

Inhaltsverzeichnis

Vorwort zur ersten Auflage	v
I Allgemeine Grundlagen	1
1 Grundlegende Begriffe	3
1.1 Programmierung	3
1.2 Abstraktion	3
1.3 Referenz	4
1.4 Synthese oder Applikation	4
2 Wichtige Abstraktionsmechanismen	5
2.1 Bildung von Funktionen	5
2.2 Bildung von Modulen und Klassen	6
2.3 Bildung von Paketen in Java	8
3 Operationen in Funktionen	9
3.1 Allgemeine Grundfunktionen	9
3.1.1 Abstraktion	9
3.1.2 Referenz	9
3.1.3 Synthese oder Applikation	9
3.2 Definition von Variablen	9
3.3 Definition von Funktionen	9
3.4 Erzeugung von Objekten	10
3.5 Zuweisung von Werten	10
3.6 Testen von Werten	10
3.7 Wiederholung von Operationen	11
3.8 Ein-/Ausgabeoperationen	13
3.9 Arithmetische Operationen	13
3.10 Relationale Operationen	13
3.11 Boolesche Operationen	13
3.12 Bibliotheksfunktionen	13
4 Grundlegende Datentypen	15
4.1 Atom	15
4.2 Listen	15
4.3 Verarbeitung von Listen	17
4.4 Dictionaries	18
5 Programmiersprachen	21
5.1 Allgemeine Programmiersprachen	21
5.2 Spezielle Programmiersprachen	25

II Die Programmiersprache A++

27

1 Einführung	29
1.1 A++ – Die kleinste Programmiersprache der Welt	29
1.1.1 Wesen und Zweck von A++	29
1.1.2 Weitere Eigenschaften von A++	30
1.2 Ursprung von A++	30
1.2.1 Verallgemeinerung des Lambda-Kalküls	30
1.3 Konstitutive Prinzipien in A++	31
1.3.1 Abstraktion	31
1.3.2 Referenz	31
1.3.3 Synthese	31
1.3.4 Closure	32
1.3.5 Lexical Scope	34
2 Sprachdefinition	35
2.1 Syntax und Semantik von A++	35
2.2 Beispiele zur Syntax von A++	36
2.2.1 Beispiele zur Abstraktion 1. Alternative in 2.2	36
2.2.2 Beispiele zur Abstraktion 2. Alternative in 2.2	36
2.2.3 Beispiele zur Referenz 2.3	36
2.2.4 Beispiele zur Synthese 2.4	36
2.3 A++ Erweiterung	38
2.3.1 Syntax von A++ mit vorgegebenen Primitiv-Abstraktionen	39
2.3.2 Beispiele zu den Erweiterungen in A++	39
3 Erste Entfaltung von A++	43
3.1 Programmierstil in A++	43
3.2 Grundlegende Logische Abstraktionen	43
3.2.1 Abstraktionen 'true' und 'false'	43
3.2.2 Abstraktionen 'lif'	44
3.3 Erweiterte Logische Abstraktionen	44
3.4 Numerische Abstraktionen	45
3.4.1 Abstraktion für die Zahl '0'	45
3.4.2 Abstraktion für die Zahl '1'	45
3.4.3 Abstraktion für die Zahl '2'	45
3.4.4 Abstraktion für das Prädikat 'zerop'	46
3.4.5 Abstraktion für die Zahl '3'	46
3.4.6 Utility-Abstraktion 'compose'	46
3.4.7 Abstraktion für die Addition	46
3.4.8 Abstraktion für die Inkrementierung	47
3.4.9 Abstraktion für die Multiplikation	47
3.5 Abstraktionen für allgemeine Listen	47
3.5.1 Abstraktion für den Konstruktor	47
3.5.2 Abstraktion für den Selektor 'lcar'	48
3.5.3 Abstraktion für den Selektor 'lcdr'	48
3.5.4 Anwendung der grundlegenden Operationen für Listen	48
3.5.5 Abstraktion für das Ende einer Liste	48
3.5.6 Abstraktion für das Prädikat 'nullp'	49
3.5.7 Abstraktion für die Längenabfrage	49
3.5.8 Abstraktion zum Entfernen eines Objektes aus einer Liste	49
3.6 Erweiterte Arithmetische Abstraktionen	50
3.6.1 Abstraktion für 'zeropair'	50
3.6.2 Abstraktion für die Dekrementierung	50
3.6.3 Abstraktion für die Subtraktion	50

