Technische Universität München
Fakultät für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Werner Meixner, Alexander Krauss

Sommersemester 2008
Lösungsblatt Abschlussklausur
28. Juli 2008

Einführung in die Theoretische Informatik

Name	Vorname	Studiengang	Matrikelnummer
....	....	☐ Diplom ☐ Inform. ☐ Bachelor ☐ BioInf. ☐ Lehramt ☐ Mathe.	....

Hörsaal	Reihe	Sitzplatz	Unterschrift
....	....	....	....

Allgemeine Hinweise

- Bitte füllen Sie obige Felder in Druckbuchstaben aus und unterschreiben Sie!
- Bitte schreiben Sie nicht mit Bleistift oder in roter/grüner Farbe!
- Die Arbeitszeit beträgt 120 Minuten.
- Alle Antworten sind in die geheftete Angabe auf den jeweiligen Seiten (bzw. Rückseiten) der betreffenden Aufgaben einzutragen. Auf dem Schmierblattbogen können Sie Nebenrechnungen machen. Der Schmierblattbogen muss ebenfalls abgegeben werden, wird aber in der Regel nicht bewertet.
- Es sind keine Hilfsmittel außer einem handbeschriebenen DIN-A4-Blatt zugelassen.

Hörsaal verlassen von ...... bis ...... / von ...... bis ......
Vorzeitig abgegeben um ......
Besondere Bemerkungen:

	A1	A2	A3	A4	A5	Σ	Korrektor
Erstkorrektur							
Zweitkorrektur							

Aufgabe 1 (9 Punkte)

Wahr oder falsch? Begründen Sie im Folgenden Ihre Antworten möglichst knapp!

1. Eine Sprache $A \subseteq \Sigma^*$ mit $A \neq \emptyset$ ist genau dann rekursiv aufzählbar, wenn es eine totale und berechenbare Funktion $f : \mathbb{N} \to \Sigma^*$ gibt, so dass
$$A = \{f(0), f(2), f(4), f(6), f(8), \ldots\} .$$
2. Das Halteproblem für LOOP-Programme ist entscheidbar.
3. Die folgende Instanz des Post'schen Korrespondenzproblems hat eine Lösung:
$$(001, 10), (01, 0100), (011, 11)$$
4. Es ist unentscheidbar, ob es für ein LOOP-Programm ein WHILE-Programm gibt, das dieselbe Funktion berechnet.
5. Es gibt kontextfreie Sprachen $L_1 \neq L_2$, so dass deren Durchschnitt $L_1 \cap L_2$ nicht leer und kontextfrei ist.
6. Die Sprache $L = \{w \mid \exists x.\ M_w[x] \text{ hält nach höchstens } 2^{10} \text{ Schritten}\}$ ist entscheidbar.
7. Ein PCP P hat entweder keine Lösung oder unendlich viele Lösungen.
8. Für die Problemklasse P gilt: $\{A \mid \overline{A} \in \mathrm{P}\} \cap \mathrm{P} = \emptyset$.
9. Wenn $A \leq_p B$ und $B \in TIME(O(1))$, dann ist $A \in \mathrm{P}$.

Lösungsvorschlag

1. (w) wähle z.B. $g(n) = f(2n)$. Die Gegenrichtung funktioniert ähnlich.
2. (w) LOOP-Programme halten immer.
3. (w) 2,1,3
4. (f) trivial entscheidbar, da jedes LOOP-Programm auch ein WHILE-Programm ist.
5. (w) z.B. $L_1 = \{a, b\}, L_2 = \{b, c\}$
6. (w) Maschine für *alle(!) Eingaben* $|w| < 2^10$ jeweils 2^10 Schritte simulieren.
7. (w) Das Verdoppeln einer Lösung erzeugt wieder eine Lösung.
8. (f) $\ldots = \mathrm{P}$, wg. Abschluss unter Komplement.
9. (w) Denn B ist ja in P, und mit der Reduktion somit auch A. Achtung: $A \notin TIME(O(1))$.

