

Grundbegriffe der Theoretischen Informatik

Sommersemester 2017 - Beate Bollig

Die Folien basieren auf den Materialien von Thomas Schwentick.

Teil B: Kontextfreie Sprachen

8: Normalformen und Korrektheitsbeweise

Normalformen: Einleitung


Erinnerung an die Logik-Vorlesung

- Für aussagenlogische Formeln gibt es verschiedene Normalformen, die eine bestimmte syntaktische Struktur haben:
 - konjunktive Normalform
 - disjunktive Normalform
 - Negationsnormalform
- Für viele Zwecke nützlich:
 - Um den Resolutionskalkül anzuwenden, muss die Formel in KNF vorliegen
 - Um den Tableauekalkül anzuwenden, muss die Formel in NNF vorliegen

- Jetzt: Normalformen für kontextfreie Grammatiken

- **Chomsky-Normalform:** Nur Regeln der Art
 - $X \rightarrow YZ$ und
 - $X \rightarrow \sigma$

- **Greibach-Normalform:** Nur Regeln der Art
 - $X \rightarrow \sigma X_1 \cdots X_k, k \geq 0$

 In beiden Fällen wird bei Bedarf zusätzlich eine Regel $S \rightarrow \epsilon$ ergänzt

- Nutzen dieser Normalformen:
 - Die automatische Verarbeitung von Grammatiken wird erleichtert
 - weniger mögliche Fälle zu implementieren
 - Beweise werden übersichtlicher
 - weniger mögliche Fälle zu untersuchen

Inhalt

▷ 8.1 Chomsky-Normalform

8.2 Greibach-Normalform


8.3 Korrektheitsbeweise für kontextfreie Grammatiken

8.4 Anhang: Beweisdetails

Chomsky-Normalform: Definition

Definition

- Eine Variable X einer Grammatik $G = (V, \Sigma, S, P)$ heißt ⊕
 - **nützlich**, falls es eine Ableitung $S \Rightarrow_G^* \alpha X \beta \Rightarrow_G^* w$ mit $w \in \Sigma^*$ gibt

 Eine Variable ist also nützlich, wenn sie in mindestens einer Ableitung eines Wortes vorkommt

Definition

- $G = (V, \Sigma, S, P)$ ist in **Chomsky-Normalform (CNF)**, wenn
 - G nur nützliche Variablen enthält und
 - alle Regeln von G die Form
 - * $X \rightarrow YZ$ oder
 - * $X \rightarrow \sigma$haben mit $X, Y, Z \in V, \sigma \in \Sigma$
- Falls S in keiner rechten Regelseite vorkommt, ist zusätzlich die Regel $S \rightarrow \epsilon$ erlaubt
- Wir werden jetzt konstruktiv beweisen, dass es zu jeder kontextfreien Grammatik eine äquivalente Grammatik in CNF gibt

Chomsky-Normalform: Ausblick

- Grammatiken werden durch die Umwandlung in CNF meist größer, aber homogener, wie sich an der Beispiel-Grammatik zeigen wird

Beispiel-Grammatik: G_0

$$\begin{aligned} S &\rightarrow bDD \mid Ca \mid bc \\ A &\rightarrow B \mid aCC \mid baD \\ B &\rightarrow cBD \mid \epsilon \mid AC \\ C &\rightarrow bD \mid aBA \\ D &\rightarrow CD \mid a \mid EF \\ E &\rightarrow Eb \\ F &\rightarrow a \end{aligned}$$

Beispiel-Grammatik in CNF

$$\begin{aligned} S &\rightarrow W_b S_1 \mid C W_a \mid W_b W_c \\ S_1 &\rightarrow DD \\ A &\rightarrow W_a A_1 \mid W_b A_2 \mid W_c B_1 \mid AC \mid \\ &\quad W_b D \mid W_a C_1 \mid a \\ A_1 &\rightarrow CC \\ A_2 &\rightarrow W_a D \\ B &\rightarrow W_c B_1 \mid AC \mid W_b D \mid W_a C_1 \mid a \\ B_1 &\rightarrow BD \mid CD \mid a \\ C &\rightarrow W_b D \mid W_a C_1 \mid a \\ C_1 &\rightarrow BA \mid W_a A_1 \mid W_b A_2 \mid W_c B_1 \mid \\ &\quad AC \mid W_b D \mid W_a C_1 \mid a \\ D &\rightarrow CD \mid a \\ W_a &\rightarrow a \\ W_b &\rightarrow b \\ W_c &\rightarrow c \end{aligned}$$

Chomsky-Normalform: Satz

Satz 8.1 [Chomsky 59]

- Für jede kontextfreie Sprache L gibt es eine Grammatik G in Chomsky-Normalform mit $L(G) = L$

Beweisskizze

- Sei $G_0 = (V, \Sigma, S, P)$ eine kontextfreie Grammatik für L
- Wir entfernen in G_0 nach und nach die Merkmale, die der CNF im Weg stehen und konstruieren dafür Grammatiken mit folgenden Eigenschaften:

(G_1) nur nützliche Variablen

(G_2) rechte Seiten enthalten genau ein Terminalsymbol oder nur Variablen

(G_3) ohne Regeln $X \rightarrow \beta$ mit $|\beta| > 2$

(G_4) ohne ϵ -Regeln

(G_5) ohne Regeln der Art $X \rightarrow Y$

- Es gilt dann $L(G_5) = L - \{\epsilon\}$, deshalb muss G_5 evtl. um $S \rightarrow \epsilon$ ergänzt werden

CNF: (1) Entfernen von nutzlosen Variablen (1/4)

Definition

- Eine Variable X einer Grammatik $G = (V, \Sigma, S, P)$ heißt ⊕
 - **erreichbar**, falls es eine Ableitung $S \Rightarrow_G^* \alpha X \beta$ gibt
 - **erzeugend**, falls es eine Ableitung $X \Rightarrow_G^* w$ mit $w \in \Sigma^*$ gibt

- Klar: nützlich \Rightarrow erreichbar und erzeugend


Beispiel

- In der Grammatik
$$\begin{array}{l} S \rightarrow AB \mid a \\ A \rightarrow b \\ C \rightarrow ab \end{array}$$
ist
 - S nützlich
 - C erzeugend, aber nicht erreichbar
 - B erreichbar, aber nicht erzeugend
 - A erreichbar und erzeugend, aber nicht nützlich

