Hochschule RheinMain

Automatentheorie und Formale Sprachen

für die Studiengänge
- Angewandte Informatik
- Informatik - Technische Systeme

02 Chomsky-Grammatiken

Prof. Dr. David Sabel
Sommersemester 2025

Stand der Folien: 20. Mai 2025

Formale Sprachen darstellen

Hochschule RheinMain

- Sei Σ ein Alphabet.
- Eine Sprache über Σ ist eine Teilmenge von Σ^*.
- Z.B. für $\Sigma = \{(,), +, -, *, /, a\}$ sei
  L_{ArEx} die Sprache aller korrekt geklammerten Ausdrücke

  Z.B. $((a + a) - a) * a \in L_{ArEx}$ aber $(a-) + a) \notin L_{ArEx}$
- Unsere bisherigen Operationen auf Sprachen (Mengen) können das nicht darstellen

Benötigt: Formalismus, um L_{ArEx} zu beschreiben

Formale Sprachen darstellen (2)

Hochschule RheinMain

Anforderungen:
- **Endliche** Beschreibung
- Sprache selbst muss aber auch unendlich viele Objekte erlauben

Zwei wesentliche solchen Formalismen sind
- Grammatiken
- Automaten

Grammatiken

Hochschule RheinMain

Grammatik für einen sehr kleinen Teil der deutschen Sprache:

<Satz> → <Subjekt><Prädikat><Objekt>
<Subjekt> → <Artikel><Attribut><Nomen>
<Objekt> → <Artikel><Attribut><Nomen>
<Artikel> → ε
<Artikel> → der
<Artikel> → das
<Attribut> → <Adjektiv>
<Attribut> → <Adjektiv><Attribut>
<Adjektiv> → kleine
<Adjektiv> → große
<Adjektiv> → nette
<Adjektiv> → blaue
<Nomen> → Mann
<Nomen> → Auto
<Prädikat> → fährt
<Prädikat> → liebt

Grammatiken

Hochschule RheinMain

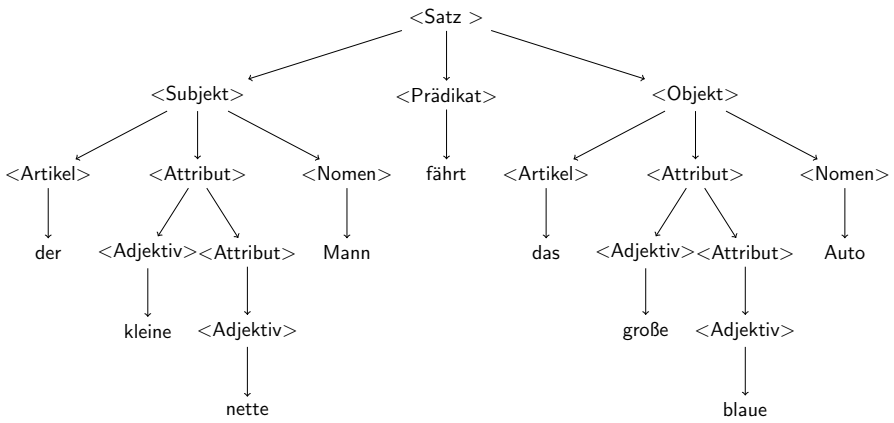
- Endliche Menge von Regeln „linke Seite → rechte Seite"
- Symbole in spitzen Klammern wie <Artikel> sind Variablen, d.h. sie sind Platzhalter, die weiter ersetzt werden müssen.
- Ableiten: Ersetze Vorkommen von linken Seiten durch rechte Seiten
- Z.B. kann

  „der kleine nette Mann fährt das große blaue Auto"

  durch die obige Grammatik abgeleitet werden

Syntaxbaum zum Beispiel

Hochschule RheinMain



Definition einer Grammatik

Hochschule RheinMain

Definition (Grammatik)

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

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

Manchmal notieren wir nur P (wenn klar ist, was Variablen, Zeichen, Startsymbol sind)

Beispiel für eine Grammatik

Hochschule RheinMain

$G = (V, \Sigma, P, E)$ mit

$$V = \{E, M, Z\},$$
$$\Sigma = \{+, *, 1, 2, (,)\}$$
$$\begin{aligned} P = \{ E &\to M, \\ E &\to E + M, \\ M &\to Z, \\ M &\to M * Z, \\ Z &\to 1, \\ Z &\to 2, \\ Z &\to (E)\} \end{aligned}$$

Ableitung

Hochschule RheinMain

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

Ableitungsschritt $\Rightarrow_G$

Für Satzformen u, v (d.h. Wörter aus $(V \cup \Sigma)^*$) sagen wir:

u geht unter Grammatik G unmittelbar in v über, $u \Rightarrow_G v$, wenn

$$u = w_1 \ell w_2 \text{ und } w_1 r w_2 = v \text{ mit } (\ell \to r) \in P$$

- Wenn G klar ist, schreiben wir $u \Rightarrow v$ statt $u \Rightarrow_G v$
- $\Rightarrow_G^*$ sei die reflexiv-transitive Hülle von $\Rightarrow_G$

Ableitung

Eine Folge $(w_0, w_1, \ldots, w_n)$ mit $w_0 = S$, $w_n \in \Sigma^*$ und $w_{i-1} \Rightarrow w_i$ für $i = 1, \ldots, n$ heißt Ableitung von w_n. Statt $(w_0, \ldots, w_n)$ schreiben wir auch $w_0 \Rightarrow \ldots \Rightarrow w_n$

