Learning Discrete Timed Automata

Kush Grover

Chennai Mathematical Institute

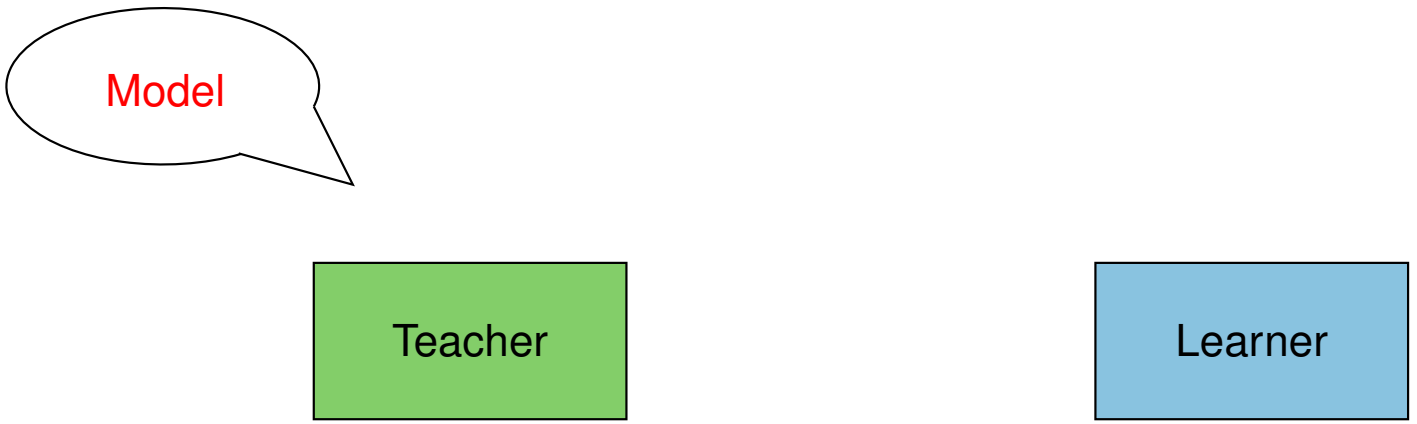
July 23, 2019

Schedule

Learning

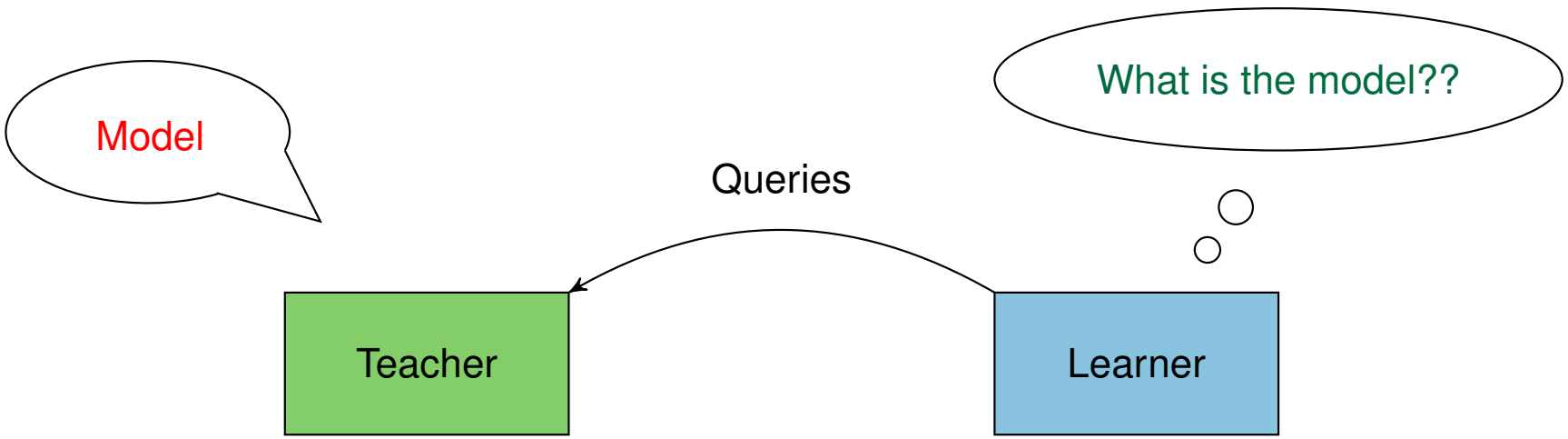
Learning Problem

Learning Problem

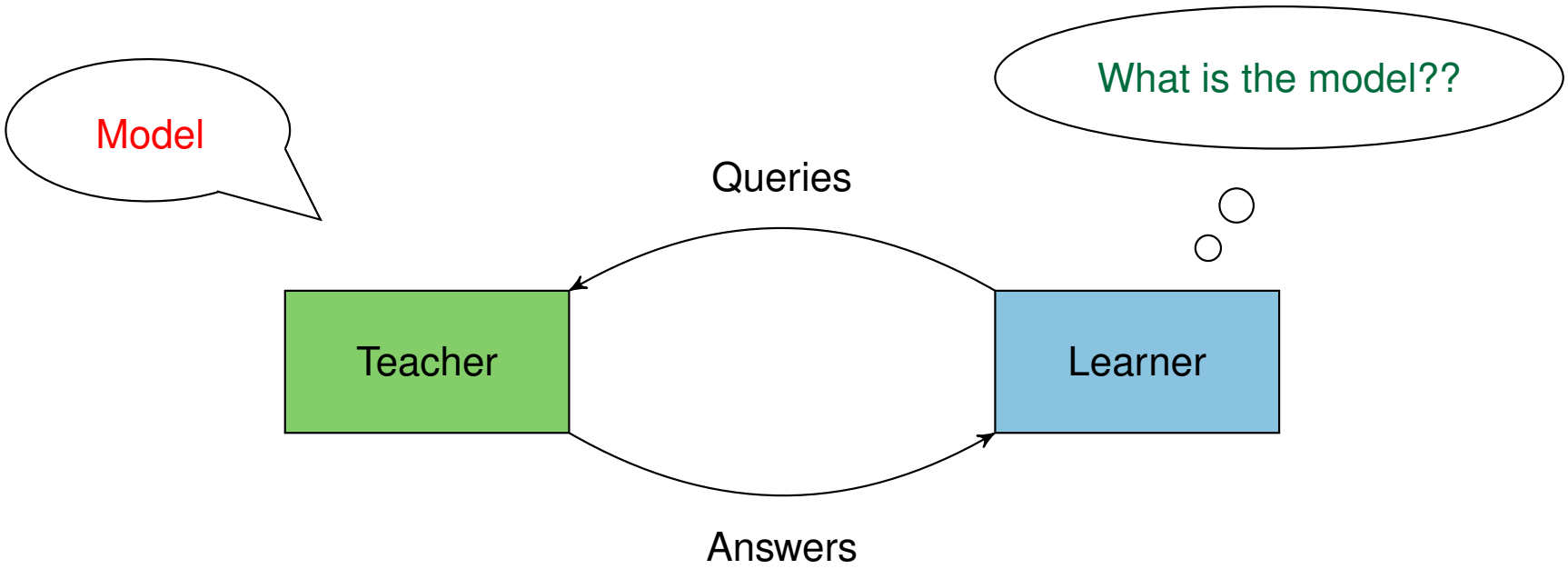


Learning Problem

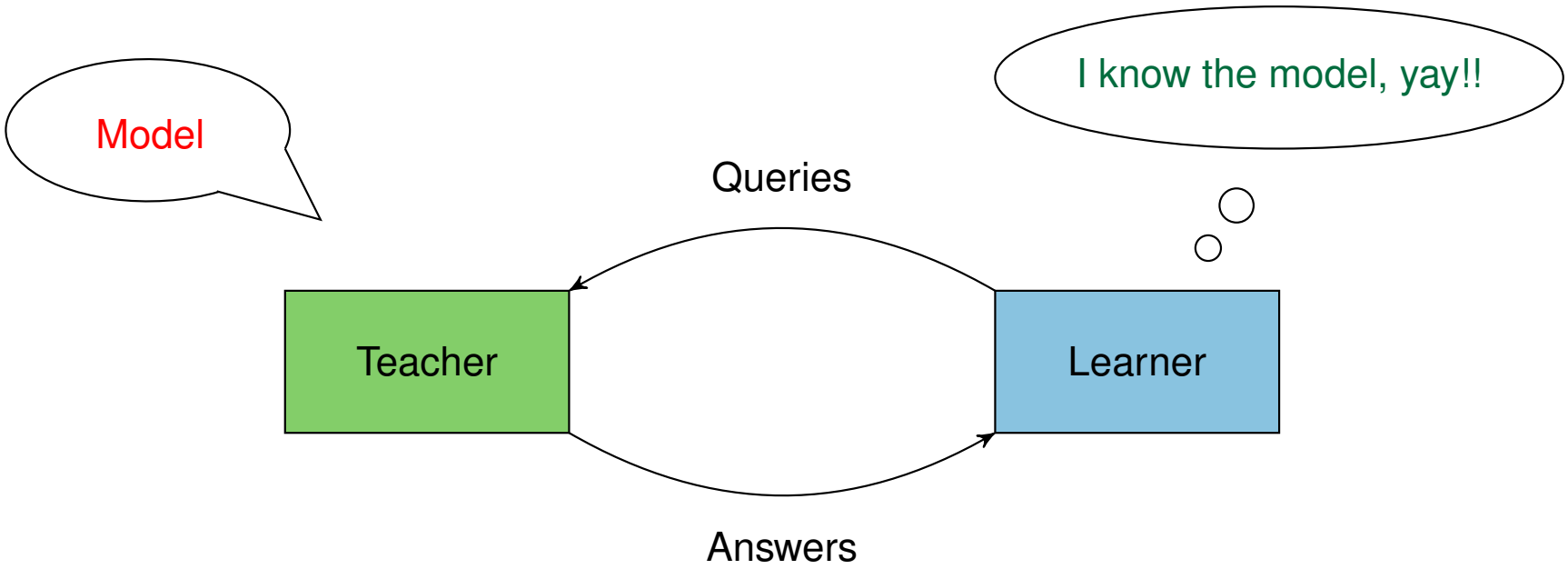
Learning Problem



Learning Problem



Learning Problem

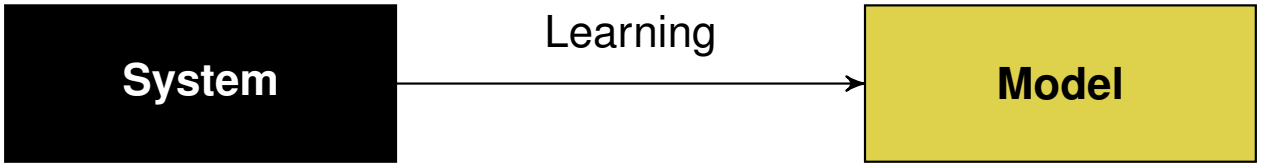


Motivation

System

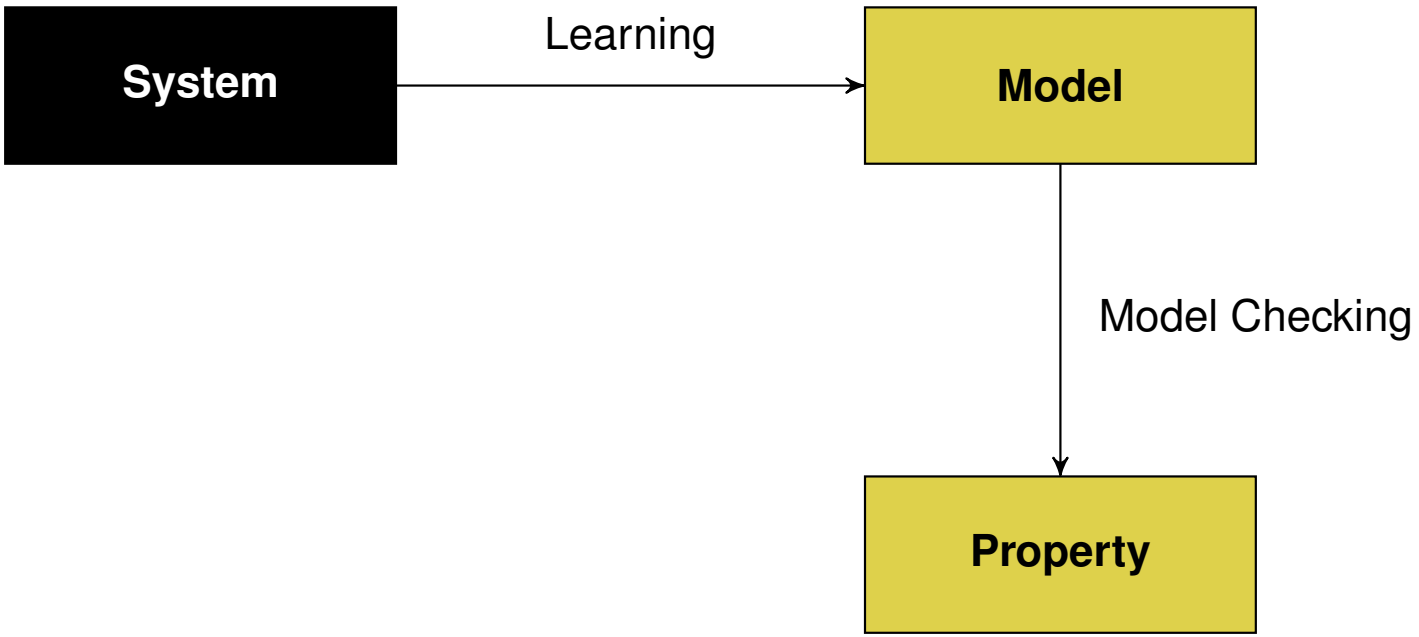
[1] Peled D., Vardi M.Y., Yannakakis M. (1999) Black Box Checking. Formal Methods for Protocol Engineering and Distributed Systems. PSTV 1999, FORTE 1999.

Motivation



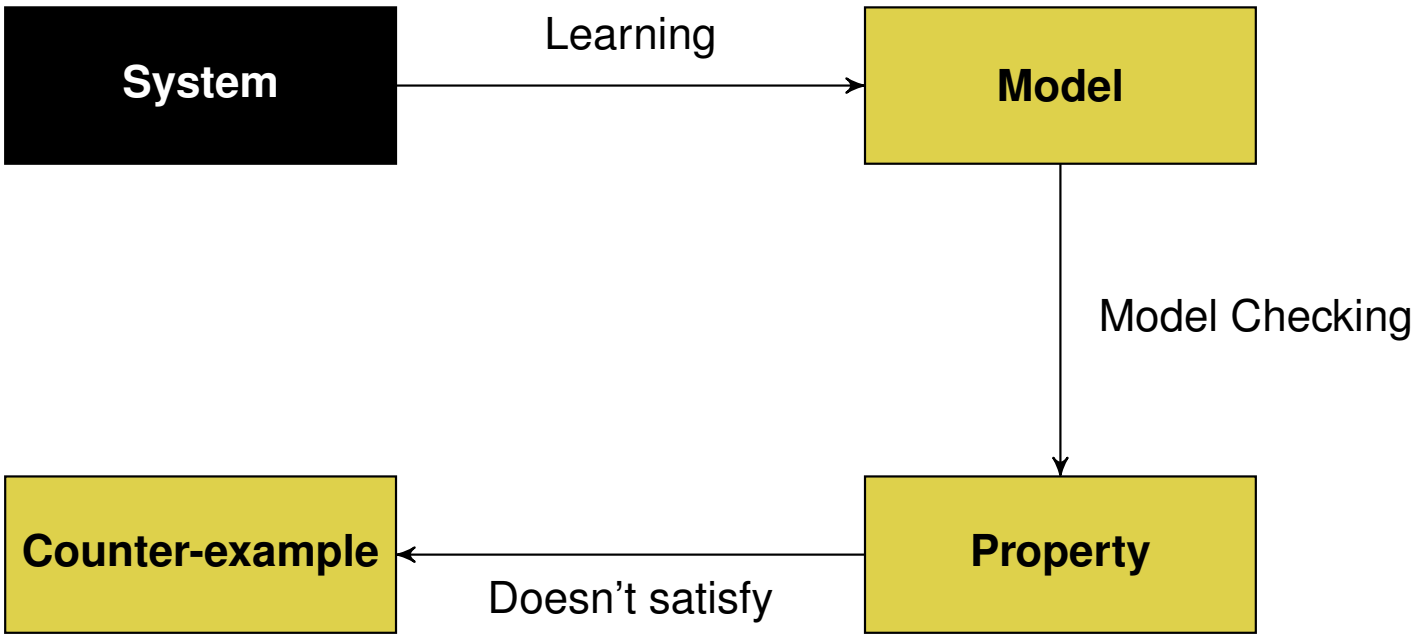
[1] Peled D., Vardi M.Y., Yannakakis M. (1999) Black Box Checking. Formal Methods for Protocol Engineering and Distributed Systems. PSTV 1999, FORTE 1999.

Motivation



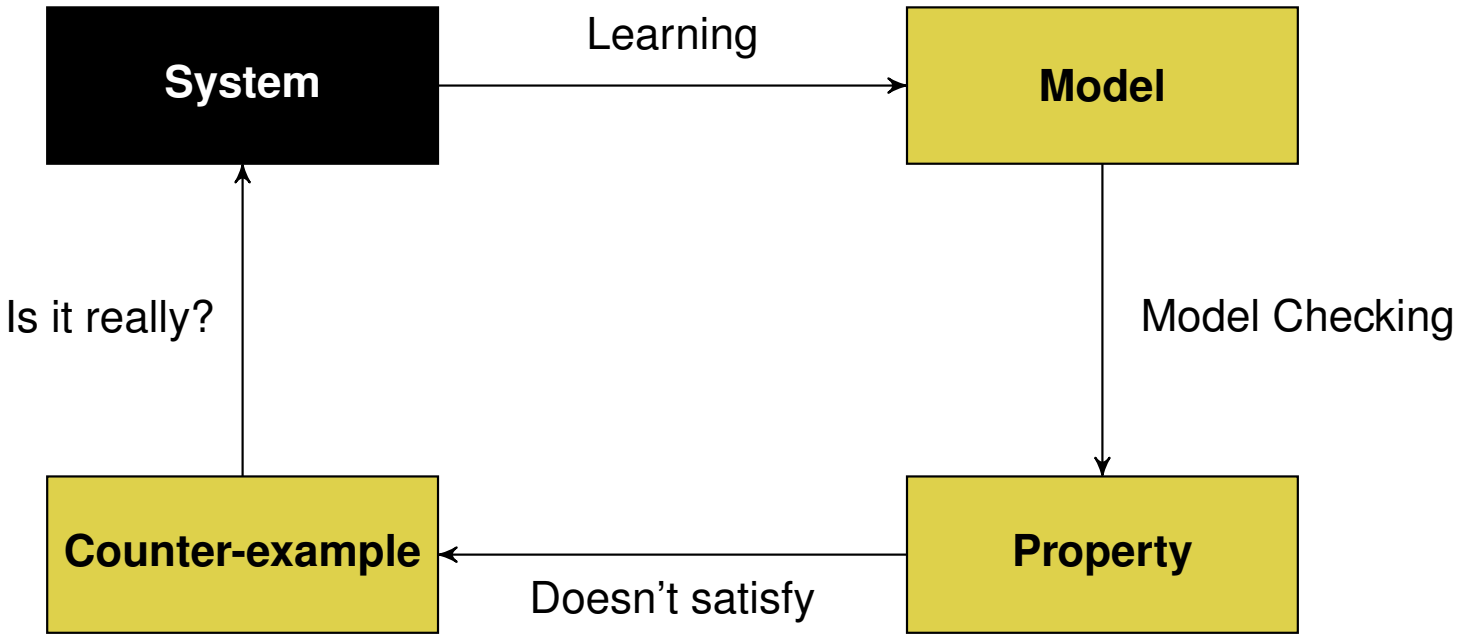
[1] Peled D., Vardi M.Y., Yannakakis M. (1999) Black Box Checking. Formal Methods for Protocol Engineering and Distributed Systems. PSTV 1999, FORTE 1999.

