

MUSTERLÖSUNG

Klausur Informatik I

Nachklausur Wintersemester 2003/2004

14.04.2004

Prof. Dr. G. Goos
Dipl.-Inform. T. Gelhausen

Vorname: _____
Nachname: _____
Matrikelnr.: _____

Aufkleber

Zur Klausur sind keine Hilfsmittel und kein eigenes Papier zugelassen. Die Bearbeitungszeit beträgt 60 Minuten.

Die Klausur ist komplett und geheftet abzugeben. Nur Blätter, die Namen und Matrikelnummer tragen, gehen in die Bewertung ein.

Sie dürfen die Rückseite der Aufgabenblätter als Konzeptpapier benutzen. Nur Lösungen, die sich auf dem entsprechenden Aufgabenblatt oder seiner Rückseite befinden, gehen in die Bewertung ein.

Aufgabe	1	2	3	4	5	6	Σ
Maximal	8	10	7	10	13	12	60
K1							
K2							
K3							

Punkte:

Note:

Aufgabe 1: Endliche Automaten (1+6+1=8P)

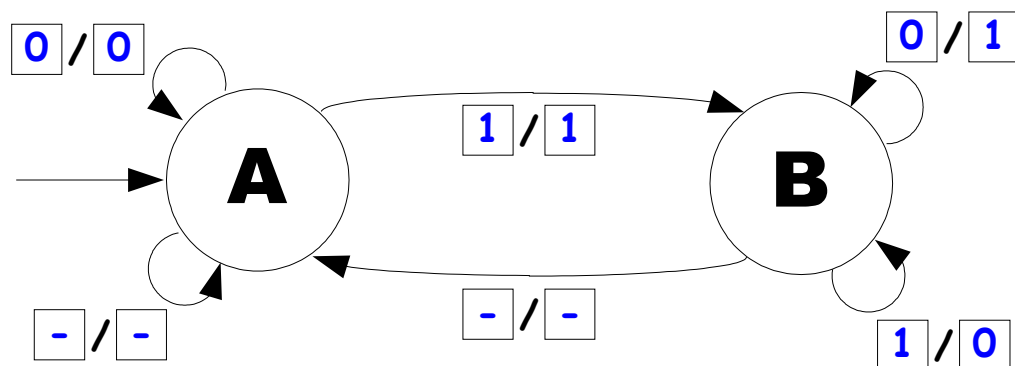
- a.) Ein endlicher Automat wird formal durch ein 5-Tupel definiert. (bitte Anzahl eintragen) (1P)
- b.) Gegeben seien beliebig lange Binärzahlen und das Fragment eines endlichen Automaten. Der Automat verarbeitet die Zahlen, wenn ihm die niederwertigste Stelle zuerst, danach die zweitniederwertigste Stelle (und so weiter) als Eingabe übergeben werden. Jede der Binärzahlen ist mit mindestens einer führenden Null dargestellt. Ergänzen Sie das angegebene Automatenfragment zu einem deterministischen Mealy-Automaten, der das Zweierkomplement einer eingegebenen Binärzahl ausgibt. Schreiben Sie die Eingabe in das Kästchen vor dem Schrägstrich („/“) und die Ausgabe in das Kästchen dahinter. Wenn ein Übergang nicht verwendet werden soll, markieren Sie die Kästchen mit einem Querstrich („-“). (6P)

Hinweis: Richtig ausgefüllte Kästchen ergeben in dieser Aufgabe $\frac{1}{2}$ Punkt, falsch ausgefüllte Kästchen ergeben in dieser Aufgabe $\frac{1}{2}$ Punkt Abzug. Insgesamt wird die Teilaufgabe mit mindestens 0 Punkten bewertet.

Beispiel: $z=0011$ ergibt als Eingabe [1, 1, 0, 0]

Legende: - / - bedeutet „diese Kante wird nicht verwendet“

a / b bedeutet „bei der Eingabe a wird b ausgegeben“



- c.) Im Automat aus Aufgabenteil b.) wurden noch keine Endzustände gekennzeichnet. Welche müssen es sein? (bitte zutreffendes ankreuzen) (1P)

- Zustand 1
- Zustand B
- Zustand 0
- Zustand A
- Keiner der Zustände darf ein Endzustand sein!