- Wie die Variable A im Beispiel zeigt, ist eine erreichbare und erzeugende Variable nicht immer nützlich
- Es genügt, zuerst die nicht erzeugenden und danach die (dann) nicht erreichbaren Variablen zu entfernen

Algorithmus CNF1


- 1: $G'_1 := G_0$ ohne die nicht erzeugenden Variablen
- 2: $G_1 := G'_1$ ohne die nicht erreichbaren Variablen

 Dabei werden jeweils außer den Variablen auch alle Regeln entfernt, in denen sie (links oder rechts) vorkommen

CNF: (1) Entfernen von nutzlosen Variablen (2/4)

- Berechnung der Menge V_e der **erzeugenden Variablen**:

- Initialisiere V_e durch die Menge aller Variablen X , für die es eine Regel gibt, deren rechte Seite nur aus Terminalsymbolen besteht

 Also eine Regel $X \rightarrow \alpha$ mit $\alpha \in \Sigma^*$

- Solange möglich, füge Variablen X zu V_e hinzu, wenn sie eine Regel haben, die auf der rechten Seite nur Terminalsymbole und Variablen aus V_e enthält

- Variablen, die am Ende nicht in V_e sind, sind nicht erzeugend

Beispiel-Grammatik: G_0

$$\begin{aligned} S &\rightarrow bDD \mid Ca \mid bc \\ A &\rightarrow B \mid aCC \mid baD \\ B &\rightarrow cBD \mid \epsilon \mid AC \\ C &\rightarrow bD \mid aBA \\ D &\rightarrow CD \mid a \mid EF \\ E &\rightarrow Eb \\ F &\rightarrow a \end{aligned}$$

- E ist nicht erzeugend und wird daher nicht in G'_1 aufgenommen

G'_1

$$\begin{aligned} S &\rightarrow bDD \mid Ca \mid bc \\ A &\rightarrow B \mid aCC \mid baD \\ B &\rightarrow cBD \mid \epsilon \mid AC \\ C &\rightarrow bD \mid aBA \\ D &\rightarrow CD \mid a \\ F &\rightarrow a \end{aligned}$$

CNF: (1) Entfernen von nutzlosen Variablen (3/4)

- Berechnung der **nicht erreichbaren Variablen**:
- Konstruiere einen Graphen $H(G'_1)$ zu G'_1 :
 - Knotenmenge: Variablen von G'_1
 - Kante von X nach Y , wenn Y auf der rechten Seite einer Regel zu X vorkommt
- Berechne alle in diesem Graphen von S aus erreichbaren Knoten
- Die übrigen Knoten sind nicht erreichbar

G'_1

$$\begin{aligned} S &\rightarrow bDD \mid Ca \mid bc \\ A &\rightarrow B \mid aCC \mid baD \\ B &\rightarrow cBD \mid \epsilon \mid AC \\ C &\rightarrow bD \mid aBA \\ D &\rightarrow CD \mid a \\ F &\rightarrow a \end{aligned}$$

- F ist in G'_1 nicht erreichbar

G_1

$$\begin{aligned} S &\rightarrow bDD \mid Ca \mid bc \\ A &\rightarrow B \mid aCC \mid baD \\ B &\rightarrow cBD \mid \epsilon \mid AC \\ C &\rightarrow bD \mid aBA \\ D &\rightarrow CD \mid a \end{aligned}$$

CNF: (1) Entfernen von nutzlosen Variablen (4/4)

Lemma 8.2

- Sei G_1 durch Algorithmus CNF1 berechnet
- Dann gelten:
 - (a) $L(G_1) = L(G_0)$
 - (b) G_1 enthält nur nützliche Variablen

Beweisidee

- (a) – $L(G_1) \subseteq L(G_0)$:
 - * weil alle Regeln aus G_1 auch in G_0 vorkommen
- $L(G_0) \subseteq L(G_1)$:
 - * weil alle in einer Ableitung $S \Rightarrow_{G_0}^* w$ vorkommenden Variablen und Regeln bei der Umwandlung in G_1 erhalten bleiben
- (b) – Sei X eine Variable von G_1
 - ➡ X ist erreichbar in G'_1
 - ➡ $S \Rightarrow_{G'_1}^* \alpha X \beta$ für gewisse α, β
 - Wegen Schritt (1) des Algorithmus sind X und alle Symbole aus α, β erzeugend in G'_1
(und damit auch in G_1)
 - ➡ X ist nützlich

CNF: (2) Variablen und Terminalsymbole trennen

Algorithmus CNF2

- 1: **for** jedes Terminalsymbol $\sigma \in \Sigma$ **do**
- 2: Füge eine neue Variable W_σ hinzu
- 3: Ersetze in allen rechten Regelseiten σ durch W_σ
- 4: Füge eine neue Regel $W_\sigma \rightarrow \sigma$ hinzu

Lemma 8.3

- Sei G_2 durch Algorithmus CNF2 aus G_1 berechnet
- Dann gelten:
 - (a) $L(G_2) = L(G_1)$
 - (b) Jede rechte Regelseite von G_2 ist von einem der folgenden drei Typen:
 - ein Terminalsymbol
 - ϵ
 - ein oder mehrere Variablen

CNF: (2) Variablen und Terminalsymbole trennen: Beispiel

G_1

$$\begin{aligned} S &\rightarrow bDD \mid Ca \mid bc \\ A &\rightarrow B \mid aCC \mid baD \\ B &\rightarrow cBD \mid \epsilon \mid AC \\ C &\rightarrow bD \mid aBA \\ D &\rightarrow CD \mid a \end{aligned}$$

G_2

$$\begin{aligned} S &\rightarrow W_bDD \mid CW_a \mid W_bW_c \\ A &\rightarrow B \mid W_aCC \mid W_bW_aD \\ B &\rightarrow W_cBD \mid \epsilon \mid AC \\ C &\rightarrow W_bD \mid W_aBA \\ D &\rightarrow CD \mid W_a \\ W_a &\rightarrow a \\ W_b &\rightarrow b \\ W_c &\rightarrow c \end{aligned}$$

