Automates, algèbre, applications - AAA

CM 4

Uli Fahrenberg Sven Dziadek Philipp Schlehuber
Adrien Pommellet Etienne Renault

EPITA

S6 2022

Foreword

Program of the Course

1. CM 1 : Weighted automata — 5 May
2. TD : Weighted automata — 12 May
3. CM 2 : LTL model checking — 12 May
4. CM 3 : ω-Automata — 19 May
5. DM
6. TP : ω-Automata — 2 June
7. CM 4 : Automata learning — 16 June

Program of the Course

1. CM 1 : Weighted automata — 5 May
2. TD : Weighted automata — 12 May
3. CM 2 : LTL model checking — 12 May
4. CM 3 : ω-Automata — 19 May
5. DM
6. TP : ω-Automata — 2 June
7. CM 4 : Automata learning — 16 June

CM 4: Active Learning of Automata

Various Approaches

Verification

- Model a program as an automaton $\mathcal{M}$.
- Model a specification as a LTL formula φ.
- Check if $\mathcal{M} \models \varphi$.

Various Approaches

Verification

- Model a program as an automaton $\mathcal{M}$.
- Model a specification as a LTL formula φ.
- Check if $\mathcal{M} \models \varphi$.

Synthesis

- Consider a LTL specification φ.
- Compute an automaton $\mathcal{M}$ such that $\mathcal{M} \models \varphi$.
- Design a program based on $\mathcal{M}$.

Various Approaches

Verification

- Model a program as an automaton $\mathcal{M}$.
- Model a specification as a LTL formula φ.
- Check if $\mathcal{M} \models \varphi$.

Synthesis

- Consider a LTL specification φ.
- Compute an automaton $\mathcal{M}$ such that $\mathcal{M} \models \varphi$.
- Design a program based on $\mathcal{M}$.

Active learning

- Consider a program $\mathcal{P}$.
- Submit various queries to $\mathcal{P}$.
- Find an automaton $\mathcal{M}$ that models $\mathcal{P}$.

Various Approaches

Verification

- Model a program as an automaton $\mathcal{M}$.
- Model a specification as a LTL formula φ.
- Check if $\mathcal{M} \models \varphi$.

Synthesis

- Consider a LTL specification φ.
- Compute an automaton $\mathcal{M}$ such that $\mathcal{M} \models \varphi$.
- Design a program based on $\mathcal{M}$.

Active learning

- Consider a program $\mathcal{P}$.
- Submit various queries to $\mathcal{P}$.
- Find an automaton $\mathcal{M}$ that models $\mathcal{P}$.

Why Active Learning?

- Writing complex specifications is hard work; formalizing them using LTL makes it even harder for the uninitiated.

Why Active Learning?

- Writing complex specifications is hard work; formalizing them using LTL makes it even harder for the uninitiated.
- Understanding black box systems by intuiting rules that determine their outputs.

Why Active Learning?

- Writing complex specifications is hard work; formalizing them using LTL makes it even harder for the uninitiated.
- Understanding black box systems by intuiting rules that determine their outputs.
- A common pattern: a person asking questions, and an 'expert' (human or machine) that can answer them.

A Theoretical Active Learning Framework

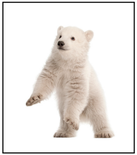
Learning Languages

Consider a (possibly infinite) language $L \subseteq \Sigma^*$ of finite words on Σ.

Learning Languages

Consider a (possibly infinite) language $L \subseteq \Sigma^*$ of finite words on Σ.

A student

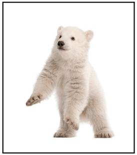


- Knows nothing of L (yet!).
- Wants to find a model (which type?) $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = L$.

Learning Languages

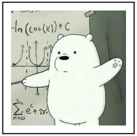
Consider a (possibly infinite) language $L \subseteq \Sigma^*$ of finite words on Σ.

A student



- Knows nothing of L (yet!).
- Wants to find a model (which type?) $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = L$.

A teacher

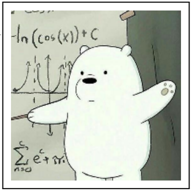


- Knows L.
- Can answer various types of queries on L, but can't explicitly give L.

Membership Queries

The student can submit membership queries.

Does $w \in L$?



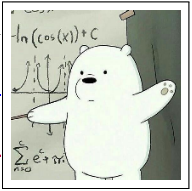
Membership Queries

The student can submit membership queries.

Does $w \in L$?

Yes.

No.



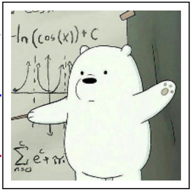
Membership Queries

The student can submit membership queries.

Does $w \in L$?

Yes.

No.

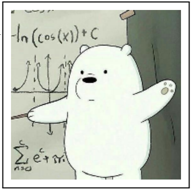


These queries can be answered by merely running the black box.

Equivalence Queries

The student can also submit equivalence queries.

Does $\mathcal{L}(\mathcal{M}) = L$?



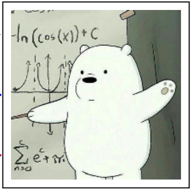
Equivalence Queries

The student can also submit equivalence queries.

Does $\mathcal{L}(\mathcal{M}) = L$?

Yes.

No. Counter-example: w.



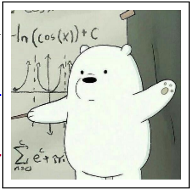
Equivalence Queries

The student can also submit equivalence queries.

Does $\mathcal{L}(\mathcal{M}) = L$?

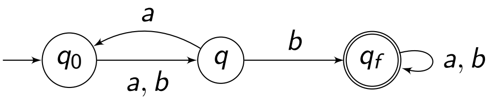
Yes.

No. Counter-example: w.



These queries are complex to answer (if it is even possible) and should be used conservatively.

A Reminder on Deterministic Finite Automata

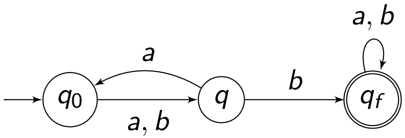


Deterministic, complete finite automaton

A DFA is a 5-uplet $A = \langle \Sigma, Q, q_0, Q, \delta \rangle$ where:

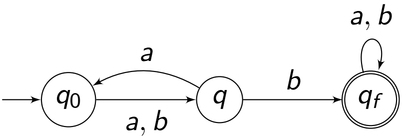
- Σ is the alphabet,
- Q a finite set of states,
- $q_0 \in Q$ a subset of initial states,
- $F \subseteq Q$ a set of accepting states,
- $\delta : Q \times \Sigma \mapsto Q$ the transition relation.

Naming States



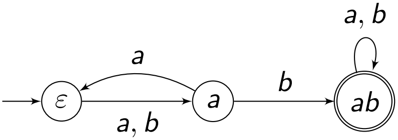
- What can we learn about $\mathcal{A}$, using nothing but membership queries?

Naming States



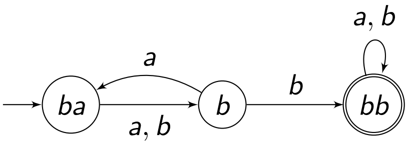
- What can we learn about $\mathcal{A}$, using nothing but membership queries?
- Note that states are being given arbitrary names.

Naming States



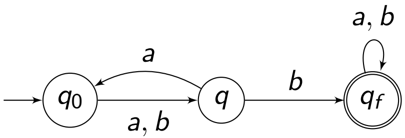
- What can we learn about $\mathcal{A}$, using nothing but membership queries?
- Note that states are being given arbitrary names.
- We could instead label them according to their path from the initial state.

