

Praktische Informatik 3: Funktionale Programmierung  
Vorlesung 14 vom 15.02.21: Rückblick & Ausblick

Christoph Lüth



Wintersemester 2020/21



## Fahrplan

- ▶ Teil I: Funktionale Programmierung im Kleinen
- ▶ Teil II: Funktionale Programmierung im Großen
- ▶ **Teil III: Funktionale Programmierung im richtigen Leben**
  - ▶ Aktionen und Zustände
  - ▶ Monaden als Berechnungsmuster
  - ▶ Funktionale Webanwendungen
  - ▶ Scala — Eine praktische Einführung
  - ▶ **Rückblick & Ausblick**



## I. Prüfungen



## Inhalt

- ▶ Warum **Fachgespräche**?
- ▶ Was bedeutet das für die **Scheinbedingungen**?
- ▶ **Ablauf** der Fachgespräche
- ▶ **Termin** für die Fachgespräche
- ▶ **Beispiele**



## Warum Fachgespräche?

- ▶ Fachgespräche dienen dazu, „Individualität der Leistung“ sicherzustellen.
  - ▶ Besonders in Corona-Zeiten.
- ▶ Format: Eine Gruppe (3 Studierende), 15 Minuten
  - ▶ Individuelle Prüfungen (30 Minuten) zeitlich nicht darstellbar (127 (Pabo) – 190 (stud.ip))
  - ▶ Open-Book Klausur o.ä. keine Alternative, da keine wesentliche Verbesserung zu Übungsbetrieb
  - ▶ Parallelisierung erlaubt höheren Durchsatz und längere Antwortzeit
- ▶ Gruppe muss keine Übungsgruppe sein



## Scheinkriterien

- ▶ Fachgespräch am Ende (Individualität der Leistung)
- ▶ Mind. 50% in den Einzelübungsblättern und mind. 50% in allen Übungsblättern
- ▶ **Vornote** aus den Übungsblättern
- ▶ **Notenspiegel** (in Prozent aller Punkte):

Pkt.-%	Note	Pkt.-%	Note	Pkt.-%	Note	Pkt.-%	Note
≥ 95	1.0	89.5-85	1.7	74.5-70	2.7	59.5-55	3.7
94.5-90	1.3	84.5-80	2.0	69.5-65	3.0	54.5-50	4.0
		79.5-75	2.3	64.5-60	3.3	49.5-0	<b>n/b</b>

- ▶ Fachgespräch **bestätigt** Vornote (Abänderung bis max. 1 Notenstufe)



## Ablauf des Fachgesprächs

- ▶ Fachgespräche finden über **Zoom** statt
- ▶ Fachgespräche bestehen aus der schriftlichen Bearbeitung einer kurzen **Programmieraufgabe** sowie kurzen **Verständnisfragen** dazu
- ▶ Aufgaben werden online auf HedgeDoc bearbeitet.
  - ▶ Kein Syntaxcheck, kein Compiler
- ▶ Beispielfragen auf der Webseite



## Ablauf und Ziel des Fachgesprächs

- ▶ Kamera ist **Pflicht** (**eingeschaltet** lassen)
- ▶ **Ruhiger** Ort ohne Hintergrundgeräusche
- ▶ Niemand **sonst** im Raum
- ▶ Bearbeitung **live** im HedgeDoc, nicht aus Editor kopieren
- ▶ Der Weg ist das Ziel...
- ▶ **Nicht:** Gelöste Aufgabe.
- ▶ **Sondern:** wie löst ihr die Aufgabe?



## Termine

- ▶ Fachgespräche finden an **drei** Tagen statt.
- ▶ Mögliche **Termine**: 11/12.03, 18/19.03., 25/26.03.
- ▶ Dazu **Umfrage**.
- ▶ Anmeldung wird **nach der Vorlesung** freigeschaltet.

## Beispielfrage (leicht)

Definieren Sie eine Funktion `format`, die eine Zahl in einer Zeichenkette gegebener Länge rechtsbündig ausgibt.

Bsp. `format 4 35`  $\rightsquigarrow$  `" 35"`

Lösung:

```
format :: Int -> Int -> String
```

```
format n x =  
  replicate ' ' (n - length s) ++ s  
  where s = show x
```

Fragen:

- ▶ Was ist daran falsch?
- ▶ Funktioniert das auch für negative Zahlen?
- ▶ Wie kann ich Länge auf `n` begrenzen?

## Beispielfrage (mittel)

Definieren Sie eine Funktion `mean`, die den arithmetischen Durchschnitt einer Liste von ganzen Zahlen berechnet.

Bsp. `mean [2,1,5,4,3]`  $\rightsquigarrow$  `3.0`

Lösung:

```
mean :: [Int] -> Double
```

```
mean xs = sum xs / length xs
```

Fragen:

- ▶ Was ist daran falsch?
- ▶ Wie konvertiert man `Int` nach `Double`?  

```
mean xs = fromIntegral (sum xs) / ...
```
- ▶ Schwer: wie vereinfacht man `fromIntegral (length xs)`

## Beispielfrage (schwer)

Zwei `Int`-Listen sollen **ähnlich** heißen, wenn Sie die gleichen Zahlen unabhängig von ihrer Reihenfolge und Häufigkeit enthalten. Schreiben Sie eine Testfunktion `sim` dafür.

