Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Vorlesung am 11.11.2010

INSTITUT FÜR THEORETISCHE INFORMATIK

Satz von Rice - Motivation

- Wir haben bisher gezeigt, dass wir kein Programm schreiben können, das für ein Turing-Maschinen-Programm $\langle \mathcal{M} \rangle$ und eine Eingabe w entscheidet, ob $\mathcal{M}$ auf der Eingabe w hält.
- Wir werden im folgenden sehen, dass wir aus einem Programm im allgemeinen keine nicht-trivialen Eigenschaften der von dem Programm realisierten Funktion ableiten können.

Satz von Rice

Satz (Satz von Rice):
Sei R die Menge der von Turing-Maschinen berechenbaren Funktionen und S eine nicht-triviale Teilmenge von R ($\emptyset \neq S \neq R$). Dann ist die Sprache

$$L(S) := \{\langle \mathcal{M} \rangle \mid \mathcal{M} \text{ berechnet eine Funktion aus } S\}$$

nicht entscheidbar.

Satz von Rice

Satz (Satz von Rice):
Sei R die Menge der von Turing-Maschinen berechenbaren Funktionen und S eine nicht-triviale Teilmenge von R ($\emptyset \neq S \neq R$). Dann ist die Sprache

$$L(S) := \{\langle \mathcal{M} \rangle \mid \mathcal{M} \text{ berechnet eine Funktion aus } S\}$$

nicht entscheidbar.

Beweisskizze:

- Zeige: $\mathcal{H}_\varepsilon := \{\langle \mathcal{M} \rangle \mid \mathcal{M}$ hält auf der Eingabe $\varepsilon\}$ ist unentscheidbar
- Zeige: $\mathcal{H}_\varepsilon^C$ ist unentscheidbar
- Führe den Widerspruchsbeweis für die Unentscheidbarkeit von $L(S)$:
- Konstruiere TM für $\mathcal{H}_\varepsilon^C$ unter Benutzung von TM $\mathcal{M}'$ für $L(S)$

Bemerkungen zum Satz von Rice

Karlsruher Institut für Technologie

Der Satz von Rice hat weitreichende Konsequenzen:

> Es ist für Programme nicht entscheidbar, ob die durch sie definierte Sprache endlich, leer, unendlich oder ganz Σ^* ist.

Wir haben hier nur die Unentscheidbarkeit von L_d direkt bewiesen.
Die anderen Beweise folgten dem folgenden Schema:

> Um zu zeigen, dass ein Problem A unentscheidbar ist, zeigen wir, wie man mit einem Entscheidungsverfahren für A ein bekanntermaßen unentscheidbares Problem B entscheiden kann. Dies liefert den gewünschten Widerspruch.

Das Post'sche Korrespondenzproblem

Post'sches Korrespondenzproblems
Gegeben ist endliche Folge von Wortpaaren

$$K = ((x_1, y_1), \ldots, (x_n, y_n))$$

über einem endlichen Alphabet Σ. Es gilt $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$. Gefragt ist, ob es eine endliche Folge von Indizes $i_1, \ldots, i_k \in \{1, \ldots, n\}$ gibt, so dass $x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$ gilt.

Beispiele

- $K = ((1, 111), (10111, 10), (10, 0))$ hat die Lösung $(2, 1, 1, 3)$, denn es gilt:
$$x_2 x_1 x_1 x_3 = 101111110 = y_2 y_1 y_1 y_3$$
- $K = ((10, 101), (011, 11), (101, 011))$ hat keine Lösung
- $K = ((001, 0), (01, 011), (01, 101), (10, 001))$ hat eine Lösung der Länge 66.

Das Post'sche Korrespondenzproblem

Karlsruher Institut für Technologie

Satz (Unentscheidbarkeit des PKP):
Das Post'sche Korrespondezproblem ist nicht entscheidbar

Beweis:
Beweis über Nicht-Entscheidbarkeit des Halteproblems.

Eigenschaften von (semi-)entscheibaren Sprachen

- Die entscheidbaren Sprachen sind abgeschlossen unter Komplementbildung, Schnitt und Vereinigung.
- Die semi-entscheidbaren Sprachen sind abgeschlossen unter Schnitt und Vereinigung, aber nicht unter Komplementbildung.

Satz:

Sei $L \subseteq \Sigma^*$ und $L^c = \Sigma^* \setminus L$. Dann gilt

- L entscheidbar gdw L^c entscheidbar.
- L und L^c semientscheidbar gdw L entscheidbar.

Beweis: Übung und Tutorien.

Kapitel

- Sprachen
- Probleme
- Zeitkomplexität

Thema dieses Kapitels

Karlsruher Institut für Technologie

Fragestellung bisher:

- Ist eine Sprache L entscheidbar oder nicht?
- Ist eine Funktion berechenbar oder nicht?
- Benutzung von deterministischen Turing-Maschinen.

In diesem Kapitel:

- Wie effizient kann ein Problem gelöst werden?
- Betrachtung von nichtdeterministischen Turing-Maschinen.

Frage (P vs. NP):
Gibt es einen wesentlichen Effizienzgewinn beim Übergang der deterministischen Turing-Maschine zur nichtdeterministischen Turing-Maschine?

Wie sieht ein Problem aus?

Karlsruher Institut für Technologie

Beispiel: Traveling Salesman Problem (TSP)

Gegeben sei ein vollständiger Graph $G = (V, E, c)$, d.h.

- $V := \{1, \ldots, n\}$
- $E := \{\{u, v\} \mid u, v \in V, u \neq v\}$
- $c\colon E \to \mathbb{Z}^+$.

Wir betrachten folgende Problemvarianten

- **Optimierungsproblem:**
  Gesucht ist eine Tour (Rundreise), die alle Elemente aus V enthält und minimale Gesamtlänge unter allen solchen Touren hat.
- **Optimalwertproblem:**
  Gesucht ist die Länge einer minimalen Tour.
- **Entscheidungsproblem:**
  Gegeben sei zusätzlich auch ein Parameter $k \in \mathbb{Z}^+$. Die Frage ist nun: Gibt es eine Tour, deren Länge höchstens k ist?

Wie sieht ein Problem aus?

Karlsruher Institut für Technologie

Wir betrachten folgende Problemvarianten

- **Optimierungsproblem:**
  Gesucht ist eine Tour (Rundreise), die alle Elemente aus V enthält und minimale Gesamtlänge unter allen solchen Touren hat.
- **Optimalwertproblem:**
  Gesucht ist die Länge einer minimalen Tour.
- **Entscheidungsproblem:**
  Gegeben sei zusätzlich auch ein Parameter $k \in \mathbb{Z}^+$. Die Frage ist nun: Gibt es eine Tour, deren Länge höchstens k ist?

Bemerkung:

- Mit einer Lösung des Optimierungsproblems kann man leicht auch das Optimalwertproblem und das Entscheidungsproblem lösen.
- Mit einer Lösung des Optimalwertproblems kann man leicht auch das Entscheidungsproblem lösen.

Definition: Problem

Ein **Problem** Π ist gegeben durch:

- eine allgemeine Beschreibung aller vorkommenden Parameter;
- eine genaue Beschreibung der Eigenschaften, die die Lösung haben soll.

Ein **Problembeispiel** / (**Instanz**) von Π erhalten wir, indem wir die Parameter von Π festlegen.

Definition: Kodierungsschema

- Wir interessieren uns für die **Laufzeit** von Algorithmen.
- Diese wird in der Größe des Problems gemessen.

Die Größe eines Problems ist abhängig von der Beschreibung oder Kodierung der Problembeispiele

- Ein **Kodierungsschema** *s* ordnet jedem Problembeispiel eines Problems eine Zeichenkette oder *Kodierung* über einem Alphabet Σ zu.
- Die **Inputlänge** eines Problembeispiels ist die Anzahl des Symbole seiner Kodierung.

Kodierungsschema

Es gibt verschiedene Kodierungsschemata für ein bestimmtes Problem.

Beispiel:

- Zahlen können dezimal, binär, unär, usw. kodiert werden.
- Die Inputlänge von 5127 beträgt dann 4 für dezimal, 13 für binär und 5127 für unär.

Wir werden uns auf vernünftige Schemata festlegen:

- Die Kodierung eines Problembeispiels sollte keine überflüssigen Informationen enthalten.
- Zahlen sollen binär (oder k-är für $k \neq 1$) kodiert sein.

Kodierungsschema

Dies bedeutet, die Kodierungslänge

- einer ganzen Zahl n ist $\lfloor \log_k |n| + 1 \rfloor + 1 =: \langle n \rangle$
  (eine 1 benötigt man für das Vorzeichen);
- einer rationalen Zahl $r = \frac{p}{q}$ ist $\langle r \rangle = \langle p \rangle + \langle q \rangle$;
- eines Vektors $X = (x_1, \ldots, x_n)$ ist $\langle X \rangle := \sum_{i=1}^{n} \langle x_i \rangle$;
- einer Matrix $A \in \mathbb{Q}^{m \times n}$ ist $\langle A \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} \langle a_{ij} \rangle$.
- eines Graphen $G = (V, E)$ kann zum Beispiel durch die Kodierung seiner *Adjazenzmatrix*, die eines gewichteten Graphen durch die Kodierung der *Gewichtsmatrix* beschrieben werden.

