A Categorical Framework for Learning Generalised Tree Automata

Gerco van Heerdt Tobias Kappé Jurriaan Rot
Matteo Sammartino Alexandra Silva

April 2, 2022

The $\mathrm{L}^\star$ algorithm (Angluin, 1987)

Finite alphabet A

System behaviour captured by a **regular language** $\mathcal{L} \subseteq A^*$

$\mathrm{L}^\star$ learns *minimal* DFA for $\mathcal{L}$ assuming an *oracle* that answers

- **Membership queries**

$$w \in \mathcal{L}?$$

- **Equivalence queries**

$$\mathcal{L}(H) = \mathcal{L}?$$

  Negative result $\Longrightarrow$ *counterexample*

$L^\star$ observation table

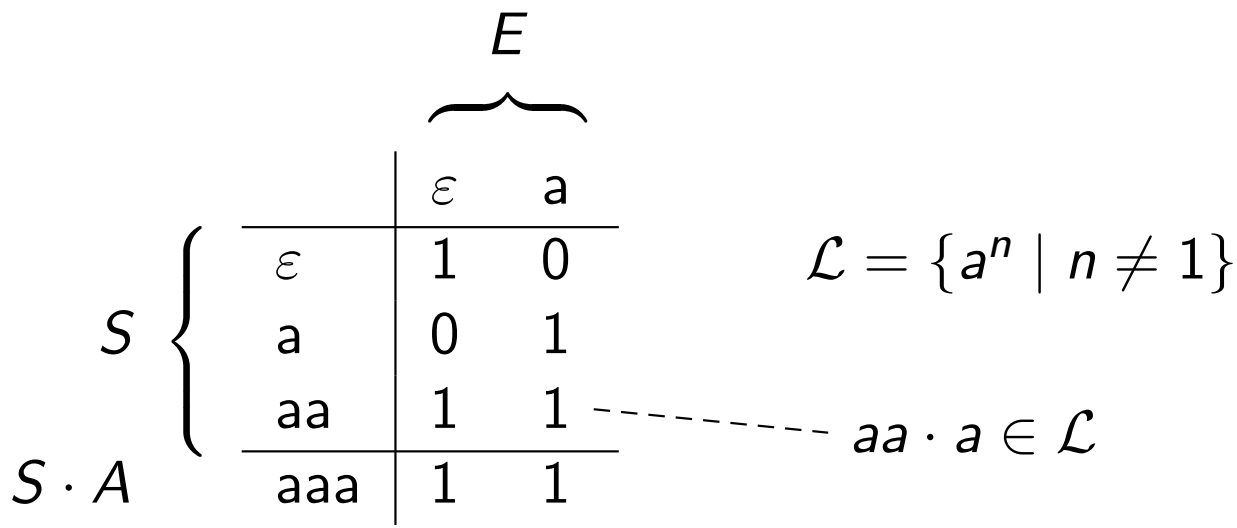
$L^\star$ maintains $S, E \subseteq A^*$ inducing a table

E

		ε	a
S	ε	1	0
	a	0	1
	aa	1	1
$S \cdot A$	aaa	1	1

$L^\star$ observation table

$L^\star$ maintains $S, E \subseteq A^*$ inducing a table



Prepend *row label* to *column label* and pose **membership query**

$$(s, e) \mapsto \begin{cases} 1 & \text{if } se \in \mathcal{L} \\ 0 & \text{if } se \notin \mathcal{L} \end{cases}$$

$L^\star$ hypothesis DFA

Hypothesis states are upper rows of the table; transitions append symbols to row labels

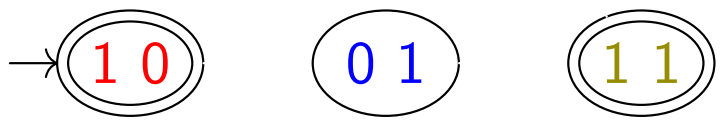
	ε	a
ε	1	0
a	0	1
aa	1	1
aaa	1	1

Requires properties **closedness** and **consistency** to be well-defined

$L^\star$ hypothesis DFA

Hypothesis states are upper rows of the table; transitions append symbols to row labels

	ε	a
ε	1	0
a	0	1
aa	1	1
aaa	1	1

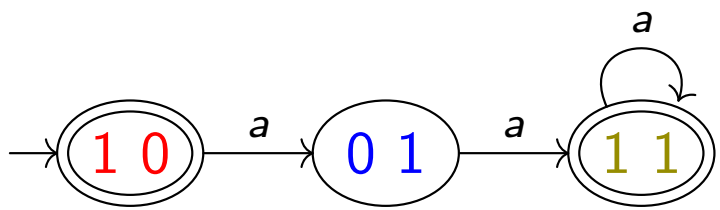


Requires properties **closedness** and **consistency** to be well-defined

$L^\star$ hypothesis DFA

Hypothesis states are upper rows of the table; transitions append symbols to row labels

	ε	a
ε	1	0
a	0	1
aa	1	1
aaa	1	1



Requires properties **closedness** and **consistency** to be well-defined

$L^\star$ algorithm overview

table updated using membership queries

1. Initialise $S = E = \{\varepsilon\}$
2. Satisfy closedness and consistency (by augmenting S and E)
3. Construct hypothesis
4. Pose equivalence query
5. On a counterexample, add its prefixes to S and repeat from 2

DFAs vs (bottom-up) generalised tree automata

DFA

$$1 + Q \times A \downarrow Q$$

accept/reject map $Q \to 2$

DFAs vs (bottom-up) generalised tree automata

DFA

$$1 + Q \times A \downarrow Q$$

accept/reject map $Q \to 2$

tree automaton

$$I + FQ \downarrow Q$$

accept/reject map $Q \to 2$

DFAs vs (bottom-up) generalised tree automata

DFA

$$1 + Q \times A \downarrow Q$$

accept/reject map $Q \to 2$

tree automaton

$$I + FQ \downarrow Q$$

accept/reject map $Q \to 2$

Semantics: language of trees generated by F and I

DFAs vs (bottom-up) generalised tree automata

DFA

$$1 + Q \times A \downarrow Q$$

accept/reject map $Q \to 2$

tree automaton

$$I + FQ \downarrow Q$$

accept/reject map $Q \to 2$

Semantics: language of trees generated by F and I

I finite, $F : \mathbf{Set} \to \mathbf{Set}$ *strongly finitary*

Trees generated by functor

Trees generated by F with leaves in I:

$$F^* I = \mathsf{lfp}(I + F(-))$$

F^* is the *free monad* over F

Trees generated by functor

Trees generated by F with leaves in I:

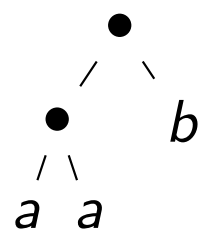
$$F^* I = \mathsf{lfp}(I + F(-))$$

F^* is the *free monad* over F

Example:

$$FX = X \times X \qquad\qquad I = \{a, b\}$$

Then e.g.



$\in F^* I$

Strongly finitary functor

Finitary: $u \in FX$ "contains" finitely many elements of X

- $FX = \mathcal{P}X = \{U \mid U \subseteq X\}$ is not but
- $FX = \mathcal{P}_{\mathrm{fin}}X = \{U \mid U \subseteq X, U \text{ finite}\}$ is

Strongly finitary functor

Finitary: $u \in FX$ "contains" finitely many elements of X

- $FX = \mathcal{P}X = \{U \mid U \subseteq X\}$ is not but
- $FX = \mathcal{P}_{\mathrm{fin}}X = \{U \mid U \subseteq X, U \text{ finite}\}$ is

Strongly finitary: F also preserves finite sets

- $FX = X^*$ is not but
- $FX = \mathcal{P}_{\mathrm{fin}}X$ is

Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$a \mapsto q_{\text{odd}}$ $\quad$ $b \mapsto q_{\text{even}}$

Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$$a \mapsto q_{\text{odd}} \qquad b \mapsto q_{\text{even}}$$

$$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}} \qquad (q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$$

$$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}} \qquad (q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$$

Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$$a \mapsto q_{\text{odd}} \qquad b \mapsto q_{\text{even}}$$

$$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}} \qquad (q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$$

$$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}} \qquad (q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$$

$$o(q_{\text{even}}) = 1 \qquad o(q_{\text{odd}}) = 0$$

Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

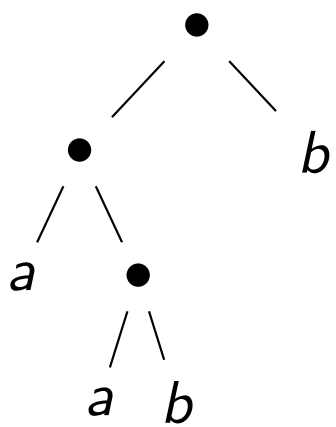
$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$$a \mapsto q_{\text{odd}} \qquad b \mapsto q_{\text{even}}$$

$$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}} \qquad (q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$$

$$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}} \qquad (q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$$

$$o(q_{\text{even}}) = 1 \qquad o(q_{\text{odd}}) = 0$$



Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$a \mapsto q_{\text{odd}}$

$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}}$

$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}}$

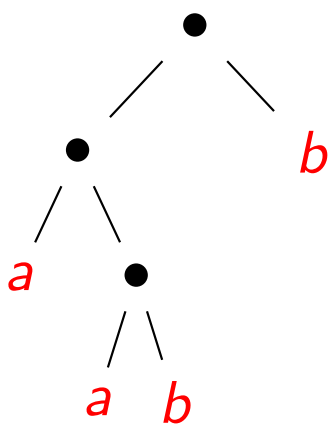
$o(q_{\text{even}}) = 1$

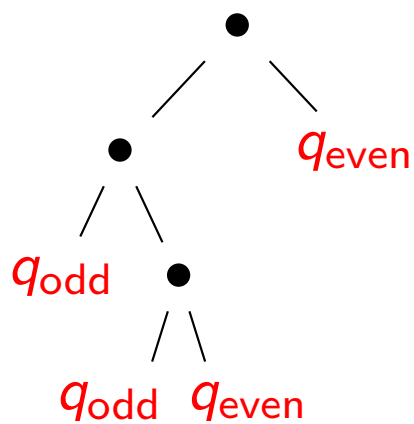
$b \mapsto q_{\text{even}}$

$(q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$

$(q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$

$o(q_{\text{odd}}) = 0$





Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$a \mapsto q_{\text{odd}}$

$b \mapsto q_{\text{even}}$

$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}}$

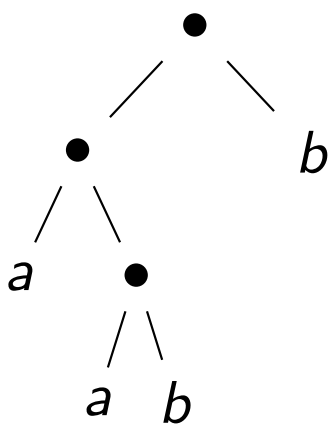
$(q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$

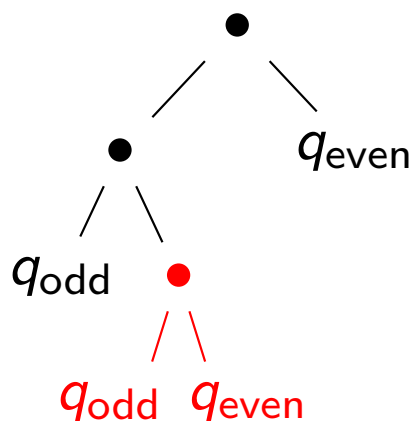
$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}}$

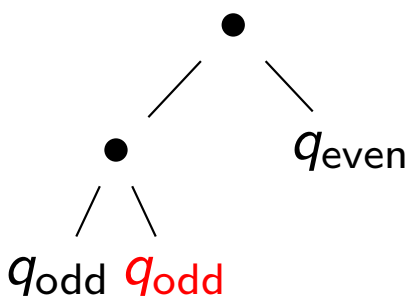
$(q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$

$o(q_{\text{even}}) = 1$

$o(q_{\text{odd}}) = 0$







Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

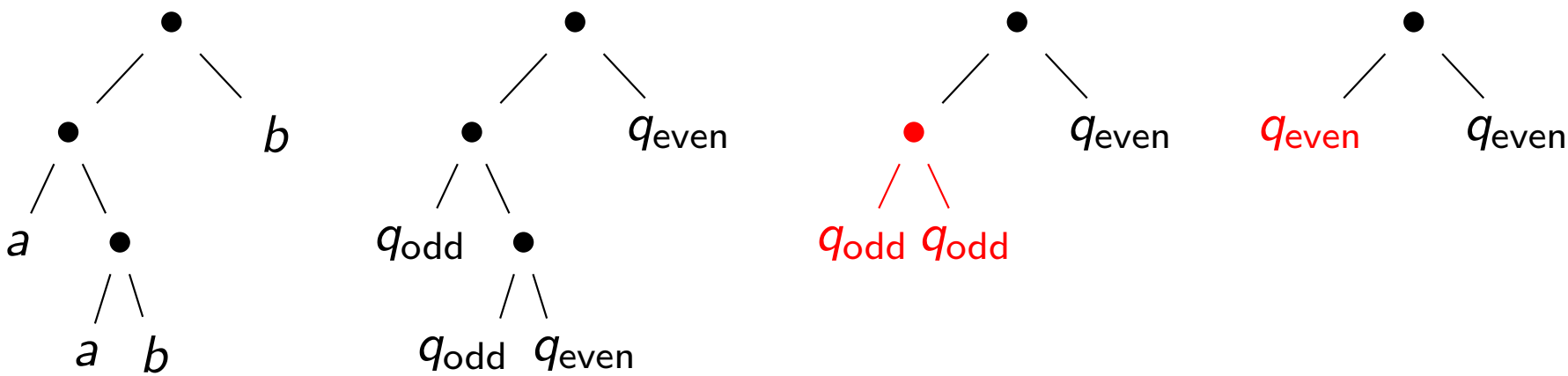
$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$$a \mapsto q_{\text{odd}} \qquad b \mapsto q_{\text{even}}$$

$$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}} \qquad (q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$$

$$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}} \qquad (q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$$

$$o(q_{\text{even}}) = 1 \qquad o(q_{\text{odd}}) = 0$$



Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

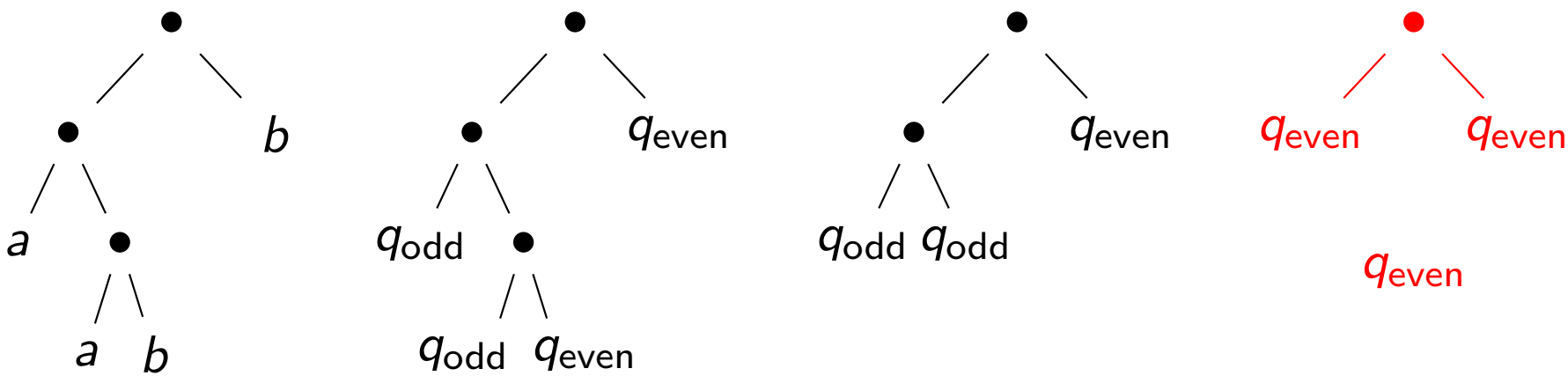
$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$a \mapsto q_{\text{odd}}$ $\quad$ $b \mapsto q_{\text{even}}$

$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}}$ $\quad$ $(q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$

$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}}$ $\quad$ $(q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$

