

Aufgaben zur Übergangsprüfung Grundlagen der Programmierung im WS 02/03  
Zeit: 60 Minuten, erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den freien Stellen nach den jeweiligen Aufgaben ein (ggf. auf der jeweiligen Rückseite weiterschreiben). Sollten Unklarheiten oder Mehrdeutigkeiten bei der Aufgabenstellung auftreten, so notieren Sie bitte, wie Sie die Aufgabe interpretiert haben.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 5 Seiten.

### Aufgabe 1.

Gegeben sei folgende aussagenlogische Formel:  $((\neg A \vee \neg B) \rightarrow \neg (A \vee B))$

- Transformieren Sie die Formel in eine äquivalente Formel in Klauselform.
- Prüfen Sie, ob die Formel erfüllbar ist.
- Ist die Formel auch tautologisch? Bitte begründen Sie Ihre Antwort.

Lösungsvorschlag:

a)

1. Die Formel muß in KNF transformiert werden:

$$\begin{aligned} & ((\neg A \vee \neg B) \rightarrow \neg (A \vee B)) \equiv \\ & (\neg(\neg A \vee \neg B) \vee \neg (A \vee B)) \equiv \\ & ((A \wedge B) \vee (\neg A \wedge \neg B)) \equiv \\ & ((A \vee (\neg A \wedge \neg B)) \wedge (B \vee (\neg A \wedge \neg B))) \equiv \\ & ((A \vee \neg A) \wedge (A \vee \neg B) \wedge (B \vee \neg A) \wedge (B \vee \neg B)) \equiv \\ & \text{(ist bereits KNF, man kann aber weiter vereinfachen)} \\ & (T \wedge (A \vee \neg B) \wedge (B \vee \neg A) \wedge T) \equiv \\ & \mathbf{((A \vee \neg B) \wedge (B \vee \neg A))} \end{aligned}$$

2.  $((A \vee \neg B) \wedge (B \vee \neg A))$  in Klauselform:

$$\mathbf{\{\{A, \neg B\}, \{B, \neg A\}\}}$$

b)  $((\neg A \vee \neg B) \rightarrow \neg (A \vee B)) \equiv$

(s. die Umformung unter a)1. )

$$((A \wedge B) \vee (\neg A \wedge \neg B)) \equiv$$

$$\mathbf{A \leftrightarrow B}$$

Wahrheitstabelle für Biimplikation:

A	B	$A \leftrightarrow B$
F	F	W
F	W	F
W	F	F
W	W	W

Die Formel ist erfüllbar (es gibt eine Belegung (z.B.  $A=F, B=F$ ), die diese Formel wahr macht)

c) Die Formel ist nicht tautologisch (es gibt eine Belegung (z.B.  $A=F, B=W$ ), für die diese Formel den Wert F hat).

## Aufgabe 2.

- a) Geben Sie die Spezifikation für eine Funktion an, die als Eingabe ein Array der Länge  $N$  mit natürlichen Zahlen bekommt und bestimmt, ob in dem Eingabearray keine doppelten Elemente vorkommen.
- b) Geben Sie ein While-Programm zur Implementierung der unter a) spezifizierten Funktion an.
- c) Schätzen Sie die asymptotische Komplexität Ihres unter b) angegebenen Programmes ab.

Lösungsvorschlag:

a)

1. Variante:  $\forall 1 \leq i \leq N . \forall 1 \leq j \leq N . (i \neq j \rightarrow a[i] \neq a[j])$
2. Variante :  $\forall 1 \leq i < N . \forall i + 1 \leq j \leq N . a[i] \neq a[j]$

b) While-Implementation der 2. Variante (mit einer kleinen Optimierung: vorzeitiger Abbruch beim Finden zweier gleicher Elemente)

```
pruefe-doppelte (a: array [1 .. N] of N0): Bool
  var i, j : N0, doppelt: Bool;
  i, doppelt := 1, false;
  while (i < N ∧ ¬doppelt) do
    j := i + 1;
    while (j ≤ N ∧ ¬doppelt) do
      doppelt := a[i] = a[j];
      j := j + 1;
    end while
    i := i + 1;
  end while
  doppelt
```

c) Die Komplexität ist im schlimmsten Fall  $O(N^2)$  wegen beider geschachtelten Schleifen der Länge  $N$  bzw.  $N/2$ .

