Theoretische Informatik für Wirtschaftsinformatik und Lehramt

Eigenschaften regulärer Sprachen

Priv.-Doz. Dr. Stefan Milius
stefan.milius@fau.de

Theoretische Informatik
Friedrich-Alexander Universität Erlangen-Nürnberg

SS 2016

Gliederung

Worum geht es in diesem Abschnitt?

Bisher: Verschiedene Darstellungsformen regulärer Sprachen.

Jetzt:

- Aus einfachen regulären Sprachen komplexere konstruieren.
  (Durch zuvor betrachtete Operationen auf Sprachen.)
- Methode, um zu zeigen, dass eine Sprache nicht regulär ist.
  (Pumping Lemma)
- Minimierung von Automaten
  (DFA mit möglichst wenigen Zuständen für eine Sprache)

Lernziele

- Abschlusseigenschaften regulärer Sprachen kennen und beweisen können
- das Pumping-Lemma für reguläre Sprachen kennen und es zum Nachweis der Nicht-Regularität einer Sprache einsetzen können
- einen endlichen Automaten systematisch minimieren können

Zur Erinnerung: Operationen auf Sprachen

- $L_1, L_2 \subseteq \Sigma^*$ seien Sprachen:

  Durchschnitt und **Vereinigung**: $L_1 \cap L_2$ bzw. $L_1 \cup L_2$.

  Komplement der Sprache $L \subseteq \Sigma^*$: $\overline{L} = \Sigma^* \setminus L$.

- **Sprachprodukt (Konkatenation von Sprachen):**

  $L_1 \circ L_2 = \{vw \mid v \in L_1, w \in L_2\}$ (Schreibweise: $L_1 L_2$)

- **Sprachpotenz:**

  $L^n = \{w_1 w_2 \cdots w_n \mid w_i \in L \text{ für } i = 1, \ldots, n\}$ (für $n \in \mathbb{N}$)

  (Beachte: $L^0 = \{\varepsilon\}$.)

- **Kleene-Stern:** $L^* = \bigcup_{n \geq 0} L^n$ (außerdem: $L^+ = \bigcup_{n > 0} L^n$)

- **Spiegelung:** $L^R = \{w^R \mid w \in L\}$ $(\varepsilon^R = \varepsilon, (av)^R = v^R a)$

Zur Erinnerung: Gesetze für die Operationen auf Sprachen

Satz 1.2

Für beliebige $L, L_1, L_2 \subseteq \Sigma^*$ gilt:

a) $\overline{\overline{L}} = L$.

b) $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

c) $L(L_1 \cup L_2) = LL_1 \cup LL_2$.

d) $(L^*)^* = L^*$.

e) $(L_1 \cup L_2)^R = L_1^R \cup L_2^R$.

f) $(L_1 L_2)^R = L_2^R L_1^R$.

g) $(L^*)^R = (L^R)^*$.

Abschlusseigenschaften regulärer Sprachen (I)

Satz 3.1

Die regulären Sprachen sind abgeschlossen unter den Operationen

Vereinigung, Durchschnitt, Komplement, Produkt, Potenz, Kleene-Stern und Spiegelung.

D.h.: Sind L, L_1, L_2 reguläre Sprachen, $n \in \mathbb{N}$, dann sind auch die folgenden Sprachen regulär:

a) $L_1 \cup L_2$,
b) $L_1 \cap L_2$,
c) $\overline{L}$,
d) $L_1 \circ L_2$,
e) L^n,
f) L^*,
g) L^R

Hinweis: Wir können diese Eigenschaften nutzen, um aus einfachen komplexere reguläre Sprachen zusammenzubauen.

Abschlusseigenschaften regulärer Sprachen (II)

Für den Beweis von Satz 3.1 benötigen wir noch:

Satz 2.8 (zur Erinnerung)

Sei L eine Sprache. Die folgenden Aussagen sind äquivalent:

a) L wird von einer regulären Grammatik erzeugt (d. h. L ist regulär).

b) L wird von einem DFA akzeptiert.

c) L wird von einem NFA akzeptiert.

d) L wird durch einen regulären Ausdruck beschrieben.

Abschlusseigenschaften regulärer Sprachen (III)

Beweis (von Satz 3.1):

Seien L, L_1, L_2 reguläre Sprachen. Dann gilt (z. T. mit Satz 2.8):

- a) $L_1 \cup L_2$, d) $L_1 \circ L_2$, f) L^*:
  Es gibt reguläre Ausdrücke r, r_1, r_2 mit

  $$L(r) = L, \quad L(r_1) = L_1 \quad \text{und} \quad L(r_2) = L_2.$$

  Dann sind $r_1|r_2, r_1 r_2, r^*$ reguläre Ausdrücke, und die von diesen erzeugten Sprachen sind regulär:

  $$L_1 \cup L_2 = L(r_1|r_2), \quad L_1 \circ L_2 = L(r_1 r_2) \quad \text{und} \quad L^* = L(r^*).$$

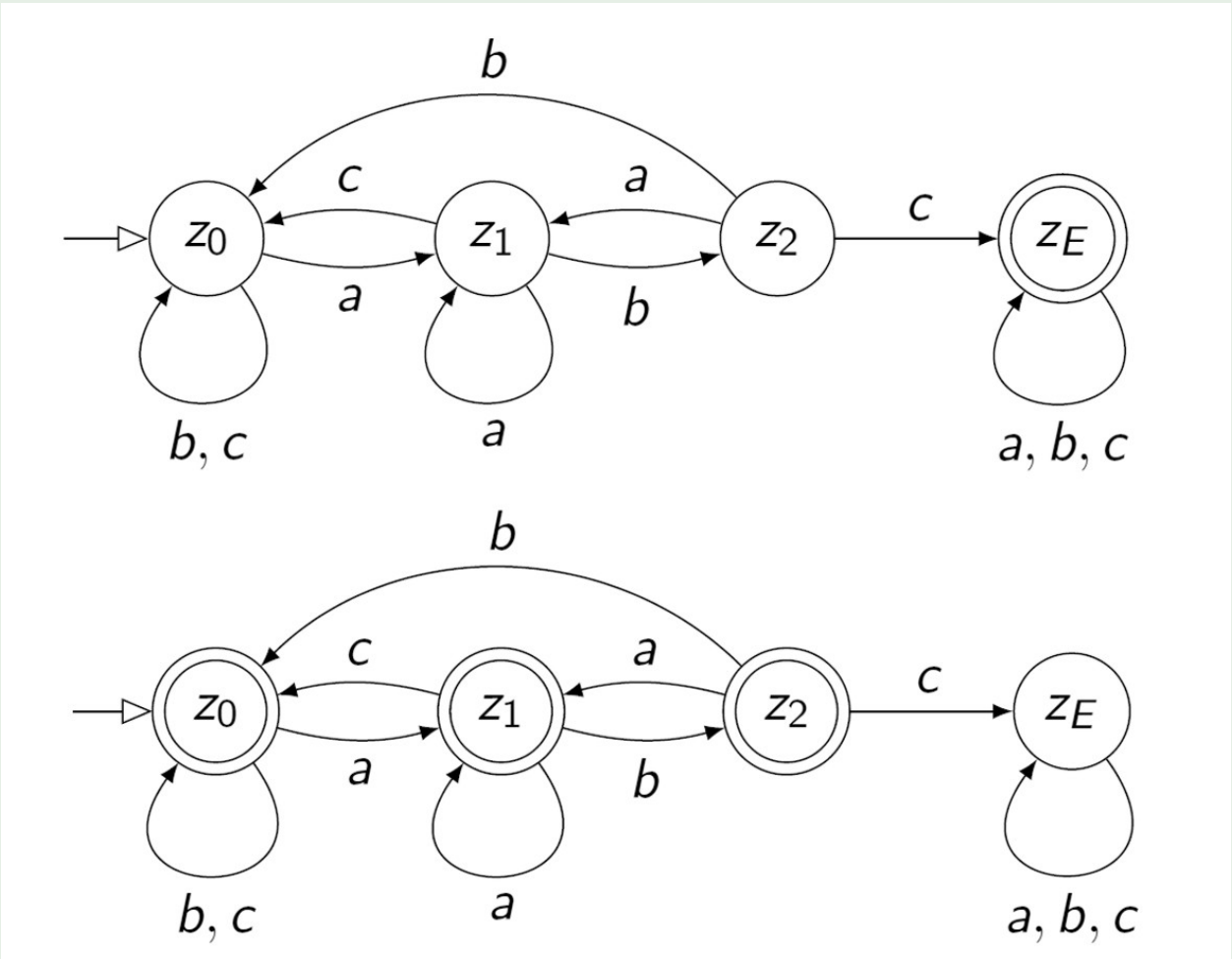
- c) $\overline{L}$: Es gibt einen DFA $M = (Z, \Sigma, \delta, z_0, E)$ mit $L(M) = L$.
  Für den DFA $M' = (Z, \Sigma, \delta, z_0, Z \setminus E)$ gilt:

  $$w \in L(M') \iff \hat{\delta}(z_0, w) \in Z \setminus E \iff \hat{\delta}(z_0, w) \notin E \iff w \notin L(M.)$$

  Also: $L(M') = \overline{L}$, d. h. $\overline{L}$ ist regulär, da es den DFA M' für $\overline{L}$ gibt.

