Grundlagen der Theoretischen Informatik

Wintersemester 2024/25

Prof. Dr. Heribert Vollmer

Institut für Theoretische Informatik
Leibniz Universität Hannover

Link für Kurzfragen:

URL: https://pingo.coactum.de/events/739935

Bitte scannen!

URL: `https://pingo.coactum.de/events/739935`

Definition

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

- V ist eine endliche Menge, die so genannte Menge der Variablen
- Σ ist ein Alphabet, das so genannte Terminalalphabet, mit $V \cap \Sigma = \emptyset$
- P ist die endliche Menge der Produktionen, $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$
- $S \in V$ ist die so genannte Startvariable

Definition

Sei $G = (V, \Sigma, P, S)$ eine Grammatik und seien $u, v \in (V \cup \Sigma)^*$. Wir definieren eine Relation $\Rightarrow_G$ wie folgt:

- $u \Rightarrow_G v$, falls u, v zerlegt werden können in Teilwörter $u = xyz$ und $v = xy'z$ mit $x, z \in (V \cup \Sigma)^*$ und $y \to y'$ ist Regel in P.
  „u geht unter (Anwendung einer Regel in) G unmittelbar über in v“
- $u \Rightarrow_G^* v$, falls $u = v$ oder es Wörter $w_1, \ldots, w_k \in (V \cup \Sigma)^*$ gibt mit $u = w_1$, $w_i \Rightarrow_G w_{i+1}$ für $i = 1, 2. \ldots, k-1$ und $v = w_k$.
  Wir lassen den Index G weg, falls dieser eindeutig ist.

Die von G erzeugte Sprache ist $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.
Eine Ableitung von $w \in L(G)$ in k Schritten ist eine Folge $(w_0, w_1, \ldots, w_k)$ mit $w_0 = S$, $w_k = w$ und $w_i \Rightarrow_G w_{i+1}$ für $i = 0, 1, \ldots, k-1$.

Beispiel für eine Grammatik:

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

$$
\begin{aligned}
P = \{\; & S \xrightarrow{(1)} aSBC, \\
& S \xrightarrow{(2)} aBC, \\
& CB \xrightarrow{(3)} BC, \\
& aB \xrightarrow{(4)} ab, \\
& bB \xrightarrow{(5)} bb, \\
& bC \xrightarrow{(6)} bc, \\
& cC \xrightarrow{(7)} cc \;\}
\end{aligned}
$$

TP ∧

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

$\supseteq$: $S \overset{(1)}{\Longrightarrow}{}^{*} a^{n-1} S (BC)^{n-1}$

((n-1)-mal Regel (1))

$\overset{(2)}{\Longrightarrow} a^n (BC)^n$

$\overset{(3)}{\Longrightarrow}{}^{*} a^n B^n C^n$

$\Longrightarrow^{*} a^n b^n c^n$

$\subseteq$: sh. Skript.

Beispiel für eine Ableitung:

$S \overset{(1)}{\Longrightarrow} \underbrace{\underbrace{a}_{x}\underbrace{S}_{y}\underbrace{BC}_{z}}_{u}$

$y \to y'$

$\hat{=}\; S \to aSBC$

$\overset{(1)}{\Longrightarrow} \underbrace{a}_{x}\underbrace{aSBC}_{y'}\underbrace{BC}_{z}$

$\overset{(1)}{\Longrightarrow} \underbrace{aa}_{x}\underbrace{S}_{y}\underbrace{BCBC}_{z}$

$S \to aBC$

$\overset{(2)}{\Longrightarrow} \underbrace{aa}_{x}\underbrace{aBC}_{y'}\underbrace{BCBC}_{z}$