Je 0,5P. für Aussage und 0,5P. für Begründung.

Aufgabe 2 (7 Punkte)

Wir betrachten die Sprache $L = \{a^{2n}b^n \mid n \in \mathbb{N}\}$ über dem Alphabet $\Sigma = \{a, b\}$ zusammen mit ihrer gespiegelten Sprache $L^R = \{w^R \mid w \in L\}$. Sei $L' = LL^R$.

1. Konstruieren Sie einen Kellerautomaten $K' = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, der L' mit Endzuständen akzeptiert. Geben Sie insbesondere eine knappe informelle Beschreibung Ihrer Konstruktionsidee an!

2. Gibt es einen deterministischen Kellerautomaten, der $L' \setminus \{\epsilon\}$ mit leerem Keller akzeptiert? Begründen Sie Ihre Antwort!

Lösungsvorschlag

1. Seien $Q = \{q_0, q_1, q_1', q_2, q_3, q_4, q_4', q_f\}$, $F = \{q_f\}$ und $\Gamma = \{Z_0, A, B\}$. ($\frac{1}{2}$ P.)

   Konstruktionsidee: Zunächst werden im Anfangszustand Übergänge zur Behandlung der Fälle $n = 0$ bzw. $m = 0$ definiert. In Phase 1 werden doppelte a's gelesen und je ein A gekellert, in Phase 2 werden b's gelesen und jeweils ein A im Keller gelöscht. Phase 3 beginnt bei Kellerende mit Lesen von b's und Kellern von jeweils ein B, die dann in Phase 4 mit je 2 a's wieder gelöscht werden. Nach der zweiten Kellerleerung wird der Endzustand angenommen.

   (1 P.)

   Übergangsfunktion:

   $$\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0), (q_3, Z_0), (q_f, Z_0)\} \qquad \text{(1 P.)}$$

   $$\delta(q_1, a, *) = (q_1', A*)\,, \quad \delta(q_1', a, *) = (q_1, *)\,, \quad \delta(q_1, \epsilon, A) = (q_2, A)\,,$$

   $$\delta(q_2, b, A) = (q_2, \epsilon)\,, \quad \delta(q_2, \epsilon, Z_0) = (q_3, Z_0)\,, \qquad \text{(1 P.)}$$

   $$\delta(q_3, b, *) = (q_3, B*)\,, \quad \delta(q_3, \epsilon, B) = (q_4, B)\,,$$

   $$\delta(q_4, a, B) = (q_4', \epsilon)\,, \quad \delta(q_4', a, *) = (q_4, *)\,, \quad \delta(q_4, \epsilon, Z_0) = (q_f, Z_0)\,. \qquad \text{(2 P.)}$$

2. Nein, ($\frac{1}{2}$ P.)
   denn $L' \setminus \{\epsilon\}$ erfüllt nicht die Präfixbedingung. (1 P.)

Korrekturvorgaben

1. 5P., die wie folgt verteilt werden:

0,5P.	für Angabe des PDA-Tupels (Syntax)
1P.	für 1. "Teilautomat"
1P.	für 2. "Teilautomat"
1P.	für richtige Behandlung des ϵ (bei auch sonst vernünftigem Automat).
1P.	für richtiges Akzeptieren mit Endzuständen
1P.	für informelle Beschreibung (muss zum Automaten passen).

Die Konstruktion über Grammatik wird analog bewertet.

Wird versucht für die falsche Sprache $\{a^{2n}b^{2n}a^{2n} \mid n \in \mathbb{N}\}$ einen PDA zu bauen, kann man da nur wenig Punkte vergeben. Die Konstruktion kann gar nicht funktionieren, da diese Sprache nicht kontextfrei ist.

2. Siehe oben. Für sinnvollen Zusammenhang zur Präfixbedingung, die aber falsch auf die Sprache angewandt wird, gibt es noch 0,5P. Richtiges "Nein" mit total abwegiger Begründung sind 0P.

