Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik Übung

2. Übungstermin · 5. November 2019

Franziska Wegner

INSTITUT FÜR THEORETISCHE INFORMATIK · LEHRSTUHL ALGORITHMIK

Gliederung

Karlsruher Institut für Technologie

Inhalt

- Pumping-Lemma
- Bestimmung eines regulären Ausdrucks
- Minimierung von Automaten
  - Entfernen überflüssiger Zustände
  - Äquivalenzklassenkonstruktion
- Nerode-Relation
- Cantors 2. Diagonalargument

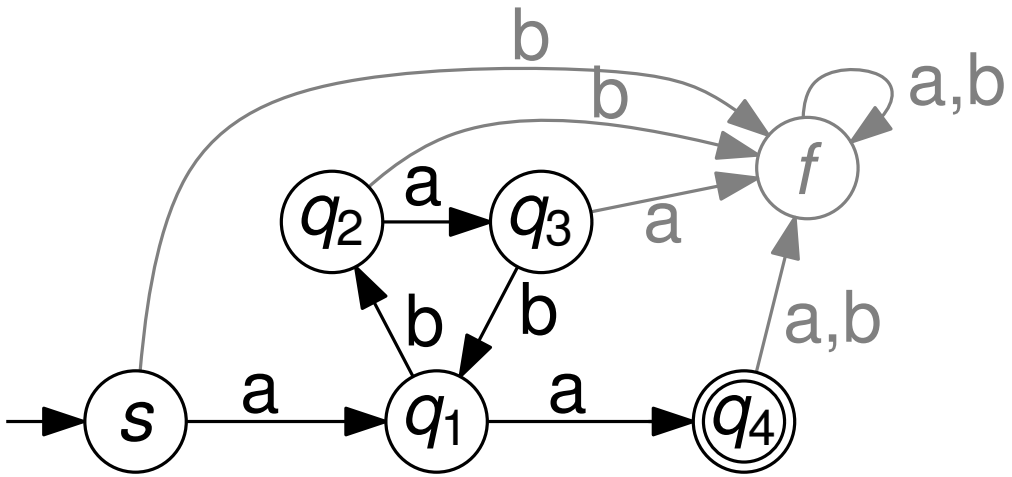
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

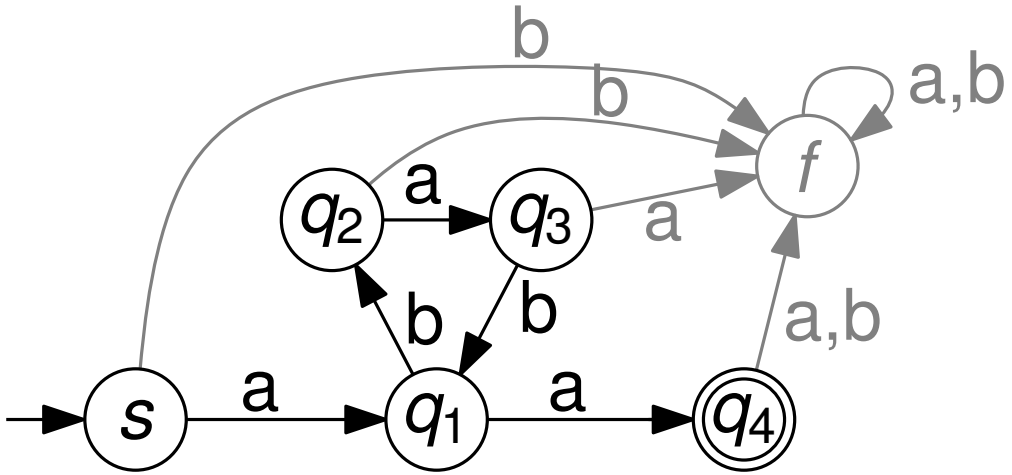
Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

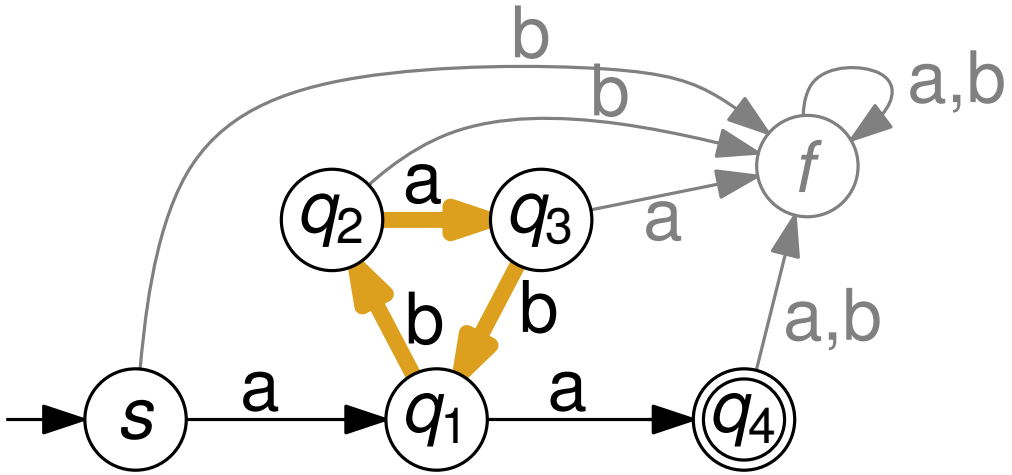
$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.

Egal, wie viele Zustände für $\mathcal{A}$ verwendet werden, für jedes Wort $w \in L$, das mehr Zeichen hat als $\mathcal{A}$ Zustände, gilt:

Während der Abarbeitung von w durchläuft man einen Zyklus $\mathcal{Z}$ in $\mathcal{A}$.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.

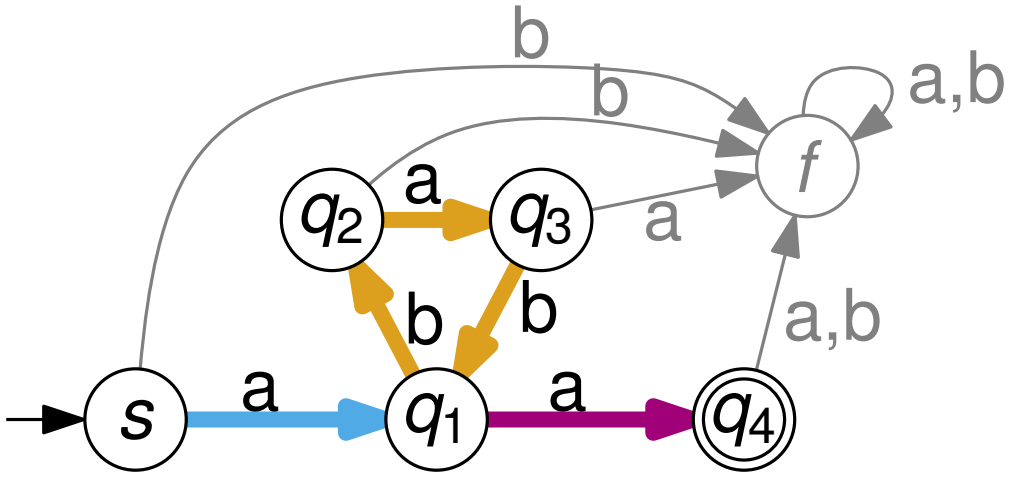
Egal, wie viele Zustände für $\mathcal{A}$ verwendet werden, für jedes Wort $w \in L$, das mehr Zeichen hat als $\mathcal{A}$ Zustände, gilt:

Während der Abarbeitung von w durchläuft man einen Zyklus $\mathcal{Z}$ in $\mathcal{A}$.

Sei

- u das Teilwort von w, das vor $\mathcal{Z}$,
- v das Teilwort von w, das in $\mathcal{Z}$, und
- x das Teilwort von w, das nach $\mathcal{Z}$

abgearbeitet wird.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, sodass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.

Egal, wie viele Zustände für $\mathcal{A}$ verwendet werden, für jedes Wort $w \in L$, das mehr Zeichen hat als $\mathcal{A}$ Zustände, gilt:

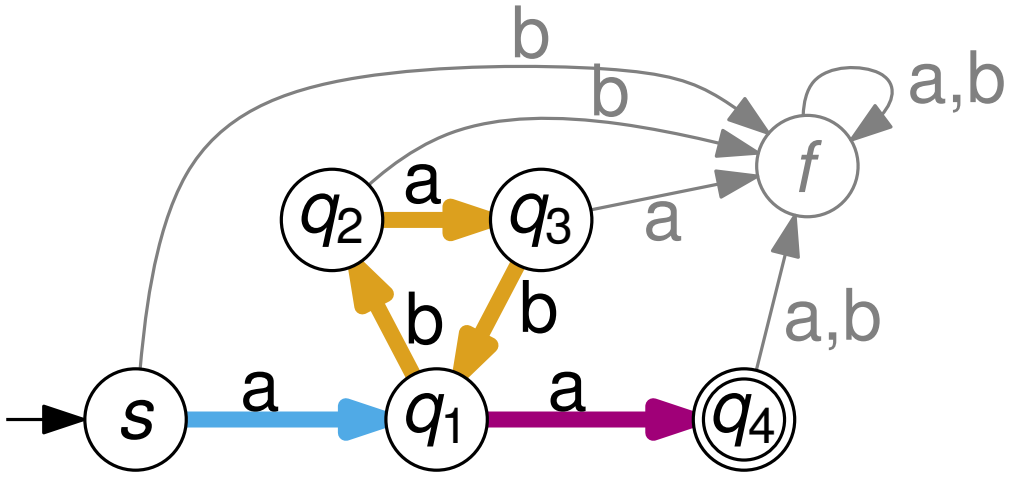
Während der Abarbeitung von w durchläuft man einen Zyklus $\mathcal{Z}$ in $\mathcal{A}$.

Sei

- u das Teilwort von w, das vor $\mathcal{Z}$,
- v das Teilwort von w, das in $\mathcal{Z}$, und
- x das Teilwort von w, das nach $\mathcal{Z}$

abgearbeitet wird.

→ $uv^i x$ (mit $i \in \mathbb{N}_0$) ist auch in L enthalten.
Durchlaufe $\mathcal{Z}$ entsprechend häufig.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Umkehrung der Aussage des Pumping-Lemmas

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Umkehrung der Aussage des Pumping-Lemmas

für alle $\forall n \in \mathbb{N}$

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Umkehrung der Aussage des Pumping-Lemmas

für alle $\forall n \in \mathbb{N}$

existiert $\exists w \in L$ mit $|w| > n$

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Umkehrung der Aussage des Pumping-Lemmas

für alle $\forall n \in \mathbb{N}$

existiert $\exists w \in L$ mit $|w| > n$

für alle $\forall u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Umkehrung der Aussage des Pumping-Lemmas

für alle $\forall n \in \mathbb{N}$

existiert $\exists w \in L$ mit $|w| > n$

für alle $\forall u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

existiert $\exists i \in \mathbb{N}_0$:

Pumping-Lemma

Aussage des Pumping-Lemmas

existiert $\exists n \in \mathbb{N}$

für alle $\forall w \in L$ mit $|w| > n$

existiert $\exists u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

für alle $\forall i \in \mathbb{N}_0$:

gilt $uv^i x \in L$

Umkehrung der Aussage des Pumping-Lemmas

für alle $\forall n \in \mathbb{N}$

existiert $\exists w \in L$ mit $|w| > n$

für alle $\forall u, v, x \in \Sigma^*$ mit $w = uvx, |uv| \leq n, v \neq \varepsilon$

existiert $\exists i \in \mathbb{N}_0$:

gilt $uv^i x \notin L$

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(a) $L_1 = \{a^{2i} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$

(b) $L_2 = \{a^{i^2} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$

(c) $L_3 = \{w \in \{0, 1\}^* \mid w$ enthält das Teilwort 000 genauso häufig wie das Teilwort 111$\}$

(d) $L_4 = \{w \in \{0, 1\}^* \mid$ auf jedes Symbol 0 in w folgt das Symbol 1 und nach maximal dreimal 1 in w folgt das Symbol 0. $\}$

(e) $L_5 = \{a\}$

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

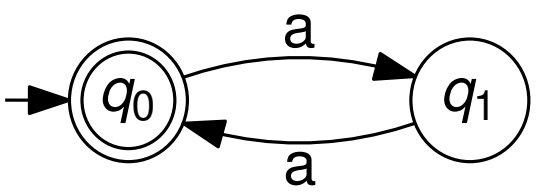
(a) $L_1 = \{a^{2i} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(a) $L_1 = \{a^{2i} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$

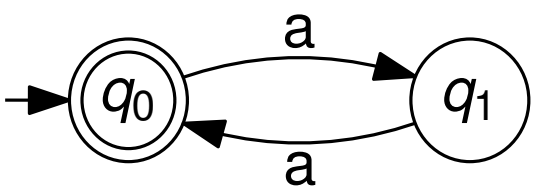


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(a) $L_1 = \{a^{2i} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$ ✓



Aussage des Pumping-Lemmas ist erfüllt:

"$\exists$" Wähle $n = 2$.

"$\forall$" Betrachte beliebiges $w \in L$ mit $|w| > 2$.

$\Rightarrow w = a^{2j}$ mit $j \geq 2$.

"$\exists$" Wähle Zerlegung $w = uvx$ mit $u = \varepsilon$, $v = aa$, $x = a^{2(j-1)}$.

"$\forall$" Für alle $i \in \mathbb{N}_0$: $uv^ix = a^{2i}a^{2(j-1)} = a^{2(i+j-1)} \in L$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(b) $L_2 = \{a^{i^2} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(b) $L_2 = \{a^{i^2} \in \{a,b\}^* \mid i \in \mathbb{N}_0\}$

Zeige Umkehrung des Aussage des Pumping-Lemmas:

"$\forall$" Betrachte beliebiges $n \in \mathbb{N}$. Sei $m = n + 1$.

"$\exists$" Wähle das Wort $w = a^{m^2} \in L_2$. Es gilt $|w| \geq m > n$.

"$\forall$" Betrachte beliebige Zerlegung $w = u\,v\,x \in L_2$ mit $|uv| \leq n$ und $v \neq \varepsilon$.

Beschreibe alle so möglichen Zerlegungen als: $\underset{u}{a^p}\ \underset{v}{a^q}\ \underset{x}{a^r}$

mit $p + q + r = m^2$, $p + q \leq n$ und $1 \leq q \leq n$.

"$\exists$" Wähle $i = 2$. Zeige: $u\,v^2\,x \notin L_2$.

$$|a^{m^2}| < |uv^2x| = |a^p a^{2q} a^r| = |a^{m^2} a^q| \leq |a^{m^2} a^n| < |a^{m^2} a^{2m} a| = |a^{(m+1)^2}|$$

Es gibt kein $j \in \mathbb{N}$, sodass $|\, uv^2x \,| = a^{j^2}$.

✗

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(c) $L_3 = \{w \in \{0,1\}^* \mid w$ enthält das Teilwort 000 genauso häufig wie das Teilwort 111$\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(c) $L_3 = \{w \in \{0,1\}^* \mid w$ enthält das Teilwort 000 genauso häufig wie das Teilwort 111$\}$

Zeige Umkehrung des Aussage des Pumping-Lemmas:

"$\forall$" Betrachte beliebiges $n \in \mathbb{N}$.

"$\exists$" Wähle das Wort $w = (000)^n(111)^n \in L_3$. Es gilt $|w| > n$.

"$\forall$" Betrachte beliebige Zerlegung $w = u\,v\,x \in L_3$ mit $|uv| \leq n$ und $v \neq \varepsilon$.

Beschreibe alle so möglichen Zerlegungen als: $\underbrace{0^p}_{u}\ \underbrace{0^q}_{v}\ \underbrace{0^{3n-p-q}\ (111)^n}_{x}$

mit $p + q \leq n$ und $1 \leq q \leq n$.

"$\exists$" Wähle $i = 2$. Zeige: $u\,v^2\,x \notin L_3$.

$0^p\ 0^{2q}\ 0^{3n-p-q}\ (111)^n = 0^{3n+q}\ (111)^n \notin L_3$

✗

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

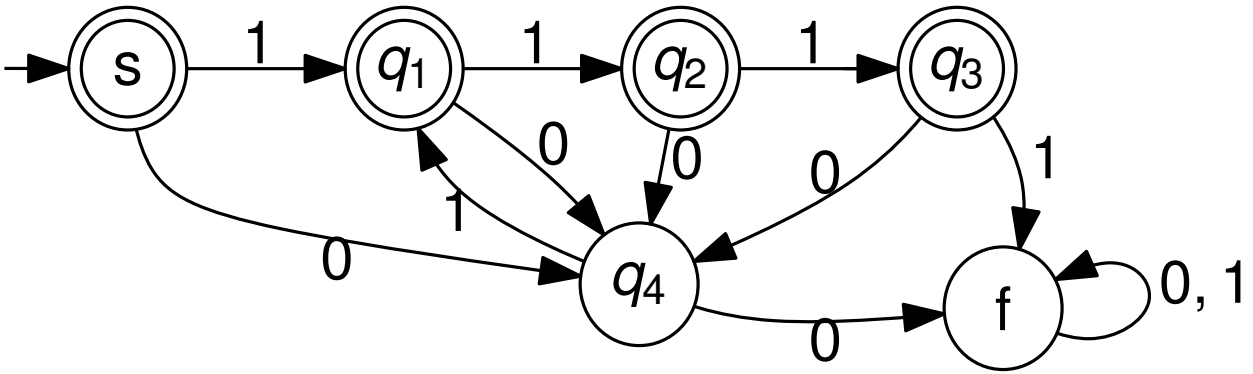
(d) $L_4 = \{w \in \{0,1\}^* \mid$ auf jedes Symbol 0 in w folgt das Symbol 1 und nach maximal dreimal 1 in w folgt das Symbol 0. $\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(d) $L_4 = \{w \in \{0,1\}^* \mid$ auf jedes Symbol 0 in w folgt das Symbol 1 und nach maximal dreimal 1 in w folgt das Symbol 0. $\}$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

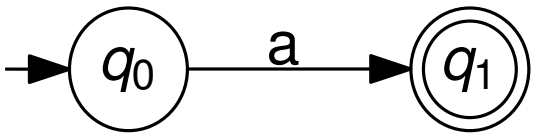
(e) $L_5 = \{a\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(e) $L_5 = \{a\}$

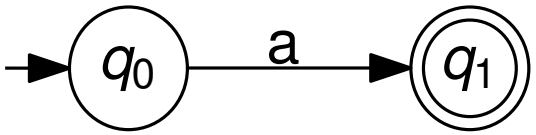


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(e) $L_5 = \{a\}$

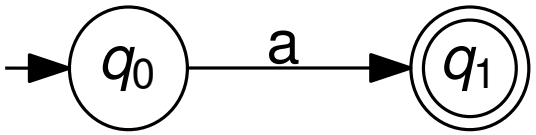


kein Zyklus $\rightarrow$ Aussage des Pumping-Lemmas nicht erfüllt?

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(e) $L_5 = \{a\}$



kein Zyklus $\rightarrow$ Aussage des Pumping-Lemmas nicht erfüllt?

✓

Doch:

"$\exists$" Wähle $n = 1$.

"$\forall$" Betrachte beliebiges $w \in L$ mit $|w| > 1$.

$\Rightarrow$ gibt es nicht!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(a) $L_1 = \{a^{2i} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$ ✓

(b) $L_2 = \{a^{i^2} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$ ✗

(c) $L_3 = \{w \in \{0, 1\}^* \mid w$ enthält das Teilwort 000 genauso häufig wie das Teilwort 111$\}$ ✗

(d) $L_4 = \{w \in \{0, 1\}^* \mid$ auf jedes Symbol 0 in w folgt das Symbol 1 und nach maximal dreimal 1 in w folgt das Symbol 0. $\}$ ✓

(e) $L_5 = \{a\}$ ✓

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumping-Lemma

Zeigen oder widerlegen Sie, dass folgende Sprachen regulär sind.

