Logic and Theoretical Foundation of Computer Science

LaTFoCS

Pamela Fleischmann

fpa@informatik.uni-kiel.de

Winter Semester 2019

Kiel University

Dependable Systems Group

Limitations of Finite Automata

Stupidity of Finite Automata

Consider

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$$

Stupidity of Finite Automata

Consider

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$$

- L is not recognisable by a finite automata

Stupidity of Finite Automata

Consider

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$$

- L is not recognisable by a finite automata
- NFA/DFA would have to count the numbers of a

Stupidity of Finite Automata

Consider

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$$

- L is not recognisable by a finite automata
- NFA/DFA would have to count the numbers of a
- they only know where they are and where they may go

Stupidity of Finite Automata

Consider

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$$

- L is not recognisable by a finite automata
- NFA/DFA would have to count the numbers of a
- they only know where they are and where they may go
- NFA/DFA **do not** know where they come from

Stupidity of Finite Automata

Consider

$$L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$$

- L is not recognisable by a finite automata
- NFA/DFA would have to count the numbers of a
- they only know where they are and where they may go
- NFA/DFA **do not** know where they come from
- if an NFA/DFA could recognize L it would have infinitely many states (see whiteboard)

Formalisation of Non-Regularity

Theorem (Pumping Lemma for Regular Sets)

$L \subseteq \Sigma^*$ *regular set* $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:

1. $w = xyz$
2. $|y| \geq 1$
3. $|xy| \leq p$
4. $\forall i \in \mathbb{N}_0 : xy^iz \in L$

Explanations: Pumping Lemma

- DO NOT (NEVER EVER) use the Pumping Lemma for proving regularity!
- Contradiction for proving that a language is NOT REGULAR
- idea: word longer than number of states $\Rightarrow$ loop

Using the Pumping Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular
- PL $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:
  1. $w = xyz$
  2. $|y| \geq 1$
  3. $|xy| \leq p$
  4. $\forall i \in \mathbb{N}_0 : xy^i z \in L$

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular
- PL $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:
  1. $w = xyz$
  2. $|y| \geq 1$
  3. $|xy| \leq p$
  4. $\forall i \in \mathbb{N}_0 : xy^i z \in L$
- take $p \in \mathbb{N}$ (existential quantifier)

The language $L = \{a^n b^n \mid n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n \mid n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular
- PL $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:
    1. $w = xyz$
    2. $|y| \geq 1$
    3. $|xy| \leq p$
    4. $\forall i \in \mathbb{N}_0 : xy^i z \in L$
- take $p \in \mathbb{N}$ (existential quantifier)
- choose $w = a^p b^p$

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular
- PL $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:
  1. $w = xyz$
  2. $|y| \geq 1$
  3. $|xy| \leq p$
  4. $\forall i \in \mathbb{N}_0 : xy^i z \in L$
- take $p \in \mathbb{N}$ (existential quantifier)
- choose $w = a^p b^p$
  - if $w \in L$ and $|w| \geq p$ we can proceed (claim only holds for these ws)

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular
- PL $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:
  1. $w = xyz$
  2. $|y| \geq 1$
  3. $|xy| \leq p$
  4. $\forall i \in \mathbb{N}_0 : xy^i z \in L$
- take $p \in \mathbb{N}$ (existential quantifier)
- choose $w = a^p b^p$
  - if $w \in L$ and $|w| \geq p$ we can proceed (claim only holds for these ws
  - $w \in L$ immediately

The language $L = \{a^n b^n | n \in \mathbb{N}_0\}$

Lemma

The language $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}$ is not regular.

Proof with Pumping Lemma

- Contradiction: Suppose L is regular
- PL $\Rightarrow \exists p \in \mathbb{N} \forall w \in L^{\geq p} \exists x, y, z \in \Sigma^*$:
  1. $w = xyz$
  2. $|y| \geq 1$
  3. $|xy| \leq p$
  4. $\forall i \in \mathbb{N}_0 : xy^i z \in L$
- take $p \in \mathbb{N}$ (existential quantifier)
- choose $w = a^p b^p$
  - if $w \in L$ and $|w| \geq p$ we can proceed (claim only holds for these ws
  - $w \in L$ immediately
  - $|w| = 2p > p$ since $p > 0$

Cont.

- take $x, y, z \in \Sigma^*$ with

$$w = a^p b^p = xyz,\ |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 :\ xy^i z \in L$$

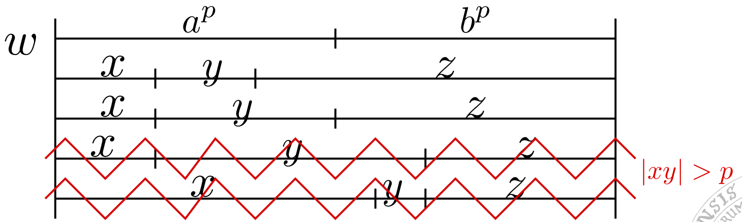
$a^p b^p = xyz$ vague $\rightsquigarrow$ how can it be achieved?

