Automaten und Formale Sprachen
2. Vorlesung

Prof. Dr. Dietrich Kuske

FG Automaten und Logik, TU Ilmenau

Wintersemester 2022/23

Grammatiken

Grammatiken in der Informatik sind – ähnlich wie Grammatiken für natürliche Sprachen – ein Mittel, um alle syntaktisch korrekten Sätze (hier: Wörter) einer Sprache zu erzeugen.

Beispiel: Eine vereinfachte Grammatik zur Erzeugung deutscher Sätze:

- in spitzen Klammern: Nicht-Terminale / Variable
- ohne spitze Klammern: Terminale
- $\ell \to r_1 \mid r_2 \mid \ldots r_n$ steht für die Regeln $(\ell \to r_1)$, $(\ell \to r_2)$, $\ldots$, $(\ell \to r_n)$

$$
\begin{aligned}
\langle \text{Satz} \rangle &\to \langle \text{Subjekt} \rangle \langle \text{Prädikat} \rangle \langle \text{Objekt} \rangle \\
\langle \text{Subjekt} \rangle &\to \langle \text{Artikel} \rangle \langle \text{Attribut} \rangle \langle \text{Substantiv} \rangle \\
\langle \text{Artikel} \rangle &\to \varepsilon \mid \text{der} \mid \text{die} \mid \text{das} \\
\langle \text{Attribut} \rangle &\to \varepsilon \mid \langle \text{Adjektiv} \rangle \mid \langle \text{Adjektiv} \rangle \langle \text{Attribut} \rangle \\
\langle \text{Adjektiv} \rangle &\to \text{kleine} \mid \text{bissige} \mid \text{große} \\
\langle \text{Substantiv} \rangle &\to \text{Hund} \mid \text{Katze} \\
\langle \text{Prädikat} \rangle &\to \text{jagt} \\
\langle \text{Objekt} \rangle &\to \langle \text{Artikel} \rangle \langle \text{Attribut} \rangle \langle \text{Substantiv} \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle \text{Satz} \rangle &\Rightarrow \langle \text{Subjekt} \rangle \langle \text{Prädikat} \rangle \langle \text{Objekt} \rangle \\
&\Rightarrow \langle \text{Subjekt} \rangle \text{ jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \langle \text{Artikel} \rangle \langle \text{Attr.} \rangle \langle \text{Substantiv} \rangle \text{ jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \langle \text{Artikel} \rangle \langle \text{Attr.} \rangle \text{ Hund jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \text{der } \langle \text{Attr.} \rangle \text{ Hund jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \text{der } \langle \text{Adj.} \rangle \langle \text{Attr.} \rangle \text{ Hund jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \text{der kleine } \langle \text{Attr.} \rangle \text{ Hund jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \text{der kleine } \langle \text{Adj.} \rangle \text{ Hund jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \text{der kleine bissige Hund jagt } \langle \text{Objekt} \rangle \\
&\Rightarrow \text{der kleine bissige Hund jagt } \langle \text{Artikel} \rangle \langle \text{Attr.} \rangle \langle \text{Subst.} \rangle \\
&\Rightarrow \text{der kleine bissige Hund jagt die } \langle \text{Attr.} \rangle \langle \text{Subst.} \rangle \\
&\Rightarrow \text{der kleine bissige Hund jagt die } \langle \text{Adj.} \rangle \langle \text{Subst.} \rangle \\
&\Rightarrow \text{der kleine bissige Hund jagt die große } \langle \text{Subst.} \rangle \\
&\Rightarrow \text{der kleine bissige Hund jagt die große Katze}
\end{aligned}
$$

Diese Folge nennt man eine „Ableitung“. Sie belegt, daß der Satz

der kleine bissige Hund jagt die große Katze

zu der Sprache gehört, die von der Grammatik erzeugt wird.
Mit Hilfe dieser (endlichen) Grammatik ist es möglich, unendlich viele Sätze zu erzeugen:

der Hund jagt die kleine kleine kleine . . . Katze

Das heißt, die zu der Grammatik gehörende Sprache ist unendlich.

Grammatiken besitzen Regeln der Form

$$\text{linke Seite} \rightarrow \text{rechte Seite}$$

Sowohl auf der linken, als auch auf der rechten Seite können zwei Typen von Symbolen vorkommen:

- Nicht-Terminale (oder Variablen), aus denen noch weitere Wortbestandteile abgeleitet werden sollen
- Terminale (die „eigentlichen“ Symbole)

Im vorherigen Beispiel: auf der linken Seite befindet sich immer genau ein Nicht-Terminal; man spricht von einer kontext-freien Grammatik.

