Prof. T. Nipkow
K. Kappelmann, J. Rädle, L. Stevens **ABGABE 13.07.2020, 23:59 (Σ 6 P.)** Technische Universität München
Lehrstuhl für Logik und Verifikation

Einführung in die Theoretische Informatik

Sommersemester 2020 – Übungsblatt Lösungsskizze 11

AUFGABE 11.1. (*Unbezahltes Praktikum*) 0,5 + 1,5 + 0,5 P

Die TUM plant eine innovative neues Lehrveranstaltung; eine Mischung aus Seminar und Praktikum. In den ersten v Wochen der Veranstaltung gibt es jeweils einen Vortrag eines Studierenden, der zweite Teil besteht dann aus p Praktika. Dabei gibt es jedoch potenzielle Terminkonflikte: Von den insgesamt n Studierenden ist in jeder Woche $k \in \{1, \dots, v\}$ nur eine Teilmenge S_k verfügbar um einen Vortrag zu halten. Außerdem hat jedes Projekt als Voraussetzung, dass ein gewisser Vortrag gehalten wurde. Für jedes Projekt $i \in \{1, \dots, p\}$, gibt es eine Menge P_i an Studierenden, von denen mindestens eine/r einen Vortrag gehalten haben muss, damit das Projekt bearbeitet werden kann.

PRAKTIKUM:

Gegeben: Endliche Menge S (Studierende), Anzahl Vorträge v, Anzahl Praktika p. Teilmengen $S_k \subseteq S$ (Verfügbarkeit in Woche k), $P_i \subseteq S$ (Voraussetzungen für Praktika)

Problem: Gibt es eine Zuordnung von Studierenden zu Veranstaltungswochen, so dass jede Woche genau ein/e Student/in (der/die in dieser Woche verfügbar ist) vorträgt und dass jedes Projekt bearbeitet werden kann.

(a) Zeigen Sie, dass PRAKTIKUM in NP liegt.

(b) Zeigen Sie, dass PRAKTIKUM NP-schwer ist, indem sie eine geeignete polynomielle Reduktion von 3-KNF-SAT angeben. Zeigen Sie dabei auch die Korrektheit Ihrer Reduktion.

(c) Wenden Sie Ihre Reduktionsfunktion auf die folgende Instanz von 3-KNF-SAT an:

$$(x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

Vergewissern Sie sich dass Ihr Ergebnis Sinn macht, also dass eine erfüllende Belegung für die SAT Instanz eine Lösung für die PRAKTIKUM Instanz ergibt und umgekehrt.

Lösungsskizze

(a) Als Zertifikat dient hier eine Reihung von Studierenden. Wir überprüfen dann, dass jede/r Student/in in der zugeordneten Woche verfügbar ist und dass für jedes Projekt mindestens ein Vortrag geplant wurde. Das ist in polynomieller Zeit möglich.

(b) Sei $F := \bigwedge_{i=1}^{k} C_i$ und $C_i := \bigvee_{j=1}^{3} L_{i,j}$ mit Variablen $x_i \dots x_n$ die Eingabe für das 3KNF-SAT Problem. Wir definieren die Reduktion f, die eine Probleminstanz für PRAKTIKUM folgendermaßen konstruiert:

Erzeuge für jede Variable x_i zwei Studierende, s_i und s_i', die x_i und $\neg x_i$ repräsentieren. Es gibt dann n Veranstaltungswochen ($v := n$), in Woche i hat jeweils nur s_i und s_i' Zeit, d.h. $S_i := \{s_i, s_i'\}$.
Für jede der k Klauseln gibt es dann ein Praktikum ($p := k$), dass als Voraussetzungen genau die Studierenden hat, die zu den Literalen in der Klausel korrespondieren. Formal $P_i := \{g(L_{i,1}), g(L_{i,2}), g(L_{i,3})\}$, wobei $g(x_i) = s_i$ und $g(\neg x_i) = s_i'$. Die Reduktion f ist in polynomieller Zeit berechenbar.

Korrektheit:
Sei σ eine erfüllende Belegung für eine Formel F.
Sei $v_1, \dots, v_n$ eine Reihung mit

$$v_i = \begin{cases} s_i \text{ falls } \sigma(x_i) = 1 \\ s_i' \text{ falls } \sigma(x_i) = 0 \end{cases}$$

Dann gilt wegen der Definition von f dass $v_i \in S_i$ für alle $i \in \{1, \dots, n\}$.
Da σ eine erfüllende Belegung ist, gibt es mindestens ein Literal $L_{i,j}$ mit $\sigma(L_{i,j}) = 1$ für alle $i \in \{1, \dots, k\}$. Für ein solches Literal $L_{i,j}$ gilt dann per Konstruktion auch dass $g(L_{i,j}) \in P_i$ und dass $g(L_{i,j})$ in der Reihung $v_1, \dots, v_n$ enthalten ist. Damit erfüllt die Reihung die Anforderungen von PRAKTIKUM.
Sei umgekehrt $v_1, \dots, v_n$ eine valide Reihung von Studierenden. Konstruiere σ so, dass

$$\sigma(x_i) = \begin{cases} 1 \text{ falls } v_i = s_i \\ 0 \text{ falls } v_i = s_i' \end{cases}$$

Laut der Definition von PRAKTIKUM gilt, dass die Reihung für jedes $i \in \{1, \dots, k\}$ ein $v_j \in P_i$ enthält. Damit folgt per Konstruktion, dass es für jede Klausel C_i ein j gibt, so dass $\sigma(L_{i,j}) = 1$. Damit ist σ eine erfüllende Belegung.

(c)
- $v = 4$

- $p = 2$
- $S_i = \{s_i, s_i'\}, 1 \leq i \leq 4$
- $P_1 = \{s_1, s_3', s_4\}$
- $P_2 = \{s_1', s_2, s_3\}$

AUFGABE 11.2. (*In Form L*) 0,5 + 0,5 + 1P

Geben Sie für jede der folgenden Aussage eine informelle aber umfassende Begründung an. Ein Korrektheitsbeweis ist jeweils nicht notwendig.

(a) Sei G eine CFG in Chomsky-Normalform. Zeigen oder widerlegen Sie: $\mathsf{L}(G) \leq_p SAT$.

(b) Zeigen Sie, dass $H = \{w\#x \mid M_w[x] \downarrow\}$ NP-schwer aber nicht NP-vollständig ist. Geben Sie für die NP-Schwere eine geeignete Reduktion $A \leq_p H$ für ein NP-schweres Problem A an.

(c) Wir betrachten das RUCKSACK-Problem (siehe Folie 409) mit der Modifikation, dass b unär kodiert ist. Weiterhin gelte für alle i, dass $a_i \leq b$ Zeigen Sie, dass das RUCKSACK-Problem in P liegt, indem Sie einen Algorithmus angeben, der RUCKSACK entscheidet. Begründen Sie kurz, dass ihr Algorithmus polynomielle Laufzeit in der Länge der Eingabe hat.

Lösungsskizze

(a) Die Aussage ist wahr. Wir definieren die Reduktionsfunktion

$$f(w) = \begin{cases} \mathsf{true} \coloneqq x_0 \vee \neg x_0 & \text{wenn } w \in \mathsf{L}(G), \\ \mathsf{false} \coloneqq x_0 \wedge \neg x_0 & \text{sonst.} \end{cases}$$

Wir entscheiden $w \in \mathsf{L}(G)$ mit dem CYK-Algorithms. Da der CYK-Algorithms polynomielle Laufzeit hat, handelt es sich hierbei um eine polynomielle Reduktion.

(b) Aufgrund der Unentscheidbarkeit von H gilt $H \notin$ NP. Wir zeigen, dass H NP-schwer ist, indem wir eine Reduktion $\mathsf{SAT} \leq_p H$ skizzieren. Sei M mit Kodierung w eine TM, die als Eingabe eine binär kodierte logische Formel nimmt und alle möglichen Variablenbelegungen ausprobiert. Wenn eine erfüllende Belegung gefunden wird, dann terminiert M, ansonsten geht M in eine Endlosschleife. Die Reduktionsfunktion konstruiert für eine Formel F die Instanz $w\#F$ des allgemeinen Halteproblems. Diese Reduktion ist polynomiell, da w konstante Länge für alle Eingaben F hat. Sie ist außerdem korrekt, weil M_w genau dann für eine Eingabe F hält, wenn F erfüllbar ist.

(c) Wir benutzen dynamische Programmierung, um das Problem zu lösen. Dazu speichern wir in der Zelle $c_{i,j}$ einer Matrix C, ob es eine Teilmenge $R \subseteq \{1, \ldots, i\}$ gibt, sodass $\Sigma_{i \in R} a_i = j - 1$. Wenn nach der Ausführung $c_{n,b+1} = 1$ gilt, dann gibt es eine Menge $R \subseteq \{1, \ldots, n\}$ mit $\Sigma_{i \in R} a_i = b$. In Pseudocode:

- **Eingabe**: Binär kodierte Zahlen $a_1, \ldots, a_n \in \mathbb{N}$ und unär kodiertes $b \in \mathbb{N}$.
- Lege eine Matrix $C \in \{0,1\}^{n \times b+1}$ an, wobei anfangs $c_{i,j} = 0$ für alle i, j.
- Setze $c_{1,a_1+1} = 1$ und $c_{1,1} = 1$.
- Für alle $i \in \{2, \ldots, n\}$ und alle $j \in \{1, \ldots, b+1\}$:
  - $c_{i,j} = c_{i-1,j}$.
  - Wenn $j - a_i \geq 1$ und $c_{i-1,j-a_i} = 1$, dann $c_{i,j} = 1$.
- Gebe $c_{n,b+1}$ zurück.

Wir haben zwei geschachtelte Schleifen mit $\mathcal{O}(n)$ beziehungsweise $\mathcal{O}(b)$ Iterationen. Im Körper der Schleife werden arithmetische Operationen auf Zahlen ausgeführt, deren Größe durch b beschränkt ist. Weiterhin fällte für die Inkrementierung und Vergleich der Schleifenzähler eine Laufzeit von $\mathcal{O}(n * \log(n) + b * \log(b))$ an. Insgesamt ergibt sich also eine Laufzeit von $\mathcal{O}(n * b * \log(b) + n * \log(n))$. Da $a_i \leq b$ für alle i und b unär kodiert ist, ergibt sich eine Eingabelänge in $\mathcal{O}(n * \log(b) + b)$. Damit hat der Algorithmus polynomielle Laufzeit in der Eingabelänge.
Anmerkung: NP-schwere Probleme, die bei einer unären Kodierung der Eingabe in P liegen, werden als schwach NP-schwer bezeichnet. Umgekehrt werden Probleme, bei denen das nicht der Fall ist, stark NP-schwer genannt.

AUFGABE 11.3. (*Total abge*SPACE*d*) 0.5 + 1P

Zeit ist nicht die einzig wertvolle Ressource der Komplexitätstheorie. Die Turingmaschinen der Praxis (aka Computer) besitzen leider nur endlich viel Speicher, weswegen auch der Speicherbedarf eines Programmes von Interesse ist. Wir werden uns daher in dieser und einer kommenden Aufgabe zumindest auch ein wenig mit Speichergrenzen beschäftigen. Analog zu time_M und TIME aus der Vorlesung definieren wir hierzu:

- $\mathsf{space}_M(w)$: die Anzahl an verschiedenen, besuchten Zellen der DTM M bei Eingabe w.
- $\mathsf{SPACE}(f(n)) \coloneqq \{A \subseteq \Sigma^* \mid \exists \mathsf{DTM}\, M.\ L(M) = A \wedge \forall w \in \Sigma^*.\ \mathsf{space}_M(w) \leq f(|w|)\}$.

Unser Ziel ist es zu zeigen, dass mehr Zeit/Speicher auch wirklich mehr Power bedeutet. Wir legen hierfür einen ersten Grundstein in dieser Hausaufgabe und führen den Beweis im nächsten Hausaufgabenblatt fort. Sei $f : \mathbb{N} \to \mathbb{N}$ hierfür berechenbar und total.

(a) Zeigen Sie: $\mathsf{TIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$.

(b) Zeigen Sie: Es existiert eine *entscheidbare* Sprache $L \notin \mathsf{TIME}(f(n))$. Verwenden Sie hierfür einen Diagonalisierungsansatz ähnlich zu Satz 5.63.

Lösungsskizze

(a) Sei $A \in \mathsf{TIME}(f(n))$. Dann existiert eine DTM M mit $L(M) = A$ und $\mathsf{time}_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$. Da M maximal $f(|w|)$ Schritte macht, kann M auch maximal $f(|w|)$ viele Zellen besuchen. Somit folgt $\mathsf{space}_M(w) \leq f(|w|)$ und damit $A \in \mathsf{SPACE}(f(n))$.

(b) Definiere

$$L \coloneqq \{w \in \{0,1\}^* \mid M_w[w] \text{ akzeptiert nicht in } \leq f(|w|) \text{ Schritten}\}.$$

L ist entscheidbar: Wir simulieren bei Eingabe w die Maschine $M_w[w]$ für $f(|w|)$ Schritte (f ist total und berechenbar). Falls $M_w[w]$ in dieser Dauer akzeptiert, lehnen wir ab, sonst akzeptieren wir.
$L \notin \mathsf{TIME}(f(n))$: Angenommen $L \in \mathsf{TIME}(f(n))$. Dann gibt es eine Gödelnummer w für eine DTM M_w mit $L(M_w) = L$ und $\mathsf{time}_{M_w}(x) \leq f(|x|)$ für alle $x \in \Sigma^*$. Insbesondere gilt daher $\mathsf{time}_{M_w}(w) \leq f(|w|)$ (1). Nun folgt

$$\begin{aligned} w \in L(M_w) = L &\iff M_w \text{ akzeptiert } w \text{ und } \mathsf{time}_{M_w}(w) \leq f(|w|) && \text{(Def. } w \in L(M_w) \text{ und (1))} \\ &\iff M_w \text{ akzeptiert } w \text{ in maximal } f(|w|) \text{ Schritten} \\ &\iff w \notin L = L(M_w) && \text{(Def. } L\text{).} \end{aligned}$$

Widerspruch! Somit gilt $L \notin \mathsf{TIME}(f(n))$.

> "yields falsehood when preceded by its quotation" yields falsehood when preceded by its quotation.
>
> — Williard Van Orman Quine

Welcome to the wonderful world of paradoxes, uncertainty, and incompleteness.