

# Formale Sprachen

## Script, Kapitel 4

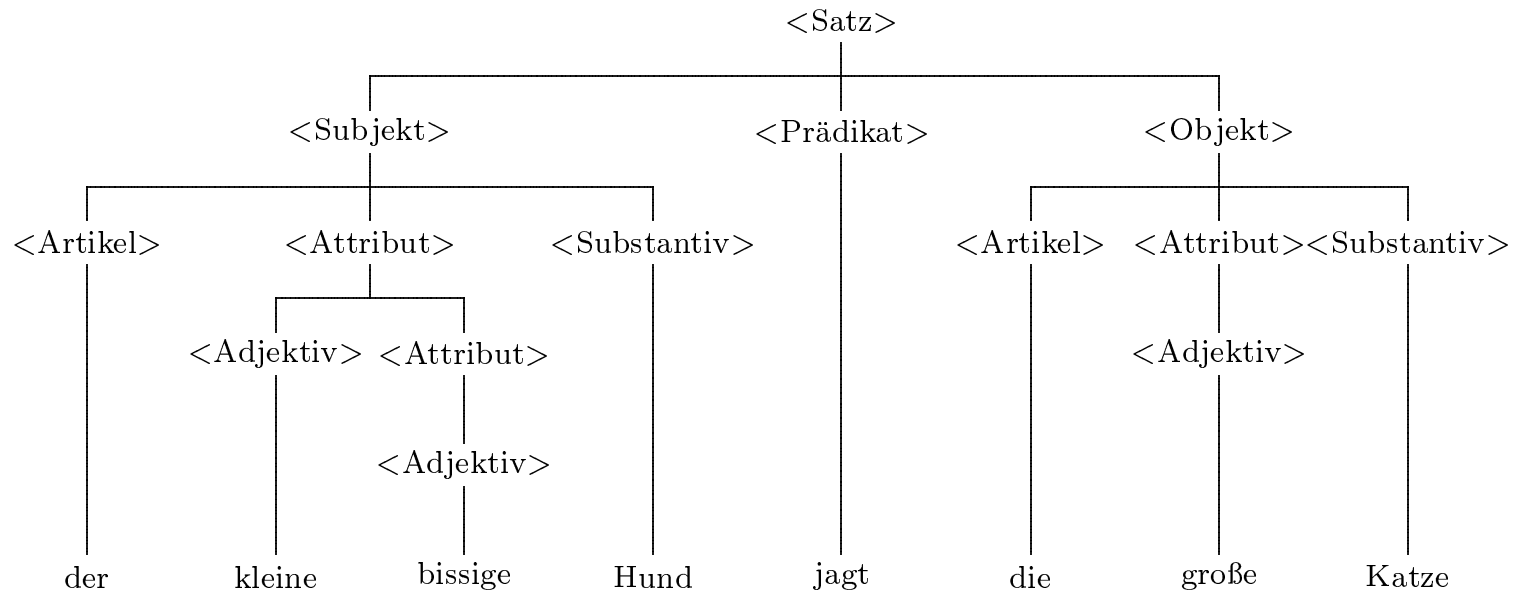
- **Grammatiken**
  - erzeugen Sprachen
  - eingeführt von Chomsky zur Beschreibung natürlicher Sprache
  - bedeutend für die Syntaxdefinition und -analyse von Programmiersprachen
- **Automaten**
  - akzeptieren Sprachen
  - enge Beziehungen zu Grammatiken

# Beispiel für eine Grammatik

$\langle \text{Satz} \rangle \rightarrow \langle \text{Subjekt} \rangle \langle \text{Prädikat} \rangle \langle \text{Objekt} \rangle$   
 $\langle \text{Subjekt} \rangle \rightarrow \langle \text{Artikel} \rangle \langle \text{Attribut} \rangle \langle \text{Substantiv} \rangle$   
 $\langle \text{Prädikat} \rangle \rightarrow \text{jagt}$   
 $\langle \text{Objekt} \rangle \rightarrow \langle \text{Artikel} \rangle \langle \text{Attribut} \rangle \langle \text{Substantiv} \rangle$   
 $\langle \text{Artikel} \rangle \rightarrow \varepsilon \mid \text{der} \mid \text{die} \mid \text{das}$   
 $\langle \text{Attribut} \rangle \rightarrow \varepsilon \mid \langle \text{Adjektiv} \rangle \mid \langle \text{Adjektiv} \rangle \langle \text{Attribut} \rangle$   
 $\langle \text{Adjektiv} \rangle \rightarrow \text{kleine} \mid \text{bissige} \mid \text{große}$   
 $\langle \text{Substantiv} \rangle \rightarrow \text{Hund} \mid \text{Katze}$