Abschlusseigenschaften regulärer Sprachen (IV)

Beispiel: DFAs für Sprache und Komplementsprache



Abschlusseigenschaften regulärer Sprachen (V)

Beweis (Fortsetzung):

- b) $L_1 \cap L_2$: Wegen $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ (Satz 1.2 b)) folgt die Behauptung für $L_1 \cap L_2$ aus Satz 3.1 a) und c).
- e) L^n: $L^0 = \{\varepsilon\}$ und $L^1 = L$ sind trivialerweise regulär.

  Für $n \geq 2$: wegen $L^n = L \circ L \circ \cdots \circ L$ (n-mal) aus Satz 3.1 d).
- g) L^R: Es gibt einen regulären Ausdruck r mit $L(r) = L$.

  Zeige: Für jeden regulären Ausdruck r ist $L(r)^R$ regulär.

  Dann gilt: $L^R = L(r)^R$ ist regulär.

Abschlusseigenschaften regulärer Sprachen (VI)

Beweis (Fortsetzung):

- g) L^R (Fortsetzung):

  Zeige: Für jeden regulären Ausdruck r ist $L(r)^R$ regulär.
  Durch Induktion über den Aufbau von r:

  1. $r = \emptyset, r = \varepsilon, r = a$: In allen drei Fällen ist $L(r)^R = L(r)$.
     Also: $L(r)^R$ regulär.
  2. $r = r_1|r_2$: Dann ist $L(r) = L(r_1) \cup L(r_2)$.

     Also mit Satz 1.2 e): $L(r)^R = L(r_1)^R \cup L(r_2)^R$.

     Nach Induktionsvoraussetzung: $L(r_1)^R$ und $L(r_2)^R$ regulär.

     Daher: mit Satz 3.1 a) auch $L(r)^R$.
  3. $r = r_1 r_2$: Ebenso mit Satz 1.2 f) und Satz 3.1 d).
  4. $r = r_1^*$: Ebenso mit Satz 1.2 g) und Satz 3.1 f).

Reguläre vs. nicht-reguläre Sprachen (I)

Bisher:

- Methode zum Beweisen, dass eine Sprache regulär ist:

  Endlichen Automaten angeben
  (oder reguläre Grammatik oder regulären Ausdruck)

Im Folgenden:

1. Wie beweist man, dass eine Sprache L nicht regulär ist?
   - Idee: Wir beweisen, dass es keinen endlichen Automaten geben kann, der die Sprache L akzeptiert.
   - Dazu versuchen wir auszunutzen, dass ein endlicher Automat nur endlich viele Zustände enthält.
2. Wie groß ist der kleinste Automat, der eine gegebene reguläre Sprache akzeptiert?

   Gibt es überhaupt den kleinsten Automaten?

Reguläre vs. nicht-reguläre Sprachen (II)

Frage: Was unterscheidet die regulären von den nicht regulären Sprachen „charakteristisch“?

- Die Sprache $L = \{a^n b^n \in \{a, b\}^* \mid n \in \mathbb{N}\}$ ist nicht regulär.
- Warum ist das so?
  - **Annahme:** L regulär.

    Dann gibt es einen DFA $M = (Z, \Sigma, \delta, z_0, E)$, der L akzeptiert.
  - Für $i = 0, 1, 2, \ldots$ sei z_i der Zustand, den M erreicht, wenn $a^i = \underbrace{aa \cdots a}_{i\text{-mal}}$ gelesen wird.

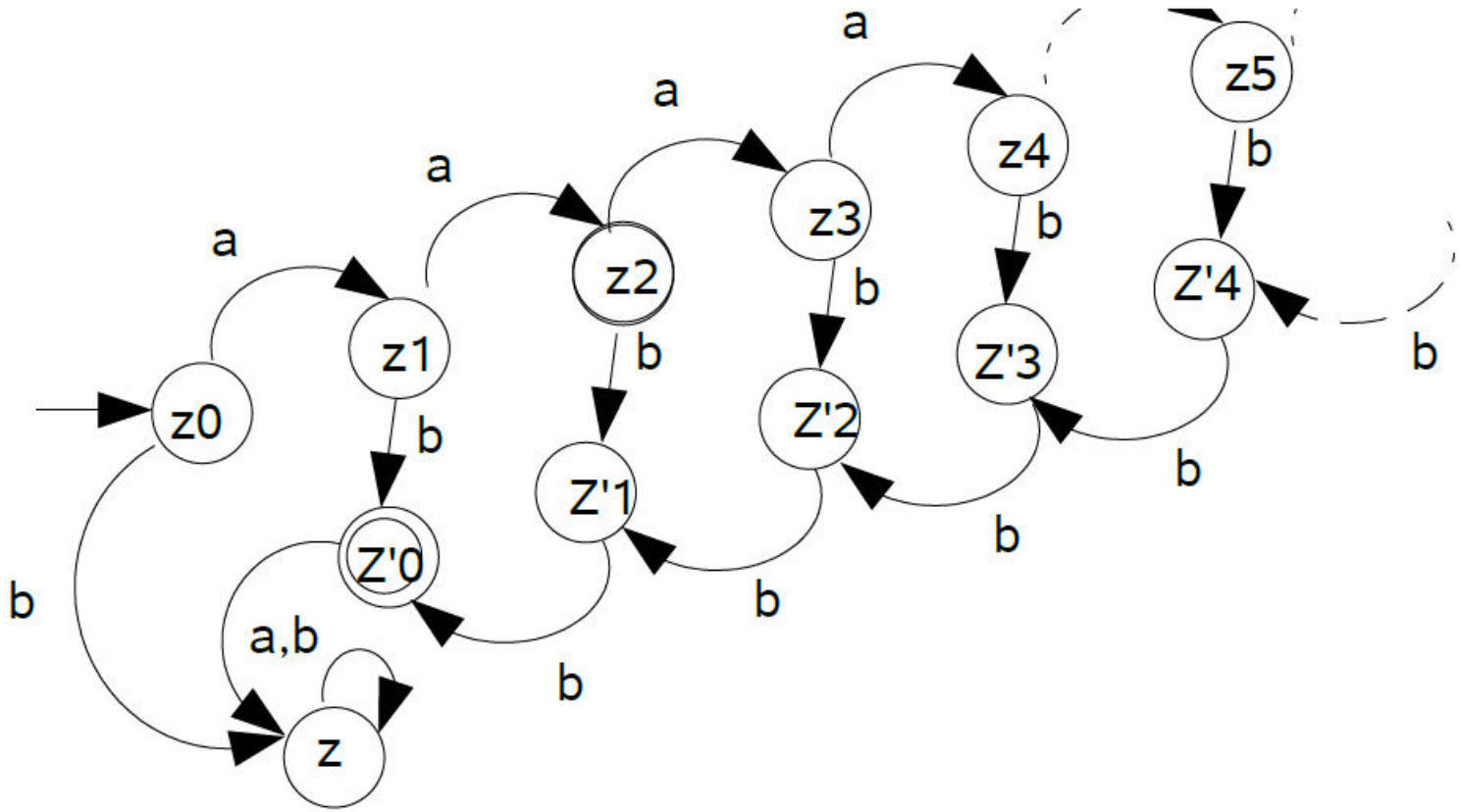
    (Anschließend wird das erste b „erwartet“.
  - Für $i \neq j$ gilt: $a^i b^i \in L$ und $a^j b^i \notin L$.

    Also: $\hat{\delta}(z_i, b^i) \in E$ und $\hat{\delta}(z_j, b^i) \notin E$.

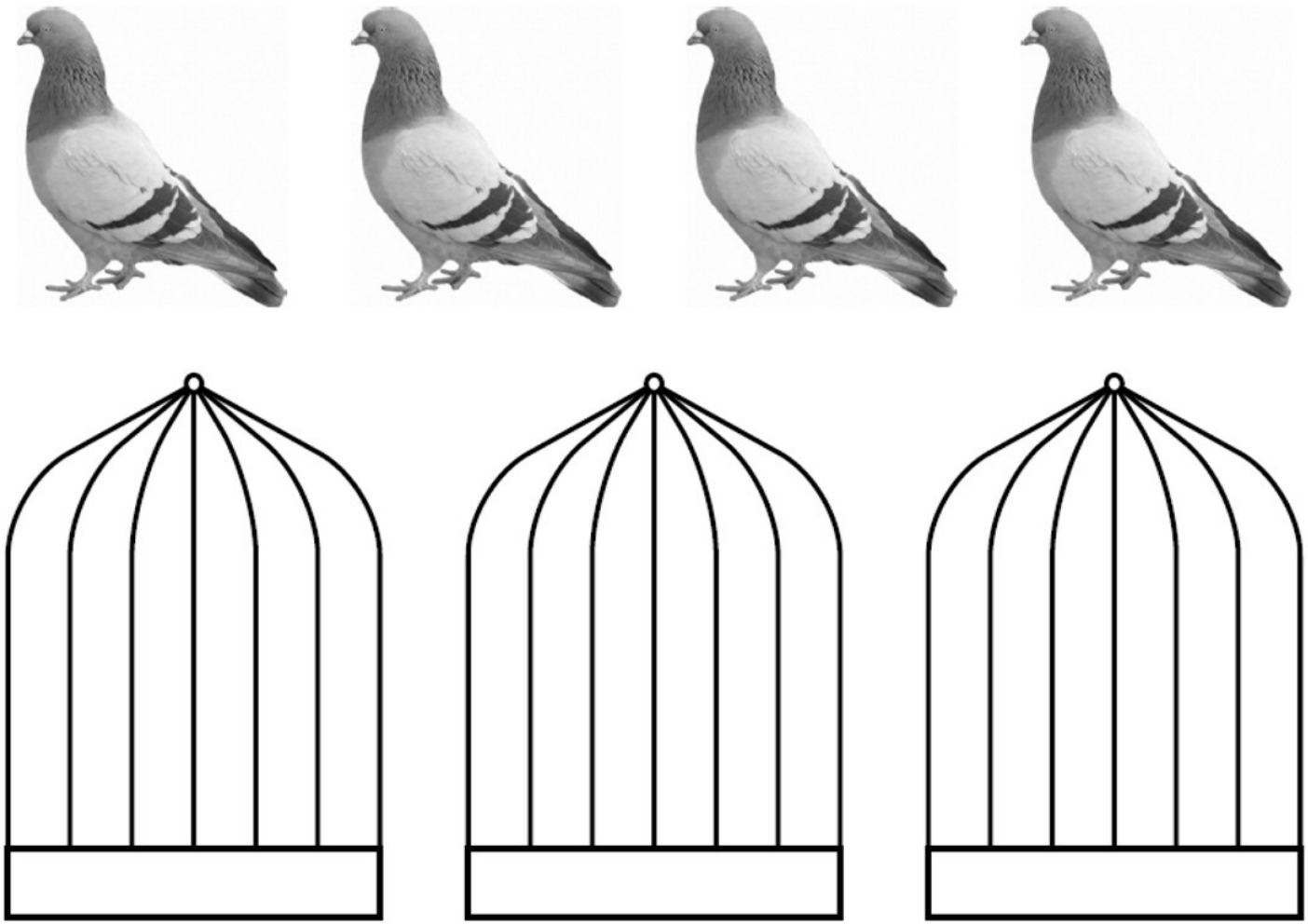
    Daher gilt: $z_i \neq z_j$.
  - Dann hat M unendlich viele paarweise verschiedene Zustände $z_0, z_1, z_2, \ldots$

