Prof. J. Esparza
S. Sickert, J. Krämer

Technische Universität München
KEINE ABGABE

Einführung in die theoretische Informatik

Sommersemester 2017 – Übungsblatt Lösungsskizze 13

Übungsblatt

Wir unterscheiden zwischen Übungs- und Abgabeblättern. 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
- $\mathsf{time}_{\mathsf{M}}(w)$
- $\mathsf{TIME}(f(n))$
- $\mathsf{NTIME}(f(n))$
- 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 ($f\colon \Sigma^* \to \Gamma^*$). 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*) Stufe B

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

(a) Behauptung: $A \leq_p B$ mit $A \coloneqq \{0^i 1^i \mid i \in \mathbb{N}_0\}$ und $B \coloneqq \{w \in \Sigma^* \mid \exists i \in \mathbb{N}_0.\ 3i = |w|\}$
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: $\mathsf{SAT} \leq_p C$ mit $C \coloneqq \{1\}$
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: $\mathsf{SAT} \leq_p D$ mit $D \coloneqq \{w \in \{0,1\}^* \mid |w|_1 \geq 1\}$
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 ($F \in \mathsf{SAT}$), so gibt es mindestens eine 1 in $f(F)$ und somit $f(F) \in D$. Gibt es keine erfüllende Belegung ($F \notin \mathsf{SAT}$), so gilt $f(F) \in \{0\}^*$ und $f(F) \notin D$.

Lösungsskizze

(a) Die Reduktion gilt für $B \leq_p A$ und nicht für $A \leq_p B$.
(b) Reduktion muss von einer DTM in polynomieller Zeit berechnet werden können.
(c) Enumeration produziert 2^n Belegungen und somit ist die Laufzeit sowie die Ausgabe nicht polynomiell beschränkt.

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-KNF-SAT $\leq_p X$ angeben und $X \in \mathsf{NP}$ 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 $\texttt{ITE}(x, y, z) \coloneqq (x \to y) \wedge (\neg x \to z)$. Eine ITE-Formel genügt der folgenden Grammatik:

$$F \to \texttt{ITE}(F, F, F) \mid x \mid \mathsf{true} \mid \mathsf{false} \qquad \text{für Variablen } x \in \mathcal{V}$$

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?

Lösungsskizze

(a)
- **NP-schwer:** Sei F eine Formel in 3-KNF. Somit $F = \bigwedge_{i=1}^{n} K_i$ und $K_i = \bigvee_{j=1}^{m} L_{ij}$ mit $m \leq 3$. O.b.d.A. kommen keine Variablen doppelt in einer Klausel vor. Sei v die Anzahl der in F verwendeten Variablen. Dann ersetzten wir in jeder Klausel die Variablen mit für diese Klausel spezifische Variablen und fügen eine Bedingung ein, die erzwingt, dass alle Kopien einer Variable in der Eingabe den gleichen Wert zugewiesen bekommen. Diese Bedingung hat eine Ringstruktur, um Variablenverwendungen zu sparen: $(x_{k,1} \to x_{k,2}) \wedge (x_{k,2} \to x_{k,3}) \wedge \ldots (x_{k,n} \to x_{k,1})$.

$$f(F) = \bigwedge_{i=1}^{n} f(K_i) \wedge \bigwedge_{k=1}^{v} \bigwedge_{i=1}^{n} (\neg x_{k,i} \vee x_{k,(i \bmod n)+1})$$

$$f(K_i) = \bigvee_{j=1}^{m} f(L_{ij})$$

$$f(L_{ij}) = \begin{cases} x_{k,i} & \text{falls } L_{ij} = x_k \\ \neg x_{k,i} & \text{falls } L_{ij} = \neg x_k \end{cases}$$

  Diese Reduktion ist in polynomieller Zeit berechenbar, da nur die Eingabe einmal kopiert werden muss und ein zusätzlicher Term der Größe $m \cdot v$ erzeugt wird.
  Korrektheit: Aus jeder erfüllenden Belegung σ für F können wir eine erfüllende Belegung σ' für $f(F)$ konstruieren mit $\sigma'(x_{k,i}) = \sigma(x_k)$. Auch können wir aus jeder erfüllenden Belegung σ für $f(F)$ ein Belegung für F konstruieren. Da $\sigma(x_{k,i}) = \sigma(x_{k,j})$ für alle i, j gelten muss, können wir einfach σ' über $\sigma'(x_k) = \sigma(x_{k,1})$ definieren. Somit ist die Konstruktion erfüllbarkeitserhaltend und die Reduktion korrekt.
- $\in$ **NP**: Eine erfüllende Belegung ist ein geeignetes Zertifikat, das es maximal genauso groß ist wie die

Formel F und in polynomieller Zeit geprüft werden kann.

(b)
- **NP-schwer:** Sei F eine Formel in 3-KNF. Wir wandeln F in polynomieller Zeit unter der Verwendung der folgenden semantischen Äquivalenzen in eine ITE-Formel:

$$\neg x \equiv \texttt{ITE}(x, \mathsf{false}, \mathsf{true}) \qquad x \wedge y \equiv \texttt{ITE}(x, y, \mathsf{false}) \qquad x \vee y \equiv \texttt{ITE}(x, \mathsf{true}, y)$$

Eine Klausel mit drei Literalen, z.B. $\neg x \vee \neg y \vee \neg z$ lässt sich damit umformen zu

$$\neg x \vee \neg y \vee \neg z \equiv \texttt{ITE}(\texttt{ITE}(x, \mathsf{false}, \mathsf{true}), \mathsf{true}, \texttt{ITE}(\texttt{ITE}(y, \mathsf{false}, \mathsf{true}), \mathsf{true}, \texttt{ITE}(z, \mathsf{false}, \mathsf{true})))$$

Jede Klausel wird in eine ITE-Formel übersetzt, welche nur um einen konstanten Faktor größer ist. Entsprechend übersetzt sich jede 3KNF-Formel in eine semantisch äquivalente ITE-Formel, die nur um einen konstanten Faktor größer ist. Die Korrektheit folgt aus der Verwendung semantisch-äquivalenten Umformungen.
- $\in$ **NP**: Eine erfüllende Belegung ist ein geeignetes Zertifikat, das es maximal genauso groß ist wie die Formel F und in polynomieller Zeit geprüft werden kann.

(c) Eine ausführliche Lösung finden Sie auf: `http://cs.nyu.edu/courses/fall14/CSCI-UA.0310-001/1in3sat.pdf`

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

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

mit $a_{i,j}, b_i \in \mathbb{Z}$ und $m, n > 0$.
Frage: Gibt es für die Variablen $y_1, \ldots, y_n$ Werte aus $\{0, 1\}$, so dass alle Ungleichungen gleichzeitig erfüllt sind?

Lösungsskizze
Sei $F := \bigwedge_{i=1}^{k} C_i$ und $C_i := \bigvee_{j=1}^{3} L_{i,j}$ mit z Variablen die Eingabe für das 3KNF-SAT Problem. Die Reduktion f modelliert jede Variable des SAT-Problems mit zwei Variablen des ZOLP-Problems. Sei x_i eine Variable dann encodiert y_{2i}, ob das Literal x_i wahr ist, und y_{2i-1}, ob das Literal $\neg x_i$ wahr ist. Somit haben wir

$$f(x_i) \mapsto y_{2i} \qquad f(\neg x_i) \mapsto y_{2i-1}$$

Für das ZOLP wählen wir $m = k + 2z$ und $n = 2z$. Die Zeile $i \in [1, k]$ ist dann definiert in Abhängigkeit von C_i als

$$f(L_{i,1} \vee L_{i,2} \vee L_{i,3}) \qquad \mapsto \qquad 1 \leq f(L_{i,1}) + f(L_{i,2}) + f(L_{i,3})$$

Zusätzlich fügen wir folgende Konsistenzgleichungen ein damit y_{2i} und $y_{2i} - 1$ unterschiedliche Werte erhalten für alle $i \in [1, z]$

$$\begin{aligned} 1 &\leq y_{2i} + y_{2i-1} \\ -1 &\leq -y_{2i} - y_{2i-1} \end{aligned}$$

Die Reduktion ist offensichtlich polynomiell in der Eingabegröße.
Anwendung auf Beispiel:

$$\begin{aligned} 1 &\leq y_1 + y_4 + y_6 \\ 1 &\leq y_2 + y_3 + y_8 \\ 1 &\leq y_1 + y_2 \\ -1 &\leq -y_1 - y_2 \\ 1 &\leq y_3 + y_4 \\ -1 &\leq -y_3 - y_4 \\ 1 &\leq y_5 + y_6 \\ -1 &\leq -y_5 - y_6 \\ 1 &\leq y_7 + y_8 \\ -1 &\leq -y_7 - y_8 \end{aligned}$$

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:

$$K_1 = x \qquad K_2 = y \qquad K_3 = z \qquad K_4 = w$$

$$K_5 = \neg x \vee \neg y \qquad K_6 = \neg y \vee \neg z \qquad K_7 = \neg z \vee \neg x \qquad K_8 = x \vee \neg w \qquad K_9 = y \vee \neg w \qquad K_{10} = z \vee \neg w$$

Zeigen Sie:

(i) Ist β' eine erfüllende Belegung von $x \vee y \vee z$, so lässt sich β' zu einer zu $K_1 \wedge \ldots \wedge K_{10}$ passenden Belegung β erweitern, welche maximal sieben Klauseln erfüllt.

(ii) Die Belegung $\beta' \colon \{x, y, z\} \to \{0, 1\} \colon v \mapsto 0$ 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 $(R_\varphi, 7k) \in$ MAX-2KNF-SAT gdw. $\varphi \in$ 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.

Lösungsskizze

(a) Erfüllende Belegung raten, Formel auswerten und die erfüllten Klauseln zählen. Die Verifikation kann in polynomieller Zeit von einer DTM ausgeführt werden.

(b) (i) Da die Klauseln symmetrisch bzgl. x, y, z sind, betrachten wir nur den Fall $x \mapsto 1$ und $y, z \mapsto 0$. Dann sind K_1, K_5, K_6, K_7, K_8 erfüllt. Durch die Wahl $w \mapsto 0$ werden noch K_9 und K_{10} erfüllt, somit werden insgesamt 7 Klauseln erfüllt. Durch die Wahl $w \mapsto 1$ wird nur noch K_4 erfüllt und somit nur 6 Klauseln.
Wir betrachten nun den Fall $x, y \mapsto 1$ und $z \mapsto 0$. Dann sind $K_1, K_2, K_6, K_7, K_8, K_9$ erfüllt. Durch die Wahl $w \mapsto 0$ wird noch K_{10} erfüllt, somit werden insgesamt 7 Klauseln erfüllt. Durch die Wahl $w \mapsto 1$ wird nur noch K_4 erfüllt und somit auch 7 Klauseln.
Wir betrachten nun den Fall $x, y, z \mapsto 1$. Dann sind $K_1, K_2, K_3, K_8, K_9, K_{10}$ erfüllt. Durch die Wahl $w \mapsto 0$ werden keine weiteren Klauseln erfüllt. Durch die Wahl $w \mapsto 1$ wird nur noch K_4 erfüllt und somit 7 Klauseln.

(ii) Unter dieser partiellen Belegung sind K_1, K_2, K_3 trivialer Weise nicht erfüllt und K_5, K_6, K_7 sind erfüllt. Wenn wir $w \mapsto 0$ setzen, dann sind noch K_8, K_9, K_{10} erfüllt. Somit erhalten wir 6 erfüllte Klauseln. Durch die Wahl $w \mapsto 1$ erhalten wir nur 4 erfüllte Klauseln.

(c) In (b) haben wir gezeigt, dass von R_C maximal 7 Klauseln erfüllt werden können und das ist nur der Fall, wenn $x \vee y \vee z$ erfüllt ist. Wenn $(R_\varphi, 7k) \in$ MAX-2KNF-SAT gilt, dann hat R_C für jedes $C = L_1 \vee L_2 \vee L_3$ genau 7 erfüllte Klauseln. Somit existiert eine Belegung β, die jedes C erfüllt und somit $\varphi \in$ 3KNF-SAT. Gegenrichtung analog.

(d) Man muss sich nur noch vergewissern, dass die Abbildung $\varphi \mapsto (R_\varphi, 7k)$ in P berechenbar ist. Da k die Anzahl der Klauseln von φ ist, ist k unär durch φ gegeben. Aus jeder k Klauseln erzeugt man 10 neue Klauseln, was in Zeit $\mathcal{O}(10k)$ geht.

(e) (i) Binäre Suche nach dem maximalen c benötigt maximal $\log_2 k \leq \log_2 |\varphi|$ viele Aufrufe der P-Entscheidungsprozedur für MAX-2KNF-SAT.
(ii) Seien $x_1, \ldots, x_m$ die in φ vorkommenden Variablen (also $m \leq |\varphi|$).
Algorithmus:
- Bestimme maximales c_φ entsprechend (i) in P.
- Sei β die überall undefinierte Belegung.
- Für $i = 1, \ldots, m$:
  Sei $\varphi' := \varphi[x_i \mapsto 1]$.
  Bestimme maximales $c_{\varphi'}$ in P.
  Falls $c_{\varphi'} = c_\varphi$, setze $\beta(x_i) := 1$ und $\varphi := \varphi'$; sonst $\beta(x_i) := 0$ und $\varphi := \varphi[x_i \mapsto 0]$.
- Gib β zurück.

Der Algorithmus führt die Funktion aus (i) genau $m \leq |\varphi|$ Mal aus, ist somit auch polynomiell in $|\varphi|$.

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

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

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

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

(a) Beweisen Sie die folgende Behauptung:

Wenn die Sprache L NP-vollständig ist, so ist $\overline{\mathsf{L}}$ co-NP-vollständig.

(b) Sei VALID die Menge aller Tautologien:

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

Beweisen Sie, dass VALID co-NP-vollständig ist

(c) Betrachten Sie Definition 6.7 sowie Satz 6.9 der Vorlesung erneut (Zertifikat und polynomiell beschränkter Verifikator). Diskutieren Sie, wie eine analoge Version von Definition 6.7 und Satz 6.9 für co-NP aussehen kann. Beweisen Sie dann, ihre Analogie zu Satz 6.9.

Lösungsskizze

(a) Offensichtlich folgt aus $\mathsf{L} \in \mathsf{NP}$: $\overline{\mathsf{L}} \in \mathsf{co\text{-}NP}$. Sei nun $\mathsf{L}' \in \mathsf{co\text{-}NP}$ eine beliebige Sprache. Weil L NP-vollständig ist, haben wir $\overline{\mathsf{L}'} \leq_p \mathsf{L}$ und somit auch $\mathsf{L}' \leq_p \overline{\mathsf{L}}$. Damit ist $\overline{\mathsf{L}}$ co-NP-vollständig.

(b) Um (b) zu zeigen, verwenden wir (a) und zeigen, dass $\overline{\mathsf{VALID}} = \{F \in \mathcal{F} \mid \exists \sigma : \mathcal{V} \to \{0,1\}^*.\ \sigma(F) = 0\}$ NP-vollständig ist. $\overline{\mathsf{VALID}} \in \mathsf{NP}$ folgt daraus, dass σ mit $\sigma(F) = 0$ ein polynomielles Zertifikat ist, das wir in polynomieller Zeit verifizieren können. Aus der Reduktion $f(F) = \neg F$ erhalten wir $\mathsf{SAT} \leq_p \overline{\mathsf{VALID}}$ und somit is VALID co-NP-vollständig.

(c) Anstatt kurzer Zertifikate für $w \in \mathsf{L}$ haben wir jetzt kurze Zertifikate für $w \notin \mathsf{L}$.