Theoretische Informatik III (T3INF2002)

Formale Sprachen und Automaten | Einführung Compilerbau

Vorlesung im Wintersemester 2022/23

Formale Sprachen und Automaten

- Kontextfreie Sprachen
- Chomsky-Normalform
- Backus-Naur-Form

Kontextfreie Sprachen

Definitionen und Eigenschaften

- Kontextfreie Grammatiken sind eine Erweiterung der regulären Grammatiken
- In beiden sind die Produktionen so aufgebaut, dass auf der linken Seite nur ein einzelnes Nonterminal steht
- Im Gegensatz zu regulären Grammatiken, die die Form der rechten Seite einschränken, kann diese in kontextfreien Grammatiken aus einer beliebigen Sequenz von Terminal- und Nonterminalzeichen bestehen

-> Eine Grammatik ist genau dann kontextfrei, wenn jede Produktion die Form $l \rightarrow r$ mit $l \in V$ und $r \in (\Sigma \cup V)^*$ besitzt.

Grammatik zur Erzeugung der Sprache $\{a^n b^n \mid n \in N^+\}$

- Pumping-Lemma zeigt, dass die Sprache $\{a^n b^n \mid n \in N^+\}$ nicht regulär ist und von keiner regulären Grammatik erzeugt werden kann
- Kontextfreie Sprachen sind hingegen ausdrucksstark genug, um die Sprache zu beschreiben

$G := (\{S\}, \{a, b\}, P, S)$

Produktionsmenge P: $S \rightarrow aSb \mid ab$

Beispiel: Ableitung des Worts *aaaabbbb*

Grammatik zur Erzeugung der Sprache $\{a^i b^j a^k \mid i \in \mathbb{N}^+, j, k \in \mathbb{N}\}$

- Ebenfalls eine kontextfreie Sprache ist $\{a^i b^j a^k \mid i \in \mathbb{N}^+, j, k \in \mathbb{N}\}$, da die linken Seiten der Produktionen nur aus Variablen bestehen

  $G := (\{S,A,B\},\{a,b\},P,S)$

  Produktionsmenge P:

  $$S \to AB \mid ABA$$
  $$A \to aA \mid a$$
  $$B \to Bb \mid \varepsilon$$

Beispiel: Ableitung des Worts *aabbaa*

Chomsky-Normalform

Chomsky-Normalform

- Es gibt Situationen in denen es erforderlich ist, dass die kontextfreie Grammatik in einer speziellen Form - der Chomsky-Normalform – vorliegt
- **Beispiel:** für das Cocke-Younger-Kasami (CYK)-Parsing-Verfahren oder für den Beweis des Pumping-Lemmas für kontextfreie Sprachen
- Zu jeder kontextfreien Grammatik gibt es eine äquivalente kontextfreie Grammatik in Chomsky-Normalform

Chomsky-Normalform

- Eine Grammatik $G=(V, \Sigma, P, S)$ liegt in Chomsky-Normalform vor, wenn alle Produktionen die Form

  $S \to \varepsilon$,
  $A \to \sigma$ oder $A \to BC$

besitzen mit $A \in V$, $B, C \in V \setminus \{S\}$ und $\sigma \in \Sigma$.

- Als Ausnahme ist die Produktion S -> ε erlaubt, wobei dann das Startsymbol S nicht rekursiv sein darf

Chomsky-Normalform

Produktionen:

$S \rightarrow \varepsilon$,
$A \rightarrow \sigma$ oder $A \rightarrow BC$

- Auf der rechten Seite jeder Produktion stehen entweder genau zwei Variablen oder genau ein Terminalzeichen
- Obwohl die Form der Produktionen stark eingeschränkt ist, lassen sich mit Grammatiken in Chomsky-Normalform die kontextfreien Sprachen erzeugen

Chomsky-Normalform

Um eine kontextfreie Grammatik G mit $\varepsilon \notin L(G)$ in die Chomsky- Normalform zu überführen, sind drei Schritte auszuführen:

Schritt 1: Basis-Normalisierung

Schritt 2: Terminalzeichen überbrücken

Schritt 3: Produktionen aufspalten

Schritt 1: Basis-Normalisierung

Definition:

Sei G = (V, T, P, S) eine kontextfreie Grammatik. Man bezeichnet die Grammatik als basis-normalisiert, wenn sie *keine nutzlosen* Variablen, *kein rekursives* Startsymbol, *keine Epsilon-Produktionen* außer S -> ε und *keine Kettenproduktionen* enthält.

