

## 4. Grammatiken

### 4.1. Grundlegende Definitionen

Wie lassen sich formale Sprachen beschreiben?

im endlichen Fall: Aufzählung der Wörter der Sprache

im unendlichen Fall: akzeptierende Automaten, Mengenausdrücke: z.B.  $\{a^n \mid n \text{ Primzahl}\}$ , reguläre Ausdrücke (für reguläre Sprachen) oder Grammatiken

Def.: Eine Grammatik  $G = (V, \Sigma, P, S)$  besteht aus

1. einer endlichen Menge  $V$  von Variablen,
2. einem endlichen Terminalalphabet  $\Sigma$ , wobei gelten muss:  $V \cap \Sigma = \emptyset$ ,
3. einer endlichen Menge  $P$  von Regeln, d.h. Elementen aus  $(V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ ,
4. einer Startvariable  $S \in V$ .

Regeln (auch: Produktionen) werden oft in der Form *linke Seite*  $\rightarrow$  *rechte Seite* notiert.

Def.: Seien  $u, v \in (V \cup \Sigma)^*$ ,  $G = (V, \Sigma, P, S)$  eine Grammatik. Wir definieren:  $u \Rightarrow_G v$  ( $u$  geht unter  $G$  direkt in  $v$  über) falls gilt:

1.  $u = xyz$ ,
2.  $v = xy'z$ ,
3.  $(y, y') \in P$ . (alternative Schreibweise:  $y \rightarrow y' \in P$ ).

Sei  $\Rightarrow_G^*$  die reflexive und transitive Hülle von  $\Rightarrow_G$ . (die kleinste Relation, die  $\Rightarrow_G$  enthält und reflexiv und transitiv ist; in anderen Worten: die Relation, die genau die Wortpaare enthält, die man durch Hintereinanderausführen einer beliebigen Anzahl (einschließlich 0) von direkten Übergängen erhält.)

Die von  $G$  erzeugte Sprache ist  $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$ .

Beispiel:  $G = (V, \Sigma, P, S)$  mit

$V = \{S, B, C\}$

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

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

Beispielableitung (falls  $G$  aus Kontext klar, lassen wir den Index  $\Rightarrow_G$  weg):

$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabBCC \Rightarrow aabbCC \Rightarrow aabbccC \Rightarrow aabbcc$

$L(G) = \{a^n b^n c^n \mid n > 0\}$

Bei Ableitungen obiger Art gibt es in vielen Schritten Alternativen, z.B. hätte man von  $S$  aus auch in  $aBC$  übergehen können, von  $aSBC$  in  $aaSBCBC$  etc. Die möglichen Ableitungen bilden einen Baum (genannt Erzeugungsbaum) mit Wurzel  $S$ . Knoten sind mit Wörtern beschriftet. Nachfolgerknoten beschreiben jeweils alle möglichen direkten Übergänge.

Es kann Sackgassen geben:

$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabcBC$   
kein Nachfolger!

#### 4.2 Die Chomsky-Hierarchie

Spezialfälle von Grammatiken:

Typ	Bezeichnung	zusätzliche Einschränkung
Typ 0		keine
Typ 1	kontextsensitiv	für alle Regeln $w_1 \rightarrow w_2$ gilt $ w_1  \leq  w_2 $
Typ 2	kontextfrei	wie Typ 1 und $w_1 \in V$
Typ 3	regulär	wie Typ 2 und: $w_2 \in \Sigma \cup \Sigma V$

Die Beispielgrammatik für  $a^n b^n c^n$  ist kontextsensitiv, aber nicht kontextfrei, da einige Regeln mehr als 1 Variable auf der linken Seite haben.

Sonderregelung für  $\epsilon$ :

1. Das leere Wort  $\epsilon$  kann nach obiger Definition nicht aus Typ 1, 2, 3 Grammatiken hergeleitet werden. Das ist unschön. Man erlaubt deshalb den Sonderfall  $S \rightarrow \epsilon$  unter der Voraussetzung, dass  $S$  nicht auf der rechten Seite einer Regel vorkommt.