Beispiel

Hochschule RheinMain

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

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

Eine Ableitung von (2+1) * (2+2):

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

Beispiel: Ableitungen sind nicht eindeutig

Hochschule RheinMain

Ableitung von letzter Folie (keine Linksableitung):

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

Linksableitung: ersetzt immer das linkeste Nichtterminal

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

Quiz 1

Hochschule RheinMain

Sei $G = (\{A, B, C\}, \{d, e\}, P, A)$ mit

$P = \{A \to BBCC, B \to d, C \to e, dC \to Cd, ed \to \varepsilon\}$

Welches Wort liegt in $L(G)$?

a) ε

b) $BBdd$

c) e

arsnova.hs-rm.de
6750 1376

Nichtdeterminismus beim Ableiten

Hochschule RheinMain

Für eine Satzform u kann es verschiedene Satzformen v_i geben mit $u \Rightarrow_G v_i$.

Quellen des Nichtdeterminismus:

- Wähle, **welche Produktion** $\ell \to r$ aus P angewendet wird
- Wähle die **Position des Teilworts** ℓ in u, das durch r ersetzt wird.

Aber: Es gibt **nur endliche viele** v_i für jeden Schritt!

Erzeugte Sprache

Hochschule RheinMain

Erzeugte Sprache einer Grammatik

Die von einer Grammatik $G = (V, \Sigma, P, S)$ **erzeugte Sprache** $L(G)$ ist

$$L(G) := \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}.$$

Quiz 2

Hochschule RheinMain

Sei $G_1 = (\{S\}, \{a\}, \{S \to aS\}, S)$ und
$G_2 = (\{S'\}, \{a, b\}, \{S' \to aS', S' \to b\}, S')$.

Welche Sprache erzeugt G_1?
Welche Sprache erzeugt G_2?

arsnova.hs-rm.de
6750 1376

Antworten

Hochschule RheinMain

$G_1 = (\{S\}, \{a\}, \{S \to aS\}, S)$

- $S \Rightarrow aS \Rightarrow aaS \Rightarrow \ldots$ endet nie
- Andere Ableitungen gibt es nicht
- Daher sind keine Wörter aus $\{a\}^*$ ableitbar

$G_2 = (\{S'\}, \{a, b\}, \{S' \to aS', S' \to b\}, S')$

$$\begin{array}{ccccccccc} S' & \Longrightarrow & aS' & \Longrightarrow & aaS' & \Longrightarrow & aaaS' & \Longrightarrow & aaaaS' & \Longrightarrow & \cdots \\ \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow & & \\ b & & ab & & aab & & aaab & & aaaab & & \end{array}$$

- Für alle $i \in \mathbb{N}_0$ gilt $S \Rightarrow^i a^i S \Rightarrow a^i b$

Hochschule RheinMain

CHOMSKY-HIERARCHIE

- Grammatiken der Typen 0,1,2,3
- Typ-i Sprachen

Die Chomsky-Hierarchie

Hochschule RheinMain

Noam Chomsky teilte die Grammatiken in Typen 0 bis 3:

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

G ist vom Typ 0

G ist automatisch vom Typ 0 (alles erlaubt)

G ist vom Typ 1 (kontextsensitive Grammatik), wenn ...

für alle $(\ell \to r) \in P$: $|\ell| \leq |r|$ (keine verkürzende Produktionen)

G ist vom Typ 2 (kontextfreie Grammatik), wenn ...

G ist vom Typ 1 und für alle $(\ell \to r) \in P$ gilt: $\ell = A \in V$ (linke Seite = 1 Variable)

G ist vom Typ 3 (reguläre Grammatik), wenn ...

G ist vom Typ 2 und für alle $(A \to r) \in P$ gilt: $r = a$ oder $r = aA'$ für $a \in \Sigma, A' \in V$ (rechte Seite = Wort aus $(\Sigma \cup (\Sigma V))$)

Typ i-Sprachen

Hochschule RheinMain

Definition

Für $i = 0, 1, 2, 3$ nennt man eine formale **Sprache** $L \subseteq \Sigma^*$ **vom Typ** i, falls es eine Typ i-Grammatik G gibt, sodass $L(G) = L$ gilt.

Spricht man von **dem Typ einer formalen Sprache**, so ist stets der größtmögliche Typ gemeint.

Bemerkung: Die Definition erlaubt Aussagen der Form:

> *Typ $i+k$-Sprachen sind eine Teilmenge der Typ i-Sprachen, da jede Typ $i+k$-Grammatik auch eine Typ i-Grammatik ist.*

Beispiele

Hochschule RheinMain

- $G_1 = (\{S\}, \{a, b\}, \{S \to aS, S \to b\}, S)$ ist regulär (Typ 3)
- $G_2 = (\{E, M, Z\}, \{+, *, 1, 2, (,)\}, P, E)$ mit
  $P = \{E \to M,\ \ E \to E + M,\ \ M \to Z,\ \ M \to M * Z,$
  $Z \to 1,\ \ Z \to 2,\ \ Z \to (E)\}$ ist kontextfrei (Typ 2)
- $G_3 = (\{S, B, C\}, \{a, b, c\}, P, S)$ mit
  $P = \{S \to aSBC, S \to aBC, CB \to BC, aB \to ab,$
  $bB \to bb, bC \to bc, cC \to cc\}$ ist kontextsensitiv (Typ 1)
- $G_4 = (\{S, T, A, B, \$\}, \{a, b\}, P, S)$ mit
  $P = \{S \to \$T\$, T \to aAT, T \to bBT, T \to \varepsilon, \$a \to a\$,$
  $\$b \to b\$, Aa \to aA, Ab \to bA, Ba \to aB, Bb \to bB,$
  $A\$ \to \$a, B\$ \to \$b, \$\$ \to \varepsilon\}$ ist vom Typ 0