Aufgabe 2: Prolog (2+2+2+3+1=10P)

Gegeben seien endlich viele Prolog-Fakten der Form

Faktum	Bedeutung
$p(X)$	<i>X ist eine Person</i>
$k(X)$	<i>X ist eine Katze</i>
$m(X)$	<i>X ist männlich</i>
$w(X)$	<i>X ist weiblich</i>
$e(X, Y)$	<i>X ist direkter Vorfahre (Eltern) von Y</i>

Beispiel: $p(\text{otto}) . e(\text{ulla}, \text{otto}) . p(\text{Ulla}) . m(\text{otto}) . k(\text{miau}) .$
 $m(\text{hans}) . e(\text{hans}, \text{otto}) . w(\text{miau}) . p(\text{hans}) . w(\text{ulla}) .$

Hinweise: Sie dürfen außer `not()` keine Bibliotheksfunktionen verwenden. Sie dürfen jedoch annehmen, dass der Datenbasis die übliche Biologie der Säugetiere zugrunde liegt, daß also insbesondere maximal 2 Vorfahren für jedes Subjekt (Person oder Katze) eingetragen sind und daß die Eltern-Relation nicht reflexiv ist.

- a.) Geben Sie eine Klausel `verschieden(X, Y, Z)` an, die dann zu Wahr ausgewertet wird, wenn es sich bei X, Y und Z um **drei verschiedene Personen** handelt. (2P)

**verschieden(X, Y, Z) :- p(X), p(Y), p(Z),
not(X==Y), not(Y==Z), not(X==Z).**

+1P wenn es sich um 3 Personen handelt
+1P wenn X, Y und Z verschieden sind
-½P für leichte, -1P für schwere Syntaxfehler

- b.) Geben Sie eine Klausel `hatSchwester(X)` an, welche genau dann Wahr wird, wenn X eine Schwester hat, also ein direkter Vorfahre ein weiteres, weibliches Kind hat. Ihre Klausel soll für Menschen und Katzen funktionieren. (2P)

hatSchwester(X) :- e(E, X), e(E, Y), not(X==Y), w(Y).

+1P für Eltern mit 2 Kindern
+½P wenn Kinder verschieden
+½P wenn richtiges Kind weiblich
-½P für leichte, -1P für schwere Syntaxfehler

- c.) Geben Sie eine Klausel `hatHalbschwester(X)` an, welche genau dann Wahr wird, wenn X eine Halbschwester hat. (Echte Schwestern haben 2 gemeinsame Vorfahren, Halbschwestern nicht) (2P)

Hinweise: Sie dürfen in dieser Teilaufgabe die Klauseln `verschieden(X, Y, Z)` und `hatSchwester(X)` aus den Teilaufgaben a.) und b.) verwenden. Ihre Klausel braucht nur für Menschen zu funktionieren.

**`hatHalbschwester(X) :- e(E1,X), e(E2,X),
e(E2,Y), e(E3,Y), w(Y),
verschieden(E1,E2,E3).`**

+ $\frac{1}{2}$ P für 3 Eltern, + $\frac{1}{2}$ P wenn Eltern verschieden
+ $\frac{1}{2}$ P für 2 Kinder („not(X==Y)“ unnötig wegen „üblicher Biologie“)
+ $\frac{1}{2}$ P wenn richtiges Kind weiblich
- $\frac{1}{2}$ P für leichte, -1P für schwere Syntaxfehler

- d.) Geben Sie eine Klausel `minDreiTanten(X)` an, mit der alle Personen der Datenbasis ermittelt werden können, die mindestens 3 Tanten haben. (3P)

Hinweise: Beachten Sie bitte, dass auch Halbschwestern Tantenbeziehung erzeugen können! Sie dürfen in dieser Teilaufgabe die Klauseln `verschieden(X, Y, Z)`, `hatSchwester(X)` und `hatHalbschwester(X)` aus den Teilaufgaben a.), b.) und c.) verwenden. Ihre Klausel braucht nur für Menschen zu funktionieren.

