185.A03 Funktionale Programmierung WS 2020/2021

Schriftlicher Online-Test 1 auf Papier

Donnerstag, 14.01.2021, 16:00–18:00 Uhr

Aufgabe	1	2	3	4	5	6	7	8	Summe Punkte
Punkte	18	10	18	9	7	12	8	18	100
Erreichte Punkte									

Vergessen Sie nicht, die unterfertigte eideststattliche Erklärung mit abzugeben!

Aufgabe 1 ((1+1)+10+6 = 18 Punkte)

Zwei Zeichenreihen heißen *umstellungsgleich* gdw.: die beiden Zeichenreihen können durch Ändern der Reihenfolge ihrer Zeichen ineinander überführt werden.

1. Geben Sie je ein Paar von Zeichenreihen an, die
   (a) umstellungsgleich
   (b) nicht umstellungsgleich

   sind.
2. Schreiben Sie eine uncurryfizierte Haskell-Wahrheitswertfunktion `sind_ug` einschließlich ihrer syntaktischen Signatur, die angewendet auf ein Paar von Zeichenreihen bestimmt, ob die Zeichenreihen umstellungsgleich sind.
3. Erklären Sie knapp, aber gut nachvollziehbar, wie ihre Wahrheitswertfunktion vorgeht.

Aufgabe 2 (4+6 = 10 Punkte)

Gegeben sind folgende Typ- und Instanzdeklarationen:

```
type Info a = [a]
data Baum a = B (Info a)
            | K (Baum a) (Info a,Info a) (Baum a)

instance (Ord a,Show a) => Eq (Baum a) where
 B xs       == B ys          = xs == ys
 B _        == K _ _ _       = False
 K _ _ _    == B _           = False
 K b1 ps b2 == K b1' ps' b2' = b1 == b1' && ps == ps' && b2 == b2'
```

Geben Sie jeweils das Zutreffende an und ergänzen Sie anstelle der Auslassungspunkte (...) die Begründungen: In der Instanzdeklaration für `Baum a`

1. kann die Kontextbedingung `(Ord a,Show a) =>`
   (a) weggelassen werden
   (b) nicht weggelassen werden
   (c) nicht weggelassen werden, aber vereinfacht werden bis hin zu ...

   weil ...
2. haben die verschiedenen Vorkommen von `==`
   (a) dieselbe
   (b) nicht dieselbe

   Bedeutung, weil ... und sind dabei jeweils vom Typ ... bzw. den Typen ...

Aufgabe 3 (6+6+6 = 18 Punkte)

1. Was haben normale und faule (Ausdrucks-) Auswertung gemeinsam, worin stimmen sie überein?
2. Was unterscheidet sie, worin stimmen sie nicht überein?
3. Welche Gewissheiten ergeben sich für den Programmierer aus den Gemeinsamkeiten und Unterschieden für die Programmierung in einer funktionalen Programmiersprache mit fauler Auswertung?

Aufgabe 4 (3+3+3 = 9 Punkte)

Gegeben ist die Funktion `f`:

```
f x y z = if (((x*x) - 42) > 0) then x * (y+y) else x*y*z
```

und der Funktionsterm: `f (5*(3+2)) (2+3*5) (3+5)`.

Wie oft wird der Ausdruck

1. `(5*(3+2))` bei
   (a) applikativer
   (b) normaler
   (c) fauler

   Auswertung von `f (5*(3+2)) (2+3*5) (3+5)` ausgewertet?
2. `(2+3*5)` bei
   (a) applikativer
   (b) normaler
   (c) fauler

   Auswertung von `f (5*(3+2)) (2+3*5) (3+5)` ausgewertet?
3. `(3+5)` bei
   (a) applikativer
   (b) normaler
   (c) fauler

   Auswertung von `f (5*(3+2)) (2+3*5) (3+5)` ausgewertet?

Aufgabe 5 (3+2+2 = 7 Punkte)

1. Welches ist der allgemeinste Typ von `f` aus Aufgabe 4? Begründen Sie Ihre Antwort.
2. Ist der allgemeinste Typ von `f` aus Aufgabe 4 monomorph oder polymorph? Falls polymorph, geben Sie die Polymorphieart möglichst genau an und erklären Sie, woran die Polymorphieart hier zu erkennen ist.
3. Welches ist der allgemeinste Typ von `f` aus Aufgabe 4, wenn die Bedingung `(((x*x) - 42) > 0)` im Fallunterscheidungsausdruck durch `(((x*x) - 42) /= 0)` ersetzt wird? Begründen Sie Ihre Antwort.

