

Praktische Informatik 3
Einführung in die Funktionale Programmierung
Vorlesung vom 04.02.09:
Schlußbemerkungen

Christoph Lüth

WS 08/09



Fahrplan

- Teil I: Grundlagen
- Teil II: Abstraktion
- Teil III: Beispiele, Anwendungen, Ausblicke
 - Datenmodellierung mit XML
 - Effizienzerwägungen
 - Grafik
 - Schluss

Organisatorisches

- Ausgefüllten **Scheinvordruck** zum Fachgespräch **mitbringen**
- Nur wer ausgefüllten Scheinvordruck abgibt, erhält auch einen.
- Evaluationsbogen ausfüllen

Inhalt

- Wiederholung

- Rückblick, Ausblick

Vorlesung vom 29.10.08: Grundbegriffe

- Was sind die Bestandteile einer Funktionsdefinition?

Vorlesung vom 29.10.08: Grundbegriffe

- Was sind die Bestandteile einer Funktionsdefinition?
- Welche Auswertungsstrategien gibt es, welche benutzt Haskell?

Vorlesung vom 29.10.08: Grundbegriffe

- Was sind die Bestandteile einer **Funktionsdefinition**?
- Welche **Auswertungsstrategien** gibt es, welche benutzt Haskell?
- Was bedeutet **Striktheit**?

Vorlesung vom 05.11.08: Funktionen und Datentypen

- Wie werden **syntaktisch** Funktionen definiert?

Vorlesung vom 05.11.08: Funktionen und Datentypen

- Wie werden **syntaktisch** Funktionen definiert?
- Was ist die **Abseitsregel**?

Vorlesung vom 05.11.08: Funktionen und Datentypen

- Wie werden **syntaktisch** Funktionen definiert?
- Was ist die **Abseitsregel**?
- Welches sind die **wesentlichen** Eigenschaften der **Konstruktoren** eines algebraischen Datentyps?

Vorlesung vom 05.11.08: Funktionen und Datentypen

- Wie werden **syntaktisch** Funktionen definiert?
- Was ist die **Abseitsregel**?
- Welches sind die **wesentlichen** Eigenschaften der **Konstruktoren** eines algebraischen Datentyps?
- Welche vordefinierten **Basisdatentypen** gibt es in Haskell?

Vorlesung vom 12.11.08: Typvariablen und Polymorphie

- Was ist **Polymorphie** in funktionalen Sprachen?

Vorlesung vom 12.11.08: Typvariablen und Polymorphie

- Was ist **Polymorphie** in funktionalen Sprachen?
- Wo ist der Unterschied zu Java?

Vorlesung vom 12.11.08: Typvariablen und Polymorphie

- Was ist **Polymorphie** in funktionalen Sprachen?
- Wo ist der Unterschied zu Java?
- Woran kann Typableitung **scheitern**?

Vorlesung vom 12.11.08: Typvariablen und Polymorphie

- Was ist **Polymorphie** in funktionalen Sprachen?
- Wo ist der Unterschied zu Java?
- Woran kann Typableitung **scheitern**?
- Was ist der Typ von $\lambda x y \rightarrow (x, 3) : [(, y)]$?
- Was ist ein Beispiel für einen Ausdruck vom Typ $[[a], \text{Int}]$?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist **einfache Rekursion**?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist **einfache Rekursion**?
- Was ist einfache Rekursion auf **Listen**?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist **einfache Rekursion**?
- Was ist einfache Rekursion auf **Listen**?
- Was ist der Unterschied zwischen `foldr` und `foldl`?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist **einfache Rekursion**?
- Was ist einfache Rekursion auf **Listen**?
- Was ist der Unterschied zwischen `foldr` und `foldl`?
- Was sind Typklassen in Haskell?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist **einfache Rekursion**?
- Was ist einfache Rekursion auf **Listen**?
- Was ist der Unterschied zwischen `foldr` und `foldl`?
- Was sind Typklassen in Haskell?
- Was ist der Unterschied zwischen Polymorphie und Typklassen?

Vorlesung vom 19.11.08: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist **einfache Rekursion**?
- Was ist einfache Rekursion auf **Listen**?
- Was ist der Unterschied zwischen `foldr` und `foldl`?
- Was sind Typklassen in Haskell?
- Was ist der Unterschied zwischen Polymorphie und Typklassen?

