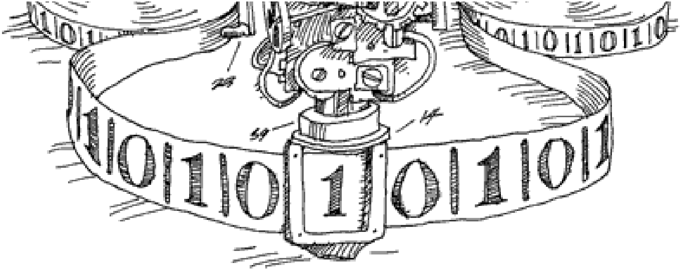
Theoretische Grundlagen der Informatik

Tutorium 8

Institut für Theoretische Informatik



Komplementsprachen

Definition

Zu einer Sprache $L \subseteq \Sigma^*$ definieren wir co-L als das Komplement der Sprache, also co-$L := \Sigma^* \setminus L$.

Für eine Klasse von Sprachen wie $\mathcal{P}$ definieren wir die „co"-Klasse (wie z.B. co-$\mathcal{P}$) als Menge der Komplemente (und nicht etwa als Komplement der Klasse).

Beispiel: co-3SAT: Die aussagenlogischen Formeln mit 3 Variablen pro Klausel, die nicht erfüllbar sind.

Anmerkungen

- Ob $\mathcal{NP} =$ co-$\mathcal{NP}$ ist eine offene Frage.
- $\mathcal{NP} \neq$ co-$\mathcal{NP}$ impliziert $\mathcal{P} \neq \mathcal{NP}$
- Für $L \in \mathcal{NPC}$ gilt: $L \in$ co-$\mathcal{NP} \iff \mathcal{NP} =$ co-$\mathcal{NP}$

Komplementsprachen

Definition

Zu einer Sprache $L \subseteq \Sigma^*$ definieren wir co-L als das Komplement der Sprache, also co-$L := \Sigma^* \setminus L$.

Für eine Klasse von Sprachen wie $\mathcal{P}$ definieren wir die „co"-Klasse (wie z.B. co-$\mathcal{P}$) als Menge der Komplemente (und nicht etwa als Komplement der Klasse).

Beispiel: co-3SAT: Die aussagenlogischen Formeln mit 3 Variablen pro Klausel, die nicht erfüllbar sind.

Anmerkungen

- Ob $\mathcal{NP} =$ co-$\mathcal{NP}$ ist eine offene Frage.
- $\mathcal{NP} \neq$ co-$\mathcal{NP}$ impliziert $\mathcal{P} \neq \mathcal{NP}$
- Für $L \in \mathcal{NPC}$ gilt: $L \in$ co-$\mathcal{NP} \iff \mathcal{NP} =$ co-$\mathcal{NP}$

NPI

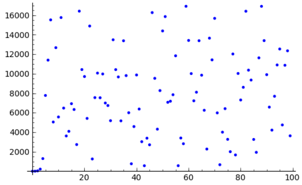
Definition

$\mathcal{NP}$-Intermediate $(\mathcal{NPI}) := \mathcal{NP} \setminus (\mathcal{NPC} \cup \mathcal{P})$

Da $\mathcal{NPI} \neq \emptyset \iff \mathcal{P} \neq \mathcal{NP}$ (*Ladner's Theorem, 1975*), sind natürlich keine Probleme aus $\mathcal{NPI}$ bekannt, es gibt jedoch einige Kandidaten:

- Graphisomorphie
- Faktorisieren
- Diskrete Logarithmen

Aufgrund ihrer Eigenschaften sind diese Probleme in der Kryptographie alle von großer Bedeutung.



Aufgabe

Karlsruher Institut für Technologie

Betrachte das Problem NEAR TAUT: Gegeben sei ein Boolescher Ausdruck A. Es ist zu entscheiden, ob es höchstens eine Belegung der Variablen gibt, so dass A falsch wird.

1. Formuliere das komplementäre Problem co-NEAR TAUT.
2. Zeige, dass NEAR TAUT in co-$\mathcal{NP}$ liegt.

Pseudopolynomielle Algorithmen

Definition

Ein Algorithmus wird **pseudopolyomiell** genannt, wenn seine Laufzeit polynomiell in der Eingabe *bei Unärkodierung* ist.

