Grundlagen der Programmierung (Vorlesung 22)

Ralf Möller, FH-Wedel

- Vorige Vorlesung
  - Formale Sprachen, Grammatik, Grammatiktypen Wortproblem, Ableitungsproblem, Entscheidbarkeit für Typ 1
- Inhalt dieser Vorlesung
  - Automatentheorie
- Lernziele
  - Kennenlernen von abstrakten Maschinen zur Generierung von Sprachen oder zur Entscheidung des Wortproblems

Danksagung

- Die Präsentationen sind an den Inhalt des Buches "Theoretische Informatik kurzgefaßt" von Uwe Schöning angeleht und wurden aus den Unterlagen zu der Vorlesung "Informatik IV - Theoretische Informatik" an der TU München von Angelika Steger übernommen
- Die Originalunterlagen befinden sich unter: http://www14.in.tum.de/lehre/2000SS/info4/

Entscheidungsprobleme

Abstrakt kann man diese Probleme wie folgt formulieren:

<u>WORTPROBLEM</u>

Gegeben: Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$.

Frage: Gilt $w \in L(G)$?

<u>ABLEITUNGSPROBLEM</u>

Gegeben: Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in L(G)$.

Aufgabe: Konstruiere eine Ableitung von w, d.h., Worte

$w_0, w_1, \ldots, w_n \in (\Sigma \cup V)^*$ mit $w_0 = S$,

$w_n = w$ und $w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$.

Entscheidbarkeit des Wortproblems für Typ 1-Grammatiken

Satz: Für kontextsensitive Sprachen ist das Wortproblem entscheidbar.

Genauer: Es gibt einen Algorithmus, der bei Eingabe einer kontextsensitiven Grammatik G und eines Wortes w in endlicher Zeit entscheidet, ob $w \in L(G)$.

Beweisidee: Angenommen $w \in L(G)$. Dann gibt es $w_0, w_1, \ldots, w_n \in (\Sigma \cup V)^*$ mit $w_0 = S$, $w_n = w$ und $w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$.

WICHTIG: Da G kontextsensitiv ist, gilt $|w_0| \leq |w_1| \leq \ldots \leq |w_n|$. D.h., es genügt alle Worte in $L(G)$ der Länge $\leq n$ zu erzeugen.

Beweis (1)

Formaler Beweis: Definiere

$$T_m^n := \{w \in (\Sigma \cup V)^* \mid |w| \leq n \text{ und } w \text{ läßt sich aus } S \text{ in}$$
$$\text{höchstens } m \text{ Schritten herleiten } \}$$

Diese Mengen kann man für alle n induktiv wie folgt berechnen:

$$\begin{aligned} T_0^n &:= \{S\} \\ T_{m+1}^n &:= T_m^n \cup \{w \in (\Sigma \cup V)^* \mid |w| \leq n \text{ und} \\ &\qquad\qquad w' \Rightarrow w \text{ für ein } w' \in T_m^n\} \end{aligned}$$

Beachte: Für alle m gilt: $|T_m^n| \leq \sum_{i=1}^{n} |\Sigma \cup V|^i$.

Es muß daher immer ein m_0 geben mit $T_{m_0}^n = T_{m_0+1}^n = \ldots$.

Beweis (2)

Algorithmus:

$n := |w|;$

$T_0^n := \{S\};\ m := 0;$

repeat

$\quad T_{m+1}^n := <\text{wie oben}>;\ m := m + 1;$

until $T_m^n = T_{m-1}^n$ or $w \in T_m^n;$

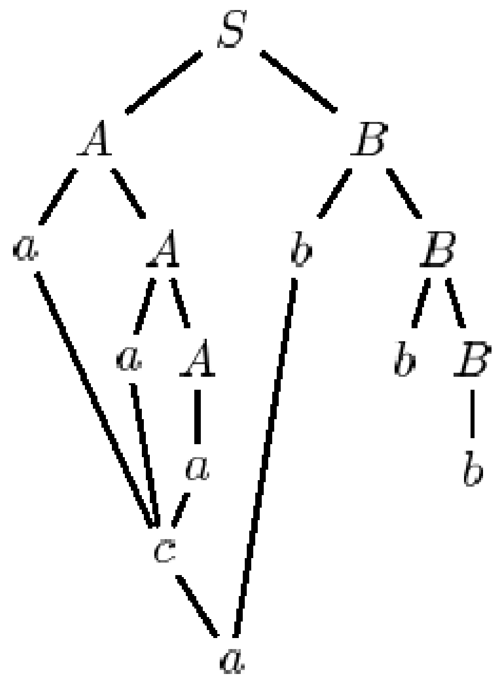
$w \in T_m^n$

Ableitungsgraphen für allgemeine Grammatiken

Grammatik:

$$\begin{array}{rcl} S & \to & AB \\ A & \to & aA \\ A & \to & a \\ B & \to & bB \\ B & \to & b \\ aaa & \to & c \\ cb & \to & a \end{array}$$

Beispiel:



Die Terminale ohne Kante nach unten entsprechen, von links nach rechts gelesen, dem durch den Ableitungsgraphen dargestellten Wort.