Quiz 3

Hochschule RheinMain

Sei $G = (\{A, B, C\}, \{d, e\}, P, A)$ mit

$$P = \{A \to BBCC, B \to d, C \to e, dC \to Cd\}$$

Welchen Typ hat G?

a) 0

b) 1

c) 2

d) 3

arsnova.hs-rm.de
6750 1376

Quiz 4

Hochschule RheinMain

Sei $G = (\{A, B, C\}, \{d, e\}, P, A)$ mit

$$P = \{A \to BBCC, B \to d, C \to e, dC \to Cd\}$$

Welchen Typ hat $L(G)$?

a) 0

b) 1

c) 2

d) 3

d) stimmt, da $L(G) = \{ddee, dede, deed, edde, eded, eedd\}$ und es reguläre Grammatik gibt, die $L(G)$ erzeugt.

arsnova.hs-rm.de
6750 1376

Beispiel (kontextsensitive Grammatik)

Hochschule RheinMain

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

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

Beispiel-Ableitung:

$$\begin{aligned} S &\Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaSBCBCBC \Rightarrow aaaaBCBCBCBC \\ &\Rightarrow aaaabCBCBCBC \Rightarrow aaaabBCCBCBC \Rightarrow aaaabbCCBCBC \\ &\Rightarrow aaaabbCBCCBC \Rightarrow aaaabbBCCCBC \Rightarrow aaaabbBCCBCC \\ &\Rightarrow aaaabbBCBCCC \Rightarrow aaaabbBBCCCC \Rightarrow aaaabbbBCCCC \\ &\Rightarrow aaaabbbbCCCC \Rightarrow aaaabbbbcCCC \Rightarrow aaaabbbbccCC \\ &\Rightarrow aaaabbbbccCC \Rightarrow aaaabbbbcccc \end{aligned}$$

Steckengebliebene Folge von Ableitungsschritten:

$$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabcBC$$

Grammatik erzeugt $\{a^n b^n c^n \mid n \in \mathbb{N}\}$

Hochschule RheinMain

Satz

$L(G) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ für $G = (\{S, B, C\}, \{a, b, c\}, P, S)$ mit
$P = \{S \to aSBC, S \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\}$

„$\supseteq$“: Zeige $a^n b^n c^n \in L(G)$ für alle $n \in \mathbb{N}$

- Wende $n-1$ mal $S \to aSBC$ und dann einmal $S \to aBC$ an:
  $S \Rightarrow^* a^{n-1} S (BC)^{n-1} \Rightarrow a^n (BC)^n$
- Wende $CB \to BC$ solange an, bis es kein Teilwort CB mehr gibt.
  $a^n (BC)^n \Rightarrow^* a^n B^n C^n$
- Wende $aB \to ab$ und anschließend $n-1$ mal $bB \to bb$ an.
  $a^n B^n C^n \Rightarrow a^n b B^{n-1} C^n \Rightarrow^* a^n b^n C^n$
- Wende einmal $bC \to bc$ und anschließend $n-1$ mal $cC \to cc$ an
  $a^n b^n C^n \Rightarrow a^n b^n c C^{n-1} \Rightarrow^* a^n b^n c^n$

Zusammensetzen aller Ableitungsschritte zeigt $S \Rightarrow^* a^n b^n c^n$.

Grammatik erzeugt $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ (2)

Hochschule RheinMain

Satz

$L(G) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ für $G = (\{S, B, C\}, \{a, b, c\}, P, S)$ mit
$P = \{S \to aSBC, S \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\}$

„$\subseteq$“: Zeige, dass alle von G erzeugten Wörter von der Form $a^n b^n c^n$ sind.

- Für $S \Rightarrow_G^* u$ mit u Satzform zeigen die Regeln:
  $\#_a(u) = \#_b(u) + \#_B(u) = \#_c(u) + \#_C(u)$
- Für $S \Rightarrow_G^* w$ mit $w \in \{a, b, c\}^*$ gilt: a's werden ganz links erzeugt, d.h. $w = a^n w'$ mit $w' \in \{b, c\}^*$ und $n = \#_b(w') = \#_c(w')$
- Es gilt $w' = bw_1$, da jedes auf a folgende Symbol durch $aB \to ab$ erzeugt wird und die Regeln keine Terminalsymbole vertauschen.

Grammatik, die $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ erzeugt (3)

Hochschule RheinMain

Satz

$L(G) = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ für $G = (\{S, B, C\}, \{a, b, c\}, P, S)$ mit
$P = \{S \to aSBC, S \to aBC, CB \to BC, aB \to ab, bB \to bb, bC \to bc, cC \to cc\}$

„$\subseteq$“: Zeige, dass alle von G erzeugten Wörter von der Form $a^n b^n c^n$ sind.

