

Inhaltsübersicht

Vorwort	XXIX
Zur deutschen Ausgabe	XXXIII
Kapitel 1 Einleitung	1
Kapitel 2 Ein einfacher syntaxgerichteter Übersetzer ..	49
Kapitel 3 Lexikalische Analyse	133
Kapitel 4 Syntaktische Analyse	231
Kapitel 5 Syntaxgerichtete Übersetzung	363
Kapitel 6 Zwischencodeerzeugung	431
Kapitel 7 Laufzeitumgebungen	519
Kapitel 8 Codeerzeugung	617
Kapitel 9 Maschinenunabhängige Optimierungen ...	711
Kapitel 10 Parallelität auf Befehlsebene	857
Kapitel 11 Optimierungen für Parallelität und Lokalität	929
Kapitel 12 Interprozedurale Analyse	1091
Anhang	1163
Liste mit englischen Begriffen und deren Übersetzung ...	1223
Liste mit deutschen Begriffen und deren Übersetzung ...	1225
Index	1227
Die Autoren	1253

Inhaltsverzeichnis

Vorwort	XXIX
Zur deutschen Ausgabe	XXXIII
Kapitel 1 Einleitung	1
1.1 Sprachprozessoren	3
Übungen zu Abschnitt 1.1	5
1.2 Die Struktur eines Compilers	6
1.2.1 Lexikalische Analyse	7
1.2.2 Syntaxanalyse	9
1.2.3 Semantische Analyse	9
1.2.4 Zwischencodeerzeugung	11
1.2.5 Codeoptimierung	12
1.2.6 Codeerzeugung	13
1.2.7 Umgang mit Symboltabellen	14
1.2.8 Gruppieren von Phasen in Läufe	14
1.2.9 Werkzeuge zum Compilerbau	15
1.3 Die Evolution der Programmiersprachen	16
1.3.1 Der Weg zu höheren Programmiersprachen	16
1.3.2 Einfluss auf Compiler	17
Übung zu Abschnitt 1.3	18
1.4 Die Wissenschaft des Compilerbaus	19
1.4.1 Modellierung bei Compilerdesign und -implementation	19
1.4.2 Die Wissenschaft der Codeoptimierung	20
1.5 Anwendungen der Compilertechnologie	22
1.5.1 Implementierung von höheren Programmier- sprachen	22
1.5.2 Optimierungen für Computerarchitekturen	24
1.5.3 Entwurf neuer Computerarchitekturen	26
1.5.4 Programmübersetzung	27
1.5.5 Werkzeuge zur Produktivitätssteigerung	29

1.6	Grundlagen von Programmiersprachen	32
1.6.1	Unterscheidung zwischen statisch und dynamisch	32
1.6.2	Umgebungen und Zustände	33
1.6.3	Statischer Gültigkeitsbereich und Blockstruktur	35
1.6.4	Explizite Zugriffskontrolle	39
1.6.5	Dynamischer Gültigkeitsbereich	40
1.6.6	Mechanismen zur Parameterübergabe	42
1.6.7	Aliasing	44
	Übungen zu Abschnitt 1.6	44
	Zusammenfassung	46
	Literatur zu Kapitel 1	48
Kapitel 2 Ein einfacher syntaxgerichteter Übersetzer		49
2.1	Einführung	51
2.2	Syntaxdefinition	53
2.2.1	Grammatikdefinition	54
2.2.2	Ableitungen	56
2.2.3	Parse-Bäume	57
2.2.4	Mehrdeutigkeit	59
2.2.5	Assoziativität von Operatoren	60
2.2.6	Operatorenpräzedenz	61
	Übungen zu Abschnitt 2.2	64
2.3	Syntaxgerichtete Übersetzung	66
2.3.1	Postfixnotation	67
2.3.2	Synthetisierte Attribute	68
2.3.3	Einfache syntaxgerichtete Definitionen	70
2.3.4	Durchlaufen von Bäumen	71
2.3.5	Übersetzungsverfahren	73
	Übungen zu Abschnitt 2.3	75
2.4	Syntaxanalyse (Parsing)	76
2.4.1	Top-Down-Syntaxanalyse	76
2.4.2	Prädiktive Syntaxanalyse	79
2.4.3	Verwendungszweck von ϵ -Produktionen	82
2.4.4	Entwurf eines prädiktiven Parsers	82
2.4.5	Linksrekursion	84
	Übung zu Abschnitt 2.4	85

