**Institut für Informatik** **Ludwig-Maximilians-Universität München**
Prof. Dr. François Bry

**Programmierung und Modellierung**, SoSe 16
**Übungsblatt 10**

*Abgabe: bis Mo 20.06.2016 10:00 Uhr*

Besprechung: am ab Di 21.06.2016

## Aufgabe 10-1 Monoide

Monoide sind eine Typklasse in Haskell. In der Dokumentation werden Monoide als "types with an associative binary operation that has an identity" beschrieben. Das heißt, dass man für einen Monoiden ein neutrales Element und eine irgendwie-geartete assoziative Operation für zwei Argumente diesen Types implementieren muss. So ist die Operationen `+` über Integer ein Monoide, die Operation `/` jedoch nicht.

Implementieren Sie die folgenden Beschreibungen von Datentypen als Monoid. Benutzen Sie dazu die von Haskell vordefinierte Typklasse `Data.Monoid`.[1]

a) Ein Student, welcher der Vorlesung nicht allzu aufmerksam gefolgt hat, hat eine zündende Idee: Ein Datentyp, mit dem man beliebige Zeichenkette darstellen und auch noch miteinander verknüpfen kann! Einen passenden Namen hat er sich auch schon dafür ausgedacht: `CharacterChain`.

   Helfen Sie dem Studenten, den Datentyp `CharacterChain` als Monoid zu implementieren.

b) Implementieren Sie einen Datentyp `ComplexNumber`, der eine komplexe Zahl darstellt. Implementieren Sie dann die Addition zweier komplexer Zahlen als Monoid. Implementieren Sie zusätzlich für `ComplexNumber` die Typklasse `Show` welche die komplexe Zahl in der Form $a + bi$ (wobei $a$ der Realteil und $b$ der Imaginärteil der Zahl ist) aus.

c) Im RGB-Farbraum wird eine Farbe als Tripel `(Red, Green, Blue)` von drei ganzen Zahlen im Bereich von 0 bis 255 dargestellt. Jede Komponente des Tripels repräsentiert dabei den Anteil der jeweiligen Farbe an der Gesamtfarbe.

   Für das Mischen von Farben gibt es verschiedene Modelle. In dieser Aufgabe werden Farben additiv gemischt, d.h. die entsprechenden Farbwerte werden einfach addiert. Zu beachten gibt es hier nur, dass der Wert einer Farbe nicht über 255 steigen darf. Implementieren Sie einen Datentyp für RGB-Werte und additive Farbmischung als Monoid.

   Handelt es sich bei der subtraktiven Farbmischung (wie additive Farbmischung, nur mit Substraktion anstatt Addition) auch um einen Monoiden? Wenn ja, dann implementieren Sie diese Art der Farbmischung für Ihren Datentyp. Wenn nicht, begründen Sie *kurz*, wieso es sich hierbei um keinen Monoiden handelt.

## Aufgabe 10-2 Funktoren und Applikative

a) Die Typklasse `Functor` verallgemeinert die Listen mit `map`. Die Funktion `fmap` eines Functors überträgt eine normale „Funktion“ auf Functor-Werte. Nutzen Sie dies aus, um einen eigenen Datentyp `List` zu implementieren, der keine oder beliebig viele Elemente enthalten kann. Achten Sie darauf, dass der neue Listen Datentyp alle von der Typklasse `Functor` geforderten Gesetze umsetzt.

[1] https://hackage.haskell.org/package/base-4.9.0.0/docs/Data-Monoid.html

b) Schreiben Sie eine eigene String Repräsentation des neuen Typen `List` mit Hilfe der Typklasse `Show` und setzten Sie auch die Gleichheit mittels der Typklasse `Eq` um.

c) Realisieren Sie eine Funktion `scale :: (Functor f, Num b) => b -> f b -> f b`, die alle Elemente einer Liste vom Typ `List` mit einem Faktor vom Typ `Num` multipliziert.

d) Die Funktion `<*>` eines Applicativen Functors ermöglicht die Anwendung einer Functor-Funktion innerhalb des Funktors. Definieren Sie ein *Applicative* für den Datentyp `List` und definieren Sie die Funktion `lzipWith` für den Datentyp `List` analog zur Haskell Funktion `zipWith`. Denken Sie daran, das Modul `Control.Applicative` zu importieren.

## Aufgabe 10-3 Funktoren und Applikative

a) Dreidimensionale Punkte werden häufig als Tripel (Vektoren) dargestellt. Implementieren Sie einen Datentyp `Triple` mit einer eigenen String Repräsentation durch die Typklasse `Show`.

b) Um auf die einzelnen Komponenten des Tripels zugreifen zu können, werden, analog zu den Standard Haskell Tupel Funktionen `fst` und `snd`, Äquivalente für den neuen Datentyp `Triple` benötigt; implementieren Sie diese.

c) Konvertierungsmethoden aus oder auf bestehende Datentypen sind sehr praktisch, um einen neuen Datentyp verwenden zu können. Implementieren Sie daher eine Funktion, die einen Tripel aus einer Liste und eine Funktion, die aus einer Liste ein Tripel erzeugt.

d) Als nächstes sollen Sie das Kreuzprodukt zweier Tripel implementieren. Im dreidimensionalen kartesischen Koordinatensystem lässt sich das Kreuzprodukt wie folgt berechnen:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \times \begin{pmatrix} d \\ e \\ f \end{pmatrix} = \begin{pmatrix} bf - ce \\ cd - af \\ ae - bd \end{pmatrix}$$

e) Implementieren Sie auch noch eine Funktion, um ein Tripel mit einem Skalar zu multiplizieren können. Definieren Sie hierfür den Funktor `Triple` (die Funktion `fmap` wird für die Implementierung nützlich sein). Die Skalarmultiplikation ist wie folgt definiert:

$$s \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} sa \\ sb \\ sd \end{pmatrix}$$

f) Nützlich ist auch noch das Skalarprodukt und die Vektor-Addition/-Subtraktion. Um diese zu realisieren, definieren Sie ein Applicative `Triple`. Implementieren Sie dann die Funktionen nach den folgenden Definitionen:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} * \begin{pmatrix} d \\ e \\ f \end{pmatrix} = ad + be + cf \qquad \begin{pmatrix} a \\ b \\ c \end{pmatrix} \pm \begin{pmatrix} d \\ e \\ f \end{pmatrix} = \begin{pmatrix} a \pm d \\ b \pm e \\ c \pm f \end{pmatrix}$$

g) Zum Schluss muss noch die Länge eines Tripels berechnet werden. Implementieren Sie diese wie folgt:

$$\left| \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right| = \sqrt{a^2 + b^2 + c^2}$$