Naming States



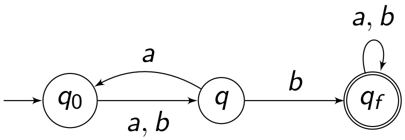
- What can we learn about $\mathcal{A}$, using nothing but membership queries?
- Note that states are being given arbitrary names.
- We could instead label them according to their path from the initial state.
- But there may be more than one such path.

Distinguishing States



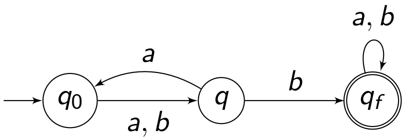
- Assume that the teacher knows $\mathcal{L}(\mathcal{A})$ but we don't.

Distinguishing States



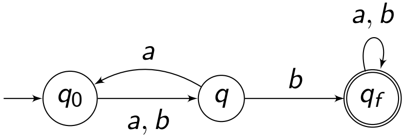
- Assume that the teacher knows $\mathcal{L}(\mathcal{A})$ but we don't.
- Note that *bb* is accepted but *ba* isn't.

Distinguishing States



- Assume that the teacher knows $\mathcal{L}(\mathcal{A})$ but we don't.
- Note that *bb* is accepted but *ba* isn't.
- Thus, they cannot lead to the same state.

Distinguishing States



- Assume that the teacher knows $\mathcal{L}(\mathcal{A})$ but we don't.
- Note that *bb* is accepted but *ba* isn't.
- Thus, they cannot lead to the same state.
- Membership queries thus allow us to infer information on states.

Indistinguishable States

Indinstinguishable states

Two states q and q' are said to be indistinguishable if they accept the same language (also written $\mathcal{L}_q(\mathcal{A}) = \mathcal{L}_{q'}(\mathcal{A})$).

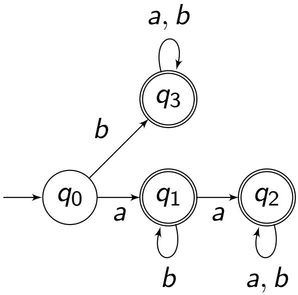
We then write $q \equiv_{\mathcal{A}} q'$.

Indistinguishable States

Indinstinguishable states

Two states q and q' are said to be indistinguishable if they accept the same language (also written $\mathcal{L}_q(\mathcal{A}) = \mathcal{L}_{q'}(\mathcal{A})$).

We then write $q \equiv_{\mathcal{A}} q'$.

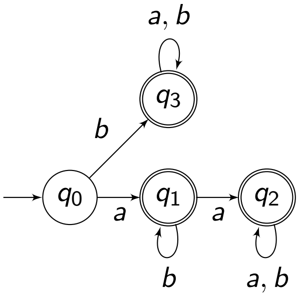


Indistinguishable States

Indinstinguishable states

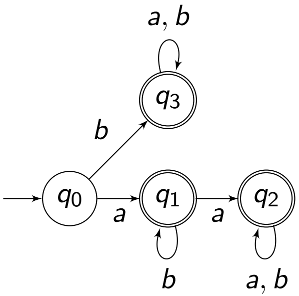
Two states q and q' are said to be indistinguishable if they accept the same language (also written $\mathcal{L}_q(\mathcal{A}) = \mathcal{L}_{q'}(\mathcal{A})$).

We then write $q \equiv_{\mathcal{A}} q'$.

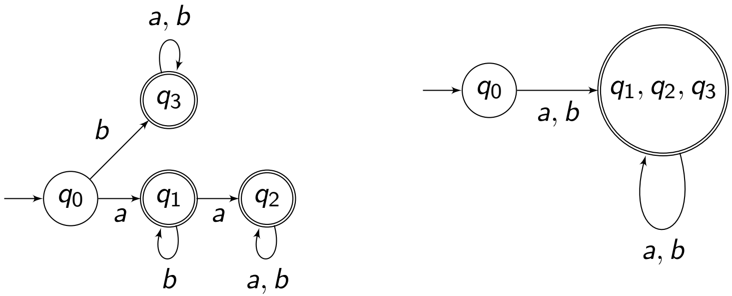


Here, $q_1 \equiv_{\mathcal{A}} q_2$ and $q_2 \equiv_{\mathcal{A}} q_3$.

A Minimal DFA



A Minimal DFA



By merging indistinguishable states, we obtain an equivalent, minimal DFA (remember Moore's algorithm).

Application to Minimization

- Formally, $\equiv_{\mathcal{A}}$ is an equivalence relation on Q.

Application to Minimization

- Formally, $\equiv_{\mathcal{A}}$ is an equivalence relation on Q.
- We partition Q according to its equivalence classes.

Application to Minimization

- Formally, $\equiv_{\mathcal{A}}$ is an equivalence relation on Q.
- We partition Q according to its equivalence classes.
- We then define an equivalent automaton whose set of states is the quotient space $Q/\equiv_{\mathcal{A}}$.

Application to Minimization

- Formally, $\equiv_{\mathcal{A}}$ is an equivalence relation on Q.
- We partition Q according to its equivalence classes.
- We then define an equivalent automaton whose set of states is the quotient space $Q/\equiv_{\mathcal{A}}$.

As a well-known consequence (already discussed in THLR):

A consequence of Myhill-Nerode's Theorem

Given a rational language L, there exists an unique (graph isomorphism notwithstanding), minimal (in terms of states) DFA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = L$, also known as the canonical DFA of L.

Distinguishing States

Distinguishable states

Two states q and q' are said to be distinguishable if there exists a word w such that q accepts w but not q' or q' accepts w but not q. We then say that w distinguishes q and q'.

Distinguishing States

Distinguishable states

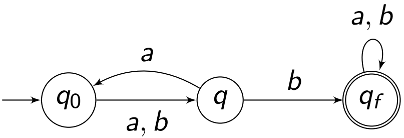
Two states q and q' are said to be distinguishable if there exists a word w such that q accepts w but not q' or q' accepts w but not q. We then say that w distinguishes q and q'.

As a consequence:

A property of the canonical DFA

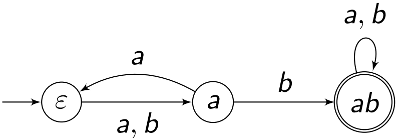
All the states of a canonical DFA are distinguishable.

Distinguishing States



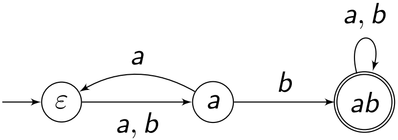
- Note that q_0 and q here are distinguishable: the former refuses the word b, the latter accepts.

Distinguishing States



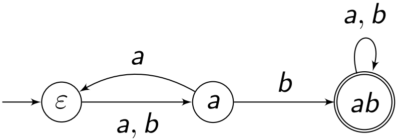
- Note that q_0 and q here are distinguishable: the former refuses the word b, the latter accepts.
- As a consequence, the automaton refuses $\varepsilon \cdot b$ and accepts $a \cdot b$.

Distinguishing States



- Note that q_0 and q here are distinguishable: the former refuses the word b, the latter accepts.
- As a consequence, the automaton refuses $\varepsilon \cdot b$ and accepts $a \cdot b$.
- More generally, if we know that u (resp. v) leads to q (resp. q'), then the membership queries $u \cdot w$ and $v \cdot w$ may allow us to prove that w distinguishes q and q' if they yield different results.

