Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Übungsblatt 5

Vorlesung Theoretische Grundlagen der Informatik im WS 16/17

Ausgabe 9. Dezember 2016
Abgabe 20. Dezember 2016, 11:00 Uhr (im Kasten im UG von Gebäude 50.34)

Bitte nutzen Sie den WebInScribe Deckblattgenerator
und heften Sie das Deckblatt an Ihr Übungsblatt.
`https://webinscribe.ira.uka.de/deckblatt/index.php?course=10588`.

Aufgabe 1

(2 + 3 = 5 Punkte)

Der ebenso geniale, wie auch frustrierte Wissenschaftler und Superbösewicht Doktor Meta ist in Geldnot. Es ist zum aus der Haut fahren! Sven van Hagen befindet sich in seiner Gewalt, Turing-Man ist in einer Endlosschleife gefangen und die schöne Elsa unterstützt Metas Pläne. Doch nach dem Bau seines neuen Hauptquartiers ist der Doktor finanziell in Schwierigkeiten. Schon seit langem hofft Doktor Meta deswegen auf die Unterstützung der Allianz für Diktatoren, die unter Superbösewichten sehr bekannt ist. Dort wird eine Liste von Bösewichten geführt, die einer (sehr größzügigen) Spende würdig sind. Offensichtlich steht Metas Name nicht darauf. Um sich in die Liste eintragen zu lassen muss man dieses Jahr allerdings *nur* ein dort noch nicht bekanntes co-$\mathcal{NP}$-Vollständiges Problem einschicken. Das ist seine Chance. Elsa erinnert sich, dass das Problem $\overline{\text{SAT}}$ co-$\mathcal{NP}$-vollständig ist.
Das Problem $\overline{\text{SAT}}$ ist folgendermaßen definiert:

> **Gegeben:** Menge U von Variablen, Menge C von Klauseln.
>
> **Frage:** Existiert *keine* Wahrheitsbelegung von U, so dass C erfüllt wird? Das heißt, nimmt für jede Wahrheitsbelegung von U mindestens eine Klausel aus C den Wahrheitswert `falsch` an?

Da Doktor Meta aufgrund der Bekanntheit von SAT befürchtet, dass $\overline{\text{SAT}}$ bereits bekannt ist, denkt er sich ein neues Problem aus. Die Sprache L_{reg} ist über einem endlichen Alphabet Σ folgendermaßen definiert:

$$L_{reg} = \{(R, S) \mid R, S \text{ sind einfache reguläre Ausdrücke mit } L(R) = L(S)\}$$

In einfachen regulären Ausdrücken wird der Kleensche Abschluss nicht verwendet.

Helfen Sie Dr. Meta und beweisen Sie, dass L_{reg} co-$\mathcal{NP}$-vollständig ist, indem Sie wie folgt vorgehen:

(a) Zeigen Sie $L_{reg} \in$ co-$\mathcal{NP}$. Sie dürfen davon ausgehen, dass ein Algorithmus existiert, der zu einem einfachen regulären Ausdruck R der Länge m und einem Wort w der Länge n in $\mathcal{O}(mn)$ Zeit entscheidet, ob $w \in L(R)$.

(b) Zeigen Sie, dass L_{reg} co-$\mathcal{NP}$-schwer ist, indem Sie $\overline{\text{SAT}} \propto L_{reg}$ zeigen.

Lösung:

(a) Seien R, S einfache reguläre Ausdrücke mit $(R, S) \notin L_{reg}$. Dann existiert ein Wort w, so dass gilt $w \in L(R)$ und $w \notin L(S)$, oder $w \notin L(R)$ und $w \in L(S)$. Also ist w ein Zeuge dafür, dass R und S nicht dieselben Sprachen beschreiben. Zu einem gegebenen Wort kann mit dem in der Aufgabenstellung genannten Algorithmus mit polynomialem Zeitaufwand entschieden werden, ob es ein Zeuge dafür ist, dass $(R, S) \notin L_{reg}$. Somit ist $L_{reg} \in$ co-$\mathcal{NP}$.