Motivation



[1] Peled D., Vardi M.Y., Yannakakis M. (1999) Black Box Checking. Formal Methods for Protocol Engineering and Distributed Systems. PSTV 1999, FORTE 1999.

Motivation



[1] Peled D., Vardi M.Y., Yannakakis M. (1999) Black Box Checking. Formal Methods for Protocol Engineering and Distributed Systems. PSTV 1999, FORTE 1999.

Types of learning

Two types of learning:

Active Learning

Passive Learning

Types of learning

Two types of learning:

Active Learning

Learner asks if some instance satisfies the model on the fly.

Passive Learning

Sets $\mathcal{P}$ and $\mathcal{Q}$ of positive and negative instances are known to the learner.

Automata Learning

Learning DFA

$\mathbf{L}^*$: an active learning algorithm by D. Angluin[1987].

Learning DFA

L*: an active learning algorithm by D. Angluin[1987].

Two types of queries learner can ask:

Learning DFA

L*: an active learning algorithm by D. Angluin[1987].

Two types of queries learner can ask:

- **Membership queries**: ask if some string is accepted by the target automaton.

Learning DFA

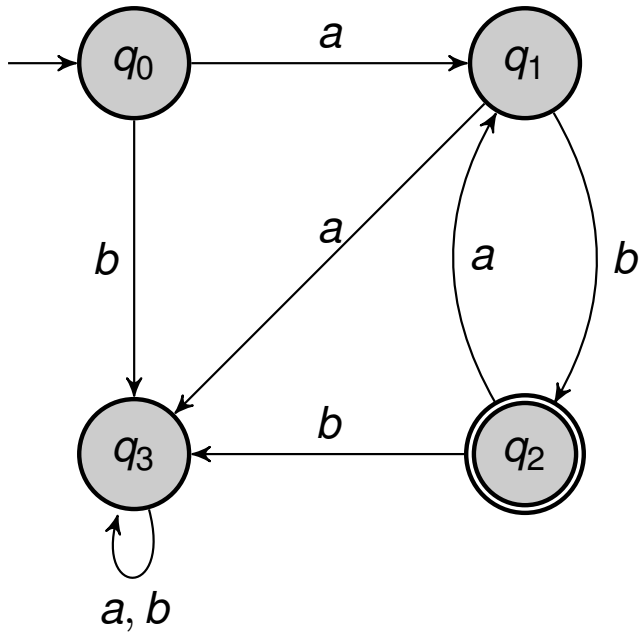
L*: an active learning algorithm by D. Angluin[1987].

Two types of queries learner can ask:

- **Membership queries**: ask if some string is accepted by the target automaton.
- **Equivalence queries**: give a hypothesis automaton to the teacher and ask if it is equal to the target automaton. If not, teacher returns a counter-example.

Learning DFAs

$$\mathcal{L}(D) = (ab)^+$$

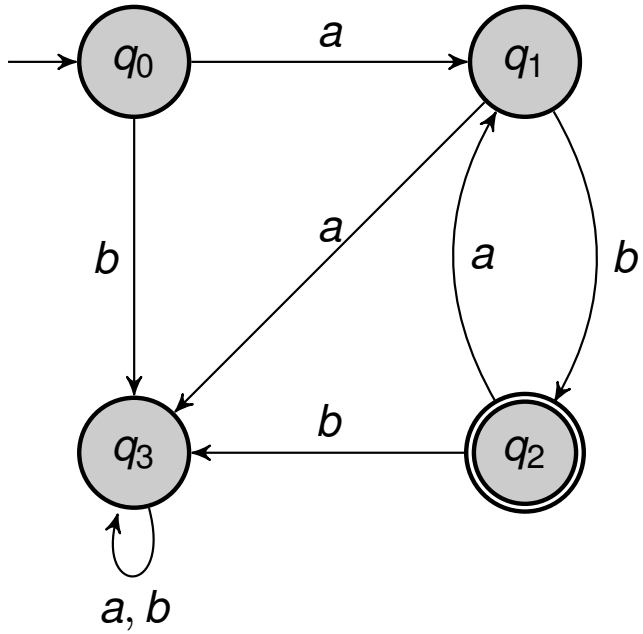


Deterministic Finite Automaton D

Learning DFAs

$\mathcal{L}(D) = (ab)^+$

Run L^* on this.

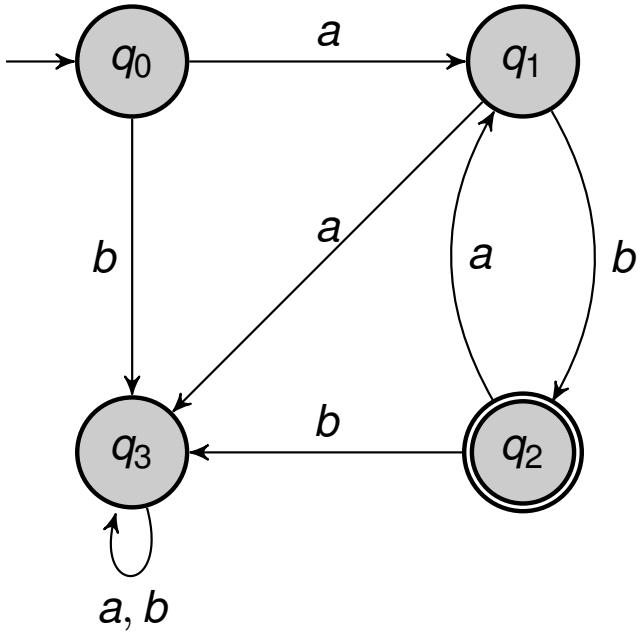


Learning DFAs

Observation Table	
T	E
S	$-/+$
R	$-/+$

$T = (\Sigma, S, R, E, f)$

- Σ is a finite alphabet,
- $S, R, E \subseteq \Sigma^*$ and $R = S \cdot \Sigma$,
- $f(u) = +$ if $u \in \mathcal{L}(D)$ and $f(u) = -$ otherwise.

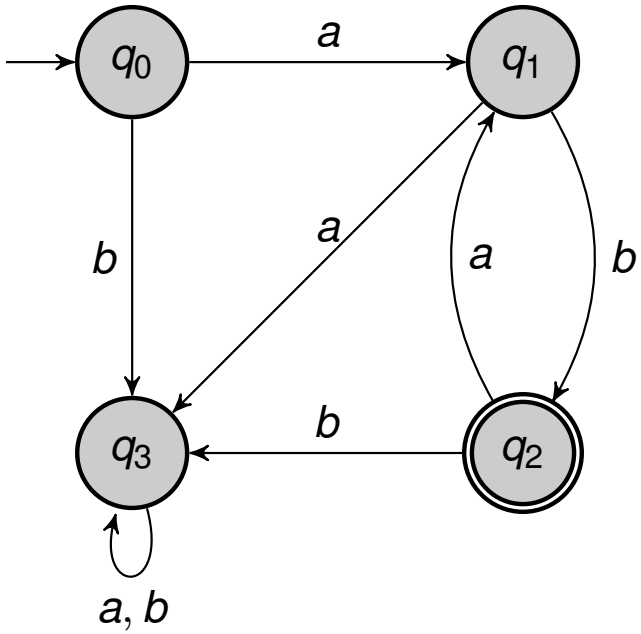


Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–

Initially

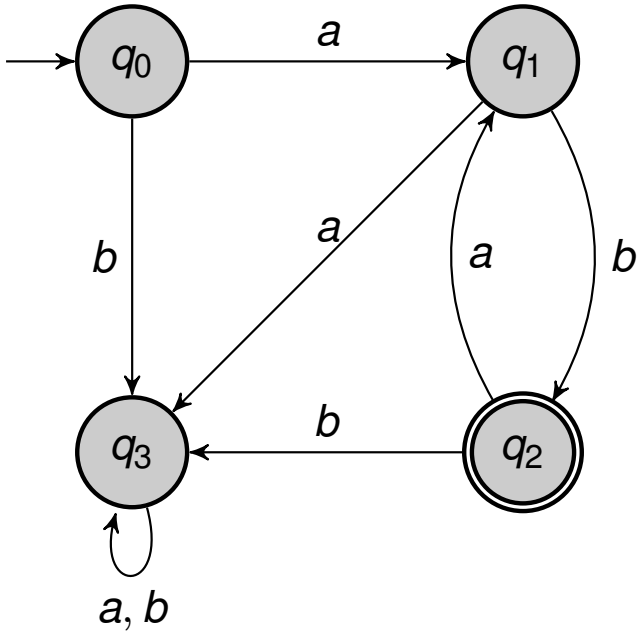
- $S = E = \{\epsilon\}$
- $R = \{a, b\}$
- $f(\epsilon) = f(a) = f(b) = -$



Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–

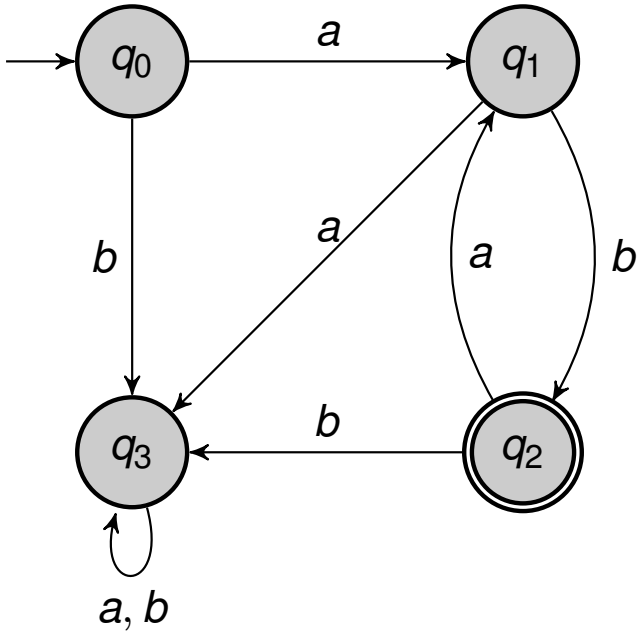
Check if it is { ***closed*** ***consistent***



Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–

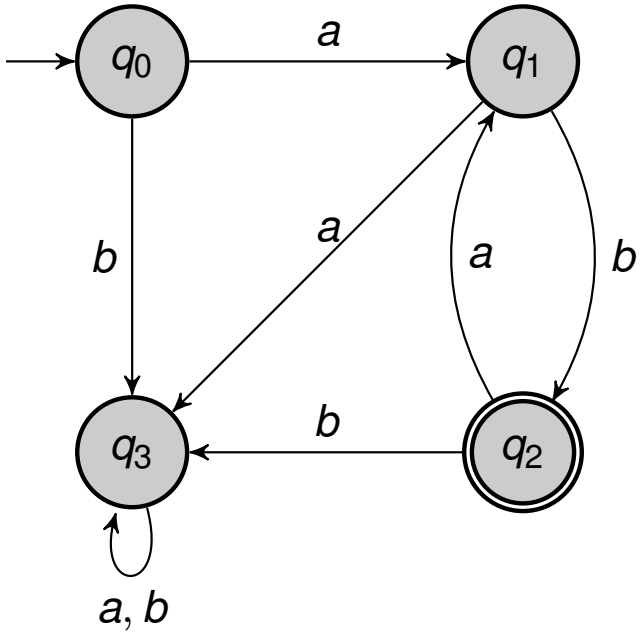
Check if it is { ***closed*** ✓ ***consistent***



Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–

Check if it is { ***closed*** ✓ ***consistent*** ✓

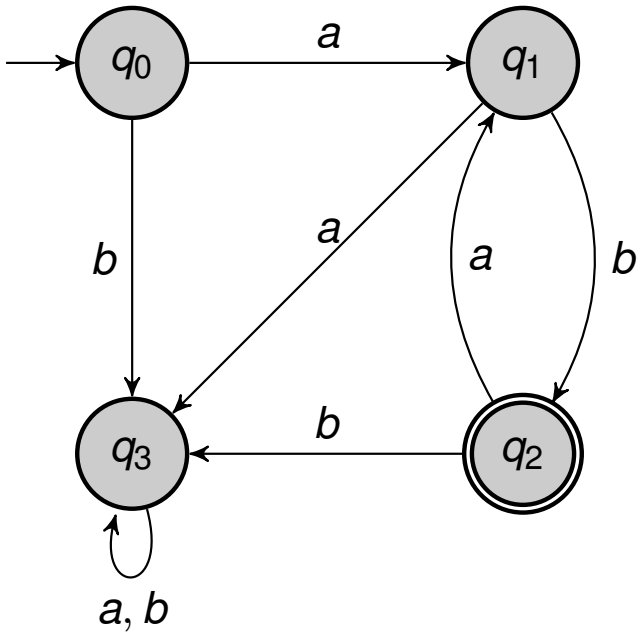


Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–

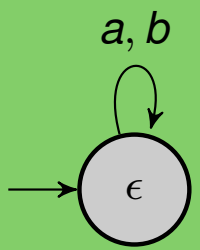
Construct a hypothesis automaton $M(T)$ from the table.

- S is the set of states
- R represents the transitions
- ϵ is the initial state
- $s \in S$ is final if first entry of $row(s)$ is $+$

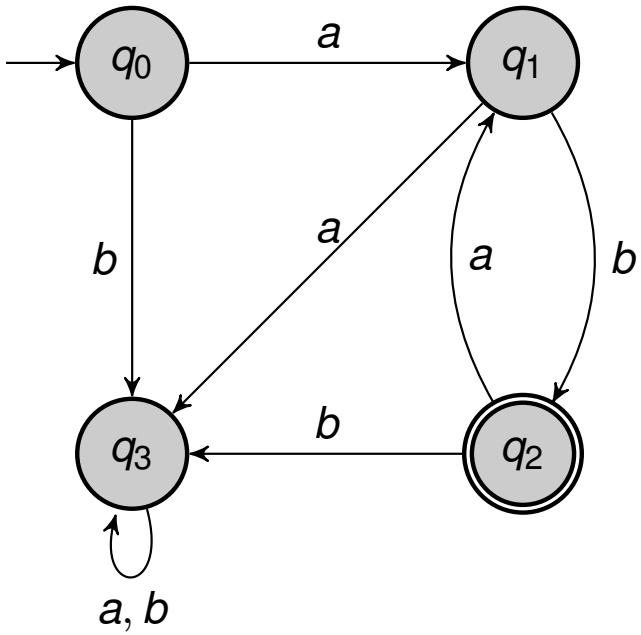


Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–

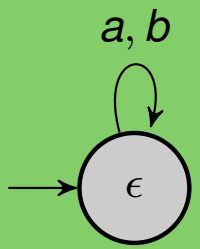


Hypothesis automaton $H_0 = M(T_0)$



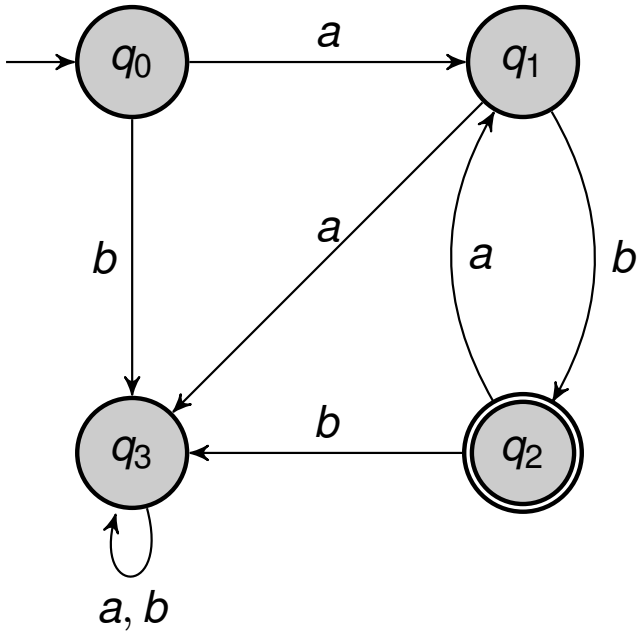
Learning DFAs

Observation Table	
T_0	ϵ
ϵ	–
a	–
b	–



Hypothesis automaton $H_0 = M(T_0)$

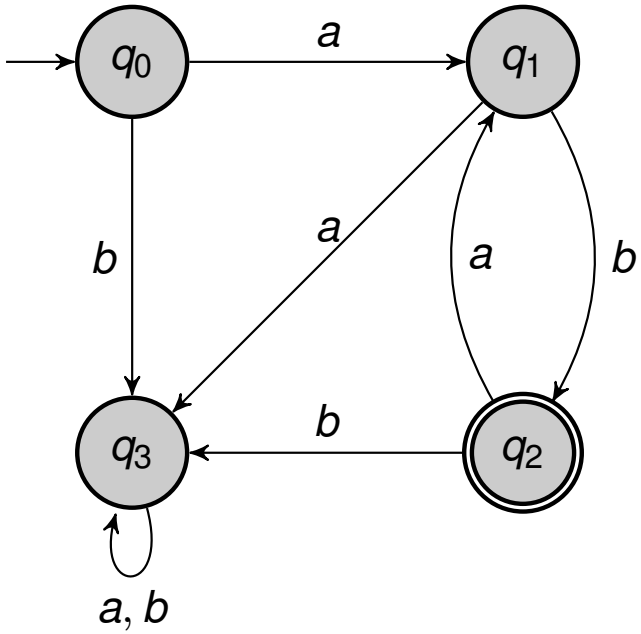
✗



Learning DFAs

Teacher returns a counter-example

$w = ab$

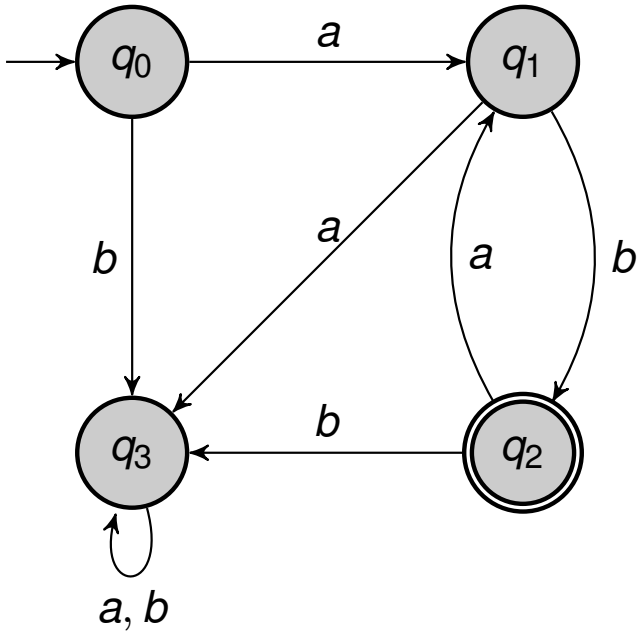


Learning DFAs

Teacher returns a counter-example

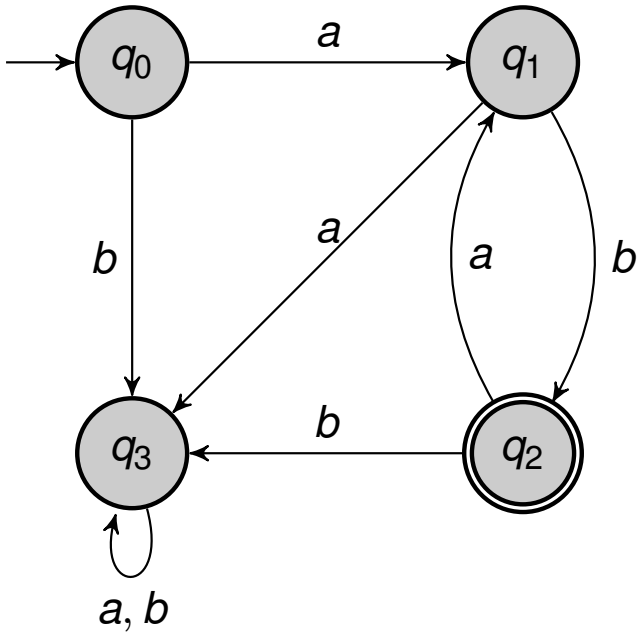
$w = ab$

Add all prefixes of w to S.



