Theoretische Informatik II

Einheit 9

Theoretische Informatik im Rückblick

1. Berechenbarkeitsmodelle
2. Berechenbarkeitstheorie
3. Komplexitätstheorie
4. Methodik des Aufgabenlösens

Berechenbarkeitsmodelle

- **Turingmaschinen** $\boldsymbol{\tau = (S, X, \Gamma, \delta, s_0, b)}$
  – Endlicher Automat mit unendlichem Band als Gedächtnis
  – Beschreibung durch Zustandsüberführungstabellen
  – Semantik definiert über Konfigurationen
  – Berechenbarkeitsbegriff auf Worten, Zahlen, Mengen, ...
  – Viele Varianten

- **Registermaschinen** $\boldsymbol{\rho = (S, k, \delta, s_0, F)}$
  – Vereinfachte Standardarchitektur von Einprozessorsystemen
  – Einfache Registeroperationen erweitert durch Unterprogrammtechnik
  – Berechenbarkeitsbegriff auf Zahlen äquivalent zu Turing-Berechenbarkeit

- **μ-rekursive Funktionen**
  – Mathematischer Funktionenkalkül auf Zahlen
  – Anwendung von Operationen $(\circ, Pr, \mu)$ auf Grundfunktionen (s, pr_k^n, c_k^n)
  – Programmiertechniken simulierbar
  – Primitiv-rekursive Funktionen als wichtige Teilklasse
  – Äquivalent zu Register- und Turing-Berechenbarkeit

Berechenbarkeitsmodelle

- **Typ-0 Grammatiken**
  - Regeln zur Produktion des Funktionsgraphen
  - Äquivalent zu bisherigen Modellen
- **λ-Kalkül**
  - Einfaches mathematisches Modell funktionaler Programmiersprachen
  - Funktionsdefinition, -anwendung und -auswertung
  - Datenstrukturen wie Zahlen, Listen, Boole'sche Operatoren codierbar
  - Äquivalent zu μ-rekursiven Funktionen
- **Weitere Modelle ebenfalls äquivalent**
- **Church'sche These**
  - Intuitiv berechenbar $\equiv$ Turing-berechenbar
  - Unbeweisbare Arbeitshypothese

Elementare Berechenbarkeitstheorie

- **Aufzählbarkeit und Entscheidbarkeit**
  - Berechenbarkeit der (partiellen) charakteristischen Funktion einer Menge
  - Viele äquivalente Charakterisierungen
  - M entscheidbar $\Leftrightarrow$ M und $\underline{M}$ aufzählbar
  - Abschlußeigenschaften: Vereinigung, Durchschnitt, Urbild
    Entscheidbarkeit auch Komplement und Differenz

- **Universelle Maschinen**
  - Numerierung ϕ berechenbarer Funktionen (Aufzählung der Programme)
  - Universelle Funktion $u(i,j) := \phi_i(j)$ ist berechenbar
  - Programme können effektiv kombiniert werden $(\phi_{h(i,j)} = \phi_i \circ \phi_j)$
  - Rechenzeit $\Phi_i(n) = t$ ist entscheidbar

- **Beweistechniken für unlösbare Probleme**
  - Diagonalisierung: konstruiere Widerspruch aus Annahme der Lösbarkeit
  - Monotonieargumente: Funktion wächst zu stark, um berechenbar zu sein
  - Problemreduktion: Abbildung auf bekanntes unlösbares Problem
  - Satz von Rice: keine extensionale Eigenschaft berechenbarer Funktionen
    ist entscheidbar

Komplexitätstheorie

- **Komplexitätsmaße**
  - Zeit- und Platzkomplexität abhängig von Größe der Eingabe
  - Vereinfachte Komplexitätsabschätzungen genügen
  - Asymptotische Meßgröße $\mathcal{O}(f)$
  - Obergrenze für Handhabbarkeit ist polynomielles Wachstum
- **Komplexität von Algorithmen**
  - Suchverfahren - lineare und logarithmische Laufzeit
  - Sortierverfahren - quadratische Laufzeit und $\mathcal{O}(n * \log_2 n)$
- **Komplexität von Problemen**
  - Untere Schranken für Komplexität von Sortieren: $\mathcal{O}(n * \log_2 n)$
  - Nichtdeterministische Komplexität (Orakel oder Parallelverarbeitung)
  - Komplexitätsklassen: ... $LOGSPACE \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq PSPACE \subseteq$ ..

