Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik

Vorlesung am 7.12.1010

INSTITUT FÜR THEORETISCHE INFORMATIK

Kapitel

- **Komplementsprachen**

Die Klassen NPI, co-P und co-NP

- Die Klasse $\mathcal{NPC}$ ($\mathcal{NP}$-complete) sei die Klasse der $\mathcal{NP}$-vollständigen Sprachen/Probleme.
- Die Klasse $\mathcal{NPI}$ ($\mathcal{NP}$-intermediate) ist definiert durch $\mathcal{NPI} := \mathcal{NP} \backslash (\mathcal{P} \cup \mathcal{NPC})$.

Klasse der Komplementsprachen

- Die Klasse **co** $-\,\mathcal{P}$ ist die Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$.
- Die Klasse **co** $-\,\mathcal{NP}$ ist die Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{NP}$.

Die Klassen NPI, co-P und co-NP

Karlsruher Institut für Technologie

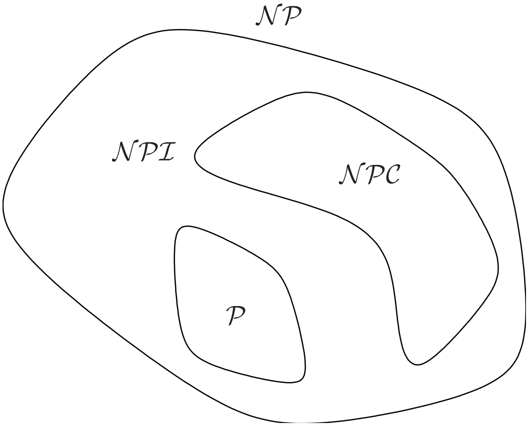
- Die Klasse $\mathcal{NPC}$ ($\mathcal{NP}$-complete) sei die Klasse der $\mathcal{NP}$-vollständigen Sprachen/Probleme.
- Die Klasse $\mathcal{NPI}$ ($\mathcal{NP}$-intermediate) ist definiert durch $\mathcal{NPI} := \mathcal{NP} \backslash (\mathcal{P} \cup \mathcal{NPC})$.

Klasse der Komplementsprachen

- Die Klasse **co** $- \mathcal{P}$ ist die Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{P}$.
- Die Klasse **co** $- \mathcal{NP}$ ist die Klasse aller Sprachen $\Sigma^* \backslash L$ für $L \subseteq \Sigma^*$ und $L \in \mathcal{NP}$.

Satz (Ladner (1975)):
Falls $\mathcal{P} \neq \mathcal{NP}$, so folgt $\mathcal{NPI} \neq \emptyset$.

Vermutete Situation



Offensichtlich: $\mathcal{P} = \text{co} - \mathcal{P}$.

Frage: Gilt auch $\mathcal{NP} = \text{co} - \mathcal{NP}$?

- Natürlich folgt aus $\mathcal{NP} \neq \text{co} - \mathcal{NP}$, dass $\mathcal{P} \neq \mathcal{NP}$ gilt.
- Aber was folgt aus $\mathcal{NP} = \text{co} - \mathcal{NP}$?
- Vermutlich ist $\mathcal{NP} \neq \text{co} - \mathcal{NP}$
  (Verschärfung der $\mathcal{P} \neq \mathcal{NP}$–Vermutung).

Das TSP-Komplement-Problem

Problem co–TSP

Gegeben: Graph $G = (V, E)$, $c\colon E \to \mathbb{Z}^+$ und ein Parameter K.

Aufgabe: Gibt es *keine* Tour der Länge $\leq K$?

- **Bemerkung:** Für ein vernünftiges Kodierungsschema von TSP ist es leicht nachzuweisen, ob ein gegebener String eine gültige TSP-Instanz repräsentiert.
- co–TSP in co $- \mathcal{NP}$, denn TSP in $\mathcal{NP}$.
- Frage: Ist co–TSP in $\mathcal{NP}$?
- Vermutung: Nein.

Lemma

Satz (Lemma):
Falls L $\mathcal{NP}$-vollständig ist und $L \in \text{co} - \mathcal{NP}$, so ist $\mathcal{NP} = \text{co} - \mathcal{NP}$.