    Widerspruch!

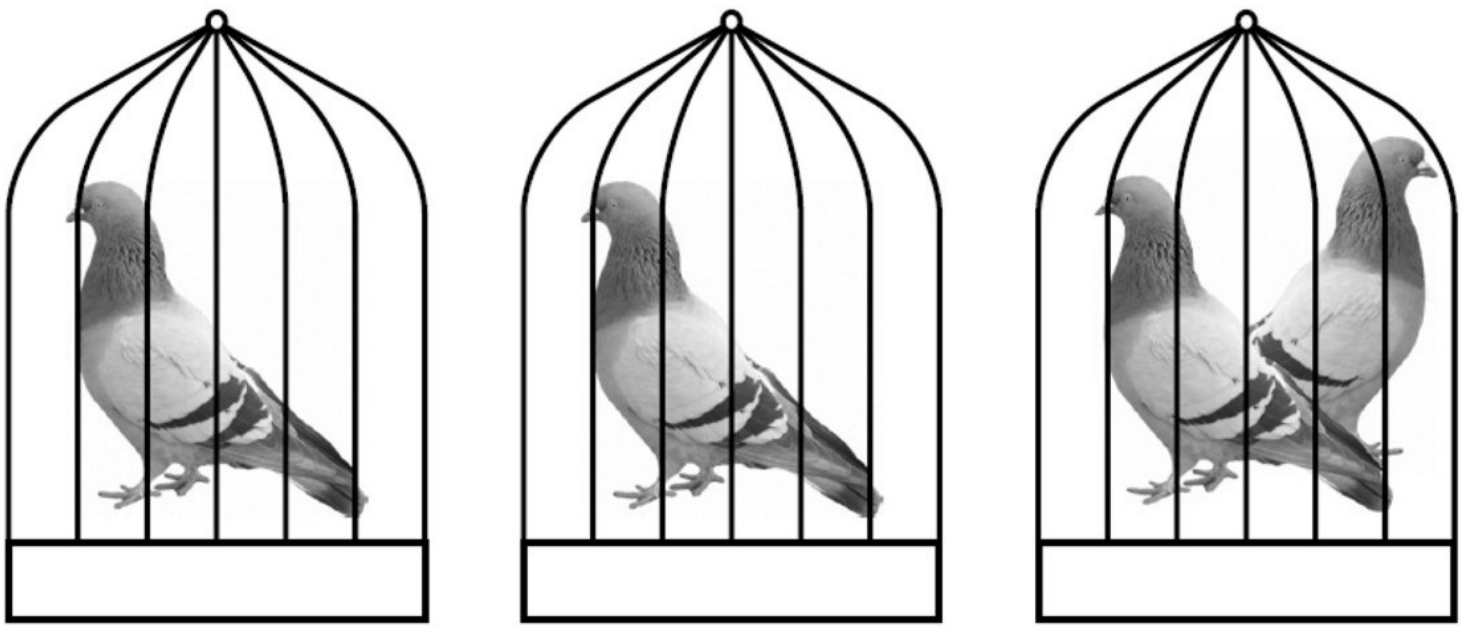
Reguläre vs. nicht-reguläre Sprachen (III)



Das Schubfach-Prinzip (I)



Das Schubfach-Prinzip (II)



Das Schubfach-Prinzip (III)

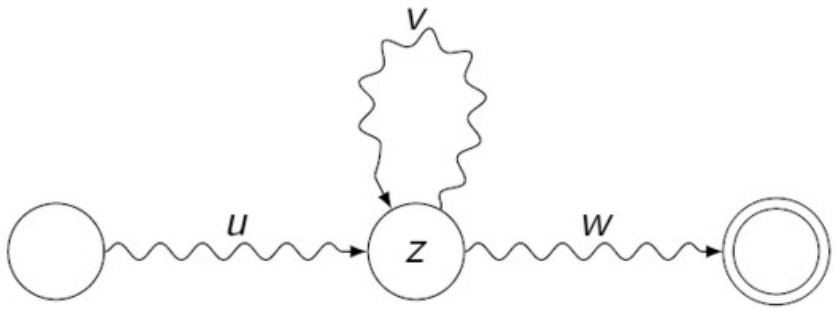
- Das Schubfachprinzip (englisch: „pigeon hole principle“):

  Wenn man m Objekte über n Mengen verteilen möchte und $m > n$ ist, dann gibt es mindestens eine Menge, die mindestens zwei Elemente enthält.

- Das „Schubfachprinzip“ für endliche Automaten:

  Wenn ein Automat mit n Zuständen einen Pfad der Länge m enthält und $m \geq n$ ist, dann gibt es mindestens einen Zustand, der mehrfach auf dem Pfad vorkommt.

Das Pumping-Lemma (I)

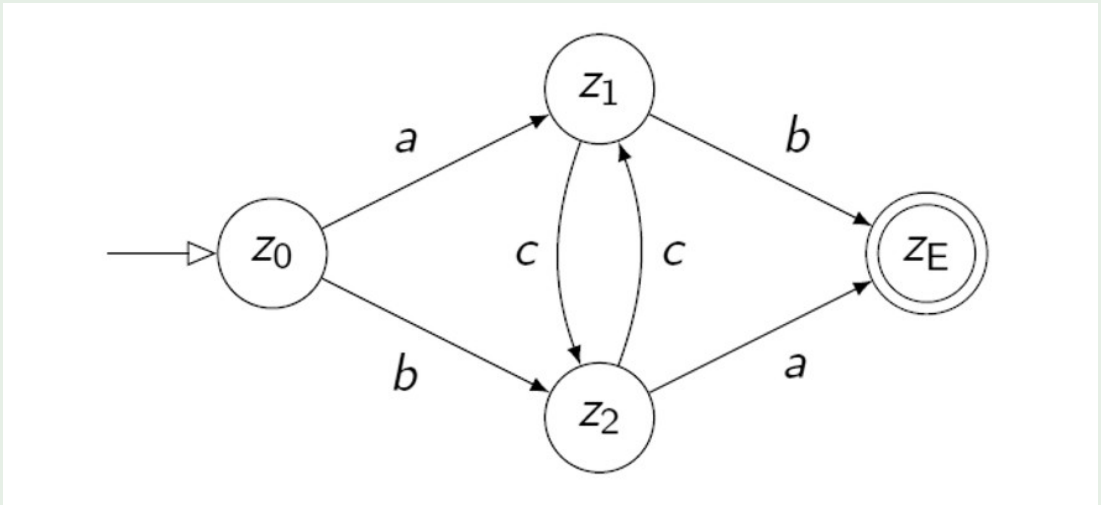


- Dadurch entsteht eine Schleife.
  Diese kann mehrfach (oder gar nicht) durchlaufen werden.
  Dadurch wird $x = uvw$ „aufgepumpt“:
  auch uw, uv^2w, uv^3w, ... liegen in L.
- Außerdem kann man für u, v, w folgende Eigenschaften verlangen:
  (n = Anzahl der Zustände)
  1. $|v| \geq 1$: Die Schleife ist auf jeden Fall nicht trivial und enthält zumindest einen Übergang.
  2. $|uv| \leq n$: Spätestens nach n Alphabetsymbolen wird der Zustand z das zweite Mal erreicht.

