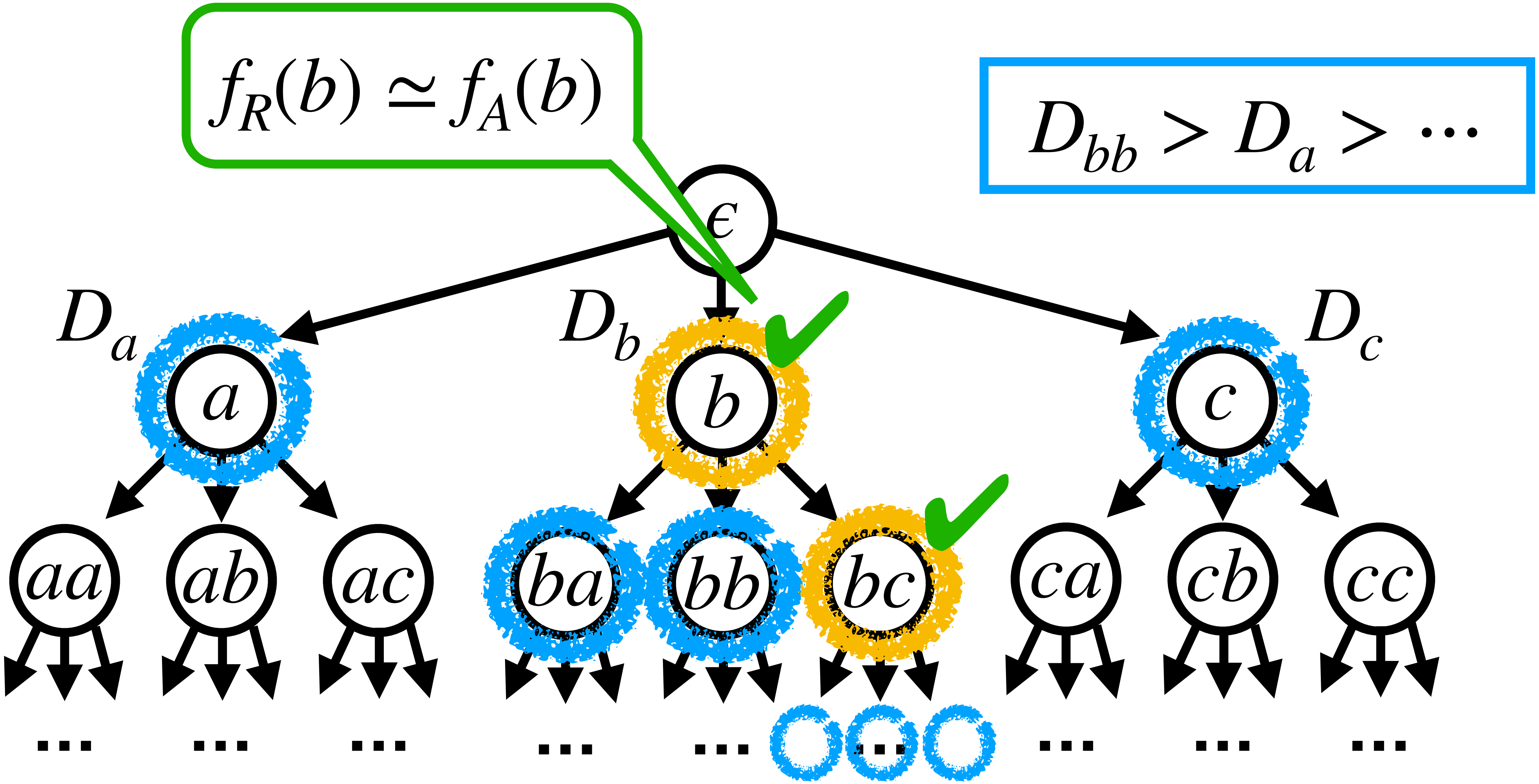
Running example



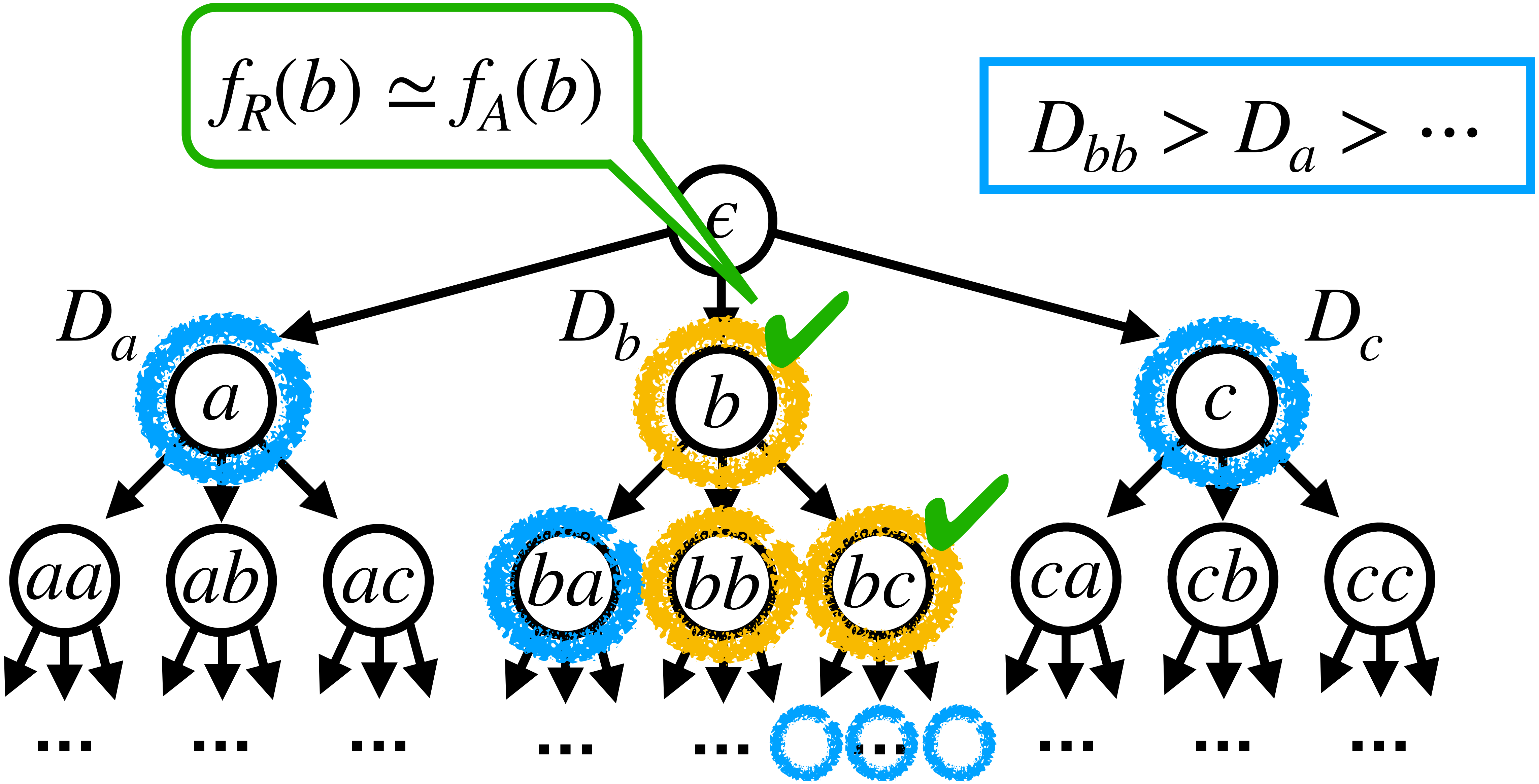
Running example



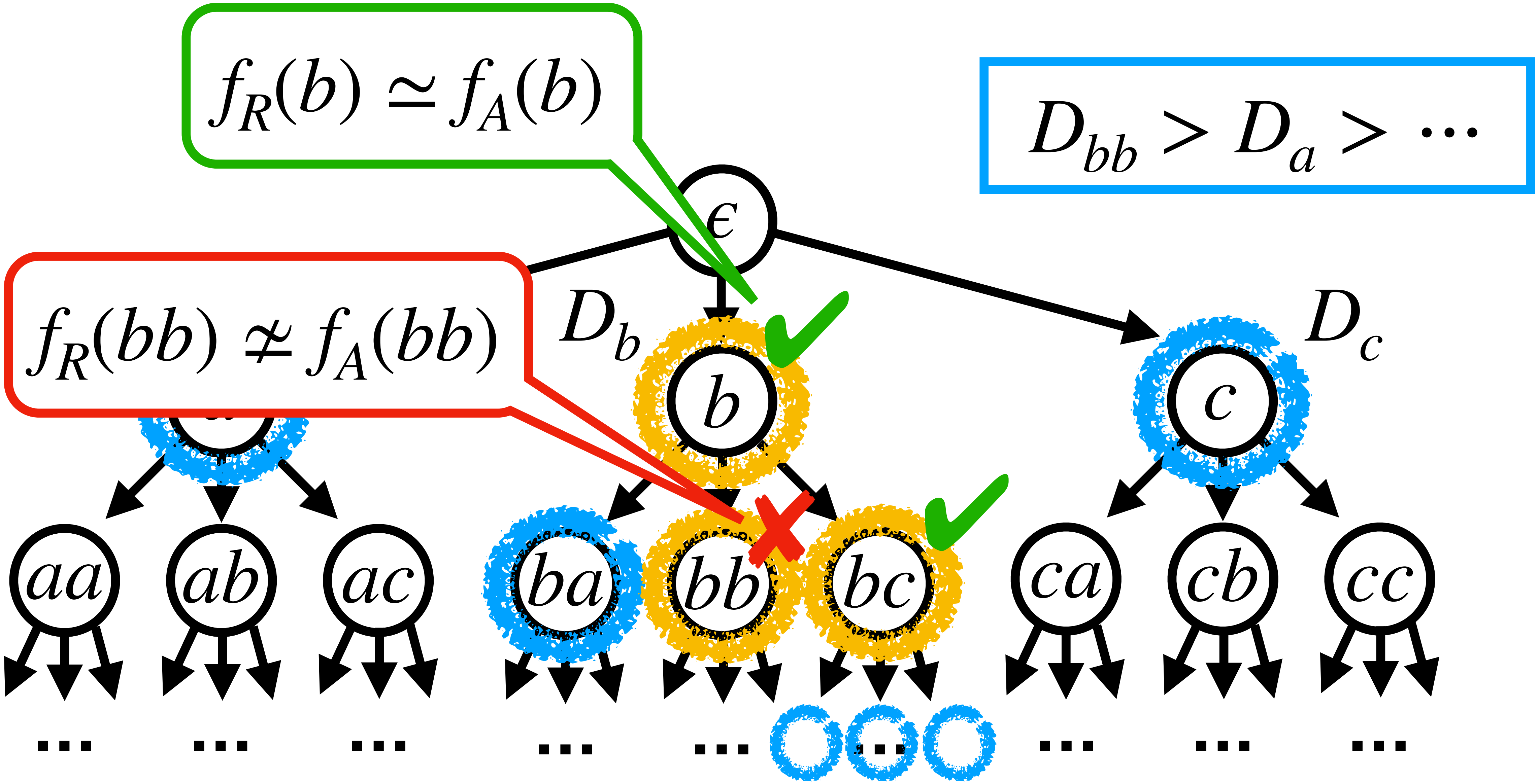
Running example



Running example



Running example



Strategy for $e_R(A)$