Schritt 1: Basis-Normalisierung

- Nutzlose Variablen entfernen
- Rekursives Startsymbol entfernen
- Epsilon-Produktionen entfernen
- Kettenproduktionen entfernen

Normalisierung von kontextfreien Grammatiken

- Bei kontextfreien Grammatiken bestehen die linken Seiten aller Produktionen jeweils aus einer einzigen Variablen.
  - Für rechten Seiten der Produktionen bestehen keine Einschränkungen.
- Bestimmte Form der rechten Seite manchmal wünschenswert -> Durch Umformung der Grammatik ist es möglich, die rechten Seiten der Produktionen in bestimmte Form zu bringen, ohne dass sich die von der Grammatik erzeugte Sprache ändert.
- Beispielsweise lassen sich bei Bedarf alle Kettenproduktionen (Produktionen der Form X -> Y) beseitigen.
- Prozess wird als Normalisierung bezeichnet:
  - Zunächst führt man die Basis-Normalisierung durch
  - Nach weiteren Normalisierungsschritten steht am Ende die Chomsky-Normalform für kontextfreie Grammatiken

Äquivalenz von Grammatiken

Definition: Zwei Grammatiken G und G' werden als äquivalent bezeichnet, wenn sie dieselbe Sprache erzeugen, d.h. wenn gilt $L(G) = L(G')$

Beispiel: Grammatik G_0 mit den Produktionen

S -> aSb | ε

und die Grammatik G_1 mit den Produktionen

S -> C

C -> D

E -> ab

D -> S | aSb | aF | ε

sind äquivalent, denn beide erzeugen die Sprache { $a^n b^n$ | n Element natürliche Zahlen0 }.

Nutzlose Variablen

Definition: Sei $G = (V, T, P, S)$ eine kontextfreie Grammatik und sei $A = V \cup T$. Eine Variable $X \in V$ heißt

- erreichbar, wenn S Ableitung uXv mit $u, v \in A^*$ gilt,
- produktiv, wenn X Ableitung w mit $w \in T^*$ gilt,
- nutzlos, wenn X nicht erreichbar oder nicht produktiv ist.

-> Zu jeder kontextfreien Grammatik $G = (V, T, P, S)$ gibt es eine äquivalente kontextfreie Grammatik $G' = (V', T, P', S)$ ohne nutzlose Variablen.

Nutzlose Variablen

Beweis: Sei w ein Wort mit w ∈ L(G), dann gibt es eine Ableitungsfolge S -> w. Jede Variable, die in den Ableitungsschritten auftritt, ist erreichbar und produktiv. In keiner Ableitungsfolge kann eine nutzlose Variable vorkommen.

G' geht aus G hervor, indem ***aus V alle nutzlosen Variablen entfernt*** werden und ***aus P alle Produktionen, in denen nutzlose Variablen vorkommen***.

Beispiel: In Grammatik G_1 ist die Variable E nicht erreichbar und die Variable F nicht produktiv, beide Variablen sind nutzlos. Diese Variablen und die zugehörigen Produktionen werden entfernt. Es verbleibt die Grammatik G_2:

S -> C

C -> D

D -> S | aSb | ε

Rekursives Startsymbol

Definition: Sei $G = (V, T, P, S)$ eine Grammatik und sei $A = V \cup T$. Eine Variable $X \in V$ heißt rekursiv, wenn

$$X \ \text{->}\ uXv \qquad \text{mit } u, v \in A^* \text{ gilt.}$$

Beispiel: In Grammatik G_3 des Beispiels ist das Startsymbol S rekursiv.

-> Zu jeder Grammatik $G = (V, T, P, S)$ gibt es eine äquivalente Grammatik $G' = (V', T, P', S')$, deren Startsymbol S' nicht rekursiv ist.

Rekursives Startsymbol

Beweis: Grammatik G' geht aus der Grammatik G hervor, indem ein neues Startsymbol S' zur Menge der Variablen V hinzugefügt wird und die neue Produktion S' -> S zur Menge der Produktionen P hinzugefügt wird.

