1c

Weitere Grammatikbegriffe sowie Eigenschaften von Sprachen

PD Dr. Jan Johannsen

Institut für Informatik

Stand: 25. März 2026
Basierend auf Folien von PD Dr. David Sabel und Prof. Dr. Jasmin Blanchette

Wiederholung: Definition einer Grammatik

Definition

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

- V ist eine endliche Menge von Variablen (alternativ Nichtterminalen)
- Σ (mit $V \cap \Sigma = \emptyset$) ist ein Alphabet von Zeichen (alternativ Terminalen)
- P ist eine endliche Menge von Produktionen (alternativ Regeln) von der Form $\ell \to r$ wobei $\ell \in (V \cup \Sigma)^+$ und $r \in (V \cup \Sigma)^*$
- $S \in V$ ist das Startsymbol (alternativ Startvariable).

Wiederholung: Die Chomsky-Hierarchie

Definition

Sei $G = (V, \Sigma, P, S)$ eine Grammatik.

- G ist immer vom Typ 0.
- G ist vom Typ 1 (alternativ kontextsensitiv), wenn: für alle $\ell \to r \in P$ ist $|\ell| \leq |r|$.
- G ist vom Typ 2 (alternativ kontextfrei), wenn: G ist vom Typ 1 und für alle $\ell \to r \in P$ ist $\ell \in V$.
- G ist vom Typ 3 (alternativ regulär), wenn: G ist vom Typ 2 und für alle $A \to r \in P$ gilt $r = a$ oder $r = aA'$ für $a \in \Sigma, A' \in V$ (d.h. die rechten Seiten sind Wörter aus $\Sigma \cup \Sigma V$).

Typ 3 (regulär) vs. Typ 2 (kontextfrei)

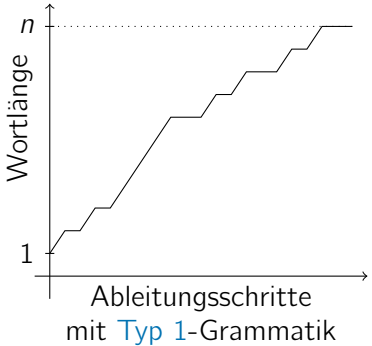
- Typ 3- und Typ 2-Produktionen sind immer von der Form $A \to r$, wobei $r \neq \varepsilon$.
- Bei Typ 3 muss zusätzlich r von der Form $a \in \Sigma$ oder $aA' \in \Sigma V$ sein.

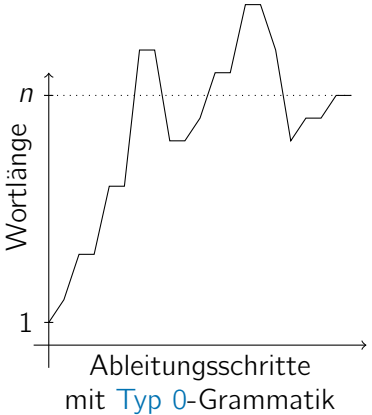
Typ 2 (kontextfrei) vs. Typ 1 (kontextsensitiv)

- Typ 2-Produktionen $A \to r$ sind immer auf ein Vorkommen von A anwendbar.
- Typ 1-Produktionen können solche Ersetzungen auf einen Kontext einschränken. Sie erlauben Regeln von der Form $uAv \to urv$, die die Ersetzung von A durch r nur erlauben, wenn A durch u und v umrahmt ist.

Typ 1 (kontextsensitiv) vs. Typ 0

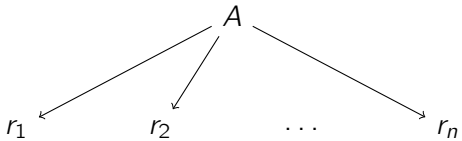
Ableitung eines Wortes der Länge n





Wiederholung: Syntaxbäume

- Ein Syntaxbaum stellt dar, wie ein Wort entsteht.
- Die Anwendung von $A \to r_1 r_2 \dots r_n$ wird durch den Teilbaum



dargestellt.

Definition eines Syntaxbaums

Definition

Sei $G = (V, \Sigma, P, S)$ eine Typ 2-Grammatik und $S = w_0 \Rightarrow_G \cdots \Rightarrow_G w_n$ eine Ableitung von $w_n \in \Sigma^*$.