# Syntaxbaum für den Satz

*der kleine bissige Hund jagt die große Katze*



## Definition Grammatik

Eine **Grammatik** ist ein 4-Tupel  $G = (V, \Sigma, P, S)$ , wobei

- $V$  ein Alphabet ist (**Nichtterminalalphabet** oder Alphabet der **Variablen**),
- $\Sigma$  ein Alphabet ist (**Terminalalphabet**),
- $V \cap \Sigma = \emptyset$  gilt,
- $P$  eine **endliche** Teilmenge von  $((V \cup \Sigma)^* \setminus \Sigma^*) \times (V \cup \Sigma)^*$  ist (Menge der **Regeln**),
- $S \in V$  ist (die **Startvariable** oder **Axiom**).

Zur besseren Lesbarkeit werden wir  $u \rightarrow v \in P$  für  $(u, v) \in P$  schreiben.

# Definitionen direkte Ableitung einer Grammatik

Sei  $G = (V, \Sigma, P, S)$  eine Grammatik und  $u, v \in (V \cup \Sigma)^*$  Wörter. Dann gilt  $u \Longrightarrow_G v$  (in Worten:  $u$  erzeugt bezüglich  $G$  direkt  $v$ ) genau dann, wenn

- (i)  $u = \gamma_1 \alpha \gamma_2$  mit  $\gamma_1, \gamma_2 \in (V \cup \Sigma)^*$ ,
- (ii)  $v = \gamma_1 \beta \gamma_2$  und
- (iii)  $\alpha \rightarrow \beta \in P$  ist.

Wenn keine Verwechslungsgefahr besteht, schreiben wir statt " $\Longrightarrow_G$ " einfach " $\Longrightarrow$ ".

# Definitionen Ableitung und erzeugte Sprache

Sei  $G = (V, \Sigma, P, S)$  eine Grammatik und  $u, v \in (V \cup \Sigma)^*$  Wörter. Dann gilt  $u \xRightarrow{*}_G v$  (in Worten:  $u$  erzeugt bezüglich  $G$  in endlich vielen Schritten  $v$ ) genau dann, wenn  $u = v$  gilt oder es ein  $n \in \mathbb{N}$  und Wörter  $w_0, w_1, \dots, w_n$  gibt, so dass

$$u = w_0 \Longrightarrow_G w_1 \Longrightarrow_G w_2 \Longrightarrow_G \cdots \Longrightarrow_G w_n = v$$

gilt.

Sei  $G = (V, \Sigma, P, S)$  eine Grammatik. Die von  $G$  erzeugte Sprache  $L(G)$  wird definiert als

$$L(G) = \{w \in \Sigma^* \mid S \xRightarrow{*}_G w\}.$$

## Beispiel einer Grammatik

Es sei

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow ab\}, S)$$

eine Grammatik, dann gilt

$$L(G) = \{a^n b^n \mid n \geq 1\}.$$

Eine Ableitung für das Wort  $a^4 b^4$  sieht dann so aus:

$$S \Longrightarrow aSb \Longrightarrow aaSbb \Longrightarrow aaaSbbb \Longrightarrow aaaabbbb.$$

## Weiteres Beispiel einer Grammatik

Es sei

$$G = (\{S\}, \{a\}, \{S \rightarrow aaS, S \rightarrow a\}, S)$$

eine Grammatik, dann gilt

$$L(G) = \{a^{2n+1} \mid n \geq 0\}.$$

Eine Ableitung für das Wort  $a^7$  sieht dann so aus:

$$S \Longrightarrow aaS \Longrightarrow aaaaS \Longrightarrow aaaaaaS \Longrightarrow aaaaaaa.$$



## Weiteres Beispiel einer Grammatik

Es sei

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aS, S \rightarrow bS, S \rightarrow a, S \rightarrow b\}, S)$$

eine Grammatik, dann gilt

$$L(G) = \{a, b\}^+ = \{w \in \{a, b\}^* \mid w \neq \varepsilon\}.$$

Eine Ableitung für das Wort  $aaba$  sieht dann so aus:

$$S \Longrightarrow aS \Longrightarrow aaS \Longrightarrow aabS \Longrightarrow aaba.$$

