Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik
Übung

1. Übungstermin · 25. Oktober 2018
Jonas Sauer

Institut für Theoretische Informatik · Lehrstuhl Algorithmik

Übungsleiter

ab ca. Mitte November:

Jonas Sauer
jonas.sauer2@kit.edu
Raum 317

Guido Brückner
brueckner@kit.edu
Raum 317

Sprechzeiten: Termin nach Vereinbarung

Hinweis

Tutorium 6 von Liran Dattner (Dienstag 14:00-15:30 Uhr) wurde von SR -118 in SR -107 verlegt!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Gliederung

Organisatorisches

- Übungsblätter

Inhalt

- Formale Sprachen und reguläre Ausdrücke
- Nichtdeterministische endliche Automaten
- Eigenschaften von endlichen Automaten
- Potenzmengenkonstruktion
- Entfernen von ε-Übergängen
- Kontextfreie Grammatiken

Übungsblatt (ÜB)

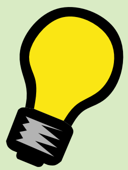
Aus- und Abgabe: jeden zweiten Dienstag

Abgabe um **11:00 Uhr** im Kasten im UG von Gebäude 50.34

Tag der Abgabe für aktuelles ÜB

=

Tag der Ausgabe für nächstes ÜB



Ab 25% der erreichbaren Punkte gibt es 1 Bonuspunkt auf die **bestandene** Klausur.

- ab 50% gibt es 2 Bonuspunkte
- ab 75% gibt es 3 Bonuspunkte

Übungsblattabgabe

Titelblatt:

WebInScribe Deckblattgenerator benutzen!

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Übungsblattabgabe

Titelblatt:

WebInScribe Deckblattgenerator benutzen!

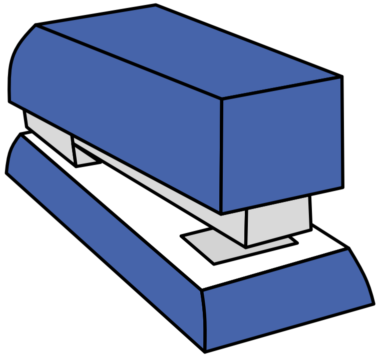
Handschriftliche Abgabe!

Übungsblattabgabe

Titelblatt:

WebInScribe Deckblattgenerator benutzen!

Handschriftliche Abgabe!



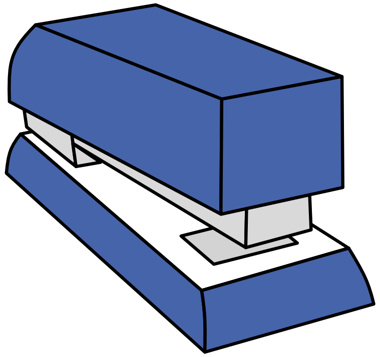
Übungsblatt heften!

Übungsblattabgabe

Titelblatt:

WebInScribe Deckblattgenerator benutzen!

Handschriftliche Abgabe!



Übungsblatt heften!

Doppelabgabe ist erlaubt.

→ Beide müssen im selben Tutorium angemeldet sein.

Gliederung

Organisatorisches

- Übungsblätter

Inhalt

- Formale Sprachen und reguläre Ausdrücke
- Nichtdeterministische endliche Automaten
- Eigenschaften von endlichen Automaten
- Potenzmengenkonstruktion
- Entfernen von ε-Übergängen
- Kontextfreie Grammatiken

Formale Sprachen

Alphabet: Endliche Menge Σ an Zeichen/Symbolen.

Beispiel: $\Sigma = \{0, 1\}$

Wort: Ein Wort w über dem Alphabet Σ ist eine (endliche) Folge von Zeichen aus Σ.

Beispiel: $\Sigma = \{0, 1\}$, w=0101, w=ε

Leere Wort: ε

Alle Worte: $\Sigma^\star$

Beispiel: $\Sigma = \{0, 1\}$, $\Sigma^\star = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$

Sprache: (beliebige) Menge an Worten über einem Alphabet Σ

Beispiel: $\Sigma = \{0, 1\}$

$$L_1 = \{00, 01, 10, 11\}$$

$$L_2 = \{w \in \Sigma^\star \mid w \text{ endet mit } 0\}$$

Formale Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^\star$ Sprachen.

Produktsprache	$L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
k-faches Produkt	$L^k := \{w_1 \cdot w_2 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^0 := \{\varepsilon\}$
Kleene'scher Abschluss	$L^\star := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Quotientensprache	$L_1 / L_2 := \{w \in \Sigma^\star \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Komplementsprache	$L^c := \Sigma^\star \setminus L$

Beispiel: Produktsprache

$L_1 = \{01, 10, 11\}$

$L_2 = \{1, 00, 101\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Formale Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^\star$ Sprachen.

Produktsprache	$L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
k-faches Produkt	$L^k := \{w_1 \cdot w_2 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^0 := \{\varepsilon\}$
Kleene'scher Abschluss	$L^\star := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Quotientensprache	$L_1 / L_2 := \{w \in \Sigma^\star \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Komplementsprache	$L^c := \Sigma^\star \setminus L$

Beispiel: Produktsprache

$L_1 = \{01, 10, 11\}$

$L_2 = \{1, 00, 101\}$

$L_1 \cdot L_2 = \{011, 0100, 01101,$

$\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Formale Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^\star$ Sprachen.

Produktsprache	$L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
k-faches Produkt	$L^k := \{w_1 \cdot w_2 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^0 := \{\varepsilon\}$
Kleene'scher Abschluss	$L^\star := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Quotientensprache	$L_1 / L_2 := \{w \in \Sigma^\star \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Komplementsprache	$L^c := \Sigma^\star \setminus L$

Beispiel: Produktsprache

$L_1 = \{01, 10, 11\}$

$L_2 = \{1, 00, 101\}$

$L_1 \cdot L_2 = \{011, 0100, 01101, 101, 1000, 10101, \}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Formale Sprachen

Seien $L, L_1, L_2 \subseteq \Sigma^\star$ Sprachen.

Produktsprache	$L_1 \cdot L_2 := \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
k-faches Produkt	$L^k := \{w_1 \cdot w_2 \cdot \ldots \cdot w_k \mid w_i \in L \text{ für } 1 \leq i \leq k\}$ $L^0 := \{\varepsilon\}$
Kleene'scher Abschluss	$L^\star := \bigcup_{i \geq 0} L^i$
Positiver Abschluss	$L^+ := \bigcup_{i \geq 1} L^i$
Quotientensprache	$L_1 / L_2 := \{w \in \Sigma^\star \mid \exists z \in L_2 \text{ mit } w \cdot z \in L_1\}$
Komplementsprache	$L^c := \Sigma^\star \setminus L$

Beispiel: Produktsprache

$L_1 = \{01, 10, 11\}$

$L_2 = \{1, 00, 101\}$

$L_1 \cdot L_2 = \{011, 0100, 01101, 101, 1000, 10101, 111, 1100, 11101\}$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reguläre Sprachen

Eine Sprache $L \subseteq \Sigma^\star$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

1. **Verankerung**:
   (a) $L = \{a\}$ mit $a \in \Sigma \cup \{\varepsilon\}$ oder
   (b) $L = \emptyset$
2. **Induktion**: Seien L_1, L_2 reguläre Sprachen
   (a) $L = L_1 \cdot L_2$ oder
   (b) $L = L_1 \cup L_2$ oder
   (c) $L = L_1^\star$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Reguläre Sprachen

Eine Sprache $L \subseteq \Sigma^\star$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

1. **Verankerung**:
   (a) $L = \{a\}$ mit $a \in \Sigma \cup \{\varepsilon\}$ oder
   (b) $L = \emptyset$
2. **Induktion**: Seien L_1, L_2 reguläre Sprachen
   (a) $L = L_1 \cdot L_2$ oder
   (b) $L = L_1 \cup L_2$ oder
   (c) $L = L_1^\star$

Reguläre Ausdrücke:

- $a \in \Sigma \cup \{\varepsilon\}$ für $L = \{a\}$
- $(\alpha) \cup (\beta)$ für $L(\alpha) \cup L(\beta)$
- $(\alpha) \cdot (\beta)$ für $L(\alpha) \cdot L(\beta)$
- $(\alpha)^+$ für $L(\alpha)^+$
- $(\alpha)^\star$ für $L(\alpha)^\star$

Reguläre Sprachen

Karlsruher Institut für Technologie

Eine Sprache $L \subseteq \Sigma^\star$ heißt *regulär*, wenn für sie einer der folgenden Punkte gilt: (induktive Definition)

1. **Verankerung**:
   (a) $L = \{a\}$ mit $a \in \Sigma \cup \{\varepsilon\}$ oder
   (b) $L = \emptyset$
2. **Induktion**: Seien L_1, L_2 reguläre Sprachen
   (a) $L = L_1 \cdot L_2$ oder
   (b) $L = L_1 \cup L_2$ oder
   (c) $L = L_1^\star$

Reguläre Ausdrücke:

- $a \in \Sigma \cup \{\varepsilon\}$ für $L = \{a\}$
- $(\alpha) \cup (\beta)$ für $L(\alpha) \cup L(\beta)$
- $(\alpha) \cdot (\beta)$ für $L(\alpha) \cdot L(\beta)$
- $(\alpha)^+$ für $L(\alpha)^+$
- $(\alpha)^\star$ für $L(\alpha)^\star$

Klammern werden häufig weggelassen.

Aufgabe

Seien A, B reguläre Ausdrücke. Welche regulären Ausdrücke beschreiben die gleiche Sprache?

(a) $(A^\star)^\star \stackrel{?}{=} A^\star$

(b) $(A \cup B)^\star \stackrel{?}{=} (A^\star B^\star)^\star$

(c) $A^\star \cup B^\star \stackrel{?}{=} (A \cup B)^\star$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

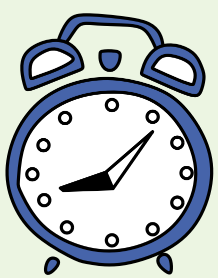
Karlsruher Institut für Technologie

Seien A, B reguläre Ausdrücke. Welche regulären Ausdrücke beschreiben die gleiche Sprache?

(a) $(A^\star)^\star \stackrel{?}{=} A^\star$

(b) $(A \cup B)^\star \stackrel{?}{=} (A^\star B^\star)^\star$

(c) $A^\star \cup B^\star \stackrel{?}{=} (A \cup B)^\star$



3 min Zeit

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Seien A, B reguläre Ausdrücke. Welche regulären Ausdrücke beschreiben die gleiche Sprache?

(a) $(A^\star)^\star \stackrel{?}{=} A^\star$ ✓

(b) $(A \cup B)^\star \stackrel{?}{=} (A^\star B^\star)^\star$

(c) $A^\star \cup B^\star \stackrel{?}{=} (A \cup B)^\star$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Seien A, B reguläre Ausdrücke. Welche regulären Ausdrücke beschreiben die gleiche Sprache?

(a) $(A^\star)^\star \stackrel{?}{=} A^\star$ ✓

(b) $(A \cup B)^\star \stackrel{?}{=} (A^\star B^\star)^\star$ ✓

(c) $A^\star \cup B^\star \stackrel{?}{=} (A \cup B)^\star$

Aufgabe

Seien A, B reguläre Ausdrücke. Welche regulären Ausdrücke beschreiben die gleiche Sprache?

