Universität Osnabrück / FB6 / Theoretische Informatik — Prof. Dr. M. Chimani

Informatik D: Einführung in die Theoretische Informatik

Klausur — SoSe 2017 — 10. Juli 2017

Haupttermin, Prüfungsnr. 1007049 — Gruppe: Christopher Reeve / Lyle Talbot

Unbedingt ausfüllen

Matrikelnummer | Studiengang/Abschluss | Fachsemester

Nachname | Vorname

Unterschrift | Identifikator (Beliebiges Wort zur Identifikation im anonymen Notenaushang)

Grundregeln

- Die Bearbeitungszeit der Klausur beträgt **120 Minuten**.
- Sie schreiben diese Klausur **vorbehaltlich** der Erfüllung der **Zulassungsvoraussetzung**. Das heißt: Wir werden Ihre Zulassung vor Korrektur prüfen; die Tatsache, dass Sie die Klausur mitschreiben, bedeutet keine implizite Zulassung.
- Es sind **keine Unterlagen** und auch **keine** anderen **Hilfsmittel** erlaubt.
- Benutzen sie nur dokumentenechten (blauen oder zur Not schwarzen) **Kugelschreiber**! Bleistiftlösungen werden nicht gewertet!
- Es zählt die Antwort, die sich im dafür vorgesehenen Kästchen befindet! Soll eine andere Antwort gewertet werden, so ist diese **eindeutig** zu kennzeichnen!
- Jegliches Schummeln, und auch der Versuch desselben, führt zum Ausschluss von der Klausur und einer Bewertung mit **5,0**.

Wird vom Korrektor/Prüfer ausgefüllt

Aufgabe	1	2	3	4	5	6	7	8	9	10	Σ
Punkte (max)	12	14	6	6	12	18	12	16	12	28	**136**
Punkte (erreicht)											

Punkte	0..67	68..75	76..83	84..88	89..94	95..99	100..104	105..110	111..115	116..123	124..136
Note	**5,0**	**4,0**	**3,7**	**3,3**	**3,0**	**2,7**	**2,3**	**2,0**	**1,7**	**1,3**	**1,0**

Note:

Aufgabe 1: Sprachgrundlagen (12 Punkte)

(a) Faktenwissen (6 Punkte)

Wie ist die Form einer kontextfreien Grammatik?

Sei Σ die Menge der Symbole und V die Menge der Variablen.

Was ist die Laufzeitkomplexität (abhängig wovon?) des Wortproblems bei KF Sprachen?

Durch welches Maschinenmodell wird bzw. durch welche Maschinenmodelle werden kontextsensitive Sprachen beschrieben?

(b) Endlichkeitsproblem (6 Punkte)

Ist die durch die folgende Grammatik erzeugte Sprache L endlich? Begründen Sie!

$S \to A \mid BC \qquad A \to D \mid b \qquad B \to A \mid AC \qquad C \to aC \qquad D \to B \mid C$

Aufgabe 2: Sprachen klassifizieren (14 Punkte)

Zu welcher der *fünf* in der Vorlesung besprochenen Sprachfamilien innerhalb der Chomsky-Hierarchie gehören die folgenden Sprachen? Geben Sie dabei die *kleinste* korrekte Antwort an, also die Sprachfamilie, die gerade mächtig genug ist, die entsprechende Sprache zu erkennen.

L, wobei $\bar{L}$ eine deterministisch kontextfreie Sprache ist	
$\{a^i b^j c^k d^j e^i \mid i, j, k \in \mathbb{N}\}$	
$\{w \mid w$ ist eine durch vier teilbare Binärzahl, $\lvert w\rvert \geq 42\}$	
$S \to aAb \mid cAa \mid bAc \mid \varepsilon \qquad A \to bSa \mid aSc \mid cSb \mid \varepsilon$	
$\{w \mid w$ ist ein JAVA-Programm, das die ersten fünf Nachkommastellen der Eulerschen Zahl e berechnet$\}$	
$\{wvw \mid w \in \{a, b\}^*, v \in \{c, d\}\}$	
Eine Sprache, die keine monotone Grammatik erlaubt	

Aufgabe 3: Umwandlung: RegEx → endlicher Automat **(6 Punkte)**

Wandeln Sie den regulären Ausdruck $a(b|c)^*$ gemäß dem Vorgehen aus der Vorlesung in einen äquivalenten endlichen Automaten um.

Sie müssen keine Zwischenschritte angeben. Lassen Sie keine Kanten weg!

Aufgabe 4: Kellerautomat **(6 Punkte)**

Sei $\Sigma = \{0, 1, 2\}$. Geben Sie einen Kellerautomat an, der genau die durch die folgenden Regeln eingeschränkte Sprache $L \subseteq \Sigma^*$ akzeptiert.

- Ist $a_1 a_2 \cdots a_n \in L$ mit $a_1, \ldots, a_n \in \Sigma$, so enthält jeder Präfix (d. h. jedes Wort $a_1 a_2 \cdots a_k$ mit $k \in \{1, \ldots, n\}$) höchstens so viele 2en wie 1en.
- Jedes Wort enthält mindestens eine 0.

Aufgabe 5: Abschlusseigenschaften (12 Punkte)

(a) Abgeschlossenheit beweisen (8 Punkte)

Zeigen Sie, dass reguläre Sprachen bezüglich Vereinigung, Komplement und Schnitt abgeschlossen sind.

Zeigen Sie, dass kontextfreie Sprachen bezüglich Schnitt nicht abgeschlossen sind.

(b) Anwendung (4 Punkte)

Gegeben $L_1 := \{\$^\ell \in^{(\ell^2)} \mid \ell \in \mathbb{N}_+\}$. Zeigen Sie mittels Abschlusseigenschaften, dass L_1 nicht regulär ist. Sie können dazu als bekannt voraussetzen, dass die Sprache aller unär kodierten Quadratzahlen nicht regulär ist.

Aufgabe 6: Pumping Lemma (18 Punkte)

(a) Definition (4 Punkte)

Wie lautet das Pumping Lemma für kontextfreie Sprachen?

Für jede KF Sprache L gibt es ein n, sodass

sich zerlegen lassen als $z = u\,v\,w\,x\,y$, sodass (1) ,

(2) und (3) .

(b) Beweisidee (2 Punkte)

Der Beweis des Pumping Lemmas nutzt den Ableitungsbaum einer kontextfreien Grammatik. Warum dürfen wir annehmen, dass dieser Ableitungsbaum binär ist?

(c) Anwendung (12 Punkte)

Beweisen Sie mittels Pumping Lemma, dass $L := \{ss^R t \mid s \in \Sigma^*, t \in \{s, s^R\}\}$ mit $\Sigma := \{0, 1, 2\}$ nicht kontextfrei ist.

Angenommen ...

Aufgabe 7: Rechnende Turingmaschine **(12 Punkte)**

Geben Sie eine deterministische Turingmaschine an, die für ein unär kodiertes $\alpha \in \mathbb{N}$ die folgende Funktion $f(\alpha)$ berechnet. Die Ausgabe soll abermals unär kodiert sein.

$$f(\alpha) = \begin{cases} 1 & \text{wenn 2 Teiler von } \alpha \text{ ist, aber nicht 3,} \\ 2 & \text{wenn 3 Teiler von } \alpha \text{ ist, aber nicht 2,} \\ 3 & \text{wenn weder 2 noch 3 Teiler von } \alpha \text{ sind,} \\ \mathit{undef} & \text{sonst.} \end{cases}$$

Stellen Sie sicher, dass nach der Berechnung der Turingmaschine ausschließlich die Ausgabe auf dem Band steht. Beachten Sie, dass auch $\alpha = 0$ als Eingabe gültig ist.