Learning DFAs

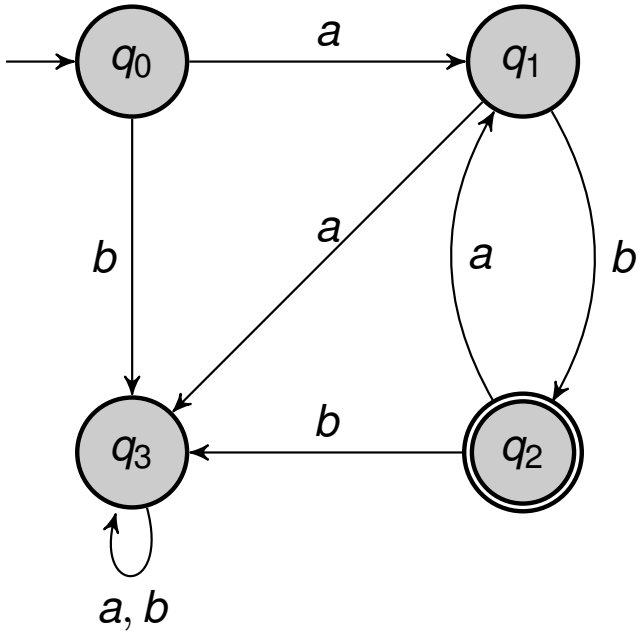
Observation Table	
T_1	ϵ
ϵ	$-$
a	$-$
ab	$+$
b	$-$
aa	$-$
aba	$-$
abb	$-$



Learning DFAs

Observation Table	
T_1	ϵ
ϵ	$-$
a	$-$
ab	$+$
b	$-$
aa	$-$
aba	$-$
abb	$-$

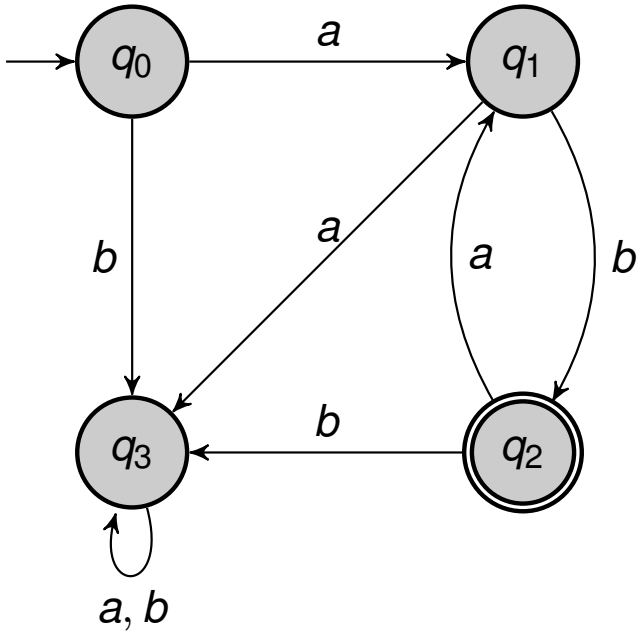
Check if it is { ***closed*** ***consistent***



Learning DFAs

Observation Table	
T_1	ϵ
ϵ	$-$
a	$-$
ab	$+$
b	$-$
aa	$-$
aba	$-$
abb	$-$

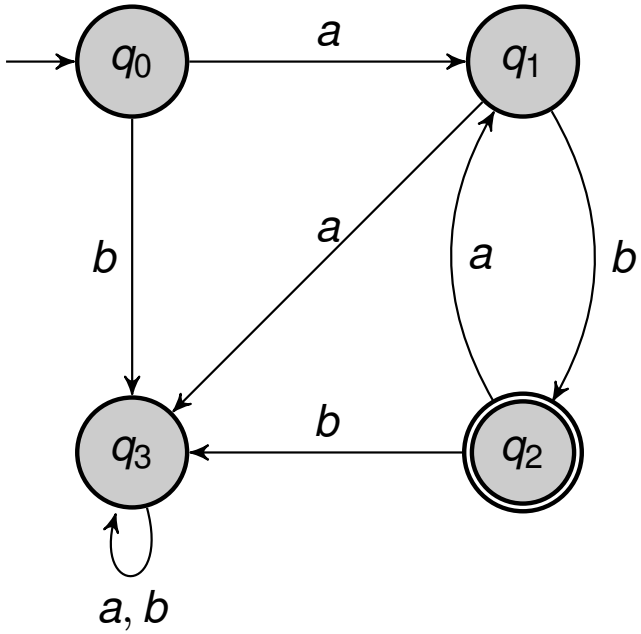
Check if it is { ***closed*** ✓ ***consistent***



Learning DFAs

Observation Table	
T_1	ϵ
ϵ	$-$
a	$-$
ab	$+$
b	$-$
aa	$-$
aba	$-$
abb	$-$

Check if it is { ***closed*** ✓ ***consistent*** ✗

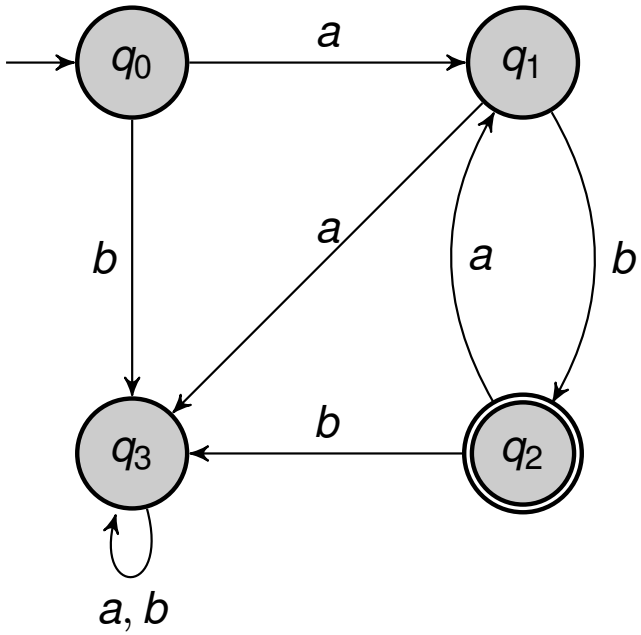


Learning DFAs

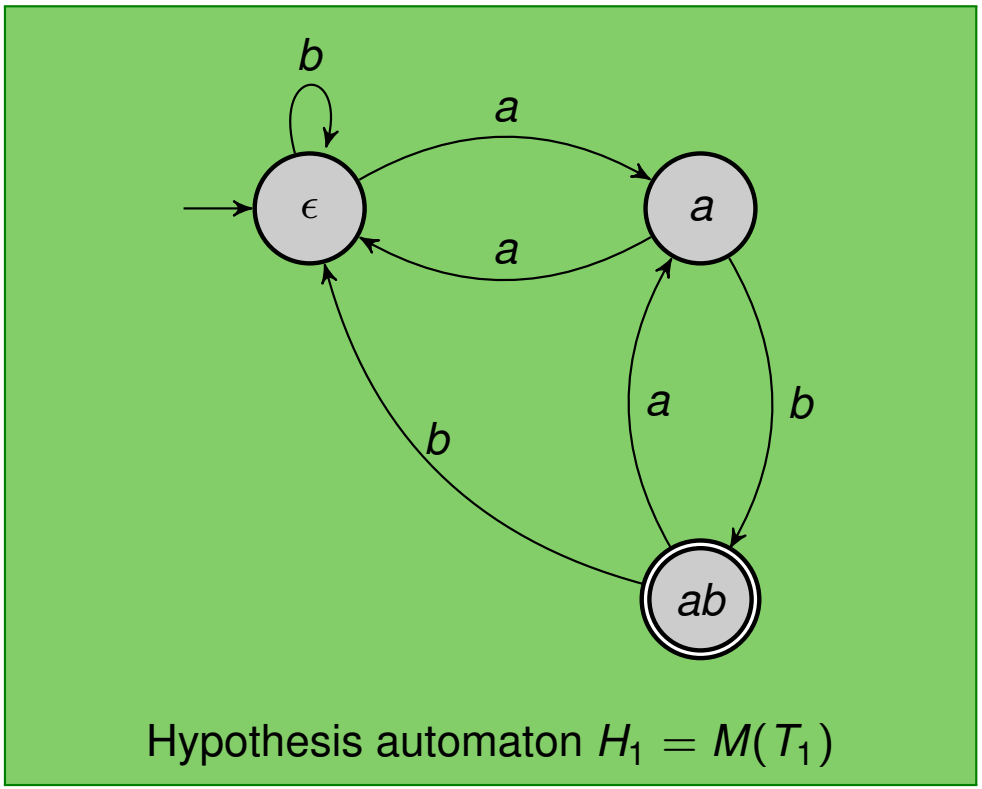
Observation Table

T_1	ϵ	b
ϵ	$-$	$-$
a	$-$	$+$
ab	$+$	$-$
b	$-$	$-$
aa	$-$	$-$
aba	$-$	$+$
abb	$-$	$-$

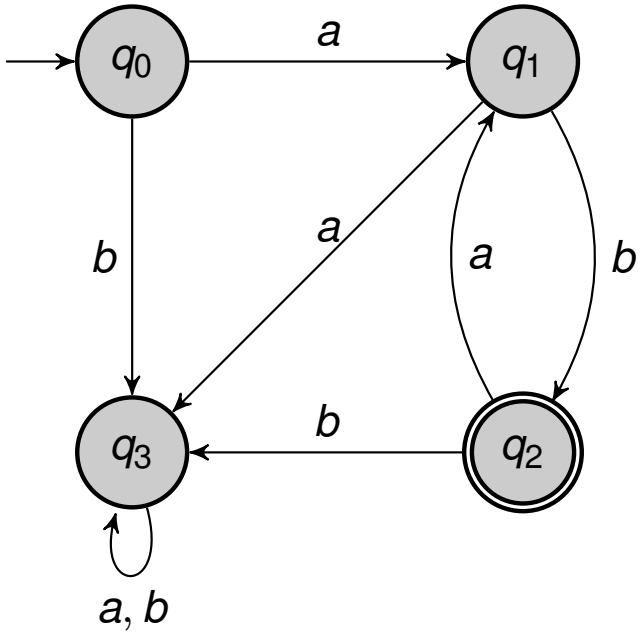
Add b to **E**



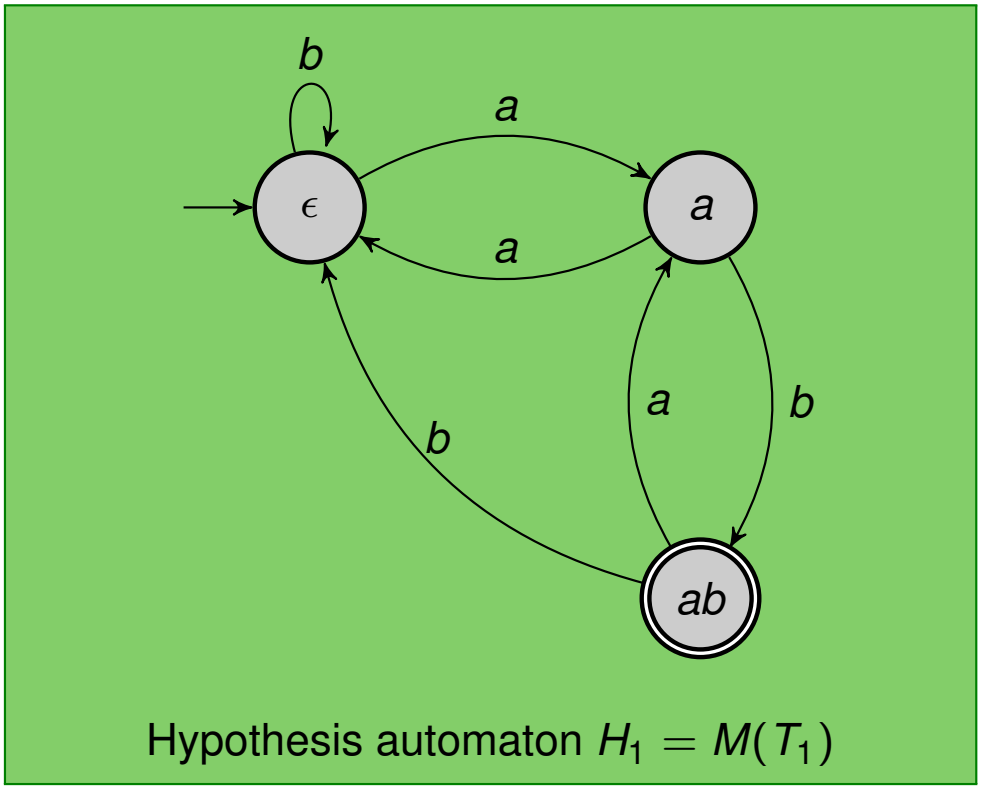
Learning DFAs



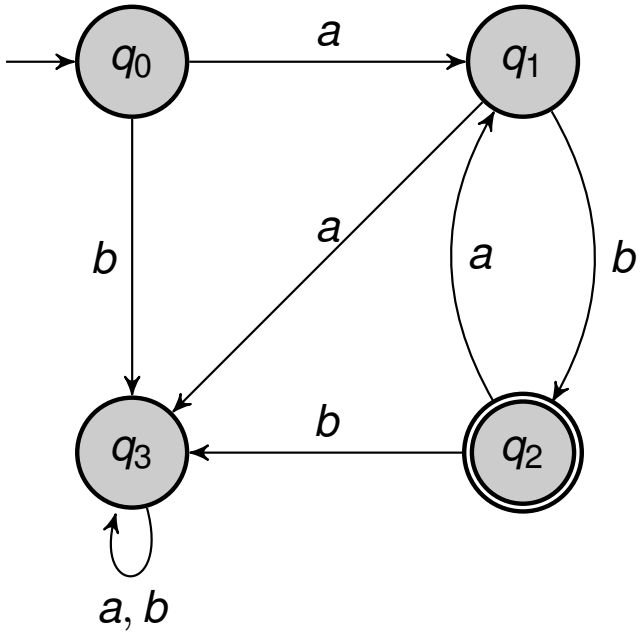
Hypothesis automaton $H_1 = M(T_1)$



Learning DFAs



Hypothesis automaton $H_1 = M(T_1)$

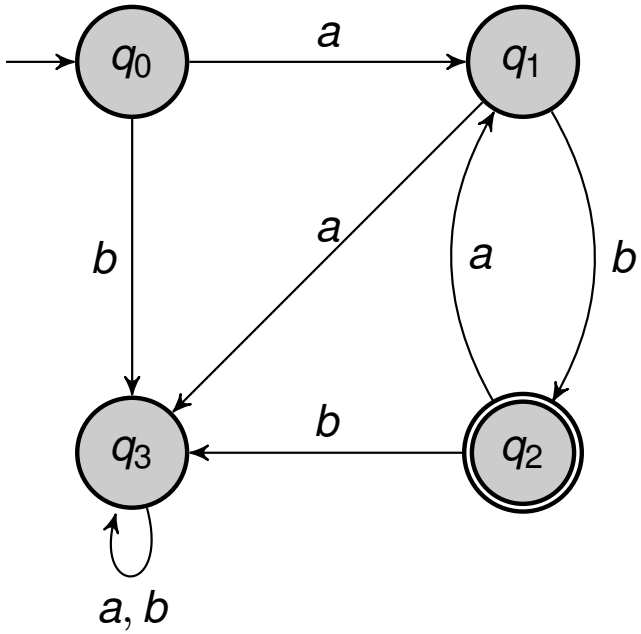


✗

Learning DFAs

Get a counter-example

$w' = bab$

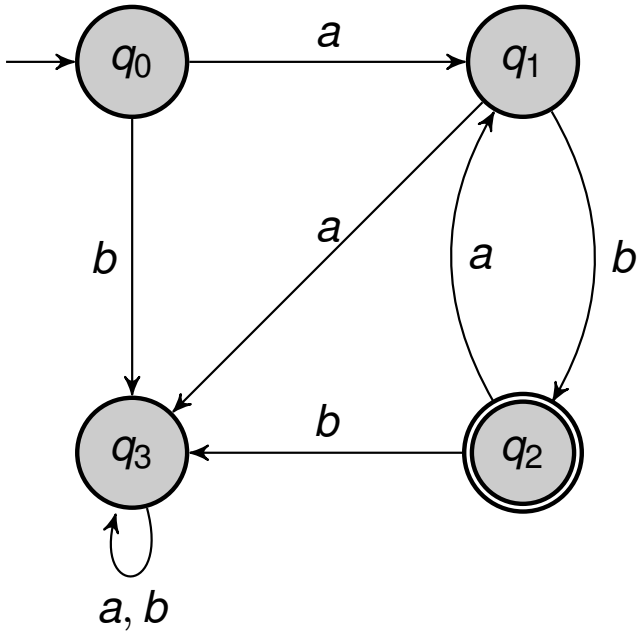


Learning DFAs

Get a counter-example

$w' = bab$

Add all prefixes to S.