- ...
- Ebenso können die Terminalsymbole des Wortes $w' \in \{b, c\}^*$ nur durch $bB \to bb$, $bC \to bc$ und $cC \to cc$ erzeugt worden sein.
  Diese Produktionen erlauben nur einen Wechsel von b zu c und keine Wechsel von c zu b. Auch ein Umordnen der Terminalsymbole ist nicht möglich
  (da es keine Produktion dafür gibt).
- Daher gilt $w' = b^i c^j$ und mit $n = \#_b(w') = \#_c(w')$ sogar $w' = b^n c^n$. □

Beispiel einer Typ 0-Grammatik

Hochschule RheinMain

Grammatik $G = (\{S, T, A, B, \$\}, \{a, b\}, P, S)$ mit

$$P = \{S \to \$T\$,\ T \to aAT,\ T \to bBT,\ T \to \varepsilon,\ \$a \to a\$,\ \$b \to b\$,\ Aa \to aA,\\ Ab \to bA,\ Ba \to aB,\ Bb \to bB,\ A\$ \to \$a,\ B\$ \to \$b,\ \$\$ \to \varepsilon\}$$

Eine Ableitung:

$$\begin{aligned} S &\Rightarrow \$T\$ \Rightarrow \$aAT\$ \Rightarrow \$aAaAT\$ \Rightarrow \$aAaAbBT\$\\ &\Rightarrow \$aAaAbB\$ \Rightarrow \$aaAAbB\$ \Rightarrow \$aaAbAB\$ \Rightarrow \$aabAAB\$\\ &\Rightarrow \$aabAA\$b \Rightarrow \$aabA\$ab \Rightarrow \$aab\$aab \Rightarrow a\$ab\$aab\\ &\Rightarrow aa\$b\$aab \Rightarrow aab\$\$aab \Rightarrow aabaab \end{aligned}$$

Beachte: $L(G) = \{ww \mid w \in \{a, b\}^*\}$ und $L(G)$ ist Typ 1-Sprache

Beispiel einer Typ 0-Grammatik (2)

Hochschule RheinMain

Grammatik $G = (\{S, T, A, B, \$\}, \{a, b\}, P, S)$ mit

$$P = \{S \to \$T\$,\ T \to aAT,\ T \to bBT,\ T \to \varepsilon,\ \$a \to a\$,\ \$b \to b\$,\ Aa \to aA,\\ Ab \to bA,\ Ba \to aB,\ Bb \to bB,\ A\$ \to \$a,\ B\$ \to \$b,\ \$\$ \to \varepsilon\}$$

Begründung dafür, dass $L(G) = \{ww \mid w \in \{a, b\}^*\}$ gilt:

- Mit $S \to \$T\$$ wird zunächst eine Umrahmung mit \$\$ erzeugt
- Mit $T \to aAT$, $T \to bBT$, $T \to \varepsilon$ wird ein Wort aus 2er Blöcken aA und bB erzeugt
- Mit $Aa \to aA, Ab \to bA, Ba \to aB, Bb \to bB$ werden A's und B's bis vor \$ geschoben
- Mit $A\$ \to \$a$ und $B\$ \to \$b$ werden die A's und B's in a's und b's verwandelt, indem sie über das rechte \$ hüpfen.
- Mit $\$a \to a\$$, $\$b \to b\$$ wird das linke \$ zum rechten geschoben, mit $\$\$ \to \varepsilon$ werden sie dann eliminiert.
- Bei allen Schritten wird die relative Lage aller a zu b sowie aller A zu B nicht geändert.

Hochschule RheinMain

ε-PRODUKTIONEN

- in Typ 1,2,3-Grammatiken
- in Typ 2,3-Grammatiken

ε-Regel für Typ 1,2,3-Grammatiken

Hochschule RheinMain

- Das leere Wort ε kann bisher nicht für Typ 1,2,3 Grammatiken erzeugt werden:

  Produktion $S \to \varepsilon$ erfüllt die Typ 1-Bedingung $|S| \leq |\varepsilon|$ nicht. Daher Sonderregel:

ε-Regel für Typ 1-Grammatiken

Eine Grammatik $G = (V, \Sigma, P, S)$ vom Typ 1 darf eine Produktion $(S \to \varepsilon) \in P$ enthalten, vorausgesetzt, dass keine rechte Seite einer Produktion in P, die Variable S enthält.

Sonderregel erlaubt nicht:

$G = (\{S\}, \{a\}, \{S \to \varepsilon, S \to aSa\}, S)$

Sonderregel erlaubt:

$G = (\{S', S\}, \{a\}, \{S' \to \varepsilon, S' \to aSa, S' \to aa, S \to aSa, S \to aa\}, S')$

Leeres Wort hinzufügen geht mit Sonderregel immer

Hochschule RheinMain

Satz

Sei $G = (V, \Sigma, P, S)$ vom Typ $i \in \{1, 2, 3\}$ mit $\varepsilon \notin L(G)$. Sei $S' \notin V$.
Dann erzeugt $G' = (V \cup \{S'\}, \Sigma, P \cup \{S' \to \varepsilon\} \cup \{S' \to r \mid S \to r \in P\}, S')$ die Sprache $L(G') = L(G) \cup \{\varepsilon\}$ und
G' erfüllt die ε-Regel für Typ 1,2,3-Grammatiken und G' ist vom Typ i.

