Funktionale Programmierung

LVA 185.A03, VU 2.0, ECTS 3.0
WS 2016/2017

– Vorbesprechung –

(Stand: 04.10.2016)

Jens Knoop

Technische Universität Wien
Institut für Computersprachen

Funktionale Programmierung

...komplementiert und rundet die grundlegenden Lehrveranstaltungen zu wichtigen Programmierparadigmen und -stilen ab.

- Objektorientierte Programmierung
  LVA 185.A01 Objektorientierte Programmiertechniken
  VU 2.0 ECTS 3.0 WS 2016/17
- Logikorientierte Programmierung
  LVA 185.A12 Logikprogrammierung und Constraints
  VU 4.0 ECTS 6.0 WS 2016/17
- **Funktionale Programmierung**
  LVA 185.A03 Funktionale Programmierung
  VU 2.0 ECTS 3.0 WS 2016/17

...die in zugehörigen fortgeschrittenen Lehrveranstaltungen fortgeführt und vertieft werden.

Themen der Lehrveranstaltung

- Funktionaler Programmierstil mit Programmen als Systeme (wechselweise) rekursiver Rechenvorschriften.
- Lambda-Kalkül als Grundlage der semantischen Fundierung funktionaler Programmiersprachen.
- Auswertungsstrategien für Ausdrücke und Programme, insbesondere die Auswertungsstrategien lazy und eager.
- Funktionen höherer Ordnung, Programmieren mit Funktionen als Argument und Resultat.
- Polymorphie in den Varianten parametrisch und überladen.
- Ströme, Stromverarbeitung, Programmieren mit 'unendlichen' Listen.
- ...

...und die Umsetzung&Anwendung dieser Konzepte in Haskell.

Gliederung der Lehrveranstaltung

- Teil I: Einführung
- Teil II: Applikative Programmierung
- Teil III: Funktionale Programmierung
- Teil IV: Fundierung funktionaler Programmierung
- Teil V: Ergänzungen und weiterführende Konzepte
- Teil VI: Resümee und Perspektiven
- Literaturverzeichnis
- Anhänge

Gliederung im Detail (1)

Teil I: Einführung

- Kap. 1: Motivation
  - 1.1 Ein Beispiel sagt (oft) mehr als 1000 Worte
  - 1.2 Funktionale Programmierung: Warum? Warum mit Haskell?
  - 1.3 Nützliche Werkzeuge: Hugs, GHC, Hoogle und Hayoo, Leksah
  - 1.4 Literaturverzeichnis, Leseempfehlungen
- Kap. 2: Grundlagen
  - 2.1 Elementare Datentypen
  - 2.2 Tupel und Listen
  - 2.3 Funktionen
  - 2.4 Funktionssignaturen, -terme und -stelligkeiten
  - 2.5 Mehr Würze: Curry bitte!
  - 2.6 Programmlayout und Abseitsregel
  - 2.7 Literaturverzeichnis, Leseempfehlungen

Gliederung im Detail (2)

Gliederung im Detail (3)

Gliederung im Detail (4)

Gliederung im Detail (5)

Teil V: Ergänzungen und weiterführende Konzepte

- Kap. 11: Muster und mehr
- Kap. 12: Module
  - 12.1 Programmieren im Großen
  - 12.2 Module in Haskell
  - 12.3 Anwendung: Abstrakte Datentypen
  - 12.4 Literaturverzeichnis, Leseempfehlungen
- Kap. 13: Typüberprüfung, Typinferenz
  - 13.1 Monomorphe Typüberprüfung
  - 13.2 Polymorphe Typüberprüfung
  - 13.3 Typsysteme und Typinferenz
  - 13.4 Literaturverzeichnis, Leseempfehlungen

Gliederung im Detail (6)

Gliederung im Detail (7)

Teil VI: Resümee und Perspektiven

- Kap. 17: Abschluss und Ausblick
- Literaturverzeichnis
- Anhänge
  - A Formale Rechenmodelle
    - A.1 Turing-Maschinen
    - A.2 Markov-Algorithmen
    - A.3 Primitiv-rekursive Funktionen
    - A.4 μ-rekursive Funktionen
    - A.5 Literaturverzeichnis, Leseempfehlungen
  - B Auswertungsordnungen
    - B.1 Applikative vs. normale Auswertungsordnung
  - C Datentypdeklarationen in Pascal
  - D Hinweise zur schriftlichen Prüfung

Funktionale Programmierung: Warum?