Vorlesung vom 26.11.08: Standarddatentypen

- Was sind die vier Phasen der Programmentwicklung?

Vorlesung vom 26.11.08: Standarddatentypen

- Was sind die vier Phasen der Programmentwicklung?
- Welche Standarddatentypen gibt es?

Vorlesung vom 26.11.08: Standarddatentypen

- Was sind die vier Phasen der Programmentwicklung?
- Welche Standarddatentypen gibt es?
- Wie definiere ich an den Blättern markierte Bäume?

Vorlesung vom 26.11.08: Standarddatentypen

- Was sind die vier Phasen der Programmentwicklung?
- Welche Standarddatentypen gibt es?
- Wie definiere ich an den Blättern markierte Bäume?
- Was ist das:

```
data X a = X a [X a]
```

Vorlesung vom 03.12.08: ADTs

- Was ist ein **abstrakter Datentyp**?

Vorlesung vom 03.12.08: ADTs

- Was ist ein **abstrakter Datentyp**?
- Was ist ein **Modul**, und was sind seine **Bestandteile**?

Vorlesung vom 03.12.08: ADTs

- Was ist ein **abstrakter Datentyp**?
- Was ist ein **Modul**, und was sind seine **Bestandteile**?
- Wieso sind geordnete Bäume ein abstrakter und kein algebraischer Datentyp?

Vorlesung vom 03.12.08: ADTs

- Was ist ein **abstrakter Datentyp**?
- Was ist ein **Modul**, und was sind seine **Bestandteile**?
- Wieso sind geordnete Bäume ein abstrakter und kein algebraischer Datentyp?
- Wieso sind Listen ein algebraischer und kein abstrakter Datentyp?

Vorlesung vom 03.12.08: ADTs

- Was ist ein **abstrakter Datentyp**?
- Was ist ein **Modul**, und was sind seine **Bestandteile**?
- Wieso sind geordnete Bäume ein abstrakter und kein algebraischer Datentyp?
- Wieso sind Listen ein algebraischer und kein abstrakter Datentyp?
- Haben abstrakte Datentypen einen verkapselten **Zustand**?

Vorlesung vom 10.12.08: Signaturen und Eigenschaften

- Was wäre ein ADT für `Array a` aus Aufgabenblatt 2?

Vorlesung vom 10.12.08: Signaturen und Eigenschaften

- Was wäre ein ADT für `Array a` aus Aufgabenblatt 2?
- Welche Operationen könnte ein ADT für Graphen haben?

Vorlesung vom 10.12.08: Signaturen und Eigenschaften

- Was wäre ein ADT für `Array a` aus Aufgabenblatt 2?
- Welche Operationen könnte ein ADT für Graphen haben?
- Welche Eigenschaften müssten gelten?

Vorlesung vom 17.12.08: Induktion und Beweis

- Was ist strukturelle **Induktion**?

Vorlesung vom 17.12.08: Induktion und Beweis

- Was ist strukturelle **Induktion**?
- Wie beweist man folgende Behauptung:

$$\text{map } f \text{ (map } g \text{ xs)} = \text{map } (f. g) \text{ xs}$$

Vorlesung vom 17.12.08: Induktion und Beweis

- Was ist strukturelle **Induktion**?
- Wie beweist man folgende Behauptung:

$$\text{map } f \text{ (map } g \text{ xs)} = \text{map } (f. g) \text{ xs}$$

- Wie beweist man folgende Behauptung:

$$\text{reverse (reverse xs ++ ys)} = \text{reverse ys ++ xs}$$

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (IO a) von anderen ADTs?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen ($\text{IO } a$) von anderen ADTs?
- Was sind die Operationen des ADT $\text{IO } a$?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?
 - ... die Kommandozeilenargumente lesen?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (IO a) von anderen ADTs?
- Was sind die Operationen des ADT IO a?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?
 - ... die Kommandozeilenargumente lesen?
 - ... eine Zeichenkette ausgeben?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?
 - ... die Kommandozeilenargumente lesen?
 - ... eine Zeichenkette ausgeben?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (IO a) von anderen ADTs?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen ($\text{IO } a$) von anderen ADTs?
- Was sind die Operationen des ADT $\text{IO } a$?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?
 - ... die Kommandozeilenargumente lesen?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?
 - ... die Kommandozeilenargumente lesen?
 - ... eine Zeichenkette ausgeben?

