Theoretische Informatik

Prof. Dr. Barbara Staehle

HTWG Konstanz
Fakultaet für Informatik

WS 2021/2022

Teil II

Formale Sprachen, Grammatiken und die Chomsky Hierarchie

Teil II Formale Sprachen

Abschnitt 1

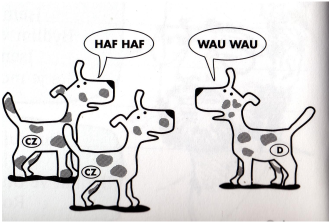
Formale Sprachen

Sprachen aus unserem Alltag

Beispiele:

- Mutter- und Fremdsprachen
- Fachsprachen (Architekten, Bäcker, Chemiker, . . .)
- erfundene Sprachen (Klingonisch, Esperanto, Elbisch, Java, . . .)
- . . .

Quelle: leaderonomics.com



Quelle: dolmetschbar.de

Quelle: news.vanderbilt.edu

Sprachen und Computer

Gemeinsamkeiten der von Menschen genutzten Sprachen

- Von Menschen für Menschen, enthalten oft Mehrdeutigkeiten, Ironie, schlampige Formulierungen, Umgangssprache, ...
- Für Computer ist ein sprachlicher Satz
  - leicht auf seine grammatikalische Korrektheit (Syntax) zu überprüfen,
  - schwer hinsichtlich seiner Bedeutung (Semantik) zu verstehen.
    - Siri, Alexa & Co. haben in den letzten Jahren große Fortschritte gemacht
    - aber immer noch ein aktives Forschungsgebiet, siehe Association for Computational Linguistics (ACL)
- Generell sind von Menschen benutzten Sprachen ungeeignet, um sich mit einem Computer zu verständigen.

Von Computern besser verstandene Sprachen

- Computer verstehen nur „0“ (Strom aus), „1“ (Strom an).
- Formale Sprachen (z.B. Programmiersprachen) basieren auf mathematischen Regeln, sind eindeutig und können daher gut für Computer übersetzt werden.

Formale Sprachen und Computer

Übersetzung (Compilierung) eines Programmes:

- Die Eingabe (z.B. in Java, C, C++, C#, ...) wird von einem Scanner in Symbole (Schlüsselwörter, Variablen, Operatoren, ...) zerlegt.
- Der Parser überprüft, ob die Syntax korrekt ist (Klammerung, Schleifen, Bedingungen, ...).
- Der entstehende Ableitungsbaum wird dann weiter analysiert, optimiert und in einen vom Zielsystem lesbaren Maschinencode übersetzt.
- Für diese Vorlesung interessant:
  - reguläre Sprachen (siehe Teil 3) sind die Grundlagen aller Scanner
  - kontextfreie Sprachen (siehe Teil 2) sind die Grundlagen aller Parser.

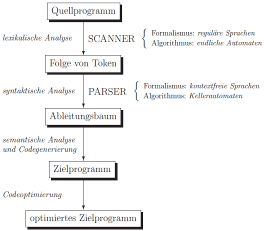


Bild: Arbeitsschritte der Compilierung [Hedtstück, 2012]

Definitionen für Formale Sprachen

Definition

- Ein Alphabet Σ ist eine nichtleere endliche Menge von Symbolen, oder Zeichen.
- Jedes Element $\sigma \in \Sigma$ ist ein Zeichen des Alphabets.
- Jedes Element $\omega \in \Sigma^*$ heißt Wort (über Σ).
- Jede Teilmenge $L \subseteq \Sigma^*$ heißt formale Sprache (über Σ).

Bemerkungen:

- Ein Alphabet muss endlich sein und mindestens ein Symbol enthalten, sonst gibt es keine Einschränkungen.
  - Beispiele: $\Sigma_1 = \{a, \ldots, z\}, \Sigma_2 = \{0, 1\}, \Sigma_3 = \{☺\}$, $\Sigma_4 = \{\text{wau}, \text{miau}, \text{blubb}, \text{piep}\}, \Sigma_5 = \{0, 1, 2, \ldots, 9, A, B, \ldots, F\}, \ldots$
- Σ^* nennt man die Kleensche Hülle von Σ. Σ^*ist abzählbar unendlich, mehr Details folgen.

  $\Rightarrow$ Formale Sprachen können endlich oder abzählbar unendlich sein.

