Formale Grundlagen der Informatik 1
Kapitel 6

Eigenschaften kontextfreier Sprachen

Frank Heitmann
heitmann@informatik.uni-hamburg.de

22. April 2014

Informales Beispiel

$$\begin{array}{rcl} I & \longrightarrow & L \mid IL \mid ID \\ L & \longrightarrow & a \mid b \mid c \\ D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{array}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$I \Rightarrow ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD$$

Informales Beispiel

$$\begin{array}{rcl} I & \longrightarrow & L \mid IL \mid ID \\ L & \longrightarrow & a \mid b \mid c \\ D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{array}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$I \Rightarrow ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD$$

Informales Beispiel

$$\begin{array}{rcl} I & \longrightarrow & L \mid IL \mid ID \\ L & \longrightarrow & a \mid b \mid c \\ D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{array}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$I \Rightarrow ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD$$

Informales Beispiel

$$\begin{array}{lcl} I & \longrightarrow & L \mid IL \mid ID \\ L & \longrightarrow & a \mid b \mid c \\ D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{array}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$I \Rightarrow ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD$$

Informales Beispiel

$$
\begin{array}{rcl}
I & \longrightarrow & L \mid IL \mid ID \\
L & \longrightarrow & a \mid b \mid c \\
D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4
\end{array}
$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$I \Rightarrow ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD$$

Informales Beispiel

$$\begin{array}{rcl} I & \longrightarrow & L \mid IL \mid ID \\ L & \longrightarrow & a \mid b \mid c \\ D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{array}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$\begin{array}{rcl} I & \Rightarrow & ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD \\ & \Rightarrow & aLLD \Rightarrow abLD \Rightarrow abL1 \Rightarrow abc1 \end{array}$$

Informales Beispiel

$$\begin{array}{rcl} I & \longrightarrow & L \mid IL \mid ID \\ L & \longrightarrow & a \mid b \mid c \\ D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{array}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$\begin{array}{rcl} I & \Rightarrow & ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD \\ & \Rightarrow & aLLD \Rightarrow abLD \Rightarrow abL1 \Rightarrow abc1 \end{array}$$

Informales Beispiel

$$
\begin{array}{rcl}
I & \longrightarrow & L \mid IL \mid ID \\
L & \longrightarrow & a \mid b \mid c \\
D & \longrightarrow & 0 \mid 1 \mid 2 \mid 3 \mid 4
\end{array}
$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$
\begin{array}{rcl}
I & \Rightarrow & ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD \\
 & \Rightarrow & aLLD \Rightarrow abLD \Rightarrow abL1 \Rightarrow abc1
\end{array}
$$

Informales Beispiel

$$\begin{aligned} I &\longrightarrow L \mid IL \mid ID \\ L &\longrightarrow a \mid b \mid c \\ D &\longrightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \end{aligned}$$

Eine mögliche Ableitung eines Identifiers (Ziel: $abc1$):

$$\begin{aligned} I &\Rightarrow ID \Rightarrow ILD \Rightarrow ILLD \Rightarrow LLLD \\ &\Rightarrow aLLD \Rightarrow abLD \Rightarrow abL1 \Rightarrow abc1 \end{aligned}$$

Grammatiken. Formale Definition

Definition (Grammatik)

Eine **Grammatik** ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq (V^* \setminus V_T^*) \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Grammatiken. Formale Definition

Definition (Grammatik)

Eine **Grammatik** ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq (V^* \setminus V_T^*) \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Grammatiken. Formale Definition

Definition (Grammatik)

Eine **Grammatik** ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq (V^* \setminus V_T^*) \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Grammatiken. Formale Definition

Definition (Grammatik)

Eine **Grammatik** ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq (V^* \setminus V_T^*) \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Grammatiken. Formale Definition

Definition (Grammatik)

Eine **Grammatik** ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq (V^* \setminus V_T^*) \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Grammatiken. Anmerkungen

Bemerkung

- Eine Regel $(u, v) \in P$ wird meist als $u \to v$ notiert.
- Mehreren Reglen $u \to v$ und $u \to w$ werden als $u \to v \mid w$ abgekürzt.
- Auf der linken Seite einer Regel steht stets mindestens ein Nonterminal!

$$P \subseteq (V^* \setminus V_T^*) \times V^*$$

Grammatiken. Anmerkungen

Bemerkung

- Eine Regel $(u, v) \in P$ wird meist als $u \to v$ notiert.
- Mehreren Reglen $u \to v$ und $u \to w$ werden als $u \to v \mid w$ abgekürzt.
- Auf der linken Seite einer Regel steht stets mindestens ein Nonterminal!

$$P \subseteq (V^* \setminus V_T^*) \times V^*$$

Grammatiken. Anmerkungen

Bemerkung

- Eine Regel $(u, v) \in P$ wird meist als $u \to v$ notiert.
- Mehreren Reglen $u \to v$ und $u \to w$ werden als $u \to v \mid w$ abgekürzt.
- Auf der linken Seite einer Regel steht stets mindestens ein Nonterminal!

$$P \subseteq (V^* \setminus V_T^*) \times V^*$$

Ableitung

Definition (Ableitung)

Die **einschrittige Ableitung** eines Wortes v aus einem Wort u mittels einer Produktion einer Grammatik G wird notiert als $u \underset{G}{\Longrightarrow} v$. Dabei ist die Relation $\underset{G}{\Longrightarrow} \subseteq V^* \times V^*$ für alle $u, v \in V^*$ definert durch: $u \underset{G}{\Longrightarrow} v$ gdw.

$$\exists u_1, u_2 \in V^* \exists (w_l, w_r) \in P : u = u_1 w_l u_2 \text{ und } v = u_1 w_r u_2$$

Ist der Kontext klar, wird das tief gestellte G weggelassen. Ferner bedienen wir uns wieder der reflexiven, transitiven Hülle $\overset{*}{\underset{G}{\Longrightarrow}}$ für **mehrschrittige Ableitungen**.

Ableitung

Definition (Ableitung)

Die **einschrittige Ableitung** eines Wortes v aus einem Wort u mittels einer Produktion einer Grammatik G wird notiert als $u \underset{G}{\Longrightarrow} v$. Dabei ist die Relation $\underset{G}{\Longrightarrow} \subseteq V^* \times V^*$ für alle $u, v \in V^*$ definert durch: $u \underset{G}{\Longrightarrow} v$ gdw.

$$\exists u_1, u_2 \in V^* \exists (w_l, w_r) \in P : u = u_1 w_l u_2 \text{ und } v = u_1 w_r u_2$$

Ist der Kontext klar, wird das tief gestellte G weggelassen. Ferner bedienen wir uns wieder der reflexiven, transitiven Hülle $\overset{*}{\underset{G}{\Longrightarrow}}$ für **mehrschrittige Ableitungen**.

Ableitung

Definition (Ableitung)

Die **einschrittige Ableitung** eines Wortes v aus einem Wort u mittels einer Produktion einer Grammatik G wird notiert als $u \underset{G}{\Longrightarrow} v$. Dabei ist die Relation $\underset{G}{\Longrightarrow} \subseteq V^* \times V^*$ für alle $u, v \in V^*$ definert durch: $u \underset{G}{\Longrightarrow} v$ gdw.

$$\exists u_1, u_2 \in V^* \exists (w_l, w_r) \in P : u = u_1 w_l u_2 \text{ und } v = u_1 w_r u_2$$

Ist der Kontext klar, wird das tief gestellte G weggelassen. Ferner bedienen wir uns wieder der reflexiven, transitiven Hülle $\overset{*}{\underset{G}{\Longrightarrow}}$ für **mehrschrittige Ableitungen**.