2.5	Übersetzer für einfache Ausdrücke	86
2.5.1	Abstrakte und konkrete Syntax	87
2.5.2	Anpassen des Übersetzungsverfahrens	88
2.5.3	Prozeduren für die Nichtterminale	89
2.5.4	Vereinfachen des Übersetzers	90
2.5.5	Das vollständige Programm	91
2.6	Lexikalische Analyse	94
2.6.1	Entfernen von Leerzeichen und Kommentaren	96
2.6.2	Vorausschauendes Lesen	96
2.6.3	Konstanten	97
2.6.4	Erkennen von Schlüsselwörtern und Bezeichnern	98
2.6.5	Ein lexikalischer Scanner (kurz Lexer)	100
	Übungen zu Abschnitt 2.6	104
2.7	Symboltabellen	105
2.7.1	Symboltabellen nach Gültigkeitsbereich	106
2.7.2	Verwendung von Symboltabellen	110
2.8	Zwischencodeerzeugung	113
2.8.1	Zwei Arten der Zwischendarstellung	113
2.8.2	Konstruktion von Syntaxbäumen	114
2.8.3	Statische Überprüfung	119
2.8.4	Drei-Adress-Code	121
	Übungen zu Abschnitt 2.8	129
	Zusammenfassung	130
Kapitel 3 Lexikalische Analyse		133
3.1	Die Rolle des Lexers	135
3.1.1	Lexikalische Analyse und Syntaxanalyse im Vergleich	136
3.1.2	Token, Muster und Lexeme	136
3.1.3	Attribute für Token	138
3.1.4	Lexikalische Fehler	139
	Übungen zu Abschnitt 3.1	140
3.2	Eingabepuffer	141
3.2.1	Pufferpaare	141
3.2.2	Wächter	143

3.3	Spezifikation von Token	144
3.3.1	Strings und Sprachen	144
3.3.2	Operationen an Sprachen	146
3.3.3	Reguläre Ausdrücke	147
3.3.4	Reguläre Definitionen	149
3.3.5	Erweiterungen regulärer Ausdrücke	150
	Übungen zu Abschnitt 3.3	151
3.4	Tokenerkennung	156
3.4.1	Übergangsdigramme	158
3.4.2	Erkennen von reservierten Wörtern und Bezeichnern	160
3.4.3	Abschluss des Beispiels	161
3.4.4	Architektur eines Lexers auf der Grundlage von Übergangsdigrammen	162
	Übungen zu Abschnitt 3.4	165
3.5	Der Generator Lex für lexikalische Scanner	170
3.5.1	Verwendung von Lex	170
3.5.2	Struktur von Lex-Programmen	171
3.5.3	Konfliktlösung in Lex	174
3.5.4	Der Lookahead-Operator	175
	Übungen zu Abschnitt 3.5	176
3.6	Endliche Automaten	178
3.6.1	Nichtdeterministische endliche Automaten	178
3.6.2	Übergangstabellen	179
3.6.3	Akzeptieren von Eingabestrings durch Automaten	180
3.6.4	Deterministische endliche Automaten	181
	Übungen zu Abschnitt 3.6	183
3.7	Von regulären Ausdrücken zu Automaten	184
3.7.1	Umwandlung eines NFA in einen DFA	184
3.7.2	Simulation eines NFA	188
3.7.3	Effizienz der NFA-Simulation	189
3.7.4	Aufbau eines NFA aus einem regulären Ausdruck	192
3.7.5	Effizienz von stringverarbeitenden Algorithmen	197
	Übungen zu Abschnitt 3.7	200
3.8	Entwurf eines Generators für lexikalische Scanner	201
3.8.1	Die Struktur des generierten Scanners	201

	3.8.2	Pattern Matching auf der Grundlage von NFAs	203
	3.8.3	DFAs für lexikalische Scanner	205
	3.8.4	Implementieren des Lookahead-Operators	206
		Übungen zu Abschnitt 3.8	208
3.9		Optimierung des Pattern Matching auf DFA-Grundlage	209
	3.9.1	Wichtige Zustände eines NFA	209
	3.9.2	Aus dem Syntaxbaum berechnete Funktionen	211
	3.9.3	Berechnen von nullable, firstpos und lastpos	213
	3.9.4	Berechnen von followpos	214
	3.9.5	Direkte Konvertierung eines regulären Ausdruckes in einen DFA	216
	3.9.6	Minimierung der Anzahl von Zuständen eines DFA	217
	3.9.7	Zustandsminimierung in lexikalischen Scannern	222
	3.9.8	Kompromisse zwischen Raum und Zeit bei der DFA-Simulation	223
		Übungen zu Abschnitt 3.9	225
		Zusammenfassung	226
		Literatur zu Kapitel 3	228
Kapitel 4 Syntaktische Analyse			231
4.1		Einführung	233
	4.1.1	Die Rolle des Parsers	233
	4.1.2	Repräsentative Grammatiken	234
	4.1.3	Behandlung von Syntaxfehlern	235
	4.1.4	Strategien für die Fehlerbehebung	237
4.2		Kontextfreie Grammatiken	239
	4.2.1	Formale Definition einer kontextfreien Grammatik	239
	4.2.2	Konventionen für die Notation	240
	4.2.3	Ableitungen	242
	4.2.4	Parse-Bäume und Ableitungen	244
	4.2.5	Mehrdeutigkeit	246
	4.2.6	Verifizieren der von einer Grammatik generierten Sprache	247