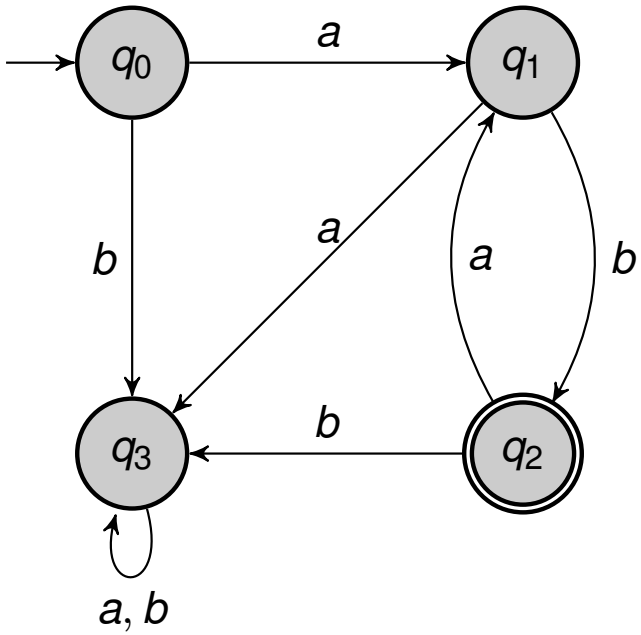
Learning DFAs

Observation Table		
T_1	ϵ	b
ϵ	−	−
a	−	+
b	−	−
ab	+	−
ba	−	−
bab	−	−

bb	−	−
aa	−	−
aba	−	+
abb	−	−
baa	−	−
bab	−	−
$baba$	−	−
$babb$	−	−

closed ✓

consistent ✗



Learning DFAs

Observation Table

T_1	ϵ	b
ϵ	−	−
a	−	+
b	−	−
ab	+	−
ba	−	−
bab	−	−

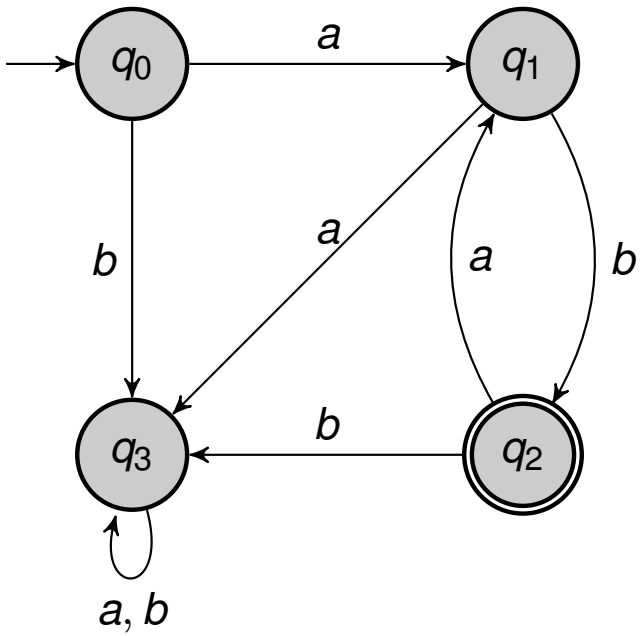
bb	−	−
aa	−	−
aba	−	+
abb	−	−
baa	−	−
bab	−	−
$baba$	−	−
$babb$	−	−

closed ✓

consistent ✗

Consider ϵ and b

$$f(\epsilon \cdot a \cdot b) \neq f(b \cdot a \cdot b)$$

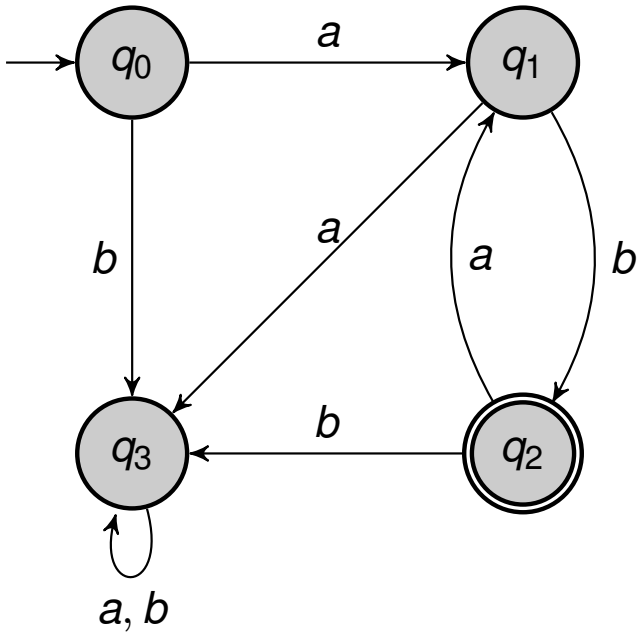


Learning DFAs

Observation Table

T_1	ϵ	b
ϵ	−	−
a	−	+
b	−	−
ab	+	−
ba	−	−
bab	−	−

bb	−	−
aa	−	−
aba	−	+
abb	−	−
baa	−	−
bab	−	−
$baba$	−	−
$babb$	−	−



closed ✓

consistent ✗

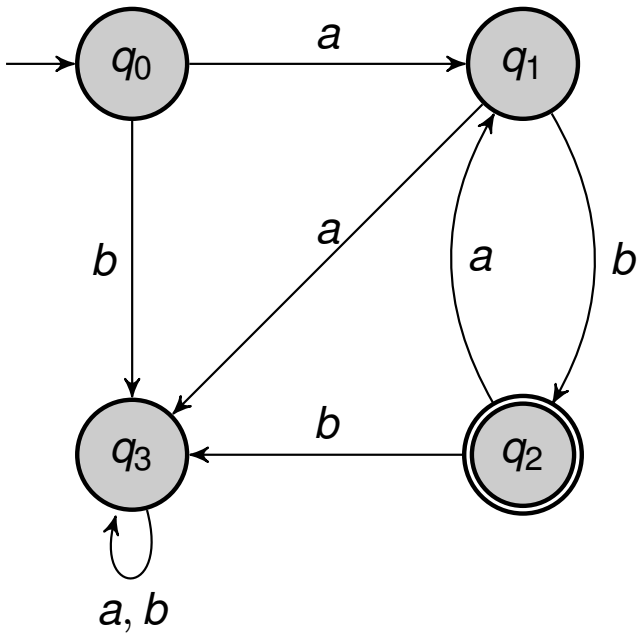
Consider ϵ and b

$$f(\epsilon \cdot a \cdot b) \neq f(b \cdot a \cdot b)$$

Add all suffixes of $a \cdot b$ to E.

Learning DFAs

Observation Table			
T_2	ϵ	b	ab
ϵ	−	−	+
a	−	+	−
b	−	−	−
ab	+	−	+
ba	−	−	−
bab	−	−	−
bb	−	−	−
aa	−	−	−
aba	−	+	−
abb	−	−	−
baa	−	−	−
bab	−	−	−
$baba$	−	−	−
$babb$	−	−	−



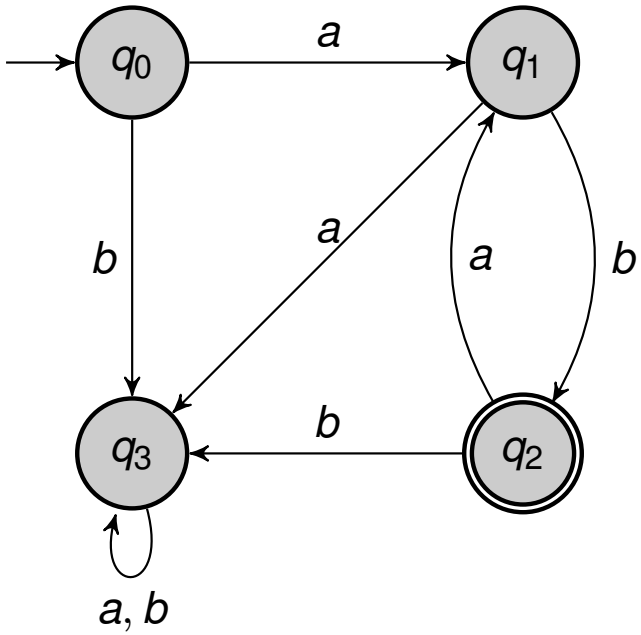
Learning DFAs

Observation Table

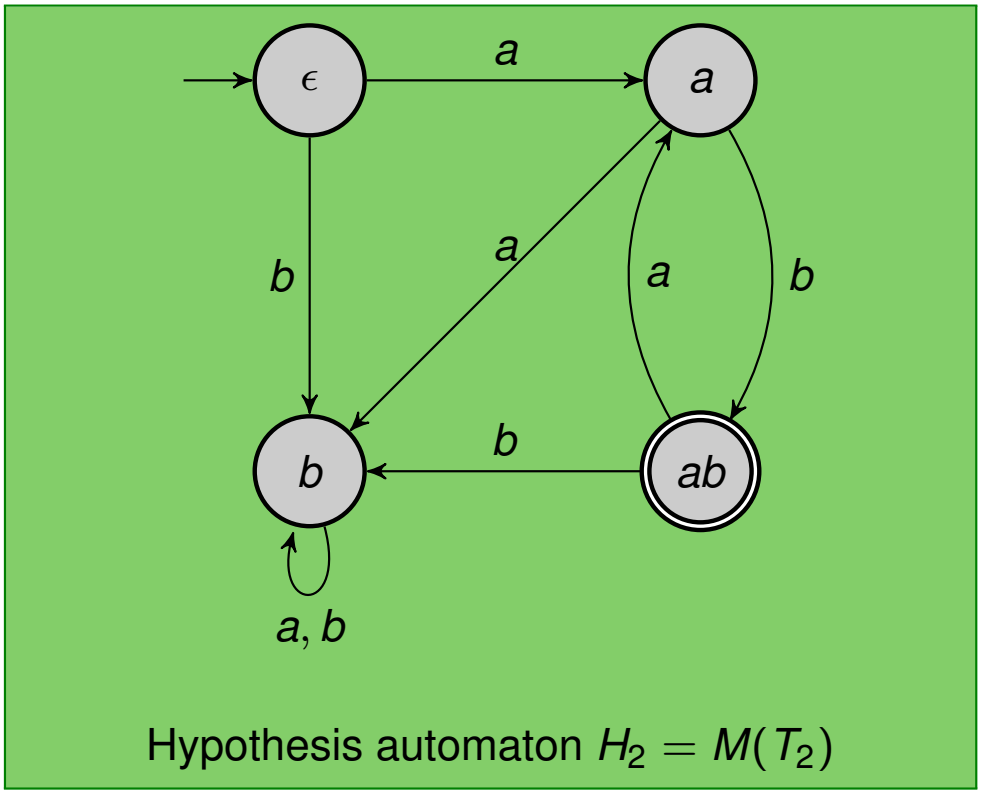
T_2	ϵ	b	ab
ϵ	−	−	+
a	−	+	−
b	−	−	−
ab	+	−	+
ba	−	−	−
bab	−	−	−
bb	−	−	−
aa	−	−	−
aba	−	+	−
abb	−	−	−
baa	−	−	−
bab	−	−	−
$baba$	−	−	−
$babb$	−	−	−

closed ✓

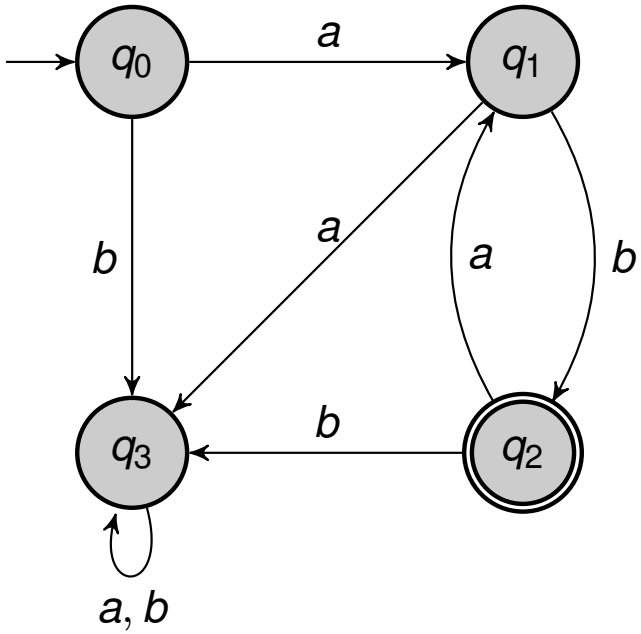
consistent ✓



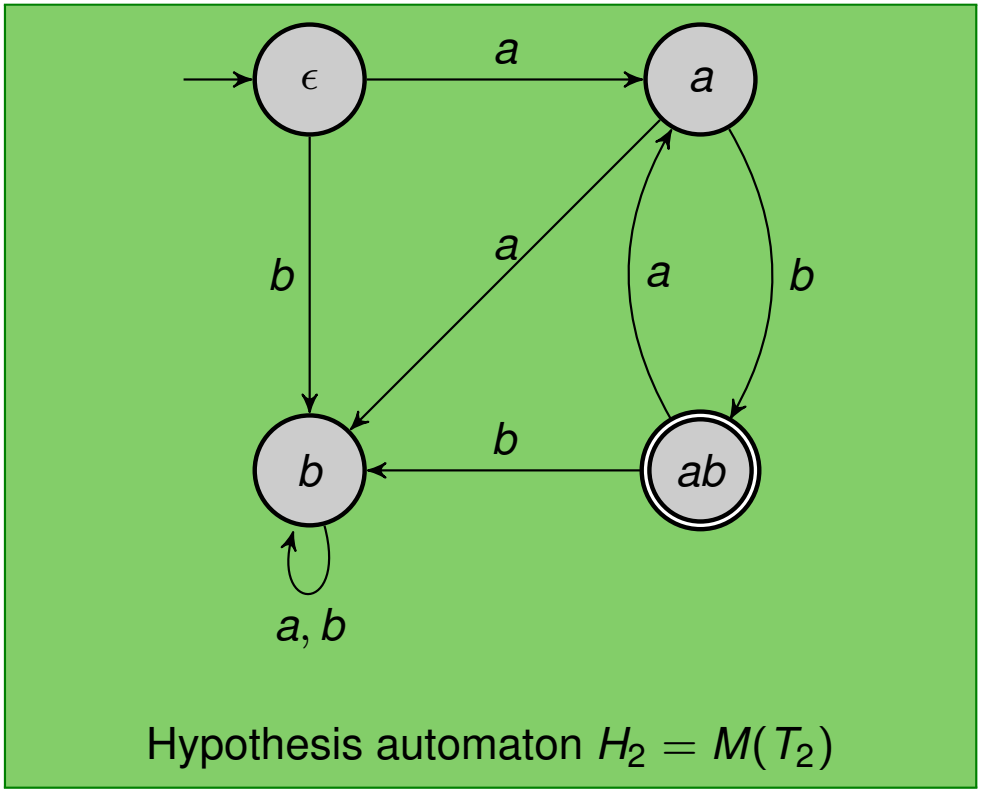
Learning DFAs

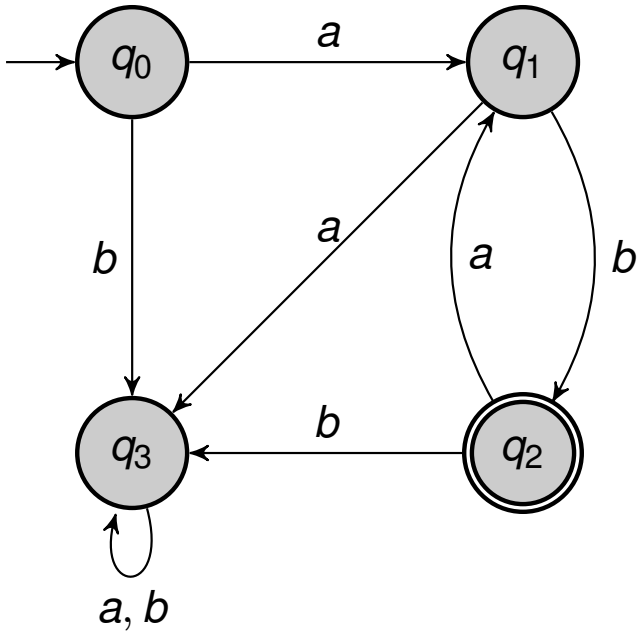


Hypothesis automaton $H_2 = M(T_2)$



Learning DFAs





✓

Implementation

There have been several **implementations** of learning algorithms, some notable ones are:

Implementation

There have been several **implementations** of learning algorithms, some notable ones are:

- **libalf**: The Automata Learning Framework
  Bollig B., Katoen JP., Kern C., Leucker M., Neider D., Piegdon D.R. (2010) libalf: The Automata Learning Framework. CAV 2010.

Implementation

There have been several **implementations** of learning algorithms, some notable ones are:

- **libalf**: The Automata Learning Framework
  Bollig B., Katoen JP., Kern C., Leucker M., Neider D., Piegdon D.R. (2010) libalf: The Automata Learning Framework. CAV 2010.
- The Open-Source **LearnLib**
  Isberner M., Howar F., Steffen B. (2015) The Open-Source LearnLib. CAV 2015.
- **Symbolic Automata**
  Drews S., D'Antoni L. (2017) Learning Symbolic Automata. TACAS 2017.

Learning

Learning Automata

Learning

Learning DFA by
D. Angluin in 1987

Learning

Learning DFA by D. Angluin in 1987

Learning NFA by B. Bollig et al. IJCAI 2009

Learning Automata

Learning

Learning DFA by D. Angluin in 1987

Learning NFA by B. Bollig et al. IJCAI 2009

Learning Languages Over Large Alphabets by O. Maler et al. TACAS 2014

Learning Automata

Learning

Learning DFA by D. Angluin in 1987

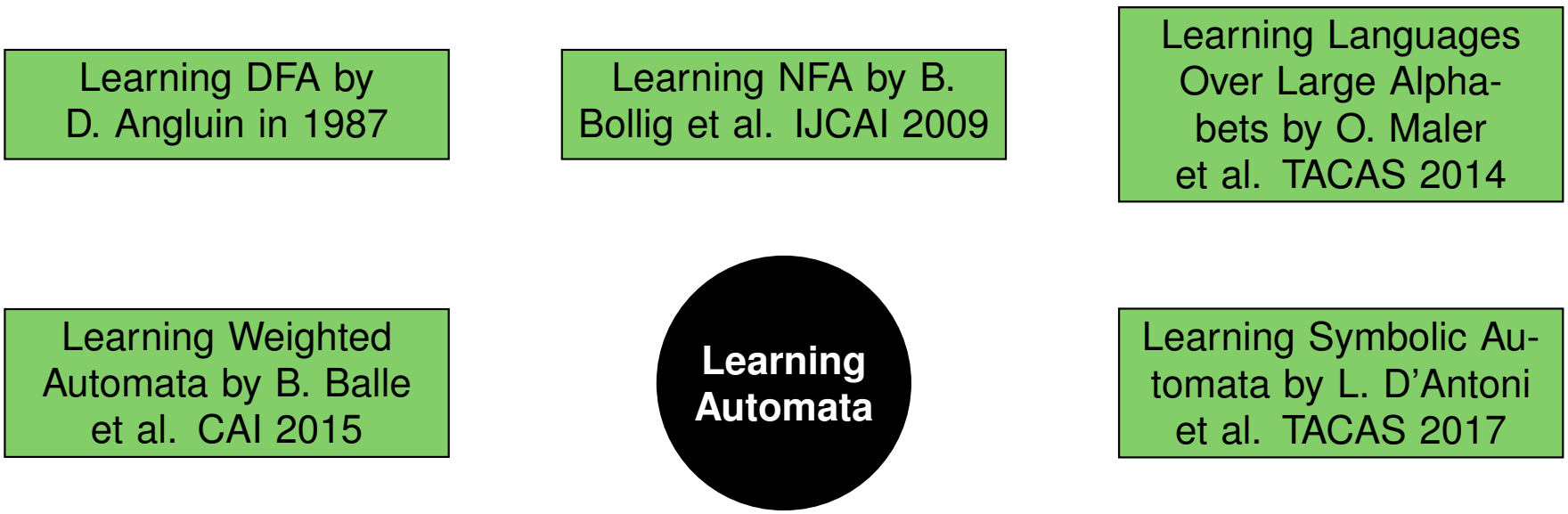
Learning NFA by B. Bollig et al. IJCAI 2009

