Komplexitätstheorie

Kapitel 2: Turingmaschinen

Einleitung

Wir interessieren uns nicht für konkrete Algorithmen, sondern für alle Algorithmen, die ein Problem lösen (bzw. den besten davon).

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 zu handhaben (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!

Cobham-Edmonds 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, kommt auf's Berechnungsmodell an)

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

Anm.: Der Grad des Polynoms kann sich aber durchaus unterscheiden.

Kapitel 2

Turingmaschinen

Turingmaschinen

Turingmaschine

- 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

(Ähnlich menschlichem Rechnen auf Karopapier)

Turingmaschinen

Definition Turingmaschine

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

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

mit

- Q endlicher Menge von Zuständen
- Σ endliches Eingabealphabet
- Γ endliches Bandalphabet mit $\Sigma \subseteq \Gamma$ und $\{\perp, \lhd\} \subseteq \Gamma \setminus \Sigma$
- $\delta : ((Q \setminus \{q_{\mathsf{acc}}, q_{\mathsf{rej}}\}) \times \Gamma) \to (Q \times \Gamma \times \{L, R, N\})$ 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 $a = \lhd$, dann $a' = \lhd$ und $B = R$;
- wenn $a \neq \lhd$, dann $a' \neq \lhd$.

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

Definition Konfiguration

Konfiguration einer TM M hat Form uqv, wobei

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

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

Turingmaschinen

Mehr Begriffe:

- Für Eingabe $w \in \Sigma^*$ ist $\lhd q_0 w$ 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

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

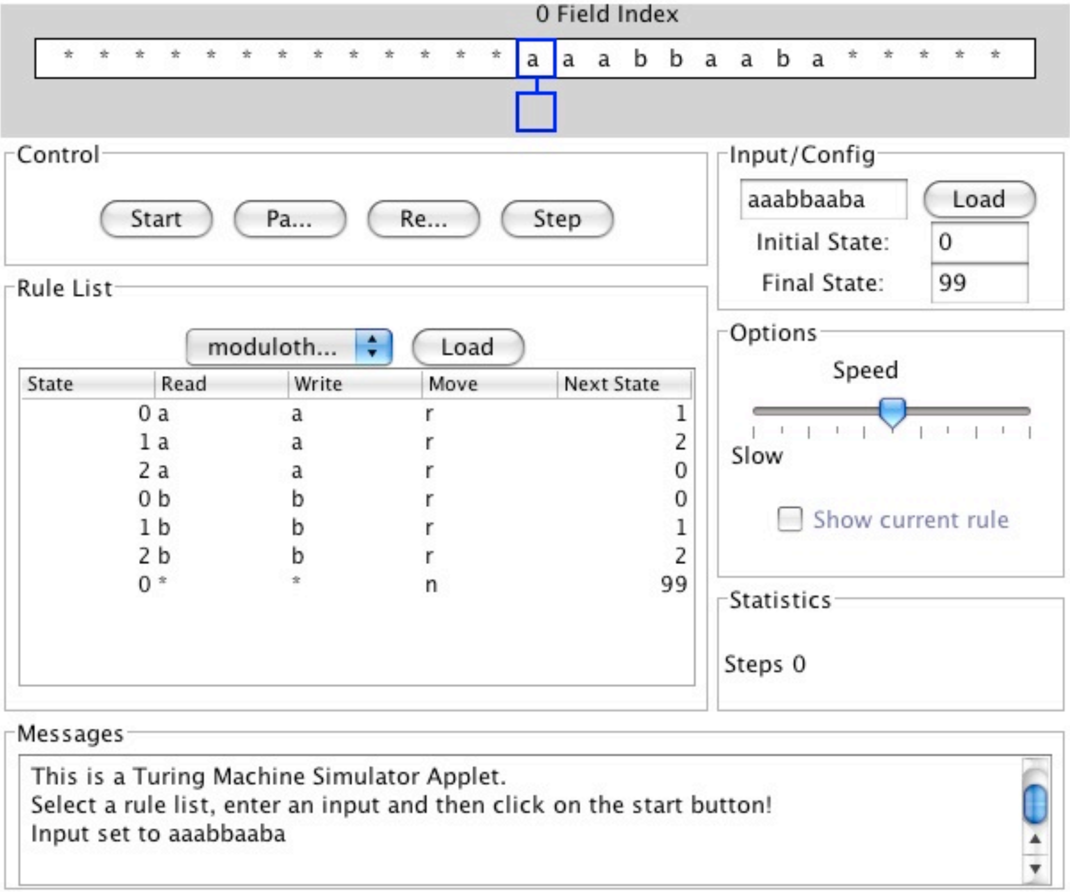
Anm.: Es gibt zwei Wege, auf denen DTM Eingabe verwerfen kann:

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

TM Simulatoren

Man findet viele Simulatoren im Netz, z.B.:

http://ais.informatik.uni-freiburg.de/turing-applet/



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 $w \in L$;
- 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 (basierend auf Ressourcenverbrauch)

Kapitel 2

Zeit-Komplexitätsklassen

Komplexitätsklassen

Komplexitätsklasse ist Klasse von Problemen, zusammengefasst nach gemeinsamer oberer Schranke des Ressourcenverbrauchs

Je nach betrachteter Ressource:

- Zeitkomplexitätsklassen
- Platzkomplexitätsklassen

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

$$\mathsf{time}_M(w) = n$$

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 $\mathsf{time}_M(w) \leq t(n)$ für alle w der Länge n

Zeitkomplexitätsklasse DTIME(t) ist definiert als Menge der Probleme

$$\{L \subseteq \Sigma^* \mid \text{es existiert } \mathcal{O}(t)\text{-zeitbeschränkte DTM } M \text{ mit } L(M) = L\}$$

Z.B.:
- DTIME(n^2) Menge der Probleme, die in quadratischer Zeit entschieden werden können;
- DTIME(2^n) Menge aller Probleme, die in Zeit 2^n entschieden werden können.

Zeitkomplexitätsklassen

Da wir uns in Zeitabschätzungen nicht für Konstanten interessieren, verwenden wir oft die Landau Symbole, insbesondere:

- $f \in \mathcal{O}(g)$ "f wächst nicht (wesentlich) schneller als g";
- $f \in \Omega(g)$ "f wächst nicht (wesentlich) langsamer als g";

Für formale Definition siehe Literatur.

Einfache Analyse der Algorithmen aus Kapitel 1 liefert:

- GAP $\in$ DTIME(n^2)
- CLIQUE $\in$ DTIME$(2^{\mathcal{O}(n^2)}) := \bigcup_{t \in \mathcal{O}(n^2)}$ DTIME$(2^{t(n)})$

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.

Mehrband TMs

Für $t : \mathbb{N} \rightarrow \mathbb{N}$ und $k \geq 1$ ist DTIME$_k(t)$ definiert wie DTIME(t), aber mittels k-Band DTMen

Also: DTIME$_1(t) =$ DTIME(t).

Viele Bänder vs. wenige:

Bandreduktion / Satz von Hennie und Stearns

Für alle $k \geq 1$ und t:

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

Mehrband TMs

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.