



Alfred V. Aho
Monica S. Lam
Ravi Sethi
Jeffrey D. Ullman

it
informatik

Compiler

Prinzipien, Techniken und Werkzeuge

2., aktualisierte Auflage

Compiler

Prinzipien, Techniken und Werkzeuge

2., aktualisierte Auflage

Compiler

Inhaltsverzeichnis

Compiler - Prinzipien, Techniken und Werkzeuge - 2.
aktualisierte Auflage

Inhaltsübersicht

Inhaltsverzeichnis

Vorwort

Zur deutschen Ausgabe

Kapitel 1 Einleitung

 1.1 Sprachprozessoren

 Übungen zu Abschnitt 1.1

 1.2 Die Struktur eines Compilers

 1.2.1 Lexikalische Analyse

 1.2.2 Syntaxanalyse

 1.2.3 Semantische Analyse

 1.2.4 Zwischencodeerzeugung

 1.2.5 Codeoptimierung

 1.2.6 Codeerzeugung

 1.2.7 Umgang mit Symboltabellen

 1.2.8 Gruppieren von Phasen in Läufe

 1.2.9 Werkzeuge zum Compilerbau

 1.3 Die Evolution der Programmiersprachen

 1.3.1 Der Weg zu höheren Programmiersprachen

 1.3.2 Einfluss auf Compiler

 Übung zu Abschnitt 1.3

 1.4 Die Wissenschaft des Compilerbaus

 1.4.1 Modellierung bei Compilerdesign und Implementierung

 1.4.2 Die Wissenschaft der Codeoptimierung

 1.5 Anwendungen der Compilertechnologie



Inhaltsverzeichnis

1.5.1 Implementierung von höheren Programmiersprachen

1.5.2 Optimierungen für Computerarchitekturen

1.5.3 Entwurf neuer Computerarchitekturen

1.5.4 Programmübersetzung

1.5.5 Werkzeuge zur Produktivitätssteigerung

1.6 Grundlagen von Programmiersprachen

1.6.1 Unterscheidung zwischen statisch und dynamisch

1.6.2 Umgebungen und Zustände

1.6.3 Statischer Gültigkeitsbereich und Blockstruktur

1.6.4 Explizite Zugriffskontrolle

1.6.5 Dynamischer Gültigkeitsbereich

1.6.6 Mechanismen zur Parameterübergabe

1.6.7 Aliasing

Übungen zu Abschnitt 1.6

Zusammenfassung

Literatur zu Kapitel 1

Kapitel 2 Ein einfacher syntaxgerichteter Übersetzer

2.1 Einführung

2.2 Syntaxdefinition

2.2.1 Grammatikdefinition

2.2.2 Ableitungen

2.2.3 Parse-Bäume

2.2.4 Mehrdeutigkeit

2.2.5 Assoziativität von Operatoren

2.2.6 Operatorenpräzedenz

Übungen zu Abschnitt 2.2

2.3 Syntaxgerichtete Übersetzung

2.3.1 Postfixnotation

2.3.2 Synthetisierte Attribute

2.3.3 Einfache syntaxgerichtete Definitionen

2.3.4 Durchlaufen von Bäumen

2.3.5 Übersetzungsverfahren

Übungen zu Abschnitt 2.3

2.4 Syntaxanalyse (Parsing)



Inhaltsverzeichnis

- 2.4.1 Top-Down-Syntaxanalyse
- 2.4.2 Prädiktive Syntaxanalyse
- 2.4.3 Verwendungszweck von -Produktionen
- 2.4.4 Entwurf eines prädiktiven Parsers
- 2.4.5 Linksrekursion

Übung zu Abschnitt 2.4

2.5 Übersetzer für einfache Ausdrücke

- 2.5.1 Abstrakte und konkrete Syntax
- 2.5.2 Anpassen des Übersetzungsverfahrens
- 2.5.3 Prozeduren für die Nichtterminale
- 2.5.4 Vereinfachen des Übersetzers
- 2.5.5 Das vollständige Programm

2.6 Lexikalische Analyse

- 2.6.1 Entfernen von Leerzeichen und Kommentaren
- 2.6.2 Vorausschauendes Lesen
- 2.6.3 Konstanten
- 2.6.4 Erkennen von Schlüsselwörtern und Bezeichnern
- 2.6.5 Ein lexikalischer Scanner (kurz Lexer)

Übungen zu Abschnitt 2.6

2.7 Symboltabellen

- 2.7.1 Symboltabellen nach Gültigkeitsbereich
- 2.7.2 Verwendung von Symboltabellen

2.8 Zwischencodeerzeugung

- 2.8.1 Zwei Arten der Zwischendarstellung
- 2.8.2 Konstruktion von Syntaxbäumen
- 2.8.3 Statische Überprüfung
- 2.8.4 Drei-Adress-Code

Übungen zu Abschnitt 2.8

Zusammenfassung

Kapitel 3 Lexikalische Analyse

3.1 Die Rolle des Lexers

- 3.1.1 Lexikalische Analyse und Syntaxanalyse im Vergleich
- 3.1.2 Token,Muster und Lexeme
- 3.1.3 Attribute für Token

