

11. Woche: Turingmaschinen und Komplexität
Rekursive Aufzählbarkeit, Entscheidbarkeit
Laufzeit, Klassen DTIME und \mathcal{P}

Einführung in die NP-Vollständigkeitstheorie

Notationen

- Alphabet $A = \{a_1, \dots, a_m\}$ aus Buchstaben a_i
- Worte der Länge n sind Elemente aus $A^n = \{a_{i_1} \dots a_{i_n} \mid a_{i_j} \in A\}$.
- $A^0 = \epsilon$, ϵ ist das leere Wort.
- $A^* = \bigcup_{n=0}^{\infty} A^n$, $A^+ = A^* \setminus \epsilon$, $A^{\leq m} = \bigcup_{n=0}^m A^n$
- Länge $|a_1 \dots a_n| = n$. $\text{bin}(a_1)$ ist Binärcodierung 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.
- \sqcup bedeutet „Bandzelle noch nicht beschrieben“.
- 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 nur, falls Kontrolle in Zuständen q_a oder q_r .
 - ▶ Falls M in q_a hält: M akzeptiert (accepts) die Eingabe $a_1 \dots a_n$.
 - ▶ Falls M in q_r hält: M verwirft (rejects) 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 7-Tupel $(Q, \Sigma, \Gamma, \delta, s, \sqcup, E)$ bestehend aus

- 1 Zustandmenge Q : Enthält Zustände q_a, q_r, s .
- 2 Bandalphabet Γ mit $\sqcup, \triangleright \in \Gamma$
- 3 Eingabealphabet $\Sigma \subseteq \Gamma \setminus \{\sqcup, \triangleright\}$.
- 4 Bewegungen: E ist entweder $\{R, L\}$ oder $\{R, N, L\}$ wobei $L =$ Links, $R =$ Rechts, $N =$ Keine Bewegung.
- 5 Übergangsfunktion $\delta : Q \setminus \{q_a, q_r\} \times \Gamma \rightarrow Q \times \Gamma \times E$
 - ▶ Es gilt stets am linken Bandende $\delta(q, \triangleright) = (q', \triangleright, R)$.
 - ▶ Es gilt nie $\delta(q, a) = (q', \triangleright, L/R)$ (nicht am linken Bandende).

Man kann zeigen, dass die Definitionen mit $E = \{R, L\}$ und $E = \{R, N, L\}$ äquivalent sind (polynomielle Äquivalenz: siehe später).

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 $\Gamma = \{\sqcup, \triangleright, a\}$
- Übergangsfunktion

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

Notation der Konfigurationen bei Eingabe a^2 :

$q_0 \triangleright aa$
 $\vdash \triangleright q_0 aa$
 $\vdash \triangleright aq_1 a$
 $\vdash \triangleright aaq_1 \sqcup$
 $\vdash \triangleright aa \sqcup q_a \sqcup$

Nachfolgekongfigurationen

Notation Nachfolgekongfiguration

- Direkte Nachfolgekongfiguration: $aqb \vdash a'q'b'$
- i -te Nachfolgekongfiguration: $aqb \vdash^i a'q'b'$
- Indirekte Nachfolgekongfiguration $aqb \vdash^* a'b'q'$, 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 $w \Leftrightarrow \exists a, b \in \Gamma^*$ mit $s \triangleright w \vdash^* aq_a b$
 - ▶ M lehnt w ab $\Leftrightarrow \exists a, b \in \Gamma^*$ mit $s \triangleright w \vdash^* aq_r b$

Akzeptierte Sprache, L rekursiv aufzählbar

Definition Akzeptierte Sprache, Rekursive Aufzählbarkeit

Sei M eine DTM. Dann bezeichne

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

die von M *akzeptierte Sprache*.

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. Wir sagen, dass M die Sprache $L(M)$ *entscheidet* gdw M alle Eingaben $w \in \bar{L}$ 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 $f : \mathbb{N}^n \rightarrow \mathbb{N}$, falls M für jedes (a_1, \dots, a_n) bei Eingabe $\text{bin}(a_1)\# \dots \# \text{bin}(a_n)$ den Bandinhalt $\text{bin}(f(a_1, \dots, a_n))$ 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

$T_M(n) : \mathbb{N} \rightarrow \mathbb{N}$ 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 $t : \mathbb{N} \rightarrow \mathbb{N}$ eine monoton wachsende Funktion. Die Klasse DTIME ist definiert als

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

- 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 $t : \mathbb{N} \rightarrow \mathbb{N}$ 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 Funktionen.“

- 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

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

- $L \in \mathcal{P}$ 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.

Codierung der Eingabe

- Erinnerung: Zeitkomplexität $T_M(n)$ ist eine Funktion in $|w| = n$.
- Benötigen geeignete Codierung der Eingabe w .
- Codierung einer Zahl $n \in \mathbb{N}$
 - ▶ Verwenden Binärkodierung $\text{bin}(n)$ mit Eingabelänge $\Theta(\log n)$.
- Codierung eines Graphen $G = (V, E)$
 - ▶ Codieren Knotenanzahl n unär, d.h. $|V| = n$.
 - ▶ m Kanten mit Adjazenzliste $|E| = m$ oder Adjazenzmatrix $|E| = n^2$.

Bsp:

- PFAD := $\{(G, s, t) \mid G \text{ ist Graph mit Pfad von } s \text{ nach } t.\} \in \mathcal{P}$.
 - ▶ Starte Breitensuche in s .
 - ▶ Falls t erreicht wird, akzeptiere. Sonst lehne ab.
 - ▶ Laufzeit $\mathcal{O}(|V| + |E|)$, d.h. linear in der Eingabelänge von G .
- TEILERFREMD := $\{(x, y) \mid \text{gcd}(x, y) = 1\} \in \mathcal{P}$.
 - ▶ Berechne mittels Euklidischem Algorithmus $d = \text{gcd}(x, y)$.
 - ▶ Falls $d = 1$, akzeptiere. Sonst lehne ab.
 - ▶ $\mathcal{O}(\log^2(\max\{x, y\}))$, quadratisch in $|x| = \Theta(\log x)$, $|y| = \Theta(\log y)$.

Optimierungsvariante vs Entscheidungsvariante

RUCKSACK_{opt}

- Gegeben: Gegenstände $1, \dots, n$ mit Gewichten $W = \{w_1, \dots, w_n\}$ und Profiten $P = \{p_1, \dots, p_n\}$. Kapazität B .
- Gesucht: $I \subseteq [n] : \sum_{i \in I} w_i \leq B$, so dass $\sum_{i \in I} p_i$ maximiert wird.

Sprache RUCKSACK:

RUCKSACK := $\{(W, P, B, k) \mid \exists I \subseteq [n] : \sum_{i \in I} w_i \leq B \text{ und } \sum_{i \in I} p_i \geq k\}$.

Naiver Algorithmus zum Entscheiden von RUCKSACK

- 1 Für alle $I \subseteq [n]$:
 - 1 Falls $\sum_{i \in I} w_i \leq B$ und $\sum_{i \in I} p_i \geq k$, akzeptiere.
 - 2 Lehne ab.
- Prüfung von 2^n vielen Untermengen in Schritt 1.
 - D.h. die Gesamtlaufzeit ist exponentiell in der Eingabelänge.
 - Prüfung *einzelner potentieller Lösungen* in Schritt 1.1 ist effizient.