

Frage, Fragen und nochmals Fragen

Berthold Hoffmann

Universität Bremen and DFKI Bremen
`hof@informatik.uni-bremen.de`

In diesem Text stehen einige Fragen, die man sich zu den Folien der Veranstaltung *Funktionales Programmieren* (Praktische Informatik 3) im Winter 2008/2009 stellen könnte.

Alle TeilnehmerInnen der Veranstaltung sind eingeladen, die Fragen zu beantworten. Das sollte bei regelmäßigem Besuch der Veranstaltung nicht schwer fallen, ist aber völlig freiwillig.

Vielleicht stellen sich Ihnen auch noch weitere Fragen, die Sie dringlicher, interessanter und schwieriger finden? Dann können sie natürlich auch im Tutorium besprochen werden.

Solche Fragen könnten auch Gegenstand der *Fachgespräche* am Ende des Semester sein.

1 Fragen zur Vorlesung am 29. 10. 2008

1. Welche JAVA-Operationen sind nicht *referentiell transparent*? (Folie 10)
2. Welche Signaturen würden Sie den Funktionen `inc` und `addDouble` geben? (Folie 11)
3. Für welche Teilmenge von `Int` ist die Funktion `fac` definiert? (Folie 12)
4. Ändern Sie die Definition der Funktion `fac`, so dass sie total ist. (Folie 12)
5. Definieren Sie die Funktion `repeat` in JAVA.
Vergleichen Sie sie mit der Funktion auf Folie 14.
 - (a) Ist die JAVA-Funktion rekursiv?
 - (b) Ist die JAVA-Funktion total?
6. Welche Operationen von JAVA sind nicht strikt?
Geben Sie einen Ausdruck an, dessen Auswertung das belegt. (Folie 19 und 20)
7. Weshalb darf die Fallunterscheidung in keiner Programmiersprache strikt ausgewertet werden? (Folie 20)

2 Fragen zur Vorlesung am 5. 11. 2008

1. Definieren Sie die Funktion `max` mit einer bedingten Definition. (Folie 4 und 7)
2. Benennen Sie die Namen in einer von beiden Definitionen der Funktion `f` so um, dass keiner der Namen mehr einen anderen überlagert. (Folie 8)
3. Wenn man schreibt “`data Month = ... deriving (Eq, Ord)`”, sind die Werte des Datentyps `Days` total geordnet. Es gilt `Mon < Tue < Wed < Thu < Fri < Sat < Sun`.¹
Können Sie die Funktion `weekend` einfacher definieren, wenn Sie die Ordnungsrelation `(< :: Date -> Date -> Bool)` ausnutzen?
Ein einziger Vergleich müsste reichen! (Folien 11 und 12)
4. Welche Signatur hat der Wertkonstruktor `Date` auf der rechten Seite der Definition des gleichnamigen Datentyps auf Folie 17?
5. Welche Signatur haben die Wertkonstrukturen `ci` auf Folie 19?
6. Definieren Sie die Funktion `add` von Folie 21 mit einer Fallunterscheidung (`case`) bzw. einer bedingten Definition.
Welche Definition finden Sie am klarsten?
7. Definieren Sie die Multiplikation und die Fakultät auf `Nat`.
Ist die Fakultäts-Funktion partiell, wie die auf Folie 12 zur Vorlesung vom 29.10? (Siehe auch Frage ???.?)
8. Welche der drei Eigenschaften von algebraischen Datentypen auf Folie 19 garantiert dass

`Cons a : s ≠ Empty` (für alle `Char a` und `MyString s`)

9. Definieren Sie eine Funktion `first`, die das erste Zeichen eines `MyString` zurückgibt. (Folie 22)
10. Warum gilt für die “ungünstige” Definition von `MyString` unten auf Folie 22:

`Concat (Concat (Single 'a') (Single 'b')) (Single 'c')`
`≠ (Concat (Single 'a') (Concat (Single 'b') (Single 'c')))`

¹ Was es allgemein mit “`deriving`” auf sich hat, wird in einer der nächsten Vorlesungen erklärt.

3 Fragen zur Vorlesung am 12. 11. 2008

1. Die Funktionen `len`, `cat`, `rev` auf Folie 5 haben einige Eigenschaften (*properties*), die mit Gleichungen spezifiziert werden können, z. B.:

$$\begin{aligned} \text{len}(x) &= \text{len}(\text{rev } x) && \text{für alle } x \in \text{MyString} \\ \text{cat } x (\text{cat } y \ z) &= \text{cat } (\text{cat } x \ y) \ z && \text{für alle } x, y, z \in \text{MyString} \end{aligned}$$

Definieren Sie mindestens drei weitere Eigenschaften.