Das Pumping-Lemma (II)

Beispiel

$M = (\{z_0, z_1, z_2, z_3\}, \{a, b, c\}, \delta, z_0, \{z_E\})$



$$x = a\,c\,c\,c\,a = \underbrace{a\,c}_{u}\,\underbrace{c\,c}_{v}\,\underbrace{a}_{w} \in L(M)$$

$$uv^0w = \underbrace{a\,c}_{u}\,\underbrace{a}_{w} \in L(M) \qquad uv^2w = \underbrace{a\,c}_{u}\,\underbrace{c\,c}_{v}\,\underbrace{c\,c}_{v}\,\underbrace{a}_{w} \in L(M)$$

Das Pumping-Lemma (III)

- Wenn ein endlicher Automat beliebig lange Wörter akzeptiert, muss er einen Zyklus bzw. eine Schleife haben.
- Aus dieser Eigenschaft von endlichen Automaten leiten wir eine Eigenschaft von regulären Sprachen ab (das *Pumping Lemma*, nächste Folie).
- Das Pumping-Lemma für reguläre Sprachen sagt aus, dass reguläre Sprachen diese abgeleitete Eigenschaft haben.
- Das Pumping-Lemma macht *keine* direkten Aussagen über endliche Automaten.

Das Pumping-Lemma (IV)

Satz 3.2 (Pumping-Lemma)

Sei L eine reguläre Sprache über einem Alphabet Σ.

Dann gibt es eine natürlich Zahl $n \geq 1$ (**Pumping-Zahl für** L), so dass gilt:

Zu jedem $x \in L$ mit $|x| \geq n$ gibt es $u, v, w \in \Sigma^*$ mit $x = uvw$ und

1. $|v| \geq 1$,
2. $|uv| \leq n$,
3. Für jedes $k \in \mathbb{N}$ ist $uv^k w \in L$.

Das Pumping-Lemma (V)

Beweis: Da L regulär ist:

es gibt einen DFA $M = (Z, \Sigma, \delta, z_0, E)$ mit $L(M) = L$.

Setze: $n = |Z|$. (Dann gilt $n \geq 1$.)

Sei nun $x = a_1 \cdots a_m$ mit $|x| \geq n$, also $m \geq n$.

Bei der Akzeptierung von x durchläuft M $m+1$ Zustände

$$z_0, z_1, \ldots, z_m$$

mit $z_m \in E$ und $\delta(z_i, a_{i+1}) = z_{i+1}$ für $i = 0, \ldots, m-1$.

Da M insgesamt nur n Zustände hat: in der Teilfolge

$z_0, \ldots, z_n$ muss (mindestens) ein Zustand zweimal vorkommen.

Es gibt also i, j mit $0 \leq i < j \leq n$, $z_i = z_j$ und

$$\hat{\delta}(z_0, a_1 \ldots a_i) = z_i,$$

$$\hat{\delta}(z_i, a_{i+1} \ldots a_j) = z_j = z_i,$$

$$\hat{\delta}(z_j, a_{j+1} \ldots a_m) = z_m \in E.$$

Das Pumping-Lemma (VI)

Beweis (Fortsetzung):

Setze: $u = a_1 \cdots a_i$, $v = a_{i+1} \cdots a_j$ und $w = a_{j+1} \cdots a_m$.

Dann ist $x = uvw$ und

1. $|v| \geq 1$,
2. $|uv| \leq n$,
3. $\hat{\delta}(z_i, v^k) = z_j = z_i$ für alle $k \in \mathbb{N}$.

Aus 3. folgt: $\hat{\delta}(z_0, uv^k w) \in E$.

Also: $uv^k w \in L$ für alle $k \in \mathbb{N}$. □

Das Pumping-Lemma (VII)

- Wie kann man das Pumping-Lemma dazu nutzen, um zu zeigen, dass eine Sprache nicht regulär ist?
- Alle regulären Sprachen erfüllen die Bedingungen des Pumping-Lemmas.
  Das heißt: wenn eine Sprache L die Bedingungen nicht erfüllt, dann ist sie nicht regulär.
- **Beachte** (häufige Fehlerquelle!):
  Es gibt auch nicht-reguläre Sprachen, die die Bedingungen des Pumping-Lemmas erfüllen.

  (Das Pumping-Lemma liefert nur eine notwendige Bedingung für Regularität, nicht aber eine hinreichende!)
- **Das heißt:** Man kann das Pumping-Lemma nicht benutzen, um zu zeigen, dass eine Sprache regulär ist.

Das Pumping-Lemma (VIII)

Beispiel

Behauptung: $L = \{a^i b^i \in \{a,b\}^* \mid i \in \mathbb{N}\}$ ist nicht regulär.

Beweis:

- **Annahme:** $L = \{a^i b^i \in \{a,b\}^* \mid i \in \mathbb{N}\}$ ist regulär.
- Sei n die Pumping-Zahl von L. Betrachte: $x = a^n b^n \in L$.
  Es ist $|x| = 2n \geq n$, also gibt es
  $$u, v, w \in \{a,b\}^* \text{ mit } x = uvw$$
  und den in Satz 3.2 genannten Eigenschaften 1-3.
- Aus $|uv| \leq n$ folgt:
  in v kommt kein b vor, also: $v = a^m$.
- Wegen $|v| \geq 1$ ist $m \geq 1$ und damit
  $$uv^0w = uw = a^{n-m}b^n \notin L.$$
- **Widerspruch** (denn gemäß dritter Eigenschaft: $uv^0w \in L$).

Minimierung von Automaten

Minimierung von Automaten (I)

Motivation

Bisher:

- Endliche Automaten im Zusammenhang mit regulären Sprachen
- systematische Konstruktionen:

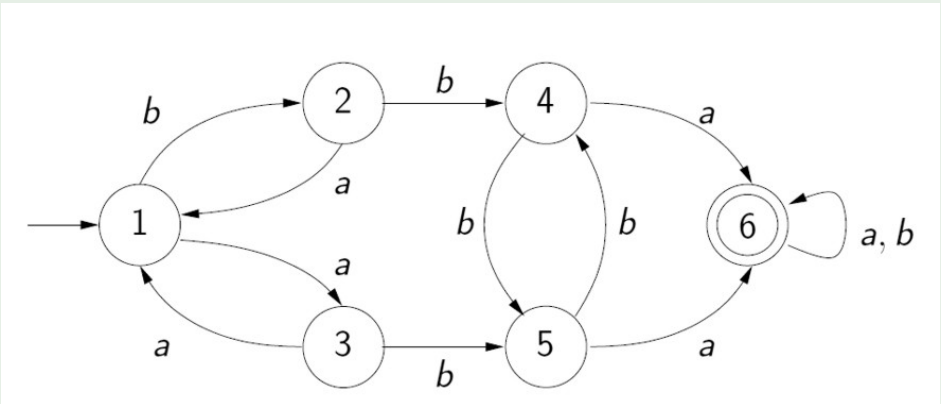
$$\text{DFA} \rightsquigarrow \text{reguläre Grammatiken} \rightsquigarrow \text{NFA} \rightsquigarrow \text{DFA}$$

Jetzt:

- Vereinfachung eines gegebenen DFA, so dass immer noch dieselbe Sprache akzeptiert wird
- möglichst kleine Realisierung von regulären Sprachen durch DFA
- **Also:** Minimierung endlicher Automaten

Minimierung von Automaten (II)

Beispiel



Feststellung: für die Zustände 4, 5 gilt

- Mit einem Wort, das ein *a* enthält, landet man von dort aus immer in einem Endzustand (nämlich in 6).
- Mit einem Wort, das kein *a* enthält, landet man von dort aus immer in einem Nicht-Endzustand (in 4 bzw. in 5).

Minimierung von Automaten (III)

Beispiel (Fortsetzung)

Daraus folgt:

4 und 5 sind *erkennungsäquivalent*

und können zu einem Zustand verschmolzen werden.

Äquivalenzklassenautomaten (I)

- Pumping-Lemma: notwendige Bedingung für Regularität.
- Jetzt: notwendige und hinreichende Charakterisierung der regulären Sprachen.

Definition (Myhill-Nerode-Äquivalenz)

Sei $L \subseteq \Sigma^*$ formale Sprache.
Die binäre Relation $\sim_L \subseteq \Sigma^* \times \Sigma^*$ ist gegeben durch:

$$x \sim_L y \quad :\Longleftrightarrow \quad \text{für alle } w \in \Sigma^* \text{ gilt } xw \in L \Leftrightarrow yw \in L.$$

Bemerkung: $x \sim_L y$ bedeutet:

eine (Weiter-) Verarbeitung beliebiger Wörter w führt zum gleichen Akzeptanz- (bzw. Nicht-Akzeptanz)-Ergebnis.

Äquivalenzklassenautomaten (II)

Beispiel

Für $L = \{a^i b^i \in \{a,b\}^* \mid i \in \mathbb{N}\}$ gilt:

- Für beliebige $i, j \in \mathbb{N}$ mit $i \neq j$ gilt: $a^i \not\sim_L a^j$,
  denn: $a^i b^i \in L$ aber $a^j b^i \notin L$.

Dagegen gilt z. B.

- $aab \sim_L aaabb$ und auch
- $abb \sim_L ba$, wegen

$$x \sim_L y \iff \text{für alle } w \in \Sigma^* : xw \notin L \Leftrightarrow yw \notin L.$$

Für die nächsten Schritte benötigen wir einige mathematische Grundbegriffe, nämlich:

- Äquivalenzrelation
- Äquivalenzklasse

Äquivalenzrelationen (I)

Idee:

- Ähnliche Objekte als gleich ansehen.
- Formalisierung der Gleichwertigkeitsbegriffs

Definition

Sei M eine Menge. Die binäre Relation $R \subseteq M \times M$ heißt **Äquivalenzrelation**, wenn sie reflexiv, transitiv und symmetrisch ist.

