

# Automaten und Sprachen

§ 2–6 Endliche Automaten und reguläre Sprachen:  
wichtige Resultate und Techniken



## Teil I: Endliche Automaten und reguläre Sprachen

- 0. Grundbegriffe
- 1. Endliche Automaten
- 2. Nachweis der Nichterkennbarkeit
- 3. Abschlusseigenschaften
- 4. Entscheidungsprobleme
- 5. Reguläre Ausdrücke und Sprachen
- 6. Minimale DEAs und die Nerode-Rechtskongruenz



## Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

- 7. Die Chomsky-Hierarchie
- 8. Rechtslineare Grammatiken und reguläre Sprachen
- 9. Normalformen und Entscheidungsprobleme
- 10. Abschlusseigenschaften und Pumping-Lemma
- 11. Kellerautomaten
- 12. Die Struktur kontextfreier Sprachen



## §2 Nachweis der Nichterkennbarkeit



# Nachweis der Nichterkennbarkeit

## Nachweis der Erkennbarkeit:

Konstruiere einen endlichen Automaten (DEA, NEA) für die Sprache.

## Nachweis der Nichterkennbarkeit:

Beweise, dass es **keinen** DEA/NEA für die Sprache geben kann.

☞ Viel schwieriger!

## Generelle Idee:

Verwende, dass DEAs/NEAs nur **endlich viele** Zustände haben dürfen.

Finde darauf basierende Eigenschaft, die **jede** erkennbare Sprache erfüllt.  
Weise dann nach, dass die betreffende Sprache die Eigenschaft **verletzt**.

Dies wird formalisiert durch das **Pumping Lemma**.



# Ein Beispiel

## Beispiel 2.1

$L = \{a^n b^n \mid n \geq 0\}$  ist nicht erkennbar.

### Beweis:

Angenommen, es gibt doch NEA  $\mathcal{A}$  mit  $L(\mathcal{A}) = L$ .

Dann hat  $\mathcal{A}$  endlich viele Zustände, sagen wir  $n_0$  Stück.

Wir betrachten Wort  $w = a^{n_0} b^{n_0} \in L$ :

aaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbb  
 $q_0 \quad x \quad q \quad y \quad q \quad z \quad q_f \in F$

Es gibt also Pfad  $q_0 \xrightarrow{x} \mathcal{A} q \xrightarrow{y} \mathcal{A} q \xrightarrow{z} \mathcal{A} q_f$

und damit auch  $q_0 \xrightarrow{x} \mathcal{A} q \xrightarrow{y} \mathcal{A} q \xrightarrow{y} \mathcal{A} q \xrightarrow{z} \mathcal{A} q_f$

Also enthält  $L(\mathcal{A})$  das Wort  $w' = xyzy$ ; offensichtlich gilt  $|w'|_a > |w'|_b$ . 

**Widerspruch!**

Also war die Annahme falsch, dass  $L$  erkennbar ist!



# Das Pumping-Lemma

Lemma 2.2 (Pumping-Lemma) [Bar-Hillel, Perles, Shamir]

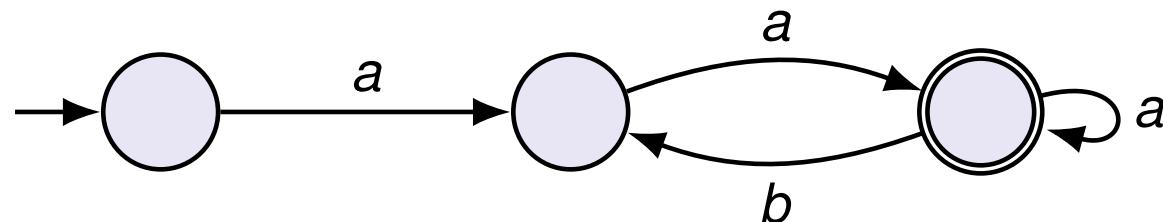
Sei  $L$  eine erkennbare Sprache.

Dann gibt es eine natürliche Zahl  $n_0 \geq 1$ , so dass gilt:

Jedes Wort  $w \in L$  mit  $|w| \geq n_0$  lässt sich zerlegen in  $w = xyz$  mit

- $y \neq \varepsilon$  und  $|xy| \leq n_0$
- $xy^kz \in L$  für alle  $k \geq 0$

## Beispiel



Wähle  $n_0 = 3$  (Zahl der Zustände!)

Als Beispiel für Wort in  $L$  mit Länge  $\geq n_0$  betrachte  $\underline{a} \underline{a} \underline{b} \underline{a} \underline{a}$

Dann:  $xy^0z = aaa$     $xy^1z = a\textcolor{red}{ab}aa$     $xy^2z = a\textcolor{red}{abab}aa$    etc.



# Zurück zum Beispiel

## Beispiel 2.3

$L = \{a^n b^n \mid n \geq 0\}$  ist nicht erkennbar.

### Beweis mittels Pumping-Lemma:

Wir führen einen Widerspruchsbeweis und nehmen an,  $L$  sei erkennbar.

Es gibt also Zahl  $n_0$ , so dass sich jedes Wort  $w \in L$  mit  $|w| \geq n_0$  wie in Lemma 2.2 zerlegen lässt.

Betrachte das Wort  $w = a^{n_0} b^{n_0} \in L$ .

Da  $|w| \geq n_0$ , muss es eine Zerlegung geben

$$a^{n_0} b^{n_0} = xyz \quad \text{mit } y \neq \varepsilon, \quad |xy| \leq n_0 \quad \text{und} \quad xy^k z \in L \quad \text{für alle } k \geq 0.$$

Also muss  $y$  vollständig in den  $a$ 's liegen:

aaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbb  
y

Dann gilt für  $k = 2$ , dass  $xy^k z \notin L$  (Bsp. 2.1)



Widerspruch!



# Anwendung des Pumping-Lemmas

## Lemma 2.2 (Pumping-Lemma)

Sei  $L$  eine erkennbare Sprache.

Dann gibt es eine natürliche Zahl  $n_0 \geq 1$ , so dass gilt:

Jedes Wort  $w \in L$  mit  $|w| \geq n_0$  lässt sich zerlegen in  $w = xyz$  mit

- $y \neq \varepsilon$  und  $|xy| \leq n_0$
- $xy^kz \in L$  für alle  $k \geq 0$

## Lemma 2.4 (Pumping-Lemma als Kontraposition)

Angenommen, für eine Sprache  $L$  gilt das Folgende:

für alle natürlichen Zahlen  $n_0 \geq 1$

gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ , so dass

für alle Zerlegungen  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$  gilt:

es gibt  $k \geq 0$  mit  $xy^kz \notin L$ .

Dann ist  $L$  nicht erkennbar.



# Anwendung des Pumping-Lemmas

Lemma 2.4 (Pumping-Lemma als Kontraposition)

Angenommen, für eine Sprache  $L$  gilt das Folgende:

für alle natürlichen Zahlen  $n_0 \geq 1$

gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ , so dass

für alle Zerlegungen  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$  gilt:

es gibt  $k \geq 0$  mit  $xy^k z \notin L$ .

Dann ist  $L$  nicht erkennbar.



pixabay.com (Public domain)

Diese Formulierung legt **spieltheoretische Sicht** nahe:

- Man spielt in Runden gegen einen Gegner.  
Gegner will zeigen: Sprache ist erkennbar; wir wollen Gegenteil zeigen.
- bei „für alle“ ist Gegner dran; bei „es gibt“ wir.
- Wenn wir das Spiel stets gewinnen können, egal was der Gegner tut,  
dann ist die Eigenschaft aus Lemma 2.4 erfüllt, also  $L$  nicht erkennbar.

# Ein weiteres Beispiel

## Beispiel 2.5

$L = \{a^n \mid n \text{ ist Quadratzahl}\}$  ist nicht erkennbar.

### Beweis:

- „für alle natürlichen Zahlen  $n_0 \geq 1$ “: Gegner wählt  $n_0$
- „gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ “: Wir wählen  $w = a^m$  mit  $m = (n_0 + 1)^2$
- „so dass für alle Zerleg.  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$ “: Gegner wählt  $xyz$
- „gilt: es gibt  $k \geq 0$  mit  $xy^k z \notin L$ “:  $z \neq \varepsilon$

Offensichtlich ist  $|w| > n_0$ .

Wegen  $|xy| \leq n_0$  muss also  $z \neq \varepsilon$  sein.

**Idee:** pumpen  $y$  so, dass Wortlänge  $s^2 + t$  für geeignete  $s, t$  mit  $0 < t < s$

So eine Zahl kann niemals Quadratzahl sein:

$$s^2 < s^2 + t < s^2 + 2s + 1 = (s + 1)^2$$



# Ein weiteres Beispiel

## Beispiel 2.5

$L = \{a^n \mid n \text{ ist Quadratzahl}\}$  ist nicht erkennbar.

### Beweis:

- „für alle natürlichen Zahlen  $n_0 \geq 1$ “: Gegner wählt  $n_0$
- „gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ “: Wir wählen  $w = a^m$  mit  $m = (n_0 + 1)^2$
- „so dass für alle Zerleg.  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$ “: Gegner wählt  $xyz$
- „gilt: es gibt  $k \geq 0$  mit  $xy^k z \notin L$ “:  $z \neq \varepsilon$

Wir pumpen  $y$  so, dass Wortlänge  $s^2 + t$  für geeignete  $s, t$  mit  $0 < t < s$

Wähle  $k = 4 \cdot |y| \cdot |w|^2$

$$\text{Dann } |xy^k z| = |y| \cdot k + \underbrace{|x| + |z|}_{=t} = \underbrace{4 \cdot |y|^2 \cdot |w|^2}_{s=2 \cdot |y| \cdot |w|} + t = s^2 + t$$

Es gilt  $0 < t < s$ , also  $xy^k z \notin L$ , und wir gewinnen das Spiel!



# Kurze Erholungspause



# Noch ein Beispiel

## Beispiel 2.6

$L = \{a^n b^m \mid n \neq m\}$  ist nicht erkennbar.

### Beweis:

- „für alle natürlichen Zahlen  $n_0 \geq 1$ “: Gegner wählt  $n_0$
- „gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ “: Wir wählen  $w = a^{n_0} b^{n_0!+n_0}$
- „so dass für alle Zerleg.  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$ “: Gegner wählt  $xyz$
- „gilt: es gibt  $k \geq 0$  mit  $xy^k z \notin L$ “:

Sei  $x = a^{k_1}$ ,  $y = a^{k_2}$ ,  $z = a^{k_3} b^{n_0!+n_0}$ , wobei  $k_1 + k_2 + k_3 = n_0$  und  $k_2 > 0$

Offenbar existiert ein  $\ell$  mit  $\ell \cdot k_2 = n_0!$  (da  $0 < k_2 \leq n_0$ ).

Wir wählen  $k = \ell + 1$ . Die Anzahl a's im Wort  $xy^{\ell+1}z$  ist

$$k_1 + (\ell + 1)k_2 + k_3 = k_1 + k_2 + k_3 + (\ell \cdot k_2) = n_0 + n_0!$$

und die Anzahl von b's ebenso; das Wort ist also nicht in  $L$ .

Damit gewinnen wir das Spiel!



# Kurze Erholungspause



# Grenzen des Pumping-Lemmas

Das eben formulierte Pumping-Lemma ist nützlich,  
um **Nichterkennbarkeit** nachzuweisen.

Es kann aber nicht immer verwendet werden:

Es gibt Sprachen, die **nicht erkennbar** sind,  
aber die Eigenschaften des Lemmas **trotzdem erfüllen**.

Die formulierte Eigenschaft ist also **notwendig** für Erkennbarkeit,  
aber **nicht hinreichend**.



# Ein Beispiel

**Beispiel 2.7:** eine nicht erkennbare Sprache, die Lemma 2.2 erfüllt:

$$L = \{a^m b^n c^n \mid m, n \geq 1\} \cup \{b^m c^n \mid m, n \geq 0\}$$

Versucht man, Nichterkennbarkeit mit Lemma 2.2 zu zeigen, so scheitert man:

- „für alle natürlichen Zahlen  $n_0 \geq 1$ “: Gegner wählt  $n_0 = 3$
  - „gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ “: Wir wählen **schlaues**  $w \in L$
  - „so dass für alle Zerleg.  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$ “:

**1. Fall:** Wir haben  $w = a^m b^n c^n$  gewählt mit  $m, n \geq 1$  (1. Teil von  $L$ )

Gegner wählt  $x = \varepsilon$ ,  $y = a$ ,  $z = \text{Restwort}$

- „gilt: es gibt  $k \geq 0$  mit  $xy^kz \notin L$ “:

Für jedes  $k \geq 0$  ist nun  $xy^k z = a^{m+k-1} b^n c^n \in L$

( $m = 1$  und  $k = 0$  ist Sonderfall:  $xy^kz$  hat keine führenden a's)



# Ein Beispiel

**Beispiel 2.7:** eine nicht erkennbare Sprache, die Lemma 2.2 erfüllt:

$$L = \{a^m b^n c^n \mid m, n \geq 1\} \cup \{b^m c^n \mid m, n \geq 0\}$$

Versucht man, Nichterkennbarkeit mit Lemma 2.2 zu zeigen, so scheitert man:

- „für alle natürlichen Zahlen  $n_0 \geq 1$ “: Gegner wählt  $n_0 = 3$
  - „gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ “: Wir wählen schlaues  $w \in L$
  - „so dass für alle Zerleg.  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$ “:

**2. Fall:** Wir haben  $w = b^m c^n$  gewählt mit  $m \geq 1, n \geq 0$  (2. Teil von L)

Gegner wählt  $x = \varepsilon$ ,  $y = b$ ,  $z = \text{Restwort}$

- „gilt: es gibt  $k \geq 0$  mit  $xy^kz \notin L$ “:

Für jedes  $k \geq 0$  ist nun  $xy^k z = b^{m+k-1}c^n \in L$



# Ein Beispiel

**Beispiel 2.7:** eine nicht erkennbare Sprache, die Lemma 2.2 erfüllt:

$$L = \{a^m b^n c^n \mid m, n \geq 1\} \cup \{b^m c^n \mid m, n \geq 0\}$$

Versucht man, Nichterkennbarkeit mit Lemma 2.2 zu zeigen, so scheitert man:

- „für alle natürlichen Zahlen  $n_0 \geq 1$ “: Gegner wählt  $n_0 = 3$
- „gibt es ein Wort  $w \in L$  mit  $|w| \geq n_0$ “: Wir wählen **schlaues**  $w \in L$
- „so dass für alle Zerleg.  $w = xyz$  mit  $y \neq \varepsilon$  und  $|xy| \leq n_0$ “:

**3. Fall:** Wir haben  $w = c^n$  gewählt mit  $n \geq 1$  (2. Teil von  $L$ )

Gegner wählt  $x = \varepsilon$ ,  $y = c$ ,  $z = \text{Restwort}$

- „gilt: es gibt  $k \geq 0$  mit  $xy^k z \notin L$ “:

Für jedes  $k \geq 0$  ist nun  $xy^k z = c^{n+k-1} \in L$

In **keinem** Fall gewinnen wir das Spiel!



# Abschlussbemerkungen Pumping-Lemma

## Zusammenfassung

- ★ Pumping-Lemma ist ein wichtiges Werkzeug zum Nachweis der **Nichterkennbarkeit**.
- ★ Es kann **nicht** verwendet werden um zu zeigen, dass eine Sprache **erkennbar** ist!
- ★ Bequem ist die Betrachtungsweise als Kontraposition und Spiel.
- ★ Die Methode ist jedoch kein Automatismus, **erfordert Kreativität**.
- ★ Es gibt stärkere Versionen des Pumping-Lemmas die für alle nichterkennbaren Sprachen „funktionieren“.

## Intuition



Sprachen, deren Erkennen **unbeschränktes Zählen** erfordert, sind **nicht** erkennbar.



## Teil I: Endliche Automaten und reguläre Sprachen

- 0. Grundbegriffe
- 1. Endliche Automaten
- 2. Nachweis der Nichterkennbarkeit
- 3. Abschlusseigenschaften
- 4. Entscheidungsprobleme
- 5. Reguläre Ausdrücke und Sprachen
- 6. Minimale DEAs und die Nerode-Rechtskongruenz



## Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

- 7. Die Chomsky-Hierarchie
- 8. Rechtslineare Grammatiken und reguläre Sprachen
- 9. Normalformen und Entscheidungsprobleme
- 10. Abschlusseigenschaften und Pumping-Lemma
- 11. Kellerautomaten
- 12. Die Struktur kontextfreier Sprachen

## §3 Abschlusseigenschaften



# Abschlusseigenschaften

Endliche Automaten definieren die **Klasse** der **erkennbaren Sprachen**.

Anstatt Eigenschaften einzelner Sprachen zu studieren,  
kann man auch die **Eigenschaften ganzer Sprachklassen** analysieren.

Wir beweisen hier **Abschlusseigenschaften** der Klasse der erkennbaren Sprachen, wie z. B.:

Wenn  $L_1$  und  $L_2$  erkennbar sind, dann ist auch  $L_1 \cap L_2$  erkennbar.  
(wir bleiben also in derselben Sprachklasse)

Solche Eigenschaften sind sehr **nützlich in Konstruktionen und Beweisen**, z. B. um manchmal die Anwendung des Pumping-Lemmas zu vermeiden.