3.7	Relationale Abstraktionen	51
3.7.1	Abstraktion für das Prädikat 'gleich'	51
3.7.2	Abstraktion für das Prädikat 'größer als'	51
3.7.3	Abstraktion für das Prädikat 'kleiner als'	51
3.7.4	Abstraktion für das Prädikat 'größer gleich'	52
3.8	Rekursion	52
3.8.1	Abstraktion für die Berechnung der Fakultät	53
3.8.2	Abstraktion für die Summation	53
3.9	Funktionen Höherer Ordnung	54
3.9.1	Abstraktion 'compose'	54
3.9.2	Abstraktion für die 'curry'-Funktion	54
3.9.3	Abstraktion für die Abbildung einer Liste	55
3.9.4	Abstraktion für die 'curry map'-Funktion	55
3.9.5	Abstraktion für die Auswahl aus einer Liste	56
3.9.6	Abstraktion für die Suche nach einem Objekt in einer Liste	56
3.9.7	Abstraktion für den Zugriff auf ein Element einer Liste	57
3.9.8	Abstraktion für die Iteration über die Elemente einer Liste	57
3.10	Mengen-Operationen	57
3.10.1	Abstraktion für das Prädikat 'memberp'	58
3.10.2	Abstraktion für das Hinzufügen eines Elementes	58
3.10.3	Abstraktion für die Vereinigung von Mengen	58
4	Erste Anwendung von A++	61
4.1	Sortierung	61
4.1.1	Abstraktion für das sortierte Einfügen in eine Liste	61
4.1.2	Abstraktion für die Sortierung	61
4.2	Assoziative Listen in A++	62
4.2.1	Abstraktionen für assoziative Listen	62
4.2.2	Anwendung der Abstraktionen für assoziative Listen	63
5	Imperative Programmierung in A++	65
5.1	Ausführung von Anweisungen	65
5.2	Auswertung von Ausdrücken	65
5.3	Programmierstile in A++	66
5.4	Beispiele für imperative Programmierung	66
5.4.1	Beispiele für OOP sind auch Beispiele für imperative Programmie- rung	66
5.4.2	Besonderes Beispiel für imperative Programmierung	66
6	Objekt-Orientierte Anwendung von A++	69
6.1	Einleitung	69
6.1.1	Bildung von Klassen	69
6.1.2	Instanzen von Klassen	69
6.1.3	Beispiele für Objekt-orientierung in A++	70
6.2	Erstes Beispiel zur Objektorientierung in A++	70
6.2.1	Konstruktor für Objekte der Klasse "account"	70
6.2.2	Erzeugung des Objektes "acc1" durch Aufruf von "make-account"	72
6.2.3	Senden der Botschaft "deposit" an das Objekt "acc1"	72
6.2.4	Ausführen der Funktion "deposit"	72
6.2.5	Detaillierter Code in A++	76
6.2.6	Anmerkungen zu dem ersten Beispiel zur Objektorientierung	77
6.2.7	Anmerkungen zum Aufruf der Funktion 'konto'	78
6.3	Zweites Beispiel zur Objektorientierung in A++	78
6.3.1	Anmerkung zu den einzelnen Klassen	78
6.3.2	Beziehungen zwischen den Klassen	81

6.3.3	Zusammenfassung	81
6.4	Drittes Beispiel zur Objektorientierung in A++	85
6.4.1	Anmerkung zu den einzelnen Klassen	85
7	Abstraktion, Referenz und Synthese im Detail	95
7.1	Addition der zwei Zahlen 'two' und 'three'	95
7.1.1	Synthese von 'add' und 'two three' (1)	95
7.1.2	Abstraktion von 'add' (1)	95
7.1.3	Auflösung der Referenz von 'add' in [1]	95
7.1.4	Synthese von $(\lambda(m\ n) \dots)$ und 'two three' in [3]	95
7.1.5	Abstraktion von 'compose'	95
7.1.6	Auflösung der Referenz von 'compose' in [4]	96
7.1.7	Synthese von $(\lambda(f\ g) \dots)$ und '(two f) (three f)' in [6]	96
7.1.8	Abstraktion von three:	96
7.1.9	Auflösung der Referenz von 'three' in [7]	96
7.1.10	Synthese von $(\lambda(f) \dots)$ und 'f' in [9]	96
7.1.11	Synthese von $(\lambda(x) \dots)$ und 'x' in [10]	96
7.1.12	Abstraktion von two:	96
7.1.13	Auflösung der Referenz von 'two' in [11]	96
7.1.14	Synthese von $(\lambda(f) \dots)$ und 'f' in [13]	96
7.1.15	Synthese vom inneren $(\lambda(x) \dots)$ und ' $f(f(x))$ ' in [14]	97
7.2	Multiplikation der zwei Zahlen 'two' und 'three'	97
7.2.1	Synthese von mult und 'two three'	97
7.2.2	Abstraktion von mult:	97
7.2.3	Auflösung der Referenz von 'mult' in [17]	97
7.2.4	Synthese von $(\lambda(m\ n) \dots)$ und 'two three' in [19]	97
7.2.5	Abstraktion von compose:	97
7.2.6	Auflösung der Referenz von 'compose' in [20]	97
7.2.7	Synthese von $(\lambda(f\ g) \dots)$ und 'two three' in [22]	97
7.2.8	Abstraktion von two:	97
7.2.9	Abstraktion von three:	97
7.2.10	Auflösung der Referenz von 'two' und 'three' in [23]	97
7.2.11	Synthese vom inneren $(\lambda(f) \dots)$ und 'x' in [26]	98
7.2.12	Synthese von $(\lambda(f) \dots)$ und ' $(\lambda(x0) \dots)$ ' in [28]	98
7.2.13	Synthese vom inneren $(\lambda(x0) \dots)$ und 'x1'	98
7.2.14	Synthese von $(\lambda(x0) \dots)$ und ' $(x(x\ x1))$ '	99
7.2.15	Umbenennung der Variablen: $x \rightarrow f$ und $x1 \rightarrow x$	99
8	Infrastruktur für A++	101
8.1	Support-Funktionen	101
8.1.1	Abstraktion für die Ausgabe einer Zahl	101
8.1.2	Abstraktion für die Ausgabe eines booleschen Wertes	101
8.1.3	Abstraktion für die Ausgabe von Listen	101
8.2	A++ Interpreter	101
8.2.1	Linux	102
8.2.2	MS-Windows	103
8.2.3	Basis-Abstraktionen	104
8.2.4	Programmbeendigung	104
8.3	Initialisierungsdatei für den ARS-Interpreter	104
III	Von A++ nach ARS++	109
1	ARS++ – Erste Ausbaustufe	111
1.1	Die Programmiersprache ARS++	111

