

Inhaltsverzeichnis

Einleitung	1
Kaffee oder Tee?	1
Die t.Sprachen	2
Die Java-Implementierung	5
Ockhams Rasiermesser	6
Leserkreis	7
Hinweise zur Benutzung	8
Webseite zum Buch	8
Das Umschlagbild	9
Danksagung	9
1 t.Zero: Deklarative Programmierung	11
1.1 Sprachelemente	12
1.1.1 Zahlen	12
1.1.2 Funktionen	13
1.1.3 Bedingte Ausdrücke	15
1.1.4 Rekursion	15
1.1.5 Fehler	16
1.2 Beispiele	17
1.2.1 Zahlenspielereien	18
1.2.2 Größter gemeinsamer Teiler	19

1.2.3	Quadratwurzel	20
1.2.4	Primzahlen	22
1.2.5	Potenzen	23
1.2.6	Berechnung von π	25
1.2.7	Exponentialfunktion	27
1.2.8	Logarithmus	30
1.3	Deklarativer Programmierstil	32
1.3.1	Sprachumfang und Programmierstil	32
1.3.2	Probleme des deklarativen Programmierstils	34
1.4	Syntax und Semantik	35
1.4.1	Lexikalische Struktur	35
1.4.2	Syntax	36
1.4.3	Semantik	38
1.5	Der Interpreter	39
1.5.1	Die Read-Eval-Print-Schleife	39
1.5.2	Ausdrücke und ihre Auswertung	40
2	t.Lisp: Listenbasierte Programmierung	45
2.1	Lisp	45
2.2	Sprachelemente	46
2.2.1	Listen	46
2.2.2	Symbole, Quotierung und Binden	49
2.2.3	Typen	51
2.2.4	Funktionen	53
2.2.5	Bedingte Ausdrücke	54
2.3	Beispiele	55
2.3.1	Elementare Listenfunktionen	55
2.3.2	Vereinigen und Zerlegen von Listen	58
2.3.3	Umkehren einer Liste	60
2.3.4	Sortieren	62
2.3.5	Quotierung	65
2.3.6	Arithmetische Ausdrücke	68
2.3.7	Symbolische Differentiation	70
2.4	Datenabstraktion	76
2.4.1	Natürliche Zahlen	77
2.4.2	Mengen	83
2.4.3	Binärbäume	88
2.5	m.Lisp – ein metazirkuläres Lisp	96
2.5.1	Der m.Lisp-Interpreter im Überblick	97
2.5.2	Auswertung: eval und apply	101

2.5.3	Metazirkuläre Anwendung von m.Lisp	106
2.5.4	Statischer und dynamischer Scope	107
2.6	Die Implementierung von t.Lisp	110
2.6.1	Initialisierungsdateien	110
2.6.2	Entwurf der Implementierung	112
2.6.3	Die abstrakte Klasse Procedure	115
2.6.4	Operatoren	117
2.6.5	Laufzeitprüfungen	122
2.6.6	Listen	123
3	t.Pascal: Imperative Programmierung	129
3.1	Imperativer Programmierstil	130
3.1.1	Rechnen mit Zuständen	130
3.1.2	Pascal	131
3.2	Sprachelemente	132
3.2.1	Arrays und Records	132
3.2.2	Variablen	134
3.2.3	Schleifen	135
3.2.4	Blöcke	136
3.2.5	Sprungbefehle	138
3.2.6	Sonstiges	140
3.3	Beispiele	142
3.3.1	Ducci-Folgen	142
3.3.2	Vektoren und Matrizen	143
3.3.3	Sortieren von Arrays	149
3.3.4	Neue Typen: Komplexe Zahlen	152
3.3.5	Rekursive Typen: Listen	156
3.3.6	Manipulationen an Listen	158
3.3.7	Stapel und Schlangen	161
3.4	Die Harvard-Maschine	169
3.4.1	Befehlssatz	170
3.4.2	Implementierung	171
3.4.3	Beispielprogramm	175
3.5	Die Implementierung von t.Pascal	176
3.5.1	Die Klasse Record und der Operator record	176
3.5.2	Die Operatoren block und var	179
3.5.3	Weitere Operatoren und Initialisierung	180
4	t.Scheme: Funktionale Programmierung	183
4.1	Funktionen als Daten erster Klasse	183