Cont.

- take $x, y, z \in \Sigma^*$ with

$$w = a^p b^p = xyz,\ |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 :\ xy^i z \in L$$

$a^p b^p = xyz$ vague $\rightsquigarrow$ how can it be achieved?

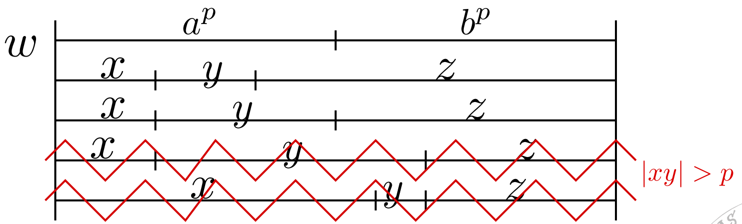


Cont.

- take $x, y, z \in \Sigma^*$ with

$$w = a^p b^p = xyz,\ |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 :\ xy^i z \in L$$

$a^p b^p = xyz$ vague $\rightsquigarrow$ how can it be achieved?



(1)-(3) fulfilled in the two cases $\Rightarrow$ we need contradiction to (4)

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

- $\exists k_1, k_2, k_3 < p : x = a^{k_1} \wedge y = a^{k_2} \wedge z = a^{k_3} b^p \wedge k_1 + k_2 + k_3 = p$

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

- $\exists k_1, k_2, k_3 < p : x = a^{k_1} \wedge y = a^{k_2} \wedge z = a^{k_3} b^p \wedge k_1 + k_2 + k_3 = p$
- $y \neq \varepsilon \Rightarrow k_2 > 0 \Rightarrow k_1 + k_3 < p$

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

- $\exists k_1, k_2, k_3 < p : x = a^{k_1} \wedge y = a^{k_2} \wedge z = a^{k_3} b^p \wedge k_1 + k_2 + k_3 = p$
- $y \neq \varepsilon \Rightarrow k_2 > 0 \Rightarrow k_1 + k_3 < p$
- Choose $i = 0$

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

- $\exists k_1, k_2, k_3 < p : x = a^{k_1} \wedge y = a^{k_2} \wedge z = a^{k_3} b^p \wedge k_1 + k_2 + k_3 = p$
- $y \neq \varepsilon \Rightarrow k_2 > 0 \Rightarrow k_1 + k_3 < p$
- Choose $i = 0$
- $\Rightarrow xy^i z = xy^0 z = a^{k_1} a^{k_3} b^p = a^{k_1 + k_3} b^p \notin L$

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

- $\exists k_1, k_2, k_3 < p : x = a^{k_1} \wedge y = a^{k_2} \wedge z = a^{k_3} b^p \wedge k_1 + k_2 + k_3 = p$
- $y \neq \varepsilon \Rightarrow k_2 > 0 \Rightarrow k_1 + k_3 < p$
- Choose $i = 0$
- $\Rightarrow xy^i z = xy^0 z = a^{k_1} a^{k_3} b^p = a^{k_1 + k_3} b^p \notin L$
- Contradiction to (4)

Cont.

- $p \in \mathbb{N}, w = a^p b^p = xyz, |y| \geq 1, |xy| \leq p, \forall i \in \mathbb{N}_0 : xy^i z \in L$

Case 1: $|xy| < p$

- $\exists k_1, k_2, k_3 < p : x = a^{k_1} \wedge y = a^{k_2} \wedge z = a^{k_3} b^p \wedge k_1 + k_2 + k_3 = p$
- $y \neq \varepsilon \Rightarrow k_2 > 0 \Rightarrow k_1 + k_3 < p$
- Choose $i = 0$
- $\Rightarrow xy^i z = xy^0 z = a^{k_1} a^{k_3} b^p = a^{k_1+k_3} b^p \notin L$
- Contradiction to (4)

Case 2: $|xy| = p$ analogously

Remarks

- we are lazy, so start finding the contradiction for $i = 0$ or small i

Remarks

- we are lazy, so start finding the contradiction for $i = 0$ or small i
- we only know that a $p \in \mathbb{N}$ exists, we do not know which specific number p is $\rightsquigarrow$ do not take a specific one!

Remarks

- we are lazy, so start finding the contradiction for $i = 0$ or small i
- we only know that a $p \in \mathbb{N}$ exists, we do not know which specific number p is $\rightsquigarrow$ do not take a specific one!
- we only know that w can be decomposed in xyz - we do not know anything how x, y, z look specifically!

Remarks

- we are lazy, so start finding the contradiction for $i = 0$ or small i
- we only know that a $p \in \mathbb{N}$ exists, we do not know which specific number p is $\rightsquigarrow$ do not take a specific one!
- we only know that w can be decomposed in xyz - we do not know anything how x, y, z look specifically!
- the difficulty is to find a *nice* w such that the cases for x, y, z are not so horrible

More non-regular languages

Every language in which data have to be stored (e.g. for counting) are not regular, e.g.