1.1.1	Wesen und Zweck von ARS++	111
1.1.2	ARS++ und Scheme	112
1.1.3	Unterschied zwischen ARS++ und Scheme	112
1.1.4	Ausbau von A++ in ARS++	113
1.2	Struktur von ARS++ in der ersten Ausbaustufe	113
1.2.1	EBNF-Notation	113
1.2.2	Special-Forms in ARS++	116
1.2.3	Makros in ARS++	116
1.2.4	Auswertung der Argumente von Funktionen	117
1.2.5	Anmerkung zur Bedeutung der Klammern	118
1.3	Wichtige Abstraktionsmechanismen	118
1.3.1	Etwas einen Namen geben	119
1.3.2	Bildung von Funktionen	119
1.3.3	Bildung von Klassen	121
1.3.4	Bildung von Makros	128
1.4	Operationen in Funktionen	129
1.4.1	Abstraktion	129
1.4.2	Referenz	129
1.4.3	Synthese	130
1.4.4	Definition von Variablen	130
1.4.5	Erzeugung von Objekten	133
1.4.6	Zuweisung von Werten	133
1.4.7	Testen von Werten	133
1.4.8	Wiederholung von Operationen	136
1.4.9	Ein-/Ausgabeoperationen	138
1.4.10	Arithmetische Operationen	138
1.4.11	Relationale Operationen	138
1.4.12	Boolesche Operationen	138
1.4.13	Bibliotheksfunktionen	140
1.5	Grundlegende Datentypen	141
1.5.1	Atom	141
1.5.2	Listen	141
1.5.3	Verarbeitung von Listen	147
1.5.4	Dictionaries	151
1.5.5	Assoziative Listen	151
1.5.6	Vektoren	154
1.6	Besonderheiten	155
1.6.1	Sequentielle Programmierung	155
1.6.2	“Continuations”	155
1.6.3	“Delayed Evaluation”	158
2	ARS++ – Zweite Ausbaustufe	159
2.1	Überblick	159
2.2	Struktur von ARS++ in der zweiten Ausbaustufe	159
2.3	Erweiterungen	162
2.3.1	Datenbankfunktionen	162
2.3.2	Netzwerkfunktionen	163
2.3.3	Funktionen für reguläre Ausdrücke	165
2.3.4	POSIX-Dateifunktionen	165
2.3.5	POSIX-Konstanten	167

IV Anwendung von ARS++

169

1 Anwendung der Werkzeuge

171

1.1	Anwendung des Compilers ACOMP	171
1.1.1	Anwendung von 'arscomp.scm'	171
1.1.2	Anwendung von 'acomp.avm'	172
1.2	Anwendung des Interpreters AVIM im Normalmodus	173
1.3	Anwendung des Interpreters AVIM im Debugmodus	173
1.4	Anwendung von AVIM mit internem Garbage Collector	178
1.4.1	Aufruf von AVIM 1.0i	178
2	Einfaches Beispiel: Bibliotheksverwaltung	181
2.1	Einleitung und Überblick	181
2.2	Anmerkung zu den einzelnen Klassen	181
2.2.1	Klasse: 'base-object-class'	181
2.2.2	Klasse: 'library'	181
2.2.3	Klasse: 'book'	183
2.2.4	Klasse: 'person'	183
2.2.5	Klasse: 'author'	183
2.2.6	Klasse: 'reader'	183
3	'Full Continuations' im Einsatz	193
3.1	Mit Backtracking Probleme lösen	193
3.1.1	Backtracking-Funktionen	193
3.1.2	Das Problem der acht Königinnen	195
3.2	Multi-Tasking 'home made'	198
3.2.1	Einführung	198
3.2.2	Dispatcher für 'non-preemptive Multi-Tasking'	199
4	ARS++ - Anwendung: ARS-Interpreter	205
4.1	Einführung zu dem A++ - Interpreter in ARS++	205
4.2	Syntax von ARS	205
4.3	Beschreibung der Funktionen	206
4.3.1	Hauptfunktionen	206
4.3.2	ars-eval	207
4.3.3	ars-apply	208
4.3.4	get-referenced-value	210
4.3.5	binding-in-env	210
4.3.6	tagged-by?	210
4.3.7	add-modify-variable!	210
4.3.8	make-closure	211
4.3.9	force-eval!	211
4.3.10	list-of-real-values	212
4.3.11	list-of-args	212
4.3.12	eval-body-of-closure	212
4.3.13	add-frame-to-environment	212
4.3.14	make-environment-frame	213
4.3.15	print-object	213
4.3.16	driver	213
4.3.17	primitive-abstractions	213
4.3.18	build-global-environment	215
4.3.19	print-frame	215
4.3.20	print-env	215
4.3.21	print-composed	215
4.3.22	display-all	215
4.3.23	display-alln	215
4.3.24	the-global-environment	215
4.4	Quell-Code des A++ - Interpreters	216

