Diskrete Mathematik II

Alexander May

Fakultät für Mathematik
Ruhr-Universität Bochum

Sommersemester 2011

Organisatorisches

- Vorlesung: **Mo 12-14** in HZO 70 , **Di 09-10** in NA 6/99 (3+1 SWS, 6.75 CP)
- Übung: **Di 10-12** in NA 5/99
  - Assistent: **Gottfried Herold**, Korrektor: **Ilya Ozerov**
  - Präsenzübung ist **zweiwöchentlich**: 05.04., 19.04., 03.05., . . .
  - Vorrechenübung ist **zweiwöchentlich**: 12.04., 26.04., 10.05., . . .
    Abgabe der Übungen am selben Tag **vor** der Vorlesung.
  - Gruppenabgaben bis 3 Personen
  - Bonussystem:
    1/3-Notenstufe für 50%, 2/3-Notenstufe für 75%
    Gilt nur, wenn man die Klausur besteht!
- Klausur: September(?)

Themengebiete

1. Komplexitätstheorie
   - Klassen P und NP
   - Reduktionen
   - Anwendung: Sicherheitsbeweise in der Kryptographie
2. Algorithmische Zahlentheorie
   - Quadratische Reste
   - Beispiel Anwendungen: Zufallszahlengenerator, Identity-Based Encryption
3. Kodierungstheorie
   - Komprimierende Codes
   - Beispiel Anwendungen: Kommunikation (Mobilfunk, Internet), Speicher (MP3)

   - Fehlererkennende Codes
   - Ausfalltolerante Codes
   - Beispiel Anwendungen: Mobilfunk, Internet, CD, Secret Sharing, Kryptosystem

Weiterführende Referenzen

Ziel: Einfaches aber mächtiges Rechnermodell.

- Michael R. Garey, David S. Johnson, “Computers and Intractability”, Freeman, 2000
- J. Blömer, “Einführung in Algorithmen und Komplexität”, Vorlesungsskript Universität Paderborn, 2002
- N. Koblitz, “A Course in Number Theory and Cryptography”, Springer Verlag, 1994
- Steven Roman, “Introduction to Coding and Information Theory”, Springer Verlag, 1996

Einführung in die NP-Vollständigkeitstheorie

Notationen

- Alphabet $A = \{a_1, \ldots, a_m\}$ aus Buchstaben a_i
- Worte der Länge n sind Elemente aus $A^n = \{a_{i_1} \ldots a_{i_n} \mid a_{i_j} \in A\}$.
- $A^0 = \{\epsilon\}$, wobei ϵ das leere Wort ist.
- $A^* = \bigcup_{n=0}^{\infty} A^n, A^+ = A^* \setminus \{\epsilon\}, A^{\leq m} = \bigcup_{n=0}^{m} A^n$
- Länge $|a_1 \ldots a_n| = n$. $\mathrm{bin}(a_1)$ ist Binärkodierung von a_1.

Definition Sprache L

Sei A ein Alphabet. Eine Menge $L \subseteq A^*$ heißt *Sprache* über dem Alphabet A. Das *Komplement* von L über A ist definiert als $\bar{L} = A^* \setminus L$.

Turingmaschine (informal)

Turingmaschine besteht aus:

- Einseitig unendlichem Band mit Zellen (Speicher),
- Kontrolle und einem Lesekopf, der auf einer Zelle steht.

Arbeitsweise einer Turingmaschine

- Bandsymbol $\triangleright$ steht in der Zelle am linken Bandende.
- Kontrolle besitzt Zustände einer endlichen Zustandsmenge.
- Abhängig vom Zelleninhalt und Zustand schreibt die Kontrolle ein Zeichen und bewegt den Lesekopf nach links oder rechts.
- Zu Beginn der Berechnung gilt:
  - Lesekopf befindet sich auf dem linken Bandende $\triangleright$.
  - Band enthält $\triangleright a_1 \dots a_n \sqcup \sqcup \dots$, wobei $a_1 \dots a_n$ die Eingabe ist.
- Turingmaschine M hält gdw Kontrolle im Zustand q_a oder q_r.
  - Falls M in q_a hält, so akzeptiert M die Eingabe $a_1 \dots a_n$.
  - Falls M in q_r hält, so verwirft M die Eingabe $a_1 \dots a_n$.
  - Falls M nie in die Zustände q_a, q_r kommt: M läuft unendlich.