Beispiel: Indem die Vorgehensweise auf die Grammatik G_2 angewendet wird, ergibt sich die Grammatik G_3, deren Startsymbol S' nicht rekursiv ist:

S' -> S

S -> C

C -> D

D -> S | aSb | ε

Epsilon-Produktionen

Definition: Sei $G = (V, T, P, S)$ eine kontextfreie Grammatik. Eine Produktion der Form $X \rightarrow \varepsilon$ mit $X \in V$ wird als Epsilon-Produktion bezeichnet.

-> Zu jeder kontextfreien Grammatik $G = (V, T, P, S)$ gibt es eine äquivalente kontextfreie Grammatik $G' = (V, T, P', S')$

- ohne Epsilon-Produktionen, falls $\varepsilon \notin L(G)$
- mit $S' \rightarrow \varepsilon$ als einziger Epsilon-Produktion, falls $\varepsilon \in L(G)$

Epsilon-Produktionen

Beweis: Die kontextfreie Grammatik G wird wie folgt in die Grammatik G' umgeformt:

1. führe die Produktion S' -> S ein, damit das Startsymbol nicht rekursiv ist
2. bestimme alle Variablen X, aus denen sich das leere Wort ε ableiten lässt:

$$V_\varepsilon = \{ X \in V \mid X \rightarrow \varepsilon \}$$

3. entferne alle Epsilon-Produktionen aus der Grammatik

4. wiederhole solange Y -> uXv mit $X \in V_\varepsilon$ und $|uv| \geq 1$ eine Produktion ist, aber Y -> uv noch keine Produktion ist:

führe die Produktion Y -> uv zusätzlich ein

5. wenn $S' \in V_\varepsilon$ dann

führe die Produktion S' -> ε ein

Epsilon-Produktionen

Beispiel: In Grammatik G_3 wurde *Schritt 1* des Verfahrens bereits durchgeführt.

In *Schritt 2* bestimmt man V_ε = { S', S, C, D }.

In *Schritt 3* entfernt man die Produktion D -> ε.

In *Schritt 4* findet man D -> aSb als einzige Produktion mit der dort angegebenen Eigenschaft und führt die Produktion D -> ab neu ein.

In *Schritt 5* führt man die Produktion S' -> ε ein, da S' $\in V_\varepsilon$.

Ergebnis ist die Grammatik G_4:

S' -> S | ε

S -> C

C -> D

D -> S | aSb | ab

Kettenproduktionen

Definition: Sei G = (V, T, P, S) eine kontextfreie Grammatik. Eine Produktion der Form X -> Y mit X, Y $\in$ V wird als Kettenproduktion bezeichnet.

-> Zu jeder kontextfreien Grammatik G = (V, T, P, S) gibt es eine äquivalente kontextfreie Grammatik G' = (V', T, P', S) ohne Kettenproduktionen.

Kettenproduktionen entfernen:

Jede Kettenproduktion X -> Y wird so umgeformt, dass für Y die aus Y direkt ableitbaren rechten Seiten eingesetzt werden, so lange, bis keine Kettenproduktionen mehr vorhanden sind.

Kettenproduktionen

Beispiel: In Grammatik G_4 ist S -> C, C -> D, D -> S ein Zyklus von Kettenproduktionen. Man entfernt die Produktionen des Zyklus und ersetzen alle weiteren Vorkommen von S und D durch C. Das Ergebnis ist die Grammatik G_5:

S' -> C | ε

C -> aCb | ab

Kettenproduktion S' -> C entfernen, indem man für die rechte Seite C die aus C direkt ableitbaren Wörter einsetzt. Es ergibt sich die Grammatik G_6:

S' -> aCb | ab | ε

C -> aCb | ab

→

Schritt 1: Basis-Normalisierung

Sprache: $\{a^n b^n \mid n \in N_0\}$

Das Ergebnis der Basis-Normalisierung für die Grammatik S → C, C → aSb | ε ist die Grammatik

S´ → aSb | ab | ε   -> S´ ist neues Startsymbol

S → aSb | ab

Schritt 2: Terminalzeichen überbrücken

- Im zweiten Schritt wird für jedes Terminalzeichen eine neue Variable eingeführt
- Alle Vorkommen von Terminalzeichen in den Produktionen werden durch die entsprechenden neuen Variablen ersetzt
- Abschließend werden neue Produktionen, in denen die neuen Variablen wieder in Terminalzeichen überführt werden, hinzugefügt