CNF: (3) Rechte Seiten verkürzen

Algorithmus CNF3

1: Entferne jede Regel der Art

$$X \rightarrow Y_1 \cdots Y_k, \text{ mit } k > 2$$

2: Füge dafür folgende Regeln hinzu:

$$X \rightarrow Y_1 Z_1$$

$$Z_1 \rightarrow Y_2 Z_2$$

$$\vdots \rightarrow \vdots$$

$$Z_{k-2} \rightarrow Y_{k-1} Y_k$$

- Dabei sind die Z_i neue (und für jede ersetzte Regel andere) Variablen, für alle $i \in \{1, \dots, k-2\}$

Lemma 8.4

- Sei G_3 durch Algorithmus CNF3 aus G_2 berechnet
- Dann gelten:
 - (a) $L(G_3) = L(G_2)$
 - (b) Jede rechte Regelseite von G_3 ist von einem der folgenden Typen:
 - ein Terminalsymbol
 - ϵ
 - eine Variable
 - zwei Variablen

CNF: (3) Rechte Seiten verkürzen: Beispiel

G_2

$$S \rightarrow W_b D D \mid C W_a \mid W_b W_c$$

$$A \rightarrow B \mid W_a C C \mid W_b W_a D$$

$$B \rightarrow W_c B D \mid \epsilon \mid A C$$

$$C \rightarrow W_b D \mid W_a B A$$

$$D \rightarrow C D \mid W_a$$

$$W_a \rightarrow a$$

$$W_b \rightarrow b$$

$$W_c \rightarrow c$$

G_3

$$S \rightarrow W_b S_1 \mid C W_a \mid W_b W_c$$

$$S_1 \rightarrow D D$$

$$A \rightarrow B \mid W_a A_1 \mid W_b A_2$$

$$A_1 \rightarrow C C$$

$$A_2 \rightarrow W_a D$$

$$B \rightarrow W_c B_1 \mid \epsilon \mid A C$$

$$B_1 \rightarrow B D$$

$$C \rightarrow W_b D \mid W_a C_1$$

$$C_1 \rightarrow B A$$

$$D \rightarrow C D \mid W_a$$

$$W_a \rightarrow a$$

$$W_b \rightarrow b$$

$$W_c \rightarrow c$$

CNF: (4) ϵ -Regeln entfernen (1/3)

Beispiel

- Was ist zu beachten, wenn eine Grammatik (unter anderem) die Regeln

$$A \rightarrow BC$$

$$B \rightarrow EF$$

$$C \rightarrow DE$$

$$D \rightarrow E$$

$$E \rightarrow \epsilon$$

enthält und wir die ϵ -Regel $E \rightarrow \epsilon$ entfernen wollen?

- Um weiterhin den Effekt der (Teil-)Ableitung $B \Rightarrow EF \Rightarrow F$ zu ermöglichen, sollte die Regel $B \rightarrow F$ hinzugefügt werden
- Das Entfernen von $E \rightarrow \epsilon$ bewirkt aber auch, dass ϵ nicht mehr von D und C abgeleitet werden kann
- Deshalb müssen auch für Vorkommen dieser Variablen auf rechten Regelseiten Ergänzungen vorgenommen werden:
 \rightarrow z.B. neue Regel: $A \rightarrow B$

- Der folgende Algorithmus CNF4 berechnet deshalb zunächst die Menge V' aller Variablen, von denen der Leerstring abgeleitet werden kann
- Für jedes Vorkommen dieser Variablen in rechten Regelseiten fügt er dann neue Regeln ein
- Alle ϵ -Regeln werden dann entfernt
- Es ist klar, dass der Leerstring danach nicht mehr erzeugt werden kann

CNF: (4) ϵ -Regeln entfernen (2/3)

Algorithmus CNF4

- 1: $V' := \{X \mid X \Rightarrow^* \epsilon\}$
- 2: **for** $X \rightarrow YZ$ in P **do**
- 3: **if** $Z \in V'$ **then**
- 4: Füge Regel $X \rightarrow Y$ hinzu
- 5: **if** $Y \in V'$ **then**
- 6: Füge Regel $X \rightarrow Z$ hinzu
- 7: Entferne alle Regeln der Art $X \rightarrow \epsilon$
- 8: Entferne nutzlos gewordene Variablen mit CNF1

Lemma 8.5

- Sei G_4 durch Algorithmus CNF4 aus G_3 berechnet
- Dann gelten:
 - (a) $L(G_4) = L(G_3) - \{\epsilon\}$
 - (b) Jede rechte Regelseite von G_4 ist von einem der folgenden 3 Typen:
 - ein Terminalsymbol
 - eine Variable
 - zwei Variablen

CNF: (4) ϵ -Regeln entfernen: Beispiel

G_3

$$\begin{aligned} S &\rightarrow W_b S_1 \mid C W_a \mid W_b W_c \\ S_1 &\rightarrow D D \\ A &\rightarrow B \mid W_a A_1 \mid W_b A_2 \\ A_1 &\rightarrow C C \\ A_2 &\rightarrow W_a D \\ B &\rightarrow W_c B_1 \mid \epsilon \mid A C \\ B_1 &\rightarrow B D \\ C &\rightarrow W_b D \mid W_a C_1 \\ C_1 &\rightarrow B A \\ D &\rightarrow C D \mid W_a \\ W_a &\rightarrow a \\ W_b &\rightarrow b \\ W_c &\rightarrow c \end{aligned}$$

- $V' = \{A, B, C_1\}$

G_4

$$\begin{aligned} S &\rightarrow W_b S_1 \mid C W_a \mid W_b W_c \\ S_1 &\rightarrow D D \\ A &\rightarrow B \mid W_a A_1 \mid W_b A_2 \\ A_1 &\rightarrow C C \\ A_2 &\rightarrow W_a D \\ B &\rightarrow W_c B_1 \mid A C \mid C \\ B_1 &\rightarrow B D \mid D \\ C &\rightarrow W_b D \mid W_a C_1 \mid W_a \\ C_1 &\rightarrow B A \mid A \mid B \\ D &\rightarrow C D \mid W_a \\ W_a &\rightarrow a \\ W_b &\rightarrow b \\ W_c &\rightarrow c \end{aligned}$$