4.5	Test-Suite für den A++ - Interpreters	221
5	ARS++ - Anwendung : XMLPROC	225
5.1	Implementierung einer XML-Datenbank	225
5.2	XML	225
5.2.1	Beispiel eines XML-Dokumentes	228
5.2.2	XML-Struktur in EBNF-Notation	229
5.3	Struktur der Datenbank	231
5.3.1	Datenhaltung	231
5.3.2	Detail-Struktur von XMLDB	232
5.4	Aufteilung in Module: Überblick	232
5.4.1	Generic Function User Interface (GENFUI)	233
5.4.2	Datenbank-Modul (XMLDB)	234
5.4.3	Allgemeiner Stream-Handler (XMLSTREAM)	235
5.4.4	Importierung von XML-Dokumenten (XML2DB)	236
5.4.5	Strukturanalyse der Eingabedaten (XMLPARSE)	237
5.4.6	XML-Lexical-Analyzer (XMLLEX)	238
5.4.7	Utility-Funktionen (XMLUTIL)	239
5.5	Beispiel eines Anwendungsdialoges mit XMLPROC	241
5.5.1	Starten der Anwendung	241
5.5.2	Anwendungsdialog	242
5.6	Implementierung von GENFUI	243
5.6.1	Tabelle der Funktionen enthalten in GENFUI	243
5.6.2	Implementierung der Klasse GENFUI	248
5.7	Implementierung von XMLSTREAM	252
5.7.1	Tabelle der Funktionen enthalten in XMLSTREAM	253
5.7.2	Implementierung der Klasse XMLSTREAM	253
5.8	Implementierung von XMLDB	255
5.8.1	Tabelle der Funktionen enthalten in XMLDB	256
5.8.2	Implementierung der Klasse XMLDB	257
5.9	Implementierung von XMLPARSE	265
5.9.1	Tabelle der Funktionen enthalten in XMLPARSE	265
5.9.2	Implementierung der Klasse XMLPARSE	269
5.10	Implementierung von XMLLEX	275
5.10.1	Reservierte Wörter für XMLLEX	275
5.10.2	Sonderzeichen für XMLLEX	276
5.10.3	Trennzeichen für XMLLEX	276
5.10.4	Tabelle der Funktionen in XMLLEX	277
5.10.5	Implementierung der Klasse XMLLEX	277
5.11	Implementierung von XML2DB	281
5.12	Implementierung von XMLUTIL	289
5.12.1	Tabelle der Funktionen enthalten in XMLUTIL	289
5.12.2	Implementierung der Klasse ASSOC	290
5.12.3	Tabelle der Methoden der Klasse ASSOC	291
5.12.4	Implementierung des Moduls XMLUTIL	291
6	XMLPROC mit Client/Server-Architektur	301
6.1	Anwendung von ARS++ in der zweiten Ausbaustufe	301
6.2	Überblick	303
6.2.1	Client Graphical User Interface	303
6.2.2	Kommunikationsschnittstelle zur Datenbank	304
6.2.3	Schnittstelle zur Berkeley-Datenbank	304
6.3	Ausführung des Programm-Codes	305
6.3.1	Aktivierung des Datenbank-Servers in ARS++	305
6.3.2	Aktivierung des Datenbank-Servers in 'dbscheme'	306

6.3.3	Aktivierung des Client-Programms	306
6.4	Beispiele für den Dialog mit dem Programm 'CLGUP'	306
6.4.1	Eröffnung des Programms 'CLGUP'	306
6.4.2	Anlegen einer Datenbank	306
6.4.3	Datenbank-Funktionen	308
6.4.4	Navigation durch die Datenbank	311
6.5	Implementierung der Benutzerschnittstelle	311
6.5.1	Implementierung des Client-Moduls in Tcl/Tk	312
6.5.2	Implementierung des Server-Moduls in ARS++	321

V ARS und Java

337

1	Programmierung in Java	339
1.1	Zweck und Umfang der Einführung in Java	339
1.2	Wichtige Abstraktionsmechanismen	339
1.2.1	Etwas einen Namen geben	339
1.2.2	Bildung von Klassen	340
1.3	Closures	342
1.4	Operationen in Funktionen	345
1.4.1	Allgemeine Grundfunktionen	345
1.4.2	Praktische Konkretisierung der allgemeinen Grundfunktionen	346
1.4.3	Bildung von Funktionen	347
1.5	Grundlegende Datentypen	351
1.5.1	Atom	351
1.5.2	Listen	352
1.5.3	Verarbeitung von Listen	357
1.5.4	Implementierung von Listen in Java	363
1.5.5	Dictionaries	371
1.6	Besonderheiten in Java	376
1.6.1	Fehler-Behandlung mit "try" und "catch"	376
1.6.2	"Threads"	377
1.7	Alternative Implementierung von Closures in Java	377
1.7.1	Datentypen-Überblick	377
1.7.2	Umgebung einer Lambda-Abstraktion	379
1.7.3	Klasse LClosure	386
1.7.4	Anwendungsbeispiele	387
2	ARS-Interpreter in Java	391
2.1	Einleitung	391
2.1.1	Syntax eines Lambda-Ausdrucks in EBNF-Notation:	391
2.1.2	Funktion und Struktur des Interpreters	391
2.2	Klasse für Symbole: Symbol	394
2.3	Klasse für Token: ARSToken	395
2.4	Klasse für den Parser: ARSParser	396
2.5	Klasse für Closures: ARSClosure	401
2.6	Klasse für den Evaluator: ARSEval	402
2.7	Klasse für die Fehlerbehandlung: ARSException	413
2.8	Klasse: ARSInterpreter	413
3	Komplette Anwendung in Java: OntoSimula	415
3.1	Einleitung	415
3.1.1	Simulation des Lebens von fünf Philosophen	415
3.1.2	Als 'applet' im Internet ausführbar	415
3.1.3	Detailliertes Szenario	415

