Prof. Dr. Christoph Bockisch
Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Übungen zur Vorlesung
Deklarative Programmierung: Sommersemester 2018

Nr. 6, Abgabe bis 11.6.2018 11:59 Uhr

Aufgabe 6.1: Funktionen als Parameter — 4 Punkte

Sie haben in der Vorlesung gelernt, wie man Funktionen als Parameter übergeben kann. Nutzen Sie dies nun, um folgende Funktionen zu implementieren:

a) `; (Number -> Number) Number -> Number`
`(sumTo f n)`, die die Summe $\sum_{k=1}^{n} f(k)$ berechnet.

b) `; (Number -> Number) Number Number Number -> Number`
`(integral f a b n)`, die das Integral $\int_a^b f(x)dx$ durch die Summe $\sum_{k=0}^{n-1} f(a + k * dx) * dx$ mit $dx = \frac{b-a}{n}$ annähert.

Hinweis: Lagern Sie evtl. benötigte Teilfunktionen in lokale Definitionen aus.

Aufgabe 6.2: Black Mesa — 4 Punkte

Erstellen Sie Funktionen, die

a) `; [X,Y,Z] (X -> Y) (Y -> Z) -> (X -> Z)`
`(combine f g)`, die die beiden unären Funktionen verkettet und eine neue Funktion h liefert, sodass gilt: $h(x) = g(f(x))$

b) `; [X,Y,Z] (X Y -> Z) X -> (Y -> Z)`
`(partial f x)`, die den ersten Parameter der binären Funktion f mit dem Wert x fixiert und eine neue unäre Funktion zurückliefert.

c) `; [X] Number (X -> X) -> (X -> X)`
`(repeated n f)`, die eine Funktion g zurückliefert, die f insgesamt n-mal auf ihr Argument x anwendet. D.h.: $g(x) = f^n(x) = f(f(f(...f(x)...)))$

Hinweis: Lösen Sie die Aufgaben unter Verwendung von Lambda-Ausdrücken. Die Benutzung lokaler Definitionen ist nicht zulässig.

Aufgabe 6.3: Raise! 4 Punkte

Implementieren Sie folgende Listenfunktionen mithilfe von `foldr` und Lambdas:

a) `; [X,Y] (X -> Y) (list-of X) -> (list-of Y)`
`(my-map f l)`, die jedes Element einer Liste mithilfe von f abbildet und eine Liste der abgebildeten Werte erzeugt.

b) `; [X] (X -> Boolean) (list-of X) -> (list-of X)`
`(my-filter p l)`, die alle Elemente aus der Liste l ausgibt, die die Bedingung p erfüllen.

c) `; [X] (X -> Boolean) (list-of X) -> Boolean`
`(forall p l)`, die überprüft, ob alle Elemente der Liste die gegebene Eigenschaft erfüllen.

d) `; [X] (X -> Boolean) (list-of X) -> Boolean`
`(exists p l)`, die überprüft, ob mindestens ein Element der Liste die gegebene Eigenschaft erfüllt.

Hinweis: Sie dürfen Ihre Funktionen in den folgenden Teilaufgaben verwenden.

Aufgabe 6.4: What's your favourite idea? Mine is to be creative! 4 Punkte

In dieser Aufgabe sollen Sie kreativ werden und eine möglichst „schöne" Animation erstellen. Im ILIAS finden Sie dazu ein Programmskelett (`animate.rkt`). Machen Sie sich zunächst mit dem Programmcode vertraut und versuchen Sie das Beispiel nachzuvollziehen.

Die Animation soll aus verschiedenen Animationen einzelner Shapes zusammengesetzt werden. Jede dieser Einzelanimationen besteht aus drei Transformationen (Position, Rotation und Farbe), einem Start- und Endzeitpunkt sowie einem Shape. Shapes sind Funktionen, die ein Bild nur durch Übergabe der Farbe erzeugen. Die Transformationen haben alle Start- und Endzustände sowie eine sog. easing-Funktion, mit der ihr Verlauf beeinflusst werden kann (Beispiele finden Sie hier: `http://easings.net/de`).

Die einzigen Bedingungen an die Animation sind, dass Sie wenigestens 2 komplexe (zusammengesetzte) Shapes, sowie vier unterschiedliche easing-Funktionen verwenden sollen. Viel Spaß!

Aufgabe 6.5: I like to move it, move it!

$\lim_{n\to\infty}(\sqrt[n]{5}-1)_{n\in\mathbb{N}}$ Punkte