(a) $L_1 = \{a^{2i} \in \{a, b\}^* \mid i \in \mathbb{N}_0\}$ ✓

Mit dem Pumping-Lemma kann man nicht zeigen, dass eine Sprache regulär ist, man kann es höchstens widerlegen!

wie

und

nach maximal dreimal 1 in w folgt das Symbol 0. $\}$ ✓

(e) $L_5 = \{a\}$ ✓

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t}\right)$$

= Sprache, die jedes Wort w enthält, für das gilt: Wenn man in $\mathcal{A}$ im Zustand q_r startet, w abarbeitet und dabei nur die Zustände $q_1, \ldots, q_{i+1}$ benutzt, endet man in Zustand q_t.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

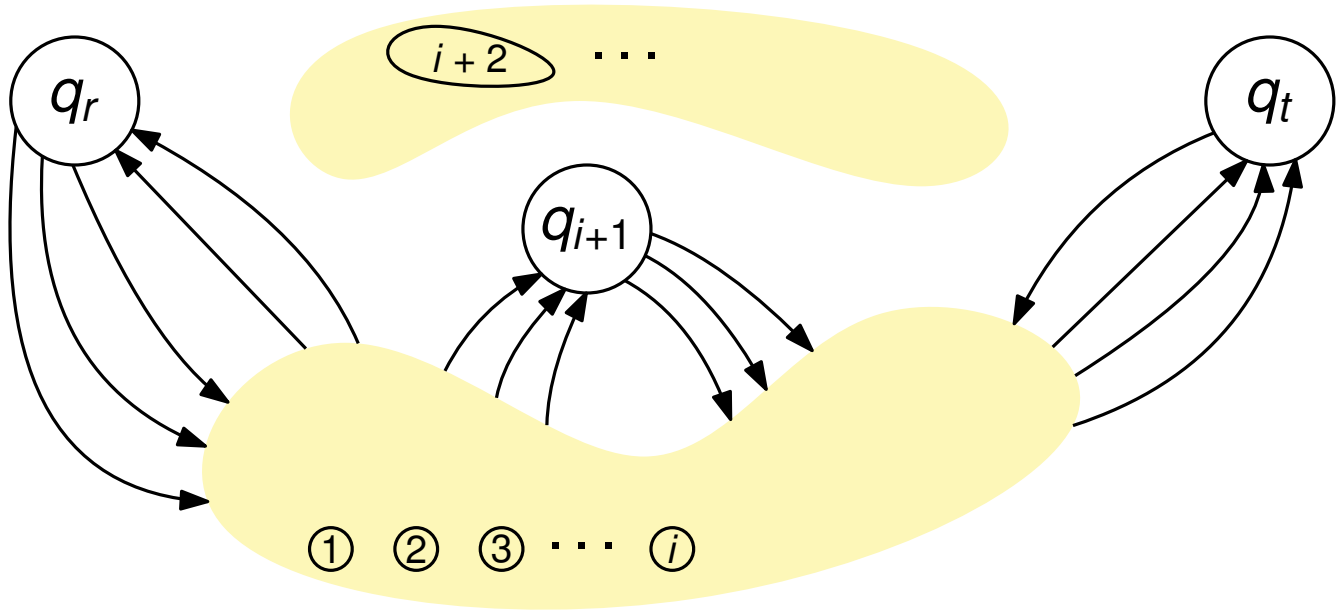
Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t}\right)$$

Erklärung:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

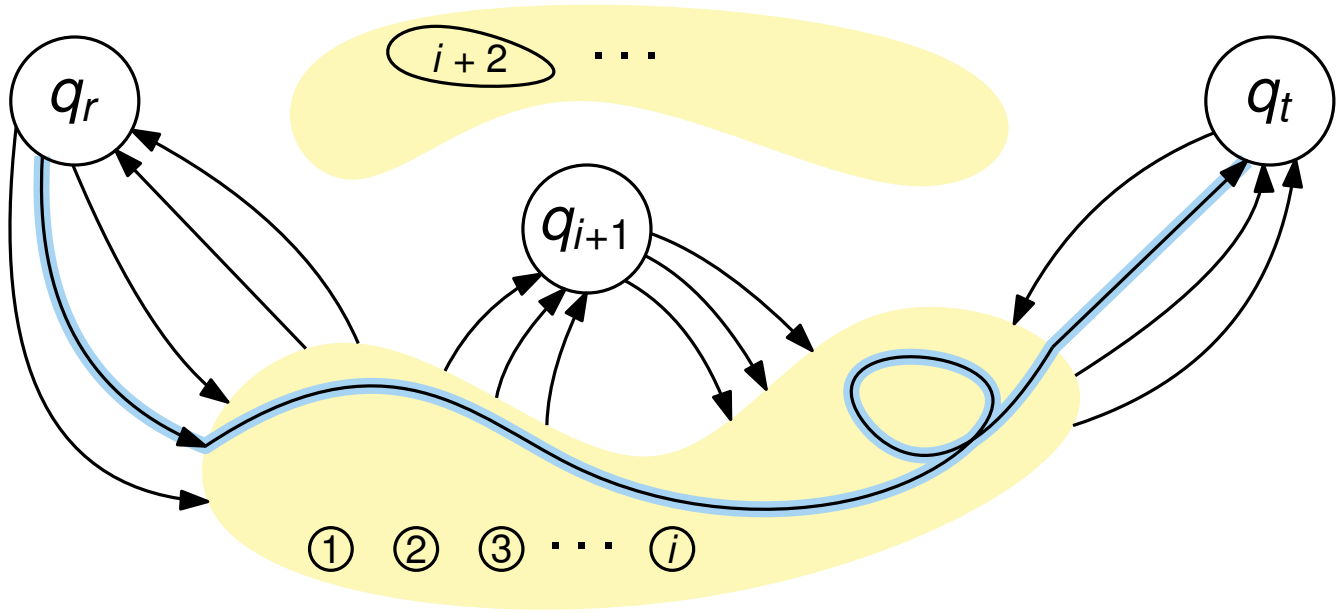
Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t}\right)$$

Erklärung:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

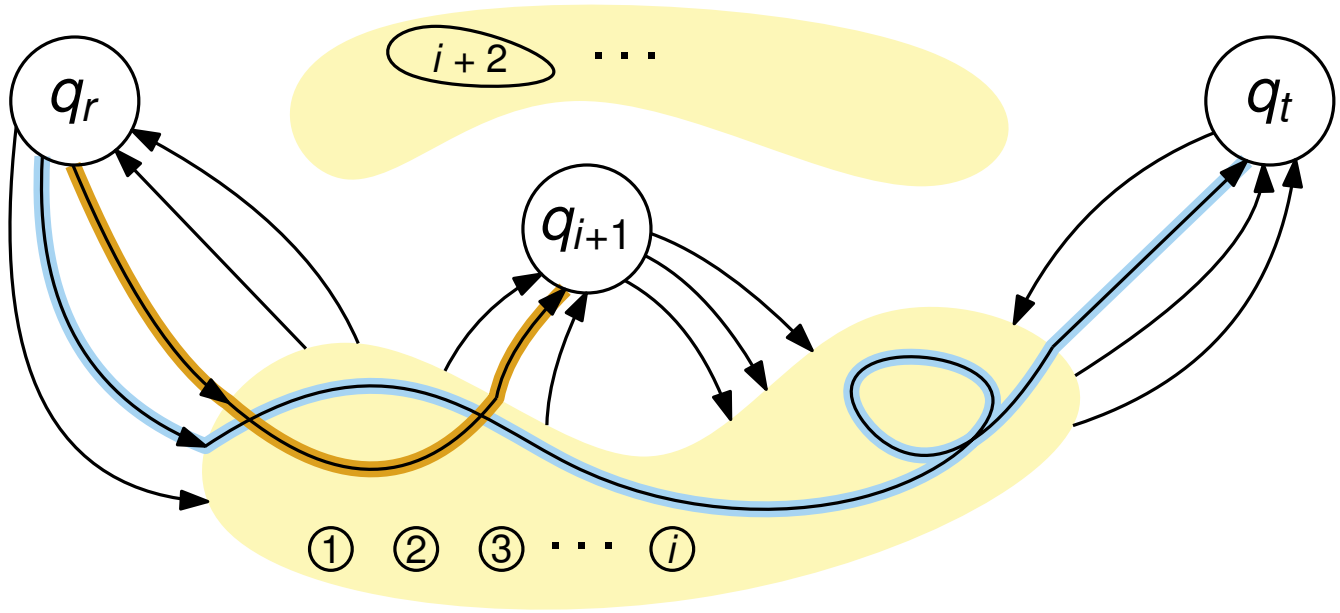
Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Erklärung:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

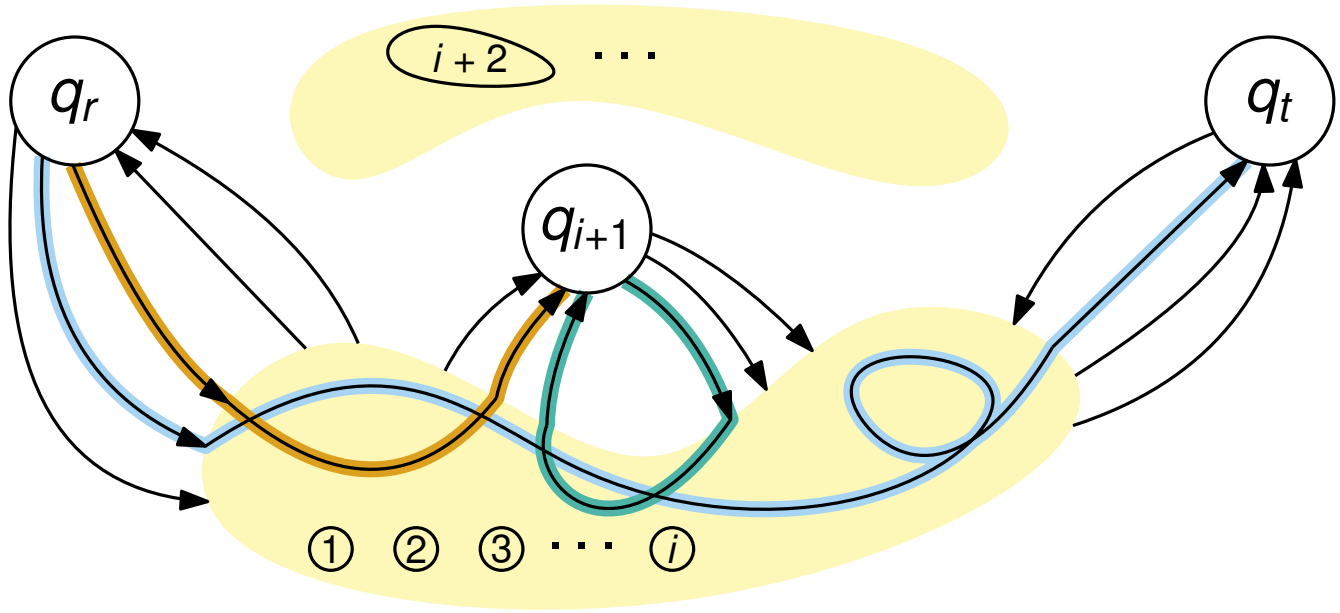
Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Erklärung:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

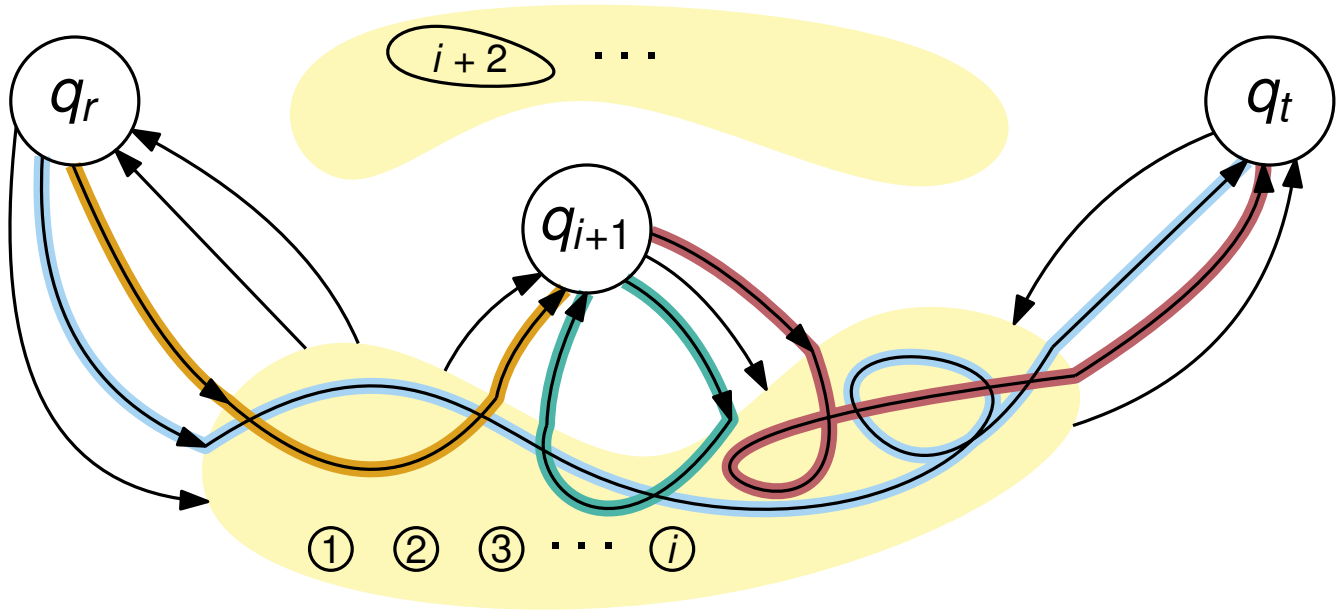
Bestimmung eines regulären Ausdrucks

Gegeben: deterministischer endlicher Automat $\mathcal{A}$

Gesucht: regulärer Ausdruck, der $L(\mathcal{A})$ erzeugt

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} \left(L_{q_{i+1},i,q_{i+1}} \right)^* L_{q_{i+1},i,q_t} \right)$$

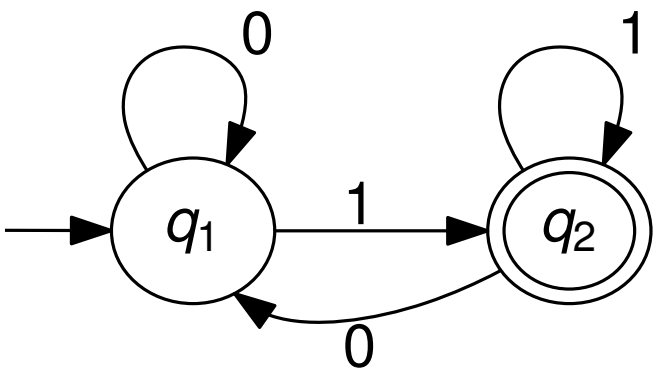
Erklärung:



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.

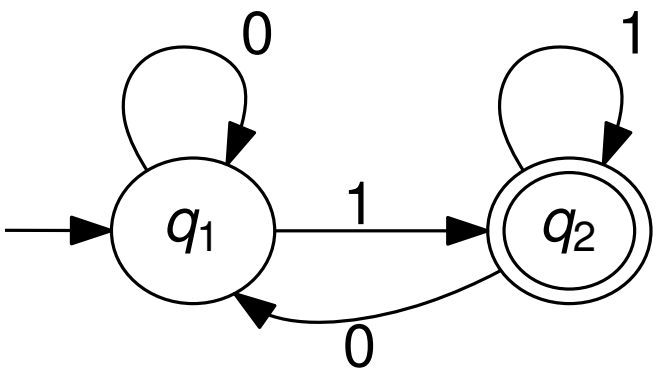


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.

$$L_{1,2,2}$$



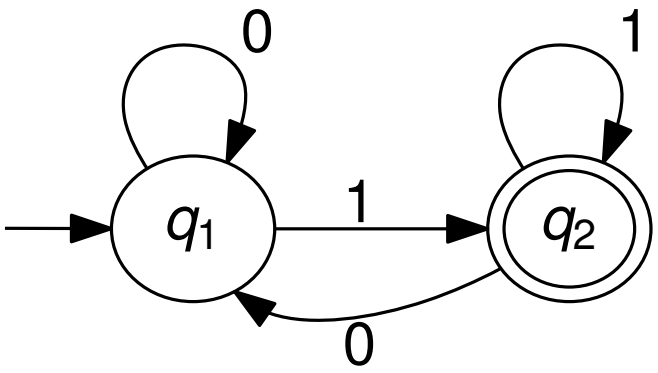
$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.

$$L_{1,2,2} = L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2})$$



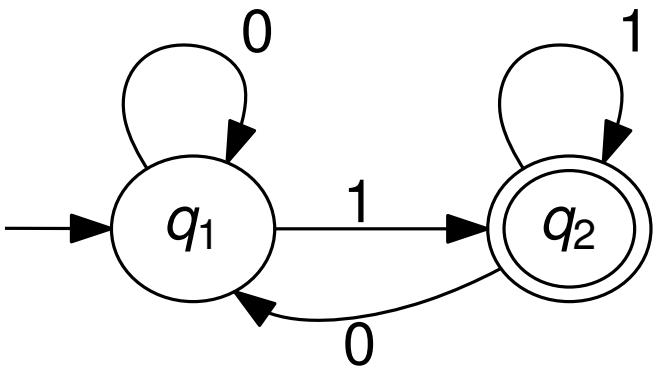
$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.

$$L_{1,2,2} = L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2})$$
$$L_{1,1,2} = L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2})$$

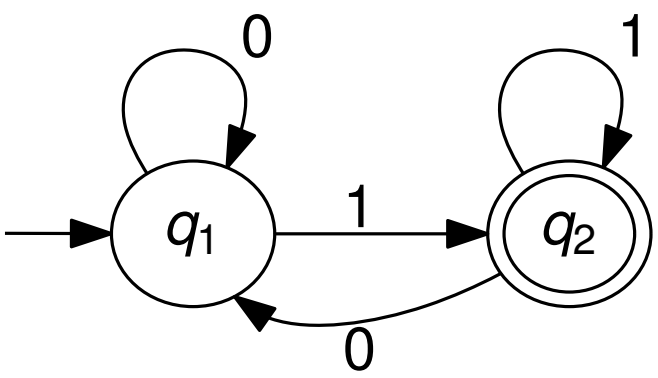


$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



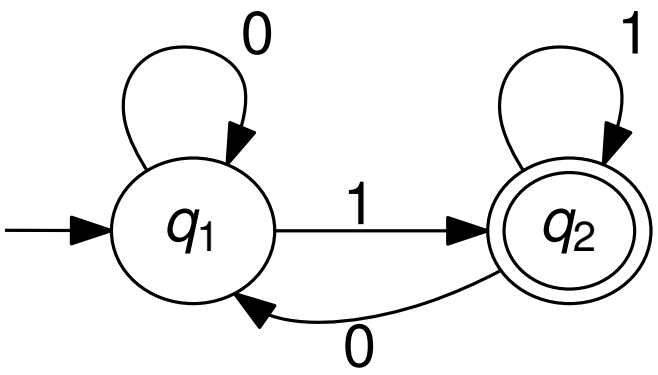
$$L_{1,2,2} = L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2})$$
$$L_{1,1,2} = L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2})$$
$$L_{2,1,2} = L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2})$$

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



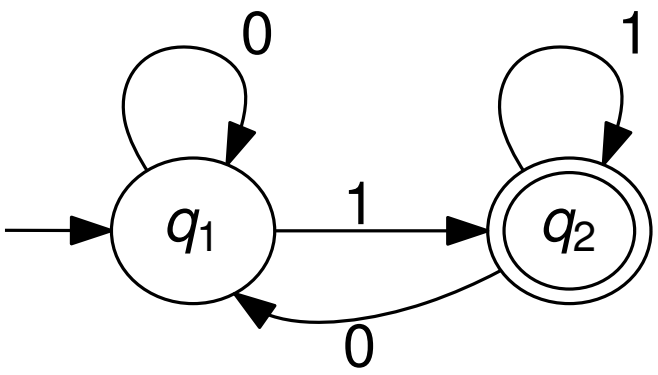
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



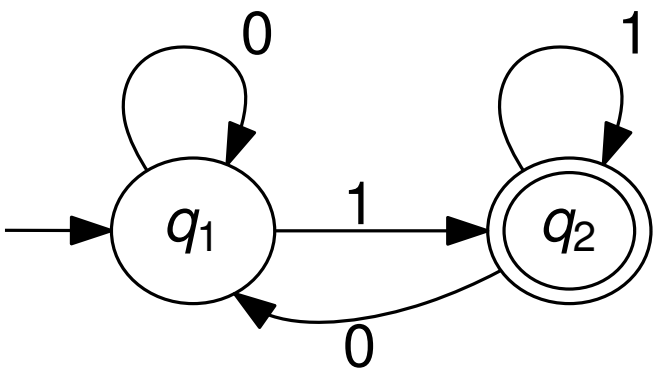
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



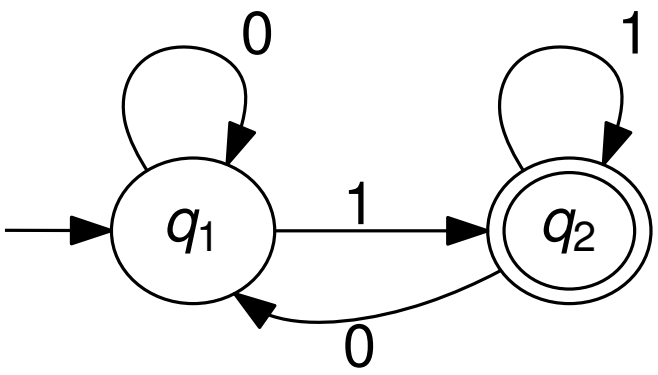
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