- **$\mathcal{NP}$-Vollständigkeit**
  - Polynomielle Reduzierbarkeit $\leq_p$
  - $\mathcal{NP}$-Vollständigkeit als schwierigste Klasse in $\mathcal{NP}$
  - Satz von Cook: Expliziter Vollständigkeitsbeweis für SAT
  - Vollständigkeitsbeweise via $\leq_p$: $3SAT$, $CLIQUE$, VC, KP, GC, ...
  - Klassen jenseits von $\mathcal{NP}$:
- **Grenzüberschreitung**
  - Pseudopolynomielle Algorithmen
  - Approximationsalgorithmen
  - Probabilistische Algorithmen

Methodik des Aufgabenlösens

1. **Voraussetzungen präzisieren**
   – Welche Begriffe sind zum Verständnis der Aufgabe erforderlich
   – Was ist eigentlich genau zu tun?
2. **Lösungsweg konkretisieren**
   – Welche Einzelschritte benötigt man, um das Problem zu lösen?
   – Lösungen der einzelnen Schritte knapp, aber präzise aufschreiben
3. **Argumente zu Lösung zusammenfassen**
   – Lösungen der Einzelschritte zu Gesamtergebnis zusammenführen
   – Noch einmal hinschreiben, was jetzt insgesamt gezeigt ist

Beispiel I: primitiv-rekursive Funktionen

Zeige, daß $fak:\mathbb{N}\to\mathbb{N}$ mit $fak(n) = n!$ primitiv rekursiv ist.
Stelle fak explizit durch Operatoren und Grundfunktionen dar

1. Voraussetzungen präzisieren

– $f{:}\mathbb{N}^k\to\mathbb{N}$ ist primitiv-rekursiv, wenn f aus primitiv-rekursiven Funktionen durch Komposition oder primitive Rekursion entsteht

– Die Grundfunktionen s, pr_k^n, und c_k^n sind primitiv-rekursiv

– Primitiv-rekursive Funktionen aus Vorlesung, Übungen, Probeklausur

· p, sub, mul, exp, ...

· Fallunterscheidung, Summierung, beschränkte Minimierung, ...

– Zu tun:

Drücke fak durch obige Funktionen und Operatoren aus

Beispiel I: fak ist primitiv-rekursiv

2. Lösungsweg konkretisieren

–Einzelschritte: versuche fak durch ein Operatorenschema zu beschreiben

· Einfache Komposition bekannter Funktionen reicht nicht

· Fallunterscheidung und Minimierung passen nicht

· Versuche Schema der primitiven Rekursion

– $fak{=}Pr[f,g]$ gilt, wenn $fak(\vec{x},0){=}f(\vec{x})$, $fak(\vec{x},y{+}1){=}g(\vec{x},y,fak(\vec{x},y))$

– Dabei muß $f{:}\mathbb{N}^0{\to}\mathbb{N}$ und $g{:}\mathbb{N}^2{\to}\mathbb{N}$ sein, also fällt $\vec{x}$ ganz weg.

– Eingesetzt: $fak(0) = 0! = 1 = f()$,

$$fak(y{+}1) = (y{+}1)! = (y{+}1)*y! = (y{+}1)*fak(y) = g(y,fak(y))$$

– Es folgt $f() = 1$, also $\boldsymbol{f = c_1^0}$

und $g(y,z) = (y{+}1)*z = mul(s(y),z)$, also $\boldsymbol{g = mul \circ (s \circ pr_1^2, pr_2^2)}$

3. Argumente zu Lösung zusammenfassen

– Da f und g primitiv rekursiv sind, folgt daß fak primitiv-rekursiv ist

– Operatorenschema: $fak = Pr[c_1^0, (mul \circ (s \circ pr_1^2, pr_2^2))]$

– Wir setzen ein: $mul = Pr[c_0^1, (add \circ (pr_1^3, pr_3^3))]$ und $add = Pr[pr_1^1, s \circ pr_3^3]$

$$\boldsymbol{fak = Pr[c_1^0, (Pr[c_0^1, (Pr[pr_1^1, s{\circ}pr_3^3]{\circ}(pr_1^3, pr_3^3))]{\circ}(s{\circ}pr_1^2, pr_2^2))]}$$

Beispiel I: fak ist primitiv-rekursiv

Zeige, daß $fak : \mathbb{N} \rightarrow \mathbb{N}$ mit $fak(n) = n!$ primitiv rekursiv ist.
Stelle fak explizit durch Operatoren und Grundfunktionen dar