$o(q_{\text{even}}) = 1$ $\quad$ $o(q_{\text{odd}}) = 0$



Tree automaton example

$I = \{a, b\}$, $FX = X \times X$

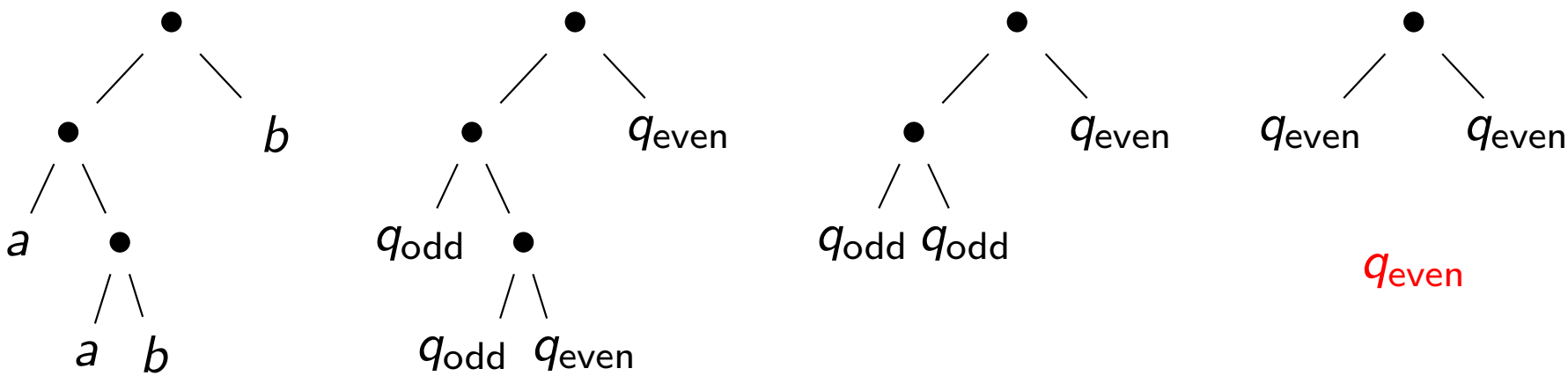
$Q = \{q_{\text{even}}, q_{\text{odd}}\}$

$$a \mapsto q_{\text{odd}} \qquad b \mapsto q_{\text{even}}$$

$$(q_{\text{even}}, q_{\text{even}}) \mapsto q_{\text{even}} \qquad (q_{\text{even}}, q_{\text{odd}}) \mapsto q_{\text{odd}}$$

$$(q_{\text{odd}}, q_{\text{even}}) \mapsto q_{\text{odd}} \qquad (q_{\text{odd}}, q_{\text{odd}}) \mapsto q_{\text{even}}$$