# Abschlusseigenschaften der erkennbaren Sprachen

## Satz 3.1

Sind  $L_1, L_2$  erkennbar, so auch

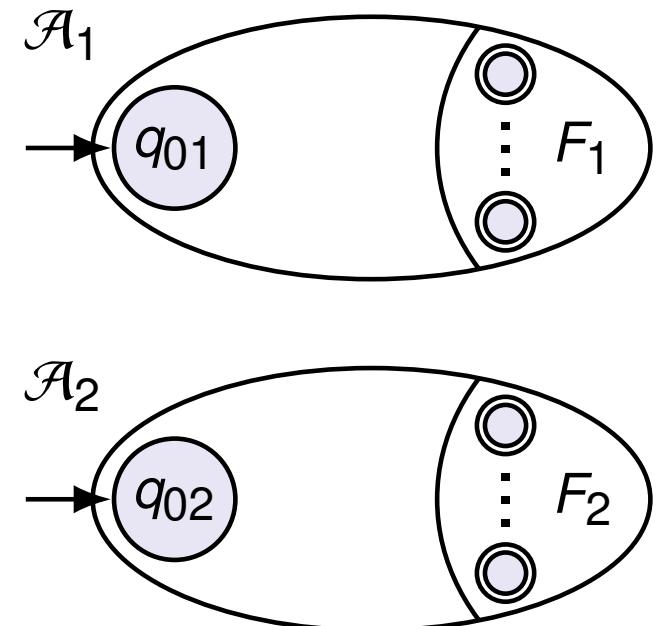
- (1)  $L_1 \cup L_2$  (Vereinigung)
- (2)  $\overline{L_1}$  (Komplement)
- (3)  $L_1 \cap L_2$  (Schnitt)
- (4)  $L_1 \cdot L_2$  (Konkatenation)
- (5)  $L_1^*$  (Kleene-Stern)

## Beweis.

Sei  $\mathcal{A}_i = (Q_i, \Sigma, q_{0i}, \Delta_i, F_i)$  ein NEA für  $L_i$  ( $i = 1, 2$ ).

O. B. d. A. sei  $Q_1 \cap Q_2 = \emptyset$ .

**Ziel:** Konstruiere NEAs für  $L_1 \cup L_2$ ,  $\overline{L_1}$ ,  $L_1 \cap L_2$ ,  $L_1 \cdot L_2$ ,  $L_1^*$



# Abschluss unter Vereinigung

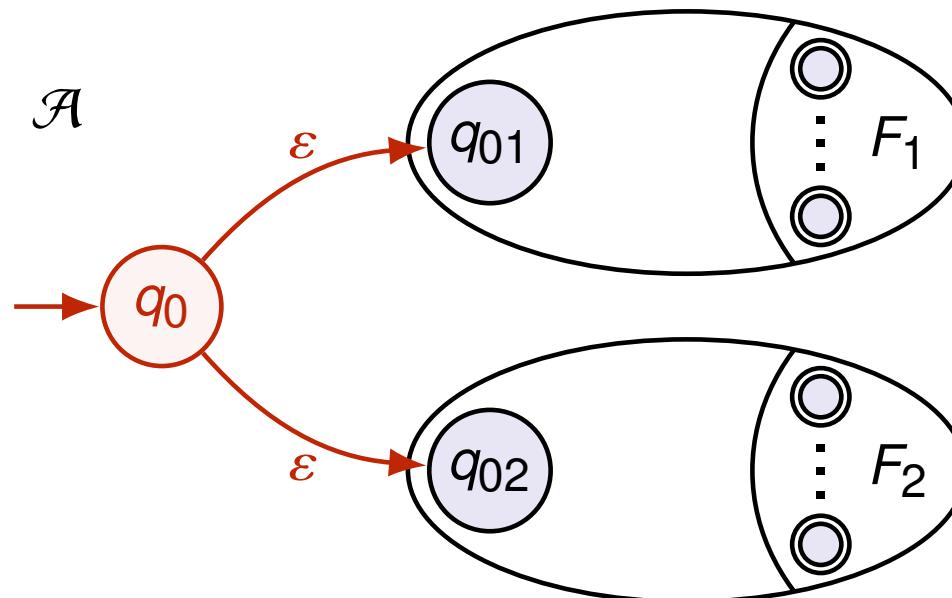
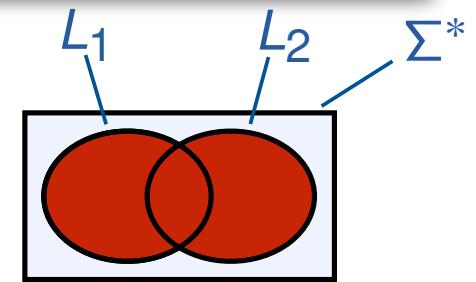
## Teil (1)

Der folgende  $\varepsilon$ -NEA erkennt  $L_1 \cup L_2$ :

$$\mathcal{A} := (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, q_0, \Delta, F_1 \cup F_2)$$

wobei  $q_0 \notin Q_1 \cup Q_2$  und

$$\Delta := \Delta_1 \cup \Delta_2 \cup \{(q_0, \varepsilon, q_{01}), (q_0, \varepsilon, q_{02})\}$$



Mit Lemma 1.18 gibt es zu  $\mathcal{A}$  einen äquivalenten NEA.

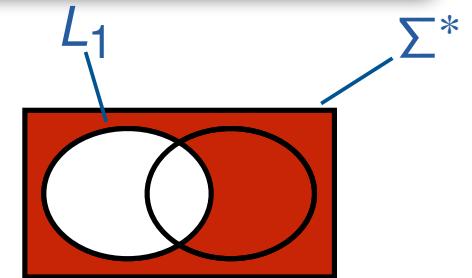


# Abschluss unter Komplement

## Teil (2) 2 Schritte:

- (i) Potenzmengenkonstr. liefert zu  $\mathcal{A}_1$  äquivalenten  
DEA  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ .
- (ii) Vertauschen der akzeptierenden und nicht-akzeptierenden Zustände  
liefert DEA für  $\overline{L_1}$ :

$$\overline{\mathcal{A}} := (Q, \Sigma, q_0, \delta, Q \setminus F)$$



Es gilt nämlich:

$$\begin{aligned} w \in \overline{L_1} &\quad \text{gdw. } w \notin L(\mathcal{A}) \\ &\quad \text{gdw. } \hat{\delta}(q_0, w) \notin F \\ &\quad \text{gdw. } \hat{\delta}(q_0, w) \in Q \setminus F \\ &\quad \text{gdw. } w \in L(\overline{\mathcal{A}}) \end{aligned}$$

**Beachte:** Mit einem NEA funktioniert diese Konstruktion **nicht!**



# Abschluss unter Schnitt

## Teil (3)

Wir konstruieren den **Produktautomaten**

$$\mathcal{A} := (Q_1 \times Q_2, \Sigma, (q_{01}, q_{02}), \Delta, F_1 \times F_2)$$

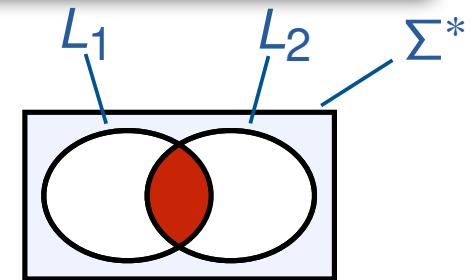
mit

$$\Delta := \left\{ \left( (q_1, q_2), a, (q'_1, q'_2) \right) \mid \begin{array}{l} (q_1, a, q'_1) \in \Delta_1 \text{ und} \\ (q_2, a, q'_2) \in \Delta_2 \end{array} \right\}$$

d. h.: ein Übergang in  $\mathcal{A}$  ist genau dann möglich,  
wenn der entsprechende Übergang in  $\mathcal{A}_1$  **und**  $\mathcal{A}_2$  möglich ist.

Wir beweisen nun, dass gilt:

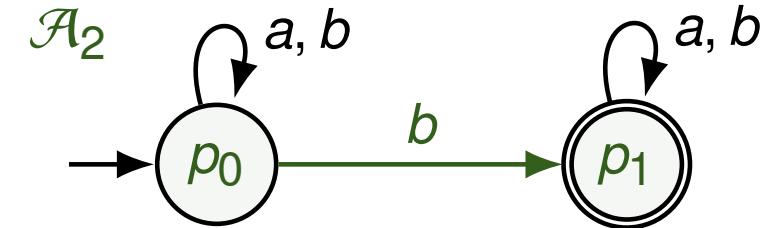
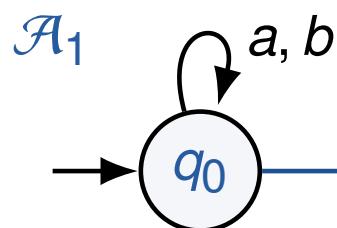
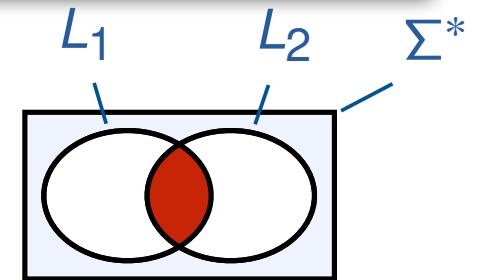
$$L(\mathcal{A}) = L_1 \cap L_2$$



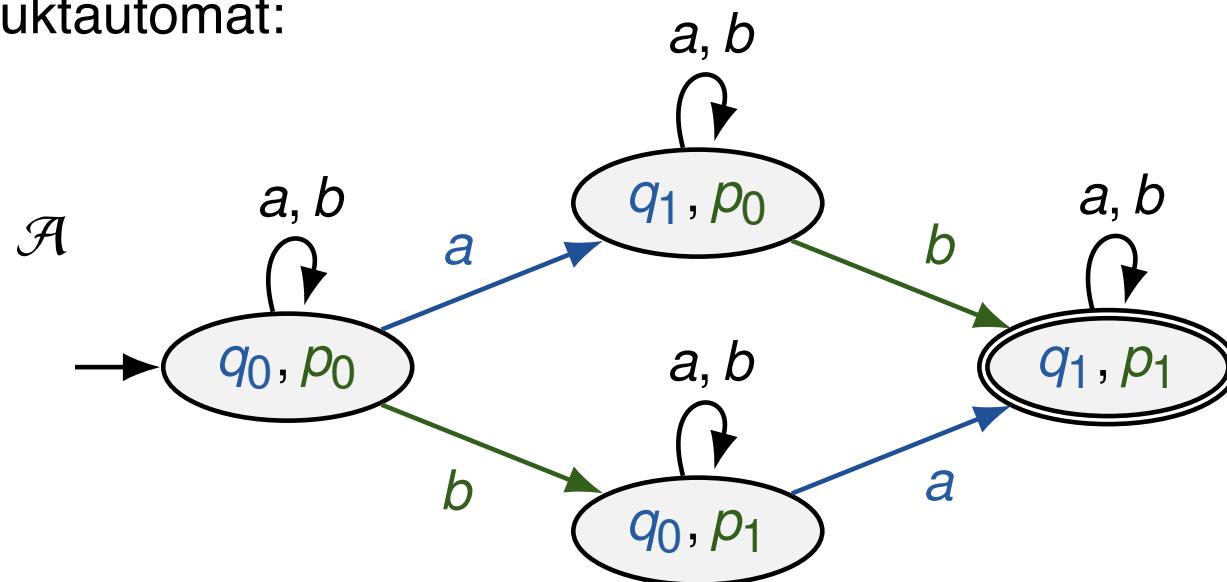
# Abschluss unter Schnitt

## Teil (3)

Beispiel für die Produktkonstruktion:



Produktautomat:



# Abschluss unter Konkatenation

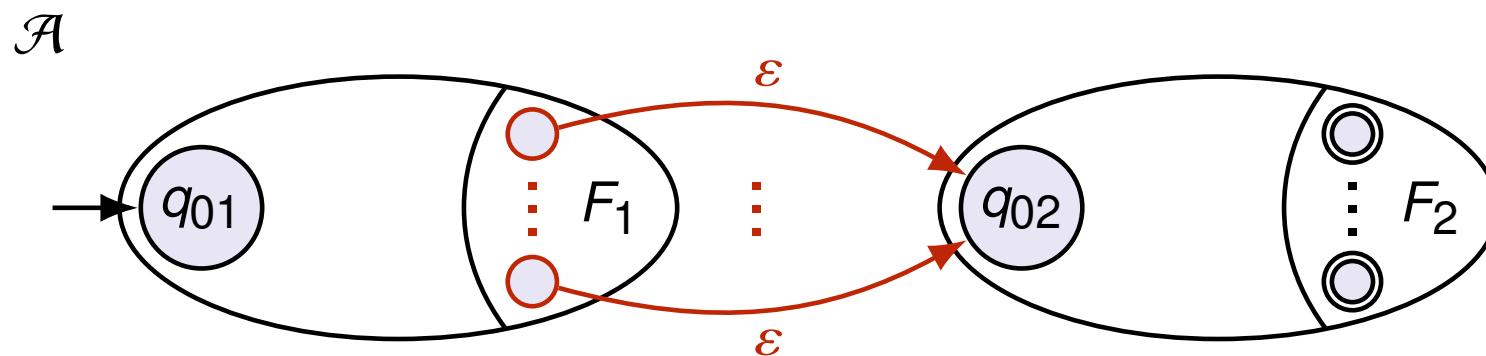
## Teil (4)

Der folgende  $\varepsilon$ -NEA erkennt  $L_1 \cdot L_2$ :

$$\mathcal{A} := (Q_1 \cup Q_2, \Sigma, q_{01}, \Delta, F_2)$$

wobei

$$\Delta := \Delta_1 \cup \Delta_2 \cup \{(f, \varepsilon, q_{02}) \mid f \in F_1\}$$



# Abschluss unter Kleene-Stern

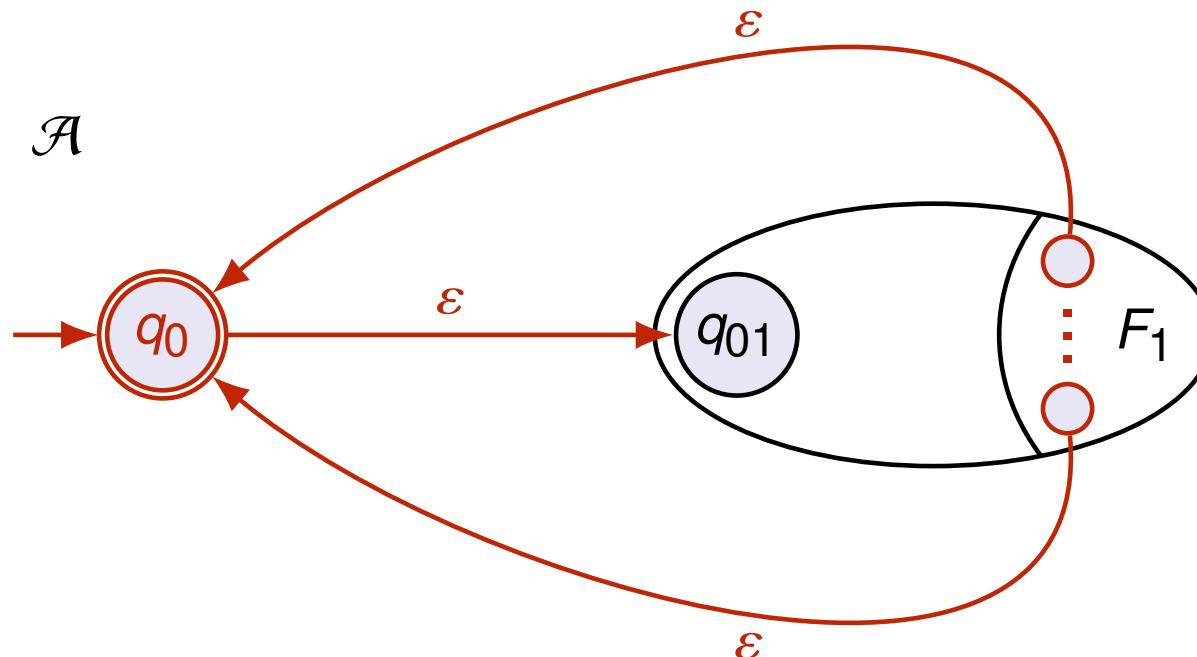
## Teil (5)

Der folgende  $\varepsilon$ -NEA erkennt  $L_1^*$ :

$$\mathcal{A} := (Q_1 \cup \{q_0\}, \Sigma, q_0, \Delta, \{q_0\})$$

wobei

$$\Delta := \Delta_1 \cup \{(f, \varepsilon, q_0) \mid f \in F_1\} \cup \{(q_0, \varepsilon, q_{01})\}$$



# Größe der konstruierten Automaten

Die Automaten für  $L_1 \cup L_2$ ,  $L_1 \cap L_2$ ,  $L_1 \cdot L_2$  und  $L_1^*$  sind **polynomiell groß** in der Größe der Automaten für  $L_1$ ,  $L_2$ .

- für  $L_1 \cup L_2$ :  $|Q| = |Q_1| + |Q_2| + 1$  (linear viele Zustände)
- für  $L_1 \cap L_2$ :  $|Q| = |Q_1| \cdot |Q_2|$  (quadratisch viele Zustände)
- für  $L_1 \cdot L_2$ :  $|Q| = |Q_1| + |Q_2|$  (linear viele Zustände)
- für  $L_1^*$ :  $|Q| = |Q_1| + 1$  (linear viele Zustände)

Beim **Komplement** kann die Konstruktion **exponentiell** sein, wenn man von einem NEA ausgeht.

$$\text{d. h. } |Q| = 2^{|Q_1|}$$

Man kann zeigen, dass sich das nicht vermeiden lässt.



# Nachweis der Nichterkennbarkeit

„Trick“, mit dem man manchmal das Pumping-Lemma vermeiden kann:

## Beispiel 3.2

$L = \{a^n b^m \mid n \neq m\}$  ist nicht erkennbar.

### Beweis.

Anstatt dies mit dem Pumping-Lemma zu zeigen,  
kann man auch verwenden, dass

$$L' := \{a^n b^n \mid n \geq 0\}$$

nicht erkennbar ist.

Es gilt nämlich  $L' = \overline{L} \cap (\{a\}^* \cdot \{b\}^*)$ .

Wäre also  $L$  erkennbar, dann mit Satz 3.1 auch  $L'$

Da  $L'$  nicht erkennbar ist (Bsp. 2.3), kann auch  $L$  nicht erkennbar sein.



# Homomorphismen

Wir schauen uns eine weitere Operation auf Sprachen an, die später in der Vorlesung nochmal eine Rolle spielen wird

## Definition 3.3 (Homomorphismus)

Seien  $\Sigma$  und  $\Gamma$  Alphabete. Ein Homomorphismus von  $\Sigma^*$  nach  $\Gamma^*$  ist eine Abbildung  $h : \Sigma^* \rightarrow \Gamma^*$ , so dass  $h(wv) = h(w)h(v)$  für alle  $w, v \in \Sigma^*$ .