CNF: (4) ϵ -Regeln entfernen (3/3)

Algorithmus CNF4

- 1: $V' := \{X \mid X \Rightarrow^* \epsilon\}$
- 2: **for** $X \rightarrow YZ$ in P **do**
- 3: **if** $Z \in V'$ **then**
- 4: Füge Regel $X \rightarrow Y$ hinzu
- 5: **if** $Y \in V'$ **then**
- 6: Füge Regel $X \rightarrow Z$ hinzu
- 7: Entferne alle Regeln der Art $X \rightarrow \epsilon$
- 8: Entferne nutzlos gewordene Variablen (CNF1)

Beweisidee für Lemma 8.5

- Da der Algorithmus sowohl Regeln hinzufügt als auch Regeln entfernt, ist der Nachweis, dass die neue Grammatik äquivalent ist (bis auf ϵ), etwas kompliziert
- Es wird bewiesen, dass für jede Variable X von G_3 und jeden String $w \in \Sigma^* - \{\epsilon\}$ äquivalent sind:
 - (1) $X \Rightarrow_{G_4}^* w$
 - (2) $X \Rightarrow_{G_3}^* w$
- Für „(1) \Rightarrow (2)“ lässt sich für jede neue Regel $X \rightarrow Y$ in G_4 zeigen:
$$X \Rightarrow_{G_3}^* Y$$
- Für „(2) \Rightarrow (1)“ lässt sich durch Induktion nach n zeigen:
$$X \Rightarrow_{G_3}^n w \Rightarrow X \Rightarrow_{G_4}^* w$$
- Weitere Details im Anhang

CNF: (5) Einzel-Variablen der rechten Seite entfernen (1/2)

- Jetzt müssen nur noch die **Einheitsregeln** entfernt werden, also Regeln der Form $X \rightarrow Y$

Beispiel

- Was ist beim Entfernen der Einheitsregeln aus einer Grammatik, die folgende Regeln enthält, zu beachten?

$$A \rightarrow BC$$

$$B \rightarrow D$$

$$C \rightarrow DF$$

$$D \rightarrow E$$

$$E \rightarrow e$$

- Um den Effekt der Ableitung $C \Rightarrow DF \Rightarrow EF \Rightarrow eF$ zu erhalten, nehmen wir die Regel $D \rightarrow e$ auf:

$$C \Rightarrow DF \Rightarrow eF$$

- Um den Effekt der Ableitung

$$A \Rightarrow BC \Rightarrow DC \Rightarrow EC \Rightarrow eC$$

zu erhalten, nehmen wir auch noch $B \rightarrow e$ auf:

$$A \Rightarrow BC \Rightarrow eC$$

- Der folgende Algorithmus CNF5 berechnet zunächst alle Variablenpaare $X \neq Y$, für die $X \Rightarrow^* Y$ gilt

- Dann fügt er für jedes solche Paar zu jeder binären Regel $Y \rightarrow \alpha$ eine neue Regel $X \rightarrow \alpha$ hinzu

CNF: (5) Einzel-Variablen der rechten Seite entfernen (2/2)


Algorithmus CNF5

- 1: $U := \{(X, Y) \mid X \Rightarrow_{G_4}^* Y, X \neq Y\}$
- 2: **for** (X, Y) in U und jede Nicht-Einheitsregel $Y \rightarrow \alpha$ **do**
- 3: Füge neue Regel $X \rightarrow \alpha$ ein
- 4: Entferne alle Einheitsregeln
- 5: Entferne alle nicht erreichbaren Variablen (und ihre Regeln)

Lemma 8.6

- Sei G_5 durch Algorithmus CNF5 aus G_4 berechnet
- Dann gelten:
 - (a) $L(G_5) = L(G_4)$
 - (b) G_5 ist in Chomsky-Normalform

Beweisidee

- „ $L(G_5) \subseteq L(G_4)$ “:
 - Für jede neue Regel $X \rightarrow \alpha$ von G_5 gibt es in G_4 eine Variable Y , so dass gilt:
 - * $X \Rightarrow_{G_4}^* Y$
 - * $Y \rightarrow \alpha$ ist Regel in G_4
 - Also gibt es für jede neue Regel $X \rightarrow \alpha$ von G_5 in G_4 eine Ableitung $X \Rightarrow_{G_4}^* \alpha$
- „ $L(G_4) \subseteq L(G_5)$ “:
 - Jede Folge $X \Rightarrow_{G_4} \dots \Rightarrow_{G_4} Y \Rightarrow_{G_4} \alpha$ kann in G_5 durch Anwendung von $X \rightarrow \alpha$ ersetzt werden
 -  Dabei ist α keine Variable!
- Weitere Details im Anhang

CNF: (5) Einzel-Variablen entfernen (Beispiel)

G_4

$$\begin{aligned}
 S &\rightarrow W_b S_1 \mid C W_a \mid W_b W_c \\
 S_1 &\rightarrow D D \\
 A &\rightarrow B \mid W_a A_1 \mid W_b A_2 \\
 A_1 &\rightarrow C C \\
 A_2 &\rightarrow W_a D \\
 B &\rightarrow W_c B_1 \mid A C \mid C \\
 B_1 &\rightarrow B D \mid D \\
 C &\rightarrow W_b D \mid W_a C_1 \mid W_a \\
 C_1 &\rightarrow B A \mid A \mid B \\
 D &\rightarrow C D \mid W_a \\
 W_a &\rightarrow a \\
 W_b &\rightarrow b \\
 W_c &\rightarrow c
 \end{aligned}$$

$$U = \{(C_1, A), (A, B), (B, C), \\
 (C, W_a), (B_1, D), (D, W_a), \\
 (C_1, B), (C_1, C), (C_1, W_a), \\
 (A, C), (A, W_a), (B, W_a), (B_1, W_a)\}$$