Goal: implement $e_R(A) = \begin{cases} \text{Eq} & (\text{if } f_R \simeq f_A) \\ w & (\text{if } f_R(w) \not\simeq f_A(w)) \end{cases}$

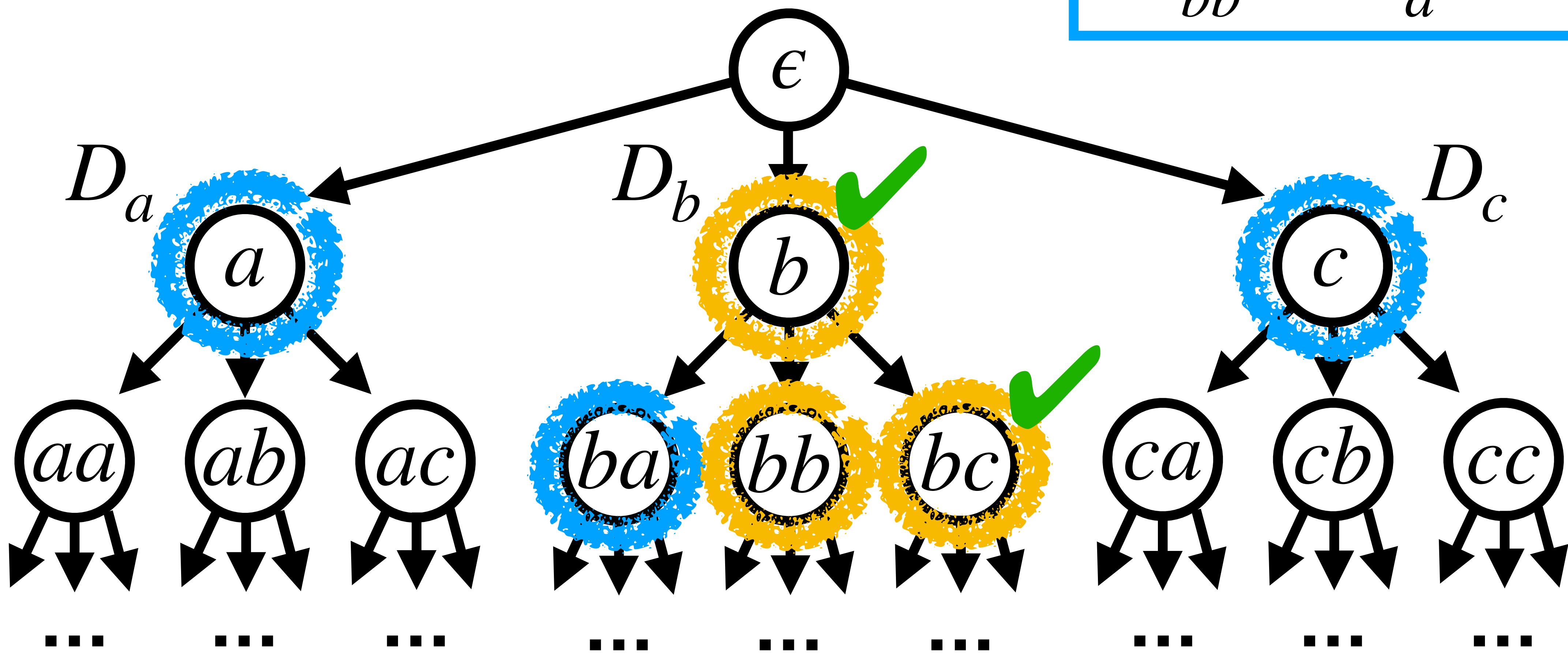
1. Trying to find counterexample w s.t. $f_R(w) \neq f_A(w)$