4.2	Sprachelemente	184
4.2.1	Funktionen	184
4.2.2	Umgebungen	187
4.2.3	Makros	190
4.2.4	Exceptions	192
4.2.5	Sonstiges	194
4.3	Beispiele	199
4.3.1	Die Collatz-Funktion	199
4.3.2	Anwenden von Funktionen auf Listen und Arrays	200
4.3.3	Erweitern von Funktionen	202
4.3.4	Komposition und Iteration von Funktionen	206
4.3.5	Der Ableitungsoperator	207
4.3.6	Ein Minitutorial zu Makros	210
4.4	Der Kontext einer Funktion	219
4.4.1	Kontexterzeugung mit <code>let</code>	219
4.4.2	Anonyme rekursive Funktionen	220
4.5	Funktionen als Objekte	222
4.5.1	Zufallszahlen	223
4.5.2	Stack-Objekte	223
4.5.3	Iteratoren und Mengen	226
4.6	Endrekursive Funktionen	229
4.6.1	Endrekursion	229
4.6.2	Der Auswertungsstack	231
4.6.3	Endständigkeit	234
4.6.4	Die Konstruktion endrekursiver Funktionen	239
4.7	Die Implementierung von <code>t.Scheme</code>	245
4.7.1	Umgebungen und zugehörige Operatoren	245
4.7.2	Die Klassen <code>Function</code> und <code>Macro</code>	251
4.7.3	Tracing-Modus und Protokoll-Modus	254
4.7.4	Der Exception-Mechanismus	257
4.7.5	Die Verarbeitung von Tail Calls	259
4.7.6	Der Operator <code>part</code>	265
5	<code>t.Lambda</code>: Rein funktionale Programmierung	267
5.1	Der Lambda-Kalkül	267
5.1.1	Lambda-Ausdrücke	268
5.1.2	Reduktion	268
5.2	<code>Function</code> als universeller Typ	270
5.2.1	Spielregeln	270
5.2.2	Boolesche Werte	271

5.2.3	Church-Listen	272
5.2.4	Church-Zahlen	273
5.3	Rekursion in t.Lambda	275
5.3.1	Primitive Rekursion	275
5.3.2	Bedingte Ausdrücke und Rekursion	276
5.3.3	Elimination von Rekursion	277
5.4	m.Lambda – ein makrobasierter Lambda-Kalkül	282
5.4.1	Funktionale Programmierung ohne Funktionen	282
5.4.2	Der Fixpunkt-Kombinator	283
6	t.Java: Objektorientierte Programmierung	285
6.1	Objektorientierung	285
6.1.1	Grundkonzepte	285
6.1.2	t.Java – erste Schritte	286
6.2	Sprachelemente	288
6.2.1	Klassen	288
6.2.2	Objekte	290
6.2.3	Methoden	292
6.3	Beispiele	298
6.3.1	Dynamische Arrays	298
6.3.2	Turingmaschinen	301
6.3.3	Private Attribute und Methoden	309
6.3.4	Innere Klassen	313
6.4	Ströme	317
6.4.1	Der Typ Stream	317
6.4.2	Beispiele	320
6.4.3	Die Thue-Morse-Folge	325
6.4.4	Laziness und die t.Java-Klasse Stream	328
6.4.5	Die Java-Klasse Stream	332
6.5	Die Implementierung von t.Java	334
6.5.1	Entwurfsüberlegungen	334
6.5.2	Die Klasse Obj	336
6.5.3	Die Klasse ClassObj	342
6.5.4	Die Klasse Method	348
7	t.Prolog: Logische Programmierung	351
7.1	Einführung	351
7.1.1	Syllogismen	352
7.1.2	Familienbeziehungen	353
7.1.3	Färbung von Graphen	357

7.2	Grundbegriffe der logischen Programmierung	359
7.2.1	Implizite Quantoren	359
7.2.2	Resolution	359
7.2.3	Substitution	362
7.2.4	Unifikation	363
7.3	Sprachelemente	366
7.3.1	Unbestimmte und Terme	366
7.3.2	Die Operatoren <code>unify</code> und <code>substitute</code>	367
7.3.3	Regelbasen	368
7.3.4	Regeln	370
7.3.5	Anfragen	372
7.3.6	Escape nach <code>t.Scheme</code>	373
7.3.7	Der Cut-Mechanismus	377
7.3.8	Wildcards	378
7.3.9	Schleifenerkennung	379
7.4	Beispiele	380
7.4.1	Unifikation	380
7.4.2	Relationen	384
7.4.3	Suche in Graphen	388
7.4.4	Einstein-Rätsel	392
7.4.5	Datenstrukturen	397
7.4.6	Das Acht-Damen-Problem	403
7.4.7	Arithmetik in <code>t.Prolog</code>	405
7.4.8	Cut und Negation	408
7.5	Die Implementierung von <code>t.Prolog</code>	411
7.5.1	Der Entwurf im Überblick	411
7.5.2	Unifikation und die Klasse <code>Indeterminate</code>	414
7.5.3	Die Klasse <code>Substitution</code>	415
7.5.4	Resolution und die Klasse <code>RuleBase</code>	416
A	Installation	425
B	Quellcode	427
B.1	Das Paket <code>tanagra</code>	428
B.1.1	Die Klassen <code>Token</code> und <code>Lexer</code>	428
B.1.2	Die Klasse <code>Parser</code>	430
B.1.3	Die Klasse <code>Interpreter</code>	432
B.1.4	„Odds and ends“: Die Klasse <code>Sys</code>	433
B.2	Das Paket <code>expressions</code>	435
B.2.1	Die Klassen im Überblick	436

B.2.2 Dateistruktur	437
B.3 Initialisierungsdateien	437
B.4 Beispieldateien	438
Literaturverzeichnis	441
Stichwortverzeichnis	445