Sei  $L(G)$  die Sprache einer Grammatik  $G = (V, \Sigma, P, S)$  mit  $\epsilon \notin L(G)$ . Sei  $S'$  ein neues Startsymbol. Die Grammatik

$$G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \rightarrow \epsilon, S' \rightarrow S\}, S')$$

erzeugt  $L(G) \cup \{\epsilon\}$  und erfüllt obige Bedingungen.

2. Für kontextfreie und reguläre (nur diese!) Grammatiken ist die Verwendung von Regeln der Form  $V \rightarrow \epsilon$  unproblematisch, denn man kann sie eliminieren, falls  $\epsilon \notin L(G)$ , bzw. auf die in 1. genannte Form beschränken, falls  $\epsilon \in L(G)$ .

Die Elimination von Regeln  $V \rightarrow \epsilon$  lässt sich folgendermaßen durchführen:

a) Zerlege die Variablen in disjunkte Teilmengen  $V_1$  und  $V_2$ , so dass  $V_1$  diejenigen Variablen enthält, aus denen das leere Wort hergeleitet werden kann.  $V_1$  ist die kleinste Menge, für die gilt:

- (1)  $A \rightarrow \epsilon \in P$  impliziert  $A \in V_1$ ,
- (2)  $B \rightarrow C_1 \dots C_k \in P$  und  $C_i \in V_1$  für alle  $i$  impliziert  $B \in V_1$ .

b) Lösche alle  $\epsilon$ -Regeln und füge für jede Regel  $D \rightarrow w$ , in der mindestens eine Variable aus  $V_1$  auf der rechten Seite vorkommt, alle Regeln der Form  $D \rightarrow w'$  hinzu, wobei  $w'$  ein nichtleeres Wort ist, das durch Weglassen von Variablen aus  $V_1$  in  $w$  entsteht.

Zu beachten: wenn auf der rechten Seite mehrfach Variablen aus  $V_1$  vorkommen, so müssen alle Möglichkeiten, diese wegzulassen, berücksichtigt werden.

Beispiel:  $D \rightarrow AaBbA$ ,  $A, B \in V_1$

neue Regeln:  $D \rightarrow aBbA$ ,  $D \rightarrow AabA$ ,  $D \rightarrow AaBb$ ,  $D \rightarrow abA$ ,  $D \rightarrow aBb$ ,  $D \rightarrow Aab$ ,  $D \rightarrow ab$

also: falls  $k$  Variablen aus  $V_1$  vorkommen,  $2^k - 1$  neue Regeln!

c) Falls  $S \in V_1$  erweitere die Grammatik entsprechend 1.

Beispiel:

$S \rightarrow A$        $A \rightarrow aBb$        $B \rightarrow bAa$        $A \rightarrow \epsilon$        $B \rightarrow \epsilon$

(mit den Konventionen:

$S$  Startsymbol, Großbuchstaben Variablen, Kleinbuchstaben Terminalsymbole,  $V$  und  $\Sigma$  enthalten genau die in den Regeln vorkommenden Variablen bzw. Terminalsymbole reichen die Regeln allein für die vollständige Spezifikation der Grammatik)

$V_1 = \{S, A, B\}$

neue Grammatik (Startsymbol  $S'$ ):

$S' \rightarrow \epsilon$        $S' \rightarrow S$        $S \rightarrow A$        $A \rightarrow aBb$        $A \rightarrow ab$        $B \rightarrow bAa$        $B \rightarrow ba$

$(L(G) = (ab)^*)$ : reguläre Grammatik:  $S \rightarrow \epsilon$ ,  $S \rightarrow aB$ ,  $B \rightarrow bA$ ,  $B \rightarrow b$ ,  $A \rightarrow aB$ )

Eine Sprache  $L$  heißt vom Typ  $i$  ( $0 \leq i \leq 3$ ), wenn es eine Grammatik  $G$  vom Typ  $i$  gibt mit  $L(G) = L$ . Die Bezeichner kontextsensitiv, kontextfrei, regulär werden auch für die entsprechenden Sprachen verwendet.