Lemma

Satz (Lemma):
Falls L $\mathcal{NP}$-vollständig ist und $L \in \text{co} - \mathcal{NP}$, so ist $\mathcal{NP} = \text{co} - \mathcal{NP}$.

Beweis:

- Sei $L \in \text{co} - \mathcal{NP}$.
- Dann existiert eine polynomiale nichtdet. Berechnung für L^c.
- Für alle $L' \in \mathcal{NP}$ gilt: $L' \propto L$
- Also existiert eine det. poly. Transformation ${L'}^c \propto L^c$.
- Deshalb existiert eine poly. nichtdet. Berechnung für ${L'}^c$
- Also $L' \in \text{co} - \mathcal{NP}$.

Bemerkung

- Mit der Vermutung $\mathcal{NP} \neq \text{co} - \mathcal{NP}$ folgt auch $\mathcal{NPC} \cap \text{co} - \mathcal{NP} = \emptyset$.
- Wenn ein Problem in $\mathcal{NP}$ und $\text{co} - \mathcal{NP}$ ist, vermutlich aber nicht in $\mathcal{P}$, so ist es in $\mathcal{NPI}$.

Das Problem Subgraphisomorphie

Problem Subgraphisomorphie

Gegeben: Graphen $G = (V, E)$ und $H = (V', E')$ mit $|V'| < |V|$

Frage: Gibt es eine Menge $U \subseteq V$ mit $|U| = |V'|$ und eine bijektive Abbildung $\mathrm{Iso}\colon V' \to U$, so dass für alle $x, y \in V'$ gilt: $\{x, y\} \in E' \iff \{\mathrm{Iso}(x), \mathrm{Iso}(y)\} \in E$

Frage anschaulich: Ist H isomorph zu einem Subgraphen von G?

Das Problem Subgraphisomorphie

Problem Subgraphisomorphie

Gegeben: Graphen $G = (V, E)$ und $H = (V', E')$ mit $|V'| < |V|$

Frage: Gibt es eine Menge $U \subseteq V$ mit $|U| = |V'|$ und eine bijektive Abbildung $\mathrm{Iso}\colon V' \to U$, so dass für alle $x, y \in V'$ gilt:
$\{x, y\} \in E' \Longleftrightarrow \{\mathrm{Iso}(x), \mathrm{Iso}(y)\} \in E$

Problem Subgraphisomorphie ist $\mathcal{NP}$-vollständig (ohne Beweis).

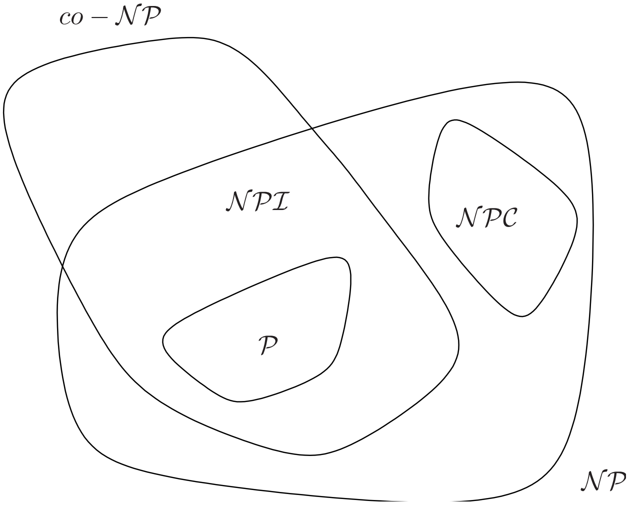
Das Problem Graphisomorphie

Problem Graphisomorphie

Gegeben: Graphen $G = (V, E)$ und $H = (V', E')$ mit $|V| = |V'|$.

Frage: Existiert eine bijektive Abbildung $\mathrm{Iso}\colon V' \to V$ mit $\{x, y\} \in E' \Longleftrightarrow \{\mathrm{Iso}(x), \mathrm{Iso}(y)\} \in E$?

- Frage anschaulich: Sind G und H isomorph?
- Graphisomorphie ist ein Kandidat für ein Problem aus $\mathcal{NPI}$
- Graphisomorphie liegt in $\mathcal{NP}$ und co $- \mathcal{NP}$.