(b) Definiere $\varphi : C \times U \to \{\{0\}, \{1\}, \{0, 1\}\}$ durch $\varphi : (c, u) \mapsto \{0\}$ falls u ein positives Literal in c ist, $\varphi : (c, u) \mapsto \{1\}$, falls $\neg u$ ein negiertes Literal in c ist, und schließlich $\varphi : (c, u) \mapsto \{0, 1\}$, falls u kein Literal in c ist. Sei $U = \{u_1, u_2, \ldots, u_n\}$ und $C = \{c_1, c_2, \ldots, c_m\}$. Konstruiere dann die einfachen regulären Ausdrücke

$$R = \bigcup_{i=1}^{m} (\varphi(c_i, u_1) \cdot \varphi(c_i, u_2) \cdot \ldots \cdot \varphi(c_i, u_n)) \qquad \text{und} \qquad S = \bigcup_{j=1}^{mn} \{0, 1\}\,.$$

Sowohl R als auch S können durch eine deterministische Turingmaschine mit polynomialem Zeitaufwand berechnet werden.

Weiter gilt $L(R) = L(S)$ genau dann, wenn C durch keine Wahrheitsbelegung von U erfüllt wird. Die Sprache $L(R)$ enthält nämlich genau die kodierten Wahrheitsbelegungen von U, die C nicht erfüllen. Um eine einzelne Klausel zu erfüllen reicht es aus, dass ein einziges enthaltenes Literal den Wahrheitswert `wahr` erhält. Dies wird dadurch verhindert, dass φ alle vorkommenden Literale mit dem Wahrheitswert `falsch` belegt. Alle anderen Variablen spielen für die Wahrheitsbelegung der Klausel keine Rolle und bekommen also keinen Wahrheitswert zugewiesen.

Aufgabe 2

(3 + 1 = 4 Punkte)

Zeigen Sie, dass aus $\mathcal{P} = \mathcal{NP}$ folgt, dass alle Sprachen in $\mathcal{P}$, bis auf $\emptyset$ und $\Sigma^\star$, $\mathcal{NP}$-vollständig sind. Wieso gilt die Aussage nicht für $\emptyset$ und $\Sigma^\star$?

Lösung:
Sei $L \in \mathcal{P}$. Da $\mathcal{P} = \mathcal{NP}$ folgt direkt $L \in \mathcal{NP}$. Es bleibt zu zeigen, dass zu jeder Sprache $L' \in \mathcal{NP}$ eine polynomiale Transformation f existiert, so dass $x \in L' \iff f(x) \in L$. Da $L' \in \mathcal{NP}$ und deshalb auch $L' \in \mathcal{P}$, existiert eine deterministische Turingmaschine $\mathcal{M}$ mit polynomialer Laufzeit, die L' entscheidet. Wähle $w^+ \in L$ und $w^- \notin L$, was genau dann möglich ist, wenn $\emptyset \subsetneq L \subsetneq \Sigma^\star$. Die Tranformation f simuliert zur einer Eingabe x dann einfach $\mathcal{M}$. Akzeptiert $\mathcal{M}$, setze $f(x) = w^+$. Lehnt $\mathcal{M}$ die Eingabe x ab, setze $f(x) = w^-$. Da $\mathcal{M}$ in polynomialer Zeit läuft existiert ebenfalls eine deterministische Turingmaschine $\mathcal{M}'$, die in polynomialer Zeit die Funktion f berechnet. Somit ist f eine geeignete polynomiale Transformation.

Die Aussage gilt nicht für $\emptyset$ und $\Sigma^\star$, da dann „Ja“-Instanzen nicht mehr von „Nein“-Instanzen unterschieden werden können.

Aufgabe 3

(3 + 3 = 6 Punkte)

$$w \in L_{J\Pi} \iff \exists x \in \Sigma^\star : \mathcal{M} \text{ akzeptiert } \langle w, x \rangle,$$

wobei die Laufzeit von $\mathcal{M}$ polynomial in $n = |w|$ ist.

Hinweis: dies ist die „Zeugendefinition“ der Klasse $\mathcal{NP}$, da x die Zugehörigkeit von w zu L bezeugt.

(a) Beweisen Sie, dass jede Sprache $L \in \mathcal{NP}$ von einer deterministischen Turingmaschine in $\mathcal{O}(2^{p(n)})$ entschieden werden kann, wobei $p(n)$ ein Polynom über n ist.

