Kontextfreie (Typ-2) Sprachen

Bsp.:

- $L_1 = \{0^n1^n \mid n>0\} \subset \{0,1\}^*$

  $S \rightarrow 0S1 \mid 01$

- $L_2 = \{w \mid w \in \{0,1\}^* \text{ und } w=w^{rev}\} \subset \{0,1\}^*$

  $S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

- $L_3 = \{w \mid w \text{ hat genausoviele Nullen wie Einsen}\} \subset \{0,1\}^*$

  $S \rightarrow 0B \mid 1A \mid \varepsilon$

  $A \rightarrow 0 \mid 0S \mid 1AA$

  $B \rightarrow 1 \mid 1S \mid 0BB$

  - $S \rightarrow 1A \rightarrow 11AA \rightarrow 11A0 \rightarrow 110S0 \rightarrow 1100B0 \rightarrow 110010$
  - $S \rightarrow 1A \rightarrow 11AA \rightarrow 110SA \rightarrow 1100BA \rightarrow 11001A \rightarrow 110010$

Syntaxbaum für kontextfreie Sprachen

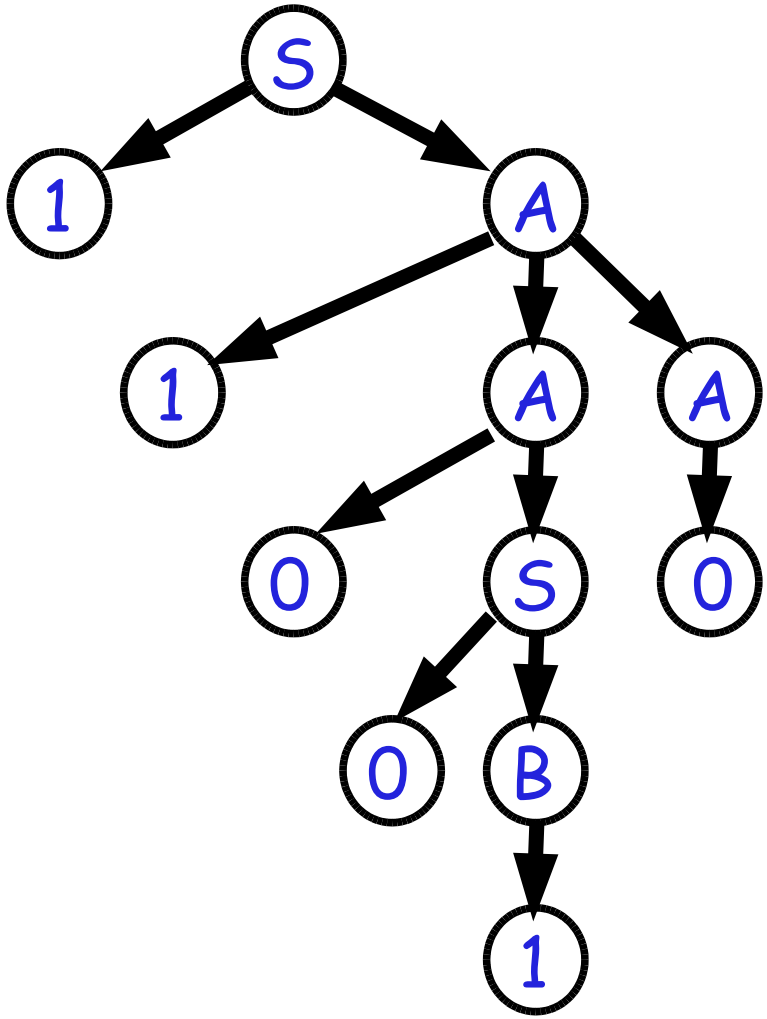
Def.: Sei $G=(\Sigma,V,S,R)$ Typ-2 Grammatik. Baum T heisst Syntaxbaum für G, falls

- Wurzel mit S beschriftet
- innere Knoten mit V beschriftet
- Blätter mit $\{\varepsilon\}\cup\Sigma$ beschriftet
- Markierung der inneren Knoten und aller Kinder entspricht einer Regel aus R

T heisst Syntaxbaum für $w\epsilon L(G)$, falls Beschriftung der Blätter von T (von links nach rechts gelesen) w ergibt

Bsp. Syntaxbaum für Typ-2 Sprachen

$S \rightarrow 0B \mid 1A,\ A \rightarrow 0 \mid 0S \mid 1AA,\ B \rightarrow 1 \mid 1S \mid 0BB$