3.2	Quellen für die Entwicklung der Fallstudie	416
3.3	Dynamisches Objektsystem für Java	416
3.3.1	Überblick	416
3.3.2	Klasse LJoops	420
3.3.3	Klasse LJoopsError	421
3.3.4	Klasse LJClasstable	421
3.3.5	Klasse LJclass	422
3.3.6	Klasse LJobject	423
3.3.7	Testprogramm für LJoops	427
3.4	Lambda-Klassen in Java	428
3.4.1	Überblick	428
3.5	Applet-Steuerklasse: PhilGraph	428
3.5.1	Darstellung auf dem Bildschirm	429
3.5.2	Ausführung in der Java-VM	429
3.6	Haupt-Anwendungsklasse OntoSimula	433
3.6.1	Dynamische Klassen	433
3.7	Haupt-Graphikklassen	444
3.7.1	Klasse für das Schloss: PhilCastle	444
3.7.2	Klasse für die Bedieneroberfläche	448
3.8	Statische Klassen für Sachen	453
3.8.1	Basis-Klasse ElementD	454
3.8.2	Klasse Chopstick	454
3.8.3	Klasse Room	455
3.9	Klassen für graphische Objekte	456
3.9.1	Basisklasse ImageObjectD	457
3.9.2	Klasse PersonImage	458
3.9.3	Klasse ChopstickImage	459
3.9.4	Hilfsklasse Semaphore	459
3.10	Methoden zur Auswahl des Verhaltensmusters	460
3.10.1	Methoden zur Auswahl des Verhaltensmusters für Aristoteles	460
3.10.2	Methoden zur Auswahl des Verhaltensmusters für Heraklit	461
3.10.3	Methoden zur Auswahl des Verhaltensmusters für Parmenides	462
3.10.4	Methoden zur Auswahl des Verhaltensmusters für Plato	463
3.10.5	Methoden zur Auswahl des Verhaltensmusters für Sokrates	464
3.11	Verhaltensmuster für alle Philosophen	465
3.11.1	Verhaltensmuster für heroische Einsatzbereitschaft	465
3.11.2	Verhaltensmuster für hohe Einsatzbereitschaft	465
3.11.3	Verhaltensmuster für normale Einsatzbereitschaft	466
3.11.4	Verhaltensmuster für gemäßigte Einsatzbereitschaft	467
3.11.5	Verhaltensmuster bei Krankheit	467
3.12	Spezielle Verhaltensmuster für Heraklit	468
3.12.1	Verhaltensmuster für heroische Einsatzbereitschaft	468
3.12.2	Verhaltensmuster für hohe Einsatzbereitschaft	469
3.12.3	Verhaltensmuster für normale Einsatzbereitschaft	469
3.13	Aktivitäten für alle Philosophen	470
3.13.1	Denken	470
3.13.2	Holen der Chopsticks	470
3.13.3	Essen	471
3.13.4	Zurückgeben der Chopsticks	471
3.13.5	Schlafen	472
3.13.6	Krankwerden	472
3.13.7	Altern	473
3.14	Spezielle Aktivitäten für Heraklit	473
3.14.1	Altern	473
3.15	Spezielle Aktivitäten für Sokrates	474

3.15.1	Holen der Chopsticks	474
3.15.2	Zurückgeben der Chopsticks	474
3.15.3	Altern	475
3.16	Spezielle Klasse für den Seins-Akt aller Seienden	475
3.17	Utility Klasse für die Verwaltung der Chopsticks	477

VI ARS und Python

479

1	Programmierung in Python	481
1.1	Allgemeine Syntax von Python	481
1.2	Wichtige Abstraktionsmechanismen	481
1.2.1	Bildung von Funktionen	481
1.2.2	Bildung von Klassen	482
1.2.3	Bildung von Modulen	484
1.3	Operationen in Funktionen	485
1.3.1	Definition von Variablen	485
1.3.2	Definition von Funktionen	485
1.3.3	Erzeugung von Objekten	485
1.3.4	Zuweisung von Werten	485
1.3.5	Testen von Werten	485
1.3.6	Wiederholung von Operationen	486
1.3.7	Rekursion	486
1.3.8	Ein-/Ausgabe-Operationen	487
1.3.9	Arithmetische Operationen	488
1.3.10	Relationale Operationen	488
1.3.11	Boolesche Operationen	488
1.4	Grundlegende Datentypen	489
1.4.1	Atome	489
1.4.2	Listen	489
1.4.3	Methoden für Listenobjekte	490
1.4.4	Hilfsfunktionen für Listen	490
1.5	Dictionaries	494
1.5.1	Methoden für Assoziative Listen	494
1.5.2	Hash-Tabellen	494
1.6	Besonderheiten in Python	495
2	ARS-Interpreter in Python	497
2.1	Einleitung	497
2.1.1	Syntax eines Lambda-Ausdrucks in EBNF-Notation:	497
2.1.2	Komponenten des Interpreters	497
2.2	Klasse X	499
2.3	Globale Variable	499
2.4	Klasse ARSSymbol	500
2.5	Klasse ARSToken	500
2.6	Klasse für den Parser: ARSParser	501
2.7	Klasse ARSClosure	502
2.8	Funktionen für die Auswertung von Ausdrücken	503
2.9	Klasse Interpreter	506
2.10	Austesten des Interpreters	507
2.11	Quellcode des Interpreters	508
2.12	Kompatibilitäts Modul für Listen	514

VII	ARS und C++	517
1	Hauptunterschied zwischen Java und C++	519
1.1	Fehlerdiagnostik	519
1.2	'Garbage Collection' in C++	519
2	Kurzeinführung in C++	521
2.1	Modularisierung	521
2.2	Variable, Zeiger und Referenzen	521
2.3	Definition von Klassen in C++	524
2.4	Erzeugung von Objekten	525
2.5	Kommunikation mit Objekten	526
2.6	Überladen von Operatoren	526
2.7	Zeiger auf das implizite Objekt	527
2.8	Operator zur Festlegung des Namensraumes	527
3	Einführung zum ARS-Interpreter in C++	529
3.1	Klassen-Hierarchie	529
3.2	Klasse LObject	530
3.3	Klasse Object	532
3.4	Klasse Integer	535
3.5	Klasse Symbol	536
3.6	Klasse String	537
3.7	Klasse List	538
3.8	Globale Funktionen	540
3.9	Klasse Null	542
3.10	Klasse Assoc	557
3.11	Klasse ARSParser	561
3.12	Klasse ARSClosure	566
3.13	Klasse ARSEval	569
3.14	Klasse LCar	580
3.15	Klasse LCadr	582
3.16	Klasse LIncr	583
3.17	Klasse LLoad	585
3.18	Klasse LPrint	587
3.19	ARS-Fehlerbehandlung	588
3.20	Hauptprogramm	589
3.21	Prozedur für die Erzeugung des ausführbaren Programmes	590
4	Direkte Umsetzung von ARS in C++	591
4.1	Schlüssel für die Umsetzung von ARS in C++	591
4.2	Datentypen für die Umsetzung von ARS in C++	591
4.3	Klasse 'Env' im Detail	594
4.4	Klasse 'Clam' im Detail	595
4.5	Anwendung der ARS-Klassen in C++	597
4.5.1	Abstraktion	597
4.5.2	Referenz	597
4.5.3	Synthese	597
4.5.4	Einfaches Beispiel	597
4.6	Testprogramm	598
4.6.1	Hauptprogramm	598
4.6.2	Test des funktionalen Programmierstils	599
4.6.3	Test des objekt-orientierten Programmierstils	600
4.6.4	Externe Hilfsmittel	601
4.7	Listen des Quell-Codes	601