**`tante(T,X) :- e(E,X), e(GE,E), e(GE,T), w(T), not(E==T).
minDreiTanten(X) :- tante(T1,X), tante(T2,X), tante(T3,X),
verschieden(T1,T2,T3).`**

+1P für richtige Tantenbeziehung über Großeltern (GE)
+ $\frac{1}{2}$ P wenn Tante≠Eltern
+ $\frac{1}{2}$ P wenn Tante weiblich
+ $\frac{1}{2}$ P für 3 Tanten
+ $\frac{1}{2}$ P wenn die Tanten unterschiedlich sind
- $\frac{1}{2}$ P für leichte, -1P für schwere Syntaxfehler
insges. 1P für syntaktisch korrekte Angabe einer Klausel die bei Vorhandensein einer Tante (unter Verwendung von `hatSchwester()`) Wahr ausgibt

- e.) Beschreiben Sie kurz, wie man alle Personen der Datenbasis ermittelt, die mindestens drei Tanten haben. Geben Sie auch die Aufrufsyntax mit an! (1P)

**?- minDreiTanten(X).
Diese Anfrage gibt eine Person aus. Weitere Personen
werden durch die Eingabe von ";" ausgegeben.**

+ $\frac{1}{2}$ P für korrekte Syntax
+ $\frac{1}{2}$ P für ";"

Aufgabe 3: Ausdrücke (3+2+2=7P)

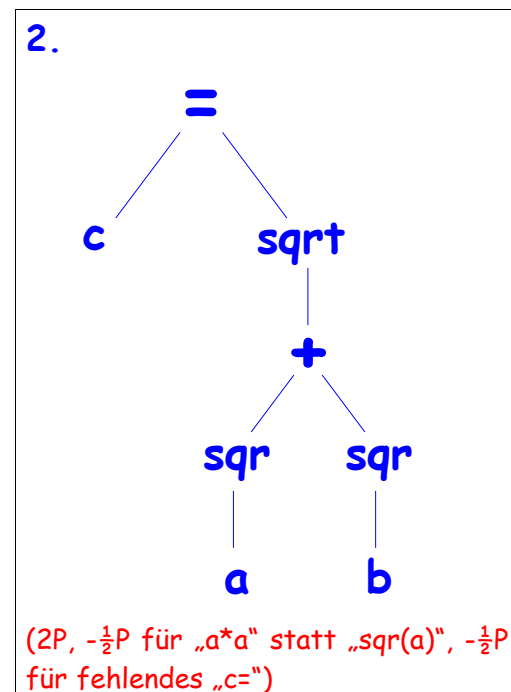
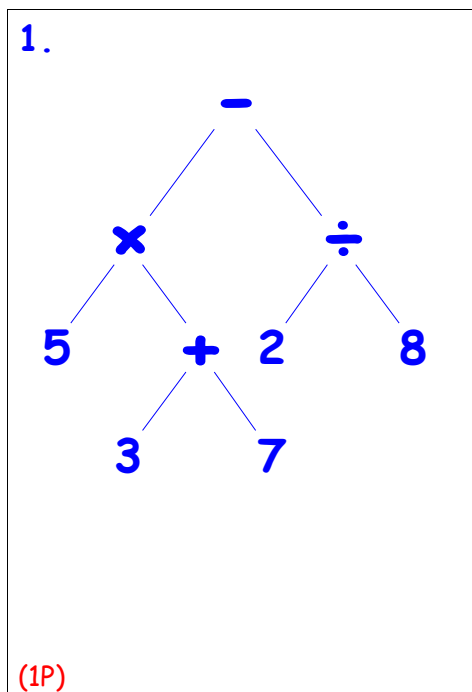
Gegeben seien folgende mathematische Ausdrücke mit den üblichen Vorrangregeln:

1. $5 * (3 + 7) - \frac{2}{8}$

2. $c = \sqrt{a^2 + b^2}$

Hinweise: Führen Sie ggf. neue Funktionsnamen ein und dokumentieren Sie diese so kurz wie möglich. Tipp: Zählen Sie sich als erstes die Operatoren bzw. Funktionsnamen auf, die in den beiden Formeln vorkommen.

