Theorie der Informatik

19. P, NP und polynomielle Reduktionen

Malte Helmert Gabriele Röger

Universität Basel

12. Mai 2014

Überblick: Vorlesung

Vorlesungsteile

I. Logik ✓
II. Automatentheorie und formale Sprachen ✓
III. Berechenbarkeitstheorie ✓
IV. Komplexitätstheorie

Überblick: Komplexitätstheorie

IV. Komplexitätstheorie

18. Motivation und Einführung ✓
19. P, NP und polynomielle Reduktionen
20. Satz von Cook und Levin
21. einige NP-vollständige Probleme

Nachlesen

Literatur zu diesem Vorlesungskapitel

Theoretische Informatik - kurz gefasst
von Uwe Schöning (5. Auflage)

- Kapitel 3.1 und 3.2

19.1 P und NP

Akzeptanz eines Worts in Zeit n

Definition (Akzeptanz eines Worts in Zeit n)

Sei M eine DTM oder NTM mit Eingabealphabet Σ,
$w \in \Sigma^*$ ein Wort und $n \in \mathbb{N}_0$.

M akzeptiert w in Zeit n, falls es eine Folge von Konfigurationen $c_0, \ldots, c_k$ mit $k \leq n$ gibt, für die gilt:

- c_0 ist die Startkonfiguration für w,
- $c_0 \vdash c_1 \vdash \cdots \vdash c_k$, und
- c_k ist eine Endkonfiguration.

Akzeptanz einer Sprache in Zeit f

Definition (Akzeptanz einer Sprache in Zeit f)

Sei M eine DTM oder NTM mit Eingabealphabet Σ,
$L \subseteq \Sigma^*$ eine Sprache und $f : \mathbb{N}_0 \to \mathbb{N}_0$ eine Funktion.

M akzeptiert L in Zeit f, falls gilt:

1. für alle Wörter $w \in L$ gilt: M akzeptiert w in Zeit $f(|w|)$
2. für alle Wörter $w \notin L$: M akzeptiert w nicht

P und NP

Definition (P und NP)

P ist die Menge aller Sprachen L,
für die eine DTM M und ein Polynom p existieren,
so dass M die Sprache L in Zeit p akzeptiert.

NP ist die Menge aller Sprachen L,
für die eine NTM M und ein Polynom p existieren,
so dass M die Sprache L in Zeit p akzeptiert.

Bemerkungen:

- Mengen von Sprachen wie P und NP, die über den Rechenaufwand von TMs (oder anderen Berechnungsmodellen) definiert sind, heissen Komplexitätsklassen.
- Wir wissen, dass $\mathrm{P} \subseteq \mathrm{NP}$. (Warum?)
- Ob die Umkehrung gilt, ist eine offene Frage: das berühmte P-NP-Problem

Beispiel: DirHamiltonCycle $\in$ NP

Beispiel (DirHamiltonCycle $\in$ NP)

Der im vorigen Kapitel angegebene nichtdeterministische Algorithmus löst das Problem und kann auf einer NTM mit polynomiellem Zeitaufwand implementiert werden.

- Gilt DirHamiltonCycle $\in$ P?
- Die Antwort ist unbekannt.
- Bisher kennt man nur exponentielle deterministische Algorithmen für das Problem.

Simulation von NTMs durch DTMs

- Anders als DTMs sind NTMs kein realistisches Berechnungsmodell: man kann sie nicht direkt auf Computern implementieren.
- Allerdings kann man NTMs simulieren, indem man die verschiedenen Berechnungspfade systematisch durchprobiert, z. B. mit einer Breitensuche.

Genauer:

- Sei M eine NTM, die Sprache L in Zeit f akzeptiert.
- Dann kann man eine DTM M' angeben, die L in Zeit f' akzeptiert, wobei $f'(n) = O(f(n) \cdot c^{f(n)})$.
- Dabei ist c der Verzweigungsgrad von M, d. h. die maximale Anzahl Nachfolgekonfigurationen von Konfigurationen von M.

19.2 Polynomielle Reduktionen

Polynomielle Reduktionen: Idee

- Reduktionen sind ein verbreitetes und mächtiges Konzept in der Informatik. Wir kennen sie aus dem vorigen Vorlesungsteil.
- Die Idee ist, dass wir ein neues Problem lösen, indem wir es auf ein bekanntes Problem zurückführen (reduzieren).
- In der Komplexitätstheorie wollen wir Reduktionen für Aussagen der folgenden Art verwenden:

  Problem A kann effizient gelöst werden,
  sofern Problem B effizient gelöst werden kann.

- Dafür benötigen wir eine Reduktion von A auf B, die ihrerseits effizient berechnet werden kann (sonst wäre sie für die effiziente Lösung von A unbrauchbar).

Polynomielle Reduktionen

Definition (polynomielle Reduktion)

Seien $A \subseteq \Sigma^*$ und $B \subseteq \Sigma'^*$ Entscheidungsprobleme.
Wir sagen, dass A polynomiell auf B reduzierbar ist, geschrieben $A \leq_p B$, wenn eine Funktion $f : \Sigma^* \to \Sigma'^*$ mit folgenden Eigenschaften existiert:

- f ist in polynomieller Zeit durch eine DTM berechenbar
  - d. h., es gibt ein Polynom p und eine DTM M, so dass M auf Eingabe $w \in \Sigma^*$ in höchstens $p(|w|)$ Schritten $f(w)$ berechnet
- f reduziert A auf B
  - d. h. für alle $w \in \Sigma^*$ gilt: $w \in A$ gdw. $f(w) \in B$

f heisst polynomielle Reduktion von A auf B.

Polynomielle Reduktionen: Bemerkungen

- Polynomielle Reduktionen heissen auch Karp-Reduktionen (nach Richard Karp, der 1972 die ersten Reduktionsbeweise führte)
- Natürlich müssen wir in der Praxis keine DTM angeben, die f berechnet:
  es muss nur klar sein, dass f in polynomieller Zeit durch einen deterministischen Algorithmus berechenbar ist.

Polynomielle Reduktionen: Beispiel (1)

Definition (HamiltonCycle)

HamiltonCycle ist das folgende Entscheidungsproblem:

- Gegeben: ungerichteter Graph $G = \langle V, E \rangle$
- Frage: Enthält G einen Hamiltonkreis?

Polynomielle Reduktionen: Beispiel (2)

Definition (TSP)

TSP (Travelling Salesperson Problem) ist das folgende Entscheidungsproblem:

- Gegeben: Menge S von Städten (endlich, nicht-leer), symmetrische Kostenfunktion $cost : S \times S \to \mathbb{N}_0$, Kostenschranke $K \in \mathbb{N}_0$
- Frage: Gibt es eine Rundreise mit Gesamtkosten höchstens K, d. h. eine Permutation $\langle s_1, \ldots, s_n \rangle$ der Städte, so dass $\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K$?

Polynomielle Reduktionen: Beispiel (3)

Satz (HamiltonCycle $\leq_p$ TSP)

HamiltonCycle $\leq_p$ TSP.

Beweis.

$\rightsquigarrow$ Tafel □

Eigenschaften von polynomiellen Reduktionen (1)

Satz (Eigenschaften von polynomiellen Reduktionen)

Seien A, B und C Entscheidungsprobleme.

1. *Wenn $A \leq_p B$ und $B \in P$, dann $A \in P$.*
2. *Wenn $A \leq_p B$ und $B \in NP$, dann $A \in NP$.*
3. *Wenn $A \leq_p B$ und $A \notin P$, dann $B \notin P$.*
4. *Wenn $A \leq_p B$ und $A \notin NP$, dann $B \notin NP$.*
5. *Wenn $A \leq_p B$ und $B \leq_p C$, dann $A \leq_p C$.*

Eigenschaften von polynomiellen Reduktionen (2)

Beweis.

zu 1.:

Wir müssen zeigen, dass es eine DTM gibt, die in polynomieller Zeit A akzeptiert.

Wir wissen:

- Es gibt eine DTM M_B, die B akzeptiert, und zwar in Zeit p, wobei p ein Polynom ist.
- Es gibt eine DTM M_f, die eine Reduktion von A nach B in polynomieller Zeit q berechnet.

...

Eigenschaften von polynomiellen Reduktionen (3)

Beweis (Fortsetzung).

Betrachte die Maschine M, die sich zuerst wie M_f verhält und dann (nach Halten von M_f) wie M_B.

M akzeptiert A:

- M verhält sich auf Eingabe w so wie M_B auf Eingabe $f(w)$, akzeptiert also w genau dann, wenn $f(w) \in B$.
- Da f eine Reduktion ist, gilt $w \in A$ gdw. $f(w) \in B$.

...

Eigenschaften von polynomiellen Reduktionen (4)

Beweis (Fortsetzung).

Berechnungszeit von M auf Eingabe w:

- zunächst rechnet M_f auf Eingabe w: $\leq q(|w|)$ Schritte
- dann rechnet M_B auf Eingabe $f(w)$: $\leq p(|f(w)|)$ Schritte
- $|f(w)| \leq |w| + q(|w|)$, denn in $q(|w|)$ Schritten kann M_f maximal $q(|w|)$ zusätzliche Zeichen auf das Band schreiben

$\rightsquigarrow$ Gesamtlaufzeit $\leq q(|w|) + p(|f(w)|)$
$\leq q(|w|) + p(|w| + q(|w|))$

$\rightsquigarrow$ das ist polynomiell in $|w|$ $\rightsquigarrow$ $A \in \mathrm{P}$.

...

Eigenschaften von polynomiellen Reduktionen (5)

Beweis (Fortsetzung).

zu 2.:
analog zu 1., nur dass M_B und M NTMs sind

zu 3.+4.:
äquivalente Formulierungen zu 1.+2. (Kontraposition)

zu 5.:
Gelte $A \leq_{\mathrm{p}} B$ mit Reduktion f und $B \leq_{\mathrm{p}} C$ mit Reduktion g.
Dann ist $g \circ f$ eine Reduktion von A nach C.

Die Laufzeit der hintereinander geschalteten Berechnungen ist polynomiell mit dem gleichen Argument wie im Beweisteil 1. □

19.3 NP-Härte und NP-Vollständigkeit

NP-Härte und NP-Vollständigkeit

Definition (NP-hart, NP-vollständig)

Sei B ein Entscheidungsproblem.

B heisst NP-hart, wenn $A \leq_p B$ für alle Probleme $A \in \text{NP}$.

B heisst NP-vollständig, wenn $B \in \text{NP}$ und B NP-hart ist.

NP-vollständige Probleme: Bedeutung

- NP-harte Probleme sind „mindestens so schwer“ wie alle Probleme in NP.
- NP-vollständige Probleme sind „die schwersten“ Probleme in NP: alle Probleme in NP können auf sie zurückgeführt werden.
- Wenn $A \in \text{P}$ für irgendein NP-vollständiges Problem gilt, dann ist $\text{P} = \text{NP}$. (Warum?)
- Es gibt also entweder für alle NP-vollständigen Probleme effiziente Algorithmen, oder für keines.
- Gibt es überhaupt NP-vollständige Probleme?

19.4 Zusammenfassung

Zusammenfassung

- P: von DTMs in polynomieller Zeit akzeptierte Sprachen
- NP: von NTMs in polynomieller Zeit akzeptierte Sprachen
- polynomielle Reduktion: $A \leq_p B$, wenn in polynomieller Zeit berechenbare totale Funktion f existiert, so dass für alle Wörter w gilt: $w \in A$ gdw. $f(w) \in B$
- $A \leq_p B$ impliziert, dass A „höchstens so schwer“ ist wie B
- polynomielle Reduzierbarkeit ist transitiv
- NP-harte Probleme B: $A \leq_p B$ für alle $B \in \text{NP}$
- NP-vollständige Probleme B: $B \in \text{NP}$ und B ist NP-hart