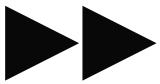
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Gliederung der Vorlesung

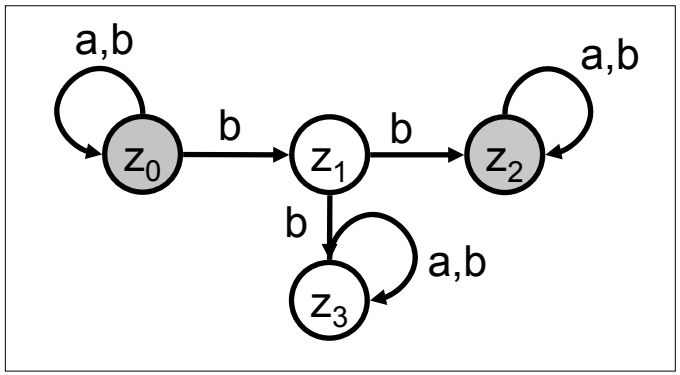
0. Grundbegriffe
1. Formale Sprachen/Automatentheorie
    1.1. Grammatiken
    1.2. Reguläre Sprachen
    1.3. Kontext-freie Sprachen
2. Berechnungstheorie
    2.1. Entscheidungsprobleme
    2.2. Berechenbarkeitsmodelle
    2.3. Die Churchsche These
    2.4. Unentscheidbarkeit
3. Komplexitätstheorie
    3.1. Nicht-deterministische Turing Maschinen
    3.1 Komplexitätsmaße
    3.2. Das P=NP? Problem

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

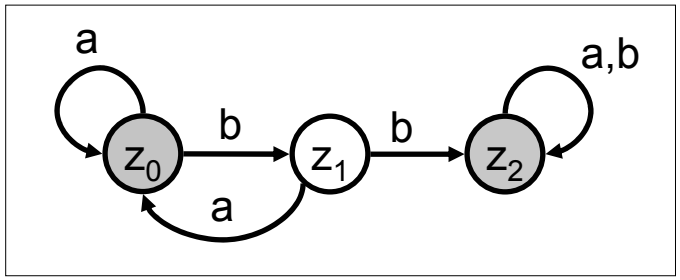
Reguläre Sprachen

Was haben deterministische endlichen Automaten mit nicht-deterministischen endlichen Automaten zu tun?

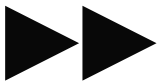
$L = \{ vbbw \mid v \in \{ a,b \}^*, w \in \{ a,b \}^* \}$



NFA für L



DFA für L



Reguläre Sprachen

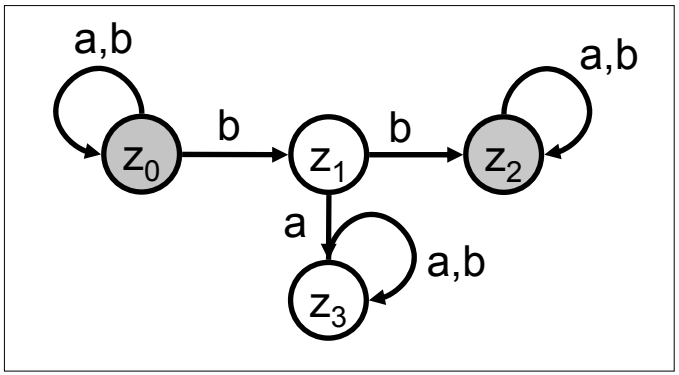
werden diskutieren, wie man aus zu einem gegebenen NFA einen äquivalenten DFA konstruieren kann

NFA B	DFA A
es gibt mehre Zustände z mit $z \in \delta^*(\{z_0\},w)$	es gibt genau einen Zustand z mit $z = \delta^*(z_0,w)$
w gehört zu L(B) gdw. es einen Zustand z mit $z \in \delta^*(\{z_0\},w)$, der ein Endzustand ist	w gehört zu L(A) gdw. der Zustand z mit $z = \delta^*(z_0,w)$ ein Endzustand ist

... beim Übergang von B zu A muss ein Zustand von A eine Menge von Zuständen von B „repräsentieren“

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen



infrage kommende Zustände für A

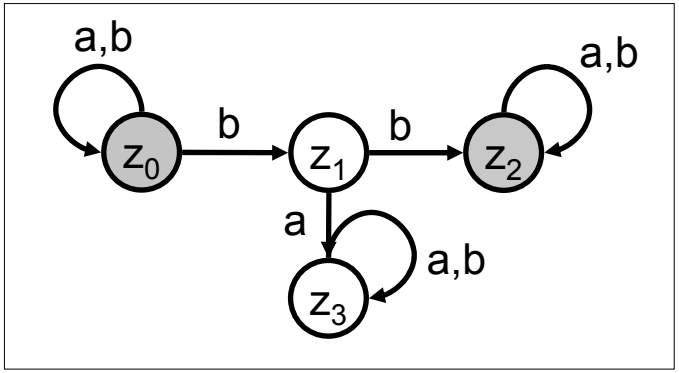
Name	Bedeutung
0	$\{ z_0 \}$
1	$\{ z_1 \}$
2	$\{ z_2 \}$
3	$\{ z_3 \}$
...	
012	$\{ z_0,z_1,z_2\}$
...	
0123	$\{ z_0,z_1,z_2,z_3 \}$