Vorlesung vom 07.01.09: Ein-/Ausgabe

- Was unterscheidet Aktionen (`IO a`) von anderen ADTs?
- Was sind die Operationen des ADT `IO a`?
- Wozu dient `return`?
- Welche **Eigenschaften** haben die Operationen?
- Wie kann man...
 - ... aus einer Datei lesen?
 - ... die Kommandozeilenargumente lesen?
 - ... eine Zeichenkette ausgeben?

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
- Dokumentation
 - z.B. im Vergleich zu Java's APIs
- Büchereien
- Noch viel im Fluß
 - Tools ändern sich
 - Zum Beispiel HGL
- Entwicklungsumgebungen

Andere Funktionale Sprachen

- **Standard ML (SML):**
 - Streng typisiert, strikte Auswertung
 - 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 & native
 - Nur ein Compiler (O'Caml)
 - <http://caml.inria.fr/>

Andere Funktionale Sprachen

- LISP & Scheme
 - Ungetypt/schwach getypt
 - Seiteneffekte
 - Viele effiziente Compiler, aber viele Dialekte
 - Auch industriell verwendet

Funktionale Programmierung in der Industrie

- Erlang
 - schwach typisiert, nebenläufig, strikt
 - Fa. Ericsson — Telekom-Anwendungen
- FL
 - ML-artige Sprache
 - Chip-Verifikation der Fa. Intel
- Galois Connections
 - Hochqualitätssoftware in Haskell
 - Hochsicherheitswebserver, Cryptoalgorithmen
- Verschiedene andere Gruppen

Perspektiven

- Funktionale Programmierung in 10 Jahren?
- Anwendungen:
 - Integration von XML, DBS (X#/Xen, Microsoft)
 - Integration in Rahmenwerke (F# & .Net, Microsoft)
 - Eingebettete domänenspezifische Sprachen
- Forschung:
 - Ausdrucksstärkere Typsysteme
 - für effiziente Implementierungen
 - und eingebaute Korrektheit (Typ als Spezifikation)
 - Parallelität?

Warum funktionale Programmierung nie Erfolg haben wird

- Programmierung nur kleiner Teil der SW-Entwicklung
- Unterstützung:
 - Libraries, Dokumentation, Entwicklungsumgebungen
- Nicht verbreitet — funktionale Programmierer zu teuer
- Konservatives Management
 - “Nobody ever got fired for buying IBM”

Was lernt uns funktionale Programmierung?

- Abstraktion
 - Denken in Algorithmen, nicht in Programmiersprachen

Was lernt uns funktionale Programmierung?

- Abstraktion
 - Denken in Algorithmen, nicht in Programmiersprachen
- Konzentration auf **wesentliche** Elemente moderner Programmierung:
 - Typisierung und Spezifikation
 - Datenabstraktion
 - Modularisierung und Dekomposition

Was lernt uns funktionale Programmierung?

- Abstraktion
 - Denken in Algorithmen, nicht in Programmiersprachen
- Konzentration auf **wesentliche** Elemente moderner Programmierung:
 - Typisierung und Spezifikation
 - Datenabstraktion
 - Modularisierung und Dekomposition
- Blick über den Tellerrand — Blick in die Zukunft
 - Studium \neq Programmierkurs — was kommt in 10 Jahren?

Hilfe!

- Haskell: primäre Entwicklungssprache am DFKI, FG SKS
 - Formale Programmentwicklung: <http://www.tzi.de/cofi/hets>
 - Sicherheit in der Robotik: <http://www.dfki.de/sks/sams>
- Wir suchen studentische Hilfskräfte
 - für diese Projekte
- Wir bieten:
 - Angenehmes Arbeitsumfeld
 - Interessante Tätigkeit
- Wir suchen Tutoren für PI3
 - im WS 09/10 — meldet Euch bei Berthold Hoffmann!

Tschüß!