"Can programming be liberated from the von Neumann style?"

John W. Backus, 1978

Funktionale Programmierung: Warum?

"Can programming be liberated from the von Neumann style?"

John W. Backus, 1978

Dieser Frage wollen wir in der LVA nachgehen!

Ausgangspunkt und Grundlage:

- John W. Backus. Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs. Communications of the ACM 21(8):613-641, 1978. (Turing Award Speech)

Ziele der Lehrveranstaltung

Theoretische und praktische Einsicht in und Verständnis für

- die grundlegenden und zentralen Konzepte und Prinzipien funktionaler Programmierung und ihrer theoretischen Fundierung.
- die Umsetzung dieser Konzepte in einer state-of-the-art Programmiersprache, hier Haskell.
- den sinnvollen und angemessenen Einsatz funktionaler Programmierung und ihrer Konzepte für die Lösung programmiertechnischer Aufgaben, darunter auch Tipps, Tricks und mehr!

Protagonisten und Anhänger sind überzeugt:

Functional Programming is **Fun!**

Ziele der Lehrveranstaltung

Theoretische und praktische Einsicht in und Verständnis für

- die grundlegenden und zentralen Konzepte und Prinzipien funktionaler Programmierung und ihrer theoretischen Fundierung.
- die Umsetzung dieser Konzepte in einer state-of-the-art Programmiersprache, hier Haskell.
- den sinnvollen und angemessenen Einsatz funktionaler Programmierung und ihrer Konzepte für die Lösung programmiertechnischer Aufgaben, darunter auch Tipps, Tricks und mehr!

Protagonisten und Anhänger sind überzeugt:

Functional Programming is **Fun!**

...dem werden wir nachgehen!

Wenn ja, warum?

Eine Antwort:

...weil funktionale Programmierung
etwas von der Eleganz der Mathematik
in die Programmierung bringt!

Peter Pepper. *Funktionale Programmierung in OPAL, ML, Haskell und Gofer*. Springer-V., 2003.

Ein "Klassiker" in diesem Zusammenhang:

- John Hughes. Why Functional Programming Matters. The Computer Journal 32(2):98-107, 1989.

Für Einstieg & Motivation bes. empfohlen (1)

Was Sie erwarten können:

Konrad Hinsen. *The Promises of Functional Programming*. Computing in Science and Engineering 11(4):86-90, 2009.
...adopting a functional programming style could make your programs more robust, more compact, and more easily parallelizable.

Konstantin Läufer, Geoge K. Thiruvathukal. *The Promises of Typed, Pure, and Lazy Functional Programming: Part II*. Computing in Science and Engineering 11(5):68-75, 2009.
...this second installment picks up where Konrad Hinsen's article "The Promises of Functional Programming" [...] left off, covering static type inference and lazy evaluation in functional programming languages.

Für Einstieg & Motivation bes. empfohlen (2)

Warum es sich für Sie lohnt:

Yaron Minsky. *OCaml for the Masses*. Communications of the ACM 54(11):53-58, 2011.
...why the next language you learn should be functional.

Wo Sie es anwenden können (1)

Von Wissenschaft...

Jerzy Karczmarczuk. *Scientific Computation and Functional Programming*. Computing in Science and Engineering 1(3):64-72, 1999.
...modern functional programming languages and lazy functional techniques are useful for describing and implementing abstract mathematical objects in quantum mechanics.

Noah M. Daniels, Andrew Gallant, Norman Ramsey. *Experience Report: Haskell in Computational Biology*. In Proc. 17th ACM SIGPLAN International Conference on Functional Programming (ICFP 2012), 227-234, 2012.
...Haskell gives computational biologists the flexibility and rapid prototyping of a scripting language, plus the performance of native code.

Wo Sie es anwenden können (2)

...über Wirtschaft

Curt J. Simpson. *Experience Report: Haskell in the "Real World": Writing a Commercial Application in a Lazy Functional Language*. In Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming (ICFP 2009), 185-190, 2009.