G_5

$$\begin{aligned}
 S &\rightarrow W_b S_1 \mid C W_a \mid W_b W_c \\
 S_1 &\rightarrow D D \\
 A &\rightarrow W_a A_1 \mid W_b A_2 \mid W_c B_1 \mid A C \mid \\
 &\quad W_b D \mid W_a C_1 \mid a \\
 A_1 &\rightarrow C C \\
 A_2 &\rightarrow W_a D \\
 B &\rightarrow W_c B_1 \mid A C \mid W_b D \mid W_a C_1 \mid a \\
 B_1 &\rightarrow B D \mid C D \mid a \\
 C &\rightarrow W_b D \mid W_a C_1 \mid a \\
 C_1 &\rightarrow B A \mid W_a A_1 \mid W_b A_2 \mid W_c B_1 \mid \\
 &\quad A C \mid W_b D \mid W_a C_1 \mid a \\
 D &\rightarrow C D \mid a \\
 W_a &\rightarrow a \\
 W_b &\rightarrow b \\
 W_c &\rightarrow c
 \end{aligned}$$

Chomsky-Normalform: Abschluss der Beweisskizze

- Falls die Sprache L den String ϵ enthält, ergänzen wir noch einen weiteren Schritt

Algorithmus CNF6

- 1: **if** $\epsilon \in L$ und S kommt in keiner rechten Seite einer Regel vor **then**
- 2: Füge neue Regel $S \rightarrow \epsilon$ ein
- 3: **else**
- 4: Füge ein neues Startsymbol S' und die Regel $S' \rightarrow \epsilon$ hinzu
- 5: Füge für jede Regel $S \rightarrow \alpha$ die Regel $S' \rightarrow \alpha$ hinzu

Satz 8.1 [Chomsky 59]

- Für jede kontextfreie Sprache L gibt es eine Grammatik G in Chomsky-Normalform mit $L(G) = L$

Beweisskizze

- Sei $G_0 = (V, \Sigma, S, P)$ eine Grammatik für L
- Für die wie beschrieben konstruierten Grammatiken G_1, \dots, G_5 gilt:

$$\begin{aligned} L(G_5) &= L(G_4) && \text{Lemma 8.6} \\ &= L(G_3) - \{\epsilon\} && \text{Lemma 8.5} \\ &= L(G_2) - \{\epsilon\} && \text{Lemma 8.4} \\ &= L(G_1) - \{\epsilon\} && \text{Lemma 8.3} \\ &= L(G_0) - \{\epsilon\} && \text{Lemma 8.2} \end{aligned}$$

- Falls $\epsilon \notin L(G_0)$, gilt also $L(G_5) = L(G_0)$
- Falls $\epsilon \in L(G_0)$, gilt für die in CNF6 konstruierte Grammatik: $L(G_6) = L(G_0)$

CNF: Größe

- Wie groß wird die durch den Algorithmus insgesamt konstruierte CNF-Grammatik im Vergleich zur Ausgangsgrammatik?
- Sei $m = |\Sigma|$ und für jedes i jeweils
 - n_i die Anzahl der Variablen von G_i
 - k_i die Anzahl der Produktionen von G_i
 - l_i die maximale Länge einer rechten Seite von G_i
- Dann gilt für die einzelnen Teilschritte:
 1. Alle Parameter können allenfalls kleiner werden
 2. $n_2 = n_1 + m, k_2 = k_1 + m$
 3. $n_3 = \mathcal{O}(k_2 l_2), k_3 = \mathcal{O}(k_2 l_2), l_3 = 2$
 4. $k_4 \leq 3k_3$
 5. $k_5 = \mathcal{O}(k_4 n_4)$
- Insgesamt also:
 - $k_5 = \mathcal{O}(l_0^2 (m + k_0)^2) = \mathcal{O}(|G_0|^4)$
 - $n_5 = \mathcal{O}(l_0 (m + k_0))$

Inhalt

8.1 Chomsky-Normalform

▷ **8.2 Greibach-Normalform**

8.3 Korrektheitsbeweise für kontextfreie Grammatiken

8.4 Anhang: Beweisdetails

Greibach-Normalform (1/2)

Definition

- Eine kontextfreie Grammatik $G = (V, \Sigma, S, P)$ ist in **Greibach-Normalform (GNF)**, wenn
 - V nur nützliche Variablen enthält und
 - alle Regeln von G von der Form
 - * $X \rightarrow \sigma Y_1 \cdots Y_k$
mit $X, Y_1, \dots, Y_k \in V, \sigma \in \Sigma$ sind
- Falls S in keiner rechten Regelseite vorkommt, ist zusätzlich die Regel $S \rightarrow \epsilon$ erlaubt

Satz 8.7 [Greibach 65]

- Ist L kontextfrei, so gibt es eine Grammatik G in Greibach-Normalform, so dass $L(G) = L$

Greibach-Normalform (2/2)

- Es ist in der Greibach-Normalform möglich, die Anzahl der Variablen in rechten Regel-seiten auf maximal zwei zu beschränken
- Es gibt dann nur Regeln der Art:
 - $X \rightarrow \sigma$
 - $X \rightarrow \sigma Y$
 - $X \rightarrow \sigma Y Z$

Beispiel

- Die Grammatik $G'_{a=b}$:
$$\begin{aligned} S &\rightarrow aB \mid bA \mid \epsilon \\ S' &\rightarrow aB \mid bA \\ A &\rightarrow aS' \mid bAA \mid a \\ B &\rightarrow bS' \mid aBB \mid b \end{aligned}$$
ist in Greibach-Normalform

Bemerkungen

- Ist G in GNF, so haben Strings der Länge n Ableitungen der Länge n
- Der Beweis von Satz 8.7 ist aufwändiger als der Beweis von Satz 8.1
→ Buch von Ingo Wegener
(siehe auch: [Blum, Koch 99])

