

### Bemerkung:

Das uniforme wie auch das nicht-uniforme Wortproblem ist für Typ-0-Sprachen (also die rekursiv-aufzählbare Sprachen) im Allgemeinen nicht entscheidbar. Wir werden später sehen, dass es zum **Halteproblem für Turingmaschinen** äquivalent ist.

Es gilt jedoch

### Satz 20

*Für kontextsensitive Grammatiken ist das Wortproblem entscheidbar.*

*Genauer: Es gibt einen Algorithmus, der bei Eingabe einer kontextsensitiven Grammatik  $G = (V, \Sigma, P, S)$  und eines Wortes  $w$  in endlicher Zeit entscheidet, ob  $w \in L(G)$ .*

### Beweisidee:

Angenommen  $w \in L(G)$ . Dann gibt es eine Ableitung

$$S = w^{(0)} \rightarrow_G w^{(1)} \rightarrow_G \cdots \rightarrow_G w^{(\ell)} = w$$

mit  $w^{(i)} \in (\Sigma \cup V)^*$  für  $i = 1, \dots, \ell$ .

Da aber  $G$  kontextsensitiv ist, gilt (falls  $w \neq \epsilon$ )

$$|w^{(0)}| \leq |w^{(1)}| \leq \cdots \leq |w^{(\ell)}| ,$$

d.h., es genügt, nur Wörter in  $(\Sigma \cup V)^*$  der Länge  $\leq |w|$  zu betrachten.

## Beweis:

Sei o.B.d.A.  $w \neq \epsilon$  und sei  $T_m^n := \{w' \in (\Sigma \cup V)^*; |w'| \leq n \text{ und } w' \text{ lässt sich aus } S \text{ in } \leq m \text{ Schritten ableiten}\}$

Diese Mengen kann man für alle  $n$  und  $m$  induktiv wie folgt berechnen:

$$\begin{aligned} T_0^n &:= \{S\} \\ T_{m+1}^n &:= T_m^n \cup \{w' \in (\Sigma \cup V)^*; |w'| \leq n \text{ und } w'' \rightarrow w' \text{ für ein } w'' \in T_m^n\} \end{aligned}$$

**Beachte:** Für alle  $m$  gilt:  $|T_m^n| \leq \sum_{i=1}^n |\Sigma \cup V|^i$ .

Es muss daher, für festes  $n$ , immer ein  $m_0$  geben mit

$$T_{m_0}^n = T_{m_0+1}^n = \dots$$

Beweis (Forts.):

Algorithmus:

$n := |w|$

$T := \{S\}$

$T' := \emptyset$

**while**  $T \neq T'$  **do**

$T' := T$

$T := T' \cup \{w' \in (V \cup \Sigma)^+; |w'| \leq n, (\exists w'' \in T')[w'' \rightarrow w']\}$

**od**

**if**  $w \in T$  **return** „ja“ **else return** „nein“ **fi**



## Beispiel 21

Gegeben sei die Typ-2-Grammatik mit den Produktionen

$$S \rightarrow ab \text{ und } S \rightarrow aSb$$

sowie das Wort  $w = abab$ .

$$T_0^4 = \{S\}$$

$$T_1^4 = \{S, ab, aSb\}$$

$$T_2^4 = \{S, ab, aSb, aabb\} \quad aaSbb \text{ ist zu lang!}$$

$$T_3^4 = \{S, ab, aSb, aabb\}$$

Also lässt sich das Wort  $w$  mit der gegebenen Grammatik **nicht** erzeugen!

**Bemerkung:**

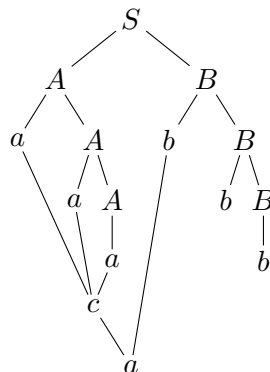
Der angegebene Algorithmus ist nicht sehr effizient! Für **kontextfreie** Grammatiken gibt es wesentlich effizientere Verfahren, die wir später kennenlernen werden!