Aufgabe 8: Zusammenhänge in Berechenbarkeit und Komplexität (16 Punkte)

In jeder Teilaufgabe führt genau ein Kreuz zu einer wahrheitsgemäßen Aussage. Kreuzen Sie die korrekten Aussagen an.
Achtung: Pro Teilaufgabe gibt es $2/0/-1$ *Punkte bei einer richtigen/keinen/falschen Antwort! Es gibt jedoch keine negativen Punkte für die gesamte Aufgabe.*

Beim Post'schen Korrespondenzproblem kann man in endlicher Zeit

- ☐ ...einen Nein-Zeugen finden, falls er existiert.
- ☐ ...einen Nein-Zeugen finden oder zeigen, dass er nicht existiert.
- ☐ ...einen Ja-Zeugen finden oder zeigen, dass er nicht existiert.
- ☐ ...einen Ja-Zeugen finden, falls er existiert.

Sei SITH eine Programmiersprache. Man kann die Turing-Vollständigkeit von SITH beweisen, indem man zeigt,

- ☐ ...wie SITH LOOP-Programme simulieren kann.
- ☐ ...wie WHILE-Programme SITH-Programme simulieren können.
- ☐ ...wie WHILE-Programme mittels SITH-Programmen simuliert werden können.
- ☐ ...wie SITH die Ackermann-Funktion berechnen kann.

Das Addieren zweier Zahlen α, β benötigt eine Laufzeit von

- ☐ ...$\Theta(1)$ im uniformen Kostenmaß und $\mathcal{O}(\log \alpha + \log \beta)$ im logarithmischen Kostenmaß.
- ☐ ...$\Theta(\alpha + \beta)$ im uniformen Kostenmaß und $\Omega(\log(\alpha + \beta))$ im logarithmischen Kostenmaß.
- ☐ ...$\Theta(\max(\log \alpha, \log \beta))$ sowohl im uniformen als auch im logarithmischen Kostenmaß.

Wenn ein Entscheidungsproblem in ***PSPACE*** liegt,

- ☐ ...ist es deterministisch in polynomieller Zeit entscheidbar.
- ☐ ...ist es deterministisch in exponentieller Zeit entscheidbar.
- ☐ ...ist es in polynomiellen Platz entscheidbar, aber man kann nichts über die Zeit sagen.

Ein Entscheidungsproblem mit Eingabegröße n ist in ***FPT*** in Bezug auf Parameter k genau dann, wenn es sich in $\mathcal{O}(f_1(n) \cdot f_2(k))$ Zeit lösen lässt,

- ☐ ...wobei $f_1(n)$ ein Polynom in n und $f_2(k)$ berechenbar ist.
- ☐ ...wobei $f_1(n)$ berechenbar und $f_2(k)$ ein Polynom in k ist.
- ☐ ...wobei $f_1(n)$ ein Polynom in n und $f_2(k)$ ein Polynom in k ist.

Das Entscheidungsproblem zu FACTORIZATION liegt nach aktuellem Wissensstand

- ☐ ...in ***P***.
- ☐ ...in ***NP*** $\cap$ ***Co-NP***.
- ☐ ...entweder in ***NP*** oder in ***Co-NP***.

Sei (A, b) eine SUBSETSUM-Instanz. Bei der dynamischen Programmierung ist $Q[i, s] =$ `true`,

- ☐ ...gdw. eine Teilmenge der ersten i Elemente die Summe s ergeben kann.
- ☐ ...gdw. die ersten i Elemente die Summe s ergeben.
- ☐ ...gdw. eine s-elementige Teilmenge der ersten i Elemente die Summe b ergibt.

Wir reduzieren ein schwach ***NP***-vollständiges Problem $\mathcal{Y}$ deterministisch in polynomieller Zeit auf ein Problem $\mathcal{X}$. Unter der Annahme ***P*** $\neq$ ***NP*** gilt:

- ☐ ...$\mathcal{X}$ ist ***NP***-schwer.
- ☐ ...$\mathcal{X} \in$ ***P***.
- ☐ ...es gibt einen pseudopolynomiellen Algorithmus für $\mathcal{X}$.

Aufgabe 9: GOTO-Programmierung (12 Punkte)

(a) WHILE → GOTO (4 Punkte)

Wandeln Sie das folgende WHILE-Programm in ein äquivalentes GOTO-Programm um. Beim bedingten Sprung sind Bedingungen $x_i \circ 0$ erlaubt, wobei $\circ \in \{=, \neq, <, >\}$.

```
x2 := x1;
x3 := 0;
while (x2 ≠ 0) {
   x2 := x2 − 1;
   x3 := x3 + 5
};
x3 := x3 + 17
```

(b) Semi-Entscheidbarkeit (4 Punkte)

Gegeben sei ein GOTO-Programm P mit Variablen $x_1, \ldots, x_k$, $k \geq 2$. Beweisen oder widerlegen Sie die Semi-Entscheidbarkeit des folgenden Problems: Wird die Variable x_2 jemals initialisiert?

(c) Co-Semi-Entscheidbarkeit (4 Punkte)

Ist das Problem aus **(b)** co-semi-entscheidbar? Begründen Sie!

Aufgabe 10: Musikalische Graphen (28 Punkte)

Betrachten Sie das folgende Problem:

Problem: k-VERTONUNG
Gegeben: Ein ungerichteter, zusammenhängender Graph $G = (V, E)$.
Gefragt: Gibt es eine Funktion $\mu\colon V \to \{0, \ldots, k-1\}$ mit $\mu(u) \neq \mu(v)$ für alle $uv \in E$?

Die Zahlen $0, \ldots, k-1$ stehen für Töne, die den Knoten zugewiesen werden. Die Anzahl k ist ein Parameter des Problems und gehört nicht zur Eingabe. Ja-Instanzen des Problems werden *k-vertonbar* genannt. Einen Ja-Zeugen μ bezeichnen wir als *(gültige) k-Vertonung*.

(a) Entscheidungsalgorithmus für 2-VERTONUNG (8 Punkte)

Wir betrachten in dieser Teilaufgabe $k = 2$.

Geben Sie eine Nein- und Ja-Instanz für dieses Problem mit mindestens drei Knoten an.

Nein-Instanz:

Ja-Instanz:

Beschreiben Sie, wie Sie eine *Tiefensuche* verändern können, um 2-VERTONUNG deterministisch in Polynomialzeit zu entscheiden. (Wie erkennen Sie dabei eine Nein- und eine Ja-Instanz?)

Wie ist die Laufzeit Ihres Algorithmus in $\mathcal{O}$-Notation?

(b) ***NP*-Vollständigkeit von 3-VERTONUNG (20 Punkte)**

Wir betrachten in dieser Teilaufgabe $k = 3$. Auf den folgenden Seiten ist ein ***NP***-Vollständigkeitsbeweis für 3-VERTONUNG ausgeführt, den Sie durch korrektes Ankreuzen und Ausfüllen der Freifelder vervollständigen müssen. Beim Ankreuzen ist pro Block nur genau eine Antwort richtig.

Als erstes müssten wir zeigen, dass man einen Ja-Zeugen für 3-Vertonung deterministisch in/mit polynomieller

☐	☐	☐	☐
Größe finden	Zeit finden	Größe validieren	Zeit validieren

kann. Wir verzichten auf diesen Beweisteil und nehmen an, wir hätten dies schon gezeigt. Es bleibt mittels Reduktion zu zeigen, dass 3-Vertonung ***NP***-schwer ist.

Beweisidee:

☐	Wir reduzieren von 3-Vertonung auf 3-Sat.
☐	Wir reduzieren von 3-Sat auf 3-Vertonung.
☐	Wir reduzieren von 2-Sat auf 3-Vertonung.
☐	Wir reduzieren von 2-Vertonung auf 3-Vertonung.

Wir verwenden dazu Gadgets dreier verschiedener Typen.

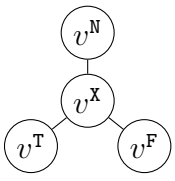
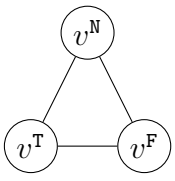
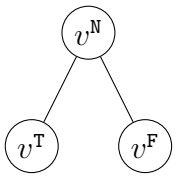
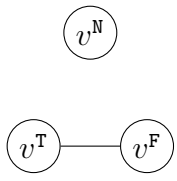
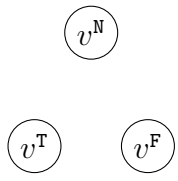
- Im einzigen Gadget des ersten Typs $\mathcal{G}_1$ werden die Töne $0, 1, 2$ mit den Werten `false`, `true`, „nicht belegt“ verknüpft.
- Für jede Variable gibt es ein Gadget vom Typ $\mathcal{G}_2$. Es weist den zugehörigen Literalen den Wert `true` bzw. `false` zu.
- Für jede Klausel gibt es ein zugehöriges Gadget vom Typ $\mathcal{G}_3$, das sie simuliert.

Die 3-Vertonbarkeit von $\mathcal{G}_1$ bildet die Grundlage für unsere Argumentation.

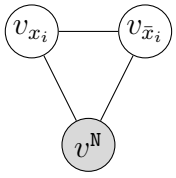
Reduktion:

☐	Gegeben eine beliebige 3-Sat-Instanz F mit n Variablen $x_1, \ldots, x_n$ und o. B. d. A. exakt drei Literalen pro Klausel. Wir erzeugen daraus eine 3-Vertonung-Instanz $G = (V, E)$.
☐	Gegeben eine beliebige 3-Vertonung-Instanz $G = (V, E)$. Wir erzeugen daraus eine 3-Sat-Instanz F.
☐	Gegeben eine beliebige 2-Sat-Instanz F mit n Variablen $x_1, \ldots, x_n$ und o. B. d. A. exakt zwei Literalen pro Klausel. Wir erzeugen daraus eine 3-Vertonung-Instanz $G = (V, E)$.
☐	Gegeben eine beliebige 2-Vertonung-Instanz. Wir erzeugen daraus eine 3-Vertonung-Instanz $G = (V, E)$.

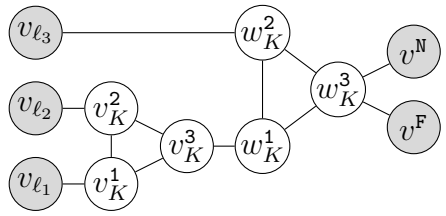
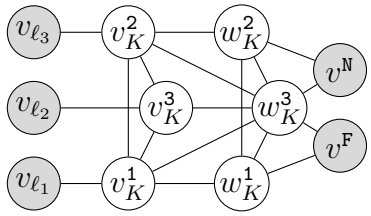
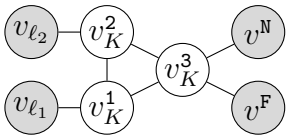
Wir haben genau ein Gadget des Typs $\mathcal{G}_1$:

☐	☐	☐	☐	☐
				

Für jede vorkommende Variable x_i haben wir ein Gadget des Typs $\mathcal{G}_2$, wobei der Knoten $v^{\mathtt{N}}$ aus dem Gadget des Typs $\mathcal{G}_1$ stammt:



Für jede Klausel K mit Literalen ℓ_i haben wir ein Gadget des Typs $\mathcal{G}_3$, wobei die Knoten $v^{\mathtt{N}}$ und $v^{\mathtt{F}}$ aus $\mathcal{G}_1$ und die v_{ℓ_i} aus den entsprechenden Gadgets des Typs $\mathcal{G}_2$ stammen:

☐	☐	☐
		

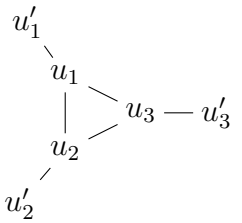
Laufzeit der Reduktion: Wir müssen für die Reduktion zeigen, dass sie

Sei m die Anzahl der Klauseln und n die Anzahl der Variablen. Die Gesamtlaufzeit der Reduktion ist $\mathcal{O}($ ☐ $)$.

Korrektheit der Reduktion: Für die Korrektheit der Reduktion müssen wir zeigen, dass

☐	F ist unerfüllbar $\Leftrightarrow$ G ist eine Ja-Instanz.
☐	F hat einen Ja-Zeugen $\Leftrightarrow$ G ist nicht 3-vertonbar.
☐	F ist eine Ja-Instanz $\Leftrightarrow$ G hat einen Ja-Zeugen.

Dazu erst eine Beobachtung: Sei (u_1, u_2, u_3) ein Dreieck (d. h. ein Kreis der Länge 3). Die Knoten u_1, u_2, u_3 müssen paarweise verschieden vertont sein. Seien die Knoten u_i jeweils adjazent zu Knoten u_i' außerhalb des Dreiecks (siehe Abbildung). Wenn u_1' oder u_2', aber nicht u_3' mit Ton 1 vertont sind, dann kann u_3 mit 1 vertont werden. Haben u_1' und u_2' den gleichen Ton, so muss auch u_3 diesen Ton bekommen.



„⇒“: Sei $x_1', \ldots, x_n'$ ein Ja-Zeuge, der F erfüllt. Wir erzeugen eine 3-Vertonung $\mu' \colon V \to \{0, 1, 2\}$ wie folgt: Wir setzen $\mu'(v^{\mathtt{F}}) := 0$, $\mu'(v^{\mathtt{T}}) := 1$ und $\mu'(v^{\mathtt{N}}) := 2$. Für alle $i = 1, \ldots, n$ gilt: Ist $x_i' = \mathtt{false}$, so setzen wir $\mu'(v_{x_i}) := 0$. Ist $x_i' = \mathtt{true}$, so setzen wir $\mu'(v_{x_i}) := 1$. Wir setzen entsprechend $\mu'(v_{\bar{x}_i}) := 1 - \mu'(v_{x_i})$.

Die Gadgets der Typen $\mathcal{G}_1$ und $\mathcal{G}_2$ sind gültig mit drei Tönen vertont. Wir müssen noch über die Vertonbarkeit der $\mathcal{G}_3$-Gadgets argumentieren: Sei K eine Klausel mit Literalen ℓ_i. Weil F und damit K erfüllt ist, existiert ein i mit $\mu'(v_{\ell_i}) = 1$. Dann existiert nach obiger Beobachtung eine gültige 3-Vertonung,

☐	sodass $\mu'(v_K^3) = \mu'(w_K^3) = 1$.
☐	sodass $\mu'(v_K^3) = \max\{\mu'(v_{\ell_1}), \mu'(v_{\ell_2})\}$ und $\mu'(w_K^3) = 1$.
☐	weil $\mathcal{G}_3$ die Klauseln auf `false` setzt. Ein Widerspruch zur Erfüllbarkeit von F.
☐	weil $\mathcal{G}_2$ die Klauseln auf `true` setzt.

„⇐“: Sei $\mu' \colon V \to \{0, 1, 2\}$ eine gültige 3-Vertonung für den konstruierten Graphen $G = (V, E)$. Wir erzeugen einen Ja-Zeugen $x'_1, \ldots, x'_n$ für F, wie folgt: In $\mathcal{G}_1$ müssen $v^{\mathtt{F}}$, $v^{\mathtt{T}}$, $v^{\mathtt{N}}$ verschieden vertont sein. Sei o. B. d. A. $\mu'(v^{\mathtt{F}}) = 0, \mu'(v^{\mathtt{T}}) = 1, \mu'(v^{\mathtt{N}}) = 2$. Für $i = 1, \ldots, n$ sind wegen $\mathcal{G}_2$ die Knoten v_{x_i} und $v_{\bar{x}_i}$ unterschiedlich und jeweils nicht mit 2 vertont. Ist $\mu'(v_{x_i}) = 1$, so setzen wir $x'_i := \texttt{true}$, sonst $x'_i := \texttt{false}$.

Wir analysieren $\mathcal{G}_3$ für eine beliebige Klausel K mit Literalen ℓ_i.

Angenommen ...

Dies ist ein Widerspruch zur 3-Vertonung μ'. □