Da eine Typ  $j$  Grammatik jeweils ein Spezialfall einer Typ  $i$  Grammatik ist wenn  $i < j$ , bilden die Sprachen eine Hierarchie. Wir werden später zeigen, dass es sich jeweils um eine echte Inklusion handelt, also

Typ 3 Sprachen  $\subset$  Typ 2 Sprachen  $\subset$  Typ 1 Sprachen  $\subset$  Typ 0 Sprachen

Warum die Beschränkung bei Typ-1 Sprachen, dass rechte Seite mindestens so lang sein muss wie linke?

garantiert Entscheidbarkeit des Wortproblems:

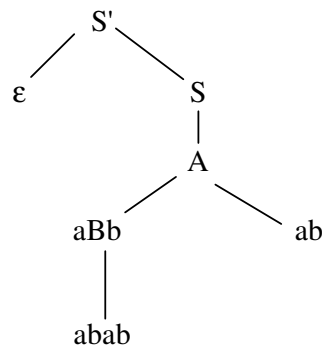
Wortproblem: gegeben Grammatik  $G$ ,  $w \in \Sigma^*$ , gilt  $w \in L(G)$ ?

*Satz: Das Wortproblem für Typ-1 Sprachen ist entscheidbar, d.h. es gibt einen Algorithmus, der bei Eingabe einer kontextsensitiven Grammatik  $G$  und eines Wortes  $w \in \Sigma^*$  entscheidet, ob  $w \in L(G)$ .*

Beweis: Sei  $|w| = n$ . Da die Länge eines einmal erzeugten Wortes aus  $(V \cup \Sigma)^*$  sich nicht wieder reduzieren kann, genügt es, den Erzeugungsbaum so weit zu erzeugen, dass alle weiteren Ableitungen entweder zu bereits erzeugten Wörtern oder zu Wörtern einer Länge  $> n$

führen würden. Da es nur endlich viele Wörter in  $(V \cup \Sigma)^*$  der Länge  $m \leq n$  gibt, ist dieser Baum endlich. Es gilt  $w \in L(G)$  gdw. ein Knoten des so erzeugten Baums mit  $w$  beschriftet ist.

Beispiel: obige Grammatik. Ist  $abab \in L(G)$ ? Baum aller Ableitungen bis Länge 4:



Jede weitere Ableitung führt zu Länge  $> 4$ .

Da  $abab$  im Baum vorkommt, gehört es zur Sprache,  $abba$  dagegen gehört nicht zur Sprache.

## 5. Reguläre Grammatiken und reguläre Sprachen

Wir müssen noch zeigen, dass die Bezeichnung "reguläre Sprachen" für die von regulären Grammatiken erzeugten Sprachen gerechtfertigt ist:

*Satz: Jede von einem endlichen Automaten akzeptierte Sprache wird von einer regulären Grammatik erzeugt.*

Beweis: Sei  $A = (Z, \Sigma, \delta, z_0, F)$  ein DEA. Wir definieren eine Typ-3 Grammatik  $G$ , so dass  $L(A) = L(G)$ .

$G = (V, \Sigma, P, S)$  mit  $V = Z$ ,  $S = z_0$ , und  $P$  besteht aus folgenden Regeln:

falls $\delta(z_1, a) = z_2$	dann $z_1 \rightarrow az_2 \in P$
falls $\delta(z_1, a) = z_2$ und $z_2 \in F$	dann $z_1 \rightarrow a \in P$
falls $\epsilon \in L(A)$	dann $z_0 \rightarrow \epsilon \in P$

Jetzt gilt:

$a_1 a_2 \dots a_n \in L(A)$  gdw.

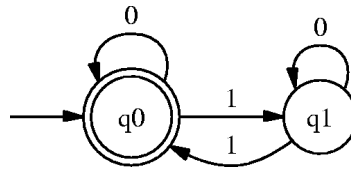
$\Leftrightarrow$  es gibt Folge  $z_0 z_1 \dots z_n$  mit  $z_0$  Startzustand,  $z_n \in F$ , und für alle  $0 < i \leq n$ :  $\delta(z_{i-1}, a_i) = z_i$

$\Leftrightarrow$  es gibt Folge  $z_0 z_1 \dots z_{n-1}$  mit  $z_0$  Startvariable und es gilt:

$$z_0 \Rightarrow a_1 z_1 \Rightarrow \dots \Rightarrow a_1 \dots a_{n-1} z_{n-1} \Rightarrow a_1 a_2 \dots a_n$$

$\Leftrightarrow a_1 a_2 \dots a_n \in L(G)$ .