$co - \mathcal{NP}$
$\mathcal{NPI}$
$\mathcal{NPC}$
$\mathcal{P}$
$\mathcal{NP}$

Kapitel

- **Weitere Komplexitätsklassen über NP hinaus**

Suchprobleme

Ein **Suchproblem** Π wird beschrieben durch

- die Menge der Problembeispiele / Instanzen D_Π und
- für $I \in D_\Pi$ die Menge $S_\Pi(I)$ *aller* Lösungen von I.

Die **Lösung** eines Suchproblems für eine Instanz D_Π ist

- ein beliebiges Element aus $S_\Pi(I)$ falls $S_\Pi(I) \neq \emptyset$
- $\emptyset$ sonst

Beispiel: TSP-Suchproblem

TSP-Suchproblem (Variante 1)

Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit Gewichtsfunktion $c \colon E \to \mathbb{Q}$.

Aufgabe: Gib eine optimale Tour zu G bezüglich c an.

- Bemerkung: $S_\Pi(G)$ ist die Menge aller optimalen Touren zu G.

TSP-Suchproblem (Variante 2)

Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit Gewichtsfunktion $c \colon E \to \mathbb{Q}$, Parameter $k \in \mathbb{Q}$.

Aufgabe: Gib eine Tour zu G bezüglich c mit Maximallänge k an, falls eine existiert.

Beispiel: Hamilton–Kreis Suchproblem

Gegeben ist ein Graph $G = (V, E)$.

Ein Hamilton–Kreis in G ist eine Permutation π auf V, so dass

$$\{\pi(n), \pi(1)\} \in E \text{ und } \{\pi(i), \pi(i+1)\} \in E \text{ für } 1 \leq i \leq n-1 \text{ ist.}$$

Hamilton–Kreis Suchproblem

Gegeben: Ein ungerichteter, ungewichteter Graph $G = (V, E)$.

Aufgabe: Gib einen Hamilton–Kreis in G an, falls einer existiert.

- Bemerkung: $S_\Pi(G)$ ist die Menge aller Hamilton-Kreise in G.

Aufzählungsprobleme

Ein **Aufzählungsproblem** Π ist gegeben durch

- die Menge der Problembeispiele D_Π und
- für $I \in D_\Pi$ die Menge $S_\Pi(I)$ aller Lösungen von I.

Die **Lösung** der Instanz I eines Aufzählungsproblem Π besteht in der Angabe der Kardinalität von $S_\Pi(I)$, d.h. von $|S_\Pi(I)|$.

Beispiel: Hamilton–Kreis Aufzählungsproblem

Hamilton–Kreis Aufzählungsproblem

Gegeben: Ein ungerichteter, ungewichteter Graph $G = (V, E)$.

Aufgabe: Wieviele Hamilton–Kreise gibt es in G?

Reduzierbarkeit für Suchprobleme

Zu einem Suchproblem Π sei R_Π folgende Relation:

$$R_\Pi := \{(x, s) \mid x \in D_\Pi, s \in S_\Pi(x)\}$$

Eine Funktion $f\colon \Sigma^* \to \Sigma^*$ **realisiert** eine Relation R, wenn für alle $x \in \Sigma^*$ gilt:

$$f(x) = \begin{cases} \varepsilon & \nexists y \in \Sigma^* \setminus \{\varepsilon\} \ : \ (x, y) \in R \\ y & \text{sonst, mit beliebigem } y : (x, y) \in R \end{cases}$$

Ein Algorithmus **löst** das durch R_Π beschriebene Suchproblem Π, wenn er eine Funktion berechnet, die R_Π realisiert.

Orakel-Turing-Maschine

Eine **Orakel-Turing-Maschine** zum Orakel $G\colon \Sigma^* \to \Sigma^*$ ist eine deterministische Turing-Maschine mit

- einem ausgezeichnetem **Orakelband**
- zwei zusätzlichen Zuständen q_f und q_a.

Dabei ist

- q_f der **Fragezustand**
- q_a der **Antwortzustand**

des Orakels.

- Die Arbeitsweise ist in allen Zuständen $q \neq q_f$ wie bei der normalen Turing-Maschine.