2(a). Returning w if found

2(b). Returning "Eq" if no counterexample is found even by sufficient search

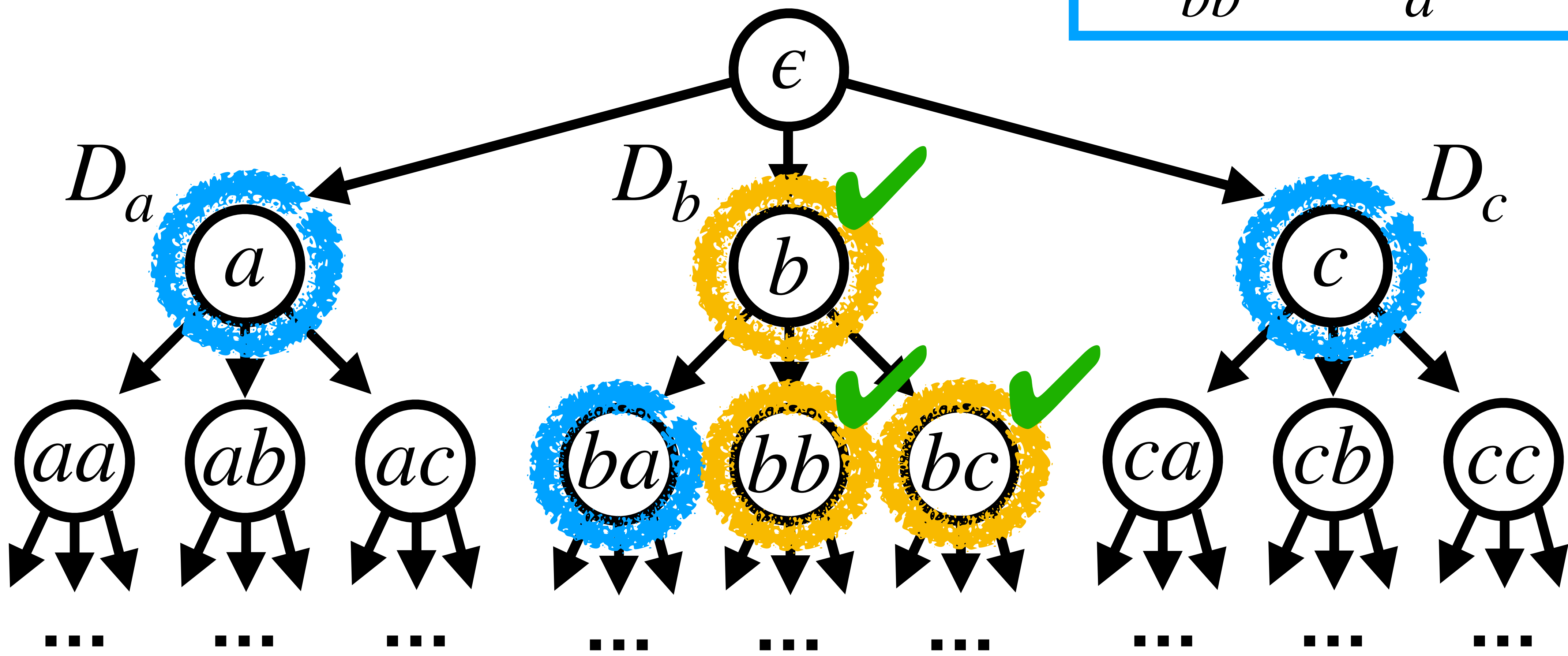
Running example

$$D_{bb} > D_a > \cdots$$



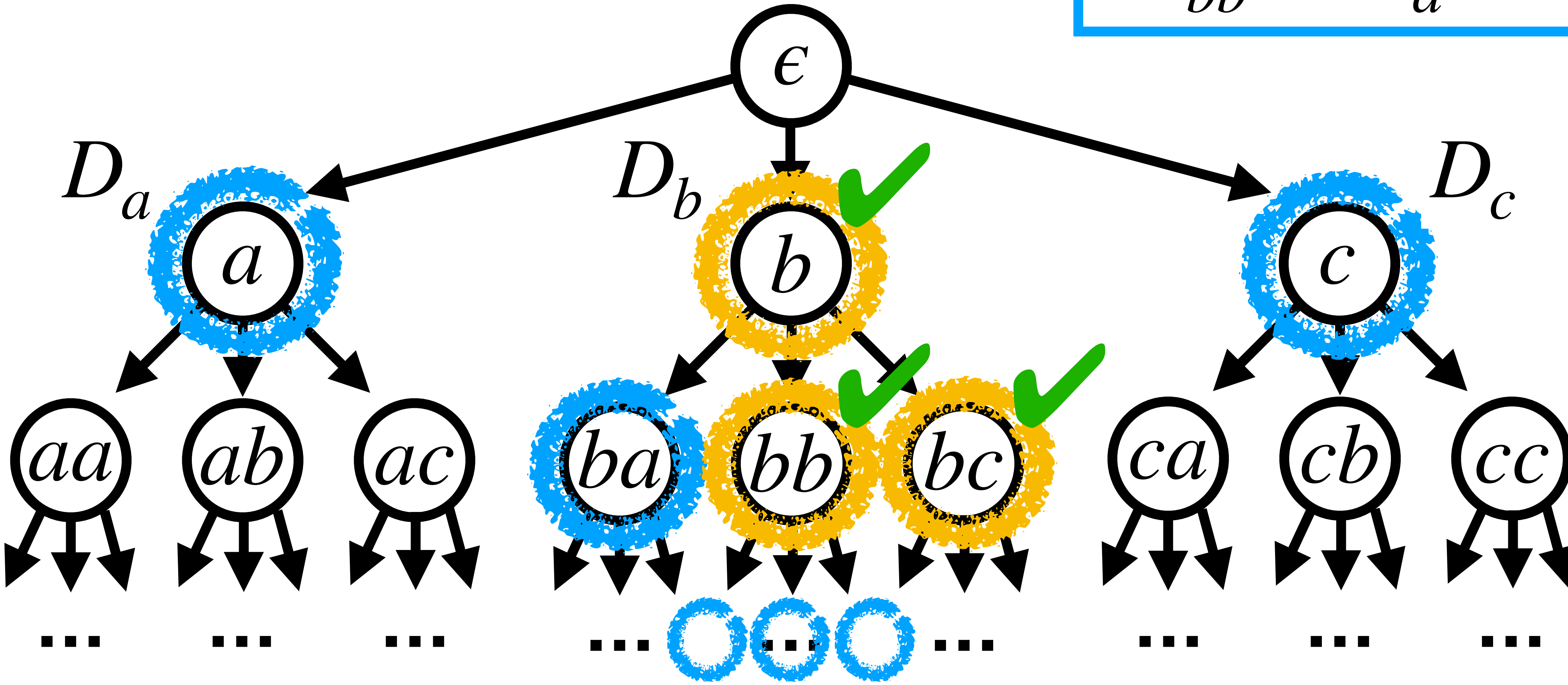
Running example

$$D_{bb} > D_a > \cdots$$