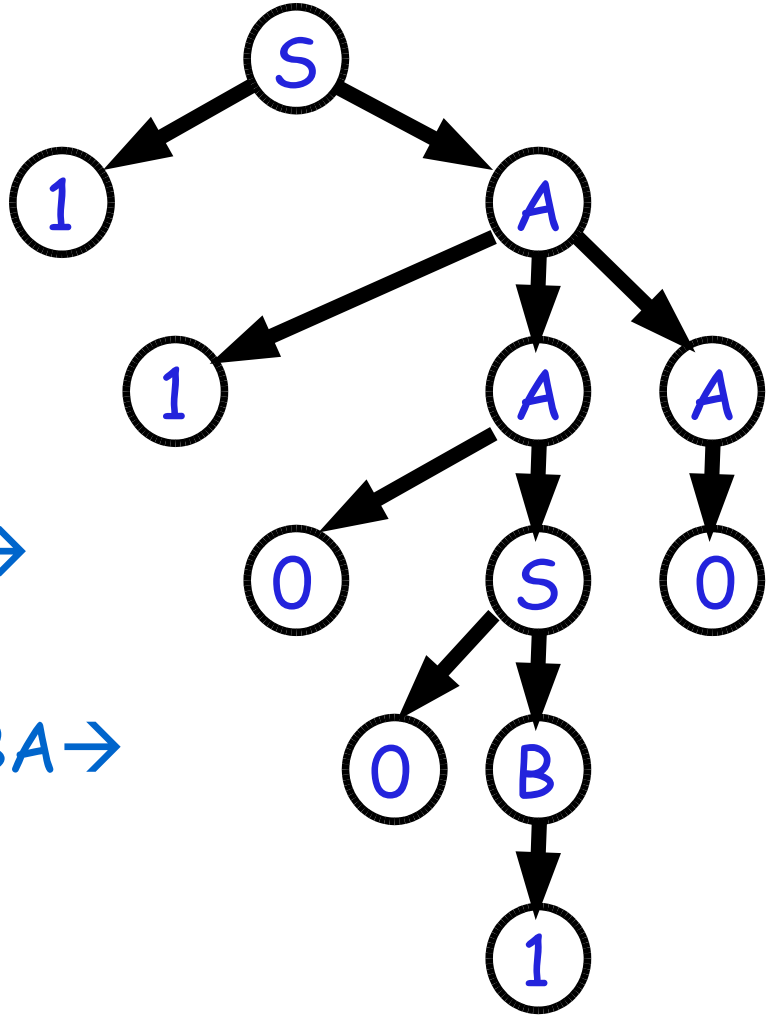


Linksableitung

Bem.:

- Ableitung, bei der immer die "linkeste" Regel angewendet wird, heisst Linksableitung
- Syntaxbaum kann mehrere Ableitungen darstellen:
  - S→1A→11AA→11A0→110S0→ 1100B0→110010
  - S→1A→11AA→110SA→1100BA→ 11001A→110010

  (aber nur eine Linksableitung)



Eindeutige kontextfreie Sprachen

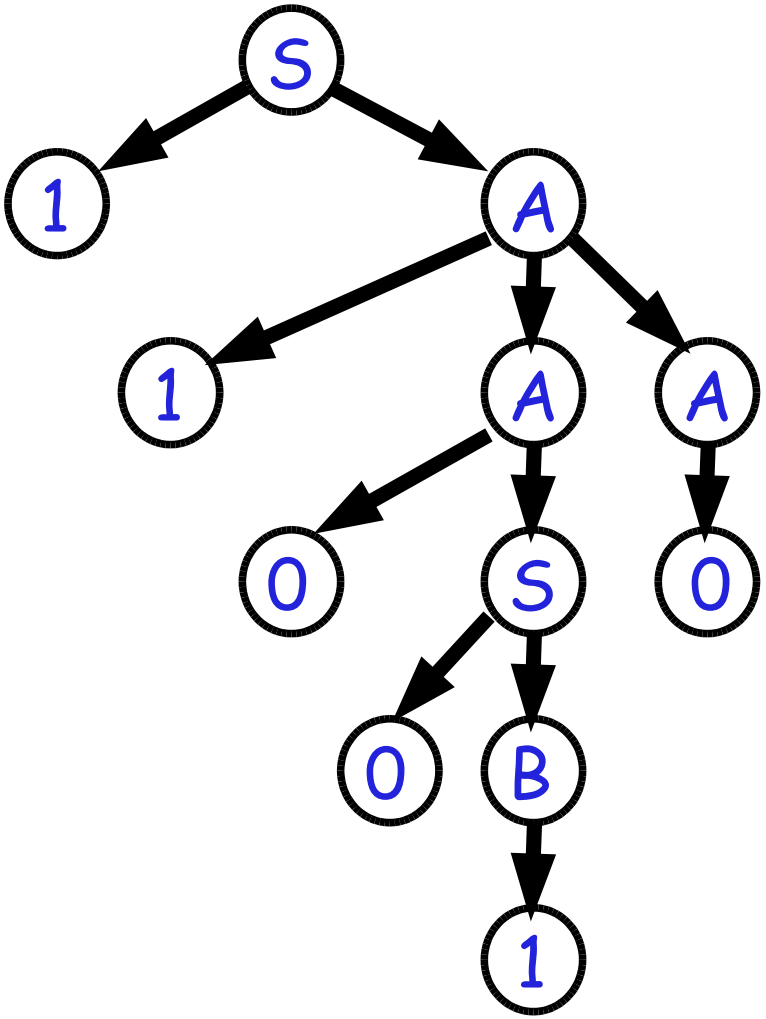
Def.:

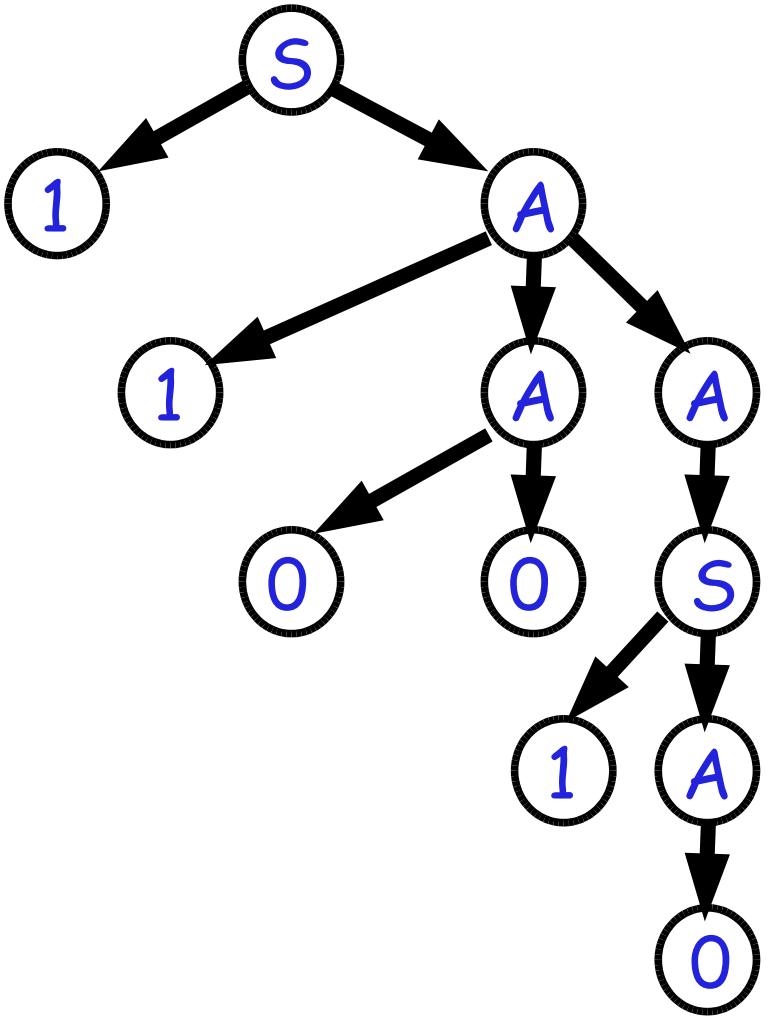
- Typ-2 Grammatik $G=(\Sigma,V,S,R)$ heisst eindeutig, falls es für jedes Wort $w \in L(G)$ genau einen Syntaxbaum gibt
- Typ-2 Sprache L heisst eindeutig, falls es eine eindeutige Typ-2 Grammatik G mit $L=L(G)$ gibt; ansonsten heisst L inhärent mehrdeutig

Motivation: Wortproblem für eindeutige Grammatiken sollte einfacher sein

Bsp. Mehrdeutige Grammatik

$S \rightarrow 0B \mid 1A,\ A \rightarrow 0 \mid 0S \mid 1AA,\ B \rightarrow 1 \mid 1S \mid 0BB$