VIII	ARS und C	615
1	Die Programmiersprache C	617
1.1	Charakterisierung der Programmiersprache C	617
1.2	Wichtige Abstraktionsmechanismen	617
1.2.1	Etwas einen Namen geben	617
1.2.2	Definition von abstrakten Datentypen (ADT)	618
1.2.3	Bildung von Funktionen	618
1.3	Operationen in Funktionen	619
1.3.1	Allgemeine Grundfunktionen	619
1.3.2	Definition von Variablen	619
1.3.3	Zuweisung von Werten	620
1.3.4	Bildung von Funktionen	620
1.3.5	Aufruf von Funktionen	621
1.3.6	Testen von Werten	621
1.3.7	Wiederholung von Operationen	622
1.3.8	Ein-/Ausgabeoperationen	623
1.3.9	Arithmetische Operationen	623
1.3.10	Relationale Operationen	623
1.3.11	Boolesche Operationen	624
1.3.12	Bibliotheksfunktionen	624
1.3.13	Testprogramm für die Grundoperationen	624
1.4	Einfache und zusammengesetzte Ausdrücke	625
1.4.1	Atom	625
1.4.2	Zusammengesetzter Ausdruck	626
1.5	Strukturierungs-Hilfsmittel in C	626
1.5.1	typedef	626
1.5.2	enum	626
1.5.3	struct	627
1.5.4	union	627
1.5.5	setjmp/longjmp	628
2	ARS-Interpreter in C	629
2.1	Einführung	629
2.2	Datenstrukturen im ARS-Interpreter	629
2.2.1	NAME	629
2.2.2	PRIM	629
2.2.3	VTYPE	630
2.2.4	EXPTYPE	630
2.2.5	ABS	630
2.2.6	SYN	631
2.2.7	EXP	631
2.2.8	ELIST	631
2.2.9	NLIST	631
2.2.10	VLIST	632
2.2.11	ENV	632
2.2.12	LFUN	633
2.2.13	CLAM	633
2.2.14	ACL	634
2.2.15	THUNK	634
2.2.16	VALUE	635
2.3	Beschreibung der Funktionen	635
2.3.1	Konstruktoren	635
2.3.2	Der Umgang mit Variablennamen	638
2.3.3	Variable und Umgebungen	638

2.3.4	Auswertung von Ausdrücken	639
2.3.5	Schnittstellen für die Eingabe	642
2.3.6	Einlesen von ARS-Code durch den ARS-Parser	643
2.4	Quellcode des ARS-Interpreters	644
2.4.1	Modul ARSC: Datentypen und Prototypen	644
2.4.2	Modul ARSC: Implementierung	647
2.4.3	Modul ARSParser	659
2.4.4	Prozedur zur Erstellung des Programms	663
2.5	Performance-Vergleich der ARS-Interpreter	663
3	Direkte Umsetzung von ARS in C	665
3.1	Die Muschel als Schlüssel für die Umsetzung von ARS in C	665
3.2	Datentypen	665
3.3	Anwendung	667
3.3.1	Abstraktion	667
3.3.2	Referenz	668
3.3.3	Synthese	668
3.4	Einfaches Beispiel	668
3.4.1	Bildung der C-Funktion	668
3.4.2	Erweiterung der Funktion zu einer Lambda-Abstraktion	669
3.4.3	Synthese der Lambda-Abstraktion mit einem Argument	669
3.5	Komplexeres Beispiel	669
3.5.1	Hauptprogramm	669
3.5.2	Test des objekt-orientierten Programmierstils	670
3.5.3	Test des funktionalen Programmierstils	672
3.5.4	Testmodul: Funktionsprototypen	673
3.5.5	Hauptsteuerung und Testmodul	673
3.6	Externe Hilfsmittel	677
IX	ARSAPI	679
1	Einleitung	681
1.1	Das Akronym ARSAPI	681
1.2	Das Bild der Muschel	681
1.3	Anwendung von ARSAPI in der Praxis	681
2	ARS (A++)	685
2.1	Syntax und Semantik von ARS	685
2.2	Grundlegende Logische Abstraktionen in ARS	685
2.3	Erweiterte Logische Abstraktionen in ARS	686
2.4	Numerische Abstraktionen in ARS	686
2.5	Grundlegende Abstraktionen für Listen (1) in ARS	687
2.6	Grundlegende Abstraktionen für Listen (2) in ARS	687
2.7	Erweiterte Arithmetische Abstraktionen in ARS	688
2.8	Relationale Abstraktionen in ARS	688
2.9	Erweiterte Abstraktionen für Listen in ARS	689
2.10	Funktionen Höherer Ordnung	689
2.11	Mengen-Operationen	690
2.12	Imperative Programmierung in ARS	691
2.13	Support-Funktionen für ARS	691
2.14	Objekt-Orientierte Programmierung in ARS	692
3	ARSAPI für Java	693
3.1	Datentypen-Überblick	693
3.2	Umgebung einer Lambda-Abstraktion	695