Inhaltsverzeichnis

3.1.4 Lexikalische Fehler

Übungen zu Abschnitt 3.1

3.2 Eingabepuffer

3.2.1 Pufferpaare

3.2.2 Wächter

3.3 Spezifikation von Token

3.3.1 Strings und Sprachen

3.3.2 Operationen an Sprachen

3.3.3 Reguläre Ausdrücke

3.3.4 Reguläre Definitionen

3.3.5 Erweiterungen regulärer Ausdrücke

Übungen zu Abschnitt 3.3

3.4 Tokenerkennung

3.4.1 Übergangsdiagramme

3.4.2 Erkennen von reservierten Wörtern und Bezeichnern

3.4.3 Abschluss des Beispiels

3.4.4 Architektur eines Lexers auf der Grundlage von Übergangsdiagrammen

Übungen zu Abschnitt 3.4

3.5 Der Generator Lex für lexikalische Scanner

3.5.1 Verwendung von Lex

3.5.2 Struktur von Lex-Programmen

3.5.3 Konfliktlösung in Lex

3.5.4 Der Lookahead-Operator

3.6 Endliche Automaten

3.6.1 Nichtdeterministische endliche Automaten

3.6.2 Übergangstabellen

3.6.3 Akzeptieren von Eingabestrings durch Automaten

3.6.4 Deterministische endliche Automaten

Übungen zu Abschnitt 3.6

3.7 Von regulären Ausdrücken zu Automaten

3.7.1 Umwandlung eines NFA in einen DFA

3.7.2 Simulation eines NFA

3.7.3 Effizienz der NFA-Simulation

3.7.4 Aufbau eines NFA aus einem regulären Ausdruck



Inhaltsverzeichnis

3.7.5 Effizienz von stringverarbeitenden Algorithmen

Übungen zu Abschnitt 3.7

3.8 Entwurf eines Generators für lexikalische Scanner

3.8.1 Die Struktur des generierten Scanners

3.8.2 Pattern Matching auf der Grundlage von NFAs

3.8.3 DFAs für lexikalische Scanner

3.8.4 Implementieren des Lookahead-Operators

Übungen zu Abschnitt 3.8

3.9 Optimierung des Pattern Matching auf DFA-Grundlage

3.9.1 Wichtige Zustände eines NFA

3.9.2 Aus dem Syntaxbaum berechnete Funktionen

3.9.3 Berechnen von nullable, firstpos und lastpos

3.9.4 Berechnen von followpos

3.9.5 Direkte Konvertierung eines regulären Ausdrucks in einen DFA

3.9.6 Minimierung der Anzahl von Zuständen eines DFA

3.9.7 Zustandsminimierung in lexikalischen Scannern

3.9.8 Kompromisse zwischen Raum und Zeit bei der DFA-Simulation

Zusammenfassung

Übungen zu Abschnitt 3.9

Literatur zu Kapitel 3

Kapitel 4 Syntaktische Analyse

4.1 Einführung

4.1.1 Die Rolle des Parsers

4.1.2 Repräsentative Grammatiken

4.1.3 Behandlung von Syntaxfehlern

4.1.4 Strategien für die Fehlerbehebung

4.2 Kontextfreie Grammatiken

4.2.1 Formale Definition einer kontextfreien Grammatik

4.2.2 Konventionen für die Notation

4.2.3 Ableitungen

4.2.4 Parse-Bäume und Ableitungen

4.2.5 Mehrdeutigkeit

4.2.6 Verifizieren der von einer Grammatik generierten Sprache

4.2.7 Kontextfreie Grammatiken und reguläre Ausdrücke im Vergleich



Inhaltsverzeichnis

Übungen zu Abschnitt 4.2

4.3 Schreiben einer Grammatik

- 4.3.1 Lexikalische und syntaktische Analyse
- 4.3.2 Eliminieren von Mehrdeutigkeiten
- 4.3.3 Eliminieren der Linksrekursion
- 4.3.4 Linksfaktorisierung
- 4.3.5 Nicht kontextfreie Sprachkonstrukte

Übungen zu Abschnitt 4.3

4.4 Top-Down-Parsing

- 4.4.1 Rekursiv absteigendes Parsing
- 4.4.2 FIRST und FOLLOW
- 4.4.3 LL(1)-Grammatiken
- 4.4.4 Nichtrekursive prädiktive Syntaxanalyse
- 4.4.5 Fehlerbehebung bei der prädiktiven Syntaxanalyse

Übungen zu Abschnitt 4.4

4.5 Bottom-Up-Parsing

- 4.5.1 Reduktionen
- 4.5.2 Handle-Stützung
- 4.5.3 Shift-Reduce-Syntaxanalyse
- 4.5.4 Konflikte bei der Shift-Reduce-Syntaxanalyse