$o(q_{\text{even}}) = 1$ $\qquad$ $o(q_{\text{odd}}) = 0$



Contexts

Contexts over F: $F^*(I + \{\Box\})$

$\Box$ placeholder to plug in trees

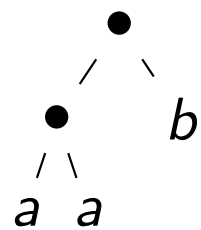
Contexts

Contexts over F: $F^*(I + \{\Box\})$

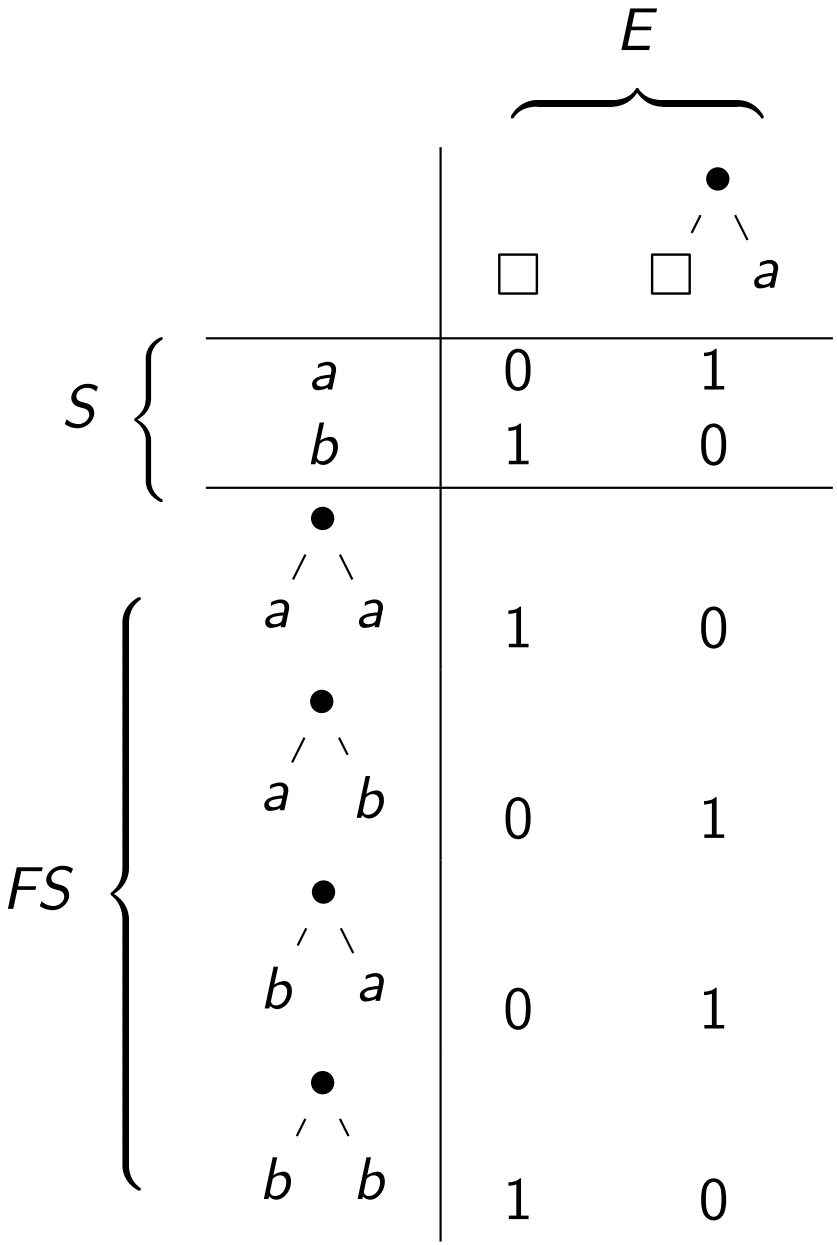
$\Box$ placeholder to plug in trees

Example: plugging

gives



Observation table

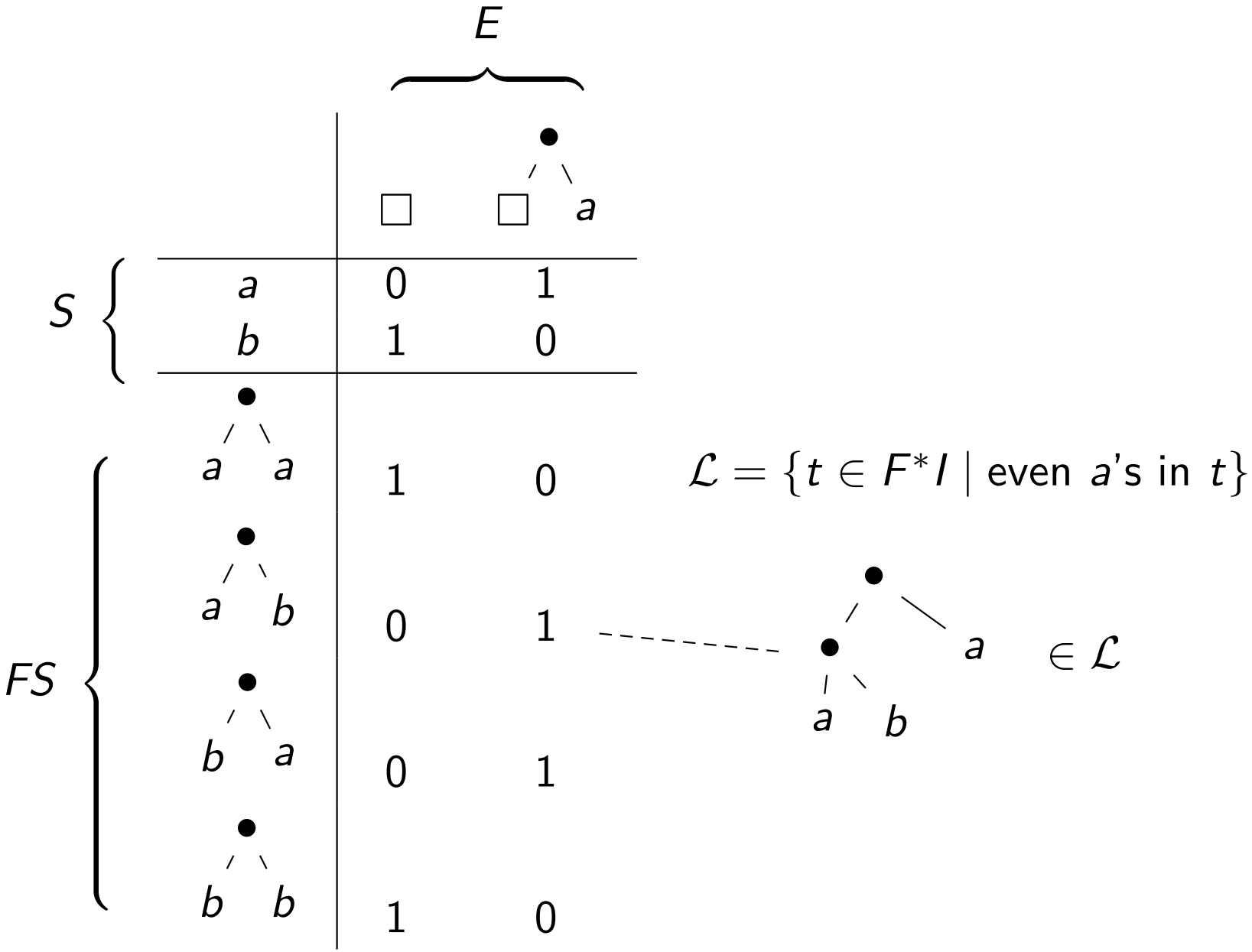


Observation table

		E: $\square$	E: $\square$ a
S	a	0	1
	b	1	0
FS	a a	1	0
	a b	0	1
	b a	0	1
	b b	1	0

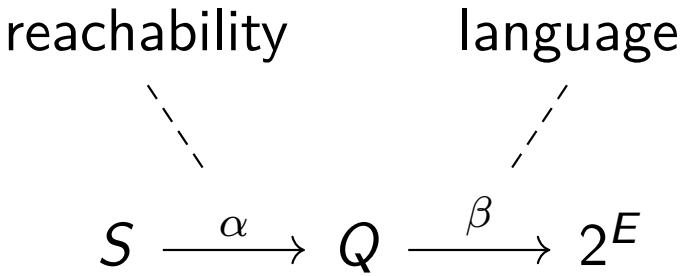
$$\mathcal{L} = \{t \in F^* I \mid \text{even } a\text{'s in } t\}$$

Observation table



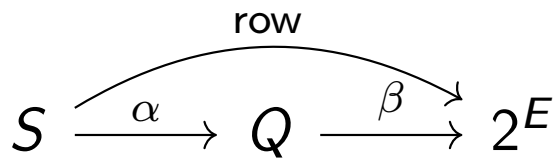
$\mathrm{L}^\star$ observation table, abstractly: wrapper

$S, E \subseteq A^*$



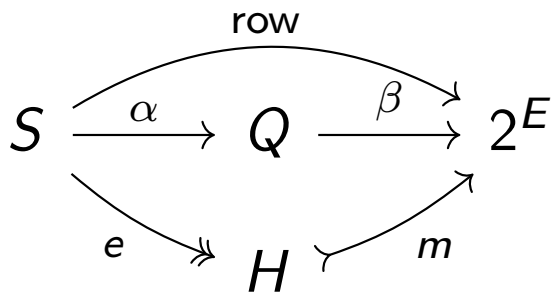
$\mathrm{L}^\star$ observation table, abstractly: wrapper

$S, E \subseteq A^*$



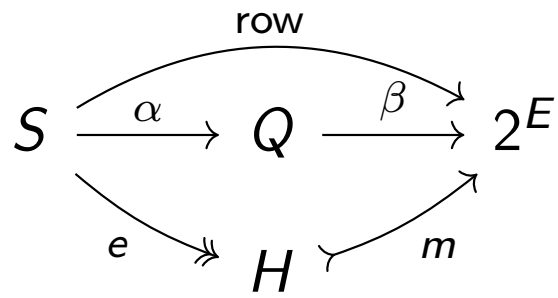
$\mathrm{L}^\star$ observation table, abstractly: wrapper

$S, E \subseteq A^*$



$\mathrm{L}^\star$ observation table, abstractly: wrapper

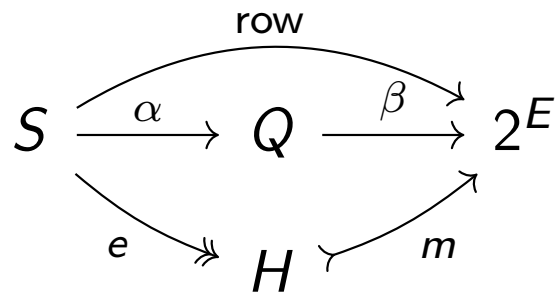
$S, E \subseteq A^*$

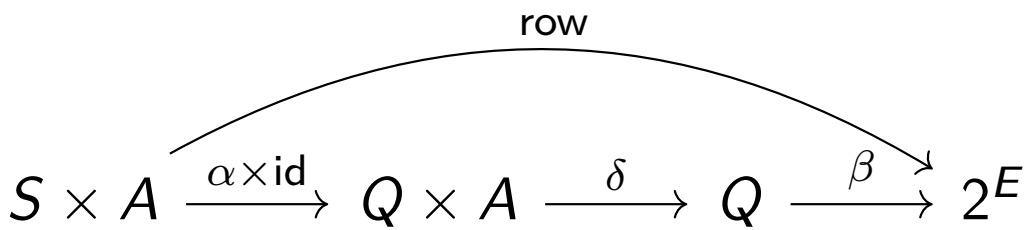


$$S \times A \xrightarrow{\alpha \times \mathsf{id}} Q \times A \xrightarrow{\delta} Q \xrightarrow{\beta} 2^E$$