Beweis:

- Da S' neu, kommt S' auf keiner rechten Seite vor.
- Da $S \to r \in P$ vom Typ i, sind auch $S' \to r$ vom Typ i
- Da $S' \Rightarrow \varepsilon$, gilt $\varepsilon \in L(G')$
- Für $w \neq \varepsilon$ gilt: $S \Rightarrow_G^* w$ g.d.w. $S' \Rightarrow_{G'}^* w$
  Der jeweils erste Ableitungsschritt muss ausgetauscht werden,
  d.h. $S \Rightarrow_G r$ vs. $S' \Rightarrow_{G'} r$

ε-Produktionen für Typ 2- und Typ 3-Grammatiken

Hochschule RheinMain

Sonderregel für Typ 2- und Typ 3-Grammatiken:

ε-Produktionen in kontextfreien und regulären Grammatiken

In Grammatiken des Typs 2 und des Typs 3 erlauben wir Produktionen der Form $A \to \varepsilon$ (sogenannte ε-Produktionen).

Das ist keine echte Erweiterung, denn:

Satz (Entfernen von ε-Produktionen)

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie (bzw. reguläre) Grammatik mit $\varepsilon \notin L(G)$. Dann gibt es eine kontextfreie (bzw. reguläre) Grammatik G' mit $L(G) = L(G')$ und G' enthält keine ε-Produktionen.

Kontextfrei vs. kontextsensitiv

Hochschule RheinMain

- Kontextfreie Produktionen $A \to r$ sind immer auf ein Vorkommen von A anwendbar.
- Kontextsensitive Produktionen können solche Ersetzungen

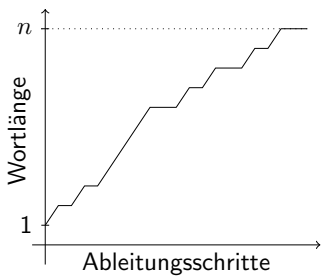
  auf einen Kontext einschränken

  und erlauben Regeln $uAv \to urv$, die die Ersetzung von A durch r nur erlauben, wenn A durch u und v umrahmt ist.

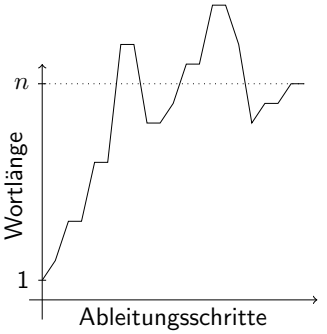
Typ 0 vs. Typ 1

Hochschule RheinMain

Ableitung eines Wortes der Länge n



mit Typ 1-Grammatik



mit Typ 0-Grammatik

Hochschule RheinMain

SYNTAXBÄUME, ABLEITUNGEN, MEHRDEUTIGE GRAMMATIKEN, BACKUS-NAUR-FORM

Syntaxbäume

Hochschule RheinMain

Definition (Syntaxbaum)

Sei $G = (V, \Sigma, P, S)$ eine Typ 2-Grammatik und

$$S \Rightarrow w_0 \Rightarrow \ldots \Rightarrow w_n$$

eine Ableitung von $w_n \in \Sigma^*$.

Der **Syntaxbaum** zur Ableitung wird wie folgt erstellt:

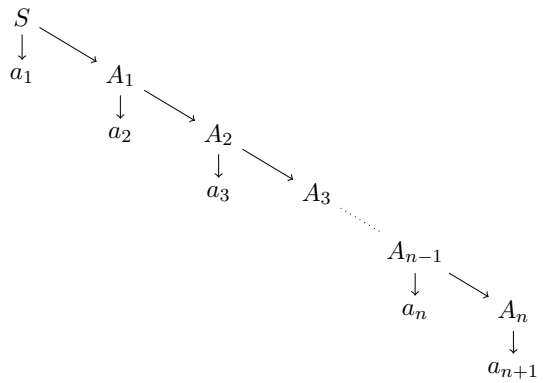
- Die Wurzel des Baums ist mit S markiert.
- Wenn $w_i \Rightarrow w_{i+1}$, $w_i = uAv$ und $w_{i+1} = urv$ (Produktion $A \to r$ verwendet), dann erzeuge im Syntaxbaum $|r|$ viele Knoten als Kinder des mit A markierten Knotens. Markiere die Kinder mit den Symbolen aus r (in der Reihenfolge von links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

Die Blätter sind daher genau mit dem Wort w_n markiert.

Syntaxbäume bei Typ 3-Grammatiken

Hochschule RheinMain

Sind immer Listenartig:



Beispiel

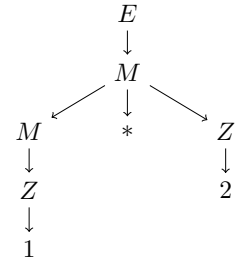
Hochschule RheinMain

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

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.



Links- und Rechtsableitungen

Hochschule RheinMain

- **Linksableitung**: Ersetze immer das linkeste Nichtterminal der Satzform.
- **Rechtsableitung**: Ersetze immer das rechteste Nichtterminal der Satzform.

Beispiele:

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

Links- und Rechtsableitungen (2)

Hochschule RheinMain

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 Syntaxbaum zu dieser Ableitung.
- Lese Links- bzw. Rechtsableitung am Syntaxbaum ab.