$\overset{(3)}{\Longrightarrow} aaaBBCCBC$

$\overset{(3)}{\Longrightarrow} aaaBBCBCC$

$\overset{(3)}{\Longrightarrow} aaaBBBCCC$

$\overset{(4)}{\Longrightarrow} aaabBBCCC$

$\overset{(5)}{\Longrightarrow} aaabbBCCC$

$\overset{(5)}{\Longrightarrow} aaabbbCCC$

$\overset{(6)}{\Longrightarrow} aaabbbcCC$

$\overset{(7)}{\Longrightarrow} aaabbbccC$

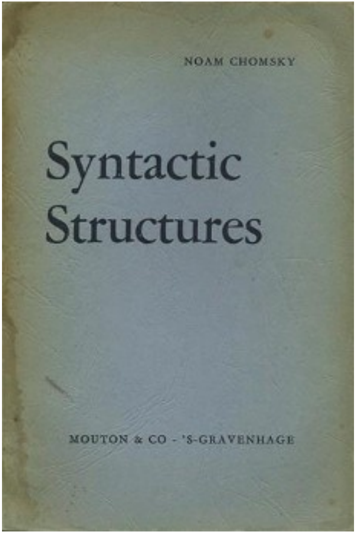
$\overset{(7)}{\Longrightarrow} aaabbbccc$

$= a^3 b^3 c^3 \in L(G)$

Die Chomsky-Hierarchie

Noam Chomsky

∗ 7. Dez. 1928, Philadelphia



1957: Syntactic Structures

Definition

- Jede Grammatik ist vom Typ 0 (d. h. keine Einschränkungen).
- Eine Grammatik ist vom Typ 1 (oder: kontextsensitiv), falls für alle ihre Regeln $u \to v$ gilt: $|u| \le |v|$. nicht verkürzend
- Eine Typ-1-Grammatik ist vom Typ 2 (oder: kontextfrei), falls für alle ihre Regeln $u \to v$ gilt, dass u eine einzelne Variable ist (d. h. $u \in V$). (Bsp.: arithm. Ausdrücke
- Eine Typ-2-Grammatik ist vom Typ 3 (oder: regulär), falls für alle ihre Regeln $u \to v$ gilt, dass v ein einzelnes Terminalzeichen ist ($v \in \Sigma$) oder v aus einem Terminalzeichen gefolgt von einer Variablen besteht.

$A \to a$

$A \to aB$

Bsp. für Typ-2-Grammatik:

$G' \quad G = (\{S\}, \{a,b\}, P, S)$ mit

$P = \{ S \to aSb, \quad S \to ab \}$, $S \to \varepsilon$

$L(G) = \{a^n b^n / n \geq 1\}$ Typ-2-Sprache

$L(G') = \{a^n b^n / n \geq 0\}$ Typ 3 ?

$S \Rightarrow ab$

$S \Rightarrow aSb \Rightarrow aabb$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow a^{n-1}Sb^{n-1} \Rightarrow a^n b^n$

Zurück zur Syntax der Aussagenlogik

$\neg(p_1 \wedge p_2)$ $(p_1 \wedge p_2)$

EBNF:

$\phi ::= p \mid 0 \mid 1 \mid \neg\phi \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \phi)$,

wobei p eine aussagenlogische Variable ist,
also $p \in \{p_1, p_2, p_3, \ldots\}$.

Typ-2-Grammatik:

$S \rightarrow V \mid C \mid \neg S \mid (S \wedge S) \mid (S \vee S) \mid (S \rightarrow S) \mid (S \leftrightarrow S)$
$V \rightarrow p_1 \mid p_2 \mid p_3 \mid \ldots$
$C \rightarrow 0 \mid 1$

Problem: unendliches Alphabet!

Zurück zur Syntax der Aussagenlogik

$p_1 \cong pI$
$p_2 \cong pII$

Lösung:

Für p_i schreiben wir: pI^i.