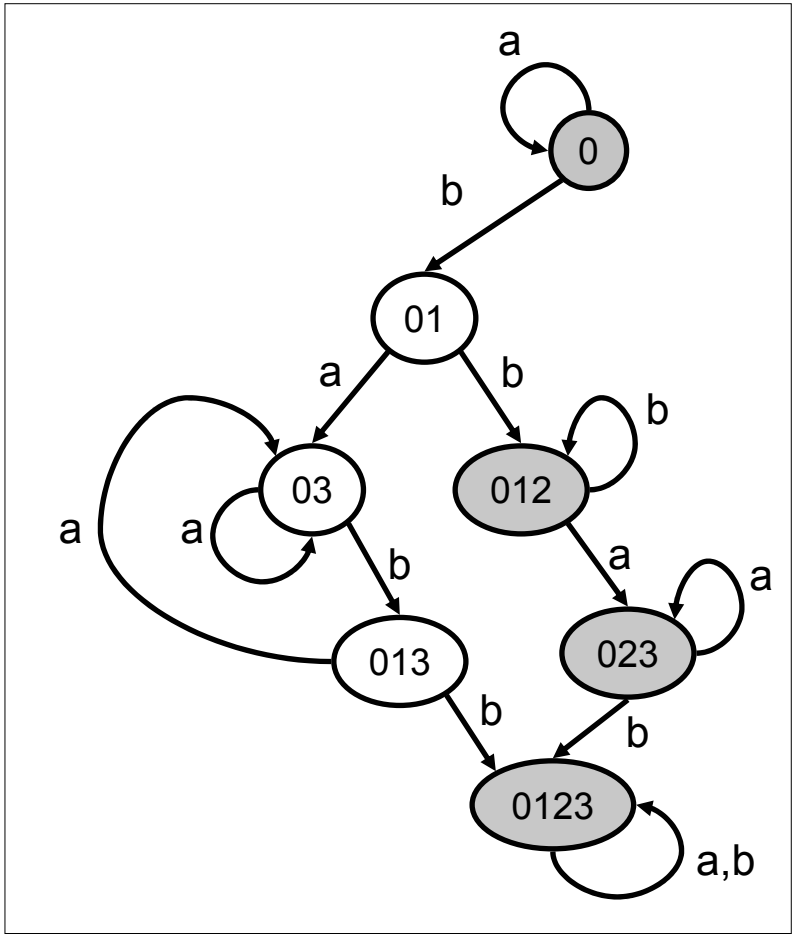
Anfangszustand von A

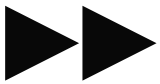
potentieller
Endzustand von A

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen





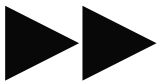


Reguläre Sprachen

allgemein

> Es sei B ein nicht-deterministischer endlicher Automat. Dann gibt es einen deterministischen endlichen Automat A, so daß gilt: L(A) = L(B).

m.a.W.: zu jedem NFA B gibt es einen DFA A, der dieselbe Sprache wie der NFA B akzeptiert



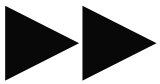
Reguläre Sprachen

zugrunde liegende Konstruktion

es sei B = $[Z,\Sigma',\delta,z_0,E]$ der gegebene NFA

bilde den gesuchten DFA A = $[Z',\Sigma',\delta',z'_0,E']$ für L(B) wie folgt

- setze $\Sigma' = \Sigma$
- für jedes $M \in 2^Z$ mit $M \neq \varnothing$ nimm einen Zustand z'_M in die Menge Z' auf
- setze $z'_0 = z_{\{z_0\}}$
- $E' = \{ z'_M \mid z'_M \in Z' , M \cap E \neq \varnothing \}$
- für jedes $z'_M \in Z'$ und alle $a \in \Sigma'$:
  - setze $\delta'(z_M,a) = z_M'$ mit $M' = \{ z' \mid z' \in \delta(z,a)$ für ein $z \in M \}$



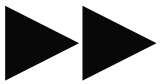
Reguläre Sprachen

Zwischenfazit

es sei Σ das zugrunde liegende Alphabet

Es sei $L \subseteq \Sigma^*$. Dann sind die folgenden Aussagen äquivalent:

(1) L ist eine reguläre Sprache.
(2) Es gibt eine reguläre Grammatik G mit L(G) = L.
(3) Es gibt einen NFA B mit L(B) = L.
(4) Es gibt einen DFA A mit L(A) = L.



Reguläre Sprachen

Beobachtungen

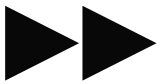
- bei der „Übersetzung“ eines NFA mit n Zuständen kann ein äquivalenter DFA entstehen, der $2^n - 1$ viele Zustände hat

... Liegt das am verwendeten Algorithmus oder steckt da mehr dahinter?

- bei der „Übersetzung“ eines NFA entsteht offenbar nicht immer der kleinste äquivalente DFA

... siehe Beispiel

... Wie kann man dieses Problem umgehen?



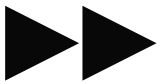
Reguläre Sprachen

Illustration

Zu jedem n > 0 gibt es eine reguläre Sprache L_n mit folgenden Eigenschaften:

- es gibt einen NFA B mit L(B) = L_n, der n + 2 Zustände hat
- jeder DFA A mit L(A) = L_n hat mindestens 2^n Zustände

... die Problematik, daß bei der Übersetzung eines NFA ein deutlich größerer DFA entstehen kann, kann nicht durch die Wahl eines „cleveren“ Algorithmus überwunden werden



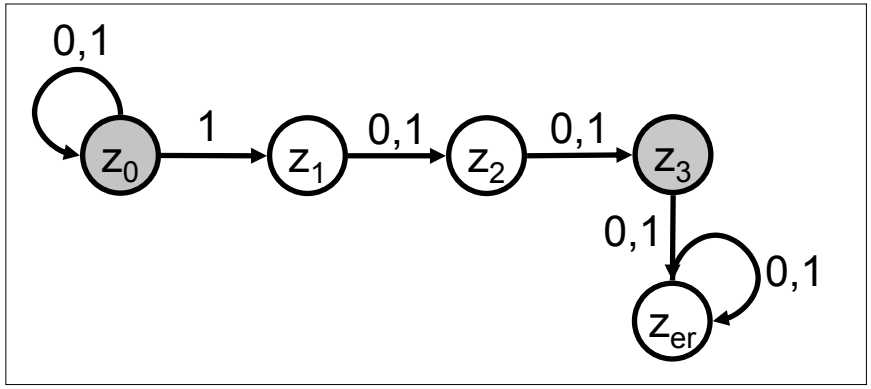
Reguläre Sprachen

Ein Beispiel

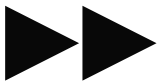
für jedes $n > 0$ sei $L_n = \{ w1v \mid w,v \in \{0,1\}^*, |v| = n - 1 \}$

... m.a.W. Wörter in L_n haben eine 1 als n-ten Buchstaben von hinten

NFA für L_3



ÜA: „übersetzen“ Sie diesen NFA in einen äquivalenten DFA



Reguläre Sprachen

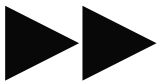
Warum muß ein DFA für L_n mindestens 2^n Zustände haben?

zugrunde liegende Beobachtung:

es gibt 2^n viele Wörter in { 0,1 }* der Länge n

Annahme:

es seien n > 0 gegeben und A = $[Z,\Sigma',\delta,z_0,E]$ ein DFA mit L(A) = L_n, der weniger als 2^n Zustände hat



Reguläre Sprachen

weil A weniger als 2^n viele Zustände hat, gibt es Wörter u und u‘ mit:

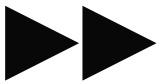
- $|u| = |u'| = n$
- $u \neq u'$
- $\delta^*(z_0,u) = \delta^*(z_0,u')$

weil u ≠ u‘ gilt, gibt es Wörter w, v und v‘, so daß gilt:

- $|w| > 0$
- $u = w1v$
- $u' = w0v'$

dann gilt für die Wörter uw = w1vw und u'w = w0v'w:

- $\delta^*(z_0,uw) = \delta^*(\delta^*(z_0,u),w) = \delta^*(\delta^*(z_0,u'),w) = \delta^*(z_0,u'w)$



Reguläre Sprachen

dann gilt für die Wörter uw = w1vw und u'w = w0v'w :

- $\delta^*(q_0,uw) = \delta^*(\delta^*(q_0,u),w) = \delta^*(\delta^*(q_0,u'),w) = \delta^*(q_0,u'w)$

also gilt:

- $uw \in L(A)$ und $u'w \in L(A)$ <u>oder</u>
- $uw \notin L(A)$ und $u'w \notin L(A)$

weil w1v = u und w0v' = u' und |u| = |u'| gilt:

- |1vw| = |0v'w| = n

und damit gilt

- $uw = w1vw \in L_n$ <u>und</u> $u'w = w0v'w \notin L_n$

also ist $L(A) \neq L_n$, ein Widerspruch

Minimierung von Automaten

den DFA, der bei der „Übersetzung“ eines NFA entsteht bzw. selbstständig entworfen wurde, „kleiner“ machen

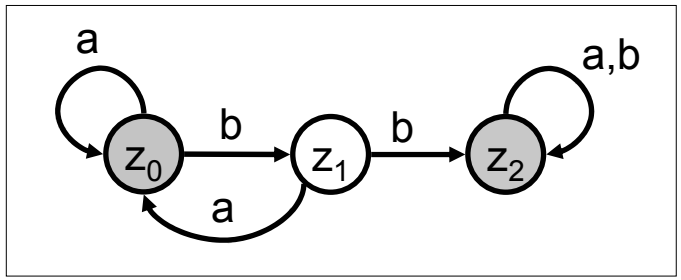
Maß für die Größe eines DFA:

die Anzahl der Zustände

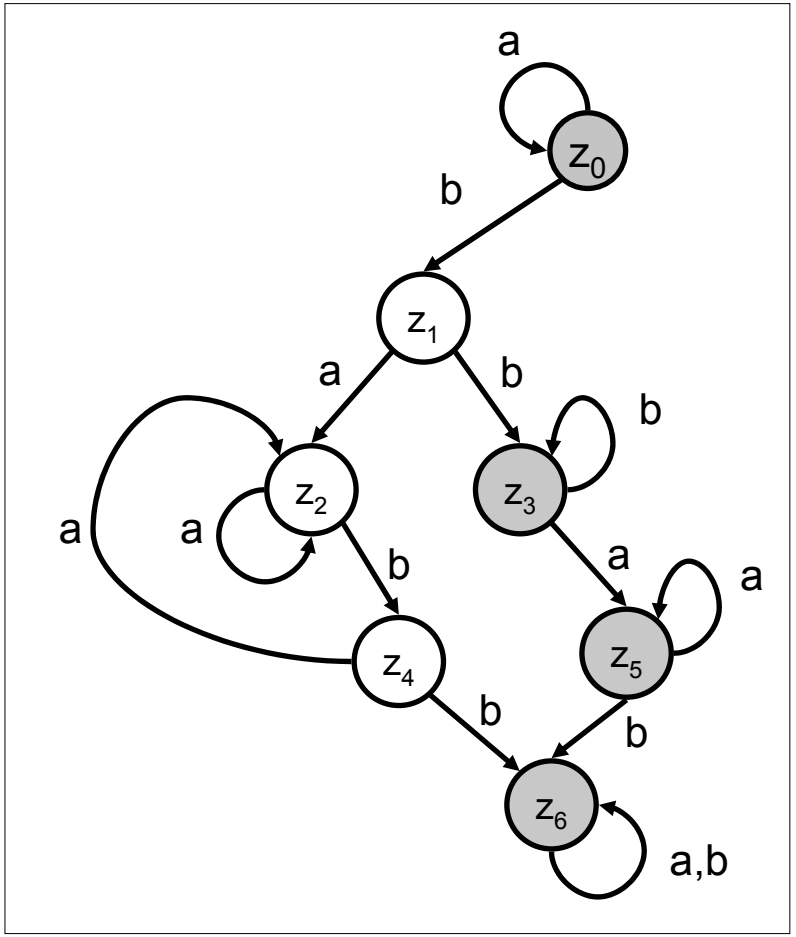
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen

$L = \{ vbbw \mid v \in \{ a,b \}^*, w \in \{ a,b \}^* \}$

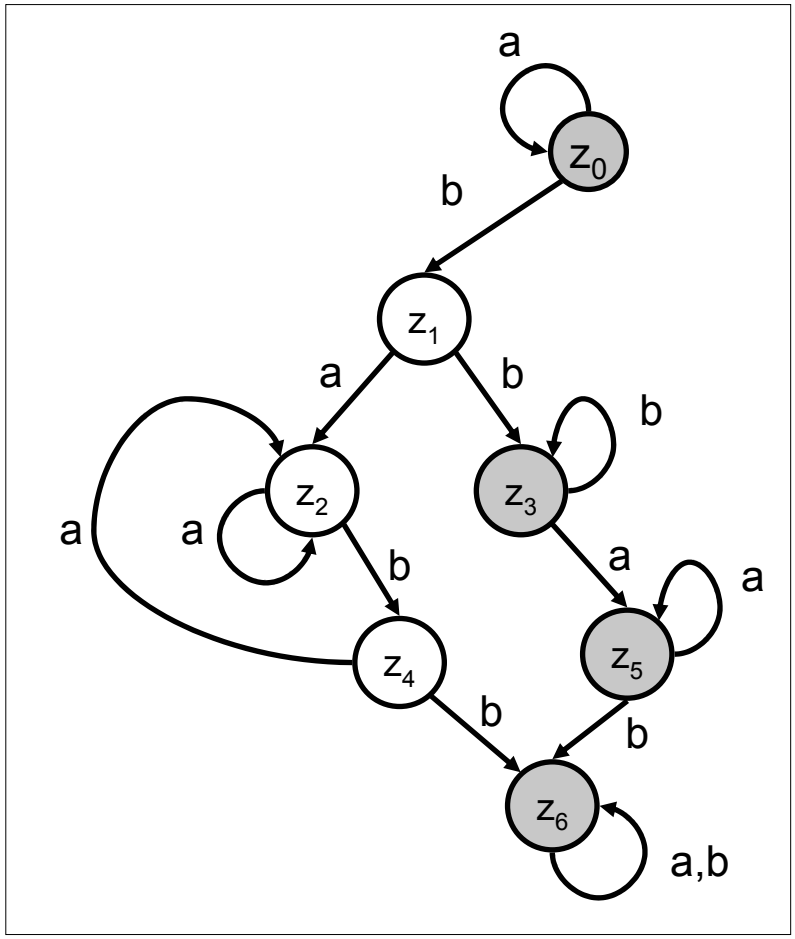


ein DFA für L



ein anderer DFA für L

Reguläre Sprachen

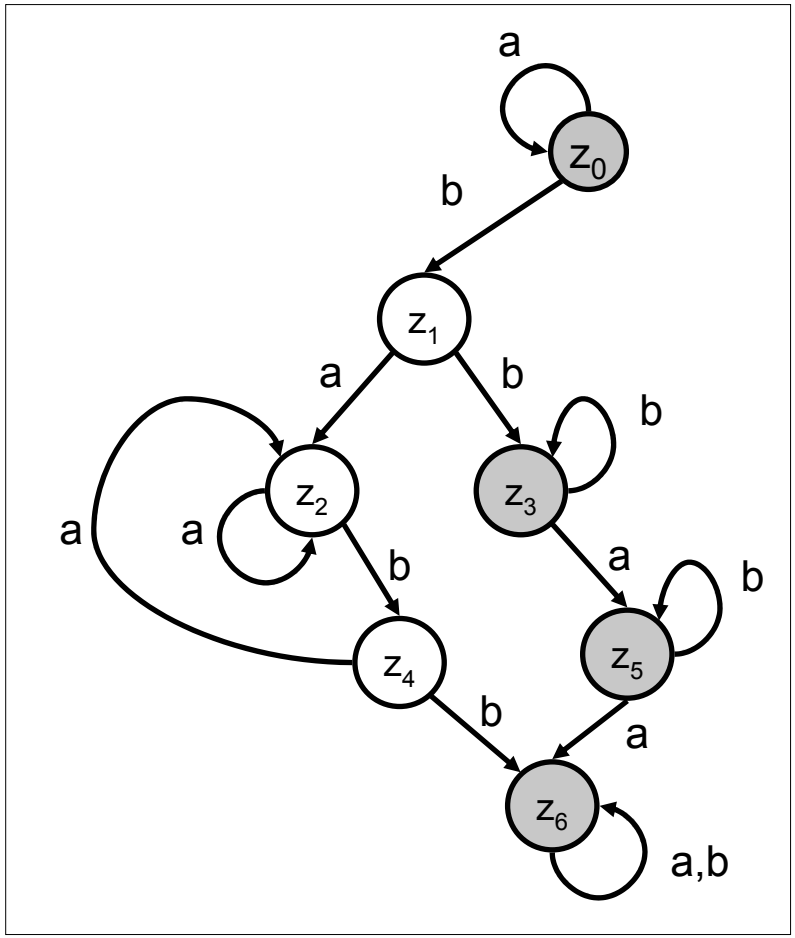


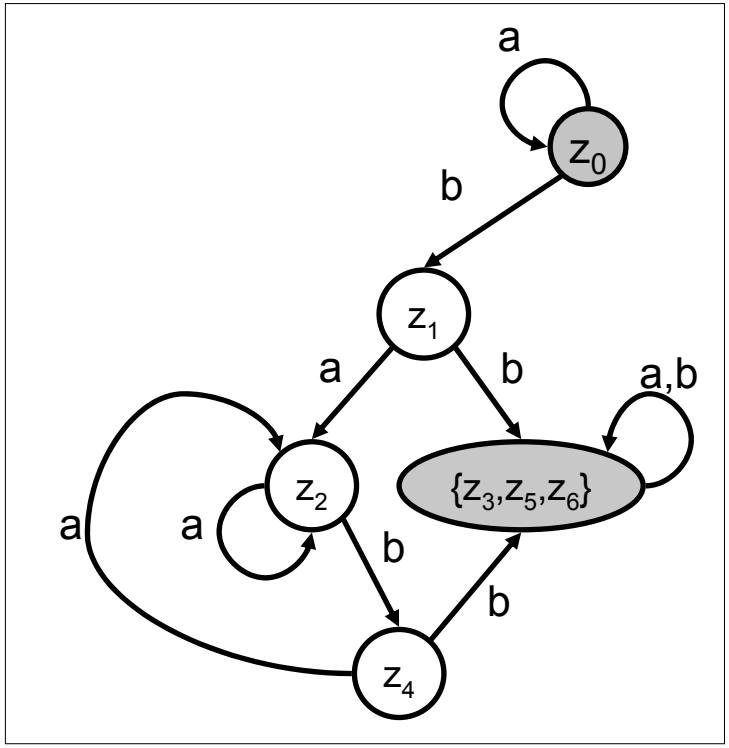
offenbar unterscheiden sich die Zustände z_3, z_4 und z_5 nicht

... es gilt für alle $w \in \{ a,b \}^*$:

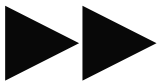
- $\delta(z_3,w) \in E$
- $\delta(z_4,w) \in E$
- $\delta(z_5,w) \in E$

Reguläre Sprachen





... alle Mengen äquivalenter Zustände finden



Reguläre Sprachen

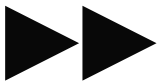
zentraler Begriff

es sei $A = [Z,\Sigma,\delta,z_0,E]$

es seien z, z‘ zwei Zustände in Z

z und z‘ heißen äquivalent, wenn für alle Wörter $w \in \Sigma^*$ gilt:

1. wenn $\delta^*(z,w) \in E$, so $\delta^*(z',w) \in E$
2. wenn $\delta^*(z,w) \notin E$, so $\delta^*(z',w) \notin E$



Reguläre Sprachen

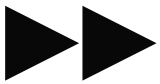
zentraler Begriff (/* handhabbare Einschränkung */)

es sei $A = [Z,\Sigma,\delta,z_0,E]$

es seien z, z‘ zwei Zustände in Z, es sei $k \geq 0$

z und z‘ heißen k-äquivalent, wenn für alle Wörter $w \in \Sigma^*$ mit $|w| \leq k$ gilt:

1. wenn $\delta^*(z,w) \in E$, so $\delta^*(z',w) \in E$
2. wenn $\delta^*(z,w) \notin E$, so $\delta^*(z',w) \notin E$



Reguläre Sprachen

grundlegender Zusammenhang

es sei $A = [Z,\Sigma,\delta,z_0,E]$

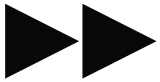
es seien z, z‘ zwei Zustände in Z, es sei $k \geq 0$

z und z‘ sind (k+1)-äquivalent, wenn für alle $a \in \Sigma$ gilt, daß die Zustände $\delta(z,a)$ und $\delta(z',a)$ k-äquivalent sind.

„spannender“ Teil der Begründung

es sei w = aw‘ mit |w| = k; dann gilt:

- $\delta^*(z,aw) = \delta^*(\delta(z,a),w)$
- $\delta^*(z',aw) = \delta^*(\delta(z',a),w)$



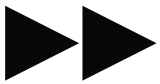
Reguläre Sprachen

Minimierungsalgorithmus (/* Teil 1 */)

es sei $A = [Z,\Sigma,\delta,z_0,E]$

Schritt 0:
Setze $M = E$, $M' = Z \setminus E$ und $K_0 = \{ M, M' \}$. Gehe zu Schritt 1.

Schritt $k + 1$ ($k \geq 0$):
(/* Es sei K_k eine Klasseneinteilung von Z in Mengen von k-äquivalenten Zuständen. */)
- Bestimme eine Klasseneinteilung K_{k+1} von Z in Mengen von (k+1)-äquivalenten Zuständen.
- Falls $K_{k+1} = K_k$, so stoppe.
- Andernfalls gehe zu Schritt $k + 2$.