$\mathrm{L}^\star$ observation table, abstractly: wrapper

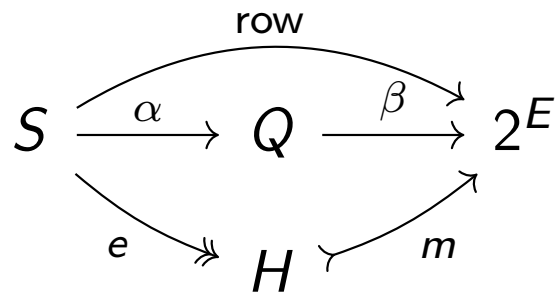
$S, E \subseteq A^*$

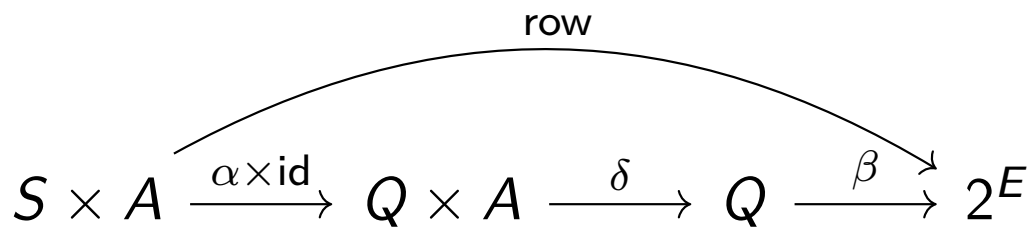


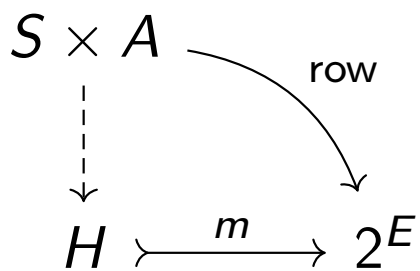


$\mathrm{L}^\star$ observation table, abstractly: wrapper

$S, E \subseteq A^*$



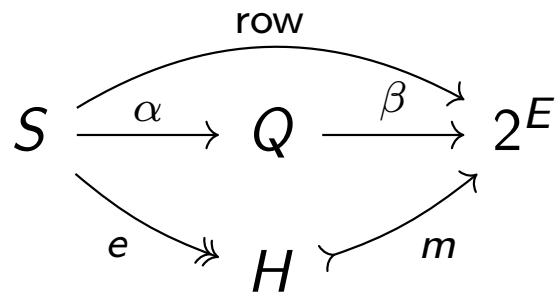


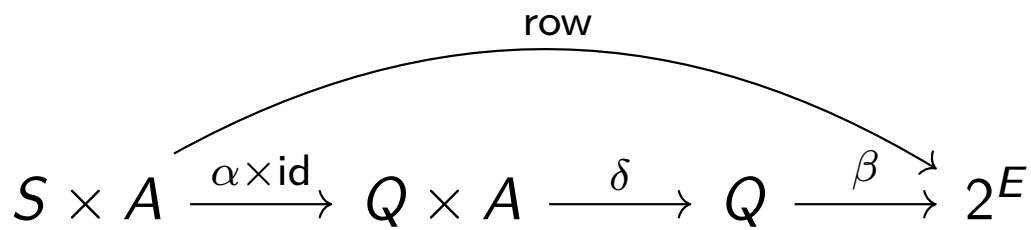


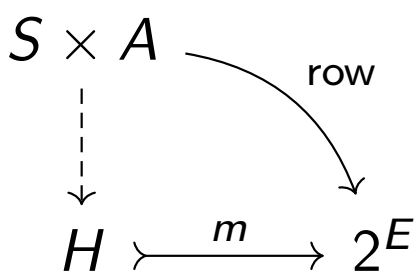
closedness

$\mathrm{L}^\star$ observation table, abstractly: wrapper

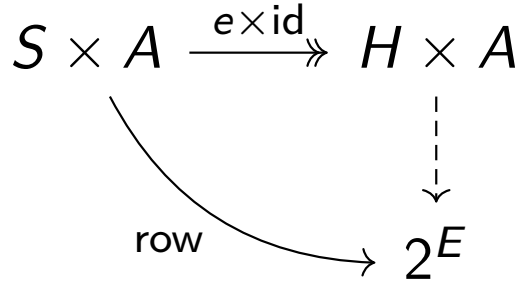
$S, E \subseteq A^*$





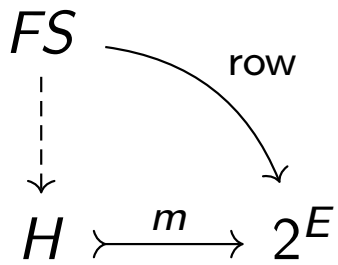


closedness



consistency

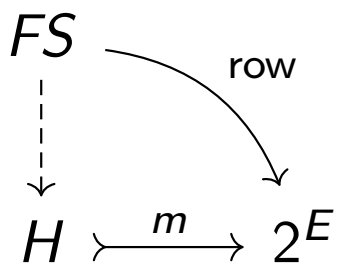
Closedness and consistency for tree automata



closedness:

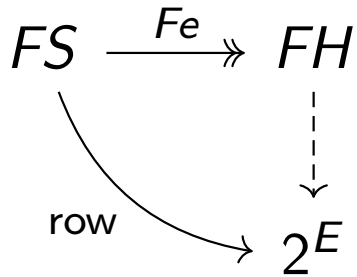
$\forall x \in FS\, \exists s \in S.$
$\mathrm{row}(s) = \mathrm{row}(x)$

Closedness and consistency for tree automata



closedness:

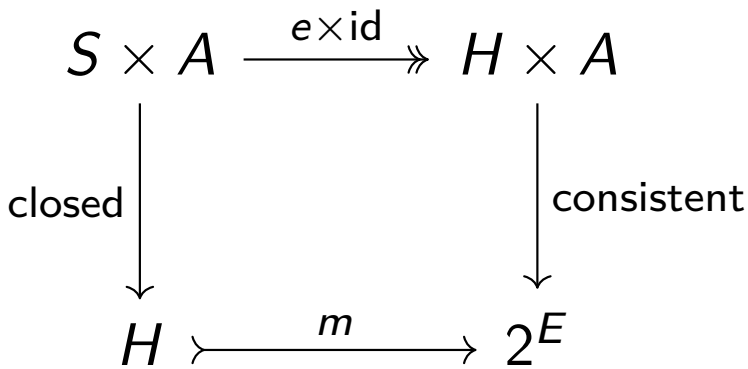
$$\forall x \in FS\, \exists s \in S.\ \mathrm{row}(s) = \mathrm{row}(x)$$



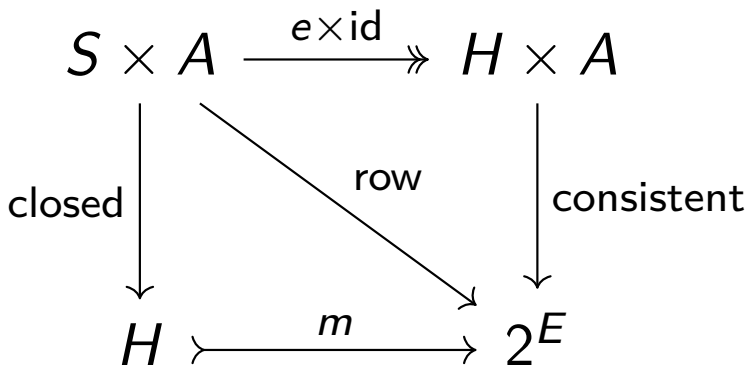
consistency:

$$\forall x \in F(S \times S).\ F(\mathrm{row} \circ \pi_1)(x) = F(\mathrm{row} \circ \pi_2)(x) \implies (\mathrm{row} \circ F\pi_1)(x) = (\mathrm{row} \circ F\pi_2)(x)$$

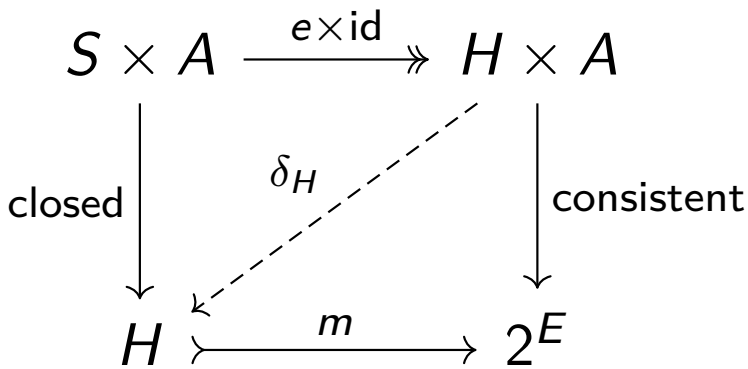
Hypothesis, abstractly



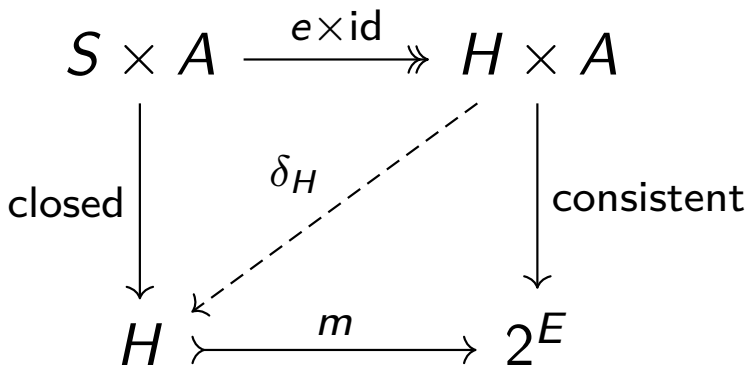
Hypothesis, abstractly



Hypothesis, abstractly

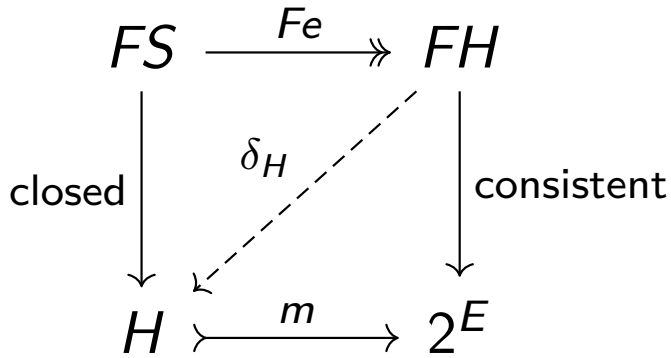


Hypothesis, abstractly



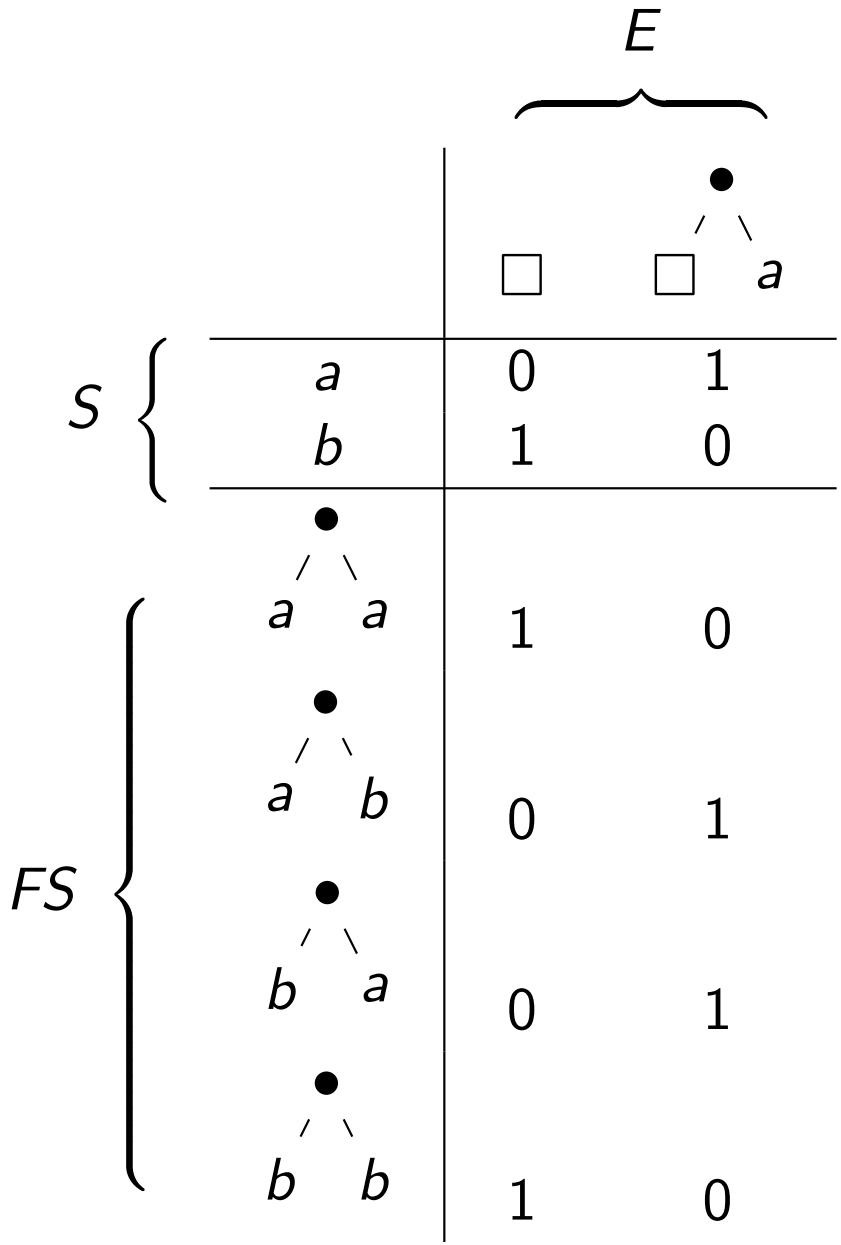
$$\delta_H(\mathrm{row}(s), a) = \mathrm{row}(sa)$$