	4.2.7	Kontextfreie Grammatiken und reguläre Ausdrücke im Vergleich	248
		Übungen zu Abschnitt 4.2	250
4.3		Schreiben einer Grammatik	254
	4.3.1	Lexikalische und syntaktische Analyse	254
	4.3.2	Eliminieren von Mehrdeutigkeiten	255
	4.3.3	Eliminieren der Linksrekursion	257
	4.3.4	Linksfaktorisierung	259
	4.3.5	Nicht kontextfreie Sprachkonstrukte	260
		Übungen zu Abschnitt 4.3	262
4.4		Top-Down-Parsing	263
	4.4.1	Rekursiv absteigendes Parsing	264
	4.4.2	FIRST und FOLLOW	266
	4.4.3	LL(1)-Grammatiken	269
	4.4.4	Nichtrekursive prädiktive Syntaxanalyse	273
	4.4.5	Fehlerbehebung bei der prädiktiven Syntaxanalyse	275
		Übungen zu Abschnitt 4.4	278
4.5		Bottom-Up-Parsing	282
	4.5.1	Reduktionen	282
	4.5.2	Handle-Stützung	283
	4.5.3	Shift-Reduce-Syntaxanalyse	285
	4.5.4	Konflikte bei der Shift-Reduce-Syntaxanalyse	287
		Übungen zu Abschnitt 4.5	290
4.6		Einführung in die LR-Syntaxanalyse: einfaches LR	291
	4.6.1	Warum LR-Parser?	291
	4.6.2	Items und der LR(0)-Automat	292
	4.6.3	Der LR-Parsealgorithmus	299
	4.6.4	Aufbau von SLR-Parsertabellen	304
	4.6.5	Sinnvolle Präfixe	307
		Übungen zu Abschnitt 4.6	310
4.7		Leistungsfähigere LR-Parser	312
	4.7.1	Kanonische LR(1)-Items	312
	4.7.2	Aufbau von LR(1)-Item-Mengen	314
	4.7.3	Kanonische LR(1)-Parsertabellen	318
	4.7.4	Aufbau von LALR-Parsertabellen	319
	4.7.5	Effizienter Aufbau von LALR-Parsertabellen	324

	4.7.6	Komprimierung von LR-Parsertabellen	329
		Übungen zu Abschnitt 4.7	332
4.8		Mehrdeutige Grammatiken	333
	4.8.1	Präzedenz und Assoziativität zur Konfliktlösung	333
	4.8.2	Mehrdeutigkeit durch ein „hängendes else“	336
	4.8.3	Fehlerbehebung beim LR-Parsing	338
		Übungen zu Abschnitt 4.8	341
4.9		Parsergeneratoren	343
	4.9.1	Der Parsergenerator Yacc	343
	4.9.2	Einsatz von Yacc bei mehrdeutigen Grammatiken	347
	4.9.3	Erstellen von Yacc-Lexern mit Lex	350
	4.9.4	Fehlerbehebung bei Yacc	351
		Übungen zu Abschnitt 4.9	354
		Zusammenfassung	355
		Literatur zu Kapitel 4	358
Kapitel 5 Syntaxgerichtete Übersetzung			363
5.1		Syntaxgerichtete Definitionen	366
	5.1.1	Erebt und synthetisierte Attribute	366
	5.1.2	Auswerten einer syntaxgerichteten Definition an den Knoten eines Parse-Baumes	368
		Übungen zu Abschnitt 5.1	372
5.2		Auswerten einer syntaxgerichteten Definition an den Knoten eines Parse-Baumes	373
	5.2.1	Abhängigkeitsgraphen	373
	5.2.2	Reihenfolge der Auswertung von Attributen	375
	5.2.3	S-attributierte Definitionen	376
	5.2.4	L-attributierte Definitionen	377
	5.2.5	Semantische Regeln mit kontrollierten Nebenwirkungen	378
		Übungen zu Abschnitt 5.2	381
5.3		Anwendungen der syntaxgerichteten Übersetzung	383
	5.3.1	Aufbau von Syntaxbäumen	383
	5.3.2	Die Struktur eines Typs	387
		Übungen zu Abschnitt 5.3	389

5.4	Verfahren zur syntaxgerichteten Übersetzung	390
5.4.1	Postfix-Übersetzungsverfahren	391
5.4.2	Parserstack-Implementierungen von syntaxgerichteten Postfix-Übersetzungen	391
5.4.3	Syntaxgerichtete Übersetzungen mit Aktionen innerhalb von Produktionen	393
5.4.4	Eliminieren der Linksrekursion aus syntaxgerichteten Übersetzungen	395
5.4.5	Syntaxgerichtete Übersetzungen für L-attributierte Definitionen	398
	Übungen zu Abschnitt 5.4	405
5.5	Implementieren von L-attribuierten syntaxgerichteten Definitionen	407
5.5.1	Übersetzung bei der rekursiv absteigenden Syntaxanalyse	408
5.5.2	Codeerzeugung im laufenden Betrieb	411
5.5.3	L-attributierte syntaxgerichtete Definitionen und LL-Syntaxanalyse	413
5.5.4	Bottom-Up-Syntaxanalyse von L-attribuierten syntaxgerichteten Definitionen	420
	Übungen zu Abschnitt 5.5	425
	Zusammenfassung	426
	Literatur zu Kapitel 5	428
Kapitel 6 Zwischencodeerzeugung		431
6.1	Varianten von Syntaxbäumen	434
6.1.1	Gerichtete azyklische Graphen für Ausdrücke	434
6.1.2	Die Wertenummermethode für die Konstruktion von DAGs	436
	Übungen zu Abschnitt 6.1	439
6.2	Drei-Adress-Code	440
6.2.1	Adressen und Befehle	440
6.2.2	Quadrupel	443
6.2.3	Tripel	444
6.2.4	Statische Einzelzuweisungsform	446
	Übungen zu Abschnitt 6.2	448

