Karlsruher Institut für Technologie

Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Nachklausur zur Vorlesung
Theoretische Grundlagen der Informatik
Wintersemester 2011/2012

Hier Aufkleber mit Name und Matrikelnr. anbringen

Vorname: _____

Nachname: _____

Matrikelnummer: _____

Beachten Sie:

- Bringen Sie den Aufkleber mit Ihrem Namen und Ihrer Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Ihrer Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Zum Bestehen der Klausur sind **20** der möglichen **60** Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.

Aufgabe	Mögliche Punkte					Erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
1	1	3	-	-	4			-	-	
2	1	4	1	-	6				-	
3	2	4	-	-	6			-	-	
4	1	4	-	-	5			-	-	
5	4	-	-	-	4		-	-	-	
6	2	1	1	-	4				-	
7	1	1	1	1	4					
8	2	1	3	3	9					
9	2	3	3	-	8				-	
10	10x1				10					
Σ					60					

Aufgabe 1: (1 + 3 = 4 Punkte)

Gegeben sei die Grammatik $G = (\Sigma, V, S, R)$ mit Terminalen $\Sigma = \{a, b, e, f\}$, Nichtterminalen $V = \{S, A, B, E, F\}$, Startsymbol S und Produktionen

$$\begin{aligned} R = \{S &\rightarrow EB \\ A &\rightarrow EF \mid a \mid f \\ B &\rightarrow AE \mid b \\ E &\rightarrow FA \mid b \mid e \\ F &\rightarrow AB \mid f\}. \end{aligned}$$

(a) Welchen maximalen Chomsky-Typ hat G? Begründen Sie Ihre Antwort.

Lösung:
G ist maximal vom Chomsky-Typ 2, da G in Chomsky-Normalform ist. Damit ist G insbesondere nicht vom Typ 3; jede der Regeln widerspricht dem Typ 3.

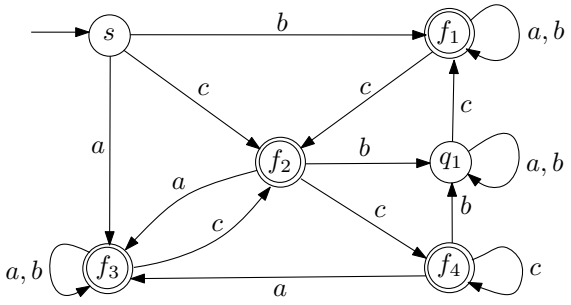
(b) Überprüfen Sie mit dem Cocke-Younger-Kasami Algorithmus, ob das Wort $fabfe$ in der Sprache $L(G)$ enthalten ist. Geben Sie dazu alle Zwischenergebnisse an.

Lösung:
Das Wort $fabfe$ ist in $L(G)$, da nach Anwendung des CYK-Algorithmus das Startsymbol S in der Spitze der Pyramide steht.

		S		
	B, E		F	
S, F		E		S, B
E	B, F		A	B
A, F	A	B, E	A, F	E
f	a	b	f	e

Aufgabe 2: (1 + 4 + 1 = 6 Punkte)

Gegeben sei folgender endlicher Automat $\mathcal{A}$ über dem Alphabet $\Sigma = \{a, b, c\}$.

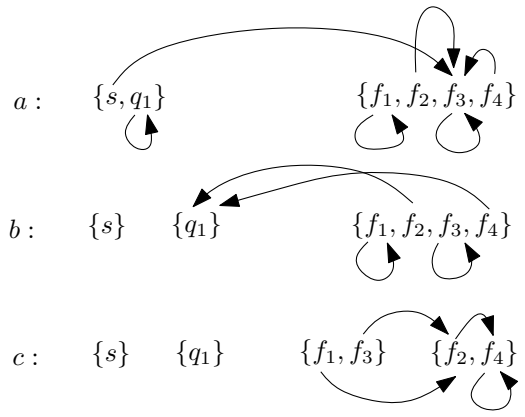


(a) Ist $\mathcal{A}$ determinisitisch? Begründen Sie Ihre Antwort.

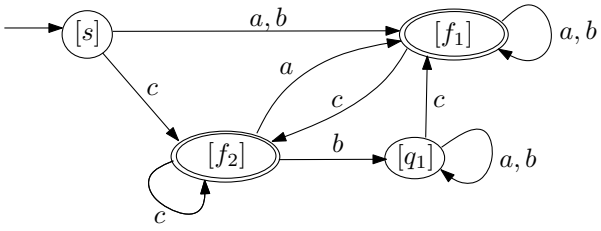
Lösung:
Ja, denn für jedes Paar $(q, x) \in Q \times \Sigma$ gibt es genau einen Übergang in genau einen Zustand in Q.

(b) Konstruieren Sie den Minimalautomaten zu $\mathcal{A}$. Geben Sie die daraus folgenden Äquivalenzklassen der Zustände in $\mathcal{A}$ explizit an. Geben Sie das Ergebnis als Zustandsübergangsdiagramm an und bezeichnen Sie die Zustände mit den Äquivalenzklassen, die sie repräsentieren.