$\Rightarrow$ Die Laufzeit hängt polynomiell von der Eingabelänge und der größten vorkommenden Zahl in der Eingabe ab.

Beispiel KNAPSACK: Entscheiden einer Instanz mit n Objekten und maximalen Kosten W in $\mathcal{O}(nW)$ über dynamische Programmierung

Ein Problem heißt **schwach $\mathcal{NP}$-vollständig,** wenn es $\mathcal{NP}$-vollständig ist und ein pseudopolynomieller Algorithmus existiert, der das Problem entscheidet.

Existiert ein solcher Algorithmus nicht, so spricht man von einem **stark $\mathcal{NP}$-vollständigen** Problem.

Pseudopolynomielle Algorithmen

Definition

Ein Algorithmus wird **pseudopolyomiell** genannt, wenn seine Laufzeit polynomiell in der Eingabe *bei Unärkodierung* ist.

$\Rightarrow$ Die Laufzeit hängt polynomiell von der Eingabelänge und der größten vorkommenden Zahl in der Eingabe ab.

Beispiel KNAPSACK: Entscheiden einer Instanz mit n Objekten und maximalen Kosten W in $\mathcal{O}(nW)$ über dynamische Programmierung

Ein Problem heißt **schwach $\mathcal{NP}$-vollständig,** wenn es $\mathcal{NP}$-vollständig ist und ein pseudopolynomieller Algorithmus existiert, der das Problem entscheidet.

Existiert ein solcher Algorithmus nicht, so spricht man von einem **stark $\mathcal{NP}$-vollständigen** Problem.

Aufgabe

Das Entscheidungsproblem *PRIMES* besteht darin, zu entscheiden, ob es sich bei einer gegebenen natürlichen Zahl $p > 1$ um eine Primzahl handelt. Eine Probleminstanz von *PRIMES* wird also durch eine natürliche Zahl kodiert.

Ein naiver Algorithmus für *PRIMES* könnte alle Zahlen $2, 3, \ldots, p-1$ daraufhin überprüfen, ob sie die gegebene Zahl p teilen.

Zeige, dass dieser Algorithmus pseudopolynomiell ist. (Gib dazu eine Schranke für die Laufzeit an, die polynomiell in der Länge der Eingabe und der größten vorkommenden Zahl ist).

Suchprobleme

Definition

Ein Suchproblem Π wird beschrieben durch

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

Lösung

Die Lösung eines beliebigen Suchproblems für eine Instanz D_Π ist

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

Suchprobleme als Relationen

Ein Suchproblem kann man auch als Relation auffassen, für Π sei

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

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

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

Eine Turingmaschine löst das durch R_Π beschriebene Suchproblem Π, wenn sie eine Funktion berechnet, die R_Π realisiert.

Orakelturingmaschinen

Karlsruher Institut für Technologie

Definition

Eine Orakel-Turingmaschine $\mathcal{M}^G$ ist eine **deterministische** Turingmaschine, die ein zusätzliches Hilfsorakel enthält.

Dieses Hilfsorakel berechnet (zuverlässig!) in $\mathcal{O}(1)$ eine beliebige Funktion $G : \Sigma^* \to \Sigma^*$.

Zu jedem Zeitpunkt der Berechnung kann $\mathcal{M}$ Folgendes machen:

1. Schreibe ein Wort w auf ein spezielles „Orakelband“.
2. Gehe über in einen Fragezustand q_f
3. $\to$ Das Orakel schreibt in $\mathcal{O}(1)$ den Funktionswert $G(w)$ auf das Orakelband und $\mathcal{M}$ geht in den Antwortzustand q_a über.
4. Fahre mit der normalen Berechnung fort.

Aufgabe: Was hat dieses Hilfsorakel mit dem Orakel einer NTM zu tun?

Orakelturingmaschinen

Definition

Eine Orakel-Turingmaschine $\mathcal{M}^G$ ist eine **deterministische** Turingmaschine, die ein zusätzliches Hilfsorakel enthält.

Dieses Hilfsorakel berechnet (zuverlässig!) in $\mathcal{O}(1)$ eine beliebige Funktion $G : \Sigma^* \rightarrow \Sigma^*$.