Beispiele

- die „=“-Relation
- die Gleichmächtigkeit endlicher Mengen, wobei die Mächtigkeit die Anzahl der Elemente beschreibt

Äquivalenzrelationen (II)

Beispiele (Fortsetzung)

Gegeben: M = alle Schüler einer Schule

$s_1 \, R \, s_2$ = „s_1 geht in die gleiche Klasse, wie s_2“

- Ein Schüler geht in die gleiche Klasse, wie er selbst (Reflexivität).
- Paul geht in die gleiche Klasse wie Petra und Petra in die gleiche Klasse wie Nicole.
  Deshalb gehen auch Paul und Nicole in die gleiche Klasse (Transitivität).
- Paul geht in die gleiche Klasse wie Petra und Petra in die gleiche Klasse wie Paul (Symmetrie).

Äquivalenzklassen

Definition

Sei $R \subseteq M \times M$ eine Äquivalenzrelation.
Dann heißen für $a \in M$ die Mengen

$$[a]_R = \{b \in M \mid a\, R\, b\}$$

Äquivalenzklassen von R.

Beispiel (Fortsetzung)

Äquivalenzklasse eines Schülers:

Menge aller seiner Mitschüler der gleichen Schulklasse
(ihn selbst mit eingeschlossen wegen der Reflexivität).

Die Menge der Äquivalenzklassen ist die Menge der Schulklassen.

Index (I)

Die Myhill-Nerode-Äquivalenz $\sim_L \subseteq \Sigma^* \times \Sigma^*$ einer Sprache L ist eine Äquivalenzrelation.

Bilde die Äquivalenzklassen:

Definition (Index)

Sei L eine Sprache über einem Alphabet Σ.

Für $x \in \Sigma^*$ bezeichne $[x]_{\sim_L}$ die Äquivalenzklasse von x bzgl. $\sim_L$:

$$[x]_{\sim_L} = \{y \in \Sigma^* \mid x \sim_L y\}.$$

Der **Index** $\mathcal{I}(L)$ von L ist die Anzahl der Äquivalenzklassen in Σ^* bzgl. $\sim_L$:

$$\mathcal{I}(L) = |\{\, [x]_{\sim_L} \mid x \in \Sigma^* \,\}|$$

Ist der Kontext klar, schreibe $[x]$ statt $[x]_{\sim_L}$.

Index (II)

Beispiel

Sei $L = \{v \in \{a, b\}^* \mid v \text{ enthält genau ein } b\}$.
Für beliebiges $x \in \{a, b\}^*$ gilt:

x enthält kein $b \iff x \sim_L \varepsilon$	$(xw \in L \iff \varepsilon w = w \in L$ für alle $w)$,
x enthält genau ein $b \iff x \sim_L b$	$(xw \in L \iff bw \in L$ für alle $w)$,
x enthält mehr als ein $b \iff x \sim_L bb$	$(xw \notin L \iff bbw \notin L$ für alle $w)$,

und $\varepsilon \not\sim_L b, \varepsilon \not\sim_L bb$ und $b \not\sim_L bb$.

Damit sind $[\varepsilon]$, $[b]$, $[bb]$
alle Äquivalenzklassen in $\{a, b\}^*$ bezüglich $\sim_L$, und es ist $\mathcal{I}(L) = 3$.

Index (III)

Vermutung: Zusammenhang zwischen dem Index einer Sprache und der notwendigen Anzahl von Zuständen eines DFA, der diese Sprache akzeptiert.

Es gilt zunächst:

Satz 3.3

Sei M ein DFA mit n Zuständen. Dann gilt: $n \geq \mathcal{I}(L(M))$.

Index (IV)

Beweis: Seien $u, v \in \Sigma^*$ und $[u]$ bzw. $[v]$ die zugehörigen Äquivalenzklassen.

Zeige: verschiedene Klassen bestimmen verschiedene Zustände:

$$[u] \neq [v] \qquad \Longrightarrow \qquad \hat{\delta}(z, u) \neq \hat{\delta}(z, v).$$

Es gelte $\hat{\delta}(z, u) = \hat{\delta}(z, v)$.

Dann gilt:

$$\forall w \in \Sigma^* : \hat{\delta}(z, uw) = \hat{\delta}(z, vw)$$
$$\forall w \in \Sigma^* : \hat{\delta}(z, uw) \in E \Leftrightarrow \hat{\delta}(z, vw) \in E$$
$$\forall w \in \Sigma^* : (uw \in L(M) \Leftrightarrow vw \in L(M)) \text{ und damit } [u] = [v].$$

Äquivalenzklassenautomat (I)

Folgerung: Für jede Sprache L gilt:

- Jeder DFA, der L akzeptiert, hat mindestens $\mathcal{I}(L)$ Zustände (Satz 3.3).
- Ein solcher DFA ist nur möglich, falls $\mathcal{I}(L)$ endlich ist.
- Ist dies der Fall, so können wir zu L auch einen DFA angeben, der mit genau $\mathcal{I}(L)$ Zuständen auskommt.

  (Beachte: $\mathcal{I}(L) \geq 1$ gilt immer.)

Äquivalenzklassenautomat (II)

Definition (Äquivalenzklassenautomat)

Sei L eine Sprache über einem Alphabet Σ mit $\mathcal{I}(L) = n$ und den n verschiedenen Äquivalenzklassen bzgl. $\sim_L$: $[x_1], [x_2], \ldots, [x_n]$.

Der **Äquivalenzklassenautomat** von L ist der DFA $M = (Z, \Sigma, \delta, z_0, E)$ mit

$$\begin{aligned} Z &= \{[x_1], [x_2], \ldots, [x_n]\}, \\ \delta([x], a) &= [xa] \quad \text{für alle } x \in \Sigma^*, a \in \Sigma, \\ z_0 &= [\varepsilon], \\ E &= \{[x] \mid x \in L\}. \end{aligned}$$

Bemerkungen: Man muss prüfen, ob der DFA M **wohldefiniert** ist, d.h.:

die Festlegungen für δ und E sind unabhängig von den gewählten Repräsentanten der Äquivalenzklassen

Äquivalenzklassenautomat (III)

Satz 3.4

Für den Äquivalenzklassenautomaten M einer Sprache L gilt: $L(M) = L$.

Beweis: Per Induktion über w zeige: für M gilt

$$\hat{\delta}([x], w) = [xw] \qquad \text{für alle } x, w \in \Sigma^*.$$

Induktionsanfang: $\hat{\delta}([x], \varepsilon) = [x] = [x\varepsilon]$,

Induktionsschritt:

$$\hat{\delta}([x], aw) \overset{\text{Def. } \hat{\delta}}{=} \hat{\delta}(\delta([x], a), w) \overset{\text{Def. Äqu.Aut.}}{=} \hat{\delta}([xa], w) \overset{\text{I.V.}}{=} [xaw].$$

Damit gilt:

$$w \in L(M) \iff \hat{\delta}([\varepsilon], w) \in E \iff [w] \in E \iff w \in L.$$

Also $L(M) = L$. □

Äquivalenzklassenautomat (IV)

Beispiel

$L = \{v \in \{a, b\}^* \mid v \text{ enthält genau ein } b\}$
Äquivalenzklassen: $[\varepsilon]$, $[b]$ und $[bb]$.

Äquivalenzklassenautomat

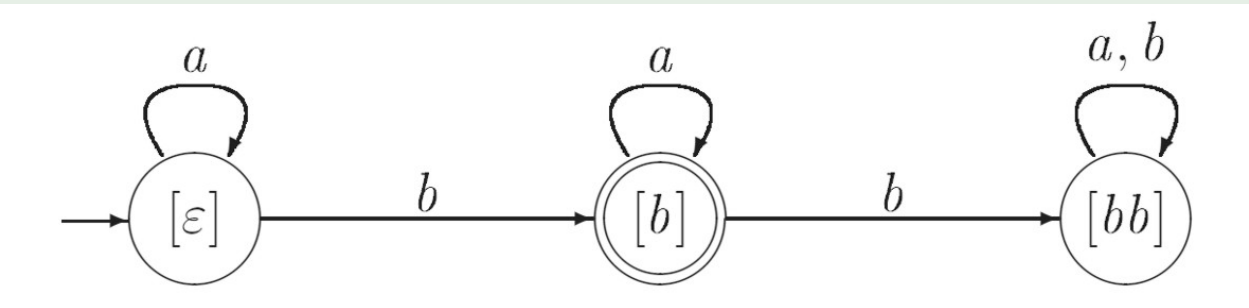
Zustände: $[\varepsilon], [b]$ und $[bb]$.
Anfangszustand: $[\varepsilon]$
Akzeptierende Zustände: $E = \{[b]\}$ (da $b \in L$ und $\varepsilon, bb \notin L$).
Überführungsfunktion δ:

$$\delta([\varepsilon], a) = [\varepsilon a] = [\varepsilon] \qquad \delta([bb], a) = [bba] = [bb]$$
$$\delta([\varepsilon], b) = [\varepsilon b] = [b] \qquad \delta([bb], b) = [bbb] = [bb]$$
$$\delta([b], a) = [ba] = [b]$$
$$\delta([b], b) = [bb]$$

Äquivalenzklassenautomat (V)

Beispiel (Fortsetzung)

Grafisch:



Äquivalenzklassenautomat (VI)

Satz 3.5 (Satz von Myhill-Nerode)

Eine Sprache L ist genau dann regulär, wenn der Index von L endlich ist.

- Genau dann wenn L regulär ist, lässt sich L durch einen DFA akzeptieren, d.h. mit endlich vielen Zuständen.
- Der Index – Mindestanzahl der Zustände – ist dann endlich.

