Vorlesung

Theoretische Informatik II

Bernhard Beckert

Institut für Informatik

UNIVERSITÄT KOBLENZ · LANDAU

Wintersemester 2007/2008

Dank

Diese Vorlesungsmaterialien basieren zum Teil auf den Folien zu den Vorlesungen von

Katrin Erk (gehalten an der Universität Koblenz-Landau)

Jürgen Dix (gehalten an der TU Clausthal)

Christoph Kreitz (gehalten an der Universität Potsdam)

Ihnen gilt mein herzlicher Dank.

– Bernhard Beckert, Oktober 2007

NP

Theorem 15.9 (Charakterisierung von NP)

Eine Sprache L is in **NP** *genau dann wenn es eine Sprache L' in* **P** *und ein $k \geq 0$ gibt, so das für alle $w \in \Sigma$ gilt:*

$$w \in L \text{ gdw. es gibt ein } c : \langle w, c \rangle \in L' \text{ und } |c| < |w|^k.$$

c wird ***Zeuge*** *(witness oder Zertifikat/certificate) von w in L genannt. Eine DTM, die die Sprache L' akzeptiert, wird* ***Prüfer*** *(verifier) von L genannt.*

Wichtig:

Ein Entscheidungsproblem ist in **NP** genau dann wenn **jede Ja-Instanz ein kurzes Zertifikat** hat (d.h. seine Länge polynomial in der Länge der Eingabe ist), welche in polynomial-Zeit verifiziert werden kann.

Teil VI

1 Wiederholung: Die Struktur von PSPACE

2 **Wiederholung: Vollständige und harte Probleme**

3 Beispiele

Vollständige und harte Probleme

Definition 16.1 (NP-vollständig, NP-hart)

- Eine Sprache L heißt **NP-hart (NP-schwer)** wenn jede Sprache $L' \in \mathbf{NP}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **NP-vollständig** wenn sie
  1. in **NP** ist ($L \in \mathbf{NP}$), und
  2. **NP**-hart ist

Definition 16.2 (PSPACE-vollständig, PSPACE-hart)

- Eine Sprache L heißt **PSPACE-hart (PSPACE-schwer)** wenn jede Sprache $L' \in \mathbf{PSPACE}$ polynomial-zeit-reduzibel auf L ist.
- Eine Sprache L heißt **PSPACE-vollständig** wenn sie
  1. in **PSPACE** ist ($L \in \mathbf{PSPACE}$) und
  2. **PSPACE**-hart ist

Vollständige und harte Probleme

Bemerkenswert

- Wenn gezeigt werden kann, daß auch nur ein einziges **NP**-hartes Problem in **P** liegt, dann ist $\mathbf{P} = \mathbf{NP}$.
- Wenn $\mathbf{P} \neq \mathbf{NP}$ gilt, dann ist kein einziges **NP**-vollständiges Problem in polynomieller Zeit lösbar.

  Eine Million Euro für den, der das „P = NP“-Problem löst!
  (Millenium Probleme)

Vollständige und harte Probleme

Wie zeigt man NP-Vollständigkeit?

Um zu zeigen, dass eine Sprache L **NP**-vollständig ist:

Finde bekanntermaßen **NP**-vollständige Sprache L' und reduzieren sie auf L:

$$L' \preceq L$$

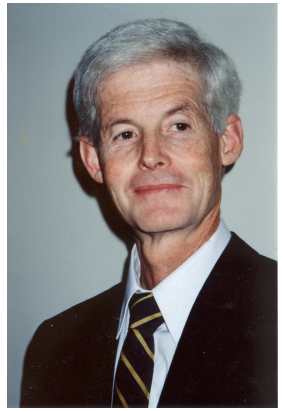
Das genügt, da jede Sprache aus **NP** auf L' reduzierbar ist und wegen $L' \preceq L$ dann auch auf L.

Hierfür häufig verwendet:
SAT-Problem, d.h.

$$L' = L_{\text{sat}} = \{w \mid w \text{ ist eine erfüllbare aussagenlogische Formel}\}$$

Satz von Cook

Stephen A. Cook ⋆1939



- Informatik-Professor and der Universität Toronto
- 1971: „Das Erfüllbarkeitsproblem der Aussagenlogik (SAT) ist NP-vollständig“
- Turing-Preisträger

Abgschlossenheit der Komplexitätsklassen

P, PSPACE abgeschlossen unter Komplement

Alle Komplexitätsklassen, die mittels **deterministischer Turing-Maschinen** definiert sind, sind **abgeschlossen unter Komplement-Bildung**

Denn:
Wenn eine Sprache L dazu gehört, dann auch ihr Komplement
(einfach die alte Maschine ausführen und die Ausgabe invertieren).

Abgschlossenheit der Komplexitätsklassen

Abgeschlossenheit von NP unter Komplement

Frage:
Ist **NP** abgeschlossen unter Komplementbildung?

Antwort:
Keiner weiß es!

Die Kompexitätsklasse co-NP

Definition 16.3 (co-NP)

co-NP ist die Klasse dr Sprachen deren Komplemente in **NP** liegen:

$$\textbf{co-NP} = \{L \mid \overline{L} \in \textbf{NP}\}$$

Beziehungen zwischen Komplexitätsklassen

Die folgenden Beziehungen sind momentan noch unbekannt

1. **P** $\subsetneq$ **NP**.
2. **NP** $\subsetneq$ **co-NP**.
3. **P** $\subsetneq$ **PSPACE**.
4. **NP** $\subsetneq$ **PSPACE**.

