Leitfäden der Informatik
Peter Thiemann
Grundlagen der funktionalen
Programmierung

Leitfäden der Informatik

Herausgegeben von

Prof. Dr. Hans-Jürgen Appelrath, Oldenburg

Prof. Dr. Volker Claus, Stuttgart

Prof. Dr. Günter Hotz, Saarbrücken

Prof. Dr. Lutz Richter, Zürich

Prof. Dr. Wolffried Stucky, Karlsruhe

Prof. Dr. Klaus Waldschmidt, Frankfurt

Die Leitfäden der Informatik behandeln

- Themen aus der Theoretischen, Praktischen und Technischen Informatik entsprechend dem aktuellen Stand der Wissenschaft in einer systematischen und fundierten Darstellung des jeweiligen Gebietes.
- Methoden und Ergebnisse der Informatik, aufgearbeitet und dargestellt aus Sicht der Anwendungen in einer für Anwender verständlichen, exakten und präzisen Form.

Die Bände der Reihe wenden sich zum einen als Grundlage und Ergänzung zu Vorlesungen der Informatik an Studierende und Lehrende in Informatik-Studiengängen an Hochschulen, zum anderen an „Praktiker“, die sich einen Überblick über die Anwendungen der Informatik(-Methoden) verschaffen wollen; sie dienen aber auch in Wirtschaft, Industrie und Verwaltung tätigen Informatikern und Informatikerinnen zur Fortbildung in praxisrelevanten Fragestellungen ihres Faches.

Grundlagen der funktionalen Programmierung

Von Dr. rer. nat. Peter Thiemann
Universität Tübingen

B. G. Teubner Stuttgart 1994

Dr. rer. nat. Peter Thiemann

Geboren 1964 in Neuss/Rhein. Studium der Informatik mit Nebenfach Mathematik an der RWTH Aachen (1983–1987). Seit 1989 wiss. Mitarbeiter am Wilhelm-Schickard-Institut für Informatik der Eberhard-Karls-Universität Tübingen, Promotion 1991 bei H. Klaeren.

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Thiemann, Peter:

Grundlagen der funktionalen Programmierung / von Peter

Thiemann. – Stuttgart : Teubner, 1994

(Leitfäden der Informatik)

ISBN-13: 978-3-519-02137-7 e-ISBN-13: 978-3-322-89207-2

DOI: 10.1007/978-3-322-89207-2

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt besonders für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

© B. G. Teubner Stuttgart 1994

Gesamtherstellung: Zehnersche Buchdruckerei GmbH, Speyer

Einband: Peter Pfitz, Stuttgart

Vorwort

Das vorliegende Buch entstand aus einer Reihe von Vorlesungen, die der Autor an der Eberhard-Karls-Universität Tübingen unter dem Titel „Einführung in die funktionale Programmierung“ gehalten hat. Die Zielgruppe der Vorlesung sind Studenten im Hauptstudium, die Informatik als Haupt- oder Nebenfach belegen. Voraussetzungen zum Verständnis des Buches sind die Kenntnis von Grundbegriffen der Informatik und Programmierung.

Die Vorlesung, wie auch das Buch, besteht aus zwei Teilen. Der erste Teil umfaßt die Kapitel 1 bis 8 und ist praktisch orientiert. Er gibt eine kurze Einführung in die rein-funktionale Programmiersprache Gofer mit grundlegenden Programmiertechniken und Methoden der Verifikation und Transformation von Programmen gefolgt von einem kurzen Ausblick auf fortgeschrittene Techniken und weiterführende Konzepte. Insbesondere wird auf Typklassen, Konstruktorklassen und Monaden, sowie rein-funktionale Ein- und Ausgabe eingegangen. Typklassen und Konstruktorklassen erlauben die kontrollierte Überladung von benutzerdefinierten Funktionen. Monaden ermöglichen unter anderem die Integration von Variablen im herkömmlichen Sinn in rein-funktionale Programmiersprachen.