Grammatiken. Beispiel

Vorgehen

Eine Ableitung funktioniert also so:

- Das aktuelle Wort w betrachten
- Treten in w linke Seiten von Regeln auf?
- Falls ja, wähle eine dieser Regeln und
- ersetze die in w auftretende linke Seite der Regel
- durch die rechte Seite dieser Regel

Beispiel

Mit $S \to ASB \mid AB$, $AB \to c$, $AcB \to c$ gilt

$$S \Rightarrow ASB \Rightarrow AABB \Rightarrow AcB \Rightarrow c$$

Grammatiken. Beispiel

Vorgehen

Eine Ableitung funktioniert also so:

- Das aktuelle Wort w betrachten
- Treten in w linke Seiten von Regeln auf?
- Falls ja, wähle eine dieser Regeln und
- ersetze die in w auftretende linke Seite der Regel
- durch die rechte Seite dieser Regel

Beispiel

Mit $S \to ASB \mid AB$, $AB \to c$, $AcB \to c$ gilt

$$S \Rightarrow ASB \Rightarrow AABB \Rightarrow AcB \Rightarrow c$$

Grammatiken. Beispiel

Vorgehen

Eine Ableitung funktioniert also so:

- Das aktuelle Wort w betrachten
- Treten in w linke Seiten von Regeln auf?
- Falls ja, wähle eine dieser Regeln und
- ersetze die in w auftretende linke Seite der Regel
- durch die rechte Seite dieser Regel

Beispiel

Mit $S \to ASB \mid AB$, $AB \to c$, $AcB \to c$ gilt

$$S \Rightarrow ASB \Rightarrow AABB \Rightarrow AcB \Rightarrow c$$

Grammatiken. Beispiel

Vorgehen

Eine Ableitung funktioniert also so:

- Das aktuelle Wort w betrachten
- Treten in w linke Seiten von Regeln auf?
- Falls ja, wähle eine dieser Regeln und
- ersetze die in w auftretende linke Seite der Regel
- durch die rechte Seite dieser Regel

Beispiel

Mit $S \to ASB \mid AB$, $AB \to c$, $AcB \to c$ gilt

$$S \Rightarrow ASB \Rightarrow AABB \Rightarrow AcB \Rightarrow c$$

Grammatiken. Beispiel

Vorgehen

Eine Ableitung funktioniert also so:

- Das aktuelle Wort w betrachten
- Treten in w linke Seiten von Regeln auf?
- Falls ja, wähle eine dieser Regeln und
- ersetze die in w auftretende linke Seite der Regel
- durch die rechte Seite dieser Regel

Beispiel

Mit $S \to ASB \mid AB$, $AB \to c$, $AcB \to c$ gilt

$$S \Rightarrow ASB \Rightarrow AABB \Rightarrow AcB \Rightarrow c$$

Grammatiken. Beispiel

Vorgehen

Eine Ableitung funktioniert also so:

- Das aktuelle Wort w betrachten
- Treten in w linke Seiten von Regeln auf?
- Falls ja, wähle eine dieser Regeln und
- ersetze die in w auftretende linke Seite der Regel
- durch die rechte Seite dieser Regel

Beispiel

Mit $S \to ASB \mid AB$, $AB \to c$, $AcB \to c$ gilt

$$S \Rightarrow ASB \Rightarrow AABB \Rightarrow AcB \Rightarrow c$$

Generierte Sprache

Definition (Generierte Sprache)

Sei $G = (V_N, V_T, P, S)$ eine Grammatik. Die von G **generierte** oder **erzeugte** Sprache ist

$$L(G) := \{w \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$$

Kontextfreie Grammatiken. Formale Definition

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** (CFG) ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq V_N \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Kontextfreie Grammatiken werden auch als **Typ-2**-Grammatiken bezeichnet.

Kontextfreie Grammatiken. Formale Definition

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** (CFG) ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq V_N \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Kontextfreie Grammatiken werden auch als **Typ-2**-Grammatiken bezeichnet.

Kontextfreie Grammatiken. Formale Definition

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** (CFG) ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq V_N \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Kontextfreie Grammatiken werden auch als **Typ-2**-Grammatiken bezeichnet.

Kontextfreie Grammatiken. Formale Definition

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** (CFG) ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq V_N \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Kontextfreie Grammatiken werden auch als **Typ-2**-Grammatiken bezeichnet.

Kontextfreie Grammatiken. Formale Definition

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** (CFG) ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq V_N \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Kontextfreie Grammatiken werden auch als **Typ-2**-Grammatiken bezeichnet.

Kontextfreie Grammatiken. Formale Definition

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** (CFG) ist ein Quadrupel $G = (V_N, V_T, P, S)$ mit

1. Dem endlichen Alphabet von **Nonterminalen** V_N.
2. Dem endlichen Alphabet von **Terminalen** V_T mit $V_T \cap V_N = \emptyset$. Das *Gesamtalphabet* wird mit $V := V_T \cup V_N$ bezeichnet.
3. Der endlichen Menge von **Produktionen** (oder *Regeln*) $P \subseteq V_N \times V^*$.
4. Dem **Startsymbol** $S \in V_N$.

Kontextfreie Grammatiken werden auch als **Typ-2**-Grammatiken bezeichnet.

Sprachfamilie CF

Definition (Sprachfamilie CF)

Ableitung und akzeptierte Sprache ist wie bei Grammatiken definiert. Die **Familie der kontextfreien Sprachen** ist dann jene Familie von Sprachen, für die es eine kontextfreie Grammatik, die sie generiert. Abgekürzt wird diese Sprachfamilie mit CF.

Typ-3 Grammatiken

Definition (Kontextfreie Grammatik (CFG))

Eine **kontextfreie Grammatik** $G = (V_N, V_T, P, S)$ heißt

- **linear**, falls $P \subseteq V_N \times (V_T^* \cdot V_N \cdot V_T^* \cup V_T^*)$
- **linkslinear**, falls $P \subseteq V_N \times (V_N \cdot V_T^* \cup V_T^*)$
- **rechtslinear**, falls $P \subseteq V_N \times (V_T^* \cdot V_N \cup V_T^*)$

Wir werden uns auf rechtslineare Grammatiken beschränken. Diese heißen auch **Typ-3**-Grammatiken oder **reguläre** Grammatiken.

Typische Produktionen rechtslinearer Grammatiken:

$$A \to bcX, A \to bY, A \to a$$

Grammatiken und Automaten

Satz

Eine Sprache $R \subseteq \Sigma^*$ ist regulär genau dann, wenn es eine rechtslineare Grammatik G gibt mit $L(G) = R$.

Die Beweisidee:

- Wird ein Buchstabe im Automaten gelesen, wird eine Kante (z, a, z') benutzt ("entlang gegangen").
- Man kann dies durch eine Produktion $[z] \to a[z']$ in der Grammatik "nachbauen" (wobei $[z]$ ein Nonterminal ist).
- Andersherum geht das auch!

Grammatiken und Automaten

Satz

Eine Sprache $R \subseteq \Sigma^*$ ist regulär genau dann, wenn es eine rechtslineare Grammatik G gibt mit $L(G) = R$.