- **Kurze, aufgeschriebene Lösung**
  - Wir beschreiben fak durch das Schema der primitiven Rekursion
  - Es ist $fak(0) = 0! = 1 = f()$,

    $$fak(y{+}1) = (y{+}1)! = (y{+}1)*y! = (y{+}1)*fak(y) = g(y, fak(y))$$
  - Es folgt $f() = 1$, also $\qquad f = c_1^0$

    und $g(y,z) = (y{+}1)*z = mul(s(y), z)$, also $\qquad g = mul \circ (s \circ pr_1^2, pr_2^2)$
  - Da f und g primitiv rekursiv sind, folgt daß fak primitiv-rekursiv ist

---

  - Operatorenschema: $fak = Pr[c_1^0, (mul \circ (s \circ pr_1^2, pr_2^2))]$
  - Nach Einsetzen

    $$fak = Pr[c_1^0, (Pr[c_0^1, (Pr[pr_1^1, s \circ pr_3^3] \circ (pr_1^3, pr_3^3))] \circ (s \circ pr_1^2, pr_2^2))]$$

In vielen Fällen greift ein anderes Schema (Komposition, Minimierung, etc.) besser

Beispiel II: Diagonalisierung

Zeige, daß $\boldsymbol{RG_\phi} = \{(i,y) \mid \exists n \in \mathbb{N}.\ \phi_i(n){=}y\}$ nicht entscheidbar ist

1. Voraussetzungen präzisieren

– M ist entscheidbar, wenn χ_M berechenbar ist

– Charakteristische Funktion $\chi_M(\vec{x}) = \begin{cases} 1 & \text{falls } \vec{x} \in M, \\ 0 & \text{sonst} \end{cases}$

– ϕ: Numerierung berechenbarer Funktionen,
  Für jede berechenbare Funktion f gibt es in j mit $f = \phi_j$

– Zeige, daß die Annahme "RG_ϕ ist entscheidbar" zum Widerspruch führt

– Mögliche Techniken: Diagonalisierung, Monotonieargumente,
  Problemreduktion, Satz von Rice

2. Lösungsweg konkretisieren

– Einzelschritte der Diagonalisierung

1. Annahme RG_ϕ ist entscheidbar
2. Konstruiere eine Diagonalfunktion f mittels χ_{RG_ϕ}
3. Zeige, daß f berechenbar ist, also $f = \phi_j$ für ein j
4. Zeige, daß f auf seinem eigenen Index j widersprüchlich ist

zu 2.: definiere $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\phi \\ i & \text{sonst} \end{cases}$ Schlüsselidee für Widerspruch auf (j,j)

zu 3.: f berechenbar, da erzeugt durch Fallunterscheidung mit Test $(i,i) \in RG_\phi$

Es gilt $(i,i) \in RG_\phi \Leftrightarrow \chi_{RG_\phi}(i,i) = 1$, also $f(i) = \mu_z[\chi_{RG_\phi}(i,i) = 0] + i$

Da f berechenbar ist, gibt es einen Index j mit $f = \phi_j$

zu 4.: Wir betrachten das Verhalten von f auf j

Es gilt $(j,j) \in RG_\phi$

$\Leftrightarrow \exists n \in \mathbb{N}.\ \phi_j(n) = j$ (nach Definition von RG_ϕ)

$\Leftrightarrow \exists n \in \mathbb{N}.\ f(n) = j$ $(f = \phi_j)$

$\Leftrightarrow (j,j) \notin RG_\phi$ (nach Konstruktion von f)

Dies ist ein Widerspruch. **Also kann RG_ϕ nicht entscheidbar sein**

Beispiel II: Diagonalisierung

Zeige, daß $\boldsymbol{RG_\phi} = \{(i,y) \mid \exists n \in \mathbb{N}.\ \phi_i(n){=}y\}$ nicht entscheidbar ist

- **Kurze, aufgeschriebene Lösung**

  Wir nehmen an RG_ϕ sei entscheidbar.

  Dann ist die charakteristische Funktion χ_{RG_ϕ} berechenbar.

  Wir konstruieren mit χ_{RG_ϕ} eine berechenbare Funktion f, die sich auf ihrer eigenen Gödelnummer widersprüchlich verhält.