### Aufgabe 3.

Es seien Säugetier, Fleischfresser, Pflanzenfresser, Mensch, Pflanze und Vegetarier einstellige Prädikatensymbole. Weiterhin sei  $x$  eine Variable und tom eine Konstante.

Mit diesen syntaktischen Einheiten läßt sich eine Formelmengende  $F$  bilden:

$$F := \{ \forall x (\text{Fleischfresser}(x) \rightarrow \text{Säugetier}(x) \vee \text{Pflanze}(x)), \quad (1)$$

$$\forall x (\text{Pflanzenfresser}(x) \rightarrow \text{Säugetier}(x)), \quad (2)$$

$$\forall x (\text{Mensch}(x) \rightarrow (\text{Fleischfresser}(x) \vee \text{Pflanzenfresser}(x))), \quad (3)$$

$$\forall x (\text{Vegetarier}(x) \rightarrow (\text{Mensch}(x) \wedge \neg \text{Fleischfresser}(x))), \quad (4)$$

$$\text{Vegetarier}(\text{tom}) \} \quad (5)$$

- a) Argumentieren Sie, daß die Formel  $\text{Säugetier}(\text{tom})$  eine Folgerung aus  $F$  ist.
- b) Geben Sie ein beliebiges Modell für die Formelmengende  $F$  an.

Lösungsvorschlag:

a)

$$\text{Aus (5) und (4) folgt: } (\text{Mensch}(\text{tom}) \wedge \neg \text{Fleischfresser}(\text{tom})) \quad (6)$$

$$\text{Aus (6) und (3) folgt: } (\text{Fleischfresser}(\text{tom}) \vee \text{Pflanzenfresser}(\text{tom})) \quad (7)$$

$$\text{Aus (6) und (7) folgt: } \text{Pflanzenfresser}(\text{tom}) \quad (8)$$

$$\text{Aus (8) und (2) folgt: } \mathbf{\text{Säugetier}(\text{tom})}$$

b)

Sei  $A = (U_A, I_A)$  ein Modell von  $F$ .

$U_A = \{\text{tom}\}$  und die Interpretationsfunktion  $I_A$  sei wie folgt definiert:

$$\text{Fleischfresser}^A = \{\}$$

$$\text{Säugetier}^A = \{(\text{tom})\}$$

$$\text{Pflanze}^A = \{\}$$

$$\text{Pflanzenfresser}^A = \{(\text{tom})\}$$

$$\text{Mensch}^A = \{(\text{tom})\}$$

$$\text{Vegetarier}^A = \{(\text{tom})\}$$

#### Aufgabe 4.

Leider passierte es dem Professor Schusselig immer wieder, daß er vergißt, welche Aufgaben er den Studierenden gegeben hat. Leider sind die Lösungen auch nicht immer leicht zu verstehen. Nun hält er folgende Lösung des Studierenden Schlau in der Hand:

```
finde-index (a: array [1..N] of N0, i,j: N0): N0
  var m1, m2 : N0
  if i ≥ j
    then i
    else m1, m2 := finde-index(a, i, i+((j-i) div 2)), finde-index(a, i+((j-i) div 2) + 1, j);
      if a[m1] > a[m2]
        then m1
        else m2
      end if
  end if
```

```
f (a: array [1..N] of N0): N0
  a[finde-index(a,1,N)]
```

- Um welche Art der Rekursion handelt es sich bei der Definition der Funktion finde-index?
- Nehmen wir an, N sei 3. Weiterhin sei das Array b: array [1..N] of N<sub>0</sub> wie folgt initialisiert: b := (100, 42, 101). Geben Sie die einzelnen Schritte zur Auswertung des Ausdrucks f(b) an.
- Analysieren Sie Ihre Auswertung und helfen Sie dem Professor Schusselig. Welche Aufgabe hatte er gestellt, d.h. was sollte die Funktion f berechnen?
- Ist der Algorithmus des Studierenden Schlau in Hinblick auf die asymptotische Komplexität des Problems, das er lösen soll, optimal? Bitte begründen Sie Ihre Antwort.