...describe[s] the initial attempt of experienced business software developers with minimal functional programming background to writ a non-trivial, business-critical application entirely in Haskell. ...discuss[es] the advantages and difficulties of Haskell in these circumstances, with a particular focus on issues that commercial developers find important but that may receive less attention from the academic community.

Wo Sie es anwenden können (3)

...bis Systemprogrammierung:

Iavor S. Dachki, Thomas Hallgren, Mark P. Jones, Rebekah Leslie, Andrew Tolmach. *Writing System Software in a Functional Language: An Experience Report*. In Proceedings of the 4th International Workshop on Programming Languages and Operating Systems (PLOS 2007), Article No. 1, 5 pages, 2007.

...we describe our experience developing a prototype operating system, House, in which the kernel, device drivers, and even a simple GUI, are all written in Haskell.

Weitere sehr lesenswerte Einstiegsartikel (1)

- John Hughes. *Why Functional Programming Matters*. The Computer Journal 32(2):98-107, 1989.
- Philip Wadler. *The Essence of Functional Programming*. In Conference Record of the 19th Annual Symposium on Principles of Programming Languages (POPL'92), 1-14, 1992.
- Paul Hudak. *Conception, Evolution and Applications of Functional Programming Languages*. Communications of the ACM 21(3):359-411, 1989.
- Benjamin Goldberg. *Functional Programming Languages*. ACM Computing Surveys 28(1):249-251, 1996.

Weitere sehr lesenswerte Einstiegsartikel (2)


Mark P. Jones. *Functional Thinking*. Lecture at the 6th International Summer School on Advanced Functional Programming, Boxmeer, The Netherlands, 2008.


Richard Bird. *Thinking Functionally with Haskell*. Cambridge University Press, 2015.


Neal Ford. *Functional Thinking: Why Functional Programming is on the Rise*. IBM developerWorks, 11 pages, 2013.
`www.ibm.com/developerworks/java/library/j-ft20/j-ft20-pdf.pdf`

Weitere sehr lesenswerte Einstiegsartikel (3)

- Philip Wadler. *An Angry Half-dozen*. ACM SIGPLAN Notices 33(2):25-30, 1998.
- Philip Wadler. *Why no one uses Functional Languages*. ACM SIGPLAN Notices 33(8):23-27, 1998.
- Leo A. Meyerovich, Ariel S. Rabkin. *Empirical Analysis of Programming Language Adoption*. In Proceedings of the 2013 ACM SIGPLAN International Conference on Object-Oriented Programming Systems Languages and Applications (SPLASH 2013), 1-18, 2013.
- Paul Hudak, Mark P. Jones. *Haskell vs. Ada vs. C++ vs... An Experiment in Software Prototyping Productivity*. Research Report YALEU/DCS/RR-1049, Yale University, 1994. `www.cs.yale.edu/publications/techreports.html#1994`

Ausgewählte Lehrbücher

- Simon Thompson. *Haskell: The Craft of Functional Programming*. Addison-Wesley (Pearson), 3rd edition, 2011.
- Ernst-Erich Doberkat. *Haskell: Eine Einführung für Objektorientierte*. Oldenbourg Verlag, 2012.
- Richard Bird. *Introduction to Functional Programming using Haskell*. Prentice-Hall, 2nd edition, 1998.
- Graham Hutton. *Programming in Haskell*. Cambridge University Press, 2007.
- Peter Pepper. *Funktionale Programmierung in OPAL, ML, Haskell und Gofer*. Springer-Verlag, 2. Auflage, 2003.
- Manuel Chakravarty, Gabriele Keller. *Einführung in die Programmierung mit Haskell*. Pearson Studium, 2004.
- Paul Hudak. *The Haskell School of Expression: Learning Functional Programming through Multimedia*. Cambridge University Press, 2004.
- ...

Ausgewählte Informationsquellen im Web

- Haskell-Homepage: `www.haskell.org/`
- Haskell-Wiki: `wiki.haskell.org/Haskell/`
- Haskell-Tutorial: `www.haskell.org/tutorial/`
- Hugs-Interpretierer: `www.haskell.org/hugs`
- ...

Ausgewählte Informationsquellen im Web

- Haskell-Homepage: `www.haskell.org/`
- Haskell-Wiki: `wiki.haskell.org/Haskell/`
- Haskell-Tutorial: `www.haskell.org/tutorial/`
- Hugs-Interpretierer: `www.haskell.org/hugs`
- ...