Konkatenation - Erzeugung von Worten

Definition

Jede Konkatenation, Verkettung (Hintereinanderreihung) beliebig vieler Symbole $\sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma$ aus einem Alphabet Σ erzeugt ein Wort ω der Länge n aus Σ^*:

$$\omega = \sigma_1\sigma_2 \ldots \sigma_n \quad \text{mit} \quad \omega \in \Sigma^* \quad \text{und} \quad |\omega| = n.$$

ε heißt das leere Wort mit Länge 0: $|\varepsilon| = 0$.

Beispiele:

- Alphabet $\Sigma_1 = \{\text{a,b,c,}\ldots\text{,z}\}$
  - Worte aus Σ_1: a, b, z, aaab, xyz, katze, cat, chat, qwertz, ...
  - formale Sprachen über Σ_1: $\emptyset$, $\{\text{a,b,c}\}$, $\{\text{one}, \text{two}, \ldots, \text{forty-two}\}$, $\{x \in \Sigma_k^* \mid x = \text{a}^n, n \in \mathbb{N}_0\}$, ...
- Alphabet $\Sigma_2 = \{0, 1\}$
  - Worte aus Σ_2: $\varepsilon, 0, 1, 11, 00, 000000000, 100011101, \ldots$
  - formale Sprachen über Σ_2: $\emptyset, \{00, 11, 10, 01\}, \{x \in \Sigma_2^* \mid |x| = 2\}, \ldots$

Konkatenation - Verkettung von Worten I

Beobachtung: Die Konkatenation der zweier Worte $\omega_1, \omega_2 \in \Sigma^*$ ist wieder ein Wort aus Σ^*. Die Länge des neuen Wortes entspricht der Summe der Länge der Worte ω_1, ω_2.

Begründung: Nehmen wir an, $\omega_1 = a_1 a_2 \dots a_n$ und $\omega_2 = b_1 b_2 \dots b_m$. Daraus folgt für die Konkatenation von ω_1 und ω_2:

$$\omega_1 \omega_2 = a_1 a_2 \dots a_n b_1 b_2 \dots b_m = \omega_3.$$

Daher gilt für das neue Wort ω_3:

$$\omega_3 \in \Sigma^* \quad \text{und} \quad |\omega_3| = n + m = |\omega_1| + |\omega_2|$$

Bemerkungen:

- Die Konkatenation ist assoziativ, aber NICHT kommutativ. Für beliebige Worte u, v, w gilt: $u(vw) = (uv)w$ und $uv \neq vu$.
- Für die Konkatenation mit dem leeren Wort gilt für beliebige Worte w: $w\varepsilon = \varepsilon w$.

Konkatenation - Verkettung von Worten II

Mit diesem Wissen können wir nun Σ^* induktiv definieren:

$\Sigma^0 := \{\varepsilon\}$	Die Sprache, die nur aus dem leeren Wort besteht. Achtung: $\Sigma^0 = \{\varepsilon\} \neq \emptyset$!
$\Sigma^1 := \Sigma$	Die Sprache, die genau aus den Wörtern der Länge 1, also den Elementen von Σ besteht.
$\Sigma^{n+1} := \{xy \vert x \in \Sigma^n, y \in \Sigma\}$	Die Sprache, die aus den Wörtern der Länge $n+1$ besteht, entsteht durch Anhängen genau eines Zeichens an alle Wörter der Länge n.
$\Sigma^* := \bigcup_{i=0}^{\infty} \Sigma^i$	Die Menge aller Wörter über Σ.
$\Sigma^+ := \bigcup_{i=1}^{\infty} \Sigma^i$	Die Menge aller nichtleeren Wörter über Σ.

Beobachtung: $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$

Beispiele zum Mitdenken

- Alphabet $\Sigma_{ab} = \{a, b\}$
  - $\Sigma_{ab}^0 = \{\varepsilon\}$
  - $\Sigma_{ab}^1 = \Sigma_{ab} = \{a, b\}, |\Sigma_{ab}^1| = 2$
  - $\Sigma_{ab}^2 = \{aa, ab, ba, bb\}, |\Sigma_{ab}^2| = 4$
  - ...
  - $\Sigma_{ab}^+ = \{a, b, aa, ab, ba, bb, aaa, \ldots\}$
  - $\Sigma_{ab}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$
- Alphabet $\Sigma_4 = \{0, 1, 2, 3\}$
  - $\Sigma_4^0 = \{\varepsilon\}$
  - $\Sigma_4^1 = \Sigma_4 = \{0, 1, 2, 3\}, |\Sigma_4^1| = 4$
  - $\Sigma_4^2 = \{00, 01, 02, 03, 10, 11, \ldots, 33\}$ $|\Sigma_4^2| = 16$
  - ...
  - $\Sigma_4^+ = \{0, 1, 2, 3, 00, 01, 02, \ldots\}$
  - $\Sigma_4^* = \{\varepsilon, 0, 1, 2, 3, 00, 01, 02, \ldots\}$

Bemerkung: Für ein Alphabet Σ der Länge s hat Σ^n genau s^n Elemente.

Beispiele:

- $\Sigma_0 = \{0, 1\}, |\Sigma_0| = 2$
  - $|\Sigma_0^1| = |\{0, 1\}| = 2 = 2^1$
  - $|\Sigma_0^2| = |\{00, 01, 10, 11\}| = 4 = 2^2$
  - $|\Sigma_0^3| = |\{000, 001, \ldots, 111\}| = 8 = 2^3$
  - ...

Exkurs: Interessante Fragestellungen zu formalen Sprachen

Wortproblem	Gilt für ein Wort $\omega \in \Sigma$ und eine Sprache L die Beziehung $\omega \in L$?
Leerheitsproblem	Enthält eine Sprache L mindestens ein Wort, gilt also $L \neq \emptyset$?
Endlichkeitsproblem	Besitzt eine Sprache L nur endlich viele Elemente?
Äquivalenzproblem	Gilt für zwei Sprachen L_1 und L_2 die Beziehung $L_1 = L_2$?
Spracherzeugungsproblem	Gibt es für eine Sprache L eine Beschreibung, aus der sich alle Wörter systematisch ableiten lassen?

Fragestellungen 1-4 (Entscheidungsprobleme) eher grundlegender und akademischer Natur, das Erzeugungsproblem ist das Wesentlichste. Daher gehen wir das an - wie können Sprachen erzeugt werden?

Abschnitt 2

Grammatiken

Grammatiken - Einführung I

> *Die Grammatik ist beim Sprachenlernen normalerweise der unangenehme Teil – hier muss man Regeln lernen, die oft so völlig anders sind als in der Muttersprache.* (*Lingolia*)

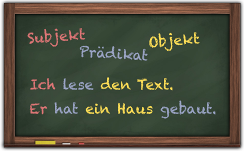


Bild: Aufbau eines einfachen deutschen Satzes Quelle: Lingolia

Beispiele für Subjekt-Prädikat-Objekt Sätze

- Der Eisbär fängt Fische ✓
- Das schnelle Ozelot jagt Vögel ✓
- Die schlaue Maus isst Käse ✓
- Fische Eisbär schlaue ↯
- Jagt Käse die flinke Fische ↯
- Das flinke Ozelot fängt Käse ✓(Inhaltlich falsch)
- Die schlaue Eisbär isst Vögel ✓(Inhaltlich falsch)

Beobachtung:

Grammatiken können nur den richtigen Aufbau (Syntax) des Satzes überprüfen, bzw. richtig aufgebaute Sätze erzeugen. Die Bedeutung (Semantik) muss anders geprüft werden.

Grammatiken - Einführung II

Eine etwas formellere Beschreibung des Satz-Aufbaus:

- <Satz> → <Subjekt> <Prädikat> <Objekt>
- <Subjekt> → <Artikel> <Adjektiv> <Substantiv>
- <Artikel> → Der | Die | Das
- <Adjektiv> → schnelle | schlaue | flinke
- <Substantiv> → Eisbär | Ozelot | Maus
- <Prädikat> → fängt | jagt | isst
- <Objekt> → Käse | Vögel | Fische

Bemerkungen

- Wörter in spitzen Klammern kommen im fertigen Satz nicht mehr vor. Sie werden immer durch andere ersetzt und daher Platzhalter oder Nonterminal(symbol)e genannt.
- Wörter ohne spitze Klammern können nicht mehr ersetzt werden und heißen daher Terminal(symbol)e.
- Jede Ableitung (Ersetzung von Nonterminalen durch andere Symbole) beginnt mit <Satz>. Dieses heißt daher Startsymbol.

Beispiel zum Mitdenken

Wie kann „Das schlaue Ozelot fängt Fische“ abgeleitet werden?

<Satz>
$\Rightarrow$ <Subjekt> <Prädikat> <Objekt>
$\Rightarrow$ <Artikel> <Adjektiv> <Substantiv> <Prädikat> <Objekt>
$\Rightarrow$ Das <Adjektiv> <Substantiv> <Prädikat> <Objekt>
$\Rightarrow$ Das schlaue <Substantiv> <Prädikat> <Objekt>
$\Rightarrow$ Das schlaue Ozelot <Prädikat> <Objekt>
$\Rightarrow$ Das schlaue Ozelot fängt <Objekt>
$\Rightarrow$ Das schlaue Ozelot fängt Fische

Bemerkung: So es nicht explizit vorgegeben ist, ist die Reihenfolge der Anwendung der Regeln ist egal.

Grammatik - Definition

Definition

Eine Grammatik G ist ein Viertupel $G = (N, \Sigma, P, S)$. Sie besteht aus

- dem endlichen Nonterminalalphabet (auch Variablenmenge) N,
- dem endlichen Terminalalphabet Σ mit $\Sigma \cap N = \emptyset$,
- der endlichen Regelmenge (auch Produktionsmenge) P und
- der Startvariablen S mit $S \in N$.

Jede Regel hat die Form $l \to r$ mit

- $l \in (N \cup \Sigma)^+ \setminus \Sigma^+$
- $r \in (N \cup \Sigma)^*$.

Verdeutlichung: Die linke Seite einer Regel muss immer aus mindestens einem Zeichen, davon mindestens ein Nonterminalsymbol, bestehen, die rechte Seite kann auch das leere Wort sein.

Ableitung eines Wortes

Gegeben: Grammatik $G = (N, \Sigma, P, S)$, Worte $x, y \in (N \cup \Sigma)^*$ der Form $x = l\mathbf{u}r$ und $y = l\mathbf{v}r$ mit $l, r, v \in (N \cup \Sigma)^*$, $u \in (N \cup \Sigma)^+$ (x und y sind bis auf die Teilworte **u** und **v** gleich).

Ableitung durch Anwendung der Regeln

- $x \Rightarrow y$, falls y in **einem** Schritt aus x abgeleitet werden kann, durch Anwendung der **einen** Regel $\mathbf{u} \rightarrow \mathbf{v} \in P$
- $x \Rightarrow^* y$, falls y in **null oder endlich vielen** Schritten aus x abgeleitet werden kann (durch Anwendung mehrerer Regeln hintereinander)

Beispiele:

- Das schlaue Ozelot <Prädikat> <Objekt> $\Rightarrow$ Das schlaue Ozelot fängt <Objekt>
- Das schlaue Ozelot <Prädikat> <Objekt> $\Rightarrow^*$ Das schlaue Ozelot fängt Fische
- <Satz> $\Rightarrow^*$ Das schlaue Ozelot fängt Fische

Beispiele zum Mitdenken

Beispiel: Sei die Grammatik G_{xy} gegeben mit

- $G_{xy} = \{N, \Sigma, P, S\} = \{\{S, T, U\}, \{x, y\}, P, S\}$
- $P: \begin{array}{lcl} S & \rightarrow & xT \\ T & \rightarrow & yU \\ U & \rightarrow & xT \\ U & \rightarrow & \varepsilon \end{array}$

Fragen:

- Gilt $xyU \Rightarrow xyxT$?
  Lösung: Ja, durch Anwendung der Regel $U \rightarrow xT$.
- Gilt $xyU \Rightarrow xyyT$?
  Lösung: Nein, hierfür gibt es keine passende Regel.
- Gilt $xyU \Rightarrow^* xyxy$?
  Lösung: Ja, verwende Regeln $U \rightarrow xT$, $T \rightarrow yU$ und $U \rightarrow \varepsilon$.
- Kann man das Wort xy aus dem Startsymbol ableiten?
  Lösung: Ja, verwende Regeln $S \rightarrow xT$, $T \rightarrow yU$ und $U \rightarrow \varepsilon$.
- Kann man das Wort xyx aus dem Startsymbol ableiten?
  Lösung: Nein, hierfür gibt es keine passenden Regeln.

Sprache einer Grammatik

Definition

Die Menge der Wörter über dem Terminalalphabet Σ, die sich durch die Anwendung und Kombination beliebig vieler Regeln aus dem Startsymbol S einer Grammatik G ableiten lassen, heißt Sprache von G:

$$\mathcal{L}(G) := \{\omega \in \Sigma^* \mid S \Rightarrow^* \omega\}$$

Beispiel: Die Sprache der Grammatik G_{xy}

- Erinnerung: $G_{xy} = \{N, \Sigma, P, S\} = \{\{S, T, U\}, \{x, y\}, P, S\}$, mit $P = \{S \to xT, T \to yU, U \to xT \mid \varepsilon\}$
- Beispiele für von G_{xy} erzeugte Worte: $xy, xyxy, xyxyxy, \ldots$
- Die von G_{xy} erzeugte Sprache $\mathcal{L}(G_{xy}) = \{xy, xyxy, xyxyxy, \ldots\} = \{(xy)^n \mid n \in \mathbb{N}\}$

Beispiele: Zahlengrammatiken

Eine Grammatik für unäre Zahlen

- Unäre Zahlen werden lediglich mit einem Symbol (1) dargestellt, ε steht für die 0. Beispiel: Strichlisten.
- Gesucht: die Grammatik G_1, die alle möglichen unären Zahlen erzeugt, für die also $\mathcal{L}(G_1) = \{\varepsilon, 1, 11, 111, 1111, \ldots\}$ gilt.
- Lösung: $G_1 = (N, \Sigma, P, S) = (\{S\}, \{1\}, P, S)$
  P enthält drei Regeln: $P = \{S \to \varepsilon, S \to 1, S \to SS\}$

Eine Grammatik für Dualzahlen

- Dualzahlen bestehen aus $0, 1$, führende 0en werden vermieden.
- Gesucht: die Grammatik G_d, die alle gewünschten Dualzahlen erzeugt, für die also $\mathcal{L}(G_d) = \{0, 1, 10, 11, 100, \ldots\}$ gilt.
- Lösung: $G_d = (N, \Sigma, P, S) = (\{S, T\}, \{0, 1\}, P, S)$ mit

$$P : \begin{array}{lcl} S & \to & 0 \mid 1 \mid 1T \\ T & \to & \varepsilon \mid 0T \mid 1T \end{array}$$

Kurzschreibweise: Alle Regeln mit gleicher linker Seite lassen sich als Alternativen zusammenfassen (rechte Seite per „|“ verodern).

Beispiel: Dyck-Sprache

Die Dyck-Sprache D_n ist als die Menge der korrekt geklammerten Ausdrücke für n verschiedene Klammerpaare definiert.

Verwendung: Wichtige für alle Programmier- und Markupsprachen!

Beispiel: D_2, Klammerpaare () und []

- $(), [], ([]) \in D_2$
- $(,], ([)] \notin D_2$
- D_2 wird von der Grammatik G_2 erzeugt: $\mathcal{L}(G_2) = D_2$
  - $G_2 = \{N, \Sigma, P, S\} = \{\{S\}, \{(,), [,]\}, P, S\}$
  - Die Produktionsmenge P besteht aus vier Regeln:

$$\begin{array}{lcc} S & \rightarrow & \varepsilon \\ S & \rightarrow & SS \\ S & \rightarrow & [S] \\ S & \rightarrow & (S) \end{array}$$

  - Kurzschreibweise:

$$S \quad \rightarrow \quad \varepsilon \mid SS \mid [S] \mid (S)$$

- $\mathcal{L}(D_2) = \{\varepsilon, (), [], (()), ([]), \ldots\}$

Ableitungssequenzen

Verwenden Sie D_2 um das Wort „()[()]()“ abzuleiten.

3 Möglichkeiten:

Ableitung Nr. 1

$$\begin{aligned} S &\Rightarrow SS \\ &\Rightarrow (S)S \\ &\Rightarrow ()S \\ &\Rightarrow ()SS \\ &\Rightarrow ()[S]S \\ &\Rightarrow ()[(S)]S \\ &\Rightarrow ()[()]S \\ &\Rightarrow ()[()](S) \\ &\Rightarrow ()[()]() \end{aligned}$$

Ableitung Nr. 2

$$\begin{aligned} S &\Rightarrow SS \\ &\Rightarrow S(S) \\ &\Rightarrow S() \\ &\Rightarrow SS() \\ &\Rightarrow S[S]() \\ &\Rightarrow S[(S)]() \\ &\Rightarrow S[()]() \\ &\Rightarrow (S)[()]() \\ &\Rightarrow ()[()]() \end{aligned}$$

Ableitung Nr. 3

$$\begin{aligned} S &\Rightarrow SS \\ &\Rightarrow SSS \\ &\Rightarrow (S)SS \\ &\Rightarrow ()SS \\ &\Rightarrow ()[S]S \\ &\Rightarrow ()[(S)]S \\ &\Rightarrow ()[()]S \\ &\Rightarrow ()[()](S) \\ &\Rightarrow ()[()]() \end{aligned}$$

Syntaxbäume

- Jede Ableitungssequenz wird durch einen Syntaxbaum modelliert.
- Nonterminale: innere Knoten, Blätter zeigen abgeleitete Wort.
- Jede Ebene: ein Ableitungsschritt, jedes Kind: ein abgeleitetes nichtleeres Zeichen
- Verwendung: z.B. Compilerbau, Linguistik

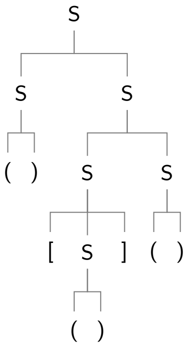


Bild: Syntaxbaum zu Ableitung Nr. 1

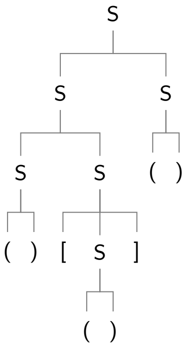


Bild: Syntaxbaum zu Ableitung Nr. 2

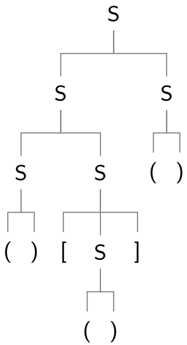


Bild: Syntaxbaum zu Ableitung Nr. 3

Eigenschaften von Grammatiken

Beobachtungen:

- Bei der Ableitung Nr. 1 wurde immer das linkeste Nichtterminal ersetzt. Daher nennt man solch eine Ableitung auch Linksableitung.
- Bei der Ableitung Nr. 2 wurde immer das rechteste Nichtterminal ersetzt. Daher nennt man solch eine Ableitung auch Rechtsableitung.
- Auch Ableitung Nr. 3 ist eine Linksableitung, hat jedoch einen anderen Syntaxbaum als Ableitung Nr. 1 . Dagegen führen die Links- bzw. Rechtsableitung Nr. 2 und 3 zum selben Baum.
- Eine Grammatik G heißt eindeutig, falls alle Ableitungen (links und rechts) eines Wortes immer zum selben Syntaxbaum führen.
- Andernfalls (wie in unserem Beispiel) heißt G mehrdeutig.
- Vorteil von eindeutigen Grammatiken: Hierfür können effiziente Parser geschrieben werden. Mehr Details siehe Vorlesung „Sprachkonzepte“ oder [Wagenknecht and Hielscher, 2014].

Abschnitt 3

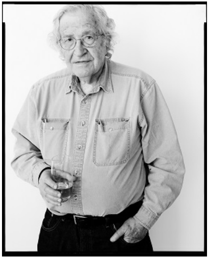
Chomsky-Hierarchie

Klassifizierung von Grammatiken und Sprachen

Frage: Kann man Grammatiken und die von ihnen erzeugten Sprachen nach ihren Eigenschaften klassifizieren?

Antwort: Ja (Noam Chomsky, 1957):
Chomsky-Hierarchie erlaubt Einteilung in vier Klassen:

- Typ 0 Grammatiken oder auch Phrasenstrukturgrammatiken
- Typ 1 Grammatiken oder auch kontextsensitive Grammatiken
- Typ 2 Grammatiken oder auch kontextfreie Grammatiken
- Typ 3 Grammatiken oder auch reguläre Grammatiken



Quelle: www.chomsky.info

Bemerkung: Eine Sprache L heißt Typ n Sprache, falls sie von einer Grammatik vom Typ n erzeugt wird.

Die Grammatiken der Chomsky-Hierarchie I

Sei für die folgenden Beispiele immer $N = \{S, T\}, \Sigma = \{a, b, c\}$.

Definition

Eine Grammatik heißt Phrasenstrukturgrammatik, rekursiv aufzählbar oder Typ 0 Grammatik, falls alle Regeln die Form $l \to r$ mit $l \in (N \cup \Sigma)^+ \setminus \Sigma^+$ und $r \in (N \cup \Sigma)^*$ haben.

Bemerkung: Jede Grammatik ist (auch) eine Typ 0 Grammatik.

Beispiel: $r_0 : aSb \;\to\; Ta$

Definition

Eine Grammatik heißt kontextsensitiv, oder Typ 1 Grammatik, falls für alle Regeln $l \to r$ gilt, dass sie **nicht-verkürzend** sind, dass also $|r| \geq |l|$. Ausnahme: Für das Startsymbol S ist die Regel $S \to \varepsilon$ erlaubt, falls S in keiner rechten Regelseite vorkommt.

Beispiel: $r_1 : aSb \;\to\; aTcb$

Die Grammatiken der Chomsky-Hierarchie II

Definition

Eine Grammatik heißt kontextfrei, oder Typ 2 Grammatik, falls sie kontextsensitiv ist und für alle Regeln $l \to r$ zusätzlich gilt, dass $l \in N$. Ausnahme (von der Eigenschaft nicht-verkürzend): Für Nonterminale α sind Regeln vom Typ $\alpha \to \varepsilon$ erlaubt.

Beispiel: $r_2 : S \;\to\; aSb$

Definition

Eine Grammatik heißt regulär, oder Typ 3 Grammatik, falls sie kontextfrei ist und für alle Regeln zusätzlich $r \in \{\varepsilon\} \cup \Sigma \cup \Sigma N$ gilt.

Bemerkung: Die rechte Seite jeder Regel einer regulären Grammatik besteht entweder aus dem leeren Wort, einem einzelnen Terminal, oder aus einem Terminal gefolgt von einem Nonterminal.

Beispiel: $r_3 : S \;\to\; aT$

Bestimmung des Chomsky-Typs

Hinweis: Der Chomsky-Typ einer **Grammatik** entspricht dem kleinsten Chomsky-Typ ihrer Regeln: Eine Grammatik mit 5 Regeln vom Typ 3 und einer Regel vom Typ 0 ist insgesamt vom Typ 0.

Beispiele:
Geben Sie den (numerisch größten) Chomsky-Typ jeder der folgenden Regeln, sowie jeder der folgenden Grammatiken an, wobei $N = \{S, T, U\}$, sowie $\Sigma = \{x, y, z\}$ ist.

Regel / Grammatik	**Typ**	**Begründung**
$r_1 : S \to ST$	2	Links nur ein NT daher Typ 2 oder 3, aber rechts zwei NTs, daher Typ 2.
$r_2 : ST \to S$	0	Links mehr als ein NT daher Typ 0 oder 1, rechts kürzer als links, daher Typ 0.
$r_3 : SxTU \to xyzSTU$	1	Links mehr als ein NT daher Typ 0 oder 1, rechts länger als links, daher Typ 1.
$r_4 : U \to zU$	3	Links nur ein NT daher Typ 2 oder 3, rechts T gefolgt von NT, daher Typ 3.
$G_5 = (N, \Sigma, \{r_1, r_2, r_3\}, S)$	0	Regeln sind Typ 2, 0 und 3, daher ist G_5 vom Typ 0.
$G_6 = (N, \Sigma, \{S \to \varepsilon, S \to x\}, S)$	3	Regeln sind alle Typ 3, daher ist G_6 vom Typ 3.
$G_7 = (N, \Sigma, \{\varepsilon \to S, S \to x\}, S)$	-	erste Regel ist unkorrekt formuliert, daher ist G_7 keine Grammatik.

Abkürzungen: Nonterminal (NT), Terminal (T).

Die Chomsky-Hierarchie für Sprachen

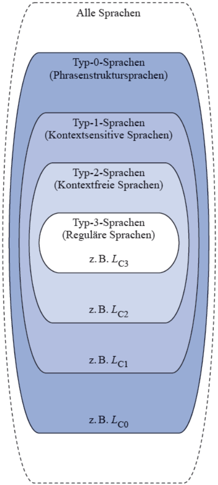
Zusammenhang Grammatiken - Sprachen:

- Eine Sprache L heißt Typ n Sprache, falls sie von einer Grammatik vom Typ n erzeugt wird.
- Die Menge aller Typ n Sprachen wird als die Sprachklasse $\mathcal{L}_n$ bezeichnet.
- Die verschiedenen Sprachklassen stehen in folgender echter Inklusion:

$$\mathcal{L}_0 \supset \mathcal{L}_1 \supset \mathcal{L}_2 \supset \mathcal{L}_3$$

- Achtung: es gibt formale Sprachen, die nicht in $\mathcal{L}_0$ enthalten sind.

Bemerkung: Je größer n, desto härter sind die Einschränkungen für die Sprachen der Klasse $\mathcal{L}_n$. Dies macht sie einerseits schneller durch Computer verarbeitbar, andererseits weniger ausdrucksstark.



Quelle: [Hoffmann, 2011]

Beispiele zum Mitdenken

r_0	$aSb \to Ta$
r_1	$aSb \to aTcb$
r_2	$S \to aSb$
r_3	$S \to aT$

Tabelle: Regeln aus den Beispielen

	r_0	r_1	r_2	r_3
Typ 0 Grammatik	j	j	j	j
Typ 1 Grammatik	n	j	j	j
Typ 2 Grammatik	n	n	j	j
Typ 3 Grammatik	n	n	n	j

Tabelle: Zugehörigkeit zu Grammatiken der verschiedenen Klassen

L_{C3}	$\{(ab)^n \mid n \in \mathbb{N}\}$
L_{C2}	$\{a^n b^n \mid n \in \mathbb{N}\}$
L_{C1}	$\{a^n b^n c^n \mid n \in \mathbb{N}\}$
L_{C0}	$\{\omega \in \{0, 1, \ldots 9\} \mid \omega$ kommt in den Nachkommastellen von π vor$\}$

Tabelle: Beispiele von Sprachen

	L_{C3}	L_{C2}	L_{C1}	L_{C0}
Typ 0 Sprache	j	j	j	j
Typ 1 Sprache	j	j	j	n
Typ 2 Sprache	j	j	n	n
Typ 3 Sprache	j	n	n	n

Tabelle: Zugehörigkeit der Sprachen zu den Sprachklassen

Bemerkung: L_{C0} ist ein gutes Beispiel für $\mathcal{L}_0$ - die Sprachen dieser Klasse haben keine wirklich praktische Bedeutung und sind schwer zu fassen.

Verwendete oder empfohlene Literatur I

[Hedtstück, 2012] Hedtstück, U. (2012).
Einführung in die theoretische Informatik: formale Sprachen und Automatentheorie.
Oldenbourg Verlag.
Als eBook in der HTWG-Bibliothek verfügbar.

[Hoffmann, 2011] Hoffmann, D. W. (2011).
Theoretische Informatik.
Carl Hanser Verlag GmbH & Co. KG, 2. Auflage.
Als eBook in der HTWG-Bibliothek verfügbar.

[Hoffmann, 2015] Hoffmann, D. W. (2015).
Theoretische Informatik.
Carl Hanser Verlag GmbH & Co. KG, 3. Auflage.
Als eBook in der HTWG-Bibliothek verfügbar.

Verwendete oder empfohlene Literatur II

[Hopcroft et al., 2011] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2011).
Einführung in die Automatentheorie, formale Sprachen und Berechenbarkeit (bzw. Komplexitätstheorie); engl.: Introduction to automata theory, languages and computation.
Pearson, 3. Auflage.
Als eBook in der HTWG-Bibliothek verfügbar.

[Kulla, 2015] Kulla, S. (2015).
Mathe für Nicht-Freaks - Grundlagen der Mathematik.
Wikibooks.

[Teschl and Teschl, 2013] Teschl, G. and Teschl, S. (2013).
Mathematik für Informatiker: Band 1: Diskrete Mathematik und Lineare Algebra.
Springer Vieweg, 4. Auflage.
Als eBook in der HTWG-Bibliothek verfügbar.

Verwendete oder empfohlene Literatur III

[Wagenknecht and Hielscher, 2014] Wagenknecht, C. and Hielscher, M. (2014).
Formale Sprachen, abstrakte Automaten und Compiler.
Springer Vieweg, 2. Auflage.
Als eBook in der HTWG-Bibliothek verfügbar.