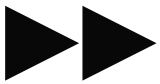
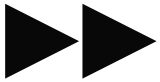
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



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



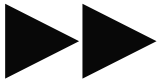
Reguläre Sprachen

es sei G = [Σ,V,S,R] eine Grammatik

G heißt reguläre Grammatik, falls für alle Regeln (l,r) $\in$ R gilt:

- |l| $\leq$ |r|
- l = x mit x $\in$ V
- r = a oder r = ax mit a $\in \Sigma$ und x $\in$ V

Eine Sprache L $\subseteq \Sigma$ heißt regulär, falls es eine reguläre Grammatik G mit L(G) = L gibt.



Reguläre Sprachen

Beispiele für reguläre Sprachen

Sprache L aller Zeichenketten, die nur aus Nullen und Einsen bestehen und in denen genau eine Eins und zwar am Ende vorkommt.

kürzer: $L = \{ v1 \mid v \in \{ 0,1 \}^* \}$

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

$S \rightarrow 0S$
$S \rightarrow 1$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen

Beispiele für 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 }* }

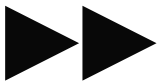
Σ = { 0, 1 }
V = { S }
S

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

kürzer für

H → 0S
H → 1H‘
H → 1

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Reguläre Sprachen

Beispiele für reguläre Sprachen

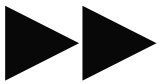
Sprache L aller Zeichenketten, die Programmiersprache C++ als Bezeichner verwendet werden können.

Vom Programmierer definierte Größen werden durch Namen angesprochen.
Diese Namen werden nach folgenden Regeln gebildet:

- Sie dürfen mit einem Buchstaben oder mit einem _ beginnen.
- Sie dürfen beliebige Zeichen (/* außer dem Leerzeichen */) enthalten.
- Sie dürfen keine Schlüsselwörter überdecken.

... zur Vereinfachung

- Σ = { _, a, b }
- es gibt nur das Schlüsselwort abb



Reguläre Sprachen

Beispiele für reguläre Sprachen

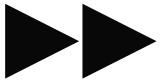
Sprache L aller Zeichenketten, die mit

- a, b oder _ beginnen und in denen das Zeichen _ höchstens als erstes Zeichen auftaucht
- die die Zeichenkette abb nicht enthalten

$L = L_1 \setminus L_2$, wobei gilt

- L_1 ist die Menge aller Zeichenketten, die mit a, b oder _ beginnen und in denen das Zeichen _ höchstens als erstes Zeichen auftaucht
- L_2 die Menge alle Zeichenketten bezeichne, welche die Zeichenkette abb enthalten

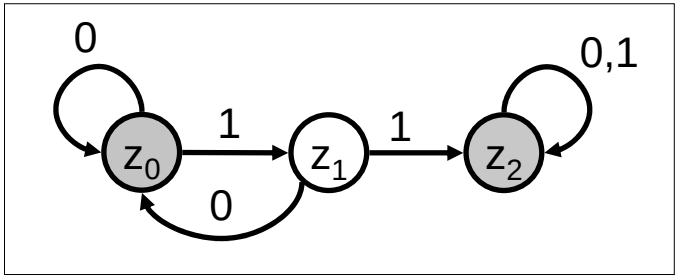
... offenbar sind L_1 und L_2 regulär
... wir werden sehen, daß deshalb auch L regulär ist



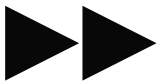
Reguläre Sprachen

... algorithmische Frage: das Membership-Problem

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



deterministischer endlicher Automat für die Sprache L



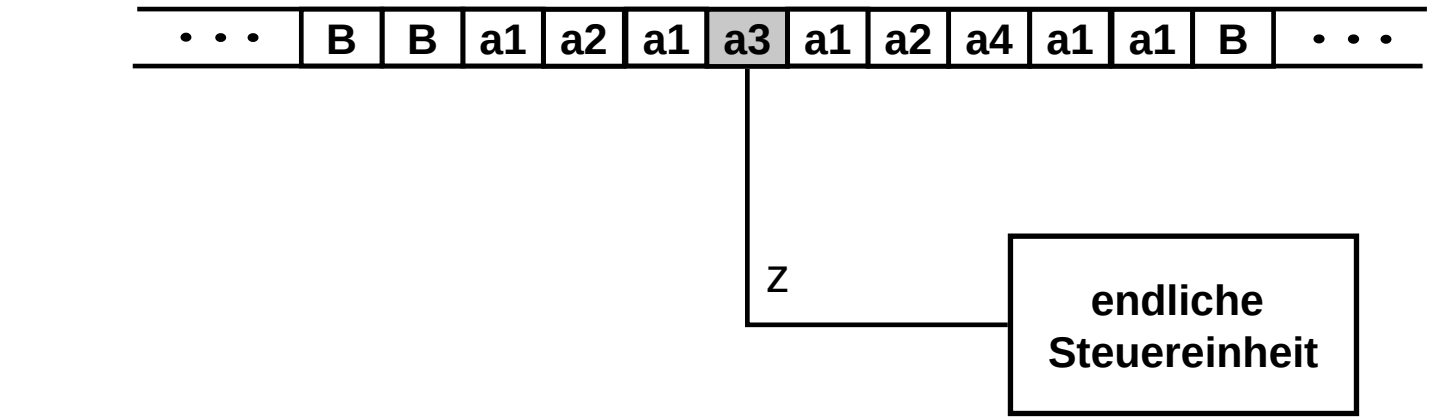
Reguläre Sprachen

... algorithmische Frage: das Membership-Problem

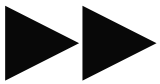
wir werden sehen, daß es zu jeder regulären Sprache L einen deterministischen endlichen Automaten A gibt, mit dem man das Membership-Problem für die Sprache L effizient lösen kann

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen



- auf dem Eingabeband steht das Wort w, für welches zu entscheiden ist, ob es zur Sprache L gehört oder nicht
- das Wort w wird zeichenweise von links nach rechts gelesen
- falls sich der deterministische endliche Automat nach Lesen des letzten Buchstabens des Wortes w in einem Endzustand befindet, so gehört w zu L; andernfalls gehört w nicht zu L



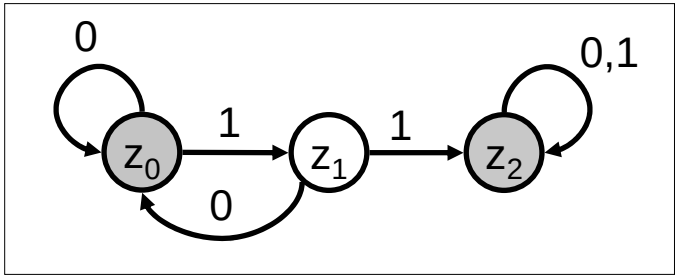
Reguläre Sprachen

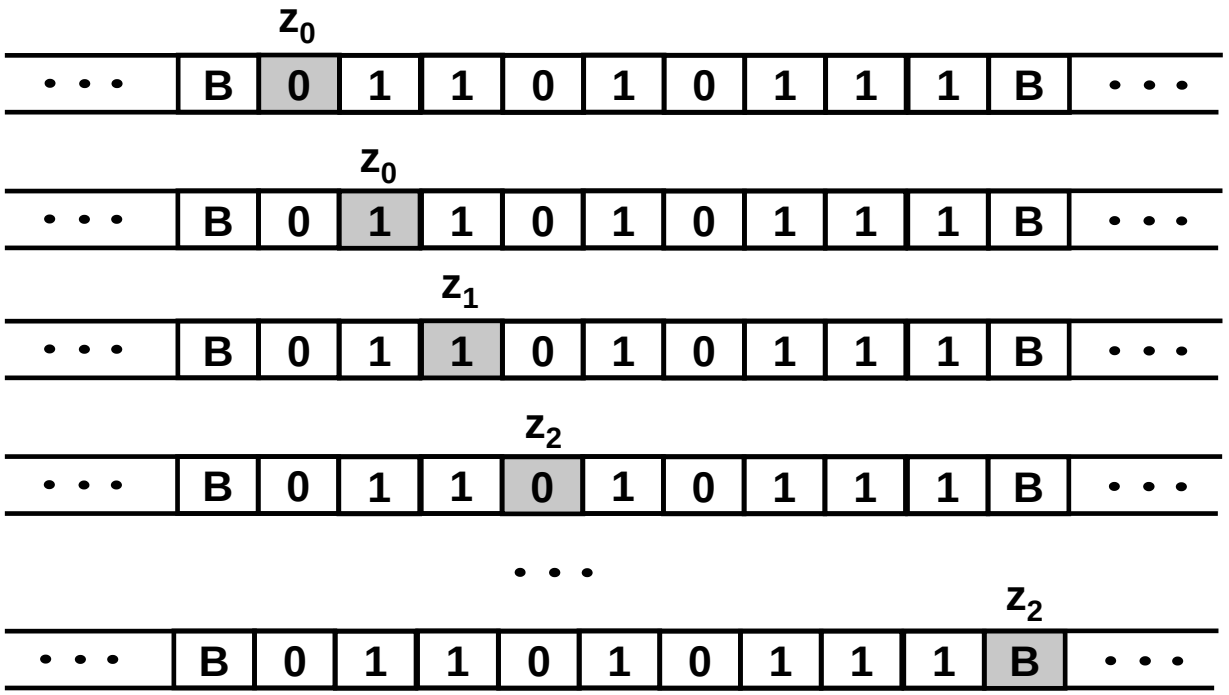
- das Eingabeband ist in Felder unterteilt; jedes Feld enthält ein Zeichen
- endliche Steuereinheit; mit einem Lesekopf verbunden
  - bekommt Information über den Inhalt eines Feldes
  - befindet sich in einem von endlich vielen Zuständen
  - legt in Abhängigkeit vom aktuellen Zustand und vom Inhalt des aktuell gesehenen Feldes fest, was im nächsten Arbeitsschritt passiert
- ein Arbeitsschritt besteht aus:
  - Bewegung des Lesekopfs um ein Feld nach rechts
  - Festlegung des nächsten Zustands

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

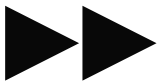
Reguläre Sprachen

$L = \{ v11w \mid v \in \{ 0,1 \}^*, w \in \{ 0,1 \}^* \}$





Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Reguläre Sprachen

Bestimmungsstücke eines deterministischen endlichen Automaten

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

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

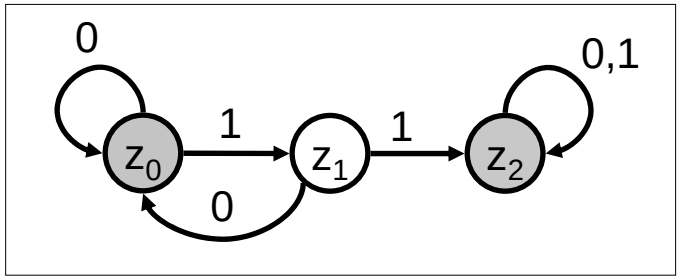
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen

Bestimmungsstücke eines endlichen deterministischen Automaten

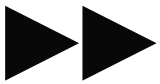
- eine Menge Z von Zuständen
- ein endliches Eingabealphabet Σ von Symbolen
- ein ausgezeichneter Startzustand $z_0 \in Z$
- eine ausgezeichnete Menge $E \subseteq Z$ (/* die Endzustände */)
- eine Überführungsfunktion δ: $Z \times \Sigma \rightarrow Z$, wobei für alle $z \in Z$ und alle $a \in \Sigma$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

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



$\delta(z,a)$	0	1
z_0	z_0	z_1
z_1	z_0	z_2
z_2	z_2	z_2

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Reguläre Sprachen

Welche Sprache wird von einem deterministischen endlichen Automaten akzeptierte?

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

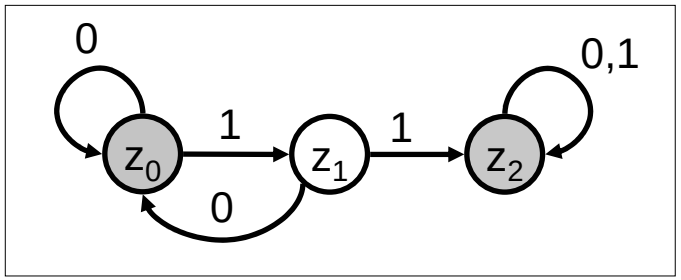
zentraler Hilfsbegriff

Fortsetzung der Überführungsfunktion δ

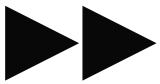
- $\delta^*(z,\varepsilon) = z$ für alle $z \in Z$
- $\delta^*(z,xw) = \delta^*(\delta(z,x),w)$ für alle $z \in Z$, $x \in \Sigma$ und $w \in \Sigma^*$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen



$\delta^*(z,w)$	ε	0	1	00	...	110	...
z_0	z_0	z_0	z_0	z_0		z_2	...
z_1	z_1	z_0	z_2	z_0		z_2	...
z_2	z_2	z_2	z_2	z_2		z_2	...



Reguläre Sprachen

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

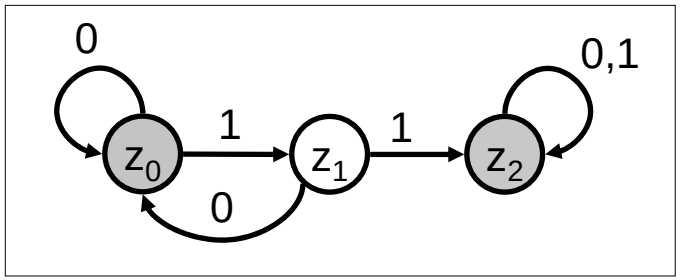
Für die vom deterministischen endlichen Automat A akzeptierte Sprache L(A) gilt:

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

m.a.W.: w gehört zu L(A), 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 nur einen mit w markierten Weg geben

Reguläre Sprachen



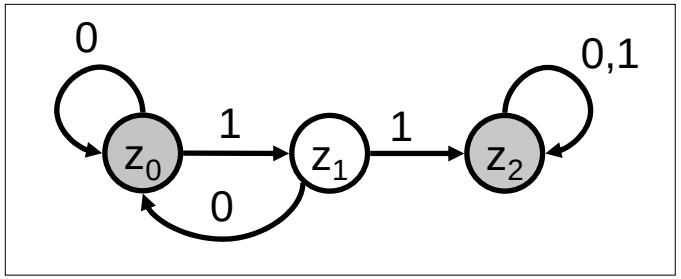
L(A) = { v11w | v $\in$ { 0,1 }*, w $\in$ { 0,1 }* }, da gilt:

- $\delta^*(z_0,v) \in \{ z_0, z_1, z_2 \}$ für alle $v \in \{ 0,1 \}^*$
- $\delta^*(z,11) = z_2$ für alle $z \in \{ z_0, z_1, z_2 \}$
- $\delta^*(z_2,w) = z_2$ für alle $w \in \{ 0,1 \}^*$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Reguläre Sprachen

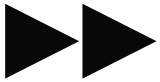
Was haben deterministische endlichen Automaten mit regulären Sprachen zu tun?



G = [Σ,V,S,R] mit

Σ = { 0, 1 }
V = { S, H, H‘ }
S

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

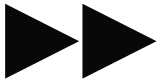


Reguläre Sprachen

allgemein

Es sei A ein deterministischer endlicher Automat. Dann gibt es eine reguläre Grammatik G, so daß gilt: L(G) = L(A).

m.a.W.: jede von einem deterministischen endlichen Automaten akzeptierte Sprache ist eine reguläre Sprache



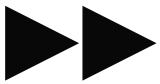
Reguläre Sprachen

zugrunde liegende Konstruktion

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

bilde die gesuchte Grammatik $G = [\Sigma',V,S,R]$ mit $L(G) = L(A)$ wie folgt

- setze $\Sigma' = \Sigma$
- für jedes $z \in Z$ nimm die Variable H_z in die Menge V auf
- setze $S = H_{z_0}$
- für alle $z, z' \in Z$ und alle $a \in \Sigma$:
  - falls $\delta(z,a) = z'$, so nimm die Regel $H_z \rightarrow aH_{z'}$ in die Menge R auf
  - falls $\delta(z,a) = z'$ und $z' \in E$, so nimm die Regel $H_z \rightarrow a$ in die Menge R auf



Reguläre Sprachen

Wofür sind deterministische endlichen Automaten sonst noch gut?

- Modellierung technischer Systeme

  ... das spiegelt sich etwa in Modellierungssprachen für den objektorientierten Software-Entwurf wider; bspw. die Zustandsdiagramme in UML

- Realisierung effizienter String-Matching-Verfahren

  ... Details siehe Foliensatz „Einschub 1“