Aus dieser Definition folgt unmittelbar:

1.  $h(\varepsilon) = \varepsilon$
2.  $h(a_1 \cdots a_n) = h(a_1) \cdots h(a_n)$ ,

Also kann man  $h$  durch Angabe von  $h(a) \in \Gamma^*$  für alle  $a \in \Sigma$  definieren

**Beispiel:**  $h(a) = ccc \quad h(b) = \varepsilon \quad h(c) = ab$

Dann gilt  $h(abcab) = cccabccc$



# Homomorphismen

Sei  $h$  ein Homomorphismus von  $\Sigma^*$  nach  $\Gamma^*$ .

Für Sprache  $L \subseteq \Sigma^*$  ist

$$h(L) = \{h(w) \mid w \in L\}$$

das homomorphe Bild von  $L$  unter  $h$ .

**Beispiel 3.4:** Sei  $L = \{a^n b^n \mid n \geq 0\}$ . Für

- $h(a) = b$  und  $h(b) = a$  gilt:  $h(L) = \{b^n a^n \mid n \geq 0\}$
- $h(a) = a$  und  $h(b) = a$  gilt:  $h(L) = \{w \in a^n \mid n \text{ geradzahlig}\}$
- $h(a) = a$  und  $h(b) = \varepsilon$  gilt:  $h(L) = \{a\}^*$



# Homomorphismen

Die erkennbaren Sprachen sind unter homomorphen Bildern abgeschlossen:

## Satz 3.5

Sei  $L \subseteq \Sigma^*$  eine erkennbare Sprache und  $h : \Sigma^* \rightarrow \Gamma^*$  ein Homomorphismus. Dann ist  $h(L)$  ebenfalls eine erkennbare Sprache.

### Beweis.

Sei  $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$  ein NEA für  $L$ .

Konstruiere NEA mit Wortübergängen  $\mathcal{A}' = (Q, \Gamma, q_0, \Delta', F)$  durch:

$$\Delta' = \{(q_1, h(a), q_2) \mid (q_1, a, q_2) \in \Delta\}$$

Man prüft leicht, dass  $L(\mathcal{A}') = h(L)$ .

□



# Nachweis der Nichterkennbarkeit

Auch mit Homomorphismen kann man manchmal das Pumping-Lemma vermeiden:

## Beispiel 3.5

$L = \{(ab)^n c^* (de)^n \mid n \geq 0\}$  ist **nicht** erkennbar.

### Beweis.

Wir verwenden wieder, dass

$$L' := \{a^n b^n \mid n \geq 0\}$$

**nicht** erkennbar ist.

Für  $h(a) = a \quad h(b) = h(c) = h(d) = \varepsilon \quad h(e) = b$

gilt  $L' = h(L)$ .

Wäre also  $L$  erkennbar, dann mit Satz 3.5 auch  $L'$ .

Da  $L'$  nicht erkennbar ist (Bsp. 2.3), kann auch  $L$  nicht erkennbar sein.



## Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz



## Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

7. Die Chomsky-Hierarchie
8. Rechtslineare Grammatiken und reguläre Sprachen
9. Normalformen und Entscheidungsprobleme
10. Abschlusseigenschaften und Pumping-Lemma
11. Kellerautomaten
12. Die Struktur kontextfreier Sprachen

# § 4 Entscheidungsprobleme



# Entscheidungsprobleme

Endliche Automaten können auf verschiedene Weise in einer **konkreten Anwendung** eingesetzt werden.

Die wichtigste Rolle spielen die folgenden Probleme:

★ das **Wortproblem**:

Gegeben Automat  $\mathcal{A}$  und Eingabe  $w$ , ist  $w \in L(\mathcal{A})$  ?

Anwendungsbeispiel: Automat beschreibt ein **Suchpattern**, z.B.

Teilwort “login” oder “username” und Teilwort “passwort”  
in beliebiger Reihenfolge und Groß/Kleinschreibung

Wörter entsprechen Dokumenten/Emails/etc (= Zeichenfolgen)

Das Wortproblem **entspricht dann**:

Gegeben ein Suchpattern  $\mathcal{A}$  und ein Dokument  $w$ ,  
enthält  $w$  das Suchpattern?



# Entscheidungsprobleme

Endliche Automaten können auf verschiedene Weise in einer **konkreten Anwendung** eingesetzt werden.

Die wichtigste Rolle spielen die folgenden Probleme:

★ das **Wortproblem**:

Gegeben Automat  $\mathcal{A}$  und Eingabe  $w$ , ist  $w \in L(\mathcal{A})$  ?

★ das **Leerheitsproblem**:

Gegeben Automat  $\mathcal{A}$ , ist  $L(\mathcal{A}) = \emptyset$  ?

Anwendungsbeispiel:

Gibt es überhaupt ein Dokument, dass dem Suchpattern entspricht?



# Entscheidungsprobleme

Endliche Automaten können auf verschiedene Weise in einer **konkreten Anwendung** eingesetzt werden.

Die wichtigste Rolle spielen die folgenden Probleme:

★ das **Wortproblem**:

Gegeben Automat  $\mathcal{A}$  und Eingabe  $w$ , ist  $w \in L(\mathcal{A})$  ?

★ das **Leerheitsproblem**:

Gegeben Automat  $\mathcal{A}$ , ist  $L(\mathcal{A}) = \emptyset$  ?

★ das **Äquivalenzproblem**:

Gegeben Automaten  $\mathcal{A}_1$  und  $\mathcal{A}_2$ , ist  $L(\mathcal{A}_1) = L(\mathcal{A}_2)$  ?

Anwendungsbeispiel:

Beschreiben zwei Patterns dieselbe Suche?



# Entscheidungsprobleme

Endliche Automaten können auf verschiedene Weise in einer **konkreten Anwendung** eingesetzt werden.

Die wichtigste Rolle spielen die folgenden Probleme:

★ das **Wortproblem**:

Gegeben Automat  $\mathcal{A}$  und Eingabe  $w$ , ist  $w \in L(\mathcal{A})$  ?

★ das **Leerheitsproblem**:

Gegeben Automat  $\mathcal{A}$ , ist  $L(\mathcal{A}) = \emptyset$  ?

★ das **Äquivalenzproblem**:

Gegeben Automaten  $\mathcal{A}_1$  und  $\mathcal{A}_2$ , ist  $L(\mathcal{A}_1) = L(\mathcal{A}_2)$  ?

Jedes Problem gibt es in **zwei Varianten**: für NEAs und für DEAs!

**Unser Ziel:** Algorithmen für diese Probleme finden,  
mit möglichst **geringer Laufzeit**.



# Das Wortproblem für DEAs

Definition: Wortproblem

Gegeben: DEA oder NEA  $\mathcal{A}$  und Eingabe  $w \in \Sigma^*$  für  $\mathcal{A}$

Frage: Gilt  $w \in L(\mathcal{A})$  ?

Wenn  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  **DEA**:

Berechne Zustand  $\hat{\delta}(q_0, w)$  durch wiederholte Anwendung von  $\delta$ ;  
überprüfe dann nur noch, ob das ein akzeptierender Zustand ist.

**Laufzeitanalyse:**

$\delta$  muss  $|w|$ -mal angewendet werden;  
jede Anwendung braucht max.  $O(|\delta|)$  Zeit.

Gesamtzahl der Übergänge

Dies liefert:

Satz 4.1

Das Wortproblem für **DEAs** ist entscheidbar in Zeit  $O(|w| \cdot |\delta|)$ .



# Das Wortproblem für DEAs

Satz 4.1

Das Wortproblem für DEAs ist entscheidbar in Zeit  $O(|w| \cdot |\delta|)$ .

Häufig betrachtet man aber den Automaten  $\mathcal{A}$  als **fest**  
und das Wort  $w$  als **die alleinige Eingabe**

Die Größe von  $\mathcal{A}$  wird dann zur Konstante und die Laufzeit **linear**, also  $O(|w|)$ .

Motivation:

- häufig werden **viele Worte  $w$**  für denselben Automaten  $\mathcal{A}$  getestet
- der Automat  $\mathcal{A}$  ist oft **klein** im Vergleich zum Wort  $w$

Zum Beispiel **Suchpattern** versus **Dokument**

Man nennt diese Art der Betrachtung auch **Datenkomplexität**  
(vergleiche VL Logik)



# Das Wortproblem für NEAs

Für einen **NEA** ist das Wortproblem nicht auf dieselbe Weise lösbar, da es für eine Eingabe  $w$  **mehrere Pfade** durch den NEA geben kann.

**Naive Ansätze** führen zu schlechter Laufzeit:

- Alle Pfade für Eingabewort **durchprobieren**:  
Im schlimmsten Fall  $|Q|^{|w|+1}$  viele, also **exponentielle Laufzeit**
- Erst **Potenzmengenkonstruktion** anwenden:  
Resultierender DEA hat  $2^{|Q|}$  viele Zustände, also  $|\delta| \approx 2^{O(|Q| \cdot |\Sigma|)}$   
Insgesamt also **exponentielle Laufzeit**  $O(|w| \cdot 2^{O(|Q| \cdot |\Sigma|)})$

Man findet aber dennoch einen Algorithmus, der polynomiell ist und **linear in Datenkomplexität**



# Das Wortproblem für NEAs

Betrachte folgenden Algorithmus:

```
procedure akz( $\mathcal{A}$ ,  $w$ )
```

Sei  $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$  und  $w = a_1 \cdots a_n$ .

$Q_0 := \{q_0\}$

for  $1 \leq i \leq n$  do

$Q_i := \{q \in Q \mid \exists p \in Q_{i-1} : (p, a_i, q) \in \Delta\}$

if  $Q_n \cap F \neq \emptyset$  then return ‘yes’

else return ‘no’

Lemma 4.2

$\text{akz}(\mathcal{A}, w) = \text{yes}$  gdw.  $w$  von  $\mathcal{A}$  akzeptiert wird.

Beweisidee:

Algorithmus simuliert DEA, der aus  $\mathcal{A}$  per Potenzmengenkonstruktion entsteht



# Das Wortproblem für NEAs

## Laufzeitanalyse:

- Der Algorithmus stoppt nach  $|w|$  Iterationen.
- In jeder Iteration gehen wir durch alle Zustände  $p \in Q_{i-1}$ , für jedes  $p$  dann durch alle Übergänge in  $\Delta$  — Zeit  $O(|Q| \cdot |\Delta|)$ .
- Naive Implementierung von „ $Q_n \cap F \neq \emptyset$ “ in Zeit  $O(|Q|^2)$

### Satz 4.3

Das Wortproblem für NEAs ist in polynomieller Zeit entscheidbar, genauer gesagt in Zeit  $O((|w| \cdot |Q| \cdot |\Delta|) + |Q|^2)$ .

Bezüglich Datenkomplexität erhalten wir also wieder Linearzeit!



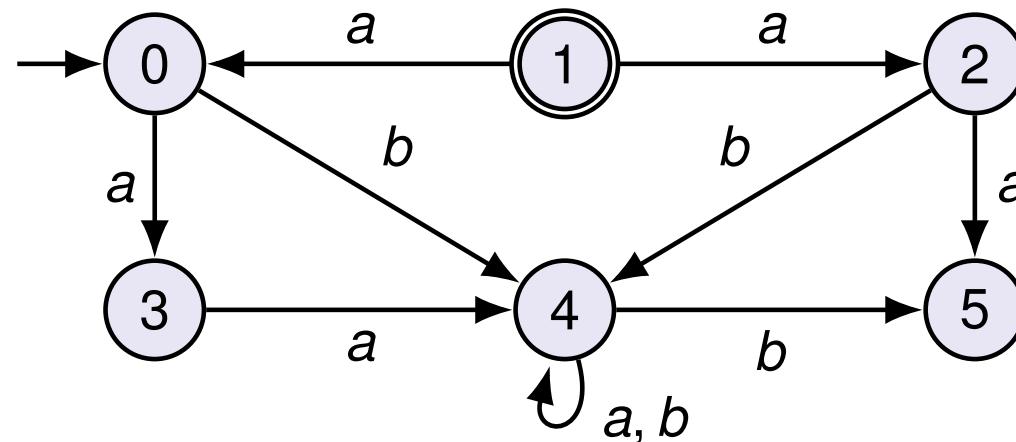
# Das Leerheitsproblem

Definition: Leerheitsproblem

Gegeben: NEA  $\mathcal{A}$

Frage: Gilt  $L(\mathcal{A}) = \emptyset$ ?

Zunächst scheint es, als ob man **alle (unendliche vielen)** Eingaben betrachten müsse — aber das ist leicht zu vermeiden:



Gilt  $L(\mathcal{A}) = 0$ ? Warum bzw. warum nicht?

# Das Leerheitsproblem

Formal definiert man zu jedem NEA  $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$  einen gerichteten Graph  $G_{\mathcal{A}} = (V, E)$  wie folgt:

$$V = Q \quad E = \{(q_1, q_2) \mid \exists a \in \Sigma : (q_1, a, q_2) \in \Delta\}$$

Folgendes ist leicht zu sehen:

Lemma 4.4

$L(\mathcal{A}) = \emptyset$  gdw. in  $G_{\mathcal{A}}$  kein  $q \in F$  von  $q_0$  aus erreichbar ist.

Das Erreichbarkeitsproblem in gerichteten Graphen ist mittels Breitensuche in Zeit  $O(|V| + |E|)$  lösbar (VL Algorithmen und Datenstrukturen)

Man nimmt dabei an, dass die Kantenmenge  $E$  als Liste gegeben ist  
(und im Folgenden: die Übergangsrelation  $\Delta$  als Liste von Tripeln)



# Das Leerheitsproblem

Wir können  $G_{\mathcal{A}} = (V, E)$  leicht in Zeit  $|Q| + |\Delta|$  aus  $\mathcal{A}$  konstruieren

Es gilt  $|V| = |Q|$  und  $|E| \leq |\Delta|$ , wir erhalten also:

Satz 4.5

Das Leerheitsproblem für NEAs ist entscheidbar in Zeit  $O(|Q| + |\Delta|)$ .



# Nochmal: Das Wortproblem für NEAs

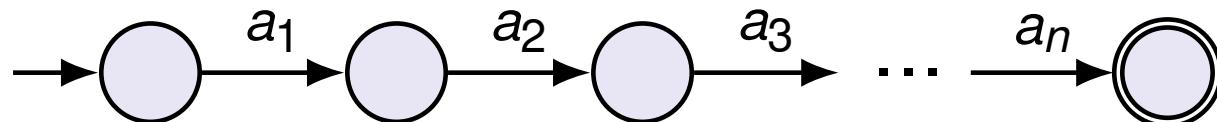
Hier nochmal mit anderem Ansatz und leicht verbesserter Laufzeit

## Satz 4.6

Das Wortproblem für NEAs ist entscheidbar in Zeit  $O(|w| \cdot (|Q| + |\Delta|))$ .

### Beweis:

Gegeben NEA  $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$  und Eingabe  $w$ , konstruiere zunächst NEA  $\mathcal{A}_w$ , der genau  $w = a_1 \cdots a_n$  akzeptiert, d. h.  $L(\mathcal{A}_w) = \{a_1 \cdots a_n\}$ :



Dieser Automat hat  $|w| + 1$  Zustände.

Offenbar ist  $w \in L(\mathcal{A})$  gdw.  $L(\mathcal{A}) \cap L(\mathcal{A}_w) \neq \emptyset$

(Reduktion des Wortproblems auf das Leerheitsproblem)

# Das Wortproblem für NEAs

Wir können also folgenden Algorithmus verwenden:

- Produktautomat für  $\mathcal{A}$  und  $\mathcal{A}_w$  erzeugen; erkennt  $L(\mathcal{A}) \cap L(\mathcal{A}_w)$
- (Nicht)-Leerheit in Zeit  $O(|Q'| + |\Delta'|)$  prüfen (Satz 4.3)  
  
des Produktautomaten!

Produktautomat kann leicht in Zeit  $O(|w| \cdot (|Q| + |\Delta|))$  erzeugt werden

Größe Produktautomat:

- Anzahl Zustände:  $|Q'| \leq |Q| \cdot (|w| + 1)$
- Anzahl Übergänge:  $|\Delta'| \leq |w| \cdot |\Delta|$   
(da  $\mathcal{A}_w$  genau  $|w|$  Übergänge hat)

Aufwand zum Testen von  $L(\mathcal{A}) \cap L(\mathcal{A}_w) \neq \emptyset$  also:

$$\begin{aligned} & O(|w| \cdot (|Q| + |\Delta|)) + && (\text{erzeugen}) \\ & O(|Q| \cdot (|w| + 1)) + |w| \cdot |\Delta| && (\text{Leerheit prüfen}) \\ = & O(|w| \cdot (|Q| + |\Delta|)) \end{aligned}$$

□

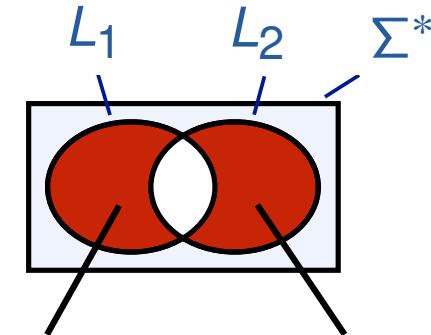


# Das Äquivalenzproblem

Definition: Äquivalenzproblem

Gegeben: DEAs oder NEAs  $\mathcal{A}_1, \mathcal{A}_2$

Frage: Gilt  $L(\mathcal{A}_1) = L(\mathcal{A}_2)$ ?



**Reduktion des ÄP auf das Leerheitsproblem:**

$$L_1 = L_2 \quad \text{gdw.} \quad (L_1 \cap \overline{L_2}) \cup (L_2 \cap \overline{L_1}) = \emptyset$$

- Vereinigung und Schnitt:  
Konstruktion vergrößert DEAs & NEAs **nur polynomiell**
- Komplement:  
für DEAs keine Vergrößerung; für NEAs **exponentielle Vergrößerung**

Satz 4.7

Das Äquivalenzproblem ist für **DEAs** entscheidbar in **polynomieller Zeit** und für **NEAs** in **exponentieller Zeit**.

