

Einführung in die theoretische Informatik
Sommersemester 2017 – Übungsblatt 13

Übungsblatt

Wir unterscheiden zwischen Übungs- und Abgabebältern. Auf diesem *Übungsblatt* finden Sie eine Übersicht über die Kernaspekte, die Sie in Kalenderwoche 30 in den Tutorien diskutieren, üben und vertiefen. Die Aufgaben auf diesem Blatt dienen dem Üben und Verstehen des Vorlesungsstoffes, sowie dem *eigenständigen Erarbeiten* der Kernaspekte. Außerdem sollen Ihnen diese Aufgaben auch helfen, ein Gefühl dafür zu bekommen, was Sie inhaltlich in der Klausur erwartet. Klausuraufgaben können jedoch deutlich von den hier gestellten Aufgaben abweichen. Abschreiben und Auswendiglernen von Lösungen wird Ihnen daher keinen dauerhaften Erfolg in der Vorlesung bringen. Fragen zu den Übungsblättern können Sie montags bis donnerstags von 12 Uhr bis 14 Uhr in der *THEO-Sprechstunde* in Raum 03.11.034 stellen.

Kernaspekte

K13.1 korrektes Wiedergeben der folgenden Definitionen und Algorithmen

- P
- NP
- $\text{time}_M(w)$
- $\text{TIME}(f(n))$
- $\text{NTIME}(f(n))$
- 3-SAT
- Hamilton-Kreis
- Euler-Kreis
- Rucksack-Problem
- Zertifikat
- polynomiell beschränkter Verifikator
- polynomiell reduzierbar
- NP-hart
- NP-vollständig
- Konjunktive Normalform (KNF)
- erfüllbarkeitsäquivalent
- Negations-Normalform
- Mengenüberdeckung
- Clique
- Partition
- Bin Packing
- Travelling Salesman
- Färbbarkeit

K13.2 Aussagen über polynomielle Reduzierbarkeit beweisen oder widerlegen

K13.3 zwischen bekannten Problemen polynomielle Reduktionen angeben

K13.4 SAT auf neue Probleme über aussagenlogischen Formeln polynomiell reduzieren

K13.5 neue Probleme über aussagenlogischen Formeln auf SAT polynomiell reduzieren

K13.6 durch Angabe von polynomiell beschränkten Verifikatoren begründen, dass ein Problem in NP enthalten ist

K13.7 polynomielle Reduktionen aufstellen und ihre Korrektheit bzw. Berechenbarkeit begründen

K13.8 begründet entscheiden, ob gegebene Beispiele neu eingeführte Definitionen erfüllen

K13.9 Aussagen mit neu eingeführten Definitionen beweisen oder widerlegen

Definition (Kodierung und Dekodierung)

In der Vorlesung haben Sie gesehen, dass Reduktionen zwischen Sprachen über verschiedenen Alphabeten abbilden ($f: \Sigma^* \rightarrow \Gamma^*$). Da wir aber meistens nicht direkt über die Wörter dieser Sprachen reden wollen, sondern über Objekte wie Turing-Maschinen, Formeln oder Graphen, gibt es immer einen impliziten Kodierungs- bzw. Dekodierungsschritt. Hierbei kann aber der Fall auftreten, dass ein Wort $w \in \Sigma^*$ keine gültige Kodierung darstellt. In diesem Fall weisen wir allen falschen Kodierungen w ein festes aber beliebiges Objekt zu, wie wir das bereits für Turing-Maschinen gemacht haben. Um die Leserlichkeit von Reduktionen und Beweisen zu vereinfachen, werden wir diesen Schritte als implizit annehmen und nicht explizit behandeln.

AUFGABE 13.1. (Falsche Reduktionen)

Seien $A, B, C, D \subseteq \Sigma^*$ Sprachen über dem Alphabet $\Sigma = \{0, 1\}$. Erklären Sie, warum die folgenden Algorithmen keine polynomielle Reduktionen sind.

Stufe B

(a) Behauptung: $A \leq_p B$ mit $A := \{0^i 1^i \mid i \in \mathbb{N}_0\}$ und $B := \{w \in \Sigma^* \mid \exists i \in \mathbb{N}_0. 3i = |w|\}$

Reduktion:

$$f(w) = \begin{cases} 01 & \text{falls } |w| \text{ durch 3 teilbar ist} \\ 00 & \text{sonst} \end{cases}$$

Eine DTM kann diese Funktion in linearer Laufzeit berechnen, da sie nur prüfen muss ob die Eingabelänge ein Vielfaches von 3 ist.

- (b) Behauptung: $\text{SAT} \leq_p C$ mit $C := \{1\}$

Reduktion: Sei $F \in \mathcal{F}$. Generiere nicht-deterministisch eine Belegung σ und prüfe $\sigma(F) = 1$ in polynomieller Zeit. Falls es so ein σ gibt, gebe 1 aus. Falls es kein solches gibt, gebe 0 aus.

- (c) Behauptung: $\text{SAT} \leq_p D$ mit $D := \{w \in \{0, 1\}^* \mid |w|_1 \geq 1\}$

Reduktion: Sei $F \in \mathcal{F}$. $f(F)$ ist dann ein Wort, das wie folgt berechnet wird: Enumeriere alle Belegungen σ über den Variablen von F aufsteigend. Für jedes σ berechne $\sigma(F)$ durch einmaliges Traversieren des Syntaxbaums von F in polynomieller Zeit und füge $\sigma(F)$ an die Ausgabe an. Somit erzeugen wir ein Wort über $\{0, 1\}^*$. Falls es eine erfüllende Belegung gibt ($F \in \text{SAT}$), so gibt es mindestens eine 1 in $f(F)$ und somit $f(F) \in D$. Gibt es keine erfüllende Belegung ($F \notin \text{SAT}$), so gilt $f(F) \in \{0\}^*$ und $f(F) \notin D$.

AUFGABE 13.2. (Spaß mit SAT)

Stufe C

Wir betrachten verschiedene Varianten von SAT, die auch NP-vollständig sind. Zeigen Sie diese NP-Vollständigkeit, indem Sie für jede Variante X eine Reduktion $3\text{-KNF-SAT} \leq_p X$ angeben und $X \in \text{NP}$ zeigen.

- (a) 3-OCC-KNF-SAT:

- Eingabe: Eine Formel F in KNF, bei der jede Variable höchstens dreimal auftritt.
- Frage: Ist F erfüllbar?

- (b) Wir betrachten den ITE-Operator mit der Semantik $\text{ITE}(x, y, z) := (x \rightarrow y) \wedge (\neg x \rightarrow z)$. Eine ITE-Formel genügt der folgenden Grammatik:

$$F \rightarrow \text{ITE}(F, F, F) \mid x \mid \text{true} \mid \text{false} \quad \text{für Variablen } x \in \mathcal{V}$$

ITE-SAT:

- Eingabe: Eine ITE-Formel F .
- Frage: Ist F erfüllbar?

- (c) 1-Pro-Klausel-KNF-SAT:

- Eingabe: Eine Formel F in KNF.
- Frage: Gibt es eine erfüllende Belegung für F , so dass genau ein Literal pro Klausel auf *wahr* gesetzt wird?

AUFGABE 13.3. (Wizard of ZOLP)

Stufe D

Zeigen Sie, dass das Entscheidungsproblem *Zero-One-Linear-Program (ZOLP)* NP-schwer ist. Geben Sie hierzu eine geeignete Reduktion von 3-KNF-SAT auf ZOLP an und beschreiben Sie **zusätzlich** das Vorgehen anhand folgender Formel:

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

Definition (ZOLP-Entscheidungsproblem)

Eingabe: Ein System von linearen Ungleichungen

$$\begin{aligned} b_1 &\leq a_{1,1}y_1 + \dots + a_{1,n}y_n \\ b_2 &\leq a_{2,1}y_1 + \dots + a_{2,n}y_n \\ &\vdots \\ b_m &\leq a_{m,1}y_1 + \dots + a_{m,n}y_n \end{aligned}$$

mit $a_{i,j}, b_i \in \mathbb{Z}$ und $m, n > 0$.

Frage: Gibt es für die Variablen y_1, \dots, y_n Werte aus $\{0, 1\}$, so dass alle Ungleichungen gleichzeitig erfüllt sind?

AUFGABE 13.4. (Immer noch SAT?!)

Stufe D

In der Vorlesung haben Sie gesehen, dass 3KNF-SAT (und damit KNF-SAT) NP-vollständig ist. Man kann zeigen, dass 2KNF-SAT (maximal 2 Literale pro Klausel) allerdings noch in P liegt. Ist eine Formel in KNF unerfüllbar, so kann man immer noch versuchen, die Anzahl der gleichzeitig erfüllten Klauseln zu maximieren. Das entsprechende Problem wird mit MAX-KNF-SAT bezeichnet. Speziell ist **MAX-2KNF-SAT** das Entscheidungsproblem:

- Gegeben: Aussagenlogische Formel φ in 2KNF, Konstante $c \in \mathbb{N}$.
- Frage: Gibt es eine Belegung β , so dass *mindestens* c Klauseln von φ unter β erfüllt sind.

Ziel ist es zu zeigen, dass MAX-2KNF-SAT bereits NP-vollständig ist.

- (a) Zeigen Sie, dass MAX-2KNF-SAT in NP liegt.
 (b) Betrachten Sie die folgenden zehn Klauseln:

$$\begin{aligned} K_1 &= x & K_2 &= y & K_3 &= z & K_4 &= w \\ K_5 &= \neg x \vee \neg y & K_6 &= \neg y \vee \neg z & K_7 &= \neg z \vee \neg x & K_8 &= x \vee \neg w & K_9 &= y \vee \neg w & K_{10} &= z \vee \neg w \end{aligned}$$

Zeigen Sie:

- (i) Ist β' eine erfüllende Belegung von $x \vee y \vee z$, so lässt sich β' zu einer zu $K_1 \wedge \dots \wedge K_{10}$ passenden Belegung β erweitern, welche maximal sieben Klauseln erfüllt.
 - (ii) Die Belegung $\beta': \{x, y, z\} \rightarrow \{0, 1\}: v \mapsto 0$ lässt sich nicht zu einer Belegung β erweitern, unter der mehr als sechs der zehn gegebenen Klauseln erfüllt sind.
- (c) Sei $C = L_1 \vee L_2 \vee L_3$ eine dreielementige Klausel. Die Formel R_C sei wie folgt definiert:

$$R_C := \bigwedge_{i=1}^{10} K_i[x \mapsto L_1, y \mapsto L_2, z \mapsto L_3, w \mapsto x_C]$$

wobei $K_i[x \mapsto L_1, y \mapsto L_2, z \mapsto L_3, w \mapsto x_C]$ die Klausel ist, welche man aus K_i erhält, indem man x durch L_1 , y durch L_2 , z durch L_3 und w durch x_C (parallel) substituiert. x_C ist dabei eine neue, zuvor noch nicht verwendete Variable.

Zeigen Sie: Sei $\varphi = \bigwedge_{i=1}^k C_i$ in 3KNF und $R_\varphi := \bigwedge_{i=1}^k R_{C_i}$. Dann gilt $(R_\varphi, 7k) \in \text{MAX-2KNF-SAT}$ gdw. $\varphi \in \text{3KNF-SAT}$.

- (d) Vervollständigen Sie den Beweis, dass MAX-2KNF-SAT NP-schwer ist.
- (e) Nehmen Sie an, MAX-2KNF-SAT liegt in P.
Zeigen Sie, dass dann auch die Probleme, zu einer gegebenen 2KNF-Formel φ (i) die maximal erfüllbare Anzahl an Klauseln bzw. (ii) eine Belegung, welche eine maximale Anzahl von Klauseln erfüllt, zu berechnen, in P liegen.

AUFGABE 13.5. (*co-NP ... oder endlich nicht mehr SAT?*)

Stufe E

Wir definieren co-NP über die Komplemente von Sprachen aus NP.

$$\text{co-NP} = \{L \mid \bar{L} \in \text{NP}\}$$

Eine Sprache L heißt co-NP-vollständig, falls $L \in \text{co-NP}$ und für jedes $L' \in \text{co-NP}$ gilt: $L' \leq_p L$.

- (a) Beweisen Sie die folgende Behauptung:

Wenn die Sprache L NP-vollständig ist, so ist \bar{L} co-NP-vollständig.

- (b) Sei VALID die Menge aller Tautologien:

$$\text{VALID} := \{F \in \mathcal{F} \mid \forall \sigma : \mathcal{V} \rightarrow \{0, 1\}^*. \sigma(F) = 1\}$$

Beweisen Sie, dass VALID co-NP-vollständig ist

- (c) Betrachten Sie Definition 6.7 sowie Satz 6.10 der Vorlesung erneut (Zertifikat und polynomiell beschränkter Verifikator). Diskutieren Sie, wie eine analoge Version von Definition 6.7 und Satz 6.10 für co-NP aussehen kann. Beweisen Sie dann, ihre Analogie zu Satz 6.10.