Mehrdeutige Grammatiken

Hochschule RheinMain

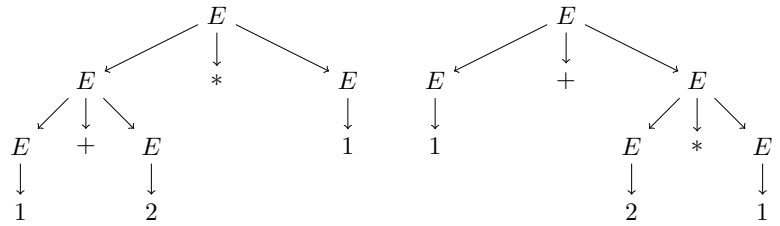
Beispiel:

$$(E, \{*, +, 1, 2\}, \{E \to E * E,\ \ E \to E + E,\ \ E \to 1,\ \ E \to 2\}, E)$$

Zwei Ableitungen für $1 + 2 * 1$:

- $E \Rightarrow \color{red}{E} * E \Rightarrow E + E * E \Rightarrow 1 + \color{red}{E} * E \Rightarrow 1 + 2 * \color{red}{E} \Rightarrow 1 + 2 * 1$
- $E \Rightarrow E + \color{red}{E} \Rightarrow E + E * E \Rightarrow 1 + \color{red}{E} * E \Rightarrow 1 + 2 * \color{red}{E} \Rightarrow 1 + 2 * 1.$

Syntaxbäume dazu:



Mehrdeutige Grammatiken (2)

Hochschule RheinMain

Mehrdeutige Grammatik

Eine Typ 2-Grammatik ist mehrdeutig, wenn es verschieden strukturierte Syntaxbäume für dasselbe Wort w gibt.

Inhärent mehrdeutige Sprache

Eine Typ 2-Sprache ist inhärent mehrdeutig, wenn es nur mehrdeutige Grammatiken gibt, die diese Sprache erzeugen.

Die Sprache

$$\{a^m b^m c^n d^n \mid m, n \in \mathbb{N}\} \cup \{a^m b^n c^n d^m \mid m, n \in \mathbb{N}\}$$

ist inhärent mehrdeutig (Beweis z.B. in Hopcroft, Motwani, Ullman, 2006)

Backus-Naur-Form

Hochschule RheinMain

Erweiterte Backus-Naur-Form (EBNF)

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

1. Statt $A \to w_1, A \to w_2, \ldots A \to w_n$ schreiben wir auch $A \to w_1 \mid w_2 \mid \ \ldots \ \mid w_n$
2. Die Schreibweise $A \to u[v]w$ steht für die beiden Produktionen $A \to uvw$ und $A \to uw$ (d. h. $[v]$ meint, dass v optional ist).
3. Die Schreibweise $A \to u\{v\}w$ 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 auch Grammatiken in erweiterter Backus-Naur-Form (EBNF)

Hochschule RheinMain

EIGENSCHAFTEN VON SPRACHEN UND SPRACHKLASSEN

- Teilmengenbeziehungen der Chomsky-Hierarchie
- Abgeschlossenheit von Sprachklassen
- Entscheidbarkeit von Sprachen

Chomsky-Hierarchie: Teilmengenbeziehungen

Hochschule RheinMain

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 (Beweise folgen im Laufe der Vorlesung):

- $L = \{a^n b^n \mid n \in \mathbb{N}\}$ ist von Typ 2, aber nicht von Typ 3
- $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ ist von Typ 1, aber nicht von Typ 2.
- $H = \{w\$x \mid \text{Turingmaschine } M_w \text{ hält für Eingabe } x\}$
  (das sogenannte Halteproblem) ist von Typ 0, aber nicht von Typ 1.

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

Abgeschlossenheit von Sprachen

Hochschule RheinMain

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

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

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

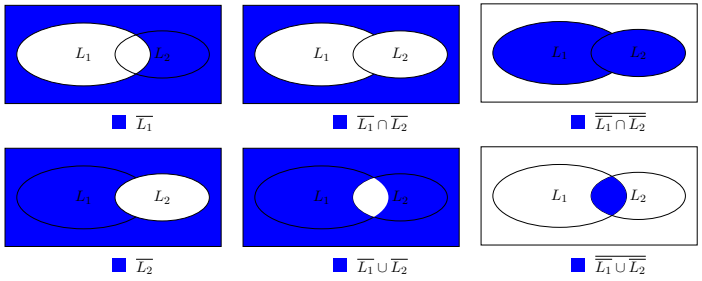
Abgeschlossenheit: Eigenschaften

Hochschule RheinMain

Satz

Sei Klasse $\mathcal{L}$ abgeschlossen bez. Komplementbildung. $\mathcal{L}$ ist abgeschlossen bez. Schnittbildung genau dann, wenn $\mathcal{L}$ abgeschlossen bez. Vereinigung ist.

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

Hochschule RheinMain

Entscheidbarkeit

Eine Sprache heißt **entscheidbar**, wenn es einen Algorithmus gibt, der bei Eingabe der Grammatik G und einem Wort w in endlicher Zeit feststellt, ob $w \in L(G)$ gilt oder nicht.

Man spricht auch von der Entscheidbarkeit des Wortproblems!

Eigenschaften der Typ i-Sprachen:

- Alle Typ 1, 2, 3-Sprachen sind **entscheidbar** (sehen wir gleich)
- Es gibt Typ 0-Sprachen, die **nicht entscheidbar** sind.
- Alle Typ 0-Sprachen sind semi-entscheidbar (rekursiv aufzählbar):
  Es gibt einen Algorithmus, der bei Eingabe der Grammatik G und einem Wort $w \in G$ in endlicher Zeit feststellt, dass $w \in L(G)$ gilt, und bei einem Wort $w \notin G$ entweder feststellt, dass $w \notin L(G)$ gilt, **oder nicht-terminiert.**

Hochschule RheinMain

DAS WORTPRBOLEM

- Definition
- Entscheidbarkeit für Typ 1-Sprachen

Das Wortproblem

Hochschule RheinMain

Definition (Wortproblem für Typ i-Sprachen)

Das **Wortproblem** für Typ i-Sprachen ist die Frage, ob für eine gegebene Typ i-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$ gilt: $w \in L(G)$ oder $w \notin L(G)$.