$G = (\Sigma_{AL}, \{S, V, C\}, P, S)$, wobei

$$\Sigma_{AL} = \{p, I, 0, 1, \wedge, \vee, \neg, \rightarrow, \leftrightarrow, (,)\}$$

$$P = \left\{ \begin{array}{l} S \rightarrow V \mid C \mid \neg S \mid (S \wedge S) \mid (S \vee S) \mid (S \rightarrow S) \mid (S \leftrightarrow S) \\ V \rightarrow p \mid VI \\ C \rightarrow 0 \mid 1 \end{array} \right\}$$

Typ 2

$Form_{AL}$

Die syntaktisch korrekten Wörter (also die aussagenlogischen Formeln) kann man nun z. B. wie folgt erzeugen:

$$S \Rightarrow \neg S \Rightarrow \neg(S \wedge S) \Rightarrow \neg(VI \wedge VI) \Rightarrow \neg(VI \wedge VII)$$
$$\Rightarrow \neg(pI \wedge pII) \simeq \neg(p_1 \wedge p_2)$$

Spezialfall des leeren Wortes

Bei einer Grammatik $G = (V, \Sigma, P, S)$ vom Typ 1, 2 oder 3 ist unabhängig von den oben genannten Restriktionen die Regel $S \to \varepsilon$ zugelassen.

Ist aber $S \to \varepsilon \in P$, so darf es keine Regel in P geben, in der S auf der rechten Seite vorkommt. kann immer hergestellt werden! (ÜA)

$$S \Rightarrow \ldots \Rightarrow \varepsilon$$

$S = u$, $|u| = 1$ … $\varepsilon = v$, $|v| = 0$

bei Typ 1 nicht möglich

- $S \Rightarrow \varepsilon$ einzige verkürzende Ableitung
- $S \Rightarrow u_1 \Rightarrow u_2 \Rightarrow u_3 \Rightarrow \ldots \Rightarrow u_n = v$ nicht verkürzend

$|S| \le |u_1| \le |u_2| \le |u_3| \ldots \le |u_n|$

Eine Sprache $L \subseteq \Sigma^*$ heißt vom Typ 0 (Typ 1, Typ 2, Typ 3), falls es eine Typ-0-Grammatik (Typ-1-Grammatik, Typ-2-Grammatik, Typ-3-Grammatik) G gibt mit $L = L(G)$.

Beispiel: G hat folgende Regeln:

$S \to ab\underline{B}$ | $S \to aA$ | Grammatik vom Typ 2!