Aufgabe 3 (8 Punkte)

Das Mengenzerteilungsproblem (MZ) ist wie folgt definiert:

Gegeben: Eine endliche Menge U, und Teilmengen $S_1, \ldots, S_k \subseteq U$.

Problem: Kann man die Elemente von U so einfärben (jeweils in Rot oder Blau), dass jedes S_i mindestens ein rotes und ein blaues Element enthält?

Formal: Gibt es eine Teilmenge $R \subseteq U$, so dass für alle $i \in \{1, \ldots, k\}$ gilt:

$$\emptyset \subsetneq S_i \cap R \subsetneq S_i$$

Intuition: R enthält genau die roten Elemente.

1. Geben Sie einen deterministischen Algorithmus an, der das Mengenzerteilungsproblem für $k = 2$ in Polynomialzeit löst.
2. Zeigen Sie, dass MZ $\leq_p$ SAT. Erläutern Sie Ihre Konstruktion auch informell. Hinweis: Verwenden Sie die Variablenmenge $\{r_x \mid x \in U\}$.
3. Seien nun $A, B \subseteq \Sigma^*$ beliebige Probleme.

   Zeigen Sie: Wenn $\emptyset \subsetneq A \subsetneq \Sigma^*$ und $B \in \mathrm{P}$, dann gilt $B \leq_p A$.

Lösungsvorschlag

1. Wenn es nur zwei Mengen gibt, hat eine Probleminstanz genau dann eine Lösung, wenn S_1 und S_2 mindestens zwei Elemente haben. Denn dann gibt es die Fälle:
   (a) $S_1 \subseteq S_2$. Dann legt man für die Elemente von S_1 eine geeignete Färbung fest, die dann auch zu S_2 passt.
   (b) $S_2 \subseteq S_1$. Analog
   (c) Andernfalls gibt es ein $x \in S_1 \setminus S_2$ und ein $y \in S_2 \setminus S_1$. Dann kann man x und y rot färben und alle anderen Elemente blau.

   Man muss also nur die Größe der Mengen anschauen.
2. Wir wählen die Variablenmenge $\{r_x \mid x \in U\}$ und konstruieren die Formel

$$F = \bigwedge_{1 \leq i \leq k} \left((\neg \bigwedge_{x \in S_i} r_x) \wedge \bigvee_{x \in S_i} r_x \right)$$

   Das ist offensichtlich in Polynomialzeit möglich, und die Belegungen kodieren genau die möglichen Lösungen des Problems.

3. Die charakteristische Funktion $\chi_B : \Sigma^* \to \mathbb{N}$ ist in polynomieller Zeit durch eine DTM M berechenbar.

   Da $\emptyset \subsetneq A \subsetneq \Sigma^*$, gibt es ein $x_1 \in A$ und ein $x_2 \notin A$.

   Wir definieren $f : \Sigma^* \to \Sigma^*$ durch

$$f(w) := \begin{cases} x_1 & \text{falls } \chi_B(w) = 1\,, \\ x_2 & \text{falls } \chi_B(w) = 0\,. \end{cases}$$

   f ist total und in polynomieller Zeit berechenbar, und es gilt:
   $w \in B \Leftrightarrow f(w) \in A\,.$

Korrekturvorgaben

1. 3P. Viele angegebenen Algorithmen konstruieren explizit eine Färbung. Das war nicht gefragt, ist aber OK, solange explizit abgebrochen wird wenn keine Lösung möglich ist (anstelle eine falsche zu produzieren). Diese "konstruktive" Variante kann sogar teilweise die Begründungen ersetzen, da hier die nötigen Fallunterscheidungen gemacht werden.

   2P. Aussage "Lösbar gdw. $|S_i| \geq 2$"
   Begründung:
   0,5P. eine Fallunterscheidung erkennbar
   0,5P. ... und ist auch vollständig.

   Dann ist der Algorithmus trivial.