Definition

Eine Grammatik G ist ein 4-Tupel $G = (V, \Sigma, P, S)$, das folgende Bedingungen erfüllt:

- V ist eine endliche Menge von Nicht-Terminalen oder Variablen.
- Σ ist ein Alphabet (Menge der Terminal(symbol)e) mit $V \cap \Sigma = \emptyset$, d.h., kein Zeichen ist gleichzeitig Terminal und Nicht-Terminal.
- $P \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ ist eine endliche Menge von Regeln oder Produktionen, d.h., auf der linken Seite einer Regel steht nie das leere Wort.
- $S \in V$ ist das Startsymbol, die Startvariable oder das Axiom.

Konventionen:

- A, B, C, ..., S, T, ... bezeichnen Nicht-Terminale (Elemente aus V)
- a, b, c, ... bezeichnen Terminalsymbole (Elemente aus Σ)

Beispiel

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

- $V = \{S, B, C\}$
- $\Sigma = \{a, b, c\}$
- $P = \{ \ (S, aSBC), \quad (S, aBC), \quad (CB, BC), \quad (aB, ab), \\ (bB, bb), \quad (bC, bc), \quad (cC, cc) \ \}$

Definition

Sei $G = (V, \Sigma, P, S)$ eine Grammatik und seien $u, v \in (V \cup \Sigma)^*$. Wir schreiben

$$u \Rightarrow_G v$$

falls eine Produktion $(\ell, r) \in P$ und Wörter $x, y \in (V \cup \Sigma)^*$ existieren mit

$$u = x\ell y \text{ und } v = xry\,.$$

Sprech- und Schreibweise:

- Für $u \Rightarrow_G v$ sagen wir „v wird aus u (in einem Schritt) abgeleitet“.
- Ist die Grammatik G klar, so schreibt man $u \Rightarrow v$ für $u \Rightarrow_G v$
- Für „$(\ell, r) \in P$“ schreibt man mitunter $\ell \to r$.

Definition

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

- Eine Ableitung ist eine endliche Folge von Wörtern $w_0, w_1, w_2, \ldots, w_n$ mit $w_0 = S$ und $w_0 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \cdots \Rightarrow w_n$.
- Ein Wort $w \in (V \cup \Sigma)^*$ heißt Satzform, wenn es eine Ableitung gibt, deren letztes Wort w ist.
- Die Sprache

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

aller Satzformen aus Σ^* heißt von G erzeugte Sprache.

Dabei ist $\Rightarrow_G^*$ der reflexive und transitive Abschluß von $\Rightarrow_G$, d.h. $u \Rightarrow_G^* v$ genau dann, wenn $n \geq 0$ und Wörter $u_0, u_1, \ldots u_n \in (V \cup \Sigma)^*$ existieren mit: $u_0 = u$, $u_i \Rightarrow_G u_{i+1}$ für alle $0 \leq i \leq n-1$ und $u_n = v$.

Mit anderen Worten: Die von G erzeugte Sprache $L(G)$ besteht genau aus den Wörtern, die in beliebig vielen Schritten aus S abgeleitet werden können und nur aus Terminalen bestehen.

Beispiel (Fortsetzung von Folie 2.8)

Die Grammatik $G = (V, \Sigma, P, S)$ mit

- $V = \{S, B, C\}$
- $\Sigma = \{a, b, c\}$
- $P = \{\ S \to aSBC,\quad S \to aBC,\quad CB \to BC,\quad aB \to ab,$
  $\qquad bB \to bb,\quad bC \to bc,\quad cC \to cc\}$

1. hat u.a. die Ableitungen:
   $S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$
   $S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aaBBCC \Rightarrow aabBCC \Rightarrow aabbCC$
   $\Rightarrow aabbcC \Rightarrow aabbcc$
2. damit sind u.a. aBC, abc, $aaBCBC$ und $aabbcc$ Satzformen
3. abc und $aabbcc$ gehören zur von G erzeugten Sprache $L(G)$, aBC, $aaBCBC$ und $aabc$ hingegen nicht.
4. man kann zeigen, daß $L(G) = \{a^n b^n c^n \mid n \geq 1\}$ gilt (schwierig: „$\subseteq$“)

