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

Informatik D: Einführung in die Theoretische Informatik

Klausur — SoSe 2015 — 27. Juli 2014

Haupttermin, Prüfungsnr. 1007049 — Gruppe: Imperium (Anakin, Sheev)

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! Falsche Kreuzchen können zu Punkteabzug innerhalb der Teilaufgabe führen.
- 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	8	16	12	16	12	12	14	10	20	**132**
Punkte (erreicht)											

Punkte	0..65	66..73	74..81	82..86	87..91	92..96	97..101	102..106	107..112	113..119	120..132
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: Sprache vs. Grammatik (12 Punkte)

(a) Hierarchie und Automaten (8 Punkte)

Zu jeder Sprache gibt es entsprechende Automaten. Vervollständigen Sie die folgende Tabelle:

Name der Sprachfamilie	Automaten
	NDEA,
rekursiv aufzählbar	
kontextsensitiv	

(b) Grammatikdefinition (4 Punkte)

Welche Sprachenklasse wird durch Grammatiken in Chomsky-Normalform genau beschrieben?

Sei V die Menge der Variablen, und Σ das betrachetet Alphabet. Definieren Sie oben genannte Normalform.

Alle Regeln haben die Form...

Aufgabe 2: Sprachen klassifizieren (8 Punkte)

Zu welcher Sprachklasse gehören die folgenden Sprachen? Kreuzen Sie dabei *alle* korrekten Antworten an.

	regulär	determ. kontextfrei	kontextfrei	kontextsensitiv	rek. aufzählbar
$\{w^R w \mid w$ ist Wort einer regulären Sprache über $\Sigma = \{\text{♪}, \text{♩}, \text{𝅗𝅥}\}\}$	☐	☐	☐	☐	☐
$\{a^i b^j c^{i+j} b^j a^i \mid i, j \geq 2\}$	☐	☐	☐	☐	☐
$\{w_1 w_2 \mid w_1, w_2 \in \{a, b, c\}^*, 2\|w_1\| \bmod 3 = 4\|w_2\| \bmod 3\}$	☐	☐	☐	☐	☐

Aufgabe 3: Pumping Lemma **(16 Punkte)**

(a) Definition (4 Punkte)

Vervollständigen Sie die Definition des Pumping Lemmas für kontextfreie Sprachen.

☐ Sei L eine nicht-kontextfreie Sprache L. Es gibt es eine Zahl n, sodass ein Wort $z \in L$ mit $|z| \geq n$ existiert, für das eine Zerlegung $z = uvwxy$ existiert, mit...

☐ Sei L eine kontextfreie Sprache. Für jede Zahl n gibt es ein Wort $z \in L$, sodass $|z| \geq n$ und eine Zerlegung $z = uvwxy$ existiert, mit...

☐ Sei L eine kontextfreie Sprache. Es gibt eine Zahl n, sodass für jedes Wort $z \in L$ mit $|z| \geq n$ eine Zerlegung $z = uvwxy$ existiert, mit...

...den folgenden drei Eigenschaften:

(1), (2), (3).

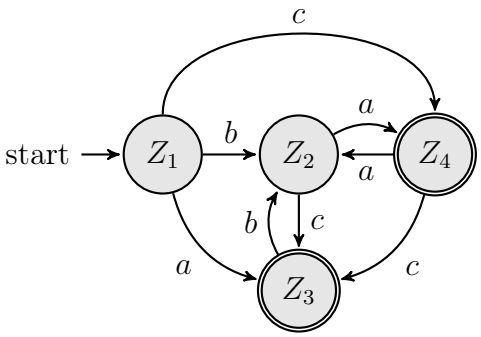
(b) Anwendung (12 Punkte)

Zeigen Sie mittels Pumping Lemma, dass die Sprache $L := \{a^{\lceil i/3 \rceil}(bb)^{\lceil i/5 \rceil}c^{\lfloor i/2 \rfloor} \mid i \geq 2\}$ nicht kontextfrei ist.

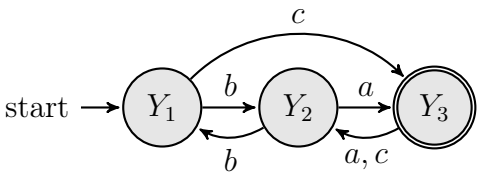
Aufgabe 4: Abgeschlossenheit regulärer Sprachen **(12 Punkte)**

Erstellen Sie – nach dem Vorgehen aus der Vorlesung! – aus den beiden gegebenen DEAs $\mathcal{A}_1$ und $\mathcal{A}_2$ einen DEA $\mathcal{B}$ mit $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.

$\mathcal{A}_1$:



$\mathcal{A}_2$:



Aufgabe 5: Algorithmische Fragestellungen **(16 Punkte)**

(a) Graphen (8 Punkte)

Ein *einfacher* gerichteter Graph besteht aus einer Menge von Knoten und einer Menge von Kanten. Eine Kante ist ein 2-Tupel von unterschiedlichen Knoten. Wieviele Kanten kann ein solcher Graph mit n Knoten maximal haben?

Gegeben ein endlicher Automat $\mathcal{A}$, der die Sprache L beschreibt. Sie möchten das Leerheitsproblem für L lösen. Dazu interpretieren Sie $\mathcal{A}$ als einen gerichteten Graphen G mit n Knoten und m Kanten.

Welchen Algorithmus benutzen Sie, um das Problem auf G zu lösen?

Was ist die benötigte Laufzeit in $\mathcal{O}$-Notation?

Welche Datenstruktur benutzt man typischerweise, um in diesem Algorithmus die Knoten zu betrachten?

☐ Stack ☐ Queue ☐ PriorityQueue (Heap) ☐ Binärer Suchbaum

(b) Dynamische Programmierung (8 Punkte)

Beschreiben Sie das generelle Konzept der *dynamischen Programmierung*:

Nennen Sie zwei Beispiele (z.B. aus Informatik D), in denen dynamische Programmierung klassischerweise zum Einsatz kommt:

-
-

Aufgabe 6: Kellerautomat **(12 Punkte)**

Gegeben die folgende kontextfreie Sprache L (als Grammatik in EBNF):

$$S \to \{A\}bBb$$
$$A \to a[A]c$$
$$B \to bBb \mid bb$$

Beschreiben Sie die Sprache L in Mengenschreibweise (also nach dem Muster $\{ab^i \mid i \geq 1\}$). Vereinfachen Sie soweit wie möglich!

Erstellen Sie einen Kellerautomaten, der genau L akzeptiert.

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

Gegeben eine unär kodierte Zahl $\alpha > 0$. Geben Sie eine Turingmachine an, die die folgende Funktion berechnet:

$$f(\alpha) := \begin{cases} \alpha/2 & \alpha \bmod 2 = 0 \\ \text{undef} & \text{sonst} \end{cases}$$

Aufgabe 8: Turing-Vollständigkeit **(14 Punkte)**

Betrachten Sie die folgende Programmiersprache MultFun:

Variablen in MultFun werden mit $v[i]$, $i \geq 0$, bezeichnet. In jeder Variable $v[i] \in \mathbb{Z}$ wird eine ganze Zahl gespeichert.

Die einzelnen Anweisungen in MultFun werden nacheinander ausgeführt und jeweils durch einen Punkt voneinander getrennt. Wir haben die folgenden Anweisungen zur Verfügung:

Anweisung	Beschreibung
copy c to $v[j]$	kopiert den Wert der Konstante $c \in \mathbb{Z}$ nach $v[j]$.
copy $v[i]$ to $v[j]$	kopiert den Wert von $v[i]$ nach $v[j]$
add $v[i]$ to $v[j]$	erhöht (Addition) $v[j]$ um $v[i]$
subtract $v[i]$ from $v[j]$	verkleinert (Subtraktion) $v[j]$ um $v[i]$
multiply $v[i]$ to $v[j]$	speichert den Wert des Produkts $v[i] * v[j]$ in $v[j]$
unless $v[i]$ do $\mathcal{A}$ end	führt die Anweisungen $\mathcal{A}$ aus, außer wenn $v[i] \leq 0$
repeat $\mathcal{A}$ until $v[i]$	führt die Anweisungen $\mathcal{A}$ aus (mind. 1-mal), bis $v[i] \leq 0$