(a) $(A^\star)^\star \stackrel{?}{=} A^\star$ ✓

(b) $(A \cup B)^\star \stackrel{?}{=} (A^\star B^\star)^\star$ ✓

(c) $A^\star \cup B^\star \stackrel{?}{=} (A \cup B)^\star$ ✗

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

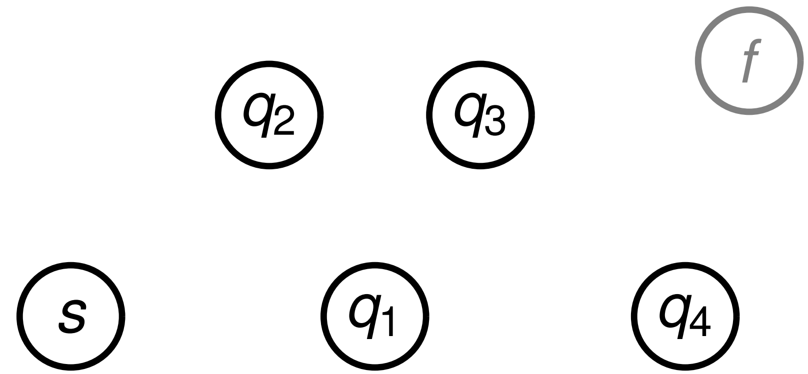
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

- Q: endliche Menge von *Zuständen*

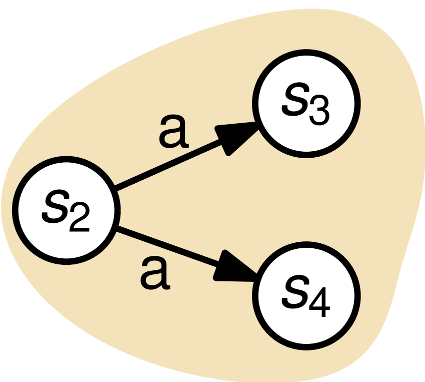


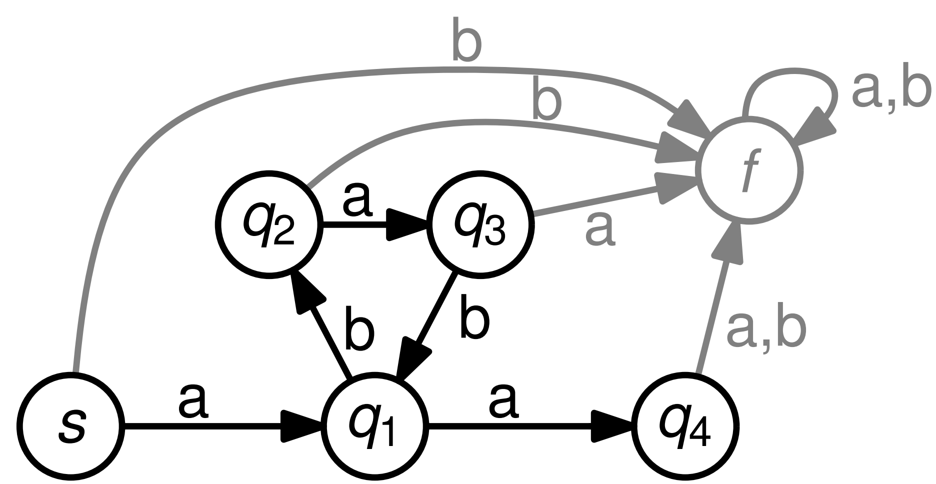
Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

- Q: endliche Menge von *Zuständen*
- Übergänge
  - eindeutig: $\delta\colon Q \times \Sigma \to Q$ deterministisch
  - mehrdeutig: $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ nicht-deterministisch



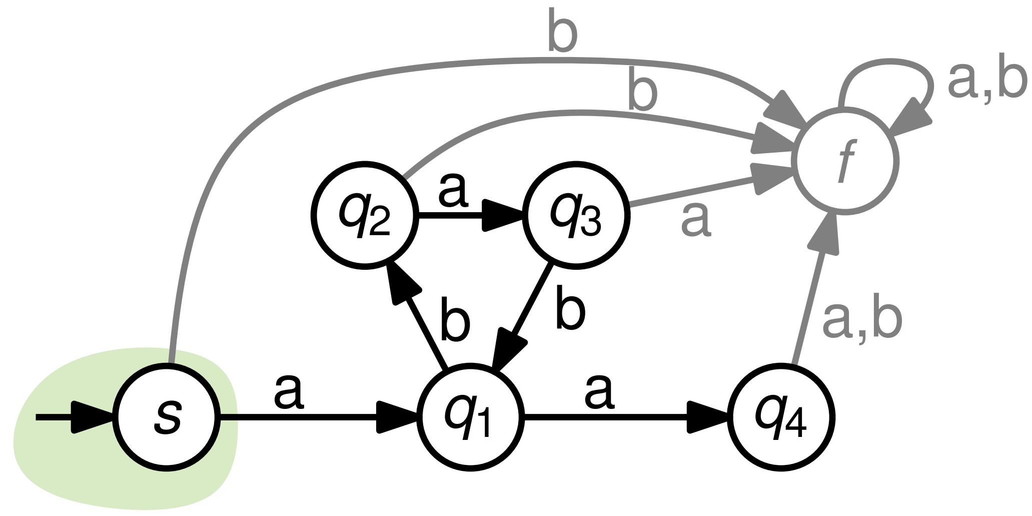


Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

- Q: endliche Menge von *Zuständen*
- Übergänge
  - eindeutig: $\delta : Q \times \Sigma \to Q$ deterministisch
  - mehrdeutig: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ nicht-deterministisch
- $s \in Q$: *Startzustand*

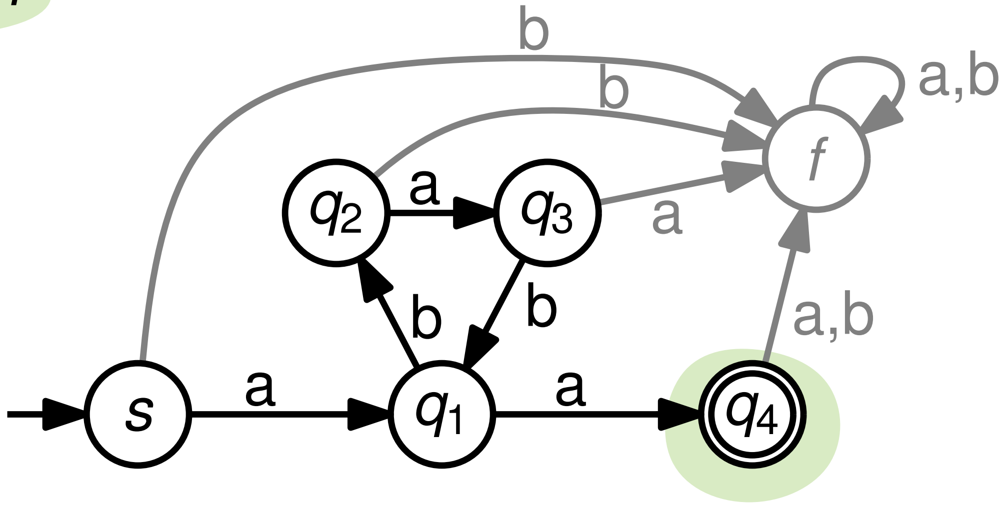


Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

- Q: endliche Menge von *Zuständen*
- Übergänge
  - eindeutig: $\delta : Q \times \Sigma \to Q$ deterministisch
  - mehrdeutig: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ nicht-deterministisch
- $s \in Q$: *Startzustand*
- $F \subseteq Q$, Menge von *Endzuständen*



Endliche Automaten

Definiert über einem Alphabet Σ

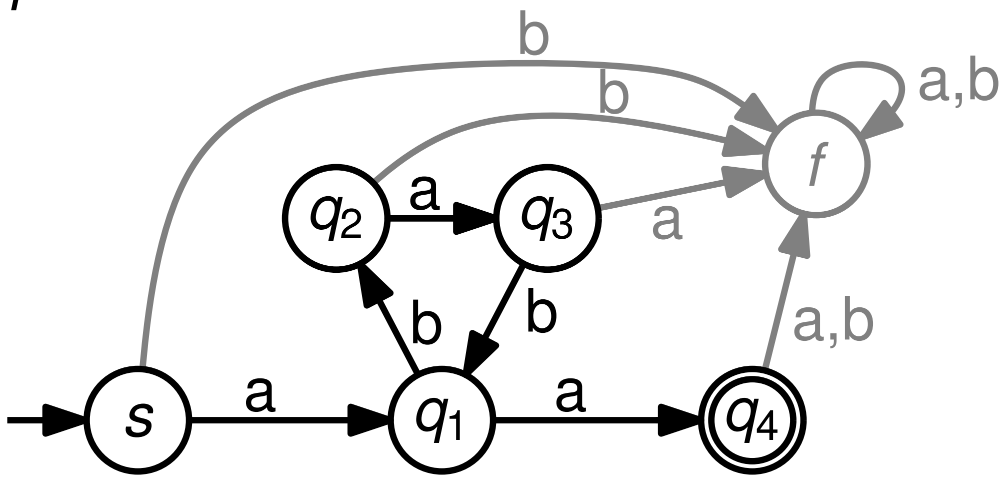
Bestehend aus:

- Q: endliche Menge von *Zuständen*
- Übergänge
  - eindeutig → $\delta\colon Q \times \Sigma \to Q$ deterministisch
  - mehrdeutig → $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ nicht-deterministisch
- $s \in Q$: *Startzustand*
- $F \subseteq Q$, Menge von *Endzuständen*

Akzeptanz des Wortes w:

DEA: Abarbeitung von w endet in Endzustand

NEA: Es gibt Abarbeitung von w, die in Endzustand endet



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

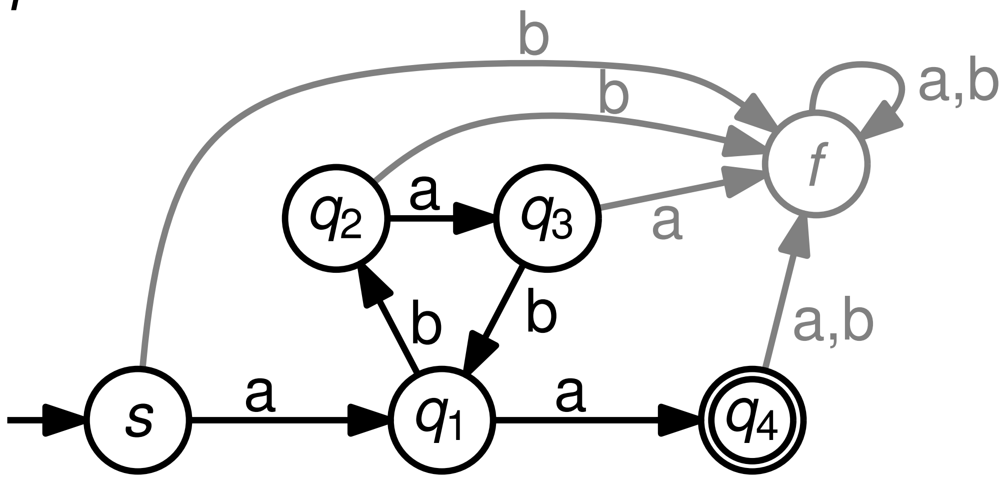
- Q: endliche Menge von *Zuständen*
- Übergänge
  - eindeutig: $\delta : Q \times \Sigma \to Q$ deterministisch
  - mehrdeutig: $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ nicht-deterministisch
- $s \in Q$: *Startzustand*
- $F \subseteq Q$, Menge von *Endzuständen*

Akzeptanz des Wortes w:

DEA: Abarbeitung von w endet in Endzustand

NEA: Es gibt Abarbeitung von w, die in Endzustand endet

Welche Sprache erkennt der Automat?



