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: Adam West / Heath Ledger

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 regulären Grammatik?

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

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

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

(b) Endlichkeitsproblem (6 Punkte)

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

$S \to AB \mid C \qquad A \to B \mid D \qquad B \to C \mid CD \qquad C \to A \mid a \qquad D \to bD$

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_1L_2, wobei L_1 und L_2 jeweils deterministisch kontextfreie Sprachen sind	
$\{a^ib^jc^kd^ie^j \mid i,j,k \in \mathbb{N}\}$	
$\{w \mid w$ ist ein C++-Programm, das die ersten fünf Nachkommastellen der Kreiszahl π berechnet$\}$	
$S \to \varepsilon \mid A \qquad A \to Aa \mid Abc \mid d$	
$\{uvw \mid u \in \{a,b\}^*, v \in \{c,d\}, w = a^{\lvert u\rvert}\}$	
$\{w \mid w$ ist eine durch acht teilbare Binärzahl, $\lvert w\rvert \geq 20\}$	
Eine Sprache, die keine Grammatik in Greibach-Normalform 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 $k \in \{1, \ldots, n\}$) mindestens so viele 0en wie 1en.
- Jedes Wort enthält mindestens eine 2.

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 $L_1 := \{\heartsuit^{(\ell^2)}\lightning^{(\ell^4)} \mid \ell \in \mathbb{N}_+\}$. 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 $L := \{s_1 s_2 s_3 \mid s_1 \in \Sigma^*, s_2 = s_1^R, s_3 \in \{s_1, s_2\}\}$ mit $\Sigma := \{a, b, c\}$ 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.

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

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.*

Sei JEDI eine Programmiersprache. Man kann zeigen, dass JEDI Turing-vollständig ist, indem man zeigt,

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

Beim Modifizierten Post'schen Korrespondenzproblem kann man

- ☐ ...einen Ja-Zeugen, falls er existiert, in polynomieller Zeit finden.
- ☐ ...einen Ja-Zeugen, falls er existiert, in endlicher Zeit finden.
- ☐ ...einen Nein-Zeugen, falls er existiert, in endlicher Zeit finden.
- ☐ ...einen Nein-Zeugen, falls er existiert, in polynomieller Zeit finden.

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

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

Wenn ein Entscheidungsproblem in exponentieller Zeit gelöst werden kann, dann

- ☐ ...ist es ***NP***-vollständig.
- ☐ ...liegt es in ***EXPSPACE***.
- ☐ ...liegt es in ***PSPACE***.

Probleme in $\mathsf{NP} \cap \mathsf{Co\text{-}NP}$ sind nach aktuellem Wissensstand

- ☐ ...zum Beispiel GRAPHZUSAMMENHANG und FACTORIZATION.
- ☐ ...alle ***NP***-vollständig.
- ☐ ...alle in ***P***.

Bei der dynamischen Programmierung für SUBSETSUM ist $Q[i, s] = \texttt{true}$, gdw. eine Teilmenge der ersten i Elemente $a_1, \ldots, a_i$ die Summe s ergeben kann. Für $i, s \geq 1$ mit $s \geq a_i$ gilt

- ☐ ... $Q[i, s] := Q[i-1, s-a_i] \wedge Q[i-1, s]$
- ☐ ... $Q[i, s] := Q[i-1, s-a_i] \vee Q[i-1, s]$
- ☐ ... $Q[i, s] := Q[i-1, s+a_i] \wedge Q[i-1, s]$

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

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

Das Addieren zweier Zahlen α, β benötigt eine Laufzeit von

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

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 $\circ \in \{=, \neq, <, >\}$.

$x_2 := 0;$
$x_3 := x_1;$
while $(x_3 \neq 0)$ {
  $x_2 := x_2 + 17;$
  $x_3 := x_3 - 1$
};
$x_2 := x_2 + 5$

(b) Semi-Entscheidbarkeit (4 Punkte)

Gegeben sei ein GOTO-Programm P mit Variablen $x_1, \ldots, x_k$, $k \geq 3$, und eine Belegung der Variablen zu Beginn des Programms. Beweisen oder widerlegen Sie die Semi-Entscheidbarkeit des folgenden Problems: Nimmt die Variable x_3 jemals den Wert 0 an?

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

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

Aufgabe 10: Schmackhafte Graphen **(28 Punkte)**

Betrachten Sie das folgende Problem:

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

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

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

Wir betrachten in dieser Teilaufgabe $k = 2$.

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

Ja-Instanz:

Nein-Instanz:

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

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

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

Wir betrachten in dieser Teilaufgabe $k = 3$. Auf den folgenden Seiten ist ein *NP*-Vollständigkeitsbeweis für 3-WÜRZUNG 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-Würzung deterministisch in/mit polynomieller

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

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

Beweisidee:

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

Wir verwenden dazu Gadgets dreier verschiedener Typen.

