Vorlesung

Grundlagen der Theoretischen Informatik / Einführung in die Theoretische Informatik I

Bernhard Beckert

Institut für Informatik

UNIVERSITÄT KOBLENZ · LANDAU

Sommersemester 2007

Dank

Diese Vorlesungsmaterialien basieren ganz wesentlich auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Ihnen beiden gilt mein herzlicher Dank.

– Bernhard Beckert, April 2007

Inhalt von Teil II

In den folgenden Abschnitten führen wir die Begriffe

- **Sprache**
- **Grammatik**

ein.

Wir untersuchen insbesondere

1. wie man Probleme aus der Mathematik, Graphentheorie, Logik als **Probleme über Sprachen** formulieren kann.
2. wie man Klassen von Grammatiken von steigendem Schwierigkeitsgrad definiert: **Chomsky-Hierarchie**.
3. **wieviele** Grammatiken und Sprachen **es überhaupt gibt** (soviele wie natürliche Zahlen, reelle Zahlen oder komplexe Zahlen?)

Teil II

Terminologie

Alphabete, Wörter

Definition 6.1 (Alphabet)

Ein **Alphabet** ist eine Menge von Zeichen/Buchstaben

- Grundlage einer Sprache (die zur Verfügung stehenden Zeichen)
- Meist endlich

Definition 6.2 (Wort)

Ein **Wort** (über einem Alphabet Σ)
ist eine endliche Folge von Zeichen aus Σ

$|w|$ bezeichnet Länge eines Wortes w

ε bezeichnet das leere Wort

Alphabete, Wörter

Operationen auf Wörtern

Verknüpfung (Konkatenation):

$w \circ w'$

assoziativ, oft geschrieben als ww'

i-te Potenz:

$$w^0 = \varepsilon, \qquad w^{i+1} = ww^i$$

Reverse:

$w^R =$ das Wort w rückwärts

Sprache

Definition 6.3 (Sprache)

Eine Sprache L (über einem Alphabet Σ)
ist eine Menge von Wörtern über Σ.

Operationen auf Sprachen

Konkatenation:

$$L \circ M = \{w \circ w' \mid w \in L,\ w' \in M\}$$

i-te Potenz:

$$L^0 = \{\varepsilon\}, \qquad L^{i+1} := LL^i$$

Reverse:

$$L^R = \{w^R : \ w \in L\}$$

Sprache

Kleene-Hülle

$$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$$

Variante:

$$L^+ = LL^* = L^1 \cup L^2 \cup \ldots$$

Σ^* bezeichnet die Menge aller Wörter über Σ

Genau genommen besteht ein Unterschied:

ein Buchstabe $\neq$ Wort, das nur aus dem einen Buchstaben besteht

Darum ist Σ selbst keine Sprache über Σ

(Oft wird über diesen Unterschied hinweggesehen)

Reguläre Ausdrücke

Definition 6.4 (Reguläre Ausdrücke)

Menge $\mathfrak{Reg}_\Sigma$ der **regulären Ausdrücke** (über Σ) ist definiert durch:

1. 0 ist ein regulärer Ausdruck
2. Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
3. Sind r und s reguläre Ausdrücke, so auch
   - $(r + s)$ (Vereinigung),
   - (rs) (Konkatenation),
   - (r^*) (Kleene Stern)

Klammern können weggelassen werden, dann

- * hat Vorrang vor Konkatenation
- Konkatenation hat Vorrang vor $+$

Reguläre Ausdrücke

Definition 6.5 (Semantik regulärer Ausdrücke)

Ein regulärer Ausdruck r stellt eine Sprache $\mathfrak{I}(r)$ über Σ wie folgt dar:

$$\begin{aligned}
\mathfrak{I}(0) &:= \emptyset \\
\mathfrak{I}(a) &:= \{a\} \qquad \text{für } a \in \Sigma \\
\mathfrak{I}(r+s) &:= \mathfrak{I}(r) \cup \mathfrak{I}(s) \\
\mathfrak{I}(rs) &:= \mathfrak{I}(r)\mathfrak{I}(s) \\
\mathfrak{I}(r^*) &:= \mathfrak{I}(r)^*
\end{aligned}$$

Wir benutzen auch das Makro ...

$$1 := 0^*$$

Es gilt: $\mathfrak{I}(1) = \{\varepsilon\}$

Reguläre Ausdrücke

Übung

Welche Sprachen werden durch die folgenden regulären Ausdrücke dargestellt?

- aa
- $(a+b)^*$
- $aa^* + bb^*$

Reguläre Ausdrücke als Suchmuster für `grep`

Das Kommando `grep` (bzw. `egrep`)

- Sucht Wörter (Strings) in Dateien
- Benutzt reguläre Ausdrücke als Suchmuster
- Sehr schnell
- Volle Funktionalität mit `egrep` (UNIX/LINUX)

Reguläre Ausdrücke als Suchmuster für `grep`

Syntax bei `grep`

`grep`	Regulärer Ausdruck
ww'	ww'
$w \mid w'$	$w + w'$
$w*$	w^*
$w+$	w^+

Syntactic Sugar

`grep`	Regulärer Ausdruck
`[abc]`	$a+b+c$
`[a-d]`	$a+b+c+d$
.	beliebiges Zeichen aus Σ

Grammatik

- Beschreibt eine Sprache
- Menge von Regeln, mit deren Hilfe man Wörter ableiten kann
- Die zu einer Grammatik gehörende Sprache besteht aus den
  - **ableitbaren**
  - **terminalen**

  Wörtern

Grammatik

Definition 6.6 (Grammatik)

Eine **Grammatik** G über einem Alphabet Σ ist ein Tupel

$$G = (V, T, R, S)$$

Dabei ist

- V eine endliche Menge von **Variablen**
- $T \subseteq \Sigma$ eine endliche Menge von **Terminalen** mit $V \cap T = \emptyset$
- R eine endliche Menge von **Regeln**
- $S \in V$ das **Startsymbol**

Grammatik

Definition 6.7 (Regel)

Eine Regel ist ein Element

$$(P, Q) \in \left((V \cup T)^* \; V \; (V \cup T)^*\right) \times (V \cup T)^*$$

Das heißt:

- P und Q sind Wörter über $(V \cup T)$
- P muss mindestens eine Variable enthalten
- Q ist beliebig

Bezeichnung:

P: Prämisse

Q: Conclusio

Grammatik

Schreibweise für Regeln

- Schreibweise für Regel (P, Q):

  $P \rightarrow_G Q$ bzw. $P \rightarrow Q$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

  $P \rightarrow Q_1 \mid Q_2 \mid Q_3$ für $P \rightarrow Q_1,\ P \rightarrow Q_2,\ P \rightarrow Q_3$

Konvention (meistens)

- **Variablen** als Großbuchstaben
- **Terminale** als Kleinbuchstaben

Beispiel 6.8

$S \rightarrow B$

$B \rightarrow \textit{do begin } B \textit{ end}$

$B \rightarrow A$

$A \rightarrow \textit{nop } A$

$A \rightarrow \varepsilon$

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

1. $aktuellWort := S$ (Startsymbol)
2. Wähle eine Regel $P \rightarrow Q$, so dass P in *aktuellWort* vorkommt
3. Ersetze (ein) Vorkommen von P in *aktuellWort* durch Q
4. Falls *aktuellWort* noch Variablen enthält (nicht terminal), GOTO 2

Ausgabe: Das terminale Wort *aktuellWort*

Beachte

Die Berechnung

- ist nicht deterministisch (Auswahl der Regel)
- kann mehr als ein Ergebnis liefern (oder auch keines)
- kann in Endlosschleifen geraten

Rechnung einer Grammatik

Beispiel 6.9 (Einfache Grammatiken)

Welche Wörter kann man ableiten?

- $G_a = (\{S\}, \{a\}, \{R_1, R_2\}, S)$

  $R_1 = S \to aS$

  $R_2 = S \to \varepsilon$

- $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$

  $R_1 = S \to aSb$

  $R_2 = S \to \varepsilon$

- Sei $G_{gerade} = (\{S, S_0\}, \{0,1,2,3,4,5,6,7,8,9\}, \{R_1, R_2\}, S)$

  $R_1 = S \to 1S \mid 2S_0 \mid 3S \mid 4S_0 \mid 5S \mid 6S_0 \mid 7S \mid 8S_0 \mid 9S$

  $R_2 = S_0 \to S \mid \varepsilon$

Rechnung einer Grammatik

Definition 6.10 (Ableitung, Rechnung)

Gegeben:

- Grammatik $G = (V, T, R, S)$
- Wörter w, w' aus $(V \cup T)^*$

Es gilt

$$w \Longrightarrow_G w' \qquad (\text{„}w \text{ geht über in } w'\text{“})$$

falls

$$\exists u, v \in (V \cup T)^* \; \exists P \to Q \in R \;\; \big(w = uPv \text{ und } w' = uQv\big)$$

Rechnung einer Grammatik

Schreibweise für Ableitung

$$w \Longrightarrow_G^* w'$$

falls es Wörter $w_0, \ldots, w_n \in (V \cup T)^*$ $(n \geq 0)$ gibt mit

- $w = w_0$
- $w_m = w'$
- $w_i \Longrightarrow_G w_{i+1}$ für $0 \leq i < n$

Merke: $w \Longrightarrow_G^* w$ gilt stets $(n = 0)$

Die Folge $w_0, \ldots, w_n$ heißt Ableitung oder Rechnung

– von w_0 nach w_n
– in G
– der Länge n

Vorsicht: Indeterminismus

Beispiel 6.11 (Indeterminismus)

Wir betrachten die Grammatik $G = (\{S, B\}, \{a, b, c\}, \{R_0, R_1, R_2, R_3\}, S)$

$$
\begin{aligned}
R_0 &= S \to aBBc \\
R_1 &= B \to b \\
R_2 &= B \to ba \\
R_3 &= BB \to bBa
\end{aligned}
$$

Drei Möglichkeiten, das Wort *abbac* zu erzeugen:

$$
\begin{aligned}
S &\Longrightarrow_{R_0} aBBc \Longrightarrow_{R_1} abBc \Longrightarrow_{R_2} abbac \\
S &\Longrightarrow_{R_0} aBBc \Longrightarrow_{R_2} aBbac \Longrightarrow_{R_1} abbac \\
S &\Longrightarrow_{R_0} aBBc \Longrightarrow_{R_3} abBac \Longrightarrow_{R_1} abbac
\end{aligned}
$$

Vorsicht: Indeterminismus

Warum ist das ein *Feature* und kein *Bug*?

- Erlaubt einfachere Definition von Grammatiken
- Für manche Sprachen gibt es keine eindeutige Grammatiken
- Eine Grammatik beschreibt die Struktur der Wörter.
  Ein Wort kann mehrere mögliche Strukturen haben.
- Für natürliche Sprachen braucht man das unbedingt:
  Manche Sätze sind mehrdeutig (in ihrer Grammatik),
  also müssen auch die Grammatiken mehrdeutig sein!

Vorsicht: Indeterminismus

Beispiel 6.12 (Mehrdeutige Grammatik natürlichsprachlicher Sätze)

Time flies like an arrow.
Fruit flies like a banana.

- Beide Sätze haben zwei mögliche grammatische Strukturen.
- Erst unser semantisches Verständnis wählt eine aus.

Erzeugte Sprache, Äquivalenz

Definition 6.13 (Erzeugte Sprache)

Gegeben: Eine Grammatik G

Die von G erzeugte Sprache $L(G)$ ist die Menge aller **terminalen** Wörter, die durch G vom Startsymbol S aus erzeugt werden können:

$$L(G) \quad := \quad \{w \in T^* \mid S \Longrightarrow_G^* w\}$$

Definition 6.14 (Äquivalenz)

Zwei Grammatiken G_1, G_2 heißen **äquivalent** gdw

$$L(G_1) = L(G_2)$$

Beispiel

Beispiel 6.15

Grammatik $G_{ab} = (\{S\}, \{a, b\}, \{R_1, R_2\}, S)$ mit

$$R_1 = S \to aSb$$
$$R_2 = S \to \varepsilon$$

Mögliche Ableitung:

$$S \Longrightarrow_{R_1} aSb \Longrightarrow_{R_1} aaSbb \Longrightarrow_{R_1} aaaSbbb \Longrightarrow_{R_2} aaabbb$$

Also: $a^3 b^3 \in L(G_2)$

Lemma 6.16

Die Grammatik G_{ab} erzeugt die Sprache

$$L(G_{ab}) = \{a^n b^n \mid n \in \mathbb{N}_0\}$$

Beispiel

Beweis

Dass G_{ab} tatsächlich genau diese Sprache erzeugt, zeigen wir allgemein, indem wir alle möglichen Ableitungen von G_{ab} betrachten.

$\subseteq$: zu zeigen: Jedes terminale Wort, das von G_{ab} erzeugt wird, hat die Form $a^n b^n$.

Wir zeigen für alle $w \in (V \cup T)^*$: Falls $S \Longrightarrow^*_{G_{ab}} w$, dann gilt entweder $w = a^n S b^n$ oder $w = a^n b^n$ für ein $n \in \mathbb{N}_0$.

Beispiel

Beweis (Forts.)

Dazu verwenden wir eine **Induktion über die Länge einer Ableitung** von S nach w.

Induktionsanfang: $w = S = a^0 S b^0$

Induktionsschritt: Es gelte $S \Longrightarrow_{G_{ab}}^{*} w \Longrightarrow_{G_{ab}} w'$, und für w gelte nach der Induktionsvoraussetzung bereits $w = a^n b^n$ oder $w = a^n S b^n$. Außerdem sei $w \Longrightarrow_{G_{ab}} w'$ eine Ableitung in einem Schritt. Nun ist zu zeigen: $w' = a^m b^m$ oder $w' = a^m S b^m$ für irgendein m.

Beispiel

Beweis (Forts.)

Fall 1: $w = a^n b^n$. Dann konnte keine Regel angewandt werden, da w schon terminal ist, also tritt dieser Fall nie auf.

Fall 2: $w = a^n S b^n$. Dann wurde von w nach w' entweder Regel R_1 oder R_2 angewandt.

Falls R_1 angewandt wurde, dann gilt $w = a^n S b^n \Longrightarrow_{R_1} a^n a S b b^n = a^{n+1} S b^{n+1} = w'$.

Falls R_2 angewandt wurde, dann gilt $w = a^n S b^n \Longrightarrow_{R_2} a^n \varepsilon b^n = w'$. Dies Wort ist terminal und hat die geforderte Form $a^n b^n$.

Beispiel

Beweis (Forts.)

$\supseteq$: zu zeigen: Für alle n kann $a^n b^n$ von G_{ab} erzeugt werden: $S \Longrightarrow_{G_{ab}}^{*} a^n b^n$ $\forall n \in \mathbb{N}_0$.

Um $a^n b^n$ zu erzeugen, wende man auf S n-mal die Regel R_1 und dann einmal die Regel R_2 an. □

Beispiel: Dycksprache

Definition 6.17 (Dycksprache)

Gegeben:

- $k \in \mathbb{N}$
- $\Sigma_k := \{x_1, \overline{x_1}, x_2 \ldots, x_k, \overline{x_k}\}$ ein Alphabet mit $2k$ Symbolen

Die Dycksprache D_k ist die kleinste Menge die folgende Bedingungen erfüllt:

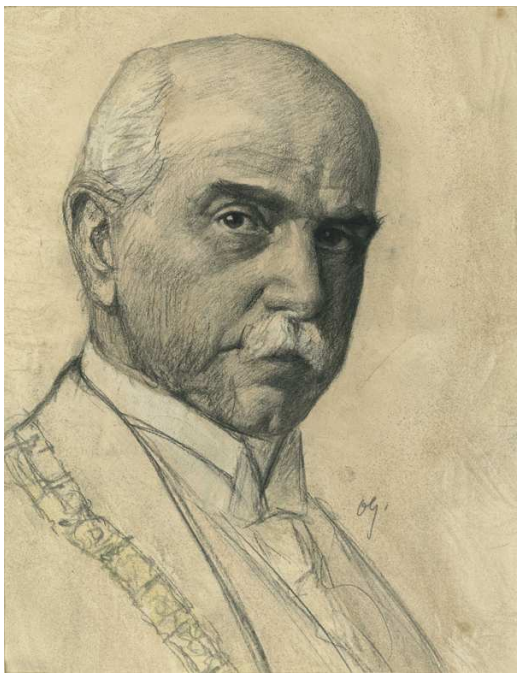
1. $\varepsilon \in D_k$,
2. Falls $w \in D_k$, so auch $x_i w \overline{x_i}$.
3. Falls $u, v \in D_k$, so auch uv.

Interpretiert man die x_i als öffnende, die $\overline{x_i}$ als zugehörige schließende Klammern, so kann man die Dycksprache als die **Menge aller korrekten Klammerausdrücke** sehen.

Beispiel: Dycksprache

Walther von Dyck ⋆1856, †1934

- Mathematiker
- Hochschulpolitiker
- Erster Rektor der TU München
- Einer der Gründungsväter des Deutschen Museums



[Foto: Deutsches Museum]

Teil II

Terminologie

Darstellung von Problemen

Fakt

So ziemlich alle Probleme können als Probleme über Sprachen formuliert werden.

Beispiel 7.1 (Primzahlen)

Alphabet $\Sigma_{\text{num}} := \{|\}$

Sprache $L_{\text{primes}} := \{\underbrace{||\ldots|}_{p \text{ mal}} \mid p \text{ prim}\}$

Darstellung von Problemen

Eingabealphabet

$$\Sigma = \{0, 1, \ldots, n-1\}$$

erlaubt Darstellung einer Ganzzahl zur Basis n

Beispiel 7.2

5 binär: 101

5 unär: ||||| (oder auch 11111)

Darstellung von Problemen

Speicheraufwand

***n*-äre Darstellung** ($n > 1$) einer Zahl k führt zu einer Speicherersparnis:

$\log_n k$ (n-är) statt k (unär)

Nur der Schritt von unär auf binär ist wesentlich, denn

$$\log_n k = \frac{1}{\log_2 n} \cdot \log_2 k = c \cdot \log_2 k$$

(von binär auf n-är nur lineare Einsparung)

Darstellung des Erfüllbarkeitsproblems SAT

Problem SAT

Gegeben: Eine aussagenlogische Formel w

Frage: Gibt es eine Belegung der booleschen Variablen in w, so dass w zu *true* auswertet?

Signatur für aussagenlogische Formeln

Signatur: $\Sigma_{\text{sat}} := \{\wedge, \vee, \neg, (,), x, 0, 1\}$

Dabei Darstellung von boolscher Variablen x_i als x gefolgt von i binär kodiert.
Dadurch Formel der Länge n um (unerheblichen) Faktor $\log n$ länger.

Darstellung des Erfüllbarkeitsproblems SAT

Definition 7.3 (Satisfiability)

Sprache

$$L_{\text{sat}} := \{ w \in \Sigma^*_{sat} : \quad w \text{ ist eine aussagenlogische Formel,} \\ \text{und es gibt eine Belegung für die } x_i, \\ \text{so dass die Formel } w \text{ zu } \textit{true} \text{ auswertet } \}$$

Darstellung des Erreichbarkeitsproblems in Graphen

Erreichbarkeitsproblem

Gegeben: Ein Graph mit Ecken v_1 bis v_n

Frage: Gibt es einen Weg von Ecke v_1 zu Ecke v_n?

Signatur für Graphen

Signatur: $\Sigma_{\text{graph}} := \{v, e, 0, 1, (,), \#\}$

Darstellung von

Ecke v_i als v gefolgt i binär kodiert

Kante $e_{i,j}$ als $e(\mathit{string}_1 \# \mathit{string}_2)$, wobei

- string_1 die binäre Darstellung von i,
- string_2 die binäre Darstellung von j

Darstellung des Erreichbarkeitsproblems in Graphen

Definition 7.4 (Erreichbarkeitsproblem)

Sprache

$$L_{\text{reach}} := \{w \in \Sigma^*_{\text{graph}} : \quad \text{es gibt einen Weg in } w \text{ von der ersten Ecke } v_1 \text{ zur letzten Ecke } v_n\}$$

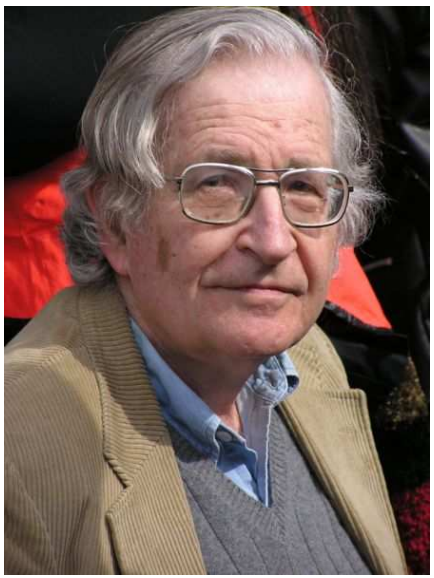
Teil II

Terminologie

Die Chomsky-Hierarchie

Noam Chomsky ⋆1928

- Professor für Linguistik und Philosophie am MIT
- Bedeutender Linguist
- Bedeutender Beitrag zur Informatik:
  Erste Beschreibung der Chomsky-Hierarchie
  (1956)
- Bedeutender linker Intellektueller und
  Globalisierungskritiker



Die Chomsky-Hierarchie

Was muss eine Grammatik erfüllen?

- Sie darf nur **endlich viele Regeln** haben
- Jede Regelprämisse muss **mindestens eine Variable** enthalten

Das Wort kann im Lauf der Ableitung beliebig wachsen und wieder schrumpfen.

(Weitere) Beschränkung der Form, die Regeln haben dürfen, führt zu
– **Grammatiktypen** und damit auch zu
– **Sprachtypen**
von verschiedenen Schwierigkeitsgraden.

Die Chomsky-Hierarchie

Definition 8.1 (Rechtlineare Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **rechtslinear** gdw

$$\forall (P \to Q) \in R \ \left(P \in V \text{ und } Q \in T^* \cup T^+ V\right)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Rechts **höchstens eine Variable**
- Wenn rechts eine Variable steht, steht sie **ganz rechts im Wort**.

Die Chomsky-Hierarchie

Definition 8.2 (Kontextfreie Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextfrei** gdw

$$\forall (P \to Q) \in R \ \left(P \in V \text{ und } Q \in (V \cup T)^*\right)$$

Das heißt, bei jeder Regelanwendung:

- Links eine **einzelne Variable**
- Die Prämisse macht keine Aussage, was der Kontext dieser Variablen ist („kontextfrei“)
- Rechts steht etwas beliebiges

Die Chomsky-Hierarchie

Definition 8.3 (Kontextsensitive Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **kontextsensitiv** gdw

$\forall (P \to Q) \in R$:

1. $\exists u, v, \alpha \in (V \cup T)^* \; \exists A \in V \; \left(P = uAv \text{ und } Q = u\alpha v \text{ mit } |\alpha| \geq 1\right)$, **oder**

   die Regel hat die Form $S \to \varepsilon$

2. S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Eine Variable A wird in einen String α mit $|\alpha| \geq 1$ überführt
- Die Ersetzung von A durch α findet nur statt, wenn der in der Regel geforderte **Kontext** (u und v), vorhanden ist
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Die Chomsky-Hierarchie

Definition 8.4 (Beschränkte Grammatik)

Eine Grammatik $G = (V, T, R, S)$ heißt **beschränkt** gdw

$\forall (P \rightarrow Q) \in R$:

1. $|P| \leq |Q|$, **oder**

   die Regel hat die Form $S \rightarrow \varepsilon$

2. S nicht in Q

Das heißt, bei jeder Regelanwendung:

- Die Conclusio ist mindestens so lang wie die Prämisse, außer bei $\varepsilon \in L$.
- Das Wort wird nicht kürzer, außer bei $\varepsilon \in L$

Die Chomsky-Hierarchie

Aufbauend auf den Grammatikarten kann man Sprachklassen definieren

Definition 8.5 (Sprachklassen)

Klasse	definiert als	Sprache heißt
$\mathbf{L_3}$, REG	$\{L(G) \mid G$ ist rechtslinear$\}$	Typ 3, **regulär**
$\mathbf{L_2}$, CFL	$\{L(G) \mid G$ ist kontextfrei$\}$	Typ 2, **kontextfrei**
$\mathbf{L_1}$, CSL	$\{L(G) \mid G$ ist kontextsensitiv$\}$	Typ 1, **kontextsensitiv**
$\mathbf{L_1}$, CSL	$\{L(G) \mid G$ ist beschränkt$\}$	Typ 1, **beschränkt**
$\mathbf{L_0}$, r.e.	$\{L(G) \mid G$ beliebig$\}$	Typ 0, **aufzählbar**
$\mathbf{L}$	$\{L \mid L \subseteq \Sigma^* \}$	**beliebige** Sprache

Die Chomsky-Hierarchie

Grammatiken können kompliziert sein!

Beispiel 8.6 (Grammatik für $a^n b^n c^n$)

Grammatik $G_{abc} = (\{S, X_1, X_2\}, \{a, b, c\}, \{R_1, \ldots R_5\}, S)$ mit

$$
\begin{array}{lllll}
R_1 = & S & \rightarrow & abc \mid aX_1bc \\
R_2 = & X_1b & \rightarrow & bX_1 \\
R_3 = & X_1c & \rightarrow & X_2bcc \\
R_4 = & bX_2 & \rightarrow & X_2b \\
R_5 = & aX_2 & \rightarrow & aa \mid aaX_1
\end{array}
$$

- Ist diese Grammatik **kontextsensitiv**?
- Ist sie **beschränkt**?

Teil II

Terminologie

Probleme über Sprachen

Interessante Probleme (informell)

- Ist ein gegebenes Wort in einer Sprache (definiert durch eine Grammatik) enthalten?
- Erzeugen zwei gegebene Grammatiken dieselbe Sprache?

Mit welchen Algorithmen können diese Probleme gelöst werden?

Probleme und Algorithmen im allgemeinen

Definition 9.1 (Problem, Algorithmus)

Ein **Problem** P ist die Frage, ob eine bestimmte Eigenschaft auf gegebene Objekte zutrifft.

Dabei ist eine bekannte, abzählbaren Grundmenge solcher Objekte gegeben.

Für jedes Objekt o gilt: die Eigenschaft trifft auf o zu oder nicht.

Definition 9.2 (Algorithmus)

Ein **Algorithmus** für ein Problem P ist eine Vorschrift (ein Programm), die zu beliebigem Objekt o berechnet, ob die Eigenschaft für o zutrifft oder nicht.

Probleme und Algorithmen im allgemeinen

Beispiel 9.3 (Einige Probleme)

- Für $n \in \mathbb{N}$:
  Ist n eine Primzahl?
- Für ein Wort $w \in \Sigma^*$ und
  ein Element G aus der Menge aller Grammatiken über Σ:
  Gilt $w \in L(G)$?
- Für ein Element G aus der Menge aller Grammatiken:
  Ist $L(G)$ leer (endlich, unendlich)?
- Für $(a, b, c) \in \mathbb{N}^3$:
  Hat $a^n + b^n = c^n$ eine Lösung in den natürlichen Zahlen?
- Für ein Programm p aus der Menge aller Java-Programme:
  Terminiert p?

Teil II

Terminologie

Abzählbarkeit

Definition 10.1 (Abzählbarkeit)

Eine Menge M heißt **abzählbar**, wenn

- es eine surjektive Funktion

$$f : \mathbb{N}_0 \to M$$

  gibt,
- oder M leer ist.

Intuition

Eine Menge ist abzählbar, wenn sie höchstens so mächtig wie $\mathbb{N}_0$ ist.

Abzählbarkeit

Lemma 10.2

Eine Menge M ist abzählbar, wenn es eine injektive Funktion

$$f : M \to \mathbb{N}_0$$

gibt.

Abzählbarkeit

Beispiel 10.3

Abzählbar sind:

- $\mathbb{N}_0$
- $\mathbb{Q}$
- alle endlichen Mengen
- die Vereinigung zweier abzählbarer Mengen
- die Vereiningung abzählbar vieler abzählbarer Mengen

Hilberts Hotel

David Hilbert ⋆1862, †1943

- Einer der bedeutensten und einflußreichsten Mathematiker aller Zeiten
- Professor in Königsberg und Göttingen
- Wichtige Beiträge zu
  - Logik
  - Funktionalanalysis
  - Zahlentheorie
  - Mathematische Grundlagen der Physik
  - uvm.

Diagonalisierungsargument für Überabzählbarkeit

Theorem 10.4

Die Menge $\mathbb{R}$ der rellen Zahlen ist überabzählbar.

Beweis.

Wir zeigen, dass schon das Intervall $[0,1]$ überabzählbar ist.

Annahme: Es gibt eine Aufzählung, also eine surjektive Funktion

$$f : \mathbb{N}_0 \to [0,1]$$

Dann sei

$$f(i) = 0, d_0^i d_1^i d_2^i \ldots$$

die Dezimaldarstellung der i-ten reellen Zahl. □

Diagonalisierungsargument für Überabzählbarkeit

Beweis.

Fortsetzung:

Wir definieren eine neue Zahl $d = 0,\overline{d}_0\overline{d}_1\overline{d}_2\ldots$ durch

$$\overline{d}_n = \begin{cases} d_n^n + 1 & \text{falls } d_n^n < 9 \\ 0 & \text{sonst} \end{cases}$$

d unterscheidet sich in der n-ten Stelle von d_n.

Also $d \neq d_n$ für alle $n \in \mathbb{N}_0$

Also kommt d in der Aufzählung nicht vor. Widerspruch! □

Wieviele gibt es?

Wieviele

– Grammatiken
– Sprachen
– Algorithmen

gibt es überhaupt?

Mögliche Antworten

- Endlich viele
- Unendlich viele
- Abzählbar viele
- Überabzählbar viele
- Nicht klar für Algorithmen, da dieser Begriff nicht genau definiert wurde

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.5

Gegeben: Signatur Σ, endlich oder abzählbar unendlich

Dann ist Σ^* abzählbar unendlich.

Beweis.

Σ ist abzählbar, also ist Σ^i abzählbar, $i \in \mathbb{N}$.

Σ^* ist die Vereinigung der abzählbar vielen abzählbaren Mengen Σ^i. □

Wieviele Wörter, Grammatiken gibt es?

Lemma 10.6

Gegeben: Signatur Σ, endlich oder abzählbar unendlich

Dann ist die Menge aller Grammatiken über Σ abzählbar unendlich

Beweis.

Grammatiken sind endlich und also als Wörter über einer geeigneten erweiterten Grammatik

$$\Sigma \cup V \cup \{\rightarrow, \ldots\}$$

darstellbar.

Die Menge der Wörter über dieser erweiterten Grammatik ist abzählbar (Lemma 10.5). □

Wieviele Algorithmen gibt es?

Lemma 10.7

Es gibt (nur) abzählbar viele Algorithmen.

Beweis.

Algorithmen müssen per Definition eine endliche Beschreibung haben.

Sie sind also als Wörter über einer Signatur Σ darstellbar
(für jedes abzählbare Σ).

Also sind sie abzählbar (Lemma 10.5). □

Wieviele Funktionen $f : \mathbb{N}_0 \to \mathbb{N}_0$ gibt es?

Lemma 10.8

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \to \mathbb{N}_0$.

Beweis.

Angenommen, es existiere eine Abzählung

$$f_1, f_2, \ldots f_n, \ldots$$

Dann sei

$$C : \mathbb{N}_0 \to \mathbb{N}_0 \quad \text{mit} \quad C(n) = \begin{cases} 1 & \text{falls } f_n(n) = 0; \\ 0 & \text{sonst} \end{cases}$$

$C(i) \neq f_i(i)$

Also: C ist von allen f_i verschieden.

Widerspruch!

□

Wieviele Funktionen $f : \mathbb{N}_0 \to \mathbb{N}_0$ gibt es?

Lemma 10.9

Es gibt überabzählbar viele Funktionen $f : \mathbb{N}_0 \to \{0, 1\}$.

Beweis.

Analog. □

Wieviele Sprachen gibt es?

Lemma 10.10

Gegeben eine Signatur Σ (endlich oder unendlich).

Die Menge der Sprachen über Σ ist überabzählbar.

Beweis.

Sei eine beliebige Abzählung aller Wörter über Σ gegeben:

$$w_1, w_2, \ldots$$

Dann kann man die Sprachen L über Σ mit den Funktionen $f : \mathbb{N}_0 \to \{0, 1\}$ identifizieren, vermittels

$$f(i) = 1 \quad \text{gdw} \quad w_i \in L$$

Von diesen gibt es überabzählbar viele. □

Korollar

Korollar 10.11

Nicht jede Sprache kann durch eine Grammatik dargestellt werden.

Zusammenfassung

Gegeben eine Signatur Σ

Abzählbar

- $\mathbb{N}$
- Menge aller Wörter
- Menge aller Grammatiken
- Menge aller Algorithmen

Überabzählbar

- Die Menge aller Teilmengen von $\mathbb{N}$
- Die Menge aller reellen Zahlen
- Die Menge aller Funktionen $f : \mathbb{N}_0 \to \mathbb{N}_0$ bzw. $f : \mathbb{N}_0 \to \{0, 1\}$
- Die Menge aller Sprachen