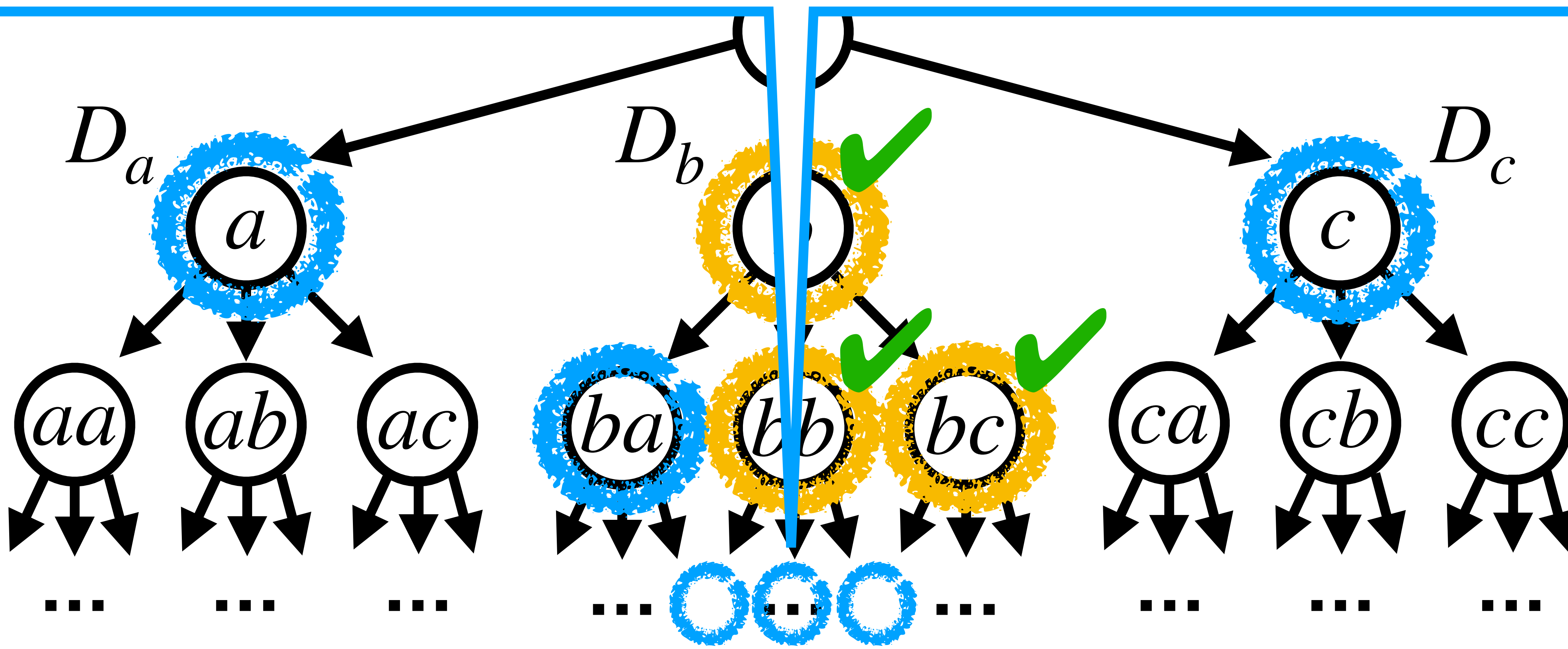


Running example

$D_{bb} > D_a > \cdots$



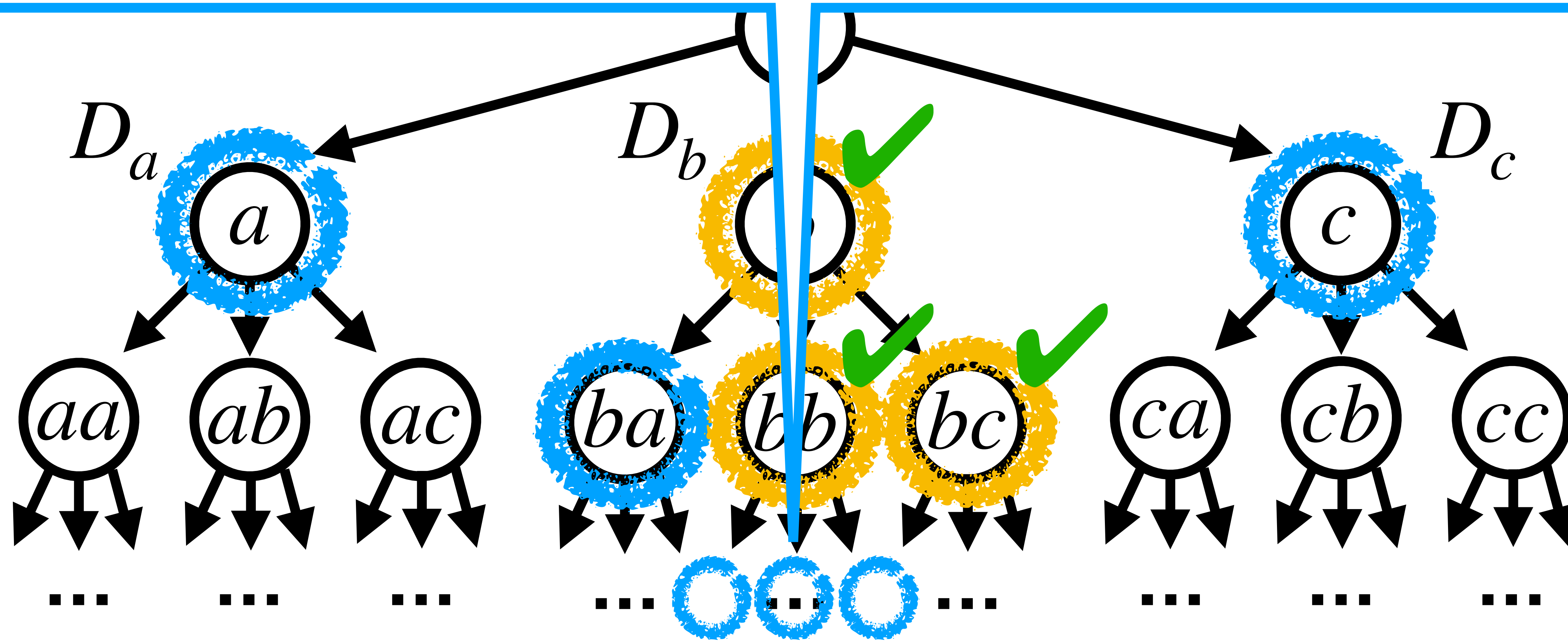
Words w (and their descendants) will not be checked if the “neighborhoods” of the WFA state for w have been investigated sufficiently