Teil VI

1 Wiederholung: Die Struktur von PSPACE

2 Wiederholung: Vollständige und harte Probleme

3 **Beispiele**

NP-vollständige Probleme

Beispiel 17.1 (NP vollständige Probleme)

- Ist ein (un-) gerichteter Graph hamiltonsch? (**Hamiltonian circle**)
- Ist eine logische Formel erfüllbar? (**Satisfiability**)
- Gibt es in einem Graphen eine Clique der Größe k? (**Clique of size k**)
- Ist ein Graph mit drei Farben zu färben? (**3-colorability**)
- Gibt es in einer Menge von ganzen Zahlen eine Teilmenge mit der Gesamtsumme x? (**Subset Sum**)
- Rucksackproblem (**knapsack**)
- Multiprozessor-Scheduling

NP-vollständige Probleme

Definition 17.2 (Hamilton Circle)

Hamilton-Kreis:
Weg entlang der Kanten in einem Graphen, der jeden Knoten genau einmal besucht

$L_{\text{Ham}_{\text{undir}}}$: Die Sprache, die aus allen ungerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

$L_{\text{Ham}_{\text{dir}}}$: Die Sprache, die aus allen gerichteten Graphen besteht, in denen es einen Hamilton-Kreis gibt.

NP-vollständige Probleme

Definition 17.3 (Maximale Clique: L_{Clique_k})

Eine **Clique** in einem Graphen ist ein **vollständiger Teilgraph von** G.

Für $k \in \mathbb{N}$:

L_{Clique_k} Die Sprache, die aus allen ungerichteten Graphen besteht, die eine Clique der Größe k enthalten.

Definition 17.4 (k-colorability: $L_{\text{Color}_{\leq k}}$)

Ein (ungerichteter) Graph heißt **k-färbbar**, falls jeder Knoten mit einer von k Farben so gefärbt werden kann, daß benachbarte Knoten verschiedene Farben haben.

Für $k \in \mathbb{N}$:

$L_{\text{Color}_{\leq k}}$ Die Sprache, die aus allen ungerichteten, mit höchstens k Farben färbbaren Graphen besteht.

Definition 17.5 (SAT, k-CNF, k-DNF)

DNF: Eine Formel ist in **disjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \wedge \ldots \wedge l_{1n_1}) \vee \ldots \vee (l_{m1} \wedge \ldots \wedge l_{mn_m})$$

CNF: Eine Formel ist in **konjunktiver Normalform**, wenn sie von folgender Form ist:

$$(l_{11} \vee \ldots \vee l_{1n_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mn_m})$$

Definition (Fortsetzung)

k*-DNF:** Eine Formel ist in ***k**-DNF* wenn sie in DNF ist und jede ihrer Konjuntionen genau ***k Literale hat.

k*-CNF:** Eine Formel ist in ***k**-CNF* wenn sie in CNF ist und jede ihrer Disjunktion genau ***k Literale hat.

Theorem 17.6 (NP-vollständige Probleme)

Die folgenden Probleme liegen in **NP** *und sind* **NP***-vollständig:*

- L_{sat}
- *CNF*
- *k-CNF für* $k \geq 3$

Theorem 17.7 (Probleme in P)

Die folgenden Probleme liegen in **P***:*

- *DNF*
- *k-DNF für alle k*
- *2-CNF*

Definition 17.8 (Rucksackproblem (knapsack))

Ein Rucksackproblem besteht aus

- n Objekten mit Gewichten $g_1, \ldots, g_n$ und Nutzwerten $a_1, \ldots, a_n$
- einem Gewichtsmaximum G
- einem Nutzwertminimum A

Das Rucksackproblem ist lösbar, wenn es eine Teilmenge der gegebenen Objekte gibt, deren Gewicht nicht größer als G und deren Nutzen nicht kleiner als A ist.

$L_{\mathbf{KP}}$ Die Sprache, die aus allen lösbaren Rucksackproblemen besteht.

Definition 17.9 (Multiprozessor-Scheduling-Problem)

Ein Scheduling-Problem besteht aus

- n Prozesse mit Laufzeiten $t_1, \ldots, t_n$
- m Prozessoren
- eine Maximallaufzeit (Deadline) D

Das Schedulinig-Problem ist lösbar, wenn es eine Verteilung der Prozesse auf die Prozessoren gibt, so dass alle Prozesse beendet sind bevor D vergangen ist.

$L_{\mathbf{schedule}}$ Die Sprache, die aus allen lösbaren Scheduling-Problemen besteht.

Beispiel 17.10 (CNF-SAT $\preceq_{\mathbf{pol}}$ Clique$_{\leq k}$)

Gegeben eine Instanz von CNF
(eine Konjunktion von Klauseln $C_1 \wedge C_2 \wedge \ldots \wedge C_k$)

Wir konstruieren daraus einen Graphen:

Knoten: die Paare (x, i), so dass x ein Literal ist, dass in der Klausel C_i vorkommt.

Kanten: Es gibt eine Kante zwischen (x, i) und (y, j) falls:
(1) $i \neq j$, und
(2) x, y sind nicht komplementär.

Es gilt dann:
Die CNF-Formel ist erfüllbar genau dann,
wenn der zugeordnete Graph eine Clique der Größe k hat.