

Vorlesung
Grundlagen der Theoretischen Informatik /
Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik



Sommersemester 2007

Dank

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

– *Bernhard Beckert, April 2007*

Beispiel 7.1

Die Grammatik $G = (\{S\}, \{a, b\}, R, S)$ mit

$$R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$$

erzeugt die Sprache $\{vv^R \mid v \in \{a, b\}^+\}$

Betrachten wir das Wort $w = abbaabba$.

Was sind mögliche letzte Schritte von Ableitungen, die zu w geführt haben können?

Wir merken uns alle möglichen einzelnen Ableitungsschritte in einer Chart, um Mehrfacharbeit zu vermeiden.

Wenn das Wort w in der Sprache $L(G)$ ist, enthält am Ende der Chart eine mit S markierte Kante, die vom ersten bis zum letzten Knoten reicht.

Beispiel 7.1

Die Grammatik $G = (\{S\}, \{a, b\}, R, S)$ mit

$$R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$$

erzeugt die Sprache $\{vv^R \mid v \in \{a, b\}^+\}$

Betrachten wir das Wort $w = abbaabba$.

Was sind mögliche letzte Schritte von Ableitungen, die zu w geführt haben können?

Wir merken uns alle möglichen einzelnen Ableitungsschritte in einer Chart, um Mehrfacharbeit zu vermeiden.

Wenn das Wort w in der Sprache $L(G)$ ist, enthält am Ende der Chart eine mit S markierte Kante, die vom ersten bis zum letzten Knoten reicht.

Beispiel 7.1

Die Grammatik $G = (\{S\}, \{a, b\}, R, S)$ mit

$$R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$$

erzeugt die Sprache $\{vv^R \mid v \in \{a, b\}^+\}$

Betrachten wir das Wort $w = abbaabba$.

Was sind mögliche letzte Schritte von Ableitungen, die zu w geführt haben können?

Wir merken uns alle möglichen einzelnen Ableitungsschritte in einer Chart, um Mehrfacharbeit zu vermeiden.

Wenn das Wort w in der Sprache $L(G)$ ist, enthält am Ende der Chart eine mit S markierte Kante, die vom ersten bis zum letzten Knoten reicht.

Beispiel 7.1

Die Grammatik $G = (\{S\}, \{a, b\}, R, S)$ mit

$$R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$$

erzeugt die Sprache $\{vv^R \mid v \in \{a, b\}^+\}$

Betrachten wir das Wort $w = abbaabba$.

Was sind mögliche letzte Schritte von Ableitungen, die zu w geführt haben können?

Wir merken uns alle möglichen einzelnen Ableitungsschritte in einer Chart, um Mehrfacharbeit zu vermeiden.

Wenn das Wort w in der Sprache $L(G)$ ist, enthält am Ende der Chart eine mit S markierte Kante, die vom ersten bis zum letzten Knoten reicht.

Beispiel 7.1

Die Grammatik $G = (\{S\}, \{a, b\}, R, S)$ mit

$$R = \{ S \rightarrow aSa \mid bSb \mid aa \mid bb \}$$

erzeugt die Sprache $\{vv^R \mid v \in \{a, b\}^+\}$

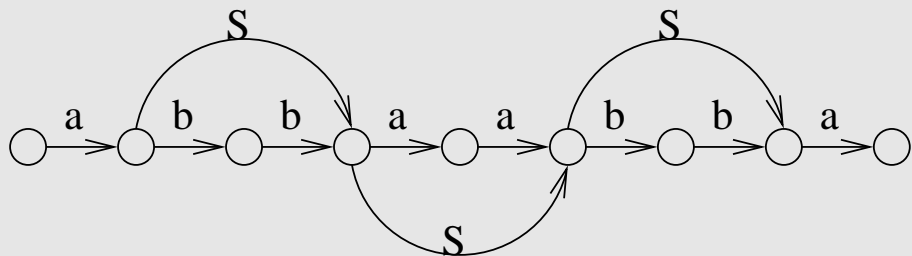
Betrachten wir das Wort $w = abbaabba$.