- $L_1 = \{\mathtt{a}^n\mathtt{b}^n + k \subseteq \{\mathtt{a},\mathtt{b}\}^* | n \in \mathbb{N}_0\}$ for a fixed $k \in \mathbb{N}_0$

More non-regular languages

Every language in which data have to be stored (e.g. for counting) are not regular, e.g.

- $L_1 = \{\mathsf{a}^n\mathsf{b}^n + k \subseteq \{\mathsf{a},\mathsf{b}\}^* | n \in \mathbb{N}_0\}$ for a fixed $k \in \mathbb{N}_0$
- $L_2 = \{\mathsf{a}^{n!} | \, n \in \mathbb{N}\}$

More non-regular languages

Every language in which data have to be stored (e.g. for counting) are not regular, e.g.

- $L_1 = \{\mathtt{a}^n\mathtt{b}^n + k \subseteq \{\mathtt{a},\mathtt{b}\}^* | n \in \mathbb{N}_0\}$ for a fixed $k \in \mathbb{N}_0$
- $L_2 = \{\mathtt{a}^{n!} | \, n \in \mathbb{N}\}$
- $L_3 = \{\mathtt{a}^n | \, n \in \mathbb{P}\}$

More non-regular languages

Every language in which data have to be stored (e.g. for counting) are not regular, e.g.

- $L_1 = \{\mathsf{a}^n\mathsf{b}^n + k \subseteq \{\mathsf{a},\mathsf{b}\}^* | n \in \mathbb{N}_0\}$ for a fixed $k \in \mathbb{N}_0$
- $L_2 = \{\mathsf{a}^{n!} | \, n \in \mathbb{N}\}$
- $L_3 = \{\mathsf{a}^{n} | \, n \in \mathbb{P}\}$
- $L_4 = \{\mathsf{a}^{n} | \, n \text{ is Fibonacci number}\}$

More non-regular languages

Every language in which data have to be stored (e.g. for counting) are not regular, e.g.

- $L_1 = \{\mathsf{a}^n\mathsf{b}^{n+k} \subseteq \{\mathsf{a},\mathsf{b}\}^* | n \in \mathbb{N}_0\}$ for a fixed $k \in \mathbb{N}_0$
- $L_2 = \{\mathsf{a}^{n!} | \, n \in \mathbb{N}\}$
- $L_3 = \{\mathsf{a}^n | \, n \in \mathbb{P}\}$
- $L_4 = \{\mathsf{a}^n | \, n \text{ is Fibonacci number}\}$

What do the last three languages tell us?

More non-regular languages

Every language in which data have to be stored (e.g. for counting) are not regular, e.g.

- $L_1 = \{\mathsf{a}^n\mathsf{b}^n + k \subseteq \{\mathsf{a},\mathsf{b}\}^* | n \in \mathbb{N}_0\}$ for a fixed $k \in \mathbb{N}_0$
- $L_2 = \{\mathsf{a}^{n!} | n \in \mathbb{N}\}$
- $L_3 = \{\mathsf{a}^n | n \in \mathbb{P}\}$
- $L_4 = \{\mathsf{a}^n | n$ is Fibonacci number$\}$

What do the last three languages tell us? We cannot build automata for calculating the factorial, the prime numbers, or the Fibonacci-numbers.

Myhill-Nerode

Pumping Lemma sucks?

Disadvantages of the Pumping lemma:

- only for proving unregularity (since it is an implication)

Pumping Lemma sucks?

Disadvantages of the Pumping lemma:

- only for proving unregularity (since it is an implication)
- we have to find a working w

Pumping Lemma sucks?

Disadvantages of the Pumping lemma:

- only for proving unregularity (since it is an implication)
- we have to find a working w
- we have to cover all options for xyz

Pumping Lemma sucks?

Disadvantages of the Pumping lemma:

- only for proving unregularity (since it is an implication)
- we have to find a working w
- we have to cover all options for xyz
- we have to find a working i

Pumping Lemma sucks?

Disadvantages of the Pumping lemma:

- only for proving unregularity (since it is an implication)
- we have to find a working w
- we have to cover all options for xyz
- we have to find a working i

Myhill-Nerode offer another option to prove not only unregularity but also regularity.