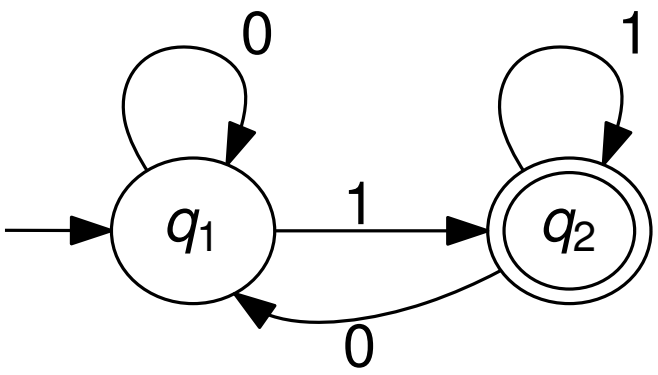
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0
\end{aligned}
$$

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



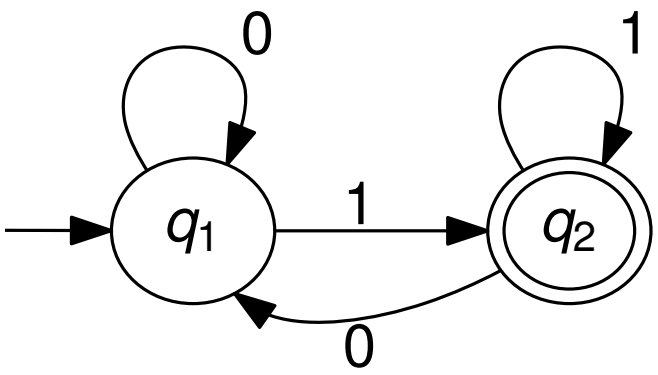
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &
\end{aligned}
$$

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



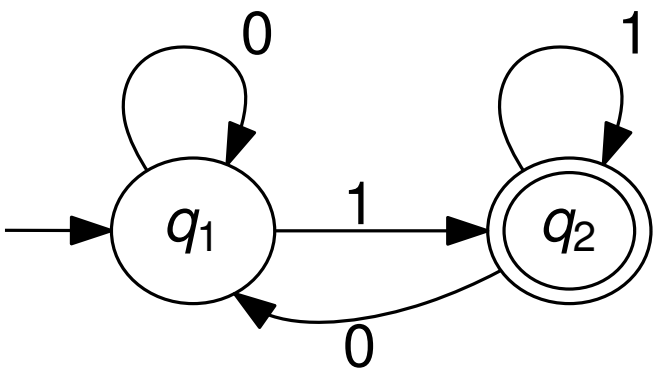
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



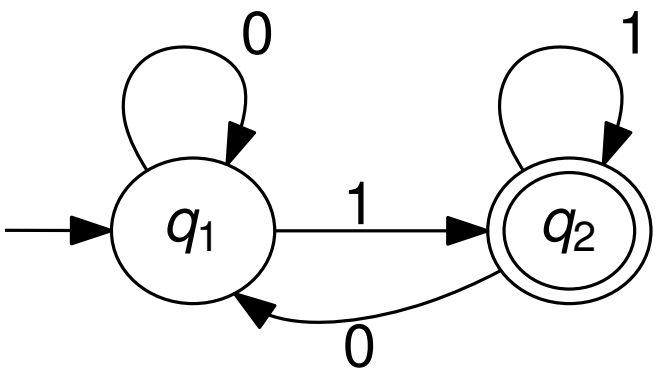
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1)
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



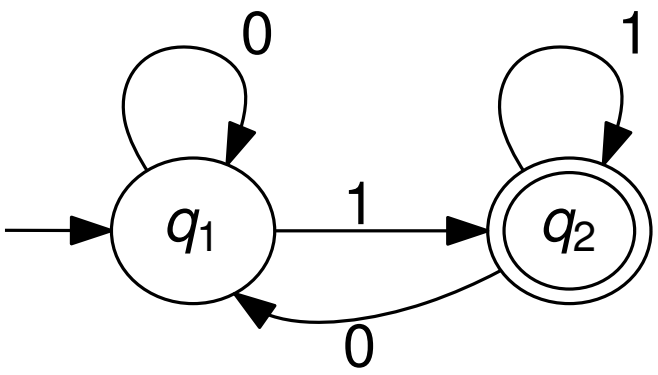
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) \\
&= 1 \cup (\quad 0^* \quad 1)
\end{aligned}
$$

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



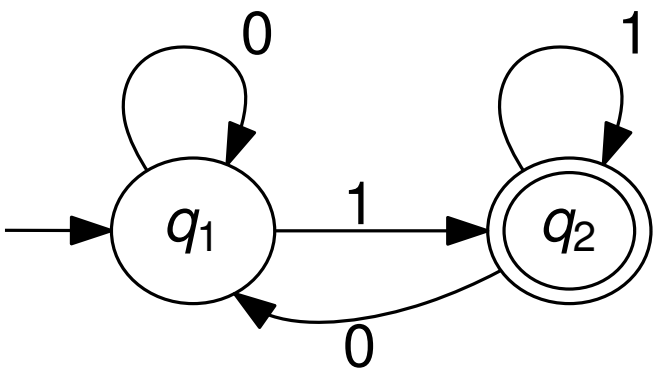
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) \\
&= 1 \cup (\quad 0^* \quad 1) = 0^* 1
\end{aligned}
$$

$$L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



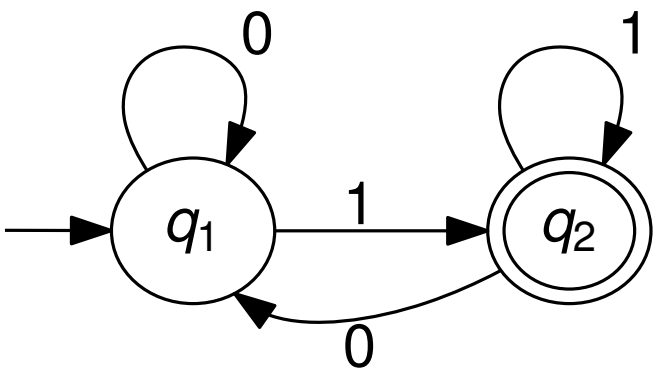
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= (\varepsilon \cup 1) \cup (0(\varepsilon \cup 0)^* 1)
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



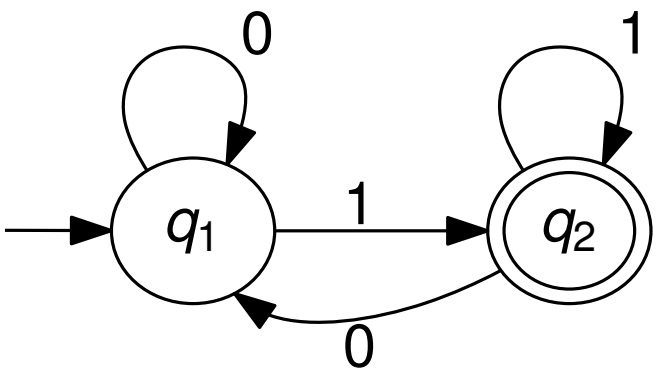
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L^*_{2,1,2} L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L^*_{1,0,1} L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L^*_{1,0,1} L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= (\varepsilon \cup 1) \cup (0(\varepsilon \cup 0)^* 1)
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.

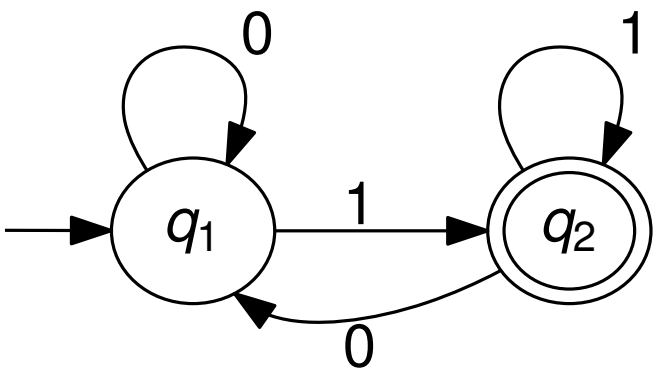


$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= (\varepsilon \cup 1) \cup (0(\varepsilon \cup 0)^* 1) \\
&= (\varepsilon \cup 1) \cup (\quad 0^+ \quad 1) \\
&= \quad \varepsilon \quad \cup (\quad 0^* \quad 1)
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



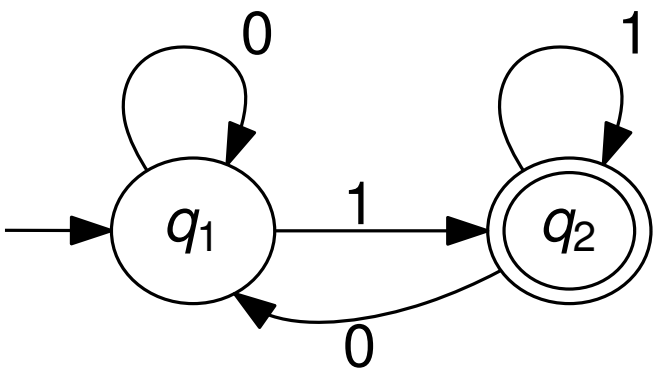
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L^*_{2,1,2} L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L^*_{1,0,1} L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L^*_{1,0,1} L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= \varepsilon \cup (0^* 1)
\end{aligned}
$$

$$
L_{1,2,2} = (0^*1) \cup ((0^*1)(\varepsilon \cup (0^*1))^* (\varepsilon \cup (0^*1)))
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



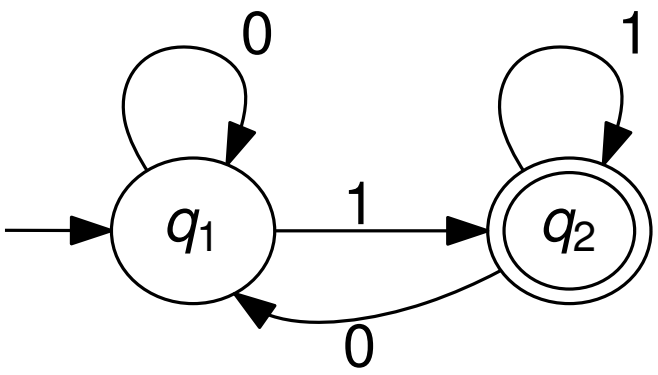
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= \varepsilon \cup (0^* 1)
\end{aligned}
$$

$$
\begin{aligned}
L_{1,2,2} &= (0^*1) \cup ((0^*1)(\varepsilon \cup (0^*1))^* (\varepsilon \cup (0^*1))) \\
&= (0^*1) \cup (0^*1)(\varepsilon \cup (0^*1))^+
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



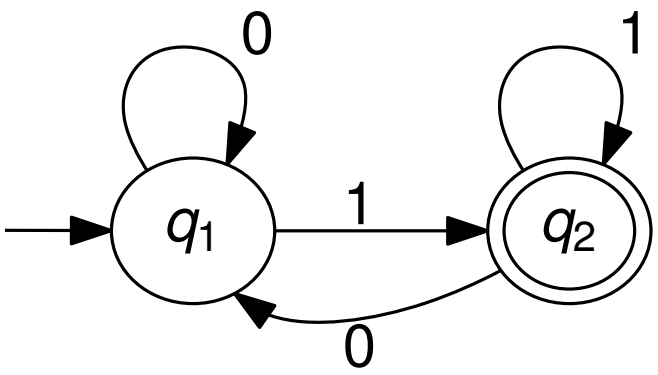
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= \varepsilon \cup (0^* 1)
\end{aligned}
$$

$$
\begin{aligned}
L_{1,2,2} &= (0^*1) \cup (0^*1)(\varepsilon \cup (0^*1))^+ \\
&= \quad (0^*1) \quad\quad (\varepsilon \cup (0^*1))^*
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



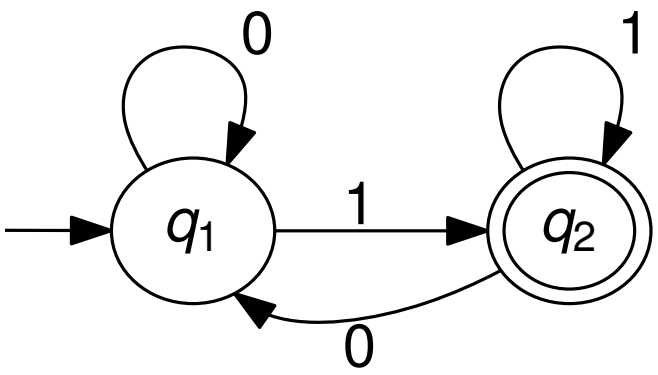
$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= \varepsilon \cup (0^* 1)
\end{aligned}
$$

$$
\begin{aligned}
L_{1,2,2} &= (0^*1)(\varepsilon \cup (0^*1))^* \\
&= (0^*1)(0^*1)^* = (0^*1)^+
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Bestimmung eines regulären Ausdrucks

Bestimmen Sie nach der Methode aus der Vorlesung (Satz von Kleene) einen regulären Ausdruck für die von folgendem Automaten $\mathcal{A}$ erkannte Sprache.



$$
\begin{aligned}
L_{1,2,2} &= L_{1,1,2} \cup (L_{1,1,2} L_{2,1,2}^* L_{2,1,2}) \\
L_{1,1,2} &= L_{1,0,2} \cup (L_{1,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{2,1,2} &= L_{2,0,2} \cup (L_{2,0,1} L_{1,0,1}^* L_{1,0,2}) \\
L_{1,0,1} &= \varepsilon \cup 0 \\
L_{1,0,2} &= 1 \\
L_{2,0,2} &= \varepsilon \cup 1 \\
L_{2,0,1} &= 0 \\
L_{1,1,2} &= 1 \cup ((\varepsilon \cup 0)(\varepsilon \cup 0)^* 1) = 0^* 1 \\
L_{2,1,2} &= \varepsilon \cup (0^* 1) \\
L_{1,2,2} &= (0^* 1)^+
\end{aligned}
$$

$$
L_{q_r,i+1,q_t} = L_{q_r,i,q_t} \cup \left(L_{q_r,i,q_{i+1}} (L_{q_{i+1},i,q_{i+1}})^* L_{q_{i+1},i,q_t} \right)
$$

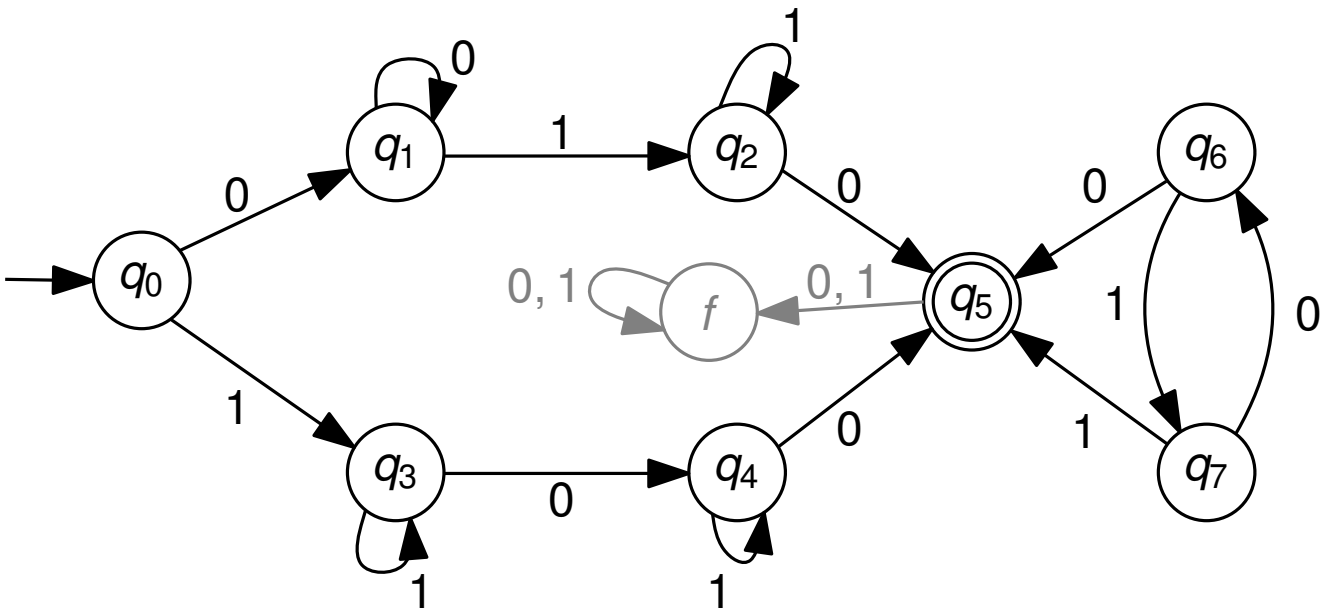
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entfernen überflüssiger Zuständе

Zustände eines endlichen Automaten, die vom Startzustand aus nicht erreichbar sind, sind **überflüssig**.

Verfahren, um nicht erreichbare Zustände zu bestimmen?

Betrachte Zustandsgraph $G = (V, E)$. Wende Tiefen- oder Breitensuche von q_0 aus an. Alle Knoten, die die Suche nicht findet, sind unerreichbar.



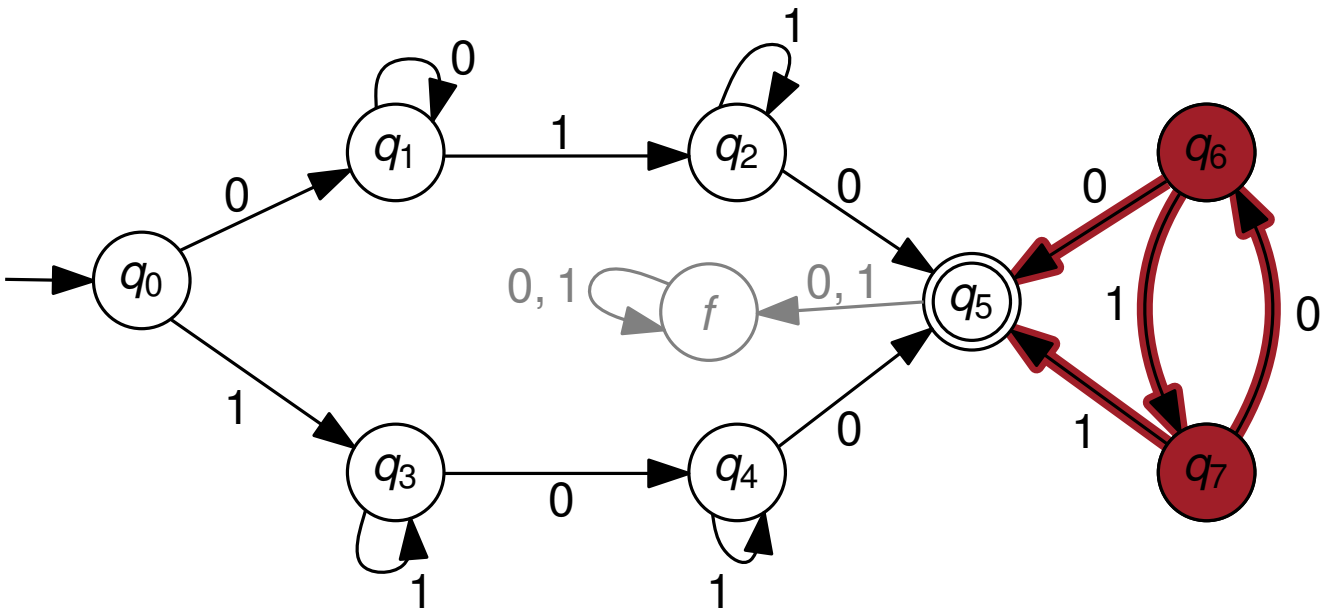
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entfernen überflüssiger Zustände

Zustände eines endlichen Automaten, die vom Startzustand aus nicht erreichbar sind, sind **überflüssig**.

Verfahren, um nicht erreichbare Zustände zu bestimmen?

Betrachte Zustandsgraph $G = (V, E)$. Wende Tiefen- oder Breitensuche von q_0 aus an. Alle Knoten, die die Suche nicht findet, sind unerreichbar.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

- Zwei Zustände haben dasselbe Akzeptanzverhalten, wenn es für das Erreichen eines Endzustandes durch Abarbeiten eines Wortes w unerheblich ist, aus welchem der beiden Zustände wir starten.
- Reduktion der Anzahl der Zustände durch Zusammenlegen der Zustände mit gleichem Akzeptanzverhalten.

