Funktionale Programmierung, SS 2018, Blatt 2

Fabian Kunze, Steven Schäfer, Kathrin Stark,
Prof. Dr. Gert Smolka
https://courses.ps.uni-saarland.de/fp_ss18/

Typklassen und Überladen von Operatoren

Aufgabe 2.1 (Typklassen) Lesen Sie Kapitel 5 und 8 in *A Gentle Introduction to Haskell* (www.haskell.org/tutorial/) von Hudak, Peterson und Fasel. Experimentieren Sie mit einigen Beispielen aus dem Text.

Aufgabe 2.2 (Komplexe Zahlen) Sei der Typ *data Comp a = Comp a a* gegeben, den wir als Datentyp von komplexen Zahlen auffassen wollen. Genauer soll *Comp Float* die komplexen Zahlen $\mathbb{C}$ repräsentieren, *Comp Int* die komplexen ganzen Zahlen $\mathbb{Z}[i]$ usw.

a) Geben Sie eine Instanzdeklaration *instance Eq a ⇒ Eq (Comp a) where ...*, die Gleichheit auf den Elementen von *Comp a* definiert.

b) Geben Sie eine Instanzdeklaration *instance Show a ⇒ Show (Comp a) where ...*, so dass *show (Comp 1 1)* den String `1+1i` liefert.

c) Geben Sie eine Instanzdeklaration *instance Num a ⇒ Num (Comp a) where ...*, die die arithmetischen Operationen auf *Comp a* definiert. Beschränken Sie sich dabei auf die Operationen

$$\begin{aligned}(+), (-), (*) &:: \mathit{Comp}\ a \to \mathit{Comp}\ a \to \mathit{Comp}\ a\\ \mathit{negate} &:: \mathit{Comp}\ a \to \mathit{Comp}\ a\\ \mathit{fromInteger} &:: \mathit{Integer} \to \mathit{Comp}\ a\end{aligned}$$

Beispielsweise soll $i * i == -1$ für $i = \mathit{Comp}\ 0\ 1$ den Wert *True* liefern.

Aufgabe 2.3 (Eq) Betrachten Sie den durch *data Set a = Set [a]* definierten Typ zur Repräsentation von Mengen über *a* durch Listen.

a) Geben Sie eine Instanzdeklaration *instance Eq a ⇒ Eq (Set a) where ...*, so dass $\mathit{Set}[3,4] == \mathit{Set}[4,3]$ den Wert *True* liefert. (Beachten Sie, dass $[3,4] == [4,3]$ den Wert *False* liefert.)

b) Betrachten Sie den Ausdruck $\mathit{Set}[\mathit{Set}[3,2]] == \mathit{Set}[\mathit{Set}[2,3], \mathit{Set}[2,3,3]]$. Geben Sie die Instanzen der Typklasse *Eq* an, die inferiert werden müssen um diesen Ausdruck zu typisieren.

Aufgabe 2.4 (Bäume) Betrachten Sie den Typ *data Tree a = Node a [Tree a]* von endlich verzweigenden Bäumen. Geben Sie eine Instanzdeklaration für die Typklasse *Eq*, so dass (==) Bäume auf strukturelle Gleichheit testet.

Aufgabe 2.5 (Konstruktorklassen) Wie in Kapitel 5 beschrieben, sind die Instanzen der Typklasse *Functor* Typkonstruktoren, die eine "map" Funktion besitzen. (Insbesondere ist *List* mit *fmap* = *map* eine Instanz von *Functor*.)

a) Geben Sie eine Instanzdeklaration für den Typkonstruktor *Set* aus Aufgabe 2.3 an, so dass beispielsweise $fmap\ (+1)\ (Set[1,2,3]) == Set[2,3,4]$ gilt.

b) Geben Sie eine Instanzdeklaration für den Typkonstruktor *Tree* aus Aufgabe 2.4 an, indem Sie eine geeignete Funktion *fmap* angeben.

c) Analog zu *Functor* lässt sich die Typklasse *BiFunctor* für Typkonstruktoren f mit *zwei* Typargumenten definieren:

$$\begin{aligned}&class\ BiFunctor\ f\ where\\ &\quad bifmap\ ::\ (a \to a') \to (b \to b') \to f\ a\ b \to f\ a'\ b'\end{aligned}$$

Geben Sie Instanzdeklarationen für Produkttypen (a, b) und Summentypen $Either\ a\ b$ an.[1]

[1] *Either* ist in der Prelude durch die Deklaration $data\ Either\ a\ b = Left\ a \mid Right\ b$ gegeben.