Myhill-Nerode-Relations

Definition

$A \subseteq \Sigma^*$ regular, $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $L(\mathcal{A}) = A$:
$\equiv_{\mathcal{A}}$ Myhill-Nerode-Relation iff

$$\forall x, y \in \Sigma^* : \; x \equiv_{\mathcal{A}} y :\Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y).$$

Myhill-Nerode-Relations

Definition

$A \subseteq \Sigma^*$ regular, $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $L(\mathcal{A}) = A$:
$\equiv_{\mathcal{A}}$ Myhill-Nerode-Relation iff

$$\forall x, y \in \Sigma^* : \; x \equiv_{\mathcal{A}} y :\Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y).$$

Some Properties:

- reflexive, symmetric, transitive

Myhill-Nerode-Relations

Definition

$A \subseteq \Sigma^*$ regular, $\mathscr{A} = (Q, \Sigma, \delta, q_0, F)$ with $L(\mathscr{A}) = A$:
$\equiv_{\mathscr{A}}$ Myhill-Nerode-Relation iff

$$\forall x, y \in \Sigma^* : \; x \equiv_{\mathscr{A}} y :\Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y).$$

Some Properties:

- reflexive, symmetric, transitive
- right congruence: $\forall x, y \in \Sigma^* \forall a \in \Sigma : \; x \equiv_{\mathscr{A}} y \Rightarrow xa \equiv_{\mathscr{A}} ya$

Myhill-Nerode-Relations

Definition

$A \subseteq \Sigma^*$ regular, $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $L(\mathcal{A}) = A$:
$\equiv_{\mathcal{A}}$ Myhill-Nerode-Relation iff

$$\forall x, y \in \Sigma^* : \; x \equiv_{\mathcal{A}} y :\Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y).$$

Some Properties:

- reflexive, symmetric, transitive
- right congruence: $\forall x, y \in \Sigma^* \forall a \in \Sigma : \; x \equiv_{\mathcal{A}} y \Rightarrow xa \equiv_{\mathcal{A}} ya$
- $\forall x, y \in \Sigma^* : \; x \equiv_{\mathcal{A}} y \Rightarrow (x \in A \Leftrightarrow y \in A)$

Myhill-Nerode-Relations

Definition

$A \subseteq \Sigma^*$ regular, $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ with $L(\mathcal{A}) = A$:

$\equiv_{\mathcal{A}}$ Myhill-Nerode-Relation iff

$$\forall x, y \in \Sigma^* : \; x \equiv_{\mathcal{A}} y :\Leftrightarrow \hat{\delta}(q_0, x) = \hat{\delta}(q_0, y).$$

Some Properties:

- reflexive, symmetric, transitive
- right congruence: $\forall x, y \in \Sigma^* \forall a \in \Sigma : \; x \equiv_{\mathcal{A}} y \Rightarrow xa \equiv_{\mathcal{A}} ya$
- $\forall x, y \in \Sigma^* : \; x \equiv_{\mathcal{A}} y \Rightarrow (x \in A \Leftrightarrow y \in A)$
- $|A/\equiv_{\mathcal{A}}| < \infty$

More Interesting Properties

- Given $\equiv$ for A with properties, $\mathscr{A}_{\equiv}$ is constructible

More Interesting Properties

- Given $\equiv$ for A with properties, $\mathscr{A}_{\equiv}$ is constructible
  - $Q = \{[x] | x \in \Sigma^*\}$
  - $q_0 = [\varepsilon]$
  - $F = \{[x] | x \in A\}$
  - $\delta([x], a) = [xa]$

More Interesting Properties

- Given $\equiv$ for A with properties, $\mathscr{A}_{\equiv}$ is constructible
  - $Q = \{[x] \mid x \in \Sigma^*\}$
  - $q_0 = [\varepsilon]$
  - $F = \{[x] \mid x \in A\}$
  - $\delta([x], a) = [xa]$
- For given $A \subseteq \Sigma^*$ regular: $\equiv \longrightarrow \mathscr{A}$ and $\mathscr{A} \longrightarrow \equiv$ are inverse to each other

More Interesting Properties

- Given $\equiv$ for A with properties, $\mathscr{A}_{\equiv}$ is constructible
  - $Q = \{[x] \mid x \in \Sigma^*\}$
  - $q_0 = [\varepsilon]$
  - $F = \{[x] \mid x \in A\}$
  - $\delta([x], a) = [xa]$
- For given $A \subseteq \Sigma^*$ regular: $\equiv \longrightarrow \mathscr{A}$ and $\mathscr{A} \longrightarrow \equiv$ are inverse to each other

Theorem

$A \subseteq \Sigma^*$ regular: $L(\mathscr{A}_{\equiv}) = A$

Myhill-Nerode Theorem

Theorem (Myhill-Nerode Theorem)

The following statements are equivalent

1. *A is regular*
2. *there exists Myhill-Nerode relation for A*
3. *$\equiv_A$ is of finite index*

Myhill-Nerode Theorem

Theorem (Myhill-Nerode Theorem)

The following statements are equivalent

1. *A is regular*
2. *there exists Myhill-Nerode relation for A*
3. *$\equiv_A$ is of finite index*

(we omit the proof)

Why did we do this? - An Application

- Theorem: equivalence to regularity

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?
- We can!

