Universität Paderborn
Prof. Dr. Johannes Blömer

Gruppe 27:
Eli Kogan-Wang (7251030)
David Noah Stamm (7249709)
Bogdan Rerich (7248483)
Jan Schreiber(7253698)

Berechenbarkeit und Komplexität – WS 2022/2023

Heimübung 7
Abgabe: 5. Dezember 2022 – 13:00 Uhr
(Dieser Übungszettel besteht aus 0 Aufgaben mit insgesamt 0 Punkten)

Aufgabe 1 (Sprachen in NP) (6 Punkte)
Betrachten Sie die folgenden Sprachen

$$\text{2-SCHEDULING} = \left\{ \langle t_1, \dots, t_n, d \rangle \;\middle|\; \begin{array}{c} \text{Es existiert eine Indexmenge } I \subseteq \{1, \dots, n\}, \\ \text{sodass } \sum_{i \in I} t_i \leq d \text{ und } \sum_{i \notin I} t_i \leq d. \end{array} \right\}$$

$$\text{VERTEXCOVER} = \left\{ \langle G, k \rangle \;\middle|\; \begin{array}{c} G = (V, E) \text{ ist ein ungerichteter Graph,} \\ \text{sodass } C \subseteq V \text{ existiert mit } |C| \leq k \text{ und} \\ \text{für alle } \{u, v\} \in E \text{ ist } u \in C \text{ oder } v \in C. \end{array} \right\}.$$

Zeigen Sie

1. 2-SCHEDULING $\in$ NP.
2. VERTEXCOVER $\in$ NP.

Lösung: Allgemein kann gezeigt werden, dass eine Sprache in NP liegt, indem man einen polynomiellen Verifizierer angebe. Sei im folgenden V ein Verifizierer.

Zu 1. Sei V eine DTM welche sich bei Eingabe x $\in \{0,1\}^*$ sich wie folgt verhält.

1. Überprüfe, ob Eingabe der Form $\langle t_1, \dots, t_n, d, S \rangle$ mit $t_1$bis t_n und d $\in \mathbb{R}$ und S $\subseteq \{1, ..., n\}$.
2. Teste, ob $\sum_{i \in S} t_i \leq d$ und $\sum_{i \notin S} t_i \leq d$. Falls dies der Fall ist so akzeptiere, sonst lehne ab.

Korrektheit:
$\langle$t$_1, ..., t_n, d \,\rangle \in$ 2-SCHEDULING $\implies$ Es existiert eine Indexmenge S sodass $\sum_{i \in S} t_i \leq d$ und $\sum_{i \notin S} t_i \leq d$ und V akzepiert V $\langle t_1, \dots, t_n, d, S \rangle$

$\langle$t$_1, ..., t_n, d \;\rangle \notin$ 2-SCHEDULING $\implies$ Es existiert keine Indexmenge für die das Geforderte gilt und V lehnt $\langle t_1, \dots, t_n, d, S \rangle$ für alle S ab.

Polynomialität von S:

Laufzeit von V:
Schritt 1: Der Formcheck lässt sich in polynomieller Zeit durchführen, da die Eingabe

gröSSen endlich sind.
Schritt 2: Besteht aus dem Vergleich zweier Summen mit d. Dies lässt sich auch in linearer Zeit umsetzten, also auch explizit in polynomieller Zeit von $|\langle t_1, ..., t_n, d\rangle|$ durchführen.

Zu 2. Sei V eine DTM, die sich bei Eingabe x $\in \{0,1\}^*$ wie folgt verhält.

1. Überprüfe ob x von von der Form $\langle G, k, C\rangle$, wobei G ein Graph, k $\in \mathbb{R}$ und C eine Menge sein soll.
2. Teste für jedes u $\in C$ ob u $\in V$.
3. Teste ob $|C| \leq k$
4. Teste für alle $\{u, v\} \in E$ ob $u \in C \lor v \in C$

Korrektheit:
w $\in$ VertexCover $\implies$ Es existiert ein C $\subseteq V, |C| \leq k$ und für alle $\{u, v\} \in E$ gilt $\{u, v\} \in E$ ob $u \in C \lor v \in C$. Dieses C erfüllt dann alle 4 Schritte.
w $\notin$ VertexCover $\implies$ Wir können kein C $\subseteq V, |C| \leq k$ finden. Spätestens bei Schritt 3 lehnt V dann also $\langle G, k, C\rangle$ ab.

Polynomialität von C:
Es gilt C $\subseteq V$ für G=(V,E). Also $|C| \leq |V|$ dadurch auch $|C| \leq |(V, E)|$ und schlieSSlich $|C| \leq |\langle G, k\rangle|^n$ für ein $n \in \mathbb{N}$

Laufzeit von V:
Schritt 1: Der Formcheck lässt sich in polynomieller Zeit durchführen.
Schritt 2: Lässt sich in $\mathcal{O}(|V|)$, also in polynomieller Zeit von w durchführen.
Schritt 3: Lässt sich in polynomieller Zeit durchführen.
Schritt 4: Im worst-case gibt es für jeden Knoten in V eine Kante zu jedem anderen Knoten in V. Es gäbe also $|V| \cdot |V-1| \cdot \frac{1}{2}$ Kanten zu überprüfen. Das liegt polynomiell unter $|(V, E)|$

Aufgabe 2 (Klasse NP) (6 Punkte)
Sei $L \in$ NP und $L^* = \{\epsilon\} \cup \bigcup_{k \in \mathbb{N}} L^k$. Die formale Definition von L^k, der Sprache aller Verkettungen von k Worten aus L, finden Sie auf Präsenzübung 3 Aufgabe 2. Zeigen Sie, dass $L^* \in$ NP.

Lösung:
Nach Präsenzübung 3 Aufgabe 2 entscheidet die folgende Turingmaschine $\langle M_{L^k}\rangle$ die Sprache L^k:
$\langle M_{L^k}\rangle$ bei Eingabe w $\in \{0,1\}$

1. Für alle Aufteilungen $v_1, ..., v_k \in L$ von w mache

a) Für i = 1...k wiederhole Simulation von $\langle M_L \rangle$ mit Eingabe v_i

b) Falls $\langle M_L \rangle$ in jedem Durchgang akzeptiert hat, akzeptiere

2. Verwerfe.

Da L $\in$ NP, ist L polynomiell verifizierbar. Sei V ein polynomieller Verifizierer für L.

Zu zeigen: L^k ist polyomiell verifizierbar, bzw. die Existenz eines polynomiellen Verifizierers V_k für L^k.
V_k bei Eingabe $\langle w, c \rangle$:

1. Teste, ob $\langle M_L \rangle$ für jede Aufteilung $v_1, v_2, ..., v_k$ von w, alle v_i als Eingabe akzeptiert. Falls ja, akzeptiere. Sonst, lehne ab.

2. Teste, ob $|\langle c \rangle| \leq |\langle w \rangle|$. Falls ja, akzeptiere. Sonst, lehne ab.

Laufzeit von V_k:
$L \in NP$, daher läuft $\langle M_L \rangle$ in polynomieller Zeit. $\langle M_L \rangle$ wird $n * k$ Mal durchgeführt (wobei n die Anzahl Aufteilungen von w in k v_i ist).
Der zweite Test, kann in linearer Zeit durchgeführt werden.
Insgesamt läuft V_k also in polynomieller Zeit.

$L^* = \{\epsilon\} \cup \bigcup_{k \in \mathbb{N}} L^k$ ist eine Vereinigung von polynomiell verifizierbarer Sprachen.
Nach Präsenzübung 7 Aufgabe 3, gilt: L_1 und $L_2 \in NP$, so gilt: $L_1 \cup L_2 \in NP$.
Folglich gilt $L^* \in NP$.

Aufgabe 3 (NP und coNP) (12 Punkte)
Betrachten Sie folgendende Definitionen, ähnlich zu den Definitionen 3.7 und 3.8 aus der Vorlesung.

Definition 3.1 *Sei L eine Sprache.*

- *Eine DTM F heiSSt* Falsifizierer *für L, falls*
$$\bar{L} = \{w \mid \exists c : F \text{ akzeptiert } \langle w, c\rangle\}.$$
- *Ein Falsifizierer F für L heiSSt* polynomieller Falsifizierer *für L, wenn es ein $k \in \mathbb{N}$ gibt, so dass*
$$\bar{L} = \{w \mid \exists c, |c| \leq |w|^k : F \text{ akzeptiert } \langle w, c\rangle\}.$$
Weiter muss die Laufzeit von F bei jeder Eingabe $\langle w, c\rangle$ polynomiell in der Länge $|w|$ von w sein.
- *Existiert ein polynomieller Falsifizierer für eine Sprache L, so heiSSt L* polynomiell falsifizierbar*.*

Definition 3.2 coNP *ist die Klasse aller Sprachen, die polynomiell falsifizierbar sind.*

Beachten Sie, dass sich die Anforderungen an einen Falsifizierer immer auf das Komplement $\bar{L}$ der Sprache L beziehen. Betrachten Sie auSSerdem folgende Sprache

$$\textsc{NonIso} = \{\langle G, H\rangle \mid G, H \text{ sind zwei ungerichtete, nicht isomorphe Graphen}\} \; .$$

Die Definition von *isomorph* finden Sie auf Präsenzübung 7 Aufgabe 2.
Zeigen Sie

1. $\textsc{NonIso} \in$ coNP.
2. $L \in$ coNP $\Leftrightarrow \bar{L} \in$ NP.
3. P $\subseteq$ NP $\cap$ coNP.
4. NP $\neq$ coNP $\Rightarrow$ P $\neq$ NP.

Lösung:

1. $\textsc{NonIso} \in$ coNP.

   Beweis:

   Wir zeigen die Existenz eines polynomiellen Falsifizierers F für $\textsc{NonIso}$.

   Wir beschreiben das verhalten von F für eine beliebige Eingabe w.

   Wenn $w \neq \langle\langle G, H\rangle, c\rangle$, mit G und H zwei ungerichtete Graphen und c eine Funktion von $V(G)$ zu $V(H)$ (Menge von Tupeln aus $V(G) \times V(H)$, wobei jedes $V(G)$ nur einmal vorkommt), dann lehnne F w ab (Form-Check).

   Wenn $w = \langle\langle G, H\rangle, c\rangle$, mit G und H zwei ungerichtete Graphen und c eine Funktion von $V(G)$ zu $V(H)$ (Menge von Tupeln aus $V(G) \times V(H)$), dann führe folgendes aus:

a) Prüfe ob c eine bijektion ist, falls nicht, lehne ab.

b) Wende c auf G an und erhalte G'. Vergleiche G' mit H. Wenn $G' \neq H$, lehne ab. Sonst, akzeptiere.

Die Laufzeit von F ist polynomiell in der Länge von w:

Begründung:
Der Form-Check prüft, ob c eine Funktion ist. Dies ist polynomiell in $|V(G)|$, da für jeden Knoten geprüft wird, ob er genau einmal abgebildet wird.
Das Hauptprogramm prüft, zuerst ob c eine bijektion ist. Dies ist polynomiell in $|V(H)|$, da für jeden Knoten von H geprüft wird, ob er genau einmal abgebildet wird.
Das Hauptprogramm wendet c auf G an. Dies ist polynomiell in $|V(G)|$, da jeder Knoten von G, als auch jede Kante von G genau einmal abgebildet wird. Es gibt maximal $|V(G)|^2$ Kanten in G.
Das Hauptprogramm vergleicht G' mit H. Dies ist polynomiell in $|V(G)|$, da jeder Knoten, als auch jede Kante auf gleichheit geprüft wird. Es gibt maximal $|V(G)|^2$ Kanten in G.

Korrektheit:
F akzeptiert genau dann, wenn G und H isomorph sind und c ein Graph-Isomorphismus zwischen G und H ist.

F ist damit ein Falsifizierer für NONISO, da, wenn $\langle G, H \rangle \in$ NONISO sind G und H nicht isomorph und es existiert kein c so dass c ein Graph-Isomorphismus zwischen G und H ist.

Ist hingegen $\langle G, H \rangle \notin$ NONISO, so ist G und H isomorph und es existiert ein c so dass c ein Graph-Isomorphismus zwischen G und H ist. □

2. $L \in$ coNP $\Leftrightarrow \bar{L} \in$ NP. **Richtung $\Rightarrow$:** Angenommen $L \in$ coNP. Dann existiert ein polynomieller Falsifizierer F für L. Da F ein Falsifizierer für L ist, ist F ein Verifizierer für $\bar{L}$ (nach Definition von coNP). Da F polynomiell ist, ist $\bar{L} \in$ NP (nach Definition von NP).

   Richtung $\Leftarrow$: Angenommen $\bar{L} \in$ NP. Dann existiert ein polynomieller Verifizierer V für $\bar{L}$ (nach Definition von NP). Da V ein Verifizierer für $\bar{L}$ ist, ist V ein Falsifizierer für L (nach Definition von coNP). Da V polynomiell ist, ist $L \in$ coNP (nach Definition von coNP). □

3. P $\subseteq$ NP $\cap$ coNP. Angenommen $L \in$ P. Nun existiert eine Turingmaschine M die L in polynomieller Zeit entscheidet. Insbesondere entscheided M auch $\bar{L}$ in polynomieller Zeit.

   Nun ist aus M ein polynomieller Verifizierer für L zu konstruieren indem man M so modifiziert, dass es das Zertifikat c ignoriert und nur w prüft.

   Genau so lässt sich mit M ein polynomieller Verifiziere für $\bar{L}$ konstruieren indem man M wie zuvor modifiziert und das Ergebnis invertiert.

   □

4. NP $\neq$ coNP $\Rightarrow$ P $\neq$ NP. Angenommen NP $\neq$ coNP. Es ist uns bekannt, dass sowohl NP als auch $coNP$ nicht leer sind, und, dass der Schnitt von NP und $coNP$ nicht leer ist (enthält P).

   Das heiSSt, dass eins von beiden gilt: Es existiert ein $L \in$ NP mit L $\notin$ coNP oder es existiert ein $L \in$ coNP mit L $\notin$ NP.

   Einer dieser Fälle impliziert den anderen, mit $L \in$ NP und L $\notin$ coNP $\implies \bar{L} \in$ coNP und $\bar{L} \notin$ NP.

   Wir können also annehmen, dass es ein $L \in$ NP mit L $\notin$ coNP gibt.

   Das heiSSt, dass es einen polynomiellen Verifizierer V für L gibt, aber keinen polynomiellen Falsifizierer F für L. Der polynomielle Falsifizierer F für L kann auch hals Polynomieller Verifizierer für $\bar{L}$ verstanden werden.

   Wir wollen zeigen, dass $P \neq NP$. Wir wissen, dass $P \subseteq NP$ und müssen damit ein $L \in$ NP mit L $\notin$ P zeigen.

   Unser Kandidat für L ist das L mit $L \in$ NP und $L \notin$ coNP. Dieses L kann nicht in P sein, da es sonst auch in $coNP$ wäre. $\square$