Hypothesis for tree automata

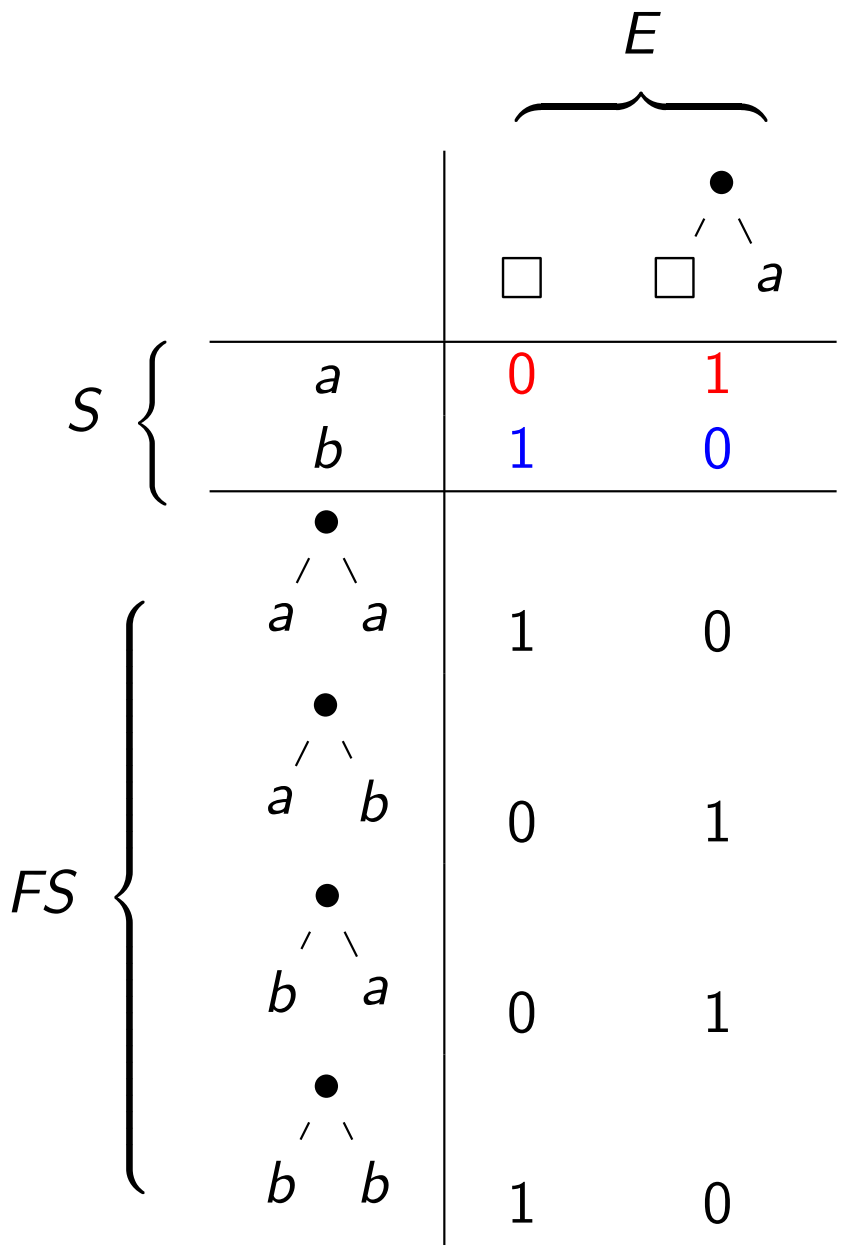


$$\delta_H(Fe(x)) = \mathsf{row}(x)$$

Hypothesis example



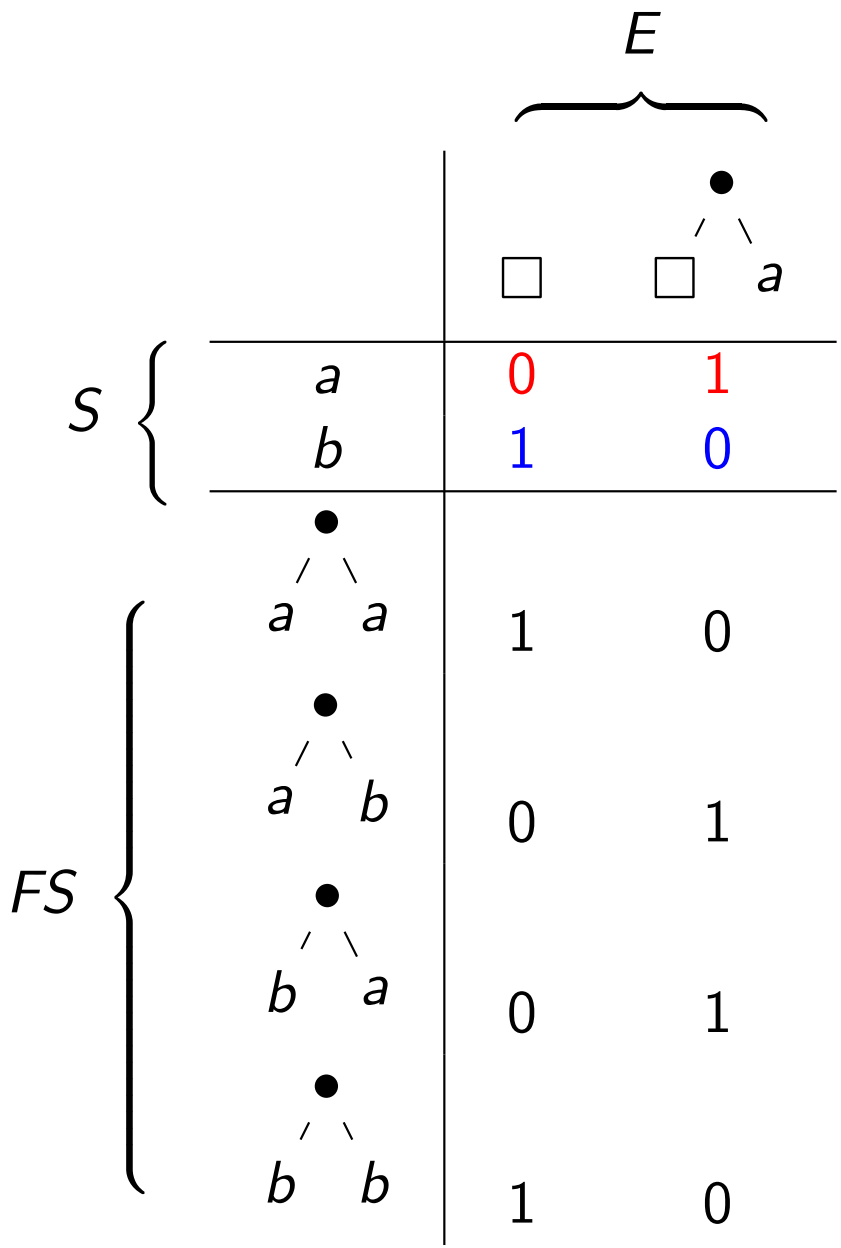
Hypothesis example



$a \mapsto q_0$

$b \mapsto q_1$

Hypothesis example

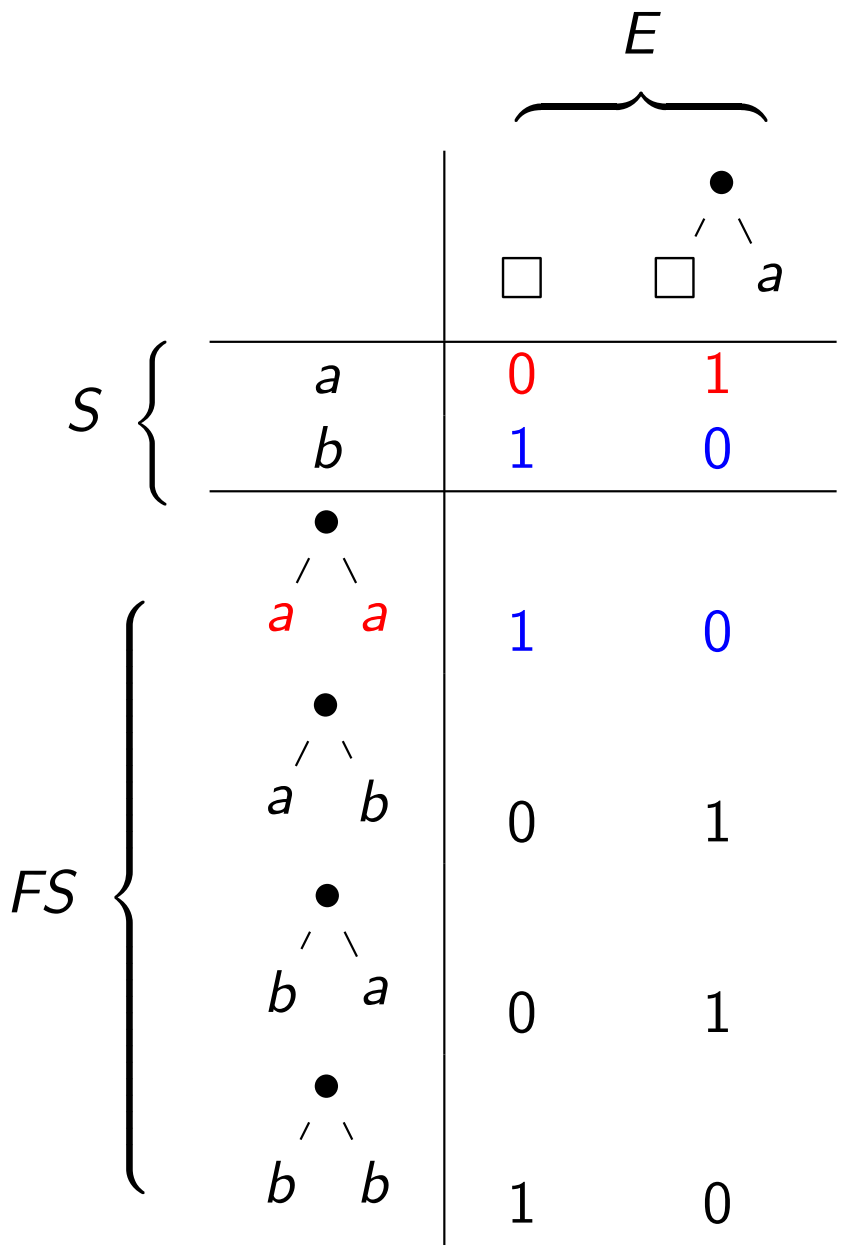


$a \mapsto q_0$

$b \mapsto q_1$

$o(q_0) = 0, o(q_1) = 1$

Hypothesis example



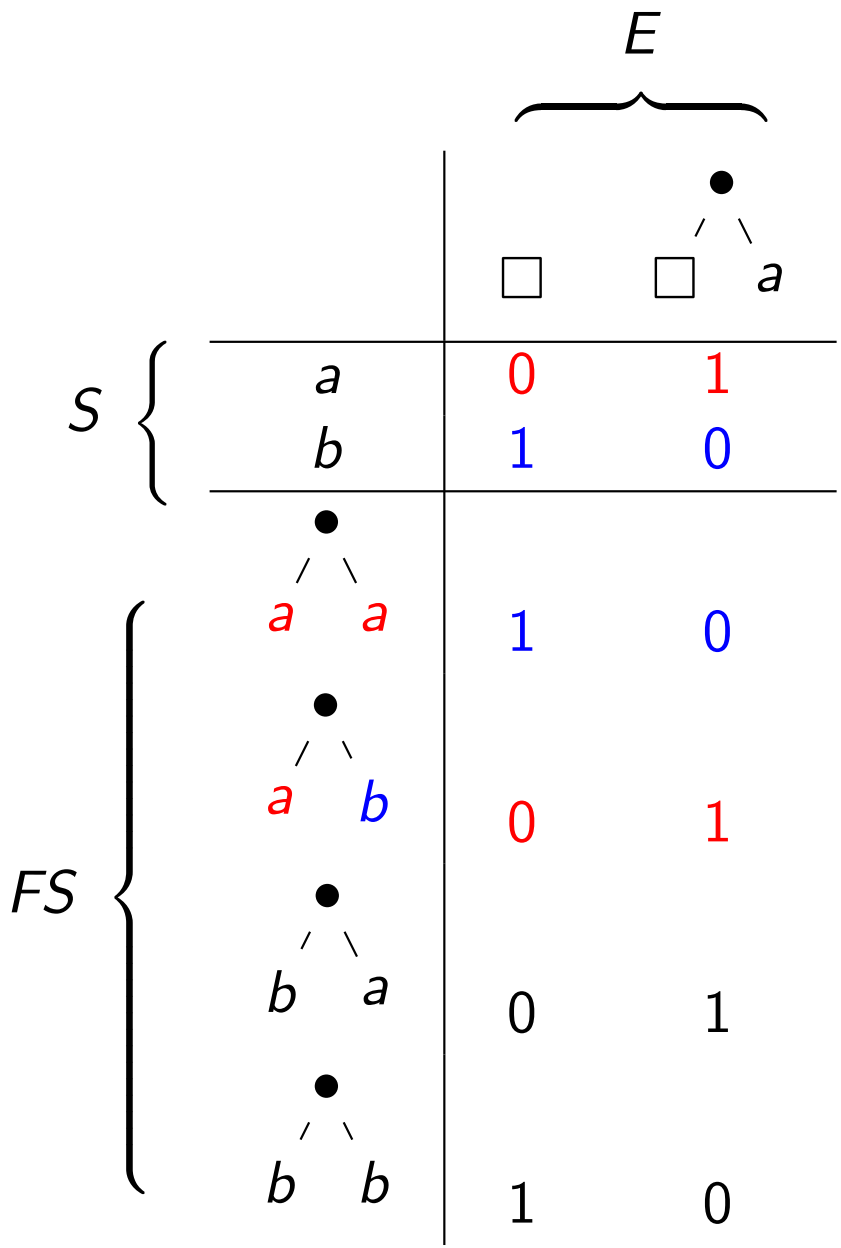
$a \mapsto q_0$

$b \mapsto q_1$

$o(q_0) = 0, o(q_1) = 1$

$(q_0, q_0) \mapsto q_1$

Hypothesis example



$a \mapsto q_0$

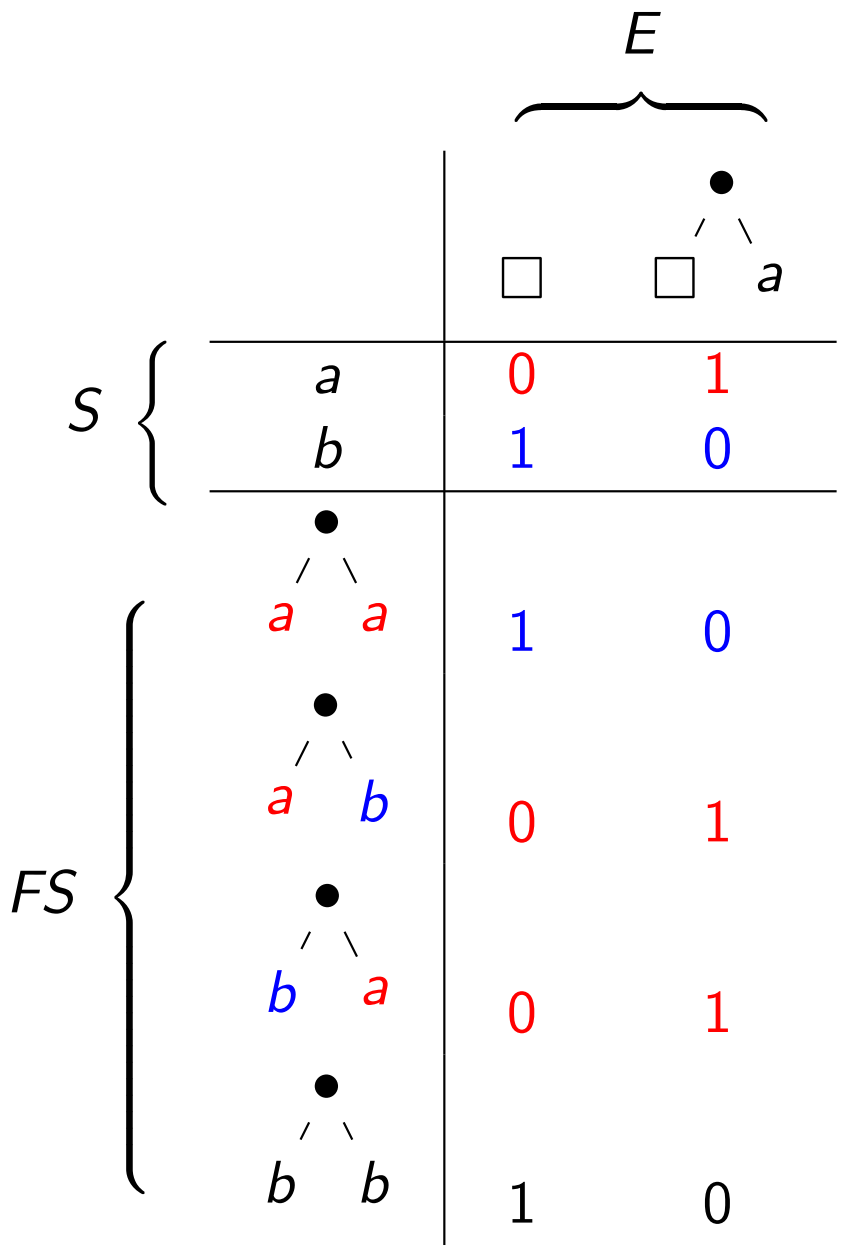
$b \mapsto q_1$

$o(q_0) = 0, o(q_1) = 1$

$(q_0, q_0) \mapsto q_1$

$(q_0, q_1) \mapsto q_0$

Hypothesis example



$a \mapsto q_0$

$b \mapsto q_1$

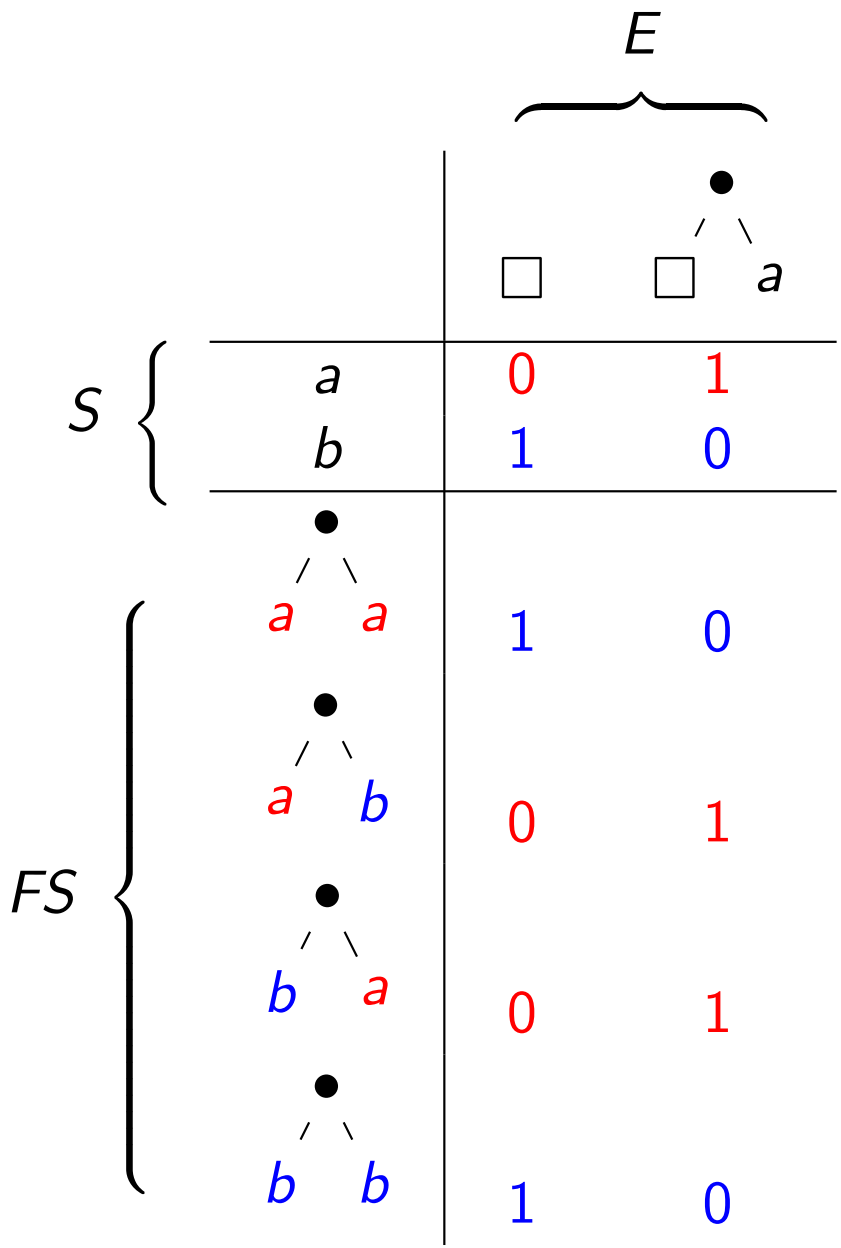
$o(q_0) = 0, o(q_1) = 1$

$(q_0, q_0) \mapsto q_1$