Zu jedem Zeitpunkt der Berechnung kann $\mathcal{M}$ Folgendes machen:

1. Schreibe ein Wort w auf ein spezielles „Orakelband".
2. Gehe über in einen Fragezustand q_f
3. $\rightarrow$ Das Orakel schreibt in $\mathcal{O}(1)$ den Funktionswert $G(w)$ auf das Orakelband und $\mathcal{M}$ geht in den Antwortzustand q_a über.
4. Fahre mit der normalen Berechnung fort.

Aufgabe: Was hat dieses Hilfsorakel mit dem Orakel einer NTM zu tun?

Orakelturingmaschinen

Definition

Eine Orakel-Turingmaschine $\mathcal{M}^G$ ist eine **deterministische** Turingmaschine, die ein zusätzliches Hilfsorakel enthält.

Dieses Hilfsorakel berechnet (zuverlässig!) in $\mathcal{O}(1)$ eine beliebige Funktion $G : \Sigma^* \to \Sigma^*$.

Zu jedem Zeitpunkt der Berechnung kann $\mathcal{M}$ Folgendes machen:

1. Schreibe ein Wort w auf ein spezielles „Orakelband".
2. Gehe über in einen Fragezustand q_f
3. $\to$ Das Orakel schreibt in $\mathcal{O}(1)$ den Funktionswert $G(w)$ auf das Orakelband und $\mathcal{M}$ geht in den Antwortzustand q_a über.
4. Fahre mit der normalen Berechnung fort.

Aufgabe: Was hat dieses Hilfsorakel mit dem Orakel einer NTM zu tun? Gar nichts!

Beispiele

Sei $\mathcal{P}^L$ die Klasse aller Entscheidungsprobleme, die in polynomieller Zeit von einer deterministischen Orakel-Turingmaschine mit Orakel für die charakteristische Funktion der Sprache L entschieden werden können. Dann ist

- $SAT \in \mathcal{P}^{SAT}$
- $TSP \in \mathcal{P}^{SAT}$
- $H_0 \in \mathcal{P}^{H_0}$

Turingreduzierbarkeit

Definition

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

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

$\mathcal{NP}$-Schwere von Suchproblemen

Karlsruher Institut für Technologie

Definition

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

CLIQUE als Suchproblem

Aufgabe

Formuliere CLIQUE als Suchproblem.

CLIQUE als Suchproblem

Karlsruher Institut für Technologie

Aufgabe

Formuliere CLIQUE als Suchproblem.

CLIQUE-Suchproblem (Variante 2)

- **Gegeben:** Graph $G = (V, E)$, Parameter $k \in \mathbb{N}$
- **Aufgabe:** Gib eine Clique in G mit Kardinalität k an, falls diese existiert.

Aufzählungsprobleme

Definition

Ein *Aufzählungsproblem* Π ist gegeben durch

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

Lösung

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

CLIQUE als Aufzählungsproblem

Aufgabe

Formuliere CLIQUE als Aufzählungsproblem.

CLIQUE als Aufzählungsproblem

Karlsruher Institut für Technologie

Aufgabe

Formuliere CLIQUE als Aufzählungsproblem.

CLIQUE-Aufzählungsproblem

Eine Möglichkeit:

- **Gegeben:** Graph $G = (V, E)$, Parameter $k \in \mathbb{N}$
- **Gesucht:** Anzahl der Cliquen in G mit mindestens k Knoten.

Aufgabe

1. Zeige (informell): Aus $L \propto L'$ folgt, dass auch $L \propto_T L'$ gilt.
2. Warum gilt nicht auch direkt die Umkehrung? Überlege dir dazu, dass gilt: $TSP \propto_T$ co-TSP, aber $TSP \propto$ co-TSP ist schwer zu zeigen.