Was sind mögliche letzte Schritte von Ableitungen, die zu w geführt haben können?

Wir merken uns alle möglichen einzelnen Ableitungsschritte in einer Chart, um Mehrfacharbeit zu vermeiden.

Wenn das Wort w in der Sprache $L(G)$ ist, enthält am Ende der Chart eine mit S markierte Kante, die vom ersten bis zum letzten Knoten reicht.

Beispiel (Forts.)



Beispiel (Forts.)

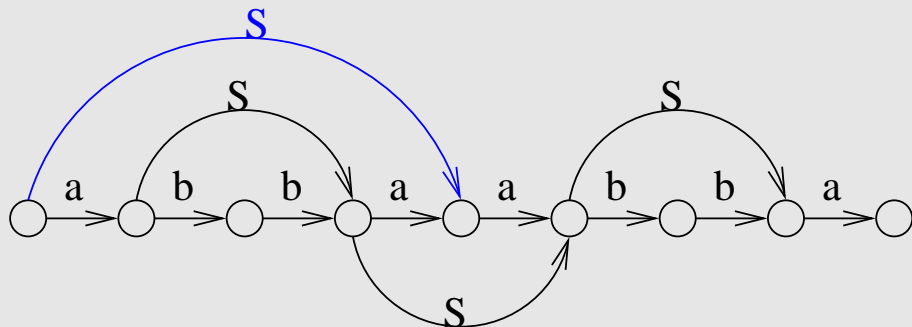


Chart-Parsing

Beispiel (Forts.)

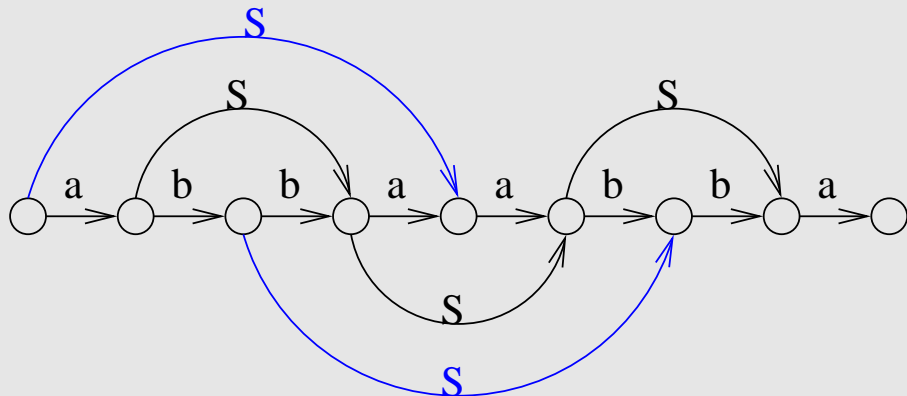


Chart-Parsing

Beispiel (Forts.)

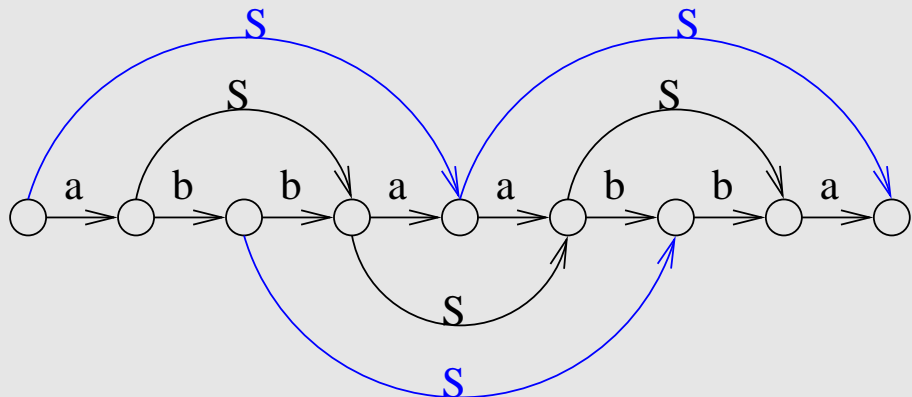


Chart-Parsing

Beispiel (Forts.)

