

---

# Inhaltsverzeichnis

---

## Teil I Elementare Funktionale Programmierung Eine Wiederholung

---

<b>0</b>	<b>Das Strittigste vorab: Notationen</b>	<b>3</b>
0.1	Jenseits von ASCII	4
0.2	Jenseits von Infix: Mixfix	5
0.3	Overloading extrem	6
0.4	Layout mit Bedeutung	7
0.5	Bindung von Variablen	8
<b>1</b>	<b>Grundlagen der Funktionalen Programmierung</b>	<b>11</b>
1.1	Funktionen	11
1.1.1	Funktionsdefinition	12
1.1.2	Ausdrücke	13
1.1.3	$\lambda$ -Notation	13
1.1.4	Funktionalitäten: Die Typisierung von Funktionen	15
1.1.5	Partielle Applikation und Currying	16
1.1.6	Typen	17
1.1.7	Musterbasierte Funktionsdefinition	18
1.1.8	Erweitertes Patternmatching	19
1.1.9	Polymorphie	20
1.1.10	Eigenschaften (Propertys)	21
1.1.11	Sequenzen (Listen, Folgen)	22
1.2	Funktionale	23
1.2.1	Allgemeine Funktionale	24
1.2.2	Catamorphismen (Map-Filter-Reduce)	25
1.3	Semantik und Auswertungsstrategien	31
1.3.1	Denotationelle Semantik	31
1.3.2	Operationale Semantik	32
1.4	Mit Programmen rechnen	35
1.4.1	Von linearer Rekursion zu Tail-Rekursion	35

1.4.2	Ein „universeller Trick“: Continuations . . . . .	38
1.4.3	Vereinfachung komplexerer Rekursionen . . . . .	41
1.5	OPAL, ML und HASKELL . . . . .	44
<b>2</b>	<b>Faulheit währt unendlich</b> . . . . .	<b>47</b>
2.1	Unendliche Objekte: die Idee . . . . .	47
2.2	LAZY, QUOTE und UNQUOTE . . . . .	49
2.2.1	LAZY als generischer Typ . . . . .	50
2.2.2	Simulation in strikten Sprachen wie OPAL oder ML . . . . .	50
2.3	LAZY Listen . . . . .	51
2.4	Programmieren mit LAZY Listen . . . . .	53
2.4.1	Unbeschränkte Folgen . . . . .	53
2.4.2	Approximationsaufgaben . . . . .	55
2.4.3	Animation („Ströme“) . . . . .	56
<b>3</b>	<b>Parser als Funktionen höherer Ordnung</b> . . . . .	<b>59</b>
3.1	Vorbemerkung zu Grammatiken und Syntaxbäumen . . . . .	61
3.2	Parser . . . . .	62
3.3	Scanner . . . . .	65
3.4	Verallgemeinerungen . . . . .	67

---

## Teil II Strukturierung von Programmen

---

<b>4</b>	<b>Gruppen: Die Basis der Modularisierung</b> . . . . .	<b>73</b>
4.1	Items . . . . .	74
4.2	Das allgemeine Konzept der Gruppen . . . . .	74
4.2.1	<i>Syntactic sugar</i> : Schlüsselwörter . . . . .	76
4.2.2	Selektoren und die Semantik von Gruppen . . . . .	77
4.3	Environments und Namensräume . . . . .	79
4.3.1	Environments . . . . .	81
4.3.2	Scopes und lokale Namensräume . . . . .	81
4.3.3	Namenserkenkung im Scope . . . . .	82
4.3.4	Namenserkenkung außerhalb des Scopes (USE) . . . . .	83
4.4	Overloading . . . . .	85
4.5	Beispiele für Strukturen und Packages . . . . .	86
4.6	Weitere syntaktische Spielereien . . . . .	88
4.6.1	Verteilte Definition von Items . . . . .	88
4.6.2	Tupel als spezielle Gruppen . . . . .	90
4.7	Programme und das Betriebssystem . . . . .	90
4.7.1	Was ist eigentlich ein Programm? . . . . .	91
4.7.2	... und was ist mit den Programmdateien? . . . . .	92