$\rightsquigarrow$ Methode zum Nachweis der Nichtregularität

Äquivalenzklassenautomat (VII)

Beispiel

Zuvor: Nicht-Regularität von $L = \{a^i b^i \in \{a, b\}^* \mid i \in \mathbb{N}\}$ mit Hilfe des Pumping-Lemmas.

Jetzt: mit Hilfe des Satzes von Myhill-Nerode:

- Für $i, j \in \mathbb{N}$ mit $i \neq j$ gilt: $a^i \not\sim_L a^j$ (siehe oben)
- Damit sind verschiedene Äquivalenzklassen von $\{a, b\}^*$ bzgl. $\sim_L$:

$$[\varepsilon], \quad [a], \quad [aa], \quad [aaa], \quad \cdots$$

- **Also:** Index von L ist unendlich und daher L nicht regulär.

Minimierung von Automaten (I)

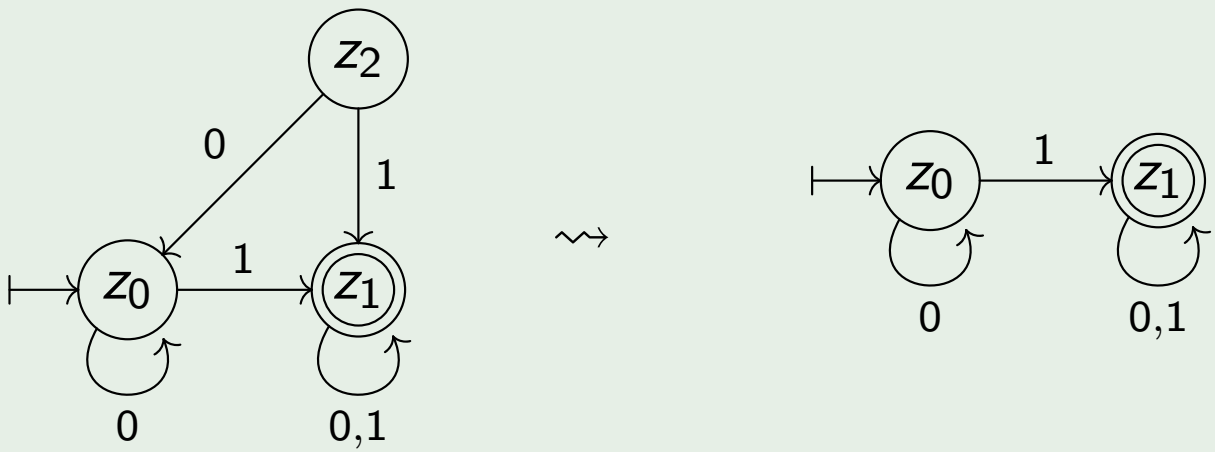
- Ist L eine reguläre Sprache, so hat jeder DFA, der L akzeptiert, mindestens $\mathcal{I}(L)$ Zustände.
- Der Äquivalenzklassenautomat von L akzeptiert L und hat *genau* $\mathcal{I}(L)$ Zustände.
- Der Äquivalenzklassenautomat ist also ein **Minimalautomat** für L:

  es gibt keinen DFA mit weniger Zuständen für L.
- Verschiedene Minimalautomaten für eine Sprache unterscheiden sich höchstens im Namen der Zustände.
- **Frage:** Kann man aus einem gegebenen (im Allgemeinen nicht minimalen) DFA M den Minimalautomaten M_{min} mit $L(M_{min}) = L(M)$ konstruieren/berechnen?

Minimierung von Automaten (II)

Unerreichbare Zustände können entfernt werden.

Beispiel



Definition (Erreichbarer Automat)

Ein DFA $M = (Z, \Sigma, \delta, z_0, E)$ heißt **erreichbar** wenn er keine unerreichbaren Zustände enthält, d.h.:

$$\text{für jedes } z \in Z \text{ existiert } w \in \Sigma^* \text{ mit } \hat{\delta}(z_0, w) = z.$$

Minimierung von Automaten (III)

Definition (Erkennungsäquivalenz)

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein DFA.

Die binäre Relation $\approx_M \subseteq Z \times Z$ ist gegeben durch:

$$z \approx_M z' :\iff \text{ für alle } x \in \Sigma^* \text{ gilt } \hat{\delta}(z, x) \in E \Leftrightarrow \hat{\delta}(z', x) \in E.$$

Zwei Zustände $z, z' \in Z$ heißen **erkennungsäquivalent** genau dann, wenn $z \approx_M z'$ gilt.

Bemerkung: $z \approx_M z'$ bedeutet:

eine (Weiter-)Verarbeitung beliebiger Wörter von z und z' aus führt zum gleichen Akzeptanzergebnis
(bzw. Nicht-Akzeptanzergebnis).

Minimierung von Automaten (IV)

Definition (reduziert)

Ein erreichbarer DFA M heißt **reduziert**, wenn für beliebige *verschiedene* Zustände z und z' von M gilt:

$$z \not\approx_M z'.$$

- z und z' mit $z \approx_M z'$ leisten in Bezug auf Akzeptanz „das Gleiche“

  $\rightsquigarrow$ nur einer von beiden wird benötigt
- Ein reduzierter Automat benötigt alle seine Zustände, ist also minimal.

Satz 3.6

Seien M ein DFA und $L = L(M)$.

M hat genau dann $\mathcal{I}(L)$ Zustände, wenn M reduziert ist.

Minimierung von Automaten (V)

- Ein reduzierter DFA ist also Minimalautomat für die von ihm akzeptierte Sprache.
- Sei M ein nicht reduzierter DFA.

  Dann erhalte reduzierten DFA mit
  Zuständen = Äquivalenzklassen bzgl. $\approx_M$:

  (analog zum Äquivalenzklassenautomaten)

Reduzierter Automat (I)

Definition (Reduzierter Automat)

Sei $M = (Z, \Sigma, \delta, z_0, E)$ ein erreichbarer DFA und den Äquivalenzklassen in Z bzgl. $\approx_M$: $[z_0], [z_1], \ldots, [z_n]$.

Der **reduzierte Automat** von M ist der DFA $M' = (Z', \Sigma, \delta', [z_0], E')$ mit

$$\begin{aligned} Z' &= \{[z_0], [z_1], \ldots, [z_n]\}, \\ \delta'([z], a) &= [\delta(z, a)] \quad \text{für alle } [z] \in Z' \text{ und } a \in \Sigma, \\ E' &= \{[z] \mid z \in E\}. \end{aligned}$$

Bemerkung:

Äquivalenzklassenautomat hat Äquivalenzklassen bzgl. $\sim_L$

reduzierter Automat hat Äquivalenzklassen bzgl. $\approx_M$

Reduzierter Automat (II)

Identifizierung von $z \approx_M z'$ erhält die akzeptierte Sprache:

Satz 3.7

Sei M ein erreichbarer DFA und M' der reduzierte Automat von M. Dann gilt: $L(M') = L(M)$.

Reduzierter Automat (III)

Beweis: Per Induktion über w zeige:

$$\hat{\delta}'([z], w) = [\hat{\delta}(z, w)] \qquad \text{für alle } z \in Z, w \in \Sigma^*.$$

Induktionsanfang: $\hat{\delta}'([z], \varepsilon) = [z] = [\hat{\delta}(z, \varepsilon)]$,

Induktionsschritt:

$$\begin{aligned}
\hat{\delta}'([z], aw) &\overset{\text{Def. } \hat{\delta}}{=} \hat{\delta}'(\delta'([z], a), w) &&\overset{\text{Def. red. Aut.}}{=} \hat{\delta}'([\delta(z, a)], w) \\
&\overset{\text{I.V.}}{=} [\hat{\delta}(\delta(z, a), w)] &&\overset{\text{Def. } \hat{\delta}}{=} [\hat{\delta}(z, aw)].
\end{aligned}$$

Daher gilt:

$$\begin{aligned}
w \in L(M') &\iff \hat{\delta}'([z_0], w) \in E' \\
&\iff [\hat{\delta}(z_0, w)] \in E' \\
&\iff \hat{\delta}(z_0, w) \in E \\
&\iff w \in L(M),
\end{aligned}$$

also $L(M') = L(M)$. □

Konstruktion eines Minimalautomaten (I)

Folgerung: Der reduzierte DFA M' kann iterativ berechnet werden.

Betrachte für $k \in \mathbb{N}$ die Relationen $R_k \subseteq Z \times Z$ mit

$$z \, R_k \, z' :\Longleftrightarrow \underbrace{\hat{\delta}(z,x) \in E \Leftrightarrow \hat{\delta}(z',x) \in E \text{ für alle } x \in \Sigma^*}_{z \approx_M z'}, |x| \leq k.$$

Konstruktion eines Minimalautomaten (II)

$$z\ R_k\ z' :\iff \underbrace{\hat{\delta}(z,x) \in E \Leftrightarrow \hat{\delta}(z',x) \in E \text{ für alle } x \in \Sigma^*}_{z \approx_M z'}, |x| \leq k.$$

Offenbar gilt: $\approx_M = \bigcap_{k=0}^{\infty} R_k$.