Reguläre Sprachen

Einfache Beobachtungen (/* Teil 1 */)

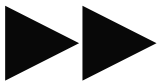
es sei $A = [Z,\Sigma,\delta,z_0,E]$

es sei K_k eine Klasseneinteilung von Z in Mengen von k-äquivalenten Zuständen; es seien $z, z' \in Z$

Dann gilt:

- wenn z und z‘ (k+1)-äquivalent sind, so gehören z und z‘ zu ein und derselben Menge in K_k
- wenn z und z‘ (k+1)-äquivalent sind, so gehören für jedes $a \in \Sigma$ die Zustände $\delta(z,a)$ und $\delta(z',a)$ zu ein und derselben Menge in K_k

... algorithmische Idee + Begründung, daß das Verfahren terminiert



Reguläre Sprachen

Einfache Beobachtungen (/* Teil 2 */)

es sei $A = [Z,\Sigma,\delta,z_0,E]$

es seien K_k eine Klasseneinteilung von Z in Mengen von k-äquivalenten Zuständen und K_{k+1} eine Klasseneinteilung von Z in Mengen von (k+1)-äquivalenten Zuständen

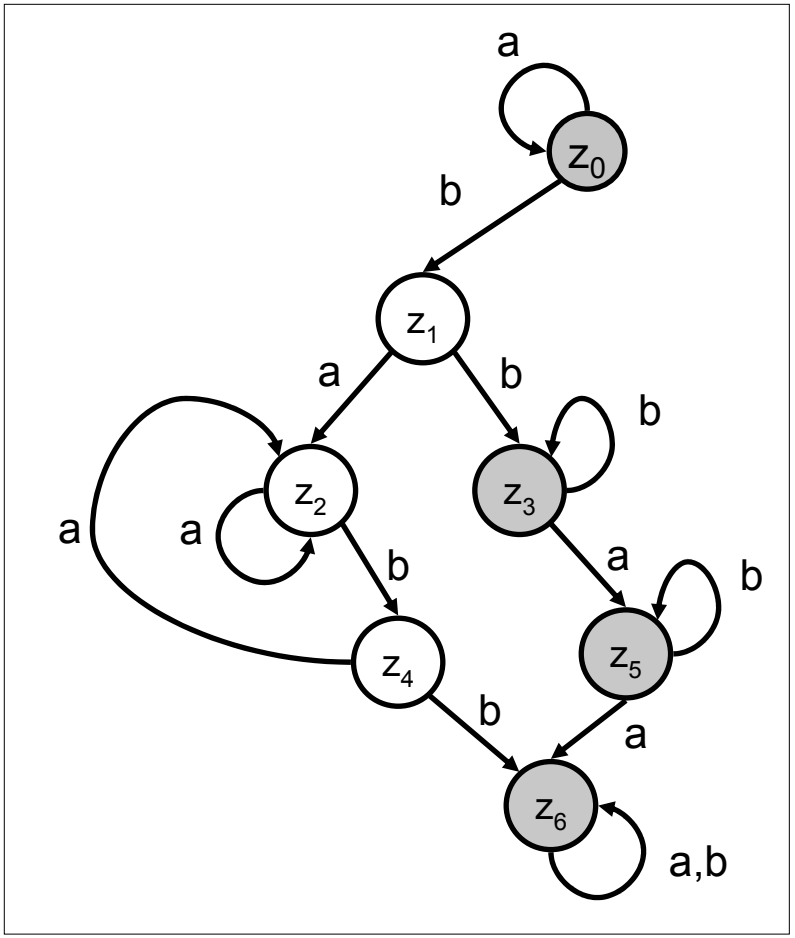
Falls $K_k = K_{k+1}$, so gilt:

- wenn z und z‘ k-äquivalent sind, so sind z und z‘ auch (k+n)-äquivalent für alle $n \geq 1$ (/* und damit äquivalent */)

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen

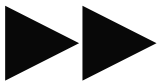
Beispiel



$K_0 = \{ \{ z_0, z_1, z_2, z_4 \}, \{ z_3, z_5, z_6 \} \}$

$K_2 = \{ \{ z_0, z_2 \}, \{ z_1, z_4 \}, \{ z_3, z_5, z_6 \} \}$

$K_3 = \{ \{ z_0, z_2 \}, \{ z_1, z_4 \}, \{ z_3, z_5, z_6 \} \}$



Reguläre Sprachen

Minimierungsalgorithmus (/* Teil 2 */)

