1. Klausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2017/2018

Lösung!

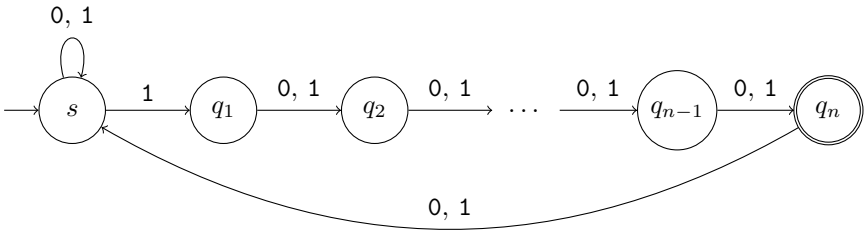
Beachten Sie:

- Bringen Sie den Aufkleber mit Ihrem Namen und Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Es sind keine Hilfsmittel zugelassen.
- You may write in English.

	Mögliche Punkte				Erreichte Punkte			
	a	b	c	Σ	a	b	c	Σ
Aufg. 1	2	5	1	8				
Aufg. 2	2	2	4	8				
Aufg. 3	4	3	3	10				
Aufg. 4	4	5	–	9			–	
Aufg. 5	5	5	–	10			–	
Aufg. 6	1	3	5	9				
Aufg. 7	6×1			6				
Σ				60				

Problem 1: Endliche Automaten $2 + 5 + 1 = 8$ Punkte

Sei für $n = \{1, 2, \ldots\}$ folgender nichtdeterministischer endlicher Automat $\mathcal{A}_n$ mit $n+1$ Zuständen gegeben, der eine Sprache über dem Alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ erkennt.



(a) Geben Sie die Sprache, die $\mathcal{A}_n$ erkennt, in Mengenschreibweise an.

(b) Zeigen Sie, dass der Index der Neroderelation von $L(\mathcal{A}_n)$ mindestens 2^n ist.

(c) Was folgt aus dem Zusammenhang aus Teilaufgabe (b) für die Anzahl an Zuständen eines zu $\mathcal{A}_n$ äquivalenten deterministischen endlichen Automaten?

Lösung:

(a) $L(\mathcal{A}_n) = \{u\mathtt{1}v \mid u, v \in \Sigma^*, |v| = n - 1\}$

(b) Beliebige zwei Wörter $w_1 \neq w_2 \in \Sigma^n$ lassen sich so zerlegen, dass gilt

$$w_1 = u\mathtt{1}v_1 \quad \text{und}$$
$$w_2 = u\mathtt{0}v_2.$$

Dabei müssen ggf. w_1 und w_2 getauscht werden. Es gilt $m = |v_1| = |v_2| \leq n - 1$. Für jedes $x \in \Sigma^{n-m-1}$ gilt dann $w_1 x \in L(\mathcal{A}_n)$ und $w_2 x \notin L(\mathcal{A}_n)$. Damit sind w_1 und w_2 bezüglich der Nerode-Relation nicht äquivalent. Da also alle Wörter der Länge n nicht äquivalent sind ist der Index der Nerode-Relation mindestens 2^n.

(c) Ein äquivalenter deterministischer endlicher Automat muss mindestens 2^n Zustände haben.

Problem 2: Kontextfreie Grammatiken $2 + 2 + 4 = 8$ Punkte

Zu einer Sprache L sei die Spiegelsprache L^{R} definiert als

$$L^{\mathsf{R}} = \left\{ w^{\mathsf{R}} \mid w \in L \right\}.$$

Dabei ist w^{R} die Spiegelung von w, d.h. $(w_1 w_2 \dots w_k)^{\mathsf{R}} = w_k w_{k-1} \dots w_1$.

(a) Sei $G = (\Sigma = \{a, b\}, V = \{S, A, B\}, S, R)$ mit folgender Produktionsmenge R

$$S \to ASB \mid AB$$
$$A \to a$$
$$B \to b.$$

Geben Sie eine Grammatik G^{R} an, die $L(G)^{\mathsf{R}}$ erzeugt.

(b) Sei G eine kontextfreie Grammatik. Beschreiben Sie eine allgemeine Vorgehensweise, um mithilfe von G eine kontextfreie Grammatik G^{R} zu bestimmen, die die Sprache $L(G)^{\mathsf{R}}$ erzeugt.

(c) Beweisen Sie die Korrektheit Ihrer Vorgehensweise aus (b).

Lösung:

(a) Die Grammatik G^{R} entspricht G bis darauf, dass die Produktionsmenge nun folgendermaßen aussieht:

$$S \to BSA \mid BA$$
$$A \to a$$
$$B \to b$$

(b) Ändere jede Produktion $A \to w$ zu $A \to w^{\mathsf{R}}$ um.

(c) Gehe induktiv vor. Trivialerweise gilt $S = S^{\mathsf{R}}$. Sei nun $w = uAv$ eine partielle Ableitung über G. Nach Induktion ist $w^{\mathsf{R}} = v^{\mathsf{R}} A u^{\mathsf{R}}$ eine partielle Ableitung über G^{R}. Sei nun $A \to x$ eine Produktion von G, womit sich $w' = uxv$ ableiten lässt. Dann ist $A \to x^{\mathsf{R}}$ eine Produktion von G^{R}, womit sich ${w'}^{\mathsf{R}} = v^{\mathsf{R}} x^{\mathsf{R}} u^{\mathsf{R}}$ ableiten lässt.

Problem 3: Entscheidbarkeit $4 + 3 + 3 = 10$ Punkte

Aus Vorlesung und Übung wissen Sie, dass die universelle Sprache

$$L_{\mathsf{u}} = \{\langle M, w\rangle \mid M \text{ akzeptiert } w\}$$

zwar semi-entscheidbar, aber nicht entscheidbar ist.

In dieser Aufgabe sollen Sie zeigen, dass weder die Sprache

$$L = \{\langle M\rangle \mid L(M) = \Sigma^*\},$$

noch ihr Komplement L^{c} semi-entscheidbar ist.

(a) Zeigen Sie, dass L^{c} nicht semi-entscheidbar ist, indem Sie L_{u} auf L reduzieren.

Aus einer gegebenen Turingmaschine M zusammen mit einer Eingabe w konstruieren wir folgendermaßen eine neue Turingmaschine N. Bei Eingabe von x simuliert N die Turingmaschine M mit Eingabe w für $|x|$ Schritte. Falls M innerhalb von $|x|$ Schritten akzeptiert, lehnt N die Eingabe x ab. Ansonsten akzeptiert N die Eingabe x.

(b) Zeigen Sie, dass M die Eingabe w genau dann nicht akzeptiert, wenn $L(N) = \Sigma^*$ gilt.

(c) Zeigen Sie, dass L nicht semi-entscheidbar ist, indem Sie $L_{\mathsf{u}}^{\mathsf{c}}$ auf L reduzieren.

Lösung:

(a) Nehme an, dass L durch eine Turingmaschine X entscheidbar ist. Konstruiere zu einer beliebigen Turingmaschine M mit Eingabe w eine Turingmaschine Y folgendermaßen. Akzeptiert M das Wort w so akzeptiert Y jede Eingabe. Andernfalls lehnt Y jede Eingabe ab. Es gilt also $L(Y) = \Sigma^*$ genau dann, wenn M die Eingabe w akzeptiert. Dann gilt $\langle Y\rangle \in L$ genau dann, wenn M die Eingabe w akzeptiert. Dann entscheidet X also L_u, was ein Widerspruch zur Unentscheidbarkeit von L_u ist.

(b) Angenommen, M akzeptiert w nicht, insbesondere also für keine Anzahl an Simulationsschritten $|x|$. Dann akzeptiert N jede Eingabe, es gilt also $L(N) = \Sigma^*$. Angenommen, M akzeptiert w. Sei n die Anzahl an Berechnungsschritten, bis M akzeptiert. Dann wird N die Eingabe σ^n für ein $\sigma \in \Sigma$ ablehnen und es gilt also $L(N) \neq \Sigma^*$.

(c) Konstruiere zu einer Eingabe $\langle M, w\rangle$ die Turingmaschine N wie oben beschrieben. Aus Teil b ist bekannt, dass $L(N) = \Sigma^*$ genau dann gilt, wenn M die Eingabe w nicht akzeptiert. Also gilt $\langle N\rangle \in L$ genau dann, wenn $\langle M, w\rangle \in L_u^{\mathsf{c}}$.

Problem 4: Kellerautomaten 4 + 5 = 9 Punkte

Sei

$$L = \{\alpha\#\beta \mid \alpha \text{ ist Teilwort von } \beta\}.$$

(a) Zeigen Sie, dass L nicht kontextfrei ist.

Das Berechnungsmodell nichtdeterministischer Kellerautomat (PDA) lässt sich so ändern, dass sich der Lesekopf, der bei PDAs stets einen Schritt nach rechts geht, nun in zwei Richtungen bewegen kann, nämlich jeweils einen Schritt nach links oder nach rechts. Der Lesekopf kann außerdem das linke und rechte Ende des Eingabeworts erkennen. Ein LRPDA akzeptiert durch akzeptierende Zustände (und nicht durch leeren Stack). Ansonsten bleibt das Berechnungsmodell unverändert. Bezeichne solche Kellerautomaten als LRPDA.

(b) Beschreiben Sie das Verhalten eines LRPDAs, der L erkennt. Gehen Sie insbesondere darauf ein, wie Sie die zusätzlichen Kopfbewegungen und den Nichtdeterminismus ausnutzen.

Lösung:

(a) Verwende das Pumping-Lemma für kontextfreie Sprachen. Sei n die vermeintliche Konstante aus dem Pumping-Lemma. Wähle das Wort $z = a^n b^n \# a^n b^n \in L$ und betrachte eine beliebige Zerlegung $z = uvwxy$.

Ist das Trennsymbol # in v oder x so gilt $uv^0wx^0y \notin L$. Befindet sich vwx komplett vor dem Trennsymbol # so gilt $uv^2wx^2y \notin L$, da α echt länger als β ist. Befindet sich vwx komplett nach dem Trennsymbol # so gilt $uv^0wx^0y \notin L$, da β echt kürzer als α ist. Ansonsten überspannt vwx das Trennsymbol. Wegen $|vwx| \leq n$ besteht dann v ausschließlich aus dem Zeichen b und x besteht ausschließlich aus dem Zeichen a. Dann gilt $uv^2wx^2y \notin L$, weil α das Zeichen b echt öfter enthält als β.

(b) Der LRPDA $\mathcal{A}$ geht folgendermaßen vor. Zunächst wird α komplett gelesen und in umgekehrter Reihenfolge auf den Stack gelegt. Dann durchläuft $\mathcal{A}$ das Wort β von links nach rechts. Ist das aktuelle Symbol gleich dem obersten Symbol auf dem Stack ist diese Position ein Kandidat für das Ende von α in β. Der LRPDA $\mathcal{A}$ entscheidet sich nichtdeterministisch, ob verifiziert werden soll, ob die Position tatsächlich das Ende von α in β markiert. Falls nein läuft $\mathcal{A}$ weiter nach rechts auf der Suche nach dem nächsten passenden Kandidat. Falls ja beginnt die Verifikation. Dazu entfernt $\mathcal{A}$ das oberste Symbol vom Stack und läuft einen Schritt nach *links*. Dort wird wieder das oberste Symbol auf dem Stack mit dem aktuellen Symbol verglichen. Sind die Symbole nicht gleich so lehnt $\mathcal{A}$ die Eingabe ab. Andernfalls wiederholt sich der Prozess so lange, bis der Stack leer ist. Ist dies der Fall akzeptiert $\mathcal{A}$ die Eingabe.

Problem 5: NP-Vollständigkeit 5 + 5 = 10 Punkte

Das Entscheidungsproblem DEGREE-RESTRICTED SPANNING TREE ist wie folgt definiert.

Input: Graph $G = (V, E)$
Zahl $k \in \mathbb{N}$
Lösung: aufspannender Baum $T \subseteq G$ mit Maximalgrad höchstens k,
d.h. T ist zusammenhängend, kreisfrei und enthält alle Knoten von G
und jeder Knoten ist in höchstens k Kanten von T enthalten

Für eine Zahl $k \in \mathbb{N}$ ist das Entscheidungsproblem DEGREE-k-RESTRICTED SPANNING TREE wie folgt definiert.

Input: Graph $G = (V, E)$
Lösung: aufspannender Baum $T \subseteq G$ mit Maximalgrad höchstens k,
d.h. T ist zusammenhängend, kreisfrei und enthält alle Knoten von G
und jeder Knoten ist in höchstens k Kanten von T enthalten

(a) Ist das Problem DEGREE-RESTRICTED SPANNING TREE NP-vollständig? Beweisen Sie!
Hinweis: Betrachten Sie den Fall $k = 2$.

Lösung: Ja, DEGREE-RESTRICTED SPANNING TREE ist NP-vollständig. Wir können eine Lösung, d.h. eine Kantenmenge $F \subseteq E$ raten und in polynomieller Zeit überprüfen, ob $T = (V, F)$ ein Baum ist und jeder Knoten in V höchstens k inzidente Kanten hat.

Um zu zeigen, dass DEGREE-RESTRICTED SPANNING TREE NP-schwer ist, reduzieren wir das NP-vollständige Problem HAMILTONPFAD auf DEGREE-RESTRICTED SPANNING TREE.

Sei $G = (V, E)$ eine Instanz von HAMILTONPFAD. Beobachte, dass ein Hamiltonpfad in G ein aufspannender Baum mit Maximalgrad 2 ist. Außerdem ist umgekehrt jeder aufspannende Baum in G mit Maximalgrad 2 ein Hamiltonpfad in G. Wir definieren also eine Instanz G', k von DEGREE-RESTRICTED SPANNING TREE durch $G' = G$ und $k = 2$. Dann ist HAMILTONPFAD auf G genau dann lösbar wenn DEGREE-RESTRICTED SPANNING TREE auf G', k lösbar ist.

Da HAMILTONPFAD NP-schwer ist, folgt, dass DEGREE-RESTRICTED SPANNING TREE auch NP-schwer ist. Mit DEGREE-RESTRICTED SPANNING TREE in NP, ist also DEGREE-RESTRICTED SPANNING TREE NP-vollständig.

(b) Für welche $k \in \mathbb{N}$ ist ob das Problem DEGREE-k-RESTRICTED SPANNING TREE NP-vollständig? Beweisen Sie!
Hinweis: Für geeignete Zahlen k können Sie DEGREE-2-RESTRICTED SPANNING TREE *auf* DEGREE-k-RESTRICTED SPANNING TREE *reduzieren.*

Lösung: Für $k = 1$ ist DEGREE-k-RESTRICTED SPANNING TREE polynomiell lösbar, also nicht NP-vollständig, solange P $\neq$ NP. Dazu reicht es zu sehen, dass ein Graph G genau dann eine Ja-Instanz von DEGREE-1-RESTRICTED SPANNING TREE ist, wenn G aus genau zwei Knoten und einer Kante besteht, was effizient überprüft werden kann.

Wir behaupten nun, dass für jedes feste $k \geq 2$ das Problem DEGREE-k-RESTRICTED SPANNING TREE NP-vollständig ist. Erstmal ist wieder klar, dass eine Lösung effizient geraten und verifiziert werden kann. DEGREE-k-RESTRICTED SPANNING TREE ist also in NP.

Für $k = 2$ folgt die NP-Schwere aus unserer Reduktion aus Aufgabenteil (a), weil diese Reduktion stets Instanzen mit $k = 2$ benutzt. Für $k \geq 3$ reduzieren wir DEGREE-2-RESTRICTED SPANNING TREE auf DEGREE-k-RESTRICTED SPANNING TREE. Sei $G = (V, E)$ eine Instanz von DEGREE-2-RESTRICTED SPANNING TREE. Wir definieren einen neuen Graphen G' auf Basis von G durch das Anhängen von $k - 2$ neuen Knoten vom Grad 1 an jeden Knoten aus G. Die Größe von G' ist höchstens das k-fache der Größe von G und G' kann effizient berechnet werden. (Beachte: k ist eine Konstante!)

Nun ist G' eine Ja-Instanz von DEGREE-k-RESTRICTED SPANNING TREE genau dann wenn G eine Ja-Instanz von DEGREE-2-RESTRICTED SPANNING TREE ist. Beobachte dazu, dass

- alle Kanten in $G' - G$ in jedem aufspannenden Baum von G' enthalten sind,
- jeder aufspannende Baum T' in G' eingeschraenkt auf die Knoten in G ein aufspannenden Baum T in G ergibt und der Maximalgrad von T genau der Maximalgrad von T' minus $(k-2)$ ist, und
- jeder aufspannende Baum T in G mit Maximalgrad 2 durch das Hinzufuegen der Kanten in $G' - G$ zu einem aufspannenden Baum T' in G' von Maximalgrad $2 + (k - 2) = k$ wird.

Es folgt, dass G' genau dann eine Ja-Instanz von DEGREE-k-RESTRICTED SPANNING TREE ist, wenn G eine Ja-Instanz von DEGREE-2-RESTRICTED SPANNING TREE ist.

Da DEGREE-k-RESTRICTED SPANNING TREE in NP liegt und DEGREE-2-RESTRICTED SPANNING TREE NP-schwer ist, haben wir dass DEGREE-k-RESTRICTED SPANNING TREE NP-vollständig ist, für jedes $k \geq 2$.

Jedes der folgenden Entscheidungsprobleme ist NP-vollständig. Sie können diese Probleme für Ihre Reduktionen nutzen.

TRAVELING SALESMAN

Input: Graph $G = (V, E)$
Längenfunktion $\ell : E \to \mathbb{R}$
Zahl $k \in \mathbb{N}$
Lösung: geschlossene Tour T durch alle Knoten von G mit Gesamtlänge höchstens k,
d.h. $\sum_{e \in T} \ell(e) \leq k$.

HAMILTONPFAD

Input: Graph $G = (V, E)$
Lösung: Pfad P durch alle Knoten von G

CLIQUE

Input: Graph $G = (V, E)$
Zahl $k \in \mathbb{N}$
Lösung: Menge $U \subseteq V$ von mindestens k Knoten in der je zwei Knoten benachbart sind,
d.h. $|U| \geq k$ und für alle $u, v \in U$ mit $u \neq v$ gilt $\{u, v\} \in E$.

COLOR

Input: Graph $G = (V, E)$
Zahl $k \in \mathbb{N}$
Lösung: Knotenfärbung mit höchstens k Farben in der keine gleichfarbigen Knoten benachbart sind,
d.h. $V = V_1 \cup \cdots \cup V_k$ und für $i = 1, \ldots, k$ und alle $u, v \in V_i$ gilt $\{u, v\} \notin E$.

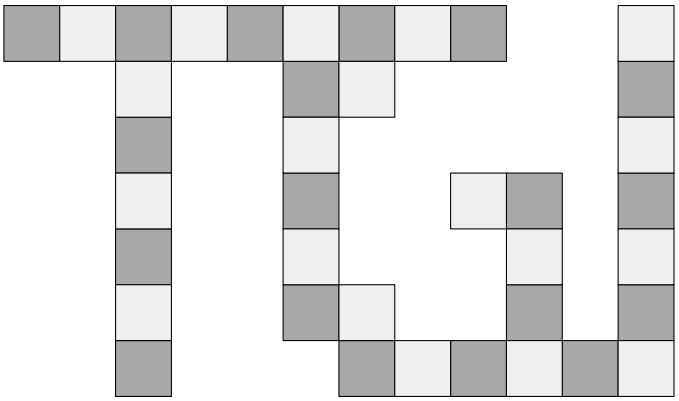
Problem 6: Approximationsalgorithmen $1 + 3 + 5 = 9$ Punkte

Das Optimierungsproblem NON-ATTACKING ROOK PLACEMENT ist wie folgt definiert.

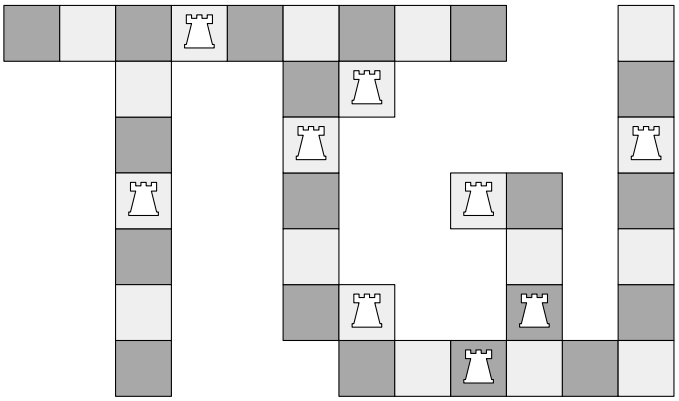
Input: endlicher Ausschnitt S des unendlichen Schachbretts in der Ebene
Lösung: Menge T von Türmen auf S die sich gegenseitig nicht bedrohen, d.h.
- jeder Turm in T steht auf genau einem Feld in S,
- auf jedem Feld in S steht höchstens ein Turm in T,
- kein Turm in T kann einen anderen Turm in T erreichen durch eine nur-horizontale oder nur-vertikale Bewegung entlang der Felder in S

Ziel: maximiere die Anzahl $k = |T|$ der Türme in T

(a) Zeichnen Sie in die folgende Instanz S des NON-ATTACKING ROOK PLACEMENT Problems eine Lösung T mit Wert $|T| = 9$ ein.



Lösung:



(b) Zeigen Sie, dass für jede Lösung T des NON-ATTACKING ROOK PLACEMENT Problems jedes Feld von S von höchstens zwei Türmen in T bedroht wird.

Lösung: Sei f ein Feld in S. Wenn zwei verschiedene Türme t_1, t_2 in T das Feld f durch eine nur-horizontale Bewegung erreichen können, dann kann auch Turm t_1 den Turm t_2 durch eine nur-horizontale Bewegung erreichen, was im Widerspruch dazu steht, dass T eine zulässige Lösung ist. Also kann höchstens ein Turm in T das Feld f durch eine nur-horizontale Bewegung erreichen.

Analog kann höchstens ein Turm in T das Feld f durch eine nur-vertikale Bewegung erreichen. Da dies alle Möglichkeiten sind, das Feld f zu bedrohen, gibt es insgesamt höchstens zwei Türme in T die f bedrohen.

(c) Zeigen Sie, dass der folgende Greedy-Algorithmus $\mathcal{A}$ ein polynomialer Approximationsalgorithmus für NON-ATTACKING ROOK PLACEMENT mit einer relativen Gütegarantie von 2 ist:

- Solange es ein Feld f in S gibt, dass von keinem bereits platzierten Turm bedroht wird, platziere einen Turm auf f.

Lösung: Sei $T_{\mathcal{A}}$ eine Lösung die von $\mathcal{A}$ berechnet wurde und T_{OPT} eine optimale Lösung für die Instanz S des NON-ATTACKING ROOK PLACEMENT Problems. Jeder Turm t in T_{OPT} wird von mindestens einem Turm in $T_{\mathcal{A}}$ bedroht, da andernfalls das Feld f auf dem t steht nicht bedroht wird und der Algorithmus $\mathcal{A}$ nicht fertig wäre. Bezeichne $b(t)$ einen (beliebigen) Turm in $T_{\mathcal{A}}$ der den Turm t in T_{OPT} bedroht.

Beobachte, dass Turm t aus T_{OPT} das Feld auf dem $b(t)$ steht bedroht. Nach Aufgabenteil (b) bedrohen höchstens zwei Türme in T_{OPT} das gleiche Feld und damit den gleichen Turm in $T_{\mathcal{A}}$. Zu jedem Turm in $T_{\mathcal{A}}$ können wir so also ein oder zwei Türme in T_{OPT} zuordnen, wobei jeder Turm in T_{OPT} getroffen wird.

Also gilt $|T_{\text{OPT}}| \leq 2 \cdot |T_{\mathcal{A}}|$ und damit hat $\mathcal{A}$ eine relative Gütegarantie von 2. Weiterhin ist die Laufzeit von $\mathcal{A}$ polynomiell in der Größe der Eingabe S, da das überprüfen der Schleifenbedingung und das Durchführen eines Schleifendurchlaufs in linearer Zeit in $|S|$ möglich ist, und insgesamt höchstens $|S|$ Durchläufe gemacht werden (Jeder Durchlauf platziert einen neuen Turm).

Problem 7: Gemischtes 6 Punkte

Sind die folgenden Aussagen korrekt? Begründen Sie jeweils kurz.

(a) Jede kontextfreie Sprache liegt in P.

Lösung: **Richtig.**

Der CYK-Algorithmus löst das Wortproblem in polynomieller Zeit.

(b) Eine Grammatik mit Startsymbol S erzeugt genau dann das leere Wort, wenn $S \to \varepsilon$ eine Produktion ist.

Lösung: **Falsch.**

Ein Gegenbeispiel erhält man mit den Produktionen $S \to T$ und $T \to \varepsilon$.

(c) Wenn $\mathsf{P} = \mathsf{NP}$, dann sind alle Sprachen über dem Alphabet Σ in NP, bis auf Σ^* und die leere Sprache, auch NP-schwer.

Lösung: **Richtig.**

Um L_1 auf L_2 polynomiell zu reduzieren, reicht es L_1 mit einem polynomiellen Algorithmus (der existiert, da $L_1 \in P$) zu entscheiden. Also kann alles in NP auf L_2 reduziert werden und L_2 ist NP-schwer.

(d) Jede kontextfreie Grammatik die nicht das leere Wort erzeugt kann in polynomieller Zeit in Greibach-Normalform umgewandelt werden.

Lösung: **Richtig.**

Das Verfahren aus der Vorlesung ist deterministisch und polynomiell.

(e) Kann ein Problem Π in Polynomialzeit entschieden werden, wenn die Eingabe unär kodiert ist, so ist Π nicht stark NP-vollständig, solange $\mathsf{P} \neq \mathsf{NP}$.

Lösung: **Richtig.**

Ein Problem ist stark NP-vollständig wenn es bei unärer Kodierung NP-vollständig bleibt.

(f) Der vom Huffman-Algorithmus berechnete Codierungsbaum ist eindeutig.

Lösung: **Falsch.**

Wir haben dazu Beispiele auf den Übungsblättern gesehen.