Inhalt

8.1 Chomsky-Normalform

8.2 Greibach-Normalform

▷ **8.3 Korrektheitsbeweise für kontextfreie Grammatiken**

8.4 Anhang: Beweisdetails

Korrektheitsbeweise für Grammatiken: allgemein

- Sei G eine Grammatik und L eine Sprache
- Wie lässt sich beweisen, dass $L(G) = L$ gilt?
- Wie üblich: Zeige $L(G) \subseteq L$ und $L \subseteq L(G)$
- $L(G) \subseteq L$ bedeutet, dass die Grammatik *nur* Strings aus L erzeugt ☞ Korrektheit
 - Der Korrektheitsbeweis ist meist durch Induktion nach der Ableitungslänge möglich
- $L \subseteq L(G)$ bedeutet, dass die Grammatik *alle* Strings aus L erzeugt ☞ Vollständigkeit
 - Meistens durch Induktion nach der Wortlänge
 - Der Induktionsschritt verwendet meist eine geeignete Fallunterscheidung
 - Hier ist oft eine weitere, nicht-induktive Definition von L nötig
- Hat eine Grammatik mehrere Variablen, so wird meist für jede Variable X gezeigt, was die Sprache $L(X)$ der von X aus ableitbaren Strings ist

✎ Faustregel:

- Zum Beweis der Korrektheit wird im Ableitungsbaum „von oben nach unten“ argumentiert
 - Zum Beweis der Vollständigkeit von unten nach oben
-
- Wir betrachten im Folgenden drei Korrektheitsbeweise:
 - für eine sehr einfache Grammatik: Palindrome
 - für eine etwas trickreichere Grammatik: $G_{a=b}$
 - für eine ziemlich trickreiche Grammatik: G_{diff}

Korrektheitsbeweise: Palindrome (1/2)

- Zur Erinnerung:

$$L_{\text{pali}} = \{w \in \{a, b\}^* \mid w^R = w\}$$

- Mit G_{pali} bezeichnen wir die Grammatik

$$P \rightarrow \epsilon \mid a \mid b \mid aPa \mid bPb$$

- Nicht-induktive Definition von Palindromen:

- Ein String w ist genau dann ein Palindrom, wenn für alle $i \in \{1, \dots, |w|\}$ gilt:

$$w[i] = w[|w| - i + 1]$$

Proposition 8.8

- $L(G_{\text{pali}}) = L_{\text{pali}}$

Beweisskizze

- Korrektheit „ $L(G_{\text{pali}}) \subseteq L_{\text{pali}}$ “:

- Sei $w \in L(G_{\text{pali}})$

$$\Rightarrow P \Rightarrow^k w \text{ für ein } k \geq 1$$

- Wir zeigen $w \in L$ durch Induktion nach der Ableitungslänge k

- $k = 1$:

- * ϵ, a, b sind Palindrome ✓

- $k > 1$:

- * Fallunterscheidung nach dem ersten Ableitungsschritt

- * 1. Fall: $P \Rightarrow aPa \Rightarrow^{k-1} ava = w$,
für ein $v \in \{a, b\}^*$

$$\Rightarrow P \Rightarrow^{k-1} v$$

$$\Rightarrow v^R = v$$

$$\Rightarrow w^R = av^Ra = ava = w$$

☞ Induktion

- * 2. Fall: $P \Rightarrow bPb \Rightarrow^{k-1} bvb = w$,
für ein $v \in \{a, b\}^*$

→ analog

Korrektheitsbeweise: Palindrome (2/2)

Beweisskizze (Forts.)

- **Vollständigkeit** „ $L_{\text{pali}} \subseteq L(G_{\text{pali}})$ “:

- Sei $w \in L_{\text{pali}}$ und $n \stackrel{\text{def}}{=} |w|$
- Wir zeigen $w \in L(G_{\text{pali}})$ durch Induktion nach n

- $n \in \{0, 1\}$:

➡ $w \in \{\epsilon, a, b\}$ ist ableitbar ✓

- $n \geq 2$:

* 1. Fall: $w[1] = a$

➡ $w[n] = a$

➡ $w = ava$ für einen String v der Länge $n - 2$

• Da $w^R = w$ ist, gilt auch $v^R = v$, denn für alle $i \leq n - 2$:

$$v[i] = w[i + 1]$$

☞ Def von v

$$= w[n - (i + 1) + 1]$$

☞ w Palindrom

$$= v[n - (i + 1)]$$

☞ Def von v

$$= v[(n - 2) - i + 1]$$

• Induktion: $P \Rightarrow^* v$

➡ $P \Rightarrow aPa \Rightarrow^* ava = w$

* 2. Fall: $w[1] = b$ analog

Korrektheitsbeweise: $L_{a=b}$ (1/2)

- Zur Erinnerung:

$$- L_{a=b} = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$$

- $G_{a=b}$ ist
 $S \rightarrow aSbS \mid bSaS \mid \epsilon$

Proposition 8.9

$$\bullet L(G_{a=b}) = L_{a=b}$$

Beweisskizze: Korrektheit

- Zu zeigen: $L(G_{a=b}) \subseteq L_{a=b}$
- Sei $w \in L(G_{a=b})$
- Induktion nach der Ableitungslänge k

$$\bullet k = 1: S \Rightarrow \epsilon \checkmark$$

- $k > 1$: Fallunterscheidung nach dem ersten Ableitungsschritt

$$- 1. \text{ Fall: } S \Rightarrow aSbS \Rightarrow^{k-1} aubv = w$$

$$\begin{aligned} * \text{ Induktion: } \#_a(u) &= \#_b(u) \text{ und} \\ \#_a(v) &= \#_b(v) \end{aligned}$$



$$\begin{aligned} \#_a(w) &= \#_a(u) + \#_a(v) + 1 \\ &= \#_b(u) + \#_b(v) + 1 \\ &= \#_b(w) \end{aligned}$$

$$- 2. \text{ Fall: } S \Rightarrow bSaS \Rightarrow^* buav = w \text{ analog}$$

Korrektheitsbeweise: $L_{a=b}$ (2/2)

- Sei für jeden String $w \in \{a, b\}^*$:

$$\underline{d(w)} \stackrel{\text{def}}{=} \#_b(w) - \#_a(w)$$
- Also: $L_{a=b} = \{w \in \{a, b\}^* \mid d(w) = 0\}$

Beweisskizze: Vollständigkeit


- Zu zeigen: $L_{a=b} \subseteq L(G_{a=b})$
- Sei $w \in L_{a=b}$
- Induktion nach $n \stackrel{\text{def}}{=} |w|$
- $n = 0$: $S \Rightarrow \epsilon \checkmark$

Beweisskizze (Forts.)