Chomsky-Normalform (CNF)

Def.: Typ-2 Grammatik $G=(\Sigma,V,S,R)$ ist in Chomsky-Normalform (CNF), falls alle Regeln folgende Form haben:

- $A \rightarrow BC$ mit $A,B,C \epsilon V$ *oder*
- $A \rightarrow a$ mit $A \epsilon V$, $a \epsilon \Sigma$

Bem.:

- ε kann von Grammatik in CNF nicht erzeugt werden
- T Syntaxbaum für CNF-Grammatik

  → alle inneren Knoten haben Grad 2,

  Väter der Blätter haben Grad 1

Herstellen der Chomsky-Normalform

Satz: Jede Typ-2 Sprache L mit $\varepsilon \in L^c$, kann von einer Typ-2 Grammatik in CNF erzeugt werden

Bew.:

Idee: Überführe Regeln einer Typ-2 Grammatik $G=(\Sigma,V,S,R)$ mit $L=L(G)$ schrittweise in CNF

Erinnerung: alle Regeln von G haben Form

$A \rightarrow w$ mit $A \in V$ und $w \in (V \cup \Sigma)^*$

Herstellen der Chomsky-Normalform

1. Schritt:

für alle $a \in \Sigma$

- neue Variable Y_a und neue Regel $Y_a \rightarrow a$
- ersetze in Regeln von G (rechte Seite) a durch Y_a

Konsequenz: alle Regeln haben Form

$A \rightarrow w$ mit $A \in V$ und $w \in V^+$ oder $w \in \Sigma \cup \{\varepsilon\}$

Herstellen der Chomsky-Normalform

2. Schritt:

für Regel $A \to B_1 \ldots B_m$ mit $m>2$

- ersetze durch neue Regeln
  - $A \to B_1 C_1$,
  - $C_i \to B_{i+1} C_{i+1}$ für $1 \leq i \leq m-3$ und
  - $C_{m-2} \to B_{m-1} B_m$
- mit neuen Variablen $C_1, \ldots, C_{m-2}$

Konsequenz: alle Regeln haben Form

$A \to w$ mit $A \in V$ und $w \in V^2$, $w \in V$ oder $w \in \Sigma \cup \{\varepsilon\}$

Herstellen der Chomsky-Normalform

3. Schritt ("Eliminieren von ε-Regeln"):

sei $E := \{A \in V \mid A \rightarrow^* \varepsilon\}$

- ersetze Regeln $A \rightarrow BC$ mit $B \in E$ durch $A \rightarrow C$
- ersetze Regeln $A \rightarrow BC$ mit $C \in E$ durch $A \rightarrow B$
- streiche Regeln $A \rightarrow \varepsilon$

Konsequenz: alle Regeln haben Form

$A \rightarrow w$ mit $A \in V$ und $w \in V^2$, $w \in V$ oder $w \in \Sigma$

Herstellen der Chomsky-Normalform

4. Schritt ("Eliminieren von Kettenregeln"):

(A) solange es Kreis $A_1 \to A_2 \to \ldots \to A_r \to A_1$ gibt

- ersetze $A_2,\ldots,A_r$ durch A_1, streiche Regel $A_1 \to A_1$

(B) bestimme Nummerierung $A_1,\ldots,A_m$ der Variablen,

so dass gilt: falls $A_i \to A_j$, so ist $i < j$

für alle $i = k,k-1,\ldots,1$ und für alle $j=i+1,\ldots,k$ mit $A_i \to A_j$

- streiche Regel $A_i \to A_j$
- falls $A_j \to a$ mit $a \in \Sigma$ Regel ist,

führe neue Regel $A_i \to a$ ein

Konsequenz: G ist in CNF

Herstellen der Chomsky-Normalform

Beh.: Die modifizierte Grammatik erzeugt die Sprache L

Bew.: (→*Übung*)

Bsp.: $L_1 = \{0^n1^n \mid n>0\}$, $S \to 0S1 \mid 01$

Herstellen der CNF:

- $Y_1 \to 1$, $Y_0 \to 0$, $S \to Y_0SY_1 \mid Y_0Y_1$
- $Y_1 \to 1$, $Y_0 \to 0$, $S \to Y_0C \mid Y_0Y_1$, $C \to SY_1$