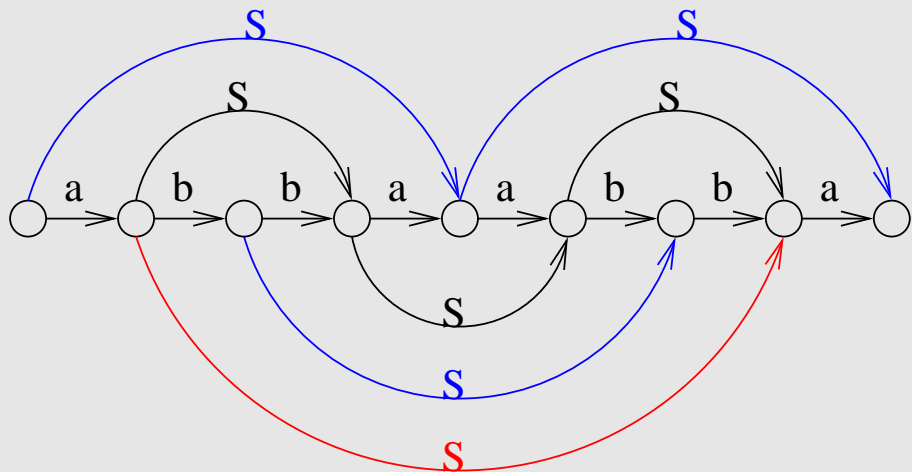
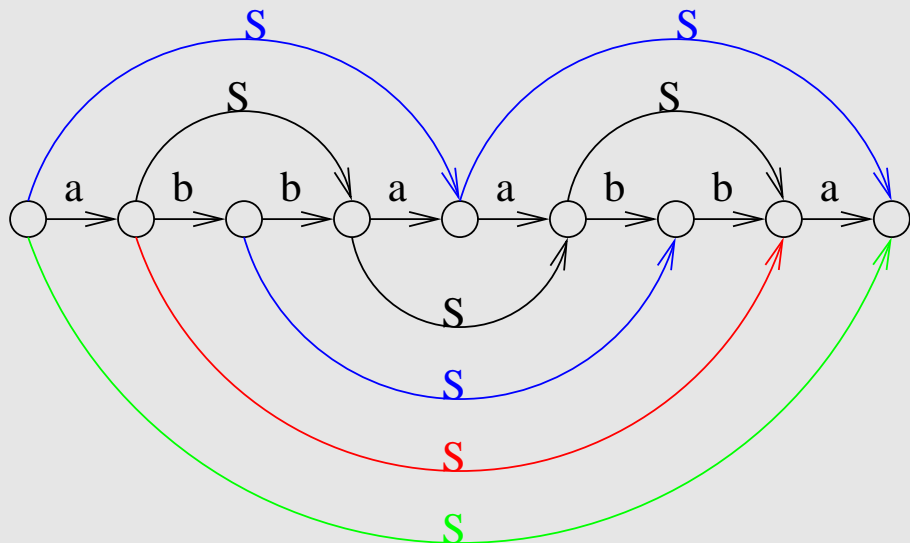


Chart-Parsing

Beispiel (Forts.)



Zur Vereinfachung

Wir fordern:

Grammatik ist in **Chomsky-Normalform**.

Dann:

Immer nur **zwei** benachbarte Kanten betrachten, um herauszufinden, ob darüber eine neue Kante eingefügt werden kann.

Zur Vereinfachung

Wir fordern:

Grammatik ist in **Chomsky-Normalform**.