Learning Languages Over Large Alphabets by O. Maler et al. TACAS 2014

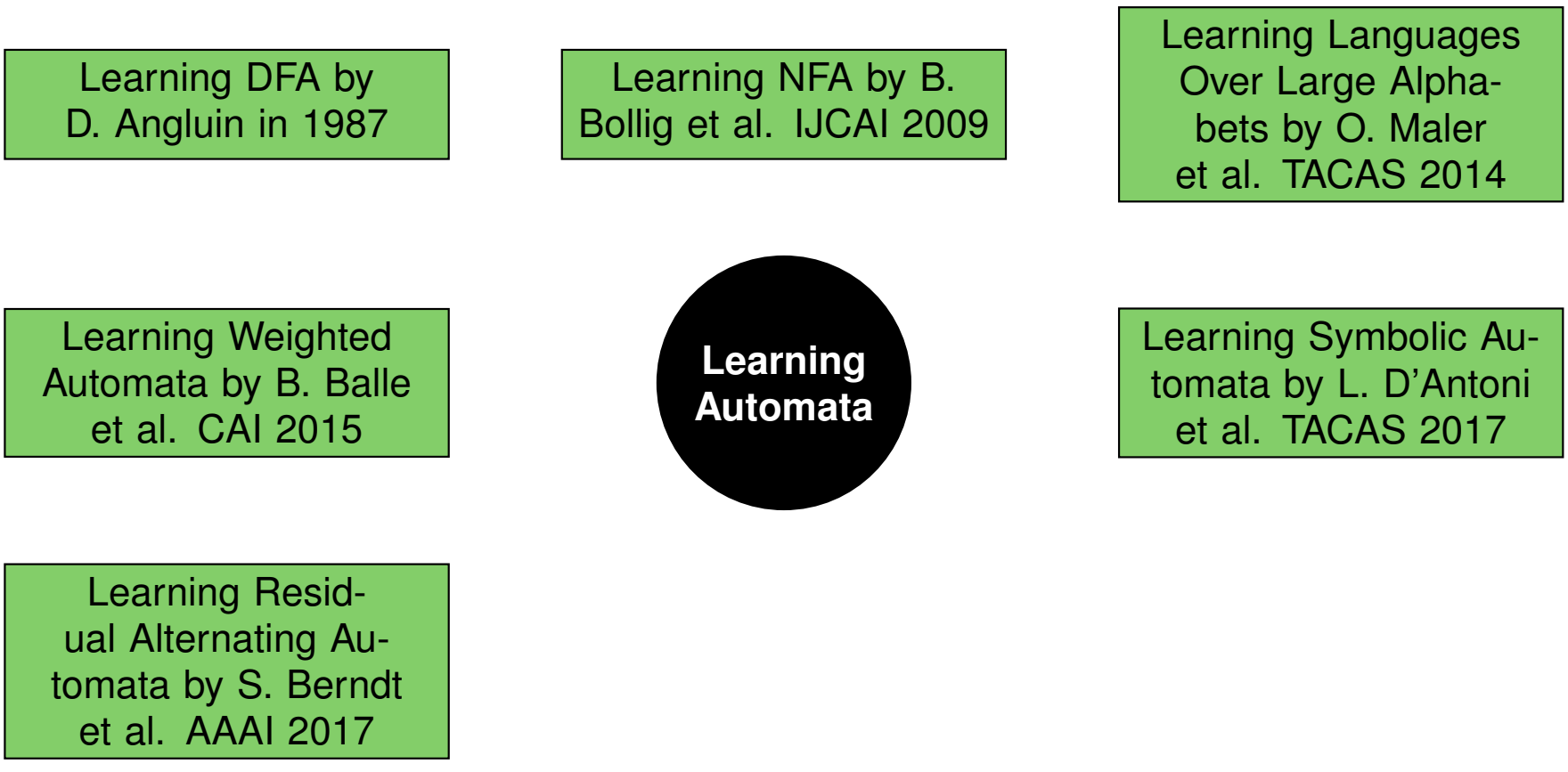
Learning Weighted Automata by B. Balle et al. CAI 2015

Learning Automata

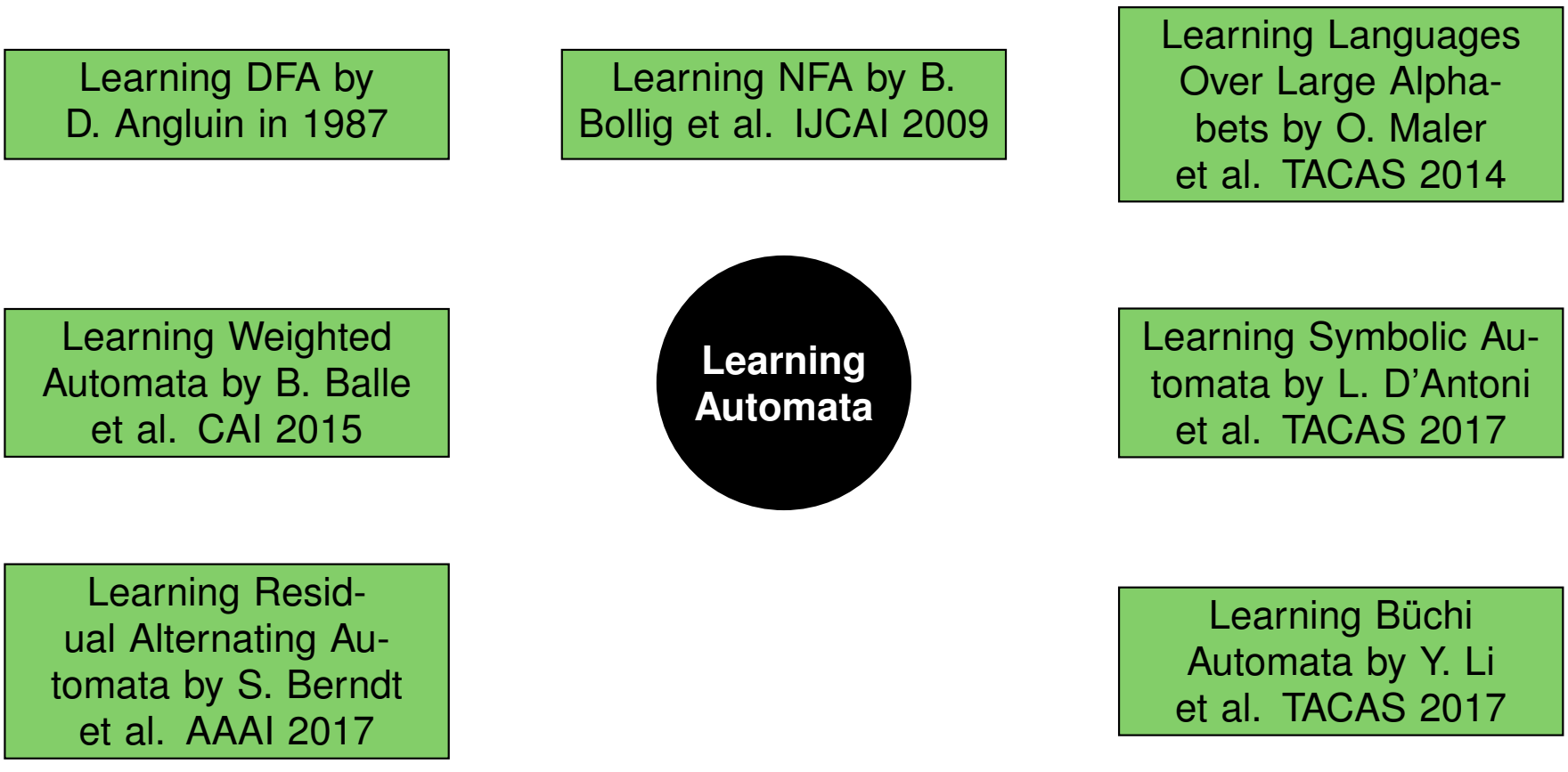
Learning


Learning DFA by D. Angluin in 1987
Learning NFA by B. Bollig et al. IJCAI 2009
Learning Languages Over Large Alphabets by O. Maler et al. TACAS 2014
Learning Weighted Automata by B. Balle et al. CAI 2015
Learning Automata
Learning Symbolic Automata by L. D'Antoni et al. TACAS 2017

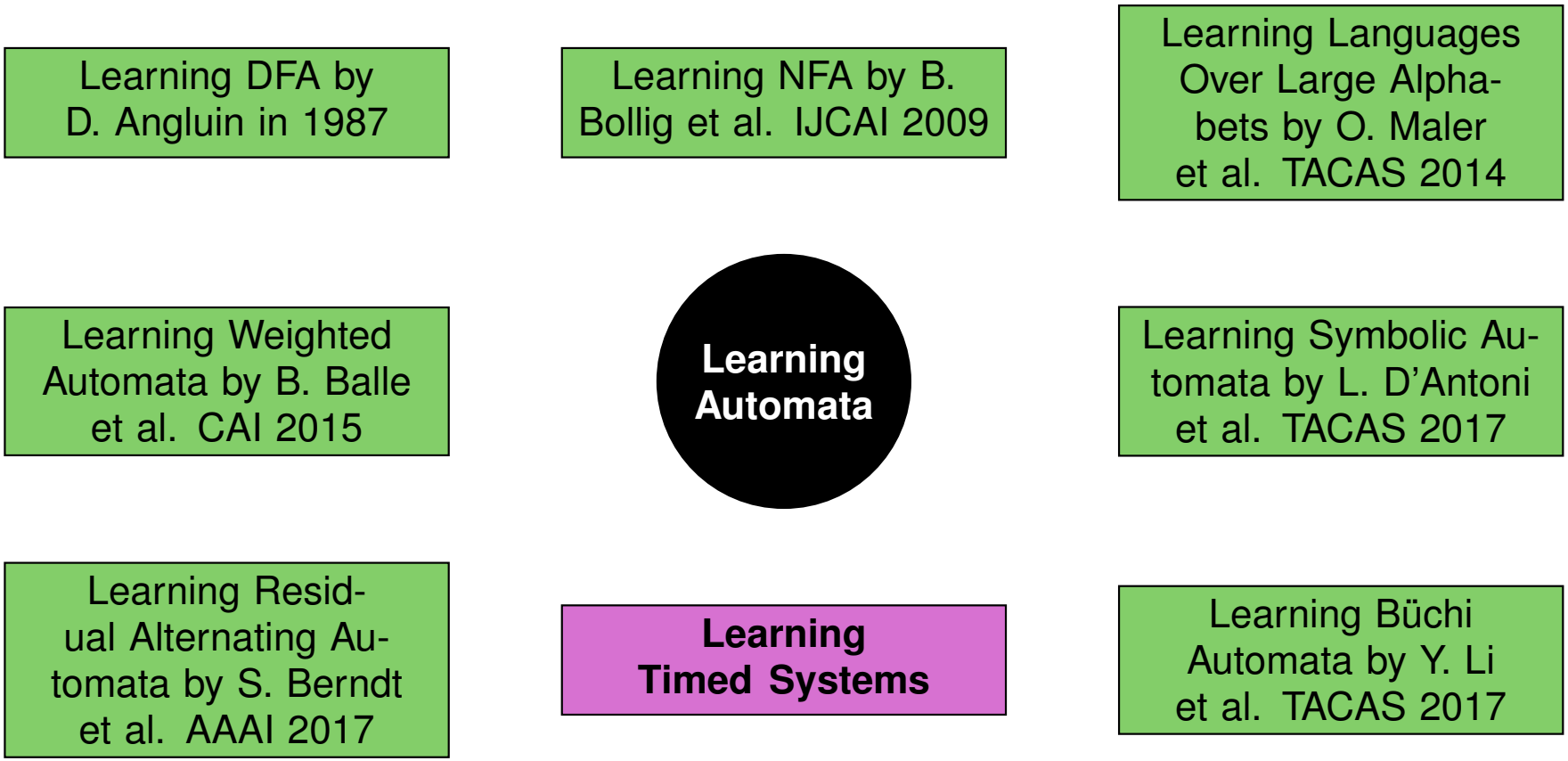
Learning


Learning DFA by D. Angluin in 1987
Learning NFA by B. Bollig et al. IJCAI 2009
Learning Languages Over Large Alphabets by O. Maler et al. TACAS 2014
Learning Weighted Automata by B. Balle et al. CAI 2015
Learning Automata
Learning Symbolic Automata by L. D'Antoni et al. TACAS 2017
Learning Residual Alternating Automata by S. Berndt et al. AAAI 2017

Learning


Learning DFA by D. Angluin in 1987
Learning NFA by B. Bollig et al. IJCAI 2009
Learning Languages Over Large Alphabets by O. Maler et al. TACAS 2014
Learning Weighted Automata by B. Balle et al. CAI 2015
Learning Automata
Learning Symbolic Automata by L. D'Antoni et al. TACAS 2017
Learning Residual Alternating Automata by S. Berndt et al. AAAI 2017
Learning Büchi Automata by Y. Li et al. TACAS 2017

Learning


Learning DFA by D. Angluin in 1987
Learning NFA by B. Bollig et al. IJCAI 2009
Learning Languages Over Large Alphabets by O. Maler et al. TACAS 2014
Learning Weighted Automata by B. Balle et al. CAI 2015
Learning Automata
Learning Symbolic Automata by L. D'Antoni et al. TACAS 2017
Learning Residual Alternating Automata by S. Berndt et al. AAAI 2017
Learning Timed Systems
Learning Büchi Automata by Y. Li et al. TACAS 2017

Learning Timed Systems

Learning Timed Systems

Efficiently Learning Simple Timed Automata by S. Verwer et al. in 2008

Learning Timed Systems

Efficiently Learning Simple Timed Automata by S. Verwer et al. in 2008

Learning Event-recording Automata by O. Grinchtein et al. FTRTFT 2004, FORMATS 2004

Learning Timed Systems

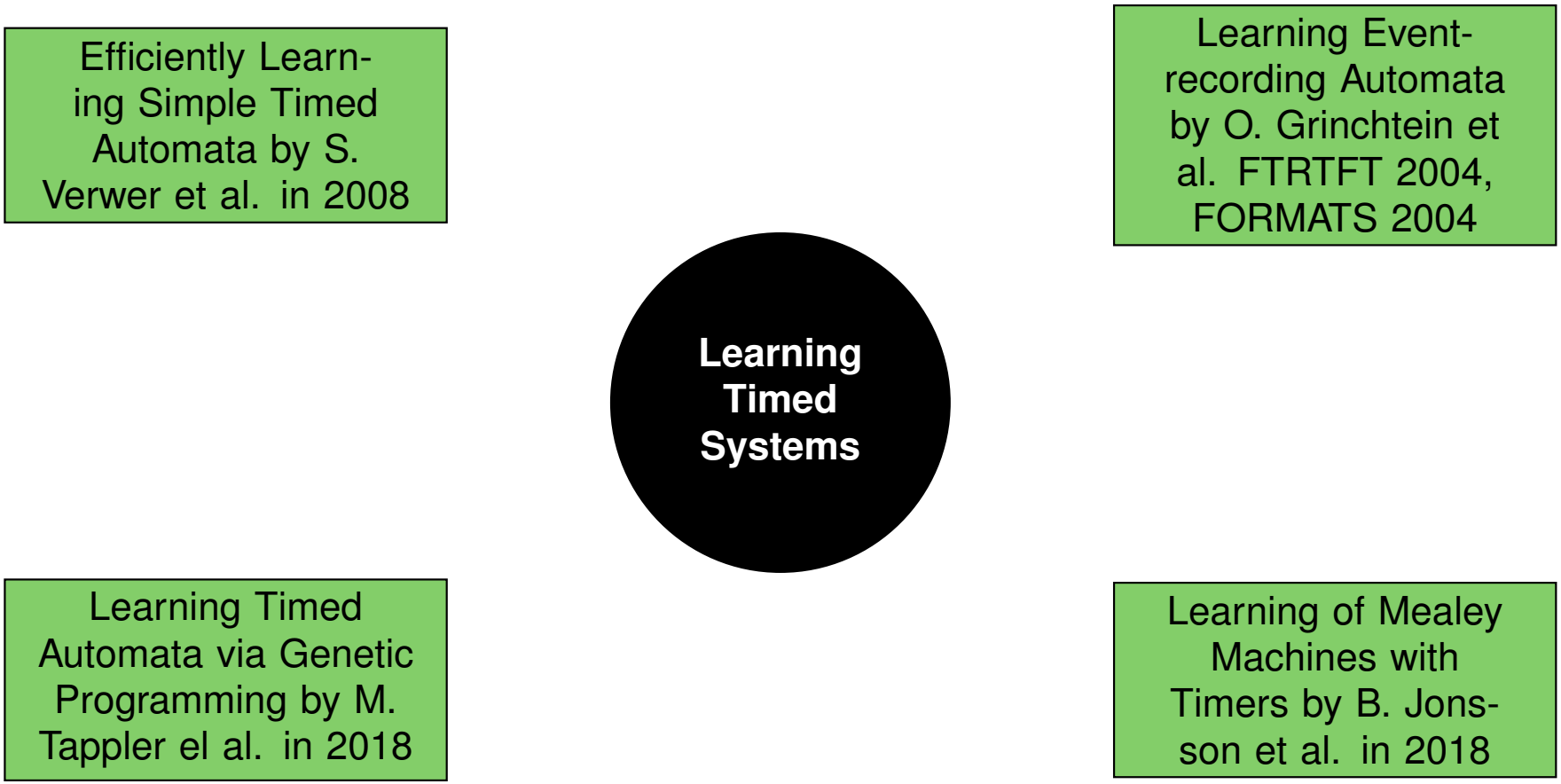
Efficiently Learning Simple Timed Automata by S. Verwer et al. in 2008

Learning Event-recording Automata by O. Grinchtein et al. FTRTFT 2004, FORMATS 2004

Learning Timed Systems

Learning Timed Automata via Genetic Programming by M. Tappler el al. in 2018

Learning Timed Systems



What we want to do and why?

Find an active learning algorithm for a subclass of timed automata.

[1] Verwer, Sicco & Weerdt, Mathijs & Witteveen, Cees. (2008). Efficiently learning simple timed automata. Techniques in Coloproctology - TECH COLOPROCTOLOGY.

What we want to do and why?

Find an active learning algorithm for a subclass of timed automata.

Why though?

[1] Verwer, Sicco & Weerdt, Mathijs & Witteveen, Cees. (2008). Efficiently learning simple timed automata. Techniques in Coloproctology - TECH COLOPROCTOLOGY.

What we want to do and why?

Find an active learning algorithm for a subclass of timed automata.

Why though?

- We can model some real-time systems: network protocols, business processes, reactive systems.[1]
  It is difficult to model these systems by hand.

[1] Verwer, Sicco & Weerdt, Mathijs & Witteveen, Cees. (2008). Efficiently learning simple timed automata. Techniques in Coloproctology - TECH COLOPROCTOLOGY.

What we want to do and why?

Find an active learning algorithm for a subclass of timed automata.

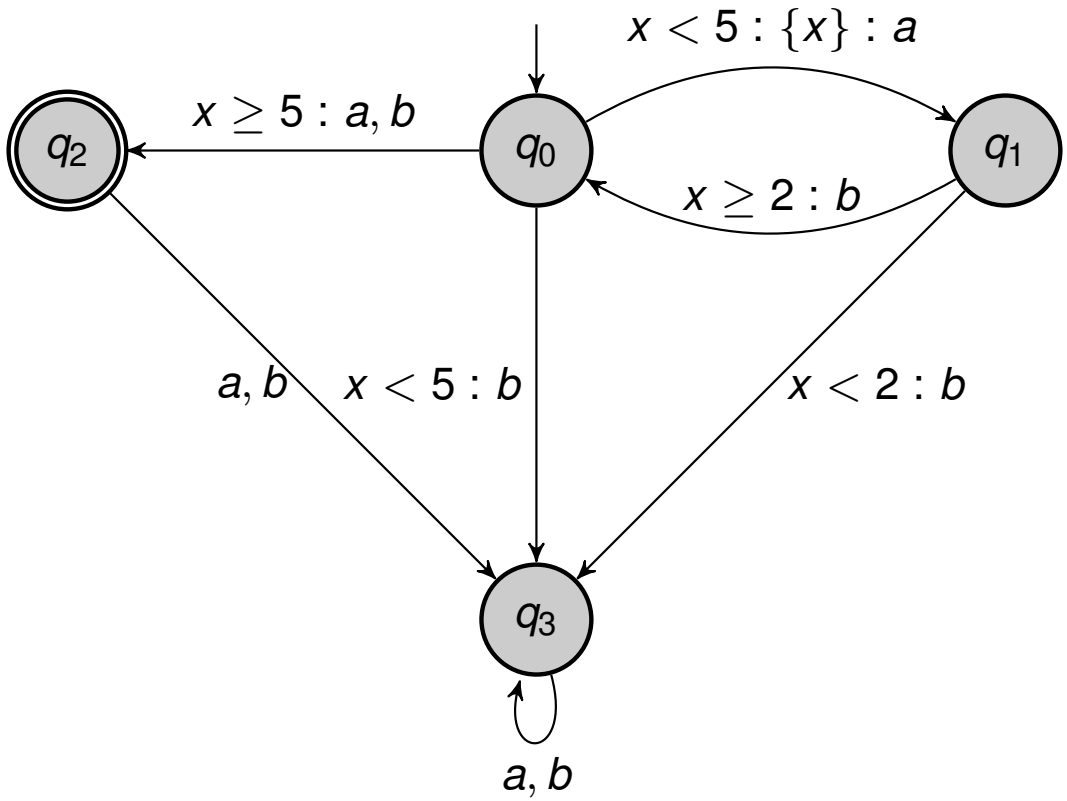
Why though?