Die Beweisidee:

- Wird ein Buchstabe im Automaten gelesen, wird eine Kante (z, a, z') benutzt ("entlang gegangen").
- Man kann dies durch eine Produktion $[z] \rightarrow a[z']$ in der Grammatik "nachbauen" (wobei $[z]$ ein Nonterminal ist).
- Andersherum geht das auch!

Automat $\rightarrow$ Grammatik

Beweis.

$R \rightarrow G$. Sei $A_1 = (Z, \Sigma, K, z_0, Z_{end})$ ein DFA mit $L(A_1) = R$. Wir definieren eine rechtslineare CFG $G_R := (V_N, V_T, P, S)$ durch:

$$\begin{aligned}
V_N &:= \{[z] \mid z \in Z\} \\
V_T &:= \Sigma \\
S &:= [z_0] \\
P &:= \{[z] \rightarrow a[z'] \mid (z, a, z') \in K\} \cup \\
&\quad\ \{[z] \rightarrow a \mid \exists z' \in Z_{end} : (z, a, z') \in K\} \cup \\
&\quad\ \{[z_0] \rightarrow \lambda \mid z_0 \in Z_{end}\}
\end{aligned}$$

□

Automat $\rightarrow$ Grammatik

Beweis.

$R \rightarrow G$. Sei $A_1 = (Z, \Sigma, K, z_0, Z_{end})$ ein DFA mit $L(A_1) = R$. Wir definieren eine rechtslineare CFG $G_R := (V_N, V_T, P, S)$ durch:

$$\begin{aligned}
V_N &:= \{[z] \mid z \in Z\} \\
V_T &:= \Sigma \\
S &:= [z_0] \\
P &:= \{[z] \rightarrow a[z'] \mid (z, a, z') \in K\} \cup \\
&\quad\ \{[z] \rightarrow a \mid \exists z' \in Z_{end} : (z, a, z') \in K\} \cup \\
&\quad\ \{[z_0] \rightarrow \lambda \mid z_0 \in Z_{end}\}
\end{aligned}$$

$\square$

Automat $\rightarrow$ Grammatik

$$\begin{aligned} P \;:=\; & \{[z] \rightarrow a[z'] \mid (z,a,z') \in K\} \cup \\ & \{[z] \rightarrow a \mid \exists z' \in Z_{end} : (z,a,z') \in K\} \cup \\ & \{[z_0] \rightarrow \lambda \mid z_0 \in Z_{end}\} \end{aligned}$$

Gibt es eine Erfolgsrechnung in A_1 und hat diese bspw. die Kanten

$$(z_0, x_1, z_1), (z_1, x_2, z_2), \ldots, (z_{n-1}, x_n, z_n)$$

mit $z_n \in Z_{end}$, so kann man mit den Regeln

$$[z_0] \rightarrow x_1 z_1, [z_1] \rightarrow x_2 z_2, \ldots, [z_{n-1}] \rightarrow x_n$$

dieses Wort auch in der Grammatik ableiten. Die Umkehrung (zu einer Ableitung gibt es auch eine Erfolgsrechnung) gilt auch, also insgesamt $L(G_R) = L(A_1)$.

Automat $\rightarrow$ Grammatik

$$\begin{aligned} P \quad := \quad & \{[z] \rightarrow a[z'] \mid (z, a, z') \in K\} \cup \\ & \{[z] \rightarrow a \mid \exists z' \in Z_{end} : (z, a, z') \in K\} \cup \\ & \{[z_0] \rightarrow \lambda \mid z_0 \in Z_{end}\} \end{aligned}$$

Gibt es eine Erfolgsrechnung in A_1 und hat diese bspw. die Kanten

$$(z_0, x_1, z_1), (z_1, x_2, z_2), \ldots, (z_{n-1}, x_n, z_n)$$

mit $z_n \in Z_{end}$, so kann man mit den Regeln

$$[z_0] \rightarrow x_1 z_1, [z_1] \rightarrow x_2 z_2, \ldots, [z_{n-1}] \rightarrow x_n$$

dieses Wort auch in der Grammatik ableiten. Die Umkehrung (zu einer Ableitung gibt es auch eine Erfolgsrechnung) gilt auch, also insgesamt $L(G_R) = L(A_1)$.

Automat → Grammatik

$$P := \{[z] \to a[z'] \mid (z,a,z') \in K\} \cup$$
$$\{[z] \to a \mid \exists z' \in Z_{end} : (z,a,z') \in K\} \cup$$
$$\{[z_0] \to \lambda \mid z_0 \in Z_{end}\}$$

Gibt es eine Erfolgsrechnung in A_1 und hat diese bspw. die Kanten

$$(z_0, x_1, z_1), (z_1, x_2, z_2), \ldots, (z_{n-1}, x_n, z_n)$$

mit $z_n \in Z_{end}$, so kann man mit den Regeln

$$[z_0] \to x_1 z_1, [z_1] \to x_2 z_2, \ldots, [z_{n-1}] \to x_n$$

dieses Wort auch in der Grammatik ableiten. Die Umkehrung (zu einer Ableitung gibt es auch eine Erfolgsrechnung) gilt auch, also insgesamt $L(G_R) = L(A_1)$.

Grammatik → Automat

Beweis.

$G \to R$. Sei $G = (V_N, V_T, P, S)$ eine rechtslineare Grammatik. Wir definieren einen NFA $A_2 := (Z, \Sigma, K, Z_{start}, Z_{end})$ mit

$$
\begin{aligned}
Z &:= \{z_A \mid A \in V_N\} \cup \{z_\lambda\} \\
\Sigma &:= V_T \\
Z_{start} &:= \{z_S\} \\
Z_{end} &:= \{z_\lambda\} \\
K &:= \{(z_Q, u, z_R) \mid Q, R \in V_N, u \in V_T^*, Q \to uR \in P\} \cup \\
&\quad\ \{(z_Q, u, z_\lambda) \mid Q \in V_N, u \in V_T^*, Q \to u \in P\}
\end{aligned}
$$

Man kann wie eben wieder einen Zusammenhang zwischen Erfolgsrechnung und Ableitung herstellen. □

Grammatik $\rightarrow$ Automat

Beweis.

$G \rightarrow R$. Sei $G = (V_N, V_T, P, S)$ eine rechtslineare Grammatik. Wir definieren einen NFA $A_2 := (Z, \Sigma, K, Z_{start}, Z_{end})$ mit

$$
\begin{aligned}
Z &:= \{z_A \mid A \in V_N\} \cup \{z_\lambda\} \\
\Sigma &:= V_T \\
Z_{start} &:= \{z_S\} \\
Z_{end} &:= \{z_\lambda\} \\
K &:= \{(z_Q, u, z_R) \mid Q, R \in V_N, u \in V_T^*, Q \rightarrow uR \in P\} \cup \\
&\quad\; \{(z_Q, u, z_\lambda) \mid Q \in V_N, u \in V_T^*, Q \rightarrow u \in P\}
\end{aligned}
$$

Man kann wie eben wieder einen Zusammenhang zwischen Erfolgsrechnung und Ableitung herstellen. □

Grammatik $\rightarrow$ Automat

Hinweis

$$
\begin{aligned}
K \quad := \quad & \{(z_Q, u, z_R) \mid Q, R \in V_N, u \in V_T^*, Q \rightarrow uR \in P\} \cup \\
& \{(z_Q, u, z_\lambda) \mid Q \in V_N, u \in V_T^*, Q \rightarrow u \in P\}
\end{aligned}
$$

Wer sich hier daran stört, dass mehr als ein Buchstabe an der Kante steht, der kann auch Zwischenzustände für die einzelnen Buchstaben des Wortes u einführen!