<b>5 Operatoren auf Gruppen (Morphismen)</b> .....	95
5.1 Vererbung (EXTEND) .....	95
5.2 Signatur-Morphismen .....	96
5.2.1 Restriktion (ONLY, WITHOUT) .....	97
5.2.2 Renaming (RENAMING) .....	97
5.2.3 Kombination und Verwendung von Morphismen .....	98
5.2.4 Vererbung mit Modifikation .....	99
5.3 Geheimniskrämerei: Import und Export .....	101
5.3.1 Schutzwall nach außen – Export (PRIVATE, PUBLIC) .....	102
5.3.2 Schutzwall nach innen – Import .....	103
5.4 Generizität: Funktionen, die Gruppen liefern .....	105

---

### Teil III Die Idee der Typisierung

---

<b>6 Typen</b> .....	109
6.1 Generelle Aspekte von Typen .....	109
6.1.1 Die Pragmatik: Statische oder dynamische Typprüfung? .....	109
6.1.2 <i>Reflection</i> : Typen als „ <i>First-class citizens</i> “ .....	111
6.1.3 Intensionalität, Reflection und der Compiler .....	113
6.1.4 Typen sind auch nur Terme .....	114
6.1.5 Typdeklarationen: Typsynonyme oder neue Typen? .....	114
6.1.6 <i>Kinding</i> : Typen höherer Stufe .....	115
6.1.7 Mehrfachtypisierung (Mehrfachvererbung) .....	116
6.2 Elementare Typen .....	116
6.3 Aufzählungstypen .....	117
6.4 Tupel- und Gruppentyp .....	119
6.4.1 Tupeltyp .....	119
6.4.2 Gruppentyp .....	121
6.5 Summentypen .....	123
6.5.1 Was für ein Typ bist du? Dieser Typ! .....	124
6.5.2 Disjunkt oder nicht? .....	126
6.5.3 Summen mit Typausdrücken .....	127
6.5.4 <i>Syntactic sugar</i> : Overloading von „ <code>:</code> “ .....	128
6.6 Funktionstypen .....	128
6.7 Rekursive Typen .....	129
6.8 Wie geht's weiter? .....	130
<b>7 Subtypen (Vererbung)</b> .....	131
7.1 Ein genereller Rahmen für Subtypen .....	131
7.1.1 Die Subtyp-Relation .....	132
7.1.2 Typanpassung ( <i>Casting</i> ) .....	133
7.1.3 <i>Coercion Semantics</i> .....	134
7.2 Direkte Subtypen: Constraints .....	135
7.3 Gruppen/Tupel und Subtypen .....	137

7.3.1	Subtypen von Tupeltypen .....	139
7.3.2	Produkttypen mit Constraints .....	139
7.4	Summentypen und Subtypen .....	140
7.4.1	Varianten und „echte“ Subtypen .....	141
7.4.2	Summen + Tupel sind ein Schutzwall .....	142
7.5	Funktionstypen und Subtypen .....	143
<b>8</b>	<b>Polymorphe und abhängige Typen</b> .....	<b>145</b>
8.1	Typfunktionen (generische Typen, Polymorphie) .....	146
8.1.1	Typvariablen .....	146
8.1.2	Typfunktionen .....	148
8.2	Abhängige Typen .....	150
8.3	Notationen für Typterme mit Variablen .....	154
8.3.1	Bindung von Variablen .....	154
8.3.2	<i>Syntactic sugar</i> : Nachgestellte Variablen .....	155
8.3.3	<i>Syntactic sugar</i> : Optionale Parameter .....	155
<b>9</b>	<b>Spezifikationen und Typklassen</b> .....	<b>157</b>
9.1	Operatoren auf Typklassen .....	161
9.2	Signaturen: Die Typisierung von Strukturen .....	162
9.2.1	<i>Syntactic sugar</i> : Traditionelle Signatur-Notation .....	163
9.2.2	<i>Syntactic sugar</i> : Verschmelzen von Struktur und Signatur .....	164
9.3	Spezifikationen: Subtypen von Signaturen .....	165
9.4	Signaturen und Spezifikationen sind existenziell .....	167
9.4.1	Zahlen generell betrachtet .....	167
9.4.2	Existenzielle Typen .....	168
9.5	Von Spezifikationen zu Typklassen .....	169
9.5.1	Typklassen à la HASKELL .....	169
9.5.2	Definition von Typklassen .....	170
9.6	Beispiele für Spezifikationen und ihre Typklassen .....	173
9.6.1	Gleichheit .....	173
9.6.2	Ordnung .....	174
9.6.3	Halbgruppen, Monoide <i>and all that</i> .....	175
9.6.4	Druckbares, Speicherbares .....	176
9.7	Subklassen .....	177
9.8	Views und Mehrfachtypisierung .....	178
9.8.1	Mehrfachtypisierung bei Typklassen .....	178
9.8.2	Views (Mehrfache Sichten) .....	179
9.9	Beispiel: Physikalische Dimensionen .....	184
<b>10</b>	<b>Beispiel: Berechnung von Fixpunkten</b> .....	<b>187</b>
10.1	Beispiel: Erreichbarkeit in einem Graphen .....	187
10.2	Ein bisschen Mathematik: CPOs und Fixpunkte .....	189
10.2.1	Vollständige partielle Ordnungen – CPOs .....	189

