Kapitel 2: Formale Sprachen

Gliederung

0. Einleitung und Grundbegriffe
1. Endliche Automaten
2. **Formale Sprachen**
3. Berechnungstheorie
4. Komplexitätstheorie

2.1. Chomsky-Grammatiken
2.2. Reguläre Sprachen
- **Reguläre Grammatiken, ND-Automaten**
- Abgeschlossenheit
- Reguläre Ausdrücke

2.3. Kontextfreie Sprachen

Erinnerung: reguläre Grammatiken/Sprachen

- Es sei G = [Σ,V,S,R] eine Chomsky-Grammatik

G heißt reguläre Grammatik, falls für alle Regeln l → r von G gilt:

- l ∈ V (d.h. die linke Seite ist eine Variable)
- r ∈ Σ (d.h. die rechte Seite ist ein Terminalzeichen)
  oder
  r = xA mit x ∈ Σ und A ∈ V (die rechte Seite ist eine Konstante gefolgt von einer Variablen)

Eine Sprache L heißt reguläre Sprache, falls es eine reguläre Grammatik G mit L(G) = L \ {ε} gibt.

Beispiel 1

- Sprache L, die genau alle Zeichenketten enthält, die nur aus 0en und 1en bestehen und die die Zeichenkette 11 enthalten; kürzer:

  $L = \{ v \circ 11 \circ w \mid v, w \in \Sigma^* \}$ mit $\Sigma = \{ 0,1 \}$

- reguläre Grammatik G = [Σ,V,S,R] für L

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B \}$
S

R:

$S \rightarrow 0S$	$B \rightarrow 0$
$S \rightarrow 1A$	$B \rightarrow 1$
	$B \rightarrow 0B$
$A \rightarrow 0S$	$B \rightarrow 1B$
$A \rightarrow 1$	
$A \rightarrow 1B$	

oder kürzer:

$S \rightarrow 0S \mid 1A$
$A \rightarrow 0S \mid 1 \mid 1B$
$B \rightarrow 0 \mid 1 \mid 0B \mid 1B$

B: beliebiges 0-1-Wort mit mindestens 1 Zeichen
A: eine 1 und dann beliebiger Rest oder
eine 0 und dann ein Wort mit 11 drin
S: Wort mit 11 drin

Beispiel 2

- $\Sigma = \{ 0,1 \}$
- $L = \{ w \in \Sigma^* \mid w \text{ enthält ungerade viele Nullen} \}$

- reguläre Grammatik $G = [\Sigma,V,S,R]$ für L

$\Sigma = \{ 0,1 \}$
$V = \{ S,A \}$

R: $S \rightarrow 0 \mid 0A \mid 1S$
$A \rightarrow 1 \mid 1A \mid 0S$

S: beliebig viele 1 und ungerade viele 0
A: beliebig viele 1 und gerade viele 0

Anmerkungen

- Uns interessiert hauptsächlich der algorithmische Aspekt, also die Frage, wie man das Wortproblem für eine reguläre Sprache effizient lösen kann:
  - gegeben: eine reguläre Grammatik $G = [\Sigma, V, S, R]$
  - gesucht: ein Algorithmus, der bei Eingabe eines beliebigen Wortes $w \in \Sigma^*$, die Antwort „1“ liefert, falls $w \in L(G)$ gilt, und die Antwort „0“ liefert, falls $w \notin L(G)$ gilt
- Wir werden sehen, dass man das Wortproblem für reguläre Sprachen sehr effizient lösen kann.

  Das liegt daran, dass man anhand einer regulären Grammatik für eine Sprache $L \setminus \{ \varepsilon \}$ einen endlichen Automaten konstruieren kann, der die Sprache L akzeptiert.

- Zentrales Ergebnis

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

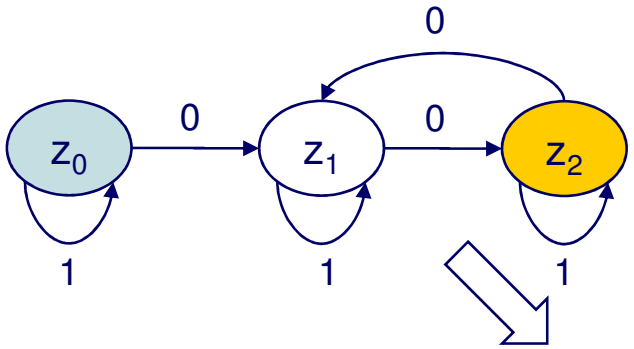
1. Es gibt eine reguläre Grammatik G mit $L(G) = L \setminus \{ \varepsilon \}$.
2. Es gibt einen endlichen Automaten A mit $L(A) = L$.

Vor allem interessiert die Richtung „(1) $\Rightarrow$ (2)“.

Wir schauen uns trotzdem auch die Richtung „(2) $\Rightarrow$ (1)“ an, um eine Idee zu entwickeln, wie man die uns interessierende Richtung beweisen kann.

Implikation „Automatensprache ⇒ reguläre Sprache“ (am Beispiel)

- Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat,
- und es gelte $z_0 \notin F$, wie im folgenden Beispiel:



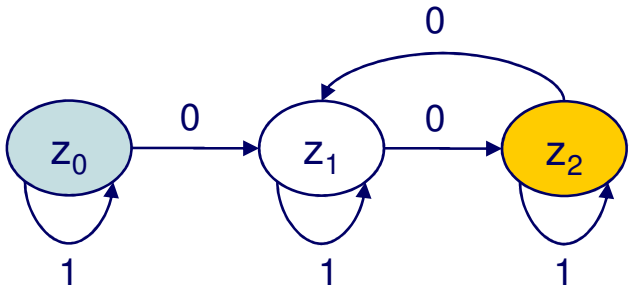
$G = [\Sigma,V,S,R]$ mit
$\Sigma = \{ 0,1 \}$
$V = \{ H_{z_0},H_{z_1},H_{z_2} \}$
H_{z_0}

$H_{z_0} \rightarrow 1H_{z_0} \mid 0H_{z_1}$
$H_{z_1} \rightarrow 1H_{z_1} \mid 0H_{z_2} \mid 0$
$H_{z_2} \rightarrow 1 \mid 1H_{z_2} \mid 0H_{z_1}$

Idee:
- eine Variable H_z für jeden Zustand z, und
- $\delta(z_i,x)=z_k$ → $H_{z_i} \rightarrow xH_{z_k}$
- und wenn $z_k \in F$, dann „Ende ermöglichen“!

d.h. zusätzlich zu $H_{z_i} \rightarrow xH_{z_k}$ auch $H_{z_i} \rightarrow x$

- Implikation „Automatensprache ⇒ reguläre Sprache“ (am Beispiel)
  - Verarbeitung des Wortes w = 0101 im Automat
  - Generierung des Wortes w = 0101 mit Grammatik



$G = [\Sigma,V,S,R]$ mit
$\Sigma = \{ 0,1 \}$
$V = \{ H_{z_0},H_{z_1},H_{z_2} \}$
H_{z_0}

$H_{z_0} \rightarrow 1H_{z_0} \mid 0H_{z_1}$
$H_{z_1} \rightarrow 1H_{z_1} \mid 0H_{z_2} \mid 0$
$H_{z_2} \rightarrow 1 \mid 1H_{z_2} \mid 0H_{z_1}$

$(z_0,0101) \rightarrow_A (z_1,101)$
$\rightarrow_A (z_1,01)$
$\rightarrow_A (z_2,1)$
$\rightarrow_A (z_2,\varepsilon)$

$H_{z_0} \rightarrow_G 0H_{z_1}$
$\rightarrow_G 01H_{z_1}$
$\rightarrow_G 010H_{z_2}$
$\rightarrow_G 0101$

links: gelesenes Zeichen = rechts: erzeugtes Terminalzeichen

Implikation „Automatensprache $\Rightarrow$ reguläre Sprache“

- Es sei $A = [Z,\Sigma,z_0,F,\delta]$ ein endlicher Automat mit $z_0 \notin F$.
- Dann erzeugt die reguläre Grammatik $G = [\Sigma,V,S,R]$ mit
  - $V = \{ H_z \mid z \in Z \}$
  - $S = H_{z_0}$
  - $R = R_1 \cup R_2$ mit:
    - $R_1 = \{ H_z \rightarrow aH_{z'} \mid z \in Z, a \in \Sigma \text{ und } \delta(z,a) = z' \}$
    - $R_2 = \{ H_z \rightarrow a \mid z \in Z, a \in \Sigma \text{ und } \delta(z,a) \in F \}$

die Sprache $L(A) \setminus \{ \varepsilon \}$.

… und das klappt sogar, wenn $z_0 \in F$ – vgl. Beispiel:

- Implikation „ reguläre Sprache ⇒ Automatensprache“ (Beispiel)

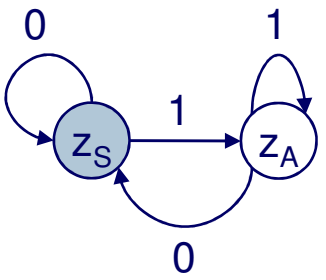
  - Es sei G = [Σ,V,S,R] die reguläre Grammatik mit

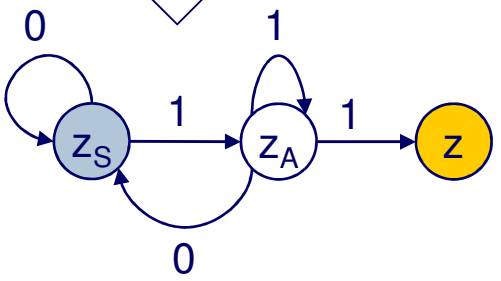
    Σ = { 0,1 }
    V = { S,A }
    Startsymbol S
    R: $S \rightarrow 0S \mid 1A$
    $A \rightarrow 0S \mid 1A \mid 1$

    Naive Umkehrung der vorigen Richtung:

    ⇒

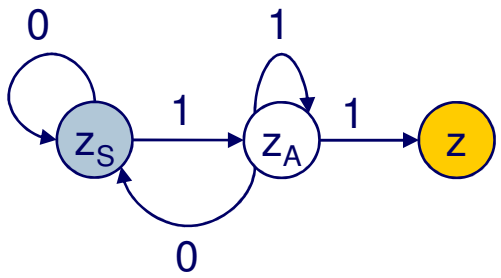
    ein Zustand Pro Variable





  - <u>Problem 1</u>: akzeptierende Zustände?
    z_S? Nein (sonst 0 akzeptiert)
    z_A? Nein (sonst 1 akzeptiert)
    <u>Problem 2</u>: $\delta(z_A,1)$ = ?
    Also muss ein neuer Zustand her: z

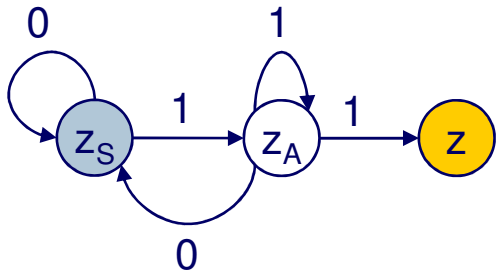
- Implikation „ reguläre Sprache ⇒ Automatensprache“ (Beispiel)



- Problem 3
  Auch dieser „partielle Automat“ hat einige Eigenheiten, die nicht zu einem endlichen Automaten passen:
  - Die Überführung ist hier keine Funktion, sondern nur eine Relation:
    - Sie ist nicht linkstotal: Es fehlt z.B. $\delta(z,0)$.
    - Sie ist nicht rechtseindeutig: $\delta(z_A,1) = z_A$ oder z?
  - Wenn man ein Wort als Wegbeschreibung nimmt, muss man evtl. irgendwann abbrechen, oder man hat stellenweise unterschiedliche Wege zur Auswahl und kann in unterschiedlichen Zuständen ankommen.

- Implikation „reguläre Sprache ⇒ Automatensprache“ (Beispiel)

Idee: *Welche Sprache erhält man, wenn man alle Wörter akzeptiert, bei denen man in F ankommen **kann**?*
Vgl.: *Automat als Wörterproduzent*



Das klappt in unserem Beispiel!

- Die „Wegbeschreibungen“, die im „partiellen Automaten“ nach z führen *können*, sind

  … genau alle Wörter mit Suffix 11.

- Die Regelmenge $S \rightarrow 0S \mid 1A$, $A \rightarrow 0S \mid 1A \mid 1$ der Grammatik G erzeugt

  … genau alle Wörter mit Suffix 11.

Zwischenbilanz

- Wenn man zu einer gegebenen regulären Grammatik G einen endlichen Automaten A mit L(A) = L(G) intuitiv konstruieren will, erhält man als Zwischenergebnis einen „partiellen Automaten“, der kein endlicher Automat ist.
- Solche Gebilde nennt man nichtdeterministische endliche Automaten.
- Wir werden uns im Folgenden das Berechnungsmodell nichtdeterministischer Automaten genauer ansehen – insbesondere, wie sich damit auch eine akzeptierte Sprache definieren lässt – und zeigen:
  - Zu jedem nichtdeterministischen endlichen Automaten kann man einen (deterministischen) endlichen Automaten konstruieren, der dieselbe Sprache akzeptiert.

Den Beweis für die noch ausstehende Richtung „reguläre Sprache ⇒ Automatensprache“ werden wir dann anschließend führen.

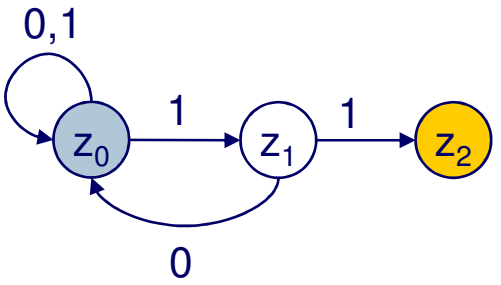
Formale Beschreibung

- Bestandteile eines nichtdeterministischen endlichen Automaten
  $B = [Z, \Sigma, z_0, F, \Delta]$
  - endliche Menge Z von Zuständen
    - ausgezeichneter Anfangszustand $z_0 \in Z$
    - Teilmenge $F \subseteq Z$ von akzeptierenden Zuständen
  - endliches Eingabealphabet Σ
  - eine Menge von Transitionen $\Delta \subseteq Z \times \Sigma \times Z$, die (dreistellige) Zustandsüberführungsrelation

Achtung: Man findet viele **unterschiedliche Definitionen** der NDAs:

- Manchmal wird die Zustandsüberführung formal scheinbar anders definiert (→ Currying!),
- oder es werden auch 0 oder mehrere Anfangszustände zugelassen,
- oder es werden auch (im Bild) Pfeile zugelassen, an denen ε oder kein Symbol steht („spontane Zustandsübergänge“),
- oder es werden ganze Zeichenketten an die Pfeile geschrieben.

- grafische Darstellung an Beispiel:
(neu aber gleichwertig zu vorigem)



- $Z = \{ z_0, z_1, z_2 \}$
- $\Sigma = \{ 0,1 \}$
- z_0
- $F = \{ z_2 \}$
- Für alle $z, z' \in Z$ und alle $a \in \Sigma$ gilt:
$(z,a,z') \in \Delta$ gdw. es eine mit a markierte Kante von z nach z' gibt:
also $\Delta = \{ (z_0,0,z_0), (z_0,1,z_0), (z_0,1,z_1), (z_1,0,z_0), (z_1,1,z_2) \}$

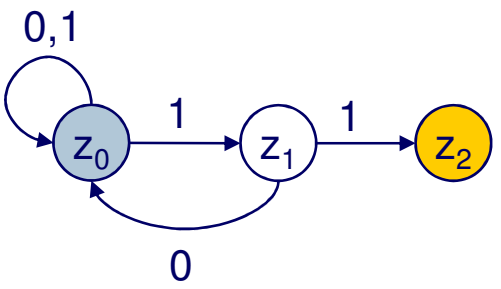
Alternative: Beschreibung von Δ auch tabellarisch durch
die Abbildung $\delta: Z \times \Sigma \to 2^Z$ mit $\delta(z,a) = \{ z' \in Z \mid (z,a,z') \in \Delta \}$.

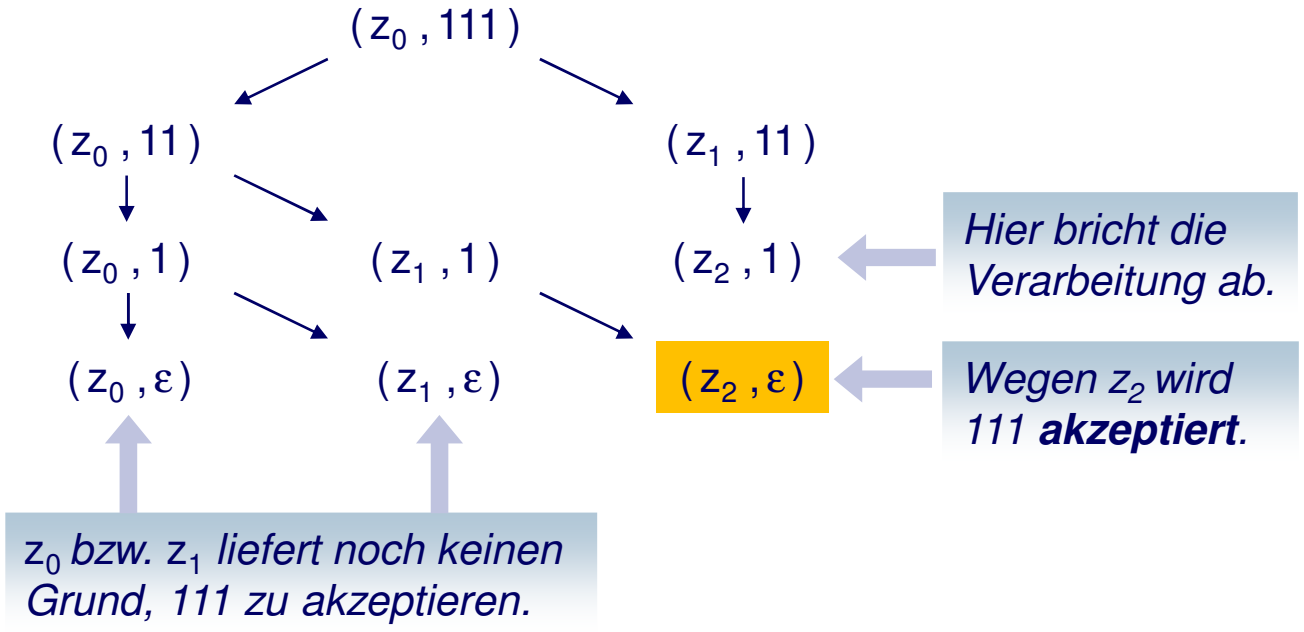
δ:

$z\backslash\Sigma$	0	1
z_0	$\{ z_0 \}$	$\{ z_0, z_1 \}$
z_1	$\{ z_0 \}$	$\{ z_2 \}$
z_2	$\varnothing$	$\varnothing$

„Currying“

- Verarbeitungsmöglichkeiten am Beispiel:





Anmerkungen

- Ein nichtdeterministischer endlicher Automat kann, wie wir gesehen haben, ein und dasselbe Wort w auf unterschiedliche Art und Weise verarbeiten.
- Um formal zu definieren, welche Sprache ein nichtdeterministischer endlicher Automat akzeptiert, gehen wir wie folgt vor:
  - Wir definieren, welche **Zustände** der Automat erreichen kann, wenn er im Startzustand beginnend ein Wort w vollständig verarbeitet hat.
  - Das Wort w soll genau dann akzeptiert werden, wenn **unter diesen** Zuständen ein **akzeptierender** Zustand ist.

Hilfsbegriff: Erreichbarkeit = erweiterte Überführungsrelation

- Sei $A = [Z,\Sigma,z_0,F,\Delta]$ ein nichtdeterministischer endlicher Automat

Die erweiterte Überführungsrelation $\Delta^* \subseteq Z \times \Sigma^* \times Z$ von A ist wie folgt definiert:

- Für alle $z \in Z$ gilt $(z, \varepsilon, z) \in \Delta^*$
- Gilt für $z_1, z_2, z_3 \in Z$, $u \in \Sigma^*$ und $a \in \Sigma$:
  $(z_1, u, z_2) \in \Delta^*$ und $(z_2, a, z_3) \in \Delta$,
  so gilt auch $(z_1, u \circ a, z_3) \in \Delta^*$.

Wir nennen einen Zustand z von A mit w erreichbar, wenn $(z_0,w,z) \in \Delta^*$.

Begriff: akzeptierte Sprache

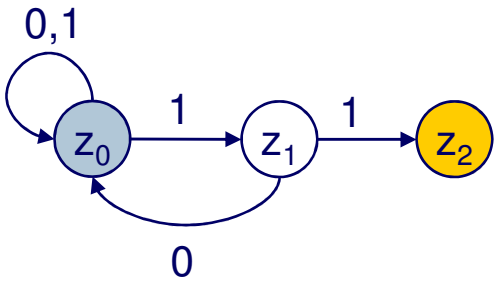
- Es sei $A = [Z,\Sigma,z_0,F,\Delta]$ ein nichtdeterministischer endlicher Automat
- Dann definieren wir die von A akzeptierte Sprache L(A) wie folgt:

$$L(A) = \{ w \in \Sigma^* \mid (z_0, w, z) \in \Delta^* \text{ für ein } z \in F \}$$

$w \in L(A) \Leftrightarrow$ In A ist mindestens ein akzeptierender Zustand mit w erreichbar.

Beispiel

- $L(A) = \{ w \in \Sigma^* \mid w \text{ hat das Suffix } 11 \}$



- Uminterpretation endlicher Automaten

Wenn man Δ definiert, indem man $\delta(z_1,x)=z_2$ als $(z_1,x,z_2) \in \Delta$ liest →
*Jeder endliche Automat **ist im Prinzip** auch ein nicht-deterministischer endlicher Automat.*

Currying!

- Wichtiges Ergebnis

Es seien Σ ein endliches Alphabet und $L \subseteq \Sigma^*$.
Dann sind die folgenden Aussagen **äquivalent**:

1. Es gibt einen (deterministischen) endlichen Automaten A mit L(A) = L.
2. Es gibt einen nichtdeterministischen endlichen Automaten B mit L(B) = L.

Wegen der „Uminterpretation“ brauchen wir nur noch die obige Implikation „(2) ⇒ (1)“ zu zeigen.

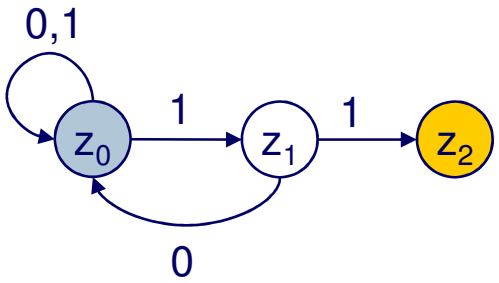
Wie könnte man zu einem gegebenen nichtdeterministischen endlichen Automaten B einen äquivalenten endlichen Automaten A konstruieren?

- Vorüberlegung: Unterschiede

n.-det. Automat B	det. Automat A
Es gibt eine Menge möglicher Zustände z mit $(z_0,w,z) \in \Delta^*$.	Es gibt genau einen Zustand z in $\delta^*(z_0,w)$.
$w \in L(B)$ gdw. $(z_0, w, z) \in \Delta^*$ für ein $z \in F$.	$w \in L(A)$ gdw. $\delta^*(z_0,w) \in F$.

- Idee
  - Im nichtdeterministischen endlichen Automaten B will man sich merken, in welchen Zuständen von B man inzwischen sein **könnte**. Die Zustände des gesuchten (deterministischen) endlichen Automaten A entsprechen diesen **Mengen** von Zuständen von B.
  - Die Überführungs**relation** Δ von B liefert eine Überführungs**funktion** δ für Zustandsmengen. (Currying)
  - Eine Zustandsmenge akzeptiert, wenn sie einen akzeptierenden B-Zustand **enthält**.

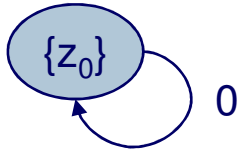
- Beispiel, Schnappschuss 1



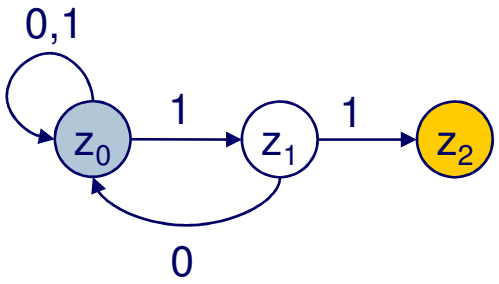
$z \in Z'$	$\delta'(z,0)$	$\delta'(z,1)$
$\{z_0\}$	$\{z_0\}$	

Konstruktion der erreichbaren Zustandsmengen und –übergänge

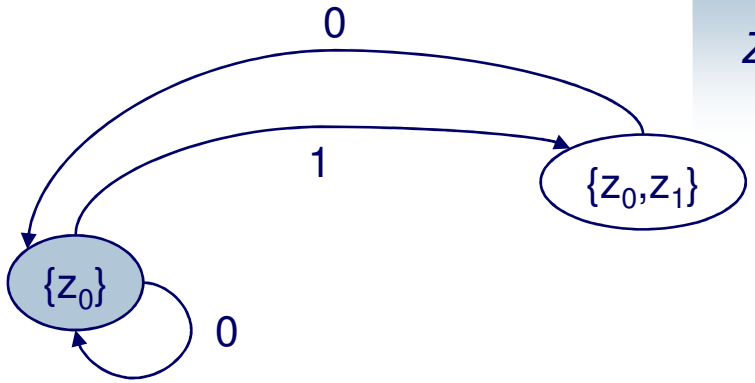
(Zwischenstand)



- Beispiel , Schnappschuss 2



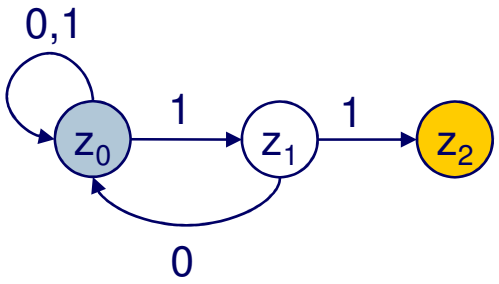
$z \in Z'$	$\delta'(z,0)$	$\delta'(z,1)$
$\{z_0\}$	$\{z_0\}$	$\{z_0,z_1\}$
$\{z_0,z_1\}$	$\{z_0\}$	



Konstruktion der erreichbaren Zustandsmengen und –übergänge

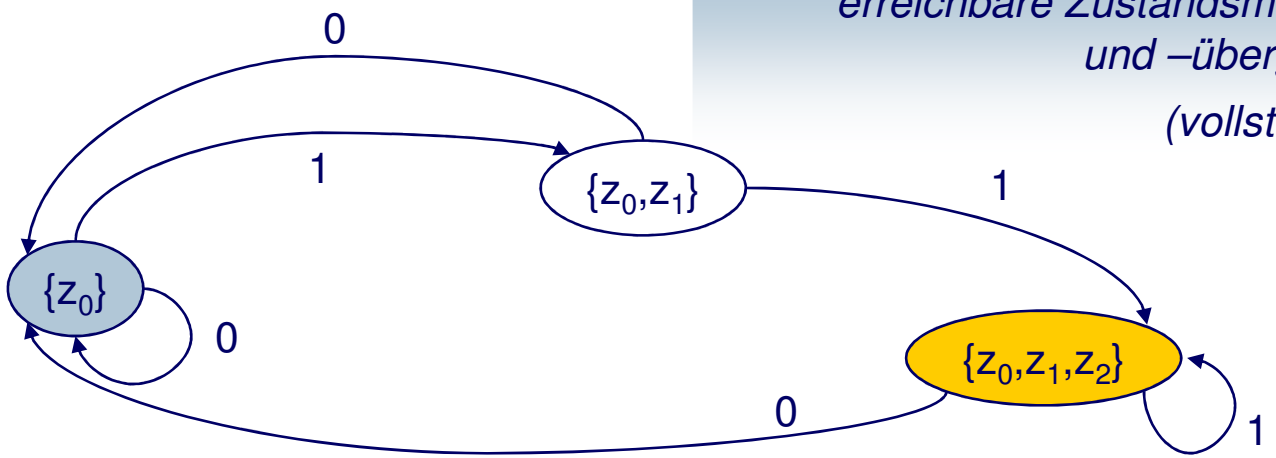
(Zwischenstand)

- Beispiel , Schnappschuss 3: Ende



$z \in Z'$	$\delta'(z,0)$	$\delta'(z,1)$
$\{z_0\}$	$\{z_0\}$	$\{z_0,z_1\}$
$\{z_0,z_1\}$	$\{z_0\}$	$\{z_0,z_1,z_2\}$
$\{z_0,z_1,z_2\}$	$\{z_0\}$	$\{z_0,z_1,z_2\}$

erreichbare Zustandsmengen und –übergänge (vollständig)



Konstruktionsvorschrift

- Es sei $B = [Z,\Sigma,z_0,F,\Delta]$ ein nichtdeterministischer endlicher Automat.
- Der gesuchte endliche Automat $A = [Z',\Sigma,z'_0,F',\delta']$ mit derselben Sprache kann wie folgt (als Potenzmengenautomat) definiert werden:

- $Z' = \{ M \mid M \subseteq Z \} = 2^Z$
- $z'_0 = \{ z_0 \}$
- $F' = \{ M \in Z' \mid M \cap F \neq \emptyset \}$
- für alle $M \in Z'$ und alle $a \in \Sigma$:
  - $\delta'(M,a) = \{ z_2 \in Z \mid (z_1,a,z_2) \in \Delta \text{ für ein } z_1 \in M \}$

Hinweis:
*Zustände, die vom Anfangszustand nicht erreichbar sind, streicht man dann am besten – oder man erzeugt gleich nur die **erreichbaren** Zustände, wie im vorigen Beispiel. Das ändert nichts an der Sprache und spart dann Rechenarbeit. (Die obige Definition spart Definitionsarbeit.)*
Übung: Definition an Rechnung anpassen, oder umgekehrt.

- zurück zum Thema „ reguläre Sprache $\Rightarrow$ Automatensprache“

  - Die Beziehung zwischen nichtdeterministischen endlichen Automaten und (deterministischen) endlichen Automaten benutzen wir jetzt, um nachzuweisen, dass das Wortproblem für reguläre Sprachen lösbar ist:

    Wir müssen nur noch genau den ersten Schritt beschreiben, nämlich wie wir aus einer regulären Grammatik
    - einen nichtdeterministischen Automaten B mit L(B)=L(G) und
    - einen nichtdeterministischen Automaten B′ mit L(B′)=L(G) $\cup$ ε

    konstruieren
    (und dann eigentlich noch beweisen, dass diese das gewünschte Ergebnis liefert. Dafür möge hier der Augenschein genügen.).

Intermezzo: ein Wermutstropfen …

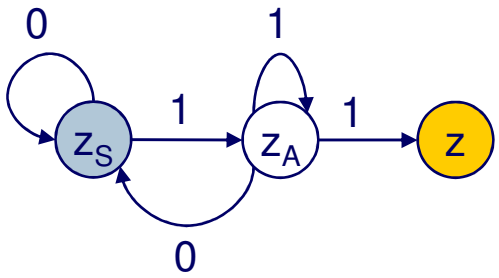
- Zwar wissen wir dann, dass zu jeder regulären Grammatik G ein deterministischer Automat existiert, der uns erlaubt, das Wortproblem für L(G) effizient zu lösen;
  aber wir kennen keine **effiziente** Konstruktion dieses Automaten!
- Wie kommt das?
- Der zu einem nichtdeterministischen endlichen Automaten B konstruierte Potenzmengenautomat A hat exponentiell mehr Zustände als B!
- Zwar kann A oft durch Weglassen nicht erreichbarer Zustände verkleinert bzw. von vornherein seine Konstruktion entsprechend verkürzt werden;
  es gibt jedoch reguläre Sprachen L_n, für die gilt:
  - Jeder (deterministische) endliche Automat für L_n hat exponentiell mehr Zustände als ein geeignet gewählter nichtdeterministischer endlicher Automat für L_n, nämlich mindestens 2^{n-1} im Vergleich zu n.

z.B. L_n: $1(0+1)^{n-1}$ (als erweiterter regulärer Ausdruck)

- Beispiel (ohne ε):
  reguläre Grammatik → nichtdeterministischer Automat

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B \}$
S

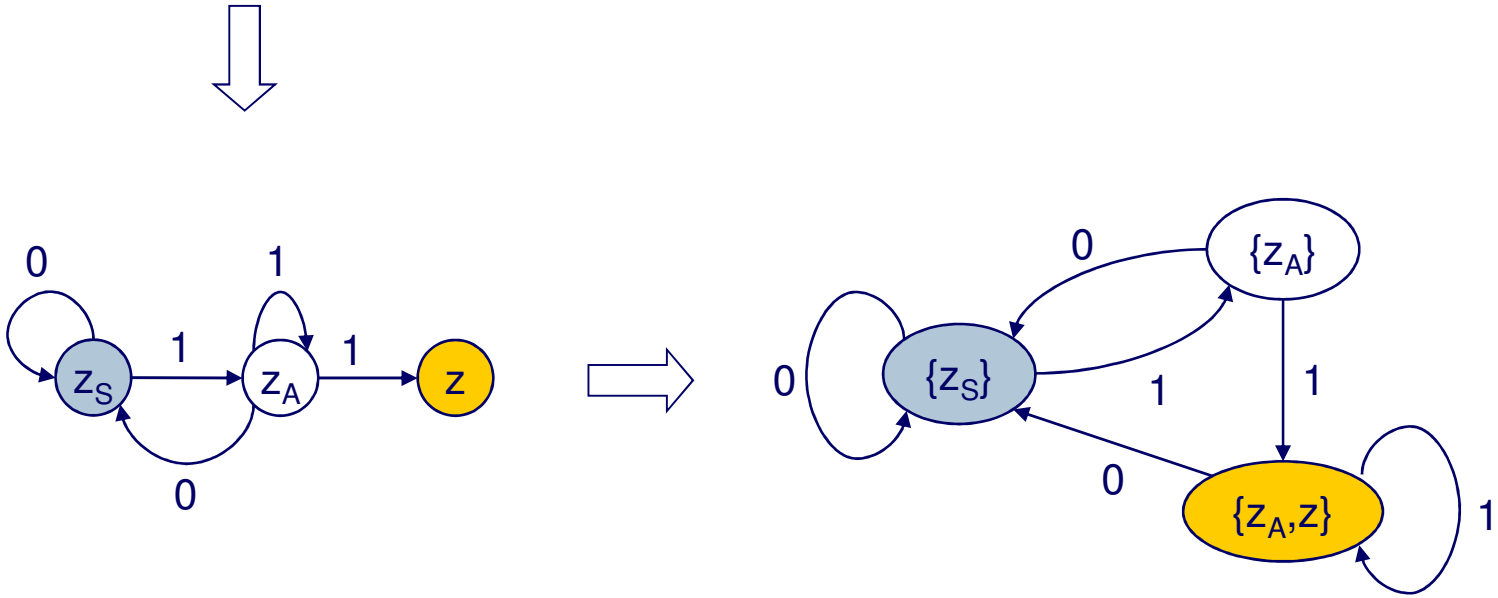
$S \rightarrow 0S \mid 1A$
$A \rightarrow 1A \mid 0S \mid 1$



- Beispiel (ohne ε):
  reguläre Grammatik → nichtdeterministischer Automat → endlicher Automat

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B \}$
S

$S \rightarrow 0S \mid 1A$
$A \rightarrow 1A \mid 0S \mid 1$



Konstruktion (ohne ε)

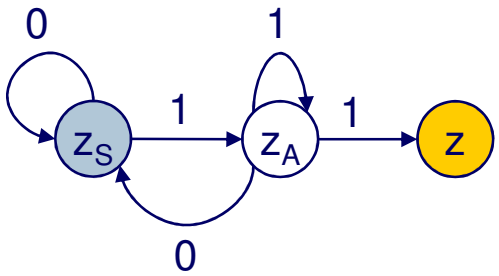
- Sei $L \subseteq \Sigma^*$ mit $\varepsilon \notin L$
- Sei $G = [\Sigma,V,S,R]$ eine reguläre Grammatik mit $L(G) = L$

- Dann definieren wir einen nichtdeterministischen endlichen Automaten $A = [Z,\Sigma,z_0,F,\Delta]$ mit $L(A) = L$ wie folgt:

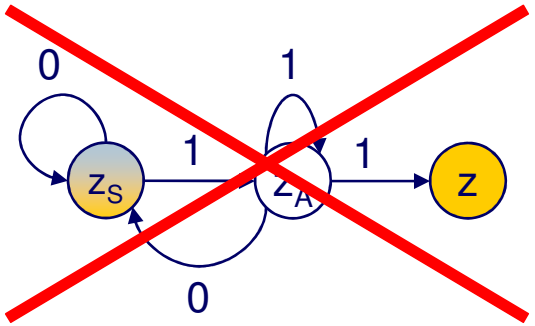
  - $Z = \{ z \} \cup \{ z_H \mid H \in V \}$
  - $z_0 = z_S$
  - $F = \{ z \}$
  - $\Delta = \Delta_1 \cup \Delta_2$ mit:
    - $\Delta_1 = \{ (z_H,a,z_{H'}) \mid$ die Regel $H \rightarrow aH'$ gehört zu $R \}$
    - $\Delta_2 = \{ (z_H,a,z) \mid$ die Regel $H \rightarrow a$ gehört zu $R \}$

Beispiel (mit ε)

- Um einen nichtdeterministischen Automaten für die Sprache L(G) ∪ { ε } zu definieren, dürfen wir den ND-Automaten für L(G)



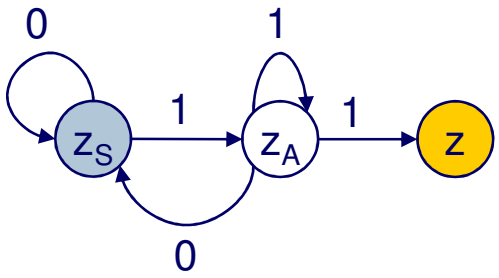
nicht einfach so modifizieren („damit ε akzeptiert wird“:



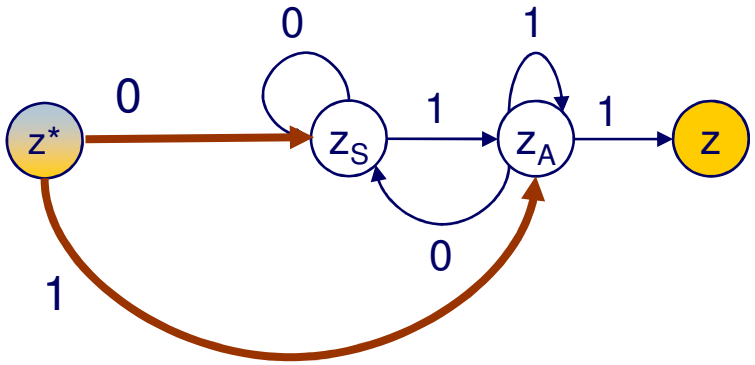
… akzeptiert leider auch 0, 00, 010, 1100 usw. ☹

Beispiel (mit ε)

- Um einen nichtdeterministischen Automaten für die Sprache L(G) ∪ { ε } zu definieren, müssen wir den ND-Automaten für L(G)



wie folgt modifizieren, um sicherzustellen, dass zusätzlich genau das leere Wort akzeptiert wird:



Neuer Startzustand z …*
- *„verhält sich wie“ z_S*
- *akzeptiert ε*
- *wird nie mehr angesteuert*

Konstruktion (mit ε)

- Sei $L \subseteq \Sigma^*$ mit $\varepsilon \in L$
- Sei $G = [\Sigma,V,S,R]$ eine reguläre Grammatik mit $L(G) = L \setminus \{ \varepsilon \}$

- Dann definieren wir einen nichtdeterministischen endlichen Automaten $A = [Z,\Sigma,z_0,F,\Delta]$ mit $L(A) = L$ wie folgt:

  - $Z = \{ z, z^* \} \cup \{ z_H \mid H \in V \}$
  - $z_0 = z^*$
  - $F = \{ z, z^* \}$
  - $\Delta = \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup \Delta_4$ mit:
    - $\Delta_1 = \{ (z_H,a,z_{H'}) \mid$ die Regel $H \rightarrow aH'$ gehört zu $R \}$
    - $\Delta_2 = \{ (z_H,a,z) \mid$ die Regel $H \rightarrow a$ gehört zu $R \}$
    - $\Delta_3 = \{ (z^*,a,z_H) \mid$ die Regel $S \rightarrow aH$ gehört zu $R \}$
    - $\Delta_4 = \{ (z^*,a,z) \mid$ die Regel $S \rightarrow a$ gehört zu $R \}$