Aufgabe 6 (4+4+4= 12 Punkte)

Die *charakteristische Funktion* $\chi_M : \mathbb{Z} \to \{wahr, falsch\}$ einer Menge ganzer Zahlen M ist definiert durch:

$$\chi_M(z) =_{df} \begin{cases} wahr & \text{falls } z \in M \\ falsch & \text{falls } z \notin M \end{cases}$$

Eine charakteristische Funktion repräsentiert damit auf eindeutige Weise eine Menge ganzer Zahlen $M_\chi \subseteq \mathbb{Z}$ definiert durch:

$$M_\chi =_{df} \{z \in \mathbb{Z} \,|\, \chi(z) = wahr\}$$

Mengen ganzer Zahlen können deshalb durch ihre charakteristische Funktionen repräsentiert und modelliert werden. In Haskell ist das auf folgende Weise möglich:

```
type Zett     = Integer
type CharFunk = Zett -> Bool
type Menge    = CharFunk
```

Schreiben Sie Haskell-Rechenvorschriften für folgende Mengenoperationen:

1. Elementeigenschaft: $z \in M$
   `ist_Element_von :: Zett -> Menge -> Bool`
2. Mengenvereinigung: $M_1 \cup M_2 =_{df} \{m \,|\, m \in M_1 \vee m \in M_2\}$
   `vereinigung :: Menge -> Menge -> Menge`
3. Mengendifferenz: $M_1 \backslash M_2 =_{df} \{m \,|\, m \in M_1 \wedge m \notin M_2\}$
   `differenz :: Menge -> Menge -> Menge`

Die Funktionen `vereinigung` und `differenz` müssen so definiert sein, dass ihre Resultate unmittelbar als Mengenargument von `ist_Element_von` verwendet werden können.

Aufgabe 7 (4+4 = 8 Punkte)

Mit den Bezeichnungen und Typen von Aufgabe 6: Implementieren Sie folgende Mengenoperationen oder begründen Sie informatisch präzise, wenn das nicht möglich ist, warum nicht:

1. Komplementmenge zu einer Grundmenge G: $\mathcal{K}_G(M) =_{df} \{m \in G \,|\, m \notin M\}$
   `komplement :: Menge -> Menge` (zur Grundmenge `Z`)
2. Mengengleichheit: $M_1 = M_2 =_{df} m \in M_1 \Leftrightarrow m \in M_2$
   `sind_gleich :: Menge -> Menge -> Bool`

Aufgabe 8 (6+4+3+1+(2+2) = 18 Punkte)

Gegeben ist die Menge $\mathbb{Z}$ ganzer Zahlen, die Betragsfunktion $|\cdot|$ auf $\mathbb{Z}$ definiert durch:

$$|\cdot| : \mathbb{Z} \to \mathbb{N}_0$$
$$\forall\, z \in \mathbb{Z}.\ |z| =_{df} \begin{cases} z & \text{falls } z \geq 0 \\ -z & \text{falls } z < 0 \end{cases}$$

und die Funktion f definiert durch:

$$f : (\mathbb{Z} \to \mathbb{Z}) \to (\mathbb{Z} \to \mathbb{Z})$$
$$\forall\, g \in \{\, h \,|\, h : \mathbb{Z} \to \mathbb{Z} \,\}\ \forall\, z \in \mathbb{Z}.\ (f(g))(z) =_{df} \sum_{i=0}^{|z|} g(i)$$

1. Implementieren Sie f als Haskell-Funktion höherer Ordnung `f` über dem Typsynonym `Z`, wobei das Resultat von `f` von funktionalem Wert sein muss:

   ```
   type Z = Integer

   f :: (Z -> Z) -> (Z -> Z)
   ```

2. Erklären Sie knapp, aber gut nachvollziehbar, wie ihre Implementierung für `f` vorgeht.

3. Geben Sie Implementierungen für `g1`, `g2`, `g3` an, so dass die geforderte Eigenschaft gilt:

   (a) $\forall\, z \in$ `Z. f g1` $z = \frac{z*(z+1)}{2}$
   (b) $\forall\, z \in$ `Z. f g2` $z = 0$
   (c) $\forall\, z \in$ `Z. f g3` $z = 1$

4. Sind die Implementierungen für `g1`, `g2`, `g3` eindeutig bestimmt? Begründen Sie Ihre Antwort.

5. Geben Sie die

   (a) curryfizierte
   (b) uncurryfizierte

   Lesart von `f` an.

Vergessen Sie nicht, die unterfertigte eideststattliche Erklärung mit abzugeben!