Bsp. `sim [3,2,2,1,3] [1,2,3]`  $\rightsquigarrow$  `True`

```
sim :: [Int] -> [Int] -> Bool
```

```
sim xs ys = all (\x -> elem x ys) xs
```

Fragen:

- ▶ Was fehlt da?  

```
sim xs ys =  
  .. && all (\y -> elem y xs) ys
```
- ▶ Kann man die Signatur verallgemeinern?  

```
sim :: Eq a => [a] -> [a] -> Bool
```

## Mündliche Prüfung

- ▶ **Dauer**: in der Regel 20-30 Minuten
- ▶ **Einzelprüfung**, ggf. mit Beisitzer
- ▶ **Inhalt**: Programmieren mit Haskell und Vorlesungsstoff
- ▶ **Ablauf**: "Fachgespräch plus"
  - ▶ Einstieg mit leichter Programmieraufgabe wie im Fachgespräch
  - ▶ Daran anschließend Fragen über den Stoff
- ▶ Liste von **Verständnisfragen** auf der Webseite.

## II. Rückblick und Ausblick

## Warum funktionale Programmierung lernen?

- ▶ Funktionale Programmierung macht aus Programmierern Informatiker
- ▶ Blick über den Tellerrand — was kommt in 10 Jahren?
- ▶ **Herausforderungen** der Zukunft
- ▶ Enthält die **wesentlichen** Elemente moderner Programmierung

## Zusammenfassung Haskell

**Stärken:**

- ▶ Abstraktion durch
  - ▶ Polymorphie und Typsystem
  - ▶ algebraische Datentypen
  - ▶ Funktionen höherer Ordnung
- ▶ Flexible Syntax
- ▶ Haskell als **Meta-Sprache**
- ▶ Ausgereifter Compiler
- ▶ Viele Büchereien

**Schwächen:**

- ▶ Komplexität
- ▶ Büchereien
  - ▶ Nicht immer gut gepflegt
- ▶ Viel im Fluß
  - ▶ Kein stabiler und brauchbarer Standard
- ▶ Divergierende Ziele:
  - ▶ Forschungsplattform **und** nutzbares Werkzeug

## Andere Funktionale Sprachen

- ▶ **Standard ML (SML):**
  - ▶ Streng typisiert, strikte Auswertung
  - ▶ Standardisiert, formal definierte Semantik
  - ▶ Drei aktiv unterstützte Compiler
  - ▶ Verwendet in Theorembeweisern (Isabelle, HOL)
  - ▶ <http://www.standardml.org/>
- ▶ **CamL, O'CamL:**
  - ▶ Streng typisiert, strikte Auswertung
  - ▶ Hocheffizienter Compiler, byte code & nativ
  - ▶ Nur ein Compiler (O'CamL)
  - ▶ <http://caml.inria.fr/>



## Andere Funktionale Sprachen

- ▶ **LISP und Scheme**
  - ▶ Ungetypt/schwach getypt
  - ▶ Seiteneffekte
  - ▶ Viele effiziente Compiler, aber viele Dialekte
  - ▶ Auch industriell verwendet
- ▶ **Hybridsprachen:**
  - ▶ Scala (Functional-OO, JVM)
  - ▶ F# (Functional-OO, .Net)
  - ▶ Clojure (Lisp, JVM)



## Was spricht gegen funktionale Programmierung?

- ▶ **Mangelnde Unterstützung:**
  - ▶ Libraries, Dokumentation, Entwicklungsumgebungen
  - ▶ Wird besser (Scala)...
- ▶ **Programmierung** nur kleiner Teil der SW-Entwicklung
- ▶ **Nicht verbreitet** — funktionale Programmierer zu teuer
- ▶ **Konservatives Management**
  - ▶ "Nobody ever got fired for buying IBMSAP"



## Haskell in der Industrie

- ▶ Simon Marlow bei Facebook
- ▶ Simon Peyton-Jones bei Microsoft.
- ▶ secuCloud in Hamburg (<https://www.secuccloud.com/>)
- ▶ Bluespec: Schaltkreisentwicklung, DSL auf Haskell-Basis
- ▶ Galois, Inc: Cryptography (Cryptol DSL)
- ▶ Finanzindustrie: Barclays Capital, Credit Suisse, Deutsche Bank
- ▶ Siehe auch: Haskell in Industry ([https://wiki.haskell.org/Haskell\\_in\\_industry](https://wiki.haskell.org/Haskell_in_industry))
- ▶ Andere Sprachen: Scala, Erlang, FL, ...



## Perspektiven funktionaler Programmierung

- ▶ **Forschung:**
  - ▶ Ausdrucksstärkere Typsysteme
  - ▶ für effiziente Implementierungen
  - ▶ und eingebaute Korrektheit (Typ als Spezifikation)
  - ▶ Parallelität?
- ▶ **Anwendungen:**
  - ▶ Eingebettete domänenspezifische Sprachen
  - ▶ Zustandsfreie Berechnungen (MapReduce, Hadoop, Spark)
  - ▶ **Big Data** and **Cloud Computing**



## If you liked this course, you might also like ...

- ▶ Die Veranstaltung **Reaktive Programmierung** (findet irregulär stattt)
  - ▶ Scala, nebenläufige Programmierung, fortgeschrittene Techniken der funktionalen Programmierung
- ▶ Wir suchen **studentische Hilfskräfte** am DFKI, FB CPS
  - ▶ Scala als Entwicklungssprache
- ▶ Wir suchen **Tutoren für PI3**
  - ▶ Im WS 2021/22 — **meldet Euch** bei Thomas Barkowsky (oder bei mir)!



Tschüß!