  Es sei $f(i) = \begin{cases} \perp & \text{falls } (i,i) \in RG_\phi \\ i & \text{sonst} \end{cases}$

  f ist berechenbar, da $(i,i) \in RG_\phi \Leftrightarrow \chi_{RG_\phi}(i,i) = 1$.

  Damit gibt es einen Index j mit $f = \phi_j$

  Wir betrachten das Verhalten von f auf j

  Es gilt $(j,j) \in RG_\phi \Leftrightarrow \exists n \in \mathbb{N}.\ \phi_j(n){=}j \Leftrightarrow \exists n \in \mathbb{N}.\ f(n){=}j \Leftrightarrow (j,j) \notin RG_\phi$

  Dies ist ein Widerspruch. **Also kann RG_ϕ nicht entscheidbar sein**

Beispiel III: $\mathcal{NP}$-Vollständigkeit

Zeige, daß VC (das Vertex Cover Problem) $\mathcal{NP}$-vollständig ist

- **Voraussetzungen präzisieren**
  - L ist $\mathcal{NP}$-vollständig, falls $L \in \mathcal{NP}$ und $L' \leq_p L$ für jedes $L' \in \mathcal{NP}$
  - $L \in \mathcal{NP}$, falls L von einer NTM in polynomieller Zeit entschieden wird
  - $L' \leq_p L$, falls es eine polynomiell bb. Funktion f gibt mit $x \in L' \Leftrightarrow f(x) \in L$
  - $\boldsymbol{VC} = \{ (G,k) \mid \exists V' \subseteq V.\ |V'| \leq k \wedge V'$ Knotenüberdeckung von $G \}$
- **Standard-Lösungsweg**

  Zeige $\boldsymbol{VC \in \mathcal{NP}}$:

  a) Beschreibe, welchen Lösungsvorschlag das Orakel generiert

  b) Beschreibe, wie Lösungsvorschlag überprüft wird

  c) Zeige, daß das Prüfverfahren polynomiell ist

  Zeige $\boldsymbol{\exists L' \in \mathcal{NPC}.\ L' \leq_p VC}$

  d) Wähle ein ähnliches, bekanntes Problem $L' \in \mathcal{NPC}$

  e) Beschreibe Transformationsfunktion f, welche Eingaben aus der Sprache für L' in Worte der Sprache für VC umwandelt

  f) Zeige für alle x: $x \in L' \Leftrightarrow f(x) \in VC$

  g) Zeige, daß f in polynomieller Zeit berechnet werden kann

Beispiel III: $\mathcal{NP}$-Vollständigkeit von VC

Zeige $VC \in \mathcal{NP}$:

a) Rate eine Kantenmenge $V' \subseteq V$

b) Prüfe $|V'| \leq k$ *maximal $|V'|$ Schritte*

Prüfe: $\forall \{v, v'\} \in E.\ v \in V' \vee v' \in V'$ *maximal $|V'| * |E| \leq |V|^3$ Schritte*

c) Gesamte Anzahl der Schritte ist in $\mathcal{O}(|V|^3)$

Zeige $\exists L' \in \mathcal{NPC}.\ L' \leq_p VC$

d) Wähle das $\mathcal{NP}$-vollständige Cliquen Problem und zeige $CLIQUE \leq_p VC$

e) Es ist V' eine Clique in $G = (V, E)$

$\Leftrightarrow \forall v, v' \in V'.\ v \neq v' \Rightarrow \{v, v'\} \in E \qquad \Leftrightarrow \forall v, v' \in V'.\{v, v'\} \notin E^c$

$\Leftrightarrow \forall \{v, v'\} \in E^c. v \in V - V' \vee v' \in V - V$

$\Leftrightarrow V - V'$ Knotenüberdeckung von $G^c = (V, E^c)$

Setze $f(G, k) := (G^c, |V| - k)$

f) Es folgt $(G, k) \in CLIQUE$

$\Leftrightarrow G$ hat Clique V_c der Mindestgröße k

$\Leftrightarrow G^c$ hat Knotenüberdeckung $V' = V - V_c$ der Maximalgröße $|V| - k$

$\Leftrightarrow f(G, k) = (G^c, |V| - k) \in VC$

g) f ist in polynomieller Zeit $\mathcal{O}(|V|^2)$ berechenbar

Aus $VC \in \mathcal{NP}$ und $CLIQUE \leq_p VC$ folgt **VC ist $\mathcal{NP}$-Vollständig**

FRAGEN ?