Gliederung

Minimierungsalgorithmus

▶ Problemstellung

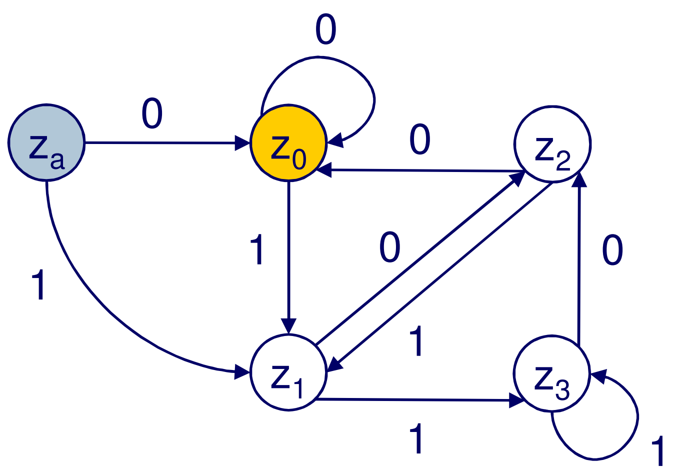
- Gegeben seien eine Sprache L und ein endlicher Automat A mit L(A) = L.
- Frage: Gibt es einen kleineren (bzgl. der Anzahl der Zustände) endlichen Automaten A° mit L(A°) = L ?

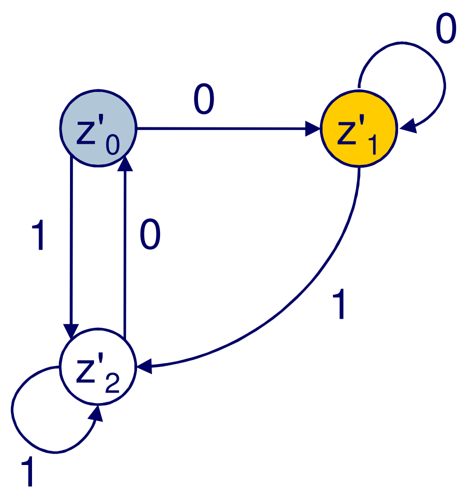
... relevante Fragestellung, weil ...

- *man A° evtl. besser versteht als A*
- *man Speicherplatz und/oder Rechenaufwand spart, wenn man A° anstelle von A benutzt, und*
- *über die Verkleinerung von Automaten später eine Möglichkeit geschaffen wird, zu prüfen, ob zwei Automaten **dieselbe Sprache** akzeptieren.*

ein Beispiel

- Sei $\Sigma = \{ 0,1 \}$.
- Sei $L = \{ w \in \Sigma^* \mid$ w ist Binärdarstellung einer durch 4 teilbaren natürlichen Zahl $\}$.   (Führende Nullen seien erlaubt.)
- A und A‘ sind endliche Automaten mit L(A) = L bzw. L(A‘) = L .





Das kann man daran sehen, dass beide Automaten genau das Wort 0 und alle 0-1-Folgen mit Suffix 00 akzeptieren.

▶ Algorithmische Aufgabe

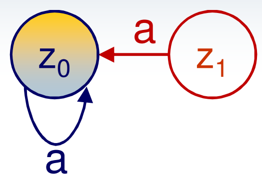
- zulässige Eingabe: ein endlicher Automat A, in dem jeder Zustand vom Startzustand aus erreichbar ist

- zulässige Ausgabe: ein endlicher Automat A° mit folgenden zwei Eigenschaften:

  (1) L(A°) = L(A)
  (2) Es gibt keinen endlichen Automaten A‘ mit L(A‘) = L(A), der weniger Zustände als A° hat.

Hinweis: *Zustände, die vom Startzustand aus nicht erreichbar sind, kann man bedenkenlos streichen (gleiche Sprache, weniger Zustände). Dies wird ab jetzt beim Minimieren stets als erledigt vorausgesetzt!*

- Beispiel:



- Einschub: alte und neue Sprachen rund um einen Automaten

Erinnerung: Die Menge $K_A(z)$ aller Wörter, die den Automaten von z_0 nach z bringen, wurde definiert als $K_A(z) = \{ w \in \Sigma^* \mid \delta^*(z_0,w) = z \}$.

Die Menge L(A,z) bzw. $L_A(z)$ aller Wörter, die den Automaten von z in einen akzeptierenden Zustand bringen, ist $L_A(z) := \{ w \in \Sigma^* \mid \delta^*(z,w) \in F \}$.

Zu $L_A(z)$ gehören genau alle Wörter, die A akzeptieren würde, wenn z der Startzustand wäre.

Erinnerung: Die Menge L(A) aller Wörter, die den Automaten von z_0 in einen akzeptierenden Zustand bringen, ist $L(A) = \{ w \in \Sigma^* \mid \delta^*(z_0,w) \in F \} = L_A(z_0)$.

Also ist $L(A) = L_A(z_0)$.

Merkregel:
$K_A(z)$: *nach z **k**ommen*
$L_A(z)$: ***l**eaving z / z ver**l**assen*

Einschub: Sprachen rund um einen Automaten – Zusammenhänge

$$\begin{tikzcd} z_0 \arrow[r, Rightarrow, "K_A(z)"] \arrow[rr, Rightarrow, bend right, "L(A)"'] & z \arrow[r, Rightarrow, "L_A(z)"] & F \end{tikzcd}$$

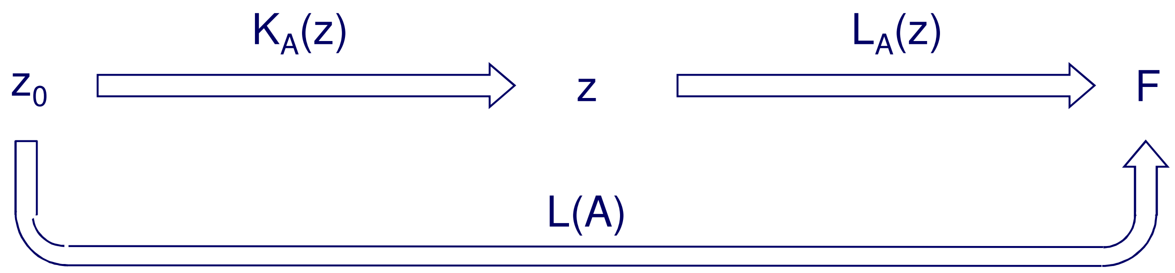
- $L(A) = \bigcup_{z \in F} K_A(z)$
- $L(A) = L_A(z_0)$

anstatt z_0

- $L_A(z)$ ist die Sprache des Automaten A‘ mit A‘ = $[Z,\Sigma, z ,F,\delta]$
- $K_A(z)$ ist die Sprache des Automaten A‘ mit A‘ = $[Z,\Sigma,z_0, \{z\} ,\delta]$

anstatt F

▶ Einschub: Sprachen rund um einen Automaten – Verständnistest: Frage



Welche Zusammenhänge bestehen zwischen

- $K_A(z) \circ L_A(z)$ *und*
- $L(A)$ *?*

▶ Einschub: Sprachen rund um einen Automaten – Verständnistest: Antworten

Welche Zusammenhänge bestehen zwischen
- $K_A(z) \circ L_A(z)$ *und*
- $L(A)$ *?*

- Die Wörter in $K_A(z) \circ L_A(z)$ „führen vom Startzustand über z zu einem akzeptierenden Zustand“, gehören also zu L(A):
  $K_A(z) \circ L_A(z) \subseteq L(A)$

- $L(A) = \bigcup_{z \in Z} K_A(z) \circ L_A(z)$ (→ Sonderfälle überlegen: z=z0, $z \in F$)

- Wenn $K_A(z) \circ L_A(z)$ leer ist, …
  - ist $K_A(z)$ leer, also vom Startzustand aus z unerreichbar, oder
  - $L_A(z)$ ist leer, also von z aus F unerreichbar,

  (oder beides).

Minimierungsalgorithmus

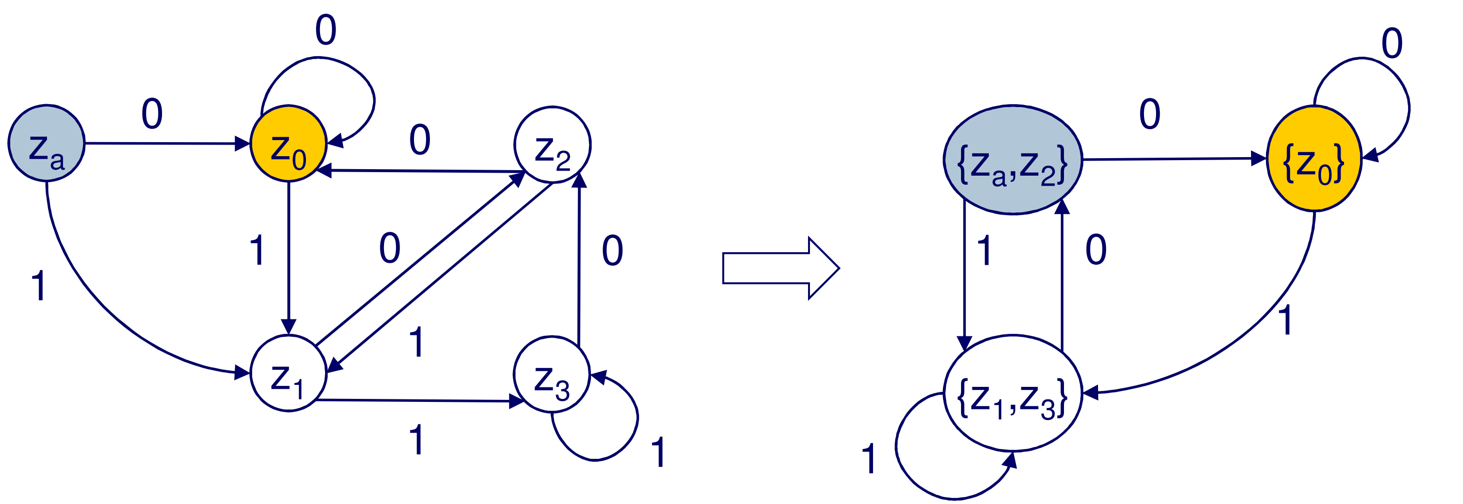
- Zustandsäquivalenz

Die Zustände z und z‘ heißen äquivalent, kurz $z \equiv z'$, gdw. $L_A(z) = L_A(z')$ gilt.

- Grundidee der Minimierung – Vorschau

  - Bestimme Zustände, die „äquivalent“ sind, und ersetze diese jeweils durch einen Zustand.

→ *Automat? (Ja!)*



- Problem: *Wenn $L_A[z]$ <u>unendlich</u> ist, wie kann man Äquivalenz <u>mit endlichem Aufwand</u> prüfen?*

- mit endlichem Aufwand prüfbar: k-Zustandsäquivalenz

  - Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat;
  - es seien $z, z' \in Z$;
  - es sei $k \in N$.

> Die Menge $L^k_A(z)$ aller Wörter der Länge $\leq k$, die akzeptiert würden, wenn z der Startzustand wäre, ist definiert als:
> $L^k_A(z) = \{ w \in \Sigma^* \mid \delta^*(z,w) \in F \text{ und } |w| \leq k \}$.

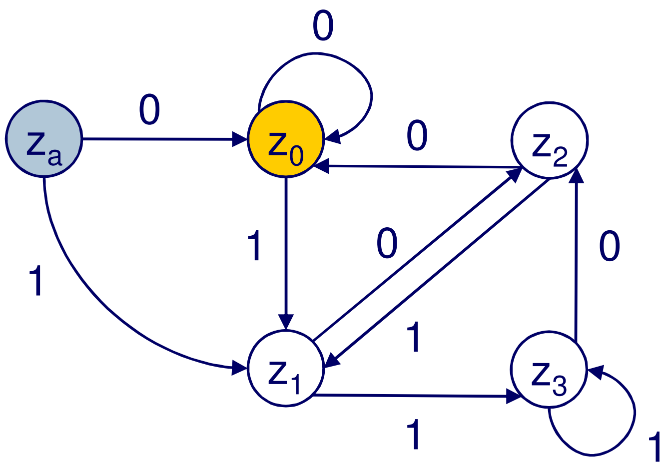
> Die Zustände z und z' heißen k-äquivalent, kurz $z \equiv_k z'$, gdw. $L^k_A(z) = L^k_A(z')$ gilt.

- grundlegender Zusammenhang

(1) Die Zustände z und z' sind genau dann äquivalent, wenn sie für alle k = 0,1, 2, ... k-äquivalent sind.

... weil jedes Wort irgendeine endliche Länge hat.

▶ Beispiele für k-Zustandsäquivalenz



Klasseneinteilung bzgl. 0-Äquivalenz:

$K_0 = \{ \ \{ z_0 \}, \ \{ z_1,z_2,z_3,z_a \} \}$

L^0_A: $\{\varepsilon\}$ $\emptyset$

Klasseneinteilung bzgl. 1-Äquivalenz:

$K_1 = \{ \ \{ z_0 \}, \ \{ z_1,z_3 \}, \{ z_2,z_a \} \}$

L^1_A: $\{\varepsilon,0\}$ $\emptyset$ $\{0\}$

Klasseneinteilung bzgl. 2-Äquivalenz:

$K_2 = \{ \ \{ z_0 \}, \ \{ z_1,z_3 \}, \{ z_2,z_a \} \}$

L^2_A: $\{\varepsilon,0,00\}$ $\{00\}$ $\{0,00\}$

3-äquivalent? 4-äquivalent? usw. ...
immer noch unendlich viele Prüfschritte?

Minimierungsalgorithmus

▶ weitere Zusammenhänge

- Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat;
- es seien $z,z' \in Z$;
- es sei $k \in N$.

(2) Die Zustände z und z‘ sind 0-äquivalent gdw. entweder beide in F (d.h. $L^0_A[z] = L^0_A[z'] = \{\varepsilon\}$) oder beide nicht in F (d.h. $L^0_A[z] = L^0_A[z'] = \emptyset$).

klar

(3) Wenn die Zustände z und z‘ (k+1)-äquivalent sind, so sind die die Zustände z und z‘ auch k-äquivalent.

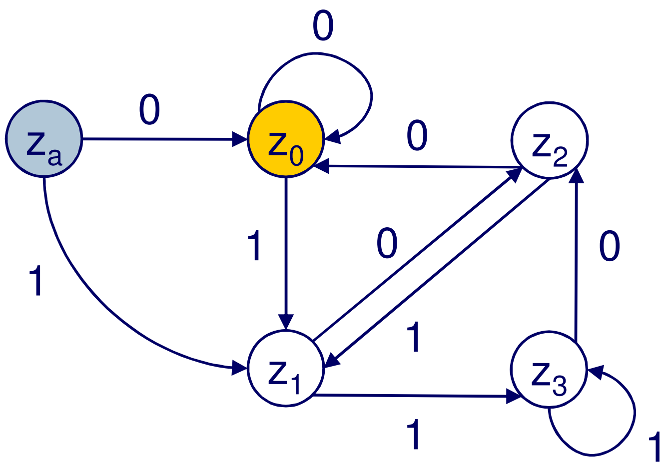
... weil die $\leq k$ langen Wörter auch zu den $\leq k+1$ langen gehören

(4) Die Zustände z und z‘ sind (k+1)-äquivalent gdw. für alle $x \in \Sigma$ gilt, dass die Zustände $\delta(z,x)$ und $\delta(z',x)$ k-äquivalent sind.

*... weil alles was ich in max. k+1 Schritten erreichen kann, allem entspricht, was ich in einem ersten und max. k weiteren erreiche.**

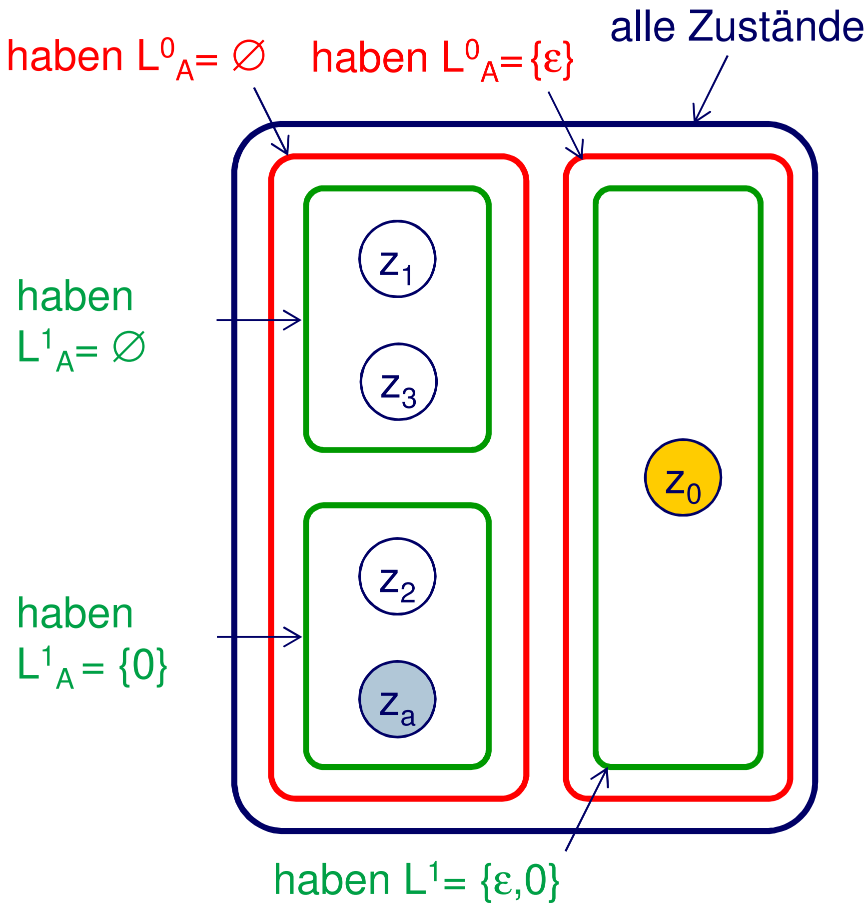
*) streng mathematisch mit mehr Formeln, aber nicht tiefsinniger.

- (3) $\Rightarrow$ (k+1)-Äquivalenz ist feiner als k-Äquivalenz
  hier im Automatenbeispiel – für k=0 …



$Kl_0 = \{\ \{ z_0 \},\ \{ z_1,z_2,z_3,z_a \}\ \}$
$L^0{}_A$: $\{\varepsilon\}$ $\varnothing$

$Kl_1 = \{\ \{ z_0 \},\ \{ z_1,z_3 \},\ \{ z_2,z_a \}\ \}$
$L^1{}_A$: $\{\varepsilon,0\}$ $\varnothing$ $\{0\}$



- Algorithmisch nutzbarer Zusammenhang:

  - Seien $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat,
  - $k \in N_0$,
  - K_k und K_{k+1} die durch $\equiv_k$ und $\equiv_{k+1}$ induzierten Klasseneinteilungen
  - und K die durch $\equiv$ induzierte Klasseneinteilung.
  - Dann gilt: $\boxed{K_k = K \text{ gdw. } K_k = K_{k+1}.}$

$\Rightarrow$: *Wegen (2): (k+1)-äquivalent $\rightarrow$ k-äquivalent.*
Mit $K_k = K$: k-äquivalent $\rightarrow$ äquivalent.
Wegen(1): äquivalent $\rightarrow$ k-äquivalent.
Also: (k+1)-äquivalent $\leftrightarrow$ k-äquivalent.

$\Leftarrow$: *Aus $z_1 \equiv_{k+1} z_2$ folgt mit (4) auch $\delta(z_1,a) \equiv_k \delta(z_2,a)$ für alle $a \in \Sigma$.*
Wegen $K_k = K_{k+1}$ folgt $\delta(z_1,a) \equiv_{k+1} \delta(z_2,a)$ für alle $a \in \Sigma$.
Zusammen mit $z_1 \equiv_{k+1} z_2$ folgt dann wegen (4) auch $z_1 \equiv_{k+2} z_2$ und analog $z_1 \equiv_{k+3} z_2$, $z_1 \equiv_{k+4} z_2$ usw., also wegen (2) auch $z_1 \equiv z_2$.

- Abbruchkriterium: Aufhören, sobald keine echte Verfeinerung mehr: $K_k = K_{k+1}$!

▶ Minimierungsalgorithmus (Teil 1)

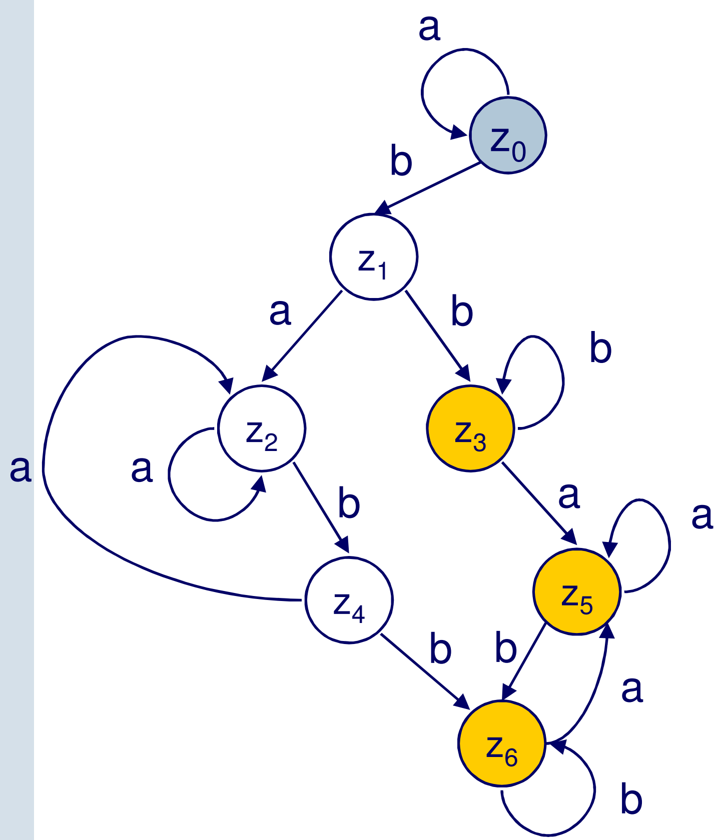
- Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat.

k:=0.
/* Bestimmung der durch $\equiv_0$ induzierten Klasseneinteilung */
Setze $M_{0,1}$:= F, $M_{0,2}$:= Z \ F und $K_0 = \{ M_{0,1},M_{0,2}\}$.

Nächster_Schritt:
/* $K_k = \{ M_{k,1},...,M_{k,n} \}$ ist eine Klasseneinteilung von Z in Klassen von k-äquivalenten Zuständen. Nun Bestimmung von K_{k+1} */
Setze k := k+1:
Setze $K_k := \varnothing$. /* Fülle nun K_k mit Unterklassen der Klasseneinteilung K_{k-1}.*/
Für jedes i = 1,...,n:
- Bestimme eine Klasseneinteilung $H_{k,i}$ von $M_{k-1,i}$ in Klassen von k-äquivalenten Zuständen.
  /* z und z‘ kommen in eine Klasse, falls für alle $x \in \Sigma$ gilt: $\delta(z,x)$ und $\delta(z',x)$ gehören zu ein und derselben Klasse in K_{k-1} */
- Setze $K_k = K_k \cup H_{k,i}$.

Falls $K_k = K_{k-1}$, so setze $K := K_k$, Stop;
Wenn nicht, gehe zu Nächster_Schritt.

▶ Beispiel



Klassen 0-äquivalenter Zustände:

$K_0 = \{ M_{0,1}, M_{0,2} \}$ mit
$M_{0,1} = \{ z_3, z_5, z_6 \}$; $M_{0,2} = \{ z_0, z_1, z_2, z_4 \}$

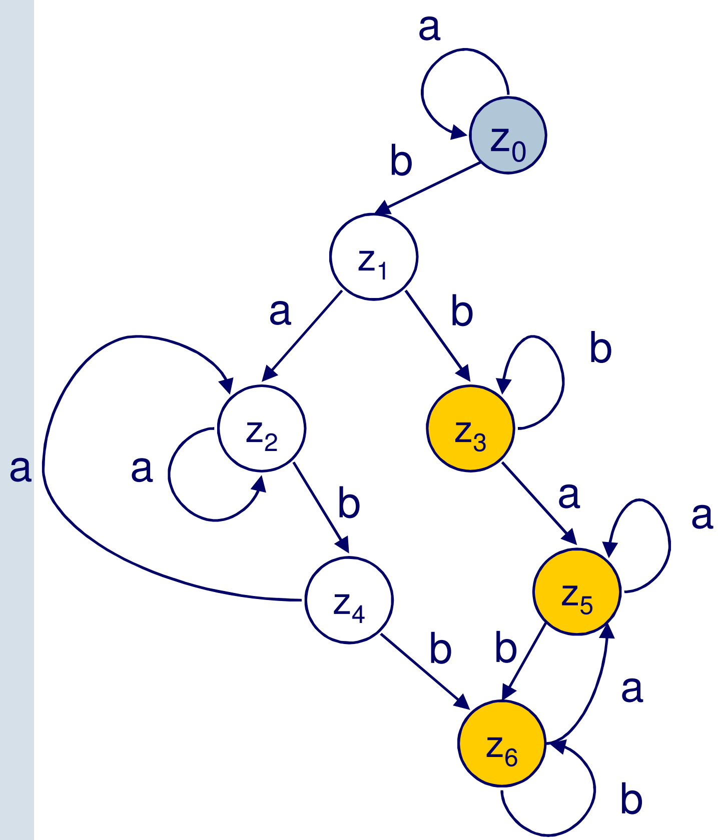
Klasseneinteilung $H_{1,1}$ für $M_{0,1}$:

$\delta(z_3,a) = z_5 \in M_{0,1}$ usw.

$Z \setminus \Sigma$	a	b
z_3	$M_{0,1}$	$M_{0,1}$
z_5	$M_{0,1}$	$M_{0,1}$
z_6	$M_{0,1}$	$M_{0,1}$

$H_{1,1} := \{\{ z_3, z_5, z_6 \}\} = \{M_{0,1}\}$, d.h. $M_{0,1}$ wird **nicht** unterteilt. $M_{1,1} = \{ z_3, z_5, z_6 \}$,

Beispiel



Klassen 0-äquivalenter Zustände:

$K_0 = \{ M_{0,1}, M_{0,2} \}$ mit
$M_{0,1} = \{ z_3, z_5, z_6 \}$; $M_{0,2} = \{ z_0, z_1, z_2, z_4 \}$

Klasseneinteilung $H_{1,2}$ für $M_{0,2}$:

$z \backslash \Sigma$	a	b
z_0	$M_{0,2}$	$M_{0,2}$
z_1	$M_{0,2}$	$M_{0,1}$
z_2	$M_{0,2}$	$M_{0,2}$
z_4	$M_{0,2}$	$M_{0,1}$

$H_{1,2} = \{ M_{1,2}, M_{1,3} \}$ mit
$M_{1,2} = \{ z_0, z_2 \}$; $M_{1,2} = \{ z_1, z_4 \}$

Minimierungsalgorithmus (Teil 2)

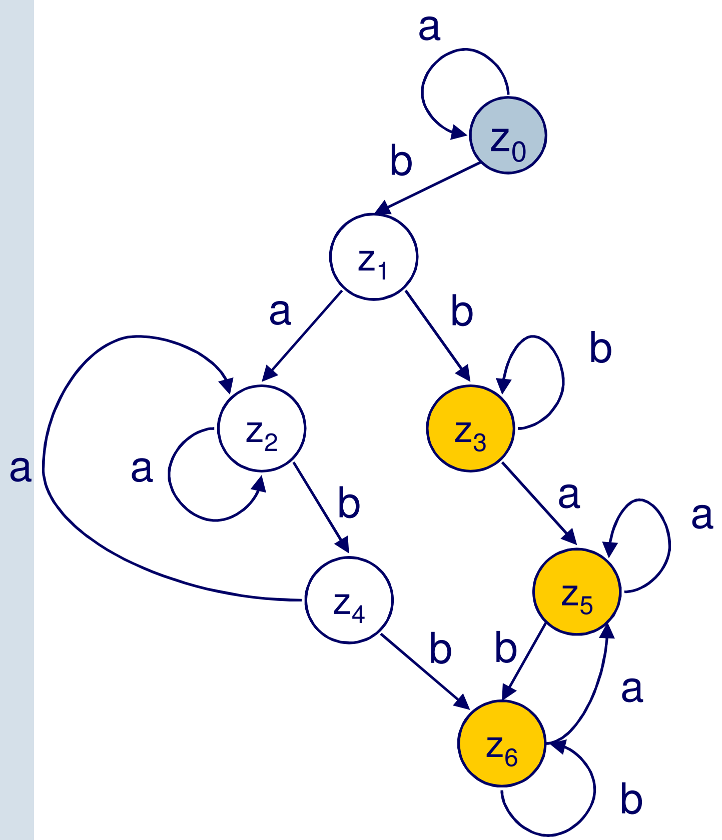
- Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat.
- Es sei K die Klasseneinteilung von Z in Klassen äquivalenter Zustände.

- Der gesuchte minimale endliche Automat $A° = [Z°,\Sigma°,z°_0,F°,\delta°]$ mit $L(A°) = L(A)$, der sog. Quotientenautomat, 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 $z_0 \in M$ /* Es gibt genau **ein** solches M. */
- $F° := \{ z_M \in Z° \mid M \cap F \neq \emptyset \}$
  /* Eine Äquivalenzklasse akzeptiert, wenn einer Ihrer Zustände akzeptiert.*/
- Für jedes $z_M \in Z°$ und jedes $x \in \Sigma$:
  - setze $\delta°(z_M,x) := z_{M'}$ mit $M' := \{ \delta(z,x) \mid z \in M \}$

  /* … die Menge alle x-Ziele von Zuständen aus M.
  Sie ist (zum Glück) immer selbst eine Äquivalenzklasse ☺ */

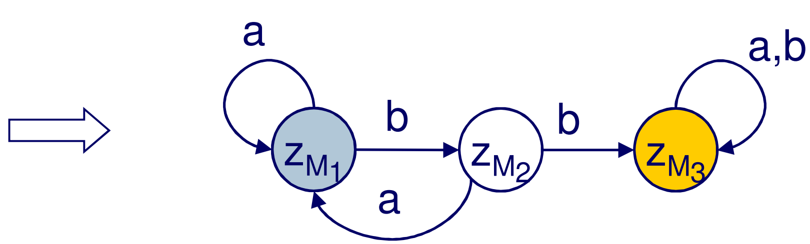
- Beispiel



Klassen äquivalenter Zustände:

$K = \{ M_1, M_2, M_3 \}$ mit

$M_1 = \{ z_0, z_2 \}$; $M_2 = \{ z_1, z_4 \}$; $M_3 = \{ z_3, z_5, z_6 \}$



Minimierungsalgorithmus

- Manuelle Minimierung, hier ohne Mengenindizes (bis zu 26 Zuständen OK)
  … anhand des vorigen Beispiels TEIL 1

$z \backslash \Sigma$	a	b
z_0	z_0	z_1
z_1	z_2	z_3
z_2	z_2	z_4
z_3	z_5	z_3
z_4	z_2	z_6
z_5	z_5	z_6
z_6	z_5	z_6

Klassen Z\F, F benennen, z.B. mit A und Z. Einträge: In welcher Klasse liegt der Zustand?

K0	a	b
A	A	A
A	A	Z
A	A	A
Z	Z	Z
A	A	Z
Z	Z	Z
Z	Z	Z

Minimierungsalgorithmus

- Manuelle Minimierung, hier ohne Mengenindizes (bis zu 26 Zuständen OK)

  … anhand des vorigen Beispiels   TEIL 2

  … Klassenverfeinerung von K_0 nach K_1 ausnahmsweise farblich markiert.

z\Σ	K0	a	b
z_0	**A**	A	A
z_1	**A**	A	Z
z_2	**A**	A	A
z_3	Z	Z	Z
z_4	**A**	A	Z
z_5	Z	Z	Z
z_6	Z	Z	Z

Unterschiedliche „Zukunft" → unter-schiedliche **Teil**klasse neu benennen.

K1
A
B
A
Z
B
Z
Z

Minimierungsalgorithmus

▶ Manuelle Minimierung, hier ohne Mengenindizes (bis zu 26 Zuständen OK)
… anhand des vorigen Beispiels TEIL 3

z\Σ	a	b
z_0	z_0	z_1
z_1	z_2	z_3
z_2	z_2	z_4
z_3	z_5	z_3
z_4	z_2	z_6
z_5	z_5	z_6
z_6	z_5	z_6

z\Σ	K1	a	b
z_0	A	A	B
z_1	B	A	Z
z_2	A	A	B
z_3	Z	Z	Z
z_4	B	A	Z
z_5	Z	Z	Z
z_6	Z	Z	Z

keine weitere Aufspaltung $\Rightarrow$ $K = K_1$; **Stop**.

Minimierungsalgorithmus

- Manuelle Minimierung, hier ohne Mengenindizes (bis zu 26 Zuständen OK)

… anhand des vorigen Beispiels TEIL 4

evtl. Klassen-Tabellen nebeneinander, aber nicht in falsche Tabelle schauen!

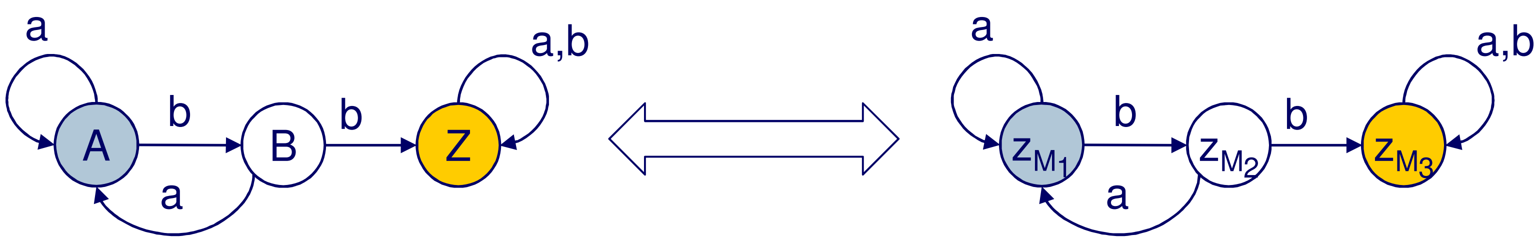
z\Σ
z_0
z_1
z_2
z_3
z_4
z_5
z_6

K0	a	b
A	A	A
A	A	Z
A	A	A
Z	Z	Z
A	A	Z
Z	Z	Z
Z	Z	Z

erledigt!

K1	a	b
A	A	B
B	A	Z
A	A	B
Z	Z	Z
B	A	Z
Z	Z	Z
Z	Z	Z

Anfangszustand:
$z_0 \in A \Rightarrow$ **A**
Akz. Zustände: Wo finden sich z_3,z_5,z_6?
In Z/Z/Z, $\Rightarrow$ nur **Z**



Minimierungsalgorithmus

▶ Eigenschaften des Minimierungsalgorithmus und der Minimalautomaten

Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat mit n Zuständen, in dem jeder Zustand vom Startzustand aus erreichbar ist.

- Das Ergebnis der Anwendung des Minimierungsalgorithmus auf A liefert einen minimalen endlichen Automaten A° mit L(A°) = L(A) (minimal in der Anzahl der Zustände).
- Geeignet implementiert benötigt der Minimierungsalgorithmus $O(n^2)$ viele Rechenschritte, um A° zu bestimmen.

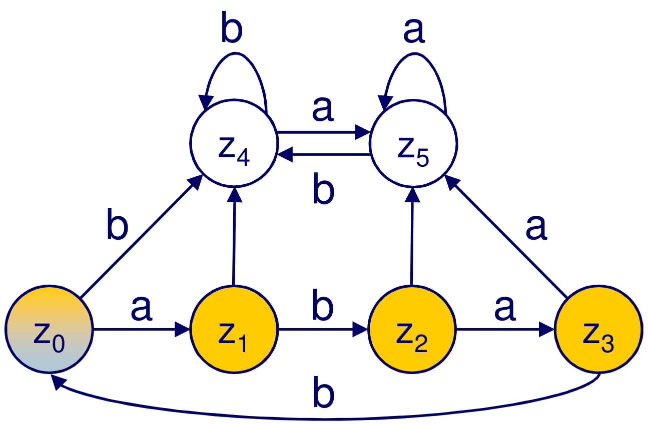
Fakt: Alle Automaten mit der gleichen Sprache wie A und mit der Minimalzahl von Zuständen (also mit gleich vielen wie A°) sind isomorph, d.h. sie unterscheiden sich höchstens in den Namen der Zustände.

Dies liefert einen Algorithmus zur Prüfung, ob zwei Automaten die **gleiche Sprache** haben:
Sie haben „gleiche“ (genauer: isomorphe) Minimalautomaten.

→ Überlegen Sie, wie Sie das systematisch prüfen (lassen) würden!

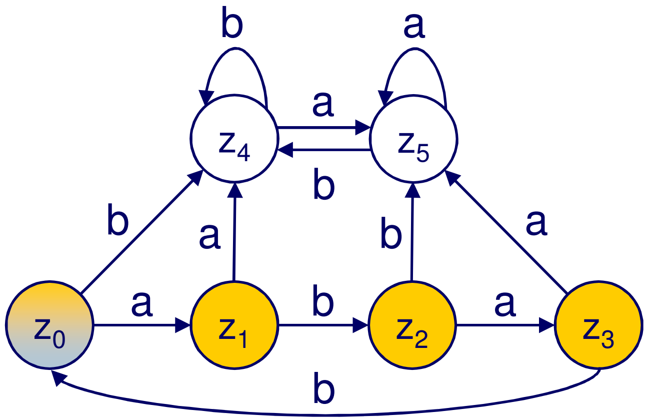
Minimierungsalgorithmus

- Beispiel zum Üben: Wir minimieren von Hand …



Minimierungsalgorithmus

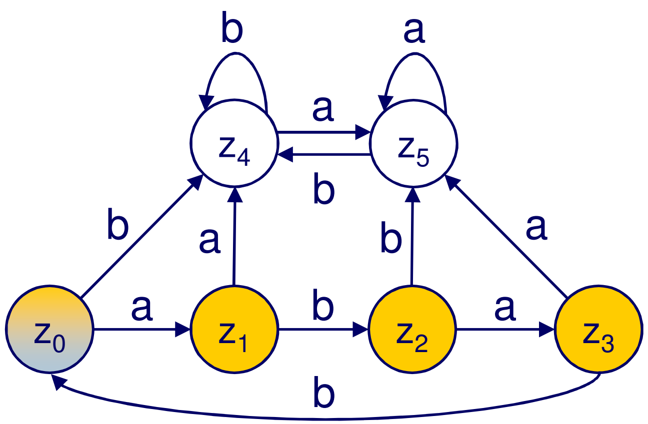
- Beispiel zum Üben: Wir minimieren von Hand …



$z\backslash\Sigma$	a	b
z_0	z_1	z_4
z_1	z_4	z_2
z_2	z_3	z_5
z_3	z_5	z_0
z_4	z_5	z_4
z_5	z_5	z_4

Minimierungsalgorithmus

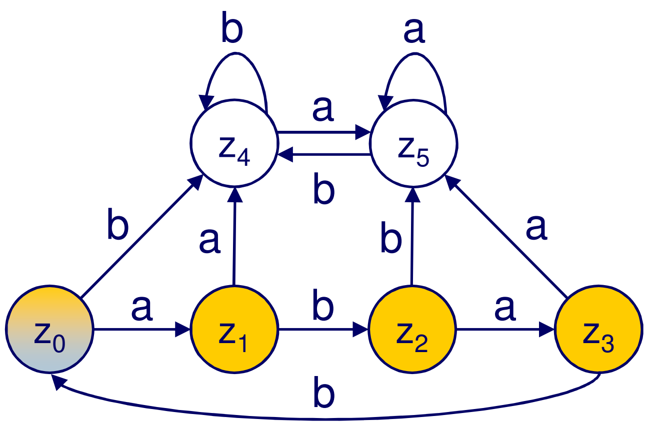
- Beispiel zum Üben: Wir minimieren von Hand …



$z \backslash \Sigma$	a	b
z_0	z_1	z_4
z_1	z_4	z_2
z_2	z_3	z_5
z_3	z_5	z_0
z_4	z_5	z_4
z_5	z_5	z_4

$z \backslash \Sigma$	a	b
Z	Z	A
Z	A	Z
Z	Z	A
Z	A	Z
A	A	A
A	A	A

▶ Beispiel zum Üben: Wir minimieren von Hand …



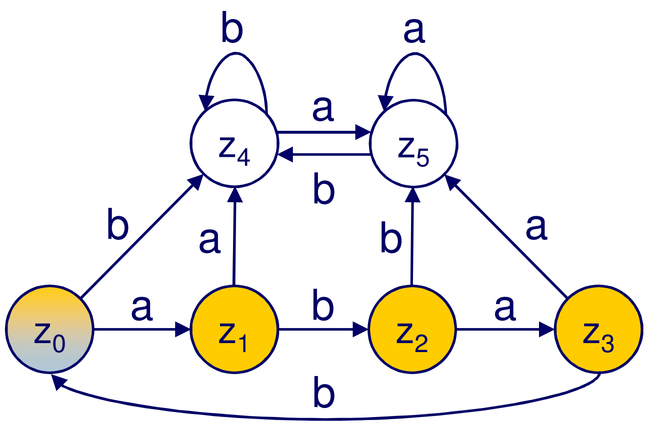
$z\backslash\Sigma$	a	b
z_0	z_1	z_4
z_1	z_4	z_2
z_2	z_3	z_5
z_3	z_5	z_0
z_4	z_5	z_4
z_5	z_5	z_4

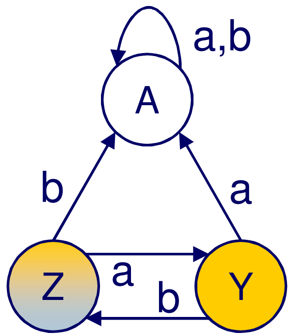
$z\backslash\Sigma$	a	b
Z	Z	A
Z	A	Z
Z	Z	A
Z	A	Z
A	A	A
A	A	A

$z\backslash\Sigma$	a	b
Z	Y	A
Y	A	Z
Z	Y	A
Y	A	Z
A	A	A
A	A	A

Minimierungsalgorithmus

- Beispiel zum Üben: Wir minimieren von Hand …





z\Σ	a	b
z_0	z_0	z_1
z_1	z_2	z_3
z_2	z_2	z_4
z_3	z_2	z_6
z_4	z_5	z_6
z_5	z_5	z_6

z\Σ	a	b
Z	Z	A
Z	A	Z
Z	Z	A
Z	A	Z
A	A	A
A	A	A

z\Σ	a	b
Z	Y	A
Y	A	Z
Z	Y	A
Y	A	Z
A	A	A
A	A	A

Minimierungsalgorithmus

- Warnung vor „beliebtem Fehler“

Annahme: irgendein Zwischenergebnis fast wie im vorigen Beispiel …
Nächste Klasseneinteilung?

z\Σ	a	b
Z	Z	A
Z	A	Z
Z	Z	A
Z	A	Z
A	A	A
A	A	Z

⇨

z\Σ
Z
Y
Z
Y
A
Y??

z\Σ
Z
Y
Z
Y
A
B!!

Es wird immer feiner zerlegt – und nicht Zerlegtes wieder zusammengeklebt!