Praktische Informatik 3: Einführung in die Funktionale Programmierung
Vorlesung vom 09.02.11: Schlußbemerkungen

Christoph Lüth & Dennis Walter

Universität Bremen

Wintersemester 2010/11

Fahrplan

- Teil I: Funktionale Programmierung im Kleinen
- Teil II: Funktionale Programmierung im Großen
- Teil III: Funktionale Programmierung im richtigen Leben
  - Effizient Funktional Programmieren
  - Fallstudie: Kombinatoren
  - Eine Einführung in Scala
  - Rückblich & Ausblick

Organisatorisches

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

Beispiel

Beispiel

Definieren Sie eine Funktion leastSpaces, welche aus einer Liste von Zeichenketten diejenige zurückgibt, welche die wenigsten Leerzeichen enthält.

Inhalt

- Wiederholung
- Rückblick, Ausblick

Vorlesung vom 27.10.10: Grundbegriffe

- Was sind die Bestandteile einer Funktionsdefinition?

Vorlesung vom 27.10.10: Grundbegriffe

- Was sind die Bestandteile einer Funktionsdefinition?
- Was bedeutet referentielle Transparenz?

Vorlesung vom 27.10.10: Grundbegriffe

- Was sind die Bestandteile einer Funktionsdefinition?
- Was bedeutet referentielle Transparenz?
- Was ist eigentlich syntaktischer Zucker? (Beispiel?)

Vorlesung vom 03.11.10: Funktionen und Datentypen

- Welche Auswertungsstrategien gibt es, welche benutzt Haskell?

Vorlesung vom 03.11.10: Funktionen und Datentypen

- Welche Auswertungsstrategien gibt es, welche benutzt Haskell?
- Was bedeutet Striktheit?

Vorlesung vom 03.11.10: Funktionen und Datentypen

- Welche Auswertungsstrategien gibt es, welche benutzt Haskell?
- Was bedeutet Striktheit?
- Was ist die Abseitsregel?

Vorlesung vom 03.11.10: Funktionen und Datentypen

- Welche Auswertungsstrategien gibt es, welche benutzt Haskell?
- Was bedeutet Striktheit?
- Was ist die Abseitsregel?
- Welche vordefinierten Basisdatentypen gibt es in Haskell?

Vorlesung vom 10.11.10: Rekursive Datentypen

- Welches sind die wesentlichen Eigenschaften der Konstruktoren eines algebraischen Datentyps?

Vorlesung vom 10.11.10: Rekursive Datentypen

- Welches sind die wesentlichen Eigenschaften der Konstruktoren eines algebraischen Datentyps?
- Was ist das:

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

Vorlesung vom 10.11.10: Rekursive Datentypen

- Welches sind die wesentlichen Eigenschaften der Konstruktoren eines algebraischen Datentyps?
- Was ist das:

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

- Was bedeutet strukturelle Induktion?

Vorlesung vom 10.11.10: Rekursive Datentypen

- Welches sind die wesentlichen Eigenschaften der Konstruktoren eines algebraischen Datentyps?
- Was ist das:

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

- Was bedeutet strukturelle Induktion?
- Wie beweist man folgende Behauptung:

```
map f (map g xs) = map (f . g) xs
```

Vorlesung vom 17.11.10: Typvariablen und Polymorphie

- Was ist parametrische Polymorphie in funktionalen Sprachen?

Vorlesung vom 17.11.10: Typvariablen und Polymorphie

- Was ist parametrische Polymorphie in funktionalen Sprachen?
- Wo ist der Unterschied zu Polymorphie Java, und was entspricht der parametrischen Polymorphie in Java?

Vorlesung vom 17.11.10: Typvariablen und Polymorphie

- Was ist parametrische Polymorphie in funktionalen Sprachen?
- Wo ist der Unterschied zu Polymorphie Java, und was entspricht der parametrischen Polymorphie in Java?
- Welche Standarddatentypen gibt es in Haskell?

Vorlesung vom 17.11.10: Typvariablen und Polymorphie

- Was ist parametrische Polymorphie in funktionalen Sprachen?
- Wo ist der Unterschied zu Polymorphie Java, und was entspricht der parametrischen Polymorphie in Java?
- Welche Standarddatentypen gibt es in Haskell?
- Wozu dienen Typklassen in Haskell?

Vorlesung vom 24.11.10: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?

Vorlesung vom 24.11.10: Funktionen höherer Ordnung

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

Vorlesung vom 24.11.10: Funktionen höherer Ordnung

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

Vorlesung vom 24.11.10: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist einfache Rekursion?
- Was ist Listenkomprehension?
- Wie läßt sich Listenkomprehension durch map und filter darstellen?