Orakel-TM: Verhalten im Fragezustand

Wenn der

- Zustand q_f angenommen wird,
- Kopf sich auf Position i des Orakelbandes befindet
- Inhalt des Orakelbandes auf Position $1, \ldots, i$ das Wort $y = y_1 \ldots y_i$ ist,

dann verhält sich die Orakel-TM wie folgt:

- falls $y \notin \Sigma^*$: Fehlermeldung und die Orakel-TM hält.
- In einem Schritt wird y auf dem Orakelband gelöscht
- $G(y)$ wird auf Positionen $1, \ldots, |G(y)|$ des Orakelbandes geschrieben
- Der Kopf des Orakelbandes springt auf Position 1
- Folgezustand ist q_a.

Bemerkung

- Orakel-TM und Nichtdeterministische TM sind verschiedene Konzepte.

Turing-Reduktion

Turing-Reduktion
Seien R, R' Relationen über Σ^*. Eine **Turing-Reduktion** $\propto_T$ von R auf R' ($R \propto_T R'$), ist eine Orakel-Turing-Maschine $\mathcal{M}$,

- deren Orakel die Relation R' realisiert
- die selbst in polynomialer Zeit die Funktion f berechnet, die R realisiert.

Bemerkung:

- Falls R' in polynomialer Zeit realisierbar ist und $R \propto_T R'$, so ist auch R in polynomialer Zeit realisierbar.
- Falls $R \propto_T R'$ und $R' \propto_T R''$ so auch $R \propto_T R''$.

NP-schwer

Ein Suchproblem Π heißt $\mathcal{NP}$–schwer, falls es eine $\mathcal{NP}$-vollständige Sprache L gibt mit $L \propto_T \Pi$.

Bemerkung

- Ein Problem das $\mathcal{NP}$–schwer ist, muss nicht notwendigerweise in $\mathcal{NP}$ sein.

Das TSP-Suchproblem ist NP-schwer

TSP-Suchproblem (Variante 1)

Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit Gewichtsfunktion $c\colon E \to \mathbb{Q}$.

Aufgabe: Gib eine optimale Tour zu G bezüglich c an.

TSP-Entscheidungsproblem

Gegeben: Graph $G = (V, E)$ vollständig und gewichtet mit Gewichtsfunktion $c\colon E \to \mathbb{Q}$, Parameter $k \in \mathbb{Q}$.

Aufgabe: Gibt es eine Tour der Länge höchstens k?

Satz:
Das TSP-Suchproblem ist NP-schwer.

Beweisskizze

- Bezeichne TSP_E das Entscheidungsproblem.
- Bezeichne TSP_S das Suchproblem.

Die zu TSP_E bzw, TSP_S gehörenden Relationen R_E und R_S sind gegeben durch

$$\begin{aligned} R_E &:= \{(x, J) \mid x \in J_{TSP_E}\} \\ R_S &:= \{(x, y) \mid x \in D_{TSP_S}, y \in S_{TSP_S}(x)\} \ . \end{aligned}$$

Beweisskizze

$$\begin{aligned} R_E &:= \{(x, J) \mid x \in J_{TSP_E}\} \\ R_S &:= \{(x, y) \mid x \in D_{TSP_S}, y \in S_{TSP_S}(x)\} . \end{aligned}$$

Wir zeigen $R_E \propto_T R_S$:

Dazu geben wir eine OTM (Orakel-Turing-Maschine) mit Orakel $\Omega : \Sigma^* \to \Sigma^*$ an. Ω realisiert R_S.

Die OTM arbeitet wie folgt für eine Eingabe w:

- Schreibe die Eingabe auf das Orakelband und gehe in Zustand q_f.
- Weise das Orakel an, in einem Schritt $\Omega(w)$ auf das Orakelband zu schreiben und anschließend in den Zustand q_a zu wechseln.
- Prüfe, ob $\Omega(w)$ eine Tour der Länge $\leq k$ kodiert. Falls ja, lösche das Band und schreibe J, andernfalls lösche das Band.

Die gegebene OTM realisiert R_E und hat polynomial beschränkte Laufzeit.