- We can model some real-time systems: network protocols, business processes, reactive systems.[1]
  It is difficult to model these systems by hand.
- A lot of other examples are given in 'The Unmet Challenge of Timed Systems' by O. Maler.

[1] Verwer, Sicco & Weerdt, Mathijs & Witteveen, Cees. (2008). Efficiently learning simple timed automata. Techniques in Coloproctology - TECH COLOPROCTOLOGY.

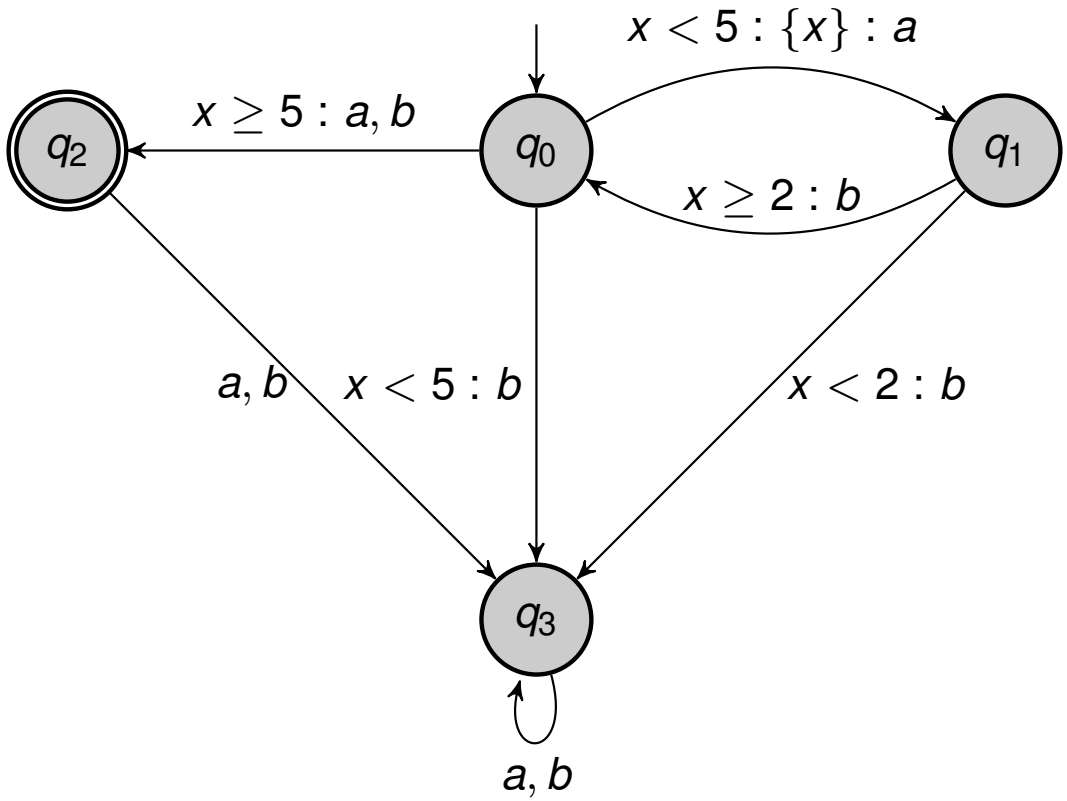
Discrete Timed Automata

Timed Automata



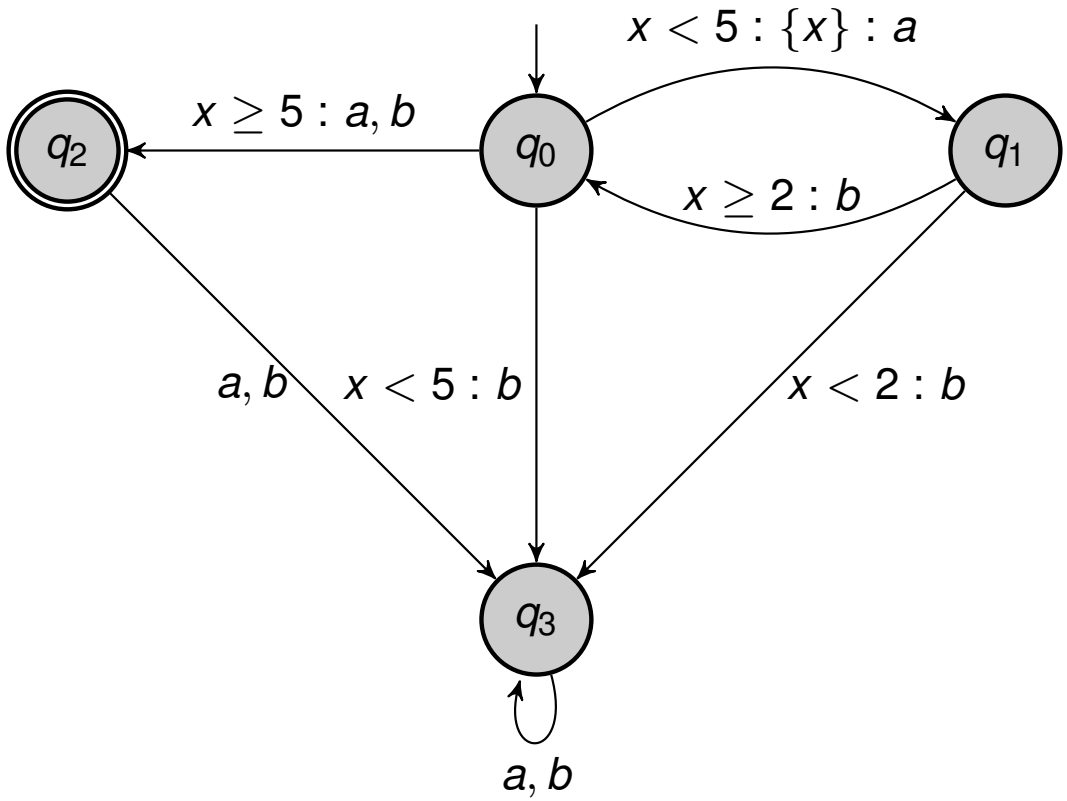
Example of a **timed automaton**.

Timed Automata



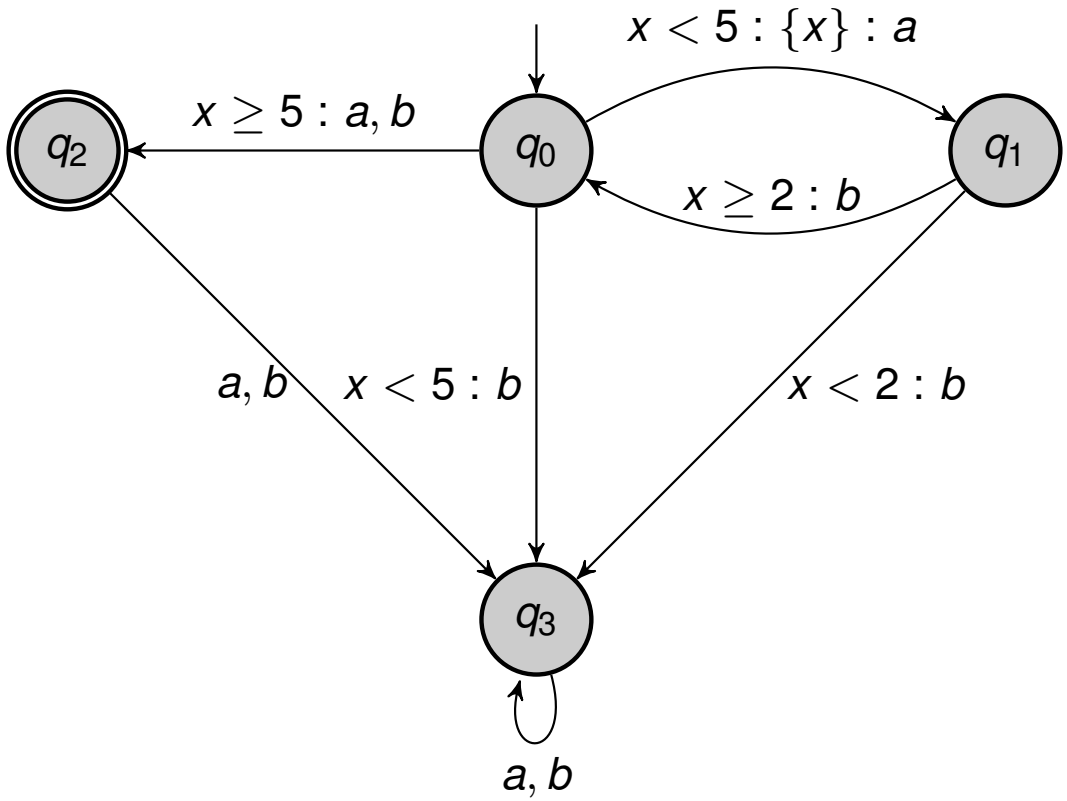
Example of a **timed automaton**. $q_0 \overset{a}{\underset{2.5}{\longrightarrow}} q_1 \overset{b}{\underset{4.7}{\longrightarrow}} q_0 \overset{a}{\underset{6.3}{\longrightarrow}} q_1 \overset{b}{\underset{10.1}{\longrightarrow}} q_0 \overset{a}{\underset{12.0}{\longrightarrow}} q_2$

Timed Automata



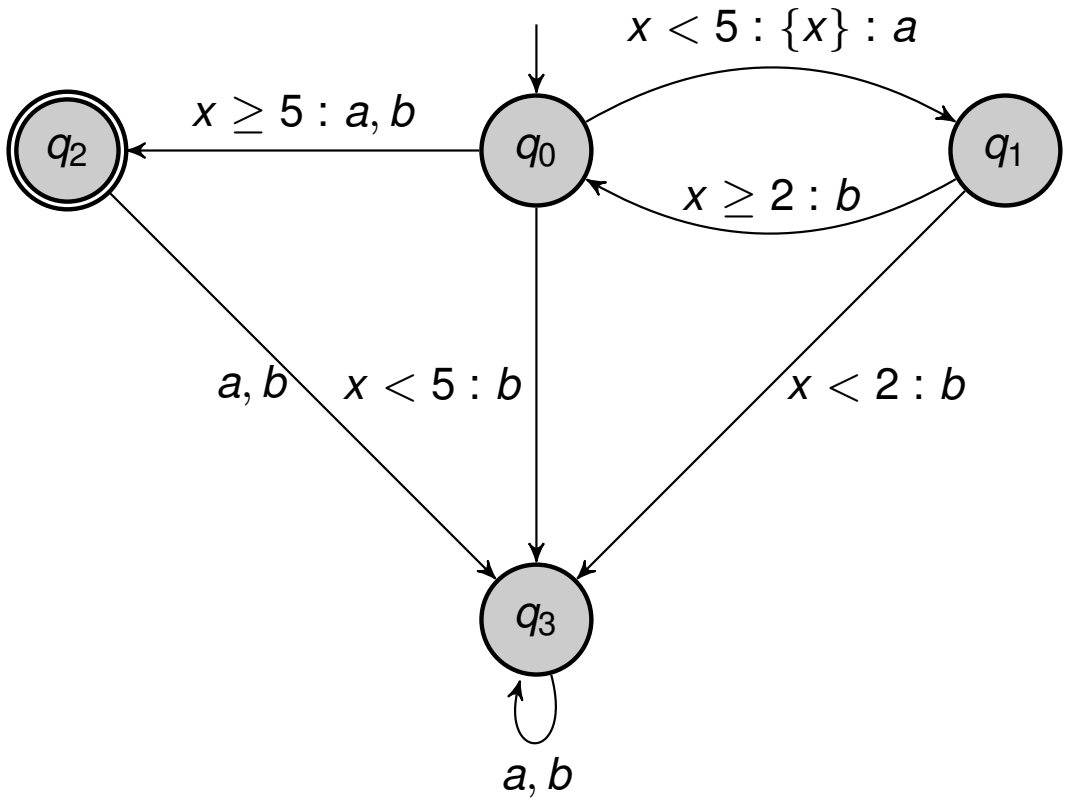
Example of a **timed automaton**. $q_0 \overset{a}{\underset{2.5}{\longrightarrow}} q_1 \overset{b}{\underset{4.7}{\longrightarrow}} q_0 \overset{a}{\underset{6.3}{\longrightarrow}} q_1 \overset{b}{\underset{10.1}{\longrightarrow}} q_0 \overset{a}{\underset{12.0}{\longrightarrow}} q_2$ ✓

Discrete Timed Automata



This is a **discrete timed automaton** also.

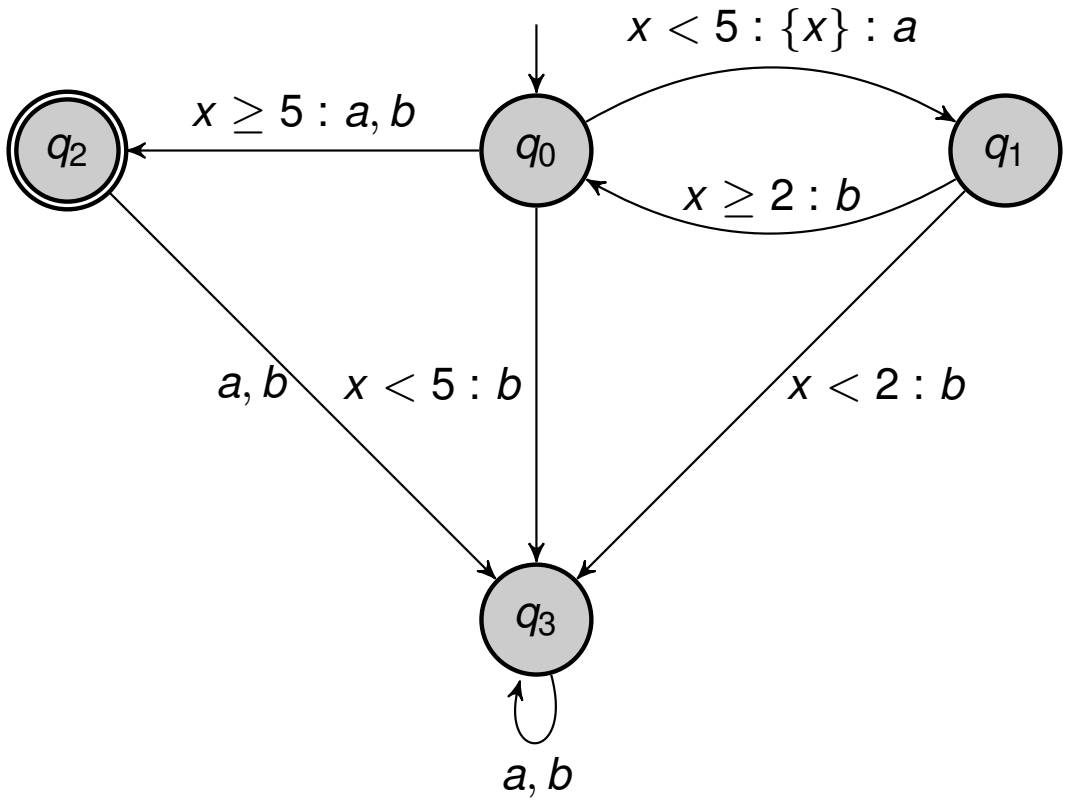
Discrete Timed Automata



This is a **discrete timed automaton** also.

$$q_0 \xrightarrow[3]{a} q_1 \xrightarrow[6]{b} q_0 \xrightarrow[6]{a} q_1 \xrightarrow[8]{b} q_0 \xrightarrow[12]{a} q_2$$

Discrete Timed Automata



This is a **discrete timed automaton** also.

$$q_0 \xrightarrow[3]{a} q_1 \xrightarrow[6]{b} q_0 \xrightarrow[6]{a} q_1 \xrightarrow[8]{b} q_0 \xrightarrow[12]{a} q_2 \quad \checkmark$$

Learning Discrete Timed Automata

Why learn discrete timed automata

- Discrete time is easier to handle than dense time.
- Expressive enough.
- Can be applied to real world systems.

Learning Discrete Timed Automata Using L^*

Learn *discrete timed automata* as a **DFA**.

Learning Discrete Timed Automata Using L^*

Learn *discrete timed automata* as a **DFA**.

We can convert each *timed* string into an *untimed* string using a new alphabet symbol ✓ , for e.g.

Learning Discrete Timed Automata Using L^*

Learn *discrete timed automata* as a **DFA**.

We can convert each *timed* string into an *untimed* string using a new alphabet symbol ✓ , for e.g.

$(2, a)(3, b)(6, b)(6, a)$

Learning Discrete Timed Automata Using L^*

Learn *discrete timed automata* as a **DFA**.

We can convert each *timed* string into an *untimed* string using a new alphabet symbol ✓ , for e.g.

$(2, a)(3, b)(6, b)(6, a) \longrightarrow$ ✓ ✓ a ✓ b ✓ ✓ ✓ b a

Learning Discrete Timed Automata Using L^*

Learn *discrete timed automata* as a **DFA**.

We can convert each *timed* string into an *untimed* string using a new alphabet symbol ✓ , for e.g.

$(2, a)(3, b)(6, b)(6, a) \longrightarrow$ ✓ ✓ a ✓ b ✓ ✓ ✓ b a

Each tick represents one time unit.

For a timed language $L \subseteq (\mathbb{N} \times \Sigma)^*$, we can convert it into a regular language $L' \subseteq (\Sigma \cup \{✓\})^*$

Learning Discrete Timed Automata Using L^*

Learn *discrete timed automata* as a **DFA**.

We can convert each *timed* string into an *untimed* string using a new alphabet symbol ✓ , for e.g.

$(2, a)(3, b)(6, b)(6, a) \longrightarrow$ ✓ ✓ a ✓ b ✓ ✓ ✓ b a

Each tick represents one time unit.

For a timed language $L \subseteq (\mathbb{N} \times \Sigma)^*$, we can convert it into a regular language $L' \subseteq (\Sigma \cup \{✓\})^*$

Size of this automaton will be **HUGE**.

But, can be learned using $\mathbf{L}^*$.

A better approach

If we can group the *ticks* together, we can get a smaller automaton.