Dann:

Immer nur **zwei** benachbarte Kanten betrachten, um herauszufinden, ob darüber eine neue Kante eingefügt werden kann.

Beispiel (Forts.)

Grammatik in CNF, die dieselbe Sprache wie oben erzeugt:

$G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$ mit

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Darstellung als Array

Für eine Kante, die den i . bis j . Buchstaben überspannt und mit A markiert ist, steht im $[i, j]$ -Element des Arrays die Eintragung A .

Definition 7.2 ($M * N$)

Sei $L = L(G)$ kontextfrei, und $G = (V, T, R, S)$ in Chomsky-Normalform. Mit $M, N \subseteq V$ sei

$$M * N := \{A \in V \mid \exists B \in M, \exists C \in N : A \rightarrow BC \in R\}$$

Darstellung als Array

Für eine Kante, die den i . bis j . Buchstaben überspannt und mit A markiert ist, steht im $[i, j]$ -Element des Arrays die Eintragung A .

Definition 7.2 ($M * N$)

Sei $L = L(G)$ kontextfrei, und $G = (V, T, R, S)$ in Chomsky-Normalform. Mit $M, N \subseteq V$ sei

$$M * N := \{A \in V \mid \exists B \in M, \exists C \in N : A \rightarrow BC \in R\}$$

Definition 7.3 ($w_{i,j}$, $V_{i,j}$)

Sei $w = a_1 \dots a_n$ mit $a_i \in \Sigma$.

Dann:

- $w_{i,j} := a_i \dots a_j$ ist das Infix von w vom i -ten bis zum j -ten Buchstaben
- $V_{i,j} := \{A \in V \mid A \Rightarrow_G^* w_{i,j}\}$

Definition 7.3 ($w_{i,j}$, $V_{i,j}$)

Sei $w = a_1 \dots a_n$ mit $a_i \in \Sigma$.

Dann:

- $w_{i,j} := a_i \dots a_j$ ist das Infix von w vom i -ten bis zum j -ten Buchstaben
- $V_{i,j} := \{A \in V \mid A \xRightarrow{*}_G w_{i,j}\}$

Lemma 7.4

Sei $w = a_1 \dots a_n$, $a_i \in \Sigma$, d.h. $|w| = n$. Dann gilt:

① $V_{i,i} = \{A \in V \mid A \rightarrow a_i \in R\}$

② $V_{i,k} = \bigcup_{j=i}^{k-1} V_{i,j} * V_{j+1,k}$ für $1 \leq i < k \leq n$

Beachte:

Die Grammatik muss in Chomsky-Normalform sein!

Lemma 7.4

Sei $w = a_1 \dots a_n$, $a_i \in \Sigma$, d.h. $|w| = n$. Dann gilt:

- 1 $V_{i,i} = \{A \in V \mid A \rightarrow a_i \in R\}$
- 2 $V_{i,k} = \bigcup_{j=i}^{k-1} V_{i,j} * V_{j+1,k}$ für $1 \leq i < k \leq n$

Beachte:

Die Grammatik muss in Chomsky-Normalform sein!

Lemma 7.4

Sei $w = a_1 \dots a_n$, $a_i \in \Sigma$, d.h. $|w| = n$. Dann gilt:

- 1 $V_{i,i} = \{A \in V \mid A \rightarrow a_i \in R\}$
- 2 $V_{i,k} = \bigcup_{j=i}^{k-1} V_{i,j} * V_{j+1,k}$ für $1 \leq i < k \leq n$

Beachte:

Die Grammatik muss in Chomsky-Normalform sein!

Lemma 7.4

Sei $w = a_1 \dots a_n$, $a_i \in \Sigma$, d.h. $|w| = n$. Dann gilt:

- 1 $V_{i,i} = \{A \in V \mid A \rightarrow a_i \in R\}$
- 2 $V_{i,k} = \bigcup_{j=i}^{k-1} V_{i,j} * V_{j+1,k}$ für $1 \leq i < k \leq n$

Beachte:

Die Grammatik muss in Chomsky-Normalform sein!