Man vermutet, dass das ÄP für NEAs nicht in polynomieller Zeit lösbar ist.



# Überblick Entscheidungsprobleme

Wortproblem

Leerheitsprob.

Äquivalenzprob.

NEA

Linearzeit\*

Linearzeit

Exponentialzeit

DEA

Linearzeit\*

Linearzeit

Polynomialzeit

\* Datenkomplexität



## Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz



## Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

7. Die Chomsky-Hierarchie
8. Rechtslineare Grammatiken und reguläre Sprachen
9. Normalformen und Entscheidungsprobleme
10. Abschlusseigenschaften und Pumping-Lemma
11. Kellerautomaten
12. Die Struktur kontextfreier Sprachen

# §5 Reguläre Ausdrücke und Sprachen



# Reguläre Ausdrücke und Sprachen

**Wir kennen bereits:**

4 äquivalente Charakterisierungen der Klasse der erkennbaren Sprachen:

Eine Sprache  $L \subseteq \Sigma^*$  ist **erkennbar** gdw.

- (1)  $L = L(\mathcal{A})$  für einen **DEA**  $\mathcal{A}$
- (2)  $L = L(\mathcal{A})$  für einen **NEA**  $\mathcal{A}$
- (3)  $L = L(\mathcal{A})$  für einen  **$\varepsilon$ -NEA**  $\mathcal{A}$
- (4)  $L = L(\mathcal{A})$  für einen **NEA mit Wortübergängen**  $\mathcal{A}$

**Im Folgenden betrachten wir:**

eine weitere nützliche Charakterisierung mittels **regulärer Ausdrücke**



# Reguläre Ausdrücke: Syntax

## Definition 5.1

Sei  $\Sigma$  ein endliches Alphabet.

Die Menge  $\text{Reg}_\Sigma$  der regulären Ausdrücke über  $\Sigma$  ist induktiv definiert:

- $\emptyset, \varepsilon, a$  (für  $a \in \Sigma$ ) sind Elemente von  $\text{Reg}_\Sigma$ .
- Sind  $r, s \in \text{Reg}_\Sigma$ , so auch  $(r + s), (r \cdot s), r^*$   $\in \text{Reg}_\Sigma$ .

## Beispiel 5.2

$((a \cdot b^*) + \emptyset^*)^*$  ist regulärer Ausdruck über  $\Sigma = \{a, b\}$

$((a \cdot b)^* + (c \cdot b)^*) + a^*$

ist regulärer Ausdruck über  $\Sigma = \{a, b, c\}$



# Zur Notation regulärer Ausdrücke

## Wir vereinbaren:

- $*$  bindet stärker als  $\cdot$
- $\cdot$  bindet stärker als  $+$
- $\cdot$  wird meist ganz weggelassen

Z.B. schreiben wir statt  $((a \cdot b^*) + \emptyset^*)^*$  kürzer  $(ab^* + \emptyset^*)^*$ .

Wir werden gleich sehen:  $+$  und  $\cdot$  sind assoziativ, also z.B.:

$((r + s) + t)$  und  $(r + (s + t))$  beschreiben dieselbe Sprache.

Daher lassen wir auch hier Klammern weg, schreiben  $r + s + t$ .

Statt  $((a \cdot b)^* + (c \cdot (b \cdot a))^* + a^*)$  schreiben wir also  $(ab)^* + (cba)^* + a^*$ .



# Reguläre Ausdrücke: Semantik

## Definition 5.3

Die durch den regulären Ausdruck  $r$  definierte Sprache  $L(r)$  ist induktiv definiert:

- $L(\emptyset) := \emptyset, \quad L(\varepsilon) := \{\varepsilon\}, \quad L(a) := \{a\}$
- $L(r + s) := L(r) \cup L(s), \quad L(r \cdot s) := L(r) \cdot L(s), \quad L(r^*) := L(r)^*$

Eine Sprache  $L \subseteq \Sigma^*$  heißt regulär, falls es ein  $r \in \text{Reg}_\Sigma$  gibt mit  $L = L(r)$ .

## Beispiel 5.4

$r$	$L(r)$
$(a + b)^* ab(a + b)^*$	Menge aller Wörter $w \in \{a, b\}^*$ mit Infix $ab$
$ab^* + b$	$\{ab^i \mid i \geq 0\} \cup \{b\}$
$b^* a^*$	Menge aller Wörter $w \in \{a, b\}^*$ ohne Infix $ab$
?	Schnitt der ersten beiden Sprachen



# Reguläre Ausdrücke: Semantik

## Definition 5.3

Die durch den regulären Ausdruck  $r$  definierte Sprache  $L(r)$  ist induktiv definiert:

- $L(\emptyset) := \emptyset, \quad L(\varepsilon) := \{\varepsilon\}, \quad L(a) := \{a\}$
- $L(r + s) := L(r) \cup L(s), \quad L(r \cdot s) := L(r) \cdot L(s), \quad L(r^*) := L(r)^*$

Eine Sprache  $L \subseteq \Sigma^*$  heißt regulär, falls es ein  $r \in \text{Reg}_\Sigma$  gibt mit  $L = L(r)$ .

## Beispiel 5.4

$r$	$L(r)$
$(a + b)^* ab(a + b)^*$	Menge aller Wörter $w \in \{a, b\}^*$ mit Infix $ab$
$ab^* + b$	$\{ab^i \mid i \geq 0\} \cup \{b\}$
$b^* a^*$	Menge aller Wörter $w \in \{a, b\}^*$ ohne Infix $ab$
$abb^*$	Schnitt der ersten beiden Sprachen



# Äquivalenz

In der Praxis und auch in theoretischen Konstruktionen ist es häufig sinnvoll, **reguläre Ausdrücke** soweit wie möglich **zu vereinfachen**

Für zwei reguläre Ausdrücke  $r, s$  schreiben wir

$$r \equiv s \text{ wenn } L(r) = L(s)$$

Wir nennen  $r$  und  $s$  dann **äquivalent**.

Wenn  $r \equiv s$ , dann können wir also  $r$  durch  $s$  ersetzen

Im Folgenden betrachten wir einige **nützliche Äquivalenzen**



# Einige Äquivalenzen

Einige grundlegende Äquivalenzen für + und  $\cdot$ :

$$r + (s + t) \equiv (r + s) + t$$

Assoziativität

$$r \cdot (s \cdot t) \equiv (r \cdot s) \cdot t$$

$$r + s \equiv s + r$$

Kommutativität

$$r(s + t) \equiv rs + rt$$

Distributivität

$$(r + s)t \equiv rt + st$$

$$r + \emptyset \equiv \emptyset + r \equiv r$$

Neutrale Elemente

$$r \cdot \varepsilon \equiv \varepsilon \cdot r \equiv r$$

$$r \cdot \emptyset \equiv \emptyset$$

Absorbierendes Element

Für jedes Alphabet  $\Sigma$  ist also die **Algebra**  $(\mathcal{L}(\text{Reg}_\Sigma), +, \cdot, \emptyset, \varepsilon)$  ein  
**Halbring** mit Nullelement  $\emptyset$  und Einselement  $\varepsilon$ .

Menge aller regulären Sprachen  
über Alphabet  $\Sigma$



# Einige Äquivalenzen

Zusätzlich gilt:

$$r + r \equiv r$$

Idempotenz +

$$\varepsilon + rr^* \equiv r^*$$

Äquivalenz 1 für Kleene Stern

$$\varepsilon + r^*r \equiv r^*$$

Äquivalenz 2 für Kleene Stern

Sind obige Äquivalenzen vollständig, d.h. lassen sich alle gültigen Äquivalenzen daraus herleiten? **Leider nicht!**

Schlimmer noch:

Man kann beweisen, dass es keine endliche Menge von Äquivalenzen gibt, die in diesem Sinne vollständig ist. [Redko 1964]

Wenn man aber auch **Teilsprachen** betrachtet und gewisse **Implikationen**, dann lässt sich “das Verhalten von regulären Sprachen endlich beschreiben”.



# Teilsprachen und Implikationen

Wir schreiben  $r \leq s$  für  $s \equiv s + r$ . („ $r$  definiert Teilsprache von  $s$ “)

Man überlegt sich leicht, dass

$$r \leq s \text{ gdw. } L(r) \subseteq L(s).$$

Es gilt beispielsweise:

$$\text{wenn } rs \leq s, \text{ dann } r^*s \leq s \quad \text{Implikation 1}$$

$$\text{wenn } sr \leq s, \text{ dann } sr^* \leq s \quad \text{Implikation 2}$$

Aus den gezeigten Äquivalenzen und Implikationen kann man alle gültigen Äquivalenzen für reguläre Ausdrücke durch Umformung herleiten

Dies führt zum Begriff der Kleene Algebren



# Vereinfachung Regulärer Ausdrücke

Beispiele für weniger offensichtliche Äquivalenzen, die zur Vereinfachung verwendet werden können:

$$(rs)^*r \equiv r(sr)^*$$

$$(r^*s)^*r^* \equiv (r + s)^*$$

$$r^*(sr^*)^* \equiv (r + s)^*$$

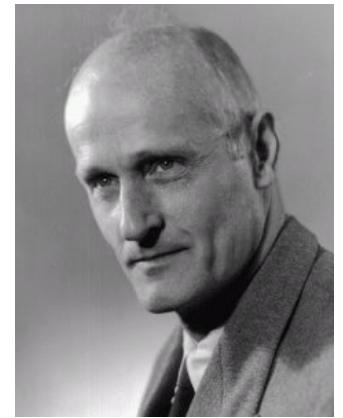
Überlegen Sie sich zur Übung gern

- Beispiele für diese Äquivalenzen,
- warum diese Äquivalenzen gelten und
- warum sie aus den gezeigten Basisäquivalenzen/Implikationen folgen



# Der Satz von Kleene

Stephen C. Kleene  
© Univ. of Wisconsin-Madison



Wir zeigen nun, dass man mit regulären Ausdrücken  
**genau** die erkennbaren Sprachen beschreiben kann.

## Satz 5.5 (Satz von Kleene)

Für eine Sprache  $L \subseteq \Sigma^*$  sind äquivalent:

- (1)  $L$  ist **regulär**.
- (2)  $L$  ist **erkennbar**.

Statt  $L(\mathcal{A}) = L(r)$  schreiben wir im Folgenden häufig einfach  $L(\mathcal{A}) = r$ .

Z.B.:  $L(\mathcal{A}) = a^* + b$



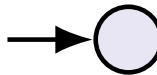
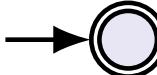
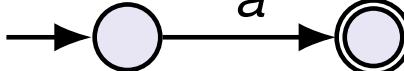
# Satz von Kleene: Beweis

**Beweis „1  $\Rightarrow$  2“** d. h. „wenn  $L$  regulär, dann  $L$  erkennbar“:

Sei  $L = L(r)$  für einen regulären Ausdruck  $r$ .

Induktion über den Aufbau von  $r$ :

## Induktionsanfang:

- $L(\emptyset) = \emptyset$  erkennbar:  ist NEA für  $\emptyset$  (kein akz. Zustand).
- $L(\varepsilon) = \{\varepsilon\}$  erkennbar:  ist NEA für  $\{\varepsilon\}$ .
- $L(a) = \{a\}$  erkennbar:  ist NEA für  $\{a\}$ .

## Induktionsschritt:

Weiß man bereits, dass  $L(r)$  und  $L(s)$  erkennbar sind, so folgt mit Satz 3.1 (Abschlusseigenschaften), dass auch

- $L(r + s) = L(r) \cup L(s)$ ,
- $L(r \cdot s) = L(r) \cdot L(s)$  und
- $L(r^*) = L(r)^*$  erkennbar sind.



# Satz von Kleene: Beweis

**Beweis „2  $\Rightarrow$  1“** d. h. „wenn  $L$  erkennbar, dann  $L$  regulär“:

Sei  $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$  ein NEA mit  $L = L(\mathcal{A})$ .

Für  $p, q \in Q$  und  $X \subseteq Q$  sei

$$L_{p,q}^X$$

die Menge der Wörter  $w = a_1 \cdots a_n$ , für die gilt: es gibt einen Pfad

$$p_0 \xrightarrow{\mathcal{A}} a_1 p_1 \xrightarrow{\mathcal{A}} a_2 p_2 \xrightarrow{\mathcal{A}} a_3 \cdots \xrightarrow{\mathcal{A}} a_n p_n$$

mit  $p_0 = p$ ,  $p_n = q$  und  $\{p_1, \dots, p_{n-1}\} \subseteq X$ .

Offensichtlich gilt:  $L(\mathcal{A}) = \bigcup_{q_f \in F} L_{q_0, q_f}^Q$

**Es genügt also zu zeigen:** Alle  $L_{p,q}^X$  sind regulär.



# Satz von Kleene: Beweis

**Zu zeigen:** Alle  $L_{p,q}^X$  sind regulär.

Induktion über die Größe von  $X$ .

**Induktionsanfang:**  $X = \emptyset$

**1. Fall:**  $p \neq q$ . Dann ist

$$L_{p,q}^\emptyset = \{a \in \Sigma \mid (p, a, q) \in \Delta\}$$

Also hat  $L_{p,q}^\emptyset$  die Form  $\{a_1, \dots, a_k\}$ . Regulärer Ausdruck:  $a_1 + \dots + a_k$

**2. Fall:**  $p = q$

$L_{p,q}^\emptyset$  enthält zusätzlich  $\varepsilon$ , also regulärer Ausdruck:  $a_1 + \dots + a_k + \varepsilon$

**Induktionsschritt:**  $X \neq \emptyset$

Wähle beliebiges  $\widehat{q} \in X$ . Dann gilt:

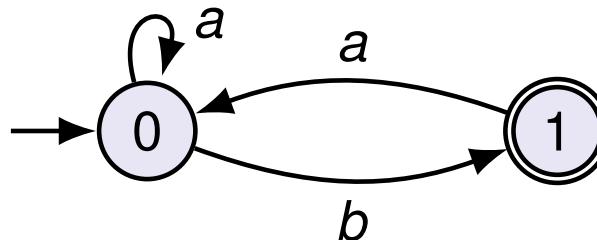
$$L_{p,q}^X = L_{p,q}^{X \setminus \{\widehat{q}\}} \cup \left( L_{p,\widehat{q}}^{X \setminus \{\widehat{q}\}} \cdot (L_{\widehat{q},\widehat{q}}^{X \setminus \{\widehat{q}\}})^* \cdot L_{\widehat{q},q}^{X \setminus \{\widehat{q}\}} \right)$$

Für die Sprachen auf der rechten Seite gibt es nach Induktionsvoraussetzung reguläre Ausdrücke — also auch für  $L_{p,q}^X$ .



# Satz von Kleene: Beispiel

## Beispiel 5.6



Da 1 der einzige akzeptierende Zustand ist, gilt:  $L(\mathcal{A}) = L_{0,1}^Q$

**Induktionsschritt liefert:**

$$L_{0,1}^Q = L_{0,1}^{\{0\}} \cup L_{0,\underline{1}}^{\{0\}} \cdot (L_{\underline{1},1}^{\{0\}})^* \cdot L_{1,1}^{\{0\}} = a^*b + a^*b \cdot (\varepsilon + aa^*b)^* \cdot (\varepsilon + aa^*b) \dots$$

$$L_{0,1}^{\{0\}} = L_{0,1}^\emptyset \cup L_{0,\underline{0}}^\emptyset \cdot (L_{\underline{0},0}^\emptyset)^* \cdot L_{0,1}^\emptyset = b + (a + \varepsilon) \cdot (a + \varepsilon)^* \cdot b = a^*b$$

$$L_{1,1}^{\{0\}} = L_{1,1}^\emptyset \cup L_{1,\underline{0}}^\emptyset \cdot (L_{\underline{0},0}^\emptyset)^* \cdot L_{0,1}^\emptyset = \varepsilon + a \cdot (a + \varepsilon)^* \cdot b \equiv \varepsilon + aa^*b$$

**Induktionsanfang liefert:**

$$L_{0,1}^\emptyset = b \quad L_{0,0}^\emptyset = a + \varepsilon \quad L_{1,1}^\emptyset = \varepsilon \quad L_{1,0}^\emptyset = a$$

Einsetzen und vereinfachen.

Der zu  $\mathcal{A}$  gehörende reguläre Ausdruck ist:  $a^*b(aa^*b)^*$



# Satz von Kleene

## Zur Größe der konstruierten Ausdrücke/Automaten

Die Konstruktion „Ausdruck → NEA“ erzeugt polynomiell großen NEA  
(im induktiven Schritt wird jeweils nur  $\leq 1$  Zustand hinzugefügt)

„NEA → Ausdruck“ kann aber exponentiell großen Ausdruck liefern

Denn in jedem der  $|Q|$  Induktionsschritte werden 4 per Induktion  
konstruierte Ausdrücke kombiniert:

$$L_{p,q}^X = L_{p,q}^{X \setminus \{\widehat{q}\}} \cup \left( L_{p,\widehat{q}}^{X \setminus \{\widehat{q}\}} \cdot (L_{\widehat{q},\widehat{q}}^{X \setminus \{\widehat{q}\}})^* \cdot L_{\widehat{q},q}^{X \setminus \{\widehat{q}\}} \right)$$

Die Größe des konstruierten Gesamtausdrucks lässt sich daher mit  
 $4^{|Q|}$   
abschätzen.