6.3	Typen und Deklarationen	449
	6.3.1 Typausdrücke	449
	6.3.2 Typäquivalenz	451
	6.3.3 Deklarationen	452
	6.3.4 Speicherlayout für lokale Namen	452
	6.3.5 Sequenzen aus Deklarationen	455
	6.3.6 Felder in Strukturen und Klassen	456
	Übungen zu Abschnitt 6.3	458
6.4	Übersetzung von Ausdrücken	459
	6.4.1 Operationen in Ausdrücken	459
	6.4.2 Inkrementelle Übersetzung	461
	6.4.3 Adressieren von Arrayelementen	462
	6.4.4 Übersetzung von Arrayreferenzen	464
	Übungen zu Abschnitt 6.4	467
6.5	Typüberprüfung	469
	6.5.1 Regeln für die Typüberprüfung	469
	6.5.2 Typkonvertierung	470
	6.5.3 Überladen von Funktionen und Operatoren	473
	6.5.4 Typinferenz und polymorphe Funktionen	474
	6.5.5 Ein Unifikationsalgorithmus	479
	Übungen zu Abschnitt 6.5	483
6.6	Kontrollfluss	484
	6.6.1 Boolesche Ausdrücke	484
	6.6.2 Short-Circuit-Code	485
	6.6.3 Kontrollflussanweisungen	486
	6.6.4 Kontrollflussübersetzung von booleschen Ausdrücken	489
	6.6.5 Vermeiden redundanter Goto-Befehle	491
	6.6.6 Boolesche Werte und Sprungcode	493
	Übungen zu Abschnitt 6.6	495
6.7	Backpatching	497
	6.7.1 Einpass-Codeerzeugung mit Backpatching	497
	6.7.2 Backpatching für boolesche Ausdrücke	498
	6.7.3 Steuerungsflussanweisungen	501
	6.7.4 Break-, continue- und goto-Anweisungen	504
	Übungen zu Abschnitt 6.7	505

6.8	Switch-Anweisungen	507
6.8.1	Übersetzung von switch-Anweisungen	507
6.8.2	Syntaxgerichtete Übersetzung von switch-Anweisungen	508
	Übung zu Abschnitt 6.8	510
6.9	Zwischencode für Prozeduren	511
	Zusammenfassung	514
	Literatur zu Kapitel 6	516
Kapitel 7 Laufzeitumgebungen		519
7.1	Speicheraufbau	521
7.1.1	Statische und dynamische Speicherzuweisung	523
7.2	Speicherzuweisung auf dem Stack	524
7.2.1	Aktivierungsbäume	524
7.2.2	Aktivierungseinträge	528
7.2.3	Aufrufsequenzen	531
7.2.4	Daten variabler Länge auf dem Stack	534
	Übungen zu Abschnitt 7.2	536
7.3	Zugriff auf nichtlokale Daten auf dem Stack	538
7.3.1	Datenzugriff ohne verschachtelte Prozeduren	538
7.3.2	Probleme bei verschachtelten Prozeduren	539
7.3.3	Eine Sprache mit Deklarationen für verschachtelte Prozeduren	539
7.3.4	Verschachtelungstiefe	541
7.3.5	Zugriffslinks	542
7.3.6	Bearbeiten von Zugriffslinks	544
7.3.7	Zugriffslinks für Prozedurparameter	545
7.3.8	Displays	547
	Übungen zu Abschnitt 7.3	550
7.4	Heap-Verwaltung	551
7.4.1	Der Speichermanager	551
7.4.2	Die Speicherhierarchie eines Computers	553
7.4.3	Lokalität in Programmen	555
7.4.4	Verringern der Fragmentierung	557
7.4.5	Manuelle Speicherfreigabe	561
	Übung zu Abschnitt 7.4	565

7.5	Einführung in die Garbage Collection	566
7.5.1	Entwurfsziele für Garbage Collectors	566
7.5.2	Erreichbarkeit	569
7.5.3	Garbage Collectors mit Referenzzählern	572
	Übungen zu Abschnitt 7.5	574
7.6	Einführung in die Garbage Collection mit Zeigerverfolgung	575
7.6.1	Ein einfacher Mark-and-Sweep-Collector	575
7.6.2	Grundlegende Abstraktion	577
7.6.3	Optimieren der Mark-and-Sweep-Collection	580
7.6.4	Mark-and-Compact-Collectors	581
7.6.5	Kopier-Collectors	585
7.6.6	Kostenvergleich	587
	Übungen zu Abschnitt 7.6	588
7.7	Garbage Collection mit kurzen Pausen	590
7.7.1	Inkrementelle Garbage Collection	590
7.7.2	Inkrementelle Erreichbarkeitsanalyse	592
7.7.3	Grundlagen der teilweisen Garbage Collection	595
7.7.4	Garbage Collection nach Generationen	596
7.7.5	Der Zugalgorithmus	598
	Übungen zu Abschnitt 7.7	603
7.8	Fortgeschrittene Themen der Garbage Collection	604
7.8.1	Parallele und gleichzeitige Garbage Collection	604
7.8.2	Teilweise Objektverschiebung	607
7.8.3	Konservative Garbage Collection für unsichere Sprachen	608
7.8.4	Schwache Referenzen	609
	Übung zu Abschnitt 7.8	610
	Zusammenfassung	611
	Literatur zu Kapitel 7	614
Kapitel 8 Codeerzeugung		617
8.1	Aspekte für den Entwurf eines Codegenerators	620
8.1.1	Eingabe in den Codegenerator	620
8.1.2	Das Zielprogramm	620
8.1.3	Befehlsauswahl	622
8.1.4	Registervergabe	624
8.1.5	Auswertungsreihenfolge	625

8.2	Die Zielsprache	626
8.2.1	Ein einfaches Modell der Zielmaschine	626
8.2.2	Programm- und Befehlskosten	629
	Übungen zu Abschnitt 8.2	630
8.3	Adressen im Zielcode	633
8.3.1	Statische Zuweisung	633
8.3.2	Stackzuweisung	636
8.3.3	Laufzeitadressen für Namen	639
	Übungen zu Abschnitt 8.3	640
8.4	Grundblöcke und Flussgraphen	642
8.4.1	Grundblöcke	643
8.4.2	Informationen über die nächste Verwendung	645
8.4.3	Flussgraphen	646
8.4.4	Darstellung von Flussgraphen	648
8.4.5	Schleifen	648
	Übungen zu Abschnitt 8.4	649
8.5	Optimierung von Grundblöcken	651
8.5.1	Die DAG-Darstellung von Grundblöcken	651
8.5.2	Suche nach lokalen gemeinsamen Teilausdrücken	652
8.5.3	Entfernen von totem Code	654
8.5.4	Algebraische Identitäten	654
8.5.5	Darstellung von Arrayreferenzen	656
8.5.6	Zeigerzuweisung und Prozeduraufrufe	658
8.5.7	Reassemblierung von Grundblöcken aus DAGs	658
	Übungen zu Abschnitt 8.5	660
8.6	Ein einfacher Codegenerator	662
8.6.1	Register- und Adressdeskriptoren	663
8.6.2	Der Algorithmus zur Codeerzeugung	663
8.6.3	Entwurf der Funktion getReg	667
	Übungen zu Abschnitt 8.6	669
8.7	Peephole-Optimierung	670
8.7.1	Entfernen redundanter Lade- und Speichervorgänge	670
8.7.2	Entfernen unerreichbaren Codes	671
8.7.3	Optimierung des Kontrollflusses	672

	8.7.4	Algebraische Vereinfachung und Kosten- reduzierung	673
	8.7.5	Verwenden von Maschinenidiomen	673
		Übungen zu Abschnitt 8.7	674
8.8		Registervergabe und -zuweisung	675
	8.8.1	Globale Registervergabe	675
	8.8.2	Verwendungszähler	676
	8.8.3	Registerzuweisung für äußere Schleifen	678
	8.8.4	Registervergabe durch Graphfärbung	679
		Übungen zu Abschnitt 8.8	680
8.9		Befehlsauswahl durch Baumersetzung	681
	8.9.1	Baumübersetzungsverfahren	681
	8.9.2	Codeerzeugung durch Zerlegung/Kachelung eines Eingabebaumes	683
	8.9.3	Mustererkennung durch Syntaxanalyse	687
	8.9.4	Routinen für die semantische Prüfung	689
	8.9.5	Allgemeine Mustererkennung für Bäume	689
		Übungen zu Abschnitt 8.9	691
8.10		Optimale Codeerzeugung für Ausdrücke	692
	8.10.1	Ershov-Zahlen	692
	8.10.2	Codeerzeugung aus Ausdrucksbäumen mit Kennzeichnungen	693
	8.10.3	Auswerten von Ausdrücken ohne ausreichenden Vorrat an Registern	696
		Übungen zu Abschnitt 8.10	699
8.11		Codeerzeugung mit dynamischer Programmierung	700
	8.11.1	Zusammenhängende Auswertung	700
	8.11.2	Der dynamische Programmieralgorithmus	701
		Übungen zu Abschnitt 8.11	705
		Zusammenfassung	706
		Literatur zu Kapitel 8	708
Kapitel 9 Maschineneunabhängige Optimierungen			711
9.1		Hauptmöglichkeiten der Optimierung	714
	9.1.1	Ursachen der Redundanz	714
	9.1.2	Praxisbeispiel: Quicksort	715
	9.1.3	Semantikerhaltende Transformationen	717