Vorlesung vom 24.11.10: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist einfache Rekursion?
- Was ist Listenkomprehension?
- Wie läßt sich Listenkomprehension durch map und filter darstellen?
- Was ist der Unterschied zwischen foldr und foldl ?

Vorlesung vom 24.11.10: Funktionen höherer Ordnung

- Was ist eine Funktion höher Ordnung?
- Was ist einfache Rekursion?
- Was ist Listenkomprehension?
- Wie läßt sich Listenkomprehension durch map und filter darstellen?
- Was ist der Unterschied zwischen foldr und foldl ?
- ... und wann benutzt man welches?

Vorlesung vom 01.12.10: Typinferenz

- Woran kann Typableitung scheitern?

Vorlesung vom 01.12.10: Typinferenz

- Woran kann Typableitung scheitern?
- Was ist der Typ von $\lambda x\ y \rightarrow (x,\ 3)\ :\ [(\text{"␣"},\ y)]$

Vorlesung vom 01.12.10: Typinferenz

- Woran kann Typableitung scheitern?
- Was ist der Typ von $\lambda x\ y \rightarrow (x,\ 3)\ :\ [(\text{"␣"},\ y)]$
- ... und warum?

Vorlesung vom 01.12.10: Typinferenz

- Woran kann Typableitung scheitern?
- Was ist der Typ von $\lambda x\ y \rightarrow (x,\ 3)\ :\ [(\text{"}\sqcup\text{"},\ y)]$
- ... und warum?
- Was ist ein Beispiel für einen Ausdruck vom Typ $[([a],\ \text{Int})]$?

Vorlesung vom 08.12.2010: ADTs

- Was ist ein abstrakter Datentyp?

Vorlesung vom 08.12.2010: ADTs

- Was ist ein abstrakter Datentyp?
- Wieso sind geordnete Bäume ein abstrakter und kein algebraischer Datentyp?

Vorlesung vom 08.12.2010: ADTs

- Was ist ein abstrakter Datentyp?
- Wieso sind geordnete Bäume ein abstrakter und kein algebraischer Datentyp?
- Wieso sind Listen ein algebraischer und kein abstrakter Datentyp?

Vorlesung vom 08.12.2010: ADTs

- Was ist ein abstrakter Datentyp?
- 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 08.12.2010: ADTs

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

Vorlesung vom 05.01.11: Signaturen und Eigenschaften

- Was ist eine Signatur?

Vorlesung vom 05.01.11: Signaturen und Eigenschaften

- Was ist eine Signatur?
- Was sind Axiome?

Vorlesung vom 05.01.11: Signaturen und Eigenschaften

- Was ist eine Signatur?
- Was sind Axiome?
- Was wäre ein ADT für Array a — welche Operationen, welche Eigenschaften?

Vorlesung vom 12.01.11: Aktionen und Zustände

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

Vorlesung vom 12.01.11: Aktionen und Zustände

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

Vorlesung vom 12.01.11: Aktionen und Zustände

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

Vorlesung vom 12.01.11: Aktionen und Zustände

- 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 12.01.11: Aktionen und Zustände

- 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...

Vorlesung vom 12.01.11: Aktionen und Zustände

- 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 12.01.11: Aktionen und Zustände

- 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 12.01.11: Aktionen und Zustände

- 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 19.01.11: Effizienzaspekte

- Was ist Endrekursion?

Vorlesung vom 19.01.11: Effizienzaspekte

- Was ist Endrekursion?
- Was ist ein Speicherleck in Haskell?

Vorlesung vom 19.01.11: Effizienzaspekte

- Was ist Endrekursion?
- Was ist ein Speicherleck in Haskell?
- Wie vermeide ich Speicherlecks?

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 unterstütze 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 & 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
- Mangelnde Unterstützung:
  - Libraries, Dokumentation, Entwicklungsumgebungen
- Nicht verbreitet — funktionale Programmierer zu teuer
- Konservatives Management
  - “Nobody ever got fired for buying IBM”

Warum funktionale Programmierung lernen?

- Denken in Algorithmen, nicht in Programmiersprachen
- Abstraktion: Konzentration auf das Wesentliche
- Wesentliche Elemente moderner Programmierung:
  - Datenabstraktion und Funktionale Abstraktion
  - Modularisierung
  - Typisierung und Spezifikation
- 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 11/12 — meldet Euch bei Berthold Hoffmann!

Tschüß!