Viele weitere Hinweise zu Lehrbüchern, zu grundlegenden, weiterführenden und vertiefenden Originalarbeiten sind zu jedem Kapitel in den Vorlesungsunterlagen angegeben.

Wichtige wiss. Zeitschriften und Konferenzen

...zur Publikation von Forschungsergebnissen im Umfeld funktionaler Programmierung sind besonders:

- Zeitschrift:
  - The Journal of Functional Programming. Matthias Felleisen, Jeremy Gibbons (Hrsg.), Cambridge, UK. `www.cambridge.org/jfp`
- Konferenzserie:
  - ACM SIGPLAN International Conference Series on Functional Programming (ICFP)
    `www.sigplan.org/Conferences/ICFP`

Aufbau und Ablauf der Lehrveranstaltung

Vier Kernbestandteile:

- Vorlesung (regelmäßig dienstags von 8-10 Uhr)
- Plenumsübung "Haskell Live" (regelmäßig freitags von 14-15 Uhr)
- Individual-Feedback "Haskell Private" (nach Vereinbarung und Kapazität; konzentriert im November und Dezember)
- Übung mit Einzelabgaben (im Regelfall wöchentliche Abgaben)

Schriftliche Prüfung

- Schriftliche 90-min. Prüfung (sog. Klausur) über Vorlesungs- und Übungsstoff und einen wissenschaftlichen Artikel, den Sie sich selbstständig im Lauf der Vorlesungszeit erschließen, und zwar:

  John W. Backus. *Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs*. Communications of the ACM 21(8):613-641, 1978. (Turing Award Speech)

  (Zugänglich aus TUW-Netz in ACM Digital Library: `http://dl.acm.org/citation.cfm?id=359579`)

  Eine Anmeldung zur Klausur ist zwingend erforderlich und erfolgt über TISS.

Anmeldung zur LVA

Die Anmeldung

- erfolgt über TISS bis spätestens Freitag, 14.10.2016, 12:00 Uhr.

Voraussetzung einer validen Anmeldung:

- Erfolgreich abgeschlossene STEOP (Ausnahme: Erasmus+ - Studierende).

Studierende mit valider Anmeldung erhalten

- ein Benutzerkonto auf der Maschine `g0.complang.tuwien.ac.at`
- Benutzerkennung und erstes Losungswort per elektronischer Nachricht an ihre Standardadresse `e<Matr.Nr>@student.tuwien.ac.at`

zur Bearbeitung und Abgabe von Übungsaufgaben.

Vorlesung und Übungsaufgaben

- Vorlesung dienstags von 08:15 Uhr - ca. 09:45 Uhr im Informatikhörsaal, Treitlstr. (Ziel: Vorlesungsteil bis Ende Dezember abschließen!)
- Beginnend (vorauss.) mit Mittwoch, den 19.10.2016, im Regelfall jeden Mittwoch ein neues Aufgabenblatt (abrufbar auf der Webseite der LVA).
- Abgabe von Lösungen: Eine Woche nach Ausgabe bis 15:00 Uhr, die automatisch aus Home-Verzeichnis (top-level! Nicht in Unterverzeichnissen!) abgesammelt und unter Benutzung von Hugs überprüft werden.
- Nachträgliche Abgabe von Lösungen bzw. verbesserten Lösungen: Eine Woche nach Erstabgabe bis 15:00 Uhr, die ebenfalls automatisch abgesammelt werden.
- Insgesamt ca. 10 Aufgabenblätter.
- Gesamtpunktzahl pro Aufgabenblatt gemäß der Formel:
  `(Punkte Erstabgabe + Punkte Zweitabgabe) / 2`

Benützung von eigenen und TUW-Rechnern

- Server für die praktischen Programmierübungen: `g0.complang.tuwien.ac.at`
- Terminals für TUW-Rechner sind im Labor Argentinierstraße 8, Erdgeschoss im Innenhof verfügbar.
- Arbeiten auf anderen, eigenen Rechnern z.B. zu Hause ist möglich.
- Abgaben allerdings ausschließlich auf `g0.complang.tuwien.ac.at`
- Nötige Software: Hugs (frei verfügbar)
- **Wichtig:** Abgaben werden auf der g0 ausschließlich unter Hugs getestet. Überzeugen Sie sich deshalb stets von der gewünschten Funktionalität Ihrer Programmierlösungen auf der g0 unter Hugs!