	9.1.4	Globale gemeinsame Teilausdrücke	718
	9.1.5	Kopiepropagation (Copy Propagation)	720
	9.1.6	Eliminieren von totem Code (Dead-Code Elimination)	721
	9.1.7	Codeverschiebung	722
	9.1.8	Induktionsvariablen und Kostenreduzierung	723
		Übungen zu Abschnitt 9.1	727
9.2		Einführung in die Datenflussanalyse	729
	9.2.1	Die Datenflussabstraktion	729
	9.2.2	Das Datenflussanalyseschema	732
	9.2.3	Datenflussschema für Grundblöcke	733
	9.2.4	Erreichende Definitionen	734
	9.2.5	Analyse lebendiger Variablen	743
	9.2.6	Verfügbare Ausdrücke	745
	9.2.7	Zusammenfassung	749
		Übungen zu Abschnitt 9.2	751
9.3		Grundlagen der Datenflussanalyse	753
	9.3.1	Halbverbände	754
	9.3.2	Transferfunktionen	760
	9.3.3	Der iterative Algorithmus für allgemeine Frameworks	762
	9.3.4	Bedeutung einer Datenflusslösung	765
		Übungen zu Abschnitt 9.3	768
9.4		Konstantenpropagation	770
	9.4.1	Datenflusswerte für das Framework der Konstantenpropagation	770
	9.4.2	Durchschnitt für das Framework der Konstantenpropagation	771
	9.4.3	Transferfunktionen für das Framework der Konstantenpropagation	772
	9.4.4	Monotonie im Framework der Konstantenpropagation	772
	9.4.5	Nichtdistributivität im Framework der Konstantenpropagation	773
	9.4.6	Interpretation der Ergebnisse	775
		Übungen zu Abschnitt 9.4	777

9.5	Eliminierung teilweiser Redundanz	778
9.5.1	Quellen der Redundanz	779
9.5.2	Lässt sich Redundanz ganz entfernen?	782
9.5.3	Das Problem der verzögerten Codeverschiebung	784
9.5.4	Vorhersehen von Ausdrücken	785
9.5.5	Algorithmus für die verzögerte Codeverschiebung	786
	Übungen zu Abschnitt 9.5	796
9.6	Schleifen in Flussgraphen	798
9.6.1	Dominatoren	798
9.6.2	Depth-First-Anordnung	802
9.6.3	Kanten in einem Depth-First-Spannbaum	804
9.6.4	Rückkanten und Reduzierbarkeit	806
9.6.5	Tiefe eines Flussgraphen	807
9.6.6	Natürliche Schleifen	808
9.6.7	Konvergenzgeschwindigkeit von iterativen Datenflussalgorithmen	810
	Übungen zu Abschnitt 9.6	814
9.7	Bereichsbasierte Analyse	817
9.7.1	Bereiche	817
9.7.2	Bereichshierarchien für reduzierbare Flussgraphen	819
9.7.3	Überblick über eine bereichsbasierte Analyse	823
9.7.4	Notwendige Annahmen über Transferfunktionen	823
9.7.5	Algorithmus für die bereichsbasierte Analyse	825
9.7.6	Umgang mit nicht reduzierbaren Flussgraphen	830
	Übungen zu Abschnitt 9.7	833
9.8	Symbolische Analyse	834
9.8.1	Affine Ausdrücke von Referenzvariablen	834
9.8.2	Formulieren von Datenflussproblemen	838
9.8.3	Bereichsbasierte symbolische Analyse	842
	Übungen zu Abschnitt 9.8	848
	Zusammenfassung	849
	Literatur zu Kapitel 9	853

Kapitel 10	Parallelität auf Befehlsebene	857
10.1	Prozessorarchitekturen	860
10.1.1	Befehlspipelines und Verzweigungsverzögerung	860
10.1.2	Ausführung in der Pipeline	861
10.1.3	Ausgabe mehrerer Befehle	861
10.2	Einschränkungen für die Codeplanung	862
10.2.1	Datenabhängigkeit	863
10.2.2	Ermitteln von Abhängigkeiten zwischen Speicherzugriffen	864
10.2.3	Kompromiss zwischen Registernutzung und Parallelität	866
10.2.4	Phasenordnung zwischen Registervergabe und Codeplanung	869
10.2.5	Steuerungsabhängigkeit	869
10.2.6	Unterstützung der spekulativen Ausführung	871
10.2.7	Ein einfaches Modell eines Computers	873
	Übungen zu Abschnitt 10.2	874
10.3	Ablaufplanung für Grundblöcke	876
10.3.1	Datenabhängigkeitsgraphen	876
10.3.2	Listenplanung von Grundblöcken	878
10.3.3	Topologische Anordnungen mit Prioritäten	879
	Übungen zu Abschnitt 10.3	881
10.4	Globale Codeplanung	882
10.4.1	Elementare Codeverschiebung	882
10.4.2	Codeverschiebung aufwärts	884
10.4.3	Codeverschiebung abwärts	885
10.4.4	Aktualisieren von Datenabhängigkeiten	887
10.4.5	Algorithmus zur globalen Ablaufplanung	888
10.4.6	Fortgeschrittene Techniken zur Codeverschiebung	891
10.4.7	Interaktion mit dynamischen Ablaufplanern	892
	Übungen zu Abschnitt 10.4	893
10.5	Softwarepipelines	894
10.5.1	Einführung	894
10.5.2	Softwarepipelines für Schleifen	897
10.5.3	Registervergabe und Codeerzeugung	899
10.5.4	Do-Across-Schleifen	900

