Übersicht

Definition 3.6.1

Eine Grammatik $G = (N, T, P, S)$ ist in Chomsky-Normalform, wenn sie nur Produktionen folgender Form enthält:

$$A \to a \text{ und } A \to BC,$$

wobei $a \in T$ und $A, B, C \in N$.

Beispiel für Grammatik in CNF:

$$S \to R_aA \mid R_bB$$
$$A \to SR_a \mid R_aR_a \mid R_bR_a$$
$$B \to SR_b \mid R_aR_b \mid R_bR_b$$
$$R_a \to a$$
$$R_b \to b$$

Transformation in CNF

1. ϵ-Produktionen entfernen
2. Neues Nonterminal R_a für jedes $a \in T$
3. Ersetze $A \rightarrow aBbC$ durch $A \rightarrow R_a B R_b C$
4. Neue Regel $R_a \rightarrow a$ für jedes $a \in T$
5. Ersetze $A \rightarrow B_1 B_2 B_3 \cdots B_k$ durch $A \rightarrow B_1 B_{23\ldots k}$ (neues Symbol)
   $B_{23\ldots k} \rightarrow B_2 B_{34\ldots k}$
   $B_{34\ldots k} \rightarrow B_3 B_{45\ldots k}$
   $\ldots$
   $B_{k-1,k} \rightarrow B_{k-1} B_k$

Jetzt fast in CNF. Aber es gibt noch „Kettenregeln“ $A \rightarrow B$.

Beispiel

$$S \longrightarrow aSa \mid bSb \mid a \mid b$$

$$S \longrightarrow R_a S R_a \mid R_b S R_b \mid R_a \mid R_b$$
$$R_a \longrightarrow a$$
$$R_b \longrightarrow b$$

$$S \longrightarrow R_a A$$
$$S \longrightarrow R_b B$$
$$A \longrightarrow S R_a$$
$$B \longrightarrow S R_b$$
$$R_a \longrightarrow a$$
$$S \longrightarrow R_a$$
$$R_b \longrightarrow b$$
$$S \longrightarrow R_b$$

Kettenregeln

So können wir Kettenregeln eliminieren:

Falls die Regel $A \to B$ existiert, dann füge für jede Produktion $C \to \alpha A \beta$ eine Produktion $C \to \alpha B \beta$ hinzu.

Füge zu $S \to D$ und $D \to \gamma \in P$ die Produktionen $S \to \gamma$ hinzu.

Dann streiche alle Kettenregeln.

Satz 3.6.2

Es gibt einen Algorithmus, der zu einer Grammatik G eine Grammatik G' in Chomsky-Normalform berechnet, wobei $L(G') = L(G) \setminus \{\epsilon\}$.

Beispiel

$S \longrightarrow R_a A$
$S \longrightarrow R_b B$
$A \longrightarrow S R_a$
$B \longrightarrow S R_b$
$R_a \longrightarrow a$
$S \longrightarrow R_a$
$R_b \longrightarrow b$
$S \longrightarrow R_b$

$S \longrightarrow R_a A \mid R_b B \mid a \mid b$
$A \longrightarrow S R_a \mid R_a R_a \mid R_b R_a$
$B \longrightarrow S R_b \mid R_a R_b \mid R_b R_b$
$R_a \longrightarrow a$
$R_b \longrightarrow b$

Das Wortproblem

- Gegeben: CFG G und Wort w der Länge n
- $w \in L(G)$ gdw. $S \in pre^*(\{w\})$
- Algorithmus:
  - Konstruiere NFA, der w erkennt.
  - Berechne daraus NFA für $pre^*(\{w\})$.
  - Überprüfe, ob es Transition von Start- zu Endzustand mit S gibt.
- Aufwand: $O(n^6)$
- Bei G in CNF lässt es sich zu $O(n^4)$ verbessern.
- Ziel: Effizienterer Algorithmus

Der CYK Algorithmus (Cocke, Younger, Kasami)

- Gegeben: CFG G in CNF und Wort $w = a_1\, a_2\, \dots\, a_n$
- $n = 1$: Ableitung nur mit Regel $A \rightarrow a$ möglich.
- $n > 1$: Ableitung nur mit Regel $A \rightarrow B\, C$ möglich:
  - B leitet die ersten k Zeichen von w her
  - C leitet die letzten $n + 1 - k$ Zeichen von w her
- *Dynamisches Programmieren*
  - Berechne Tabelle, die speichert, welche Teilwörter von w aus welchen Nonterminalen hergeleitet werden können.
  - Für kürzere Teilwörter steht Information jeweils bereits in Tabelle.
- Tabelle T ist Dreiecksmatrix:
  $T[i, j]$ = Menge der Nichtterminale, aus denen man $a_i\, a_{i+1} \dots a_{i+j-1}$ ableiten kann
- $w \in L(G)$ gdw. $S \in T[1, n]$

Der CYK Algorithmus (Cocke, Younger, Kasami)

Eingabe: $w = a_1\, a_2\, \ldots\, a_n$

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

Algorithmus

- Für alle $1 \leq i \leq n$, setze $T[i,1] = \{A \in N \mid A \to a_i \in P\}$.
- Für alle $2 \leq j \leq n$
  - Für alle $1 \leq i \leq n - j + 1$
    - Setze $T[i,j] = \varnothing$.
    - Für alle $1 \leq k \leq j-1$, setze $T[i,j] = T[i,j] \cup$ $\{A \in N \mid A \to B\,C \in P,\ B \in T[i,k],\ C \in T[i+k, j-k]\}$
- Falls $S \in T[1,n]$, dann gib "*true*" zurück, sonst "*false*".

Der CYK Algorithmus (Cocke, Younger, Kasami)

$S \to R_a A \mid R_b B \mid a \mid b$

$A \to S R_a \mid R_a R_a \mid R_b R_a$

$B \to S R_b \mid R_a R_b \mid R_b R_b$

$R_a \to a$

$R_b \to b$

$w =$	a	a	b	a	b	a	a
1	R_a, S	R_a, S	R_b, S	R_a, S	R_b, S	R_a, S	R_a, S
2	A	B	A	B	A	A	
3		S	S	S			
4		B	A				
5		S					
6		A					
7	S						

Platzbedarf: $O(n^2)$

Laufzeit: $O(n^3)$