2. Mit welchen Typen kann der Typ `List a` von Folie 8 noch instanziiert werden? (Außer mit `Int` oder `Char`?) Nennen Sie mindestens drei weitere Typen.
Geben Sie jeweils einen typkorrekten Term für Ihre `List`-Instanzen an!
Geben Sie einen Term an, der den Typ `[a]` hat (wobei `a` eine Typvariable ist).
Geben Sie einen Term an, der den Typ `[[a]]` hat.²
3. Leiten Sie mit den Inferenzregeln von Folie 14 den allgemeinsten Typ für die Funktion `rev` auf Folie 5 oder 7 ab (Ignorieren Sie die dort stehenden Signaturen und nehmen Sie an, die Typumgebung *A* enthielte die folgenden Zuordnungen von Symbolen zu Typen: `Empty :: a`, `Cons :: a -> List a -> List a` und `cat :: List a -> List a -> List a`.)
4. Definieren Sie polymorphe Listen mit JAVA *Generics*. (Folie 15)
5. Definieren Sie den Zusammenhang zwischen den Funktionen `head` und `tail` mit einer Gleichung. (Folie 18)
6. Definieren Sie den Zusammenhang zwischen den Funktionen `init` und `last` mit einer Gleichung. (Folie 18)
7. Definieren Sie den Zusammenhang zwischen den Funktionen `take`, `drop` und `splitAt` mit einer Gleichung. (Folie 18)
8. Beschreiben Sie mit Hilfe der Funktionen `++` und `rev`, wie eine Zeichenkette `x` aufgebaut sein muss, damit `palindrom(x)` den Wert `True` liefert. (Folie 21)
9. Geben Sie ein Muster vom Typ `[(String, Bool), [(Int, String)]]` an. (Folie 22)

² Ihre Antworten auf die letzten drei Fragen können Sie gut überprüfen, wenn Sie im `ghci` mit `:type t` den Typ eines Terms *t* erfragen.

4 Fragen zur Vorlesung am 19. 11. 2008

1. Definieren Sie eine Funktion `iterate n f`, die die Funktion `f` `n` Mal anwendet (für $n \geq 0$). (Folie 3)
2. Definieren Sie mit `map` eine Funktion, die alle Zahlen einer Liste quadriert, bzw. deren Vorzeichen umkehrt, bzw. deren Kehrwert bildet. (Folie 6)
3. Definieren Sie mit `filter` eine Funktion, die aus einer Liste von Zahlen alle positiven Zahlen herausfiltert, bzw. aus einer Zeichenkette alle Layoutzeichen ' ', '\n' (Zeilenwechsel) und '\t' (Tabulator) herausfiltert.³ (Folie 7)
4. Definieren Sie `map f = foldr ...` und `filter p = foldr ...`. Definieren Sie die Funktions-Parameter von `foldr` entweder mit `where` oder direkt als Lambda-Abstraktion ("`x -> ...`"). (Folie 13)
5. Benutzen Sie `foldr`, um Funktionen zu definieren, die die Zahlen einer Liste miteinander multiplizieren, bzw. die Summe ihrer Kehrwerte bilden. Definieren Sie eine Funktion, die für eine Liste von Wahrheitswerten feststellt, ob alle bzw. ob mindestens einer `True` ist. (Folie 14)
6. Definieren Sie `rev`, indem Sie die Funktion `cons` lokal definieren oder gleich als Lambda-Abstraktion einsetzen. (Folie 18)
7. Definieren Sie die Funktion `zipWith`, die auf Folie 22 erwähnt ist.
8. Geben Sie den Funktionen `qsort` und `msort` einen weiteren Parameter haben, der die Vergleichsoperation angibt, nach der sortiert werden soll.
Was sind die Typen dieser Funktionen? (Im Zweifelsfall `ghci` fragen!) (Folie 7 und 28)

Nicht direkt auf diese Vorlesung bezieht sich diese Frage von Christian Maeder: Vergleichen Sie die Definitionen `type Point = (Double,Double)` und `data Point = Point Double Double`. Was sind die Vor- und Nachteile der jeweiligen Definition?

³ Tipp: die Funktion `elem x xs` von Folie 22 überprüft, ob ein Wert `x` in einer Liste `xs` von Werten enthalten ist.

5 Fragen zur Vorlesung am 26. 11. 2009

1. Definieren Sie Eigenschaften für die Funktion `kgv x y`, die das *kleinste gemeinsame Vielfache* von zwei ganzen Zahlen x und y berechnet. (Folie 7)
2. Definieren Sie ausführbare Eigenschaften für `kgv`. (Folie 8)
3. Entwickeln Sie aus den Eigenschaften von `kgv` eine Implementierung. (Folie 9)
4. Definieren Sie eine Funktion `width :: Tree a -> Int`, die die *Breite* eines Baums t liefert, also die Anzahl der in t enthaltenen Markierungen. Finden sie mindestens zwei Eigenschaften von `width` heraus. (Folie 35)
5. Definieren Sie eine Funktion `mapT :: (a -> b) -> Tree a -> Tree b`, die eine Funktion auf alle Markierungen eines Baums abbildet. Geben Sie mindestens zwei Eigenschaften von `mapT` an. (Folie 35)
6. Definieren Sie die Funktion `width` von Frage ?? mit `foldT` (oder `preorder`). (Folie 35 und 36)
7. Definieren Sie eine Umkehrfunktion `balTree :: [a] -> Tree a` zu `inorder`, so dass der Baum möglichst balanciert ist.
(*Tipp*: die Funktion `splitAt :: Int -> [a] -> ([a], [a])` teilt eine Liste $[x_1, \dots, x_n]$ an der Stelle $1 \leq i \leq n$ in das Listen-Paar $([x_1, \dots, x_i], [x_{i+1}, \dots, x_n])$.)
Welche Eigenschaft gilt für die Höhen des linken und rechten Unterbaums eines Knotens? Finden Sie Argumente für diese Behauptung! (Folie 36)