Beispiel:



P besteht aus:

$q_0 \rightarrow 0q_0$      $q_0 \rightarrow 0$      $q_0 \rightarrow 1q_1$      $q_1 \rightarrow 0q_1$      $q_1 \rightarrow 1q_0$      $q_1 \rightarrow 1$   
 $q_0 \rightarrow \epsilon$

**Satz:** Jede von einer regulären Grammatik erzeugte Sprache wird von einem NEA (und damit auch von einem DEA) akzeptiert.

**Beweis:** Sei  $G = (V, \Sigma, P, S)$ .  $G$  sei bereits in die unter  $\epsilon$ -Sonderregelungen beschriebene Form überführt ( $\epsilon$  höchstens in der Regel  $S \rightarrow \epsilon$ , und dann  $S$  nicht rechts in Regel). Wir definieren einen NEA  $A = (Z, \Sigma, \delta, z_0, F)$ , so dass  $L(A) = L(G)$ , wie folgt:

$Z = V \cup \{X\}$      $X$  neues Symbol  
 $z_0 = S$   
 $F = \{S, X\}$  falls  $S \rightarrow \epsilon \in P$ ,  $\{X\}$  sonst

$B \in \delta(A, a)$  falls  $A \rightarrow aB \in P$   
 $X \in \delta(A, a)$  falls  $A \rightarrow a \in P$

Jetzt gilt für  $n > 0$  (der Fall  $n=0$ , also  $w = \epsilon$ , ist offensichtlich):

$a_1 a_2 \dots a_n \in L(G)$  gdw.

$\Leftrightarrow$  es gibt Folge  $A_0 A_1 \dots A_{n-1}$  von Variablen mit:

$S \Rightarrow a_1 A_1 \Rightarrow \dots \Rightarrow a_1 \dots a_{n-1} A_{n-1} \Rightarrow a_1 a_2 \dots a_n$

$\Leftrightarrow$  es gibt Folge  $A_0 A_1 \dots A_{n-1}$  von Zuständen mit:

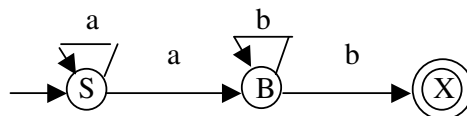
$A_1 \in \delta(S, a_1), A_2 \in \delta(A_1, a_2), \dots, X \in \delta(A_{n-1}, a_n)$ ,

$\Leftrightarrow a_1 a_2 \dots a_n \in L(A)$ .

Beispiel:

$S \rightarrow aS$      $S \rightarrow aB$   
 $B \rightarrow bB$      $B \rightarrow b$

wird zu:



## 6. Kontextfreie Grammatiken

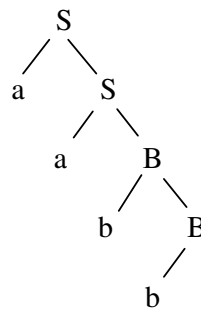
### 6.1 Syntaxbäume

Ein Syntaxbaum (auch Parsebaum, nicht zu verwechseln mit dem oben verwendeten Erzeugungsbaum, der alle möglichen Ableitungen darstellt) repräsentiert eine konkrete Ableitung in einer Typ-2 (oder Typ-3) Grammatik auf folgende Weise: Sei

$$S \Rightarrow x_0 \Rightarrow \dots \Rightarrow x_n$$

eine Ableitung des Wortes  $x$ . Die Wurzel des Baumes ist mit  $S$  beschriftet. Falls in der Ableitung ein Übergang von  $x_i$  zu  $x_{i+1}$  erfolgt, wobei die Variable  $V$  durch  $w$  ersetzt worden ist, so gibt es im Baum  $|w|$  Nachfolgeknoten von  $V$ , die mit den Symbolen in  $w$  markiert sind.