es sei $A = [Z,\Sigma,\delta,z_0,E]$

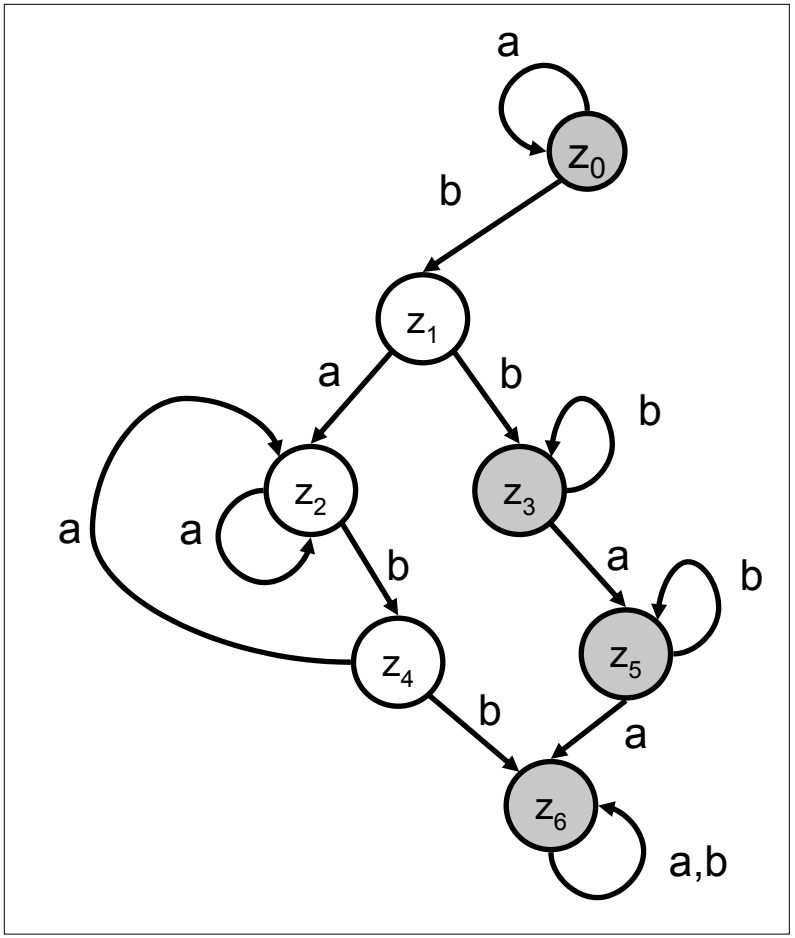
es seien K eine Klasseneinteilung von Z in äquivalente Zustände

der gesuchte minimale DFA $A' = [Z',\Sigma',\delta',z'_0,E']$ für L(A) ergibt sich wie folgt:

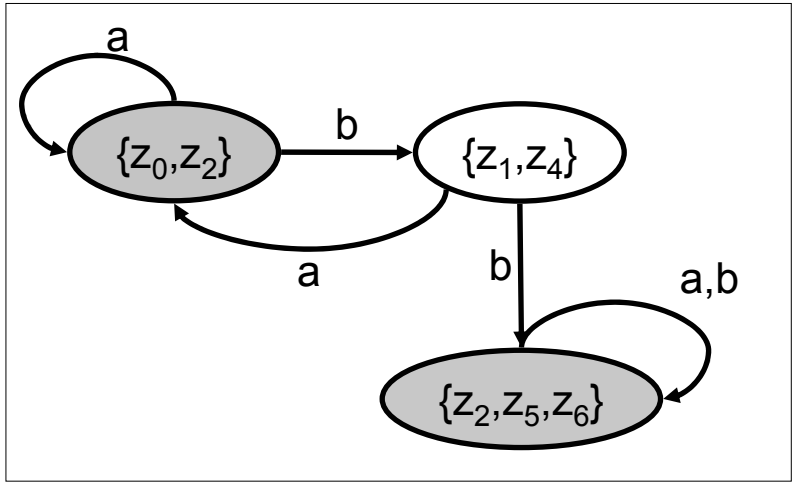
- setze $\Sigma' = \Sigma$
- für jedes $M \in K$ nimm einen Zustand z'_M in die Menge Z' auf
- setze $z'_0 = z_M$ mit $z_0 \in M$
- $E' = \{ z'_M \mid z'_M \in Z' , M \cap E \neq \emptyset \}$
- für jedes $z'_M \in Z'$ und alle $a \in \Sigma'$:
  - setze $\delta'(z_M,a) = z_M'$ mit $M' = \{ z' \mid z' \in \delta(z,a)$ für ein $z \in M \}$

Reguläre Sprachen

Beispiel



$K = \{ \{ z_0, z_2 \}, \{ z_1, z_4 \}, \{ z_3, z_5, z_6 \} \}$



Eigenschaften des Minimierungsalgorithmus

es seien L eine reguläre Sprache und A = [Z,Σ,δ,z_0,E] ein DFA für L

- das Ergebnis der Anwendung des Minimierungsalgorithmus auf A liefert einen minimalen DFA A‘ für L (/* minimal in der Anzahl der Zustände */)
- geeignet implementiert benötigt der Minimierungsalgorithmus $O(n^3)$ viele Schritte um A‘ zu bestimmen (/* n ist das Maximum von card(Z) und card(Σ) */)

ein wenig ungenau ... es sind zunächst die vom Startzustand z_0 nicht erreichbaren Zustände zu entfernen