In obigem Beispiel gilt also $abb \in L(G)$. Der Ableitungsgraph entspricht der Ableitung

$$S \Rightarrow AB \Rightarrow aAB \Rightarrow aAbB \Rightarrow aaAbB \Rightarrow aaAbbB \Rightarrow aaabbB \Rightarrow aaabbb \Rightarrow cbbb \Rightarrow abb$$

Ableitungsbäume

Beobachtung: Bei kontextfreien Sprachen sind die Ableitungsgraphen immer Bäume.

Beispiel: Grammatik:

$$\begin{aligned} S &\rightarrow aB \\ S &\rightarrow Ac \\ A &\rightarrow ab \\ B &\rightarrow bc \end{aligned}$$

Mehrfache Ableitungsmöglichkeiten

Für das Wort abc gibt es zwei verschiedene Ableitungsbäume:

Linksableitung

Definition: Eine Ableitung $S = w_0 \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_n$ eines Wortes w_n heißt *Linksableitung*, wenn für jede Anwendung einer Produktion $y \rightarrow y'$ auf Worte $w_i = xyz$, $w_{i+1} = xy'z$ gilt: Auf jedes echte Präfix von xy läßt sich *keine* Regel anwenden.

Man sagt: eine Grammatik ist *eindeutig* wenn es für jedes Wort $w \in L(G)$ genau eine eindeutige bestimmte Linksableitung gibt. Nicht eindeutige Grammatiken nennt man auch *mehrdeutig*.

Beispiel (1)

1. Beispiel: Die zugehörige Linksableitung ist:

$S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaaB \Rightarrow cB \Rightarrow cbB \Rightarrow$
$aB \Rightarrow abB \Rightarrow abb$

Eine andere Linksableitung ist:

$$S \Rightarrow AB \Rightarrow aB \Rightarrow abB \Rightarrow abb.$$

Die Grammatik ist also mehrdeutig.

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow a \\ B &\rightarrow bB \\ B &\rightarrow b \\ aaa &\rightarrow c \\ cb &\rightarrow a \end{aligned}$$

Beispiel (2)

2. Beispiel: Beide Ableitungsbäume entsprechen Linksableitungen:

$$S \Rightarrow Ac \Rightarrow abc \quad \text{bzw.} \quad S \Rightarrow aB \Rightarrow abc.$$

Die Grammatik ist also ebenfalls mehrdeutig.

$$\begin{aligned} S &\rightarrow aB \\ S &\rightarrow Ac \\ A &\rightarrow ab \\ B &\rightarrow bc \end{aligned}$$

Deterministischer Endlicher Automat

Definition. Ein *deterministischer endlicher Automat* (englisch: deterministic finite automata, kurz DFA) wird durch ein 5-Tupel $M = (Z, \Sigma, \delta, z_0, E)$ beschrieben, das folgende Bedingungen erfüllt:

- Z ist eine endliche Menge von *Zuständen*.
- Σ ist eine endliche Menge, das *Eingabealphabet*, wobei $Z \cap \Sigma = \emptyset$.
- $z_0 \in Z$ ist der *Startzustand*.
- $E \subseteq Z$ ist die Menge der *Endzustände*.
- $\delta : Z \times \Sigma \to Z$ heißt *Übergangsfunktion*.