3.2.1	Klasse 'LEnv'	695
3.2.2	Klasse 'NList'	696
3.2.3	Klasse 'VList'	697
3.3	Klasse LClosure	699
3.4	Allgemeine Listen: Klasse LList	701
4	ARSAPI für C	707
4.1	Datentyp 'NAME'	707
4.2	name. table	708
4.3	Abstrakter Datentyp 'VALUE'	709
4.4	Datentyp 'INTV'	710
4.5	Datentyp 'SYMV'	710
4.6	Datentyp 'STRV'	710
4.7	Datentyp 'PAIRV'	711
4.8	Datentyp 'NLIST'	712
4.9	Datentyp 'VLIST'	712
4.10	Datentyp 'ENV'	713
4.11	Datentyp 'CLAM'	714
4.12	Beispiel 'trivy' für C	714
4.13	Anwendungsbeispiel	715
5	ARSAPI für C++	721
5.1	Datentypen-Überblick	721
5.2	Allgemeine Typendefinitionen	723
5.3	Umgebung einer Lambda-Abstraktion	723
5.3.1	Klasse 'Env'	723
5.3.2	Klasse 'NList'	725
5.3.3	Klasse 'VList'	725
5.4	Klasse Clam	726
5.5	Allgemeine Listen: Klasse List	726
5.6	Klasse Integer	727
5.7	Klasse String	727
5.8	Klasse Symbol	728
5.9	Globale Funktionen	728
5.10	Beispiel 'trivy' für C++	729
5.11	Anwendungsbeispiel	730
X	ARS++ - Implementierung	735
1	Virtuelle Maschine von ARS++: AVIM	737
1.1	Überblick	737
1.1.1	Allgemeine Beschreibung	737
1.1.2	Eigenschaften der ARS++ - Implementierung	738
1.1.3	Hauptkomponenten von ARS++	738
1.2	Aufbau von AVIM	739
1.2.1	S-Register	739
1.2.2	E-Register	739
1.2.3	C-Register	739
1.2.4	D-Register	739
1.2.5	IS-Register	739
1.2.6	G-Register	740
1.3	Befehle der virtuellen Maschine 'AVIM'	740
1.3.1	Befehlsrepertoire: Überblick	741
1.3.2	Befehlsrepertoire: Detailbeschreibung	741

1.3.3	Basisstrukturen in der AVIM-Programmierung	747
1.4	Kleine Anwendungsbeispiele	749
1.4.1	Einfache Addition	749
1.4.2	Addition und Multiplikation	751
1.4.3	IF-THEN-ELSE-Struktur	752
1.4.4	Deklaration von Variablen	752
1.4.5	Definition von Lambda-Abstraktionen	752
1.4.6	Synthese von zwei Lambda-Ausdrücken	753
1.4.7	Rekursive Berechnung der Fakultät	754
1.4.8	Berechnung der Fakultät mit 'tail recursion'	755
2	Implementierung von ARS++ als Prototyp in Scheme	757
2.1	Besondere Eigenschaften der Implementierung von ARS++	757
2.1.1	'tail recursion'	757
2.1.2	'full continuations'	757
2.1.3	'define'-Anweisung	758
2.1.4	Makros	758
2.2	Einführung zum Prototypen von AVIM	758
2.2.1	Besonderheiten des Prototyps	758
2.3	Anwendung der ARS++ - Prototypen	759
2.3.1	Laden der ARS++ - Prototypen	759
2.3.2	Benutzung des Compilers	760
2.3.3	Ausführung des Programm-Codes in der virtuellen Maschine	760
2.3.4	Integrierter Aufruf des Compilers/Interpreters	760
2.4	Quellcode des Prototypen für die virtuelle Maschine	761
3	Erweiterung von ARSAPI für C	767
3.1	Überblick	767
3.1.1	Integration von Primitivfunktionen	767
3.1.2	Ausbau des Datentyps 'STRV'	767
3.1.3	Verzicht auf den Datentyp 'VLIST'	767
3.1.4	Einführung des Datentyps 'LAMBV'	768
3.1.5	Einführung des Datentyps 'VECV'	768
3.1.6	Integration einer Ein-/Ausgabe-Bibliothek	768
3.1.7	Integration einer Datenbankschnittstelle	769
3.1.8	Integration einer TCP/IP-Schnittstelle	769
3.1.9	Integration einer Schnittstelle für Reguläre Ausdrücke	769
3.1.10	Erweiterung der Verwaltung der Namensräume	769
3.1.11	Datentypen in ARSAPI für C	770
3.2	Datentyp 'NAME'	770
3.3	Abstrakter Datentyp 'VALUE'	774
3.4	Datentyp 'GVAR'	775
3.5	Datentypen 'INTV/DBLV'	776
3.6	Datentyp 'SYMV'	777
3.7	Datentyp 'STRV'	777
3.8	Datentyp 'PAIRV'	778
3.9	Datentyp 'VECV'	780
3.10	Datentyp 'ENVV'	781
3.11	Datentyp 'CLAM'	781
3.12	Datentyp 'LFUN'	782
3.13	Datentyp 'LAMB'	782
3.14	Datentyp 'PRIM'	783
3.15	Datentyp 'CFUN'	783
3.16	Datentyp 'INPV'	783
3.17	Datentyp 'OTPV'	784