## 2.4 Ableitungsgraph und Ableitungsbaum

Grammatik:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow a \\ B &\rightarrow bB \\ B &\rightarrow b \\ aaa &\rightarrow c \\ cb &\rightarrow a \end{aligned}$$

Beispiel:

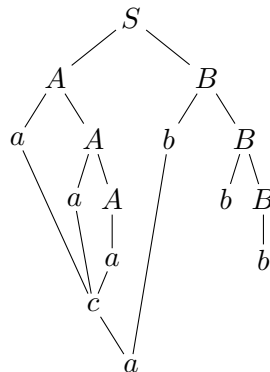


Die Symbole ohne Kante nach unten entsprechen, von links nach rechts gelesen, dem durch den Ableitungsgraphen dargestellten Wort.

## Grammatik:

$S \rightarrow AB$   
 $A \rightarrow aA$   
 $A \rightarrow a$   
 $B \rightarrow bB$   
 $B \rightarrow b$   
 $aaa \rightarrow c$   
 $cb \rightarrow a$

## Beispiel:



Dem Ableitungsgraph entspricht z.B. die Ableitung

$S \rightarrow AB \rightarrow aAB \rightarrow aAbB \rightarrow aaAbB \rightarrow aaAbbB \rightarrow$   
 $\rightarrow aaabbB \rightarrow aaabbb \rightarrow cbbb \rightarrow abb$



## Beobachtung:

Bei kontextfreien Sprachen sind die Ableitungsgraphen immer Bäume.

## Beispiel 22

Grammatik:

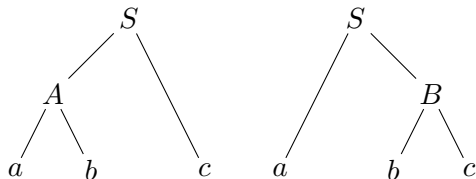
$$S \rightarrow aB$$

$$S \rightarrow Ac$$

$$A \rightarrow ab$$

$$B \rightarrow bc$$

AbleitungsbaumAbleitungsbäume:



Für das Wort  $abc$  gibt es zwei verschiedene Ableitungsbäume.

## Definition 23

- Eine Ableitung

$$S = w^{(0)} \rightarrow w^{(1)} \rightarrow \dots \rightarrow w^{(n)} = w$$

eines Wortes  $w$  heißt **Linksableitung**, wenn für jede Anwendung einer Produktion  $\alpha \rightarrow \beta$  auf  $w^{(i)} = x\alpha z$  gilt, dass sich **keine** Regel der Grammatik auf ein echtes Präfix von  $x\alpha$  anwenden lässt.

- Eine Grammatik heißt **eindeutig**, wenn es für jedes Wort  $w \in L(G)$  genau eine Linksableitung gibt. Nicht eindeutige Grammatiken nennt man auch **mehrdeutig**.
- Eine Sprache  $L$  heißt **eindeutig**, wenn es für  $L$  eine eindeutige Grammatik gibt. Ansonsten heißt  $L$  mehrdeutig.

**Bemerkung:** Eindeutigkeit wird meist für kontextfreie (und reguläre) Grammatiken betrachtet, ist aber allgemeiner definiert.

## Beispiel 24

Grammatik:

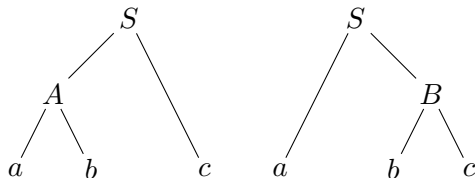
$$S \rightarrow aB$$

$$S \rightarrow Ac$$

$$A \rightarrow ab$$

$$B \rightarrow bc$$

Ableitungsbäume:



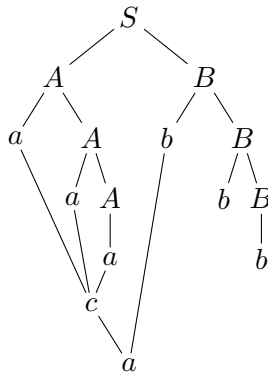
Beide Ableitungsbäume für das Wort  $abc$  entsprechen Linksableitungen.