a.) Geben Sie zu den Ausdrücken die Kantorowitsch-Bäume an. (3P)



b.) Nach welchem Schema müssen Sie den Kantorowitsch-Baum des Ausdrucks ablaufen und die Knoten ausgeben, um einen Term in **Postfixnotation** zu erhalten? Geben Sie die Postfixnotation für den 1. Ausdruck an. (2P)

Links-Abwärts-Durchlauf, Ausgabe beim Verlassen jedes Teilbaums(1P): 5 3 7 + * 2 8 / - (1P, -½P Verwechslung Prä/Post)

c.) Nach welchem Schema müssen Sie den Kantorowitsch-Baum des Ausdrucks ablaufen und die Knoten ausgeben, um einen Term in **Präfixnotation** zu erhalten? Geben Sie die Präfixnotation für den 2. Ausdruck an. (2P)

Links-Abwärts-Durchlauf, Ausgabe beim Betreten jedes Teilbaums(1P): = c sqrt + sqr a sqr b (1P, -½P Verw. Prä/Post)

Aufgabe 4: Boolesche Algebra (9+1=10P)

Sie haben eine Tutorenstelle für ein Informatik-I-Tutorium ergattert und Sie korrigieren gerade Übungsblätter. Auf dem aktuellen Blatt waren Formeln mittels Boolescher Algebra zu transformieren. Ein Student hat Ihnen ein etwas unhandliches Formelmonster abgegeben, welches Sie jetzt korrigieren müssen:

$$\begin{aligned}
 \overline{\overline{a}} &=^{(1)} \overline{\overline{a \wedge \top}} \\
 &=^{(2)} \overline{\overline{a} \wedge (\overline{\overline{a}} \vee a)} \\
 &=^{(3)} (\overline{\overline{a} \wedge \overline{\overline{a}}}) \vee (\overline{\overline{a} \wedge a}) \\
 &=^{(4)} \perp \vee (\overline{\overline{a} \wedge a}) \\
 &=^{(5)} (\overline{a} \wedge a) \vee (\overline{\overline{a} \wedge a}) \\
 &=^{(6)} (a \wedge \overline{\overline{a}}) \vee (a \wedge \overline{\overline{\overline{a}}}) \\
 &=^{(7)} a \wedge (\overline{a} \vee \overline{\overline{\overline{a}}}) \\
 &=^{(8)} a \wedge \top \\
 &=^{(9)} \top
 \end{aligned}$$

- a.) Welche der Umformungsschritte 1-9 sind korrekt und welche sind fehlerhaft? Geben Sie bei korrekten Umformungen die verwendeten Axiome an, bei fehlerhaften zusätzlich die korrigierte Version der Zeile. (9P)

Schritt Nr.	korrekter Schritt	fehlerhafter Schritt	
			Angewendete Axiome und evtl. korrigierte Zeile
1	X		Neutrale Elemente (V7)
2	X		Komplement (V8)
3	X		Distributivität (V5)
4	X		Komplement (V8)
5	X		Komplement (V8)
6	X		Kommutativität (V2)
7	X		Distributivität (V5)
8	X		Komplement (V8)
9		X	Neutrale Elemente (V7), "a" statt "T"

(1P pro richtiger Zeile, $\frac{1}{2}$ P wenn nur eine Gleichung angegeben wurde (z.B. „ $avb=bva$ “ statt „Kommutativität“), Angabe der Nummer der Gesetze nicht notwendig, keine Punkte für Kreuzel)

- b.) Welches Gesetz der booleschen Algebra hat der Student zu beweisen versucht? (1P)

Das Involutionsgesetz. (OP für „ $\overline{\overline{a}}=a$ “!)

Aufgabe 5: Haskell (5+1+4+3=13P)

Auf seine alten Tage ist Ihr Computer vergesslich geworden, außerdem fängt er an zu phantasieren. Das schöne Programm, das Sie Ihrem Tutor vorlegen (s/w)ollten, weist lauter Löcher auf und Ihren Sortieralgorithmus hat er willenlos um alberne Zeilen ergänzt.