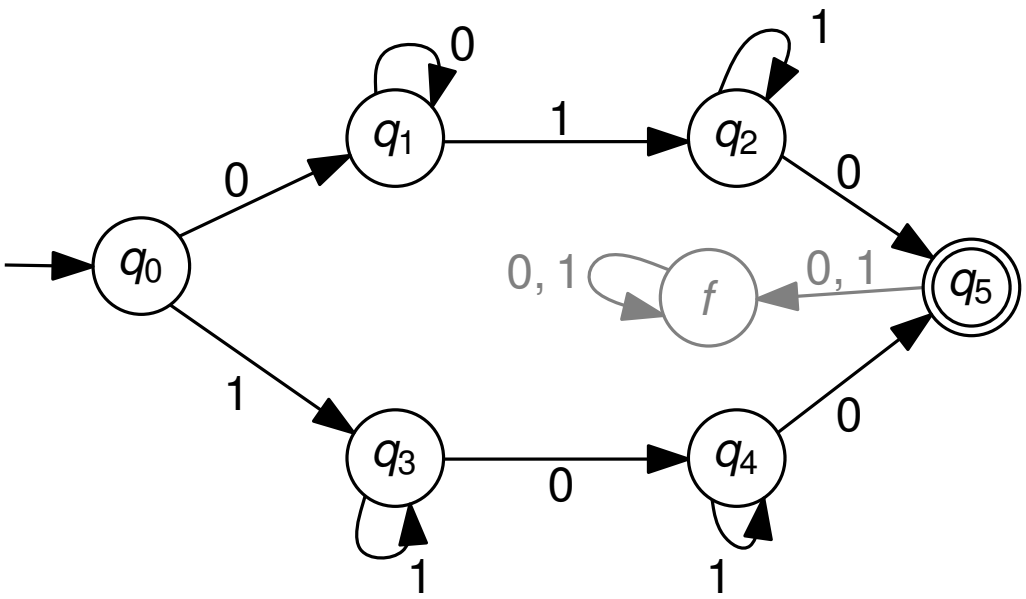
Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Welche Zustände sind äquivalent?



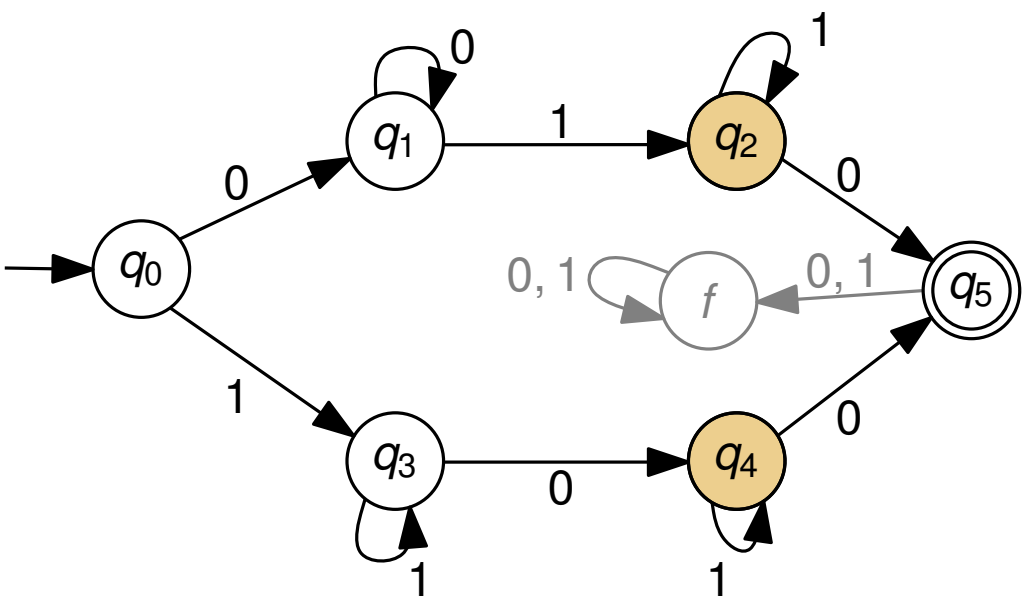
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.



Welche Zustände sind äquivalent?

q_2 und q_4?

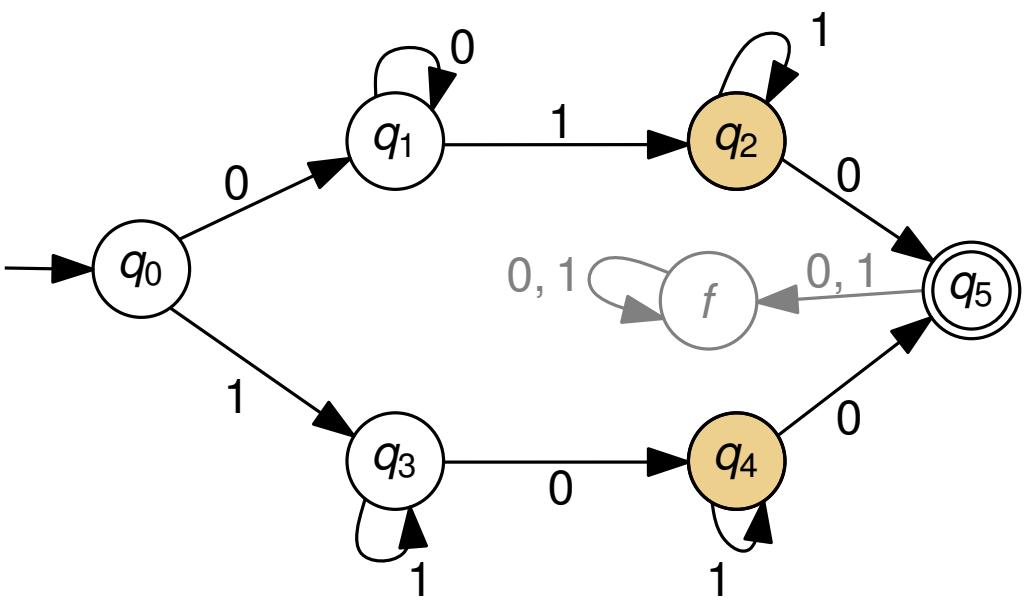
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.



Welche Zustände sind äquivalent?

q_2 und q_4?

Welche Wörter führen zu q_5?

q_2: 1^*0 q_4: 1^*0

$\Rightarrow$ Ja

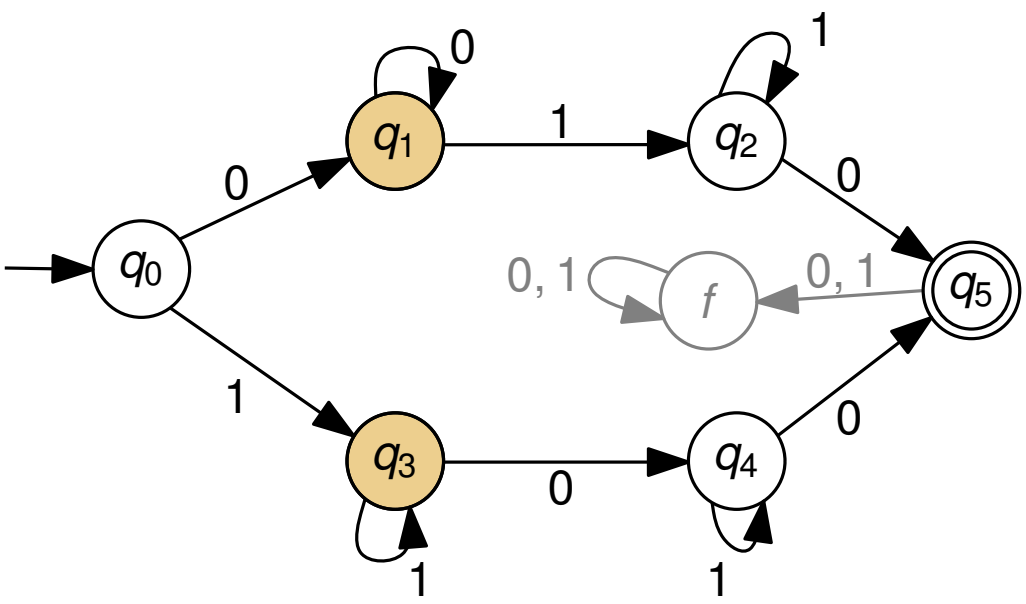
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.



Welche Zustände sind äquivalent?

q_1 und q_3?

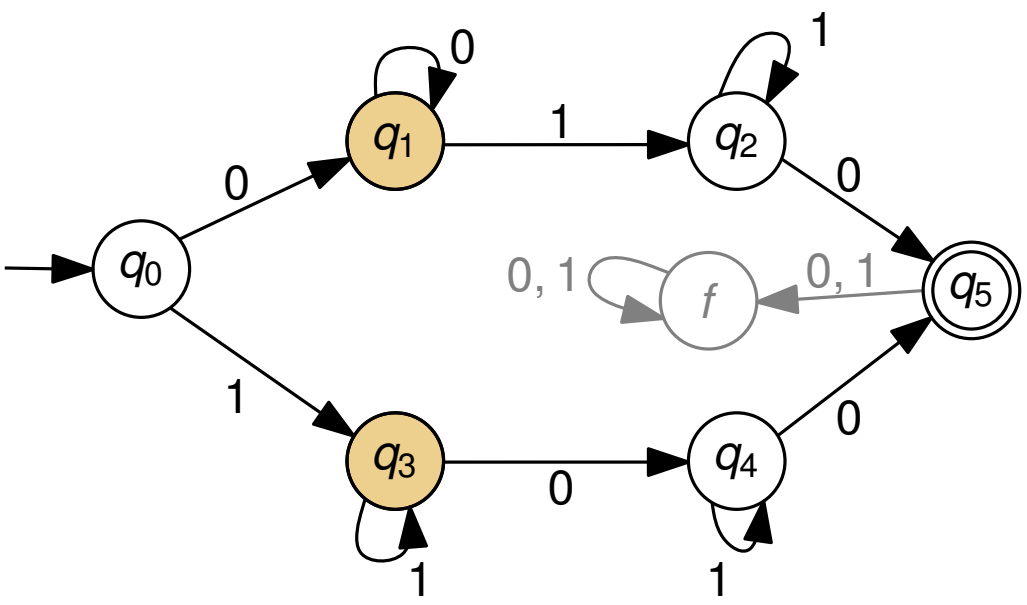
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \iff \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.



Welche Zustände sind äquivalent?

q_1 und q_3?

$\delta(q_1, 10) = q_5 \in F$

$\delta(q_3, 10) = q_4 \notin F$

$\Rightarrow$ Nein

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

- $Q^{\equiv} := \{[q]\} \quad q \in Q$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f]\} \quad f \in F$

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

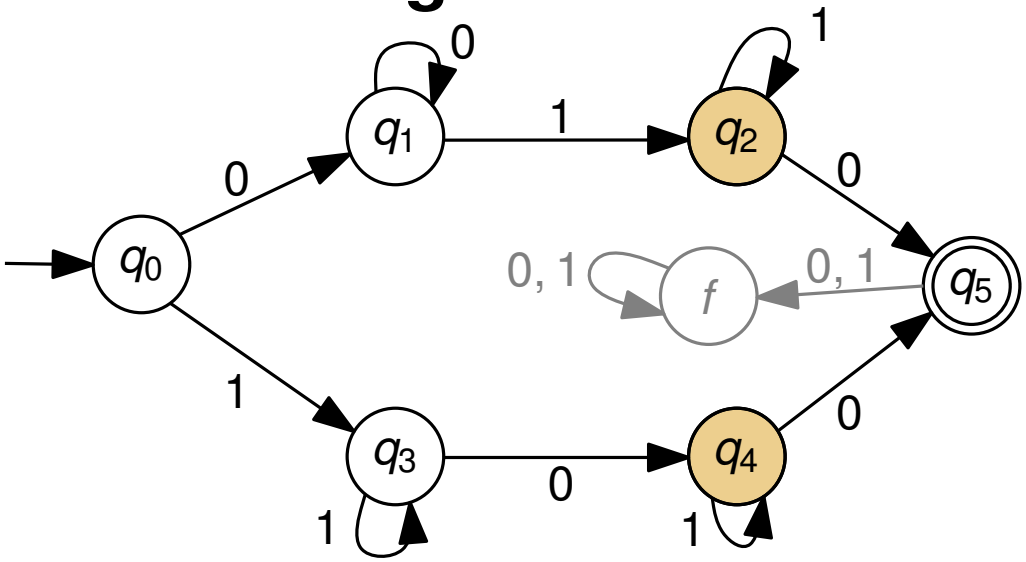
- $Q^{\equiv} := \{[q]\} \quad q \in Q$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f]\} \quad f \in F$

In der Vorlesung:

- $\mathcal{A}^{\equiv}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.
- $\mathcal{A}^{\equiv}$ ist minimal, falls $\mathcal{A}$ keine überflüssigen Zustände hat.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten



Wie sieht der Äquivalenzklassenautomat aus?

Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

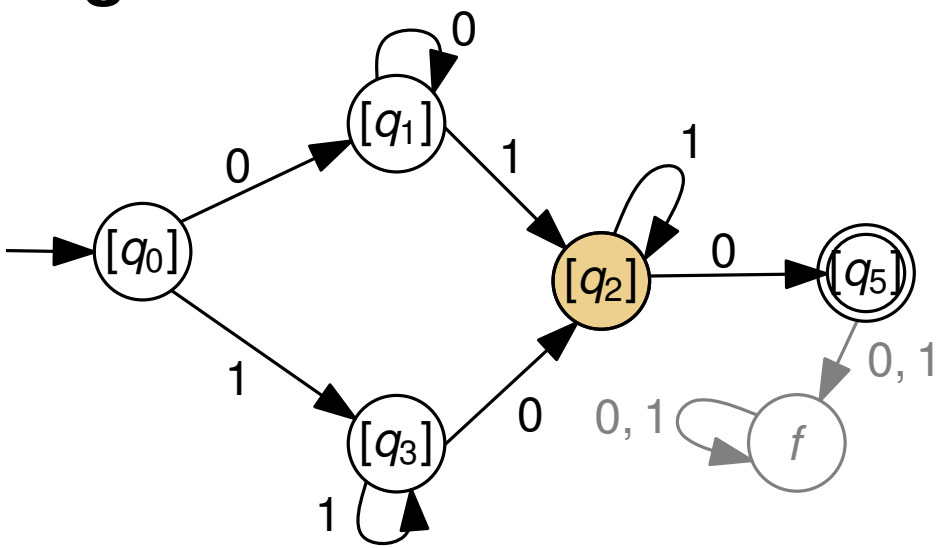
- $Q^{\equiv} := \{[q]\} \quad q \in Q$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f]\} \quad f \in F$

In der Vorlesung:

- $\mathcal{A}^{\equiv}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.
- $\mathcal{A}^{\equiv}$ ist minimal, falls $\mathcal{A}$ keine überflüssigen Zustände hat.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten



Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

- $Q^{\equiv} := \{[q]\} \quad q \in Q$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f]\} \quad f \in F$

In der Vorlesung:

- $\mathcal{A}^{\equiv}$ akzeptiert dieselbe Sprache wie $\mathcal{A}$.
- $\mathcal{A}^{\equiv}$ ist minimal, falls $\mathcal{A}$ keine überflüssigen Zustände hat.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

Zwei Zustände p und q eines deterministischen endlichen Automaten heißen *äquivalent* ($p \equiv q$), wenn für alle Wörter $w \in \Sigma^*$ gilt:

$$\delta(p, w) \in F \quad \Longleftrightarrow \quad \delta(q, w) \in F.$$

Mit $[p]$ bezeichnen wir die Äquivalenzklasse der zu p äquivalenten Zustände.

Zu einem DEA $\mathcal{A} = (Q, \Sigma, \delta, s, F)$ definieren wir den Äquivalenzklassenautomaten $\mathcal{A}^{\equiv} = (Q^{\equiv}, \Sigma^{\equiv}, \delta^{\equiv}, s^{\equiv}, F^{\equiv})$ durch:

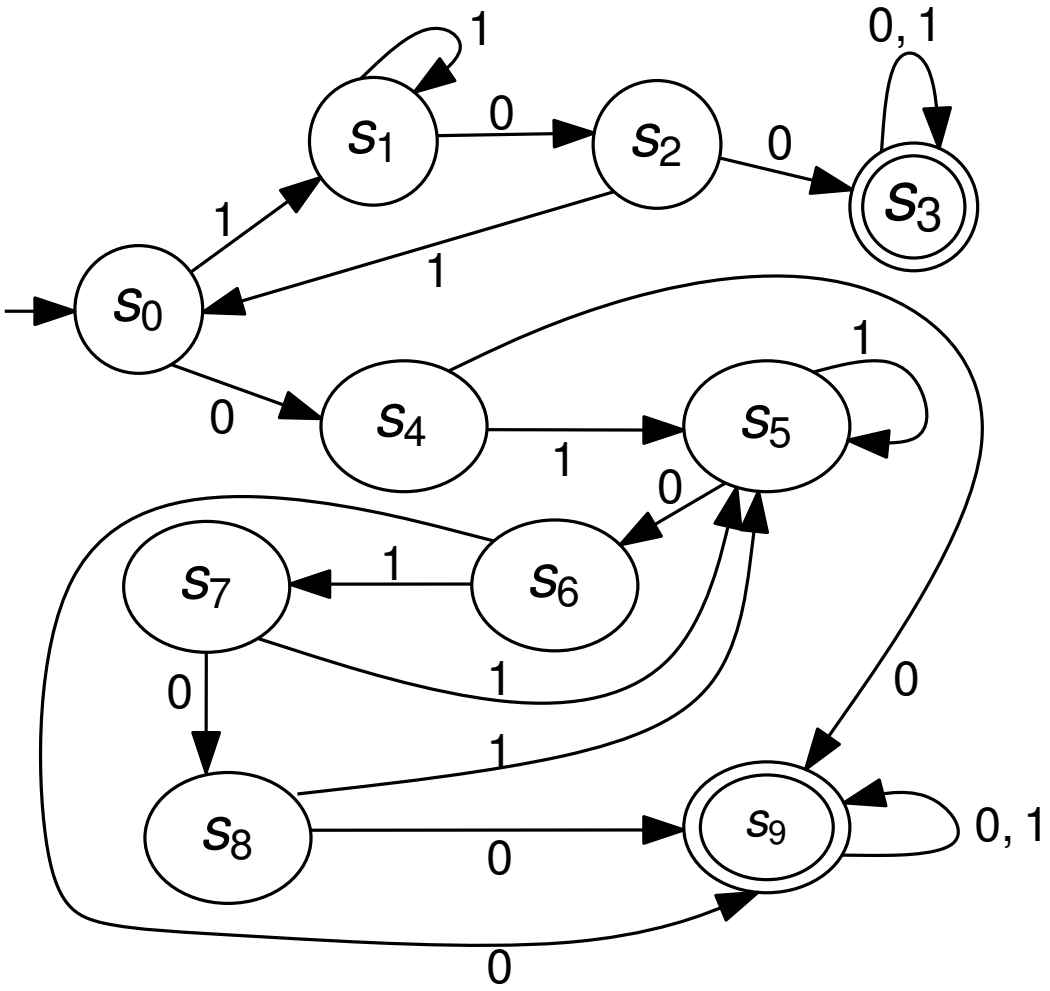
- $Q^{\equiv} := \{[q]\} \quad q \in Q$
- $\Sigma^{\equiv} := \Sigma$
- $\delta^{\equiv}([q], a) := [\delta(q, a)]$
- $s^{\equiv} := [s]$
- $F^{\equiv} := \{[f]\} \quad f \in F$

Verfahren:

Finde systematisch **Zeugen** w für Zustände p und q, die nicht äquivalent sind:

$\delta(p, w) \in F$ aber $\delta(q, w) \notin F$ oder $\delta(p, w) \notin F$ aber $\delta(q, w) \in F$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



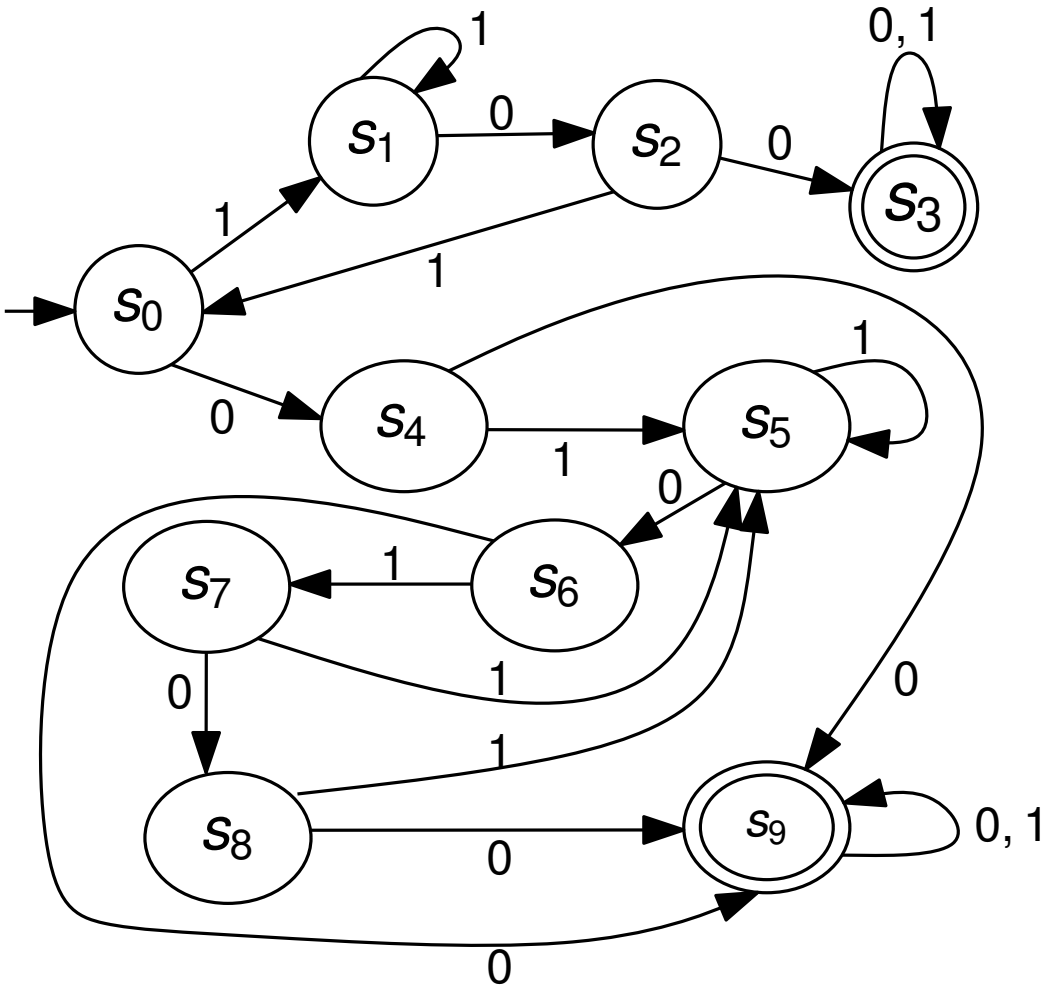
$s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9$