Words w (and their descendants) will not be checked if

$$p(\delta_R(w)) \simeq p(\delta_R(w'))$$

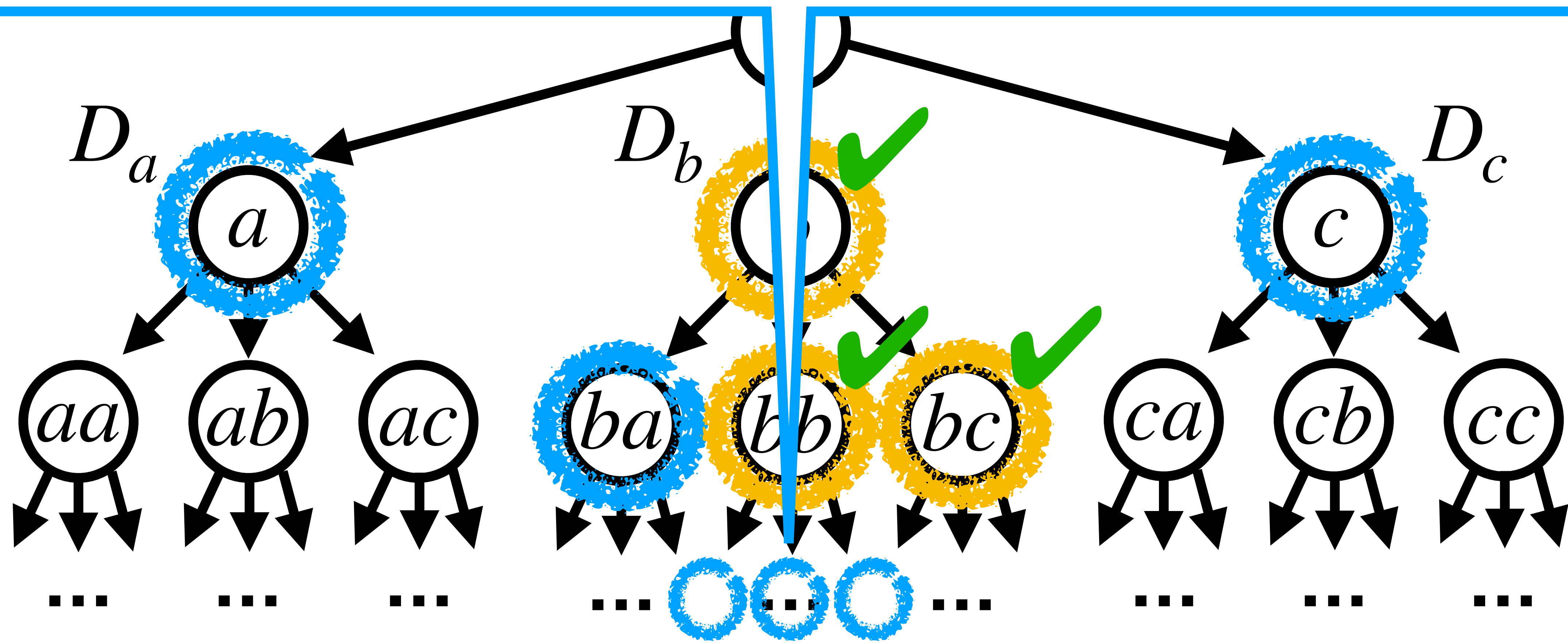
for sufficiently many already visited words w'



Words w (and their descendants) will not be checked if

$$p(\delta_R(w)) \simeq p(\delta_R(w'))$$

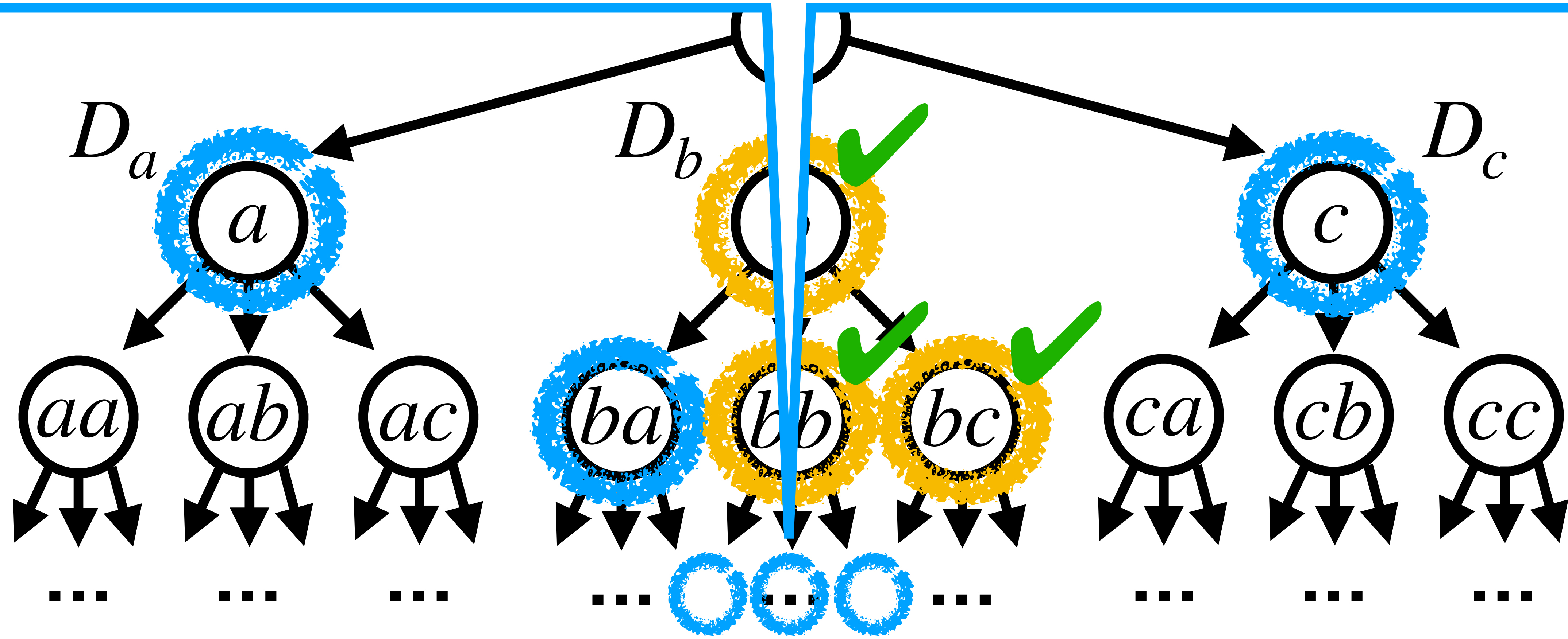
for ~~sufficiently many~~ already visited M words w' ($M \in \mathbb{N}$)



Words w (and their descendants) will not be checked if

$$p(\delta_R(w)) \simeq p(\delta_R(w'))$$

for ~~sufficiently many~~ already visited M words w' ($M \in \mathbb{N}$)



2(b). Returning “Eq” if words to check run out

Summary

We use $L^*(f_R, e_R)$ to extract WFAs from RNNs

- f_R tells the output of an RNN for an input
- e_R seeks a counterexample by regression on internal state spaces of an RNN and a WFA
  - Deeming them equivalent if counterexamples are not found by sufficient search
    - "Sufficiency" is controlled by threshold M

Outline

1. Introduction
2. Preliminaries
3. Our work: extraction of WFAs from RNNs
4. **Experiments**

Research questions

RQ1. How well do the extracted WFAs approximate to original RNNs?

RQ2. How applicable is our method to RNNs that are more expressive than WFAs?

RQ3. How are efficient the extracted WFAs, compared with original RNNs?

Experiment for RQ1

Evaluating how well the WFAs extracted by our method approximate to original RNNs

- **Ours: RGR(M)** (M is a threshold)
  - Using Gaussian process regression
- **Baseline: BFS(n)**
  - By **B**readth-**F**irst **S**earch in tree traversal for finding counterexamples
  - Returning “Eq” if no counterexample is found in consecutive **n** $+m$ words (when the previous counterexample is the m-th candidate word)

Target RNNs

- **Architecture:** two-layer LSTM networks
- **Training:** performed so that the RNNs behave similarly to randomly generated WFAs A
- **Training datasets:** pairs $(w, f_A(w))$
    - For **"WFA-like"**: w is sampled from the uniform distrib.
    - For **"Realistic"**: w is sampled from a non-uniform distrib.
        - Restricted to words where the occurrences of the same letter have to be consecutive
        - E.g., ✔ "aabccc" ✘ "aabcc<u>***a***</u>"