Distinguishing States



- Note that q_0 and q here are distinguishable: the former refuses the word b, the latter accepts.
- As a consequence, the automaton refuses $\varepsilon \cdot b$ and accepts $a \cdot b$.
- More generally, if we know that u (resp. v) leads to q (resp. q'), then the membership queries $u \cdot w$ and $v \cdot w$ may allow us to prove that w distinguishes q and q' if they yield different results.
- In particular, $w = \varepsilon$ can distinguish final and non-final states.

Extension to Languages

Indistiguishable words

Let $L \subseteq \Sigma^*$ be a language. Two words $u, v \in \Sigma^*$ are said to be indistinguishable if $\forall w \in \Sigma^*$, $u \cdot w \in L \iff v \cdot w \in L$. We then write $u \equiv_L v$, and $\equiv_L$ is an equivalence relation.

Extension to Languages

Indistiguishable words

Let $L \subseteq \Sigma^*$ be a language. Two words $u, v \in \Sigma^*$ are said to be indistinguishable if $\forall w \in \Sigma^*$, $u \cdot w \in L \iff v \cdot w \in L$. We then write $u \equiv_L v$, and $\equiv_L$ is an equivalence relation.

Myhill-Nerode's theorem

A language $L \subseteq \Sigma^*$ is rational if and only if its quotient space $\Sigma^* / \equiv_L$ is finite.

Extension to Languages

Indistiguishable words

Let $L \subseteq \Sigma^*$ be a language. Two words $u, v \in \Sigma^*$ are said to be indistinguishable if $\forall w \in \Sigma^*, u \cdot w \in L \iff v \cdot w \in L$. We then write $u \equiv_L v$, and $\equiv_L$ is an equivalence relation.

Myhill-Nerode's theorem

A language $L \subseteq \Sigma^*$ is rational if and only if its quotient space $\Sigma^* / \equiv_L$ is finite.

Intuitively, assuming L is rational, each class in $\Sigma^* / \equiv_L$ is equal to the set of prefixes leading to a given state in the canonical DFA of L. The number of classes is therefore equal to the size of the canonical DFA.

Our Conclusion So Far

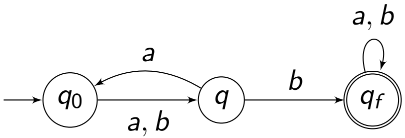
- We noted that we can finitely partition Σ^* according to $\equiv_L$: we regroup indistinguishable worlds in classes.
- Moreover, two words belonging to different classes are always distinguishable, hence admit a distinguishing word.
- We therefore expressed syntactic properties (tied to a given DFA representation) in a more generic manner that only depends on the language itself.

The L* Algorithm

Our Current Goal

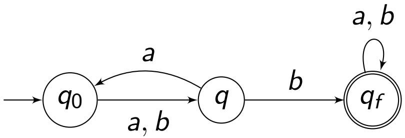
We want to design an algorithm that computes the canonical DFA of an unknown rational language L while only relying on membership and equivalence queries to a teacher that knows L.

Prefixes and Suffixes



To learn $L = \mathcal{L}(\mathcal{A})$, we will maintain two sets:

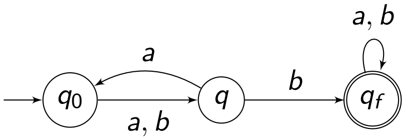
Prefixes and Suffixes



To learn $L = \mathcal{L}(\mathcal{A})$, we will maintain two sets:

- A set of prefixes P that covers every single equivalence class of $\equiv_L$ (i.e. visits every state of the canonical DFA $\mathcal{A}$) at least once.

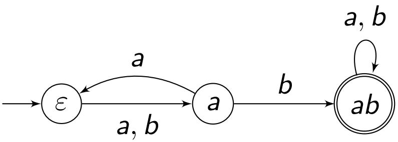
Prefixes and Suffixes



To learn $L = \mathcal{L}(\mathcal{A})$, we will maintain two sets:

- A set of prefixes P that covers every single equivalence class of $\equiv_L$ (i.e. visits every state of the canonical DFA $\mathcal{A}$) at least once.
- A set of suffixes S big enough to pairwise distinguish every pair of classes (i.e. pair of states of the canonical DFA $\mathcal{A}$) of $\equiv_L$.

Prefixes and Suffixes

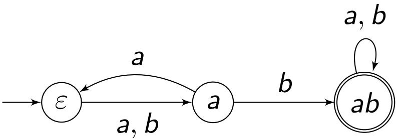


To learn $L = \mathcal{L}(\mathcal{A})$, we will maintain two sets:

- A set of prefixes P that covers every single equivalence class of $\equiv_L$ (i.e. visits every state of the canonical DFA $\mathcal{A}$) at least once.
- A set of suffixes S big enough to pairwise distinguish every pair of classes (i.e. pair of states of the canonical DFA $\mathcal{A}$) of $\equiv_L$.

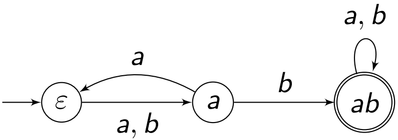
Here, $P = \{\varepsilon, a, aa, ab\}$ and $S = \{a, b\}$ match these criteria.

Distinguishing Words



Two prefixes in P may or may not lead to the same class (i.e. state).

Distinguishing Words

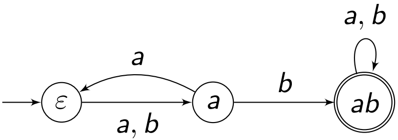


Two prefixes in P may or may not lead to the same class (i.e. state).

Assuming $S = \{s_1, \ldots, s_k\}$, we compute $\forall u \in P$ the following bit vector using only membership queries:

$$u_S = ((u \cdot s_1 \in L), \ldots, (u \cdot s_k \in L))$$

Distinguishing Words



Two prefixes in P may or may not lead to the same class (i.e. state).

$\boldsymbol{P \cdot S}$	a	b
ε	0	0
a	0	1
aa	0	0
ab	1	1

Assuming $S = \{s_1, \ldots, s_k\}$, we compute $\forall u \in P$ the following bit vector using only membership queries:

$$u_S = ((u \cdot s_1 \in L), \ldots, (u \cdot s_k \in L))$$

A New Equivalence Relation

$\boldsymbol{P \cdot S}$	a	b
ε	0	0
a	0	1
aa	0	0
ab	1	1

Here, $\varepsilon_S \neq a_S$, thus $\varepsilon \not\equiv_L a$ as at least one word in S distinguishes them. Also note that $\varepsilon_S = aa_S$, but does it imply $\varepsilon \equiv_L aa$?

A New Equivalence Relation

$\boldsymbol{P \cdot S}$	a	b
ε	0	0
a	0	1
aa	0	0
ab	1	1

Here, $\varepsilon_S \neq a_S$, thus $\varepsilon \not\equiv_L a$ as at least one word in S distinguishes them. Also note that $\varepsilon_S = aa_S$, but does it imply $\varepsilon \equiv_L aa$?

- We define a new equivalence relation $\equiv_L^S$ on P:

$$u \equiv_L^S v \iff u_S = v_S$$

- Note that $\equiv_L^S$ under-approximates $\equiv_L$ on P:

$$u \not\equiv_L^S v \implies u \not\equiv_L v$$

Approximating Indistinguishability

Here is an important consequence of this under-approximation:

Theorem

The size of the quotient space $P/\equiv_L^S$ is smaller than or equal to the size of $\Sigma^*/\equiv_L$. If it is equal, we say that (P, S) represents L.