   Nur Algorithmus (der nichts konstruiert) ist das gleiche wie Nur Aussage: 2P.
   Dafür muss er aber funktionieren.

   oder

   2,5-3P. Algorithmus, der eine Färbung konstruiert, mit der expliziten "Annahme", dass $|S_i| \geq 2$. Die Begründung steht dann in den Fallunterscheidungen im Algorithmus.

   oder Teilpunkte bei nicht funktionierenden Algorithmen:

   0,5P. wenn eine korrekte Färbung immer gefunden wird, wenn sie existiert (Vollständigkeit). Wenn unlösbar wird falsche Lösung erzeugt.
   0,5P. Problem erkannt, dass die Mengen überlappen
   0,5P. Problem erkannt, dass die Mengen zu klein sein können (aber nicht klar formuliert).

   Keine Punkte gibt es für Algorithmen, die ein anderes Problem lösen, oder die exponentielle Laufzeit haben.

2. 3P. Dafür genügt eine richtige SAT-Formel. Richtige Varianten zur obigen Lösung sind:

$$\bigwedge_{1\leq i\leq k}\left(\bigvee_{x\in S_i}\neg r_x \wedge \bigvee_{x\in S_i} r_x\right) \qquad \text{und (etwas redundant)} \qquad \bigwedge_{1\leq i\leq k}\left(\bigvee_{x\in S_i}\bigvee_{\substack{y\in S_i\\ y\neq x}}(r_x \wedge \neg r_y)\right)$$

oder

Wenn keine richtige SAT-Formel dasteht gibt es Teilpunkte:

0,5P.	Für die Aussage, dass die Variablen r_i genau für die roten Elemente stehen
1P.	Für die richtige informelle Beschreibung der codierten Eigenschaften
1P.	Für die richtige Codierung der (evtl. falschen) Eigenschaften in SAT. Davon je 0,5P. pro "Richtung".
0,5P.	Für eine Komplexitätsabschätzung einer (evtl. falschen) Codierung.

3. Hier geht es um eine triviale Reduktion, die man nur formal richtig als solche begreifen kann. Für eine informelle Argumentation kann es maximal 0,5P. geben.

1P.	Für eine formale konstruktion einer Reduktion in die richtige Richtung, unter Ausnutzung von $B \in \mathrm{P}$.
1P.	Für das explizite Ausnutzen der Eigenschaft $\emptyset \subsetneq A \subsetneq \Sigma^*$, durch die man geeignete Elemente findet, auf die man reduziert.

Aufgabe 4 (8 Punkte)

1. Zeigen Sie, dass $\mathit{even} : \mathbb{N} \to \mathbb{N}$ mit

$$\mathit{even}(n) = \begin{cases} 1 & \text{wenn } n \text{ gerade} \\ 0 & \text{sonst} \end{cases}$$

   primitiv rekursiv ist.

   Verwenden Sie hierfür *nur* die unten angegebenen Funktionen und Schemata.

2. Zeigen Sie, dass $\mathit{half} : \mathbb{N} \to \mathbb{N}$ mit

$$\mathit{half}(n) = \left\lfloor \frac{n}{2} \right\rfloor$$

   primitiv rekursiv ist.

   Hier dürfen Sie zusätzlich zu den unten angegebenen Funktionen und Schemata auch das *erweiterte* Schema der primitiven Rekursion verwenden.

3. Sei $f : \mathbb{N} \to \mathbb{N}$. Dann ist $\mathit{sum} : \mathbb{N} \to \mathbb{N}$ wie definiert als

$$\mathit{sum}(n) = \sum_{i=0}^{n} f(i) \ .$$

   Zeigen Sie: Wenn *sum* primitiv rekursiv ist, dann ist auch f primitiv rekursiv.

Bitte beachten:
Die folgenden Funktionen dürfen Sie zusätzlich zu den Basisfunktionen als primitiv rekursiv annehmen: *plus* $(+)$, *times* $(\cdot)$, *minus* $(\dot{-})$, *pred*, c, p_1, p_2, *fak* (Fakultät), *twopow*, *tower*, *ifthen*, und c_n^k (konstante k-stellige Funktion mit Ergebnis n).