## Komplexeres Beispiel einer Grammatik

Es sei die Grammatik  $G = (\{S, B, C\}, \{a, b, c\}, P, S)$  mit

$$P = \{S \rightarrow aSBC, S \rightarrow aBC, CB \rightarrow BC, aB \rightarrow ab, \\ bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc\}$$

gegeben. Wir können zum Beispiel die Ableitung

$$\begin{aligned} S &\Longrightarrow aSBC \Longrightarrow aaSBCBC \Longrightarrow aaaBCBCBC \\ &\Longrightarrow aaaBBCCBC \Longrightarrow aaaBBCBCC \Longrightarrow aaaBBBCCC \\ &\Longrightarrow aaabBBCCC \Longrightarrow aaabbBCCC \Longrightarrow aaabbbCCC \\ &\Longrightarrow aaabbbcCC \Longrightarrow aaabbbccC \Longrightarrow aaabbbccc \end{aligned}$$

aufstellen, also gehört das Wort  $aaabbbccc = a^3b^3c^3$  zur erzeugten Sprache  $L(G)$ , es gilt also  $a^3b^3c^3 \in L(G)$ .

## Komplexeres Beispiel einer Grammatik – Erzeugte Sprache Teil 1

**Vermutung:**  $L(G) = \{a^n b^n c^n \mid n \geq 1\}$ .

Zunächst wird  $L(G) \supseteq \{a^n b^n c^n \mid n \geq 1\}$  gezeigt,  
d.h.  $S \xRightarrow{*} a^n b^n c^n$  für jedes  $n \geq 1$ .

- Wende  $(n - 1)$ -mal Regel  $S \rightarrow aSBC$  und dann einmal  $S \rightarrow aBC$  an,  
d.h.:  $S \xRightarrow{*} a^n (BC)^n$ .
- Solange wie möglich wende  $CB \rightarrow BC$  an,  
d.h.:  $a^n (BC)^n \xRightarrow{*} a^n B^n C^n$ .
- Wende einmal  $aB \rightarrow ab$  und  $(n - 1)$ -mal  $bB \rightarrow bb$  an,  
d.h.:  $a^n B^n C^n \xRightarrow{*} a^n b^n C^n$ .
- Wende einmal  $bC \rightarrow bc$  und  $(n - 1)$ -mal  $cC \rightarrow cc$  an,  
d.h.:  $a^n b^n C^n \xRightarrow{*} a^n b^n c^n$ .

## Komplexeres Beispiel einer Grammatik – Erzeugte Sprache Teil 2

Schwieriger zu zeigen ist die Behauptung  $L(G) \subseteq \{a^n b^n c^n \mid n \geq 1\}$ .

- Für jedes erzeugbare Wort  $\alpha$  gilt:  $|\alpha|_a = |\alpha|_b + |\alpha|_B = |\alpha|_c + |\alpha|_C$ .
- In jedem erzeugbaren Wort stehen die  $a$ 's am Anfang.
- Ein Symbol  $B$  kann nur dann in ein  $b$  umgewandelt werden, wenn unmittelbar links vor ihm ein  $a$  oder ein  $b$  steht, d.h. kein  $c$  steht vor einem  $b$ .
- Mathematisch exakter Beweis erfolgt durch **vollständige Induktion**.

# Chomsky-Hierarchie

**Definition:** Eine **Grammatik**  $G = (V, \Sigma, P, S)$  heißt vom

- **Typ 0**, wenn sie keinen Beschränkungen unterliegt,
- **Typ 1** oder **kontextabhängig**, falls für jede Regel  $\alpha \rightarrow \beta$  gilt:  $|\alpha| \leq |\beta|$ , mit der Ausnahme  $S \rightarrow \varepsilon$ , falls  $S$  nicht auf der rechten Seite einer Regel vorkommt.
- **Typ 2** oder **kontextfrei**, wenn jede Regel von der Form  $A \rightarrow \beta$  mit  $A \in V$  und  $\beta \in (V \cup \Sigma)^*$  ist.
- **Typ 3** oder **regulär**, wenn jede Regel von der Form  $A \rightarrow wB$  oder  $A \rightarrow w$  mit  $A, B \in V$  und  $w \in \Sigma^*$  ist.