Betrachte Wort w =

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



$s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9$

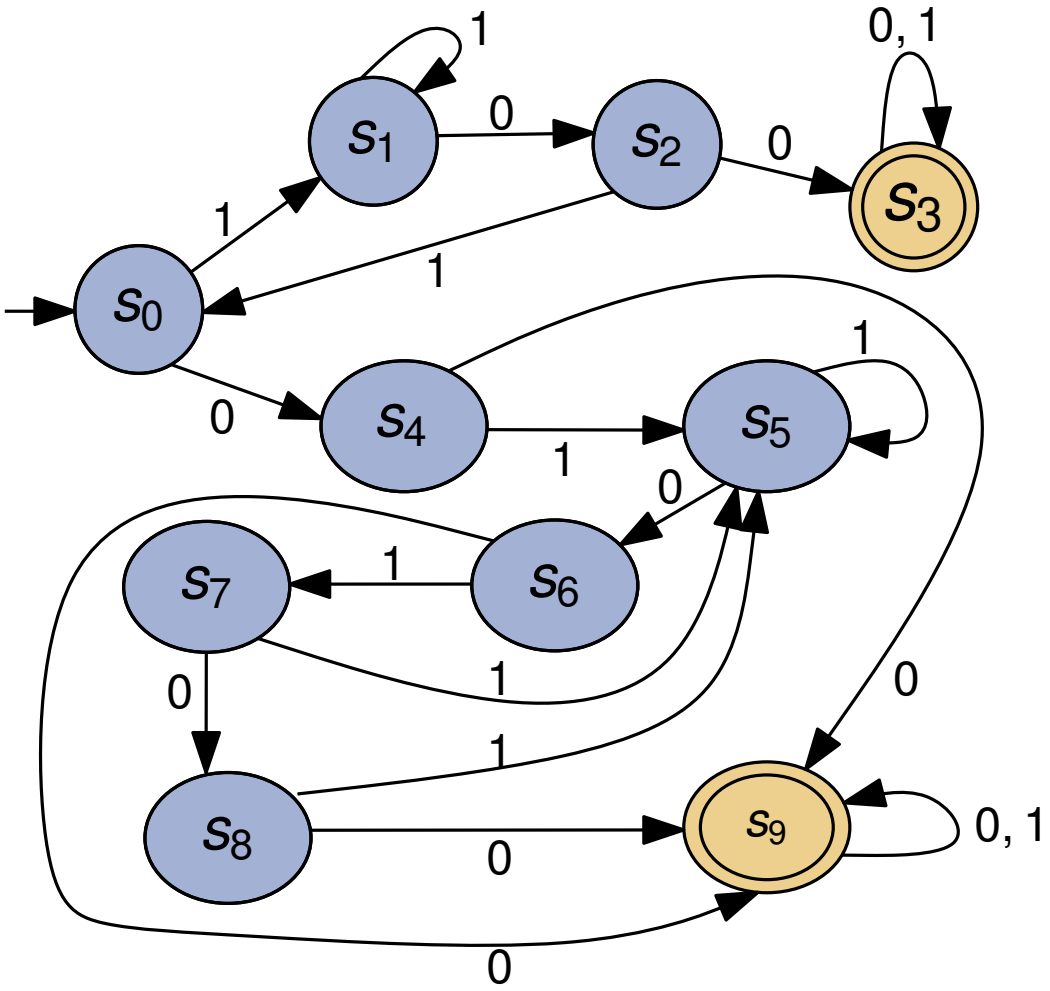
ε trennt

Betrachte Wort $w = \varepsilon$

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



$s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9$

ε trennt

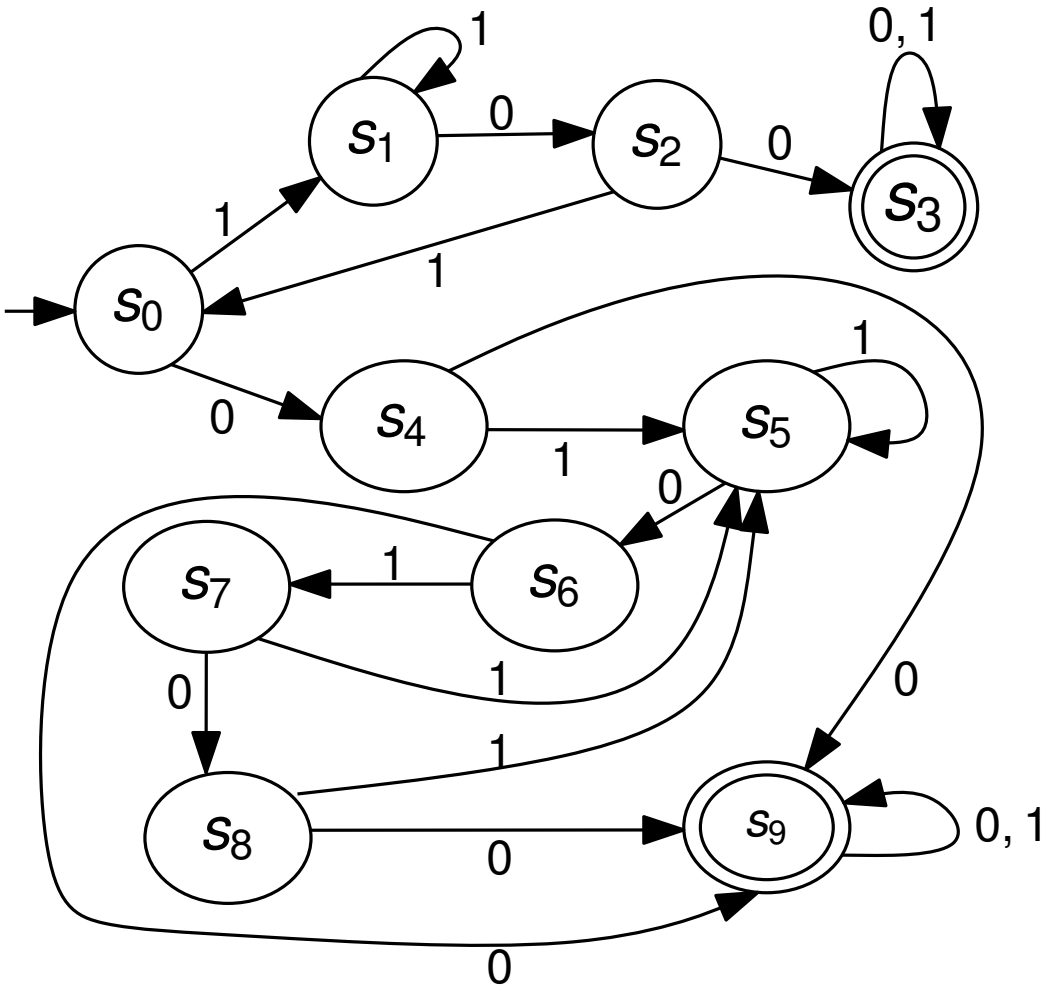
s_3, s_9 | $s_0, s_1, s_2, s_4, s_5, s_6, s_7, s_8$