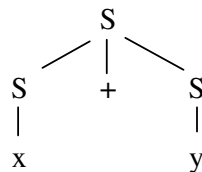
Syntaxbaum für  $aabb$ , Grammatik des obigen Beispiels:



Syntaxbäume für reguläre Grammatiken haben immer die Form einer solchen Kette.

Im Allgemeinen können Syntaxbäume mehrere Ableitungen repräsentieren:

$$S \rightarrow S * S \quad S \rightarrow S + S \quad S \rightarrow x \quad S \rightarrow y \quad S \rightarrow z$$



repräsentiert:

- 1)  $S \Rightarrow S + S \Rightarrow x + S \Rightarrow x + y$
- 2)  $S \Rightarrow S + S \Rightarrow S + y \Rightarrow x + y$

Eine Linksableitung ist eine Ableitung, bei der jeweils die am weitesten links stehende Variable ersetzt wird. Jedem Syntaxbaum entspricht genau eine Linksableitung (hier 1).

Manchmal gibt es mehr als einen Syntaxbaum für dasselbe Wort:



### 6.3 Chomsky Normalform

Def.: Eine kontextfreie Grammatik  $G$  ist in Chomsky Normalform (CNF), falls alle Regeln die Form  $A \rightarrow BC$  oder  $A \rightarrow a$  haben, wobei  $A, B, C$  Variablen sind und  $a$  ein Terminalsymbol.

Satz: Zu jeder kontextfreien Grammatik  $G$  mit  $\epsilon \notin L(G)$  gibt es eine äquivalente Grammatik  $G'$  in CNF.

Beweis: Wir erzeugen  $G'$  aus  $G$  durch folgende Schritte:

1. Eliminieren von  $\epsilon$ -Regeln nach bereits bekanntem Verfahren.
2. Eliminieren von Regeln der Form  $A \rightarrow B$ , wobei  $A, B$  Variablen sind:
  - a) Ersetzen von Variablen  $A_1, \dots, A_k$ , für die es einen Zyklus der Form  $A_1 \rightarrow A_2, \dots, A_{k-1} \rightarrow A_k, A_k \rightarrow A_1$  in  $P$  gibt, durch eine einzige Variable  $A$ .
  - b) Sortieren der verbleibenden Variablen, so dass  $V_i \rightarrow V_j$  impliziert  $i < j$ . Jetzt können, bei der letzten Variable beginnend, schrittweise Regeln  $V_i \rightarrow V_j$  ersetzt werden durch  $V_i \rightarrow x_1, \dots, V_i \rightarrow x_k$ , wobei  $V_j \rightarrow x_1, \dots, V_j \rightarrow x_k$  die Regeln sind, bei denen  $V_j$  auf der linken Seite vorkommt.

Beispiel:  $S \rightarrow A, A \rightarrow B, A \rightarrow aa, B \rightarrow aSa$                       Sortierung:  $S, A, B$   
 Ersetze  $A \rightarrow B$  durch  $A \rightarrow aSa$ :  $S \rightarrow A, A \rightarrow aSa, A \rightarrow aa, B \rightarrow aSa$   
 Ersetze  $S \rightarrow A$  durch  $S \rightarrow aSa, S \rightarrow aa$ :  $S \rightarrow aSa, S \rightarrow aa, A \rightarrow aSa, A \rightarrow aa, B \rightarrow aSa$   
 Hinweis: die letzten 3 Regeln können entfallen, da weder  $A$  noch  $B$  aus  $S$  erzeugbar.
3. Es gibt jetzt nur noch Regeln der Form  $A \rightarrow a$  sowie  $A \rightarrow x$  mit  $|x| > 1$ . Füge für jedes Terminalsymbol  $a$  eine neue Variable  $V_a$  ein, füge Regeln  $V_a \rightarrow a$  hinzu und ersetze jedes Vorkommen eines Terminalsymbols in  $x$  durch die entsprechende Variable.
4. Nur noch Regeln der Form  $A \rightarrow B_1 B_2 \dots B_k$  sind nicht in CNF. Füge neue Variablen  $C_2, \dots, C_{k-1}$  ein und ersetze obige Regel durch

$$\begin{aligned} A &\rightarrow B_1 C_2 \\ C_2 &\rightarrow B_2 C_3 \\ &\dots \\ C_{k-1} &\rightarrow B_{k-1} B_k \end{aligned}$$