Hinweis: Die Programme in dieser Aufgabe müssen den Haskell-Regeln (wie von Hugs98 akzeptiert) entsprechen.

- a.) Ergänzen Sie folgendes Programmfragment so, dass es das zweitkleinste Element einer übergebenen, mindestens zweielementigen Liste zurückgibt. Fügen Sie hierfür die passenden Operatoren und Operanden in die Kästchen ein. Die Suche soll „stabil“ sein, also dasjenige Element zurückgeben, welches als erstes in der Liste auftaucht und den zweitniedrigsten Wert besitzt. (5P)

```

zweitkleinste (x:y:xs) | x <= y = zk x y xs
                       | otherwise = zk y x xs

zk x y [] = y
zk x y (w:ws) | w < x = zk w ws
               | w <= y = zk x y ws
               | otherwise = zk x y ws
    
```

(Alternative)

- b.) In welcher Klasse liegt der Aufwand des Programms aus Aufgabenteil a.)? Machen Sie eine möglichst genaue Aussage! (1P)

$\Theta(n)$ ($O(n)$ oder $o(n) \rightarrow \frac{1}{2}P$)

- c.) Wählen Sie aus den folgenden Zeilen so (durch ankreuzen) aus, dass das sich ergebende Programm den Quicksort-Algorithmus implementiert, der übergebene Listen aufsteigend sortiert. (4P)

Hinweis: Richtig ausgefüllte Kästchen ergeben in dieser Teilaufgabe ½ bzw. 1 Punkt, falsch ausgefüllte Kästchen ergeben in dieser Teilaufgabe ½ bzw. 1 Punkt Abzug. Insgesamt wird die Teilaufgabe mit mindestens 0 Punkten bewertet.

```

qsort [] = []
qsort (x:xs) = qsort (
    {
 map
 filter
 take
 drop
 put
    }
    {
 (x!=y)
 (x<(fst xs))
 (<x)
 (>=x)
 (x>=(fst xs))
    }
    {
 qsort xs
 (x:xs)
 xs
    }
)

{
 ++ x ++
 : [x] :
 ++ xs ++
 : [xs] :
 ++ [x] ++
}

qsort (
    {
 map
 filter
 take
 drop
 put
    }
    {
 (x!=y)
 (x<(snd xs))
 (>=x)
 (x>=(snd xs))
    }
    {
 qsort xs
 (x:xs)
 xs
    }
)
    
```

- d.) Schreiben Sie eine Haskell-Funktion `mean`, die zu einer übergebenen Liste von Zahlen den Mittelwert berechnet. Die Liste soll dabei maximal 1-mal komplett durchlaufen werden. (3P, wenn Ihre Lösung die Liste öfter als 1-mal durchläuft max. 2P)

```
mean [] = 0
mean liste = summe / (fromInteger anzahl)
  where (summe,anzahl) = hilf liste

hilf [] = (0,0)
hilf (x:xs) = (summe+x,anzahl+1)
  where (summe,anzahl) = hilf xs
```

+1P für das Berechnen des Mittelwertes
+ $\frac{1}{2}$ P für Randfälle (0, [])
+ $\frac{1}{2}$ P für `fromInteger`
+1P für eine Lösung, die nur 1 mal durch die Liste läuft

Aufgabe 6: Strukturelle Induktion (12P)

Aus welchen Eigenschaften einer Funktion f folgt, dass

$$\text{foldl } f \ z = \text{foldr } f \ z$$

gilt? Geben Sie eine möglichst kleine Menge von Eigenschaften an und beweisen Sie mittels Struktureller Induktion, dass diese Eigenschaften *hinreichend* sind. Sie brauchen allerdings nicht zu beweisen, dass die von Ihnen geforderten Eigenschaften im (mathematischen Sinne) *notwendig* sind.