$S \to a$ | $A \to bB$

$B \to b$ | $S \to a$ | Typ-3-Grammatik

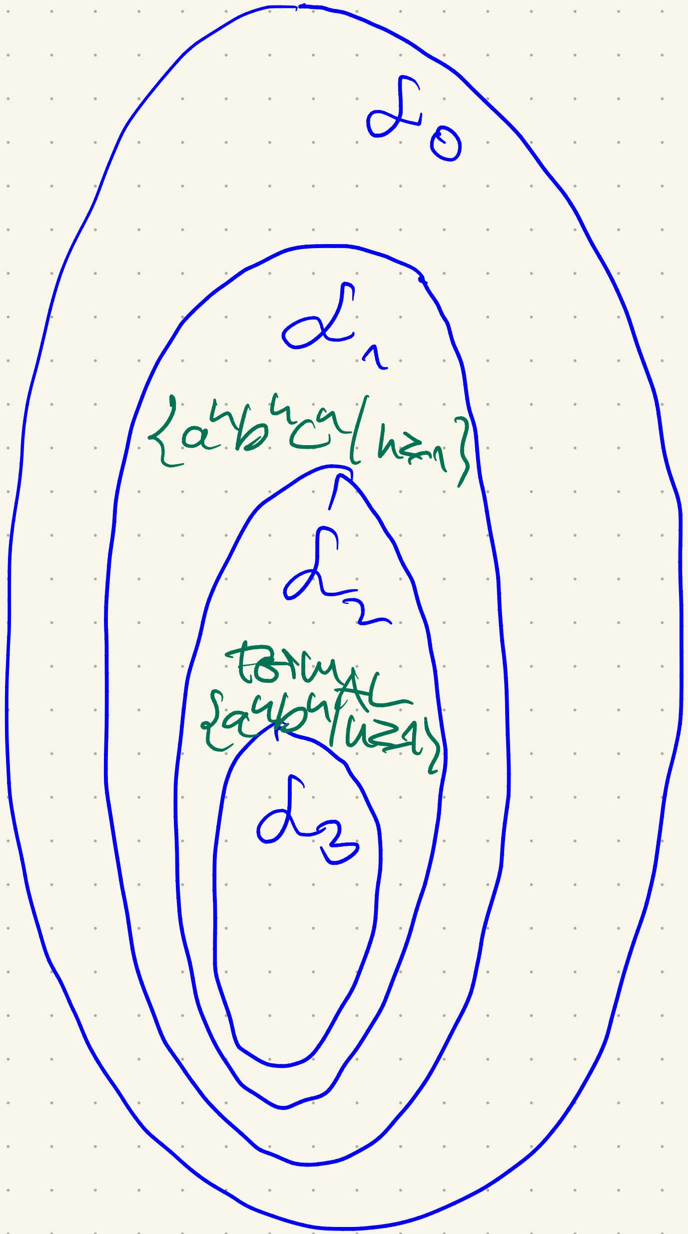
$B \to b$

$L(G) = \{a, abb\}$

$S \Rightarrow \underline{a}$, $S \Rightarrow abB \Rightarrow \underline{abb}$

L(G) ist vom Typ 3!

$\mathcal{L}_i$ = Klasse der Sprache vom Typ i



$L \subseteq \Sigma^*$, $|L| = \infty$

$L = L(G)$

↑ endl. Beschreibung

$w \to \boxed{A}$ $\to$ 1 , falls $w \in L(G)$

$\phantom{w \to \boxed{A}}$ $\to$ 0 , sonst

„Parser"

Wortproblem für G:

Eingabe: $w \in \Sigma^*$

Ausgabe: 1, falls $S \Rightarrow^* w$

0, sonst

Satz

Das Wortproblem für Typ-1-Sprachen ist „entscheidbar“ , d. h. es gibt einen Algorithmus, der bei Eingabe einer kontextsensitiven Grammatik $G = (V, \Sigma, P, S)$ und eines Wortes $w \in \Sigma^*$ nach endlicher Zeit mit der Ausgabe „$w \in L(G)$“ oder „$w \notin L(G)$“ anhält.

Beweis des Satzes von F. 15:

Sei $G = (V, \Sigma, P, S)$ vom Typ 1, $w \in \Sigma^*$, $|w| = n$.

$T_m^n =_{def} \{ u \in (V \cup \Sigma)^* \mid S \overset{*}{\underset{G}{\Rightarrow}} u,\ |u| \leq n$ und

u lässt sich aus S in $\leq m$ Schritten ableiten$\}$

Es gilt:

- $T_0^n = \{S\}$
- $T_{m+1}^n = T_m^n \cup \{ u \mid u \in (V \cup \Sigma)^*, |u| \leq n$ und $v \underset{G}{\Rightarrow} u$ mit $v \in T_m^n \}$
- $T_0^n \subseteq T_1^n \subseteq T_2^n \subseteq T_3^n \subseteq \ldots$
- Es gibt nur endl. viele Wörter der Länge $\leq n$.
- Es gibt $m_0 \geq 0$ mit

$$T_0^n \subsetneq T_1^n \subsetneq T_2^n \subsetneq \cdots \subsetneq T_{m_0}^n = T_{m_0+1}^n = T_{m_0+2}^n$$

$$= \bigcup_{m \geq 0} T_m^n$$

- Es gilt: $w \in L(G)$
  gdw. $w \in T_{m_0}^n$ für das obige m_0.

Algorithmus:

Eingabe: $G = (V, \Sigma, P, S)$, $w \in \Sigma^*$

Methode:

$n := |w|$

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

loop

$T := U;$

$U := T \cup \{u \in (V \cup \Sigma)^* \mid |u| \leq n, v \Rightarrow_G u, v \in T\}$

until $U = T$

if $w \in T$ then print "1"

else print "0"

$\square$