Approximating Indistinguishability

Here is an important consequence of this under-approximation:

Theorem

The size of the quotient space $P/\equiv_L^S$ is smaller than or equal to the size of $\Sigma^*/\equiv_L$. If it is equal, we say that (P, S) represents L.

Its proof is obvious: if $\Sigma^*/\equiv_L$ is of size n, there can't be more than n pairwise distinguishable elements in $P \subseteq \Sigma^*$.

Approximating Indistinguishability

Here is an important consequence of this under-approximation:

Theorem

The size of the quotient space $P/\equiv_L^S$ is smaller than or equal to the size of $\Sigma^*/\equiv_L$. If it is equal, we say that (P, S) represents L.

Its proof is obvious: if $\Sigma^*/\equiv_L$ is of size n, there can't be more than n pairwise distinguishable elements in $P \subseteq \Sigma^*$.

As a consequence, our algorithm will rely on the following intuition: we will make (P, S) grow until we can design a model $\mathcal{M}$ of L.

Trying to Build the Canonical DFA

How can we build a DFA $\mathcal{M}$ such that (P, S) represents $\mathcal{L}(\mathcal{M})$?

States. The equivalence classes in $P/\equiv_L^S$.

Trying to Build the Canonical DFA

How can we build a DFA $\mathcal{M}$ such that (P, S) represents $\mathcal{L}(\mathcal{M})$?

States. The equivalence classes in $P/\equiv_L^S$.

Initial state. The equivalence class $[\varepsilon]_{\equiv_L^S}$, as the prefix ε naturally leads to the initial state of any DFA. Thus, we must ensure that $\varepsilon \in P$.

Trying to Build the Canonical DFA

How can we build a DFA $\mathcal{M}$ such that (P, S) represents $\mathcal{L}(\mathcal{M})$?

States. The equivalence classes in $P/\equiv_L^S$.

Initial state. The equivalence class $[\varepsilon]_{\equiv_L^S}$, as the prefix ε naturally leads to the initial state of any DFA. Thus, we must ensure that $\varepsilon \in P$.

Final state. Every $[u]_{\equiv_L^S}$ for $u \in L$; this class accepts the word ε. Thus, we must ensure that $\varepsilon \in S$.

Trying to Build the Canonical DFA

How can we build a DFA $\mathcal{M}$ such that (P, S) represents $\mathcal{L}(\mathcal{M})$?

States. The equivalence classes in $P/\equiv_L^S$.

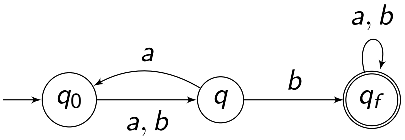
Initial state. The equivalence class $[\varepsilon]_{\equiv_L^S}$, as the prefix ε naturally leads to the initial state of any DFA. Thus, we must ensure that $\varepsilon \in P$.

Final state. Every $[u]_{\equiv_L^S}$ for $u \in L$; this class accepts the word ε. Thus, we must ensure that $\varepsilon \in S$.

Transitions. The successor of a state $[u]_{\equiv_L^S}$ according to letter $a \in \Sigma$ must naturally be $[u \cdot a]_{\equiv_L^S}$. Thus, we must also compute the bit vector $(u \cdot a)_S$ for any $u \in P$ and $a \in \Sigma$.

A Full Table

$\boldsymbol{P \cdot S}$	ε	a	b
ε	0	0	0
a	0	0	1
aa	0	0	0
ab	1	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	a	b
$\varepsilon \cdot a$	0	0	1
$\varepsilon \cdot b$	0	0	1
$a \cdot a$	0	0	0
$a \cdot b$	1	1	1
$aa \cdot a$	0	0	1
$aa \cdot b$	0	0	1
$ab \cdot a$	1	1	1
$ab \cdot b$	1	1	1



Finding the Classes

$P \cdot S$	ε	a	b
ε	0	0	0
a	0	0	1
aa	0	0	0
ab	1	1	1
$P \cdot \Sigma \cdot S$	ε	a	b
$\varepsilon \cdot a$	0	0	1
$\varepsilon \cdot b$	0	0	1
$a \cdot a$	0	0	0
$a \cdot b$	1	1	1
$aa \cdot a$	0	0	1
$aa \cdot b$	0	0	1
$ab \cdot a$	1	1	1
$ab \cdot b$	1	1	1

Finding the Classes

$P \cdot S$	ε	a	b
ε	0	0	0
a	0	0	1
aa	0	0	0
ab	1	1	1
$P \cdot \Sigma \cdot S$	ε	a	b
$\varepsilon \cdot a$	0	0	1
$\varepsilon \cdot b$	0	0	1
$a \cdot a$	0	0	0
$a \cdot b$	1	1	1
$aa \cdot a$	0	0	1
$aa \cdot b$	0	0	1
$ab \cdot a$	1	1	1
$ab \cdot b$	1	1	1

Three classes. $\{\varepsilon, aa\}$, $\{a\}$, $\{ab\}$.

Finding the Classes

$P \cdot S$	ε	a	b
ε	0	0	0
a	0	0	1
aa	0	0	0
ab	1	1	1
$P \cdot \Sigma \cdot S$	ε	a	b
$\varepsilon \cdot a$	0	0	1
$\varepsilon \cdot b$	0	0	1
$a \cdot a$	0	0	0
$a \cdot b$	1	1	1
$aa \cdot a$	0	0	1
$aa \cdot b$	0	0	1
$ab \cdot a$	1	1	1
$ab \cdot b$	1	1	1

Three classes. $\{\varepsilon, aa\}$, $\{a\}$, $\{ab\}$.

Initial class. $\{\varepsilon, aa\}$.

Finding the Classes

$P \cdot S$	ε	a	b
ε	0	0	0
a	0	0	1
aa	0	0	0
ab	1	1	1
$P \cdot \Sigma \cdot S$	ε	a	b
$\varepsilon \cdot a$	0	0	1
$\varepsilon \cdot b$	0	0	1
$a \cdot a$	0	0	0
$a \cdot b$	1	1	1
$aa \cdot a$	0	0	1
$aa \cdot b$	0	0	1
$ab \cdot a$	1	1	1
$ab \cdot b$	1	1	1

Three classes. $\{\varepsilon, aa\}$, $\{a\}$, $\{ab\}$.

Initial class. $\{\varepsilon, aa\}$.

Final class. $\{ab\}$.

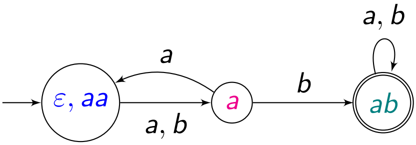
Finding the Classes

$\boldsymbol{P \cdot S}$	ε	a	b
ε	0	0	0
a	0	0	1
aa	0	0	0
ab	1	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	a	b
$\varepsilon \cdot a$	0	0	1
$\varepsilon \cdot b$	0	0	1
$a \cdot a$	0	0	0
$a \cdot b$	1	1	1
$aa \cdot a$	0	0	1
$aa \cdot b$	0	0	1
$ab \cdot a$	1	1	1
$ab \cdot b$	1	1	1

Three classes. $\{\varepsilon, aa\}$, $\{a\}$, $\{ab\}$.

Initial class. $\{\varepsilon, aa\}$.

Final class. $\{ab\}$.