Beispiel:  $S \rightarrow S', S' \rightarrow ab, S \rightarrow aSb, S' \rightarrow S$

Elimination Zyklus  $S \rightarrow S', S' \rightarrow S$ :  $S \rightarrow ab, S \rightarrow aSb$

Schritte 1,2b) irrelevant. Schritt 3: neue Variablen für  $a$  und  $b$ :  $V_a, V_b$

$$\begin{aligned} S &\rightarrow V_a V_b & V_a &\rightarrow a \\ S &\rightarrow V_a S V_b & V_b &\rightarrow b \end{aligned}$$

Schritt 4: neue Variable  $C$  benötigt

$$\begin{aligned} S &\rightarrow V_a V_b & V_a &\rightarrow a \\ S &\rightarrow V_a C & V_b &\rightarrow b \\ C &\rightarrow S V_b \end{aligned}$$

Anmerkung: es gibt eine weitere Normalform, die sogenannte Greibach-Normalform. Hier haben alle Regeln die Form  $A \rightarrow aB_1 \dots B_n, n \geq 0$ .

## 6.4 Der CYK-Algorithmus (Cocke, Younger, Kasami)

effizienter Algorithmus für das Wortproblem kontextfreier Sprachen, wobei diese in Form von CNF Grammatiken gegeben sein müssen.

Grundidee: Wort  $x = a$  der Länge 1 kann nur durch einmalige Anwendung einer Regel  $A \rightarrow a$  abgeleitet werden, Wort  $x$  der Länge  $n > 1$  nur durch Anwendung einer Regel  $A \rightarrow BC$ , wobei  $B$  Anfangsstück von  $x$  ableitet,  $C$  Endstück. Also: systematisch alle Teilworte von  $x$  der Länge  $1, \dots, n-1$  erzeugen und prüfen, ob sie erzeugbar sind. Dann prüfen, ob es eine Regel gibt mit  $BC$  auf der rechten Seite, so dass  $x$  aus dieser Regel erzeugt wird.

Sei  $x_{i,j}$  das Teilwort von  $x$ , das an Position  $i$  beginnt und Länge  $j$  hat. Wir konstruieren schrittweise eine Tabelle, die an Position  $i,j$  die Variablen enthält, aus denen Teilwort  $x_{i,j}$  des gesuchten Wortes abgeleitet werden kann.

obiges Beispiel:

$S \rightarrow AB$                        $A \rightarrow a$   
 $S \rightarrow AC$                        $B \rightarrow b$   
 $C \rightarrow SB$

zu testen: aabb

Tabelle:

		i ->			
		a	a	b	b
j	A	A	B	B	

Hier sind zunächst die Variablen eingetragen, aus denen Teilwörter der Länge 1 hergeleitet werden können. Als nächstes betrachten wir Teilwörter der Länge 2, die sich aus Regeln mit 2 passenden Variablen auf der rechten Seite erzeugen lassen (es gibt nur eins):

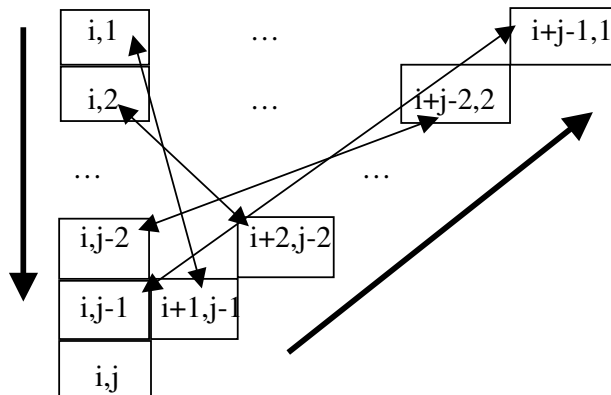
		i ->			
		a	a	b	b
j	A	A	B	B	
		S			

Wörter der Länge 3 und 4:

	i ->			
	a	a	b	b
j	A	A	B	B
		S		
		C		
	S			

Da S das gesamte Wort erzeugt, gehört dieses zur Sprache.