10.2.2	Standardkonstruktionen für CPOs	190
10.2.3	CPO-Konstruktion durch Idealvervollständigung	191
10.2.4	Fixpunkte	192
10.3	Die Programmierung von Fixpunkt-Algorithmen	197
10.3.1	CPOs als Typklasse	197
10.3.2	Fixpunktberechnung: Der Basisalgorithmus	198
10.3.3	Optimierung durch feinere Granularität	199
10.3.4	Verallgemeinerungen und Variationen	201
10.3.5	Fixpunkte als „ <i>Design Pattern</i> “	202
10.4	Datentypen als CPOs	203
10.5	Beispiel: Lösung von Gleichungssystemen	209
10.5.1	Repräsentation von Gleichungssystemen	210
10.5.2	Optimierung	212
10.5.3	Nochmals: Grammatiken und Parser	212
10.5.4	Ein Gedankenexperiment: Anpassbare Rekursion	215
<b>11</b>	<b>Beispiel: Monaden</b>	217
11.1	Kategorien, Funktoren und Monaden	218
11.1.1	Typklassen als Kategorien	218
11.1.2	Polymorphe Typen als Funktoren	219
11.1.3	Monaden	220
11.2	Beispiele für Monaden	222
11.2.1	Sequenzen als Monaden	222
11.2.2	<i>Maybe</i> als Monade	223
11.2.3	Automaten als Monaden („Zustands-Monaden“)	224
11.2.4	Spezielle Zustands-Monaden	227
11.2.5	Zähler als Zustands-Monaden	230
11.2.6	Generatoren als Zustands-Monaden	231
11.2.7	Ein-/Ausgabe als Zustands-Monade	231
11.3	Spezielle Notationen für Monaden	231
11.3.1	Die Operatoren „ $\rightarrow$ “ und „ $;$ “	232
11.3.2	Erweitertes LET	232
11.3.3	Monaden-Casting	233

---

## Teil IV Datenstrukturen

---

<b>12</b>	<b>Netter Stack und böse Queue</b>	239
12.1	Wenn Listen nur anders heißen: Stack	241
12.2	Wenn Listen zum Problem werden: Queue	242
12.2.1	Variante 1: Queue = Paar von Listen	243
12.2.2	Variante 2: Faulheit macht kalkulierbar	245
12.3	Deque und Sequence	249
12.3.1	Double-ended Queues ( <i>Deque</i> )	249
12.3.2	Sequenzen ( <i>Catenable Lists</i> )	250