- $n > 0$:
 - 1. Fall: $w = au$ für ein $u \in \{a, b\}^*$ mit $|u| = n - 1$ ⊕
 - Da $d(w) = 0$ gilt $d(u) = 1$
 - Sei $i \geq 1$ die kleinste Zahl mit $d(u[1, i]) = 1$
 - ➡ $d(u[1, i - 1]) = 0$ und $u[i] = b$
 - Außerdem: $d(u[i + 1, n - 1]) = d(w) - (d(u[1, i]) - 1) = 0$
 - Induktion: $S \Rightarrow^* u[1, i - 1]$ und $S \Rightarrow^* u[i + 1, n - 1]$
 - ➡ $S \Rightarrow aSbS \Rightarrow^* au[1, i - 1]bu[i + 1, n - 1] = au = w$
 - 2. Fall: $w = bv$ analog

Korrektheitsbeweise: L_{diff} (1/3)

- Für einen String w gerader Länge bezeichne $1^{\text{st}}(w)$ seine erste Hälfte und $2^{\text{nd}}(w)$ seine zweite Hälfte


 Für Strings ungerader Länge seien beide Funktionen undefiniert

- Sei $L_{\text{diff}} \stackrel{\text{def}}{=} \{w \in \{a, b\}^* \mid 1^{\text{st}}(w) \neq 2^{\text{nd}}(w)\}$

- Idee für die Konstruktion einer Grammatik:

- Sorge dafür, dass a in $1^{\text{st}}(w)$ an derselben Stelle steht wie b in $2^{\text{nd}}(w)$

* Erlaube ansonsten beliebige Zeichen

 Oder umgekehrt mit b in $1^{\text{st}}(w)$ und a in $2^{\text{nd}}(w)$

- Setze dazu $w = w_1 w_2$ aus Teilstrings $w_1 = u_1 a v_1$ und $w_2 = u_2 b v_2$ zusammen mit $|u_1| = |v_1|$ und $|u_2| = |v_2|$

- Setze $i \stackrel{\text{def}}{=} |u_1|$ und $j \stackrel{\text{def}}{=} |u_2|$ \boxplus

$$\Rightarrow |w| = 2i + 1 + 2j + 1 = 2(i + j + 1)$$

- Das mittlere a von w_1 ist an Position $i + 1$ von $1^{\text{st}}(w)$

- Das mittlere b von w_2 ist in w an Position $2i + 1 + j + 1 = (i + j + 1) + (i + 1)$

\Rightarrow es ist in $2^{\text{nd}}(w)$ an Position $i + 1$

- Sei G_{diff} also die Grammatik

$$S \rightarrow AB \mid BA$$

$$A \rightarrow aAa \mid bAb \mid aAb \mid bAa \mid a$$

$$B \rightarrow aBa \mid bBb \mid aBb \mid bBa \mid b$$

Satz 8.10

- $L(G_{\text{diff}}) = L_{\text{diff}}$

Korrektheitsbeweise: L_{diff} (2/3)

- Zur Illustration betrachten wir eine Ableitung des Strings $w = \mathbf{abaaabba}$
- Aus A ist \mathbf{abaaa} ableitbar (mit \mathbf{a} in der Mitte)
- Aus B ist \mathbf{bba} ableitbar (mit \mathbf{b} in der Mitte)
- Also ist w ableitbar durch


$$S \Rightarrow AB \Rightarrow^* \mathbf{abaaa}B \Rightarrow^* \mathbf{abaaabba}$$

- Zu beachten: A erzeugt *nicht* die erste Hälfte von w

Korrektheitsbeweise: L_{diff} (3/3)

Beweisskizze

- Durch Induktion ist sehr leicht zu zeigen:
 - $L(A) = \{uav \mid |u| = |v|, u, v \in \{a, b\}^*\}$
 - $L(B) = \{ubv \mid |u| = |v|, u, v \in \{a, b\}^*\}$

 Hier bezeichnet $L(A)$ die Menge der von der Variablen A ableitbaren Terminalstrings

$$\begin{aligned} \Rightarrow L(G_{\text{diff}}) = L(S) = \\ \{u_1av_1u_2bv_2 \mid |u_1|=|v_1|, |u_2|=|v_2|, \\ u_1, u_2, v_1, v_2 \in \{a, b\}^*\} \cup \\ \{u_1bv_1u_2av_2 \mid |u_1|=|v_1|, |u_2|=|v_2|, \\ u_1, u_2, v_1, v_2 \in \{a, b\}^*\} \end{aligned}$$

- Damit lässt sich $L(G_{\text{diff}}) = L_{\text{diff}}$ recht einfach zeigen

 Anhang

Zusammenfassung

- Jede kontextfreie Sprache hat eine Grammatik in Chomsky-Normalform, also nur mit Regeln der Art $X \rightarrow YZ$ und $X \rightarrow a$
- Jede kontextfreie Sprache hat eine Grammatik in Greibach-Normalform, also nur mit Regeln der Art $X \rightarrow aY_1 \cdots Y_k$
- Außerdem kann jeweils noch eine Regel $S \rightarrow \epsilon$ hinzukommen
 - dann darf S aber auf keiner rechten Seite vorkommen

Literatur für dieses Kapitel

- **Chomsky-Normalform:**

- Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959

- **Greibach-Normalform:**

- Sheila A. Greibach. A new normal-form theorem for context-free phrase structure grammars. *J. ACM*, 12(1):42–52, 1965
- Norbert Blum and Robert Koch. Greibach normal form transformation revisited. *Inf. Comput.*, 150(1):112–118, 1999

Inhalt

8.1 Chomsky-Normalform

8.2 Greibach-Normalform

8.3 Korrektheitsbeweise für kontextfreie Grammatiken

▷ **8.4 Anhang: Beweisdetails**

CNF: (4) ϵ -Regeln entfernen

Beweisskizze (Forts.)

