Technische Universität München
Prof. J. Esparza / Dr. M. Luttenberger, S. Sickert

Sommer 2016
24. Juli 2016

HA-Lösung

TA-Lösung

Einführung in die theoretische Informatik – Aufgabenblatt 12

Beachten Sie: Soweit nicht explizit angegeben, sind Ergebnisse stets zu begründen!

Hausaufgaben: Abgabe bis zum 20.07.2016 (Mittwoch) um 12:00

Aufgabe 12.1 **PCP** **1P+1P+2P+3P**

(a) Bestimmen Sie *alle* Lösungen für das folgende PCP: $P_1 = ((d, cd), (d, d), (abc, ab))$.

(b) Zeigen Sie, dass die folgende Instanz des PCPs keine Lösung hat: $P_2 = ((ab, aba), (baa, aa), (aba, baa))$.

(c) Zeigen Sie, dass das Post'sche Korrespondenzproblem über einem Alphabet mit nur einem Symbol entscheidbar ist. Geben Sie hierzu einen Algorithmus an und begründen Sie dessen Korrektheit.

(d) Sei $P = (c_1, c_2)$ ein PCP über einem beliebigem Alphabet Σ mit $c_i = (x_i, y_i)$ und $||x_i| - |y_i|| = 1$ für $i \in \{1, 2\}$. Zeigen Sie die Entscheidbarkeit für diese Variante des PCPs. Geben Sie hierzu einen Algorithmus an und begründen Sie dessen Korrektheit.

Lösung:

(a) $L = L((2 \mid (31))^*) \setminus \{\varepsilon\}$

(b) Zu Beginn kann nur das Tupel (ab, aba) verwendet werden, so dass wir mit (ε, a) fortfahren. Offensichtlich kann nun weder (ab, aba) noch (baa, aa) angewendet werden. Durch (aba, baa) erhalten wir aber $(aba, abaa)$, was wiederum zu (ε, a) reduziert werden kann. Damit kann also diese Instanz des Post'schen Korrespondenzproblems keine Lösung besitzen.

(c) In diesem Fall haben alle Karten c die Form $c = (a^i, a^j)$ für $\Sigma = \{a\}$ und $i, j \geq 0$. Falls $i = j$, ist die Karte c alleine eine Lösung. Wenn alle Karten oben länger als unten $(i > j)$ sind, gibt es keine Lösung. Analog für den umgekehrten Fall $(i < j)$.

Wähle nun zwei Karten c_1, c_2 mit $c_1 = (a^{j_1}, a^{k_1})$, $c_2 = (a^{j_2}, a^{k_2})$, $j_1 > k_1$ und $j_2 < k_2$. Sei i_1 der Index von c_1 und sei i_2 der von c_2. Dann ist $i_1^{k_2 - j_2} i_2^{j_1 - k_1}$ eine Lösung des PCP.

Eine TM kann einfach diese Vorbedingungen prüfen und somit bestimmen, ob es eine Lösung gibt.

(d) Entscheidbar, da es genau dann eine Lösung gibt, wenn 12 oder 21 eine Lösung ist, und diese beiden Fälle von einer TM geprüft werden können.

Beweis:

Sei $i_1 i_2 \dots i_k$ eine Lösung von P. Aus der Längenbedingung der Karten folgt sofort, dass $k \geq 2$ gilt. Falls $i1 \neq i2$, dann gilt $|x_{i_1} x_{i_2}| = |y_{i_1} y_{i_2}|$. Somit ist bereits $i_1 i_2$ (12 oder 21) eine Lösung.

Sei nun $i_1 = i_2$ und o.B.d.A. $i1 = i2 = 1$. Wir nehmen ebenfalls o.B.d.A. an, das $|x_1| > |y_1|$ gilt. Somit $x_1 = u_1 \dots u_n u_{n+1}$ und $y_1 = u_1 \dots u_n$ mit $u_i \in \Sigma$ gilt. Da $i_1 i_2$ Teil einer Lösung ist, gilt für $(x_1 x_1, y_1 y_1)$:

$$x_1 x_1 = y_1 y_1 u_n u_{n+1}$$

und somit

$$u_{n+1} = u_1 = u_2 = \dots = u_{n-1} = u_n$$

Die Karte c_1 hat somit die Gestalt $c_1 = (a^{n+1}, a^n)$ für irgendeinen Buchstaben $a \in \Sigma$.

In der Lösung muss es auch eine solche Teilsequenz (22) für c_2 geben, damit der Überhang von c_1 ausgeglichen wird. Analog folgt dann $c_2 = (b^m, b^{m+1})$ für ein $b \in \Sigma$. Da c_1 und c_2 sich überschneiden, gilt $a = b$. Somit ist auch 12 eine Lösung ist.

Aufgabe 12.2 **Entscheidbarkeit und kontextfreie Grammatiken** **3P+3P**

Seien G_1, G_2 CFGs.

(a) Zeigen Sie, dass das Problem $L(G_1) \not\subseteq L(G_2)$ semi-entscheidbar ist.

(b) Zeigen Sie, dass das Problem $L(G_1) \subseteq L(G_2)$ unentscheidbar ist.

Hinweis: In der Vorlesung wurde das Resultat nur erwähnt, zeigen Sie das Resultat jetzt formal. Sie dürfen verwenden, dass für G_1, G_2 CFGs das Problem $L(G_1) \cap L(G_2) = \emptyset$ unentscheidbar ist. Schauen Sie sich den entsprechenden Beweis in den Folien an. Charakterisieren Sie die dort verwendeten CFGs möglichst genau (linear, rechtslinear, linkslinear, deterministisch?).

Lösung:

(a) Es gilt $L(G_1) \not\subseteq L(G_2)$ genau dann, wenn es ein $w \in L(G_1) \setminus L(G_2)$ gibt.

Alle Ableitungen bzgl. G_1 kann man mittels BFS (Breitensuche) der Länge nach und damit alle $w \in L(G_1)$ aufzählen. Für jedes $w \in L(G_1)$ testet man z.B. mittels CYK (oBdA ist G_2 in CNF), ob $w \in L(G_2)$ gilt. Sobald man das erste $w \in L(G_1) \setminus L(G_2)$ gefunden hat, stoppt man und gibt 1 aus. Offensichtlich stoppt der Algorithmus um Fall $L(G_1) \not\subseteq L(G_2)$ stets, im Fall $L(G_1) \subseteq L(G_2)$ terminiert der Algorithmus allerdings nie. Damit ist das Problem semi-entscheidbar.

(b) Es gilt $L(G_1) \subseteq L(G_2)$ **gdw** $L(G_1) \setminus L(G_2) = \emptyset$ **gdw** $L(G_1) \cap \mathsf{A}^* \setminus L(G_2) = \emptyset$.

(OBdA verwenden G_1 und G_2 dasselbe Alphabet A.)

Sei $(x_1, y_1), \ldots, (x_l, y_l) \in \Gamma^* \times \Gamma^*$ eine PCP-Instanz. Sei $\Sigma = \{a_1, \ldots, a_l\}$. OBdA $\Gamma \cap \Sigma = \emptyset$ und $\mathsf{A} = \Sigma \cup \Gamma$, da wir Σ frei wählen können. Dann sind die in der Vorlesung verwendeten Grammatik G_1, G_2 linear und die erzeugten Sprachen sogar deterministisch:

Im Fall von G_1 liest ein DPDA zunächst die $a_{i_n} \ldots a_{i_1}$ und legt währenddessen die entsprechenden x_{i_κ} auf den Stack. Anschließend überprüft er den Stack mit der Eingabe. Sobald der Stack aufgebraucht ist, prüft der DPDA, dass das nächste Eingabezeichen \$ ist, dann pushed er alle verbleibenden Zeichen aus Γ wieder auf den Stack, um anschließend deterministisch mit den verbleibenden Zeichen aus Σ zu überprüfen, ob es sich um die richtigen y_{j_μ} handelt.

Im Fall von G_2 folgt ein DPDA demselben Muster, ist allerdings noch einfacher.

Da DCFL effektiv unter Komplement nach Vorlesung abgeschlossen sind, kann man aus G_2 bzw. dem entsprechenden DPDA eine CFG G_2' mit $L(G_2') = \overline{L(G_2)}$ konstruieren.

Damit gilt: $L(G_1) \subseteq L(G_2')$ **gdw** $L(G_1) \cap \overline{L(G_2')} = \emptyset$ **gdw** $L(G_1) \cap L(G_2) = \emptyset$ **gdw** die gegebene PCP-Instanz hat keine Lösung.

Aufgabe 12.3 **ITE-SAT** **3P**

Wir betrachten (mal wieder) den ITE-Operator für boolesche Ausdrücke:

Zur Erinnerung, die Semantik ist durch $\mathsf{ITE}(x, y, z) := (x \to y) \wedge (\neg x \to z)$ definiert. Eine ITE-Formel genügt der folgenden Grammatik in BNF:

$$F ::= \mathsf{ITE}(F, F, F) \mid x_i \mid \mathsf{true} \mid \mathsf{false}$$

Eine ITE-Formel wird mittels des ITE-Operators aus den aussagenlogischen Variablen $\{x_i \mid i \in \mathbb{N}\}$ und den konstanten Wahrheitsfunktion true und false gebildet.

Zeigen Sie, dass ITE-SAT, d.h. das Problem zu entscheiden, ob eine gegeben ITE-Formel erfüllbar ist, **NP**-vollständig ist.

Lösung:

- **NP-schwer:** Wir reduzieren KNF-SAT auf ITE-SAT in polynomieller Zeit unter der Verwendung der folgenden semantischen äquivalenzen:

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

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

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

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.

- Da die Reduktion eine ITE-Formel in eine semantisch äquivalente 3KNF-Formel in Linearzeit überführt, womit erfüllende Belegungen erhalten bleiben, folgt sofort, dass ITE-SAT auch in **NP** liegt.

Aufgabe 12.4 **Meterstab-Problem** **3P**

Zeigen Sie, dass das Meterstab-Problem NP-vollständig ist.

Gegeben: Ein Meterstab mit Gliedern unterschiedlicher Länge $l_1, \dots l_n \in \mathbb{N}$ und eine Taschenlänge $l \in \mathbb{N}$.

Problem: Läßt sich der Meterstab einpacken? Formal: Lässt er sich zu einer Länge $\leq l$ zusammenfalten?

Hinweis: Reduzieren Sie PARTITION auf das Meterstab-Problem. Zeigen Sie dafür, dass eine Eingabe $a_1, \dots, a_n$ von Partition eine positive Lösung genau dann hat, wenn sich der Meterstab mit den Gliedern der Länge A, $A/2$, a_1, …, a_n, $A/2$, A auf die Länge A falten lässt, wobei $A := \sum_{i=1}^{n} a_i$.

Lösung:

- Liegt in **NP**: Rate „Faltstellen“ und überprüfe die Längenbeschränkungen in polynomieller Zeit.
- Wir stellen uns den Meterstab entlang der Zahlengerade vor. Dann kann eine Faltung des Meterstabs durch Zahlen $x_i \in \{-1, 1\}$ codiert werden, die einfach angeben, in welche Richtung man a_i Einheiten entlang des Zahlenstrahls sich bewegt. Ein Falten des Meterstabs bei i entspricht dann $x_i x_{i+1} = -1$.

  Sei nun $a_1, \dots, a_n$ eine Probleminstanz für PARTITION.

  Wie angegeben bildet man diese auf die Segmente $b_{-1}, b_0, b_1, \dots, b_n, b_{n+1}, b_{n+2} = A, A/2, a_1, \dots, a_n, A/2, A$ mit $A = \sum_{i=1}^{n} a_i$ ab – die Abbildung lässt sich in PTIME berechnen.

  OBdA gilt dann $x_{-1} = 1$. Da die maximale Ausdehnung $l = A$ sein soll, muss damit auch $x_0 = -1$ gelten. Eine zulässige Faltung muss sich dann im Intervall $[0, A]$ auf der Zahlengerade bewegen, d.h. für alle $k = -1, 0, \dots, n+1, n+2$ muss $\sum_{i=-1}^{k} x_i a_i \in [0, A]$ gelten bzw. für alle $k = 1, 2, \dots, n$ dann $\sum_{i=1}^{k} x_i a_i \in [-A/2, A/2]$. Annahme: $A - A/2 + \sum_{i=1}^{n} x_i a_i = c \neq A/2$. Gilt $c > A/2$, so muss $x_{n+1} = -1$ gelten, also $\sum_{i=-1}^{n+1} x_i a_i = c - A/2 > 0$, womit man in beiden Fällen $x_{n+2} = \pm 1$ aus $[0, A]$ rausfliegt. Analog für $c < A/2$.

  Eine zulässige Faltung muss damit $\sum_{i=1}^{n} x_i a_i = 0$ erfüllen. Setze dann $I = \{i \in [n] \mid x_i = 1\}$. Dann folgt sofort $\sum_{i \in I} a_i = \sum_{i \in [n]-I} a_i = A/2$.

  Vollkommen analog sieht man, dass jede Partition I eine zulässige Faltung induziert.

Aufgabe 12.5 **Satz von Rice** **3P**

Sind folgende Menge unentscheidbar ($\Sigma = \{0, 1\}$)? Wenn möglich, verwenden Sie bitte den Satz von Rice. Dabei geben Sie bitte die Menge $\mathcal{F}$ genau an und argumentieren, warum die Menge nicht trivial ist.

- $L_1 = \{w \in \Sigma^* \mid L(M_w) \text{ ist regulär}\}$
- $L_2 = \{w \in \Sigma^* \mid \forall n \in \mathbb{N}.\ \varphi_w(n) = n * (n - 23) + 42\}$
- $L_3 = \{w \in \Sigma^* \mid \forall p \in \mathbb{N}.\ (|w| < p \wedge p \text{ ist prim}) \to w_p = 0\}$

Lösung:

- Sei $\mathcal{F} = \{f \mid f \text{ ist berechenbar } \wedge f^{-1}(1) \text{ ist regulär}\}$.

  Dann gilt $w \in L_1$ **gdw** $\chi_{L(M_w)} \in \mathcal{F}$. Offensichtlich ist $\mathcal{F}$ nicht die Menge aller berechenbarer Funktionen. Damit folgt mit Rice, dass L_1 unentscheidbar ist.

  Alternativ, indem man den Beweis von Rice explizit verwendet: $w \mapsto w'$ mit w' die Codierung von $M_w(\varepsilon)$; return $y \in \{a^n b^n \mid n \in \mathbb{N}_0\}$.

  Dann gilt: Hält M_w auf ε, dann gilt $L(M_{w'}) = \{a^n b^n \mid n \in \mathbb{N}\} \notin L_1$.

  Hält M_w nicht auf ε, dann hält auch $M_{w'}$ nicht, also $L(M_{w'}) = \emptyset \in L_1$.
- Sei $\mathcal{F} = \{f \mid f \text{ ist berechenbar } \wedge \forall n \in \mathbb{N}.\ f(n) = n * (n - 23) + 42\}$. Dann ist gilt für $g(n) = 0$: $g \notin \mathcal{F}$ und somit ist $\mathcal{F}$ nicht die Menge aller berechenbarer Funktionen.

  Alternativ kann wieder der Beweis von Rice verwendet werden: Bilde w auf Codierung w' von $M_w(\varepsilon)$; return $n * (n - 23) + 42$; ab.
- L_3 ist entscheidbar, da w nur syntaktischen Kriterien erfüllen muss. Eine TM kann alle Primzahlen kleiner gleich $|w|$ berechnen und an diesen Stellen in w prüfen, ob $w_p = 0$ gilt.

Aufgabe 12.6 **2P**

Wir nehmen an, dass P ein (WHILE-)Programm ist, das SAT in polynomieller Zeit entscheidet.

Konstruieren Sie unter Verwendung von P als Hilfsfunktion ein Programm P', das in polynomieller Zeit zu einer gegebenen aussagenlogischen Formel ϕ eine erfüllende Belegung β berechnet.

Lösung: P als Orakel verwenden, um schrittweise zu entscheiden, ob $\phi[x_1 := b_1, \dots x_{i-1} := b_{i-1}, x_i := 0]$ bzw. $\phi[x_1 := b_1, \dots x_{i-1} := b_{i-1}, x_i := 1]$ noch erfüllbar ist. Insgesamt werden $2\,|\mathsf{Var}(\phi)| \leq 2\,|\phi|$ viele Aufrufe von P benötigt, P' läuft damit selbst auch in **PTIME**.

Aufgabe 12.7 **Semi-Entscheidbarkeit** **2P**

Zeigen Sie:

A ist semi-entscheidbar gdw. A ist Wertebereich einer berechenbaren Funktion

Lösung: A semi-entscheidbar $\rightsquigarrow$ A aufzählbar $\rightsquigarrow$ A Wertebereich eine berechenbaren Funktion (gerade die Funktion, die A aufzählt)

A Wertebereich einer berechenbaren Funktion $f \rightsquigarrow$ sei T TM zu f, simuliere T für ansteigende Grenze N auf allen Eingaben der Länge $\leq N$ für N Schritte, falls Simulation terminiert, gib berechneten Wert aus $\rightsquigarrow$ A aufzählbar, da schließlich jede Eingabe und damit jeder Ausgabe erzeugt wird $\rightsquigarrow$ A semi-entscheidbar.

Tutoraufgaben: Besprechung in KW28

Aufgabe 12.1 Entscheidbarkeit

Geben Sie für jedes der folgenden Probleme an, ob es entscheidbar oder unentscheidbar ist. Falls es unentscheidbar ist, geben Sie an, ob das Problem selbst oder sein Komplement semi-entscheidbar ist. Begründen Sie stets Ihre Antwort.

Gegeben sei eine (1-Band-)Turing-Maschine M:

(a) Hält M auf der leeren Eingabe?

(b) Gibt es eine Eingabe, auf der M hält?

(c) Schreibt M jemals ein gegebenes x (bezogen auf alle möglichen Eingaben)?

(d) Ändert M jemals ein Zeichen auf dem Band bei leerer Eingabe?

(e) Gilt $L(M) = \{0,1\}^*$? ($L(M)$ ist die von M akzeptierte Sprache, d.h. M hat eine akzeptierende Berechnung für jedes $w \in L(M)$. OBdA gilt $L(M) \subseteq \{0,1\}^*$.)

Lösung:

(a) Unentscheidbar: H_0

Semi-entscheidbar: Simuliere M.

(b) Unentscheidbar: Reduktion von H_0: Bilde M auf $x_0{:=}M(\varepsilon)$ ab

Semi-entscheidbar: Zähle alle Eingaben w auf und simuliere alternierend M auf allen Eingaben.

(c) Unentscheidbar: Reduktion von H_0: Sei γ kein Bandzeichen von M. Bilde M auf $x_1{:=}M(\varepsilon); x_0{:=}\gamma;$ ab.

Semi-entscheidbar: Zähle alle Eingaben w auf und simuliere alternierend M auf allen Eingaben bis x geschrieben wird.

(d) Entscheidbar: Simuliere $M[\varepsilon]$ bis entweder ein Zeichen geschrieben wird oder ein Kontrollzustand q doppelt besucht wird. Im Ersten Fall antworte 1, im zweiten 0. Da die Kontrolle der TM endlich ist, tritt einer der beiden Fälle nach einer endlichen Anzahl von Schritten ein.

(e) Unentscheidbar: Sowohl $L(M) = \{0,1\}^*$ als auch $L(M) \neq \{0,1\}^*$ sind nicht semi-entscheidbar:

Wir zeigen, dass sich sowohl H_0 als auch $\overline{H_0}$ auf das Problem reduzieren lassen.

Reduktion von H_0: Bilde Codierung w auf Codierung w' der TM $M_{w'}(y) := M_w(\varepsilon)$; return 1; ab. Dann $L(M_{w'}) = \{0,1\}^*$ gdw. $w \in H_0$. (Damit reduziert sich $\overline{H_0}$ auf $L(M) \neq \{0,1\}^*$.)

Reduktion von $\overline{H_0}$: Bilde Codierung von w auf Codierung w' der TM $M_{w'}(y)$ ab, welche $M_w(\varepsilon)$ für genau $|y|$ viele Schritte simuliert; terminiert die Simulation innerhalb der Zeitschranke, lehne y ab bzw. gehe in Endlosschleife, ansonsten akzeptiere y. Dann gilt $L(M_{w'}) = \{0,1\}^*$ gdw. $M_w(\varepsilon)$ hält nicht nach endlich vielen Schritten gdw. $w \notin H_0$. (Damit reduziert sich H_0 auf $L(M) \neq \{0,1\}^*$.)

Wäre $L(M) = \{0,1\}^*$ oder $L(M) \neq \{0,1\}^*$ semi-entscheidbar, dann wären somit sowohl H_0 als auch $\overline{H_0}$ semi-entscheidbar, d.h. H_0 selbst wäre entscheidbar.

Aufgabe 12.2 Reduktionen und NP-Vollständigkeit

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

Lösung:

(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 $(R_\phi, 7k) \in$ 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 $\phi \in$ 3KNF-SAT. Gegenrichtung analog.

(d) Man muss sich nur noch vergewissern, dass die Abbildung $\phi \mapsto (R_\phi, 7k)$ 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 $\log_2 k \leq \log_2 |\phi|$ viele Aufrufe der **P**-Entscheidungsprozedur für MAX-2KNF-SAT.

(ii) Seien $x_1, \ldots, x_m$ die in ϕ vorkommenden Variablen (also $m \leq |\phi|$).

Algorithmus:

- Bestimme maximales c_ϕ entsprechend (i) in **P**.
- Sei β die überall undefinierte Belegung.
- Für $i = 1, \ldots, m$:

  Sei $\phi' := \phi[x_i \mapsto 1]$.

  Bestimme maximales $c_{\phi'}$ in **P**.

  Falls $c_{\phi'} = c_\phi$, setze $\beta(x_i) := 1$ und $\phi := \phi'$; sonst $\beta(x_i) := 0$ und $\phi := \phi[x_i \mapsto 0]$.
- Gib β zurück.

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

Aufgabe 12.3 **Entscheidbarkeit**

Sei $\Sigma = \{a_1, \ldots, a_k\}$ ein endliches Alphabet. Seien $f, g\colon \Sigma^* \to \Gamma^*$ zwei Homomorphismen, d.h. $f(\varepsilon) = \varepsilon$ und $f(uv) = f(u)f(v)$ für alle $u, v \in \Sigma^*$ und entsprechend für g. f, g seien durch Auflistung Ihrer Bilder auf Σ gegeben: $(f(a_i))_{i\in[k]}, (g(a_i))_{i\in[k]}$.

Zeigen Sie:

(a) Sei $L \subseteq \Sigma^*$ regulär. Es ist unentscheidbar, ob es ein Wort $w \in L$ mit $f(w) = g(w)$ gibt.

Hinweis: Verwenden Sie eine Reduktion von PCP.

(b) Sei $L \subseteq \Sigma^*$ regulär. Es ist entscheidbar, ob $f(w) = g(w)$ für alle $w \in L$ gilt.

Hinweis: Untersuchen Sie die Struktur des DFAs.

Lösung:

(a) Sei $(x_1, y_1), \ldots, (x_k, y_k)$ eine PCP-Instanz.

Setze $\Sigma = \{a_i \mid i = 1, \ldots, k\}$, $L = \Sigma^*$, $f(a_i) := x_i$ und $g(a_i) := y_i$.

Dann gibt es ein $w \in L = \Sigma^*$ mit $f(w) = g(w)$ genau dann, wenn die PCP-Instanz lösbar ist.

(b) Sei A ein oBdA minimaler DFA mit $L = L(A)$ mit n Zuständen.

Sei $D = \{w \in L\colon f(w) \neq g(w)\}$.

Sei $z \in D$ ein kürzestes Wort aus D.

Betrachte den akzeptierenden Ablauf $q_0, q_1, \ldots, q_l$ von A zu z mit $l = |z|$.

- Fall: Alle q_i sind verschieden. Dann gilt $l < n$. Hiervon gibt es nur endlich viele Wörter w, für welche in endlicher Zeit deterministisch $f(w) \stackrel{?}{=} g(w)$ getestet werden kann, womit man z in endlicher Zeit findet.
- Fall: Es gilt $q_i = q_j$ für $i < j$. Entsprechend dem PL für reguläre Sprachen darf man $j \leq n$ annehmen, insbesondere ist j der früheste Zeitpunkt, an dem ein Zustand von A zum zweiten Mal während der Verarbeitung von z besucht wird. Dann unterteilt sich z in $z = uvw$ mit $|u| = i$, $|uv| = j$ und $uw \in L$.

  Da z ein kürzestes Wort aus D ist, muss $f(uw) = g(uw)$ gelten, insbesondere $f(u)f(w) = g(u)g(w)$.

  OBdA gilt $|f(u)| \leq |g(u)|$. Dann muss $f(u)$ Präfix von $g(u)$ sein, d.h. es gibt ein $\gamma \in \Gamma^*$ mit $g(u) = f(u)\gamma$.

  Aus $f(u)f(w) = g(u)g(w)$ folgt $f(u)f(w) = f(u)\gamma g(w)$ und damit $f(w) = \gamma g(w)$ in diesem Fall.

  Hiermit:
  $$f(z) = f(u)f(v)f(w) = f(u)f(v)\gamma g(w) \neq f(u)\gamma g(v)g(w) = g(u)g(v)g(w) = g(z)$$

  Kürzen führt auf
  $$f(v)\gamma \neq \gamma g(v)$$

  Ersetze nun w durch ein kürzestes Wort w', so dass $uw' \in L$ gilt. Da $uw \in L$ kann $\delta(q_0, u)$ nicht der ablehende Zustand sein. In maximal $n-1$ Schritten kann man daher von $\delta(q_0, u)$ zu irgendeinem Endzustand von A kommen. Es folgt $|w'| < n$ und damit $|uvw'| < 2n$. Weiter gilt $|w'| \leq |w|$, also $|uvw'| \leq |z|$. Analog zu oben folgt $f(w') = \gamma g(w')$, da $uw' \in L \wedge |uw'| < |z|$. Somit:

  $$f(uvw') = f(u)f(v)f(w') = f(u)f(v)\gamma g(w') \neq f(u)\gamma g(v)g(w') = g(u)g(v)g(w')$$

  Damit muss bereits $|w| = |w'|$, insbesondere $|z| < 2n$ gegolten haben.

Insgesamt erhält man somit:

$$\exists w \in L(A)\colon f(w) \neq g(w) \text{ } \mathbf{gdw} \text{ } \exists w \in L(A) \cap \Sigma^{<2n}\colon f(w) \neq g(w).$$

was sich in endlicher Zeit überprüfen lässt.