12.4	Arbeiten mit listenartigen Strukturen . . . . .	252
<b>13</b>	<b>Compilertechniken für funktionale Datenstrukturen . . . . .</b>	<b>255</b>
13.1	Die Bedeutung von <i>Single-Threadedness</i> . . . . .	256
13.1.1	Die Analyse von <i>Single-Threadedness</i> . . . . .	257
13.1.2	Monaden garantieren <i>Single-Threadedness</i> . . . . .	259
13.1.3	Lineare Typen garantieren <i>Single-Threadedness</i> . . . . .	261
13.2	<i>A Dag For All Heaps (Reference-Counting)</i> . . . . .	263
13.2.1	Ein Dag-Modell für das Datenmanagement . . . . .	263
13.2.2	Sicheres Management persistenter Strukturen . . . . .	268
13.2.3	Dynamische Erkennung von <i>Single-Threadedness</i> . . . . .	273
13.2.4	„ <i>Tail-Rekursion modulo cons</i> “ . . . . .	275
13.2.5	<i>Reference-Counting</i> und Queues: Hilft Laziness? . . . . .	280
13.3	<i>Version Arrays</i> . . . . .	283
<b>14</b>	<b>Funktionale Arrays und Numerische Mathematik . . . . .</b>	<b>287</b>
14.1	Semantik von Arrays: Funktionen . . . . .	288
14.1.1	Die Typklasse der Intervalle . . . . .	288
14.1.2	Eindimensionale Arrays (Vektoren) . . . . .	290
14.1.3	Mehrdimensionale Arrays (Matrizen) . . . . .	293
14.1.4	Matrizen von besonderer Gestalt ( <i>Shapes</i> ) . . . . .	294
14.1.5	Map-Reduce auf Arrays . . . . .	296
14.2	Pragmatik von Arrays: „Eingefrorene“ Funktionen . . . . .	297
14.2.1	Memoization . . . . .	298
14.2.2	Speicherblöcke . . . . .	300
14.2.3	Sicherheit und <i>Single-Threadedness</i> : <i>Version Arrays</i> . . . . .	301
14.2.4	Von Arrays zu Speicherblöcken . . . . .	301
14.2.5	Implementierung eindimensionaler Arrays . . . . .	302
14.2.6	Implementierung mehrdimensionaler Arrays . . . . .	303
14.3	Arrays in der Numerik: Vektoren und Matrizen . . . . .	304
14.3.1	Vektoren . . . . .	304
14.3.2	Matrizen . . . . .	306
14.4	Beispiel: Gauß-Elimination . . . . .	306
14.4.1	Lösung von Dreieckssystemen . . . . .	307
14.4.2	<i>LU</i> -Zerlegung ( <i>Doolittle</i> -Variante) . . . . .	309
14.4.3	Spezialfall: Gauß-Elimination bei <i>Tridiagonalmatrizen</i> . . . . .	311
14.5	Beispiel: Interpolation . . . . .	311
14.6	Beispiel: Spline-Interpolation . . . . .	314
14.7	Beispiel: Schnelle Fourier-Transformation ( <i>FFT</i> ) . . . . .	318
<b>15</b>	<b>Map: Wenn Funktionen zu Daten werden . . . . .</b>	<b>323</b>
15.1	Variationen über Funktionen . . . . .	324
15.2	Die Typklasse der Funktionen . . . . .	325
15.3	Die Typklasse der Mappings . . . . .	326
15.3.1	Implementierungen von Maps . . . . .	329

15.3.2	Map-Filter-Reduce auf Maps	329
15.3.3	Prädikate höherer Ordnung auf Maps	331
15.4	Maps und Funktionen: Zwei Seiten einer Medaille	333
15.5	Maps, Funktionen und Memoization	334
<b>16</b>	<b>Beispiel: Synthese von Programmen</b>	<b>337</b>
16.1	Globale Suche	338
16.1.1	Suchräume	338
16.1.2	Einschränkung von Suchräumen	339
16.1.3	Suchräume und partielle Lösungen	340
16.1.4	Basisregeln für Suchräume	340
16.2	Problemlösungen als <i>Maps</i>	341
16.2.1	Beispiel: Das $n$ -Damen-Problem	341
16.2.2	Suchräume als Mengen von Maps	343
16.2.3	Constraints auf Mengen von Maps	344
16.3	Programmableitung	345
16.3.1	Das $n$ -Damen-Problem – Von der Spezifikation zum Algorithmus	346
16.3.2	Ein allgemeines Schema für die globale Suche	350
16.4	Beispiel: Scheduling	353