## Folgerung aus Satz 3.1 und 5.5

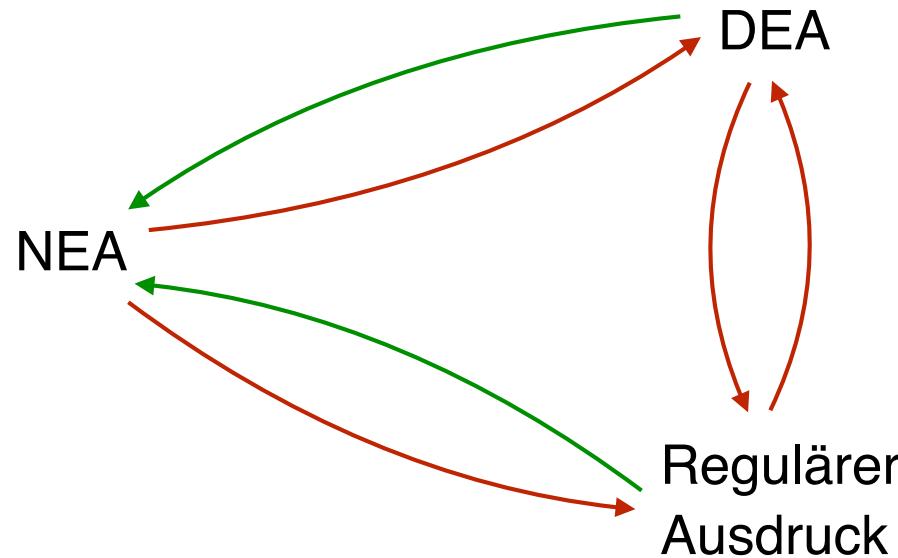
Für alle regulären Ausdrücke  $r$  und  $s$

- gibt es Ausdruck  $t$  mit  $L(t) = L(r) \cap L(s)$ ;
- gibt es Ausdruck  $t'$  mit  $L(t') = \overline{L(r)}$ .

Es ist schwierig, diese Ausdrücke **direkt** aus  $r$  und  $s$  (also ohne den **Umweg über Automaten**) zu konstruieren.

# Relative Größe

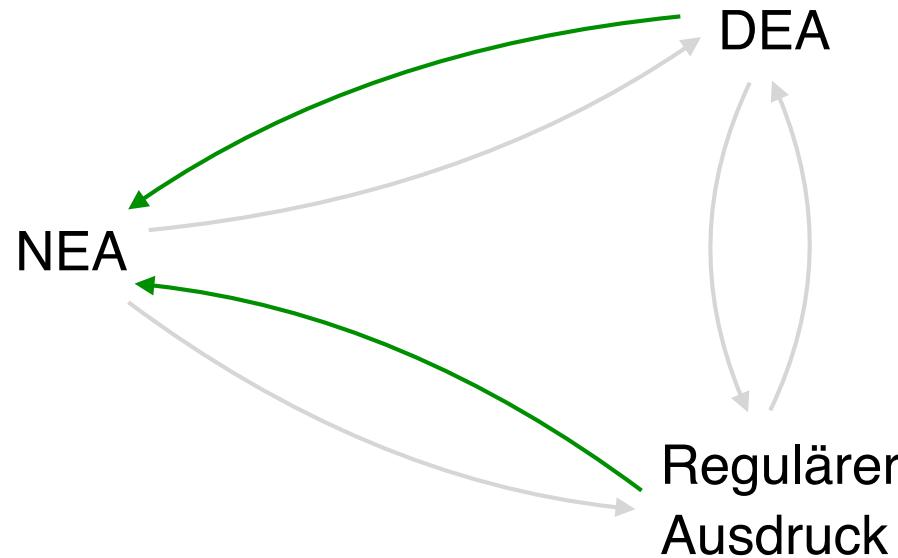
Das folgende Bild fasst die **relative Größe** der drei wichtigsten Repräsentationen von regulären Sprachen zusammen:



Roter Pfeil bedeutet: ein **exponentielles Vergrößern** ist **unvermeidbar**  
NEAs liefern also stets eine **kompakte Repräsentation**  
DEAs sind für manche Sprachen kompakter als reguläre Ausdrücke,  
für andere Sprachen ist es **anders herum**.

# Relative Größe

Das folgende Bild fasst die **relative Größe** der drei wichtigsten Repräsentationen von regulären Sprachen zusammen:

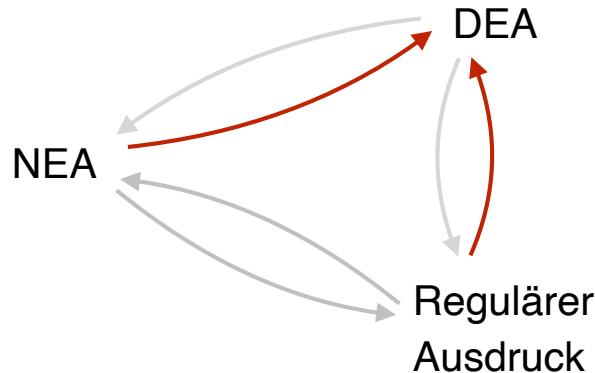


Die “Übersetzung”  $\text{DEA} \Rightarrow \text{NEA}$  ist trivial

Übersetzung regulärer Ausdruck  $\Rightarrow \text{NEA}$  (Satz 5.5)

liefert stets NEA mit  $|Q| \leq 2 * \text{Länge des regulären Ausdrucks}$

# Relative Größe



Wir zeigen: DEAs sind **weniger kompakt** als NEAs und als reguläre Ausdrücke.

Zu diesem Zweck betrachten wir die folgende **Familie von Sprachen**

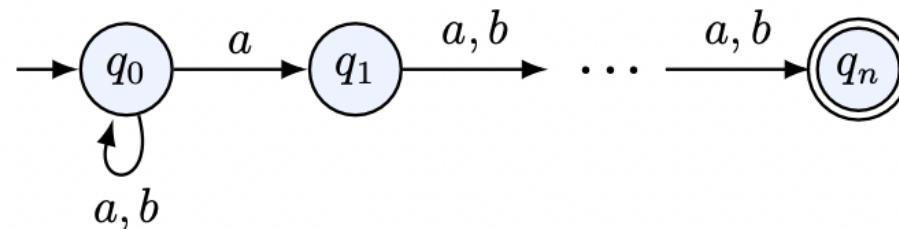
Sei  $\Sigma = \{a, b\}$ . Für alle  $n \geq 1$  sei

$$L_n = \{w \mid \exists u, v \in \Sigma^* : w = uav \text{ und } |v| = n - 1\}.$$

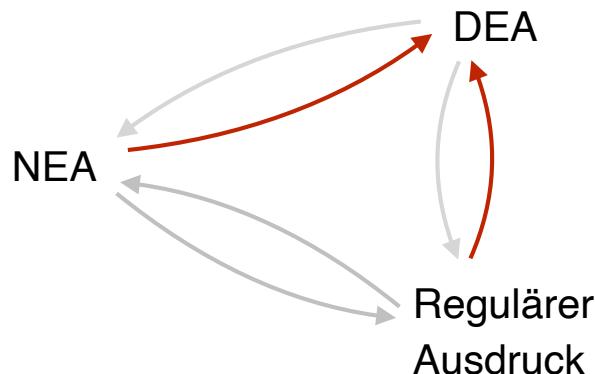
Also: alle Wörter, deren  $n$ -.letztes Symbol ein  $a$  ist.

Leicht zu sehen:

für jedes  $n \geq 1$  gibt es **NEA**  $\mathcal{A}_n$  mit  $n + 1$  Zuständen so dass  $L(\mathcal{A}_n) = L_n$ .



# Relative Größe



Wir zeigen: DEAs sind **weniger kompakt** als NEAs und als reguläre Ausdrücke.

Zu diesem Zweck betrachten wir die folgende **Familie von Sprachen**

Sei  $\Sigma = \{a, b\}$ . Für alle  $n \geq 1$  sei

$$L_n = \{w \mid \exists u, v \in \Sigma^* : w = uav \text{ und } |v| = n - 1\}.$$

Also: alle Wörter, deren  $n$ -.letztes Symbol ein  $a$  ist.

Leicht zu sehen:

für jedes  $n \geq 1$  gibt es regulären Ausdruck  $r_n$  der Länge  $O(n)$  so dass  $L(r_n) = L_n$ .

$$r_n = \underbrace{(a + b)^* a}_{n-1 \text{ mal}} \underbrace{(a + b)(a + b) \cdots (a + b)}_{n-1 \text{ mal}}$$

# Relative Größe

## Lemma 5.7

Für jedes  $n \geq 1$  gilt: jeder DEA  $\mathcal{A}_n$  mit  $L(\mathcal{A}_n) = L_n$  hat mindestens  $2^n$  Zustände.

**Beweis:** Sei  $n \geq 1$ .

Ang., es gäbe DEA  $\mathcal{A}_n = (Q, \Sigma, q_0, \Delta, F)$  mit  $L(\mathcal{A}_n) = L_n$  und  $|Q| < 2^n$ .

Es gibt über  $\Sigma = \{a, b\}$  aber  $2^n$  Wörter der Länge  $n$ .

Also existieren **verschiedene** Wörter  $w, v \in \Sigma^*$  der Länge  $n$  mit

$$\hat{\delta}(q_0, w) = \hat{\delta}(q_0, v).$$

Sei  $w = a_1 \cdots a_n$  und  $v = b_1 \cdots b_n$ . Es gibt  $j$  mit  $a_j \neq b_j$ .

ObDA sei  $a_j = a$  und  $b_j = b$ .

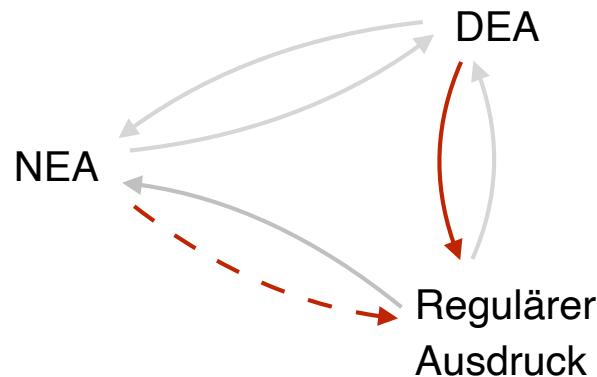
Betrachte  $\widehat{w} = wa^{j-1}$  und  $\widehat{v} = va^{j-1}$ . Es gilt:

1.  $\widehat{w} \in L_n$  und  $\widehat{v} \notin L_n$

2.  $\hat{\delta}(q_0, \widehat{w}) = \hat{\delta}(q_0, \widehat{v})$ , also  $\widehat{w}, \widehat{v} \in L_n$  oder  $L_n \cap \{\widehat{w}, \widehat{v}\} = \emptyset$ .



# Relative Größe

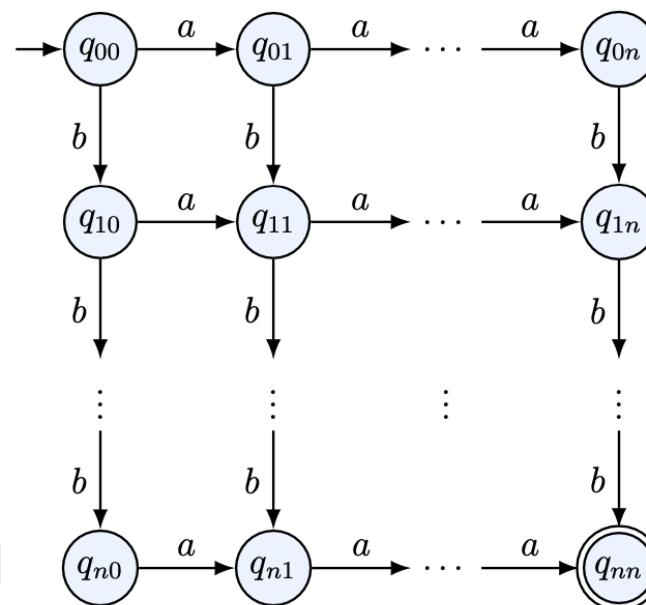


Reguläre Ausdrücke sind **weniger kompakt** als DEAs (und damit auch als NEAs)

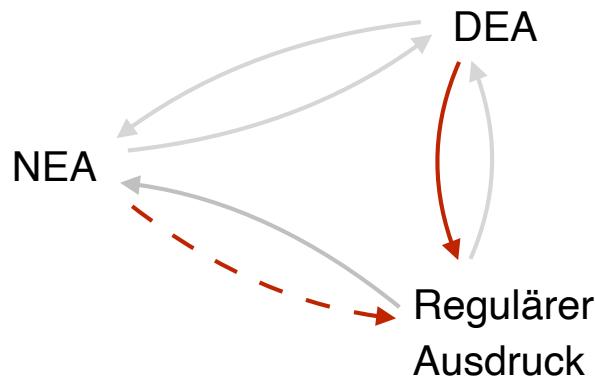
Sei  $\Sigma = \{a, b\}$ . Für alle  $n \geq 1$  sei

$$L_n = \{w \in \Sigma^* : |w|_a = |w|_b = n\}.$$

Für jedes  $n \geq 1$  gibt es **DEA  $\mathcal{A}_n$**  mit  $(n+1)^2$  Zuständen so dass  $L(\mathcal{A}_n) = L_n$ :



# Relative Größe



Reguläre Ausdrücke sind **weniger kompakt** als DEAs (und damit auch als NEAs)

Sei  $\Sigma = \{a, b\}$ . Für alle  $n \geq 1$  sei

$$L_n = \{w \in \Sigma^* : |w|_a = |w|_b = n\}.$$

Jede dieser Sprachen ist endlich, z.B.:

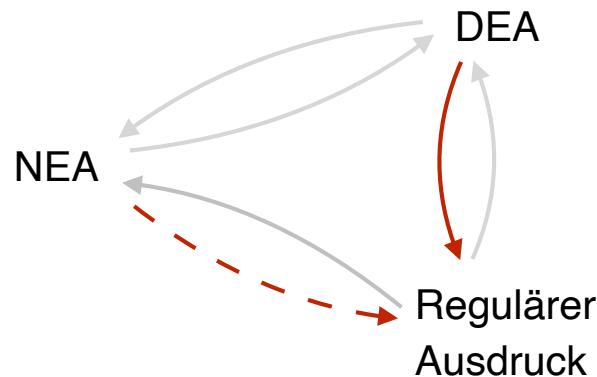
$$L_2 = \{aabb, abab, abba, baab, baba, bbaa\}$$

Man kann sich aber überlegen, dass gilt:  $|L_n| = \frac{n!}{\frac{n}{2}! \cdot \frac{n}{2}!} \geq 2^{\frac{n}{2}}$

Die Größe der Sprachen wächst also **exponentiell mit  $n$**

⇒ regulärer Ausdruck, der einfach „alle Wörter aufzählt“, ist sehr groß

# Relative Größe



Reguläre Ausdrücke sind **weniger kompakt** als DEAs (und damit auch als NEAs)

Sei  $\Sigma = \{a, b\}$ . Für alle  $n \geq 1$  sei

$$L_n = \{w \in \Sigma^* : |w|_a = |w|_b = n\}.$$

Ohne Beweis:

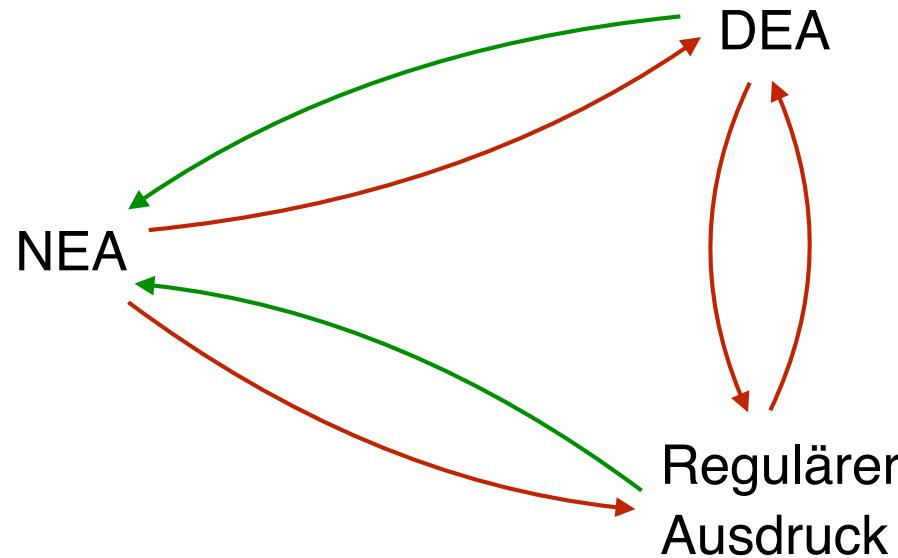
Satz 5.8

Für jedes  $n \geq 1$  gilt: jeder reguläre Ausdruck  $r_n$  mit  $L(r_n) = L_n$  hat mindestens Länge  $2^{\Omega(n)}$ .

wächst nicht wesentlich langsamer als  $2^n$

# Relative Größe

Das folgende Bild fasst die **relative Größe** der drei wichtigsten Repräsentationen von regulären Sprachen zusammen:



Roter Pfeil bedeutet: ein **exponentielles Aufblähen** ist **unvermeidbar**  
NEAs liefern also stets eine **kompakte Repräsentation**  
DEAs sind für manche Sprachen kompakter als reguläre Ausdrücke,  
für andere Sprachen ist es **anders herum**.

# § 6 Minimale DEAs und die Nerode-Rechtskongruenz



## § 6.1 Minimale DEAs



## Ziel:

zu gegebenem **DEA** einen äquivalenten DEA mit **minimaler Zustandszahl** konstruieren

## Zwei Schritte:

1. Eliminieren unerreichbarer Zustände
2. Zusammenfassen äquivalenter Zustände



# Schritt 1

## Schritt 1: Eliminieren unerreichbarer Zustände

Definition 6.1 (Erreichbarkeit eines Zustandes)

Ein Zustand  $q$  des DEA  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  heißt **erreichbar**, falls es ein  $w \in \Sigma^*$  gibt mit  $\hat{\delta}(q_0, w) = q$ .

Sonst heißt  $q$  **unerreichbar**.