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?
- We can!
  - $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?
- We can!
  - $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$
  - $k_1 \neq k_2 \Longrightarrow a^{k_1} \neq a^{k_2}$

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?
- We can!
  - $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$
  - $k_1 \neq k_2 \Longrightarrow a^{k_1} \neq a^{k_2}$
  - $\forall k \in \mathbb{N}_0 :$ $[a^k], [a^{n+k} b^n]$ are different classes
    - append $a^n b^{n+k}$ for all $n \in \mathbb{N}_0$ resp. b^k

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?
- We can!
  - $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$
  - $k_1 \neq k_2 \Longrightarrow a^{k_1} \neq a^{k_2}$
  - $\forall k \in \mathbb{N}_0 : [a^k], [a^{n+k} b^n]$ are different classes
    - append $a^n b^{n+k}$ for all $n \in \mathbb{N}_0$ resp. b^k
  - $\Rightarrow$ infinitely many classes

Why did we do this? - An Application

- Theorem: equivalence to regularity
- Can we use this for proving or contradicting regularity of a language?
- We can!
  - $L = \{a^n b^n | n \in \mathbb{N}_0\} \subseteq \{a, b\}^*$
  - $k_1 \neq k_2 \Rightarrow a^{k_1} \neq a^{k_2}$
  - $\forall k \in \mathbb{N}_0 :\ [a^k], [a^{n+k} b^n]$ are different classes
    - append $a^n b^{n+k}$ for all $n \in \mathbb{N}_0$ resp. b^k
  - $\Rightarrow$ infinitely many classes
  - L not regular

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a, b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a, b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/{\equiv_L}|$ is finite

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a, b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/\equiv_L|$ is finite
- consider w with $|w|_a \equiv_2 0$

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a,b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/{\equiv_L}|$ is finite
- consider w with $|w|_a \equiv_2 0$
- if we append z with $|z|_a \equiv_2 0$ we are in L

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a, b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/\equiv_L|$ is finite
- consider w with $|w|_a \equiv_2 0$
- if we append z with $|z|_a \equiv_2 0$ we are in L
- if we append z with $|z|_a \equiv_2 1$ we are not in L

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a, b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/\equiv_L|$ is finite
- consider w with $|w|_a \equiv_2 0$
- if we append z with $|z|_a \equiv_2 0$ we are in L
- if we append z with $|z|_a \equiv_2 1$ we are not in L
- this holds for **all** w with an even number of a

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a,b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/{\equiv_L}|$ is finite
- consider w with $|w|_a \equiv_2 0$
- if we append z with $|z|_a \equiv_2 0$ we are in L
- if we append z with $|z|_a \equiv_2 1$ we are not in L
- this holds for **all** w with an even number of a
- $\Rightarrow$ they are all equivalent

Proving Regularity with Myhill-Nerode

Lemma

$L = \{w \in \{a, b\}^* \mid |w|_a \equiv_2 0\}$ *is regular.*

Proof:

- we have to prove that $|L/\equiv_L|$ is finite
- consider w with $|w|_a \equiv_2 0$
- if we append z with $|z|_a \equiv_2 0$ we are in L
- if we append z with $|z|_a \equiv_2 1$ we are not in L
- this holds for **all** w with an even number of a
- $\Rightarrow$ they are all equivalent
- $[aa]$ is a class representing *even number of a*

Cont.

- we have to prove that $|L/\equiv_L|$ is finite
- $[aa]$ is one class representing *even number of a*

Cont.

- we have to prove that $|L/\equiv_L|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$

Cont.

- we have to prove that $|L/{\equiv_L}|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L

Cont.

- we have to prove that $|L/{\equiv_L}|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L

Cont.

- we have to prove that $|L/{\equiv_L}|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L
- this holds for **all** w with an odd number of a

Cont.

- we have to prove that $|L/{\equiv_L}|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L
- this holds for **all** w with an odd number of a
- $\Rightarrow$ they are all equivalent

Cont.

- we have to prove that $|L/{\equiv_L}|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L
- this holds for **all** w with an odd number of a
- $\Rightarrow$ they are all equivalent
- $[a]$ is another class representing *odd number of a*

Cont.

- we have to prove that $|L/\equiv_L|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L
- this holds for **all** w with an odd number of a
- $\Rightarrow$ they are all equivalent
- $[a]$ is another class representing *odd number of a*
- there aren't more options than even or odd number of a

Cont.

- we have to prove that $|L/\equiv_L|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L
- this holds for **all** w with an odd number of a
- $\Rightarrow$ they are all equivalent
- $[a]$ is another class representing *odd number of a*
- there aren't more options than even or odd number of a
- only two classes

Cont.

- we have to prove that $|L/\equiv_L|$ is finite
- $[aa]$ is one class representing *even number of a*
- consider w with $|w|_a \equiv_2 1$
- if we append z with $|z|_a \equiv_2 0$ we are not in L
- if we append z with $|z|_a \equiv_2 1$ we are in L
- this holds for **all** w with an odd number of a
- $\Rightarrow$ they are all equivalent
- $[a]$ is another class representing *odd number of a*
- there aren't more options than even or odd number of a
- only two classes
- L is regular $\square$

Two-Way Finite Automata

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets
- Tries:
    1. Non-Determinism

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets
- Tries:
    1. Non-Determinism ☹(equivalent to DFA)

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets
- Tries:
  1. Non-Determinism ☹(equivalent to DFA)
  2. ε-Transitions

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets
- Tries:
    1. Non-Determinism ☹(equivalent to DFA)
    2. ε-Transitions ☹(equivalent to without)

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets
- Tries:
  1. Non-Determinism ☹(equivalent to DFA)
  2. ε-Transitions ☹(equivalent to without)
- What if we don't read the word linearly?

Last Try Constructing a more powerful Finite Automaton

- Status: DFA equivalent to regular expression/sets
- Tries:
  1. Non-Determinism ☹(equivalent to DFA)
  2. ε-Transitions ☹(equivalent to without)
- What if we don't read the word linearly?
- Idea: We reach in $a^n b^n$ somehow the middle and read afterwards from left and right

Ideas for 2DFA/2NFA

- reading head that moves over the word
- start marker $\vdash$ and end marker $\dashv$
- automaton can accept or reject (special states)

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_0

Example

$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_0

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \land |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{1a}

Example

$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \land |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{2a}

Example

$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{2a}

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{3a}

Example

$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{3a}

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \land |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{3a}

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{fa}

Example

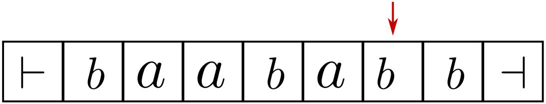
$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \land |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{1b}

Example

$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$



q_{2b}

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{2b}

Example

$$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{1b}

Example

$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{1b}

Example

$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{1b}

Example

$L = \{x \in \{a, b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{2b}

Example

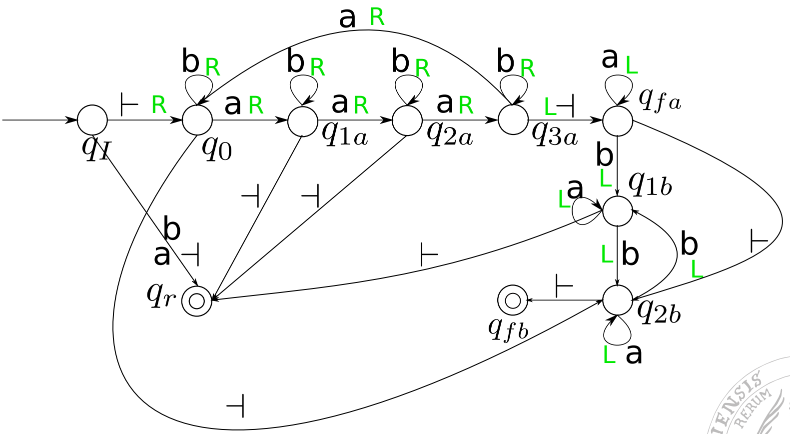
$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$

$\vdash$	b	a	a	b	a	b	b	$\dashv$

q_{fb}

Example

$$L = \{x \in \{a,b\}^* \mid |x|_a \equiv_3 0 \wedge |x|_b \equiv_2 0\}$$



Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word
- q_a accepting state

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word
- q_a accepting state
- q_r rejecting state

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word
- q_a accepting state
- q_r rejecting state
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to (Q \times \{L, R\})$ with

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word
- q_a accepting state
- q_r rejecting state
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \longrightarrow (Q \times \{L, R\})$ with
  - $\delta(q, \vdash) = (q', R)$, $\delta(q, \dashv) = (q', L)$

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word
- q_a accepting state
- q_r rejecting state
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \longrightarrow (Q \times \{L, R\})$ with
  - $\delta(q, \vdash) = (q', R)$, $\delta(q, \dashv) = (q', L)$
  - $\delta(q_a, b) = (q_a, R)$, $\delta(q_r, b) = (q_r, R)$

Formal Definition

Definition (2DFA)

$\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \delta, q_0, q_a, q_r)$ 2DFA with

- Q, Σ, q_0 as usual
- $\vdash, \dashv \notin \Sigma$ for start and end of word
- q_a accepting state
- q_r rejecting state
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \longrightarrow (Q \times \{L, R\})$ with
  - $\delta(q, \vdash) = (q', R)$, $\delta(q, \dashv) = (q', L)$
  - $\delta(q_a, b) = (q_a, R)$, $\delta(q_r, b) = (q_r, R)$
  - $\delta(q_a, \dashv) = (q_a, L)$, $\delta(q_r, \dashv) = (q_r, L)$