A better approach

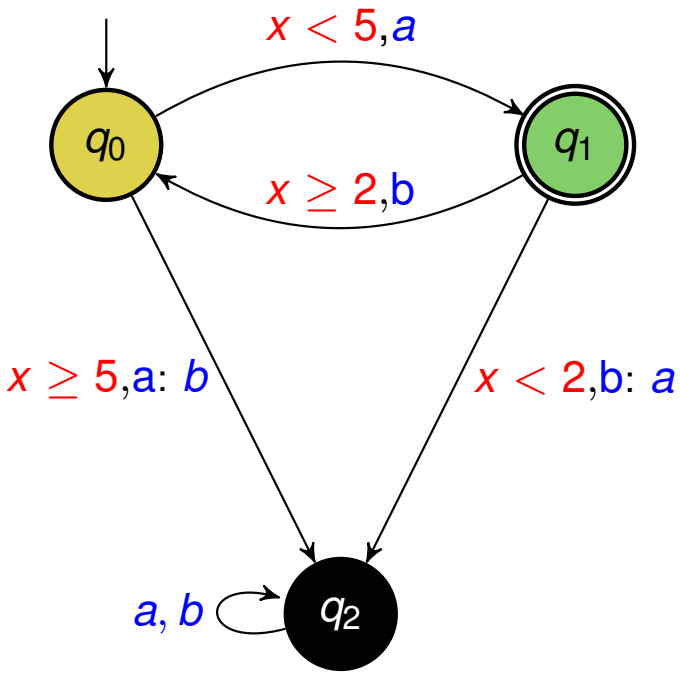
If we can group the *ticks* together, we can get a smaller automaton.

If we consider that the automaton has only **one clock which gets reset on every transition**,

A better approach

If we can group the *ticks* together, we can get a smaller automaton.

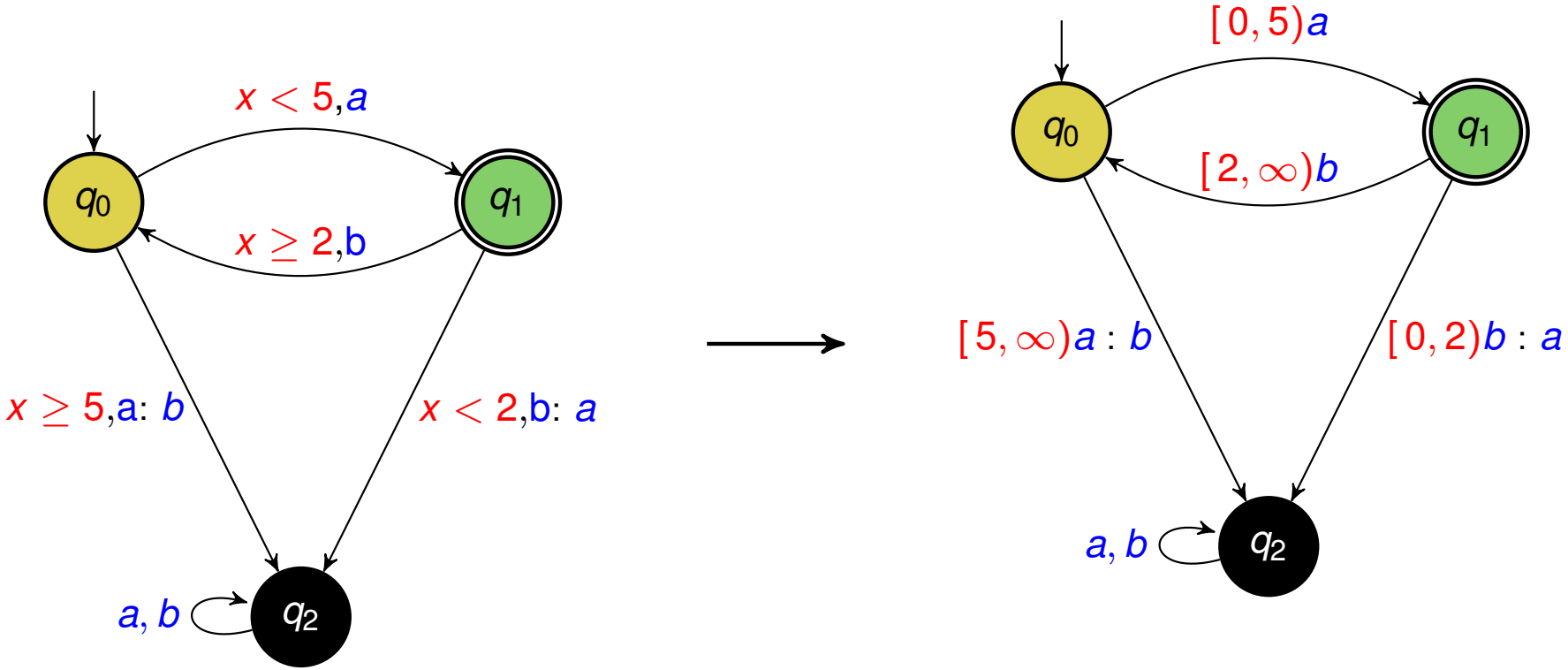
If we consider that the automaton has only **one clock which gets reset on every transition**,



A better approach

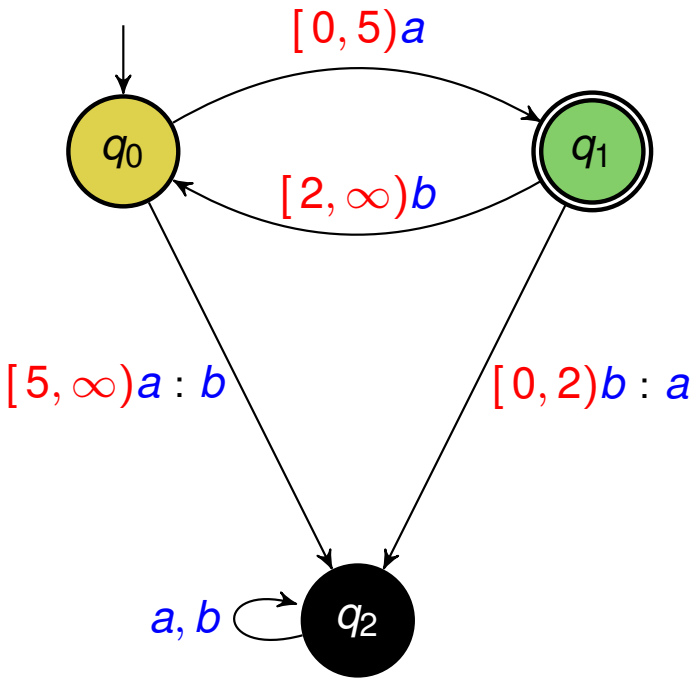
If we can group the *ticks* together, we can get a smaller automaton.

If we consider that the automaton has only **one clock which gets reset on every transition**,

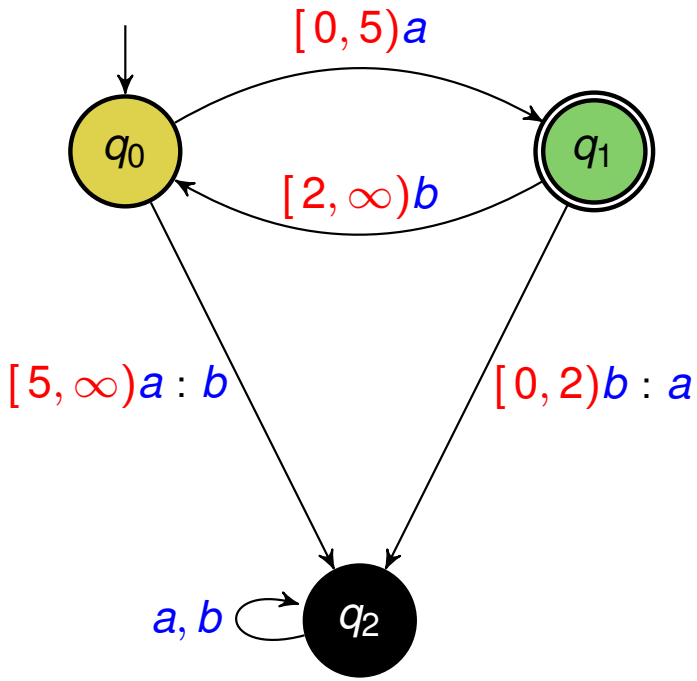


This looks like a **symbolic automata**.

A better approach

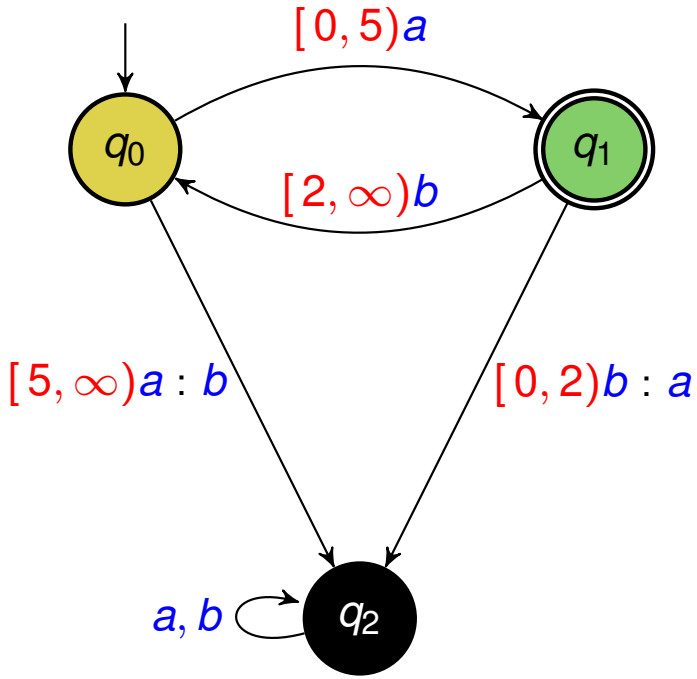


A better approach



Intervals at each transition represents the value of clock for which transition is valid.

A better approach



Intervals at each transition represents the value of clock for which transition is valid.

Consider this transformation:

$$(2,a)(3,b)(6,b)(6,a) \longrightarrow (2,a)(1,b)(3,b)(0,a)$$

Equivalence of DiTA and r-DDTA

DiTA := class of discrete timed automata.

Equivalence of DiTA and r-DDTA

DiTA := class of discrete timed automata.

r-DDTA := class of 1-clock deterministic discrete timed automata with resets on every transition.

Equivalence of DiTA and r-DDTA

DiTA := class of discrete timed automata.

r-DDTA := class of 1-clock deterministic discrete timed automata with resets on every transition.

Important Theorem:

Every *discrete timed automata* can be converted into a *1-clock deterministic discrete timed automata* with resets on every transition.

Equivalence of DiTA and r-DDTA

DiTA := class of discrete timed automata.

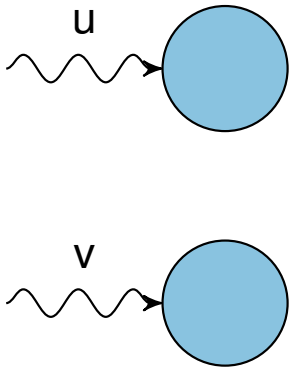
r-DDTA := class of 1-clock deterministic discrete timed automata with resets on every transition.

Important Theorem:

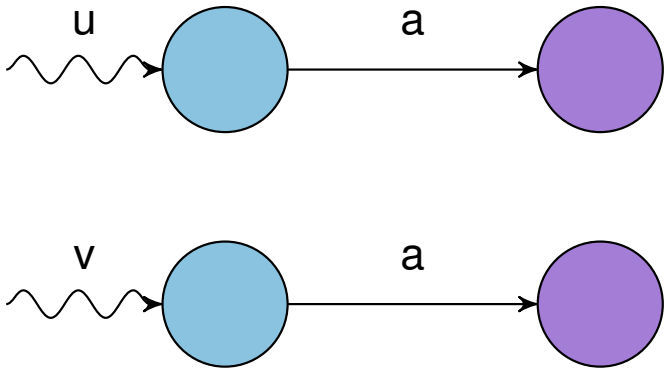
Every *discrete timed automata* can be converted into a *1-clock deterministic discrete timed automata* with resets on every transition.

Learn a r-DDTA for a given DiTA.

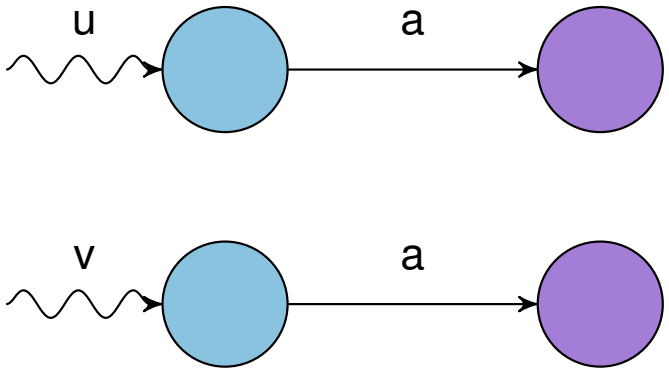
Why reset on every transition?



Why reset on every transition?

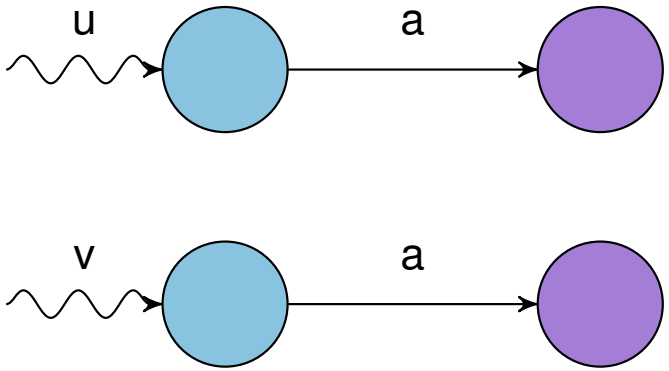


Why reset on every transition?



In some sense, the past does not matter. So, we are able to learn these equivalence classes.

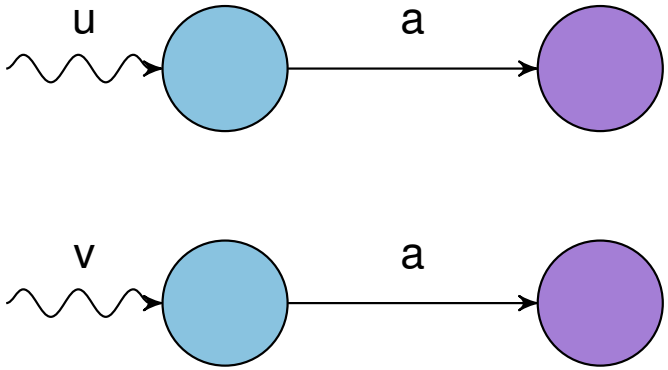
Why reset on every transition?



In some sense, the past does not matter. So, we are able to learn these equivalence classes.

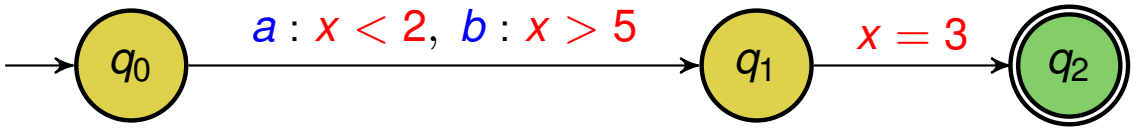
This is not the case for timed automata.

Why reset on every transition?

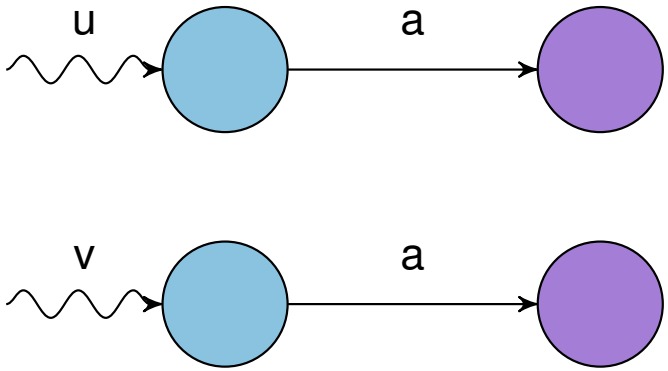


In some sense, the past does not matter. So, we are able to learn these equivalence classes.

This is not the case for timed automata.

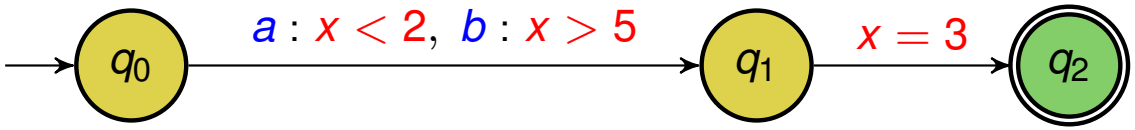


Why reset on every transition?



In some sense, the past does not matter. So, we are able to learn these equivalence classes.

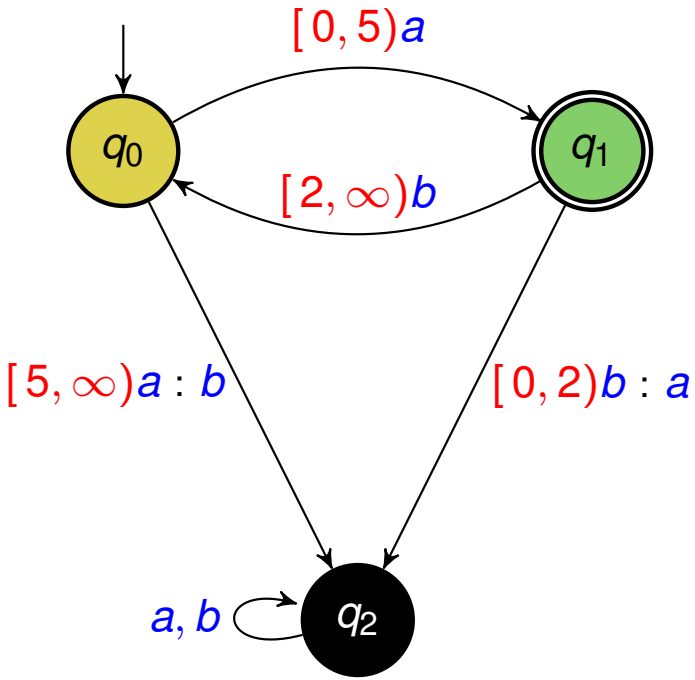
This is not the case for timed automata.



On reading $(1, a)$ and $(6, b)$, automaton is in q_1 state but $(6, b)$ cannot take transition to the accepting state.