6 Fragen zur Vorlesung am 3. 12. 2008

1. In der Vorlesung vom 26. 11. wurde auf Folien 32-36 der Typ `Tree a` mit einigen Operationen definiert.
Was müssen Sie tun, um `Tree a` als abstrakten Datentyp zu definieren?
Reichen die auf den Folien definierten Operationen aus, um mit dem abstrakten Datentyp zu arbeiten?
2. Den `CodeTree` aus Aufgabe 5 auf dem 2. Übungsblatt könnte man auch gut als abstrakten Datentypen definieren.
Wie sähe die Modulschnittstelle aus?
Reichen die in der Aufgabe geforderten Operationen, um mit dem Datentyp zu arbeiten?
3. Auch den Datentyp `Array a` aus Aufgabe 7 auf dem 2. Übungsblatt könnte man gut als abstrakten Datentypen definieren.
Wie sähe die Modulschnittstelle aus?
Reichen die in der Aufgabe geforderten Operationen, um mit dem Datentyp zu arbeiten?
4. Erweitern Sie die abstrakte Syntax von Ausdrücken um eine Alternative, mit der bedingte Ausdrücke der Form

$$\textit{CondExpr} ::= \textbf{if } Expr \textbf{ then } Expr \textbf{ else } Expr$$

repräsentieren kann.

Erweitern Sie den Parser auf Folie 25 auf bedingte Ausdrücke.

7 Fragen zur Vorlesung am 10. 12. 2008

In der Vorlesung ging es um die Spezifikation von abstrakten Datentypen, die aus Signaturen und Axiomen besteht.

Betrachten Sie die abstrakten Datentypen (ADTen), die Sie in den Fragen zur letzten Vorlesung konstruieren sollten:

- Die Bäume **Tree** **a**
- Die Code-Bäume **Codetree** **a**
- Die Felder **Array** **a**

Tun Sie für diese abstrakten Datentypen das Folgende:

1. Stellen Sie die Signaturen der ADTen zusammen.
2. Finden Sie Eigenschaften der Operationen.
3. Definieren Sie diese Eigenschaften als testbare Eigenschaften (wie auf den Folien 20–21).
4. Beweisen Sie die Eigenschaften.
5. Definieren Sie *properties* für **quickCheck** (wie auf den Folien 22–23).
6. Testen Sie die Eigenschaften. mit **quickCheck**.

8 Fragen zur Vorlesung am 17. 12. 2008

1. Beweisen Sie die folgenden Eigenschaften von Folie 12 durch strukturelle Induktion über Listen:

```
length (filter p xs) <= length xs
length (xs ++ ys) = length xs ++ length ys
map f (xs ++ ys) = map f xs ++ map f ys
length (concat xs) = sum (map length xs)
```

2. Definieren Sie eine Funktion `mirror :: Tree a -> Tree a`, die die Zweige aller Knoten rekursiv spiegelt. Zeigen Sie folgende Eigenschaften:

```
mirror (mirror t) = t
preorder t = postorder (mirror t)
inorder t = inorder (mirror t)
inorder (mirror t) = rev (inorder t)
```

9 Fragen zur Vorlesung am 07. 01. 2009

1. Schreiben Sie die Aktionen `echo1` und `ohce` in **do**-Notation. (Folien 9 und 10)
2. Schreiben Sie die Aktion `wc` *ohne* **do**-Notation.
3. Schreiben Sie mit Hilfe der Funktion `palindrom` auf Folie 20 der Vorlesung vom 12. 11. 2008 ein interaktives Programm, dass so lange Palindrome überprüft, bis ein einbuchstabiges Wort angegeben wird.
4. Schreiben Sie mit Hilfe der Funktion `makeIndex` auf Folien 22ff der Vorlesung vom 26. 11. 2008 ein interaktives Programm

`index` $\langle \textit{Dateiname} \rangle$ [$\langle \textit{Indexname} \rangle$]

das den Index einer Datei erzeugt. Der Index soll entweder ausgegeben oder in die Datei $\langle \textit{Indexname} \rangle$ geschrieben werden.

5. Erweitern sie `index` zu einem interaktiven Programm, das wiederholt Dateinamen und Indexnamen abfragt.