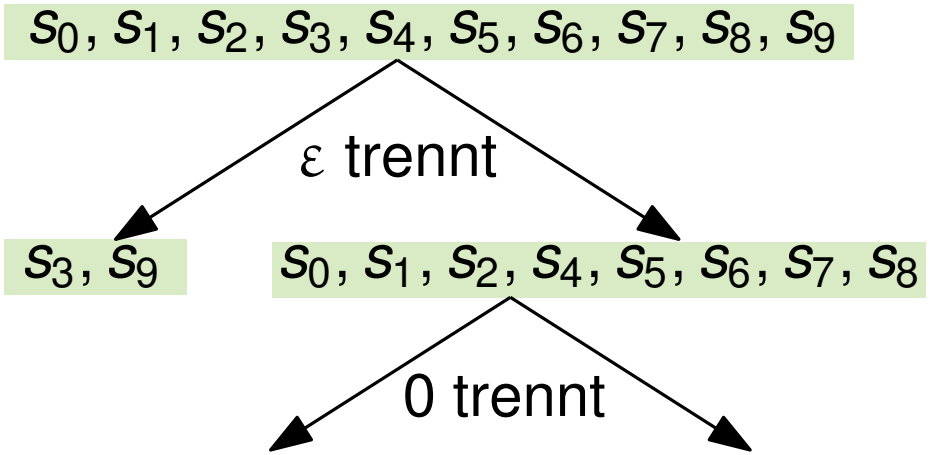
Betrachte Wort $w = \varepsilon$

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



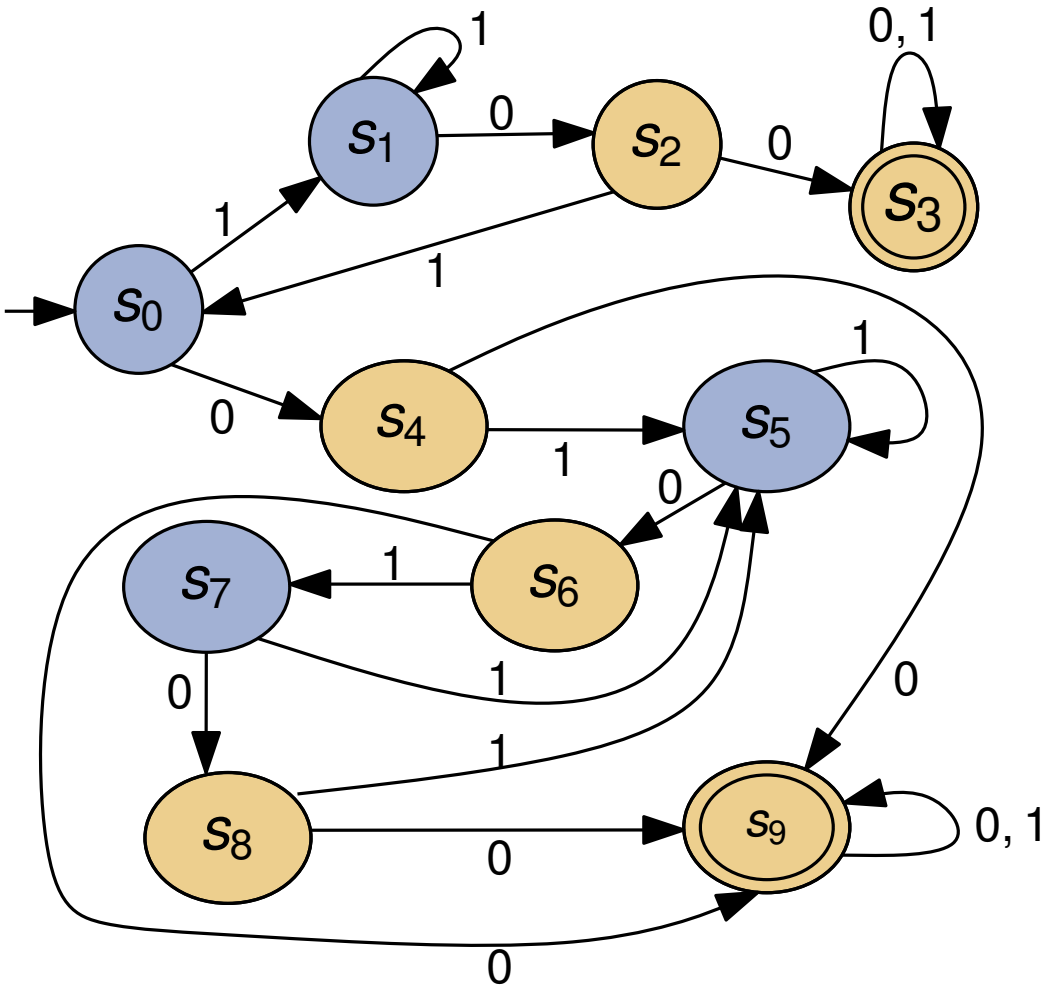


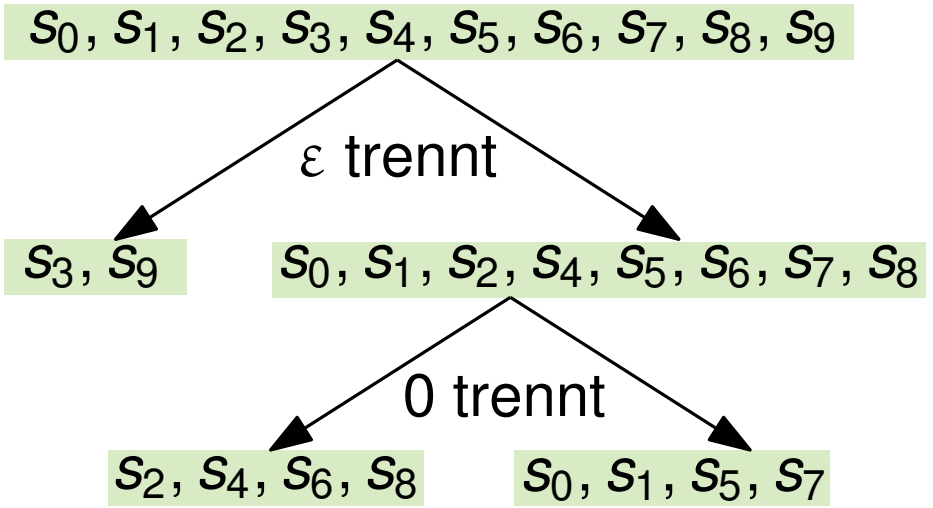
Betrachte Wort $w =$ 0

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



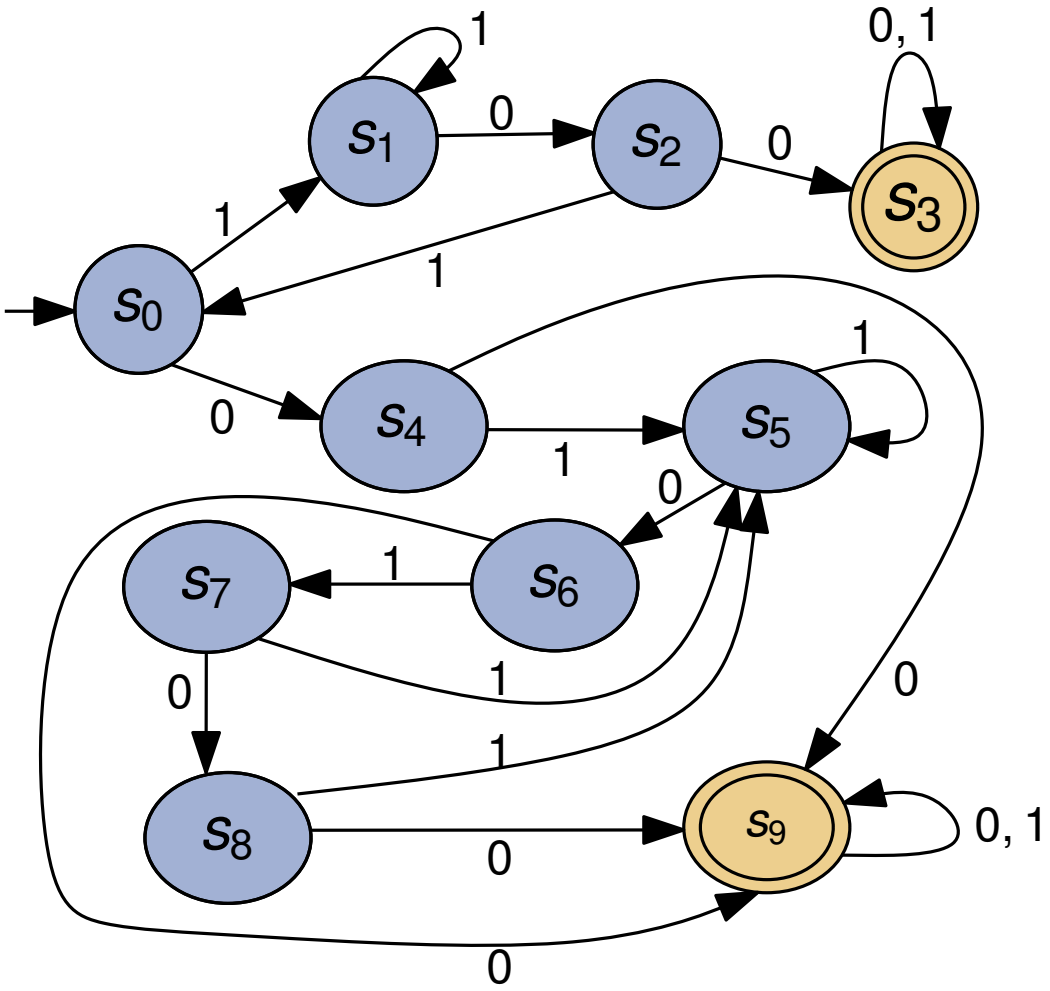


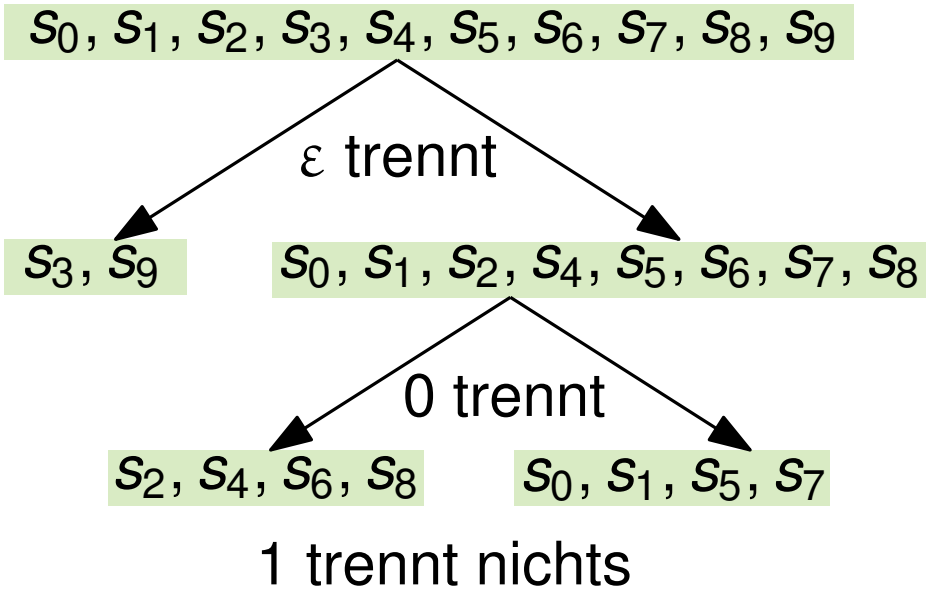
Betrachte Wort $w =$ 0

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



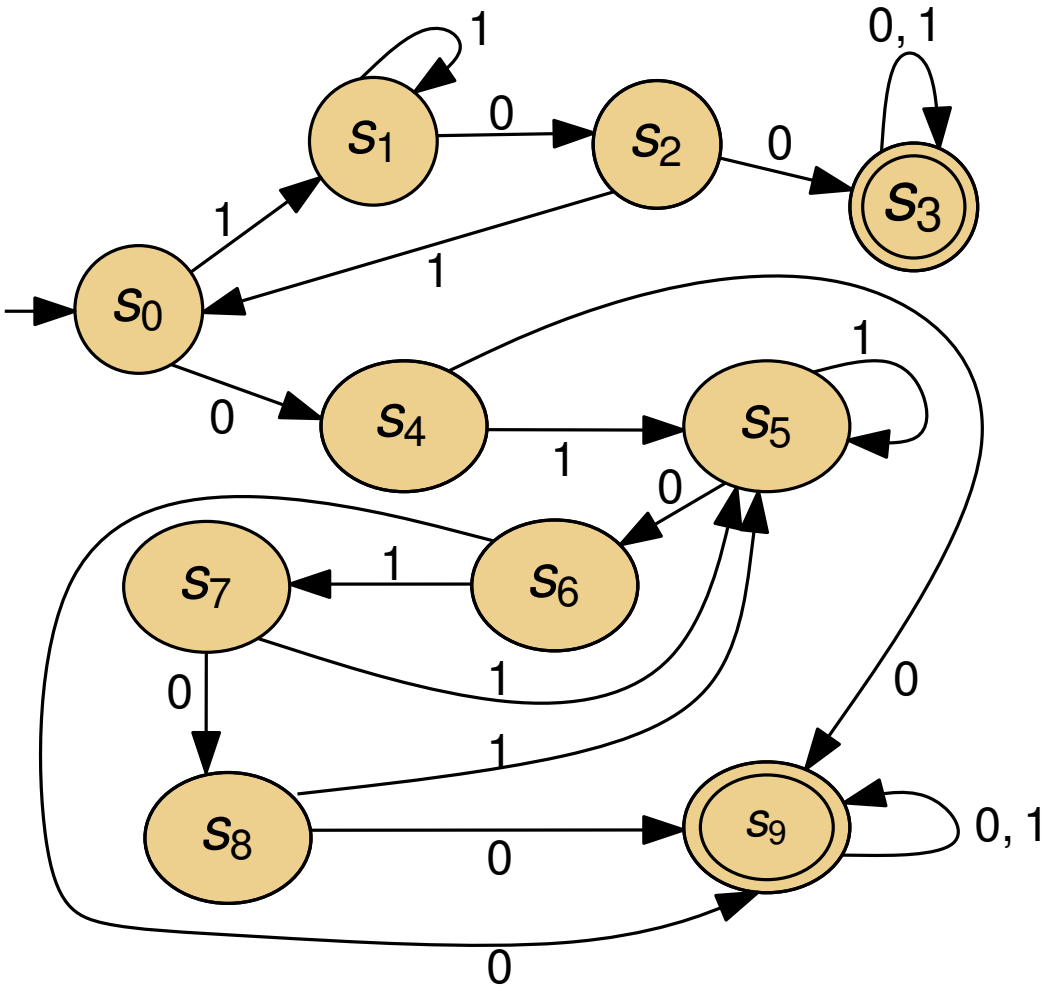


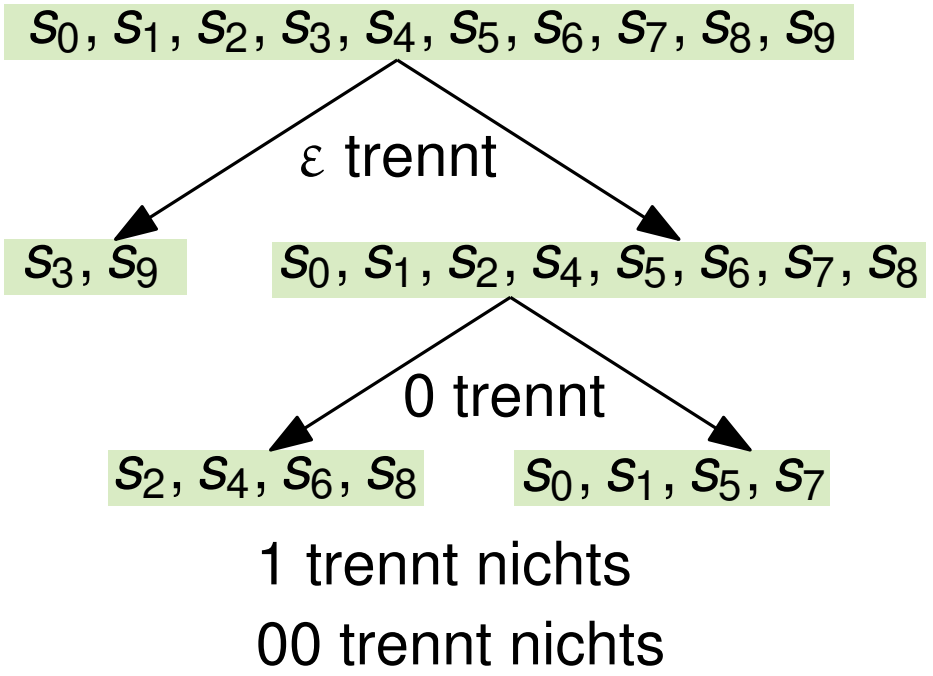
Betrachte Wort $w =$ 1

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik



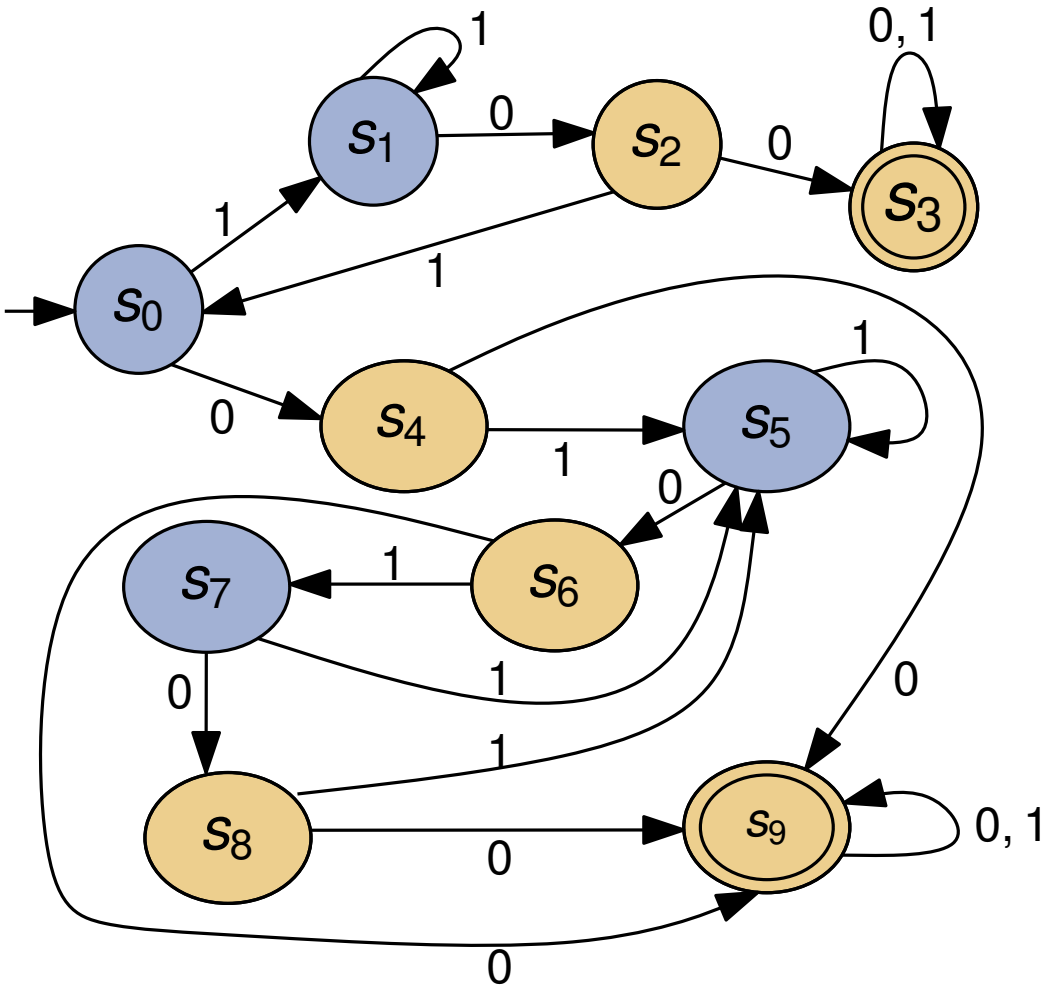


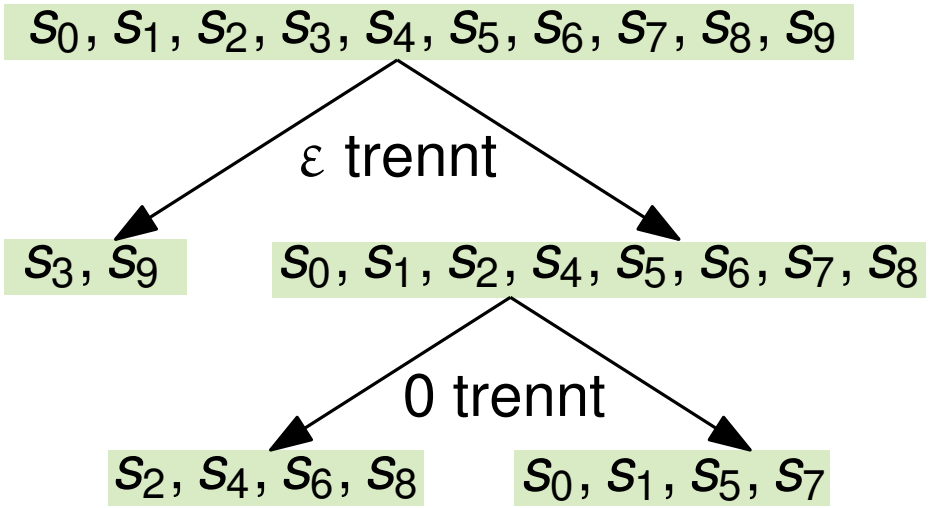
Betrachte Wort $w =$ 00

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik





1 trennt nichts

00 trennt nichts

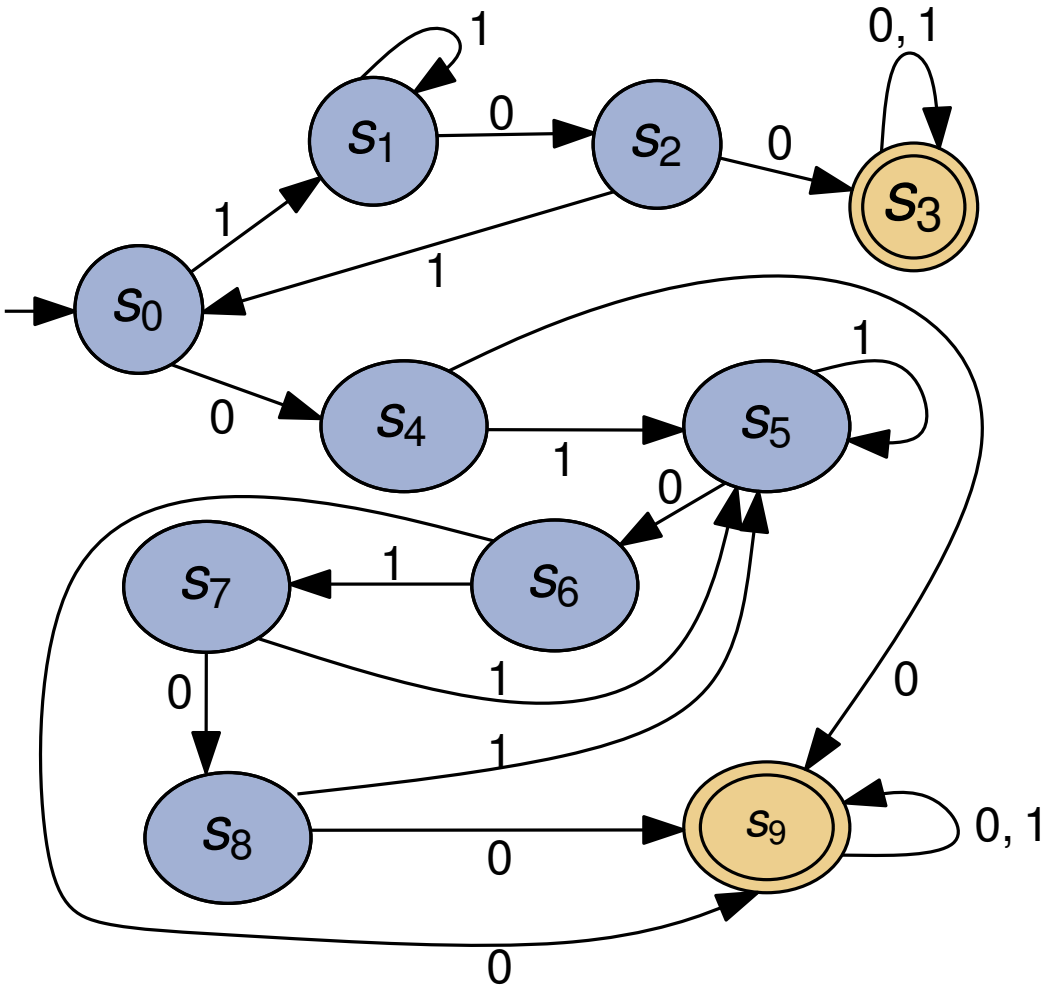
01 trennt nichts

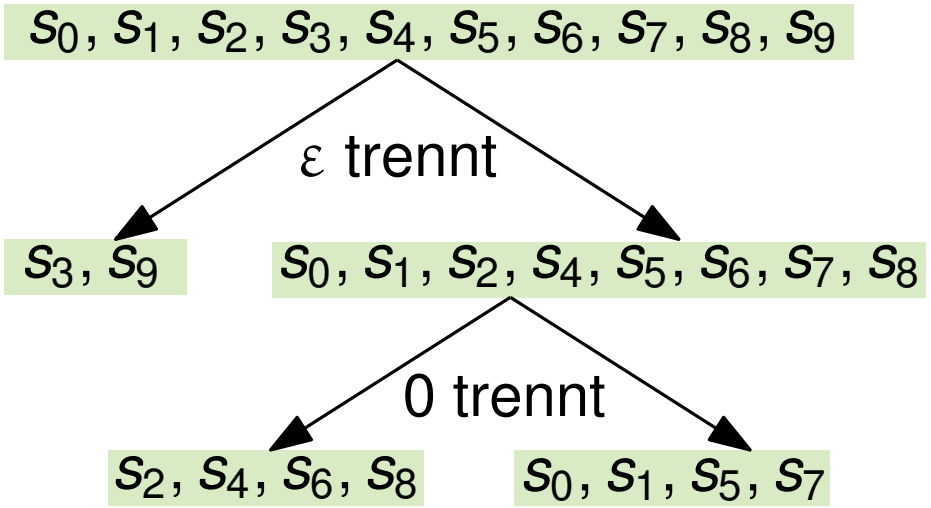
Betrachte Wort $w =$ 01

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik





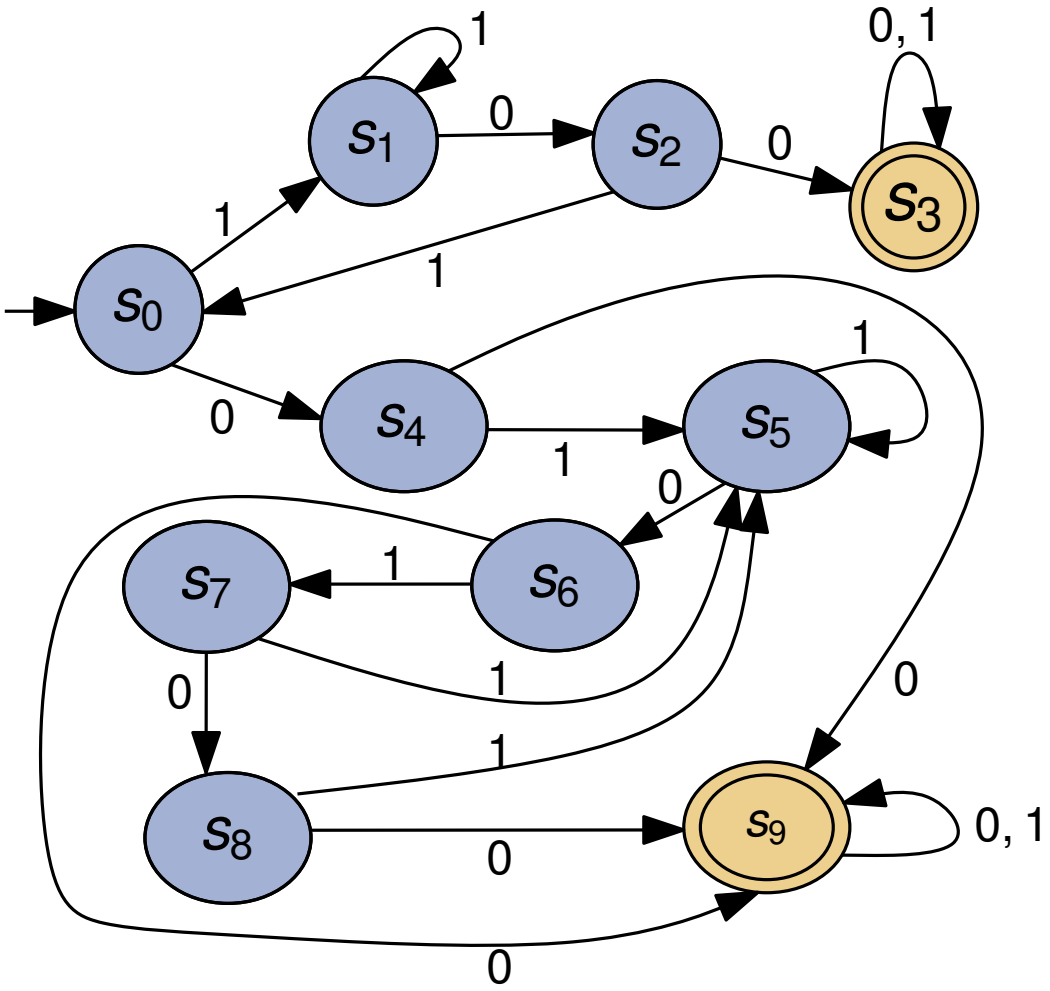
1 trennt nichts
00 trennt nichts
01 trennt nichts
10 trennt nichts

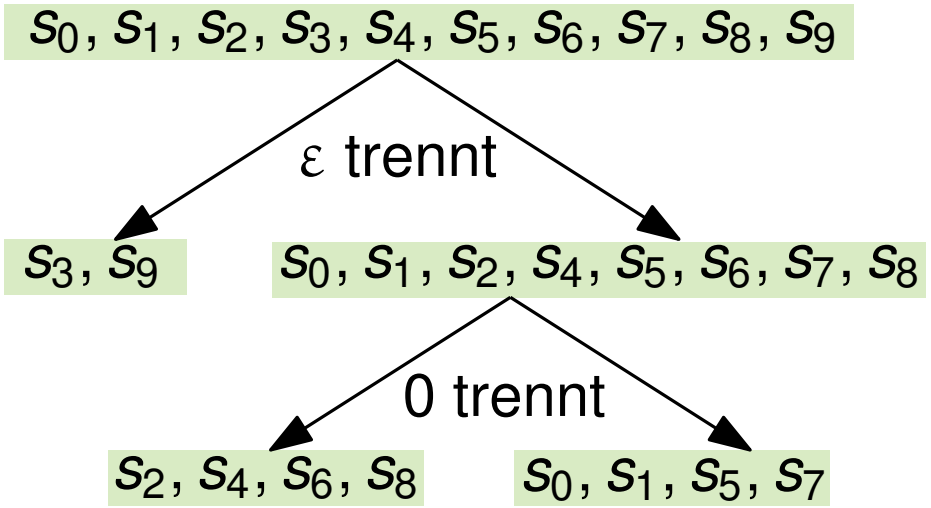
Betrachte Wort $w =$ 10

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik





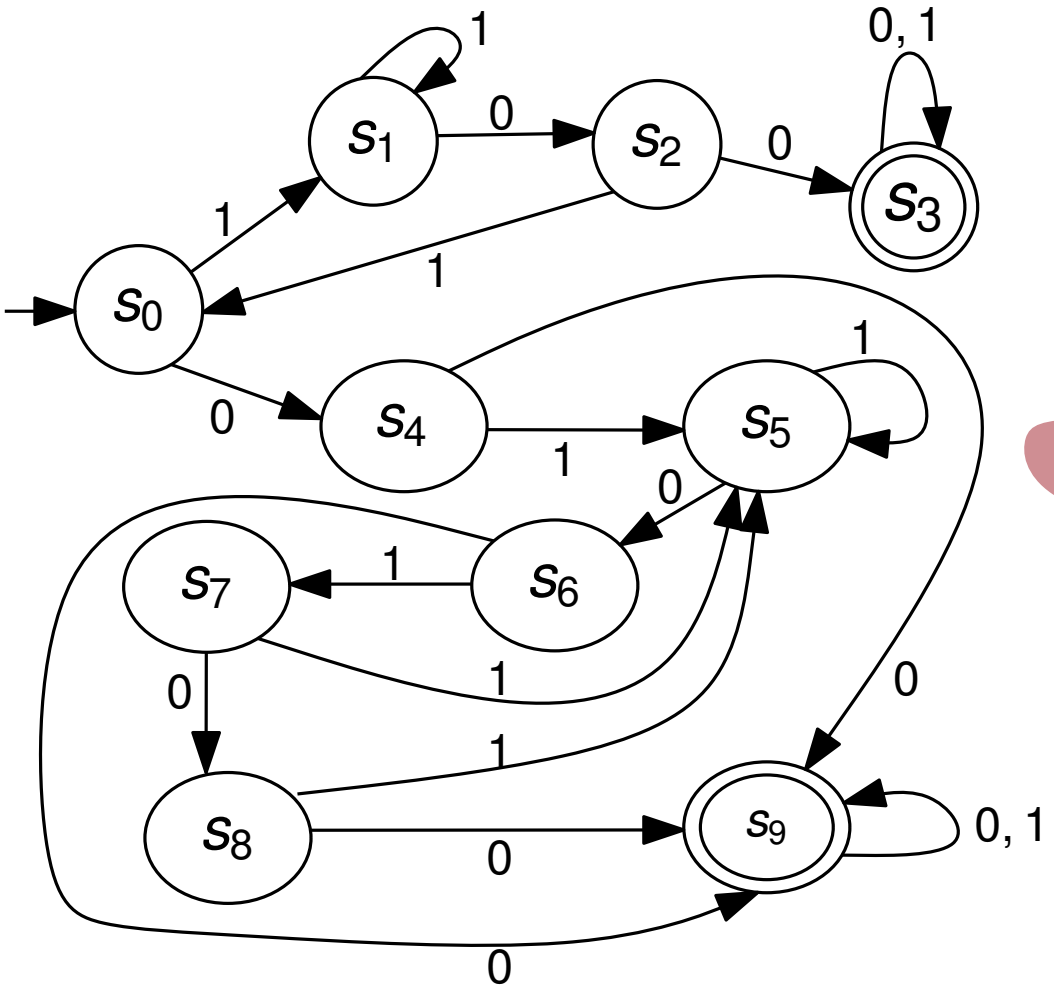
1 trennt nichts
00 trennt nichts
01 trennt nichts
10 trennt nichts
11 trennt nichts

