1. Hausaufgaben

Lösung zu H1: Der Beweis verläuft analog zu G1!

Lösung zu H2 (Partitionieren): Wir reduzieren das (nach Aufgabe G8) $\mathcal{NP}$-vollständige Problem ein perfektes 3-dimensionales Matching zu finden auf das Partitionierungsproblem. Wenn uns diese Reduktion in polynomieller Zeit gelingt, ist das Partitionierungsproblem ebenfalls $\mathcal{NP}$-vollständig, da wir ja bereits jedes Problem aus $\mathcal{NP}$ in polynomieller Zeit auf das 3-dimensionale Matching reduzieren können und somit aus der Komposition der Reduktionen eine polynomielle Reduktion auf das Partitionierungsproblem erhalten.

Gegeben sei also eine Instanz des 3-dimensionales Matching Problems, die Benennung sei so wie in Aufgabe G8. Wir werden jedem Element aus $X := U \cup V \cup W$ ein Gewicht aus $\mathbb{N}$ zuordnen um daraus ein Gewicht für jedes Tripel aus E zu berechnen, so daß jedes beliebige perfekte Matching ein einzigartiges Gewicht besitzt:

Es sei $b \in \mathbb{N}$ genau um eins größer als die maximale Anzahl an Tripeln in E, die alle mindestens ein gemeinsames Element aus X besitzen. Des weiteren ordnen wir X in einer beliebigen Reihenfolge $x_0, x_1, \ldots, x_{3k-1}$. Jedem x_i ordnen wir das Gewicht b^i zu und jedem Tripel aus E ordnen wir als Gewicht die Summe der Gewichte seiner drei Elemente zu. Wir berechnen s, die Summe der Gewichte aller Elemente von X; sowie S, die Summe der Gewichte aller Tripel in E.

Wir bemerken das diese Zuordnung umkehrbar ist: Aus der Summe der Gewichte einer Teilmenge F von E können wir ein F' rekonstruieren, welches die gleichen Elemente von X abdeckt. (Man stelle sich dazu die Summe als Zahl zur Basis b vor; ein Stellenübertrag bei der Summierung wurde dabei durch die Wahl von b ja ausgeschlossen.)

Es sei $a := |S - 2s|$. Wir setzen A als die Menge der Gewichte der Tripel in E vereinigt mit $\{a\}$.

Aus einer Partition von A in A_1 und A_2 läßt sich ein perfektes 3-dimensonales Matching konstruieren: Wir unterscheiden nun:

Fall $S - 2s \geq 0$: Es gilt also $\sum A = S + a = 2(S - s)$ Der Wert der Teilmengen beträgt somit genau $S - s$. O.B.d.A. gelte $\{a\} \in A_1$. Das Gewicht von $A_1 \setminus \{a\}$ beträgt $S - s - a = S - s - (S - 2s) = s$, d.h. jedes Gewicht eines Elements aus X ist genau einmal in $A_1 \setminus \{a\}$ enthalten.

Fall $S - 2s < 0$: Es gilt dann $\sum A = S + a = S + (-S + 2s) = 2s$ Der Wert der Teilmengen beträgt somit genau s. O.B.d.A. gelte $\{a\} \in A_2$. Jedes Gewicht eines Elements aus X ist somit genau einmal in A_1 enthalten.

In beiden Fällen erhalten wir also eine Menge von Tripel-Gewichten, deren Summe gleich der Summe der Elemente in X ist. Damit erhalten wir also auch wie oben angegeben ein perfektes Matching.

Umgekehrt erigbt ein perfektes 3-dimensionales Matching auf völlig analoger Weise eine gültige Partition.

Lösung zu H3:

Der Algorithmus ergibt sich folgt:

a) Wir starten mit einer Belegung, welche alle Variablen auf wahr setzt. Ist die Formel erfüllt, so sind wir schon fertig.

b) Ansonsten suchen wir uns eine Klausel, welche nicht erfüllt wird. Die enthält nach Voraussetzung maximal ein negatives Literal. Wenn dem so ist, so setzten wir die Variable dieses Literals in unserer Belegung auf falsch, so daß diese Klausel nun erfüllt wird. Enthält diese Klausel kein negatives Literal, so beenden wir mit der Ausgabe, das die Formel nicht erfüllbar ist.

c) Wir prüfen, ob unsere Belegung die Formel erfüllt. Wenn dem so ist sind wir fertig.

d) Wenn nein, dann prüfen wir ob unsere Belegung noch Variablen auf wahr setzt. Wenn dem so ist, so springen wir zu b).

e) Ansonsten geben wir aus, das die Formel nicht erfüllbar ist.

Laufzeit: Die Schlaufe wird höchstens so oft durchlaufen, wie es Variablen gibt, da wir (bis auf das erste Mal) vor jedem Schleifen durchlauf eine Variable auf falsch setzten, aber nie Variablen wieder auf wahr zurücksetzen. Die Anzahl der Variablen ist sicherlich polynomiell durch die Größe der Formel beschränkt. Damit läßt sich auch sicherlich prüfen, ob noch eine dieser Variablen durch die aktuell betrachtete Belegung auf wahr gesetzt wird.

Das Auswerten der Formel bezüglich einer Belegung läßt sich ebenfalls polynomiell in der Länge der Formel durchführen (jede Klausel muß höchstens einmal betrachtet werden).

Korrektheit: Wir müssen zeigen, dass jede Belegung, welche die Formel erfüllt, die vom Algorithmus in einem Schritt mit Falsch belegte Variable ebenfalls mit Falsch belegen muß. Dies beweisen wir mit Induktion über die Anzahl n der bereits auf Falsch gesetzten Variablen.

Der Algorithmus setzt eine Variable auf falsch, welche negativ in einer unter der momentan betrachteten Belegung unerfüllten Klausel vorkommt. Im Fall $n = 0$ beobachten wir, das alle unerfüllten Klauseln nur aus einem einzigen negativen Literal bestehen können, da jede Klausel ja maximal nur ein negatives Literal enthält und ein einziges vorkommendes positives Literal die Klausel unter der initialen Belegung bereits erfüllt. Offensichtlich muß jede Belegung, welche die Formel erfüllt, alle negtaiven Einerklauseln erfüllen und somit die darin enthaltenen Variablen auf falsch setzen.

Betrachten wir also den Fall $n > 0$. Die ausgewählte Variable kommt nach Konstruktion des Algorithmus und der Horn-Bedingung in einer momentan unerfüllten Klausel als einziges negatives Literal vor. Somit müssen alle anderen Variablen der Klausel bereits mit Falsch belegt worden sein, da ansonsten die Klausel ja erfült wäre. Da aber alle bereits auf falsch gesetzten Variablen zwingend auf falsch gesetzt werden müssen, kann eine gültige Belegung der Formel diese Klausel ebenfalls nur dann erfüllen, wenn die gewählte Variable ebenfalls auf Falsch gesetzt wird.

Lösung zu H4:

Die folgende Lösung wurde von Ihrem Mitstudenten Markus Latte *eingereicht und erscheint hier mit freundlicher Genehmigung und mit ein paar unwesentlichen Anpassungen. Vielen Dank!*

O.b.d.A sei 0 im Bandalphabet Σ. Für jede Sprache $A \subseteq \Sigma^*$ sei

$$L_A := \{0^n | \exists x \in A. |x| = n\}$$

und folglich

$$\overline{L_A} = \{xyz \mid x, z \in \Sigma^* \land y \in \Sigma \setminus \{0\}\} \cup \{0^n \mid \forall x. |x| = n \to x \notin A\}$$

Der erste Teil von $\overline{L_A}$ kann in linearer deterministischer Zeit entschieden werden, ist also für als betrachtete Problem nicht relevant. Im Folgenden beziehe sich $\overline{L_A}$ nur auf den zweiten Teil. Offensichtlich gilt $L_A \in \mathcal{NP}^A$, denn eine nicht deterministische Turing Maschine rät einfach ein Wort der Länge n und überprüft es anschliessend mit dem Orakel für A.

Zeige: Es gibt ein B mit $\overline{L_B} \notin \mathcal{NP}^B$. Gegeben sei eine konstruktive Aufzählung $(T_i)_{i \in \mathbf{N} \setminus \{0\}}$ aller nicht deterministischen, polynominalzeit Orkalmaschinen (siehe Aufgabe G13b). T^X stehe für die Orakelmaschine T mit der Orakelsprache X.

Konstruiere die Folge $(B_i)_{i \in \mathbf{N}}$ simultan mit $(n_i)_{i \in \mathbf{N}}$, $(m_i)_{i \in \mathbf{N}}$. $B_0 := \emptyset$ und $n_0, m_0 := 0$. Sei $i > 0$ und T_i die i-te Orakelmaschine und p_i das Polynom, das die Laufzeit von T_i beschränkt, so definiere

$$m_i := \max\{n_{i-1}, \min\{k \mid |\Sigma|^k > p_i(k)\}\}$$

und

$$B_i := B_{i-1} \cup \begin{cases} \{v_i\} & \text{falls } T_i^{B_{i-1}} \text{ die Eingabe } 0^{m_i} \text{ akzeptiert} \\ \emptyset & \text{sonst} \end{cases}$$

wobei v_i ein Wort mit $|v_i| = m_i$ sei, das bei *einer* akzeptierenden Rechnung von $T_i^{B_{i-1}}$ auf 0^{m_i} nicht an das Orakel gestellt wurde. Setze

$$n_i := |\Sigma|^{m_i} + 1$$

und

$$B := \bigcup_{i \in \mathbf{N}} B_i$$

Es gilt $\overline{L_B} \notin \mathcal{NP}^B$: Angenommen $\overline{L_B} \in \mathcal{NP}^B$, so gäbe es ein i mit $\overline{L_B} = L(T_i^B)$. Betrachte die Eingabe 0^{m_i}.

Fall T_i^B akzeptiert. Also gibt es nach Annahme kein Wort der Länge m_i in B. Somit hat nach Konstruktion $T_i^{B_{i-1}}$ die Eingabe 0^{m_i} verworfen. Damit müßte aber auch T_i^B verwerfen, da für alle $x \in B \triangle B_{i-1}$ die Ungleichung $|x| > p(m_i)$ gilt und somit die Antworten beider Orakel für alle in der erlaubten Rechenzeit stellbaren Orakelfragen übereinstimmen ($B_{i-1} \subseteq B_i$). Widerspruch.

Fall T_i^B lehnt ab. Also gibt es ein $v \in B$ mit $|v| = m_i$. Somit hat $T_i^{B_{i-1}}$ die Eingabe 0^{m_i} akzeptiert. Nach Konstruktion wird v dann aber auch von T^B akzeptiert, da es ja eine Berechnung gibt, bei der v nicht als Frage an das Orakel gestellt wird. Widerspruch.