(b) Nehmen Sie nun an, dass sie für eine $\mathcal{NP}$-vollständige Sprache $\mathcal{L}$ die Turingmaschine $\mathcal{M}$ so modifizieren können, dass $\mathcal{M}$ für jedes $w \in L$ höchstens 42 Zeugen x ablehnt. Zeigen Sie, dass dann $\mathcal{P} = \mathcal{NP}$ gilt.

Lösung:

(a) Aufgrund der polynomialen Laufzeit von $\mathcal{M}$ gilt die Aussage auch für $x \in \Sigma^{p(n)}$ mit einem Polynom $p(n)$. Dann kann eine deterministische Turingmaschine $\mathcal{M}'$ bei Eingabe von w für jeden möglichen Zeugen $x \in \Sigma^{p(n)}$ ausprobieren, ob $\mathcal{M}$ die Eingabe $\langle w, x \rangle$ akzeptiert. Ist dies der Fall, akzeptiert $\mathcal{M}'$ die Eingabe w. Werden alle Zeugen abgelehnt lehnt auch $\mathcal{M}'$ die Eingabe w ab. Dies entspricht genau der obigen Definition von $\mathcal{N}P$. Insgesamt gibt es $2^{p(n)}$ mögliche Zeugen. Die Gesamtlaufzeit beträgt dann $\mathcal{O}(2^{p(n)} \cdot \text{poly}(n)) = \mathcal{O}(2^{p(n)})$.

(b) Konstruiere eine deterministische Turingmaschine $\mathcal{M}'$, die zunächst 43 paarweise verschiedene Zeugen $x_i \in \Sigma^\star$ wählt. Für $i = 1, 2, \ldots, 43$ simuliert $\mathcal{M}'$ dann die Turingmaschine $\mathcal{M}$ auf der Eingabe $\langle w, x_i \rangle$. Akzeptiert $\mathcal{M}$, so gilt $w \in L$. Lehnt $\mathcal{M}$ alle 43 Zeugen ab, so gilt $w \notin L$, denn $\mathcal{M}$ lehnt im Fall $w \in L$ höchstens 42 Zeugen ab. Da $\mathcal{M}$ polynomiale Laufzeit hat, hat auch $\mathcal{M}'$ polynomiale Laufzeit. Mit der $\mathcal{NP}$-Volltändigkeit von L folgt dann $\mathcal{P} = \mathcal{NP}$.

Aufgabe 4

(2 + 2 + 3 + 1 + 1 = 9 Punkte)

Das Problem 3-Sum ist wie folgt definiert.

> **Gegeben:** Eine endliche Menge $S \subset \mathbb{R}$.
>
> **Frage:** Existieren drei nicht notwendigerweise unterschiedliche Zahlen $a, b, c \in S$, so dass $a + b + c = 0$?

Es ist offen, ob ein Algorithmus existiert der das Problem 3-Sum in der Laufzeit $O(n^{2-\epsilon}), \epsilon > 0$ löst. Ein Problem Π heißt 3-Sum-*schwer*, wenn ein subquadratischer Algorithmus für Π impliziert, dass 3-Sum in subquadratischer Zeit lösbar ist.

Das Problem 3-Linienschnitt ist wie folgt definiert:

Gegeben: Eine endliche Menge $\mathcal{L}$ von Linien in der euklidischen Ebene.

Frage: Existieren drei Linien aus $\mathcal{L}$ die sich in genau einem Punkt schneiden?

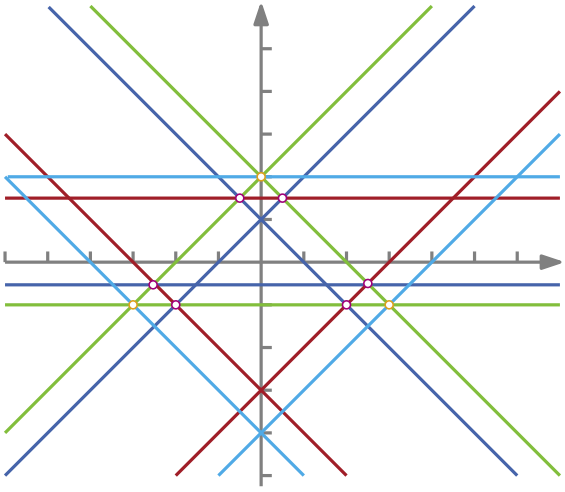
Im Folgenden sollen Sie zeigen, dass 3-Linienschnitt 3-Sum-schwer ist. Gehen Sie dafür davon aus, das Sie eine Linie $y = mx + b$ als Paar (m, b) kodieren. Sie wissen außerdem, dass die Steigung m nur die Werte $\{-1, 0, 1\}$ annehmen kann.

(a) Geben Sie für jede Zahl $x \in S$ eine Menge von Linien $\mathcal{L}_x$ an, so dass für drei Zahlen $a+b+c = 0$ genau dann gilt, wenn drei Linien $l_a \in L_a$, $l_b \in L_b$ und $l_c \in L_c$ existieren, die sich in genau einem Punkt schneiden.

Lösung:
$\mathcal{L}_x = \{(-1, x), (0, -x/2), (1, x)\}$.

(b) Konstruieren Sie zu der Instanz $S = \{1, 2, -3, -4\}$ die Mengen $L_a, a \in S$ graphisch. Kennzeichnen Sie die Mengen L_a eindeutig. Geben Sie zwei Lösungen der Form $\{(m_1, b_1), (m_2, b_2), (m_3, b_3)\}$ für die 3-Linienschnitt Instanz an. Entspricht Ihre Lösung der 3-Linienschnitt Instanz einer Lösung der 3-Sum Instanz?

Lösung:



$L_1 \cap L_2 \cap L_{-3} = \circ$ 1+2-3 = 0
$L_2 \cap L_2 \cap L_{-4} = \circ$ 2+2-4 = 0

(c) Zeigen Sie die Korrektheit Ihrer Transformation aus (a).

Lösung:
Drei Linien schneiden sich genau dann in genau einem Punt, wenn sie unterschiedliche Steigungen haben. Ein Lösungstripel ist also immer der Form $\{(-1, x), (0, y), (1, z)\}$.

Wir wählen für drei Zahlen $a, b, c \in S$ aus den Mengen $L_x, x = a, b, c$ je eine Linie mit noch nicht verwendeter Steigung. Wir zeigen, dass die Geraden $(-1, a), (0, -b/2, 1, c)$ sich genau

dann in einem Punkt schneiden, wenn $a + b + c = 0$.

$$\begin{aligned} y &= -1x + a \\ \wedge y &= -b/2 \\ \wedge y &= 1x + c \\ &\iff \\ 2y &= a + c \\ \wedge y &= -b/2 \\ &\iff \\ -b &= a + c \\ &\iff \\ 0 &= a + b + c \end{aligned}$$

(d) Benötigt Ihre Transformation subquadratische Zeit?

Lösung:
Die Transformation benötigt tatsächlich nur lineare Zeit, da für jede Zahl $a \in S$ nur drei Linien erzeugt werden.

(e) Der sogenannte Sweep-Line Algorithmus zum Schnitt von Linien benötigt $O(n \log n + k)$ Laufzeit, wobei $k \in \mathbb{N}$ die Anzahl der Schnittpunkte ist. Warum impliziert dies noch keinen subquadratischen Algorithmus für das 3-Sum Problem? Welche Voraussetzungen muss die Transformation von S nach $\mathcal{L}$ erfüllen, damit Sie mit Hilfe des Sweep-Line Algorithmus einen subquadratischen Algorithmus für 3-Sum erhalten?

Lösung:
Die in (a) angegebene Transformation erzeugt eine quadratische Anzahl an Schnittpunkten. Um den Sweep-Line Algorithmus verwenden zu können, müssen Sie sicher stellen, dass die Ihre Transformation nur eine *geringe* Anzahl an Schnittpunkten erzeugt.

Aufgabe 5 (3 + 3 + 3 = 9 Punkte)

Beweisen Sie die $\mathcal{NP}$-vollständigkeit der folgenden Probleme.

- **Gegeben:** Zwei Graphen $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ und eine Zahl $K \in \mathbb{N}$.

  Frage: Gibt es zwei Teilmengen $E_1' \subset E_1, E_2' \subset E_2$, mit $|E_1'| = |E_2'| \geq K$, so dass die Graphen $G_1' = (V(E_1'), E_1'), G_2' = (V(E_2'), E_2')$ isomorph sind?

  Nutzen Sie, dass das Problem Hamiltonischer Kreis $\mathcal{NP}$-vollständig ist.

  Lösung:
  Largest Common Subraph $\in \mathcal{NP}$:

  Sei (G_1', G_2', ϕ) mit ϕ einer Bijektion von V_1' nach V_2' eine Zeuge für die Largest Common Subgraph Instanz. In $O(|E||V|)$ Zeit kann überprüft werden

  - $|E_1'| = |E_2'| < K$
  - $V_i' \subset V_i$, $E_i' \subset E_i$
  - $\{u, v\} \in E_1'$ genau dann, wenn $\{\phi(u), \phi(v)\} \in E_2'$

Polynomiale Transformation:

Sei $G = (V, E)$ eine Instanz für das HAMILTONISCHER KREIS Problem. Wir wählen $G_1 = G$, G_2 sei ein Zyklus der Länge $|V|$ und $K = |V|$. Die Transformation ist offenbar in polynomieller Zeit berechenbar.

Korrekheit:

Sei Π (eine Permutation der Knotenmenge) ein HAMILTONISCHER KREIS in G. Sei $\phi : V_1 \to V_2$ eine beliebige Bijektion. Dann ist $(V, \phi \circ \Pi)$ eine Zeuge dafür das G_2 ein Subgraph von G_1. Desweiteren ist $K \geq |V|$ erfüllt.

Sei G' der größte gemeinsame Subgraph von G_1 und G_2. Da $|E'| \geq K = |V|$ gefordert ist, folgt das $G = G_2$. G ist also ein HAMILTONISCHER KREIS.

- **Gegeben:** Ein gerichteter Graph $G = (V, E)$, eine positive Zahl $K \in \mathbb{N}$.

  Frage: Gibt es eine Teilmenge $V' \subset V$, mit $|V'| \leq K$, so dass für jeden gerichteten Zyklus C in G, $V(C) \cap V' \neq \emptyset$ gilt.

  Nutzen Sie das Problem VERTEX COVER für die Reduktion.

  Hinweis: $V(C)$ bezeichnet die Knotenmenge des Zyklus C.

  Lösung:
  FEEDBACK VERTEX SET $\in \mathcal{NP}$:

  Es kann wie folgt in polynomial Zeit überprüft werden, ob eine Menge V' eine FEEDBACK VERTEX SET ist. Es müssen folgende Bedingungen geprüft werden.

  - $V' \subset V$
  - $|V'| \leq K$
  - Markiere mit Hilfe einer Tiefensuche jeden Zyklus der von $v \in V'$ erreichbar ist. Überprüfe für jeden nicht markierten Knoten, ob ein nicht markierter Zyklus existiert.

  Polynomiale Transformation der Transformation:

  $G' = (V, E')$ mit $E' = \{(u, v) \mid \{u, v\} \in E\} \cup \{(v, u) \mid \{u, v\} \in E\}$. Wir verwenden für beide Probleme dasselbe K. Die Transformation ist offensichtlich in polynomial Zeit, da für jede Kante $e \in E$ zwei Kanten e_1, e_2 zu E' hinzugefügt werden und sonst keine weiteren Veränderungen an dem Graphen vorgenommen werden.

  Korrektheit:

  Im Folgenden bezeichnen wir mit C_e den Kreis der beiden Kanten $e, e^{-1} \in E'$ der durch die Transformation von der Kante $e \in E$ entsteht.

  Wer zeigen zu erst, das ein VERTEX COVER V' der Größe K für G ein FEEDBACK VERTEX SET für G' ist. Sei $V' \subset V$ ein VERTEX COVER von G. Es existiert somit zu jeder Kante in $e \in E$ ein inzidenter Knoten $v \in V'$. Da der Knoten v überdeckt, überdeckt er auch den Zyklus C_e. Somit werden alle Zyklen $C_e, e \in E$ von V' überdeckt. Sei C ein Zyklus der Länge mindestens drei. Dann existiert zu jeder Kante $f \in E(C)$ eine inverse Kante $f^{-1} \in E'$. Die Kanten f und f^{-1} bilden einen Zyklus der schon von einem Knoten $v' \in V'$ überdeckt wird und somit überdeckt v' auch C.

  Sei $V' \subset V$ ein FEEDBACK ARC SET in G'. Wir zeigen das V' ein VERTEX COVER für G ist. Die Menge V' überdeckt jeden gerichteten Zyklus in G' und somit insbesondere jeden Zyklus C_e, damit wird auch jede Kante $e \in E$ von einem Knoten $v' \in V$ überdeckt.

- **Gegeben:** Eine Menge S und eine Menge $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ von Teilmengen A_i von S.

  Frage: Gibt es eine Partition $S_0 \dot{\cup} S_1 = S$, so dass kein A_i vollständig in S_0 oder S_1 enthalten ist?

Nutzen Sie das Problem 3-SAT zur Reduktion.

Lösung:
Wir zeigen SET-SPLITTING $\in \mathcal{NP}$.

Sei (S_0, S_1) ein Zeuge zu einer SET-SPLITTING Instanz $(S, \mathcal{A})$. Der Zeuge kann in $O(\Sigma_{i=1}^{n}|A_i||S|)$ Zeit wie folgt überprüft werden. Es können in polynomial Zeit Validität des Zeugens überprüft werden.

- $S_i \subset S$
- $S_0 \cup S_1 =$
- $S_0 \cup S_1 = S$
- Für jedes A_i kann in $O(|A_i||S_j|)$ Zeit überprüft werden, ob alle Element in S_j Zeit enthalten sind.

Polynomiale Transformation:

Im Folgenden bezeichnen wir mit $\overline{U} = \{\overline{x} : x \in U\}$. Wähle $S = \{f\} \cup U \cup \overline{U}$. Die Menge $\mathcal{A}$ konstruieren wir wie folgt. Für jede Variable $x \in U$, setze $A_x = \{x, \overline{x}\}$. Für eine Klausel $C_i = (x \vee y \vee z)$ konstruiere $A_i = \{x, y, z, f\}$ wobei, $x, y, z \in \overline{U} \cup U$. Die Transformation ist offensichtlich polynomial.

Korrektheit der Transformation:

Zu einer erfüllten 3-SAT Instanz konstruieren wir eine gültige SET-SPLITTING Instanz wie folgt: $S_{\text{FALSE}} = \{x \in U \mid x = \text{FALSE}\} \cup \{\overline{x} \in \overline{U} \mid x = \text{TRUE}\} \cup \{f\}$, $S_{\text{TRUE}} = \{x \in U \mid x = \text{TRUE}\} \cup \{\overline{x} \in \overline{U} \mid x = \text{FALSE}\}$. S_{FALSE} enthält somit die Literale die zu FALSE auswerten und S_{TRUE} enthält die Literale die zu TRUE auswerten. Somit ist die Menge A_x in keiner der beiden Menge vollständig enthalten. Da in einer Klausel C_i mindestens ein Literal zu TRUE auswertet, ist dieses Literal in S_{TRUE} enthalten. Da $f \in S_{\text{FALSE}}$, ist keine Menge A_i vollständig in einer der beiden Mengen enthalten.

Sei $S_0 \dot{\cup} S_1 = S$ ein Zeuge einer Ja-Instanz $(\mathcal{A}, S)$ der SPLITTING-SET Problems. Sei ohne Beschränkung der Allgemeinheit $f \in S_0$. Wir weisen jedem Literal $x \in S_1$ den Wert TRUE ($x = \text{TRUE}$, falls $x \in U$, $\overline{x} = \text{FALSE}$, falls $x \in \overline{U}$). In jeder Menge A_i existiert ein x, mit $x \in S_1$ somit wertet jede Klausel zu wahr aus. Die Menge A_x stellt sicher, dass x und $\overline{x}$ eine konsistente Wahrheitsbelegung haben.