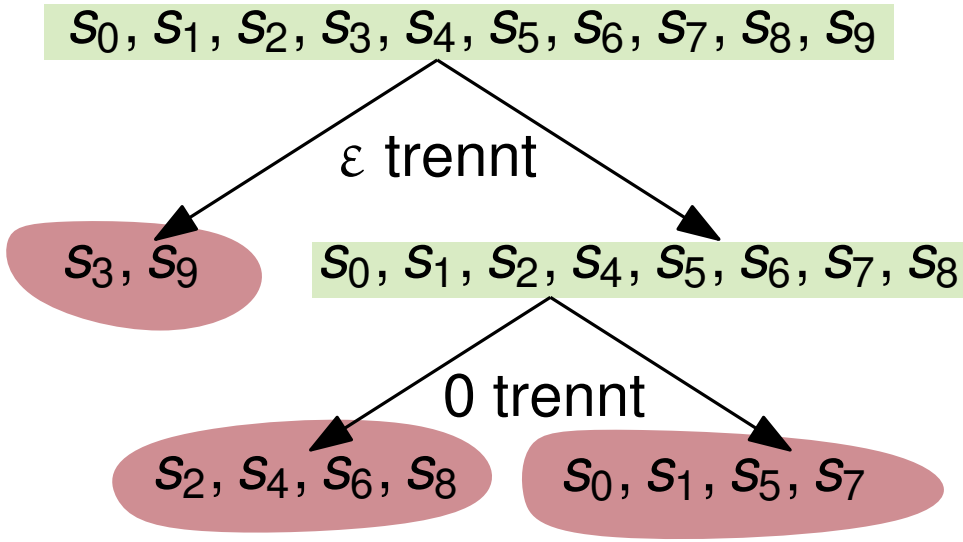
Betrachte Wort $w =$ 11

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik





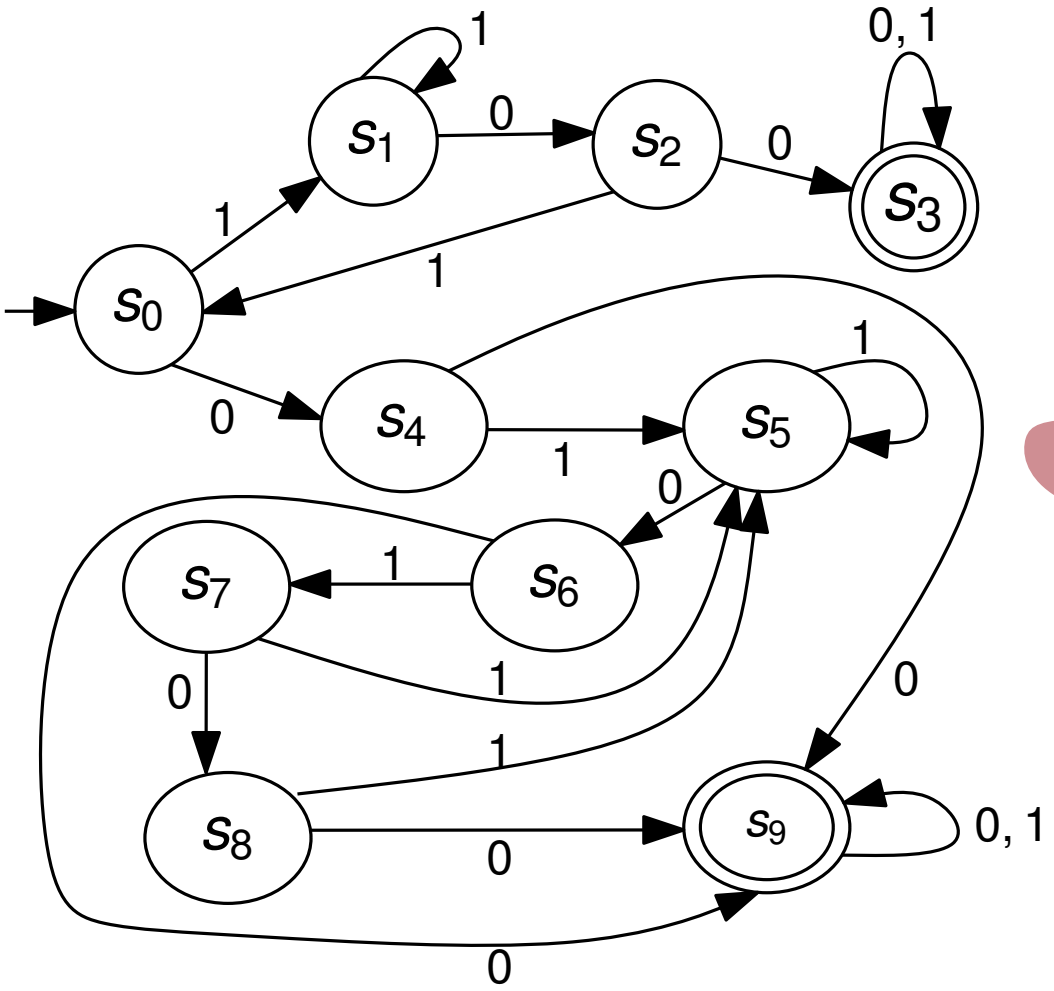
3 Äquivalenzklassen!

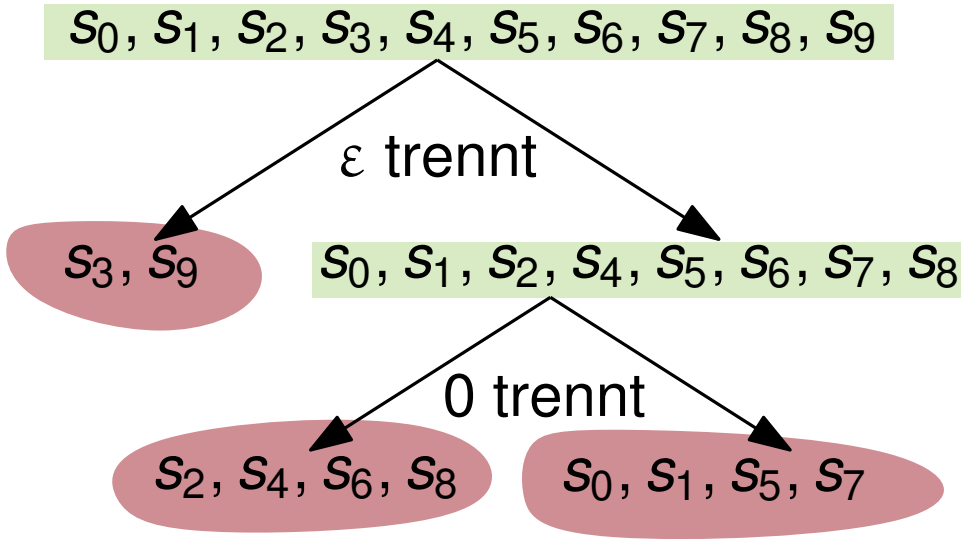
Betrachte Wort w = 11

$\delta(s_i, w)$ endet in Endzustand.

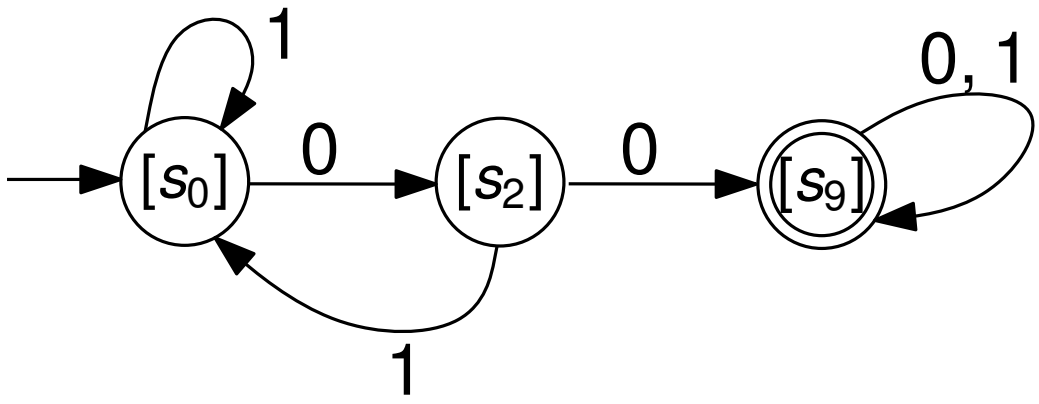
$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik





3 Äquivalenzklassen!



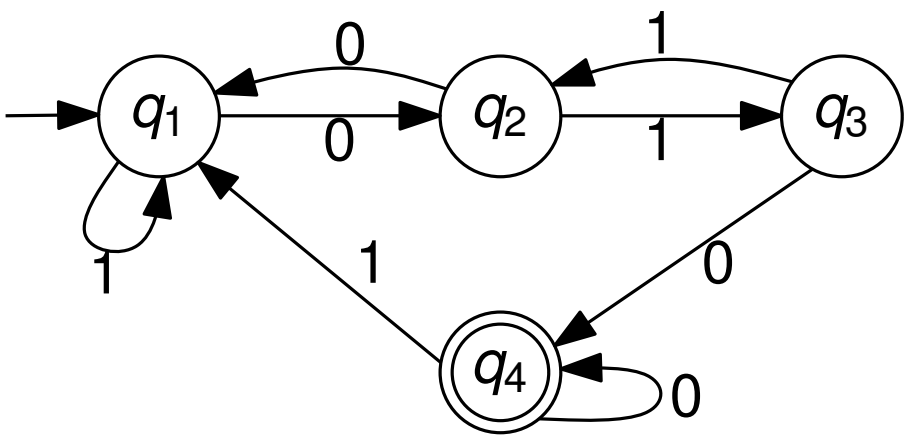
Betrachte Wort w = 11

$\delta(s_i, w)$ endet in Endzustand.

$\delta(s_i, w)$ endet nicht in Endzustand.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

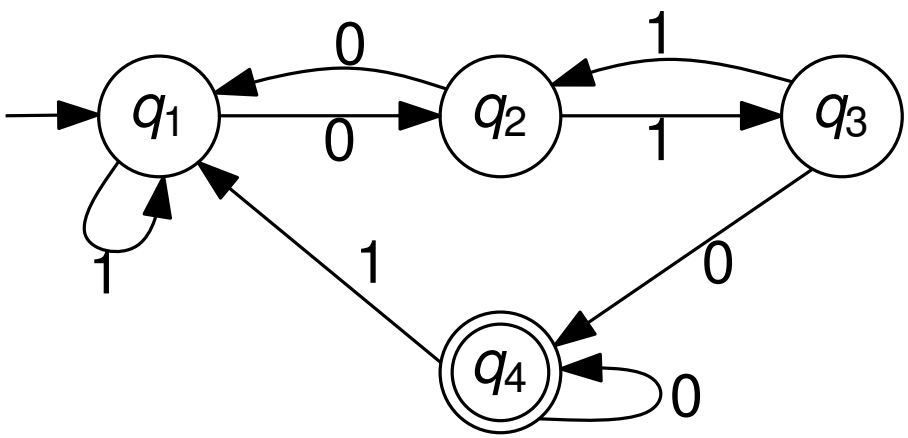
Minimierung von Automaten



Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten



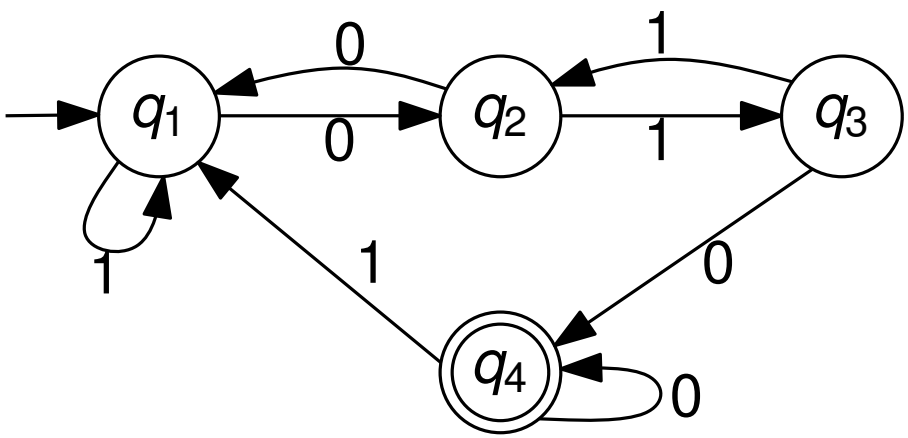
Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.

q_1, q_2, q_3, q_4

Betrachte: ε, 0, 1, 01, 10, 11,. . .

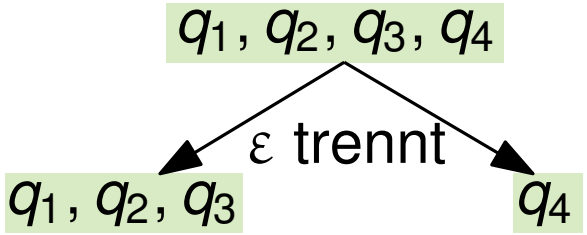
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

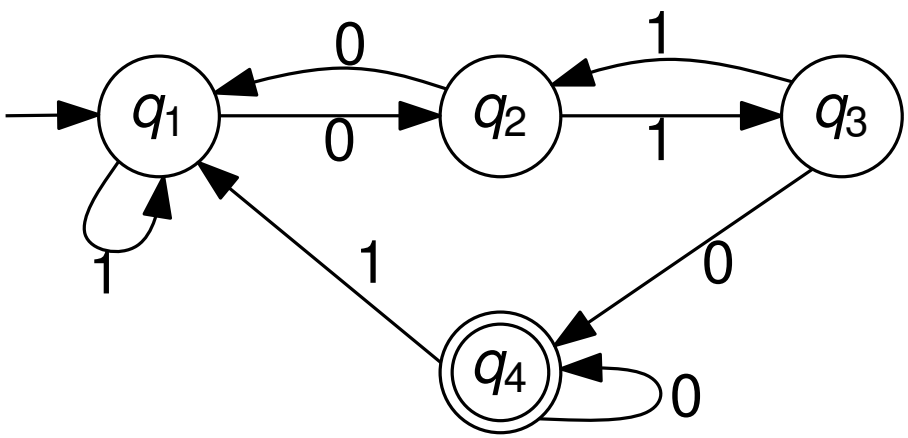


Betrachte: ε, 0, 1, 01, 10, 11,. . .

Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.

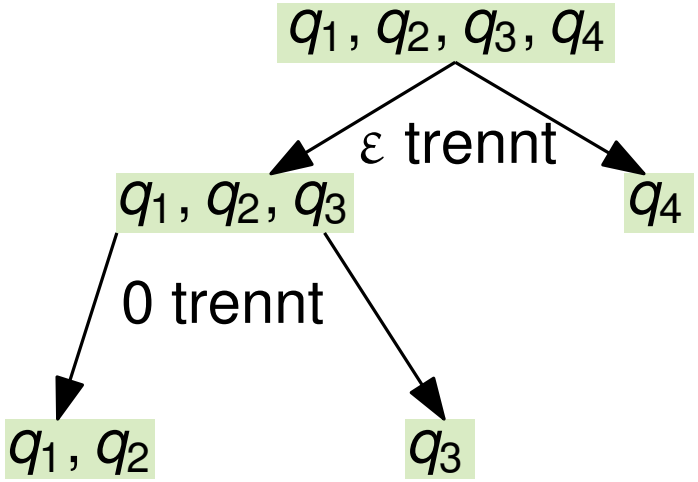


Minimierung von Automaten



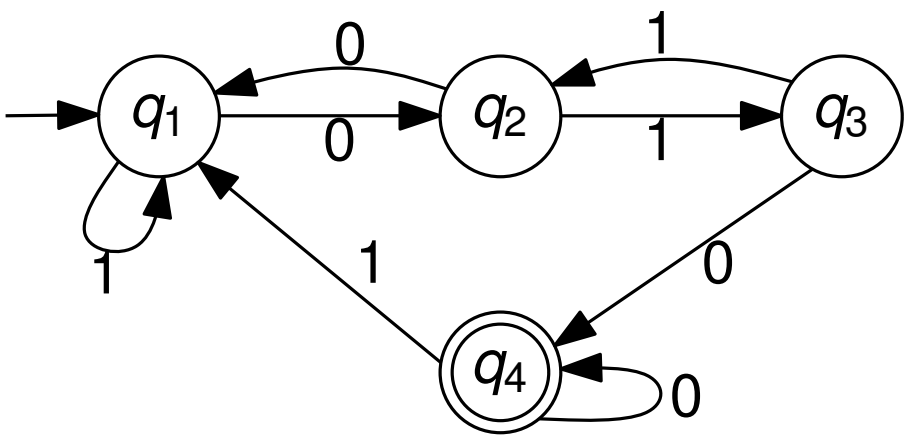
Betrachte: ε, 0, 1, 01, 10, 11,. . .

Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

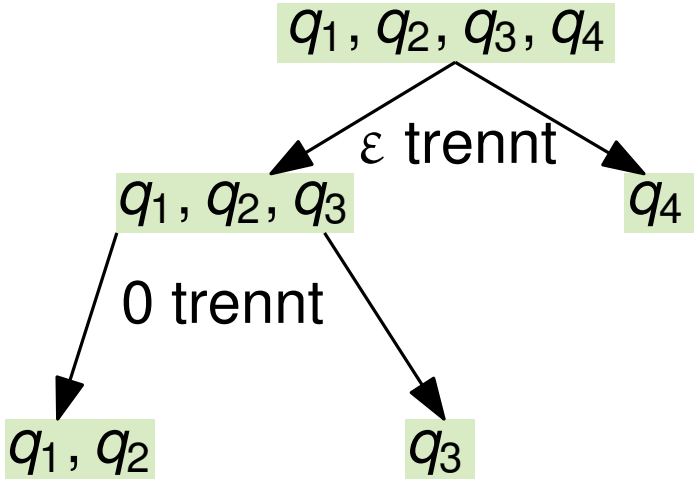
Minimierung von Automaten



Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.

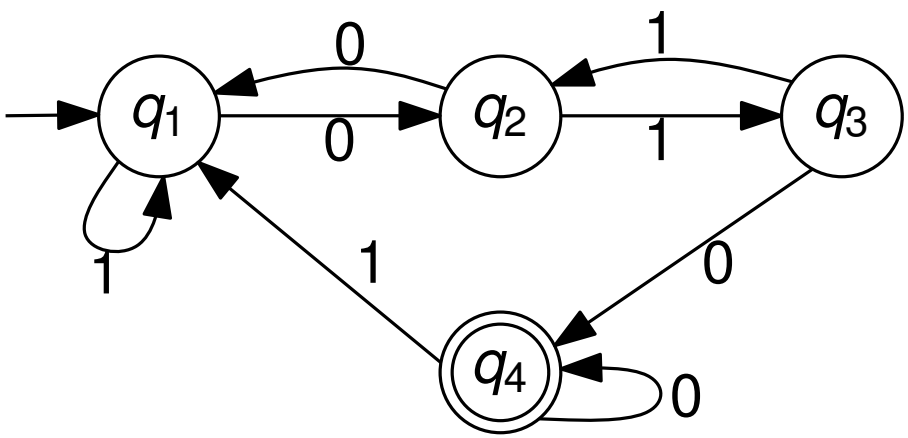
Betrachte: ε, 0, 1, 01, 10, 11,. . .