Verallgemeinerte NP-Schwere

- Wir nennen ein Problem $\mathcal{NP}$–schwer, wenn es mindestens so schwer ist, wie alle $\mathcal{NP}$–vollständigen Probleme.

Darunter fallen auch

- Optimierungsprobleme, für die das zugehörige Entscheidungsproblem $\mathcal{NP}$–vollständig ist.
- Entscheidungsprobleme Π, für die gilt, dass für alle Probleme $\Pi' \in \mathcal{NP}$ gilt $\Pi' \propto \Pi$, aber für die nicht klar ist, ob $\Pi \in \mathcal{NP}$.

Klar ist, dass ein $\mathcal{NP}$–vollständiges Problem auch $\mathcal{NP}$–schwer ist.

Das Problem INTEGER PROGRAMMING

Problem INTEGER PROGRAMMING

Gegeben: $a_{ij} \in \mathbb{N}_0$, $b_i, c_j \in \mathbb{N}_0$, $1 \leq i \leq m$, $1 \leq j \leq n$, $B \in \mathbb{N}_0$.

Frage: Existieren $x_1, \ldots, x_n \in \mathbb{N}_0$, so dass

$$\sum_{j=1}^{n} c_j \cdot x_j = B \text{ und}$$

$$\underbrace{\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i}_{A \cdot \bar{x} \leq \bar{b}} \text{ für } 1 \leq i \leq m?$$

Das Problem INTEGER PROGRAMMING

Problem INTEGER PROGRAMMING

Gegeben: $a_{ij} \in \mathbb{N}_0$, $b_i, c_j \in \mathbb{N}_0$, $1 \leq i \leq m$, $1 \leq j \leq n$, $B \in \mathbb{N}_0$.

Frage: Existieren $x_1, \ldots, x_n \in \mathbb{N}_0$, so dass

$$\sum_{j=1}^{n} c_j \cdot x_j = B \text{ und}$$

$$\underbrace{\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i}_{A \cdot \bar{x} \leq \bar{b}} \text{ für } 1 \leq i \leq m?$$

Problem INTEGER PROGRAMMING ist $\mathcal{NP}$-schwer.

Beweis

$$\exists\, x_1,\ldots,x_n \in \mathbb{N}_0, \text{ dass } \sum_{j=1}^{n} c_j \cdot x_j = B \text{ und } \underbrace{\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i}_{A\cdot\bar{x}\leq\bar{b}} \text{ für } 1 \leq i \leq m?$$

Beweis:
Zeigen: SUBSET SUM $\propto$ INTEGER PROGRAMMING.
Zu M, $w : M \to \mathbb{N}_0$ und $K \in \mathbb{N}_0$ Beispiel für SUBSET SUM wähle $m = n := |M|$, o.B.d.A. $M = \{1,\ldots,n\}$, $c_j := w(j)$, $B := K$, $b_i = 1$ und $A = (a_{ij})$ Einheitsmatrix. Dann gilt:

$$\exists M' \subseteq M \text{ mit } \sum_{j\in M'} w(j) = K$$

$$\Updownarrow$$

$$\exists x_1,\ldots,x_n \in \mathbb{N}_0 \text{ mit } \sum_{j\in M} w(j)\cdot x_j = B \text{ und } x_j \leq 1 \text{ für } 1 \leq j \leq n.$$

$M' = \{j \in M : x_j = 1\}$

Bemerkungen

- INTEGER PROGRAMMING $\in \mathcal{NP}$ ist nicht so leicht zu zeigen. Siehe: Papadimitriou „On the complexity of integer programming“, J.ACM, 28, 2, pp. 765-769, 1981.
- Wie der vorherige Beweis zeigt, ist INTEGER PROGRAMMING sogar schon $\mathcal{NP}$–schwer, falls $a_{ij}, b_i \in \{0, 1\}$ und $x_i \in \{0, 1\}$.
- Es kann sogar unter der Zusatzbedingung $c_{ij} \in \{0, 1\}$ $\mathcal{NP}$–Vollständigkeit gezeigt werden (ZERO-ONE PROGRAMMING).
- Für beliebige lineare Programme ($a_{ij}, c_j, b_i \in \mathbb{Q};\ x_i \in \mathbb{R}$) existieren polynomiale Algorithmen.