Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. François Bry, Thomas Prokosch
Kilian Rückschloß

Programmierung und Modellierung, SoSe 21
Übungsblatt 5

Abgabe: bis Mo 17.05.2021 08:00 Uhr

Aufgabe 5-1 applikative und normale Auswertungsreihenfolge

```
quadrat = \x -> x*x
summe_quadrate = \x y -> quadrat x + quadrat y
null x = 0
f n = if null (quadrat n) /= n then summe_quadrate (n-2) (n-1) else n
```

Geben Sie in der Datei `5-1.txt` für den Ausdruck `f 3` die einzelnen Schritte bei der Auswertung mit dem Substitutionsmodell an **mit applikativer Auswertungsreihenfolge (inside-out)**.

Welchen Vorteil hätte hier die normale Auswertungsreihenfolge gebracht?

Aufgabe 5-2 Normale, applikative und verzögerte Auswertung

Gegeben seien die folgenden Definitionen sowie die vordefinierten Ausdrücke `tail` und `head`.

```
f = head . tail
g = (\x -> 0)
h  = (\x -> x * x * x)
```

- `tail` bestimmt den Rest einer Liste, d.h. die Liste aller Elemente außer dem ersten Element. Der Aufruf `tail [1, 2, 3]` gibt so beispielsweise die Liste `[2, 3]` zurück.
- `head` bestimmt das erste Element einer Liste. Der Aufruf `head [1, 2, 3]` gibt so beispielsweise das Element `1` zurück.

a) Geben Sie die Auswertung von `g (h 0)` in applikativer Reihenfolge an.

b) Geben Sie die Auswertung von `g (h 0)` in normaler Reihenfolge an.

c) Geben Sie die verzögerte Auswertung von `h (h 1)` an.

d) Geben Sie die verzögerte Auswertung von `f [1,2,3]` an.

Aufgabe 5-3 Typen einfacher Ausdrücke

Mit Ihrem Wissen aus der Vorlesung sollten Sie nun die Typen der Ausdrücke in den Aufgaben 1-2 und 2-1 besser nachvollziehen können.

Mindestens genauso nützlich es aber auch zu erkennen, ob ein Ausdruck überhaupt fehlerfrei kompiliert bzw. schnell einen Fehler zu finden (wie Sie auf den vergangenen Übungsblättern sicher schon gemerkt haben).

Geben Sie den Wert und Typ der folgenden Haskell-Ausdrücke an bzw. bestimmen Sie, welche fehlerhaft sind, d.h. **keinen Wert bzw. Typ** haben und liefern Sie in diesen Fällen eine Begründung:

Anmerkung: Bei dieser Aufgabe nehmen wir an, dass Polymorphismus gestattet ist (i.e. Standardverhalten der `ghci`).

a) `let a = [1, 2, 3]`

b) `let b = "ab"++ ['c', 'd']`

c) `let c = [1, 2.0, 3]`

d) `let d = (False && not True) || (not False && True)`

e) `let e = [("a", 1), ("b", 2)]`

f) `let f = [1, 2, [3, 4, [5]]]`

g) `let g = [[[1], [2]], [], [[3], [4]]]`

Aufgabe 5-4 Typen von Funktionen

Bestimmen Sie den Typ der folgenden Funktionen.

a)
```
f x = if f x == f x then f x else f x > f x
```

b)
```
f x = if f(x) == f(x) then f(x) else (f (x + 1)) > (f (x + 2))
```

c)
```
f x y z = if x > y then z else z + 1
```

d)
```
f 0 = 1
f n = n * f (n - 1)
```

e)
```
f x = [y | y <- x, y `mod` 2 == 0]
```

f)
```
(\x -> x == '3')
```