Akzeptierte Sprache

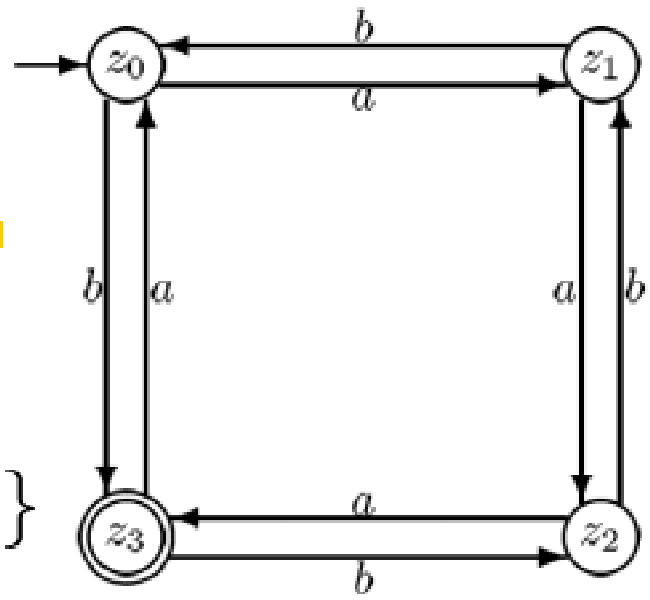
Die von M *akzeptierte* Sprache ist

$$L(M) := \{w \in \Sigma^* \mid \hat{\delta}(z_0, w) \in E\},$$

wobei $\hat{\delta} : Z \times \Sigma^* \to Z$ induktiv definiert ist durch

$$\begin{aligned} \hat{\delta}(z, \epsilon) &= z \\ \hat{\delta}(z, ax) &= \hat{\delta}(\delta(z, a), x) \end{aligned}$$

Beispiel (1)



Sei $M = (Z, \Sigma, \delta, z_0, E)$, wobei

$$\begin{aligned} Z &= \{z_0, z_1, z_2, z_3\} \\ \Sigma &= \{a, b\} \\ E &= \{z_3\} \end{aligned}$$

$$\begin{aligned} \delta(z_0, a) &= z_1 & \delta(z_2, a) &= z_3 \\ \delta(z_0, b) &= z_3 & \delta(z_2, b) &= z_1 \\ \delta(z_1, a) &= z_2 & \delta(z_3, a) &= z_0 \\ \delta(z_1, b) &= z_0 & \delta(z_3, b) &= z_2 \end{aligned}$$

Graphische Darstellung

Bem.: Endliche Automaten können durch (gerichtete und beschriftete) *Zustandsgraphen* veranschaulicht werden:

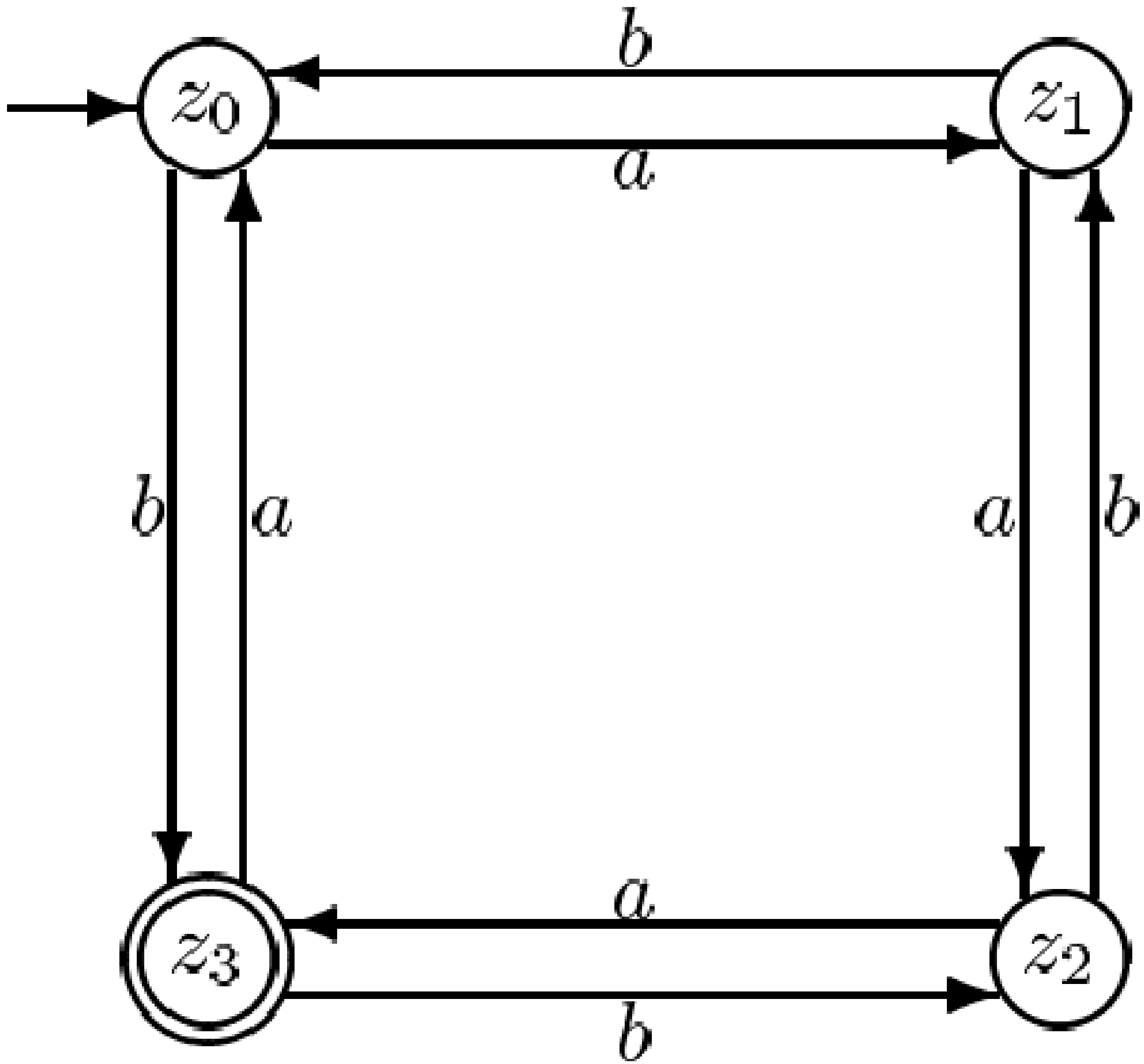
Knoten	$\hat{=}$	Zustände
Kanten	$\hat{=}$	Übergänge