Die folgenden abgeleiteten Schemata aus der Vorlesung dürfen Sie zusätzlich zum primitiven Rekursionschema verwenden: Beschränkte Maximierung, beschränkter Existenzquantor.

Lösungsvorschlag

1. Mit dem Kompositionsschema definiert man eine Hilfsfunktion:

$$h(r, n) = \mathit{minus}(c_1^2(r, n), \pi_1^2(r, n)) = 1 \dot{-} r$$

   Danach mit primitiver Rekursion:

$$\begin{aligned} \mathit{even}(0) &= 1 \\ \mathit{even}(n+1) &= h(\mathit{even}(n), n) = 1 \dot{-} \mathit{even}(n) \end{aligned}$$

(3P.)

   Die Definition à la $\mathit{even}(n+1) = \mathit{even}(n-1)$ ist nicht PR.

2. Entweder direkt als Rekursion:

$$\begin{aligned} \mathit{half}(0) &= 0 \\ \mathit{half}(n+1) &= n \dot{-} \mathit{half}(n) \end{aligned}$$

oder mit Hilfe von *even*:

$$\begin{aligned} \mathit{half}(0) &= 0 \\ \mathit{half}(n+1) &= \mathit{ifthen}(\mathit{even}(n), \mathit{half}(n), s(\mathit{half}(n))) \end{aligned}$$

oder mit beschränkter Maximierung:

$$\mathit{half}(n) = \max \{x \leq n \mid n \dot{-} 2 \cdot x = 0\} \qquad \text{(2P.)}$$

3. Mit dem erweiterten Schema (allerdings ohne wirkliche Rekursion) können wir f mit Hilfe von *sum* ausdrücken:

$$\begin{aligned} f(0) &= \mathit{sum}(0) \\ f(n+1) &= \mathit{sum}(n+1) \dot{-} \mathit{sum}(n) \qquad (3P.) \end{aligned}$$

Die Argumentation „*sum* ist ja mit f definiert“ ist unzulässig, da es ja auch eine äquivalente Definition ohne f geben könnte. (Vgl. TA8.2.3, HA9.2.2)

Korrekturvorgaben

1. Insgesamt 3P., zusammengesetzt wie folgt:

   2P. für Idee, die sich mit PR umsetzen lässt
   1P. für die technische Umsetzung (Projektionen, Stelligkeit der Fkt., Schemata)

   Bei Verwendung von nicht erlaubten Fkt, je nach Schwere des Falls
   -0.5P. bei Prädikaten $=$, $\leq$, $<$ etc.
   -1P. bei anderen Funktionen
   Insgesamt 0P, wenn dadurch die Aufgabe trivial wird.

   oder

   1P. für eine richtige rekursive Definition, die aber nicht PR ist, auch nicht nach erweitertem Schema. Faustregel: OK in OCaml

   oder

Für LOOP-Programm:	funktioniert	kleiner Fehler	Idee erkennbar	Unsinn
syntakt. korrekt	2	1,5	0,5	0
nicht syntakt. korrekt	1	0,5	0	0

2. 2P., verteilt wie oben, mit folgenden Ausnahmen: Da die technische Umsetzung trivial ist mit EPR, gibt es hier nicht extra Punkte. Bei LOOP-Programm maximal 1,5 Punkte.

3. 3P., davon

   1P. für die (auch vage) Erkenntnis, dass man f aus *sum* "zurückgewinnen" muss (auch punktweise a la "für jedes n ist dann $f(n)$ PR", obwohl falsch/sinnlos).
   1P. für die gleiche Erkenntnis bei klarer Argumentation.
   1P. für die richtige Konstruktion. (-0.5P wenn Sonderfall 0 vergessen).

Der Übergang zwischen Funktionen und Prädikaten ($= 0$, $\neq 0$, genaue Def. von *ifthen* etc) wird immer so gelesen wie er vermutlich gemeint ist.