INTEGER PROGRAMMING

Definition (aus der Vorlesung)

Gegeben:

- $A = ((a_{ij})) \in \mathbb{Z}^{m \times n}$
- $b = (b_i) \in \mathbb{Z}^m, c = (c_j) \in \mathbb{Z}^n$
- $B \in \mathbb{Z}$

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

$$\underbrace{\sum_{j=1}^{n} c_k \cdot x_j = B}_{\langle c,x \rangle = B}$$

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

Eigenschaften von INTEGER PROGRAMMING

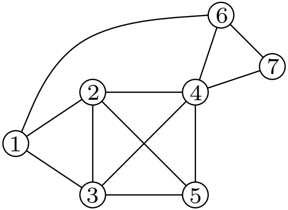
- INTEGER PROGRAMMING ist $\mathcal{NP}$-vollständig.
- Viele andere Probleme lassen sich leicht als INTEGER PROGRAMMING-Problem formulieren.

Aufgabe

Sei $G = (V, E)$ ein ungerichter Graph und $K \leq |V|$ eine natürliche Zahl. Ein *Dominating Set* von G ist eine Teilmenge $C \subseteq V$, so dass jeder Knoten v entweder selbst in C ist oder zu einem Knoten w in C adjazent ist.

Gibt es ein Dominating Set von G, das höchstens k Knoten enthält?

1. Finde in folgendem Graphen ein möglichst kleines Dominating Set:



Aufgabe

Sei $G = (V, E)$ ein ungerichter Graph und $K \leq |V|$ eine natürliche Zahl. Ein *Dominating Set* von G ist eine Teilmenge $C \subseteq V$, so dass jeder Knoten v entweder selbst in C ist oder zu einem Knoten w in C adjazent ist.

Gibt es ein Dominating Set von G, das höchstens k Knoten enthält?

2. Formuliere DOMINATING SET als *Integer Program*.

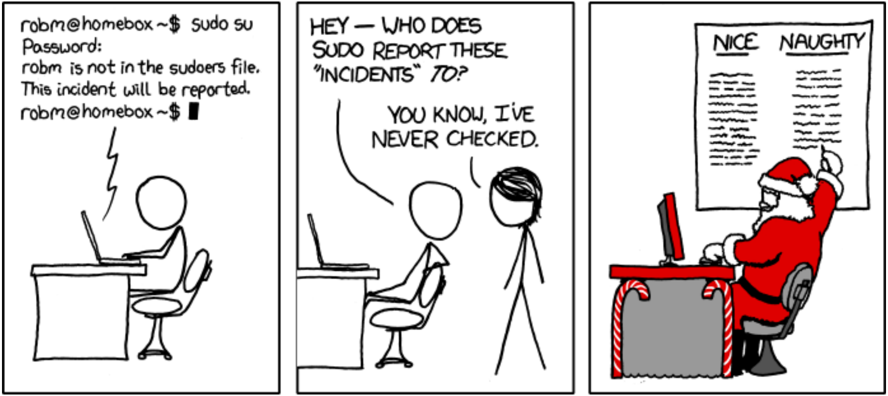
Lösungsskizze

- Bedeutung der Variablen: $x_i = \begin{cases} 1 & \text{falls } v_i \in \text{Dominating Set} \\ 0 & \text{sonst} \end{cases}$
- Modellierung der Bedingung $x_i \in \{0, 1\}$ durch $E \cdot x \leq \overrightarrow{1}$
- Für jeden Knoten muss gelten: Mindestens er selbst oder ein Nachbar ist im Dominating Set enthalten:
  - Sei $A = ((a_{ij}))$ die Adjazenzmatrix des Graphen mit Einsen auf der Diagonalen.
  - Dann muss für alle i gelten:

$$\sum_{j=1}^{|V|} x_j \cdot a_{ij} \geq 1 \text{ bzw. } \sum_{j=1}^{|V|} x_j \cdot (-a_{ij}) \leq -1$$

- Dominating Set soll die Größe K haben $\Rightarrow \sum_{i=1}^{|V|} x_i = K$
- $\Rightarrow$ Integer Program: Gibt es einen Vektor $x = (x_1, \ldots, x_{|V|})$, sodass gilt $\binom{-A}{E} \cdot x \leq \begin{pmatrix} -\overrightarrow{1} \\ \overrightarrow{1} \end{pmatrix}$ und $\sum_{i=1}^{|V|} x_i = K$?

Bis zum nächsten Mal!



He sees you when you're sleeping, he knows when you're awake, he's copied on `/var/spool/mail/root`, so be good for goodness' sake.

Lizenzen