Der Syntaxbaum zur Ableitung wird wie folgt erstellt:

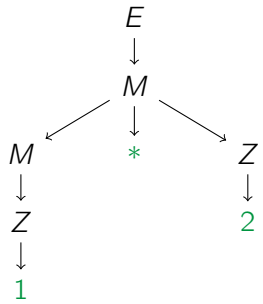
- Markiere die Wurzel des Baums mit S.
- Wenn $w_i \Rightarrow w_{i+1}$, $w_i = uAv$ und $w_{i+1} = urv$ (Produktion $A \to r$ verwendet), dann erzeuge im Syntaxbaum $|r|$ viele Knoten als Kinder des mit A markierten Knotens. Markiere die Kinder mit den Symbolen aus r (von links nach rechts).

Das Ergebnis des Syntaxbaums ist das Wort w_n an den Blättern, von links nach rechts gelesen.

Beispiel für ein Syntaxbaum

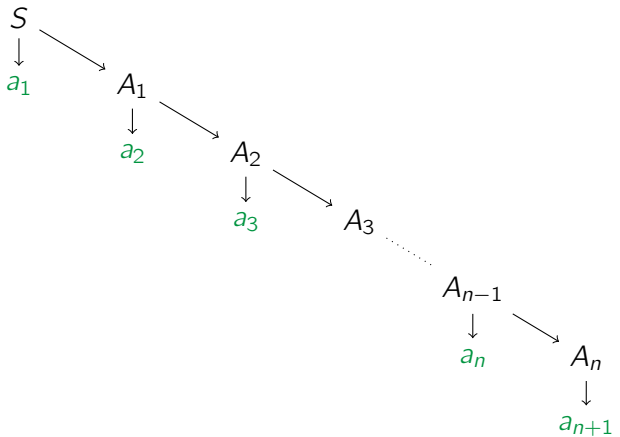
$G = (\{E, M, Z\}, \{+, *, 1, 2, (,)\}, P, E)$ mit
$P = \{E \to M,\ E \to E + M,\ M \to Z,\ M \to M * Z,\ Z \to 1,\ Z \to 2,\ Z \to (E)\}$

Beachte: Beide Ableitungen
$E \Rightarrow M \Rightarrow M * Z \Rightarrow Z * Z \Rightarrow 1 * Z \Rightarrow 1 * 2$
und
$E \Rightarrow M \Rightarrow M * Z \Rightarrow M * 2 \Rightarrow Z * 2 \Rightarrow 1 * 2$
haben denselben Syntaxbaum (rechts).



Syntaxbäume bei Typ 3-Grammatiken

Syntaxbäume bei Typ 3-Grammatiken sind immer listenartig:



Links- und Rechtsableitungen

- Linksableitung: Ersetze immer die linkeste Variable der Satzform.
- Rechtsableitung: Ersetze immer die rechteste Variable der Satzform.

Beispiele:

$$
\begin{aligned}
E &\Rightarrow E + M \\
&\Rightarrow M + M \\
&\Rightarrow M * Z + M \\
&\Rightarrow Z * Z + M \\
&\Rightarrow 1 * Z + M \\
&\Rightarrow 1 * 2 + M \\
&\Rightarrow 1 * 2 + Z \\
&\Rightarrow 1 * 2 + 3
\end{aligned}
$$

$$
\begin{aligned}
E &\Rightarrow E + M \\
&\Rightarrow E + Z \\
&\Rightarrow E + 3 \\
&\Rightarrow M + 3 \\
&\Rightarrow M * Z + 3 \\
&\Rightarrow M * 2 + 3 \\
&\Rightarrow Z * 2 + 3 \\
&\Rightarrow 1 * 2 + 3
\end{aligned}
$$

Links- und Rechtsableitungen

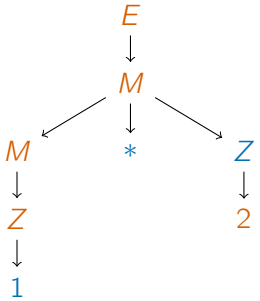
Satz

Sei G eine Typ 2-Grammatik und $w \in L(G)$.
Dann gibt es eine Linksableitung und eine Rechtsableitung von w.