Im Allgemeinen können mehrere Variablen dasselbe Teilwort erzeugen. Grundsätzlich: um Tabelleneintrag  $i,j$  zu bestimmen, müssen folgende Paare von Einträgen verglichen werden:



Algorithmus:

Eingabe:  $x = a_1 \dots a_n$ , kontextfreie Grammatik  $G$  in CNF

```

for i := 1 to n do T[i,1] := {A ∈ V | A -> ai ∈ P} end;
for j := 2 to n do
    for i := 1 to n+1-j do
        T[i,j] := ∅;
        for k := 1 to j-1 do
            T[i,j] := T[i,j] ∪ {A ∈ V | A -> BC ∈ P, B ∈ T[i,k], C ∈ T[i+k,j-k]}
        end;
    end;
end;
if S ∈ T[1,n] then "Eingabewort herleitbar" else "Eingabewort nicht herleitbar".
    
```

Initialisierung, j = 1

Komplexität  $O(n^3)$ , da 3 ineinander verschachtelte for-Schleifen existieren.

Weiteres Beispiel:  $L = \{a^n b^m c^m \mid m, n > 0\}$

CNF-Grammatik:

$S \rightarrow AD$

$A \rightarrow AA \quad A \rightarrow a$

$D \rightarrow BC \quad B \rightarrow b$

$D \rightarrow BE \quad C \rightarrow c$

$E \rightarrow DC$

Zu testendes Wort: abbcc

	i ->				
	a	b	b	c	c
	A	B	B	C	C
j			D		
			E		
		D			
	S				

Damit ist das Wort abbcc aus der Grammatik herleitbar.

## 6.5 Das Pumping Lemma für kontextfreie Sprachen

*Satz:* Sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $n$ , so dass sich alle Wörter  $z \in L$  mit  $|z| \geq n$  zerlegen lassen in  $z = uvwxy$ , wobei gilt:

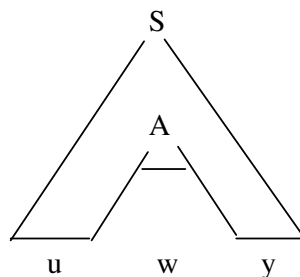
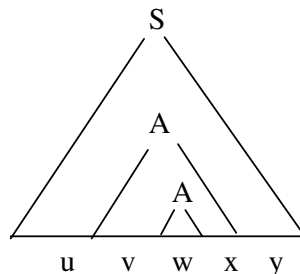
1.  $|vx| \geq 1$ ,
2.  $|vwx| \leq n$ ,
3. für alle  $i \geq 0$  gilt:  $uv^iwx^iy \in L$ .

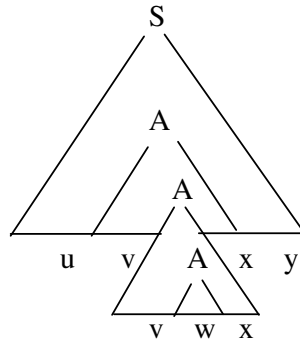
**Beweis:** Sei  $G$  CNF-Grammatik für  $L - \{\epsilon\}$ ,  $k$  die Anzahl der Variablen in  $G$ . Wähle  $n = 2^k$ . Der Syntaxbaum eines Wortes  $z \in L(G)$  ohne den jeweils letzten Ableitungsschritt (Variable  $\rightarrow$  Terminalsymbol) ist ein Binärbaum. Sei  $|z| \geq n$ . Da der Binärbaum  $\geq n$  Blätter hat, gibt es mindestens einen Pfad der Länge  $\geq k$ , d.h. an dem Pfad liegen  $\geq k+1$  Variablen. Also gibt es mindestens eine Variable  $A$ , die doppelt vorkommt, wobei der Abstand des 2. Vorkommens von  $A$  vom Blatt aus höchstens  $k$  ist. Sei  $w$  das Wort, das aus dem (vom Blatt aus gesehen) ersten Vorkommen von  $A$  abgeleitet wurde,  $vwx$  das aus dem zweiten Vorkommen. Auf das obere  $A$  muss eine Regel der Form  $A \rightarrow BC$  angewendet worden sein, deshalb kann  $vx$  nicht leer sein (Bed. 1.).