Beweis

1 $V_{i,i} = \{A \in V \mid A \Longrightarrow_G^* a_i\} = \{A \in V \mid A \rightarrow a_i \in R\}$, da G in CNF ist.

$A \in V_{i,k}$ mit $1 \leq i < k \leq n$

gdw $A \Longrightarrow_G^* a_i \dots a_k$

gdw $\exists j, i \leq j < k : \exists B, C \in V : A \Longrightarrow BC$, und

$B \Longrightarrow_G^* w_{i,j} \neq \varepsilon$

und $C \Longrightarrow_G^* w_{j+1,k} \neq \varepsilon$ (da G in CNF ist)

gdw $\exists j, i \leq j < k : \exists B, C \in V : A \Longrightarrow BC$

und $B \in V_{i,j}$ und $C \in V_{j+1,k}$

2 gdw $\exists j, i \leq j < k : A \in V_{i,j} * V_{j+1,k}$



Beweis

① $V_{i,i} = \{A \in V \mid A \Longrightarrow_G^* a_i\} = \{A \in V \mid A \rightarrow a_i \in R\}$, da G in CNF ist.

$A \in V_{i,k}$ mit $1 \leq i < k \leq n$

gdw $A \Longrightarrow_G^* a_i \dots a_k$

gdw $\exists j, i \leq j < k : \exists B, C \in V : A \Longrightarrow BC$, und

$B \Longrightarrow_G^* w_{i,j} \neq \varepsilon$

und $C \Longrightarrow_G^* w_{j+1,k} \neq \varepsilon$ (da G in CNF ist)

gdw $\exists j, i \leq j < k : \exists B, C \in V : A \Longrightarrow BC$

und $B \in V_{i,j}$ und $C \in V_{j+1,k}$

② gdw $\exists j, i \leq j < k : A \in V_{i,j} * V_{j+1,k}$



Teil IV

- 1 Ableitungsbäume
- 2 Umformung von Grammatiken
- 3 Normalformen
- 4 Pumping-Lemma für kontextfreie Sprachen
- 5 Pushdown-Automaten (PDAs)
- 6 Abschlusseigenschaften
- 7 Wortprobleme**
- 8 Der CYK-Algorithmus

Teil IV

- 1 Ableitungsbäume
- 2 Umformung von Grammatiken
- 3 Normalformen
- 4 Pumping-Lemma für kontextfreie Sprachen
- 5 Pushdown-Automaten (PDAs)
- 6 Abschlusseigenschaften
- 7 Wortprobleme
- 8 Der CYK-Algorithmus**

CYK-Algorithmus (Cocke-Younger-Kasami)

Algorithmus

Input sei eine **Grammatik G in CNF** und ein **Wort $w = a_1 \dots a_n \in \Sigma^*$** .

(i) **for** $i := 1$ **to** n **do** */*Regeln $A \rightarrow a$ eintragen */*
 $V_{i,i} := \{A \in V \mid A \rightarrow a_i \in R\}$

(ii) **for** $h := 1$ **to** $n - 1$ **do**
 for $i := 1$ **to** $n - h$ **do**

$$V_{i,i+h} = \bigcup_{j=i}^{i+h-1} V_{i,j} * V_{j+1,i+h}$$

(iii) **if** $S \in V_{1,n}$ **then return** Ausgabe $w \in L(G)$
 else return Ausgabe $w \notin L(G)$

CYK-Algorithmus (Cocke-Younger-Kasami)

Algorithmus

Input sei eine Grammatik G in CNF und ein Wort $w = a_1 \dots a_n \in \Sigma^*$.