Extraction from “WFA-like" RNNs

$(\lvert\Sigma\rvert, \lvert Q_{A^\bullet}\rvert)$	$\mathbf{RGR}(2)$	$\mathbf{RGR}(5)$	$\mathbf{BFS}(500)$	$\mathbf{BFS}(1000)$	$\mathbf{BFS}(2000)$	$\mathbf{BFS}(3000)$	$\mathbf{BFS}(5000)$
$(4, 10)$	2.17 / 286	2.39 / 338	26.8 / 165	9.77 / 279	4.36 / 545	4.07 / 716	2.33 / 1390
$(6, 10)$	2.45 / 1787	2.54 / 1302	6.99 / 386	4.48 / 641	4.08 / 1218	3.15 / 1410	2.28 / 2480
$(10, 10)$	4.68 / 7462	4.46 / 5311	22.5 / 928	11.9 / 1562	5.90 / 3521	4.55 / 3638	3.55 / 5571
$(10, 15)$	5.62 / 8941	5.78 / 8564	21.2 / 2155	10.6 / 4750	7.87 / 5692	5.71 / 7344	5.27 / 7612
$(10, 20)$	3.70 / 7610	3.79 / 7799	6.24 / 2465	10.1 / 2188	6.13 / 3106	3.70 / 5729	3.63 / 7473
$(15, 10)$	7.34 / 9569	5.52 / 10000	13.5 / 3227	8.01 / 6765	6.07 / 7916	5.98 / 8911	6.17 / 8979
$(15, 15)$	8.44 / 10000	5.58 / 9981	16.3 / 2675	9.24 / 4850	7.28 / 5135	9.88 / 7204	6.44 / 8425
$(15, 20)$	9.16 / 7344	5.15 / 7857	13.7 / 2224	7.26 / 3823	6.60 / 5744	4.96 / 5674	4.01 / 9464
Total	5.45 / 6625	4.40 / 6394	15.9 / 1778	8.92 / 3107	6.04 / 4110	5.25 / 5078	4.21 / 6549

Each cell shows “MSE ($\times 10^4$) / extraction time (sec)”

Observation & discussion

- MSEs are lower as extractions take longer times
- (Conjecture) Because the RNNs behave well for any word and the amount of counterexamples investigated is more dominant than their “qualities”

Configurations

- $|\Sigma|$: alphabet sizes
- $|Q_{A^\bullet}|$: state sizes of WFAs

used for training RNNs

om “WFA-like" RNNs

$(\lvert\Sigma\rvert, \lvert Q_{A^\bullet}\rvert)$	**RGR**(2)	**RGR**(5)	**BFS**(500)	**BFS**(1000)	**BFS**(2000)	**BFS**(3000)	**BFS**(5000)
(4, 10)	2.17 / 286	2.39 / 338	26.8 / 165	9.77 / 279	4.36 / 545	4.07 / 716	2.33 / 1390
(6, 10)	2.45 / 1787	2.54 / 1302	6.99 / 386	4.48 / 641	4.08 / 1218	3.15 / 1410	2.28 / 2480
(10, 10)	4.68 / 7462	4.46 / 5311	22.5 / 928	11.9 / 1562	5.90 / 3521	4.55 / 3638	3.55 / 5571
(10, 15)	5.62 / 8941	5.78 / 8564	21.2 / 2155	10.6 / 4750	7.87 / 5692	5.71 / 7344	5.27 / 7612
(10, 20)	3.70 / 7610	3.79 / 7799	6.24 / 2465	10.1 / 2188	6.13 / 3106	3.70 / 5729	3.63 / 7473
(15, 10)	7.34 / 9569	5.52 / 10000	13.5 / 3227	8.01 / 6765	6.07 / 7916	5.98 / 8911	6.17 / 8979
(15, 15)	8.44 / 10000	5.58 / 9981	16.3 / 2675	9.24 / 4850	7.28 / 5135	9.88 / 7204	6.44 / 8425
(15, 20)	9.16 / 7344	5.15 / 7857	13.7 / 2224	7.26 / 3823	6.60 / 5744	4.96 / 5674	4.01 / 9464
Total	5.45 / 6625	4.40 / 6394	15.9 / 1778	8.92 / 3107	6.04 / 4110	5.25 / 5078	4.21 / 6549

Each cell shows “MSE ($\times 10^4$) / extraction time (sec)”

Observation & discussion

- MSEs are lower as extractions take longer times
- (Conjecture) Because the RNNs behave well for any word and the amount of counterexamples investigated is more dominant than their “qualities”

Configurations

- $|\Sigma|$: alphabet sizes
- $|Q_{A^\bullet}|$: state sizes of WFAs

used for training RNNs

...om “WFA-like" RNNs

Ours: RGR(M)

$(\lvert\Sigma\rvert, \lvert Q_{A^\bullet}\rvert)$	**RGR**(2)	**RGR**(5)	**BFS**(500)	**BFS**(1000)	**BFS**(2000)	**BFS**(3000)	**BFS**(5000)
(4, 10)	2.17 / 286	2.39 / 338	26.8 / 165	9.77 / 279	4.36 / 545	4.07 / 716	2.33 / 1390
(6, 10)	2.45 / 1787	2.54 / 1302	6.99 / 386	4.48 / 641	4.08 / 1218	3.15 / 1410	2.28 / 2480
(10, 10)	4.68 / 7462	4.46 / 5311	22.5 / 928	11.9 / 1562	5.90 / 3521	4.55 / 3638	3.55 / 5571
(10, 15)	5.62 / 8941	5.78 / 8564	21.2 / 2155	10.6 / 4750	7.87 / 5692	5.71 / 7344	5.27 / 7612
(10, 20)	3.70 / 7610	3.79 / 7799	6.24 / 2465	10.1 / 2188	6.13 / 3106	3.70 / 5729	3.63 / 7473
(15, 10)	7.34 / 9569	5.52 / 10000	13.5 / 3227	8.01 / 6765	6.07 / 7916	5.98 / 8911	6.17 / 8979
(15, 15)	8.44 / 10000	5.58 / 9981	16.3 / 2675	9.24 / 4850	7.28 / 5135	9.88 / 7204	6.44 / 8425
(15, 20)	9.16 / 7344	5.15 / 7857	13.7 / 2224	7.26 / 3823	6.60 / 5744	4.96 / 5674	4.01 / 9464
Total	5.45 / 6625	4.40 / 6394	15.9 / 1778	8.92 / 3107	6.04 / 4110	5.25 / 5078	4.21 / 6549

Each cell shows “MSE ($\times 10^4$) / extraction time (sec)”

Observation & discussion

- MSEs are lower as extractions take longer times
- (Conjecture) Because the RNNs behave well for any word and the amount of counterexamples investigated is more dominant than their “qualities”

Configurations

- $|\Sigma|$: alphabet sizes
- $|Q_{A\bullet}|$: state sizes of WFAs used for training RNNs

om “WFA-like" RNNs

Ours: RGR(M) **Baseline: BFS(n)**

$(\lvert\Sigma\rvert, \lvert Q_{A\bullet}\rvert)$	**RGR**(2)	**RGR**(5)	**BFS**(500)	**BFS**(1000)	**BFS**(2000)	**BFS**(3000)	**BFS**(5000)
(4, 10)	2.17 / 286	2.39 / 338	26.8 / 165	9.77 / 279	4.36 / 545	4.07 / 716	2.33 / 1390
(6, 10)	2.45 / 1787	2.54 / 1302	6.99 / 386	4.48 / 641	4.08 / 1218	3.15 / 1410	2.28 / 2480
(10, 10)	4.68 / 7462	4.46 / 5311	22.5 / 928	11.9 / 1562	5.90 / 3521	4.55 / 3638	3.55 / 5571
(10, 15)	5.62 / 8941	5.78 / 8564	21.2 / 2155	10.6 / 4750	7.87 / 5692	5.71 / 7344	5.27 / 7612
(10, 20)	3.70 / 7610	3.79 / 7799	6.24 / 2465	10.1 / 2188	6.13 / 3106	3.70 / 5729	3.63 / 7473
(15, 10)	7.34 / 9569	5.52 / 10000	13.5 / 3227	8.01 / 6765	6.07 / 7916	5.98 / 8911	6.17 / 8979
(15, 15)	8.44 / 10000	5.58 / 9981	16.3 / 2675	9.24 / 4850	7.28 / 5135	9.88 / 7204	6.44 / 8425
(15, 20)	9.16 / 7344	5.15 / 7857	13.7 / 2224	7.26 / 3823	6.60 / 5744	4.96 / 5674	4.01 / 9464
Total	5.45 / 6625	4.40 / 6394	15.9 / 1778	8.92 / 3107	6.04 / 4110	5.25 / 5078	4.21 / 6549