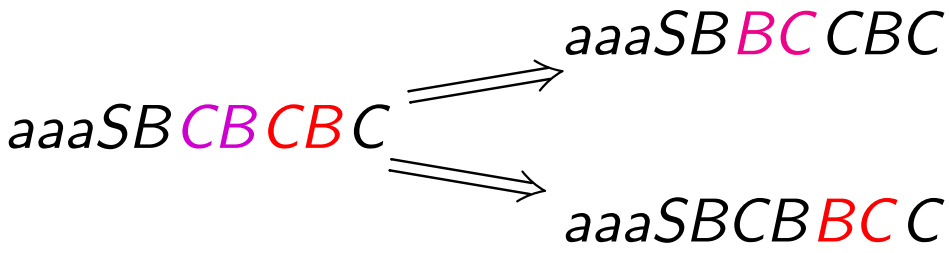
Bemerkung: Für ein $u \in (V \cup \Sigma)^*$ kann es entweder gar kein, ein oder mehrere v geben mit $u \Rightarrow_G v$. Ableiten ist also kein deterministischer, sondern ein nichtdeterministischer Prozeß.

Mit anderen Worten: $\Rightarrow_G$ ist keine Funktion.
daher sagen wir für $u \Rightarrow_G v$ „v wird aus u (in einem Schritt) abgeleitet“ und NICHT „u wird auf v abgebildet“!

Dieser Nichtdeterminismus kann durch zwei verschiedene Effekte verursacht werden ...

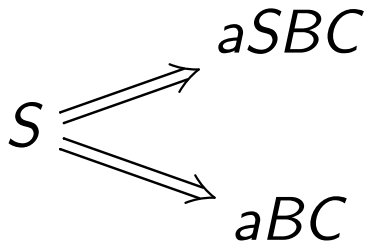
- Eine Regel ist an zwei verschiedenen Stellen anwendbar.

  Beispiel-Grammatik:



- Zwei verschiedene Regeln sind anwendbar (entweder an der gleichen Stelle – wie unten abgebildet – oder an verschiedenen Stellen):

  Beispiel-Grammatik:



Weitere Bemerkungen:

- Es kann beliebig lange Ableitungen geben, die nie zu einem Wort aus Terminalsymbolen führen:

$$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaSBCBCBC \Rightarrow \ldots$$

- Manchmal können Ableitungen in einer Sackgasse enden, d.h., obwohl noch Nichtterminale in einer Satzform vorkommen, ist keine Regel mehr anwendbar:

$$S \Rightarrow aSBC \Rightarrow aaBCBC \Rightarrow aabCBC \Rightarrow aabcBC \not\Rightarrow$$

Noam Chomsky (geb. 1928)

- Linguist am MIT
- fragte sich, wie Menschen lernen, korrekte von inkorrekten Sätzen zu unterscheiden, und warum alle menschlichen Sprachen Ähnlichkeiten aufweisen
- seine Erklärung: angeborene „Grundgrammatik“, die beim Erlernen der Muttersprache angepaßt wird
- hierzu „erfand“ er die Grammatiken, die später in der Informatik wesentlich wurden.

Chomsky-Hierarchie

Typ 0 – Chomsky-0

Jede Grammatik ist vom Typ 0 (oder ein Semi-Thue-System).

Typ 1 – Chomsky-1

1. Eine Regel $(\ell \to r)$ ist kontext-sensitiv, wenn es Wörter $u, v, w \in (V \cup \Sigma)^*$, $|v| > 0$ und ein Nicht-Terminal $A \in V$ gibt mit
$$\ell = u\,A\,w \text{ und } r = u\,v\,w\,.$$

2. Eine Grammatik $G = (V, \Sigma, P, S)$ ist vom Typ 1 (oder kontext-sensitiv), falls
   - alle Regeln aus P kontext-sensitiv sind oder
   - $(S \to \varepsilon) \in P$ die einzige nicht kontext-sensitive Regel in P ist und S auf keiner rechten Seite einer Regel aus P vorkommt.

Typ 2 – Chomsky-2

1. Eine Regel $(\ell \to r)$ ist kontext-frei, wenn $\ell \in V$ und $r \in (V \cup \Sigma)^*$ gilt.
2. Eine Grammatik $G = (V, \Sigma, P, S)$ ist vom Typ 2 (oder kontext-frei), falls sie nur kontext-freie Regeln enthält.

Typ 3 – Chomsky-3

1. Eine Regel $(\ell \to r)$ ist rechtslinear, wenn $\ell \in V$ und $r \in \Sigma V \cup \{\varepsilon\}$ gilt.
2. Eine Grammatik $G = (V, \Sigma, P, S)$ ist vom Typ 3 (oder rechtslinear), falls sie nur rechtslineare Regeln enthält.

Definition

Eine Sprache $L \subseteq \Sigma^*$ heißt vom Typ i ($i \in \{0, 1, 2, 3\}$), falls es eine Typ-i-Grammatik G gibt mit $L(G) = L$. Wir bezeichnen mit $\mathcal{L}_i$ die Klasse der Sprachen vom Typ i.

Eine Sprache vom Typ $\left\{ \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \right\}$ nennt man auch $\left\{ \begin{array}{l} \text{rekursiv aufzählbar} \\ \text{kontext-sensitiv} \\ \text{kontext-frei} \\ \text{rechtslinear} \end{array} \right\}$.

Übersicht:

Typ	alternativ	Sprachklasse	alternativ
0	semi-Thue	$\mathcal{L}_0$	RE
1	kontext-sensitiv	$\mathcal{L}_1$	CS
2	kontext-frei	$\mathcal{L}_2$	CF
3	rechtslinear	$\mathcal{L}_3$	REG

Beispiel (Fortsetzung von Folie 2.8)

Die Grammatik mit der Regelmenge

$$\{\ (S, aSBC),\quad (S, aBC),\quad (CB, BC),\quad (aB, ab),\\ (bB, bb),\quad (bC, bc),\quad (cC, cc)\ \}$$

ist von Typ 0 und nicht kontext-sensitiv. Sie erzeugt die Sprache

$$\{a^n b^n c^n \mid n \geq 1\} \in \mathcal{L}_0\,.$$

Diese Sprache ist also vom Typ 0.

Ist sie kontext-sensitiv?

Beispiel

Die Grammatik mit der Regelmenge

$$\{\ (S, aSBC),\quad (S, aBC),\quad (CB, HB),\quad (HB, HC),\quad (HC, BC),$$
$$(aB, ab),\quad (bB, bb),\quad (bC, bc),\quad (cC, cc)\ \}$$

ist kontext-sensitiv und nicht kontext-frei. Auch sie erzeugt die Sprache

$$\{a^n b^n c^n \mid n \geq 1\} \in \mathcal{L}_1\,.$$

Diese Sprache ist also sogar kontext-sensitiv.

Bemerkung: Später werden wir sehen, daß diese Sprache nicht kontext-frei ist, d.h. daß es keine kontext-freie Grammatik G gibt mit $L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

Beispiel

Die Grammatik mit der Regelmenge

$$\{(S, aSb), (S, \varepsilon)\}$$

ist kontext-frei und nicht rechtslinear. Sie erzeugt die Sprache

$$\{a^n b^n \mid n \geq 0\} \in \mathcal{L}_2 \,.$$

Diese Sprache ist also kontext-frei.

Bemerkung: Später werden wir sehen, daß diese Sprache nicht rechtslinear ist, d.h. daß es keine rechtslineare Grammatik G gibt mit $L(G) = \{a^n b^n \mid n \geq 0\}$.

Beispiel

Die Grammatik mit der Regelmenge

$$\{(S, aA), (A, aS), (S, \varepsilon)\}$$

ist rechtslinear und erzeugt die Sprache

$$\{aa\}^* = \{a^{2n} \mid n \geq 0\} \in \mathcal{L}_3 \,.$$

Diese Sprache ist also rechtslinear.

Bemerkung

- Jede Typ-3/2/1-Grammatik ist vom Typ 0, also $\mathcal{L}_3, \mathcal{L}_2, \mathcal{L}_1 \subseteq \mathcal{L}_0$.
- Jede Typ-3-Grammatik ist vom Typ 2, also $\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_0$.
- Regeln der Form $(A \to \varepsilon)$ können in Grammatiken vom Typ 2 und 3, aber nicht in Grammatiken vom Typ 1 vorkommen. Es ist also nicht klar, ob $\mathcal{L}_2 \subseteq \mathcal{L}_1$ bzw. $\mathcal{L}_3 \subseteq \mathcal{L}_1$ gilt.
- Es ist nicht klar, ob Inklusionen $\mathcal{L}_i \subseteq \mathcal{L}_j$ mit $i < j$ gelten. Zum Beispiel könnte es der Fall sein, daß $\mathcal{L}_0 \subseteq \mathcal{L}_1$ und damit $\mathcal{L}_0 = \mathcal{L}_1$ gelten.
- Es ist nicht klar, ob vielleicht alle Sprachen vom Typ 0 sind.

Satz

Es gibt einen Algorithmus, der als Eingabe eine Typ-1-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$ bekommt und nach endlicher Zeit entscheidet, ob $w \in L(G)$ gilt oder nicht.

Beweis:

1. Fall: $w = \varepsilon$: Da G vom Typ 1 ist, gilt $w \in L(G)$ gdw. $(S \to \varepsilon) \in P$.
Dies kann ein Algorithmus entscheiden.

2. Fall: $|w| \geq 1$, setze $m = |w|$.
Definiere einen gerichteten Graphen (W_m, E_m) wie folgt:

- Knoten sind die Wörter über $V \cup \Sigma$ der Länge $\leq |w| = m$ (insbes. $S, w \in W_m$)
- $(u, v) \in E_m$ gdw. $u \Rightarrow_G v$

Behauptung: $w \in L(G) \iff$ es gibt Pfad von S nach w

„$\Leftarrow$“ klar, da die Kanten des Graphen Ersetzungsschritte von G sind.

„$\Rightarrow$“ Gelte nun $w \in L(G)$

$\Rightarrow$ es gibt $u_0, u_1, \ldots, u_n \in (V \cup \Sigma)^*$ mit $S = u_0$, $u_i \Rightarrow u_{i+1}$ f.a. $0 \leq i < n$ und $u_n = w$

Da G kontext-sensitiv ist, gilt $1 = |u_0| \leq |u_1| \leq |u_2| \leq \cdots \leq |u_n| = |w|$, also $u_i \in W_m$ f.a. $1 \leq i \leq n$.

Also existiert Pfad von S nach w im Graphen (W_m, E_m), womit die Behauptung bewiesen ist.

Um also $w \in L(G)$ zu entscheiden, muß im Graphen (W_m, E_m) ein Erreichbarkeitsproblem gelöst werden, z.B. mit dem Algorithmus von Dijkstra. □

Bemerkungen

1. Für Typ-0-Grammatiken funktioniert dieser Beweis nicht: betrachte die Regelmenge $\{S \to SS, SS \to a\}$ und das Wort $w = a$.
   Die einzige Ableitung von a ist $S \Rightarrow_G SS \Rightarrow_G a$, dies ist aber kein Pfad im Graphen (W_1, E_1) (denn $|SS| > |a|$, also $SS \notin W_1$).

2. In der Vorlesung BuK werden wir sehen:
   Es gibt keinen Algorithmus, der als Eingabe eine Typ-0-Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $w \in \Sigma^*$ bekommt und nach endlicher Zeit entscheidet, ob $w \in L(G)$ gilt.

3. Der Algorithmus des Beweises ist nicht sehr effizient, da der Graph

$$|W_m| = \sum_{0 \leq i \leq m} |V \cup \Sigma|^i \geq |V \cup \Sigma|^m = |V \cup \Sigma|^{|w|}$$

viele Knoten hat. Der Algorithmus benötigt also exponentielle Zeit, um den Graphen (W_m, E_m) auszurechnen.

4. Ergebnisse der Komplexitätstheorie (Vorlesung BuK) legen nahe, daß dieser Zeitaufwand wahrscheinlich unvermeidbar ist.

Zusammenfassung 2. Vorlesung

in dieser Vorlesung neu

- Definitionen Chomsky-Grammatik, Typ 0-3, erzeugte Sprache
- Klassen $\mathcal{L}_i$ der rechtslinearen, der kontext-sensitiven, der kontext-freien und der rekursiv aufzählbaren Sprachen
- Algorithmus, der für Typ-1-Grammatik G und Wort w entscheidet, ob $w \in L(G)$ liegt

offene Fragen

- gibt es einen solchen Algorithmus auch für Typ-0-Grammatiken?
- $\mathcal{L}_2 \subseteq \mathcal{L}_1$? $\mathcal{L}_0 =$ alle Sprachen? Für welche i und j gilt $\mathcal{L}_i = \mathcal{L}_j$?

Thema kommende Vorlesung

Alternative Beschreibung der rechtslinearen Sprachen mittels „Maschinen“