---

## Teil V Integration von Paradigmen

---

<b>17</b>	<b>Zeit und Zustand in der funktionalen Welt</b>	<b>359</b>
17.1	Zeit und Zustand: Zwei Seiten einer Medaille	360
17.1.1	Ein kleines Beispiel	362
17.2	<i>Monaden</i> : Ein schicker Name für Altbekanntes	364
17.2.1	Programmieren mit <i>Continuations</i>	366
17.2.2	<i>Continuations</i> + <i>Hiding</i> = Monaden	368
17.2.3	Die Ein-/Ausgabe ist eine Compiler-interne Monade	369
17.3	Zeit: Die elementarste aller Zustands-Monaden	369
17.3.1	Zeitabhängige Operationen und Evolution	371
17.3.2	Zeit-Monade oder Zustands-Monade?	374
17.4	Die erweiterte Zeit-Monade	375
17.4.1	Exceptions	375
17.4.2	Choice	377
17.4.3	Die Systemuhr und Timeouts	378
17.4.4	Zusammenfassung: Die Zeit-Monade	378
<b>18</b>	<b>Objekte und Ein-/Ausgabe</b>	<b>381</b>
18.1	Objekte als zeitabhängige Werte	381
18.1.1	Spezielle Notationen für Objekte und Klassen	385
18.1.2	„Globale“ Objekte	387
18.2	Laufzeitsystem und andere Objekte (Zeit-Monaden)	389

18.2.1	Dateioperationen höherer Ordnung	392
18.2.2	Ein typisiertes Dateisystem?	393
<b>19</b>	<b>Agenten und Prozesse</b>	395
19.1	Service-orientierte Architekturen	396
19.2	Agenten als Monaden	398
19.3	Kommunikation: Service-Access-Points	400
19.4	Ein Beispiel	405
19.5	„Globale“ Agenten und SAPs	411
19.6	Spezialfälle: Kanäle und <i>Gates</i>	412
19.6.1	Kanäle	412
19.6.2	<i>Gates</i> : SAPs + Agenten	414
19.7	OPAL, CONCURRENT HASKELL, EDEN und ERLANG	418
<b>20</b>	<b>Graphische Schnittstellen (GUIs)</b>	421
20.1	GUIs – ein Konzept mit drei Dimensionen	422
20.2	Die Applikation ( <i>Model</i> )	423
20.3	Graphische Gestaltung ( <i>View</i> )	425
20.3.1	Arten von GUI-Elementen	427
20.3.2	Stil-Information	430
20.3.3	Geometrische Anordnung	430
20.3.4	<i>The Big Picture</i>	433
20.4	Interaktion mit der Applikation ( <i>Control</i> )	434
20.4.1	Das Fenster als Agent	434
20.4.2	Emitter als Kanäle	435
20.4.3	Regulator-Gates	437
20.4.4	Weitere Gates	440
20.4.5	Ereignisse – Events	440
20.5	HAGGIS, FUDGETS, FRANTK und andere	441
<b>21</b>	<b>Massiv parallele Programme</b>	443
21.1	<i>Skeletons</i> : Parallelität durch spezielle Funktionale	444
21.2	<i>Cover</i> : Aufteilung des Datenraums	446
21.2.1	Spezifikation von <i>Covern</i>	447
21.2.2	Skelette über <i>Covern</i>	449
21.2.3	Matrix-Cover	451
21.3	Beispiel: Matrixmultiplikation	452
21.4	Von <i>Skeletons</i> zum <i>Message passing</i>	456
<b>22</b>	<b>Integration von Konzepten anderer Programmierparadigmen</b>	459
22.1	Programmierparadigmen und deren Integration	459
22.2	Objektorientierte Erweiterungen funktionaler Sprachen	461
22.2.1	HASKELL++	462
22.2.2	O'HASKELL	462

22.2.3 OCAML .....	464
22.3 Funktional-logische Programmierung und darüber hinaus .....	464
22.4 Fazit .....	467
<b>Literatur</b> .....	469
<b>Index</b> .....	479

*Hinweis:* Eine Errata-Liste und weitere Hinweise zu diesem Buch sind über die Web-Adresse <http://www.uebb.cs.tu-berlin.de/books/fp> zu erreichen.