Da der Abstand des oberen  $A$  von den Blättern höchstens  $k$  ist, kann  $vwx$  höchstens die Länge  $2^k = n$  haben (Bed. 2).

Im Syntaxbaum können wir den Teilbaum, der am unteren  $A$  beginnt, gleich am oberen  $A$  einfügen. Man erhält so  $uw^2y$ . Man kann aber auch am unteren  $A$  den (alten) Teilbaum des oberen  $A$  einfügen und erhält  $uv^2wx^2y$ . Das kann man beliebig oft wiederholen und erhält  $uv^iwx^iy$  (Bed. 3).

Skizzen:





Das Pumping Lemma für kontextfreie Sprachen wird genauso verwendet wie das für reguläre Sprachen:

Um zu zeigen, dass eine Sprache  $L$  nicht kontextfrei ist, wähle Wort aus  $L$  (abhängig von Pumping-Zahl  $n$ ), so dass aufgrund des Pumping Lemmas ein anderes Wort  $w'$  auch in  $L$  sein müsste, was aber nach Definition von  $L$  nicht der Fall ist. Damit kann  $L$  nicht kontextfrei sein.

Standardbeispiel:  $L = \{a^m b^m c^m \mid m \geq 1\}$  ist nicht kontextfrei.

Beweis: Angenommen,  $L$  wäre kontextfrei. Sei  $n$  die Pumping-Zahl von  $L$  und betrachte  $a^n b^n c^n$ . Dieses Wort müsste sich zerlegen lassen in  $uvwxy$  mit  $|vwx| \leq n$ .  $vwx$  kann deshalb nicht sowohl  $a$ 's wie  $c$ 's enthalten, sondern mindestens eines dieser beiden Terminalsymbole kommt nicht vor. Nach dem Pumping Lemma müsste  $uwy$  ebenfalls in  $L$  sein, was aber unmöglich ist, denn da  $|vwx| \geq 1$  kommt das nicht in  $vwx$  enthaltene Terminalsymbol in  $uwy$  öfter vor als mindestens eines der anderen Symbole.

### 6.6 Abschlusseigenschaften kontextfreier Sprachen

Satz: Kontextfreie Sprachen sind abgeschlossen unter Vereinigung, Konkatenation und Kleene-Abschluss, nicht aber unter Durchschnitts- und Komplementbildung.

Beweis:

Abgeschlossenheit (Beweis genauso auch für Typ 0 und Typ 1 Sprachen):

seien  $G_1 = (V_1, \Sigma, P_1, S_1)$  und  $G_2 = (V_2, \Sigma, P_2, S_2)$  Grammatiken für die Sprachen  $L_1$  und  $L_2$ . Ohne Beschränkung der Allgemeinheit sei  $V_1 \cap V_2 = \emptyset$  und  $S$  neues Symbol .

$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$  erzeugt  $L_1 \cup L_2$ .

$G = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$  erzeugt  $L_1 L_2$ .

$G = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow S_1 S, S \rightarrow \epsilon\}, S)$  erzeugt  $L_1^*$ .

Nicht-Abgeschlossenheit:

Die Sprachen  $\{a^n b^n c^m \mid n, m > 0\}$  und  $\{a^n b^m c^m \mid n, m > 0\}$  sind kontextfrei (Übungsaufgabe). Der Schnitt dieser Sprachen  $\{a^n b^n c^n \mid n > 0\}$  ist wie im Abschnitt über das Pumping Lemma bewiesen nicht kontextfrei.

Wären die kontextfreien Sprachen abgeschlossen unter Komplement, so auch unter Schnitt, denn aus Komplement und Vereinigung lässt sich der Schnitt erzeugen:  $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ .