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

Informatik D: Einführung in die Theoretische Informatik

Klausur — SoSe 2015 — 1. Oktober 2015

Nebentermin, Prüfungsnr. 1007049 — Gruppe: Dr. Jekyll = Mr. Hyde

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)	6	10	10	16	12	12	12	14	16	24	**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: Sprachfamilien **(6 Punkte)**

Welche Automatenklasse(n) entsprechen genau den kontextfreien Sprachen?

Welche Sprachfamilie der Chomsky-Hierarchie ist die *ausdruckstärkste*?

Was ist die kleinste Sprachklasse, mit denen man (allgemeine) Palindrome beschreiben kann?

Aufgabe 2: Abschlusseigenschaften **(10 Punkte)**

Zeigen oder widerlegen Sie:

Die Vereinigung zweier regulärer Sprachen ist regulär.

Zeigen oder widerlegen Sie:

Das Komplement einer DKF Sprache ist kontextfrei.

Zeigen oder widerlegen Sie:

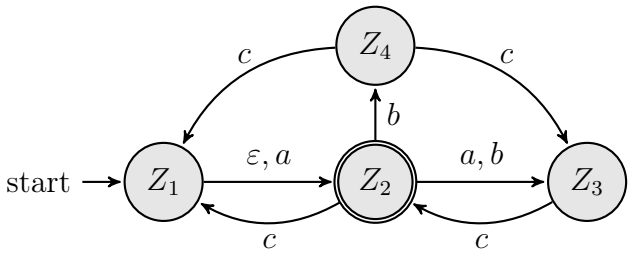
Der Schnitt einer regulären mit einer kontextsensitiven Sprache ist regulär.

Ist das *Leerheitsproblem* für rekursiv aufzählbare Sprachen entscheidbar? Ja □, Nein □

Ist das *Endlichkeitsproblem* für kontextfreie Sprachen (als Grammatik gegeben) in linearer Zeit lösbar? Ja □, Nein □

Aufgabe 3: Umwandeln: NDEA → DEA **(10 Punkte)**

Wandeln Sie – gemäß dem Vorgehen aus der Vorlesung! – den folgenden nicht-deterministischen endlichen Automaten in einen deterministischen endlichen Automaten um:



Aufgabe 4: Pumping Lemma **(16 Punkte)**

(a) Definition (4 Punkte)

Wie lautet das Pumping Lemma für reguläre Sprachen?

(b) Anwendung (12 Punkte)

Beweisen Sie, dass die Sprache $L = \{a^{(i^2)} \mid i \in \mathbb{N}\}$ nicht regulär ist.

Aufgabe 5: Kellerautomat **(12 Punkte)**

Erstellen Sie einen Kellerautomaten, der genau die Sprache mit folgenden Eigenschaften akzeptiert:

- Das Alphabet der Sprache ist $\Sigma = \{a, b, c\}$.
- Kein Wort enthält das Teilwort cc.
- Sei $n_\star^w$ die Anzahl der Vorkommen des Symbols $\star \in \Sigma$ im Wort w.
  Sei $n^w(S) := \sum_{\star \in S} n_\star^w$ die Anzahl der Vorkommen der Symbole $S \subseteq \Sigma$ im Wort w.
  Für jedes Wort w der Sprache gilt: n_a^w und $n^w(\{b, c\})$ unterscheiden sich um maximal 1.

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

An zahlreichen Stellen haben wir eine Tiefensuche (oder Breitensuche) auf einem gerichteten Graphen $G = (V, E)$, ausgehend von einem Knoten $s \in V$, bemüht, um zu argumentieren, warum diverse Sprachenprobleme leicht lösbar sind.

(a) Pseudocode (8 Punkte)

Welcher der folgenden zwei Pseudocodes repräsentiert eine Tiefensuche, die die Knoten in entsprechender Reihenfolge ausgibt?

Kreuzen Sie an und vervollständigen Sie den entsprechenden Code.

☐ **Variante A**

```
void dfs(v) {
  if(v unmarkiert) {
    print v
    markiere v
    forall                    {
      dfs(      )
    }
  }
}
```

Init: alle Knoten unmarkiert
Aufruf: `dfs(s)`

☐ **Variante B**

```
void dfs(v) {
  if(v unmarkiert) {
    print v
    forall                    {
      dfs(      )
    }
    markiere v
  }
}
```

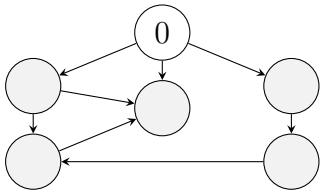
Init: alle Knoten unmarkiert
Aufruf: `dfs(s)`

(Raum für Anmerkungen etc., falls notwendig – Kann auch freigelassen werden!)

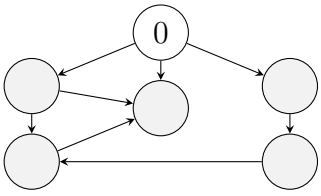
Was ist die Laufzeit einer Tiefensuche? $\mathcal{O}($ $)$

(b) Tiefensuche vs. Breitensuche (4 Punkte)

Nummerieren Sie die Knoten entsprechend einer zulässigen Reihenfolge, wenn Sie beim Knoten „0“ eine Tiefensuche bzw. Breitensuche starten:



Tiefensuche



Breitensuche

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

Gegeben zwei binär kodierte Zahlen α, β mit $\alpha > 123$ und $\beta \in \{1, 2, 4\}$, getrennt durch ein $\square$-Symbol. Geben Sie eine Turingmaschine an, die die folgende Funktion berechnet:

$$f(\alpha, \beta) := \begin{cases} \alpha/\beta & \text{falls } \beta \text{ ein Teiler von } \alpha \text{ ist,} \\ \mathsf{undef} & \text{sonst.} \end{cases}$$

Aufgabe 8: LOOP vs. WHILE **(14 Punkte)**

(a) LOOP-Programme (6 Punkte)

Definieren Sie die LOOP-Programmiersprache.

(b) WHILE $\geq$ LOOP (4 Punkte)

Zeigen Sie, dass die WHILE-Programmiersprache *mindestens so mächtig* ist wie LOOP.

(c) WHILE > LOOP (4 Punkte)

Zeigen Sie, dass die WHILE-Programmiersprache *mächtiger* ist als LOOP.

Hinweis: Stellen Sie nicht nur eine Behauptung auf!

Aufgabe 9: Theorie-Quiz (16 Punkte)

(Achtung: Falsche Antworten zählen innerhalb einer Teilaufgabe negativ! richtige/falsche/keine Antwort $\rightarrow +2/-1/0$ *Punkte)*

(a) Berechenbarkeit (8 Punkte)

korrekt	falsch	
☐	☐	Die Ackermannfunktion ist berechenbar.
☐	☐	Es gibt bestimmte Turing-vollständige Programmiersprachen, bzgl. der die Kolmogorov-Komplexität eines Wortes berechenbar ist.
☐	☐	Die Kolmogorov-Komplexität gibt eine untere Schranke an, wie sehr sich Daten komprimieren lassen.
☐	☐	Die Funktion f mit $f(i) = \mathsf{undef}$ für alle $i \in \mathbb{N}$, ist berechenbar.

(b) Komplexität (8 Punkte)

korrekt	falsch	
☐	☐	Pseudopolynomielle Algorithmen für stark NP-vollständige Probleme sind polynomiell, falls die vorkommenden Zahlenwerte nur polynomiell groß sind.
☐	☐	Ein Optimierungsproblem ist NP-schwer *genau dann* wenn das zugehörige Entscheidungsproblem NP-vollständig ist.
☐	☐	Falls ein Problem aus $\mathsf{Co\text{-}NP}$ gleichzeitig in NP liegt, dann gilt $\mathsf{P} = \mathsf{NP}$.
☐	☐	Das Problem „Ist eine gegebene 2-SAT Formel erfüllbar“ liegt in NP.