Each cell shows “MSE ($\times 10^4$) / extraction time (sec)”

Observation & discussion

- MSEs are lower as extractions take longer times
- (Conjecture) Because the RNNs behave well for any word and the amount of counterexamples investigated is more dominant than their “qualities”

Extraction from "Realistic" RNNs

Ours: RGR(M) **Baseline: BFS(n)**

$(\lvert\Sigma\rvert, \lvert Q_{A^\bullet}\rvert)$	RGR(2)	RGR(5)	BFS(500)	BFS(1000)	BFS(2000)	BFS(3000)	BFS(5000)
(4, 10)	7.73 / 696	7.07 / 1135	15.0 / 199	7.96 / 424	6.62 / 650	6.61 / 762	9.06 / 1693
(6, 10)	4.92 / 1442	7.43 / 1247	1.46 / 552	6.95 / 660	5.90 / 1217	8.78 / 1557	3.54 / 2237
(10, 10)	5.02 / 5536	4.28 / 5951	7.70 / 1117	11.0 / 1738	4.77 / 2635	3.52 / 3926	4.52 / 4777
(10, 15)	7.15 / 6977	4.35 / 8315	19.4 / 1552	13.8 / 3271	16.8 / 3209	8.57 / 5293	5.08 / 6522
(10, 20)	6.98 / 4697	8.06 / 6704	18.6 / 1465	11.8 / 2046	12.7 / 2851	9.03 / 4259	8.01 / 4856
(15, 10)	5.97 / 8747	6.77 / 8882	23.3 / 2359	11.2 / 4668	9.88 / 6186	6.24 / 7557	6.02 / 8245
(15, 15)	5.78 / 8325	8.71 / 7546	16.6 / 2874	7.31 / 4380	9.92 / 6015	9.89 / 7110	6.40 / 8358
(15, 20)	4.60 / 7652	8.56 / 8334	36.9 / 1893	23.7 / 3069	12.8 / 3987	12.0 / 5262	8.38 / 6441
Total	6.02 / 5510	6.90 / 6015	19.0 / 1502	11.7 / 2532	9.92 / 3344	8.08 / 4466	6.38 / 5391

Each cell shows "MSE ($\times 10^4$) / extraction time (sec)"

Observation & discussion

- **RGR(2)** performs the best
- (Conjecture) BFS(n) investigates the input space on which the RNNs are ***NOT*** trained in more detail
- (Conjecture) RGR(2) avoids it by early pruning

Experiment for RQ2

- Extraction from RNNs learning languages more expressive than any WFAs

Experiment for RQ2

- Extraction from RNNs learning languages more expressive than any WFAs
- RNN learning target: wparen (balanced parentheses)

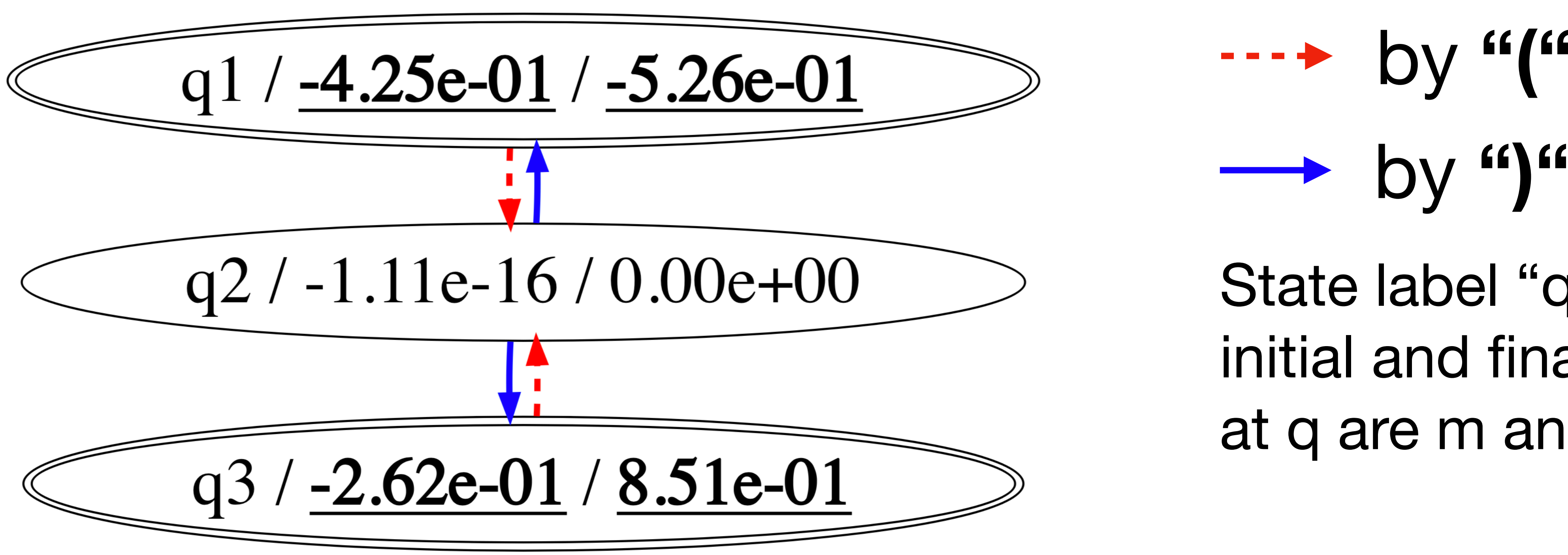
$$\mathsf{wparen}(w) = \begin{cases} 0 & \text{(if } w \text{ contains unbalanced parentheses)} \\ 1 - (1/2)^N & \text{(otherwise; } N \text{ is the depth of deepest paren.)} \end{cases}$$

Example:

$\mathsf{wparen}(''((a)(b))'') = 1 - (1/2)^2$ $\quad$ $\mathsf{wparen}(''((a)(b)'') = 0$

Extraction result

- The WFA extracted by RGR(5)
  - Only largely weighted transitions by parentheses are shown
  - Transition weights for letters other than parentheses are similar