Grammatiken und Automaten

Satz

Eine Sprache $C \subseteq \Sigma^*$ ist kontextfrei genau dann, wenn es einen PDA gibt, der C mit leerem Keller akzeptiert.

Beweis.

Umfangreicher als obiges, aber ähnliche Idee... □

Literaturhinweis

Interessierte finden den Beweis in [HMU] oder im Skript.

Bemerkung

Da die Akzeptanzbedingungen "mit leerem Keller" und "mit Endzustand" äquivalent sind, gilt obige Aussage auch für PDAs, die mit Endzustand akzeptieren. Man beachte aber, dass das Modell in jedem Fall *nichtdeterministisch* ist!

Grammatiken und Automaten

Satz

Eine Sprache $C \subseteq \Sigma^*$ ist kontextfrei genau dann, wenn es einen PDA gibt, der C mit leerem Keller akzeptiert.

Beweis.

Umfangreicher als obiges, aber ähnliche Idee... $\square$

Literaturhinweis

Interessierte finden den Beweis in [HMU] oder im Skript.

Bemerkung

Da die Akzeptanzbedingungen "mit leerem Keller" und "mit Endzustand" äquivalent sind, gilt obige Aussage auch für PDAs, die mit Endzustand akzeptieren. Man beachte aber, dass das Modell in jedem Fall *nichtdeterministisch* ist!

Grammatiken und Automaten

Satz

Eine Sprache $C \subseteq \Sigma^*$ ist kontextfrei genau dann, wenn es einen PDA gibt, der C mit leerem Keller akzeptiert.

Beweis.

Umfangreicher als obiges, aber ähnliche Idee... □

Literaturhinweis

Interessierte finden den Beweis in [HMU] oder im Skript.

Bemerkung

Da die Akzeptanzbedingungen "mit leerem Keller" und "mit Endzustand" äquivalent sind, gilt obige Aussage auch für PDAs, die mit Endzustand akzeptieren. Man beachte aber, dass das Modell in jedem Fall *nichtdeterministisch* ist!

Abschlusseigenschaften kontextfreier Sprachen

Satz

Die Familie CF ist abgeschlossen gegenüber $\cup, \cdot$ *und* $*$.

Beweis.

Seien für $i \in \{1, 2\}$ kontextfreie Grammatiken $G_i = (V_{i,N}, V_{i,T}, P_i, S_i)$ mit $L(G_i) = L_i$.

1. Sei $G_3 := (V_{1,N} \cup V_{2,N} \cup \{S_3\}, V_{1,T} \cup V_{2,T}, P_3, S_3)$ mit $P_3 := P_1 \cup P_2 \cup \{S_3 \to S_1 \mid S_2\}$, dann ist $L_1 \cup L_2 = L(G_3)$.
2. Sei $G_4 := (V_{1,N} \cup V_{2,N} \cup \{S_4\}, V_{1,T} \cup V_{2,T}, P_4, S_4)$ mit $P_4 := P_1 \cup P_2 \cup \{S_4 \to S_1 S_2\}$, dann ist $L_1 \cdot L_2 = L(G_4)$.
3. Sei $G_5 := (V_{1,N} \cup \{S_5\}, V_{1,T}, P_5, S_5)$ mit $P_5 := P_1 \cup \{S_5 \to S_1 S_5 \mid \lambda\}$, dann ist $L_1^* = L(G_5)$.

□

Abschlusseigenschaften kontextfreier Sprachen

Satz

Die Familie CF ist abgeschlossen gegenüber $\cup, \cdot$ *und* $*$.

Beweis.

Seien für $i \in \{1, 2\}$ kontextfreie Grammatiken $G_i = (V_{i,N}, V_{i,T}, P_i, S_i)$ mit $L(G_i) = L_i$.

1. Sei $G_3 := (V_{1,N} \cup V_{2,N} \cup \{S_3\}, V_{1,T} \cup V_{2,T}, P_3, S_3)$ mit $P_3 := P_1 \cup P_2 \cup \{S_3 \to S_1 \mid S_2\}$, dann ist $L_1 \cup L_2 = L(G_3)$.
2. Sei $G_4 := (V_{1,N} \cup V_{2,N} \cup \{S_4\}, V_{1,T} \cup V_{2,T}, P_4, S_4)$ mit $P_4 := P_1 \cup P_2 \cup \{S_4 \to S_1 S_2\}$, dann ist $L_1 \cdot L_2 = L(G_4)$.
3. Sei $G_5 := (V_{1,N} \cup \{S_5\}, V_{1,T}, P_5, S_5)$ mit $P_5 := P_1 \cup \{S_5 \to S_1 S_5 \mid \lambda\}$, dann ist $L_1^* = L(G_5)$.

□

Abschlusseigenschaften kontextfreier Sprachen

Satz

Die Familie CF ist abgeschlossen gegenüber $\cup, \cdot$ *und* $*$.

Beweis.

Seien für $i \in \{1,2\}$ kontextfreie Grammatiken $G_i = (V_{i,N}, V_{i,T}, P_i, S_i)$ mit $L(G_i) = L_i$.

1. Sei $G_3 := (V_{1,N} \cup V_{2,N} \cup \{S_3\}, V_{1,T} \cup V_{2,T}, P_3, S_3)$ mit $P_3 := P_1 \cup P_2 \cup \{S_3 \to S_1 \mid S_2\}$, dann ist $L_1 \cup L_2 = L(G_3)$.
2. Sei $G_4 := (V_{1,N} \cup V_{2,N} \cup \{S_4\}, V_{1,T} \cup V_{2,T}, P_4, S_4)$ mit $P_4 := P_1 \cup P_2 \cup \{S_4 \to S_1 S_2\}$, dann ist $L_1 \cdot L_2 = L(G_4)$.
3. Sei $G_5 := (V_{1,N} \cup \{S_5\}, V_{1,T}, P_5, S_5)$ mit $P_5 := P_1 \cup \{S_5 \to S_1 S_5 \mid \lambda\}$, dann ist $L_1^* = L(G_5)$.

□

Abschlusseigenschaften kontextfreier Sprachen

Satz

Die Familie CF ist abgeschlossen gegenüber $\cup, \cdot$ *und* $*$.

Beweis.

Seien für $i \in \{1, 2\}$ kontextfreie Grammatiken $G_i = (V_{i,N}, V_{i,T}, P_i, S_i)$ mit $L(G_i) = L_i$.

1. Sei $G_3 := (V_{1,N} \cup V_{2,N} \cup \{S_3\}, V_{1,T} \cup V_{2,T}, P_3, S_3)$ mit $P_3 := P_1 \cup P_2 \cup \{S_3 \to S_1 \mid S_2\}$, dann ist $L_1 \cup L_2 = L(G_3)$.
2. Sei $G_4 := (V_{1,N} \cup V_{2,N} \cup \{S_4\}, V_{1,T} \cup V_{2,T}, P_4, S_4)$ mit $P_4 := P_1 \cup P_2 \cup \{S_4 \to S_1 S_2\}$, dann ist $L_1 \cdot L_2 = L(G_4)$.
3. Sei $G_5 := (V_{1,N} \cup \{S_5\}, V_{1,T}, P_5, S_5)$ mit $P_5 := P_1 \cup \{S_5 \to S_1 S_5 \mid \lambda\}$, dann ist $L_1^* = L(G_5)$.