Für die R_k gilt:

$$\begin{aligned}
z\ R_0\ z' &\iff z \in E \Leftrightarrow z' \in E, \\
z\ R_{k+1}\ z' &\iff z\ R_k\ z' \quad \text{und} \quad \hat{\delta}(z,aw) \in E \Leftrightarrow \hat{\delta}(z',aw) \in E \\
&\qquad \text{für alle } a \in \Sigma, |w| \leq k \\
&\overset{\text{Def. } \hat{\delta}}{\iff} z\ R_k\ z' \quad \text{und} \\
&\qquad \hat{\delta}(\delta(z,a),w) \in E \Leftrightarrow \hat{\delta}(\delta(z',a),w) \in E \\
&\qquad \text{für alle } a \in \Sigma, |w| \leq k \\
&\overset{\text{Def. } R_k}{\iff} z\ R_k\ z' \quad \text{und} \quad \delta(z,a)\ R_k\ \delta(z',a) \quad \text{für alle } a \in \Sigma.
\end{aligned}$$

Konstruktion eines Minimalautomaten (III)

Die Relation $\approx_M$ kann iterativ wie folgt berechnet werden:

1. Berechne R_0.
2. Berechne R_{k+1} aus R_k:

   entferne alle (z, z') für die
   $\delta(z, a)\ R_k\ \delta(z', a)$ nicht gilt für ein $a \in \Sigma$
3. Wiederhole solange, bis nichts mehr entfernt wird.

Konstruktion eines Minimalautomaten (IV)

„Tabellenfüllalgorithmus“

Idee: suche schrittweise nicht-erkennungsäquivalente Zustandspaare

(beginned bei Paaren aus End- und Nicht-Endzuständen)

- Tabelle, die Zustandspaare verwaltet
- Jedes Zustandspaar ist entweder „markiert“ oder „unmarkiert“.

  markiert = Zustände nicht in R_k enthalten
- Am Ende des Algorithmus gilt:

  $z \approx_M z'$ für alle unmarkierten Zustandspaare.

Algorithmus Minimalautomat (I)

Eingabe: DFA $M = (Z, \Sigma, \delta, z_0, E)$

Ausgabe: Mengen von erkennungsäquivalenten Zuständen

1. Entferne alle unerreichbaren Zustände aus M.
2. Stelle eine Tabelle aller (ungeordneten) Zustandspaare $(z, z') \in Z \times Z$ mit $z \neq z'$ auf.
3. Markiere alle Paare (z, z') mit $z \in E$ und $z' \notin E$ oder umgekehrt.
   (z, z' sind sicherlich nicht erkennungsäquivalent.)

Algorithmus Minimalautomat (II)

4. Wiederhole so lange, bis sich die Tabelle nicht mehr ändert:

   Für jedes unmarkierte Paar (z, z') in der Tabelle:

   Falls es $a \in \Sigma$ gibt, so dass $(\delta(z, a), \delta(z', a))$ markiert ist, so markiere (z, z').

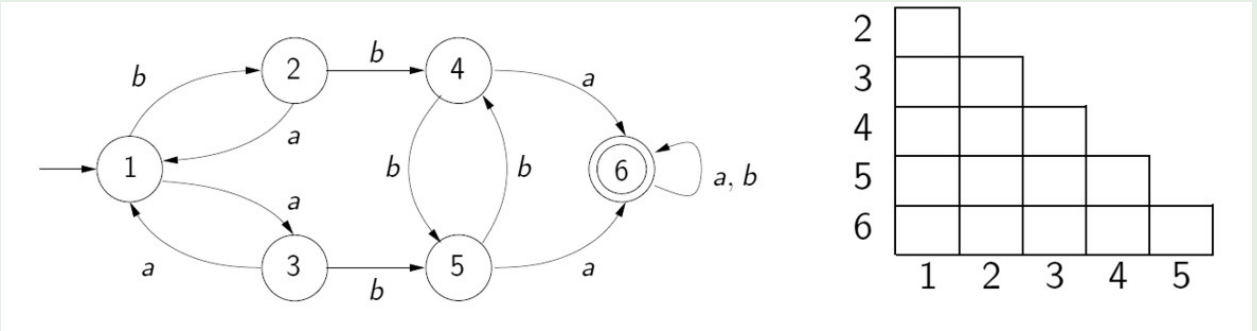
   (z, z' können nicht erkennungsäquivalent sein.)

Alle am Schluss unmarkierten Paare (z, z') sind erkennungsäquivalent.

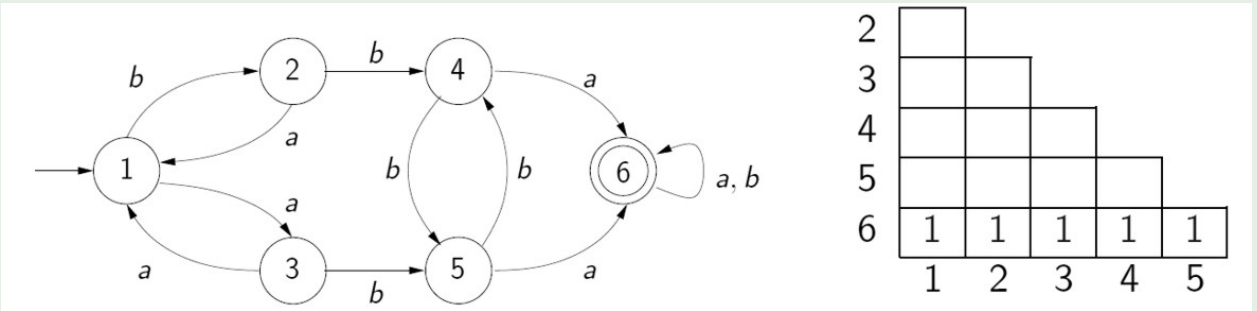
Algorithmus Minimalautomat (III)

Beispiel

Erstelle eine Tabelle aller Zustandspaare:



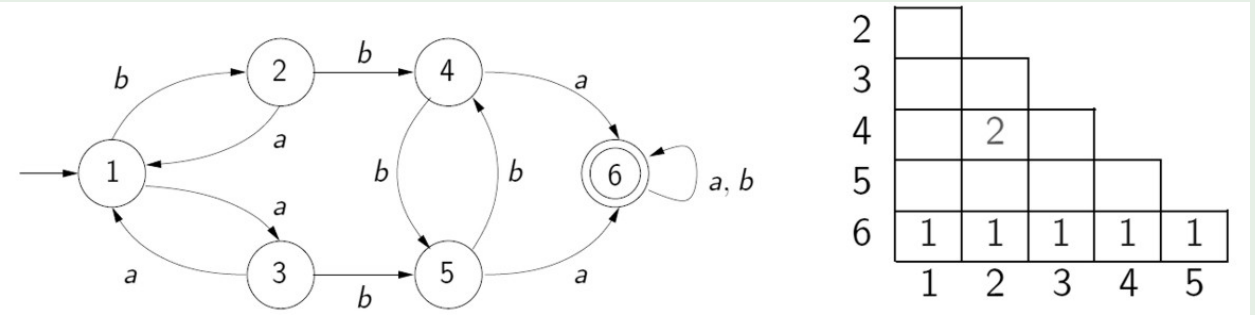
(1) Markiere Paare von Endzuständen und Nicht-Endzuständen



Algorithmus Minimalautomat (IV)

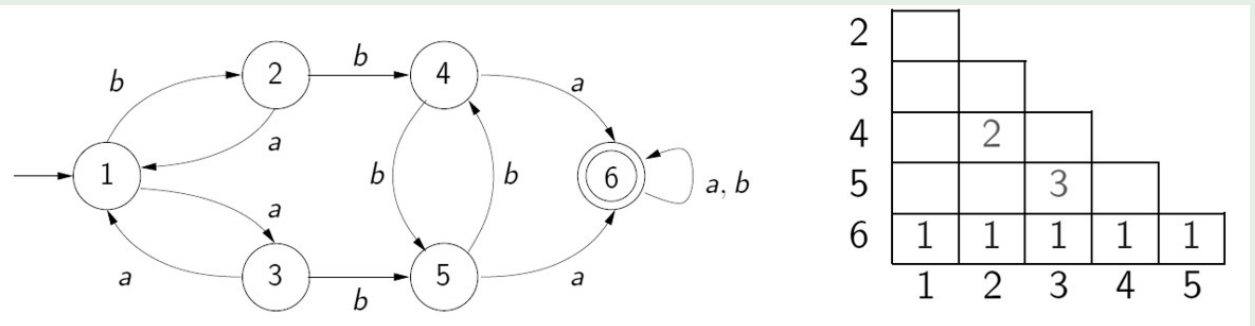
Beispiel (Fortsetzung)

(2) Markiere $\{2,4\}$ wegen $\delta(2,a) = 1, \delta(4,a) = 6$ und $\{1,6\}$ markiert



2					
3					
4		2			
5					
6	1	1	1	1	1
	1	2	3	4	5

(3) Markiere $\{3,5\}$ wegen $\delta(3,a) = 1, \delta(5,a) = 6$ und $\{1,6\}$ markiert

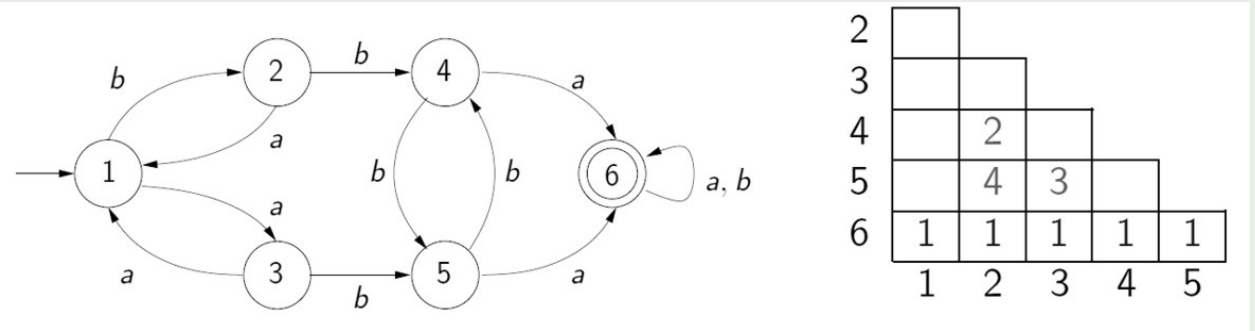


2					
3					
4		2			
5			3		
6	1	1	1	1	1
	1	2	3	4	5

Algorithmus Minimalautomat (V)