(i) **for** $i := 1$ **to** n **do** */*Regeln $A \rightarrow a$ eintragen */*

$$V_{i,i} := \{A \in V \mid A \rightarrow a_i \in R\}$$

(ii) **for** $h := 1$ **to** $n - 1$ **do**

for $i := 1$ **to** $n - h$ **do**

$$V_{i,i+h} = \bigcup_{j=i}^{i+h-1} V_{i,j} * V_{j+1,i+h}$$

(iii) **if** $S \in V_{1,n}$ **then return** Ausgabe $w \in L(G)$

else return Ausgabe $w \notin L(G)$

CYK-Algorithmus (Cocke-Younger-Kasami)

Algorithmus

Input sei eine Grammatik G in CNF und ein Wort $w = a_1 \dots a_n \in \Sigma^*$.

(i) **for** $i := 1$ **to** n **do** */*Regeln $A \rightarrow a$ eintragen */*

$$V_{i,i} := \{A \in V \mid A \rightarrow a_i \in R\}$$

(ii) **for** $h := 1$ **to** $n - 1$ **do**

for $i := 1$ **to** $n - h$ **do**

$$V_{i,i+h} = \bigcup_{j=i}^{i+h-1} V_{i,j} * V_{j+1,i+h}$$

(iii) **if** $S \in V_{1,n}$ **then return** Ausgabe $w \in L(G)$

else return Ausgabe $w \notin L(G)$

CYK-Algorithmus (Cocke-Younger-Kasami)

Algorithmus

Input sei eine **Grammatik G in CNF** und ein **Wort $w = a_1 \dots a_n \in \Sigma^*$** .

(i) **for $i := 1$ to n do** /*Regeln $A \rightarrow a$ eintragen */

$$V_{i,i} := \{A \in V \mid A \rightarrow a_i \in R\}$$

(ii) **for $h := 1$ to $n - 1$ do**

for $i := 1$ to $n - h$ do

$$V_{i,i+h} = \bigcup_{j=i}^{i+h-1} V_{i,j} * V_{j+1,i+h}$$

(iii) **if $S \in V_{1,n}$ then return** Ausgabe $w \in L(G)$

else return Ausgabe $w \notin L(G)$

CYK-Algorithmus (Cocke-Younger-Kasami)

Eigenschaften

Für Wörter der Länge $|w| = n$ entscheidet der CYK-Algorithmus in der Größenordnung von n^3 Schritten, ob $w \in L(G)$ ist.

CYK-Algorithmus (Cocke-Younger-Kasami)

Beispiel 8.1 (CYK)

Eine Grammatik in CNF, die dieselbe Sprache wie oben erzeugt:

$$G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$$

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Die Sprache ist: $L(G) = \{vv^R \mid v \in \{a, b\}^+\}$

Auführlich an der Tafel.

CYK-Algorithmus (Cocke-Younger-Kasami)

Beispiel 8.1 (CYK)

Eine Grammatik in CNF, die dieselbe Sprache wie oben erzeugt:

$$G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$$

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Die Sprache ist: $L(G) = \{vv^R \mid v \in \{a, b\}^+\}$

Auführlich an der Tafel.

CYK-Algorithmus (Cocke-Younger-Kasami)

Beispiel 8.1 (CYK)

Eine Grammatik in CNF, die dieselbe Sprache wie oben erzeugt:

$$G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$$

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Die Sprache ist: $L(G) = \{vv^R \mid v \in \{a, b\}^+\}$

Auführlich an der Tafel.

CYK-Algorithmus (Cocke-Younger-Kasami)

Beispiel 8.1 (CYK)

Eine Grammatik in CNF, die dieselbe Sprache wie oben erzeugt:

$$G = (\{S, S_a, S_b, A, B\}, \{a, b\}, R, S)$$

$$R = \{ \begin{array}{l} S \rightarrow AS_a \mid BS_b \mid AA \mid BB \\ S_a \rightarrow SA \\ S_b \rightarrow SB \\ A \rightarrow a \\ B \rightarrow b \end{array} \}$$

Die Sprache ist: $L(G) = \{vv^R \mid v \in \{a, b\}^+\}$

Auführlich an der Tafel.