Wortproblem für Typ-2 Sprachen

Satz: Wortproblem einer Typ-2 Grammatik $G=(\Sigma,V,S,R)$ in CNF kann in $O(|R||w|^3)$ Zeit entschieden werden

Gegeben: G, $w = w_1...w_n \in \Sigma^*$

Entscheide, ob $w \in L(G)$

Bew.: (Alg. von Cooke-Younger-Kasami '67 - CYK)

Idee:

- $w \in L(G) \Leftrightarrow \exists (S \rightarrow BC) \in R$ und $B \rightarrow^* p$, $C \rightarrow^* s$, $w=ps$
- berechne $V_{i,j}$ mit dynamischem Programmieren:

  mit $V_{i,j} := \{A \in V \mid A \rightarrow^* w_i...w_j\}$ gilt $w \in L(G) \Leftrightarrow S \in V_{1,n}$

CYK-Algorithmus

berechne $V_{i,j} := \{A \in V \mid A \rightarrow^* w_i \ldots w_j\}$

mit dynamischem Programmieren:

$V_{i,i} = \{A \in V \mid A \rightarrow^* w_i\} = \{A \in V \mid A \rightarrow w_i\}$, da G in CNF

$V_{i,j} = \{A \in V \mid \exists (A \rightarrow BC) \in R \; \exists k \in \{i, j-1\} : B \in V_{i,k} \text{ und } C \in V_{k+1,j}\}$

dynamisches Programmieren:

for $l=2,..,n$

  for $i=1,..,n-l+1$

    $j = i+(l-1)$

    $V_{i,j} := \cup_{i \le k \le j-1} \{A \in V \mid \exists B \in V_{i,k}, \exists C \in V_{k+1,j} \text{ und } (A \rightarrow BC) \in R\}$

CYK-Algorithmus

for $l=2,..,n$

  for $i=1,..,n-l+1$

    $j = i+(l-1)$

    $V_{i,j} := \cup_{i \le k \le j-1} \{A \in V \mid \exists B \in V_{i,k}, \exists C \in V_{k+1,j}$ und $(A \rightarrow BC) \in R\}$

- Aufwand zur Berechnung von $V_{i,j}$: $O(|R|n)$

  (betrachte $\le |R|$ Regeln und $\le n$ Werte für k)
- Gesamtaufwand: $O(|R|n^3)$

Bsp. CYK-Algorithmus

$L_1 = \{a^n b^n \mid n>0\}$

$A \rightarrow a,$

$B \rightarrow b,$

$S \rightarrow AB \mid AT,$

$T \rightarrow SB$

$a_{(j=1)}$	A			
$a_{(j=2)}$		A		
$b_{(j=3)}$			B	
$b_{(j=4)}$				B
	$a_{(i=1)}$	$a_{(i=2)}$	$b_{(i=3)}$	$b_{(i=4)}$

$w = aabb$

Bsp. CYK-Algorithmus

$L_1 = \{a^n b^n \mid n>0\}$

$A \rightarrow a,$

$B \rightarrow b,$

$S \rightarrow AB \mid AT,$

$T \rightarrow SB$

$a_{(j=1)}$	A			
$a_{(j=2)}$		A		
$b_{(j=3)}$		S	B	
$b_{(j=4)}$				B
	$a_{(i=1)}$	$a_{(i=2)}$	$b_{(i=3)}$	$b_{(i=4)}$

$w = aabb$

Bsp. CYK-Algorithmus

$L_1 = \{a^n b^n \mid n>0\}$

$A \rightarrow a,$

$B \rightarrow b,$

$S \rightarrow AB \mid AT,$

$T \rightarrow SB$

$a_{(j=1)}$	A			
$a_{(j=2)}$		A		
$b_{(j=3)}$		S	B	
$b_{(j=4)}$		T		B
	$a_{(i=1)}$	$a_{(i=2)}$	$b_{(i=3)}$	$b_{(i=4)}$

$w = aabb$

Bsp. CYK-Algorithmus

$L_1 = \{a^n b^n \mid n>0\}$

$A \rightarrow a,$

$B \rightarrow b,$

$S \rightarrow AB \mid AT,$

$T \rightarrow SB$

$w = aabb$

$a_{(j=1)}$	A			
$a_{(j=2)}$		A		
$b_{(j=3)}$		S	B	
$b_{(j=4)}$	S	T		B
	$a_{(i=1)}$	$a_{(i=2)}$	$b_{(i=3)}$	$b_{(i=4)}$