Erinnerung: Die Definitionen von `foldl` und `foldr` lauteten:

```

foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f z [] = z
foldl f z (x:xs) = foldl f (f z x) xs
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f z [] = z
foldr f z (x:xs) = f x (foldr f z xs)

```

Hinweise: Stellen Sie zuerst eine Behauptung über die Vertauschbarkeit der Anwendungsreihenfolge einer Faltungsoperation und der Funktion f auf und beweisen Sie diese.

Wenden Sie pro Beweisschritt (Folgerung oder Äquivalenz) nur ein Gesetz/Satz/Axiom an und dokumentieren Sie dies (vgl. Aufgabe 4).

Behauptung: $\text{foldl } f \ z = \text{foldr } f \ z$ gilt, wenn f kommutativ (1P) und assoziativ (1P) ist.

(Alternativ auch: assoziativ, neutrales Element und Liste endlich, siehe Goos, Vorlesung über Informatik Bd. 1, Aufgabe 5.23)

Induktionsanfang:

$$\text{foldl } f \ z \ [] =^{(D)} z =^{(D)} \text{foldr } f \ z \ []. \quad (1P)$$

Induktionsvoraussetzung:

Die Behauptung gelte für alle Listen der Länge n für ein beliebiges aber festes $n \geq 1$. ($\frac{1}{2}$ P)

Induktionsschritt $n = n+1$:

$$\begin{aligned}
 \text{foldr } f \ z \ (x:xs) &=^{(D)} f \ x \ (\text{foldr } f \ z \ xs) \\
 &=^{(IV)} f \ x \ (\text{foldl } f \ z \ xs) \\
 &=^{(HB)} \text{foldl } f \ z \ (x:xs). \quad (2P)
 \end{aligned}$$

Hilfsbehauptung: Die Anwendungsreihenfolge von foldl und f darf vertauscht werden, wenn f kommutativ und assoziativ ist, also: $\text{foldl } f \ z \ (x:xs) = f \ x \ (\text{foldl } f \ z \ xs)$ (1P)

Induktionsanfang:

$$\begin{aligned} \text{foldl } f \ z \ [x] &= \text{(D)} \text{ foldl } f \ (f \ z \ x) \ [] \\ &= \text{(D)} \ f \ z \ x \\ &= \text{(K)} \ f \ x \ z \\ &= \text{(D)} \ f \ x \ (\text{foldl } f \ z \ []). \text{ (1P)} \end{aligned}$$

$$\begin{aligned} \text{foldl } f \ z \ [x,y] &= \text{(D)} \text{ foldl } f \ (f \ z \ x) \ [y] \\ &= \text{(D)} \text{ foldl } f \ (f \ (f \ z \ x) \ y) \ [] \\ &= \text{(D)} \ f \ (f \ z \ x) \ y \\ &= \text{(A)} \ f \ z \ (f \ x \ y) \\ &= \text{(D)} \ f \ z \ (\text{foldl } f \ (f \ x \ y) \ []) \\ &= \text{(D)} \ f \ z \ (\text{foldl } f \ x \ [y]). \text{ (1P)} \end{aligned}$$

Induktionsvoraussetzung:

Die Hilfsbehauptung gelte für alle Listen der Länge n für ein beliebiges aber festes $n \geq 1$. ($\frac{1}{2}$ P)

Induktionsschritt $n = n+1$:

$$\begin{aligned} \text{foldl } f \ z \ (x:y:xs) &= \text{(D)} \text{ foldl } f \ (f \ z \ x) \ (y:xs) \\ &= \text{(K)} \text{ foldl } f \ (f \ x \ z) \ (y:xs) \\ &= \text{(IV)} \ f \ (f \ x \ z) \ (\text{foldl } f \ y \ xs) \\ &= \text{(A)} \ f \ x \ (f \ z \ (\text{foldl } f \ y \ xs)) \\ &= \text{(IV)} \ f \ x \ (\text{foldl } f \ z \ (y:xs)). \text{ (3P)} \end{aligned}$$

(K) = Kommutativität von f

(A) = Assoziativität von f

(IV) = nach Induktionsvoraussetzung

(D) = nach Definition von foldl bzw. foldr

(HB) = nach Hilfsbehauptung