Lösung:

#### a) Baumrekursion

b) Auswertung von f((100,42,101)) führt zur Auswertung von finde-index((100,42,101), 1, 3)

```
finde-index ((100,42,101), 1,3)
  var m1', m2' : N0
  if 1 ≥ 3
    then 1
    else m1', m2' :=
      finde-index((100,42,101), 1, 2),
      finde-index((100,42,101), 3, 3);
      if (100,42,101)[m1'] > (100,42,101)[m2']
        then m1'
        else m2'
      end if
  end if
```

```

finde-index ((100,42,101), 1,2)
  var m1", m2" : N0
  if 1 ≥ 2
  then 1
  else m1", m2" :=
      finde-index((100,42,101), 1, 1),
      finde-index((100,42,101), 2, 2);
  if (100,42,101)[m1"] > (100,42,101)[m2"]
  then m1"
  else m2"
  end if
end if

```

```

finde-index ((100,42,101), 1,1)
  var m1"', m2"' : N0
  if 1 ≥ 1
  then 1
  else m1"', m2"' :=
      finde-index((100,42,101), 1, 1),
      finde-index((100,42,101), 2, 1);
  if (100,42,101)[m1"' ] > (100,42,101)[m2"' ]
  then m1"'
  else m2"'
  end if
end if

```

Zwischenergebnis: m1"=1

```

finde-index ((100,42,101), 2,2)
  var m1"', m2"' : N0
  if 2 ≥ 2
  then 2
  else m1"', m2"' :=
      finde-index((100,42,101), 2, 2),
      finde-index((100,42,101), 3, 2);
  if (100,42,101)[m1"' ] > (100,42,101)[m2"' ]
  then m1"'
  else m2"'
  end if
end if

```

Zwischenergebnis: m2"=2

(100,42,101)[1] > (100,42,101)[2]

Ergebnis der Auswertung von finde-index ((100,42,101), 1,2) ist 1.

m1'=1

```

finde-index((100,42,101), 3, 3)

```

```

  var m1''', m2'''' : N0
  if 3 ≥ 3
  then 3

```

```

else m1''''', m2'''' :=
    finde-index((100,42,101), 3, 3),
    finde-index((100,42,101), 4, 3);
if (100,42,101)[m1'''''] > (100,42,101)[m2''''']
then m1'''''
else m2'''''
end if
end if

```

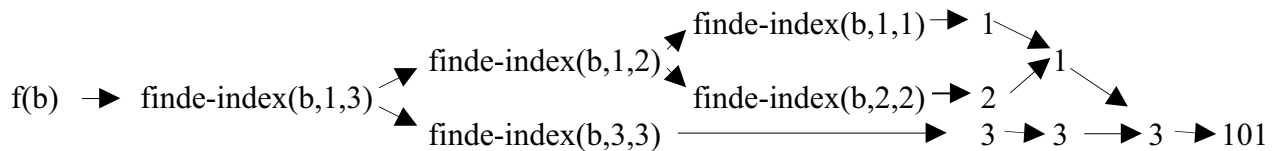
Zwischenergebnis: m2''''=3.

m2'=3

(100,42,101)[1] < (100,42,101)[3], deshalb wird finde-index ((100,42,101), 1,3) zu 3 ausgewertet.

Damit ist das Gesamtergebnis der Auswertung von f((100,42,101))=b[3]=**101**.

Die Auswertungsfolge noch einmal zusammenfassend:



Anmerkung: Antworten in dieser kurzen Form wurden auch akzeptiert.

- c) Die Funktion f soll das maximale Element im Array bestimmen.
- d) Der vorgestellte Algorithmus ist vom Typ "Teile und Herrsche" und hat die Komplexität  $O(N \lg N)$ . Die Suche nach dem größten Element in einem Array der Größe N ist aber linear, also  $O(N)$ . Der Algorithmus ist offensichtlich.