Learning r-DDTA

Learning r-DDTA



Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	
$[0, \infty)a$	
$[0, \infty)b$	

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0, \infty)a) = (0, a)$$
$$\mu([0, \infty)b) = (0, b)$$

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	
$[0,\infty)a$	
$[0,\infty)b$	

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed**

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed**

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed** ✗

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$
$[0,\infty)a[0,\infty)a$	$-$
$[0,\infty)a[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed** ✗

Add $[0,\infty)a$ to S and its continuations to R

and set $\mu([0,\infty)a[0,\infty)a) = (0,a)(0,a)$
$\mu([0,\infty)a[0,\infty)b) = (0,a)(0,b)$

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$
$[0,\infty)a[0,\infty)a$	$-$
$[0,\infty)a[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed** ✓

Add $[0,\infty)a$ to S and its continuations to R

and set $\mu([0,\infty)a[0,\infty)a) = (0,a)(0,a)$
$\mu([0,\infty)a[0,\infty)b) = (0,a)(0,b)$

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$
$[0,\infty)a[0,\infty)a$	$-$
$[0,\infty)a[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

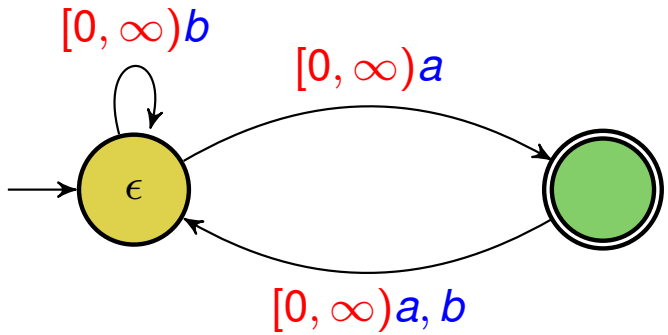
$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed** ✓

Add $[0,\infty)a$ to S and its continuations to R

and set $\mu([0,\infty)a[0,\infty)a) = (0,a)(0,a)$
$\mu([0,\infty)a[0,\infty)b) = (0,a)(0,b)$



Hypothesis automaton: H_0

Learning r-DDTA

Observation Table	
T_0	ϵ
ϵ	$-$
$[0,\infty)a$	$+$
$[0,\infty)b$	$-$
$[0,\infty)a[0,\infty)a$	$-$
$[0,\infty)a[0,\infty)b$	$-$

μ: **evidence function** (for each entry in the table there are some evidences)

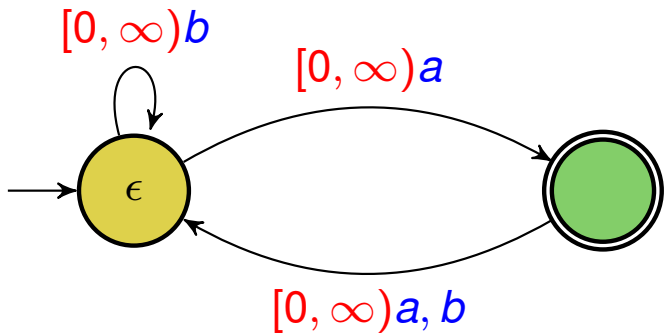
$$\mu([0,\infty)a) = (0,a)$$
$$\mu([0,\infty)b) = (0,b)$$

Fill the table by asking **membership queries** only for the evidences of an element of the table.

Check if the table is **closed** ✓

Add $[0,\infty)a$ to S and its continuations to R

and set $\mu([0,\infty)a[0,\infty)a) = (0,a)(0,a)$
$\mu([0,\infty)a[0,\infty)b) = (0,a)(0,b)$



Hypothesis automaton: H_0

✗

Learning r-DDTA

Teacher returns a minimal counter-example
$w = (5, a)$

Find factorization of $w = u \cdot b \cdot v$
such that $u \in \mu(s)$ for some $s \in S \cup R$ and $u \cdot b \notin \mu(s')$ for any $s' \in S \cup R$.

Learning r-DDTA

Teacher returns a minimal counter-example
$w = (5, a)$

Find factorization of $w = u \cdot b \cdot v$
such that $u \in \mu(s)$ for some $s \in S \cup R$ and $u \cdot b \notin \mu(s')$ for any $s' \in S \cup R$.

Now, there are two cases: $s \in S$ and $s \in R$

Learning r-DDTA

Teacher returns a minimal counter-example
$w = (5, a)$

Find factorization of $w = u \cdot b \cdot v$
such that $u \in \mu(s)$ for some $s \in S \cup R$ and $u \cdot b \notin \mu(s')$ for any $s' \in S \cup R$.

Now, there are two cases: $s \in S$ and $s \in R$

For $w = (5, a)$: $u = \epsilon$, $b = (5, a)$ and $v = \epsilon$.

Learning r-DDTA

Teacher returns a minimal counter-example
$w = (5, a)$

Find factorization of $w = u \cdot b \cdot v$
such that $u \in \mu(s)$ for some $s \in S \cup R$ and $u \cdot b \notin \mu(s')$ for any $s' \in S \cup R$.

Now, there are two cases: $s \in S$ and $s \in R$

For $w = (5, a)$: $u = \epsilon$, $b = (5, a)$ and $v = \epsilon$.

Therefore $s \in S$, which represents that u is already a state in the hypothesis but b indicates that partition boundary is not correctly defined.

Learning r-DDTA

Teacher returns a minimal counter-example
$w = (5, a)$

Find factorization of $w = u \cdot b \cdot v$
such that $u \in \mu(s)$ for some $s \in S \cup R$ and $u \cdot b \notin \mu(s')$ for any $s' \in S \cup R$.

Now, there are two cases: $s \in S$ and $s \in R$

For $w = (5, a)$: $u = \epsilon$, $b = (5, a)$ and $v = \epsilon$.

Therefore $s \in S$, which represents that u is already a state in the hypothesis but b indicates that partition boundary is not correctly defined.

Hence, replace $[0, \infty)a$ with $[0, 5)a$ in the table and add $[5, \infty)a$ to R.

Learning r-DDTA

Observation Table	
T_1	ϵ
ϵ	$-$
$[0,5)a$	$+$
$[5,\infty)a$	$-$
$[0,\infty)b$	$-$
$[0,5)a([0,\infty)a$	$-$
$[0,5)a([0,\infty)b$	$-$

Is it **closed**?

Learning r-DDTA

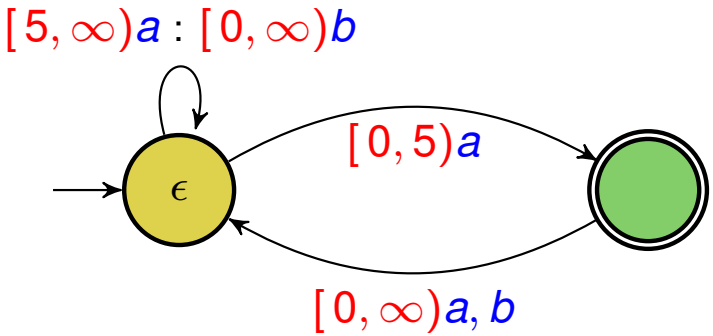
Observation Table	
T_1	ϵ
ϵ	$-$
$[0,5)a$	$+$
$[5,\infty)a$	$-$
$[0,\infty)b$	$-$
$[0,5)a([0,\infty)a$	$-$
$[0,5)a([0,\infty)b$	$-$

Is it **closed**? ✓

Learning r-DDTA

Observation Table	
T_1	ϵ
ϵ	$-$
$[0,5)a$	$+$
$[5,\infty)a$	$-$
$[0,\infty)b$	$-$
$[0,5)a([0,\infty)a$	$-$
$[0,5)a([0,\infty)b$	$-$

Is it **closed**? ✓

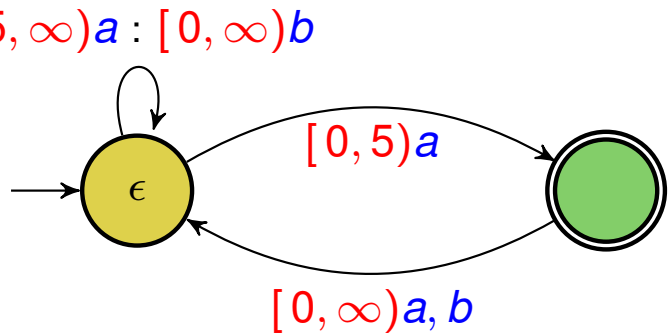


Hypothesis automaton: H_1

Learning r-DDTA

Observation Table	
T_1	ϵ
ϵ	$-$
$[0,5)a$	$+$
$[5,\infty)a$	$-$
$[0,\infty)b$	$-$
$[0,5)a([0,\infty)a$	$-$
$[0,5)a([0,\infty)b$	$-$

Is it **closed**? ✓



Hypothesis automaton: H_1

✗

Learning r-DDTA

Get a **minimal counter-example** $(w = 5, a)(0, a)$

Learning r-DDTA

Get a **minimal counter-example** $(w = 5, a)(0, a)$

$$w = u \cdot b \cdot v$$

$$u = (5, a),\ b = (0, a) \text{ and } v = \epsilon$$

$$\text{and } u = \mu([5, \infty)a)$$

Learning r-DDTA

Get a **minimal counter-example** $(w = 5, a)(0, a)$

$$w = u \cdot b \cdot v$$

$$u = (5, a),\ b = (0, a) \text{ and } v = \epsilon$$

$$\text{and } u = \mu([5, \infty)a)$$

This time we have the second case i.e. $s \in R$

Learning r-DDTA

Get a **minimal counter-example** $(w = 5, a)(0, a)$

$$w = u \cdot b \cdot v$$

$$u = (5, a),\ b = (0, a) \text{ and } v = \epsilon$$

$$\text{and } u = \mu([5, \infty)a)$$

This time we have the second case i.e. $s \in R$

This means that s is not equivalent to any state in the hypothesis automata and should form a new state.

Learning r-DDTA

Get a **minimal counter-example** $(w = 5, a)(0, a)$

$$w = u \cdot b \cdot v$$

$$u = (5, a),\ b = (0, a) \text{ and } v = \epsilon$$

$$\text{and } u = \mu([5, \infty)a)$$

This time we have the second case i.e. $s \in R$

This means that s is not equivalent to any state in the hypothesis automata and should form a new state.

Find $s' \in S$ such that $row(s) = row(s')$ Add $s \in S$ and add $b \cdot v$ and all of its suffixes to E to distinguish between s and s'.

Learning r-DDTA

Get a **minimal counter-example** $(w = 5, a)(0, a)$

$$w = u \cdot b \cdot v$$

$$u = (5, a),\ b = (0, a) \text{ and } v = \epsilon$$

$$\text{and } u = \mu([5, \infty)a)$$

This time we have the second case i.e. $s \in R$

This means that s is not equivalent to any state in the hypothesis automata and should form a new state.

Find $s' \in S$ such that $row(s) = row(s')$ Add $s \in S$ and add $b \cdot v$ and all of its suffixes to E to distinguish between s and s'.

Add $[5, \infty)a[0, \infty)a$ and $[5, \infty)a[0, \infty)b$ to R.

Learning r-DDTA

Observation Table		
T_2	ϵ	$(0, a)$
ϵ	$-$	$+$
$[0,5)a$	$+$	$-$
$[5,\infty)a$	$-$	$-$
$[0,\infty)b$	$-$	$-$
$[0,5)a[0,\infty)a$	$-$	$-$
$[0,5)a[0,\infty)b$	$-$	$-$
$[5,\infty)a[0,\infty)a$	$-$	$-$
$[5,\infty)a[0,\infty)b$	$-$	$-$

Is it **closed**?

Learning r-DDTA

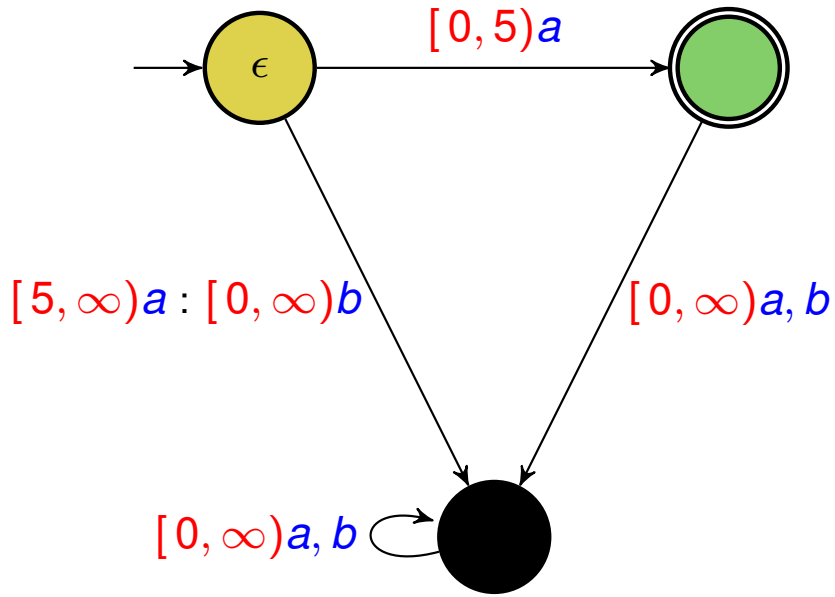
Observation Table		
T_2	ϵ	$(0, a)$
ϵ	$-$	$+$
$[0,5)a$	$+$	$-$
$[5,\infty)a$	$-$	$-$
$[0,\infty)b$	$-$	$-$
$[0,5)a[0,\infty)a$	$-$	$-$
$[0,5)a[0,\infty)b$	$-$	$-$
$[5,\infty)a[0,\infty)a$	$-$	$-$
$[5,\infty)a[0,\infty)b$	$-$	$-$

Is it **closed**? ✓

Learning r-DDTA

Observation Table		
T_2	ϵ	$(0, a)$
ϵ	$-$	$+$
$[0, 5)a$	$+$	$-$
$[5, \infty)a$	$-$	$-$
$[0, \infty)b$	$-$	$-$
$[0, 5)a[0, \infty)a$	$-$	$-$
$[0, 5)a[0, \infty)b$	$-$	$-$
$[5, \infty)a[0, \infty)a$	$-$	$-$
$[5, \infty)a[0, \infty)b$	$-$	$-$

Is it **closed**? ✓

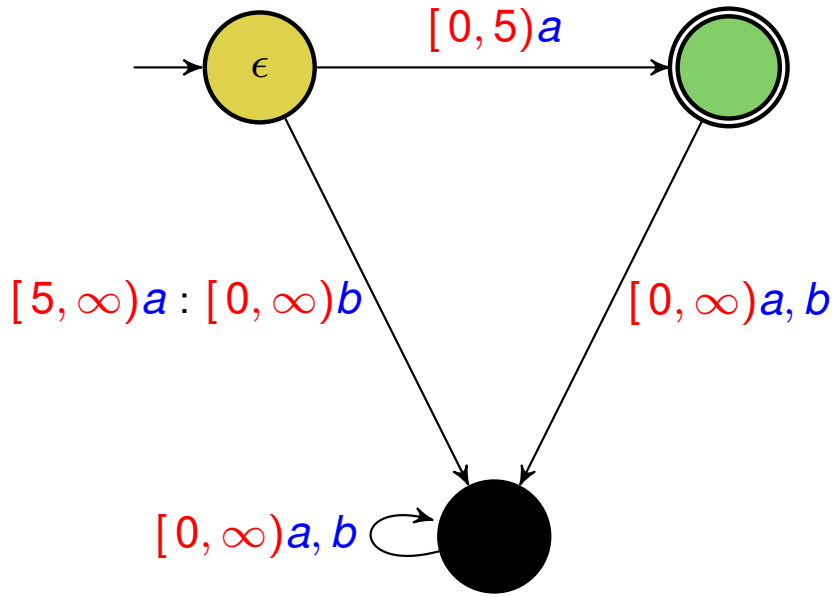


Hypothesis automaton: H_2

Learning r-DDTA

Observation Table		
T_2	ϵ	$(0, a)$
ϵ	$-$	$+$
$[0,5)a$	$+$	$-$
$[5,\infty)a$	$-$	$-$
$[0,\infty)b$	$-$	$-$
$[0,5)a[0,\infty)a$	$-$	$-$
$[0,5)a[0,\infty)b$	$-$	$-$
$[5,\infty)a[0,\infty)a$	$-$	$-$
$[5,\infty)a[0,\infty)b$	$-$	$-$

Is it **closed**? ✓



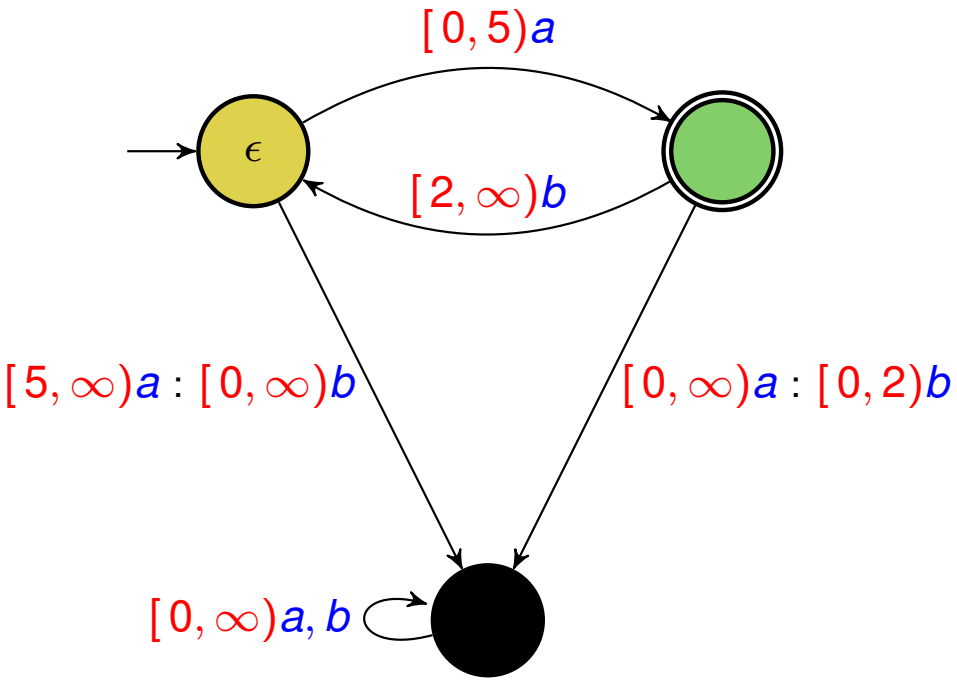
Hypothesis automaton: H_2

✗

Learning r-DDTA

In the next iteration

Observation Table		
T_3	ϵ	$(0, a)$
ϵ	$-$	$+$
$[0,5)a$	$+$	$-$
$[5,\infty)a$	$-$	$-$
$[0,\infty)b$	$-$	$-$
$[0,5)a[0,\infty)a$	$-$	$-$
$[0,5)a[0,2)b$	$-$	$-$
$[0,5)a[2,\infty)b$	$-$	$+$
$[5,\infty)a[0,\infty)a$	$-$	$-$
$[5,\infty)a[0,\infty)b$	$-$	$-$



Hypothesis automaton: H_3

✓

Canonical Automaton

Theorem. *The automaton we get from this algorithm is minimal and unique up to the renaming of states and the clock*.

Canonical Automaton

Theorem. *The automaton we get from this algorithm is minimal and unique up to the renaming of states and the clock*.

Summary:

- A learning algorithm for discrete timed automata.
- Assumption on the teacher is not drastically different from L^*.

Future work and Conclusion

Conclusion

- Learning problem and why it is useful.

Conclusion

- Learning problem and why it is useful.
- L^* with an example.

Conclusion

- Learning problem and why it is useful.
- L^* with an example.
- Discrete timed automaton.

Conclusion

- Learning problem and why it is useful.
- L^* with an example.
- Discrete timed automaton.
- Learning DiTA is equivalent to learning r-DDTA.

Conclusion

- Learning problem and why it is useful.
- L^* with an example.
- Discrete timed automaton.
- Learning DiTA is equivalent to learning r-DDTA.
- Algorithm to learn r-DDTA.

Future Work

- Implement this algorithm and run it on some examples and compare them with S. Verwer's tool.

Future Work

- Implement this algorithm and run it on some examples and compare them with S. Verwer's tool.
- Learn some real system using this algorithm.

Future Work

- Implement this algorithm and run it on some examples and compare them with S. Verwer's tool.
- Learn some real system using this algorithm.
- Generate minimal DiTA from r-DDTA.

Thank You


WHAT YOU BROUGHT TO SEMINAR AND WHAT IT SAYS ABOUT YOU:
Stuff to take notes: First year. Foolishly thinks he'll ever need notes again.
Reading material: Third year. Just here for show.
Didn't bring anything: ABD/Postdoc. Has nothing better to do.
Laptop: Young Assistant Professor. Working on three proposals at the same time.
Playing with latest Gadget/Gizmo: Full Professor. Loooves new toys.
tap tap
JORGE CHAM © 2008
WWW.PHDCOMICS.COM