Beweis Da $w \in L(G)$, gibt es irgendeine Ableitung von w.
Konstruiere den Syntaxbaum zu dieser Ableitung.
Lies eine Links- bzw. Rechtsableitung am Syntaxbaum ab. □

Beispiel für das Ablesen einer Linksableitung

Syntaxbaum:



Linksableitung:

$E \Rightarrow M \Rightarrow M * Z \Rightarrow Z * Z \Rightarrow 1 * Z \Rightarrow 1 * 2$

Erweiterte Backus-Naur-Form

Definition

Für Typ 2-Grammatiken erlauben wir abkürzende Schreibweise für die Menge der Produktionen P:

1. $A \to w_1 \mid w_2 \mid \cdots \mid w_n$ steht für $A \to w_1$, $A \to w_2$, $\ldots$, $A \to w_n$.
2. $A \to u[v]w$ steht für die beiden Produktionen $A \to uvw$ und $A \to uw$ (d.h. $[v]$ meint, dass v optional ist).
3. $A \to u\{v\}w$ steht für $A \to uw$ oder $A \to uBw$ mit $B \to v \mid vB$ (d.h. $\{v\}$ meint, dass v beliebig oft wiederholt werden kann).

Grammatiken, die diese Notation verwenden, nennen wir Grammatiken in erweiterter Backus-Naur-Form (EBNF).

Anwendungen von kontextfreien Grammatiken

- Kontextfreie Grammatiken werden zur syntaktischen Analyse von Programmiersprachen und Domain Specific Languages verwendet.
- Tools wie yacc (für C/C++), ANTLR (für Java) und PLY (für Python) generieren syntaktische Analyser („Parser“) aus Grammatiken.
  Beispiel für eine ANTLR-Grammatik (Quelle: www.antlr.org):

```
prog:   (expr NEWLINE)* ;
expr:   expr ('*' | '/') expr
    |   expr ('+' | '-') expr
    |   INT
    |   '(' expr ')';
```

- Viele Fragestellungen sind jedoch kontextsensitiv oder Typ 0.
  Praktisches Vorgehen: Nutze Typ 2-Sprache und Nebenbedingungen (z.B. Syntax als kontextfreie Grammatik und Nebenbedingungen, die prüfen, dass alle Variablen deklariert wurden).

Chomsky-Hierarchie: Teilmengenbeziehungen

Aus der Definition der Typ i-Sprachen folgt:

$$\text{Typ 3-Sprachen} \subseteq \text{Typ 2-Sprachen} \subseteq \text{Typ 1-Sprachen} \subseteq \text{Typ 0-Sprachen}$$

Es gilt sogar:

$$\text{Typ 3-Sprachen} \subset \text{Typ 2-Sprachen} \subset \text{Typ 1-Sprachen} \subset \text{Typ 0-Sprachen}$$

Trennende Beispiele sind:

- $\{a^n b^n \mid n \in \mathbb{N}_{>0}\}$ ist von Typ 2, aber nicht von Typ 3.
- $\{a^n b^n c^n \mid n \in \mathbb{N}_{>0}\}$ ist von Typ 1, aber nicht von Typ 2.
- Das sogenannte Halteproblem ist von Typ 0, aber nicht von Typ 1.

(Beweise folgen im Laufe der Veranstaltung.)

Beachte: Es gibt auch Sprachen, die nicht Typ 0 sind.
Das Komplement vom Halteproblem ist eine solche Sprache.

Abgeschlossenheit von Sprachen

Eine Klasse $\mathcal{K}$ von Sprachen (d.h. eine Menge von Mengen)
ist abgeschlossen bezüglich

- Vereinigung g.d.w. aus $L_1, L_2 \in \mathcal{K}$ folgt stets $L_1 \cup L_2 \in \mathcal{K}$
- Schnittbildung g.d.w. aus $L_1, L_2 \in \mathcal{K}$ folgt stets $L_1 \cap L_2 \in \mathcal{K}$
- Komplementbildung g.d.w. aus $L \in \mathcal{K}$ folgt stets $\overline{L} \in \mathcal{K}$
- Produktbildung g.d.w. aus $L_1, L_2 \in \mathcal{K}$ folgt stets $L_1 L_2 \in \mathcal{K}$.

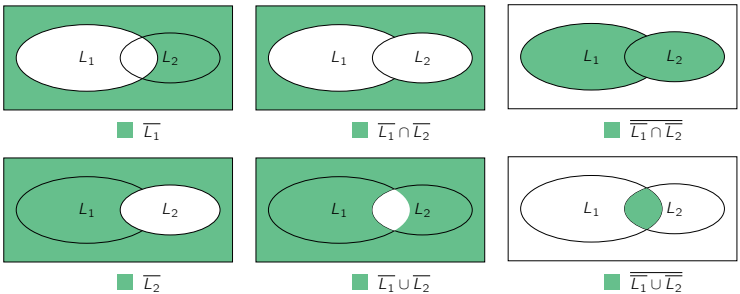
Wir werden im Laufe der Veranstaltung untersuchen, ob die Typ i-Sprachen abgeschlossen bezüglich obiger Operationen sind.