Configuration and Acceptance

Input: $a_0 \ldots a_{n+1} = \vdash x \dashv$

Definition

- configuration: $(q, i) \in Q \times [n+1]_0$ (state, position of read head)

Configuration and Acceptance

Input: $a_0 \dots a_{n+1} = \vdash x \dashv$

Definition

- configuration: $(q, i) \in Q \times [n+1]_0$ (state, position of read head)
- start configuration: $(q_0, 0)$

Configuration and Acceptance

Input: $a_0 \dots a_{n+1} = \vdash x \dashv$

Definition

- configuration: $(q, i) \in Q \times [n+1]_0$ (state, position of read head)
- start configuration: $(q_0, 0)$
- next configuration:

$$(p, i) \xrightarrow[x]{i} \begin{cases} (q, i-1) & \text{if } \delta(p, a_i) = (q, L), \\ (q, i+1) & \text{if } \delta(p, a_i) = (q, R). \end{cases}$$

Cont.

Definition

- extension:
  - $(p, i) \xrightarrow[x]{0} (p, i)$

Cont.

Definition

- extension:
  - $(p,i) \xrightarrow[x]{0} (p,i)$
  - $(p,i) \xrightarrow[x]{n+1} (r,k)$ if

    $\exists q \in Q \exists j \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{1} (q,j) \wedge (q,j) \xrightarrow[x]{n} (r,k)$

Cont.

Definition

- extension:
  - $(p,i) \xrightarrow[x]{0} (p,i)$
  - $(p,i) \xrightarrow[x]{n+1} (r,k)$ if

    $\exists q \in Q \exists j \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{1} (q,j) \wedge (q,j) \xrightarrow[x]{n} (r,k)$
  - $(p,i) \xrightarrow[x]{*} (q,j)$ if $\exists n \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{n} (q,j)$

Cont.

Definition

- extension:
  - $(p,i) \xrightarrow[x]{0} (p,i)$
  - $(p,i) \xrightarrow[x]{n+1} (r,k)$ if
    $\exists q \in Q \exists j \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{1} (q,j) \wedge (q,j) \xrightarrow[x]{n} (r,k)$
  - $(p,i) \xrightarrow[x]{*} (q,j)$ if $\exists n \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{n} (q,j)$
- $\mathcal{A}$ accepts x: $(q_0, 0) \xrightarrow[x]{*} (q_a, i)$ for some $i \in \mathbb{N}$

Cont.

Definition

- extension:
  - $(p,i) \xrightarrow[x]{0} (p,i)$
  - $(p,i) \xrightarrow[x]{n+1} (r,k)$ if
    $\exists q \in Q \exists j \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{1} (q,j) \wedge (q,j) \xrightarrow[x]{n} (r,k)$
  - $(p,i) \xrightarrow[x]{*} (q,j)$ if $\exists n \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{n} (q,j)$
- $\mathcal{A}$ accepts x: $(q_0, 0) \xrightarrow[x]{*} (q_a, i)$ for some $i \in \mathbb{N}$
- $\mathcal{A}$ rejects x: $(q_0, 0) \xrightarrow[x]{*} (q_r, i)$ for some $i \in \mathbb{N}$

Cont.

Definition

- extension:
  - $(p,i) \xrightarrow[x]{0} (p,i)$
  - $(p,i) \xrightarrow[x]{n+1} (r,k)$ if
    $\exists q \in Q \exists j \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{1} (q,j) \wedge (q,j) \xrightarrow[x]{n} (r,k)$
  - $(p,i) \xrightarrow[x]{*} (q,j)$ if $\exists n \in \mathbb{N}_0 : (p,i) \xrightarrow[x]{n} (q,j)$
- $\mathscr{A}$ accepts x: $(q_0, 0) \xrightarrow[x]{*} (q_a, i)$ for some $i \in \mathbb{N}$
- $\mathscr{A}$ rejects x: $(q_0, 0) \xrightarrow[x]{*} (q_r, i)$ for some $i \in \mathbb{N}$
- $\mathscr{A}$ loops on x: $\forall i \in \mathbb{N} : (q_0, 0) \xrightarrow[x]{*} (q, i) \Rightarrow q \notin \{q_a, q_f\}$

2DFAs and Regular Sets

Idea for the Equivalence

- we shall read a word w

Idea for the Equivalence

- we shall read a word w
- assume $w = xz$

Idea for the Equivalence

- we shall read a word w
- assume $w = xz$
- how much information can we carry when we pass from x to z?

Idea for the Equivalence

- we shall read a word w
- assume $w = xz$
- how much information can we carry when we pass from x to z?
- determinism $\Rightarrow$ each pass from x to z moves the automaton in the same state p and from z to x in the same state q

Idea for the Equivalence

- we shall read a word w
- assume $w = xz$
- how much information can we carry when we pass from x to z?
- determinism $\Rightarrow$ each pass from x to z moves the automaton in the same state p and from z to x in the same state q
- this following state is independent from z resp. from x!