Beispiel: Das folgende MultFun-Programm berechnet $(32 + v[1] - 2) \cdot v[2]$ und speichert den Wert in $v[3]$:

```
copy 32 to v[3]. add v[1] to v[3]. copy 2 to v[4]. subtract v[4] from v[3]. multiply v[2] to v[3].
```

(a) Definition (4 Punkte)

Definieren Sie Turing-Vollständigkeit:

(b) Beweis der Turing-Vollständigkeit (8 Punkte)

Zeigen Sie, dass die Programmiersprache MultFun Turing-vollständig ist. (Beschreiben Sie vor dem eigentlichen Beweis auch kurz die allgemeine Idee des Vorgehens.)

(c) Turing-Äquivalenz (2 Punkte)

Was müssten Sie darüberhinaus zeigen, um zu beweisen, dass MultFun sogar *Turing-äquivalent* ist?

Aufgabe 9: Berechenbarkeit **(10 Punkte)**

Ein *Astronaut* ist eine Turingmaschine mit trinärem Alphabet $\Gamma = \{\square, \bullet, \star\}$ (Leere, Planet, Stern), bei der die folgenden drei Eigenschaften erfüllt sind:

(i) Das Speicherband ist initial nur mit $\square$-Symbolen gefüllt.

(ii) Der SL-Kopf ändert die Richtung nur, falls gerade das Symbol $\star$ gelesen wird.
Falls aktuell also kein $\star$ gelesen wird, darf der SL-Kopf nur stehenbleiben oder einen Schritt in die Richtung machen, in die er als letztes (ggf. vor dem Stehenbleiben) einen gemacht hat.

(iii) Sie hält an.

(a) Semi-Entscheidbarkeit (6 Punkte)

Zeigen Sie, dass es semi-entscheidbar ist, ob eine gegebene TM $\mathcal{T}$ ein Astronaut ist:

(b) Unentscheidbarkeit (4 Punkte)

Ein *Angry Astronaut* ist ein Astronaut, der die größtmögliche Schrittanzahl erzielt.

Zeigen Sie, dass es unentscheidbar ist, ob eine gegebene TM $\mathcal{T}$ ein Angry Astronaut ist:

Aufgabe 10: *NP*-Vollständigkeit **(20 Punkte)**

Ein Problem $\mathcal{X}$ ist *NP*-schwer, genau dann wenn...

Ein Problem $\mathcal{X}$ ist *NP*-vollständig, genau dann wenn...

Betrachten Sie das folgende Problem

Problem: PenPacking

Gegeben: Mehrere Kugelschreibern (Pens) und Federmäppchen. Die Kugelschreiber sind gegeben als Menge $\mathcal{K}$ von k 2-Tupeln, d.h. $\{(v_i, a_i)\}_{1 \leq i \leq k}$. Dabei gibt v_i das Volumen eines Kugelschreibers und a_i die Anzahl der Kugelschreiber dieses Volumens an. Es gilt, dass $v_i \neq v_{i'}$ für $i \neq i'$. Die Federmäppchen sind gegeben als m-elementige Liste $\mathcal{M} = \langle c_j \rangle_{1 \leq j \leq m}$; das j-te Federmäppchen hat dabei die Kapazität c_j.

Frage: Kann man alle Kugelschreiber in den vorhandenen Federmäppchen unterbringen, sodass für jedes Federmäppchen gilt: die Summe der Volumina der Kugelschreiber die in ihm sind, übersteigt seine Kapazität nicht um mehr als 50%?

Wir wollen im Folgenden zeigen, dass PenPacking *NP*-vollständig ist.

Definieren Sie, wie ein (im Weiteren nutzbarer) Zeuge für dieses Problem aufgebaut ist:

Wir benötigen im Beweis ein weiteres *NP*-vollständiges Problem. Welches? Definieren Sie es:

Beweisen Sie nun, dass PenPacking **NP**-vollständig ist:

Beweisteil 1.

Beweisteil 2.

Viel Erfolg!