Schritt 2: Terminalzeichen überbrücken

- Beispiel: neue Variablen A und B werden eingeführt und lassen diese an die Stelle der Terminalzeichen a und b treten
- Zudem werden entsprechende neue Produktionen hinzugefügt, um wieder die Terminalzeichen a und b zu erzeugen:

$$S' \rightarrow ASB \mid AB \mid \varepsilon$$

$$S \rightarrow ASB \mid AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

Schritt 3: Produktionen aufspalten

- Schließlich werden die Produktionen S´ → ASB und S → ASB, die auf der rechten Seite mehr als zwei Variablen enthalten aufgespalten, indem neue Variablen, hier *D* und *E* eingeführt werden:

S´ → AD | AB | ε

D → SB

S → AE | AB

E → SB

A → a

B → b

← Grammatik in Chomsky-Normalform

Übung: Schrittweise Erzeugung der Chomsky-Normalform

$G := (\{S,A,B\},\{a,b\},P,S)$

$S \rightarrow AB|ABA$

$A \rightarrow aA|a$

$B \rightarrow Bb|\varepsilon$

Chomsky-Normalform

- Spezielle Struktur einer Grammatik in Chomsky-Normalform wirkt sich auf das Erscheinungsbild der entstehenden Syntaxbäume aus
- Da jedes Nonterminal entweder durch ein Terminalzeichen oder durch zwei weitere Nonterminale ersetzt wird, entsteht im Inneren die Struktur eines Binärbaums
- Ist ein Binärbaum B vollständig, d. h., besitzen alle Blätter die gleiche Tiefe h, so besitzt der Baum exakt 2^h Blätter
- Ist der Binärbaum nicht vollständig, so gilt die Beziehung $|B| < 2^h$.

Übung: Originalgrammatik Syntaxbaum

$G := (\{S,A,B\},\{a,b\},P,S)$

$S \rightarrow AB|ABA$
$A \rightarrow aA|a$
$B \rightarrow Bb|\varepsilon$

Ableitung des Worts
aabbaa

Übung: Chomsky-Normalform Syntaxbaum

Backus-Naur-Form

Backus-Naur-Form

- Kontextfreie Sprachen sind ausdrucksstark genug, um die Syntax der meisten Programmiersprachen zu beschreiben
- Bereits Anfang der Sechzigerjahre verwendete der amerikanische Computerpionier John Backus eine kontextfreie Grammatik, um die Syntax der Programmiersprache Algol60 formal zu spezifizieren
- Die von Backus eingeführte Notation wird als Backus-Naur-Form bezeichnet und hat sich zum De-facto-Standard für die Beschreibung von Programmiersprachen entwickelt

Backus-Naur-Form

- Auf den ersten Blick unterscheidet sich die Backus-Naur-Form von der Produktionensyntax vor allem in der Verwendung des ***Ableitungssymbols ::= anstelle von*** $\rightarrow$
- Backus führte Spezialkonstrukte ein, mit denen sich die Produktionen kontextfreier Grammatiken übersichtlich beschreiben lassen
- Hierzu gehört unter anderem die ***Strichnotation***, um Produktionen mit gleicher linker Seite zu einer einzigen Regel zusammenzufassen

Auszug aus der Algol60-Syntax

<program>	::= <block> | <compound statement>
<block>	::= <unlabelled block> | <label>: <block>
<unlabelled block>	::= <block head> ; <compound t a i l >
<<block head>	::= ’BEGIN ’ <declaration> | <block head> ; <declaration>
<compound statement>	::= <unlabelled compound> | <label>: <compound statement>

Backus-Naur-Form

- In der erweiterten Backus-Naur-Form können Wortfragmente zusätzlich in eckige und geschweifte Klammerpaare eingeschlossen werden
- So besagt der Ausdruck

$$A ::= r_1[r_2]r_3,$$

dass zwischen r_1 und r_3 optional das Wort r_2 eingefügt werden darf

Backus-Naur-Form

- Der Ausdruck

$$A ::= r_1\{r_2\}r_3$$

bedeutet, dass sich das optionale Wortfragment r_2 beliebig oft wiederholen kann

- Keines der Konstrukte führt zu einer Erweiterung der Ausdrucksstärke; beide lassen sich, auf eine gewöhnliche Produktionenmenge zurückführen
- Bei der Backus-Naur-Form handelt es sich um eine alternative Beschreibungsform für kontextfreie Grammatiken

Reduktion der Backus- Naur-Form auf normale Produktionen