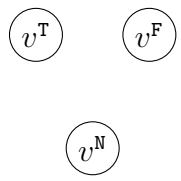
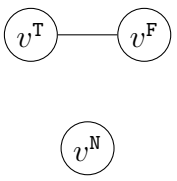
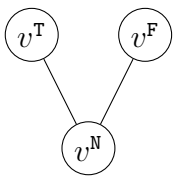
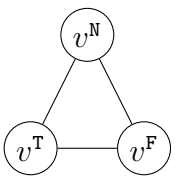
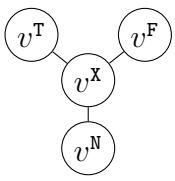
- Im einzigen Gadget des ersten Typs $\mathcal{G}_1$ werden die Gewürze $0, 1, 2$ mit den Werten „nicht belegt“, `false`, `true` verknüpft.
- Für jede Variable gibt es ein Gadget vom Typ $\mathcal{G}_2$. Es weist den zugehörigen Literalen den Wert `true` bzw. `false` zu.
- Für jede Klausel gibt es ein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

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

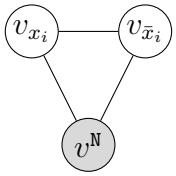
Reduktion:

☐	Gegeben eine beliebige 2-Würzung-Instanz. Wir erzeugen daraus eine 3-Würzung-Instanz $G = (V, E)$.
☐	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-Würzung-Instanz $G = (V, E)$.
☐	Gegeben eine beliebige 3-Würzung-Instanz $G = (V, E)$. Wir erzeugen daraus eine 3-Sat-Instanz F.
☐	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-Würzung-Instanz $G = (V, E)$.

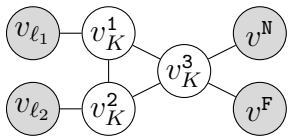
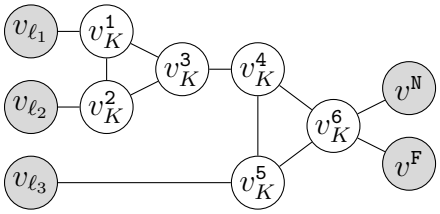
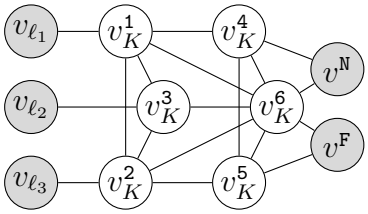
Wir haben genau ein Gadget des Typs $\mathcal{G}_1$:

☐	☐	☐	☐	☐
				

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:

☐	☐	☐
		

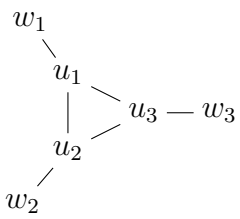
Laufzeit der Reduktion: Wir müssen für die Reduktion zeigen, dass sie

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 hat einen Ja-Zeugen.
☐	F hat einen Ja-Zeugen $\Leftrightarrow$ G ist 3-würzbar.
☐	F ist eine Ja-Instanz $\Leftrightarrow$ G hat einen Nein-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 gewürzt sein. Seien die Knoten u_i jeweils adjazent zu Knoten w_i außerhalb des Dreiecks (siehe Abbildung). Wenn w_1 oder w_2, aber nicht w_3 mit Gewürz 2 gewürzt sind, dann kann u_3 mit 2 gewürzt werden. Haben w_1 und w_2 das gleiche Gewürz, so muss auch u_3 damit gewürzt werden.



„⇒“: Sei $x'_1, \ldots, x'_n$ eine Belegung, die F erfüllt. Wir erzeugen eine 3-Würzung $\psi' \colon V \to \{0, 1, 2\}$ wie folgt: Wir setzen $\psi'(v^{\mathtt{N}}) := 0$, $\psi'(v^{\mathtt{F}}) := 1$ und $\psi'(v^{\mathtt{T}}) := 2$. Für alle $i = 1, \ldots, n$ gilt: Ist $x'_i = \texttt{false}$, so setzen wir $\psi'(v_{x_i}) := 1$. Ist $x'_i = \texttt{true}$, so setzen wir $\psi'(v_{x_i}) := 2$. Wir setzen entsprechend $\psi'(v_{\bar{x}_i}) := 3 - \psi'(v_{x_i})$.

Die Gadgets der Typen $\mathcal{G}_1$ und $\mathcal{G}_2$ sind gültig mit drei Gewürzen gewürzt. Wir müssen noch über die Würzbarkeit 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 $\psi'(v_{\ell_i}) = 2$. Dann existiert nach obiger Beobachtung eine gültige 3-Würzung,

☐	sodass $\psi'(v_K^{\mathtt{3}}) = \max\{\psi'(v_{\ell_1}), \psi'(v_{\ell_2})\}$ und $\psi'(v_K^{\mathtt{6}}) = 2$.
☐	sodass $\psi'(v_K^{\mathtt{3}}) = \psi'(v_K^{\mathtt{6}}) = 2$.
☐	weil $\mathcal{G}_2$ die Klauseln auf `true` setzt.
☐	weil $\mathcal{G}_3$ die Klauseln auf `false` setzt. Ein Widerspruch zur Erfüllbarkeit von F.

„⇐“: Sei $\psi' \colon V \to \{0, 1, 2\}$ eine gültige 3-Würzung für den konstruierten Graphen $G = (V, E)$. Wir erzeugen eine Belegung $x'_1, \ldots, x'_n$, die F erfüllt, wie folgt: In $\mathcal{G}_1$ müssen $v^{\mathtt{N}}$, $v^{\mathtt{F}}$, $v^{\mathtt{T}}$ verschieden gewürzt sein. Sei o. B. d. A. $\psi'(v^{\mathtt{N}}) = 0, \psi'(v^{\mathtt{F}}) = 1, \psi'(v^{\mathtt{T}}) = 2$. 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 0 gewürzt. Ist $\psi'(v_{x_i}) = 2$, 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-Würzung ψ'. □