Endliche Automaten

Definiert über einem Alphabet Σ

Bestehend aus:

- Q: endliche Menge von *Zuständen*
- Übergänge
  - eindeutig → $\delta \colon Q \times \Sigma \to Q$ deterministisch
  - mehrdeutig → $\delta \colon Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ nicht-deterministisch
- $s \in Q$: *Startzustand*
- $F \subseteq Q$, Menge von *Endzuständen*

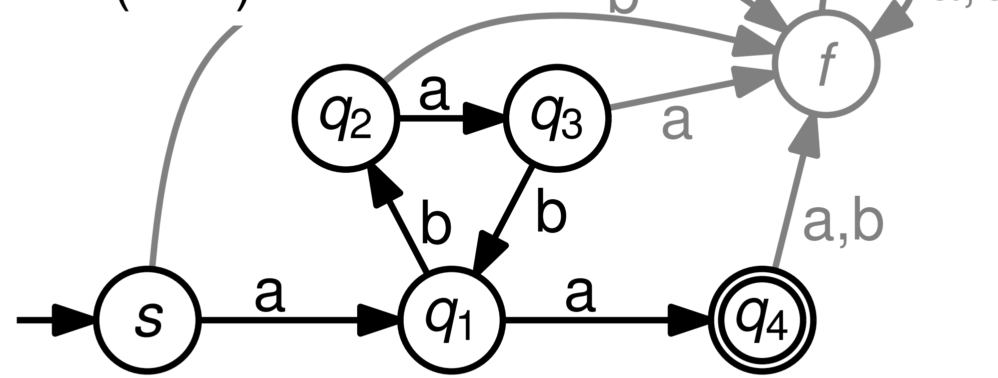
Welche Sprache erkennt der Automat?

Akzeptanz des Wortes w:

DEA: Abarbeitung von w endet in Endzustand

NEA: Es gibt Abarbeitung von w, die in Endzustand endet

$L = a(bab)^\star a$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (a \cup (ab(b)^\star ba))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (a \cup (ab(b)^\star ba))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**

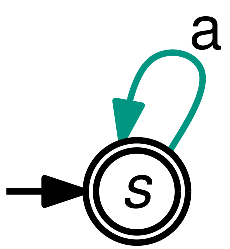
3 min Zeit

Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (a \cup (ab(b)^\star ba))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**



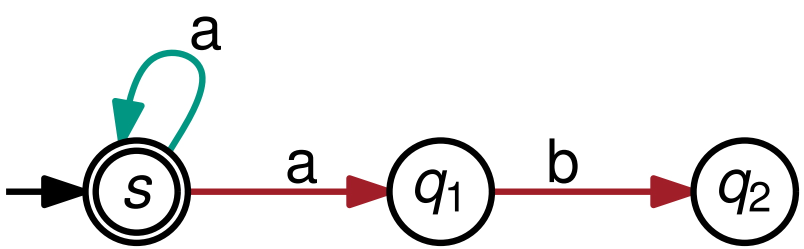
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (a \cup (ab(b)^\star ba))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**



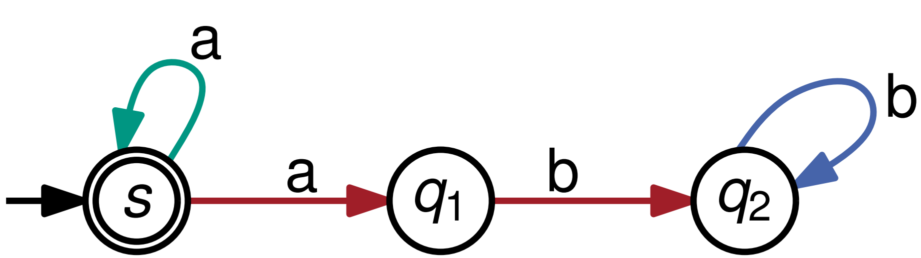
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (a \cup (ab(b)^\star ba))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**

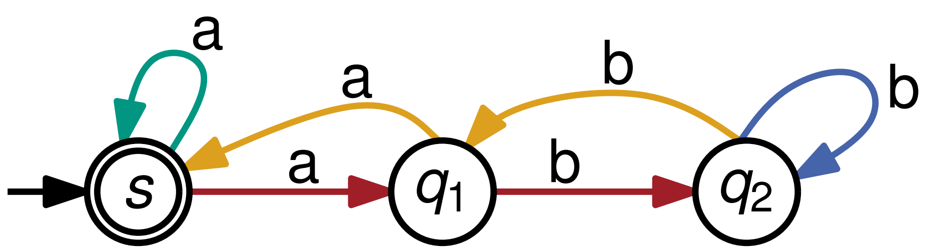


Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (\underline{a} \cup (\underline{ab}(\underline{b})^\star \underline{ba}))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**

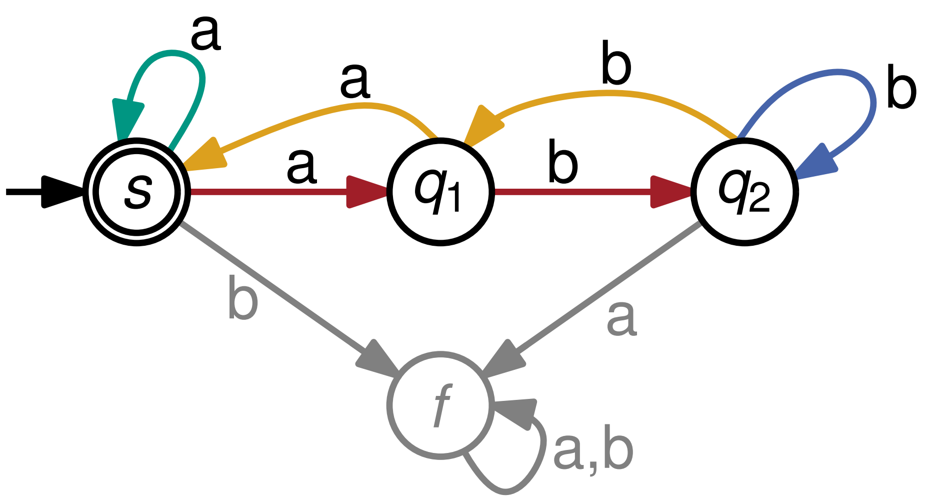


Aufgabe

Konstruieren Sie einen nicht-deterministischen endlichen Automaten, der folgende Sprache erkennt:

$$L = (\underline{a} \cup (\underline{ab}(\underline{b})^\star \underline{ba}))^\star$$

Verwenden Sie hierzu **maximal 3 Zustände + 1 Fehlerzustand.**



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid \text{ das } n\text{-letzte Zeichen von } w \text{ ist eine } 1\}$.

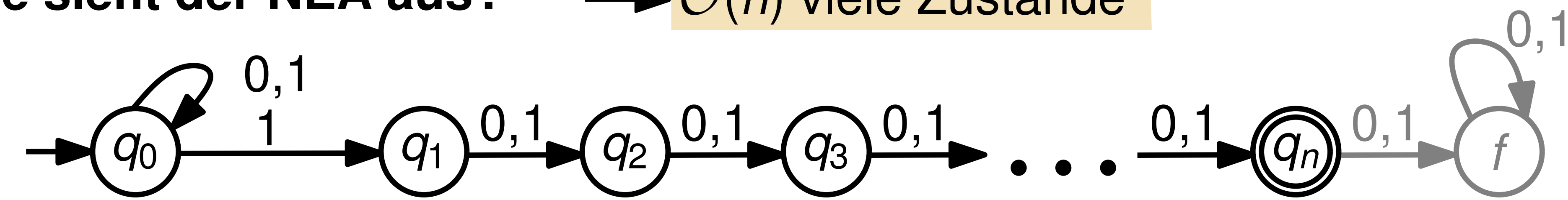
Wie sieht der NEA aus?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid \text{ das } n\text{-letzte Zeichen von } w \text{ ist eine } 1\}$.

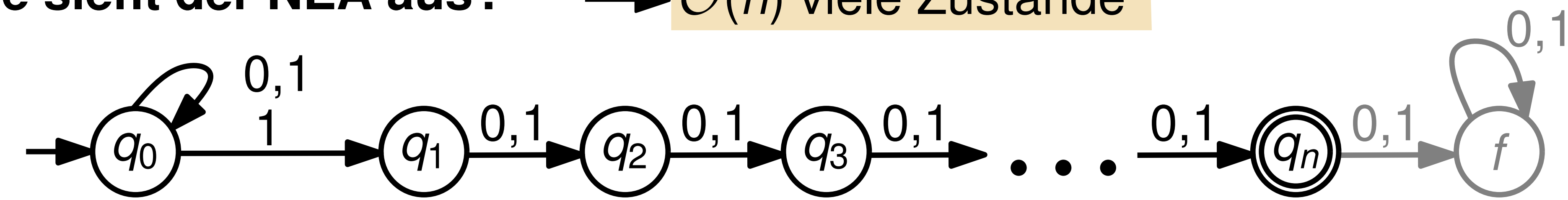
Wie sieht der NEA aus? → $\mathcal{O}(n)$ viele Zustände



Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid$ das n-letzte Zeichen von w ist eine 1$\}$.

Wie sieht der NEA aus? → $\mathcal{O}(n)$ viele Zustände

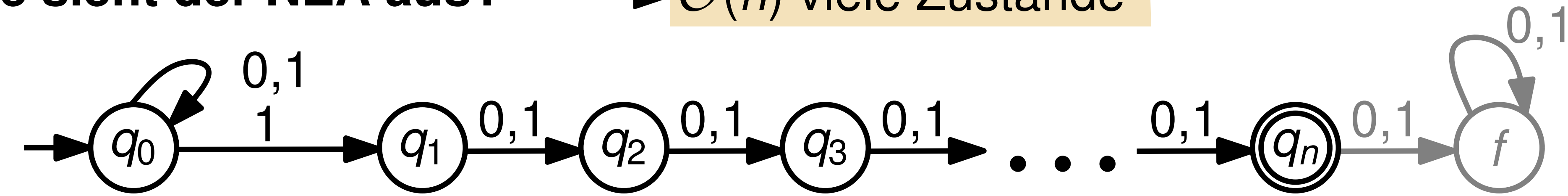


Zeige: Jeder DEA von L_n hat mindestens 2^n Zustände.

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid \text{ das } n\text{-letzte Zeichen von } w \text{ ist eine } 1\}$.

Wie sieht der NEA aus? → $\mathcal{O}(n)$ viele Zustände



Zeige: Jeder DEA von L_n hat mindestens 2^n Zustände.

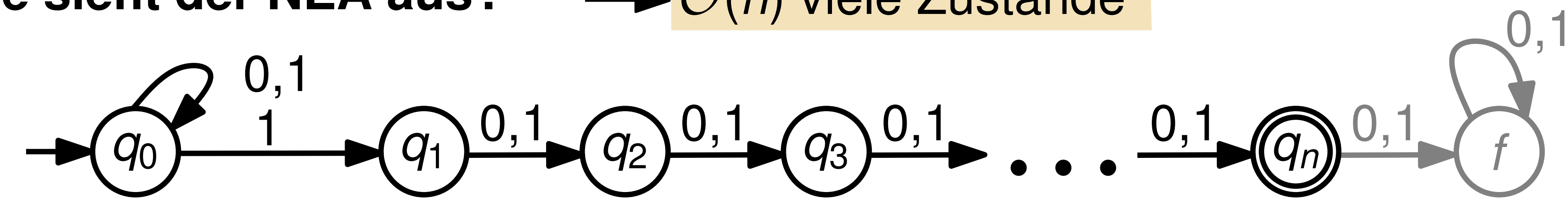
Annahme: Sei $\mathcal{D}_n = (\Sigma, Q, \delta, q_0, F)$ DEA, der L_n akzeptiert, mit $|Q| < 2^n$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid$ das n-letzte Zeichen von w ist eine $1\}$.

Wie sieht der NEA aus? $\longrightarrow$ $\mathcal{O}(n)$ viele Zustände



Zeige: Jeder DEA von L_n hat mindestens 2^n Zustände.

Annahme: Sei $\mathcal{D}_n = (\Sigma, Q, \delta, q_0, F)$ DEA, der L_n akzeptiert, mit $|Q| < 2^n$

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

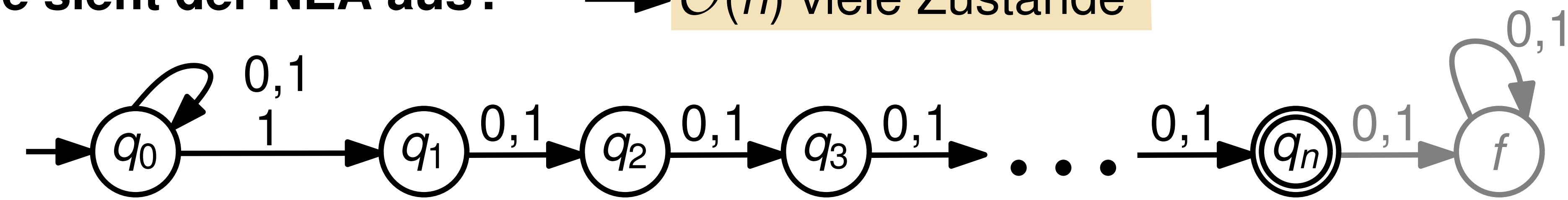
$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

Noch zu zeigen.

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid$ das n-letzte Zeichen von w ist eine $1\}$.

Wie sieht der NEA aus? $\longrightarrow$ $\mathcal{O}(n)$ viele Zustände



Zeige: Jeder DEA von L_n hat mindestens 2^n Zustände.

Annahme: Sei $\mathcal{D}_n = (\Sigma, Q, \delta, q_0, F)$ DEA, der L_n akzeptiert, mit $|Q| < 2^n$

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

Noch zu zeigen.

Dann gilt: Seien $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$ wie im Lemma gewählt.

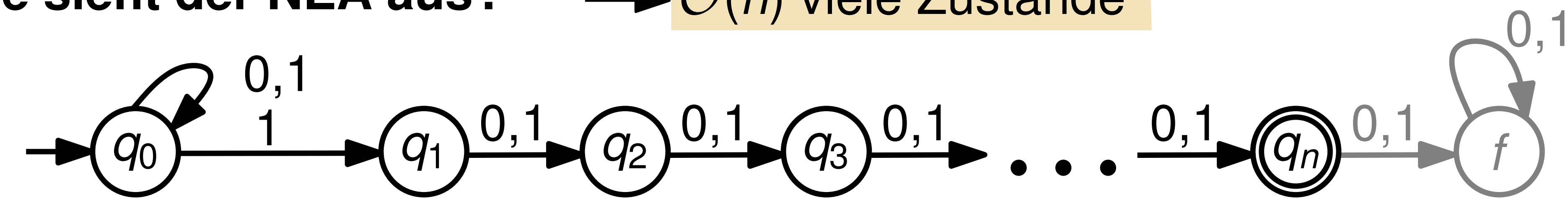
$\longrightarrow$ $x\,1\,u \in L_n$ und $y\,0\,v \notin L_n$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid$ das n-letzte Zeichen von w ist eine $1\}$.

Wie sieht der NEA aus? ⟶ $\mathcal{O}(n)$ viele Zustände



Zeige: Jeder DEA von L_n hat mindestens 2^n Zustände.

Annahme: Sei $\mathcal{D}_n = (\Sigma, Q, \delta, q_0, F)$ DEA, der L_n akzeptiert, mit $|Q| < 2^n$

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

Noch zu zeigen.

Dann gilt: Seien $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$ wie im Lemma gewählt.

⟶ $x\,1\,u \in L_n$ und $y\,0\,v \notin L_n$

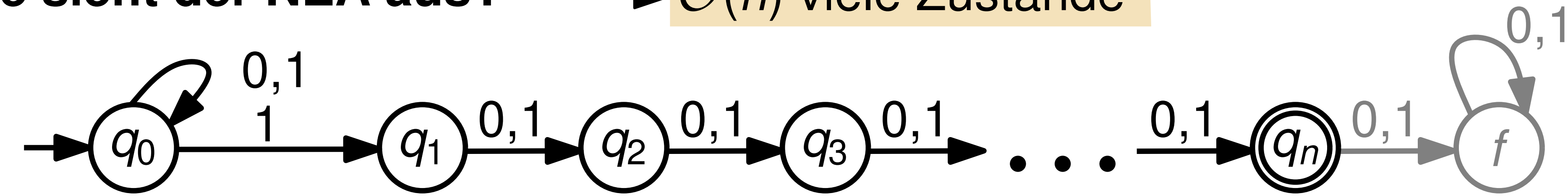
⟶ $\delta(q_0, x\,1\,u) \in F$ und $\delta(q_0, y\,0\,v) \notin F$ ↯ Lemma

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Größe von DEAs vs. NEAs

Betrachte: $L_n = \{w \in \{0,1\}^\star \mid$ das n-letzte Zeichen von w ist eine 1$\}$.

Wie sieht der NEA aus? $\longrightarrow \mathcal{O}(n)$ viele Zustände



Zeige: Jeder DEA von L_n hat mindestens 2^n Zustände.

Annahme: Sei $\mathcal{D}_n = (\Sigma, Q, \delta, q_0, F)$ DEA, der L_n akzeptiert, mit $|Q| < 2^n$

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

Noch zu zeigen.

Dann gilt: Seien $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$ wie im Lemma gewählt.

$\longrightarrow$ $x\,1\,u \in L_n$ und $y\,0\,v \notin L_n$

$\longrightarrow$ $\delta(q_0, x\,1\,u) \in F$ und $\delta(q_0, y\,0\,v) \notin F$

Lemma

$\longrightarrow \Omega(2^n)$ viele Zustände

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

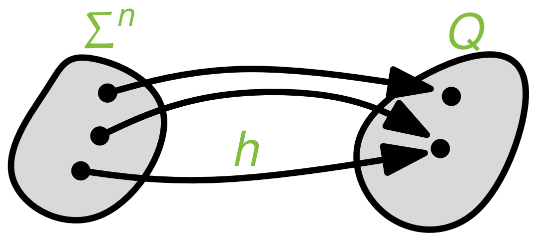
Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$

$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

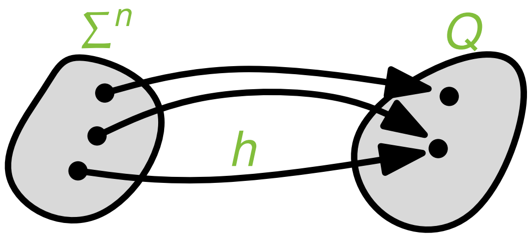


Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\ 1\ u) = \delta(q_0, y\ 0\ v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$



$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

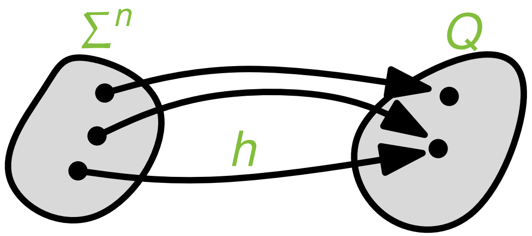
$\exists$ Sequenzen $\sigma_1 = a_1 \dots a_n$ und $\sigma_2 = b_1 \dots b_n$ mit $\sigma_1 \neq \sigma_2$ und $h(\sigma_1) = h(\sigma_2)$

Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\,1\,u) = \delta(q_0, y\,0\,v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$



$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

$\exists$ Sequenzen $\sigma_1 = a_1 \dots a_n$ und $\sigma_2 = b_1 \dots b_n$ mit $\sigma_1 \neq \sigma_2$ und $h(\sigma_1) = h(\sigma_2)$

Annahme: $a_i = 1$ und $b_i = 0$

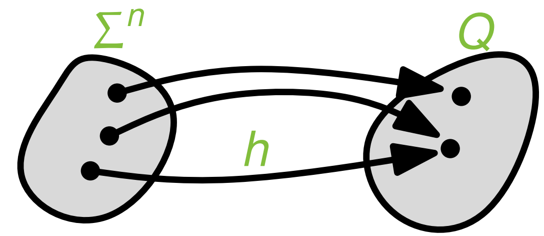
Sei $x = a_1 \dots a_{i-1}$ $\quad u = a_{i+1} \dots a_n 0^{i-1}$ $\quad y = b_1 \dots b_{i-1}$ $\quad v = b_{i+1} \dots b_n 0^{i-1}$

Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\ 1\ u) = \delta(q_0, y\ 0\ v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$



$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

$\exists$ Sequenzen $\sigma_1 = a_1 \ldots a_n$ und $\sigma_2 = b_1 \ldots b_n$ mit $\sigma_1 \neq \sigma_2$ und $h(\sigma_1) = h(\sigma_2)$

Annahme: $a_i = 1$ und $b_i = 0$

Sei $x = a_1 \ldots a_{i-1}$ $\quad u = a_{i+1} \ldots a_n 0^{i-1}$ $\quad y = b_1 \ldots b_{i-1}$ $\quad v = b_{i+1} \ldots b_n 0^{i-1}$

$$\delta(q_0, x\ 1\ u) = \delta(q_0, a_1 \ldots a_{i-1}\ 1\ a_{i+1} \ldots a_n 0^{i-1})$$

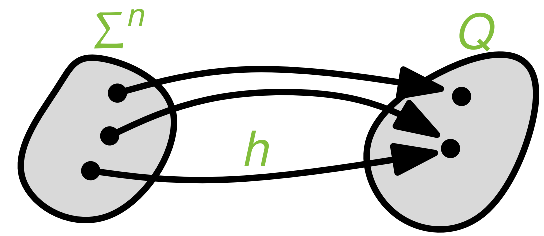
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\ 1\ u) = \delta(q_0, y\ 0\ v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$



$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

$\exists$ Sequenzen $\sigma_1 = a_1 \dots a_n$ und $\sigma_2 = b_1 \dots b_n$ mit $\sigma_1 \neq \sigma_2$ und $h(\sigma_1) = h(\sigma_2)$

Annahme: $a_i = 1$ und $b_i = 0$

Sei $x = a_1 \dots a_{i-1}$ $\quad u = a_{i+1} \dots a_n 0^{i-1}$ $\quad y = b_1 \dots b_{i-1}$ $\quad v = b_{i+1} \dots b_n 0^{i-1}$

$$\delta(q_0, x\ 1\ u) = \delta(q_0, a_1 \dots a_{i-1}\ 1\ a_{i+1} \dots a_n 0^{i-1})$$
$$= \delta(\delta(q_0, a_1 \dots a_{i-1}\ 1\ a_{i+1} \dots a_n), 0^{i-1})$$

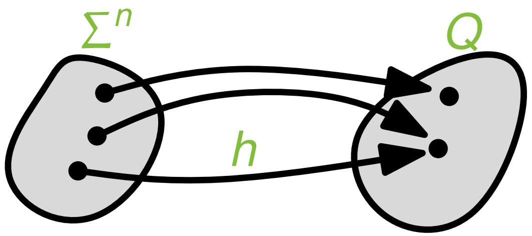
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\ 1\ u) = \delta(q_0, y\ 0\ v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$



$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

$\exists$ Sequenzen $\sigma_1 = a_1 \ldots a_n$ und $\sigma_2 = b_1 \ldots b_n$ mit $\sigma_1 \neq \sigma_2$ und $h(\sigma_1) = h(\sigma_2)$

Annahme: $a_i = 1$ und $b_i = 0$

Sei $x = a_1 \ldots a_{i-1}$ $\quad u = a_{i+1} \ldots a_n 0^{i-1}$ $\quad y = b_1 \ldots b_{i-1}$ $\quad v = b_{i+1} \ldots b_n 0^{i-1}$

$$\begin{aligned}
\delta(q_0, x\ 1\ u) &= \delta(q_0, a_1 \ldots a_{i-1}\ 1\ a_{i+1} \ldots a_n 0^{i-1}) \\
&= \delta(\delta(q_0, a_1 \ldots a_{i-1}\ 1\ a_{i+1} \ldots a_n), 0^{i-1}) \\
&= \delta(\delta(q_0, b_1 \ldots b_{i-1}\ 0\ b_{i+1} \ldots b_n), 0^{i-1})
\end{aligned}$$

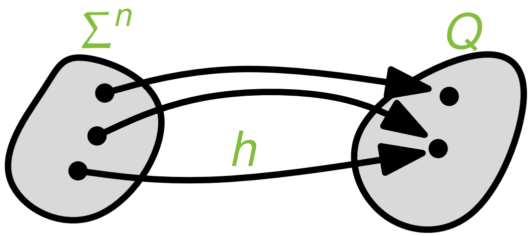
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Beweis zu Lemma

Lemma: Wenn $Q < 2^n$, dann gibt es $x, y \in \Sigma^\star$ und $u, v \in \Sigma^{n-1}$, sodass

$$\delta(q_0, x\ 1\ u) = \delta(q_0, y\ 0\ v)$$

Beweis: Sei $h\colon \Sigma^n \to Q$ sodass $h(z) = \delta(q_0, z)$



$\longrightarrow$ h ist nicht injektiv, da $|Q| < 2^n$

$\exists$ Sequenzen $\sigma_1 = a_1 \ldots a_n$ und $\sigma_2 = b_1 \ldots b_n$ mit $\sigma_1 \neq \sigma_2$ und $h(\sigma_1) = h(\sigma_2)$

Annahme: $a_i = 1$ und $b_i = 0$

Sei $x = a_1 \ldots a_{i-1}$ $\quad u = a_{i+1} \ldots a_n 0^{i-1}$ $\quad y = b_1 \ldots b_{i-1}$ $\quad v = b_{i+1} \ldots b_n 0^{i-1}$

$$\begin{aligned}
\delta(q_0, x\ 1\ u) &= \delta(q_0, a_1 \ldots a_{i-1}\ 1\ a_{i+1} \ldots a_n 0^{i-1}) \\
&= \delta(\delta(q_0, a_1 \ldots a_{i-1}\ 1\ a_{i+1} \ldots a_n), 0^{i-1}) \\
&= \delta(\delta(q_0, b_1 \ldots b_{i-1}\ 0\ b_{i+1} \ldots b_n), 0^{i-1}) \\
&= \delta(q_0, b_1 \ldots b_{i-1}\ 0\ b_{i+1} \ldots b_n 0^{i-1}) \quad = \delta(q_0, y\ 0\ v)
\end{aligned}$$

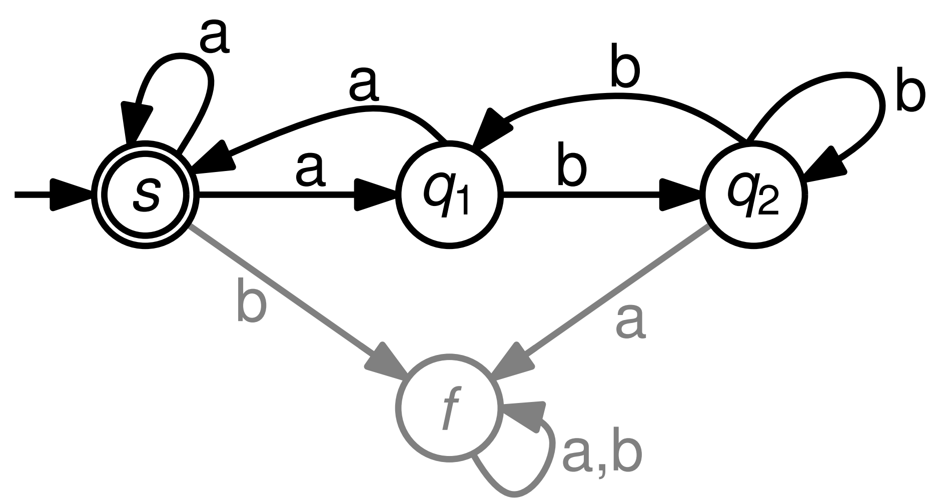
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	b

$L = (a \cup (ab(b)^\star ba))^\star$



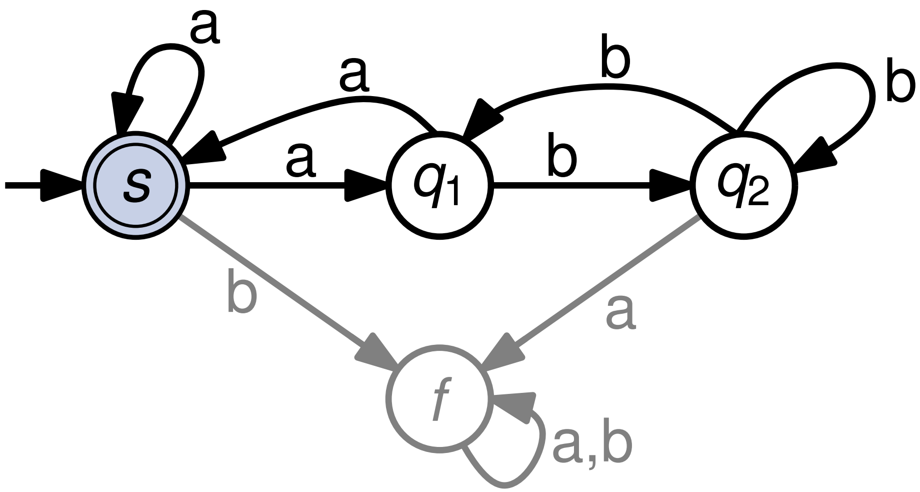
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$		

$L = (a \cup (ab(b)^\star ba))^\star$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

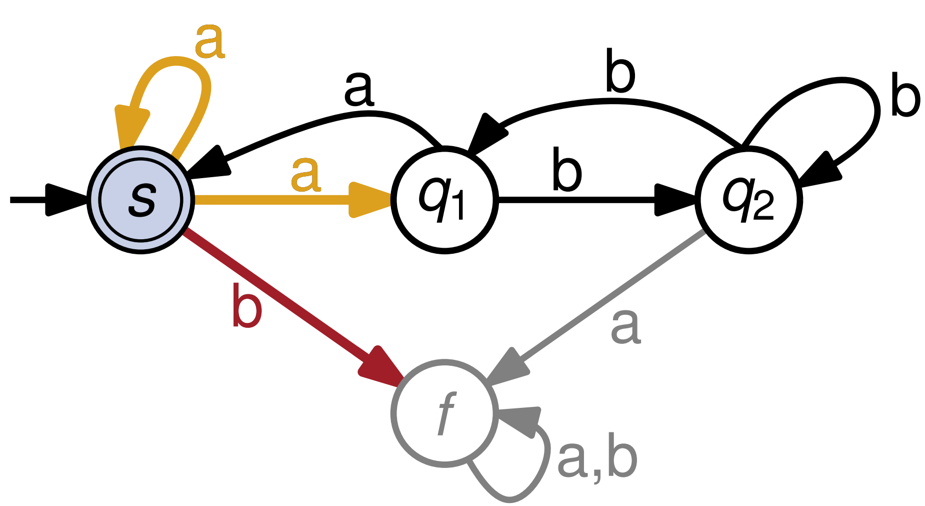
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$		
$\{f\}$		

$$L = (a \cup (ab(b)^\star ba))^\star$$



Potenzmengenkonstruktion

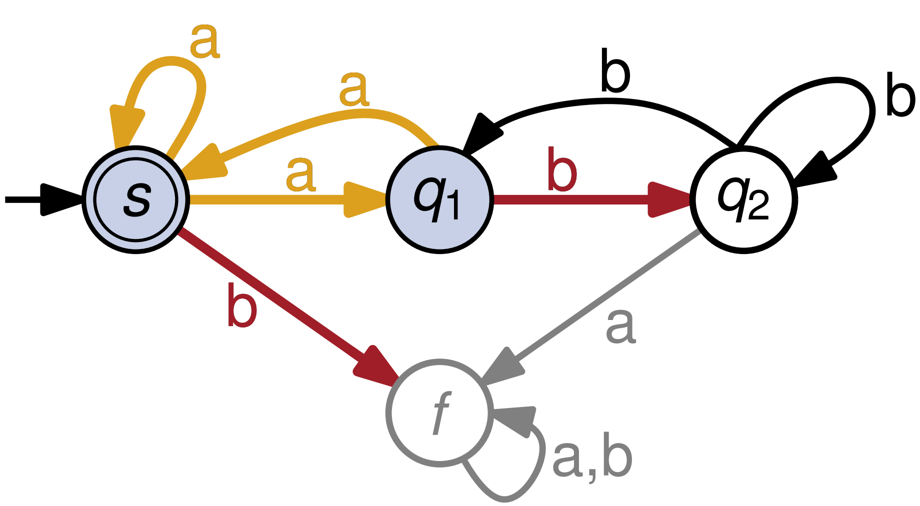
Karlsruher Institut für Technologie

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$		
$\{f, q_2\}$		

$$L = (a \cup (ab(b)^\star ba))^\star$$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

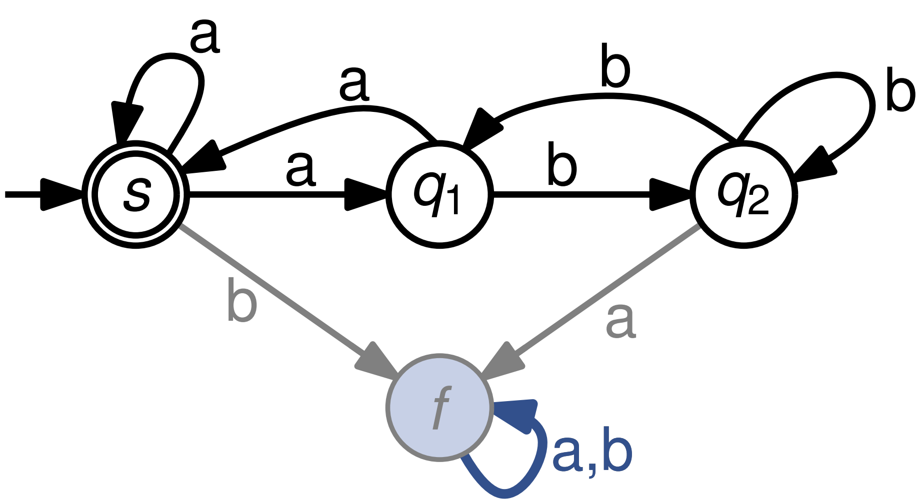
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$		

$$L = (a \cup (ab(b)^\star ba))^\star$$



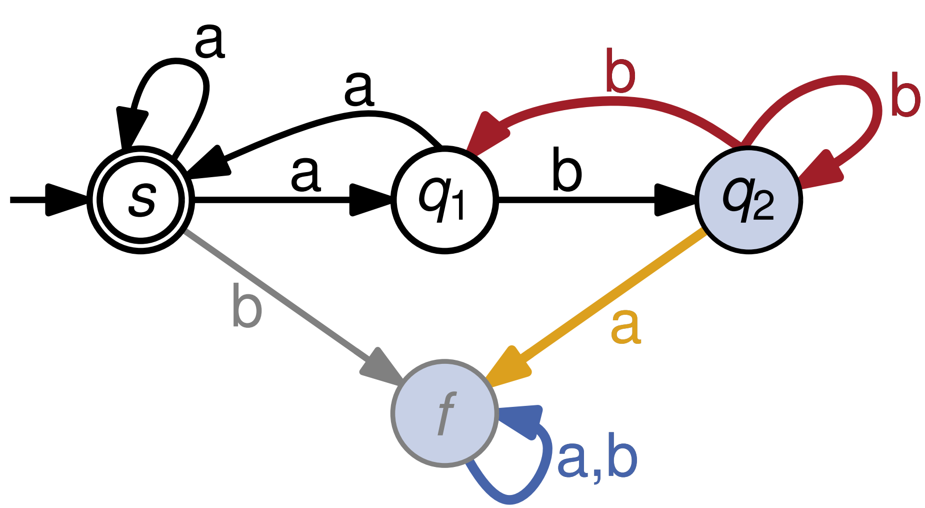
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

$$L = (a \cup (ab(b)^\star ba))^\star$$

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$		



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

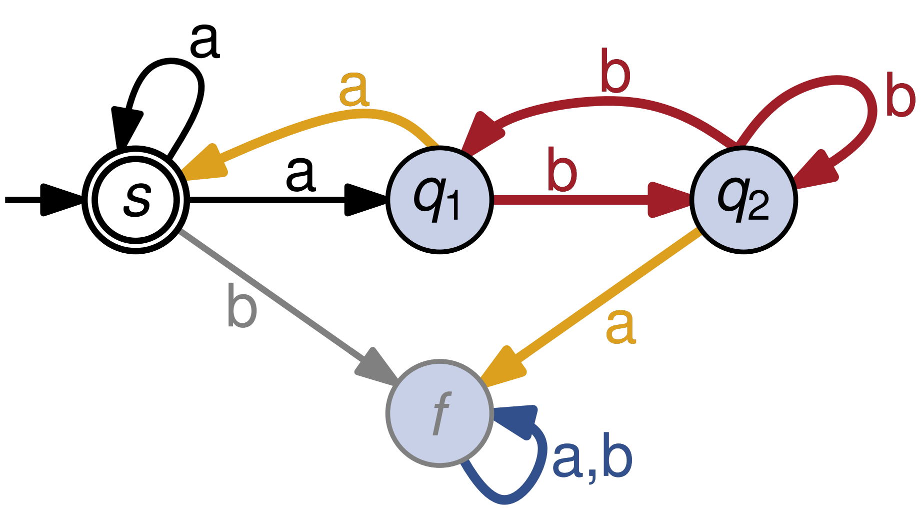
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, s\}$		

$L = (a \cup (ab(b)^\star ba))^\star$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

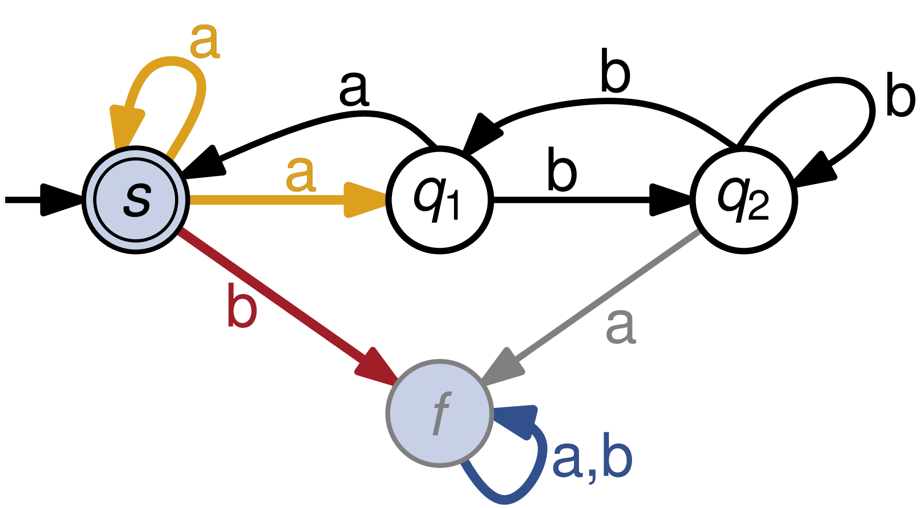
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, s\}$	$\{f, s, q_1\}$	$\{f\}$
$\{f, s, q_1\}$		

$L = (a \cup (ab(b)^\star ba))^\star$



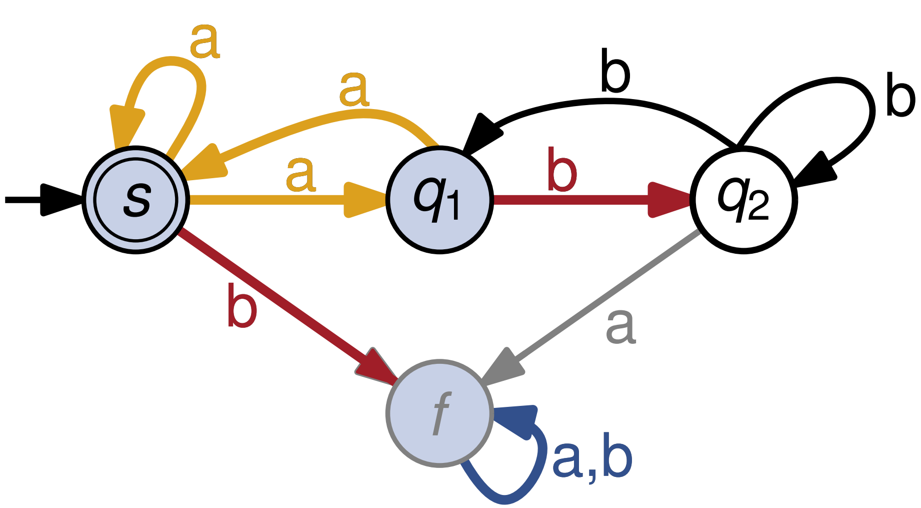
Potenzmengenkonstruktion

Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, s\}$	$\{f, s, q_1\}$	$\{f\}$
$\{f, s, q_1\}$	$\{f, s, q_1\}$	$\{f, q_2\}$

$L = (a \cup (ab(b)^\star ba))^\star$



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Potenzmengenkonstruktion

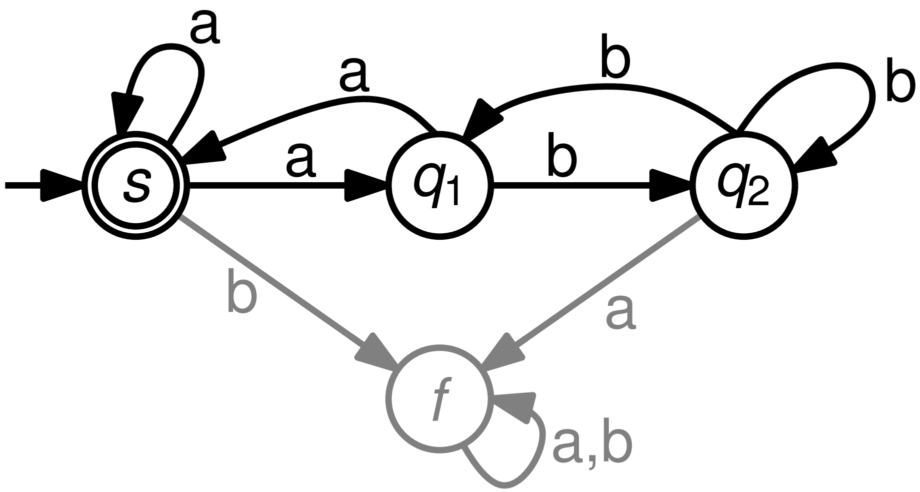
Jeder NEA kann in einen DEA überführt werden.

Potenzmengenkonstruktion: systematisches Verfahren für Transformation

$L = (a \cup (ab(b)^\star ba))^\star$

Zustand	Übergang a	Übergang b
$\{\underline{s}\}$	$\{s, q_1\}$	$\{f\}$
$\{\underline{s}, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, \underline{s}\}$	$\{f, s, q_1\}$	$\{f\}$
$\{f, \underline{s}, q_1\}$	$\{f, s, q_1\}$	$\{f, q_2\}$

Endzustände: Alle Zustände, die Endzustand aus NEA enthalten.



Potenzmengenkonstruktion

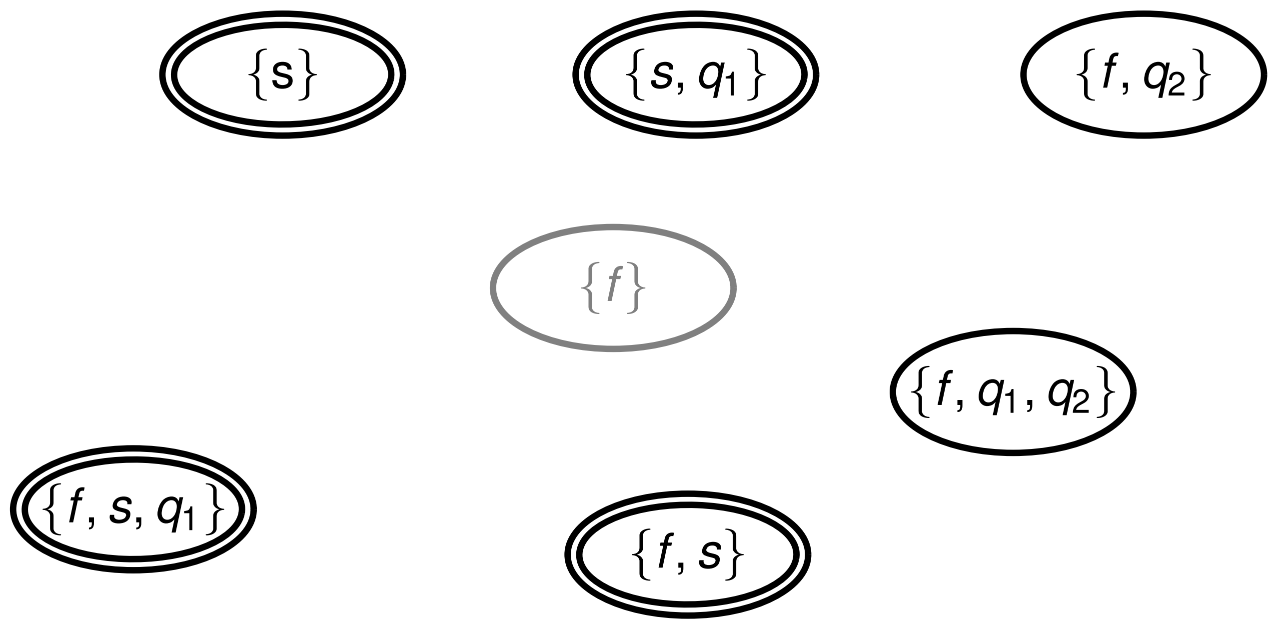
Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, s\}$	$\{f, s, q_1\}$	$\{f\}$
$\{f, s, q_1\}$	$\{f, s, q_1\}$	$\{f, q_2\}$

$$L = (a \cup (ab(b)^\star ba))^\star$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Potenzmengenkonstruktion

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, s\}$	$\{f, s, q_1\}$	$\{f\}$
$\{f, s, q_1\}$	$\{f, s, q_1\}$	$\{f, q_2\}$

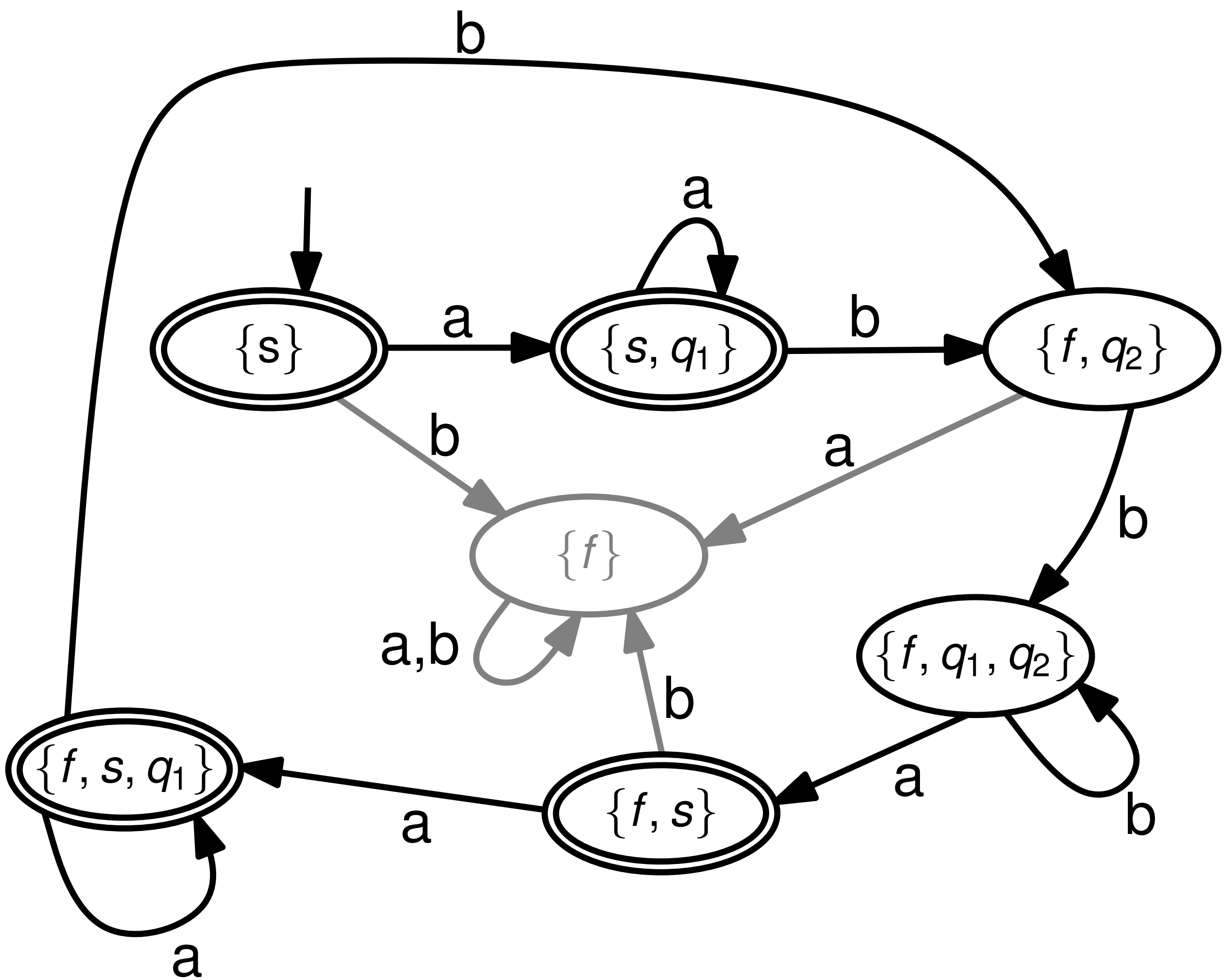


$$L = (a \cup (ab(b)^\star ba))^\star$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Potenzmengenkonstruktion

Zustand	Übergang a	Übergang b
$\{s\}$	$\{s, q_1\}$	$\{f\}$
$\{s, q_1\}$	$\{s, q_1\}$	$\{f, q_2\}$
$\{f\}$	$\{f\}$	$\{f\}$
$\{f, q_2\}$	$\{f\}$	$\{f, q_1, q_2\}$
$\{f, q_1, q_2\}$	$\{f, s\}$	$\{f, q_1, q_2\}$
$\{f, s\}$	$\{f, s, q_1\}$	$\{f\}$
$\{f, s, q_1\}$	$\{f, s, q_1\}$	$\{f, q_2\}$



$$L = (a \cup (ab(b)^\star ba))^\star$$

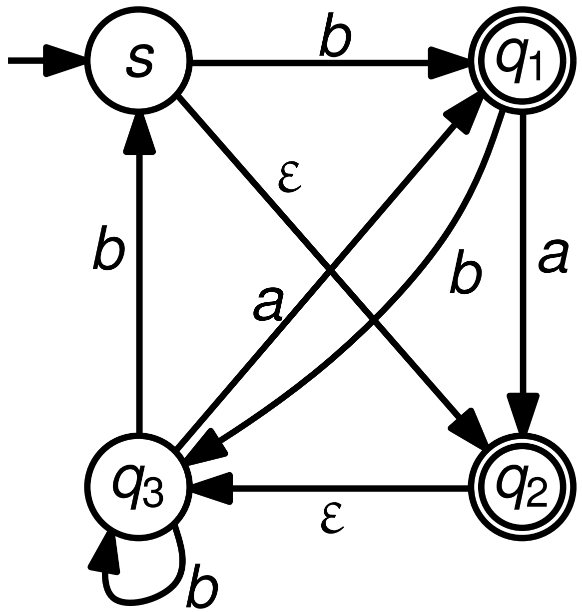
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s		
q_1		
q_2		
q_3		



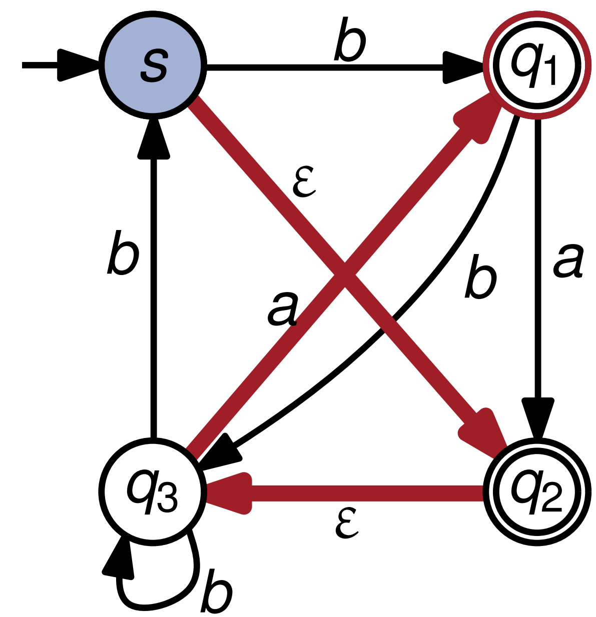
Fehlende Übergänge führen in Fehlerzustand.

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	
q_1		
q_2		
q_3		



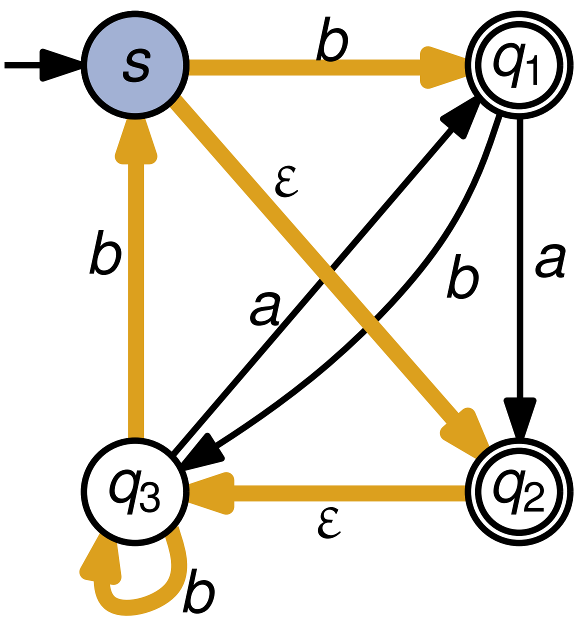
Fehlende Übergänge führen in Fehlerzustand.

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1		
q_2		
q_3		



Fehlende Übergänge führen in Fehlerzustand.

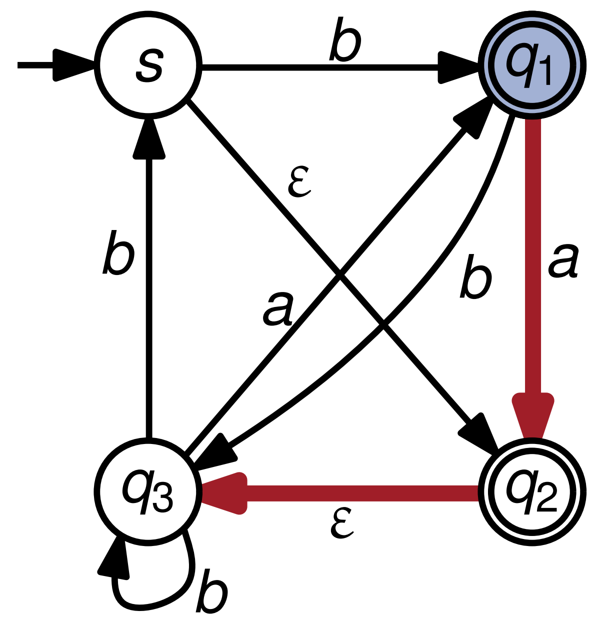
Entfernen von ε-Übergängen

Karlsruher Institut für Technologie

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	
q_2		
q_3		



Fehlende Übergänge führen in Fehlerzustand.

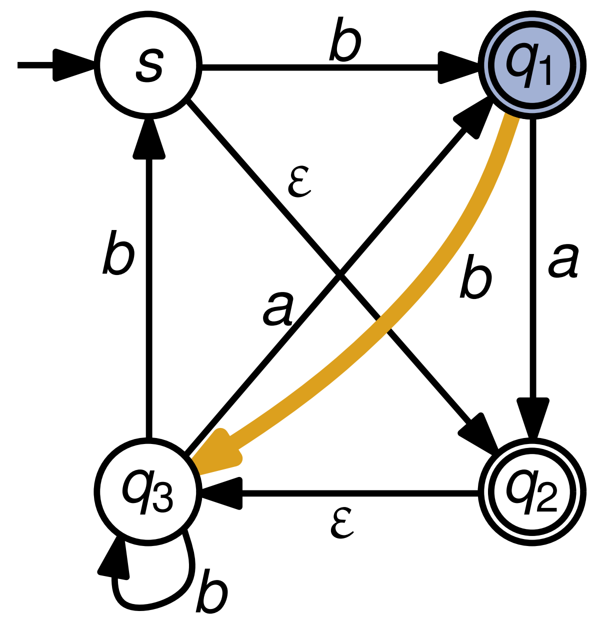
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	$, q_3$
q_2		
q_3		



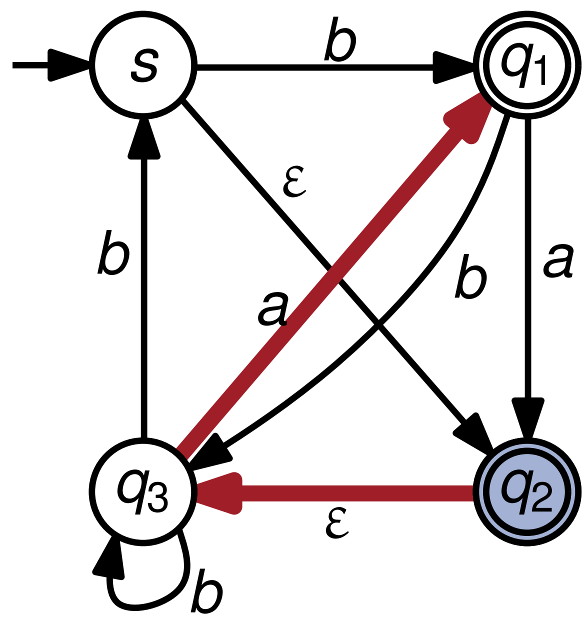
Fehlende Übergänge führen in Fehlerzustand.

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	$, q_3$
q_2	q_1	
q_3		



Fehlende Übergänge führen in Fehlerzustand.

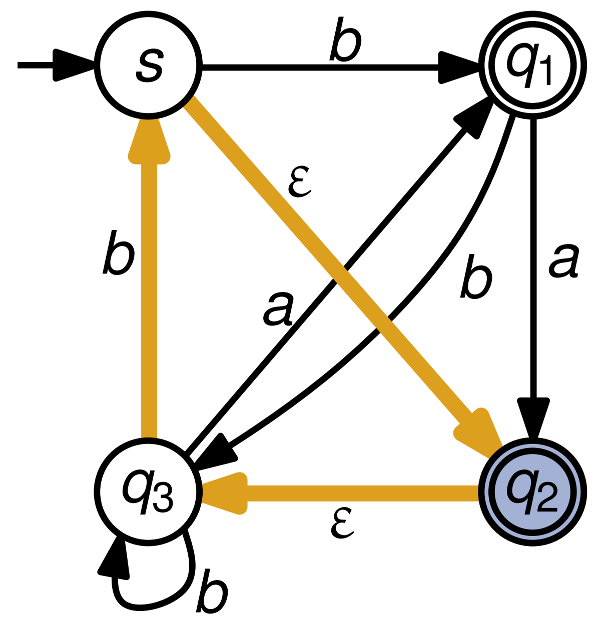
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	$, q_3$
q_2	q_1	$s \quad , q_2, q_3$
q_3		



Fehlende Übergänge führen in Fehlerzustand.

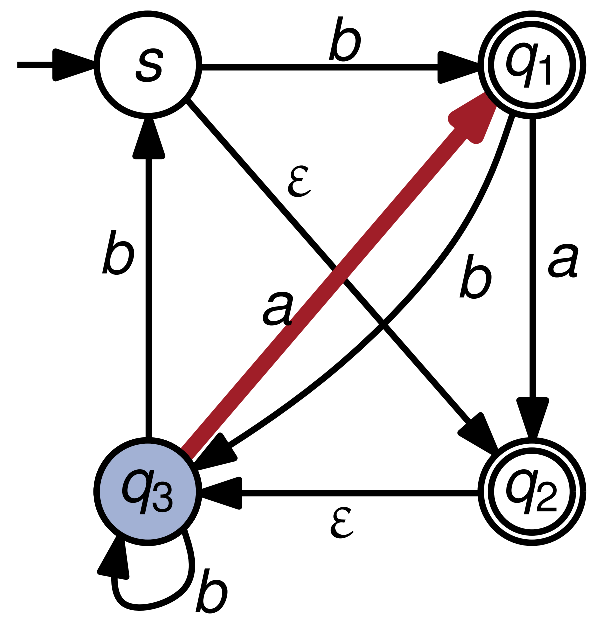
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	$, q_3$
q_2	q_1	$s \quad , q_2, q_3$
q_3	q_1	



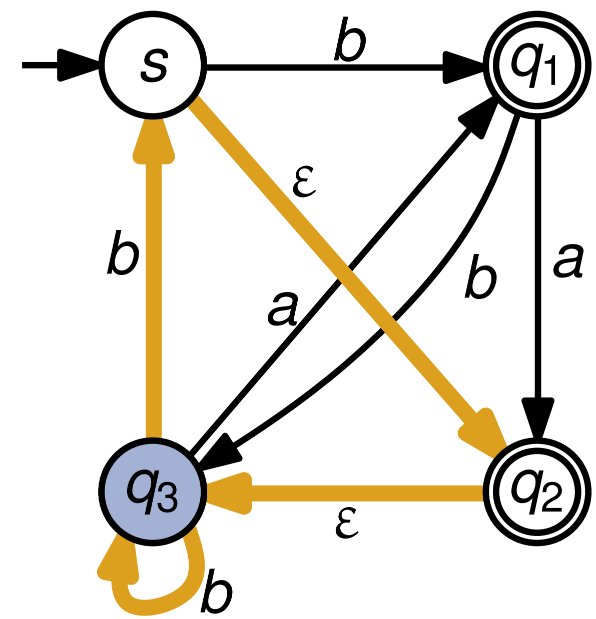
Fehlende Übergänge führen in Fehlerzustand.

Entfernen von ε-Übergängen

Jeder NEA kann in einen äquivalenten NEA ohne ε-Übergänge überführt werden, ohne die Anzahl Zustände zu erhöhen.

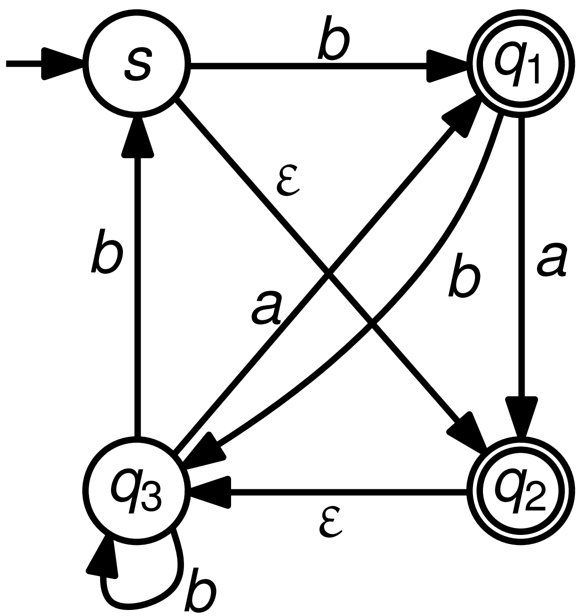
Idee: Finde via "$\varepsilon^\star x \varepsilon^\star$" erreichbare Zustände

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	$, q_3$
q_2	q_1	$s \quad , q_2, q_3$
q_3	q_1	$s \quad , q_2, q_3$



Fehlende Übergänge führen in Fehlerzustand.

Entfernen von ε-Übergängen

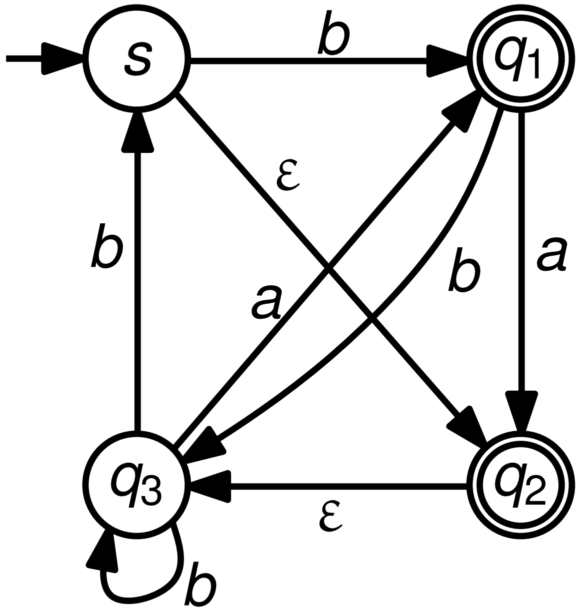


ε-Übergang führt von s zu Finalzustand

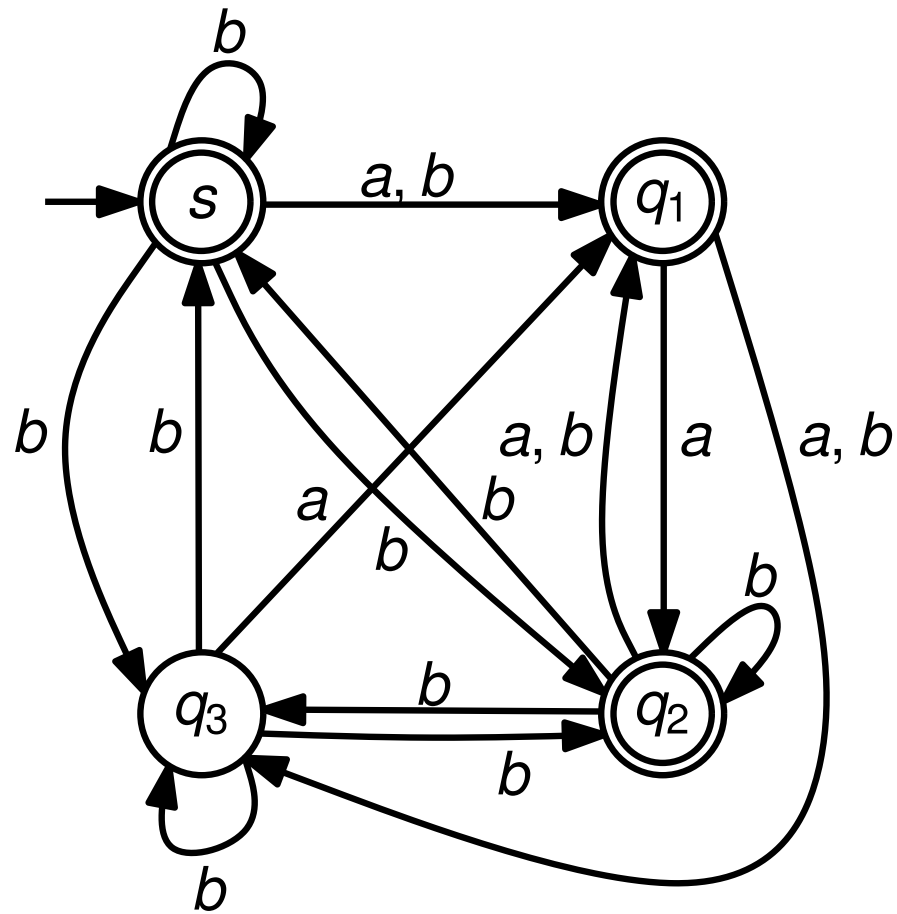
$\rightarrow$ s wird zu Finalzustand

	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	q_3
q_2	q_1	s, q_2, q_3
q_3	q_1	s, q_2, q_3

Entfernen von ε-Übergängen



ε-Übergang führt von s zu Finalzustand
$\rightarrow$ s wird zu Finalzustand



	a	b
s	q_1	s, q_1, q_2, q_3
q_1	q_2, q_3	q_3
q_2	q_1	s, q_2, q_3
q_3	q_1	s, q_2, q_3

Kontextfreie Grammatiken

Kontextfreie Grammatik: $G = (\Sigma, V, S, R)$

Alphabet: Σ (Terminalalphabet), endlich

$\Sigma = \{a, b\}$

Variablen: V (Nichtterminale), $V \cap \Sigma = \emptyset$, endlich

$V = \{A, S\}$

Startsymbol: $S \in V$

Ableitungsregeln: $R \subset V \times (\Sigma \cup V)^\star$

$S \to aSb \mid A, \quad A \to aA \mid a$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreie Grammatiken

$$
\begin{aligned}
G &= (\Sigma = \{a, b\}, V = \{S\}, S, R) \\
R &= S \rightarrow aSb \mid ab \\
L(G) &= \{a^i b^i \mid i \in \mathbb{N}\}
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreie Grammatiken

$$G = (\Sigma = \{a, b\}, V = \{S\}, S, R)$$
$$R = S \to aSb \mid ab$$
$$L(G) = \{a^i b^i \mid i \in \mathbb{N}\}$$

Ist $L(G)$ regulär?

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreie Grammatiken

Karlsruher Institut für Technologie

$$
\begin{aligned}
G &= (\Sigma = \{a, b\}, V = \{S\}, S, R) \\
R &= S \to aSb \mid ab \\
L(G) &= \{a^i b^i \mid i \in \mathbb{N}\}
\end{aligned}
$$

Ist $L(G)$ regulär?

Nein!

Kontextfreie Grammatiken

Karlsruher Institut für Technologie

$$
\begin{aligned}
G &= (\Sigma = \{a, b\}, V = \{S\}, S, R) \\
R &= S \to aSb \mid ab \\
L(G) &= \{a^i b^i \mid i \in \mathbb{N}\}
\end{aligned}
$$

Ist $L(G)$ regulär? Nein!

Problem (informell)**:** EA können nicht zählen (endliches Gedächtnis)

Kontextfreie Grammatiken

$G = (\Sigma = \{a, b\}, V = \{S\}, S, R)$

$R = S \rightarrow aSb \mid ab$

$L(G) = \{a^i b^i \mid i \in \mathbb{N}\}$

Ist $L(G)$ regulär? Nein!

Problem (informell)**:** EA können nicht zählen (endliches Gedächtnis)

Formales Hilfsmittel:

Satz (Pumping-Lemma).

Zu jeder regulären Sprache L existiert eine Zahl $n \in \mathbb{N}$, so dass jedes Wort $w \in L$ mit $|w| > n$ zerlegt werden kann in $w = uvx$ mit $|uv| \leq n$, $v \neq \varepsilon$, so dass auch $uv^ix \in L$ für alle $i = 0, 1, \ldots$.

$ux \in L \quad uvx \in L \quad uvvx \in L \quad uvvvx \in L \quad \cdots$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik