Grundlagen der Theoretischen Informatik

Sommersemester 2018

12.04.2018

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Bis jetzt

- Organisatorisches
- Literatur
- Motivation und Inhalt
- Kurzer Überblick: Logik
- Kurzer Überblick: Beweismethoden und mathematische Konzepte

Inhalt der Vorlesung

1. Terminologie
2. Endliche Automaten und reguläre Sprachen
3. Kellerautomaten und kontextfreie Sprachen
4. Turingmaschinen und rekursiv aufzählbare Sprachen
5. Berechenbarkeit, (Un-)Entscheidbarkeit
6. Komplexitätsklassen P und NP

Inhalt der Vorlesung

1. Terminologie
2. Endliche Automaten und reguläre Sprachen
3. Kellerautomaten und kontextfreie Sprachen
4. Turingmaschinen und rekursiv aufzählbare Sprachen
5. Berechenbarkeit, (Un-)Entscheidbarkeit
6. Komplexitätsklassen P und NP

Mathematische Konzepte

Grundkonzepte (z.B. aus DAS)

- Elementare Mengentheorie
  - $\{x|P(x)\}$
  - $\cup$, $\cap$, $\setminus$
- Bezug zwischen Mengen, Relationen und Funktionen **(wichtig!)**
- Elementare Gesetze der Algebra
- Strukturen wie Listen, (endliche) Folgen, Graphen, Bäume
- Wörter (Strings)

Mathematische Konzepte

Funktionen

Funktion $f : S \to S'$**:** Abbildung zwischen den Grundmengen S und S', nicht unbedingt auf allen Elementen von S definiert

Definitionsbereich D $\quad D = \{x \in S \mid \exists y \in S' \text{ mit } (x, y) \in f\}$

Wertebereich W $\quad W = \{y \in S' \mid \exists x \in S \text{ mit } (x, y) \in f\}$

f **eine totale Funktion:** f für alle Elemente in S definiert

$$\forall x \in S \; \exists y \in W \; (x, y) \in f$$

$$(x, y) \in f \qquad \mapsto \qquad f(x) = y$$

Injektiv, surjektiv, bijektiv

Injektiv: $\forall x, y (f(x) = f(y) \to x = y)$

Surjektiv: $\forall y \in S' \exists x \in S : \quad f(x) = y$

Bijektiv: injektiv + surjektiv

Terminologie

In den folgenden Abschnitten führen wir die Begriffe

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

ein.

Wir untersuchen insbesondere

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

Sprache, Grammatik

Alphabete, Wörter

Definition (Alphabet)

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

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

Alphabete, Wörter

Definition (Alphabet)

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

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

Definition (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
- $\#_a(w)$ ist die Anzahl der Vorkommen des Buchstabens a im Wort w.

Alphabete, Wörter

Beispiele

- $\Sigma_1 = \{0, 1\}$.
  Wörter über Σ_1: ε, 0, 1, 01, 1001, 100101

- $\Sigma_2 = \{a, b, c, d, e, \ldots, x, y, z\}$.
  Wörter über Σ_2: ε, a, b, aba, $baab$, *informatik*

- $\Sigma_3 = \{(,), +, -, *, a\}$.
  Wörter über Σ_3: ε, a, $(a + a)$, $(a + a) * (a - a)$, $++a)$

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

Alphabete, Wörter

Beispiele

- $\Sigma_1 = \{0, 1\}$.
  Wörter über Σ_1: ε, 0, 1, 01, 1001, 100101

  $\varepsilon \circ 01001 = 01001, \quad 01 \circ 1001 = 011001$
  $1001^3 = (1001)^3 = 100110011001$
  $01101^R = (01101)^R = 10110$

- $\Sigma_2 = \{a, b, c, d, e, \ldots, x, y, z\}$.
  Wörter über Σ_2: ε, a, b, aba, $baab$, $informatik$

  $ab \circ \varepsilon = ab, \quad aba \circ baab = ababaab$
  $aba^2 = (aba)^2 = abaaba$
  $informatik^R = kitamrofni$.

- $\Sigma_3 = \{(,), +, -, *, a\}$.
  Wörter über Σ_2: ε, a, $(a + a)$, $(a + a) * (a - a)$, $+ +a)$

Sprache

Definition (Sprache)

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

Alphabete, Wörter

Beispiele

- $\Sigma_1 = \{0, 1\}$.

  Gerade = Menge aller Wörtern über Σ_1, die die binäre Representation einer geraden Zahl sind.

- $\Sigma_2 = \{a, b, c, d, e, \ldots, x, y, z\}$.

  $\mathsf{ENGLISH} = \{w \mid w \text{ wort über } \Sigma_2, \text{ das auf English eine Bedeutung hat}\}$

- $\Sigma_3 = \{(,), +, -, *, a\}$.

  EXPR = Menge der Wörter über Σ_2, die korrekt geklammerten Ausdrücke sind.

Operationen auf Sprachen

Konkatenation:

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

oft geschrieben als LM

i-te Potenz:

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

Reverse:

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

Operationen auf Sprachen

Kleene-Hülle

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

Variante:

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

Operationen auf Sprachen

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

Operationen auf Sprachen

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 (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 (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

Übung

Welche Sprachen werden durch die folgenden regulären Ausdrücke dargestellt?

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

$\mathfrak{I}(aa) = \mathfrak{I}(a)\mathfrak{I}(a) = \{a\} \circ \{a\} = \{aa\}$

$\mathfrak{I}((a + b)^*) = \mathfrak{I}((a + b))^* = (\mathfrak{I}(a) \cup \mathfrak{I}(b))^* = (\{a\} \cup \{b\})^* = \{a, b\}^*$

$\mathfrak{I}((aa^* + bb^*)) = \mathfrak{I}(aa^*) \cup \mathfrak{I}(bb^*)) = \{a\}\{a\}^* \cup \{b\}\{b\}^* = \{a\}^+ \cup \{b\}^+$

Reguläre Ausdrücke

Übung

Geben Sie einen regulären Ausdruck über $\Sigma = \{a, b, c\}$ an, der die Sprache darstellt, die genau die Wörter enthält, die mit b beginnen.

Reguläre Ausdrücke

Übung

Geben Sie einen regulären Ausdruck über $\Sigma = \{a, b, c\}$ an, der die Sprache darstellt, die genau die Wörter enthält, die mit b beginnen.

Antwort: $b(a + b + c)^*$

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'
$w\,|\,w'$	$w + w'$
$w*$	w^*
$w+$	w^+

Reguläre Ausdrücke als Suchmuster für `grep`

Syntax bei `grep`

`grep`	Regulärer Ausdruck
ww'	ww'
$w\,\texttt{|}\,w'$	$w + w'$
w*	w^*
w+	w^+

Syntactic Sugar

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

Grammatik

Grammatik

- Beschreibt eine Sprache
- Menge von Regeln, mit deren Hilfe man Wörter ableiten kann

Grammatik: Intuition

Beispiel: Grammatik für "Hund-Katze-Sätze"

Satz	$\rightarrow$	Subjekt Prädikat Objekt
Subjekt	$\rightarrow$	Artikel Attribut Substantiv
Artikel	$\rightarrow$	ε
Artikel	$\rightarrow$	der
Artikel	$\rightarrow$	die
Artikel	$\rightarrow$	das
Attribut	$\rightarrow$	ε
Attribut	$\rightarrow$	Adjektiv
Attribut	$\rightarrow$	Adjektiv Attribut
Adjektiv	$\rightarrow$	kleine
Adjektiv	$\rightarrow$	bissige
Adjektiv	$\rightarrow$	große
Substantiv	$\rightarrow$	Hund
Substantiv	$\rightarrow$	Katze
Prädikat	$\rightarrow$	jagt
Objekt	$\rightarrow$	Artikel Attribut Substantiv

Grammatik: Intuition

Beispiel: Grammatik für "Hund-Katze-Sätze"

- durch diese Grammatik können z.B. die folgenden Sätze gebildet (abgeleitet) werden:
  - der kleine bissige Hund jagt die große Katze
  - die kleine Katze jagt der bissige Hund
  - das große Katze jagt der kleine große bissige kleine Katze

- folgende Sätze werden nicht durch diese Grammatik gebildet
  - die Katze der Hund
  - Katze und Hund
  - der Hund jagt die Katze die jagt Hund

Grammatik: Intuition

Beispiel 2: Aussagenlogische Formel, $\Pi = \{P_1, \ldots, P_n\}$

$$
\begin{array}{lcl}
F & \to & \bot \\
F & \to & \top \\
F & \to & P_1 \\
 & \ldots & \\
F & \to & P_n \\
F & \to & (\neg F) \\
F & \to & (F \wedge F) \\
F & \to & (F \vee F) \\
F & \to & (F \longrightarrow F) \\
F & \to & (F \longleftrightarrow F)
\end{array}
$$

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 (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 (Regel)

Eine Regel ist ein Element

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

Bezeichnung:

P: Prämisse

Q: Conclusio

Grammatik

Definition (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):

$$P \to_G Q \qquad \text{bzw.} \qquad P \to Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \to Q_1 \mid Q_2 \mid Q_3 \qquad \text{für} \qquad P \to Q_1,\ P \to Q_2,\ P \to Q_3$$

Grammatik

Schreibweise für Regeln

- Schreibweise für Regel (P, Q):

$$P \to_G Q \qquad \text{bzw.} \qquad P \to Q$$

- Abkürzung für mehrere Regeln mit derselben Prämisse:

$$P \to Q_1 \mid Q_2 \mid Q_3 \qquad \text{für} \qquad P \to Q_1,\ P \to Q_2,\ P \to Q_3$$

Konvention (meistens)

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

Grammatik

Beispiel

$$
\begin{array}{lcl}
S & \rightarrow & B \\
B & \rightarrow & \mathit{do}\ \mathit{begin}\ B\ \mathit{end} \\
B & \rightarrow & A \\
A & \rightarrow & \mathit{nop}\ A \\
A & \rightarrow & \varepsilon
\end{array}
$$

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

1. $aktuellWort := S$ (Startsymbol)
2. Wähle eine Regel $P \to Q$, 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

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

1. *aktuellWort* $:= S$ (Startsymbol)
2. Wähle eine Regel $P \to Q$, 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*

Rechnung einer Grammatik

Algorithmus

Eingabe: Eine Grammatik

1. *aktuellWort* $:= S$ (Startsymbol)
2. Wähle eine Regel $P \to Q$, 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:

Welche Wörter kann man ableiten?

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

$$R_1 = S \to aS$$
$$R_2 = S \to \epsilon$$

Rechnung einer Grammatik

Beispiel:

Welche Wörter kann man ableiten?

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

$$\begin{aligned} R_1 &= S \to aS \\ R_2 &= S \to \epsilon \end{aligned}$$

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

$$\begin{aligned} R_1 &= S \to aSb \\ R_2 &= S \to \epsilon \end{aligned}$$

Rechnung einer Grammatik

Beispiel:

Welche Wörter kann man ableiten?

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

$$R_1 = S \to aS$$
$$R_2 = S \to \epsilon$$

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

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

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

$$R_1 = S \to 1S \,|\, 2S_0 \,|\, 3S \,|\, 4S_0 \,|\, 5S \,|\, 6S_0 \,|\, 7S \,|\, 8S_0 \,|\, 9S$$
$$R_2 = S_0 \to S \,|\, \epsilon$$

Rechnung einer Grammatik

Beispiel:

Welche Wörter kann man ableiten?

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

$$R_1 = S \to aS$$
$$R_2 = S \to \epsilon$$

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

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

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

$$R_1 = S \to 1S \,|\, 2S_0 \,|\, 3S \,|\, 4S_0 \,|\, 5S \,|\, 6S_0 \,|\, 7S \,|\, 8S_0 \,|\, 9S$$
$$R_2 = S_0 \to S \,|\, \epsilon$$

Beispiel

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

$$R_1 \quad = \quad S \to aSb$$
$$R_2 \quad = \quad S \to \epsilon$$

Mögliche Ableitung:

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

Also: $a^3 b^3 \in L(G_{ab})$

Rechnung einer Grammatik

Definition (Ableitung, Rechnung)

Gegeben:

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

Rechnung einer Grammatik

Definition (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

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

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

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

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

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

- $w = w_0$
- $w_n = w'$
- $w_i \Longrightarrow_G w_{i+1}$ 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

Beispiel

Beispiel

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

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

Beispiel

Beispiel

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

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

Drei Möglichkeiten, das Wort *abbac* zu erzeugen:

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