Im zweiten Teil (Kap. 9 bis 15) werden verschiedene Modelle für Semantik und Ausführung funktionaler Programmiersprachen vorgestellt. Der Teil umfaßt eine Einführung in die Bereichstheorie, universelle Algebra, operationelle und denotationelle Semantik, und den Lambda-Kalkül. Ferner werden Typen und ihre Semantik, die automatische Rekonstruktion von Typen, sowie Grundbegriffe der abstrakten Interpretation und Striktheitsanalyse behandelt. Damit verzahnt werden Implementierungstechniken für funktionale Programmiersprachen auf einer abstrakten Ebene diskutiert. Aufbauend auf einem einfachen Maschinenmodell für strikte funktionale Sprachen wird der Leser Schritt für Schritt an ein paralleles Maschinenmodell für Sprachen mit nicht-strikten Funktionen und Datenstrukturen herangeführt. Aufbauend auf die Darstellung des Lambda-Kalküls werden die SECD-Maschine und die SKI-Kombinatorgraphreduktion als weitere Implementierungstechniken vorgestellt.

Zur Vorstellung der Konzepte funktionaler Programmiersprachen eignet sich die Sprache Gofer besonders. Gofer ist einerseits klein genug, um ein schnelles Erlernen der Sprache zu ermöglichen (Gofer wurde bereits an einigen Hochschulen für die Anfängerausbildung benutzt). Andererseits hat Gofer alle Merkmale einer modernen funktionalen Programmiersprache. Zusätzlich ist die Teilmenge von Gofer, der im vorliegenden Buch besprochen wird, gleichzeitig eine Teilmenge der Programmiersprache Haskell, die der de-facto Standard rein-funktionaler Programmiersprachen ist. Also kann der Text gleichzeitig als Einführung in Gofer und als Heranführung an Haskell verstanden werden. Die Besonderheiten der beiden Sprachen und einige Unterschiede werden diskutiert. Ein weiteres Argument für Gofer ist die freie Verfügbarkeit einer guten Implementierung in Form

eines Interpretierers auf handelsüblichen Arbeitsplatzrechnern und Personalcomputern. Demgegenüber sind Haskell-Systeme schon wegen ihres Umfangs besser für Arbeitsplatzrechner geeignet und sie stehen auch nur für bestimmte Rechner zur Verfügung. Überdies stellen die meisten Haskell-Systeme lediglich einen Übersetzer zur Verfügung, womit ein interaktives Arbeiten erschwert wird.

Praktische Hinweise zum Gofer-System befinden sich im Anhang: Vollständige Syntaxdiagramme (Anhang B), eine Kurzreferenz (Anhang C), Verfügbarkeit von Gofer- und Haskell-Implementierungen (Anhang D) und Hinweise zur Bedienung des Gofer-Interpretierers (Anhang E).

Die Informatik ist eine Wissenschaft, in der Anglizismen an der Tagesordnung sind, da die meisten Publikationen nur in englischer Sprache erscheinen. Im vorliegenden Text versucht der Autor, deutsche Ausdrücke zu verwenden, wenn sie den Sachverhalt wirklich treffen und den Sprachfluß nicht hemmen. Auf jeden Fall wird der englische Fachbegriff eingeführt, um dem interessierten Leser den Einstieg in die Fachliteratur zu erleichtern.

Zunächst danke ich Herrn Prof. Dr. H. Klaeren, der das Entstehen dieses Buches ermöglicht hat. Prof. Dr. K. Indermark hat mir durch seine Vorlesung „Grundlagen der funktionalen Programmierung“ erste Eindrücke der funktionalen Programmierung und ihrer Implementierung gegeben. In der Anfangsphase gab mir Prof. Dr. V. Claus wertvolle Hinweise zur Strukturierung dieses Buches. Weiterhin bedanke ich mich bei Tobias Hüttner, Adrian Krug, Elisabeth Meinhard, Martin Plümicke, Bernd Raichle, Peter v. Savigny, Peter Scheffczyk, Christoph Schmitz, Matthias Seidel, Michael Sperber, Arthur Steiner, Michael Walter, Christian Wolf. Sie alle haben in verschiedener Hinsicht zum Entstehen dieses Buches beigetragen. Vor allem danke ich meiner Frau Uta, die während meiner Arbeit an diesem Buch viel Geduld aufbringen mußte und durch kritische Anmerkungen zur Präsentation dieses Buches beigetragen hat.

Tübingen, im Juni 1994

Peter Thiemann

Alle Syntaxdiagramme in diesem Buch wurden automatisch mithilfe eines vom Autor erstellten Haskell-Programmes erzeugt.