A Correct Intuition

Our active learning algorithm will hinge on the following result:

Theorem

If (P, S) represents L, then $\mathcal{L}(\mathcal{M}) = L$, where $\mathcal{M}$ is the DFA built previously using the $\equiv_L^S$ relation.

A Correct Intuition

Our active learning algorithm will hinge on the following result:

Theorem

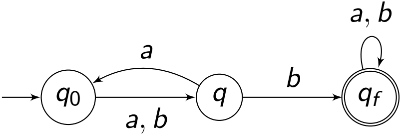
If (P, S) represents L, then $\mathcal{L}(\mathcal{M}) = L$, where $\mathcal{M}$ is the DFA built previously using the $\equiv_L^S$ relation.

The only step left is determining an iterative way to make P, S, and their matching table grow.

Closed Tables

$\boldsymbol{P \cdot S}$	ε
ε	0
a	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0
$a \cdot a$	0
$a \cdot b$	1

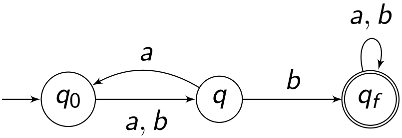
- The table associated with (P, S) is closed if any equivalence class according to $\equiv_L^S$ that appears in $P \cdot \Sigma$ also appears in P.



Closed Tables

$\boldsymbol{P \cdot S}$	ε
ε	0
a	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0
$a \cdot a$	0
$a \cdot b$	1

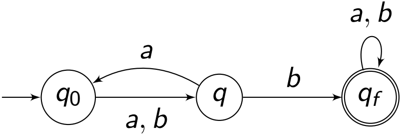
- The table associated with (P, S) is closed if any equivalence class according to $\equiv_L^S$ that appears in $P \cdot \Sigma$ also appears in P.
- Intuitively, it means the successors of P in $P \cdot \Sigma$ have already been explored.



Closed Tables

$P \cdot S$	ε
ε	0
a	0
$P \cdot \Sigma \cdot S$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0
$a \cdot a$	0
$a \cdot b$	1

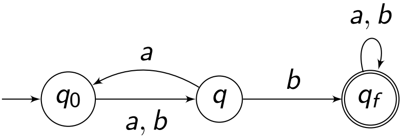
- The table associated with (P, S) is closed if any equivalence class according to $\equiv_L^S$ that appears in $P \cdot \Sigma$ also appears in P.
- Intuitively, it means the successors of P in $P \cdot \Sigma$ have already been explored.
- Here, P clearly does not cover the class of $[ab]_{\equiv_L^S}$. To make it closed, we should therefore add ab to P.



Consistent Tables

$\boldsymbol{P \cdot S}$	ε	a
ε	0	0
a	0	0
ab	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	a
$\varepsilon \cdot a$	0	0
$\varepsilon \cdot b$	0	0
$a \cdot a$	0	0
$a \cdot b$	1	1
$ab \cdot b$	1	1
$ab \cdot b$	1	1

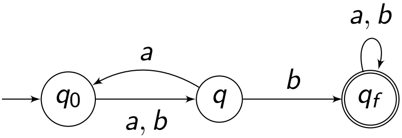
- The table associated with (P, S) is consistent if $\forall u, v \in P$, $u \equiv_L^S v$ implies that $u \cdot a \equiv_L^S v \cdot a$ for all $a \in \Sigma$.



Consistent Tables

$P \cdot S$	ε	a
ε	0	0
a	0	0
ab	1	1
$P \cdot \Sigma \cdot S$	ε	a
$\varepsilon \cdot a$	0	0
$\varepsilon \cdot b$	0	0
$a \cdot a$	0	0
$a \cdot b$	1	1
$ab \cdot b$	1	1
$ab \cdot b$	1	1

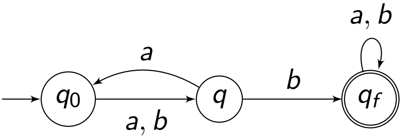
- The table associated with (P, S) is consistent if $\forall u, v \in P$, $u \equiv_L^S v$ implies that $u \cdot a \equiv_L^S v \cdot a$ for all $a \in \Sigma$.
- Intuitively, it means the successors of two equivalent states are also equivalent.



Consistent Tables

$\boldsymbol{P \cdot S}$	ε	a
ε	0	0
a	0	0
ab	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	a
$\varepsilon \cdot a$	0	0
$\varepsilon \cdot b$	0	0
$a \cdot a$	0	0
$a \cdot b$	1	1
$ab \cdot b$	1	1
$ab \cdot b$	1	1

- The table associated with (P, S) is consistent if $\forall u, v \in P$, $u \equiv_L^S v$ implies that $u \cdot a \equiv_L^S v \cdot a$ for all $a \in \Sigma$.
- Intuitively, it means the successors of two equivalent states are also equivalent.
- Here, ε and a are equivalent but their successors according to a aren't. To make the table consistent, we should therefore add b to S.



Submitting Our Results to the Teacher

- If the table is closed and consistent, then we can build an automaton $\mathcal{M}$ associated with the table and (P, S).
- If $\mathcal{L}(\mathcal{M}) \neq L$, then the teacher will at least provide a counter-example w.
- Adding w to P will ensure that our next attempt will not contradict it.
- More generally, we should add w and all its prefixes to P, to ensure every state visited by w in the canonical DFA of M is also covered by P.

The L* Algorithm At Last I

Data: two sets $P, S \subseteq \Sigma^*$, a teacher $\mathcal{T}$.
Result: a table T.

```
T ← EmptyTable;
while T is not closed or not consistent do
    T ← AskTable(P, S, 𝒯) ;                     /* Use membership queries */
    if ∃u ∈ P, ∃a ∈ Σ such that ∀v ∈ P, (T[v] ≠ T[u · a]) then
        P ← P ∪ {u · a} ;                                /* Closure issue */
    end
    if ∃u, v ∈ P, ∃a ∈ Σ, ∃s ∈ S such that (T[u] = T[v]) but
     (T[u · a][s] ≠ T[v · a][s]) then
        S ← S ∪ {a · s} ;                            /* Consistency issue */
    end
end
```

Algorithm 1: BuildTable$(P, S, \mathcal{T})$

The L* Algorithm At Last II

Data: a teacher $\mathcal{T}$ knowing a rational language L.
Result: a minimal DFA $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M}) = L$.
$P \leftarrow \{\varepsilon\}$;
$S \leftarrow \{\varepsilon\}$;
$T \leftarrow \texttt{BuildTable}(P, S, \mathcal{T})$;
$\mathcal{M} \leftarrow \texttt{BuildModel}(T)$;
while $!\texttt{EquivalenceQuery}(\mathcal{M}, \mathcal{T})$ **do**
    $c \leftarrow \texttt{CounterExample}(\mathcal{M}, \mathcal{T})$;
    $P \leftarrow P \cup \texttt{Pref}(c)$;   `/* Extending the prefixes */`
    $T \leftarrow \texttt{BuildTable}(P, S, \mathcal{T})$;
    $\mathcal{M} \leftarrow \texttt{BuildModel}(T)$;
end

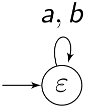
Algorithm 2: $\mathtt{L}^*(\mathcal{T})$

An Example I

$\boldsymbol{P} \cdot \boldsymbol{S}$	ε
ε	0
$\boldsymbol{P} \cdot \Sigma \cdot \boldsymbol{S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0

