Komplexitätstheorie

Kapitel 2: Turingmaschinen

Einleitung

Wir möchten auch Aussagen über alle Algorithmen treffen
(z.B. es gibt keinen, der ein bestimmtes Problem in Polyzeit löst)

Um in diesem Kontext formale Beweise führen zu können, müssen wir festlegen, was ein Algorithmus ist.

Prinzipiell gibt es viele Alternativen:

- Realistische Modelle: z.B. Java-Programme oder C-Programme
- Mathematische Modelle, inspiriert von echten Rechnern:

  z.B. Registermaschinen (RAMs)
- Rein mathematische Modelle: z.B. Turingmaschinen

Da wir einfache Beweise wollen, wählen wir ein Modell, das so einfach wie möglich ist: Turingmaschinen

Einleitung

Turingmaschinen...

- wurden in den 1930ern von Alan Turing entwickelt als Modell für menschliches Rechnen (mit Zettel und Bleistift)
- sind sehr einfach aufgebaut (Papadimitriou: "it is amazing how little we need to have everything")
- sind kein realistisches Modell für reale Computer, leisten jedoch interessanterweise genau dasselbe

Church-Turing These

Die durch Turingmaschinen berechenbaren Probleme sind genau die im intuitiven Sinn berechenbaren Probleme.

"These", denn läßt sich nicht beweisen ("im intuitiven Sinn berechenbar" kann man nicht formal fassen)

Einleitung

Die Church-Turing-These rechtfertigt noch nicht die Verwendung von Turingmaschinen in der Komplexitätstheorie!

Erweiterte Church-Turing These

Problem kann mit Zeitverbrauch t in natürlichem und generellen Berechnungsmodell gelöst werden gdw. es mit Zeitverbrauch $p(t)$ von Turingmaschinen gelöst werden kann, für ein Polynom p.

Zeitverbrauch wird gemessen durch Anzahl elementarer Schritte (was das ist, hängt vom Berechnungsmodell ab)

Wenn wir "effizient" mit "in polynomieller Zeit" lösbar gleichsetzen: Problem ist entweder in allen Modellen effizient berechenbar oder in keinem.

Der Grad des Polynoms kann sich aber u.U. unterscheiden.

Kapitel 2

Turingmaschinen

Turingmaschinen

Turingmaschine

FOLIE RAUS?!?!

- ist zu jeder Zeit in einem von endlich vielen Zuständen
- Arbeitet auf einseitig unendlichem Arbeitsband, das aus Kette von linear geordneten Zellen besteht
- Jede Zelle enthält eines von endlich vielen Symbolen
- Zu Anfang enthält das Band ganz links das Sondersymbol $\triangleleft$ gefolgt von der Eingabe, gefolgt von unendl. oft Sondersymbol $\perp$
- Die Maschine hat einen Schreib-/Lesekopf, der zu Anfang auf dem ersten (linken) Symbol der Eingabe steht
- In jedem Schritt kann die Maschine das Symbol der aktuellen Zelle lesen und schreiben, sowie den Kopf um eine Position verschieben
- Arbeitet feste Verarbeitungsvorschrift ab

Turingmaschinen

Definition Turingmaschine

(Deterministische) Turingmaschine (kurz: (D)TM) ist Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

mit

- Q endlicher Menge von Zuständen
- Σ endliches Eingabealphabet
- Γ endliches Bandalphabet mit $\Sigma \subseteq \Gamma$ und $\{\bot, \triangleleft\} \subseteq \Gamma \setminus \Sigma$
- $\delta : ((Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma) \to (Q \times \Gamma \times \{L, R, N\})$ Übergangsfunktion
- $q_0 \in Q$ Startzustand
- q_{acc} akzeptierender Endzustand
- q_{rej} verwerfender Endzustand

Turingmaschinen

Konvention: Wenn $\delta(q, a) = (q', a', B)$ verlangen wir, dass

- wenn $a = \triangleleft$, dann $a' = \triangleleft$ und $B = R$;
- wenn $a \neq \triangleleft$, dann $a' \neq \triangleleft$.

