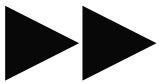
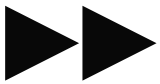
# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie



## *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*

konzeptioneller Aspekt ... reguläre Grammatiken

algorithmischer Aspekt ... deterministische endliche Automaten (DFA)

**bereits bekannt:**

jede von einem DFA A akzeptierte ist regulär,
m.a.W. es gibt eine reguläre Grammatik G
mit L(G) = L(A)

... Gilt die Umkehrung auch? Sind DFAs ausreichend, um den algorithmischen Aspekt in den Griff zu bekommen?

# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie

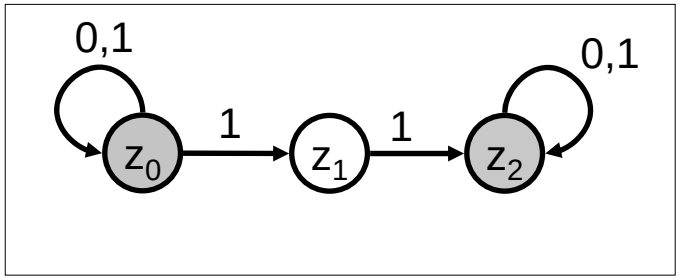
## *Reguläre Sprachen*

Sprache L aller Zeichenketten, die nur aus Nullen und Einsen bestehen und die Zeichenkette 11 enthalten.

kürzer: L = { v11w | v $\in$ { 0,1 }*, w $\in$ { 0,1 }* }

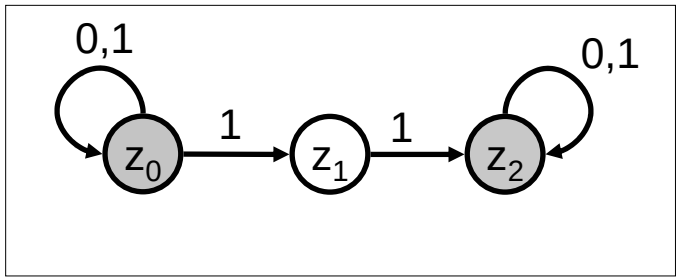
$\Sigma$ = { 0, 1 }
V = { S }
S

S → 0S |1H
H → 1H‘ | 0S | 1
H‘ → 0S | 1H‘ | 1




nicht-deterministischer endlicher Automat für L

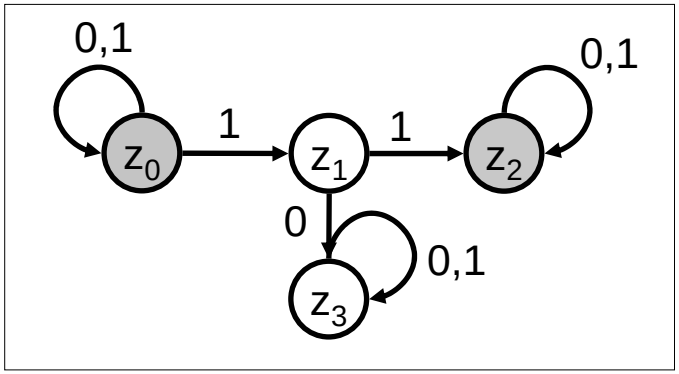
## *Reguläre Sprachen*



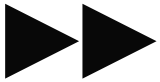

Beobachtungen:

(1) Zustandsübergänge sind nicht mehr eindeutig
(2) in einigen Situationen ist unklar, wie sich der Automat verhalten soll

(1) ist wesentlich
(2) ist „reparabel“



# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie



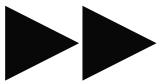
## *Reguläre Sprachen*

**Fahrplan**

- nicht-deterministische Automaten (NFA) genauer ansehen; insbesondere klären, welche Sprache ein NFA akzeptiert
- diskutieren, wie man zu einer regulären Grammatik einen äquivalenten NFA konstruiert
- diskutieren, wie man zu einem NFA einen äquivalenten DFA konstruiert

... damit bekommt man letztlich den algorithmischen Aspekt in den Griff

# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie



## *Reguläre Sprachen*

Bestimmungsstücke eines nicht-deterministischen endlichen Automaten

- eine Menge Z von Zuständen
- ein endliches Eingabealphabet $\Sigma$ von Symbolen
- ein ausgezeichneter Startzustand $z_0 \in Z$
- eine ausgezeichnete Menge $E \subseteq Z$ (/* die Endzustände */)
- eine Überführungsfunktion $\delta$: $Z \times \Sigma \rightarrow 2^Z$, wobei für alle $z \in Z$ und alle $a \in \Sigma$ gilt: $\delta(z,a) \neq \emptyset$

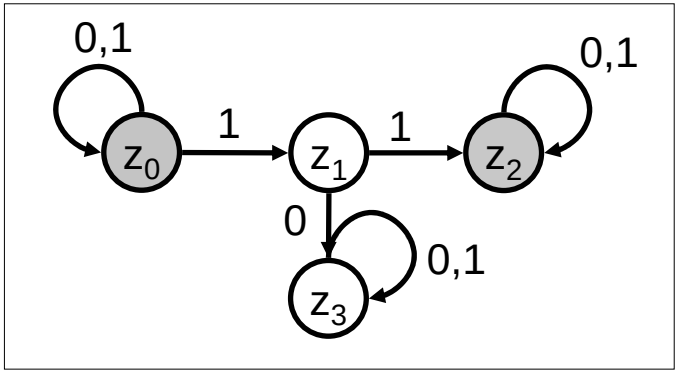
... man spricht auch von vollständig definierten nicht-deterministischen endlichen Automaten

# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie

## *Reguläre Sprachen*

Bestimmungsstücke eines nicht-deterministischen endlichen Automaten

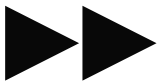
- eine Menge Z von Zuständen
- ein endliches Eingabealphabet $\Sigma$ von Symbolen
- ein ausgezeichneter Startzustand $z_0 \in Z$
- eine ausgezeichnete Menge $E \subseteq Z$ (/* die Endzustände */)
- eine Überführungsfunktion $\delta$: $Z \times \Sigma \rightarrow 2^Z$, wobei für alle $z \in Z$ und alle $a \in \Sigma$ gilt: $\delta(z,a) \neq \emptyset$




B = [{$z_0$,$z_1$,$z_2$,$z_3$},{0,1},$\delta$,$z_0$, {$z_2$}] mit

| $\delta$(z,a) | 0 | 1 |
|---|---|---|
| $z_0$ | { $z_0$ } | { $z_0$ ,$z_1$ } |
| $z_1$ | { $z_3$ } | { $z_2$ } |
| $z_2$ | { $z_2$ } | { $z_2$ } |
| $z_3$ | { $z_3$ } | { $z_3$ } |

# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie



## *Reguläre Sprachen*

**Welche Sprache wird von einem nicht-deterministischen endlichen Automaten akzeptierte?**

es sei $B = [Z,\Sigma,\delta,z_0,E]$ ein nicht-deterministischer endlicher Automat

zentraler Hilfsbegriff

Fortsetzung der Überführungsfunktion $\delta$

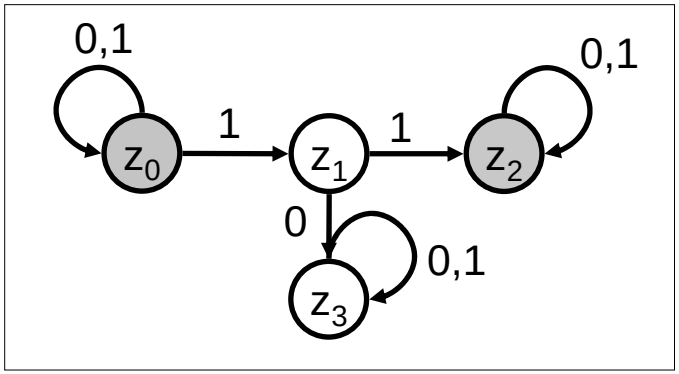
- $\delta^*(Z',\varepsilon) = Z'$ für alle $Z' \subseteq Z$
- $\delta^*(Z',xw) = \bigcup_{z \in Z'} \delta^*(\delta(z,x),w)$ für alle $Z' \subseteq Z$, $x \in \Sigma$ und $w \in \Sigma^*$



Hinweis: da B vollständig definiert ist, ist auch $\delta^*(Z',xw)$ stets definiert

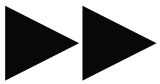
# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie

## *Reguläre Sprachen*




| $\delta^*(Z',w)$ | $\varepsilon$ | 0 | ... | 011 | ... |
|---|---|---|---|---|---|
| $\{ z_0 \}$ | $\{ z_0 \}$ | $\{ z_0 \}$ | ... | $\{ z_0,z_1,z_2 \}$ | ... |
| ... | ... | ... | ... | ... | ... |
| $\{ z_1,z_2 \}$ | $\{ z_1,z_2 \}$ | $\{ z_2,z_3 \}$ | | $\{ z_2,z_3 \}$ | |
| ... | ... | ... | ... | ... | ... |
| $\{ z_0,z_1,z_2,z_3 \}$ | $\{ z_0,z_1,z_2,z_3 \}$ | $\{ z_0,z_2,z_3 \}$ | ... | $\{ z_0,z_1,z_2,z_3 \}$ | ... |

# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie



## *Reguläre Sprachen*

es sei B = [Z,$\Sigma$,$\delta$,$z_0$,E] ein nicht-deterministischer endlicher Automat

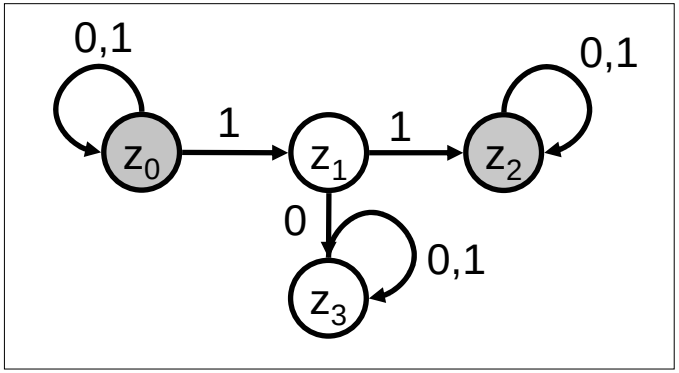
Für die vom nicht-deterministischen endlichen Automat B akzeptierte Sprache L(B) gilt:

$$L(B) = \{ w \mid w \in \Sigma^* \text{ und } \delta^*(\{ z_0 \},w) \cap E \neq \varnothing \}$$

**m.a.W.:** w gehört zu L(B), falls es einen mit dem Wort w markierten Weg durch den Automaten gibt, der in einem Startzustand beginnt und in einem Endzustand endet

**Anmerkung:** es kann mehr als einen mit w markierten Weg geben

## *Reguläre Sprachen*




L(B) = { v11w | v $\in$ { 0,1 }*, w $\in$ { 0,1 }* }, da gilt:

- $z_0 \in \delta^*(\{ z_0 \}, v)$ für alle v $\in$ { 0,1 }*
- $z_2 \in \delta^*(\{ z_0 \}, 11)$
- $z_2 \in \delta^*(\{ z_2 \}, w)$ für alle w $\in$ { 0,1 }*

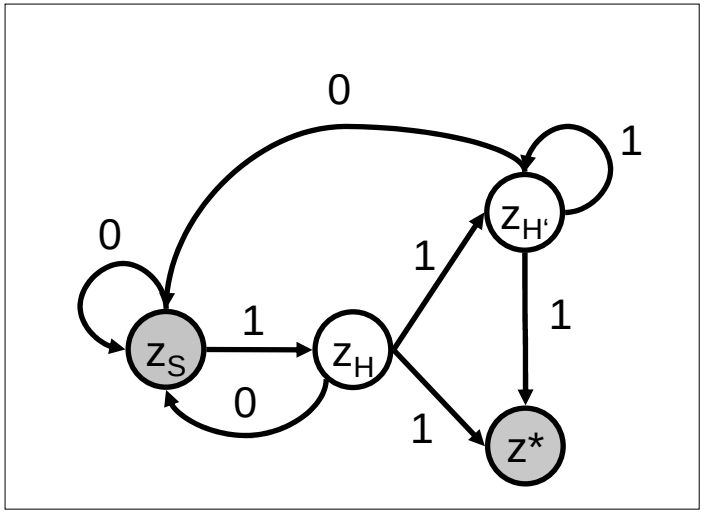
## *Reguläre Sprachen*

**Was haben nicht-deterministische endlichen Automaten mit regulären Sprachen zu tun?**

G = [Σ,V,S,R] mit

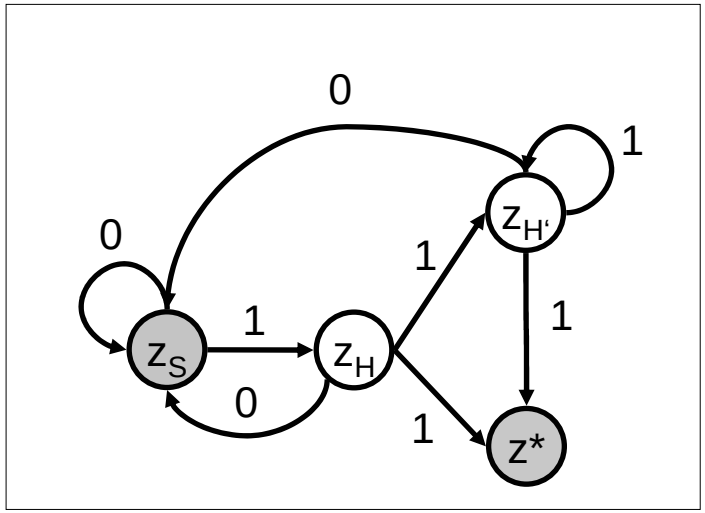
Σ = { 0, 1 }
V = { S, H, H‘ }
S

S → 0S |1H
H → 1H‘ | 0S | 1
H‘ → 0S | 1H‘ | 1

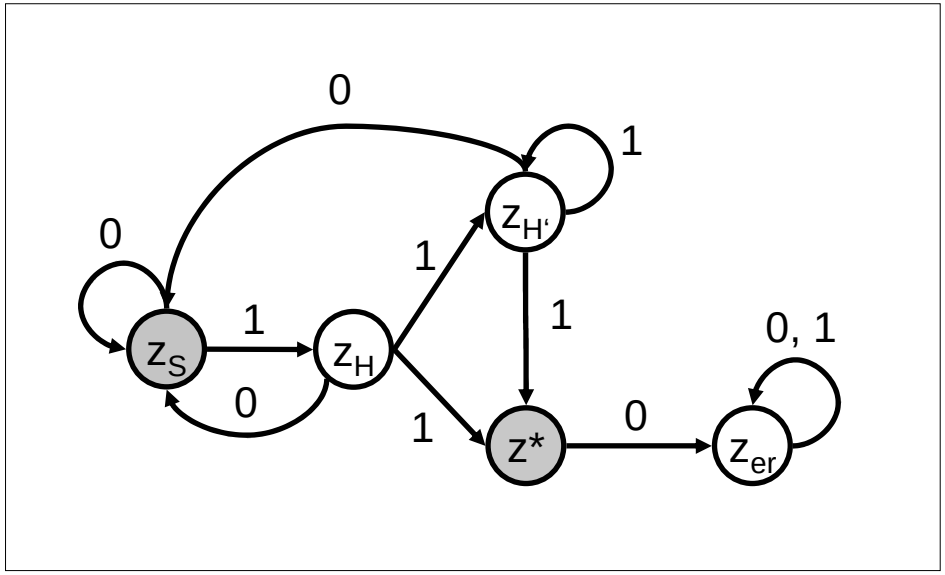


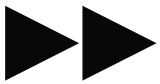
# Theoretische Informatik
# Kap 1: Formale Sprachen/Automatentheorie

## *Reguläre Sprachen*




... noch vervollständigen !!!



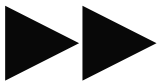


## *Reguläre Sprachen*

**allgemein**

Es sei G eine reguläre Grammatik. Dann gibt es einen nicht-deterministischen endlichen Automat B, so daß gilt: L(B) = L(G).

**m.a.W.:** zu jeder regulären Sprache L gibt es einen nicht-deterministischen endlichen Automat, der die Sprache L akzeptiert



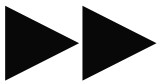
## *Reguläre Sprachen*

**zugrunde liegende Konstruktion (/* ohne Vervollständigung */)**

es sei G = [$\Sigma$,V,S,R] die gegebene Grammatik

bilde den gesuchten „NFA“ B = [Z,$\Sigma$‘,$\delta$,$z_0$,E] für L(G) wie folgt

- setze $\Sigma$‘ = $\Sigma$
- für jedes x $\in$ V nimm den Zustand $z_x$ in die Menge Z auf
- nimm zusätzlich einen neuen Zustand z* in die Menge Z auf
- setze $z_0 = H_S$ und E = { z* }
- für alle x, x‘ $\in$ V und alle a $\in$ $\Sigma$:
  - falls die Regel x → ay in R vorkommt, so nimm den Zustand $z_y$ in die Menge $\delta(z_x,a)$ auf
  - falls die Regel x → a in R vorkommt, so nimm den Zustand z* in die Menge $\delta(z_x,a)$ auf



## *Reguläre Sprachen*

**zugrunde liegende Konstruktion (/* nur Vervollständigung */)**

vervollständige den „NFA“ B = $[Z,\Sigma',\delta,z_0,E]$ für L(G) wie folgt

- nimm zusätzlich einen neuen Zustand $z_{er}$ in die Menge Z auf
- für alle $a \in \Sigma$:
  - nimm den Zustand $z_{er}$ in die Menge $\delta(z_{er},a)$ auf
- für alle $z \in Z$ und alle $a \in \Sigma$:
  - falls die Menge $\delta(z,a)$ leer ist, so nimm den Zustand $z_{er}$ in die Menge $\delta(z,a)$ auf