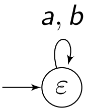
An Example I

$\boldsymbol{P \cdot S}$	ε
ε	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0

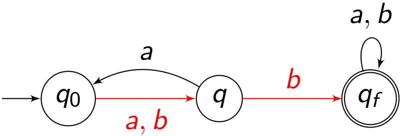


An Example I

$\boldsymbol{P \cdot S}$	ε
ε	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0

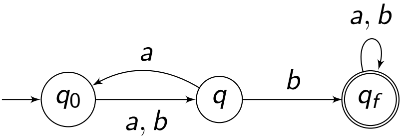


Not equivalent, counter-example ab:



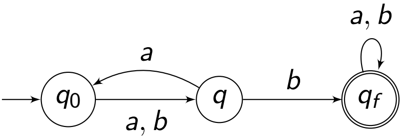
An Example II

$\boldsymbol{P \cdot S}$	ε
ε	0
a	0
ab	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0
$a \cdot a$	0
$a \cdot b$	1
$ab \cdot a$	1
$ab \cdot b$	1



An Example II

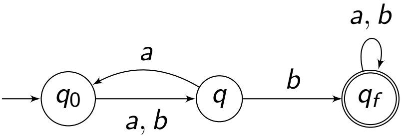
$P \cdot S$	ε
ε	0
a	0
ab	1
$P \cdot \Sigma \cdot S$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0
$a \cdot a$	0
$a \cdot b$	1
$ab \cdot a$	1
$ab \cdot b$	1



The table is not consistent:
$a \equiv_L^S \varepsilon$ but $b \not\equiv_L^S ab$.

An Example II

$\boldsymbol{P \cdot S}$	ε
ε	0
a	0
ab	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	0
$\varepsilon \cdot b$	0
$a \cdot a$	0
$a \cdot b$	1
$ab \cdot a$	1
$ab \cdot b$	1



The table is not consistent:
$a \equiv_L^S \varepsilon$ but $b \not\equiv_L^S ab$.

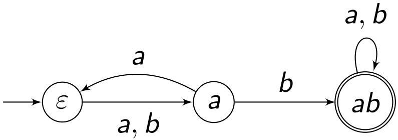
Thus we add b to S.

An Example III

$\boldsymbol{P \cdot S}$	ε	b
ε	0	0
a	0	1
ab	1	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	b
$\varepsilon \cdot a$	0	1
$\varepsilon \cdot b$	0	1
$a \cdot a$	0	0
$a \cdot b$	1	1
$ab \cdot a$	1	1
$ab \cdot b$	1	1

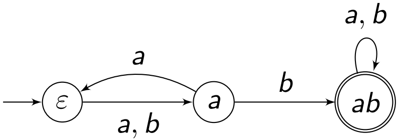
An Example III

$\boldsymbol{P \cdot S}$	ε	b
ε	0	0
a	0	1
ab	1	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	b
$\varepsilon \cdot a$	0	1
$\varepsilon \cdot b$	0	1
$a \cdot a$	0	0
$a \cdot b$	1	1
$ab \cdot a$	1	1
$ab \cdot b$	1	1

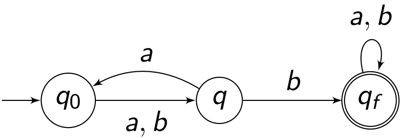


An Example III

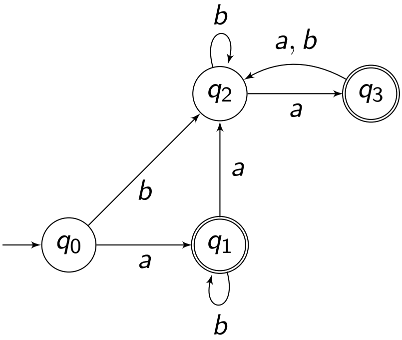
$P \cdot S$	ε	b
ε	0	0
a	0	1
ab	1	0
$P \cdot \Sigma \cdot S$	ε	b
$\varepsilon \cdot a$	0	1
$\varepsilon \cdot b$	0	1
$a \cdot a$	0	0
$a \cdot b$	1	1
$ab \cdot a$	1	1
$ab \cdot b$	1	1



Is equivalent to:



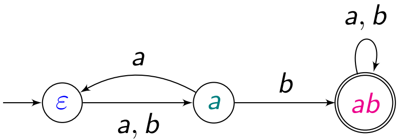
Do It Yourself



Further Optimizations

Overloading the Table

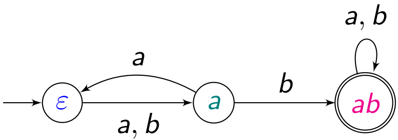
$P \cdot S$	ε	b
ε	0	0
a	0	1
aa	0	0
ab	1	0
abb	1	0



- Note that P may contain redundant prefixes that already belong to an identified equivalence class.

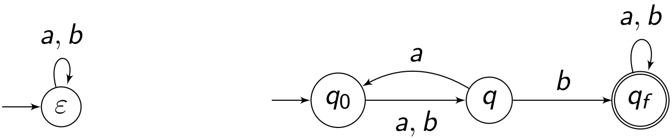
Overloading the Table

$P \cdot S$	ε	b
ε	0	0
a	0	1
aa	0	0
ab	1	0
abb	1	0



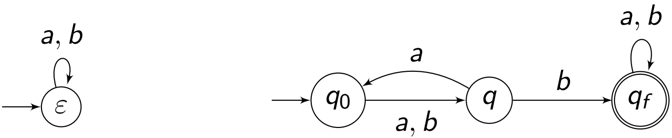
- Note that P may contain redundant prefixes that already belong to an identified equivalence class.
- Closure checks do not add such prefixes, but counter-example handling does.

Handling Counter-Examples



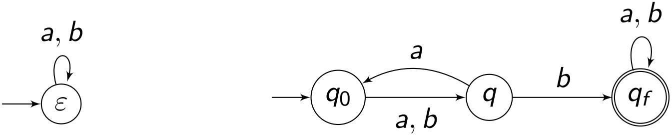
- Note that the teacher $\mathcal{T}$ must provide a counter-example, but its length is arbitrary.

Handling Counter-Examples



- Note that the teacher $\mathcal{T}$ must provide a counter-example, but its length is arbitrary.
- The word ab is a valid counter-example, but so is $(aa)^{100}ab$.

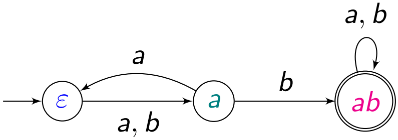
Handling Counter-Examples



- Note that the teacher $\mathcal{T}$ must provide a counter-example, but its length is arbitrary.
- The word ab is a valid counter-example, but so is $(aa)^{100}ab$.
- Since P is kept prefix-closed by design, by adding $\mathrm{Pref}((aa)^{100}ab)$ to P, we add no less than 600 lines to the table, most of which will be useless.

Simpler Prefixes

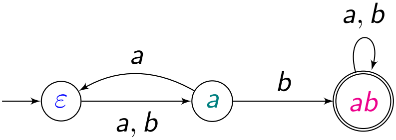
$P \cdot S$	ε	b
ε	0	0
a	0	1
ab	1	0



- We only need exactly one representative in P of each equivalence class of $\equiv_L$ to find its model.

Simpler Prefixes

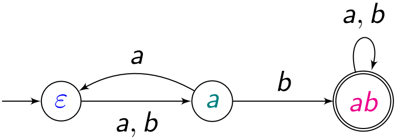
$\boldsymbol{P \cdot S}$	ε	b
ε	0	0
a	0	1
ab	1	0



- We only need exactly one representative in P of each equivalence class of $\equiv_L$ to find its model.
- In that case, the table will always be consistent.

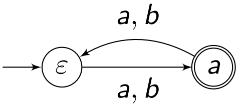
Simpler Prefixes

$P \cdot S$	ε	b
ε	0	0
a	0	1
ab	1	0



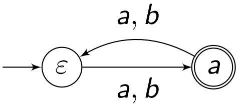
- We only need exactly one representative in P of each equivalence class of $\equiv_L$ to find its model.
- In that case, the table will always be consistent.
- We thus want to improve our use of counter-examples.

Representatives of a State



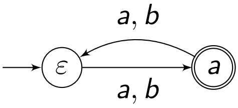
- Assume that, for a given model $\mathcal{M}$, all its prefixes in P are distinguishable (hence, each equivalence class of $\equiv_L$ has at most one representative in P).

Representatives of a State



- Assume that, for a given model $\mathcal{M}$, all its prefixes in P are distinguishable (hence, each equivalence class of $\equiv_L$ has at most one representative in P).
- For any word $w \in \Sigma^*$, we note $[w]_{\mathcal{M}}$ the only (by design) prefix u in P such that u and w lead to the same state in $\mathcal{M}$.

Representatives of a State



- Assume that, for a given model $\mathcal{M}$, all its prefixes in P are distinguishable (hence, each equivalence class of $\equiv_L$ has at most one representative in P).
- For any word $w \in \Sigma^*$, we note $[w]_{\mathcal{M}}$ the only (by design) prefix u in P such that u and w lead to the same state in $\mathcal{M}$.
- Here, $[ab]_{\mathcal{M}} = \varepsilon$ and $[a]_{\mathcal{M}} = a$.

An Important Property of Models

Theorem

Given a model $\mathcal{M}$, $u, v \in \Sigma^*$, and $x \in \Sigma$, if $\mathcal{L}(\mathcal{M}) = L$, then $[u]_{\mathcal{M}} \cdot x \cdot v \in L \iff [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

An Important Property of Models

Theorem

Given a model $\mathcal{M}$, $u, v \in \Sigma^*$, and $x \in \Sigma$, if $\mathcal{L}(\mathcal{M}) = L$, then $[u]_{\mathcal{M}} \cdot x \cdot v \in L \iff [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

The proof of this theorem is obvious: if u leads to a state q in $\mathcal{M}$, and $u \cdot x$ to a state q' that is the direct successor of q by x, then q must accept $x \cdot v$ if and only if q' accepts v.

An Important Property of Models

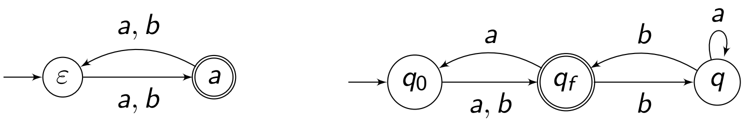
Theorem

Given a model $\mathcal{M}$, $u, v \in \Sigma^*$, and $x \in \Sigma$, if $\mathcal{L}(\mathcal{M}) = L$, then $[u]_{\mathcal{M}} \cdot x \cdot v \in L \iff [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

The proof of this theorem is obvious: if u leads to a state q in $\mathcal{M}$, and $u \cdot x$ to a state q' that is the direct successor of q by x, then q must accept $x \cdot v$ if and only if q' accepts v.

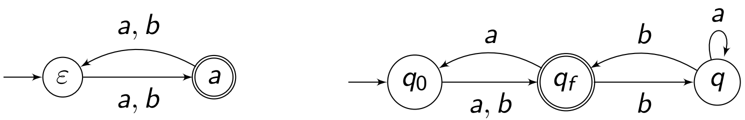
Naturally, if $\mathcal{L}(\mathcal{M}) \neq L$, then counter-examples will violate this property.

Finding Discrepancies



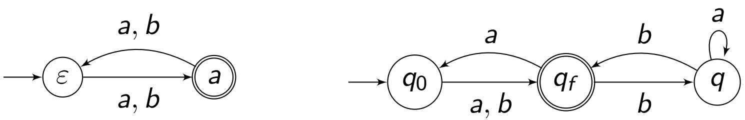
- Consider the counter-example abab to the model $\mathcal{M}$ (on the left) of L (on the right).

Finding Discrepancies



- Consider the counter-example abab to the model $\mathcal{M}$ (on the left) of L (on the right).
- $[a \cdot b]_{\mathcal{M}} \cdot ab = \varepsilon \cdot ab = ab \notin L$, but $[a]_{\mathcal{M}} \cdot b \cdot ab = abab \in L$.

Finding Discrepancies



- Consider the counter-example abab to the model $\mathcal{M}$ (on the left) of L (on the right).
- $[a \cdot b]_{\mathcal{M}} \cdot ab = \varepsilon \cdot ab = ab \notin L$, but $[a]_{\mathcal{M}} \cdot b \cdot ab = abab \in L$.
- Thus, the successor of the state $[\varepsilon]_{\equiv_L^S}$ containing ab cannot be the state $[a]_{\equiv_L^S}$.

Refining the Model

Theorem

Given a model $\mathcal{M}$ and a counter-example w proving that $\mathcal{L}(\mathcal{M}) \neq L$, there exist $u, v \in \Sigma^*$ and $x \in \Sigma$ such that $w = u \cdot a \cdot v$ and $[u]_{\mathcal{M}} \cdot x \cdot v \in L \not\Leftrightarrow [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

Refining the Model

Theorem

Given a model $\mathcal{M}$ and a counter-example w proving that $\mathcal{L}(\mathcal{M}) \neq L$, there exist $u, v \in \Sigma^*$ and $x \in \Sigma$ such that $w = u \cdot a \cdot v$ and $[u]_{\mathcal{M}} \cdot x \cdot v \in L \not\Leftrightarrow [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

- Given a counter-example of length n, we can find such a decomposition in at most $2 \times n$ membership requests.

Refining the Model

Theorem

Given a model $\mathcal{M}$ and a counter-example w proving that $\mathcal{L}(\mathcal{M}) \neq L$, there exist $u, v \in \Sigma^*$ and $x \in \Sigma$ such that $w = u \cdot a \cdot v$ and $[u]_{\mathcal{M}} \cdot x \cdot v \in L \;\not\Leftrightarrow\; [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

- Given a counter-example of length n, we can find such a decomposition in at most $2 \times n$ membership requests.
- We then add v to the set S of suffixes, ensure the table is closed, then update the model $\mathcal{M}'$.

Refining the Model

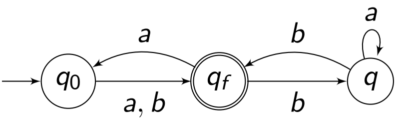
Theorem

Given a model $\mathcal{M}$ and a counter-example w proving that $\mathcal{L}(\mathcal{M}) \neq L$, there exist $u, v \in \Sigma^*$ and $x \in \Sigma$ such that $w = u \cdot a \cdot v$ and $[u]_{\mathcal{M}} \cdot x \cdot v \in L \not\Leftrightarrow [u \cdot x]_{\mathcal{M}} \cdot v \in L$.