## Beispiel 25

Grammatik:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow a \\ B &\rightarrow bB \\ B &\rightarrow b \\ aaa &\rightarrow c \\ cb &\rightarrow a \end{aligned}$$

Ableitung:



Eine Linksableitung ist

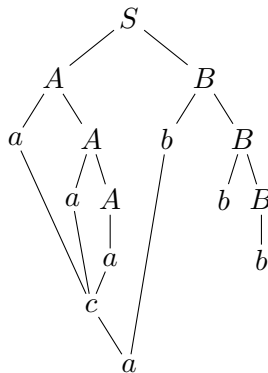
$$\begin{aligned} S &\rightarrow AB \rightarrow aAB \rightarrow aaAB \rightarrow aaaB \rightarrow cB \rightarrow \\ &\rightarrow cbB \rightarrow aB \rightarrow abB \rightarrow abb \end{aligned}$$

## Beispiel 25

Grammatik:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow a \\ B &\rightarrow bB \\ B &\rightarrow b \\ aaa &\rightarrow c \\ cb &\rightarrow a \end{aligned}$$

Ableitung:



Eine andere Linksableitung für  $abb$  ist

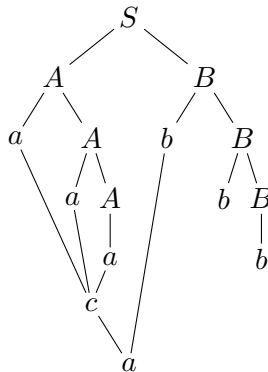
$$S \rightarrow AB \rightarrow aB \rightarrow abB \rightarrow abb .$$

## Beispiel 25

Grammatik:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow a \\ B &\rightarrow bB \\ B &\rightarrow b \\ aaa &\rightarrow c \\ cb &\rightarrow a \end{aligned}$$

Ableitung:



Die Grammatik ist also **mehrdeutig**.

## 3. Reguläre Sprachen

### 3.1 Deterministische endliche Automaten

#### Definition 26

Ein **deterministischer endlicher Automat** (englisch: deterministic finite automaton, kurz DFA) wird durch ein 5-Tupel  $M = (Q, \Sigma, \delta, q_0, F)$  beschrieben, das folgende Bedingungen erfüllt:

- 1  $Q$  ist eine endliche Menge von **Zuständen**.
- 2  $\Sigma$  ist eine endliche Menge, das **Eingabealphabet**, wobei  $Q \cap \Sigma = \emptyset$ .
- 3  $q_0 \in Q$  ist der **Startzustand**.
- 4  $F \subseteq Q$  ist die Menge der **Endzustände** (oder auch **akzeptierenden Zustände**)
- 5  $\delta : Q \times \Sigma \rightarrow Q$  heißt **Übergangsfunktion**.

Die von  $M$  akzeptierte/erkannte Sprache ist

$$L(M) := \{w \in \Sigma^*; \hat{\delta}(q_0, w) \in F\} ,$$

wobei  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$  induktiv definiert ist durch

$$\begin{aligned} \hat{\delta}(q, \epsilon) &= q && \text{für alle } q \in Q \\ \hat{\delta}(q, ax) &= \hat{\delta}(\delta(q, a), x) && \text{für alle } q \in Q, a \in \Sigma \\ &&& \text{und } x \in \Sigma^* \end{aligned}$$

**Bemerkung:** Endliche Automaten können durch (gerichtete und markierte) Zustandsgraphen veranschaulicht werden:

- Knoten  $\hat{=}$  Zuständen
- Kanten  $\hat{=}$  Übergängen
- genauer: eine mit  $a \in \Sigma$  markierte Kante  $(u, v)$  entspricht  $\delta(u, a) = v$

Der Anfangszustand wird durch einen Pfeil, Endzustände werden durch doppelte Kreise gekennzeichnet.



## Beispiel 27

Sei  $M = (Q, \Sigma, \delta, q_0, F)$ ,  
wobei

$$Q = \{q_0, q_1, q_2, q_3\}$$

$$\Sigma = \{a, b\}$$

$$F = \{q_3\}$$

$$\delta(q_0, a) = q_1$$

$$\delta(q_0, b) = q_3$$

$$\delta(q_1, a) = q_2$$

$$\delta(q_1, b) = q_0$$

$$\delta(q_2, a) = q_3$$

$$\delta(q_2, b) = q_1$$

$$\delta(q_3, a) = q_0$$

$$\delta(q_3, b) = q_2$$