[genauer: Kante (u, v), beschriftet mit $a \in \Sigma$,

entspricht $\delta(u, a) = v$]

Endzustände werden durch doppelte Kreise gekennzeichnet.

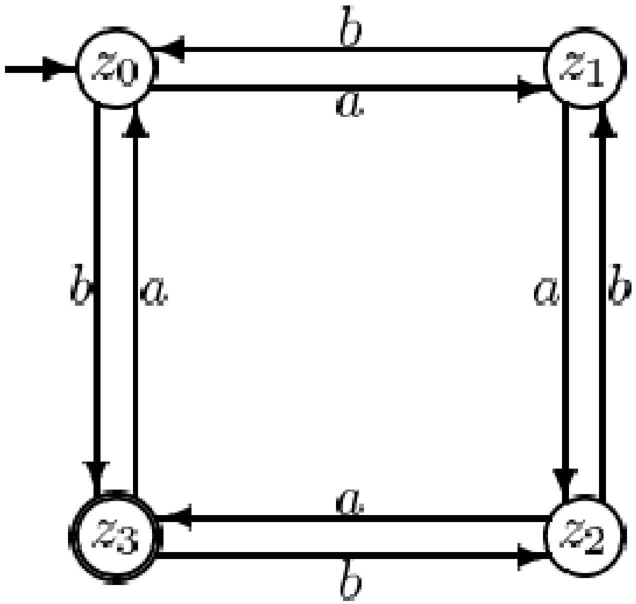
Beispiel (2)



Satz. Ist $M = (Z, \Sigma, \delta, z_0, E)$ ein deterministischer endlicher Automat, so ist die durch

$$P \quad := \quad \{z \rightarrow az' \mid \delta(z,a) = z'\} \cup \{z \rightarrow a \mid \delta(z,a) \in E\}$$

gegebene Grammatik $G = (Z, \Sigma, z_0, P)$ regulär. und L(M) = L(G).



Produktionen:

$$z_0 \rightarrow az_1, \quad z_0 \rightarrow bz_3, \quad z_1 \rightarrow az_2,$$

$$z_1 \rightarrow bz_0, \quad z_2 \rightarrow az_3, \quad z_2 \rightarrow bz_1,$$

$$z_3 \rightarrow az_0, \quad z_3 \rightarrow bz_2,$$

$$z_2 \rightarrow a, \quad z_0 \rightarrow b.,$$

Beispiel: $z_0 \Rightarrow az_1 \Rightarrow abz_0 \Rightarrow abaz_1 \Rightarrow abaaz_2 \Rightarrow abaaa \in L(G)$

Was kommt beim nächsten Mal?

- Zusammenhänge zwischen regulären Ausdrücken, endlichen Automaten und regulären Grammatiken
- "Nicht-Regularität" einiger Sprachen