- Given a counter-example of length n, we can find such a decomposition in at most $2 \times n$ membership requests.
- We then add v to the set S of suffixes, ensure the table is closed, then update the model $\mathcal{M}'$.
- We keep applying this procedure until w is no longer a counter-example proving that $\mathcal{L}(\mathcal{M}) \neq L$.

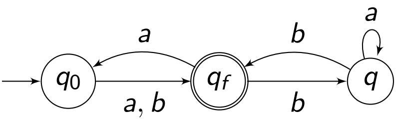
Another Example I

$\boldsymbol{P \cdot S}$	ε
ε	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	1
$\varepsilon \cdot b$	1



Another Example I

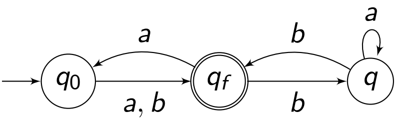
$\boldsymbol{P \cdot S}$	ε
ε	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	1
$\varepsilon \cdot b$	1



The table is not closed.

Another Example I

$\boldsymbol{P \cdot S}$	ε
ε	0
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	1
$\varepsilon \cdot b$	1



The table is not closed.

Thus we add a to P.

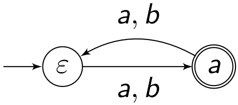
Another Example II

$\boldsymbol{P \cdot S}$	ε
ε	0
a	1

$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	1
$\varepsilon \cdot b$	1
$a \cdot a$	0
$a \cdot b$	0

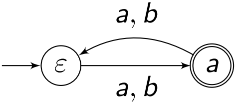
Another Example II

$\boldsymbol{P \cdot S}$	ε
ε	0
a	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	1
$\varepsilon \cdot b$	1
$a \cdot a$	0
$a \cdot b$	0

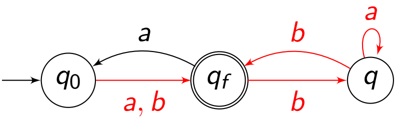


Another Example II

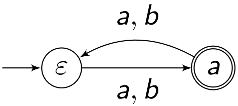
$\boldsymbol{P \cdot S}$	ε
ε	0
a	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε
$\varepsilon \cdot a$	1
$\varepsilon \cdot b$	1
$a \cdot a$	0
$a \cdot b$	0

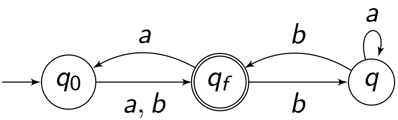


Not equivalent, counter-example $abab$:



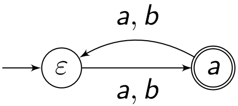
Another Example III

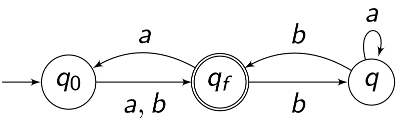




u	x	v	$[u]_{\mathcal{M}} \cdot a \cdot v \in L$	$[u \cdot a]_{\mathcal{M}} \cdot v \in L$
aba	a	ε	$aa \notin L$	$\varepsilon \notin L$
ab	a	b	$ab \notin L$	$ab \notin L$
a	b	ab	$abab \in L$	$ab \notin L$

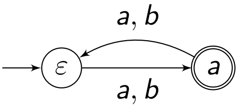
Another Example III

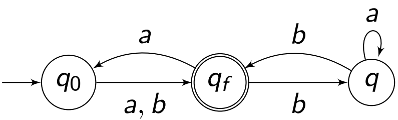




u	x	v	$[u]_{\mathcal{M}} \cdot a \cdot v \in L$	$[u \cdot a]_{\mathcal{M}} \cdot v \in L$
aba	a	ε	$aa \notin L$	$\varepsilon \notin L$
ab	a	b	$ab \notin L$	$ab \notin L$
a	b	ab	$abab \in L$	$ab \notin L$

Another Example III



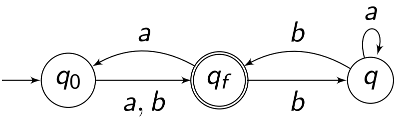


u	x	v	$[u]_{\mathcal{M}} \cdot a \cdot v \in L$	$[u \cdot a]_{\mathcal{M}} \cdot v \in L$
aba	a	ε	$aa \notin L$	$\varepsilon \notin L$
ab	a	b	$ab \notin L$	$ab \notin L$
a	b	ab	$abab \in L$	$ab \notin L$

We add ab to S.

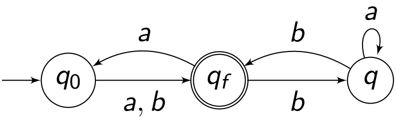
Another Example IV

$\boldsymbol{P \cdot S}$	ε	ab
ε	0	0
a	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	ab
$\varepsilon \cdot a$	1	1
$\varepsilon \cdot b$	1	1
$a \cdot a$	0	0
$a \cdot b$	0	1



Another Example IV

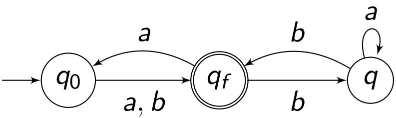
$\boldsymbol{P \cdot S}$	ε	ab
ε	0	0
a	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	ab
$\varepsilon \cdot a$	1	1
$\varepsilon \cdot b$	1	1
$a \cdot a$	0	0
$a \cdot b$	0	1



The table is not closed.

Another Example IV

$\boldsymbol{P \cdot S}$	ε	ab
ε	0	0
a	1	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	ab
$\varepsilon \cdot a$	1	1
$\varepsilon \cdot b$	1	1
$a \cdot a$	0	0
$a \cdot b$	0	1



The table is not closed.

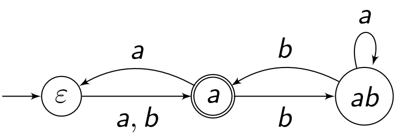
Thus we add ab to P.

Another Example V

$\boldsymbol{P \cdot S}$	ε	ab
ε	0	0
a	1	1
ab	0	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	ab
$\varepsilon \cdot a$	1	1
$\varepsilon \cdot b$	1	1
$a \cdot a$	0	0
$a \cdot b$	0	1
$ab \cdot a$	0	1
$ab \cdot b$	1	1

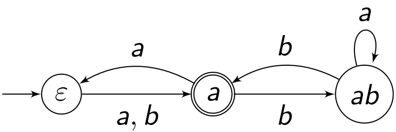
Another Example V

$\boldsymbol{P} \cdot \boldsymbol{S}$	ε	ab
ε	0	0
a	1	1
ab	0	1
$\boldsymbol{P} \cdot \boldsymbol{\Sigma} \cdot \boldsymbol{S}$	ε	ab
$\varepsilon \cdot a$	1	1
$\varepsilon \cdot b$	1	1
$a \cdot a$	0	0
$a \cdot b$	0	1
$ab \cdot a$	0	1
$ab \cdot b$	1	1

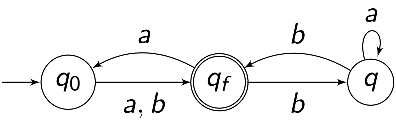


Another Example V

$\boldsymbol{P \cdot S}$	ε	ab
ε	0	0
a	1	1
ab	0	1
$\boldsymbol{P \cdot \Sigma \cdot S}$	ε	ab
$\varepsilon \cdot a$	1	1
$\varepsilon \cdot b$	1	1
$a \cdot a$	0	0
$a \cdot b$	0	1
$ab \cdot a$	0	1
$ab \cdot b$	1	1



Is equivalent to:




That's all Folks!