Lösung:



$[s] := \{s\}$, $[q_1] := \{q_1\}$, $[f_1] := \{f_1, f_3\}$, $[f_2] := \{f_2, f_4\}$



(c) Geben Sie für jede der folgenden Aussagen an, ob diese gilt.

- $[a(a^+ \cup c^* \cup b^*a)]^+ \subseteq L(\mathcal{A})$

  Lösung:
  Ja, denn keines der Worte ermöglicht den Übergang nach q_1, da cb nicht als Teilwort vorkommt.
- $[c(c^*b^* \cup a^+)]^+ \subseteq L(\mathcal{A})$

  Lösung:
  Nein, denn $cb \notin L(\mathcal{A})$.

Aufgabe 3: (2 + 4 = 6 Punkte)

(a) Geben Sie eine kontextfreie Grammatik $G = (\Sigma = \{a, b\}, V, S, R)$ an,
die die Sprache $L := \{a^n b^{2m} a^{m-n} \mid m \geq n \text{ und } n, m \in \mathbb{N}_0\}$ erzeugt. Verwenden Sie **maximal vier** Variablen, d.h. $|V| \leq 4$, und **maximal zehn** Regeln, d.h. $|R| \leq 10$.

Lösung:
$G = (\Sigma = \{a, b\}, V = \{S, X, Y\}, S, R)$

$$\begin{aligned} R = \{S &\rightarrow XY \\ X &\rightarrow aXbb \mid \varepsilon \\ Y &\rightarrow bbYa \mid \varepsilon\}. \end{aligned}$$

Erklärung: L lässt sich auch schreiben als $L := \{a^n b^{2(n+k)} a^k \mid k \geq 0 \text{ und } k \in \mathbb{N}_0\}$. Jedes a links oder rechts muss also 2 b's schreiben.

(b) Zeigen Sie, dass die Sprache L aus Teilaufgabe (a) maximal vom Chomsky-Typ 2 ist,
d.h., dass L nicht regulär ist.

Lösung:
Betrachte $w = a^n b^n \in L$. Dann ist $v = a^k$ mit $k \leq n$, wegen $|uv| \leq n$ und $v \neq \varepsilon$. Damit ist $w' = uv^i x = a^{n-k} b^n \notin L$ für $i = 0$.

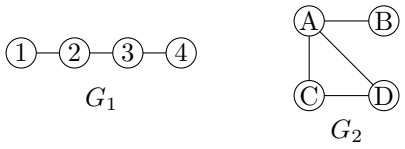
Annahme: L ist regulär, d.h., das Pumpinglemma für reguläre Sprachen gilt für L. Sei also n die untere Schranke der Wortlänge aus dem Pumpinglemma. Betrachte das Wort $w = a^n b^{2n} \in L$ (wähle m = n) mit $|w| \geq n$. Dann gilt für jede Zerlegung $w = uvx$ mit $|uv| \leq n$ und $v \neq \varepsilon$, dass $v = a^k$, $k \geq 1$. Damit ist $uv^i x = a^{n-k} b^{2n} \notin L$ für $i = 0$ (da $n - (n - k) = k \neq 0$), im Widerspruch zum Pumpinglemma.

Aufgabe 4: (1 + 4 = 5 Punkte)

Seien $G = (V_G, E_G)$ und $H = (V_H, E_H)$ ungerichtete Graphen mit $|V_G| = |V_H|$. Ein bijektiver Graphhomomorphismus von G nach H ist eine bijektive Funktion $f : V_G \to V_H$, so dass gilt:

$$\{u, v\} \in E_G \Rightarrow \{f(u), f(v)\} \in E_H$$

(a) Geben Sie einen bijektiven Graphhomomorphismus f von G_1 nach G_2 an. Geben Sie dazu explizit für jeden Knoten aus G_1 an, auf welchen Knoten aus G_2 er abgebildet wird.



G_1 G_2

Lösung:

$$f(1) = B \quad f(2) = A$$
$$f(3) = C \quad f(4) = D$$

(b) **Problem** BIJHOM
Gegeben: Ungerichtete Graphen $G = (V_G, E_G)$ und $H = (V_H, E_H)$
Frage: Gibt es einen bijektiven Graphhomomorphismus von G nach H?

Zeigen Sie, dass das Problem BIJHOM $\mathcal{NP}$-vollständig ist.
Sie dürfen dazu benutzen, dass das Problem CLIQUE $\mathcal{NP}$-vollständig ist:

Problem CLIQUE
Gegeben: Ungerichteter Graph $G = (V, E)$ und ein Parameter $k \leq |V|$
Frage: Gibt es in G eine Clique der Größe mindestens k?

Lösung:
Problem BIJHOM liegt in $\mathcal{NP}$, denn für eine gegebene Funktion kann in Polynomialzeit in der Größe des Graphen entschieden werden, ob sie bijektiv ist und ob sie die Graphhomomorphismusbedingung erfüllt.
Wir zeigen: CLIQUE $\propto$ BIJHOM. Gegeben sei eine CLIQUE-Instanz $I = (H = (V_H, E_H), k)$. Wir konstruieren daraus eine BIJHOM-Instanz $I' = (G, H)$, in der G aus einer Clique mit k Knoten und $|V_H| - k$ isolierten Knoten besteht. Diese Transformation ist offensichtlich polynomiell in $|V_H| + |E_H|$.
Zu zeigen: I' ist genau dann lösbar, wenn H eine Clique der Größe k enthält.

- $\Rightarrow$ Sei I' lösbar und f ein bijektiver Graphhomomorphismus von G nach H. Sei C das Bild der Clique in G. Aus der Homomorphismuseigenschaft und der Bijektivität folgt, dass C eine Clique der Größe k in H ist
- $\Leftarrow$ Sei C eine Clique der Größe k in H. Sei f eine beliebige bijektive Funktion, die die Knoten der Clique in G auf die Knoten in C abbildet und alle anderen Knoten in G beliebig auf die restlichen Knoten in H. Dann erfüllt f die Homomorphismuseigenschaft.

Aufgabe 5: (4 Punkte)

Zeigen Sie, dass das Post'sche Korrespondenzproblem semi-entscheidbar ist.

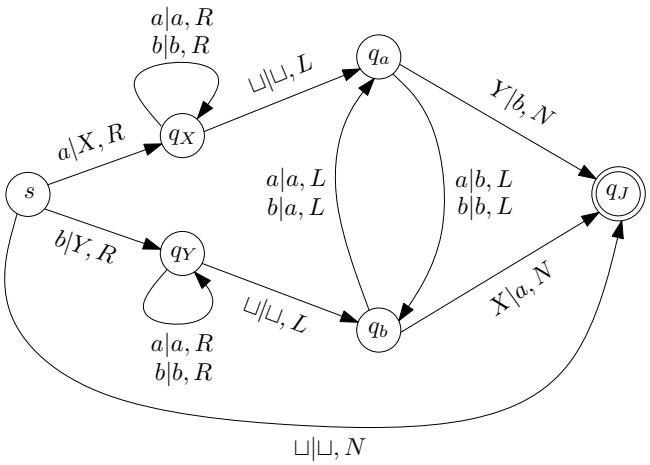
POST'SCHES KORRESPONDENZPROBLEM

Gegeben:	Endliche Folge von Wortpaaren $K = ((x_1, y_1), \ldots, (x_n, y_n))$ über einem endlichen Alphabet Σ, mit $x_i \neq \varepsilon$ und $y_i \neq \varepsilon$ für alle $i \in \{1, \ldots n\}$.
Frage:	Gibt es eine endliche Folge von Indizes $i_1, \ldots, i_k \in \{1, \ldots, n\}$, so dass $x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$ gilt?

Lösung:
Konstruiere eine Turingmaschine, die die Sprache aller lösbaren Instanzen des Post'schen Korrespondenzproblems akzeptiert (aber nicht unbedingt auf alle Eingaben hält): Betrachte nach und nach systematisch alle möglichen Indexfolgen. Zuerst alle der Länge 1, dann die der Länge 2 usw. Überprüfe jeweils, ob eine solche Indexfolge eine gültige Lösung ist, durch Konstruktion und Vergleich der beiden Wörter. Wenn eine gültige Lösung gefunden wird hält die Turingmaschine und akzeptiert die Instanz.
Falls eine Instanz lösbar ist, so gibt es eine Indexfolge, die zwei gleiche Wörter induziert. Diese Indexfolge wird dann auch gefunden, und die Turingmaschine hält und akzeptiert die Instanz. Falls eine Instanz nicht lösbar ist, hält die Maschine nie (und akzeptiert also auch nicht). Damit ist das Post'sche Korrespondenzproblem semientscheidbar.

Aufgabe 6: (2 + 1 + 1 = 4 Punkte)

Gegeben sei folgende deterministische Turingmaschine $\mathcal{M} = (Q = \{s, q_X, q_Y, q_a, q_b, q_J\}, \Sigma = \{a, b\}, \Gamma = \Sigma \cup \{\sqcup, X, Y, c\}, \delta, s, F = \{q_J\})$. Die Übergangsfunktion δ von $\mathcal{M}$ ist durch untenstehendes Zustandsübergangsdiagramm dargestellt.



Für $\mathcal{M}$ gelten folgende Konventionen: $\mathcal{M}$ hält und akzeptiert sobald ein Endzustand erreicht wird. Alle Übergänge, die aus Endzuständen heraus führen, werden demnach nie benutzt. $\mathcal{M}$ hält und **akzeptiert nicht** für nicht dargestellte Zustandsübergänge.

(a) Dokumentieren Sie die Berechnung der Worte *abab*, *baaa* und *aba* durch die angegebene Turingmaschine $\mathcal{M}$. Geben Sie dazu für jeden Schritt die aktuelle **Konfiguration** an.
Geben Sie außerdem die **Sprache** $L(\mathcal{M}) \subseteq \Sigma^*$ an, die $\mathcal{M}$ akzeptiert.

Lösung:
$abab \in L$:
$\sqcup(s)abab$, $\sqcup X(q_X)bab$, $\sqcup Xb(q_X)ab$, $\sqcup Xba(q_X)b$, $\sqcup Xbab(q_X)\sqcup$, $\sqcup Xba(q_a)b\sqcup$,
$\sqcup Xb(q_b)ab$, $\sqcup X(q_a)bab$, $\sqcup(q_b)Xbab$, $\sqcup(q_J)abab$

$baaa \in L$:
$\sqcup(s)baaa$, $\sqcup Y(q_Y)aaa$, $\sqcup Ya(q_Y)aa$, $\sqcup Yaa(q_Y)a$, $\sqcup Yaaa(q_Y)\sqcup$, $\sqcup Yaa(q_b)a\sqcup$,
$\sqcup Ya(q_a)aa$, $\sqcup Y(q_b)aba$, $\sqcup(q_a)Yaba$, $\sqcup(q_J)baba$

$aba \notin L$:
$\sqcup(s)aba$, $\sqcup X(q_X)ba$, $\sqcup Xb(q_X)a$, $\sqcup Xba(q_X)\sqcup$, $\sqcup Xb(q_a)a\sqcup$,
$\sqcup X(q_b)bb$, $\sqcup(q_a)Xab$ Übergang nicht definiert.

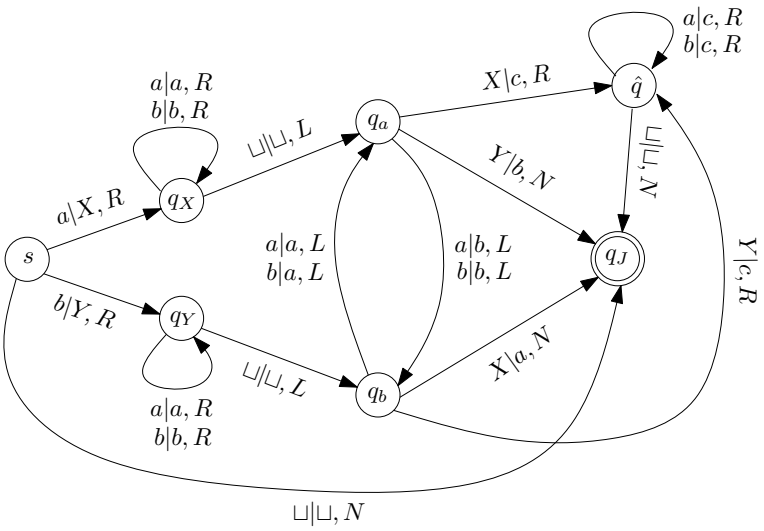
$L(\mathcal{M}) = \{w \in \Sigma^* \mid \exists k \in \mathbb{N}_0 : |w| = 2k\}$

(b) Geben Sie die Funktion $f_{\mathcal{M}} : L(\mathcal{M}) \longrightarrow \Gamma^*$ an, die $\mathcal{M}$ auf $L(\mathcal{M})$ realisiert.

Lösung:
$f_{\mathcal{M}}(w) := (ab)^{k/2}$, falls w mit a beginnt und $k := |w|$ gerade.
$f_{\mathcal{M}}(w) := (ba)^{k/2}$, falls w mit b beginnt und $k := |w|$ gerade.

(c) Erweitern Sie $\mathcal{M}$ so, dass $\mathcal{M}$ alle Worte aus Σ^* akzeptiert und nach Verarbeitung eines Wortes $w \in \Sigma^* \setminus L(\mathcal{M})$ gerade das Wort $w' := c^k$ auf dem Band steht. Dabei sei $k := |w|$ die Anzahl der Zeichen in w. Benutzen Sie für diese Erweiterung **maximal zwei** zusätzliche Zustände. Sie dürfen die Erweiterung in die vorige Abbildung zeichnen.

Lösung:


$a|c, R$
$b|c, R$
$\hat{q}$
$X|c, R$
$a|a, R$
$b|b, R$
q_a
$\sqcup|\sqcup, L$
$Y|b, N$
$\sqcup|\sqcup, N$
$a|X, R$
q_X
$a|a, L$
$b|a, L$
$a|b, L$
$b|b, L$
q_J
$Y|c, R$
s
$b|Y, R$
q_Y
$\sqcup|\sqcup, L$
$X|a, N$
$a|a, R$
$b|b, R$
q_b
$\sqcup|\sqcup, N$

Aufgabe 7: (1 + 1 + 1 + 1 = 4 Punkte)

Sei $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, Z, \delta, \{q_1\})$ der Kellerautomat mit Zustandsmenge $Q = \{q_0, q_1, q_2\}$, Eingabealphabet $\Sigma = \{0, 1, 2\}$, Stack-Alphabet $\Gamma = \{X, Z\}$, Anfangszustand q_0, Stack-Initialisierung Z, einzigem Endzustand q_1 und der folgenden Übergangsfunktion δ:

$$\delta(q_0, 0, Z) = \{(q_0, Z), (q_1, Z)\}, \quad \delta(q_1, 1, Z) = \{(q_1, X)\}, \quad \delta(q_1, 1, X) = \{(q_1, XX)\},$$
$$\delta(q_1, 2, X) = \{(q_2, \varepsilon)\}, \quad \delta(q_2, 2, X) = \{(q_2, \varepsilon)\}$$

(a) Ist $\mathcal{A}$ determininistisch? Begründen Sie Ihre Antwort.

Lösung:
Nein, $\mathcal{A}$ ist nicht deterministisch, da $|\delta(q_0, 0, Z)| = 2 > 1$.

(b) Dokumentieren Sie eine durch Endzustand akzeptierende Berechnung des Wortes 00111. Geben Sie dazu für jeden Schritt die aktuelle Konfiguration an.

Lösung:

- $(q_0, 00111, Z)$
- $(q_0, 0111, Z)$
- $(q_1, 111, Z)$
- $(q_1, 11, X)$
- $(q_1, 1, XX)$
- (q_1, ε, XXX)

(c) Welche Sprache akzeptiert $\mathcal{A}$ durch akzeptierenden Endzustand?

Lösung:
$L = \{0^i 1^j \mid i, j \in \mathbb{N}_0, i > 0\}$

(d) Welche Sprache akzeptiert $\mathcal{A}$ durch leeren Stack?

Lösung:
$L = \{0^i 1^j 2^k \mid i, j, k \in \mathbb{N}_0, i > 0, j = k > 0\}$

Aufgabe 8: (2 + 1 + 3 + 3 = 9 Punkte)

Gegegeben sei eine endliche Menge $\mathcal{U}$ von booleschen Variablen und eine endliche, nicht leere Menge $\mathcal{C}$ von Klauseln über $\mathcal{U}$. Eine Klausel ist ein boolescher Ausdruck der Form $x_1 \vee \cdots \vee x_k$ mit $x_i \in \{u \mid u \in \mathcal{U}\} \cup \{\overline{u} \mid u \in \mathcal{U}\}$. Das Optimierungsproblem MAXSAT besteht darin, eine Wahrheitsbelegung für $\mathcal{U}$ zu finden, so dass möglichst viele Klauseln aus $\mathcal{C}$ den Wert `wahr` annehmen, also erfüllt sind.

Betrachten Sie folgenden Algorithmus:

Algorithmus 1: MAXSAT APPROXIMATION

Eingabe : Endliche Menge $\mathcal{U}$ von booleschen Variablen, nicht leere Menge $\mathcal{C}$ von Klauseln über $\mathcal{U}$
Ausgabe : Wahrheitsbelegung der Variablen aus $\mathcal{U}$
Solange $\mathcal{U} \neq \emptyset$ **wiederhole**
  $u \leftarrow$ wähle ein Element $u \in \mathcal{U}$;
  $\mathcal{C}_w(u) \leftarrow \{c \in \mathcal{C} \mid c$ enthält Literal $u\}$;
  $\mathcal{C}_f(u) \leftarrow \{c \in \mathcal{C} \mid c$ enthält Literal $\overline{u}\}$;
  wenn $|\mathcal{C}_w(u)| \geq |\mathcal{C}_f(u)|$ **dann**
    $u \leftarrow$ `wahr` ;
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{C}_w(u)$;
  sonst
    $u \leftarrow$ `falsch` ;
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{C}_f(u)$;
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{u\}$;

(a) Gegeben sei die folgende MAXSAT Instanz $I = (\mathcal{U}, \mathcal{C})$ mit Variablenmenge $\mathcal{U}$ und Klauselmenge $\mathcal{C}$:

$$\mathcal{U} = \{u_1, u_2, u_3\}$$

$$\mathcal{C} = \{u_1 \vee u_2 \vee \overline{u_3}, \quad u_1 \vee \overline{u_2}, \quad \overline{u}_1 \vee u_2 \vee \overline{u_3}, \quad \overline{u_2} \vee u_3, \quad u_2 \vee \overline{u_3}, \quad \overline{u_2} \vee \overline{u_3}\}.$$

Geben Sie eine optimale Lösung für I an. Welchen Wert hat jede optimale Lösung?

Geben Sie zusätzlich die Lösung an, die Algorithmus 1 bei Eingabe von I berechnet. Nehmen Sie an, dass in jedem Schritt die Variable $u_i \in \mathcal{U}$ mit dem kleinsten Index i gewählt wird. Welchen Wert hat diese Lösung?

Lösung:

- Optimallösung: alle Variablen sind falsch. Wert: 6
- Lösung, die Algorithmus 2 berechnet: Alle Variablen sind wahr. Wert: 5

(b) Sei F die Menge der nicht erfüllten Klauseln bezüglich der Ausgabe von Algorithmus 1 und sei F_i die Menge der Klauseln, die im i-ten Schritt der Schleife durch Belegung der Variable u_i den Wert `falsch` annehmen, also unerfüllbar werden. Zeigen Sie, dass gilt:

$$F_i \cap F_j = \emptyset \text{ für } i \neq j,\ i,j \in \{1,\ldots,|\mathcal{U}|\} \quad \text{und} \quad \bigcup_{i=1}^{|\mathcal{U}|} F_i = F.$$

Lösung:

1) OBdA wird u_i vor u_j betrachtet. Unmittelbar vor der Betrachtung von u_i hat jede Klausel in F_i die Form `falsch`$\vee u_i$, d.h. die ursprüngliche Klausel enthält u_j nicht, da u_j noch nicht betrachtet und damit noch nicht belegt wurde. Jede Klausel in F_j enthält aber u_j. Damit gilt $F_i \cap F_j = \emptyset$.

2) $\bigcup_{i=1}^{|\mathcal{U}|} F_i \subseteq F$ klar. Zu zeigen: $\bigcup_{i=1}^{|\mathcal{U}|} F_i \supseteq F$. Sei $c \in F$ eine Klausel, so ist c mit der Belegung, die Algorithmus 1 berechnet nicht erfüllt, d.h., alle Literale haben den Wert `falsch`. Da Algorithmus 1 sequenziell vorgeht, muss aber eines der Literale als letztes belegt worden sein. OBdA gehöre dieses Literal zur Variable u_k. Damit ist $c \in F_k$ und es gilt $F \subseteq \bigcup_{i=1}^{|\mathcal{U}|} F_i$.

(c) Zeigen Sie, dass Algorithmus 1 1-approximativ ist, d.h. dass Algorithmus 1 eine relative Gütegarantie von $1 + 1 = 2$ hat.

Lösung:
Zu zeigen: $\frac{\mathrm{OPT}(I)}{\mathcal{A}(I)} \leq 2$. Sei T bzw. F die Menge der Klauseln, die in $\mathcal{A}(I)$, erfüllt bzw. nicht erfüllt sind. Weiterhin sei T_i die Menge der Klauseln, die vor dem i-ten Schritt nicht erfüllt waren und im i-ten Schritt durch die Belegung der i-ten Variablen erfüllt werden, es gilt, dass alle T_i disjunkt sind und nach Annahme ist $\cup_{i=1}^{|\mathcal{U}|} T_i = T$ (Argumentation analog zu b)). F_i ist die Menge der Klauseln, die im i-ten Schritt noch in $\mathcal{C}$ sind und die durch die Belegung der i-ten Variablen unerfüllbar werden. Die F_i sind paarweise disjunkt und es ist $\cup_{i=1}^{|\mathcal{U}|} F_i = F$ (mit b)). Weiterhin gilt, dass für alle i $F_i \leq T_i$ ist und damit

$$\frac{\mathrm{OPT}(I)}{\mathcal{A}(I)} \leq \frac{|\mathcal{C}|}{|T|} = \frac{\sum_{i=1}^{|\mathcal{U}|} |T_i| + \sum_{i=1}^{|\mathcal{U}|} |F_i|}{\sum_{i=1}^{|\mathcal{U}|} |T_i|} \leq \frac{2 \cdot \sum_{i=1}^{|\mathcal{U}|} |T_i|}{\sum_{i=1}^{|\mathcal{U}|} |T_i|} = 2$$

(d) Zeigen Sie: Falls $\mathcal{P} \neq \mathcal{NP}$, gibt es keinen absoluten Approximationsalgorithmus für MaxSAT.
Sie dürfen benutzen, dass MaxSAT $\mathcal{NP}$-schwer ist.

Lösung:
Annahme: Es gibt einen absoluten Approximationsalgorithmus $\mathcal{A}'$ mit $|OPT(I) - \mathcal{A}'(I)| \leq K$ für alle I. Betrachte Instanz $I' = (\mathcal{U}', \mathcal{C}')$, die aus einer beliebigen Instanz $I = (\mathcal{U}, \mathcal{C})$ hervorgeht, indem die Instanz I $K+1$ mal kopiert wird. In I' werden alle Kopien einer Variablen als verschieden betrachtet. Es gilt also $|\mathcal{U}'| = (K+1)|\mathcal{U}|$ und $|\mathcal{C}'| = (K+1)|\mathcal{C}|$. Es gilt $OPT(I') = (K+1)OPT(I)$, da eine optimale Löesung in I eine optimale Löesung in jeder Kopie von I erzeugt.
Betrachte die beste Lösung, die die Lösung $\mathcal{L}_{\mathcal{A}'}(I')$ in einer der Kopien von I induziert. Diese Lösung ist eine Lösung für I mit Wert $a \geq \frac{\mathcal{A}'(I')}{K+1}$. Damit gilt $K \geq |OPT(I') - \mathcal{A}'(I')| \geq |(K+1)OPT(I) - (K+1)a|$, und so ist $|OPT(I) - a| \leq \frac{K}{K+1} < 1$ und a entspräche dem Optimalwert (wegen Ganzzahligkeit), $\mathcal{L}$ wäre somit eine optimale Lösung für I, die polynomiell berechnet werden kann, da $\mathcal{A}'$ sowie die Umformung zwischen I und I' polynomiell sind. Dies ist ein Widerspruch zu $\mathcal{P} \neq \mathcal{NP}$, da MaxSAT $\mathcal{NP}$-schwer ist.

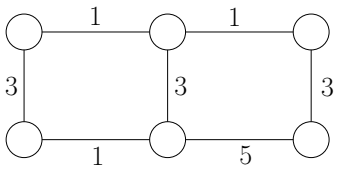
Aufgabe 9: (2 + 3 + 3 = 8 Punkte)

Gegeben sei ein einfacher, ungerichteter, positiv gewichteter Graph $G = (V, E, c)$ mit Gewichtsfunktion $c : E \to \mathbb{N}$. Eine Teilmenge $S \subseteq V$ induziert eine Zerteilung von V in zwei (möglicherweise leere) Mengen S und $V \setminus S$. Das Gewicht $c(S, V \setminus S)$ einer solchen Zerteilung ist die Summe der Gewichte aller Kanten, die zu genau einem Knoten in S inzident sind, d.h. $c(S, V \setminus S) := \sum_{\{u,v\} \in E : u \in S, v \notin S} c(\{u, v\})$.
Das Problem MAXCUT fragt nach einer Teilmenge $S \subseteq V$, sodass $c(S, V \setminus S)$ maximal ist.
Betrachten Sie folgenden Algorithmus.

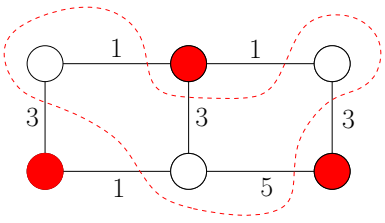
Algorithmus 2: ALMOST MAXCUT

Eingabe : Einfacher, ungerichteter Graph $G = (V, E, c)$ mit Gewichtsfunktion $c : E \to \mathbb{N}$
Ausgabe : Teilmenge $S \subseteq V$
$S \leftarrow \emptyset$;
Solange *Knoten u in S oder $V \setminus S$ existiert, sodass ein Seitenwechsel von u $c(S, V \setminus S)$ vergrößert* **wiederhole**
  Führe Seitenwechsel von u durch, d.h. $S \leftarrow S \cup \{u\}$ falls $u \notin S$ und $S \leftarrow S \setminus \{u\}$ falls $u \in S$;
Gib S aus;

(a) Zeichnen Sie eine möglichst gute Lösung S_1 für den Graphen G_1 in untenstehende Abbildung ein. Markieren Sie die Knoten in S_1 deutlich.



Lösung:



(b) Zeigen Sie, dass die Schleife in Algorithmus 2 terminiert.

Lösung:
Da die Gewichte positiv ganzzahlig sind, wächst das Gewicht in jedem Schritt mindestens um 1. Mit $c(S, V \setminus S) \leq \sum_{e \in E} c(e)$ finden somit maximal $\sum_{e \in E} c(e)$ Durchläufe statt.

(c) Zeigen Sie, dass Algorithmus 2 eine **pseudo**-polynomielle Laufzeit hat, indem Sie zeigen, dass die Laufzeit polynomiell in der Anzahl der Kanten und Knoten in G und in der größten vorkommenden Zahl der Gewichtsfunktion ist.
Sie dürfen annehmen, dass für einen Knoten u in konstanter Zeit gepüft werden kann, ob ein Wechsel das Gewicht erhöht. Ebenso sei der Seitenwechsel eines Knotens in konstanter Zeit durchführbar.

Lösung:
Pro Schleifendurchlauf wird für maximal $|V|$ Knoten die Schleifenbedingung überprüft. Dies ergibt einen Aufwand in $O(|V| \sum_{e \in E} c(e))$. Außerdem gilt $\sum_{e \in E} c(e) \leq k|E|$, mit k größte vorkommende Zahl. Damit ist die Laufzeit von Algorithmus 2 in $O(k|V||E|)$.

Aufgabe 10: (10 Punkte)

Jeder der folgenden Aussagenblöcke umfasst drei Einzelaussagen. Die sich unterscheidenden Aussagenbausteine sind durch Kästchen gekennzeichnet. Kreuzen Sie genau jene Bausteine an, die in einer wahren Einzelaussage enthalten sind. Jeder Aussagenblock enthält mindestens eine wahre Einzelaussage. Unvollständig oder falsch angekreuzte Aussagenblöcke werden mit null Punkten bewertet, Sie erhalten einen Punkt für jeden Aussagenblock, für den Sie genau die richtige Menge an Aussagenbausteinen angekreuzt haben.

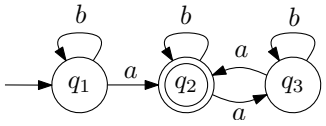
Jede Sprache, die das Pumping-Lemma für reguläre Sprachen erfüllt

- ☐ ist regulär.
- ☐ erfüllt auch das verallgemeinerte Pumping-Lemma für reguläre Sprachen.
- ☐ erfüllt auch das Pumping-Lemma für kontextfreie Sprachen.

Lösung:

a) gilt nach Skript nicht, b) gilt nicht, siehe Beispiel 5 im Skript, c) gilt, da eine Zerlegung gemäß des PL für reguläre Sprachen auch dem PL für kontextfreie Sprachen genügt mit $u = u, v = v, w = \varepsilon, x = \varepsilon$

Für den folgenden deterministischen endlichen Automaten $\mathcal{A}$ über dem Alphabet $\Sigma = \{a, b\}$ gilt:



- ☐ b ist Zeuge für die Nichtäquivalenz von q_1 und q_3.
- ☐ q_1 und q_3 sind äquivalent.
- ☐ Der Index der Neroderelation der von $\mathcal{A}$ akzeptierten Sprache $L(\mathcal{A})$ ist 3.

Lösung:

nur b) gilt, denn ε trennt $\{q_2\}$ von $\{q_1, q_3\}$, Worte der Länge 1 trennen nichts, also sind q_1 und q_3 äquivalent und der Minimalautomat hat 2 Zustände.

Falls L_1 und L_2 semi-entscheidbare Sprachen sind, dann ist

- ☐ $L_1 \cup L_2$ entscheidbar.
- ☐ $L_1 \cup L_2$ semi-entscheidbar.
- ☐ L_1^c semi-entscheidbar.

Lösung:

a) gilt nicht, wenn man z.B. die Halteproblemsprache mit der leeren Menge vereinigt, c) gilt nicht, sonst gäbe es keine echt semi-entscheidbaren Sprachen, b) gilt, siehe Übung

Falls 2SAT $\mathcal{NP}$-vollständig ist, gilt:

- ☐ 3SAT $\in \mathcal{P}$.
- ☐ 2SAT $\notin \mathcal{P}$.
- ☐ für alle $L \in \mathcal{NP} : L \propto$ 2SAT.

Lösung:

a) gilt, denn da 3SAT in $\mathcal{NP}$ liegt, gibt es eine p.T. von 3SAT auf 2SAT, aus 2SAT $\in \mathcal{P}$ folgt dann 3SAT $\in \mathcal{P}$, b) gilt nicht, siehe Skript, c) gilt siehe Skript.

Jede reguläre Sprache L ist
- ☐ kontextfrei.
- ☐ endlich.
- ☐ in $\mathcal{NTAPE}(n)$ enthalten.

Lösung:

Es gilt a) und c), siehe Skript.

Aus $\mathcal{P} \neq \mathcal{NP}$ folgt:
- ☐ 3SAT $\notin \mathcal{P}$.
- ☐ Es gibt keine deterministische 2-Band-Turingmaschine, die 3SAT in Polynomialzeit entscheidet.
- ☐ Es gibt keinen absoluten Approximationsalgorithmus für das Optimierungsproblem CLIQUE.

Lösung:

Es gilt a),b),c)

Falls es ein polynomiales Approximationsschema (PAS) für ein Optimierungsproblem Π gibt, dann
- ☐ ist Π nicht $\mathcal{NP}$-schwer.
- ☐ gibt es einen Approximationsalgorithmus mit relativer Gütegarantie für Π.
- ☐ gibt es keinen Approximationsalgorithmus mit relativer Gütegarantie für Π.

Lösung:

Es gilt nur b)

Gegeben sei die Grammatik G mit Nichtterminalmenge $\{S, A, B\}$, Terminalmenge $\{a, b\}$, Startsymbol S und folgender Regelmenge:

$$S \to aBA|a$$
$$A \to aS|a$$
$$B \to bBBS|b$$

Die oben gegebene Grammatik G ist
- ☐ in Chomsky-Normalform.
- ☐ in Greibach-Normalform.
- ☐ kontextfrei.

Lösung:

Es gelten b) und c)

Eine Sprache ist semi-entscheidbar genau dann, wenn es eine
- ☐ Grammatik gibt, die sie erzeugt.
- ☐ deterministische Turingmaschine gibt, die genau die Eingaben w akzeptiert, für die $w \in L$ gilt.
- ☐ deterministische Turingmaschine gibt, die auf allen Eingaben w hält, für die $w \in L$ gilt.

Lösung:

Es gelten a) und b), siehe Skript, c) gilt nicht, denn TM kann für jede Sprache konstruiert werden

Aus $\mathcal{NP} \neq$ co-$\mathcal{NP}$ folgt:
- ☐ $\mathcal{P} \neq \mathcal{NP}$.
- ☐ TSP liegt nicht in co-$\mathcal{NP}$.
- ☐ $\mathcal{NPI} \neq \emptyset$.

Lösung:

Es gelten a) b) c)