$\square$

Zur Nachbereitung

Hier müsste nun noch $L_1 \cup L_2 = L(G_3)$, $L_1 \cdot L_2 = L(G_4)$ und $L_1^* = L(G_5)$ gezeigt werden. Dies ist aber recht einfach. Mündlich ...

Abschlusseigenschaften kontextfreier Sprachen

Satz

Die Familie CF ist nicht abgeschlossen gegenüber $\cap$*, Komplementbildung und Mengendifferenz.*

Beweis.

Der Beweis gelingt stets durch Angabe eines Gegenbeispiels. Siehe Skript, Theorem 9.38. $\square$

Die Chomsky-Normalform

Wir wollen nun eine **Normalform** für kontextfreie Grammtiken herstellen. Dies kann uns

- bei Beweisen und
- einigen Algorithmen

helfen.

Die Chomsky-Normalform

Definition

Eine kontextfreie Grammatik $G = (V_N, V_T, P, S)$ ist in **Chomsky-Normalform** (CNF) genau dann, wenn alle Produktionen von der Form

- $A \rightarrow BC$
- $A \rightarrow a$

mit $A, B, C \in V_N$ und $a \in V_T$ sind.

Die Chomsky-Normalform

Satz

Zu jeder CFG G kann eine äquivalente CFG $G' = (V'_N, V'_T, P', S')$ konstruiert werden für die folgendes gilt:

1. *Ist $\lambda \notin L(G)$, so ist G' in CNF.*
2. *Ist $\lambda \in L(G)$, so ist $S' \to \lambda$ die einzige λ-Produktion in P' und alle anderen Produktionen sind in Chomsky-Normalform. Außerdem kommt S' auf keiner rechten Seite einer Regel aus P' vor. Dies nennen wir dann* **erweiterte Chomsky-Normalform**.

Die Chomsky-Normalform

Im Beweis werden in sechs Schritten zu $G = (V_N, V_T, P, S)$ jeweils neue, **äquivalente** Grammatiken erzeugt:

1. λ-frei machen
2. Reduzieren (produktiv und erreichbar)
3. Kettenregeln entfernen
4. Ersetzen langer Terminalregeln
5. Verkürzen zu langer Regeln
6. Evtl. Hinzunahme einer λ-Regel

Bemerkung

Nachfolgend geben wir die Konstruktion an. Die Beweise, dass die einzelnen Grammatiken äquivalent sind, überspringen wir (meist sind sie aber recht einfach).

1. λ-frei machen

1. Zu jedem $A \in V_N$ bestimmen wir, ob $A \stackrel{*}{\Longrightarrow} \lambda$ gilt:

$$\begin{aligned} M_0 &:= \{A \in V_N \mid A \to \lambda \in P\} \\ M_{i+1} &:= M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \to w \in P\} \end{aligned}$$

   Es gibt ein k mit $M_k = M_{k+1}$. Wir setzen $V_\lambda := M_k$ und hören auf.

2. Wir definieren eine Substitution $\sigma : V^* \to 2^{V^*}$ durch

$$\begin{aligned} \sigma(A) &:= \{A\}, \text{ falls } A \in (V \setminus V_\lambda) \\ \sigma(A) &:= \{\lambda, A\}, \text{ falls } A \in V_\lambda \end{aligned}$$

3. $G_1 := (V_N, V_T, P_1, S)$ entsteht durch

$$P_1 := \{A \to v \mid v \neq \lambda, v \in \sigma(w) \text{ für } A \to w \in P\}$$

   Es ist $L(G_1) = L(G) \setminus \{\lambda\}$.

1. λ-frei machen

1. Zu jedem $A \in V_N$ bestimmen wir, ob $A \stackrel{*}{\Longrightarrow} \lambda$ gilt:

$$\begin{aligned} M_0 &:= \{A \in V_N \mid A \to \lambda \in P\} \\ M_{i+1} &:= M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \to w \in P\} \end{aligned}$$

   Es gibt ein k mit $M_k = M_{k+1}$. Wir setzen $V_\lambda := M_k$ und hören auf.

2. Wir definieren eine Substitution $\sigma : V^* \to 2^{V^*}$ durch

$$\begin{aligned} \sigma(A) &:= \{A\}, \text{ falls } A \in (V \setminus V_\lambda) \\ \sigma(A) &:= \{\lambda, A\}, \text{ falls } A \in V_\lambda \end{aligned}$$

3. $G_1 := (V_N, V_T, P_1, S)$ entsteht durch

$$P_1 := \{A \to v \mid v \neq \lambda, v \in \sigma(w) \text{ für } A \to w \in P\}$$

   Es ist $L(G_1) = L(G) \setminus \{\lambda\}$.

1. λ-frei machen

1. Zu jedem $A \in V_N$ bestimmen wir, ob $A \overset{*}{\Longrightarrow} \lambda$ gilt:

$$\begin{aligned} M_0 &:= \{A \in V_N \mid A \to \lambda \in P\} \\ M_{i+1} &:= M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \to w \in P\} \end{aligned}$$

   Es gibt ein k mit $M_k = M_{k+1}$. Wir setzen $V_\lambda := M_k$ und hören auf.

2. Wir definieren eine Substitution $\sigma : V^* \to 2^{V^*}$ durch

$$\begin{aligned} \sigma(A) &:= \{A\}, \text{ falls } A \in (V \setminus V_\lambda) \\ \sigma(A) &:= \{\lambda, A\}, \text{ falls } A \in V_\lambda \end{aligned}$$

3. $G_1 := (V_N, V_T, P_1, S)$ entsteht durch

$$P_1 := \{A \to v \mid v \neq \lambda, v \in \sigma(w) \text{ für } A \to w \in P\}$$

   Es ist $L(G_1) = L(G) \setminus \{\lambda\}$.

1. λ-frei machen

Z.B. ist bei $C \to \lambda, B \to xC \mid ACx, A \to B \mid CC$

$$\begin{aligned} M_0 &= \{C\} \\ M_1 &= \{C, A\} \\ M_2 &= M_1 \end{aligned}$$

Beispiel für die Anwendung der Substitution (hat nichts mit obigem zu tun!): Wendet man σ auf ein Wort u an, werden in u einige (inkl. alle oder keins) Symbole aus V_λ gestrichen. Z.B. mit $V_\lambda = \{A, C\}$

$$\begin{aligned} \sigma(xABCy) &= \{xABCy, xBCy, xABy, xBy\} \\ \sigma(xACA) &= \{xACA, xCA, xAA, xAC, xA, xC, x\} \end{aligned}$$

1. λ-frei machen

Z.B. ist bei $C \to \lambda, B \to xC \mid ACx, A \to B \mid CC$

$$\begin{aligned} M_0 &= \{C\} \\ M_1 &= \{C, A\} \\ M_2 &= M_1 \end{aligned}$$

Beispiel für die Anwendung der Substitution (hat nichts mit obigem zu tun!): Wendet man σ auf ein Wort u an, werden in u einige (inkl. alle oder keins) Symbole aus V_λ gestrichen. Z.B. mit $V_\lambda = \{A, C\}$

$$\begin{aligned} \sigma(xABCy) &= \{xABCy, xBCy, xABy, xBy\} \\ \sigma(xACA) &= \{xACA, xCA, xAA, xAC, xA, xC, x\} \end{aligned}$$

1. λ-frei machen

Anmerkung

Man kann diese Technik auch benutzen, um $\lambda \in L(G)$ zu entscheiden oder um eine äquivalente Grammatik zu konstruieren, die nur eine λ-Produktion hat:

- $\lambda \in L(G)$ gilt, wenn $S \in V_\lambda$ ist.
- Ist $\lambda \in L(G)$, so fügt man zu der eben konstruierten Grammatik G_1 ein neues Startsymbol S_{neu} hinzu sowie die neuen Produktionen $S_{neu} \to \lambda$ sowie $S_{neu} \to w$ für jede Produktion $S \to w \in P_1$.