# Chomsky-Hierarchie – Fortsetzung

**Definition:** Eine **Sprache**  $L \subseteq \Sigma^*$  heißt vom Typ 0 (Typ 1, Typ 2, Typ 3), falls es eine Grammatik  $G = (V, \Sigma, P, S)$  vom Typ 0 (Typ 1, Typ 2, Typ 3) gibt, so dass  $L = L(G)$  gilt.

**Notation: Typ  $i$ :** Familie der **Sprachen** vom Typ  $i \in \{0, 1, 2, 3\}$

**Satz (Chomsky-Hierarchie).** Es gilt:

$$\text{Typ 3} \subsetneq \text{Typ 2} \subsetneq \text{Typ 1} \subsetneq \text{Typ 0.}$$

Der Beweis des Satzes wird in den folgenden Kapiteln erbracht.

# Typ-0-Grammatiken und Turingmaschinen

## Satz

Eine Sprache ist genau dann eine Typ-0-Sprache, wenn sie von einer NTM akzeptiert werden kann, d.h. wenn sie **rekursiv aufzählbar** ist.

## Beweisidee

- Ableitung einer Grammatik wird in umgekehrter Reihenfolge durch eine NTM simuliert.
- Konfigurationenfolge eines akzeptierenden Laufs einer NTM wird in umgekehrter Reihenfolge durch eine Grammatik erzeugt.

# Simulation einer Grammatik durch eine NTM

- gegeben: Grammatik  $G = (V, \Sigma, P, S)$
- Konstruiere NTM  $M = (Z, \Sigma, V \cup \Sigma \cup \{\square\}, \delta, z_0, \square, \{q\})$
- Arbeitsweise von  $M$  in einer *Phase*:  
Rate eine Regel  $\alpha \rightarrow \beta$  und ersetze ein Vorkommen von  $\beta$  durch  $\alpha$ ,  
d.h. Konfigurationsänderung:  $z_0 w_1 \beta w_2 \vdash^* z_0 w_1 \alpha w_2$
- $M$  akzeptiert, wenn nur noch  $S$  auf dem Band steht.



# Linear beschränkte Automaten

**Definition** Eine nichtdeterministische Turingmaschine  $M$  heißt **linear beschränkter Automat (LBA)**, wenn bei jedem Lauf von  $M$  nur die Speicherzellen der Eingabe benutzt werden.

**Offenes Problem (LBA-Problem):**

Sind **deterministische** LBA so mächtig wie nichtdeterministische LBA?

# Typ-1-Grammatiken und LBA

## Satz

Eine Sprache ist genau dann eine Typ-1-Sprache, wenn sie von einem linear beschränkten Automaten akzeptiert werden kann.

## Beweisidee

- Gleiche Konstruktionen wie für Typ-0-Grammatiken und NTM.
- Nichtverkürzende Regeln erlauben den beschränkten Platzbedarf.

# Das Wortproblem

## Definition (**Wortproblem**)

**Gegeben:** Grammatik  $G = (V, \Sigma, P, S)$  vom Typ  $i$ ,  $i \in \{0, 1, 2, 3\}$ ,  
und Wort  $w \in \Sigma^*$ ,

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

**Folgerung** Das Wortproblem für Typ-0-Grammatiken ist semi-entscheidbar, aber nicht entscheidbar.

**Satz** Das Wortproblem für Typ-1-Grammatiken ist entscheidbar.

# Weitere Entscheidungsprobleme

Leerheitsproblem:

**Gegeben:** Grammatik  $G$ .

**Frage:** Gilt  $L(G) = \emptyset$ ?

Endlichkeitsproblem:

**Gegeben:** Grammatik  $G$ .

**Frage:** Ist  $L(G)$  endlich?

Schnittproblem:

**Gegeben:** Zwei Grammatiken  $G_1, G_2$ .

**Frage:** Gilt  $L(G_1) \cap L(G_2) = \emptyset$ ?

Äquivalenzproblem:

**Gegeben:** Zwei Grammatiken  $G_1, G_2$ .

**Frage:** Gilt  $L(G_1) = L(G_2)$ ?

**Satz:** Das Leerheitsproblem, das Endlichkeitsproblem, das Äquivalenzproblem und das Schnittproblem sind unentscheidbar für Typ-1-Grammatiken.