Da unerreichbare Zustände für die erkannte Sprache irrelevant sind, erhält man durch **Weglassen** dieser Zustände einen **äquivalenten DEA**  
 $\mathcal{A}_0 := (Q_0, \Sigma, q_0, \delta_0, F_0)$  mit

$$Q_0 := \{q \in Q \mid q \text{ erreichbar}\}$$

$\delta_0$  ist wie  $\delta$ , aber eingeschränkt auf Zustände in  $Q_0$ ,  
geschrieben:  $\delta_0 = \delta|_{Q_0 \times \Sigma}$

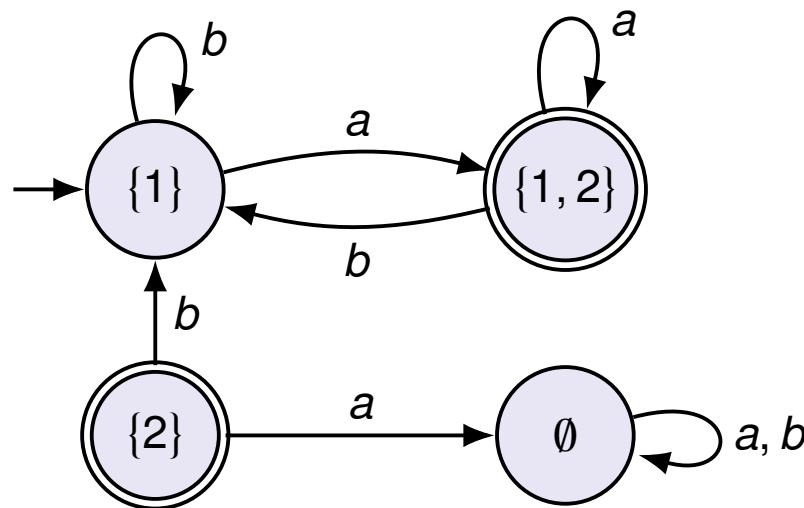
$$F_0 := F \cap Q_0$$



# Schritt 1: Beispiel

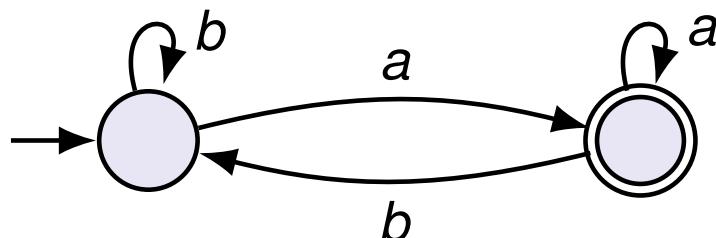
## Beispiel 6.2

Im Automaten  $\mathcal{A}'$  aus Beispiel 1.13



sind die Zustände  $\{2\}$  und  $\emptyset$  nicht erreichbar.

Durch Weglassen dieser Zustände erhält man  $\mathcal{A}'_0$ :



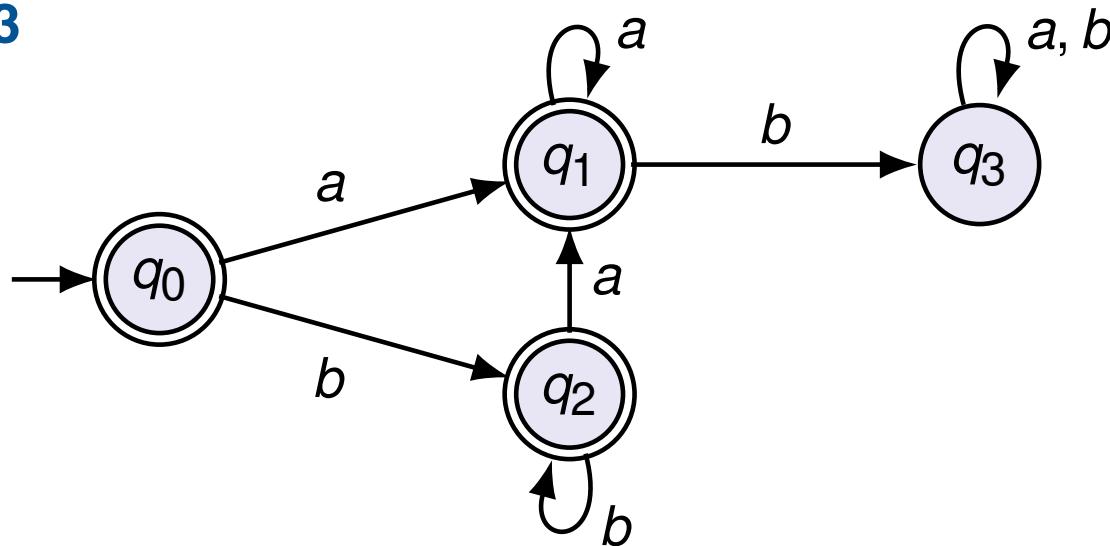
## Schritt 2

### Schritt 2: Zusammenfassen äquivalenter Zustände

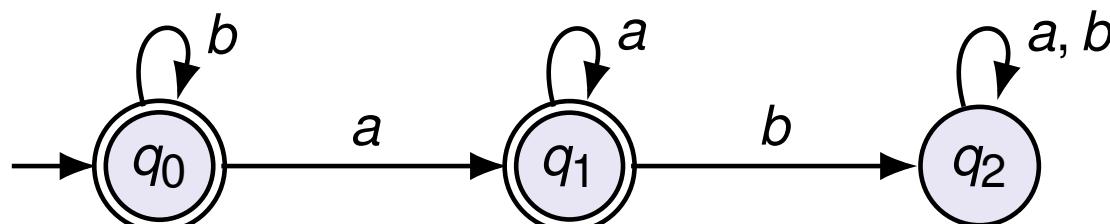
Ein DEA ohne unerreichbare Zustände muss noch nicht minimal sein, da er verschiedene Zustände enthalten kann, die sich „gleich“ verhalten.

#### Beispiel 6.3

Im DEA



sind alle Zustände erreichbar; er ist aber äquivalent zum DFA



Beide erkennen die Sprache  $b^*a^*$



# Einschub: Äquivalenzrelationen

## Definition (Äquivalenzrelation)

Eine Relation  $R \subseteq M \times M$  heißt **Äquivalenzrelation**, wenn  $R$

- **reflexiv** ist:  $(x, x) \in R$  für alle  $x \in M$
- **symmetrisch** ist:  $(x, y) \in R$  impliziert  $(y, x) \in R$
- **transitiv** ist:  $(x, y) \in R$  und  $(y, z) \in R$  impliziert  $(x, z) \in R$

Die **Äquivalenzklasse** von  $x \in M$  bezüglich  $R$  ist:  $[x] := \{y \in M \mid x R y\}$

Man verwendet Äquivalenzrelationen immer dann,  
wenn man die **Austauschbarkeit** von Objekten beschreiben möchte.

Äquivalenzklassen **partitionieren** Grundmenge  $M$  in **disjunkte Teilmengen**.

Jede Äquivalenzklasse ist **Menge** von Objekten, die austauschbar sind.



# Äquivalenz von Zuständen

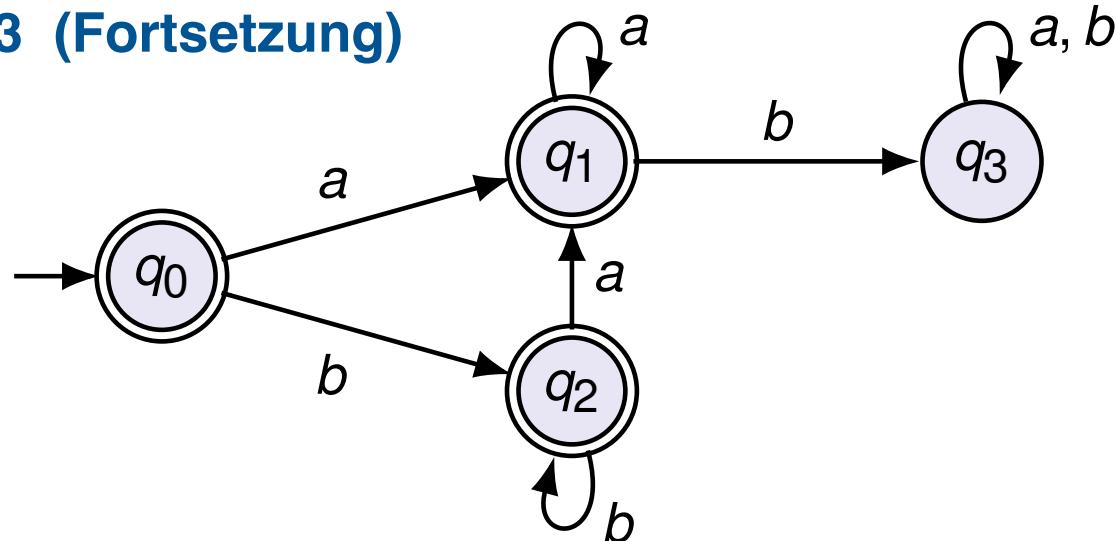
Definition 6.4 (Äquivalenz von Zuständen)

Es sei  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  ein DEA. Für  $q \in Q$  sei

$$\mathcal{A}_q = (Q, \Sigma, q, \delta, F)$$

Zwei Zustände  $q, q' \in Q$  heißen  $\mathcal{A}$ -äquivalent ( $q \sim_{\mathcal{A}} q'$ ) gdw.  
 $L(\mathcal{A}_q) = L(\mathcal{A}_{q'})$ .

## Beispiel 6.3 (Fortsetzung)



- $q_0$  und  $q_2$  sind  $\mathcal{A}$ -äquivalent:  $L(\mathcal{A}_{q_0}) = b^* a^* = L(\mathcal{A}_{q_2})$
- $q_0$  und  $q_1$  sind **nicht**  $\mathcal{A}$ -äquivalent:  $b \in L(\mathcal{A}_{q_0}) \setminus L(\mathcal{A}_{q_1})$

# Äquivalenz von Zuständen

Definition 6.4 (Äquivalenz von Zuständen)

Es sei  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  ein DEA. Für  $q \in Q$  sei

$$\mathcal{A}_q = (Q, \Sigma, q, \delta, F)$$

Zwei Zustände  $q, q' \in Q$  heißen  $\mathcal{A}$ -äquivalent ( $q \sim_{\mathcal{A}} q'$ ) gdw.  
 $L(\mathcal{A}_q) = L(\mathcal{A}_{q'})$ .

Lemma 6.5 (Eigenschaften von  $\sim_{\mathcal{A}}$ )

- (1)  $\sim_{\mathcal{A}}$  ist eine Äquivalenzrelation auf  $Q$   
(also reflexiv, transitiv und symmetrisch).
- (2)  $\sim_{\mathcal{A}}$  ist verträglich mit der Übergangsfunktion, d. h.  
 $q \sim_{\mathcal{A}} q'$  impliziert  $\forall a \in \Sigma : \delta(q, a) \sim_{\mathcal{A}} \delta(q', a)$

(1) folgt aus Def. 6.4, da „=“ Äquivalenzrelation ist.



# Der Quotientenautomat

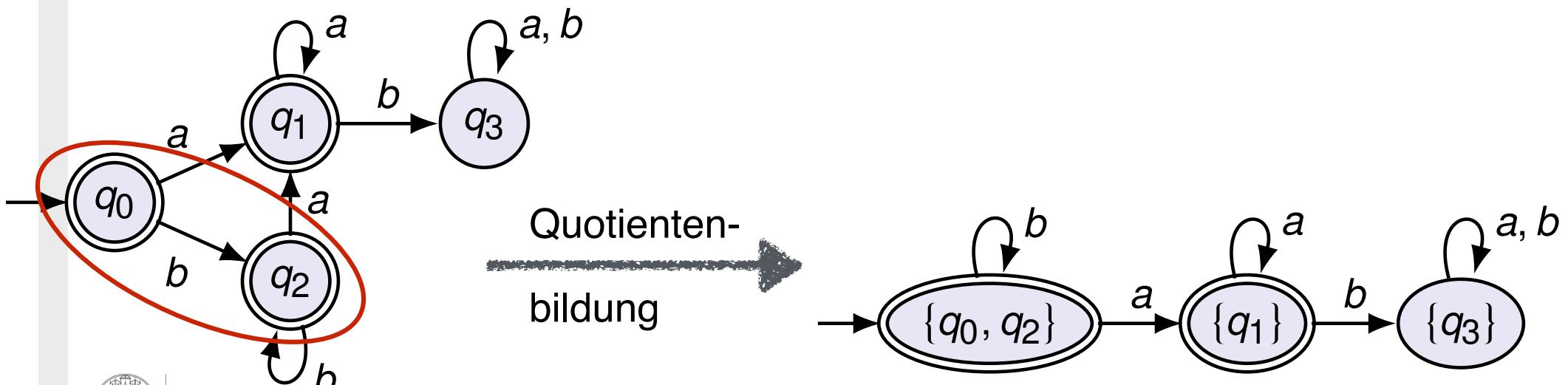
## Definition 6.6 (Quotientenautomat)

Der Quotientenautomat  $\tilde{\mathcal{A}} = (\tilde{Q}, \Sigma, [q_0]_{\mathcal{A}}, \tilde{\delta}, \tilde{F})$  zum DEA

$\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  ist definiert durch:

- $\tilde{Q} := \{[q]_{\mathcal{A}} \mid q \in Q\}$
- $\tilde{\delta}([q]_{\mathcal{A}}, a) := [\delta(q, a)]_{\mathcal{A}}$  (repräsentantenunabhängig nach Lemma 6.5)
- $\tilde{F} := \{[q]_{\mathcal{A}} \mid q \in F\}$  (repräsentantenunabhängig nach Def. von  $\sim_{\mathcal{A}}$ )

## Beispiel 6.3 (Fortsetzung)



# Der Quotientenautomat

Definition Quotientenautomat zur Erinnerung:

$$\widetilde{Q} := \{[q]_{\mathcal{A}} \mid q \in Q\} \quad \widetilde{\delta}([q]_{\mathcal{A}}, a) := [\delta(q, a)]_{\mathcal{A}} \quad \widetilde{F} := \{[q]_{\mathcal{A}} \mid q \in F\}$$

Lemma 6.7

1. Für alle  $[q]_{\mathcal{A}} \in \widetilde{Q}$  und alle  $w \in \Sigma^*$  gilt:  $\hat{\delta}([q]_{\mathcal{A}}, w) = [\hat{\delta}(q, w)]_{\mathcal{A}}$
2.  $\widetilde{\mathcal{A}}$  ist äquivalent zu  $\mathcal{A}$ .

**Beweis:**

Punkt 1 sagt intuitiv:

Die kanonische Fortsetzung  $\hat{\delta}$  von  $\tilde{\delta}$  verhält sich entsprechend der Def. von  $\tilde{\delta}$ .

Man beweist dies leicht per Induktion über  $|w|$  (Übung)



# Der Quotientenautomat

Definition Quotientenautomat zur Erinnerung:

$$\widetilde{Q} := \{[q]_{\mathcal{A}} \mid q \in Q\} \quad \widetilde{\delta}([q]_{\mathcal{A}}, a) := [\delta(q, a)]_{\mathcal{A}} \quad \widetilde{F} := \{[q]_{\mathcal{A}} \mid q \in F\}$$

Lemma 6.7

1. Für alle  $[q]_{\mathcal{A}} \in \widetilde{Q}$  und alle  $w \in \Sigma^*$  gilt:  $\widehat{\delta}([q]_{\mathcal{A}}, w) = [\widehat{\delta}(q, w)]_{\mathcal{A}}$
2.  $\widetilde{\mathcal{A}}$  ist äquivalent zu  $\mathcal{A}$ .

**Beweis:**

Punkt 2:  $w \in L(\mathcal{A})$  gdw.  $\widehat{\delta}(q_0, w) \in F$   
gdw.  $[\widehat{\delta}(q_0, w)]_{\mathcal{A}} \in \widetilde{F}$  (Def.  $\widetilde{F}$ )  
gdw.  $\widehat{\delta}([q_0]_{\mathcal{A}}, w) \in \widetilde{F}$  (Punkt 1)  
gdw.  $w \in L(\widetilde{\mathcal{A}})$



# Berechnung von $\sim_{\mathcal{A}}$

**Wir zeigen nun:**

Die Äquivalenzrelation  $\sim_{\mathcal{A}}$  kann in Polynomialzeit berechnet werden.

Im Prinzip ist das klar:

Wir müssen entscheiden ob  $L(\mathcal{A}_q) = L(\mathcal{A}_{q'})$  und das Äquivalenzproblem für DEAs ist in Polynomialzeit lösbar

Im Folgenden ein direkterer Ansatz

Wir definieren (Über)-Approximationen  $\sim_0, \sim_1, \sim_2, \dots$  von  $\sim_{\mathcal{A}}$  so dass:

- jedes  $\sim_k$  ist eine Äquivalenzrelation
- $\sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \dots$

d.h.: die Äquivalenzklassen von  $\sim_k$  verfeinern sich mit wachsendem  $k$

- nach polynomiell vielen Schritten ist  $\sim_k = \sim_{k+1} = \sim_{\mathcal{A}}$

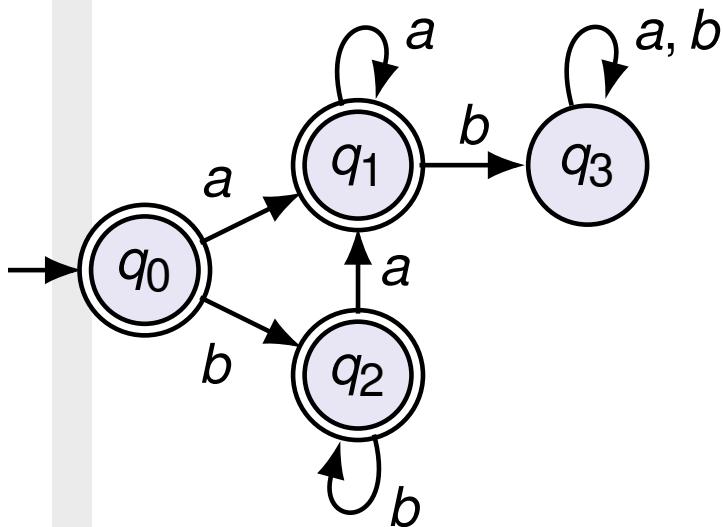


# Berechnung von $\sim_{\mathcal{A}}$

Definition von  $\sim_0, \sim_1, \sim_2, \dots$  von  $\sim_{\mathcal{A}}$ :

- $q \sim_0 q'$  gdw.  $q, q' \in F$  oder  $q, q' \notin F$
- $q \sim_{k+1} q'$  gdw.  $q \sim_k q'$  und  $\forall a \in \Sigma : \delta(q, a) \sim_k \delta(q', a)$

## Beispiel 6.3 (Fortsetzung)



- $\sim_0$  hat die Klassen  $F = \{q_0, q_1, q_2\}$  und  $Q \setminus F = \{q_3\}$
- $\sim_1$  hat die Klassen  $\{q_1\}, \{q_0, q_2\}, \{q_3\}$ : z. B. ist  $\delta(q_0, b) = \delta(q_2, b) \in F$  und  $\delta(q_1, b) \notin F$
- $\sim_2 = \sim_1 = \sim_{\mathcal{A}}$

# Berechnung von $\sim_{\mathcal{A}}$

Definition von  $\sim_0, \sim_1, \sim_2, \dots$  von  $\sim_{\mathcal{A}}$ :

- $q \sim_0 q'$  gdw.  $q, q' \in F$  oder  $q, q' \notin F$
- $q \sim_{k+1} q'$  gdw.  $q \sim_k q'$  und  $\forall a \in \Sigma : \delta(q, a) \sim_k \delta(q', a)$

Wegen  $Q \times Q \supseteq \sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \dots$  und Endlichkeit von  $Q$  gibt es  $k \geq 0$  mit  $\sim_k = \sim_{k+1}$ . Genauer:  $k \leq |Q|^2$ .

Die Relation  $\sim_k$  lässt sich also in polynomieller Zeit berechnen:

- maximal  $|Q|^2$  viele Schritte
- in jedem Schritt maximal  $|Q|^2 \cdot |\Sigma|$  viele Tests

**Zu zeigen bleibt:**

Wenn  $\sim_k = \sim_{k+1}$ , dann  $\sim_k = \sim_{\mathcal{A}}$ .



# Berechnung von $\sim_{\mathcal{A}}$

Definition von  $\sim_0, \sim_1, \sim_2, \dots$  von  $\sim_{\mathcal{A}}$ :

- $q \sim_0 q'$  gdw.  $q, q' \in F$  oder  $q, q' \notin F$
- $q \sim_{k+1} q'$  gdw.  $q \sim_k q'$  und  $\forall a \in \Sigma : \delta(q, a) \sim_k \delta(q', a)$

Wir zeigen zunächst:

Behauptung

Für alle  $k \geq 0$  gilt:

$q \sim_k q'$  gdw.  $\forall w \in \Sigma^* \text{ mit } |w| \leq k : w \in L(\mathcal{A}_q) \Leftrightarrow w \in L(\mathcal{A}_{q'})$

Wenn  $\sim_k = \sim_{k+1}$ , dann  $\sim_k = \sim_{\mathcal{A}}$ .



# Der reduzierte Automat

## Zurück zur Minimierung von DEAs

Definition 6.8 (reduzierter Automat zu einem DEA)

Für einen DEA  $\mathcal{A}$  bezeichnet  $\mathcal{A}_{\text{red}}$  den **reduzierten Automaten**, den man aus  $\mathcal{A}$  durch Eliminieren unerreichbarer Zustände und Bilden des Quotientenautomaten erhält.

**Wir haben gerade gesehen:**

Der Automat  $\mathcal{A}_{\text{red}}$  kann in **polynomieller Zeit** aus  $\mathcal{A}$  konstruiert werden

**Zentrale Eigenschaft des reduzierten Automaten**

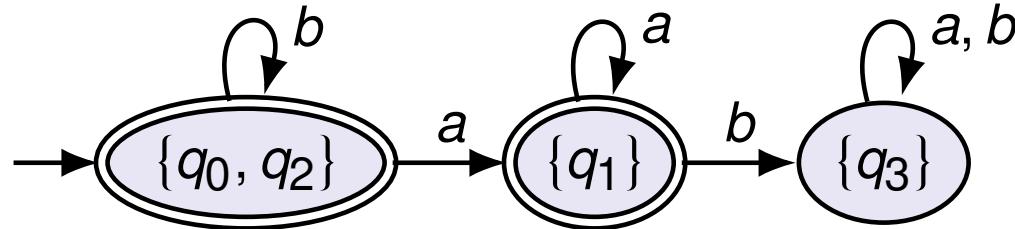
$\mathcal{A}_{\text{red}}$  kann nicht weiter vereinfacht werden,  
d. h. er hat **minimal mögliche Zustandszahl** für  $L(\mathcal{A})$ .

Dies zeigen wir (unter anderem) im nächsten Abschnitt der Vorlesung.



# Minimierung am Beispiel

In unserem Beispiel erhalten wir für  $L = b^* a^*$  den reduzierten DEA



Wir zeigen zur Illustration nun noch:

es gibt keinen DEA  $\mathcal{A}$  mit  $L(\mathcal{A}) = b^* a^*$  mit weniger als drei Zuständen.

Angenommen, es gäbe doch einen solchen DEA  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ .

Betrachte die Wörter  $w_1 = \varepsilon$ ,  $w_2 = a$  und  $w_3 = ab$ .

Da  $|Q| < 3$  gibt es  $i, j$  mit  $i \neq j$  und  $\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$ .

Wir unterscheiden drei Fälle:

1.  $\hat{\delta}(q_0, \varepsilon) = \hat{\delta}(q_0, ab)$

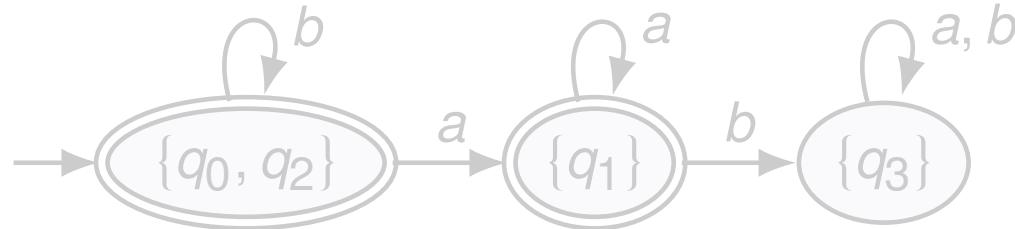
Also ( $\varepsilon \in L(\mathcal{A})$  und  $ab \in L(\mathcal{A})$ ) oder ( $\varepsilon \notin L(\mathcal{A})$  und  $ab \notin L(\mathcal{A})$ ).

Es gilt aber  $\varepsilon \in b^* a^*$  und  $ab \notin b^* a^*$ .



# Minimierung am Beispiel

In unserem Beispiel erhalten wir für  $L = b^* a^*$  den reduzierten DEA



Wir zeigen zur Illustration nun noch:

es gibt keinen DFA  $\mathcal{A}$  mit  $L(\mathcal{A}) = b^* a^*$  mit weniger als drei Zuständen.

Angenommen, es gäbe doch einen solchen DFA  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ .

Betrachte die Wörter  $w_1 = \varepsilon$ ,  $w_2 = a$  und  $w_3 = ab$ .

Da  $|Q| < 3$  gibt es  $i, j$  mit  $i \neq j$  und  $\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$ .

Wir unterscheiden drei Fälle:

2.  $\hat{\delta}(q_0, a) = \hat{\delta}(q_0, ab)$

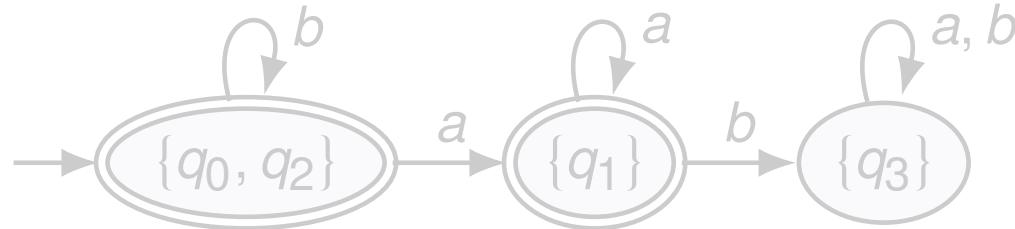
Also ( $a \in L(\mathcal{A})$  und  $ab \in L(\mathcal{A})$ ) oder ( $a \notin L(\mathcal{A})$  und  $ab \notin L(\mathcal{A})$ ).

Es gilt aber  $a \in b^* a^*$  und  $ab \notin b^* a^*$ .



# Minimierung am Beispiel

In unserem Beispiel erhalten wir für  $L = b^* a^*$  den reduzierten DEA



Wir zeigen zur Illustration nun noch:

es gibt keinen DFA  $\mathcal{A}$  mit  $L(\mathcal{A}) = b^* a^*$  mit weniger als drei Zuständen.

Angenommen, es gäbe doch einen solchen DFA  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ .

Betrachte die Wörter  $w_1 = \varepsilon$ ,  $w_2 = a$  und  $w_3 = ab$ .

Da  $|Q| < 3$  gibt es  $i, j$  mit  $i \neq j$  und  $\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$ .

Wir unterscheiden drei Fälle:

3.  $\hat{\delta}(q_0, \varepsilon) = \hat{\delta}(q_0, a)$

Dann auch  $\hat{\delta}(q_0, \varepsilon \cdot b) = \hat{\delta}(q_0, a \cdot b)$

Also ( $b \in L(\mathcal{A})$  und  $ab \in L(\mathcal{A})$ ) oder ( $b \notin L(\mathcal{A})$  und  $ab \notin L(\mathcal{A})$ ).

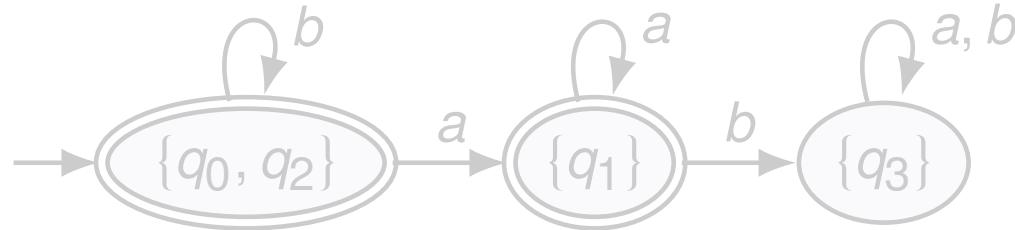
Es gilt aber  $b \in b^* a^*$  und  $ab \notin b^* a^*$ .



□

# Minimierung am Beispiel

In unserem Beispiel erhalten wir für  $L = b^* a^*$  den reduzierten DEA



Wir zeigen zur Illustration nun noch:

es gibt keinen DFA  $\mathcal{A}$  mit  $L(\mathcal{A}) = b^* a^*$  mit weniger als drei Zuständen.

Angenommen, es gäbe doch einen solchen DFA  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ .

Betrachte die Wörter  $w_1 = \varepsilon$ ,  $w_2 = a$  und  $w_3 = ab$ .

Da  $|Q| < 3$  gibt es  $i, j$  mit  $i \neq j$  und  $\hat{\delta}(q_0, w_i) = \hat{\delta}(q_0, w_j)$ .

Die im nächsten Abschnitt eingeführte **Nerode-Rechtskongruenz** erlaubt es uns, diese Art von Argumentation auf beliebige Sprachen zu generalisieren.

## Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz



## Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

7. Die Chomsky-Hierarchie
8. Rechtslineare Grammatiken und reguläre Sprachen
9. Normalformen und Entscheidungsprobleme
10. Abschlusseigenschaften und Pumping-Lemma
11. Kellerautomaten
12. Die Struktur kontextfreier Sprachen

## § 6.2 Die Nerode-Rechtskongruenz



# Zur Erinnerung: der reduzierte Automat

Definition 6.8 (reduzierter Automat zu einem DEA)

Für einen DEA  $\mathcal{A}$  bezeichnet  $\mathcal{A}_{\text{red}}$  den reduzierten Automaten, den man aus  $\mathcal{A}$  durch Eliminieren unerreichbarer Zustände und Bilden des Quotientenautomaten erhält.

**Wollen zeigen:**

$\mathcal{A}_{\text{red}}$  ist der **kleinste** DEA, der  $L(\mathcal{A})$  erkennt (wenigste Zustände).

Er ist sogar der **einige** solche DEA (bis auf Zustandsumbenennung)

Wir verwenden dazu die **Nerode-Rechtskongruenz**.

Sie ist von **unabhängigem Interesse**, liefert auch einen Weg, um

- ★ die regulären Sprachen zu **charakterisieren** und
- ★ von Sprachen zeigen, dass sie **nicht regulär** sind.



# Die Nerode-Rechtskongruenz

Definition 6.9 (Nerode-Rechtskongruenz)

Sei  $L \subseteq \Sigma^*$  eine beliebige formale Sprache. Für  $u, v \in \Sigma^*$  ist

$$u \simeq_L v \quad \text{gdw. } \forall w \in \Sigma^* : uw \in L \Leftrightarrow vw \in L$$

## Beispiel 6.10

$$L = b^* a^*$$

Es gilt:

- $\varepsilon \simeq_L b$ :  
     $\forall w : \varepsilon w \in L \quad \text{gdw. } w \in L$   
         $\text{gdw. } w \in b^* a^*$   
         $\text{gdw. } bw \in b^* a^*$   
         $\text{gdw. } bw \in L$
- $\varepsilon \not\simeq_L a$ :  
     $\varepsilon \cdot b = b \in b^* a^*$ ,    aber     $a \cdot b = ab \notin b^* a^*$

Anmerkung 6.10b    Wenn  $u \simeq_L v$ , dann  $u \in L \Leftrightarrow v \in L$



# Wichtige Eigenschaften der Nerode-Rechtskongruenz

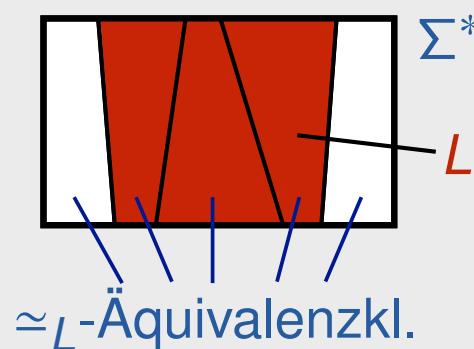
Lemma 6.11 (Eigenschaften von  $\simeq_L$ )

- (1)  $\simeq_L$  ist eine Äquivalenzrelation auf  $\Sigma^*$ .
- (2)  $\simeq_L$  ist Rechtskongruenz, d. h. zusätzlich zu (1) gilt:

$$u \simeq_L v \Rightarrow \forall w \in \Sigma^* : uw \simeq_L vw$$

- (3)  $L$  ist Vereinigung von  $\simeq_L$ -Äquivalenzklassen:

$$L = \bigcup_{u \in L} [u]_L$$



- (4) Ist  $L = L(\mathcal{A})$  für einen DEA  $\mathcal{A}$ , so ist die Zustandszahl größer oder gleich dem Index von  $\simeq_L$  (Anzahl der  $\simeq_L$ -Klassen).

(1) folgt aus Def. 6.9, da „ $\Leftrightarrow$ “ reflexiv, symmetrisch, transitiv

# Wichtige Eigenschaften der Nerode-Rechtskongruenz

## Beispiel 6.10 (Fortsetzung)

$$L = b^* a^*$$

$\simeq_L$  hat drei Klassen:

- $[\varepsilon]_L = \{v \in \Sigma^* \mid vw \in L \text{ gdw. } w \in b^* a^*\} = b^*$
- $[a]_L = \{v \in \Sigma^* \mid vw \in L \text{ gdw. } w \in a^*\} = b^* aa^*$
- $[ab]_L = \{v \in \Sigma^* \mid vw \in L \text{ für kein } w \in \Sigma^*\} = (a + b)^* ab(a + b)^*$

Es ist  $L = [\varepsilon]_L \cup [a]_L$  (vgl. Lemma 6.11 (3))  
und  $\Sigma^* \setminus L = [ab]_L$ .



# Der kanonische DEA

Wenn  $\simeq_L$  nur endlich viele Äquivalenzklassen hat, können wir diese als Zustände eines kanonischen Automaten für  $L$  verwenden.

**Definition 6.12 (Kanonischer DEA  $\mathcal{A}_L$ )**

Sei  $L \subseteq \Sigma^*$  mit  $\simeq_L$  von **endlichem Index**.

Dann ist der **kanonische DEA**  $\mathcal{A}_L := (Q', \Sigma, q'_0, \delta', F')$  definiert durch:

- $Q' := \{[u]_L \mid u \in \Sigma^*\}$
- $q'_0 := [\varepsilon]_L$
- $\delta'([u]_L, a) := [ua]_L$  (repräsentantenunabhängig nach Lem. 6.11 (2))
- $F' := \{[u]_L \mid u \in L\}$  (repräsentantenunabhängig nach Anm. 6.10 b)

Wenn  $u \simeq_L v$ , dann  $u \in L \Leftrightarrow v \in L$

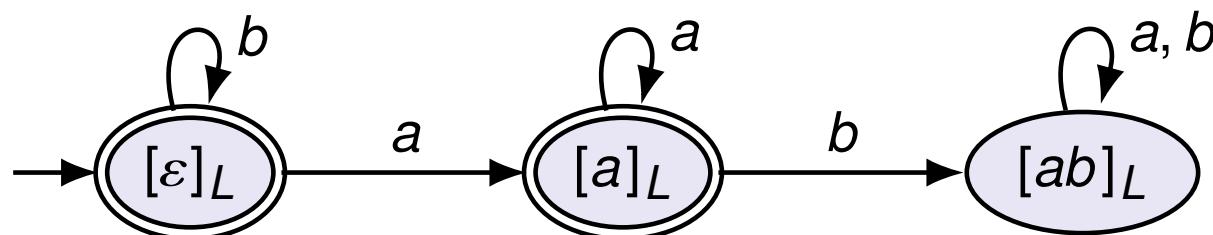


## Beispiel 6.10 (Fortsetzung)

$$L = b^* a^*$$

$\approx_L$  hat drei Klassen:

- $[\varepsilon]_L = b^*$
- $[a]_L = b^* aa^*$
- $[ab]_L = (a + b)^* ab(a + b)^*$



# Der kanonische DEA

Definition 6.12 (Kanonischer DEA  $\mathcal{A}_L$ )

Sei  $L \subseteq \Sigma^*$  mit  $\simeq_L$  von **endlichem Index**.

Dann ist der **kanonische DEA**  $\mathcal{A}_L := (Q', \Sigma, q'_0, \delta', F')$  definiert durch:

- $Q' := \{[u]_L \mid u \in \Sigma^*\}$
- $q'_0 := [\varepsilon]_L$
- $\delta'([u]_L, a) := [ua]_L$
- $F' := \{[u]_L \mid u \in L\}$

Lemma 6.13

Hat  $\simeq_L$  **endlichen Index**, so ist  $\mathcal{A}_L$  ein DEA mit  $L = L(\mathcal{A}_L)$ .

Nach Lemma 6.11 (4) hat  $\mathcal{A}_L$  minimal mögliche Zustandszahl!



# Anwendungen der Nerode-Rechtskongruenz

**Anwendung 1: Charakterisierung der regulären Sprachen,**  
die unabhängig ist von Automaten!

Satz 6.14 (Satz von Myhill und Nerode)

Eine formale Sprache  $L$  ist erkennbar  $\text{gdw. } \simeq_L$  endlichen Index hat.

## Beweis

„ $\Rightarrow$ “ Wenn  $L$  erkennbar, dann gibt es DEA  $\mathcal{A}$  mit  $L(\mathcal{A}) = L$ .

Zustandszahl ist endlich und mit Lemma 6.11 (4) größer oder gleich dem Index von  $\simeq_L$ , also ist auch der endlich.

„ $\Leftarrow$ “ Kanonischer DEA  $\mathcal{A}_L$  erkennt  $L$  (Lemma 6.13).



## Anwendung 2: Nicht-Regularität einer Sprache nachweisen

### Beispiel 6.15

Wir betrachten die Sprache  $L := \{a^n b^n \mid n \geq 0\}$ .

Für  $n \neq m$  gilt:  $a^n \not\sim_L a^m$  denn  $a^n b^n \in L$ , aber  $a^m b^n \notin L$ .

Die  $\simeq_L$ -Klassen  $[a^0]_L, [a^1]_L, [a^2]_L, \dots$  sind also alle verschieden.

Also hat  $\simeq_L$  unendlichen Index und  $L$  ist nicht regulär.

Dieses Verfahren ist also eine Alternative zur Anwendung des Pumping-Lemmas.



# Anwendungen der Nerode-Rechtskongruenz

## Anwendung 3: Nachweis, dass der reduzierte Automat minimal ist

Satz 6.16 (Minimalität des reduzierten Automaten)

Sei  $\mathcal{A}$  ein DEA. Dann hat jeder DEA, der  $L(\mathcal{A})$  erkennt, mindestens so viele Zustände wie der reduzierte Automat  $\mathcal{A}_{\text{red}}$ .

### Beweis

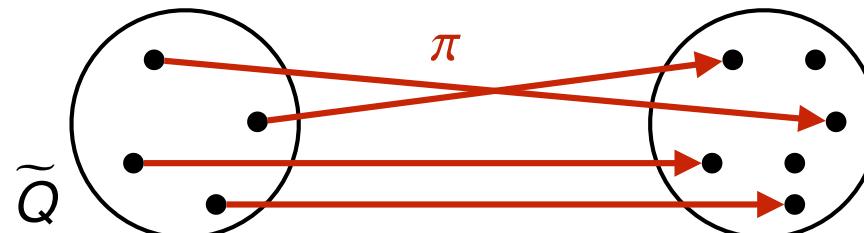
Sei  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  und  $\mathcal{A}_{\text{red}} = (\tilde{Q}, \Sigma, [q_0]_{\mathcal{A}}, \tilde{\delta}, \tilde{F})$ .

Es genügt zu zeigen:  $|\tilde{Q}| \leq \text{Index von } \simeq_L$

Die Behauptung folgt dann aus Lem. 6.11 (4).

Dafür definieren wir injektive Abbildung  $\pi$ , die jedem  $[q]_{\mathcal{A}} \in \tilde{Q}$  eine Äquivalenzklasse von  $\simeq_L$  zuordnet:

### Illustration:



# Anwendungen der Nerode-Rechtskongruenz

## Anwendung 3: Nachweis, dass der reduzierte Automat minimal ist

Satz 6.16 (Minimalität des reduzierten Automaten)

Sei  $\mathcal{A}$  ein DEA. Dann hat jeder DEA, der  $L(\mathcal{A})$  erkennt, mindestens so viele Zustände wie der reduzierte Automat  $\mathcal{A}_{\text{red}}$ .

### Beweis

Sei  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  und  $\mathcal{A}_{\text{red}} = (\tilde{Q}, \Sigma, [q_0]_{\mathcal{A}}, \tilde{\delta}, \tilde{F})$ .

Gesucht: injektive Abbildung  $\pi$  von  $\tilde{Q}$  in die Menge der Äquivalenzkl. von  $\simeq_L$

Da per Konstruktion alle Zustände in  $\mathcal{A}_{\text{red}}$  erreichbar sind, gibt es für jedes  $[q]_{\mathcal{A}} \in \tilde{Q}$  (mindestens) ein  $w_q \in \Sigma^*$  mit

$$\hat{\delta}([q_0]_{\mathcal{A}}, w_q) = [q]_{\mathcal{A}}$$

Definiere  $\pi([q]_{\mathcal{A}}) := [w_q]_L$ .

**Es bleibt zu zeigen:**  $\pi$  ist injektiv.



# Anwendungen der Nerode-Rechtskongruenz

## Anwendung 3: Nachweis, dass der reduzierte Automat minimal ist

Satz 6.16 (Minimalität des reduzierten Automaten)

Sei  $\mathcal{A}$  ein DEA. Dann hat jeder DEA, der  $L(\mathcal{A})$  erkennt, mindestens so viele Zustände wie der reduzierte Automat  $\mathcal{A}_{\text{red}}$ .

Es sind also **sowohl** der kanonische Automat  
**als auch** der reduzierte Automat minimal!

**Wir werden jetzt sogar zeigen:**  
diese beiden Automaten sind **identisch bis auf Zustandsumbenennung**.



# Isomorphie

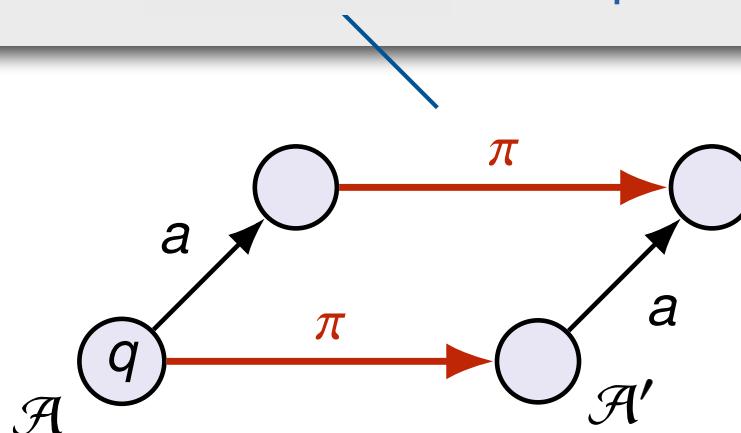
**Formale Definition von „identisch bis auf Zustandsumbenennung“:**

Satz 6.17 (Isomorphie von DEAs)

Zwei DEAs  $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$  und  $\mathcal{A}' = (Q', \Sigma, q'_0, \delta', F')$  heißen isomorph ( $\mathcal{A} \simeq \mathcal{A}'$ ), wenn es eine **Bijektion**  $\pi : Q \rightarrow Q'$  gibt mit:

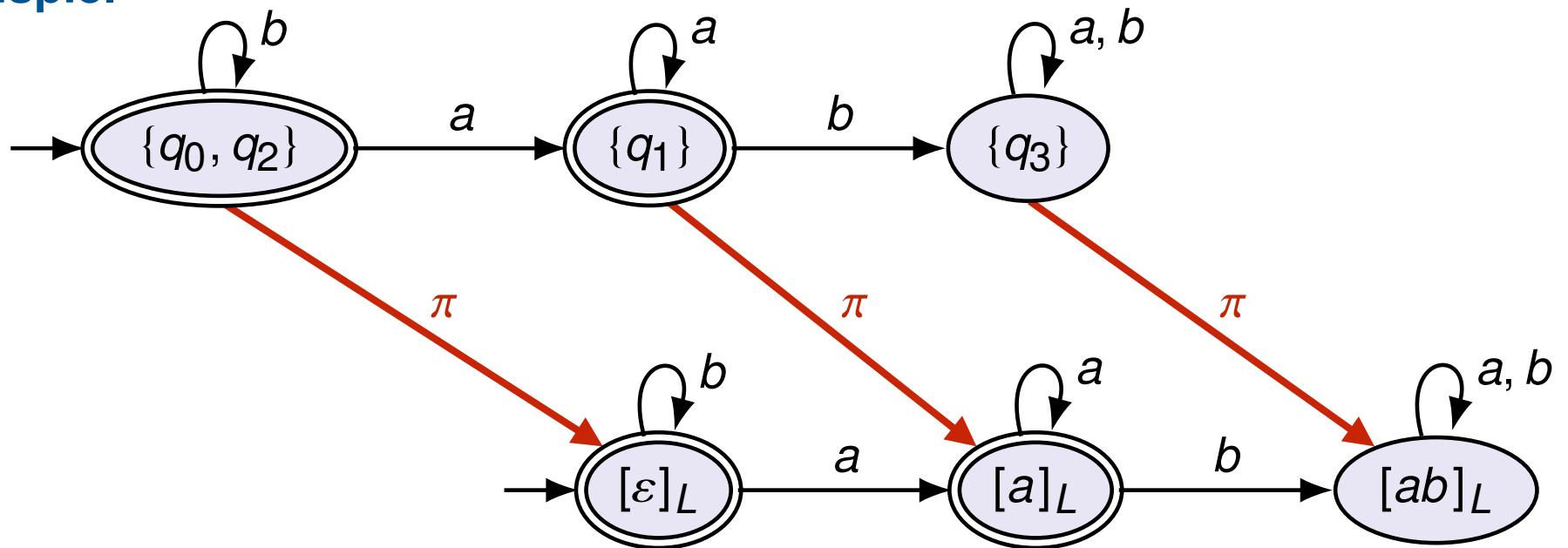
- $\pi(q_0) = q'_0$
- $q \in F$  gdw.  $\pi(q) \in F'$  für alle  $q \in Q$
- $\pi(\delta(q, a)) = \delta'(\pi(q), a)$  für alle  $q \in Q, a \in \Sigma$

Wir nennen  $\pi$  dann einen **Isomorphismus** von  $\mathcal{A}$  nach  $\mathcal{A}'$ .



# Isomorphie

## Beispiel



Lemma 6.18 (Isomorphie erhält die Sprache)

Wenn  $\mathcal{A} \simeq \mathcal{A}'$ , dann  $L(\mathcal{A}) = L(\mathcal{A}')$ .

**Beweis:** siehe Skript

# Kanonischer & reduzierter Automat sind isomorph

Satz 6.19 (Isomorphie kanonischer und reduzierter Automat)

Sei  $L$  eine erkennbare Sprache und  $\mathcal{A}$  ein DEA mit  $L(\mathcal{A}) = L$ .

Dann ist der reduzierte Automat  $\mathcal{A}_{\text{red}}$  **isomorph** zum kanonischen Automaten  $\mathcal{A}_L$ .

## Beweis

Sei  $\mathcal{A}_{\text{red}} = (\tilde{Q}, \Sigma, [q_0]_{\mathcal{A}}, \tilde{\delta}, \tilde{F})$  und  $\mathcal{A}_L = (Q', \Sigma, q'_0, \delta', F')$ .

Im Beweis von Lemma 6.16 haben wir injektive Abbildung  $\pi$  definiert, die jedem Zustand in  $\tilde{Q}$  eine Äquivalenzklasse von  $\simeq_L$  zuordnet:

Da per Konstruktion alle Zustände in  $\mathcal{A}_{\text{red}}$  erreichbar sind, gibt es **für jedes**  $[q]_{\mathcal{A}} \in \tilde{Q}$  (mindestens) ein  $w_q \in \Sigma^*$  mit

$$\hat{\delta}([q_0]_{\mathcal{A}}, w_q) = [q]_{\mathcal{A}}$$

Definiere  $\pi([q]_{\mathcal{A}}) := [w_q]_L$ .



# Kanonischer & reduzierter Automat sind isomorph

Satz 6.19 (Isomorphie kanonischer und reduzierter Automat)

Sei  $L$  eine erkennbare Sprache und  $\mathcal{A}$  ein DEA mit  $L(\mathcal{A}) = L$ .

Dann ist der reduzierte Automat  $\mathcal{A}_{\text{red}}$  **isomorph** zum kanonischen Automaten  $\mathcal{A}_L$ .

## Beweis

Sei  $\mathcal{A}_{\text{red}} = (\widetilde{Q}, \Sigma, [q_0]_{\mathcal{A}}, \widetilde{\delta}, \widetilde{F})$  und  $\mathcal{A}_L = (Q', \Sigma, q'_0, \delta', F')$ .

Im Beweis von Lemma 6.16 haben wir injektive Abbildung  $\pi$  definiert, die jedem Zustand in  $\widetilde{Q}$  eine Äquivalenzklasse von  $\simeq_L$  zuordnet.

Man kann zeigen, dass  $\pi$  Isomorphismus von  $\mathcal{A}_{\text{red}}$  nach  $\mathcal{A}_L$  ist

Z.B. ist  $\pi$  auch surjektiv:

weil  $\pi$  injektiv und  $|Q'| \leq |\widetilde{Q}|$  (da  $\mathcal{A}_L$  minimale Zustandszahl hat)

Weitere Details finden sich im Skript



# Weitere Konsequenzen der Isomorphie

Wir haben mit Satz 6.19 auch gezeigt:

- Der reduzierte Automat  $\mathcal{A}_{\text{red}}$  ist **unabhängig** vom DEA  $\mathcal{A}$ , mit dem man beginnt:

Wenn  $L(\mathcal{A}) = L(\mathcal{B})$ , dann  $\mathcal{A}_{\text{red}} \simeq \mathcal{B}_{\text{red}}$ .

Denn  $\mathcal{A}_{\text{red}}$  und  $\mathcal{B}_{\text{red}}$  sind **beide** isomorph zum kanonischen DEA  $\mathcal{A}_L$  (Satz 6.19), also sind auch  $\mathcal{A}_{\text{red}}$  und  $\mathcal{B}_{\text{red}}$  isomorph

- Für jede erkennbare Sprache gibt es einen **eindeutigen minimalen DEA**:

Wenn  $L(\mathcal{A}) = L(\mathcal{B}) = L$  und  $\mathcal{A}$  und  $\mathcal{B}$  minimale Zustandszahl unter allen  $L$  erkennenden DEAs haben, dann  $\mathcal{A} \simeq \mathcal{B}$ .

Denn:  $\mathcal{A} = \mathcal{A}_{\text{red}} \simeq \mathcal{A}_L \simeq \mathcal{B}_{\text{red}} = \mathcal{B}$

(Satz 6.19)  
( $\mathcal{A}, \mathcal{B}$  minimal)



## Korollar 6.20

Es seien  $\mathcal{A}$  und  $\mathcal{A}'$  DEAs. Dann gilt:

$$L(\mathcal{A}) = L(\mathcal{A}') \text{ gdw. } \mathcal{A}_{\text{red}} \simeq \mathcal{A}'_{\text{red}}$$

Man erhält eine alternative Methode, um das **Äquivalenzproblem für DEAs** zu entscheiden:

- Konstruiere die reduzierten Automaten wie beschrieben.
- Stelle fest, ob die so erhaltenen reduzierten DEAs isomorph sind (teste z.B. alle Bijektionen).

Dieses Verfahren ist **nicht optimal**, denn es ist **unbekannt** ob das Testen von Isomorphie in polynomieller Zeit möglich ist.



# Minimierungs von NEAs

NEAs verhalten sich bzgl Minimierung **weniger gut** als DEAs:

- NEAs mit minimaler Zustandszahl sind **nicht** eindeutig
- Die Relation  $\sim_{\mathcal{A}}$  kann (vermutlich) **nicht** in polynomieller Zeit berechnet werden (vergl. Äquivalenzproblem für NEAs!)
- Der Quotientenautomat (definiert und berechnet mittels  $\sim_{\mathcal{A}}$ ) hat **nicht** in jedem Fall minimale Zustandszahl

## Aber:

- es gibt Generalisierung der **Approximationen**  $\sim_0, \sim_1, \dots$  für NEAs
- diese können nach wie vor in **polynomieller Zeit** berechnet werden
- der Quotientenautomat (bzgl.  $\sim_K$  aber auch bzgl  $\sim_{\mathcal{A}}$ ) ist **äquivalent**



# Abschlussbemerkungen Teil I

Einige aus Zeitgründen nicht behandelte Themenbereiche:

## ★ **Andere Varianten von endlichen Automaten**

NEA/DEA mit Ausgabe (Transduktor) beschreibt Funktion  $\Sigma^* \rightarrow \Gamma^*$   
2-Wege-Automaten, alternierende Automaten

## ★ **Automaten auf unendlichen Wörtern**

Automaten, die unendliche Wörter als Eingabe erhalten  
*Erreichen* eines akzeptierenden Zustandes wird ersetzt  
durch andere Akzeptanzbedingung

## ★ **Baumautomaten**

Bäume statt Wörter als Eingaben; lineare Abarbeitung nicht mehr möglich



## Teil I: Endliche Automaten und reguläre Sprachen

0. Grundbegriffe
1. Endliche Automaten
2. Nachweis der Nichterkennbarkeit
3. Abschlusseigenschaften
4. Entscheidungsprobleme
5. Reguläre Ausdrücke und Sprachen
6. Minimale DEAs und die Nerode-Rechtskongruenz

## Teil II: Grammatiken, kontextfreie Sprachen und Kellerautomaten

- 
7. Die Chomsky-Hierarchie
  8. Rechtslineare Grammatiken und reguläre Sprachen
  9. Normalformen und Entscheidungsprobleme
  10. Abschlusseigenschaften und Pumping-Lemma
  11. Kellerautomaten
  12. Die Struktur kontextfreier Sprachen