Inhaltsverzeichnis

1	Einführung	11
2	Grundlegende Sprachstrukturen	15
2.1	Programmierung mit Funktionen	15
2.2	Lexikalische Syntax	18
2.3	Deklarationen	20
2.4	Typausdrücke	23
2.5	Ausdrücke	24
2.6	Muster	30
2.7	Deklarationen auf der Skriptebene	37
2.8	Polymorphie	45
2.9	Aufgaben	58
3	Funktionen höheren Typs	61
3.1	Die Funktion <code>map</code>	62
3.2	Die Funktion <code>foldr</code>	65
3.3	Funktionale auf Bäumen	66
3.4	Verallgemeinerte <code>map</code> - und <code>fold</code> -Funktionale	67
3.5	Literaturhinweise	69
3.6	Aufgaben	69
4	Fallstudien	71
4.1	Auswertung von Polynomen	71
4.2	Operationen auf Matrizen und Vektoren	72
4.3	Graphische Darstellung von Bäumen	74
5	Verzögerte Auswertung	79
5.1	Auswertung	79
5.2	Newtonscher Algorithmus	80
5.3	Das Sieb des Eratosthenes	82
5.4	Zirkuläre Datenstrukturen	85
5.5	Aufgaben	91

8 Inhaltsverzeichnis

6	Programmeigenschaften	93
6.1	Induktionsbeweise	93
6.2	Aussagen über Funktionen	101
6.3	Programmsynthese	107
6.4	Programmtransformation	109
6.5	Partielle Listen	112
6.6	Literaturhinweise	113
6.7	Aufgaben	113
7	Fortgeschrittene Konzepte	115
7.1	Komprehensionen für Listen	115
7.2	Parser	116
7.3	Monaden	121
7.4	Funktionale Ein-/Ausgabe	129
7.5	Spezifische Eigenschaften von Haskell und Gofer	137
7.6	Literaturhinweise	147
7.7	Aufgaben	147
8	Überblick und Anwendungen	151
8.1	Lisp	151
8.2	ISWIM und FP	154
8.3	ML	154
8.4	Hope	156
8.5	Miranda TM	156
8.6	Haskell	158
8.7	Anwendungen	159
9	Einführung in die denotationelle Semantik	161
9.1	Semantik von Funktionsgleichungen	161
9.2	Strikt vs. nicht-strikt	164
10	Bereichstheorie	167
10.1	Vollständige Halbordnungen	167
10.2	Konstruktion von Halbordnungen	171
10.3	Beziehungen zwischen Bereichen	175
10.4	Bereichsgleichungen	176
10.5	Literaturhinweise	178
10.6	Aufgaben	178

11	Universelle Algebra	181
11.1	Homogene Algebra	181
11.2	Polymorphe Algebra	187
11.3	Literaturhinweise	196
11.4	Aufgaben	197
12	Sprachen mit Funktionen erster Ordnung	199
12.1	Syntax	199
12.2	Semantik	200
12.3	Maschinenmodelle und Übersetzung	205
12.4	Aufgaben	215
13	Sprachen mit Funktionen höherer Ordnung	217
13.1	Syntax	217
13.2	Semantik	218
13.3	Maschinenmodelle und Übersetzung	221
13.4	Parallele Auswertung	233
13.5	Literaturhinweise	238
13.6	Aufgaben	238
14	Abstrakte Interpretation	241
14.1	Grundlagen	241
14.2	Striktheitsanalyse	247
14.3	Vorwärtsanalyse	248
14.4	Rückwärtsanalyse	250
14.5	Literaturhinweise	257
14.6	Aufgaben	258
15	Der λ-Kalkül	259
15.1	Syntax und Reduktionssemantik	259
15.2	Darstellung rekursiver Funktionen	263
15.3	Ein angereicherter λ -Kalkül	267
15.4	Typen für den λ -Kalkül	272
15.5	Semantik von Typen	277
15.6	Die SECD-Maschine	287
15.7	SKI-Kombinatorreduktion	290
15.8	Literaturhinweise	299
15.9	Aufgaben	300

A	Grundlegende Notation	303
B	Syntaxdiagramme von Gofer	305
B.1	Deklarationen	305
B.2	Typen	306
B.3	Klassen- und Exemplardeklarationen	307
B.4	Wert- und Funktionsdeklarationen	308
B.5	Ausdrücke	309
B.6	Muster	311
B.7	Variablen und Operatoren	312
B.8	Lexikalische Syntax	312
C	Kurzübersicht Gofer	315
D	Implementierungen von Gofer und Haskell	317
E	Bedienung des Gofer-Interpreterers	319
	Literatur	321
	Sachwortverzeichnis	337