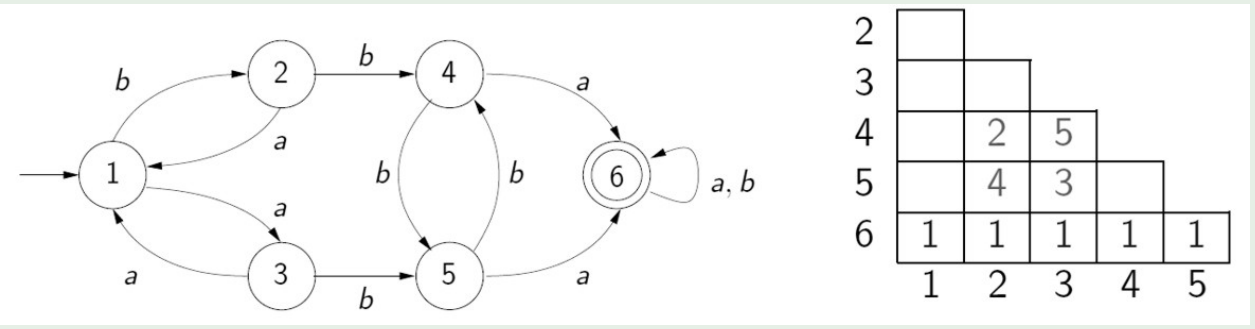
Beispiel (Fortsetzung)

(4) Markiere $\{2, 5\}$ wegen $\delta(2, a) = 1, \delta(5, a) = 6$ und $\{1, 6\}$ markiert



2					
3					
4		2			
5		4	3		
6	1	1	1	1	1
	1	2	3	4	5

(5) Markiere $\{3, 4\}$ wegen $\delta(3, a) = 1, \delta(4, a) = 6$ und $\{1, 6\}$ markiert

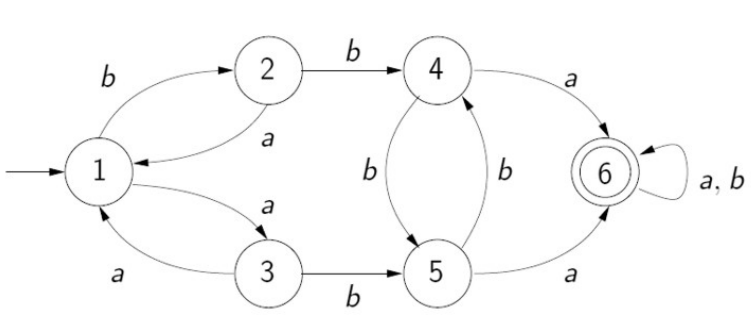


2					
3					
4		2	5		
5		4	3		
6	1	1	1	1	1
	1	2	3	4	5

Algorithmus Minimalautomat (VI)

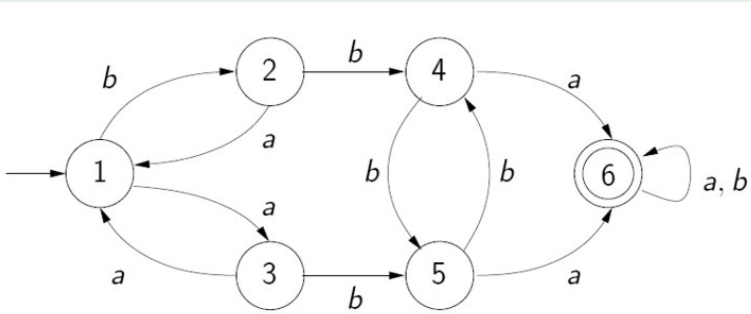
Beispiel (Fortsetzung)

(6) Markiere $\{1, 5\}$ wegen $\delta(1, a) = 3, \delta(5, a) = 6$ und $\{3, 6\}$ markiert



2					
3					
4		2	5		
5	6	4	3		
6	1	1	1	1	1
	1	2	3	4	5

(7) Markiere $\{1, 4\}$ wegen $\delta(1, a) = 3, \delta(4, a) = 6$ und $\{3, 6\}$ markiert

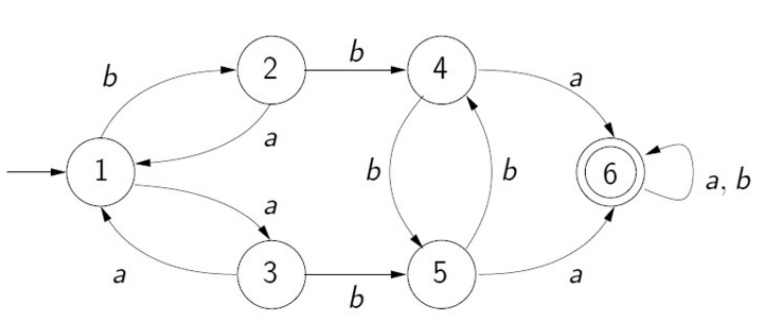


2					
3					
4	7	2	5		
5	6	4	3		
6	1	1	1	1	1
	1	2	3	4	5

Algorithmus Minimalautomat (VII)

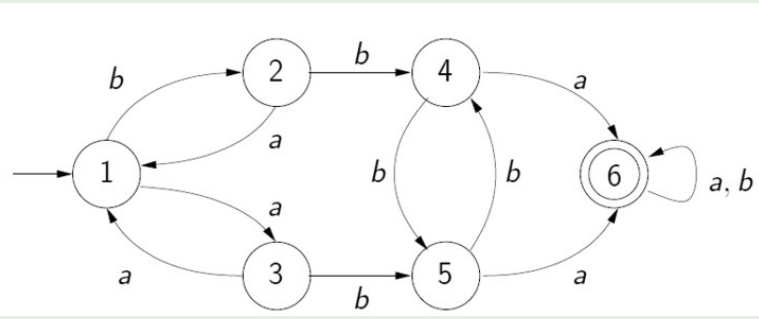
Beispiel (Fortsetzung)

(8) Markiere $\{1,3\}$ wegen $\delta(1,b) = 2, \delta(3,b) = 5$ und $\{2,5\}$ markiert



2					
3	8				
4	7	2	5		
5	6	4	3		
6	1	1	1	1	1
	1	2	3	4	5

(9) Markiere $\{1,2\}$ wegen $\delta(1,b) = 2, \delta(2,b) = 4$ und $\{2,4\}$ markiert

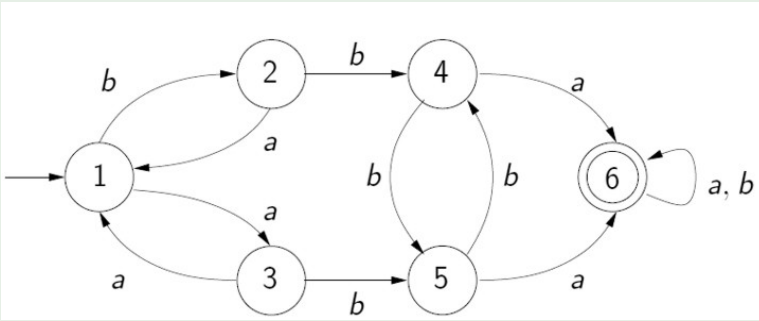


2	9				
3	8				
4	7	2	5		
5	6	4	3		
6	1	1	1	1	1
	1	2	3	4	5

Algorithmus Minimalautomat (VIII)

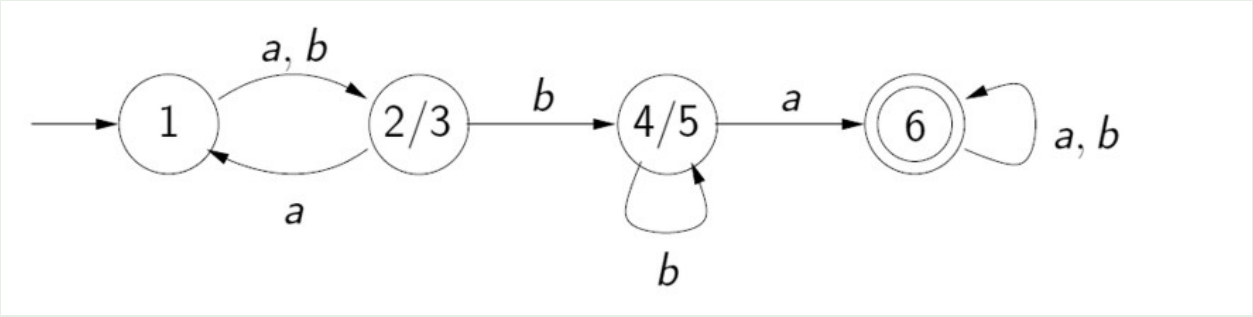
Beispiel (Fortsetzung)

Die verbleibenden Zustandspaare $\{2, 3\}$ und $\{4, 5\}$ können nicht mehr markiert werden, d. h. sie sind erkennungsäquivalent.



2	9				
3	8				
4	7	2	5		
5	6	4	3		
6	1	1	1	1	1
	1	2	3	4	5

Ergebnis:



Zusammenfassung