Turingmaschine (formal)

Definition Deterministische Turingmaschine (Turing 1936)

Eine deterministische Turingmaschine DTM ist ein 4-Tupel $(Q, \Sigma, \Gamma, \delta)$ bestehend aus

1. Zustandmenge Q: Enthält Zustände q_a, q_r, s.
2. Bandalphabet Γ mit $\sqcup, \rhd \in \Gamma$
3. Eingabealphabet $\Sigma \subseteq \Gamma \setminus \{\sqcup, \rhd\}$.
4. Übergangsfunktion $\delta : Q \setminus \{q_a, q_r\} \times \Gamma \to Q \times \Gamma \times \{L, R\}$
   - Es gilt stets $\delta(q, \rhd) = (q', \rhd, R)$ (am linken Bandende).
   - Es gilt nie $\delta(q, a) = (q', \rhd, L/R)$ (nicht am linken Bandende).

Beispiel DTM M_1

Bsp: $a^n, n \geq 1$

- $Q = \{q_0, q_1, q_a, q_r\}$ mit $s = q_0$
- $\Sigma = \{a\}$ und $\Gamma = \{\sqcup, \rhd, a\}$
- Übergangsfunktion

δ	a	$\sqcup$	$\rhd$
q_0	(q_1, a, R)	$(q_r, \sqcup, R)$	$(q_0, \rhd, R)$
q_1	(q_1, a, R)	$(q_a, \sqcup, R)$	$(q_1, \rhd, R)$

Notation der Konfigurationen bei Eingabe a^2:

$$
\begin{array}{rl}
 & q_0 \rhd aa \\
\vdash & \rhd q_0 aa \\
\vdash & \rhd a q_1 a \\
\vdash & \rhd aa q_1 \sqcup \\
\vdash & \rhd aa \sqcup q_a \sqcup
\end{array}
$$

Nachfolgekonfigurationen

Notation Nachfolgekonfiguration

- Direkte Nachfolgekonfiguration: $aqb \vdash a'q'b'$
- i-te Nachfolgekonfiguration: $aqb \vdash^i a'q'b'$
- Indirekte Nachfolgekonfiguration $aqb \vdash^* a'b'q'$, d.h. $\exists i \in \mathbb{N} : aqb \vdash^i a'q'b'$.

Akzeptanz und Ablehnen von Eingaben

- DTM M erhalte Eingabe $w \in \Sigma^*$.
  - M akzeptiert $w \Leftrightarrow \exists a, b \in \Gamma^* \text{ mit } s \triangleright w \vdash^* aq_a b$
  - M lehnt w ab $\Leftrightarrow \exists a, b \in \Gamma^* \text{ mit } s \triangleright w \vdash^* aq_r b$

Akzeptierte Sprache, L rekursiv aufzählbar

Definition Akzeptierte Sprache, Rekursive Aufzählbarkeit

Sei M eine DTM. Dann ist die von M *akzeptierte Sprache*

$$L(M) = \{w \in \Sigma^* \mid M \text{ akzeptiert Eingabe } w\}.$$

Eine Sprache L heißt *rekursiv aufzählbar* gdw eine DTM M existiert mit $L = L(M)$.

- Unsere Beispiel-DTM M_1 akzeptiert die Sprache $L(M_1) = \{a\}^+$.
- D.h. $L = \{a\}^+$ ist rekursiv aufzählbar, da für M_1 gilt $L = L(M_1)$.
- Aus der obigen Definition folgt:
  L ist nicht rekursiv aufzählbar $\Leftrightarrow \nexists$ DTM M mit $L = L(M)$.
- Es gibt Sprachen, die nicht rekursiv aufzählbar sind, z.B.
  $\bar{H} = \{\langle M, x\rangle \mid \text{DTM } M \text{ hält bei Eingabe } x \text{ nicht.}\}$. (ohne Beweis)

Entscheidbarkeit und rekursive Sprachen

Definition Entscheidbarkeit

Sei M eine DTM, die die Sprache $L(M)$ akzeptiert. M *entscheidet* die Sprache $L(M)$ gdw M alle Eingaben $w \notin L(M)$ ablehnt. D.h. insbesondere M hält auf allen Eingaben.
Eine Sprache L heißt *entscheidbar* gdw eine DTM M existiert, die L entscheidet.

- Unsere Beispiel-DTM M_1 entscheidet die Sprache $L(M_1) = \{a\}^+$.
- $L = \{a\}^+$ ist entscheidbar, da M_1 die Sprache L entscheidet.

Korollar Entscheidbarkeit impliziert rekursive Aufzählbarkeit

Sei L eine entscheidbare Sprache. Dann ist L rekursiv aufzählbar.

- Die Rückrichtung stimmt nicht:
  Es gibt rekursiv aufzählbare L, die nicht entscheidbar sind, z.B.
  $H = \{\langle M, x\rangle \mid \text{DTM } M \text{ hält auf Eingabe } x.\}$. (ohne Beweis)

Entscheiden versus Berechnen

Definition Berechnung von Funktionen

Eine DTM M berechnet die Funktion $f : \mathbb{N}^n \to \mathbb{N}$, falls M für jedes $(a_1, \ldots, a_n)$ bei Eingabe $\mathrm{bin}(a_1)\# \ldots \#\mathrm{bin}(a_n)$ den Bandinhalt $\mathrm{bin}(f(a_1, \ldots, a_n))$ berechnet und in q_a hält.

- Werden der Einfachheit halber Sprachen entscheiden, nicht Funktionen berechnen.

Laufzeit einer DTM, Klasse DTIME

Definition Laufzeit einer DTM

Sei M eine DTM mit Eingabealphabet Σ, die bei jeder Eingabe hält. Sei $T_M(w)$ die Anzahl der Rechenschritte – d.h. Bewegungen des Lesekopfes von M – bei Eingabe w. Dann bezeichnen wir die Funktion

$$T_M(n) : \mathbb{N} \to \mathbb{N} \text{ mit } T_M(n) = \max\{T_M(w) \mid w \in \Sigma^{\leq n}\}$$

als *Zeitkomplexität* bzw. *Laufzeit* der DTM M.

- Die Laufzeit wächst monoton in n.
- Unsere Beispiel-DTM M_1 mit $L(M_1) = \{a\}^*$ besitzt Laufzeit $\mathcal{O}(n)$.

Definition DTIME

Sei $t : \mathbb{N} \to \mathbb{N}$ eine monoton wachsende Funktion. Die Klasse DTIME ist definiert als

$$\mathrm{DTIME}(t(n)) := \{L \mid L \text{ wird von DTM mit Laufzeit } \mathcal{O}(t(n)) \text{ entschieden.}\}.$$

- Es gilt $L(M_1) \in DTIME(n)$.

Registermaschine RAM

Registermaschine RAM besteht aus den folgenden Komponenten:

- Eingabe-/ und Ausgabe-Register
- Speicherregister
- Programm
- Befehlszähler
- Akkumulator

Funktionsweise einer RAM:

- Liest Eingabe aus Eingaberegister und lässt Programm auf Eingabe laufen.
- Führt Arithmetik im Akkumulator aus.
- Ergebnisse können im Speicherregister gespeichert werden.
- Befehlszähler realisiert Sprünge, Schleifen und bedingte Anweisungen im Programm.
- Ausgabe erfolgt im Ausgaberegister.

DTMs versus RAMs, Churchsche These

Fakt Polynomielle Äquivalenz von DTMs und RAMs

Sei $t : \mathbb{N} \to \mathbb{N}$ eine monoton wachsende Funktion mit $t(n) \geq n$. Jede RAM mit Laufzeit $t(n)$ kann durch eine DTM M mit Laufzeit $\mathcal{O}(t(n)^3)$ simuliert werden.

Churchsche These (1936)

"Die im intuitiven Sinne berechenbaren Funktionen sind genau die durch Turingmaschinen berechenbaren Funkionen."

- These ist nicht beweisbar oder widerlegbar.
- Alle bekannten Berechenbarkeitsbegriffe führen zu DTM-berechenbaren Funktionen.

Die Klasse $\mathcal{P}$

Definition Klasse $\mathcal{P}$

Die Klasse $\mathcal{P}$ ist definiert als

$$\mathcal{P} = \bigcup_{k \in \mathbb{N}} DTIME(n^k).$$

- $L \in \mathcal{P}$ gdw eine DTM existiert, die L in Laufzeit $\mathcal{O}(n^k)$ entscheidet.
- $\mathcal{P}$ ist die Klasse aller in Polynomialzeit entscheidbaren Sprachen. (auf DTMs, RAMs, etc.)
- Hintereinanderausführung/Verzahnung von DTMs mit polynomieller Laufzeit liefert polynomielle Gesamtlaufzeit.
- $\mathcal{P}$ beinhaltet praktische und theoretisch interessante Probleme.
- Probleme ausserhalb von $\mathcal{P}$ sind in der Praxis oft nur für kleine Instanzen oder approximativ lösbar.