Also: $\triangleleft$ bleibt stets am linken Bandende und steht sonst nirgendwo.

Definition Konfiguration

Konfiguration einer TM M hat Form uqv, wobei

- $u \in \{\triangleleft\} \cdot (\Gamma \setminus \{\triangleleft\})^*$ Bandbeschriftung links des Kopfes,
- $q \in Q$ der momentane Zustand,
- $v \in (\Gamma \setminus \{\triangleleft\})^*$ Bandbeschriftung ab (inkl.) Kopf nach rechts.

Die *Länge* $|\alpha|$ einer Konfiguration $\alpha = uqv$ ist $|uv|$.

Turingmaschinen

Mehr Begriffe:

- Für Eingabe $w \in \Sigma^*$ ist $\triangleleft q_0 w$ die *Startkonfiguration*;
- β ist *Folgekonfiguration* von α wenn β aus α in einem Schritt hervorgeht; wir schreiben $\alpha \vdash_M \beta$, formale Def. als Übung Folgekonfigurationen von DTMs sind eindeutig!
- Konfiguration uqv ist *akzeptierend* wenn $q = q_{\mathsf{acc}}$, *verwerfend* wenn $q = q_{\mathsf{rej}}$;
- *Berechnung* auf Eingabe w ist Folge von Konfigurationen $\alpha_0, \alpha_1, \ldots$ (endl. oder unendl.) mit α_0 Startkonfig. für w und $\alpha_i \vdash_M \alpha_{i+1}$ für $i \geq 0$;
- Endl. Berechnung $\alpha_0, \ldots, \alpha_k$ ist *akzeptierend* wenn α_k akzeptierend, verwerfend wenn α_k verwerfend.

Turingmaschinen

Mehr Begriffe:

- DTM *akzeptiert* Eingabe w wenn (eindeutige!) maximale Berechnung auf w akzeptierend ist, sonst *verwirft* M die Eingabe w
- Von TM M *erkannte* Sprache ist

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

Anm.: Es gibt zwei Wege, auf denen DTM Eingabe verwerfen kann:

- Berechnung endet in verwerfendem Zustand
- Berechnung ist unendlich (TM terminiert nicht)

Kapitel 2

Probleme lösen mittels TMs

Entscheidungsprobleme

Eingabe für TM ist (endl.) Wort aus Σ^*.

Andere Eingaben (Graphen, Zahlen, etc) müssen kodiert werden.

Definition Problem

(Entscheidungs)problem ist Teilmenge $L \subseteq \Sigma^*$, für ein endliches Alphabet Σ.

Idee: das Problem ist die Menge aller "Ja-Instanzen", z.B.:

- GAP ist die Menge aller Tripel (G, v, v') mit $G = (V, E)$ Graph und $v, v' \in V$ (geeignet kodiert), so dass v' von v erreichbar in G
- CLIQUE ist die Menge aller Paare (G, k) so dass G eine k-Clique hat

Wir geben die Kodierung i.d.R. nicht explizit an.

Entscheidungsprobleme

Wir setzen "Turingmaschine" mit "Algorithmus" gleich

Wir interessieren uns für Algorithmen, die auf jeder Eingabe anhalten (denn nur solche Algorithmen lösen ein Entscheidungsproblem).

Definition Entscheidungsverfahren

TM M *entscheidet* Problem L: M terminiert auf jeder Eingabe und $L(M) = L$. Dann ist M *Entscheidungsverfahren* für L.

Also: wenn M Entscheidungsverfahren für L ist, dann

- akzeptiert M jedes $w \in \Sigma$ mit $w \in L$ durch halten in q_{acc};
- verwirft M jedes $w \in \Sigma$ mit $w \notin L$ durch halten in q_{rej}.

Entscheidungsprobleme

Definition Entscheidbar

Problem L ist *entscheidbar* gdw. es TM M gibt, die L entscheidet.

Sonst ist L *unentscheidbar*.

Theorie der Berechenbarkeit (veraltet: "Rekursionstheorie"):

studiert Entscheidbarkeit/Unentscheidbarkeit

Komplexitätstheorie:

- studiert entscheidbare Probleme
- enthüllt reiche Struktur innerhalb der Klasse der entscheidbaren Probleme

Kapitel 2

Mehrband Turingmaschinen

Mehrband TM

(Deterministische) Mehrband TM hat mehrere Bänder (endlich viele):

- wie Einband TM ist sie zu jedem Zeitpunkt in einem Zustand (nicht ein Zustand pro Band!)
- Es gibt einen Schreib-/Lesekopf pro Band
- in einem Schritt werden alle Bänder gleichzeitig gelesen und geschrieben
- Die Köpfe bewegen sich unabhängig voneinander (z.B.: einer nach links, ein anderer nach rechts)
- Die Eingabe ist auf dem ersten Band (*Eingabeband*) und alle anderen Bänder sind initial mit $\perp$ beschriftet

Mehrband TM

Formal, bei k Bändern:

- Übergangsfunktion hat jetzt die Form

$$((Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}) \times \Gamma^k) \to (Q \times \Gamma^k \times \{L, R, N\}^k)$$

- bei $\delta(q, a_1, \ldots, a_k) = (q', a'_1, \ldots, a'_k, B_1, \ldots, B_k)$ ist
  a_i Symbol, das auf i-tem Band gelesen wird,
  a'_i Symbol, das auf i-tem Band geschrieben wird,
  B_i Bewegung des i-ten Kopfes
- Konfiguration hat Form $(q, u_1, v_1, \ldots, u_k, v_k)$, mit
  u_i Beschriftung Band i links des Kopfes,
  v_i Beschriftung i-tes Band ab (inkl.) Kopfposition
- Akzeptanz ist wie für Einband DTMs definiert.

Kapitel 2

Zeit-Komplexitätsklassen

Komplexitätsklassen

Komplexitätsklasse ist Klasse von Problemen, die mit einer bestimmten “Menge” einer Ressource gelöst werden können.

Je nach betrachteter Ressource:

- Zeitkomplexitätsklassen
- Platzkomplexitätsklassen

Menge z.B.:

- polynomiell viel
- exponentiell viel

Wesentliche Fragestellung der Komplexitätstheorie:

wie ist der Zusammenhang zwischen verschiedenen K.-Klassen?

Zeitkomplexitätsklassen

Definition Zeitbeschränkt, DTime

Für DTM M und $w \in \Sigma^*$ schreiben wir $\text{time}_M(w) = n$ wenn M auf w nach n Schritten hält.

Sei $t : \mathbb{N} \to \mathbb{N}$ monoton wachsende Funktion mit $t(n) \geq n$.
M ist t-*zeitbeschränkt* wenn $\text{time}_M(w) \leq t(n)$ für alle w der Länge n

Wir definieren Zeitkomplexitätsklasse abhängig von Bandzahl:

$$\text{DTime}_k(t) := \{L \subseteq \Sigma^* \mid \exists \mathcal{O}(t)\text{-zeitbeschränkte } k\text{-Band DTM } M \text{ mit } L(M) = L\}$$

Z.B.:

- $\text{DTime}_1(n^2)$ Menge der Probleme, die in quadratischer Zeit von 1-Band TM entschieden werden können;
- $\text{DTime}_2(2^n)$ Menge aller Probleme, die in Zeit 2^n von 2-Band TM entschieden werden können.

Mehrband TMs

Viele Bänder vs. wenige:

Bandreduktion / Satz von Hennie und Stearns

Für alle $k \geq 1$ und t:

1. $\mathsf{DTime}_k(t) \subseteq \mathsf{DTime}_1(t^2)$;
2. wenn es $\varepsilon > 0$ gibt mit $t(n) \geq (1+\varepsilon)n$, dann gilt $\mathsf{DTime}_k(t) \subseteq \mathsf{DTime}_2(t \cdot \log(t))$.

Überblick Beweis von 1.:

- Repräsentiere k Bänder als 2k "Spuren" auf einem Band
- k Spuren für Bandbeschriftung, k Spuren für Kopfmarker
- Simuliere Schritt der k-Band TM durch zweimaliges Ablaufen des Bandes.

Mehrband TMs

Daher ist folgende Definition natürlich:

Definition DTime

Für alle monoton wachsenden $t : \mathbb{N} \to \mathbb{N}$ mit $t(n) \geq n$:

$$\mathsf{DTime}(t) := \bigcup_{k \geq 1} \mathsf{DTime}_k(t).$$

Im folgenden:

- Sind alle weiteren deterministischen Zeitkomplexitätsklassen mittels DTime definiert
- Also: wir verwenden beliebige (endliche) Bandzahl je nach Bedarf

Kapitel 2

Universelle Turingmaschinen

Universelle Turingmaschinen

Wir werden von Zeit zu Zeit universelle Turingmaschinen verwenden, die alle anderen DTMs simulieren können.

Sie sind sozusagen Interpreter für DTMs, implementiert als DTM

Zunächst: jede DTM lässt sich als ein Wort darstellen.

Einige vereinfachende Annahmen (ohne wesentliche Einschränkung):

- alle DTMs verwenden das Eingabealphabet $\Sigma = \{a, b\}$
- alle DTMs verwenden nur Symbole aus dem Alphabet $\{a_1, a_2, a_3, \dots\}$ wobei $a_1 = a$, $a_2 = b$, $a_3 = \triangleleft$ und $a_4 = \perp$
- alle DTMs verwenden nur Zustände aus der Menge $\{q_1, q_2, q_3, \dots\}$ wobei q_1 immer der Startzustand ist, $q_2 = q_{\mathsf{acc}}$ und $q_3 = q_{\mathsf{rej}}$

Universelle Turingmaschinen

DTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ dargestellt als Wort:

Transition $t =$ $(q_i, a_j) \mapsto (q_{i'}, a_{j'}, \begin{matrix} L \\ R \\ N \end{matrix})$

dargestellt als

$$\text{code}(t) = a^i b a^j b a^{i'} b a^{j'} b \begin{matrix} a \\ aa \\ aaa \end{matrix} b$$

Besteht δ aus Transitionen $t_1, \ldots, t_k$ so wird M dargestellt als:

$$\text{code}(M) = \text{code}(t_1) \cdots \text{code}(t_n) b$$

Konfiguration $K = a_{i_1} \cdots a_{i_k} q_j a_{i_{k+1}} \cdots a_{i_\ell}$ darstellbar als

$$\text{code}(K) = a^{i_1} b \cdots b a^{i_k} b a^j b b a^{i_{k+1}} b \cdots b a^{i_\ell}$$

Universelle Turingmaschinen

Theorem

Es gibt eine *universelle DTM* U, d.h. U erfüllt folgende Eigenschaft: für alle DTMs M und Eingaben w für M gilt:

$$U \text{ akzeptiert } \mathsf{code}(M)w \quad \text{gdw} \quad M \text{ akzeptiert } w.$$

Ideen:

- erzeuge zunächst $\mathsf{code}(M)\mathsf{code}(q_0 w)$
- wenn M bei Eingabe w die Berechnung $q_0 w = K_1 \vdash_M K_2 \vdash_M \cdots$ ausführt, erzeuge sukzessive

  $$\mathsf{code}(M)\mathsf{code}(K_1),\ \mathsf{code}(M)\mathsf{code}(K_2),\ \mathsf{code}(M)\mathsf{code}(K_3),\ \ldots$$

- jeder einzelne Schritt wird durch mehrmaliges Hin- und herlaufen implementiert

Übersicht Vorlesung

- Kapitel 1: Einführung
- Kapitel 2: Turingmaschinen
- Kapitel 3: P vs. NP
- Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?
- Kapitel 5: Platzkomplexität
- Kapitel 6: Schaltkreise
- Kapitel 7: Orakel