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \to (Q \cup \{\perp\})$ with
  - $\perp$ for undefined

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \to (Q \cup \{\perp\})$ with
  - $\perp$ for undefined
  - $T_x(\bullet)$ automaton's state on first time passing x

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \to (Q \cup \{\perp\})$ with
  - $\perp$ for undefined
  - $T_x(\bullet)$ automaton's state on first time passing x
  - q state when automaton reaches x from the right

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \longrightarrow (Q \cup \{\perp\})$ with
  - $\perp$ for undefined
  - $T_x(\bullet)$ automaton's state on first time passing x
  - q state when automaton reaches x from the right
  - $p =: T_x(q)$ automaton's state when automaton leaves x

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \to (Q \cup \{\perp\})$ with
  - $\perp$ for undefined
  - $T_x(\bullet)$ automaton's state on first time passing x
  - q state when automaton reaches x from the right
  - $p =: T_x(q)$ automaton's state when automaton leaves x

Notice:

- only finitely many options for $T_x(\bullet)$

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \to (Q \cup \{\perp\})$ with
  - $\perp$ for undefined
  - $T_x(\bullet)$ automaton's state on first time passing x
  - q state when automaton reaches x from the right
  - $p =: T_x(q)$ automaton's state when automaton leaves x

Notice:

- only finitely many options for $T_x(\bullet)$
- $T_x(\bullet)$ independent of z

Equivalence in more details

- $T_x : (Q \cup \{\bullet\}) \to (Q \cup \{\perp\})$ with
  - $\perp$ for undefined
  - $T_x(\bullet)$ automaton's state on first time passing x
  - q state when automaton reaches x from the right
  - $p =: T_x(q)$ automaton's state when automaton leaves x

Notice:

- only finitely many options for $T_x(\bullet)$
- $T_x(\bullet)$ independent of z
- $T_x(q)$ independent of z

T_x

- build $T_x(q)$ for all $q \in Q \cup \{\bullet\}$

T_x

- build $T_x(q)$ for all $q \in Q \cup \{\bullet\}$
- T_x is finite mapping

T_x

- build $T_x(q)$ for all $q \in Q \cup \{\bullet\}$
- T_x is finite mapping
- $T_x = T_y \Rightarrow x, y$ are not distinguishable

T_x

- build $T_x(q)$ for all $q \in Q \cup \{\bullet\}$
- T_x is finite mapping
- $T_x = T_y \Rightarrow x, y$ are not distinguishable
- only $(n+1)^{n+1}$ different possibilities for this mapping

T_x

- build $T_x(q)$ for all $q \in Q \cup \{\bullet\}$
- T_x is finite mapping
- $T_x = T_y \Rightarrow x, y$ are not distinguishable
- only $(n+1)^{n+1}$ different possibilities for this mapping
- infinitely many words $\Rightarrow$ words do have the same mapping

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

- every DFA is a special 2DFA

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

- every DFA is a special 2DFA
- $\mathscr{A}$ 2DFA $\Rightarrow$

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

- every DFA is a special 2DFA
- $\mathscr{A}$ 2DFA $\Rightarrow$
  - $T_x = T_y \Leftrightarrow (\forall z \in \Sigma^* : \ xz \in L(\mathscr{A}) \Leftrightarrow yz \in L(\mathscr{A}))$
    (x, y are indistinguishable)

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

- every DFA is a special 2DFA
- $\mathscr{A}$ 2DFA $\Rightarrow$
  - $T_x = T_y \Leftrightarrow (\forall z \in \Sigma^* : xz \in L(\mathscr{A}) \Leftrightarrow yz \in L(\mathscr{A}))$
    (x, y are indistinguishable)
  - define $x \equiv_{L(\mathscr{A})} y$ by $T_x = T_y$

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

- every DFA is a special 2DFA
- $\mathcal{A}$ 2DFA $\Rightarrow$
  - $T_x = T_y \Leftrightarrow (\forall z \in \Sigma^* : xz \in L(\mathcal{A}) \Leftrightarrow yz \in L(\mathcal{A}))$ (x, y are indistinguishable)
  - define $x \equiv_{L(\mathcal{A})} y$ by $T_x = T_y$
  - only finitely many mappings $\Rightarrow$ index is finite

2DFA equivalent to DFA

Theorem

For each 2DFA exists equivalent DFA and vice versa.

Proof:

- every DFA is a special 2DFA
- $\mathscr{A}$ 2DFA $\Rightarrow$
  - $T_x = T_y \Leftrightarrow (\forall z \in \Sigma^* : xz \in L(\mathscr{A}) \Leftrightarrow yz \in L(\mathscr{A}))$
    (x, y are indistinguishable)
  - define $x \equiv_{L(\mathscr{A})} y$ by $T_x = T_y$
  - only finitely many mappings $\Rightarrow$ index is finite
  - $L(\mathscr{A})$ is regular