Eine mögliche Ableitung ist

$$S \Rightarrow \ldots \Rightarrow aaabbbccc = a^3b^3c^3$$

Allgemein ist die durch G erzeugte Sprache L:

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

Beweis: (i) In jedem Ableitungsschritt ist die Anzahl der "a" gleich der Anzahl der "b" (der B) und der "c" (der C)

(ii) Folge der Terminale: "a" steht immer ganz links; alle Regeln sind so aufgebaut, daß die "b" vor den "c" kommen.

Einordnung von Grammatik-Typen, Chomsky-Hierarchie

Beobachtung:

- Die Regeln der Grammatik aus Beispiel a) sind so aufgebaut, daß die Variable auf der linken Seite bedingungslos durch die rechte Seite ersetzt.
- Demgegenüber können die Regeln aus Beispiel b) nur angewendet werden, wenn die Variable auf der linken Seite in einem bestimmten Kontext steht: z.B. kann B nur ersetzt werden, wenn B rechts neben C, "a" oder "b" steht.

Allgemein: $uAv \to uxv$ nur anwendbar, wenn A <u>im Kontext von</u> u und v steht.

Man bezeichnet Grammatiken des ersten Typs als "kontextfrei", der zweite Typ ist ein Beispiel für "Kontextsensitive" Grammatiken.

Grammatiken werden allgemein nach Noam Chomsky in vier Klassen eingeteilt:

<u>Typ 0</u>: Die Regeln unterliegen keinerlei Einschränkungen (sog. Phrasenstrukturgrammatiken)

<u>Typ 1</u>: Kontextsensitive Grammatiken. Regeln der Form wie oben dargestellt.

Außerdem: die Länge der abgeleiteten Wörter $|w|$ nimmt <u>nicht</u> ab von Ableitungsschritt zu Ableitungsschritt. Für alle Regeln $w_1 \to w_2$ aus P gilt: $|w_1| \leq |w_2|$

<u>Typ 2</u>: Kontextfrei. Für alle Regeln $w_1 \to w_2$ aus P gilt, daß w_1 eine einzelne Variable ist, d.h. $w_1 \in V$

Typ 3: Regulär. Zusätzlich zu Typ 2 sind die rechten Seiten der Regeln $w_1 \to w_2$ entweder Terminalsymbol oder ein Terminalsymbol gefolgt von einer Variablen, also: $w_2 \in (A \cup AV)$

In der Informatik sind für die Anwendungen Syntaxanalyse, Übersetzerbau, Nutzerschnittstellen Typ 2 und Typ 3 vom größten Interesse.

Wichtige Fragestellungen:

a) Gehört ein Wort x zur von G erzeugten Sprache?

b) Durch welchen Grammatik-Typ kann eine vorgelegte Sprache erzeugt werden?

Fragestellung a) ist das sog. Wortproblem der Informatik. Man kann zeigen, daß man in endlicher Zeit entscheiden kann, ob $x \in L(G)$.

Satz: Das Wortproblem ist für Typ 1-Sprachen entscheidbar. Es gibt also einen Algorithmus, der bei Eingabe einer Typ 1-Grammatik G und eines Wortes $x \in A^*$ in endlicher Zeit entscheidet, ob $x \in L(G)$ oder $x \notin L(G)$.

Beweisidee: Es gibt nur endlich viele Wörter $(V \cup A^*)$ der Länge kleiner oder gleich n $\rightarrow$ 'systematisches Durchprobieren'

Beweis über die Konstruktion eines Algorithmus wie folgt:

(i) Definiere Wortmengen T_m^n; diese enthalten jeweils alle Wörter der Länge $\leq n$, die in $\leq m$ Schritten ableitbar sind.

(ii) Induktiver Aufbau der Mengen durch schrittweise Ableitungen:

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

$$T_{m+1}^n = \underset{\substack{\uparrow \\ \text{Ableitungsschritt}}}{AS_n}(T_m^n) \text{ mit}$$

$$AS(X) = X \cup \{w \in (V \cup A)^* \mid |w| \leq n \text{ und } w' \rightarrow w \text{ für ein } w' \in X\}$$

(iii) Bei festgehaltenem n ist, weil nur endlich viele Wörter der Länge $\leq n$ in $(A \cup V)^*$ enthalten sind, irgendwann der Punkt erreicht, daß zusätzliche Ableitungen keine zusätzlichen Wörter mehr produzieren, $\bigcup_{m \geq 0} T_m^n$ aber nicht mehr wächst, d.h.

$T_m^n = T_{m+1}^n = T_{m+2}^n = \ldots$

(iv) Falls x in $L(G)$ liegt, so muß x in T_m^n für ein (möglicherweise großes) m liegen. Daraus folgt der Algorithmus:

```
T ← {S}
REPEAT
  T1 ← T ;
  T ← AS(T1)
UNTIL (x ∈ T) OR (T = T1)
```

Wort x in $L(G)$

T ändert sich nicht mehr, also $x \notin L(G)$

b) Fragestellung b: Gehört eine vorgelegte Sprache einem bestimmten Sprachtyp an? Hierzu nur ein Beispiel: Kann eine Typ 3-Grammatik die Sprache $L_1(G) = \{a^n b^n \mid n \geq 1\}$ erzeugen: Wörter mit n a's gefolgt von der gleichen Zahl b's. Um zu beweisen, daß dies nicht der Fall ist benutzt man das sog. Pumping-Lemma (Beweis später):

Sei L eine Typ 3-Sprache. Dann gibt es eine Zahl $n \in \mathbb{N}$, so daß alle Wörter x oberhalb einer bestimmten Länge n (also $|x| \ge n$) zerlegbar sind wie folgt:

$x = u v w$ mit

(a) $|v| \ge 1$
(b) $|uv| \le n$
(c) $uv^i w \in L$ für alle $i = 0, 1, 2, \ldots$

Eine solche Grammatik kann also neben uvw die Wörter uw, uvvw, ... erzeugen.

Im Falle von $L_1 = \{a^n b^n \mid n \ge 1\}$ müßte es eine Zahl n geben, so daß sich die Wörter $|x| \ge n$ wie im Pumping-Lemma zerlegen lassen. Sei betrachtet ein Wort $a^n b^n$ der Länge $2n$. Dann ist v in der Zerlegung $x = uvw$ nicht leer, mit (b) kann v nur aus a's bestehen. Mit (c) wäre dann auch $uw = a^{n-|v|} b^n$ in der Sprache, was allerdings der Def. von L_1 widerspricht.

Syntaxbäume

Der Ableitung eines Worts einer Typ 2 oder Typ 3-Grammatik läßt sich ein Ableitungsbaum (Syntaxbaum) zuordnen.

Konstruktionsvorschrift:

1. S wird die Wurzel des Baums
2. Für $i = 1, 2, \ldots, n$: Bei Ersetzung einer Variablen A durch ein Wort z (gemäß der Regel $A \rightarrow z$) wird an den Knoten A eine Anzahl $|z|$ von Söhnen gehängt. Die Sohnknoten werden mit den einzelnen Symbolen von z beschriftet.

Beispiel 1:

Expr => Term => Term * Factor =>
Factor * Factor => Factor * a =>
a * a

oder

Expr => Term => Term * Factor =>
Term * a => Factor * a => a * a

Baum für beide Ableitungen