$(q_0, q_1) \mapsto q_0$

$(q_1, q_0) \mapsto q_0$

Hypothesis example



$a \mapsto q_0$

$b \mapsto q_1$

$o(q_0) = 0, o(q_1) = 1$

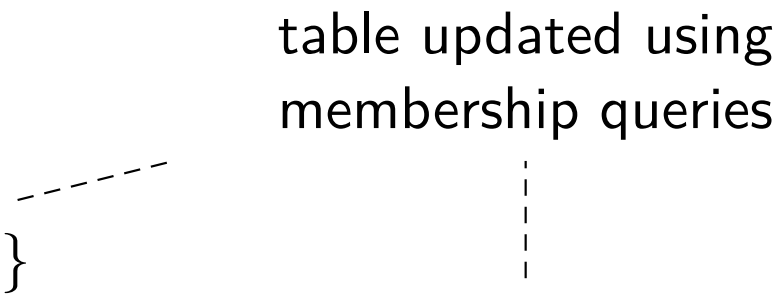
$(q_0, q_0) \mapsto q_1$

$(q_0, q_1) \mapsto q_0$

$(q_1, q_0) \mapsto q_0$

$(q_1, q_1) \mapsto q_1$

Algorithm overview



1. Initialise $S = I$, $E = \{\square\}$
2. Satisfy closedness and consistency (by augmenting S and E)
3. Construct hypothesis
4. Pose equivalence query
5. On a counterexample, add its subtrees to S and repeat from 2

Contributions

Abstract version of $\mathbb{L}^\star$

- On any category satisfying some (mild) conditions
- "Local" closedness/consistency
- Counterexamples as recursive coalgebras
  - Coalgebras with a universal coalgebra-to-algebra hom. property
- Termination proof

Instantiation to learn **generalised** tree automata in **Set**

Abstract algorithm template

Algorithm 1 Abstract automata learning algorithm

$\alpha, \beta \leftarrow \textsc{Fix}(! : 0 \to Q, ! : Q \to 1)$
while $\mathrm{EQ}(\mathcal{H}_{(\alpha,\beta)}) = \rho : S \to I + FS$ **do**
    update α to incorporate ρ
    $\alpha, \beta \leftarrow \textsc{Fix}(\alpha, \beta)$
return $\mathcal{H}_{(\alpha,\beta)}$

Algorithm 2 Make wrapper closed and consistent

function $\textsc{Fix}(\alpha, \beta)$
    while (α, β) not closed or not consistent **do**
        if (α, β) not closed **then**
            update α to be locally closed w.r.t. previous α
        else if (α, β) not consistent **then**
            update β to be locally consistent w.r.t. previous β
    return α, β

Future directions

Generalise monad F^* to arbitrary monad

Tree automata in other categories

Regular ω-tree languages