2. Reduzieren

Zuerst alle nicht produktiven Nonterminal entfernen. Produktiv ist ein Nonterminal dabei, wenn es zu einem Terminalwort abgeleitet werden kann. Wir berechnen ähnlich wie eben

$$\begin{aligned} M_0 &:= V_T \\ M_{i+1} &:= M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \to w \in P_1\} \end{aligned}$$

Es gibt ein k mit $M_{k+1} = M_k$ dort hören wir auf und bestimmen

$$G' := (V'_N, V'_T, P', S') = ((M_k \cap V_N) \cup \{S\}, V_T, P_1 \cap (M_k \times M_k^*), S)$$

2. Reduzieren

Zuerst alle nicht produktiven Nonterminal entfernen. Produktiv ist ein Nonterminal dabei, wenn es zu einem Terminalwort abgeleitet werden kann. Wir berechnen ähnlich wie eben

$$\begin{aligned} M_0 &:= V_T \\ M_{i+1} &:= M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \to w \in P_1\} \end{aligned}$$

Es gibt ein k mit $M_{k+1} = M_k$ dort hören wir auf und bestimmen

$$G' := (V'_N, V'_T, P', S') = ((M_k \cap V_N) \cup \{S\}, V_T, P_1 \cap (M_k \times M_k^*), S)$$

2. Reduzieren

Zuerst alle nicht produktiven Nonterminal entfernen. Produktiv ist ein Nonterminal dabei, wenn es zu einem Terminalwort abgeleitet werden kann. Wir berechnen ähnlich wie eben

$$\begin{aligned} M_0 &:= V_T \\ M_{i+1} &:= M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \to w \in P_1\} \end{aligned}$$

Es gibt ein k mit $M_{k+1} = M_k$ dort hören wir auf und bestimmen

$$G' := (V'_N, V'_T, P', S') = ((M_k \cap V_N) \cup \{S\}, V_T, P_1 \cap (M_k \times M_k^*), S)$$

2. Reduzieren

Als nächstes werden die erreichbaren Nonterminale bestimmt.

$$\begin{aligned} M_0 &:= \{S'\} \\ M_{i+1} &:= M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : A \to uBv \in P'\} \end{aligned}$$

Wieder gibt es ein k mit $M_{k+1} = M_k$. Wir setzen dann $G_2 := (V_{N_2}, V_{T_2}, P_2, S)$, indem wir von G' nur solche Produktionen übernehmen, deren rechte und linke Seiten nur Nonterminale aus M_k (und Terminale) enthalten. V'_N kann dabei u.U. auch verkleinert werden.

2. Reduzieren

Als nächstes werden die erreichbaren Nonterminale bestimmt.

$$\begin{aligned} M_0 &:= \{S'\} \\ M_{i+1} &:= M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : A \to uBv \in P'\} \end{aligned}$$

Wieder gibt es ein k mit $M_{k+1} = M_k$. Wir setzen dann $G_2 := (V_{N_2}, V_{T_2}, P_2, S)$, indem wir von G' nur solche Produktionen übernehmen, deren rechte und linke Seiten nur Nonterminale aus M_k (und Terminale) enthalten. V'_N kann dabei u.U. auch verkleinert werden.

2. Reduzieren

Anmerkung

Mit dieser Technik kann zu jeder Grammatik G mit $L(G) \neq \emptyset$ eine äquivalente reduzierte Grammatik G konstruiert werden. Alle Nonterminale sind dann produktiv und erreichbar, also umgangssprachlich “nützlich”.

Beispiel: In den Präsenzaufgaben ;-)

2. Reduzieren

Anmerkung

Mit dieser Technik kann zu jeder Grammatik G mit $L(G) \neq \emptyset$ eine äquivalente reduzierte Grammatik G konstruiert werden. Alle Nonterminale sind dann produktiv und erreichbar, also umgangssprachlich "nützlich".

Beispiel: In den Präsenzaufgaben ;-)

3. Kettenregeln entfernen

Im dritten Schritt ändert sich nur die Menge der Produktionen. Wir wollen keine (Ketten-)Regeln $A \rightarrow B$ haben.

Wir defineren

$$A << B \text{ gdw. } A \rightarrow B \in P_2$$

Mit $<<^*$ sei die reflexive, transitive Hülle von $<<$ bezeichnet. Es ist nun $G_3 = (V_{N_3}, V_{T_3}, P_3, S_3) = (V_{N_2}, V_{T_2}, P_3, S)$

$$P_3 := \{A \rightarrow w \mid w \notin V_{N_3} \wedge \exists B \rightarrow w \in P_2 \wedge A <<^* B\}$$

3. Kettenregeln entfernen

Im dritten Schritt ändert sich nur die Menge der Produktionen. Wir wollen keine (Ketten-)Regeln $A \to B$ haben.

Wir defineren

$$A << B \text{ gdw. } A \to B \in P_2$$

Mit $<<^*$ sei die reflexive, transitive Hülle von $<<$ bezeichnet. Es ist nun $G_3 = (V_{N_3}, V_{T_3}, P_3, S_3) = (V_{N_2}, V_{T_2}, P_3, S)$

$$P_3 := \{A \to w \mid w \notin V_{N_3} \wedge \exists B \to w \in P_2 \wedge A <<^* B\}$$

3. Kettenregeln entfernen

Beispiel

$$P_3 := \{A \to w \mid w \notin V_{N_3} \land \exists B \to w \in P_2 \land A <<^* B\}$$

Sei $A \to B \mid cB, B \to C \mid d, C \to e$, dann ist

$$A <<^* B, B <<^* C, A <<^* C$$

und damit folgt als Produktionen in P_3:

- $A \to cB, B \to d, C \to e$
- $A \to d$
- $B \to e$
- $A \to e$

3. Kettenregeln entfernen

Beispiel

$$P_3 := \{A \to w \mid w \notin V_{N_3} \wedge \exists B \to w \in P_2 \wedge A <<^* B\}$$

Sei $A \to B \mid cB, B \to C \mid d, C \to e$, dann ist

$$A <<^* B, B <<^* C, A <<^* C$$

und damit folgt als Produktionen in P_3:

- $A \to cB, B \to d, C \to e$
- $A \to d$
- $B \to e$
- $A \to e$

4. Ersetzen langer Terminalregeln

In der Chomsky-NF darf auf der rechten Seite stets nur genau ein Terminalzeichen stehen. Gibt es daher nun noch

- Produktionen, deren rechte Seite
  - länger als 1 ist und
  - Terminale enthalten.
- Dann erstezen wir auf diesen rechten Seiten jedes Terminal a durch ein Nonterminal $\langle a \rangle$.

4. Ersetzen langer Terminalregeln

Beispiel

Hat man bspw. Regeln

$$A \to bc \mid Bc \mid c$$

so entstehen daraus die Regeln

$$A \to \langle b \rangle \langle c \rangle \mid B \langle c \rangle \mid c$$

4. Ersetzen langer Terminalregeln

Beispiel

Hat man bspw. Regeln

$$A \to bc \mid Bc \mid c$$

so entstehen daraus die Regeln

$$A \to \langle b \rangle \langle c \rangle \mid B \langle c \rangle \mid c$$

5. Verkürzen zu langer Regeln

Sind nun noch Regeln $A \to w$ mit $|w| > 2$ vorhanden, so müssen wir diese auf die Länge 2 bringen. Wir machen das so:

$$
\begin{aligned}
G_5 &:= (V_{N_5}, V_{T_3}, P_5, S) \text{ mit} \\
V_{N_5} &:= \{\langle v\rangle \mid \exists u \neq \lambda : \exists A \to w \in P_4 : \; w = vu \wedge |v| \geq 2\} \cup V_{N_4} \\
P_5 &:= \{A \to \langle v\rangle x \mid A \to w \in P_4 : |w| \geq 3 \wedge w = vx \wedge x \in V_{N_4}\} \cup \\
& \{\langle v\rangle \to \langle u\rangle y \mid \langle u\rangle, \langle v\rangle \in (V_{N_5} \setminus V_{N_4}) \wedge y \in V_{N_4} \wedge v = uy\} \cup \\
& \{\langle v\rangle \to xy \mid \langle v\rangle \in (V_{N_5} \setminus V_{N_4}) \wedge x, y \in V_{N_4} \wedge v = xy\} \cup \\
& \{A \to w \mid A \to w \in P_4 \wedge |w| \leq 2\}
\end{aligned}
$$

5. Verkürzen zu langer Regeln

Sind nun noch Regeln $A \to w$ mit $|w| > 2$ vorhanden, so müssen wir diese auf die Länge 2 bringen. Wir machen das so:

$$
\begin{aligned}
G_5 &:= (V_{N_5}, V_{T_3}, P_5, S) \text{ mit} \\
V_{N_5} &:= \{\langle v \rangle \mid \exists u \neq \lambda : \exists A \to w \in P_4 : \ w = vu \wedge |v| \geq 2\} \cup V_{N_4} \\
P_5 &:= \{A \to \langle v \rangle x \mid A \to w \in P_4 : |w| \geq 3 \wedge w = vx \wedge x \in V_{N_4}\} \cup \\
&\quad \{\langle v \rangle \to \langle u \rangle y \mid \langle u \rangle, \langle v \rangle \in (V_{N_5} \setminus V_{N_4}) \wedge y \in V_{N_4} \wedge v = uy\} \cup \\
&\quad \{\langle v \rangle \to xy \mid \langle v \rangle \in (V_{N_5} \setminus V_{N_4}) \wedge x, y \in V_{N_4} \wedge v = xy\} \cup \\
&\quad \{A \to w \mid A \to w \in P_4 \wedge |w| \leq 2\}
\end{aligned}
$$

5. Verkürzen zu langer Regeln

Beispiel

Sei z.B. $A \to BCDE$, dann werden die Produktionen

$$\begin{aligned} A &\to (BCD)E \\ (BCD) &\to (BC)D \\ (BC) &\to BC \end{aligned}$$

mit den neuen Nonterminalen (BCD) und (BC) hinzugefügt.

Hinweis

Um bei den letzten beiden Schritten nicht durcheinander zu kommen, lohnt es sich manchmal mit unterschiedlichen Klammertypen zu arbeiten.

5. Verkürzen zu langer Regeln

Beispiel

Sei z.B. $A \to BCDE$, dann werden die Produktionen

$$\begin{array}{rcl} A & \to & (BCD)E \\ (BCD) & \to & (BC)D \\ (BC) & \to & BC \end{array}$$

mit den neuen Nonterminalen (BCD) und (BC) hinzugefügt.

Hinweis

Um bei den letzten beiden Schritten nicht durcheinander zu kommen, lohnt es sich manchmal mit unterschiedlichen Klammertypen zu arbeiten.

6. λ-Regel hinzunehmen

Galt im ersten Schritt $S \in V_\lambda$, war also $\lambda \in L(G)$, so fügen wir zu der zuletzt konstruierten Grammatik noch ein neues Startsymbol S_{neu} hinzu sowie die neuen Produktionen $S_{neu} \to \lambda$ sowie $S_{neu} \to w$ für jede Produktion $S \to w \in P_5$.

Die so erhaltene Grammatik G_6 erfüllt die im Satz aufgestellten Bedingungen.

Grenzen kontextfreier Sprachen

Die Chomsky-Normalform kann benutzt werden, um das *Pumping Lemma kontextfreier Sprachen* zu zeigen.

Dies kann dann wieder benutzt werden, um die Grenzen kontextfreier Sprachen nachzuweisen, d.h. um für Sprachen zu zeigen, dass sie nicht kontextfrei sind.

Pumping Lemma kontextfreier Sprachen

Lemma (Pumping Lemma II)

Sei $L \in CF$ eine kontextfreie Sprache. Dann gibt es ein $n \in \mathbb{N}$, so dass jedes Wort $z \in L$ mit $|z| \geq n$ in die Form $z = uvwxy$ zerlegt werden kann, wobei

1. $|vwx| \leq n$
2. $|vx| \geq 1$
3. $uv^iwx^iy \in L$ *für jedes* $i \in \mathbb{N}$ *(inkl. der* 0*)*

Pumping Lemma - Beweis

Beweis

Wie beim vorherigen Pumping Lemma ist auch dieser Beweis ein kombinatorisches Argument. Hier geht die Argumentation über eine Grammatik in Chomsky-Normalform. Man setzt dann $k := |V_N|$ und $n := 2^k$. Da jeder Knoten im *Ableitungsbaum* einen oder zwei Nachfolger hat (Chomsky-Normalform!), kann man dann zeigen, dass ein Nonterminal doppelt auftreten muss. Dies kann dann benutzt werden, um die Bedingungen zu zeigen.
Interessierte finden den vollständigen Beweis im Skript.

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Angenommen L wäre kontextfrei. Dann gilt das Pumping Lemma für kontextfreie Sprachen. Sei k die Zahl aus dem Pumping Lemma. Wir betrachten das Wort $z = a^k b^k c^k$. Es gilt $z \in L$ und $|z| \geq k$. Sei nun $z = uvwxy$ eine Zerlegung von z mit $|vwx| \leq k$ und $|vx| \geq 1$. Es gilt nun

1. $vx \in \{a\}^+$ oder
2. $vx \in \{b\}^+$ oder
3. $vx \in \{c\}^+$ oder
4. $vx \in \{a\}^+\{b\}^+$ oder
5. $vx \in \{b\}^+\{c\}^+$

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Angenommen L wäre kontextfrei. Dann gilt das Pumping Lemma für kontextfreie Sprachen. Sei k die Zahl aus dem Pumping Lemma. Wir betrachten das Wort $z = a^k b^k c^k$. Es gilt $z \in L$ und $|z| \geq k$. Sei nun $z = uvwxy$ eine Zerlegung von z mit $|vwx| \leq k$ und $|vx| \geq 1$. Es gilt nun

1. $vx \in \{a\}^+$ oder
2. $vx \in \{b\}^+$ oder
3. $vx \in \{c\}^+$ oder
4. $vx \in \{a\}^+\{b\}^+$ oder
5. $vx \in \{b\}^+\{c\}^+$

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Angenommen L wäre kontextfrei. Dann gilt das Pumping Lemma für kontextfreie Sprachen. Sei k die Zahl aus dem Pumping Lemma. Wir betrachten das Wort $z = a^k b^k c^k$. Es gilt $z \in L$ und $|z| \geq k$. Sei nun $z = uvwxy$ eine Zerlegung von z mit $|vwx| \leq k$ und $|vx| \geq 1$. Es gilt nun

1. $vx \in \{a\}^+$ oder
2. $vx \in \{b\}^+$ oder
3. $vx \in \{c\}^+$ oder
4. $vx \in \{a\}^+\{b\}^+$ oder
5. $vx \in \{b\}^+\{c\}^+$

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Angenommen L wäre kontextfrei. Dann gilt das Pumping Lemma für kontextfreie Sprachen. Sei k die Zahl aus dem Pumping Lemma. Wir betrachten das Wort $z = a^k b^k c^k$. Es gilt $z \in L$ und $|z| \geq k$. Sei nun $z = uvwxy$ eine Zerlegung von z mit $|vwx| \leq k$ und $|vx| \geq 1$. Es gilt nun

1. $vx \in \{a\}^+$ oder
2. $vx \in \{b\}^+$ oder
3. $vx \in \{c\}^+$ oder
4. $vx \in \{a\}^+\{b\}^+$ oder
5. $vx \in \{b\}^+\{c\}^+$

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Angenommen L wäre kontextfrei. Dann gilt das Pumping Lemma für kontextfreie Sprachen. Sei k die Zahl aus dem Pumping Lemma. Wir betrachten das Wort $z = a^k b^k c^k$. Es gilt $z \in L$ und $|z| \geq k$. Sei nun $z = uvwxy$ eine Zerlegung von z mit $|vwx| \leq k$ und $|vx| \geq 1$. Es gilt nun

1. $vx \in \{a\}^+$ oder
2. $vx \in \{b\}^+$ oder
3. $vx \in \{c\}^+$ oder
4. $vx \in \{a\}^+\{b\}^+$ oder
5. $vx \in \{b\}^+\{c\}^+$

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Angenommen L wäre kontextfrei. Dann gilt das Pumping Lemma für kontextfreie Sprachen. Sei k die Zahl aus dem Pumping Lemma. Wir betrachten das Wort $z = a^k b^k c^k$. Es gilt $z \in L$ und $|z| \geq k$. Sei nun $z = uvwxy$ eine Zerlegung von z mit $|vwx| \leq k$ und $|vx| \geq 1$. Es gilt nun

1. $vx \in \{a\}^+$ oder
2. $vx \in \{b\}^+$ oder
3. $vx \in \{c\}^+$ oder
4. $vx \in \{a\}^+\{b\}^+$ oder
5. $vx \in \{b\}^+\{c\}^+$

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Für die ersten drei Fälle führt uv^2wx^2y zum Widerspruch, da dann $a^{n+j}b^nc^n$ (bzw. $a^nb^{n+j}c^n$ usw.) nicht mehr in L ist.

Für die letzten beiden Fälle gibt es noch zwei weitere Fallunterscheidungen:

- Enthält v oder x sowohl a als auch b (bzw. b als auch c), dann ist uv^2wx^2y nicht mehr in $\{a\}^*\{b\}^*\{c\}^*$ und damit nicht in L.
- Ist bspw. $v \in \{a\}^+$ und $x \in \{b\}^+$, so ist uv^2wx^2y dennoch ein Widerspruch, da die Anzahl der c nicht mit den a und bs übereinstimmt.

□

Pumping Lemma - Beispiel

Behauptung

$L := \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist nicht kontextfrei.

Beweis.

Für die ersten drei Fälle führt uv^2wx^2y zum Widerspruch, da dann $a^{n+j}b^nc^n$ (bzw. $a^nb^{n+j}c^n$ usw.) nicht mehr in L ist.

Für die letzten beiden Fälle gibt es noch zwei weitere Fallunterscheidungen:

- Enthält v oder x sowohl a als auch b (bzw. b als auch c), dann ist uv^2wx^2y nicht mehr in $\{a\}^*\{b\}^*\{c\}^*$ und damit nicht in L.
- Ist bspw. $v \in \{a\}^+$ und $x \in \{b\}^+$, so ist uv^2wx^2y dennoch ein Widerspruch, da die Anzahl der c nicht mit den a und bs übereinstimmt.

$\square$

Pumping Lemma - Anmerkung

Hinweis

Anders als beim ersten Pumping Lemma gilt hier:

- das vwx "wandert" über das Wort
- es sind daher oft mehrere Fallunterscheidungen nötig
- oft so etwas wie "in einem Block zu sein" bzw. "an der Grenze zweier Blöcke zu sein".
- Viele Fälle kann man aber analog behandeln!

Und wie beim ersten Pumping Lemma gilt: Man darf auch das Wort uv^0wx^0y betrachten!

Pumping Lemma - Anmerkung

Hinweis

Anders als beim ersten Pumping Lemma gilt hier:

- das vwx “wandert” über das Wort
- es sind daher oft mehrere Fallunterscheidungen nötig
- oft so etwas wie “in einem Block zu sein” bzw. “an der Grenze zweier Blöcke zu sein”.
- Viele Fälle kann man aber analog behandeln!

Und wie beim ersten Pumping Lemma gilt: Man darf auch das Wort uv^0wx^0y betrachten!

Pumping Lemma - Anmerkung

Hinweis

Anders als beim ersten Pumping Lemma gilt hier:

- das vwx “wandert” über das Wort
- es sind daher oft mehrere Fallunterscheidungen nötig
- oft so etwas wie “in einem Block zu sein” bzw. “an der Grenze zweier Blöcke zu sein”.
- Viele Fälle kann man aber analog behandeln!

Und wie beim ersten Pumping Lemma gilt: Man darf auch das Wort uv^0wx^0y betrachten!

Ausblick

Die Chomsky-NF ist beim Beweis des Pumping Lemmas hilfreich.

Sie ist auch beim **Algorithmus zur Lösung des Wortproblems für kontextfreie Sprachen** wichtig. Darauf kommen wir später zu sprechen.

Zusammenfassung

Wir haben heute gesehen, dass

- Typ-3 Grammatiken und DFAs äquivalent sind
- Typ-2 Grammatiken und PDAs äquivalent sind

Wir haben weiterhin

- Abschlusseigenschaften von CF betrachtet
- Die Chomsky-Normalform hergestellt
  - Inkl. dem Test ob $\lambda \in L(G)$ gilt (Schritt 1)
  - Inkl. dem Reduzieren von Grammatiken (Schritt 2)
- Das zweite Pumping Lemma kennengelernt und damit
  - gesehen, dass es Sprachen gibt, die nicht in CF sind

Zusammenfassung

Wir haben heute gesehen, dass

- Typ-3 Grammatiken und DFAs äquivalent sind
- Typ-2 Grammatiken und PDAs äquivalent sind

Wir haben weiterhin

- Abschlusseigenschaften von CF betrachtet
- Die Chomsky-Normalform hergestellt
  - Inkl. dem Test ob $\lambda \in L(G)$ gilt (Schritt 1)
  - Inkl. dem Reduzieren von Grammatiken (Schritt 2)
- Das zweite Pumping Lemma kennengelernt und damit
  - gesehen, dass es Sprachen gibt, die nicht in CF sind