Wortproblem für Typ 1-Sprachen

Hochschule RheinMain

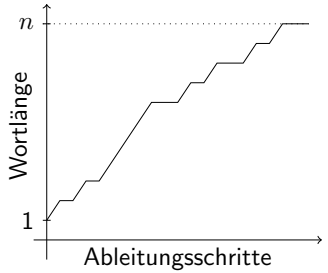
Satz

Das Wortproblem für Typ 1-Sprachen ist entscheidbar: Es gibt einen Algorithmus, der bei Eingabe von Typ 1-Grammatik G und Wort w nach endlicher Zeit entscheidet, ob $w \in L(G)$ oder $w \notin L(G)$ gilt.

Idee zum Beweis des Satzes

Hochschule RheinMain

Ableitung eines Wortes der Länge n



mit Typ 1-Grammatik

Idee zum Entscheiden des Wortproblems:

- Betrachte $S \Rightarrow w_1 \Rightarrow \ldots \Rightarrow w_m$ mit $|w_m| = n$
- Da Typ 1-Produktionen nicht verkürzend sind: $\forall 1 \leq i \leq m : |w_i| \leq n$
- Probiere systematisch für alle Satzformen der Länge $\leq n$ durch, ob sie vom Startsymbol aus ableitbar sind
- Es gibt nur endlich viele Satzformen der Länge $\leq n$ und jede Typ 1-Grammatik leitet nur endlich viele Satzformen der Länge $\leq n$ her, **ohne dabei längere Satzformen zwischendrin herzuleiten**
- Herleitbare Satzformen der Länge n können **rekursiv** aus den Satzformen der Länge $< n$ berechnet werden

Beweis: Entscheidbarkeit des Wortproblems für Typ 1-Sprachen

Hochschule RheinMain

Sei $G = (V, \Sigma, P, S)$ eine Typ 1-Grammatik und $w \in \Sigma^*$.
Für $m, n \in \mathbb{N}_0$ sei

$$\begin{aligned} L_m^n &= \text{Menge aller Satzformen der Länge höchstens } n, \\ &\quad \text{die in höchstens } m \text{ Schritten von } S \text{ aus ableitbar sind} \\ L_m^n &:= \{w \in (V \cup \Sigma)^* \mid |w| \leq n \text{ und } S \Rightarrow_G^k w \text{ mit } k \leq m\} \end{aligned}$$

Rekursive Berechnung der Mengen (für $n > 0$):

$$\begin{aligned} L_0^n &:= \{S\} \\ L_m^n &:= next(L_{m-1}^n, n) \text{ für } m > 0 \\ &\quad \text{wobei } next(L, n) := L \cup \{w' \mid w \in L, w \Rightarrow_G w', |w'| \leq n\} \end{aligned}$$

Die Berechnung terminiert, da die Mengen endlich sind: $|L_m^n| \leq |\Sigma \cup V|^{n+1}$
Wir können auch iterativ aus L_{i-1}^n die nachfolgende Menge L_i^n berechnen

Beweis: Entscheidbarkeit des Wortproblems für Typ 1-Sprachen (2)

Hochschule RheinMain

Iterative Berechnung:

- Starte mit $L_0^n = \{S\}$
- Für $i = 1, 2, 3, \ldots$ berechne $L_i^n = next(L_{i-1}^n)$
- Stoppe dabei, wenn $L_i^n = L_{i-1}^n$
- Prüfe, ob $w \in L_i^n$ gilt.

Korrektheit: Wenn $L_i^n = L_{i-1}^n$, dann gilt $L_{i+k}^n = L_{i-1}^n$ für alle $k \in \mathbb{N}_0$ und daher enthält L_{i-1}^n genau alle aus S ableitbaren Wörter der Länge n.

Terminierung: Beweis durch Widerspruch.

- Nehme an, die Berechnung stoppt nicht.
- Da $L_{i-1}^n \subseteq L_i^n$, muss für alle $i \in \mathbb{N}_0$ gelten: $L_i^n \supset L_{i-1}^n$
- Daher gilt für alle $i \in \mathbb{N}_0$: $|L_i^n| > |L_{i-1}^n|$.
- Widerspruch zu $|L_i^n| \leq |V \cup \Sigma|^{n+1}$ für alle $i \in \mathbb{N}_0$.

Algorithmus 1: Wortproblem entscheiden für Typ 1-Grammatiken

Hochschule RheinMain

Eingabe: Typ 1-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$
Ausgabe: `Ja`, wenn $w \in L(G)$ und `Nein`, wenn $w \notin L(G)$
Beginn

```
n := |w|;
L := {S};
wiederhole
    L_old := L;
    L := next(L_old, n);
bis (w ∈ L) oder (L_old = L);
wenn w ∈ L dann
    return Ja;
sonst
    return Nein;
```


Beispiel

Hochschule RheinMain

$G = (\{S, B\}, \{a, b, c\}, P, S)$ mit
$P = \{S \to aSBc, S \to abc, cB \to Bc, bB \to bb\}$

Wir berechnen L_i^6 für alle i:

$L_0^6 = \{S\}$

$L_1^6 = next(L_0^6) = L_0^6 \cup \{w' \mid w \in L_0^6, w \Rightarrow w', |w'| \leq 6\} = \{S, aSBc, abc\}$,

da $S \Rightarrow aSBc$ und $S \Rightarrow abc$

$L_2^6 = next(L_1^6) = L_1^6 \cup \{w' \mid w \in L_1^6, w \Rightarrow w', |w'| \leq 6\}$
$= \{S, aSBc, abc, aabcBc\}$, da $aSBc \Rightarrow aaSBcBc$ (zu lang) und $aSBc \Rightarrow aabcBc$

$L_3^6 = next(L_2^6) = L_2^6 \cup \{w' \mid w \in L_2^6, w \Rightarrow w', |w'| \leq 6\}$
$= \{S, aSBc, abc, aabcBc, aabBcc\}$, da $aabcBc \Rightarrow aabBcc$

Beispiel (2)

Hochschule RheinMain

$$G = (\{S, B\}, \{a, b, c\}, P, S)$$
$$P = \{S \to aSBc, S \to abc, cB \to Bc, bB \to bb\}.$$

...

$L_3^6 = \{S, aSBc, abc, aabcBc, aabBcc\}$

$L_4^6 = next(L_3^6) = L_3^6 \cup \{w' \mid w \in L_3^6, w \Rightarrow w', |w'| \leq 6\}$
$= \{S, aSBc, abc, aabcBc, aabBcc, aabbcc\}$

da $aabBcc \Rightarrow aabbcc$

$L_5^6 = next(L_4^6) = L_4^6 \cup \{w' \mid w \in L_4^6, w \Rightarrow w', |w'| \leq 6\}$
$= \{S, aSBc, abc, aabcBc, aabBcc, aabbcc\} = L_4^6$

d.h. $L_i^6 = \{S, aSBc, abc, aabcBc, aabBcc, aabbcc\}$ für alle $i \geq 4$

Erinnerung:

ANMELDUNG für die Studienleistung
im **Compass**
bis 05.05. erforderlich

Wiederholung

Hochschule RheinMain

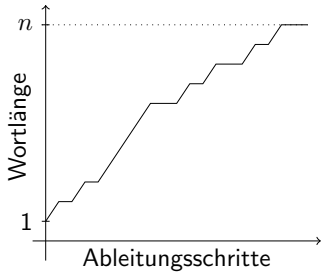
Definition (Wortproblem)

Das **Wortproblem** für Typ i-Sprachen ist die Frage, ob für eine gegebene Typ i-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$ gilt: $w \in L(G)$ oder $w \notin L(G)$.

Satz

Das Wortproblem für Typ 1-Sprachen ist entscheidbar: Es gibt einen Algorithmus, der bei Eingabe von Typ 1-Grammatik G und Wort w nach endlicher Zeit entscheidet, ob $w \in L(G)$ oder $w \notin L(G)$ gilt.

Ableitung eines Wortes der Länge n mit Typ 1-Grammatik



Wortproblem für Typ 2- und Typ 3-Sprachen

Hochschule RheinMain

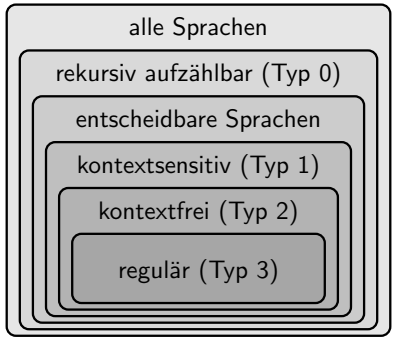
Korollar

Das Wortproblem für Typ 2- und Typ 3-Sprachen ist entscheidbar.

Bemerkungen:

- Der Algorithmus für Typ 1-Sprachen hat exponentielle Laufzeitkomplexität
- Das Wortproblem für Typ 2- und das Wortproblem für Typ 3-Sprachen sind in polynomieller Zeit lösbar.
- Das Wortproblem für Typ 0-Sprachen ist unentscheidbar (aber rekursiv aufzählbar)

Übersicht über die Sprachen



- Die Menge der Typ 0-Grammatiken ist abzählbar (jede Grammatik hat eine endliche Beschreibung, daher Grammatiken der Größe nach aufzählen)
- Menge aller Sprachen $= \mathcal{P}(\Sigma^*)$ ist überabzählbar!

Weitere Entscheidungsprobleme

Definition

- Das Leerheitsproblem für Sprachen vom Typ i ist die Frage, ob für eine Typ i-Grammatik G die Gleichheit $L(G) = \emptyset$ gilt.
- Das Endlichkeitsproblem für Sprachen vom Typ i ist die Frage, ob für eine Typ i-Grammatik G die Ungleichheit $|L| < \infty$ gilt.
- Das Schnittproblem für Sprachen vom Typ i ist die Frage, ob für Typ i-Grammatiken G_1, G_2 gilt: $L(G_1) \cap L(G_2) = \emptyset$.
- Das Äquivalenzproblem für Sprachen vom Typ i ist, die Frage, ob Typ i-Grammatiken G_1, G_2 gilt: $L(G_1) = L(G_2)$.

Typ i-Sprachen aus praktischer Sicht

Aus informatischer Sicht:

- Typ 2- und Typ 3-Sprachen sind wichtig im Rahmen des Compilerbau (lexikalische bzw. syntaktische Analyse)
- Viele Fragestellungen sind jedoch kontextsensitiv oder Typ 0.
- Praktisches Vorgehen: Typ 2-Sprache plus Nebenbedingungen, z.B. Syntax als kontextfreie Grammatik aber noch Nebenbedingungen, dass alle Variablen deklariert wurden usw.