- „ $X \Rightarrow_{G_4}^* w \Rightarrow X \Rightarrow_{G_3}^* w$ “:
Wir zeigen:

$$X \rightarrow_{G_4} Y \Rightarrow X \Rightarrow_{G_3}^* Y$$

☞ für jede neue Regel $X \rightarrow Y$

 - Denn dann gibt es zu jeder Ableitung $S \Rightarrow_{G_4}^* w$ eine Ableitung $S \Rightarrow_{G_3}^* w$
- Sei also $X \rightarrow Y$ neu in G_4
 - 1. Fall:** In G_3 gibt es eine Regel $X \rightarrow YZ$ und $Z \Rightarrow_{G_3}^* \epsilon$
 - Dann gilt:

$$X \Rightarrow_{G_3} YZ \Rightarrow_{G_3}^* Y$$
 - 2. Fall:** In G_3 gibt es eine Regel $X \rightarrow ZY$ und $Z \Rightarrow_{G_3}^* \epsilon$
 - Dann gilt:

$$X \Rightarrow_{G_3} ZY \Rightarrow_{G_3}^* Y$$

Beweisskizze (Forts.)

- „ $X \Rightarrow_{G_3}^* w \Rightarrow X \Rightarrow_{G_4}^* w$ “:
- Wir zeigen durch Induktion nach n :
 - $X \Rightarrow_{G_3}^n w \Rightarrow X \Rightarrow_{G_4}^* w$
- Es gelte also $X \Rightarrow_{G_3}^n w$
- $n = 1$: $w = \sigma$ und $X \rightarrow \sigma$ ist Regel von G_3 , also auch von G_4
- $n > 1$:
 - **1. Fall:**

$$X \Rightarrow_{G_3} YZ \Rightarrow_{G_3}^{n-1} w_1 w_2 = w$$

mit $Y \Rightarrow_{G_3}^i w_1$, $Z \Rightarrow_{G_3}^j w_2$ und $i + j = n - 1$

 - * Falls $w_1 \neq \epsilon \neq w_2$ folgt die Behauptung per Induktion
 - * Falls $w_1 = \epsilon$, $w_2 \neq \epsilon$, so ist $Y \in V'$
 $\Rightarrow X \Rightarrow_{G_4} Z \Rightarrow_{G_4}^* w_2 = w$
 - * (Analog: $w_1 \neq \epsilon$, $w_2 = \epsilon$)
 - **2. Fall:** $X \Rightarrow_{G_3} Y \Rightarrow_{G_3}^* w$
 - * Induktion liefert: $X \Rightarrow_{G_4} Y \Rightarrow_{G_4}^* w$

CNF: (5) Einzel-Variablen der rechten Seite entfernen (3/3)

Beweisskizze für Lemma 8.6

(b) klar

(a) „ $L(G_5) \subseteq L(G_4)$ “:

- Sei $X \rightarrow \alpha$ eine gegenüber G_4 neue Regel von G_5
- ➔ In G_4 gibt es dann eine Regel $Y \rightarrow \alpha$, für ein Y , und es gilt $X \Rightarrow_{G_4}^* Y$
- ➔ $X \Rightarrow_{G_4}^* \alpha$
- ➔ Zu jeder Ableitung $S \Rightarrow_{G_5}^* w$ gibt es also eine Ableitung $S \Rightarrow_{G_4}^* w$

Beweisskizze (Forts.)


- „ $L(G_4) \subseteq L(G_5)$ “:
 - Sei $w \in L(G_4)$
 - Dann gibt es eine Linksableitung für w in G_4
 - Wird in einer Linksableitung eine Einheitsregel $X \rightarrow Y$ verwendet, so muss Y im nächsten Schritt wieder ersetzt werden
 - Nach endlich vielen Schritten muss dann zum ersten Mal eine Regel $Z \rightarrow \alpha$ verwendet werden, die keine Einheitsregel ist
 - ➔ $X \Rightarrow_{G_4}^* Z$
 - ➔ Nach Konstruktion enthält G_5 dann die Regel $X \rightarrow \alpha$
 - ➔ Zu jeder G_4 -Ableitung gibt es eine äquivalente G_5 -Ableitung

Details des Korrektheitsbeweises für L_{diff}

Beweisskizze: „ $L(G_{\text{diff}}) \subseteq L_{\text{diff}}$ “

- Sei $w \in L(G_{\text{diff}})$
- 1. Fall: $w = u_1 a v_1 u_2 b v_2$ für gewisse $u_1, u_2, v_1, v_2 \in \{a, b\}^*$ mit $|u_1| = |v_1|$ und $|u_2| = |v_2|$
 - Sei $i \stackrel{\text{def}}{=} |u_1|$ und $j \stackrel{\text{def}}{=} |u_2|$
 - ➔ $|w| = 2i + 2j + 2$ und
 - * $w[i + 1] = a$
 - * $w[2i + j + 2] = b$
 - Wie schon berechnet ist dann
 - * $1^{\text{st}}(w)[i + 1] = a$
 - * $2^{\text{nd}}(w)[i + 1] = b$
 - ➔ $w \in L_{\text{diff}}$
- 2. Fall: $w = u_1 b v_1 u_2 a v_2 \dots$
 - analog

Beweisskizze: „ $L_{\text{diff}} \subseteq L(G_{\text{diff}})$ “

- Sei $w \in L_{\text{diff}}$ und $n \stackrel{\text{def}}{=} |1^{\text{st}}(w)|$  $|w| = 2n$
- OBdA gibt es ein $i \in \{0, \dots, n - 1\}$, so dass

$$1^{\text{st}}(w)[i + 1] = a \text{ und } 2^{\text{nd}}(w)[i + 1] = b$$
- Wir setzen:
 - $u_1 \stackrel{\text{def}}{=} w[1, i]$
 - $v_1 \stackrel{\text{def}}{=} w[i + 2, 2i + 1]$
 - $u_2 \stackrel{\text{def}}{=} w[2i + 2, n + i]$
 - $v_2 \stackrel{\text{def}}{=} w[n + i + 2, 2n]$
- ➔ $w = u_1 a v_1 u_2 b v_2$ und
 - $|u_1| = i = |v_1|$ und

$$\begin{aligned} |u_2| &= n + i - (2i + 2) + 1 \\ &= n - i - 1 \\ &= 2n - (n + i + 2) + 1 \\ &= |v_2| \end{aligned}$$
- ➔ $w \in L(G_{\text{diff}})$