Beurteilung

- Je zur Hälfte gewichtet die Beurteilung der praktischen Übungen und das Ergebnis der schriftlichen Prüfung.

  Hauptklausurtermin: vorauss. Do, 19.01.2017 (Anmeldung erforderlich!); danach 3 Nachtragsklausurtermine zu Beginn (vorauss. 03.03.2017), in der Mitte (vorauss. 27.04.2017) und zu Ende der Vorlesungszeit im SS 2017 (vorauss. 23.06.2017) (Anmeldung jeweils erforderlich!).

  Für jeden Klausurantritt wird ein Zeugnis ausgestellt.

  Nach Ablauf der Vorlesungszeit im SS 2017 keine weiteren Nachtragstermine. Ausstellung ggf. dann noch offener Zeugnisse im Juli/August 2017 sowie vorher nach jedem Klausurantritt).

  Merken Sie sich diese Termine bitte vor und planen Sie entsprechend!

Beurteilung (fgs.)

- Positive Note nur, wenn beide Teile positiv bestanden.
- Punkte für Übungsaufgaben: max. 100/Abgabe, ca. 10 Abgaben.
- Mindestens 50% der Punkte für positive Übungsbeurteilung.
- Halbe Punkteanzahl für nachträgliche Abgaben.
- Nachträgliche Abgaben können die Punkteanzahl positiv und negativ (bei Verschlechterung der Lösung) beeinflussen.
- Wichtig: Auch wenn Sie schon beim ersten Mal 100 Punkte hatten, müssen Sie für die Nachabgabe eine Lösung zum Absammeln vorhalten (z.B. die Lösung der Erstabgabe!)!
- Schriftliche Prüfung: Keine Hilfsmittel, Anmeldung über TISS erforderlich!

Mitveranstalter, Tutoren

- Mitveranstalter:
  - [1] Ass.Prof. Dipl.-Ing. Dr. techn. Ulrich Neumerkel
- Tutoren:
  - [2] Jakob Kremsner
  - [3] Bruno Tiefengraber
  - [4] Jannik Vierling, BSc
  - [5] Lukas Winkler

Bei Fragen und Problemen

Insbesondere:

- Webseite der LVA:
  `www.complang.tuwien.ac.at/knoop/fp185A03.html`
- Plenumsübung "Haskell Live".
- Individual-Feedback "Haskell Private".
- Tutorensprechstunde und -betreuung im Labor
  (Informationen zu den genauen Zeiten finden Sie auf der Webseite der Vorlesung).

Nächste Vorlesungs- und Übungstermine

Vorlesung, Haskell Live, Haskell Private:

- Vorlesung: Do, 06.10.2016, 16:15-17:45 Uhr im EI 3 Sahulka Hörsaal Gußhausstr. 25-29; Di, 11.10.2016, 08:15-09:45 Uhr im Informatikhörsaal, Treitlstr.; Mi, 12.10.2016, 08:15-09:45 Uhr im GM 2 Radinger Hörsaal, Getreidemarkt 9; Do, 13.10.2016, 16:00-17:30 Uhr im EI 9 Hlawka Hörsaal, Gußhausstr. 25-29.
- Haskell Live: Fr, 14.10.&21.10.2016, 14:15-15:00 Uhr im Informatikhörsaal, Treitlstr.
- Haskell Private: Nach Vereinbarung mit den Tutoren.
- Weitere Termine: Siehe Webseite der Lehrveranstaltung.

Übung:

- Erstes Aufgabenblatt: vorauss. Mo, 17.10.2016.
- Erste Abgabe: vorauss. Mi, 26.10.2016, 15:00 Uhr.

Wir, die FP-Teammitglieder, wünschen Ihnen

...viel Erfolg für diese Lehrveranstaltung und dass Sie auch langfristig davon profitieren!

Zu guter Letzt:

Die Vorlesung lebt mit Ihnen! Ihre Rückmeldungen, Anregungen, Verbesserungsvorschläge sind willkommen! Natürlich auch, wenn Ihnen etwas gut gefallen hat!