Aufgabe 5 (8 Punkte)

Zeigen Sie:

1. Wenn $A, B \subseteq \Sigma^*$ beide semi-entscheidbar sind, dann ist auch $A \cap B$ semi-entscheidbar.
2. Die Menge $L_1 = \{w \mid \exists v.\ \varphi_w(v) = vv\}$ ist unentscheidbar.
3. Die Menge $L_2 = \{w \mid M_w[0w1]\downarrow\}$ ist unentscheidbar.
   Hinweis: Verwenden Sie eine Reduktion von H_0.

Lösungsvorschlag

1. Die Funktion $\chi'_{A\cap B}$ lässt sich mit Hilfe von χ'_A und χ'_B wie folgt berechnen:

   Für ein gegebenes w

   (a) Berechne $\chi'_A(w)$.

   (b) Wenn das Ergebnis 1 ist, berechne $\chi'_B(w)$ auf.

   (c) Ist das Ergebnis hier auch positiv, dann gib 1 zurück.

   Damit ist $\chi'_{A\cap B}$ berechenbar, und somit $A \cap B$ semi-entscheidbar. (3P.)

2. Satz von Rice. Die Menge berechenbarer Funktionen $F = \{f \mid \exists v.\ f(v) = vv\}$ ist nicht leer, da es eine Turingmaschine gibt, die ein Wort verdoppeln kann. Sie ist auch nicht universell, da z.B. $\Omega \notin F$. (3P.)

3. Wir reduzieren das Halteproblem auf leerem Band H_0 auf L_2. Gegeben der Code $w \in \Sigma^*$ einer Turingmaschine, konstruieren wir eine Turingmaschine mit folgendem Verhalten:

   - Leere das Band.
   - Simuliere M_w.

   Der Code dieser Turingmaschine sei w'. Für beliebiges v gilt nun:

   $$M_{w'}[v]\downarrow \iff M_w[\epsilon]\downarrow$$

   Das gilt insbesondere für $v = 0w'1$, und wir haben eine Reduktion von H_0 auf L_2. (2P.)

Korrekturvorgaben

1. 3P., davon

1P.	Erkenntnis, dass zu zeigen ist, dass $\chi'_{A\cap B}$ berechenbar ist. (Die Angabe der Definition von $\chi'_{A\cap B}$ genügt nicht.)
0,5P.	Umformung der Bedingung $x \in A \cap B$ zu $x \in A \wedge x \in B$
1,5P.	Für den algorithmischen Teil: Wie berechnet man $\chi'_{A\cap B}$ nun: "ist trivial": 0.5P. (es wurde erkannt, dass da noch etwas zu zeigen ist) vage Beschreibung: 1P. präzise Beschreibung: 1,5P.

oder

Angabe von $\chi'_{A\cap B}$ mit Hilfe von χ'_A und χ'_B ohne Bezug auf Berechenbarkeit: 1,5P.

...mit Bezug auf Berechenbarkeit: 2P.

oder

Bei informeller Argumentationsweise ohne Bezug zu χ':

1P.	"Man muss einen Algorithmus angeben, der hält, wenn ..., nicht hält, wenn ..."
1P.	Für Angabe des Algorithmus.

oder

Bei Argumentation über $A \cap B = L(M)$:

1P.	für Ansatz (wenn richtig beschrieben)
1P.	für Durchführung. Wenn komplett formal sauber auch mehr.

2. 3P., davon

2P.	"Satz v. Rice"
0,5P.	Nennung der Voraussetzungen ($\neq \emptyset$, $\neq \Sigma^*$). Nur "nicht trivial" genügt nicht.
0,5P.	für das Überprüfen der Voraussetzungen.

oder

Bei Reduktion Bewertung wie unten.

3. 2P. Reduktionen genau lesen. Es ist schwer, eine richtige, aber schlechte beschriebene Reduktion von komplettem Unsinn zu unterscheiden!

0.5P.	Es ist erkennbar, welches Problem reduziert wird, und die Richtung stimmt.
1P.	Die Reduktion ist verständlich und funktioniert.
0.5P.	Alles formal sauber.