3.18	Datentyp 'HT'	784
3.19	Datentyp 'DBMV'	785
3.20	Datentyp 'datum'	785
3.21	Datentyp 'TCPV'	786
3.22	Datentyp 'REGXV'	786
4	Implementierung von AVIM in C	787
4.1	Auszug aus der Definitionsdatei	787
4.2	Quellcode der Virtuelle Maschine	793
4.3	Quellcode für das Namens-Management	803
4.4	Quellcode für das Umgebungs-Management	805
4.5	Quellcode für das Funktions-Management	810
4.6	Testen von AVIM	812
5	AVIM mit internem Speichermanagement	813
5.1	Einführung	813
5.2	Grundlegendes Verfahren	813
5.3	Hauptprozeduren des Garbage Collectors	814
5.3.1	Modul-globale Variable	814
5.3.2	Prozedur 'init_mem'	814
5.3.3	Prozedur 'switch_mem'	814
5.3.4	Prozedur 'init_new_half'	815
5.3.5	Prozedur 'traverse'	815
5.3.6	Prozedur 'move_value'	816
5.3.7	Prozedur 'valmem'	818
5.3.8	Prozedur 'ars_malloc'	819
5.3.9	Prozedur 'ars_vmalloc'	819
5.3.10	Utility-Prozeduren des Speichermanagers	819
6	ARS++ - Compiler: ACOMP	821
6.1	Zweck und Versionen des Compilers	821
6.2	Hauptfunktion 'compile'	821
6.3	Kompilation einer Konstanten	825
6.4	Kompilation eines Symbols	825
6.5	Kompilation eines Makros	825
6.6	Kompilation eines 'quote'-Ausdrucks	825
6.7	Kompilation einer 'define'-Anweisung	825
6.8	Kompilation einer 'set!'-Anweisung	826
6.9	Kompilation einer 'lambda'-Abstraktion	826
6.10	Kompilation eines 'if'-Konstruktes	827
6.11	Kompilation eines 'let'-Konstruktes	827
6.12	Kompilation von Synthesen	828
6.12.1	Kompilation eines allgemeinen Funktionsaufrufs	828
6.12.2	Kompilation eines Funktionsaufrufs mit 'apply'	829
6.12.3	Kompilation eines 'call-with-current-continuation'-Konstruktes	830
6.13	Kompilation eines 'begin'-Konstruktes	830
6.14	Kompilation von Spracherweiterungen (Makros)	831
6.15	Kompilation eines 'letrec'-Makros	831
6.16	Kompilation eines 'cond'-Makros	832
6.17	Kompilation eines 'case'-Makros	833
6.18	Kompilation eines 'or'-Makros	833
6.19	Kompilation eines 'and'-Makros	834
6.20	Kompilation eines 'while'-Makros	834
6.21	Kompilation eines 'do'-Makros	835
6.22	Kompilation von 'defmacro'	835

6.23	Kompilation von Benutzermakros	836
6.24	Kompilation von 'quit', 'evals' und 'load'	836
6.25	Hilfsfunktionen	837
6.25.1	Funktionen 'comp-file', 'read-file' und allg. Funktionen	837
6.25.2	Weitere Hilfsfunktionen des Compilers wie 'comp', 'comp-exps' und andere	838
6.26	Benutzerschnittstelle des Compilers	839
6.27	Hinweise zur Benutzung	843

Anhänge 847

A	Das Lambda-Kalkül	847
A.1	Syntax eines Lambda-Ausdrucks	847
A.2	Begriffe und Regeln des <i>Lambda-Kalküls</i>	847
A.2.1	Assoziativitätsregeln	847
A.2.2	Gebundene und freie Variable	848
A.2.3	Alpha-Konvertierung	848
A.2.4	Beta-Reduktion	848
A.2.5	Eta-Reduktion	849
A.2.6	Y-Kombinator	849
A.3	Beispiele für Beta-Reduktion	850
A.3.1	Lambda-Kalkül-Programmierung in Scheme-Codierung	850
A.3.2	Auszuwertende Lambda-Ausdrücke in Scheme-Codierung	851
A.3.3	Basis-Abstraktionen in Scheme-Codierung	851
A.3.4	Anwendung mit Beta-Reduktion	853
A.4	Lambda-Kalkül-Programmierung in Python	857
A.4.1	Besonderheiten in Python	857
A.4.2	Anwendungsbeispiel in Python:	857
B	Gültigkeitsbereich von Namen	859
B.1	Interpretation von Namen	859
B.1.1	Dynamic Scope	859
B.1.2	Static Scope	859
B.1.3	Global Scope	860
B.1.4	Local Scope	860
B.2	Auswirkung der Art der Symbolinterpretation auf die Programmierung	860
B.2.1	Auswirkung von „Dynamic Scope“ auf die Programmierung	860
B.2.2	Auswirkung von „Static Scope“ auf die Programmierung	861
B.2.3	Verdeutlichung der Unterschiede von „dynamic scope“ und „lexical scope“ anhand von Beispielen	861
C	Ergänzungen zu OntoSimula	865
C.1	OntoSimula in Scheme	865
C.2	Philosophischer Hintergrund zu OntoSimula	866
C.2.1	Einleitung	866
C.2.2	Abbildung der Prinzipien in OntoSimula (<i>Java</i>)	867
C.2.3	Abbildung der Prinzipien in OntoSimula (<i>Scheme</i>)	872
D	Werkzeuge und Hilfsmittel	877
D.1	Erfahrung mit Scheme-Implementierungen	877
D.1.1	gambit-C	877
D.1.2	libscheme	877
D.1.3	dbscheme	877
D.1.4	ARS++	878
D.2	In den Benchmarktests verwendete Scheme-Implementierungen	879

D.2.1	'Gambit-C'	879
D.2.2	dbscheme	879
D.2.3	ARS++	880
D.2.4	DrScheme	880
D.2.5	MzScheme	881
D.2.6	'guile'	881
D.2.7	'kawa'	881
D.2.8	'silk'	881
D.2.9	'jscheme'	882
D.2.10	'scheme-package'	882
D.2.11	'sisc'	882
D.3	Andere in diesem Buch verwendete Werkzeuge	882
D.3.1	Java Development Kit	882
D.3.2	Python Interpreter	883
D.3.3	C/C++ Compiler	883
D.3.4	Tcl/Tk Interpreter	883
D.3.5	Cygwin	883
D.4	Benchmark von Scheme-Implementierungen	884
D.5	Buch CD-ROM	887
D.6	WWW-Adressen	889
Nachwort		891
Biographische Daten zur Person des Autors		893
Literaturverzeichnis		895
Index		899