Abgeschlossenheit: Eigenschaften

Satz

Sei die Klasse von Sprachen $\mathcal{K}$ abgeschlossen bezüglich Komplementbildung.
Dann ist $\mathcal{K}$ abgeschlossen bezüglich Schnittbildung g.d.w. $\mathcal{K}$ abgeschlossen bezüglich Vereinigung ist.

Beweis Das gilt, da $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ und $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. □



Entscheidbarkeit

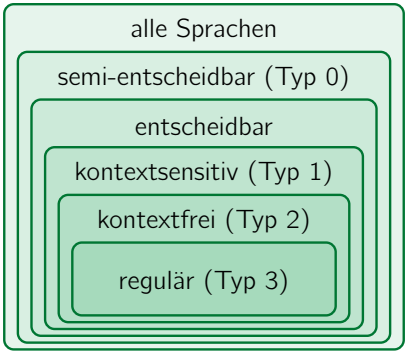
Definition

Eine Sprache L ist entscheidbar, wenn es einen Algorithmus gibt, der bei Eingabe eines Wortes w in endlicher Zeit feststellt, ob $w \in L$ gilt oder nicht.

Eigenschaften der Typ i-Sprachen

- Alle Typ 1, 2, 3-Sprachen sind entscheidbar.
- Es gibt Typ 0-Sprachen, die nicht entscheidbar sind.
- Alle Typ 0-Sprachen sind semi-entscheidbar (alternativ rekursiv aufzählbar):
  Für jede Typ 0-Sprache L gibt es einen Algorithmus, der bei Eingabe eines Wortes $w \in L$ in endlicher Zeit feststellt, dass $w \in L$ gilt, und bei einem Wort $w \notin L$ entweder feststellt, dass $w \notin L$ gilt, **oder nicht terminiert.**
- Es gibt auch Sprachen, die nicht semi-entscheidbar sind:
  - Die Menge der Typ 0-Grammatiken ist abzählbar
    (jede Grammatik hat eine endliche Beschreibung,
    d.h. Grammatiken können der Größe nach aufgezählt werden).
  - Die Menge aller Sprachen $= \mathcal{P}(\Sigma^*)$ ist überabzählbar.
  - Wir betrachten mit den Typ i-Grammatiken einen sehr kleinen Teil aller Sprachen.

Übersicht über die Sprachen



Entscheidungsprobleme

- Das Wortproblem für Typ i-Grammatiken ist die Frage, ob für eine gegebene Typ i-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$ $w \in L(G)$ gilt oder nicht.
- Das Leerheitsproblem für Typ i-Grammatiken ist die Frage, ob für eine gegebene Typ i-Grammatik G die Gleichheit $L(G) = \emptyset$ gilt.
- Das Endlichkeitsproblem für Typ i-Grammatiken ist die Frage, ob für eine gegebene Typ i-Grammatik G die Ungleichheit $|L(G)| < \infty$ gilt.
- Das Schnittproblem für Typ i-Grammatiken ist die Frage, ob für gegebene Typ i-Grammatiken G_1, G_2 gilt: $L(G_1) \cap L(G_2) = \emptyset$.
- Das Äquivalenzproblem für Typ i-Grammatiken ist, die Frage, ob für gegebene Typ i-Grammatiken G_1, G_2 gilt: $L(G_1) = L(G_2)$.

Probleme vs. Sprachen

- (Entscheidungs-)Problem: Funktion von Σ^* nach $\{\text{ja}, \text{nein}\}$
- (Formale) Sprache: Menge $\subseteq \Sigma^*$
- Die beiden Begriffe werden synonym verwendet.
  - ja (als Antwort auf ein Problem) = Element (einer Sprache)
  - nein (als Antwort auf ein Problem) = nicht Element (einer Sprache)
- Beispiele:
  - Das Primzahlproblem ist die Sprache L,
    die alle (in Dezimalnotation kodierten) Primzahlen enthält.
    D.h. $L = \{2, 3, 5, 7, 11, \dots\}$.
  - Das Leerheitsproblem für Typ i-Grammatiken ist die Sprache,
    die alle (als Wörter kodierten) Typ i-Grammatiken G enthält, für die $L(G) = \emptyset$ gilt.