Aufgabe 10: Komplexität **(24 Punkte)**

(a) *NP*-Vollständigkeit (8 Punkte)

Gegeben ein Problem SUPERSCHWER, von dem Sie zeigen wollen, dass es *NP*-vollständig ist. Beschreiben Sie das notwendige Vorgehen: *Was* ist zu tun? *Was* ist zu zeigen? *Wie* zeigt man das im Allgemeinen?

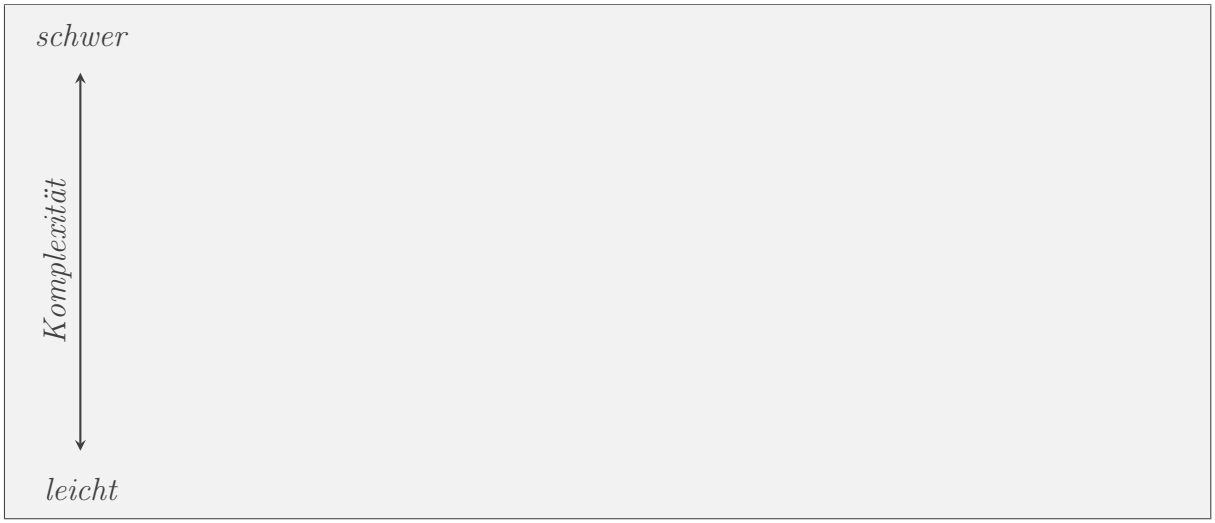
(b) *Co-NP* (4 Punkte)

Definieren Sie die Menge *Co-NP*.

(c) Mengendiagramm (6 Punkte)

Nehmen Sie innerhalb dieser Aufgabe an, dass $P \neq NP \neq Co\text{-}NP$.

- Zeichnen Sie ein Mengendiagramm (Venn-Diagramm) für die Komplexitätsklassen P, NP, NP-vollständig, $Co\text{-}NP$ und $Co\text{-}NP$-vollständig.
- Markieren Sie darin auch, wo sich NP-schwere Probleme befinden.



(d) Höhere Komplexität (6 Punkte)

$EXPTIME$ ist die Menge aller Entscheidungsprobleme, die sich mittels einer Turingmaschine in exponentieller Zeit entscheiden lassen.

$PSPACE$ ist die Menge aller Entscheidungsprobleme, die sich mittels einer Turingmaschine in polynomiell viel Platz entscheiden lassen (d.h. die Turingmaschine benötigt nur ein Speicherband polynomieller Länge).

Begründen Sie, warum $PSPACE \subseteq EXPTIME$ gilt.

Viel Erfolg!