1 trennt nichts



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

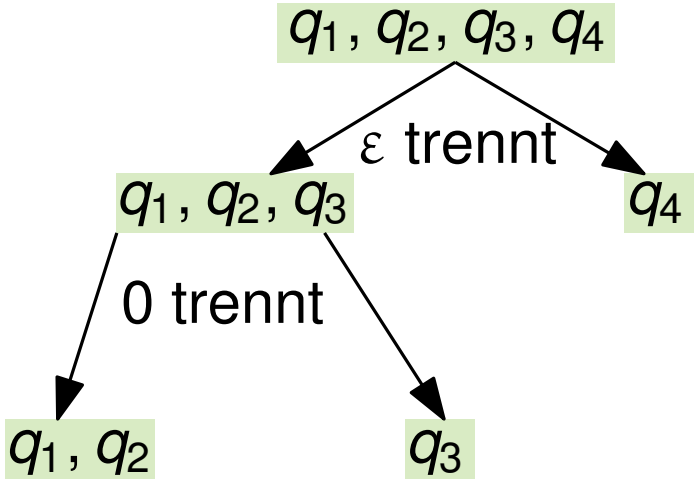


Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.

Betrachte: ε, 0, 1, 01, 10, 11,. . .

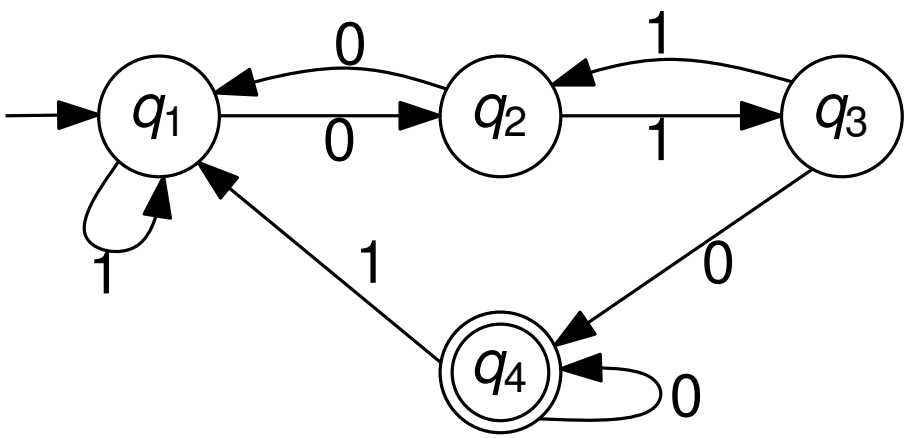
1 trennt nichts

01 trennt nichts



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Minimierung von Automaten

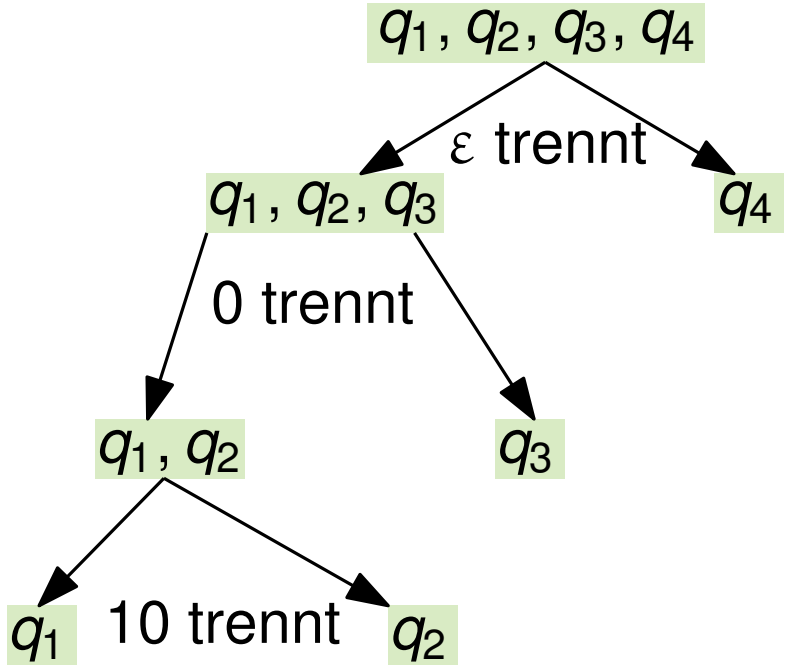


Zeigen Sie, dass dieser Automat bezüglich der Äquivalenzklassenkonstruktion minimal ist.

Betrachte: ε, 0, 1, 01, 10, 11,. . .

1 trennt nichts

01 trennt nichts



4 Äquivalenzklassen!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nerode-Relation

Nerode-Relation R_L einer Sprache L:

Für $x, y \in \Sigma^*$ gilt:

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Beispiel: $\Sigma = \{a, b\}, L = (aba)^*$

- $a \, R_L \, aba$?
- $\varepsilon \, R_L \, aba$?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nerode-Relation

Nerode-Relation R_L einer Sprache L:

Für $x, y \in \Sigma^*$ gilt:

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Beispiel: $\Sigma = \{a, b\}, L = (aba)^*$

- $a \, R_L \, aba$? Nein, denn: $a\,ba \in L$, aber $aba\,ba \notin L$
- $\varepsilon \, R_L \, aba$?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nerode-Relation

Nerode-Relation R_L einer Sprache L:

Für $x, y \in \Sigma^*$ gilt:

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Beispiel: $\Sigma = \{a, b\}, L = (aba)^*$

- $a \, R_L \, aba$? Nein, denn: $a\,ba \in L$, aber $aba\,ba \notin L$
- $\varepsilon \, R_L \, aba$? Ja, denn für jedes $z \in \Sigma^*$: $\varepsilon\, z \in L \Leftrightarrow aba\, z \in L$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nerode-Relation

Nerode-Relation R_L einer Sprache L:

Für $x, y \in \Sigma^*$ gilt:

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Nerode-Relation ist Äquivalenzrelation mit Äquivalenzklassen

$$[x] = \{y \in \Sigma^* \mid x \, R_L \, y\}.$$

Index $\text{ind}(R_L)$ ist Anzahl der Äquivalenzklassen.

Satz von Nerode

$$L \text{ regulär} \quad \Longleftrightarrow \quad \text{ind}(R_L) < \infty$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

Sprache der alternierenden Bitfolgen:
$0101010 \in L$, $101010 \in L$, $0100101 \notin L$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$$S_\varepsilon = L$$
$$S_0 = (10)^*(1 \cup \varepsilon)$$
$$S_1 = (01)^*(0 \cup \varepsilon)$$
$$S_{00} = S_{11} = \emptyset$$
$$S_{01} = (01)^*(0 \cup \varepsilon)$$
$$S_{10} = (10)^*(1 \cup \varepsilon)$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$S_\varepsilon = L$

$S_0 = (10)^*(1 \cup \varepsilon)$

$S_1 = (01)^*(0 \cup \varepsilon)$

$S_{00} = S_{11} = \emptyset$

$S_{01} = (01)^*(0 \cup \varepsilon)$

$S_{10} = (10)^*(1 \cup \varepsilon)$

$[\varepsilon] = \{\varepsilon, \ldots\} =$

$[0] = \{0, 10, \ldots\} =$

$[1] = \{1, 01, \ldots\} =$

$[00] = \{00, 11, \ldots\} =$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$S_\varepsilon = L$

$S_0 = (10)^*(1 \cup \varepsilon)$

$S_1 = (01)^*(0 \cup \varepsilon)$

$S_{00} = S_{11} = \emptyset$

$S_{01} = (01)^*(0 \cup \varepsilon)$

$S_{10} = (10)^*(1 \cup \varepsilon)$

$[\varepsilon] = \{\varepsilon, \ldots\} = \{\varepsilon\}$

$[0] = \{0, 10, \ldots\} =$

$[1] = \{1, 01, \ldots\} =$

$[00] = \{00, 11, \ldots\} =$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$S_\varepsilon = L$

$S_0 = (10)^*(1 \cup \varepsilon)$

$S_1 = (01)^*(0 \cup \varepsilon)$

$S_{00} = S_{11} = \emptyset$

$S_{01} = (01)^*(0 \cup \varepsilon)$

$S_{10} = (10)^*(1 \cup \varepsilon)$

$[\varepsilon] = \{\varepsilon, \ldots\} = \{\varepsilon\}$

$[0] = \{0, 10, \ldots\} = 0(10)^* \cup (10)^+$

$[1] = \{1, 01, \ldots\} =$

$[00] = \{00, 11, \ldots\} =$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$$S_\varepsilon = L$$
$$S_0 = (10)^*(1 \cup \varepsilon)$$
$$S_1 = (01)^*(0 \cup \varepsilon)$$
$$S_{00} = S_{11} = \emptyset$$
$$S_{01} = (01)^*(0 \cup \varepsilon)$$
$$S_{10} = (10)^*(1 \cup \varepsilon)$$

$$[\varepsilon] = \{\varepsilon, \dots\} = \{\varepsilon\}$$
$$[0] = \{0, 10, \dots\} = 0(10)^* \cup (10)^+$$
$$[1] = \{1, 01, \dots\} = 1(01)^* \cup (01)^+$$
$$[00] = \{00, 11, \dots\} =$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$$S_\varepsilon = L$$
$$S_0 = (10)^*(1 \cup \varepsilon)$$
$$S_1 = (01)^*(0 \cup \varepsilon)$$
$$S_{00} = S_{11} = \emptyset$$
$$S_{01} = (01)^*(0 \cup \varepsilon)$$
$$S_{10} = (10)^*(1 \cup \varepsilon)$$

$$[\varepsilon] = \{\varepsilon, \dots\} = \{\varepsilon\}$$
$$[0] = \{0, 10, \dots\} = 0(10)^* \cup (10)^+$$
$$[1] = \{1, 01, \dots\} = 1(01)^* \cup (01)^+$$
$$[00] = \{00, 11, \dots\} = (0 \cup 1)^*(00 \cup 11)(0 \cup 1)^*$$

Alternative Sichtweise

$$x \, R_L \, y \quad \Longleftrightarrow \quad (\forall z \in \Sigma^* : xz \in L \Leftrightarrow yz \in L)$$

Definiere für $x \in \Sigma^*$ Menge der gültigen Suffixe

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$

Dann:

$$x \, R_L \, y \quad \Longleftrightarrow \quad S_x = S_y$$

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$$S_\varepsilon = L$$
$$S_0 = (10)^*(1 \cup \varepsilon)$$
$$S_1 = (01)^*(0 \cup \varepsilon)$$
$$S_{00} = S_{11} = \emptyset$$
$$S_{01} = (01)^*(0 \cup \varepsilon)$$
$$S_{10} = (10)^*(1 \cup \varepsilon)$$

$$[\varepsilon] = \{\varepsilon, \ldots\} = \{\varepsilon\}$$
$$[0] = \{0, 10, \ldots\} = 0(10)^* \cup (10)^+$$
$$[1] = \{1, 01, \ldots\} = 1(01)^* \cup (01)^+$$
$$[00] = \{00, 11, \ldots\} = (0 \cup 1)^*(00 \cup 11)(0 \cup 1)^*$$

$\Rightarrow$ 4 Äquivalenzklassen

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Automat der Nerode-Relation

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$[\varepsilon] = \{\varepsilon\}$

$[0] = 0(10)^* \cup (10)^+$

$[1] = 1(01)^* \cup (01)^+$

$[00] = (0 \cup 1)^*(00 \cup 11)(0 \cup 1)^*$

$$Q = \{[x] \mid x \in \Sigma^*\}$$

$$\delta([x], a) = [xa]$$

$$s = [\varepsilon]$$

$$F = \{[w] \mid w \in L\}$$

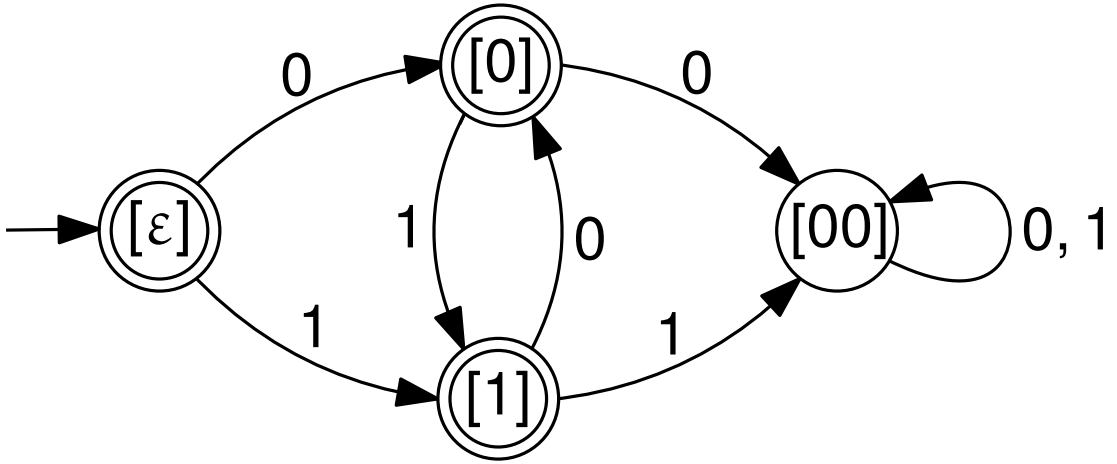
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Automat der Nerode-Relation

Beispiel: $\Sigma = \{0, 1\}, L = (0 \cup \varepsilon) \cdot (10)^* \cdot (1 \cup \varepsilon)$

$$[\varepsilon] = \{\varepsilon\}$$
$$[0] = 0(10)^* \cup (10)^+$$
$$[1] = 1(01)^* \cup (01)^+$$
$$[00] = (0 \cup 1)^*(00 \cup 11)(0 \cup 1)^*$$

$$Q = \{[x] \mid x \in \Sigma^*\}$$
$$\delta([x], a) = [xa]$$
$$s = [\varepsilon]$$
$$F = \{[w] \mid w \in L\}$$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Nerode-Relation

$$S_x = \{z \in \Sigma^* \mid xz \in L\}$$
$$x \, R_L \, y \iff S_x = S_y$$

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

Geben Sie die Äquivalenzklassen von R_L an. Ist L regulär?

Für welche Wörter ist S_x nicht leer? $P = \{a^{i+j} b^j \mid i, j \in \mathbb{N}_0\}$

$\Rightarrow [b] = \Sigma^* \setminus P$, $S_b = \emptyset$

	x	$xz \in L$	S_x
$j = 0$:	$x = a^i$	$a^i\, b^i$ $\quad$ $a^i\, a^j b^j b^i$	$S_x = \{a^j b^{i+j} \mid j \in \mathbb{N}_0\}$
$j > 0$:	$x = a^{i+j} b^j$	$a^{i+j} b^j\, b^i$	$S_x = \{b^i\}$

S_x hängt nicht von j ab $\Rightarrow$ $[a^{i+1} b] = \{a^{i+j} b^i \mid j \in \mathbb{N}_0\}$, $[a^i] = \{a^i\}$ } 2 Klassen für jedes $i \in \mathbb{N}_0$

$\Rightarrow \text{ind}(R_L) = \infty$, L nicht regulär

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

		1	2	3	4	5	6	7	8	9	10
$f(1) =$	0,	9	9	6	8	8	3	6	0	4	7
$f(2) =$	0,	1	2	3	9	8	1	2	6	4	9
$f(3) =$	0,	8	2	7	3	7	4	5	8	0	1
$f(4) =$	0,	1	7	8	7	4	9	0	3	0	7
$f(5) =$	0,	7	9	8	4	3	2	9	3	2	3

$x = 0,$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

		1	2	3	4	5	6	7	8	9	10
$f(1) =$	0,	9	9	6	8	8	3	6	0	4	7
$f(2) =$	0,	1	2	3	9	8	1	2	6	4	9
$f(3) =$	0,	8	2	7	3	7	4	5	8	0	1
$f(4) =$	0,	1	7	8	7	4	9	0	3	0	7
$f(5) =$	0,	7	9	8	4	3	2	9	3	2	3
$x =$	0,	7									

$\rightarrow x \neq f(1)$

Idee: Wähle 7 falls möglich, sonst 0.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

		1	2	3	4	5	6	7	8	9	10	
$f(1) =$	0,	9	9	6	8	8	3	6	0	4	7	$\rightarrow x \neq f(1)$
$f(2) =$	0,	1	2	3	9	8	1	2	6	4	9	$\rightarrow x \neq f(2)$
$f(3) =$	0,	8	2	7	3	7	4	5	8	0	1	
$f(4) =$	0,	1	7	8	7	4	9	0	3	0	7	
$f(5) =$	0,	7	9	8	4	3	2	9	3	2	3	
$x =$	0,	7	7									

Idee: Wähle 7 falls möglich, sonst 0.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

		1	2	3	4	5	6	7	8	9	10	
$f(1) =$	0,	9	9	6	8	8	3	6	0	4	7	$\rightarrow x \neq f(1)$
$f(2) =$	0,	1	2	3	9	8	1	2	6	4	9	$\rightarrow x \neq f(2)$
$f(3) =$	0,	8	2	7	3	7	4	5	8	0	1	$\rightarrow x \neq f(3)$
$f(4) =$	0,	1	7	8	7	4	9	0	3	0	7	
$f(5) =$	0,	7	9	8	4	3	2	9	3	2	3	
$x =$	0,	7	7	0								

Idee: Wähle 7 falls möglich, sonst 0.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

		1	2	3	4	5	6	7	8	9	10	
$f(1) =$	0,	9	9	6	8	8	3	6	0	4	7	$\rightarrow x \neq f(1)$
$f(2) =$	0,	1	2	3	9	8	1	2	6	4	9	$\rightarrow x \neq f(2)$
$f(3) =$	0,	8	2	7	3	7	4	5	8	0	1	$\rightarrow x \neq f(3)$
$f(4) =$	0,	1	7	8	7	4	9	0	3	0	7	$\rightarrow x \neq f(4)$
$f(5) =$	0,	7	9	8	4	3	2	9	3	2	3	
$x =$	0,	7	7	0	0							

Idee: Wähle 7 falls möglich, sonst 0.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

	1	2	3	4	5	6	7	8	9	10	
$f(1) =$	0, 9	9	6	8	8	3	6	0	4	7	$\to x \neq f(1)$
$f(2) =$	0, 1	2	3	9	8	1	2	6	4	9	$\to x \neq f(2)$
$f(3) =$	0, 8	2	7	3	7	4	5	8	0	1	$\to x \neq f(3)$
$f(4) =$	0, 1	7	8	7	4	9	0	3	0	7	$\to x \neq f(4)$
$f(5) =$	0, 7	9	8	4	3	2	9	3	2	3	$\to x \neq f(5)$
$x =$	0, 7	7	0	0	7	...					

Idee: Wähle 7 falls möglich, sonst 0.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0,1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0,1)$. Konstruiere Zahl $x \in (0,1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

		1	2	3	4	5	6	7	8	9	10
$f(1) =$	0,	9	9	6	8	8	3	6	0	4	7
$f(2) =$	0,	1	2	3	9	8	1	2	6	4	9
$f(3) =$	0,	8	2	7	3	7	4	5	8	0	1
$f(4) =$	0,	1	7	8	7	4	9	0	3	0	7
$f(5) =$	0,	7	9	8	4	3	2	9	3	2	3
$x =$	0,	7	7	0	0	7	...				

Allgemein: bezeichne y_i den Wert der i-ten Nachkommastelle von y.

$$x_i := \begin{cases} 7 & \text{falls } f(i)_i \neq 7 \\ 0 & \text{falls } f(i)_i = 7 \end{cases}$$

$\rightarrow x \notin \text{Bild}(f)$

Idee: Wähle 7 falls möglich, sonst 0.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Cantors 2. Diagonalargument

Theorem: Die Menge der reellen Zahlen ist überabzählbar.

Eine Menge heißt abzählbar, wenn sie entweder endlich ist oder eine Bijektion zur Menge der natürlichen Zahlen existiert.

Diagonalargument: Nehme an, $(0, 1)$ wäre abzählbar mit Bijektion $f : \mathbb{N} \to (0, 1)$. Konstruiere Zahl $x \in (0, 1)$ mit $x \notin \text{Bild}(f)$. Widerspr.

Wieso wir das machen...

In der Vorlesung wird bald die **Diagonalsprache** vorgestellt, die von Turing-Maschinen nicht erkannt werden kann.
Deren Konstruktion ist sehr ähnlich zu Cantors 2. Diagonalargument!