10.5.5	Ziele und Einschränkungen von Softwarepipelines	901
10.5.6	Algorithmus für Softwarepipelines	905
10.5.7	Ablaufplanung für azyklische Datenabhängigkeitsgraphen	906
10.5.8	Ablaufplanung für zyklische Abhängigkeitsgraphen	908
10.5.9	Verbesserungen des Pipelinealgorithmus	916
10.5.10	Modulare Variablenerweiterung	917
10.5.11	Bedingte Anweisungen	920
10.5.12	Hardwareunterstützung für Softwarepipelines	921
	Übungen zu Abschnitt 10.5	921
	Zusammenfassung	924
	Literatur zu Kapitel 10	926

Kapitel 11 Optimierungen für Parallelität und Lokalität 929

11.1	Grundkonzepte	934
11.1.1	Multiprozessorarchitektur	934
11.1.2	Parallelität in Anwendungen	937
11.1.3	Parallelität auf Schleifenebene	939
11.1.4	Datenlokalität	940
11.1.5	Einführung in die Theorie affiner Transformationen	942
11.2	Die Matrizenmultiplikation als ausführliches Beispiel	947
11.2.1	Algorithmus für die Matrizenmultiplikation	947
11.2.2	Optimierungen	950
11.2.3	Cacheinterferenz	954
	Übung zu Abschnitt 11.2	954
11.3	Iterationsräume	955
11.3.1	Konstruktion von Iterationsräumen aus verschachtelten Schleifen	955
11.3.2	Ausführungsreihenfolge für verschachtelte Schleifen	958
11.3.3	Matrixformulierung von Ungleichungen	959
11.3.4	Aufnehmen symbolischer Konstanten	960
11.3.5	Steuern der Ausführungsreihenfolge	961

	11.3.6	Ändern der Achsen	966
		Übungen zu Abschnitt 11.3	967
11.4		Affine Arrayindizes	970
	11.4.1	Affiner Zugriff	970
	11.4.2	Affine und nicht affine Zugriffe in der Praxis	972
		Übung zu Abschnitt 11.4	973
11.5		Wiederverwendung von Daten	974
	11.5.1	Arten der Wiederverwendung	975
	11.5.2	Selbstwiederverwendung	976
	11.5.3	Räumliche Selbstwiederverwendung	981
	11.5.4	Gruppenwiederverwendung	982
		Übungen zu Abschnitt 11.5	985
11.6		Analyse der Arraydatenabhängigkeiten	987
	11.6.1	Definition der Datenabhängigkeit beim Arrayzugriff	988
	11.6.2	Ganzzahlige lineare Programmierung	989
	11.6.3	Der ggT-Test	990
	11.6.4	Heuristiken für die Lösung ganzzahliger linearer Programme	993
	11.6.5	Lösen allgemeiner ganzzahliger linearer Programme	997
	11.6.6	Zusammenfassung	999
		Übungen zu Abschnitt 11.6	1000
11.7		Ermitteln synchronisierungsfreier Parallelität	1003
	11.7.1	Ein einführendes Beispiel	1003
	11.7.2	Affine Raumpartitionen	1006
	11.7.3	Einschränkungen für Raumpartitionen	1007
	11.7.4	Lösen von Einschränkungen für Raum- partitionen	1011
	11.7.5	Ein einfacher Algorithmus zur Codeerzeugung	1015
	11.7.6	Eliminieren leerer Iterationen	1019
	11.7.7	Eliminieren von Tests aus den innersten Schleifen	1022
	11.7.8	Quellcode-Transformationen	1024
		Übungen zu Abschnitt 11.7	1029

11.8	Synchronisierung zwischen parallelen Schleifen	1032
11.8.1	Konstante Anzahl von Synchronisierungen	1032
11.8.2	Programmabhängigkeitsgraphen	1034
11.8.3	Hierarchische Zeit	1036
11.8.4	Der Parallelisierungsalgorithmus	1038
	Übungen zu Abschnitt 11.8	1040
11.9	Pipelines	1041
11.9.1	Was sind Pipelines?	1041
11.9.2	Sukzessive Überrelaxation als Beispiel	1043
11.9.3	Vollständig permutierbare Schleifen	1044
11.9.4	Vollständig permutierbare Schleifen in der Pipeline	1045
11.9.5	Allgemeine Theorie	1048
11.9.6	Einschränkungen für Zeitpartitionen	1049
11.9.7	Lösen von Einschränkungen für Zeitpartitionen mit dem Lemma von Farkas	1053
11.9.8	Codetransformationen	1057
11.9.9	Parallelität mit minimaler Synchronisierung	1062
	Übungen zu Abschnitt 11.9	1065
11.10	Optimierungen der Lokalität	1068
11.10.1	Zeitliche Lokalität berechneter Daten	1068
11.10.2	Arraykontraktion	1069
11.10.3	Verschachteln von Partitionen	1072
11.10.4	Die Algorithmen im Ganzen	1076
	Übungen zu Abschnitt 11.10	1078
11.11	Andere Verwendungen für affine Transformationen	1079
11.11.1	Verteilte Speichermaschinen	1079
11.11.2	Prozessoren mit mehrfacher Befehlsausgabe	1080
11.11.3	Vektor- und SIMD-Befehle	1081
11.11.4	Vorabruf	1082
	Zusammenfassung	1084
	Literatur zu Kapitel 11	1087
Kapitel 12 Interprozedurale Analyse		1091
12.1	Grundkonzepte	1093
12.1.1	Aufrufgraphen	1093
12.1.2	Kontextsensitivität	1095

	12.1.3	Aufrufstrings	109
	12.1.4	Kontextsensitive Analyse auf Klonbasis	110
	12.1.5	Kontextsensitive Analyse mit Zusammenfassungen	1101
		Übungen zu Abschnitt 12.1	1105
12.2		Gründe für die interprozedurale Analyse	1107
	12.2.1	Virtueller Methodenaufruf	1107
	12.2.2	Zeigeralias-Analyse	1107
	12.2.3	Parallelisierung	1108
	12.2.4	Ermitteln von Softwarefehlern und Schwachstellen	1108
	12.2.5	SQL-Injektion	1109
	12.2.6	Pufferüberlauf	1111
12.3		Logische Darstellung des Datenflusses	1113
	12.3.1	Einführung in Datalog	1113
	12.3.2	Regeln in Datalog	1115
	12.3.3	Intensionale und extensionale Prädikate	1116
	12.3.4	Ausführung von Datalog-Programmen	1120
	12.3.5	Inkrementelle Auswertung von Datalog-Programmen	1121
	12.3.6	Problematische Datalog-Regeln	1123
		Übungen zu Abschnitt 12.3	1125
12.4		Einfacher Algorithmus zur Zeigeranalyse	1127
	12.4.1	Schwierigkeiten der Zeigeranalyse	1127
	12.4.2	Modell für Zeiger und Referenzen	1128
	12.4.3	Flussinsensitivität	1130
	12.4.4	Formulierung in Datalog	1131
	12.4.5	Verwenden von Typinformationen	1133
		Übungen zu Abschnitt 12.4	1134
12.5		Kontextinsensitive interprozedurale Analyse	1136
	12.5.1	Auswirkungen eines Methodenaufrufes	1136
	12.5.2	Ermitteln von Aufrufgraphen in Datalog	1138
	12.5.3	Dynamisches Laden und Reflexion	1139
		Übungen zu Abschnitt 12.5	1140
12.6		Kontextsensitive Zeigeranalyse	1141
	12.6.1	Kontexte und Aufrufstrings	1142
	12.6.2	Hinzufügen von Kontext zu Datalog-Regeln	1145

12.6.3	Weitere Aspekte der Sensitivität	1146
	Übungen zu Abschnitt 12.6	1147
12.7	Datalog-Implementierungen durch BDDs	1148
12.7.1	Binäre Entscheidungsdiagramme	1148
12.7.2	Transformationen an BDDs	1150
12.7.3	Darstellen von Relationen in BDDs	1151
12.7.4	Relationale Operationen als BDD-Operationen	1152
	Übungen zu Abschnitt 12.7	1156
	Zusammenfassung	1157
	Literatur zu Kapitel 12	1160
Anhang		1163
A	Ein vollständiges Front-End	1164
A.1	Die Quellsprache	1165
A.2	Main	1165
A.3	Der Lexer	1166
A.4	Symboltabellen und Typen	1170
A.5	Zwischencode für Ausdrücke	1172
A.6	Sprungcode für Boole'sche Ausdrücke	1176
A.7	Zwischencode für Anweisungen	1181
A.8	Parser	1186
A.9	Erstellen des Front-Ends	1193
B	Ermittlung linear unabhängiger Lösungen	1195
C	Lexer- und Parsergeneration in Java	1198
	Übungen zu Anhang C	1220
	Literatur zu Anhang C	1221
Liste mit englischen Begriffen und deren Übersetzung		1223
Liste mit deutschen Begriffen und deren Übersetzung		1225
Index		1227
Die Autoren		1253