

# Formale Grundlagen der Informatik 1

## Kapitel 8

### Eigenschaften und Grenzen kontextfreier Sprachen

Frank Heitmann  
heitmann@informatik.uni-hamburg.de

26. April 2016

## 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**  $S \in V_N$ .

## 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**  $S \in V_N$ .

## Ableitung

### Definition (Ableitung)

Die **einschrittige Ableitung** eines Wortes  $v$  aus einem Wort  $u$  mittels einer Produktion einer Grammatik  $G$  wird notiert als  $u \xrightarrow{G} v$ . Dabei ist die Relation  $\xrightarrow{G} \subseteq V^* \times V^*$  für alle

$u, v \in V^*$  definiert durch:  $u \xrightarrow{G} v$  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  $\xrightarrow{*}_G$  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  $S \rightarrow XY \mid \lambda$ ,  $X \rightarrow aXb \mid ab$ ,  $Y \rightarrow cY \mid c$  gilt

$S \Rightarrow XY \Rightarrow aXbY \Rightarrow aXbcY \Rightarrow aXbccY \Rightarrow aabbccY \Rightarrow aabbccc$

## Generierte Sprache und CF

### Definition (Generierte Sprache)

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

$$L(G) := \{w \in V_T^* \mid S \xrightarrow[G]{*} w\}$$

### Definition (Sprachfamilie CF)

Die **Familie der kontextfreien Sprachen** ist dann jene Familie von Sprachen, für die es eine kontextfreie Grammatik gibt, die sie generiert. Abgekürzt wird diese Sprachfamilie mit CF.

## Zusammenfassung

Definitionen bisher:

- Grammatik
  - Nonterminal, Terminal, Produktion, Startsymbol
  - Ableitung
  - (Generierte) Sprache, Äquivalenz
- Kontextfreie Grammatik / Typ-2-Grammatik
  - $\lambda$ -Produktionen,  $\lambda$ -frei
- Sprachfamilie CF

## Beispiel 1

$$M = a^*b^* = \{a^n b^m \mid n, m \in \mathbb{N}\}$$

Die Grammatik  $G$  ist gegeben durch die Produktionen

$$S \rightarrow AB, A \rightarrow aA \mid \lambda, B \rightarrow bB \mid \lambda$$

Der Vollständigkeit halber:

$$V_N := \{S, A, B\}$$

$$V_T := \{a, b\}$$

Das Startsymbol ist  $S$ .

## Beispiel 1

$$M = a^*b^* = \{a^n b^m \mid n, m \in \mathbb{N}\}$$

$$S \rightarrow AB, A \rightarrow aA \mid \lambda, B \rightarrow bB \mid \lambda$$

$M \subseteq L(G)$ . Sei  $w = a^n b^m \in M$ . In  $G$  können wir nun wie folgt ableiten:

$$\begin{aligned} S &\Rightarrow AB \Rightarrow aAB \Rightarrow \dots \Rightarrow a^n AB \Rightarrow a^n B \\ &\Rightarrow a^n bB \Rightarrow \dots \Rightarrow a^n b^m B \Rightarrow a^n b^m \end{aligned}$$

Also gilt auch  $w \in L(G)$ .

## Beispiel 1

$$M = a^*b^* = \{a^n b^m \mid n, m \in \mathbb{N}\}$$

$$S \rightarrow AB, A \rightarrow aA \mid \lambda, B \rightarrow bB \mid \lambda$$

$L(G) \subseteq M$ . An den Produktionen erkennen wir zunächst, dass  $w \in \{a\}^*$  für jedes  $w$  mit  $A \xRightarrow{*} w$  gilt, denn  $i$ -malige Anwendung von  $A \rightarrow aA$  führt zu der Satzform  $a^i A$ . Dann kann mit  $A \rightarrow \lambda$  das  $A$  gelöscht werden. Ebenso gilt  $v \in \{b\}^*$  für jedes  $v$  mit  $B \xRightarrow{*} v$ . Eine Ableitung eines Wortes  $u \in L(G)$  muss nun mit  $S \Rightarrow AB$  beginnen. Anschließend kann wie eben erläutert  $A$  zu  $a^i$  und  $B$  zu  $b^j$  mit  $i, j \in \mathbb{N}$  abgeleitet werden, d.h.  $u$  hat die Form  $a^i b^j$ . Dies zeigt  $u \in M$ .

## Beispiel 2

$$M = \{a^n b^n \mid n \in \mathbb{N}\}$$

$$S \rightarrow aSb \mid \lambda$$

$M \subseteq L(G)$ . Sei  $w = a^n b^n \in M$ . In  $G$  können wir nun wie folgt ableiten:

$$S \Rightarrow aSb \Rightarrow \dots \Rightarrow a^n Sb^n \Rightarrow a^n b^n$$

Also gilt auch  $w \in L(G)$ .

## Beispiel 2

$$M = \{a^n b^n \mid n \in \mathbb{N}\}$$

$$S \rightarrow aSb \mid \lambda$$

$L(G) \subseteq M$ . Sei  $w \in L(G)$ . Da wir nur zwei Regeln haben, ist die Argumentation recht einfach. Die einzige Möglichkeit zu einem Wort zu kommen ist die Produktion  $S \rightarrow \lambda$ . Vorher kann  $i$ -mal die Produktion  $S \rightarrow aSb$  benutzt werden.  $w$  kann damit nur die Form  $a^i b^i$  haben und damit gilt auch  $w \in M$ .

## Zur Konstruktion von Grammatiken

### Konstruktionstipps

Zwei Tipps bei der Konstruktion von Grammatiken:

- 1 Will man eine Zeichenkette "sequentuell" aufbauen, wie bei  $a^k$ , so kann man das Nonterminal meist an den Rand setzen  $S \rightarrow aS$  (vgl. das erste Beispiel oben).
- 2 Will man einen "linken" und einen "rechten" Teil miteinander in Beziehung setzen, wie bei  $a^n b^n$ , so klappt dies meist, indem das Nonterminal nach innen wandert:  $S \rightarrow aSb$  (vgl. das zweite Beispiel oben).

## Fragen...

Welche Sprache generiert die Grammatik  $S \rightarrow aSbc \mid \lambda$  ?

- 1  $a^n b^n c^n$
- 2  $a^n (bc)^n$
- 3  $(ab)^n c^n$
- 4  $(abc)^n$
- 5 keine davon
- 6 keine Ahnung ...

## Fragen...

Welche Sprache generiert die Grammatik  
 $S \rightarrow aSb \mid C, C \rightarrow cC \mid \lambda$  ?

- 1  $a^n b^n c^n$
- 2  $a^n c^n b^n$
- 3  $a^n b^n c^m$
- 4  $a^n c^m b^n$
- 5 keine davon
- 6 keine Ahnung ...

## Zur Nachbereitung

### Zur Nachbereitung

Richtige Antworten sind:

- 1 2
- 2 4

## Zusammenfassung und Ausblick

### Typische Fragestellung(en)

- Wie bei Automaten ist es oft nötig, zu einer gegebenen Sprache  $M$  eine Grammatik  $G$  zu konstruieren, die  $M$  generiert, für die also  $L(G) = M$  gilt. Dies ist dann wieder zu beweisen!
- Zwei Techniken zur Grammatikenkonstruktion haben wir oben kennengelernt.
- Eine weitere wichtige Fragestellung (insb. im Rahmen des einführenden Beispiels) ist zu einem Wort  $w$  zu prüfen, ob  $w \in L(G)$  ist. Dies ist bei Automaten einfach (warum?), bei Grammatiken zunächst nicht so offensichtlich.

## 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:

$$A \rightarrow bcX, A \rightarrow bY, A \rightarrow a$$

## 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  $[z] \rightarrow a[z']$  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:

$$V_N := \{[z] \mid z \in Z\}$$

$$V_T := \Sigma$$

$$S := [z_0]$$

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

□

## Automat → Grammatik

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

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

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

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

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

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

## Grammatik → 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

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

Man kann wie eben wieder einen Zusammenhang zwischen Erfolgsrechnung und Ableitung herstellen [mündlich] und hat dann  $L(G) = L(A_2)$ .  $\square$

## Grammatik → Automat

Hinweis

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

Wer sich hier (zu recht) daran stört, dass mehr als ein Buchstabe an der Kante steht, der kann noch Zwischenzustände für die einzelnen Buchstaben des Wortes  $u$  einführen!

## Die Sprachfamilie REG

Satz

Zu jeder regulären Sprache  $L$  gibt es

- einen DFA  $A$  mit  $L(A) = L$
- einen NFA  $B$  mit  $L(B) = L$
- einen NFA mit  $\lambda$ -Kanten  $C$  mit  $L(C) = L$
- einen regulären Ausdruck  $D$ , der  $L$  beschreibt ( $M_D = L$ )
- eine rechtslineare Grammatik  $G$  gibt mit  $L(G) = L$

DFAs, NFAs, NFAs mit  $\lambda$ -Kanten, reguläre Ausdrücke und rechtslineare Grammatiken sind also äquivalent.

Wichtige Bemerkung

Insbesondere haben wir damit eine Brücke zwischen Grammatiken und Automaten geschlagen!

## Die Sprachfamilie CF

### 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!

## Zusammenfassung

Unsere bisherigen Sprachfamilien:

- REG (mit DFA, NFA,  $\lambda$ -NFA, regulären Ausdrücken und rechtslinearen Grammatiken)
- CF (mit PDA und kontextfreien Grammatiken)

### Wichtige Bemerkung

Die Sprachfamilie CF ist echt größer als die Sprachfamilie REG, da jede reguläre Sprache auch eine kontextfreie Sprache ist (betrachte einen DFA als eingeschränkten PDA oder eine rechtslineare Grammatik als eingeschränkte kontextfreie Grammatik) und es Sprachen gibt, die in CF sind, aber nicht in REG (z.B.  $\{a^n b^n \mid n \in \mathbb{N}\}$ ).

## Abschlusseigenschaften

### Definition

Sei  $f_1$  eine einstellige Operation auf Mengen und  $f_2$  eine zweistellige Operationen. D.h. wenn  $M_1, M_2$  zwei Mengen sind, dann sind auch  $f_1(M_1)$  und  $f_2(M_1, M_2)$  Mengen.

- Eine Sprachfamilie  $\mathcal{C}$  ist **abgeschlossen** gegenüber der Operation  $f_1$  bzw.  $f_2$ , wenn für jedes  $R \in \mathcal{C}$  auch  $f_1(R) \in \mathcal{C}$  gilt bzw. wenn für  $R_1, R_2 \in \mathcal{C}$  auch  $f_2(R_1, R_2) \in \mathcal{C}$  gilt.

## 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  $P_3 := P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 \mid S_2\}$ , 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  $P_4 := P_1 \cup P_2 \cup \{S_4 \rightarrow S_1 S_2\}$ , 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  $P_5 := P_1 \cup \{S_5 \rightarrow S_1 S_5 \mid \lambda\}$ , dann ist  $L_1^* = L(G_5)$ .

□

### 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 ...

## Die Chomsky-Normalform

Wir wollen nun eine **Normalform** für kontextfreie Grammatiken herstellen. Diese hilft

- bei Beweisen und
- einigen Algorithmen.

Wir machen dies primär, um ein Gefühl für das Arbeiten mit den Grammatiken zu entwickeln.

## Abschlusseigenschaften kontextfreier Sprachen

### Satz

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

### Beweis.

Der Beweis gelingt stets durch Angabe eines Gegenbeispiels.  
Z.B. für  $\cap$ : Es ist

- 1  $L_1 := \{a^n b^n c^m \mid n, m \in \mathbb{N}\} \in CF$
- 2  $L_2 := \{a^i b^k c^k \mid i, k \in \mathbb{N}\} \in CF$
- 3  $L_1 \cap L_2 = \{a^r b^r c^r \mid r \in \mathbb{N}\}$

Wir sehen nachher noch, dass  $L_1 \cap L_2$  nicht kontextfrei ist. Da  $L_1$  und  $L_2$  kontextfrei sind, kann CF also nicht gegen  $\cap$  abgeschlossen sein. Die anderen folgen schnell mit de Morgan.  $\square$

## 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  $S' \rightarrow \lambda$  die einzige  $\lambda$ -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  $\lambda$ -frei machen
- 2 Kettenregeln entfernen
- 3 Reduzieren (produktiv und erreichbar)
- 4 Ersetzen langer Terminalregeln
- 5 Verkürzen zu langer Regeln
- 6 Evtl. Hinzunahme einer  $\lambda$ -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. $\lambda$ -frei machen

- 1 Zu jedem  $A \in V_N$  bestimmen wir, ob  $A \xRightarrow{*} \lambda$  gilt:

$$M_0 := \{A \in V_N \mid A \rightarrow \lambda \in P\}$$

$$M_{i+1} := M_i \cup \{A \in V_N \mid \exists w \in M_i^* : A \rightarrow w \in P\}$$

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

- 2 Wir definieren eine Substitution  $\sigma : V^* \rightarrow 2^{V^*}$  durch

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

$$\sigma(A) := \{\lambda, A\}, \text{ falls } A \in V_\lambda$$

- 3  $G_1 := (V_{N_1}, V_{T_1}, P_1, S)$  mit  $V_{N_1} = V_N, V_{T_1} = V_T$  und

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

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

## 1. $\lambda$ -frei machen

### Beispiel

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

$$M_0 = \{C\}$$

$$M_1 = \{C, A\}$$

$$M_2 = M_1$$

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

$$\sigma(xABCy) = \{xABCy, xBCy, xAB y, xB y\}$$

$$\sigma(xACA) = \{xACA, xCA, xAA, xAC, xA, xC, x\}$$

## 1. $\lambda$ -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  $\lambda$ -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  $S_{neu} \rightarrow \lambda$  sowie  $S_{neu} \rightarrow w$  für jede Produktion  $S \rightarrow w \in P_1$ .

## 2. Kettenregeln entfernen

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

Wir definieren

$$A \ll B \text{ gdw. } A \rightarrow B \in P_1$$

Mit  $\ll^*$  sei die reflexive, transitive Hülle von  $\ll$  bezeichnet. Es ist nun  $G_2 = (V_{N_2}, V_{T_2}, P_2, S_2) = (V_{N_1}, V_{T_1}, P_2, S)$  mit

$$P_2 := \{A \rightarrow w \mid w \notin V_{N_2} \wedge \exists B \rightarrow w \in P_1 \wedge A \ll^* B\}$$

Es gilt  $L(G_2) = L(G_1)$ .

## 2. Kettenregeln entfernen

### Beispiel

$$P_2 := \{A \rightarrow w \mid w \notin V_{N_2} \wedge \exists B \rightarrow w \in P_1 \wedge A \ll^* B\}$$

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

$$A \ll^* B, B \ll^* C, A \ll^* C$$

(und  $X \ll^* X$  für jedes Nonterminal  $X$ ).

Damit folgt als Produktionen in  $P_2$ :

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

## 3. Reduzieren

Der 3. Schritt besteht aus zwei Teilschritten. **Zuerst** alle nicht produktiven Nonterminale entfernen. Produktiv ist ein Nonterminal dabei, wenn es zu einem Terminalwort abgeleitet werden kann. Wir berechnen ähnlich wie im 1. Schritt

$$M_0 := V_T$$

$$M_{i+1} := M_i \cup \{A \in V_{N_2} \mid \exists w \in M_i^* : A \rightarrow w \in P_2\}$$

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

$$G' := (V'_N, V'_T, P', S') = ((M_k \cap V_{N_2}) \cup \{S\}, V_{T_2}, P_2 \cap (M_k \times M_k^*), S)$$

### 3. Reduzieren

Als nächstes werden die erreichbaren Nonterminale bestimmt.

$$M_0 := \{S'\}$$

$$M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : A \rightarrow uBv \in P'\}$$

Wieder gibt es ein  $k$  mit  $M_{k+1} = M_k$ . Wir erhalten dann  $G_3 := (V_{N_3}, V_{T_3}, P_3, S)$ , 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, um  $V_{N_3}$  zu erhalten.

Es gilt  $L(G_3) = L(G_2)$ .

### 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 ersetzen wir auf diesen rechten Seiten jedes Terminal  $a$  durch ein Nonterminal  $\langle a \rangle$ .
- Und fügen Regeln  $\langle a \rangle \rightarrow a$  ein.

Sei die so entstandene Grammatik mit  $G_4$  bezeichnet. Es gilt  $L(G_4) = L(G_3)$ .

### 3. 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 ;-)

### 4. Ersetzen langer Terminalregeln

#### Beispiel

Hat man bspw. Regeln

$$A \rightarrow bc \mid Bc \mid c$$

so entstehen daraus die Regeln

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

und

$$\langle b \rangle \rightarrow b$$

$$\langle c \rangle \rightarrow c$$

## 5. Verkürzen zu langer Regeln

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

$$G_5 := (V_{N_5}, V_{T_3}, P_5, S) \text{ mit}$$

$$V_{N_5} := \{\langle v \rangle \mid \exists u \neq \lambda : \exists A \rightarrow w \in P_4 : w = vu \wedge |v| \geq 2\} \cup V_{N_4}$$

$$P_5 := \{A \rightarrow \langle v \rangle x \mid A \rightarrow w \in P_4 : |w| \geq 3 \wedge w = vx \wedge x \in V_{N_4}\} \cup$$

$$\{\langle v \rangle \rightarrow \langle u \rangle y \mid \langle u \rangle, \langle v \rangle \in (V_{N_5} \setminus V_{N_4}) \wedge y \in V_{N_4} \wedge v = uy\} \cup$$

$$\{\langle v \rangle \rightarrow xy \mid \langle v \rangle \in (V_{N_5} \setminus V_{N_4}) \wedge x, y \in V_{N_4} \wedge v = xy\} \cup$$

$$\{A \rightarrow w \mid A \rightarrow w \in P_4 \wedge |w| \leq 2\}$$

Es gilt  $L(G_5) = L(G_4)$ .

## 6. $\lambda$ -Regel hinzunehmen

Galt im ersten Schritt  $S \in V_\lambda$ , war also  $\lambda \in L(G)$ , so verfahren wir wie folgt:

- Tritt  $S$  auf keiner rechten Seite auf, so fügen wir zu der zuletzt konstruierten Grammatik lediglich  $S \rightarrow \lambda$  hinzu.
- Tritt  $S$  auf einer rechten Seite auf, so fügen wir zu der zuletzt konstruierten Grammatik noch ein neues Startsymbol  $S_{neu}$  hinzu sowie die neuen Produktionen  $S_{neu} \rightarrow \lambda$  sowie  $S_{neu} \rightarrow w$  für jede Produktion  $S \rightarrow w \in P_5$ .

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

## 5. Verkürzen zu langer Regeln

### Beispiel

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

$$A \rightarrow (BCD)E$$

$$(BCD) \rightarrow (BC)D$$

$$(BC) \rightarrow BC$$

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.

## 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  $n := 2^{k+1}$ . 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 in z.B. [HMU].

## 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 - 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. (Kommt evtl. später noch.)

## Zusammenfassung

Definitionen gestern und heute:

- Grammatik
  - Nonterminal, Terminal, Produktion, Startsymbol
  - Ableitung
  - (Generierte) Sprache, Äquivalenz
- Kontextfreie Grammatik / Typ-2-Grammatik
  - $\lambda$ -Produktionen,  $\lambda$ -frei
- Sprachfamilie CF
- rechtslineare Grammatik

## Zusammenfassung

Wir haben heute

- Konstruktionstechniken für Grammatiken kennengelernt
- Zusammenhänge zwischen Automaten und Grammatiken gesehen:
  - Rechtslineare Grammatiken  $\equiv$  DFA
  - Kontextfreie Grammatiken  $\equiv$  PDA
- 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 3)
- Das zweite Pumping Lemma kennengelernt und damit
  - gesehen, dass es Sprachen gibt, die nicht in CF sind
  - woraus aus vorherigem folgt, dass CF nicht gegenüber Schnitt abgeschlossen ist