Äquivalenz von Kodierungsschemata

Karlsruher Institut für Technologie

Zwei Kodierungsschemata s_1, s_2 heißen **äquivalent** bezüglich eines Problems Π, falls es Polynome p_1, p_2 gibt, so dass gilt:

$$(|s_1(I)| = n \Rightarrow |s_2(I)| \leq p_2(n)) \text{ und } (|s_2(I)| = m \Rightarrow |s_1(I)| \leq p_1(m))$$

für alle Problembeispiele I von Π.

Entscheidungsprobleme

- Ein Entscheidungsproblem Π können wir als Klasse von Problembeispielen D_Π auffassen.
- Eine Teilmenge dieser Klasse ist $J_\Pi \subseteq D_\Pi$, die Klasse der **Ja–Beispiele**, d.h. die Problembeispiele deren Antwort Ja ist.
- Der Rest der Klasse $N_\Pi \subseteq D_\Pi$ ist die Klasse der **Nein–Beispiele**.

Korrespondenz von Entscheidungsproblemen und Sprachen

Ein Problem Π und ein Kodierungsschema s: $D_\Pi \to \Sigma^*$ zerlegen Σ^* in drei Klassen:

- Wörter aus Σ^*, die *nicht* Kodierung eines Beispiels aus D_Π sind,
- Wörter aus Σ^*, die Kodierung eines Beispiels $I \in N_\Pi$ sind,
- Wörter aus Σ^*, die Kodierung eines Beispiels $I \in J_\Pi$ sind.

Die dritte Klasse ist die Sprache, die zu Π im Kodierungsschema s korrespondiert.

Die zu einem Problem Π und einem Kodierungsschema s **zugehörige Sprache** ist

$$L[\Pi, s] := \left\{ x \in \Sigma^* \;\middle|\; \begin{array}{l} \Sigma \text{ ist das Alphabet zu } s \text{ und } x \text{ ist Kodierung} \\ \text{eines Ja–Beispiels } I \text{ von } \Pi \text{ unter } s, \text{ d.h. } I \in J_\Pi \end{array} \right\}$$

- Wir betrachten im folgenden deterministische Turing-Maschinen mit zwei Endzuständen q_J, q_N, wobei q_J akzeptierender Endzustand ist.
- Dann wird die Sprache $L_{\mathcal{M}}$ akzeptiert von der Turing-Maschine $\mathcal{M}$, falls

$$L_{\mathcal{M}} = \{x \in \Sigma^* \mid \mathcal{M} \text{ akzeptiert } x\} \ .$$

- Eine deterministische Turing-Maschine $\mathcal{M}$ **löst** ein Entscheidungsproblem Π unter einem Kodierungsschema s, falls $\mathcal{M}$ bei jeder Eingabe über dem Eingabe-Alphabet in einem Endzustand endet und $L_{\mathcal{M}} = L[\Pi, s]$ ist.

Zeitkomplexität

Für eine deterministische Turing-Maschine $\mathcal{M}$, die für alle Eingaben über dem Eingabe-Alphabet Σ hält, ist die **Zeitkomplexitätsfunktion** $T_{\mathcal{M}} : \mathbb{Z}^+ \to \mathbb{Z}^+$ definiert durch

$$T_{\mathcal{M}}(n) = \max \left\{ m \;\middle|\; \begin{array}{l} \text{es gibt eine Eingabe } x \in \Sigma^* \text{ mit } |x| = n, \text{ so} \\ \text{dass die Berechnung von } \mathcal{M} \text{ bei Eingabe } x \\ m \text{ Berechnungsschritte (Übergänge) benötigt,} \\ \text{bis ein Endzustand erreicht wird} \end{array} \right\}$$

Die Klasse $\mathcal{P}$

Die Klasse $\mathcal{P}$ist die Menge aller Sprachen L (Probleme), für die eine deterministische Turing-Maschine existiert, deren Zeitkomplexitätsfunktion polynomial ist, d.h. es existiert ein Polynom p mit

$$T_{\mathcal{M}}(n) \leq p(n).$$