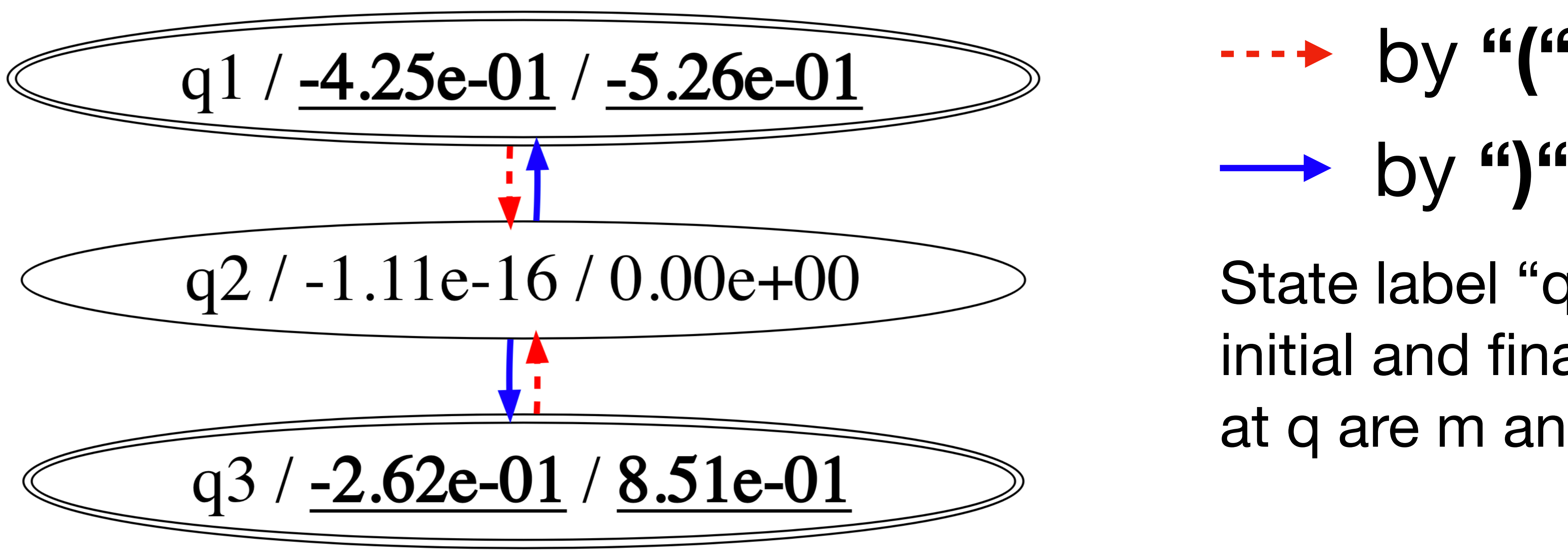


State label “q/m/n”: initial and final weights at q are m and n, resp.

Extraction result

- The WFA extracted by RGR(5)
  - Only largely weighted transitions by parentheses are shown
  - Transition weights for letters other than parentheses are similar

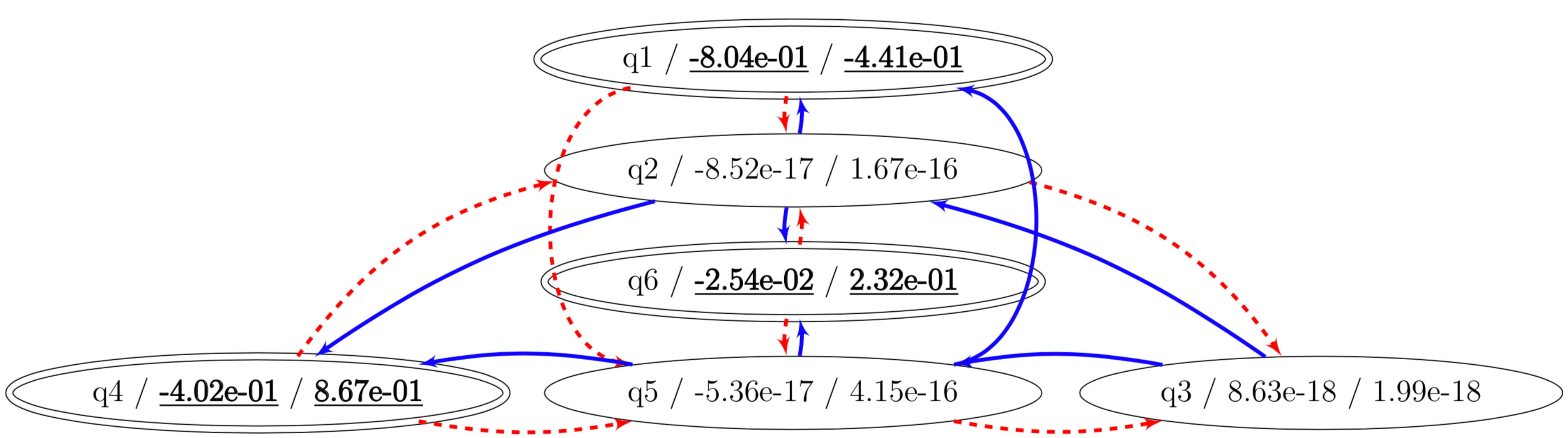
wparen is learnt successfully at least up to depth 1



State label "q/m/n": initial and final weights at q are m and n, resp.

Extraction result

- The WFA extracted by RGR(15)
    - Only largely weighted transitions by parentheses are shown
    - Transition weights for letters other than parentheses are similar



by “(“

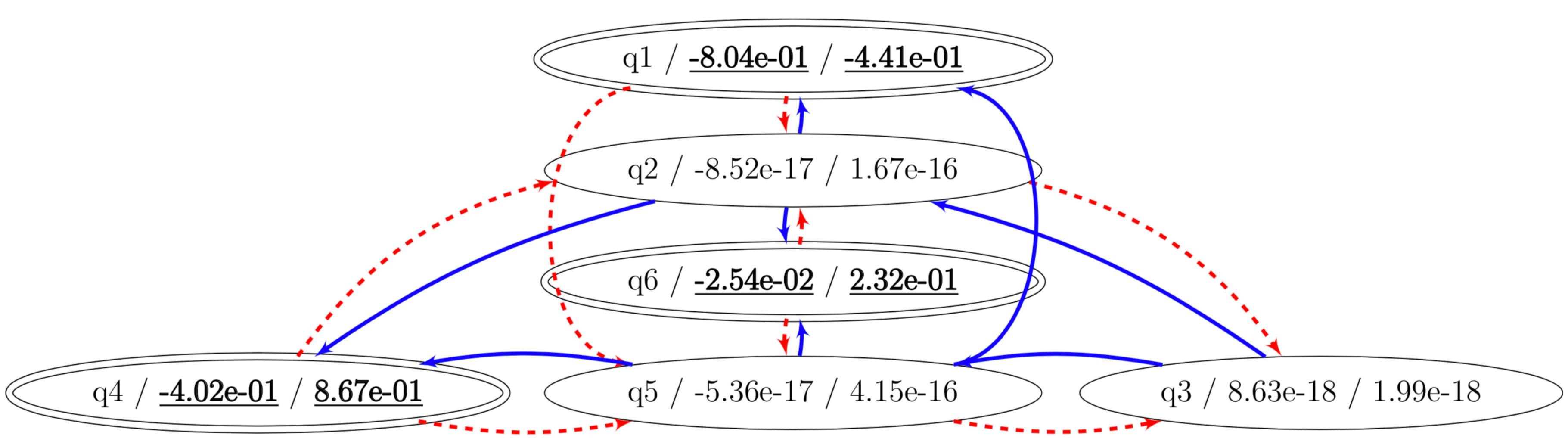
by “)“

State label “q/m/n”: initial and final weights at q are m and n, resp.

Extraction result

- The WFA extracted by RGR(15)
  - Only largely weighted transitions by parentheses are shown
  - Transition weights for letters other than parentheses are similar

wparen is learnt successfully at least up to depth 2



by “(“

by “)“

State label “q/m/n”: initial and final weights at q are m and n, resp.

Experiment for RQ3

- Comparison of efficiency of running f_R and f_A for RNN R and the extracted WFA A

- **Result:** f_A is 1,300 times faster than f_R in average
  - The RNNs are the same as in RQ1

Conclusion

- Extracting WFAs from RNNs by $L^*(f_R, e_R)$
  - f_R tells the output of an RNN for an input
  - e_R seeks a counterexample by regression on internal state spaces of an RNN and a WFA
- Future work
  - More sophisticated search for counterexamples
  - Dealing with huge alphabet sets
  - Theoretical guarantee (in an ideal case)