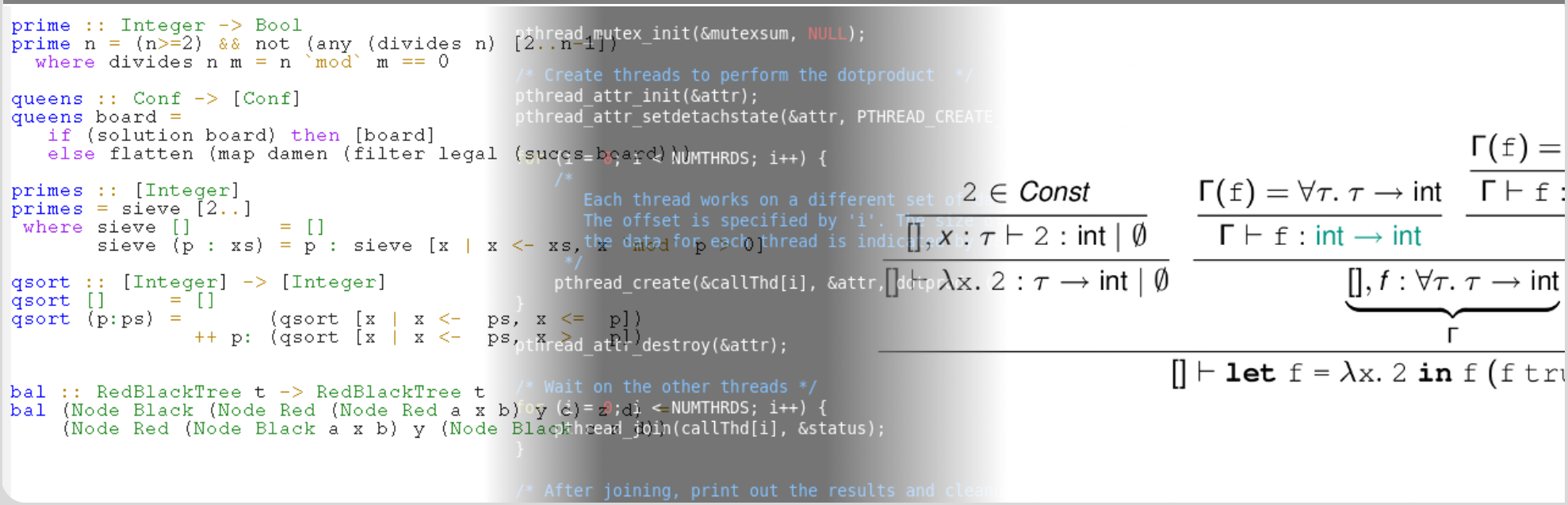
Karlsruher Institut für Technologie

Programmierparadigmen

Prof. Dr.-Ing. Gregor Snelting | WS 2013/2014

LEHRSTUHL PROGRAMMIERPARADIGMEN



Organisatorisches

Vorlesung

Vorlesung:	Zeit: Mittwoch, 14:00 – 15:30 Uhr Ort: Hertz Hörsaal (Raum 126, Geb. 10.11)
Vorlesung:	Zeit: Freitag, 14:00 – 15:30 Uhr, $\approx$ 14-tägig Ort: Hertz Hörsaal (Raum 126, Geb. 10.11) genaue Termine: Ankündigungen/Website
Prüfung:	Klausur (120 Minuten): 10.04.2014, 14:00 Uhr
Ansprechpartner:	Der eigene Tutor/Übungsleiter
Fragen, etc:	In der Übung, im Forum, Sprechstunde: Dienstag, 13:00 – 14:00 Uhr

Übungen und Tutorien

Übungsaufgaben: wöchentliche Aufgabenblätter

Ausgabe: Übungswebseite, Blatt 0 schon veröffentlicht!

Bearbeitungszeit: 1 Woche (i.d.R.)

Abgabe: Donnerstags um 11:30h (Briefkasten im Info-Bau UG)

Korrektur: Tutoren

Gruppen: 8 Übungsgruppen, wöchentlich

Einteilung via WebInScribe
(`https://webinscribe.ira.uka.de/propa2013`)

Tutorien: direkt nach Übungsgruppen

Wiederholung und Fragen zur Vorlesung

Teilnahme an den Übungen und Bearbeitung der Aufgabenblätter wird dringend empfohlen.

Termine der Übungsgruppen

Nr.	Raum	Tag	Zeit
1	SR 131 (50.34)	Mo	09.45h
2	SR 236 (50.34)	Mo	11.30h
3	SR 131 (50.34)	Mo	14.00h
4	SR 236 (50.34)	Mo	17.30h
5	SR 236 (50.34)	Di	09.45h
6	SR 131 (50.34)	Di	11.30h
7	SR 301 (50.34)	Di	11.30h
8	SR 131 (50.34)	Di	14.00h

Internet-Adressen

Lehrstuhl-Homepage:

Snelting: `http://pp.info.uni-karlsruhe.de/`

Reussner/Hummel: `http://sdq.ipd.kit.edu/`

Vorlesungs-Homepage:

`http://pp.info.uni-karlsruhe.de/lehre/WS201314/paradigmen/`

Übungs-Homepage:

`http://pp.info.uni-karlsruhe.de/lehre/WS201314/paradigmen/uebung/`

WebInScribe:

`https://webinscribe.ira.uka.de/propa2013`

Voraussetzungen

Karlsruher Institut für Technologie

Formale Voraussetzungen

- Modul Theoretische Grundlagen der Informatik

Kenntnisse:

- Objektorientierte Programmierung
- Grundlegende Algorithmen und Datenstrukturen

Bereitschaft:

- zu aktiver und eigenständiger Mitarbeit
- zur Benutzung weiterer Quellen (Bücher, API, . . .)

Arbeitsaufwand

Arbeitsaufwand:

- Laut BA-Ordnung 6 LP (1 LP $\widehat{=}$ 30 Std.)
  - ca. 33 Std. Vorlesung
  - ca. 33 Std. Nachbearbeitung
  - ca. 16 Std. Übung
  - ca. 16 Std. Tutorium
  - ca. 40 Std. Übungsaufgaben
  - ca. 2 Std. für schriftliche Prüfung (120 Minuten)
  - ca. 40 Std. Prüfungsvorbereitung

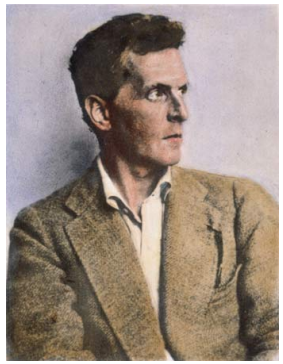
$\sum$ etwa 180 Arbeitsstunden

Programmierparadigmen – Inhalt

- Funktionale Programmierung
- λ-Kalkül, Typsysteme
- Logische Programmierung
- Parallel-Programmierung – Jun.-Prof. Dr. Oliver Hummel
- Compiler-Grundlagen

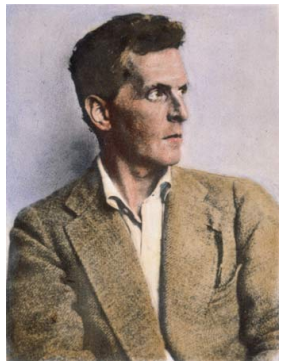
Warum Sprachtechnologie?

„Die Grenzen meiner Sprache sind die Grenzen meiner Welt“
(Wittgenstein, Tractatus)



Warum Sprachtechnologie?

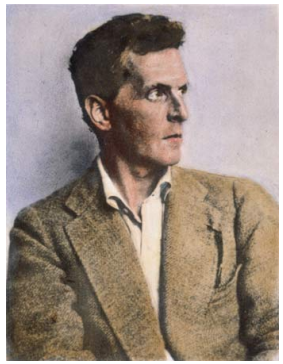
„Die Grenzen meiner Sprache sind die Grenzen meiner Welt“
(Wittgenstein, Tractatus)



„Programmiersprachliche Konstrukte sind gefrorenes Wissen über gute Softwareentwicklung“
(G. Kahn)

Warum Sprachtechnologie?

„Die Grenzen meiner Sprache sind die Grenzen meiner Welt“
(Wittgenstein, Tractatus)



„Programmiersprachliche Konstrukte sind gefrorenes Wissen über gute Softwareentwicklung“
(G. Kahn)

„Some believed we lacked the programming language to describe your perfect world“
(Agent Smith, The Matrix)

Wozu Programmierparadigmen?

- Übersicht über verschiedene Paradigmen
- Nicht immmer nur Java!
- λ-Kalkül und Typinferenz legen wichtige Grundlagen für moderne Sprachtechnologie
- KIT ist eine Forschungsuniversität.

Vorläufiger Zeitplan

Funktionale Programmierung

01	Mi. 23.10.	Fr. 25.10.	Haskell, rekursive Funktionen, Listen
02	Mi. 30.10.		Typen, Funktionale, Kombinatoren
03	Mi. 06.11.	Fr. 08.11.	rekursive Datentypen, Anwendungen
04	Mi. 13.11.		Typklassen, “lazy” Evaluation

λ-Kalkül, Typtheorie

04		Fr. 15.11.	Auswertungsstrategien, Church-Zahlen
05	Mi. 20.11.		Auswertungsstrategien, Church-Zahlen
06	Mi. 27.11.	Fr. 29.11	Rekursion, Normalisierung, Typisierung
07	Mi. 04.12.		Typinferenz, `let`-Polymorphismus

Logische Programmierung

07		Fr. 06.12.	
08	Mi. 11.12.		
09	Mi. 18.12.		
10	Mi. 08.01.		

Vorläufiger Zeitplan

Parallel-Programmierung

10		Fr. 10.01.
11	Mi. 15.01.	
12	Mi. 22.01.	Fr. 24.01.
13	Mi. 29.01.	

Compiler-Grundlagen

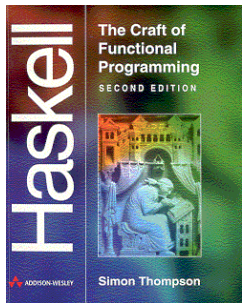
13		Fr. 31.01.
14	Mi. 05.02.	Fr. 07.02.
15		Fr. 14.02.

Literatur

Funktionale Programmierung

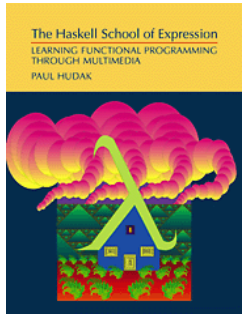
Strukturierte Einführung mittels Haskell

Simon Thompson, The Craft of Functional Programming



Anwendungs- und beispielorientiert

Paul Hudak, The Haskell School of Expression

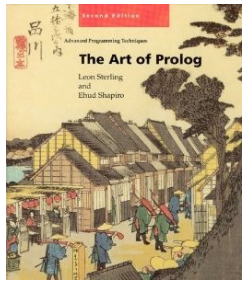


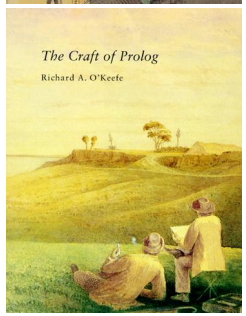
Anwendungen aus Semantik und Übersetzerbau

Gert Smolka, Programmierung – eine Einführung in die Informatik mit Standard ML

Literatur

Logische Programmierung





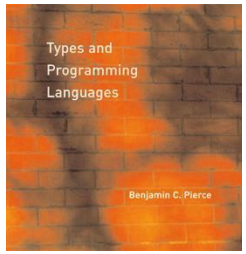
Lehrbuch, viele Beispiele

Ehud Shapiro, Leon Sterling, The Art of Prolog

Für Fortgeschrittene

R. O'Keefe, The Craft of Prolog

Theorie: λ-Kalkül, Typtheorie



Referenzwerk, Theorie und Implementierung

Benjamin C. Pierce, Types and Programming Languages