Übungen zu Abschnitt 4.5

4.6 Einführung in die LR-Syntaxanalyse: einfaches LR

- 4.6.1 Warum LR-Parser?
- 4.6.2 Items und der LR(0)-Automat
- 4.6.3 Der LR-Parsealgorithmus
- 4.6.4 Aufbau von SLR-Parsertabellen
- 4.6.5 Sinnvolle Präfixe

Übungen zu Abschnitt 4.6

4.7 Leistungsfähigere LR-Parser

- 4.7.1 Kanonische LR(1)-Items
- 4.7.2 Aufbau von LR(1)-Item-Mengen
- 4.7.3 Kanonische LR(1)-Parsertabellen
- 4.7.4 Aufbau von LALR-Parsertabellen
- 4.7.5 Effizienter Aufbau von LALR-Parsertabellen



Inhaltsverzeichnis

4.7.6 Komprimierung von LR-Parsertabellen

Übungen zu Abschnitt 4.7

4.8 Mehrdeutige Grammatiken

4.8.1 Präzedenz und Assoziativität zur Konfliktlösung

4.8.2 Mehrdeutigkeit durch ein hängendes else

4.8.3 Fehlerbehebung beim LR-Parsing

Übungen zu Abschnitt 4.8

4.9 Parsergeneratoren

4.9.1 Der Parсерgenerator Yacc

4.9.2 Einsatz von Yacc bei mehrdeutigen Grammatiken

4.9.3 Erstellen von Yacc-Lexern mit Lex

4.9.4 Fehlerbehebung bei Yacc

Übungen zu Abschnitt 4.9

Zusammenfassung

Literatur zu Kapitel 4

Kapitel 5 Syntaxgerichtete Übersetzung

5.1 Syntaxgerichtete Definitionen

5.1.1 Erbte und synthetisierte Attribute

5.1.2 Auswerten einer syntaxgerichteten Definition an den Knoten eines
Parse-Baumes

Übungen zu Abschnitt 5.1

5.2 Auswerten einer syntaxgerichteten Definition an den Knoten eines
Parse-Baumes

5.2.1 Abhängigkeitsgraphen

5.2.2 Reihenfolge der Auswertung von Attributen

5.2.3 S-attributierte Definitionen

5.2.4 L-attributierte Definitionen

5.2.5 Semantische Regeln mit kontrollierten Nebenwirkungen

Übungen zu Abschnitt 5.2

5.3 Anwendungen der syntaxgerichteten Übersetzung

5.3.1 Aufbau von Syntaxbäumen

5.3.2 Die Struktur eines Typs

Übungen zu Abschnitt 5.3



Inhaltsverzeichnis

5.4 Verfahren zur syntaxgerichteten Übersetzung

- 5.4.1 Postfix-Übersetzungsverfahren
- 5.4.2 Parserstack-Implementierungen von syntaxgerichteten Postfix-Übersetzungen
- 5.4.3 Syntaxgerichtete Übersetzungen mit Aktionen innerhalb von Produktionen
- 5.4.4 Eliminieren der Linksrekursion aus syntaxgerichteten Übersetzungen
- 5.4.5 Syntaxgerichtete Übersetzungen für L-attributierte Definitionen

Übungen zu Abschnitt 5.4

5.5 Implementieren von L-attribuierten syntaxgerichteten Definitionen

- 5.5.1 Übersetzung bei der rekursiv absteigenden Syntaxanalyse
- 5.5.2 Codeerzeugung im laufenden Betrieb
- 5.5.3 L-attributierte syntaxgerichtete Definitionen und LL-Syntaxanalyse
- 5.5.4 Bottom-Up-Syntaxanalyse von L-attribuierten syntaxgerichteten Definitionen

Übungen zu Abschnitt 5.5

Zusammenfassung

Literatur zu Kapitel 5

Kapitel 6 Zwischencodeerzeugung

6.1 Varianten von Syntaxbäumen

- 6.1.1 Gerichtete azyklische Graphen für Ausdrücke
- 6.1.2 Die Wertenummermethode für die Konstruktion von DAGs

Übungen zu Abschnitt 6.1

6.2 Drei-Adress-Code

- 6.2.1 Adressen und Befehle
- 6.2.2 Quadrupel
- 6.2.3 Tripel
- 6.2.4 Statische Einzelzuweisungsform

Übungen zu Abschnitt 6.2

6.3 Typen und Deklarationen

- 6.3.1 Typausdrücke
- 6.3.2 Typäquivalenz
- 6.3.3 Deklarationen
- 6.3.4 Speicherlayout für lokale Namen
- 6.3.5 Sequenzen aus Deklarationen
- 6.3.6 Felder in Strukturen und Klassen



Inhaltsverzeichnis

Übungen zu Abschnitt 6.3

6.4 Übersetzung von Ausdrücken

- 6.4.1 Operationen in Ausdrücken
- 6.4.2 Inkrementelle Übersetzung
- 6.4.3 Adressieren von Arrayelementen
- 6.4.4 Übersetzung von Arrayreferenzen

Übungen zu Abschnitt 6.4 .

6.5 Typüberprüfung

- 6.5.1 Regeln für die Typüberprüfung
- 6.5.2 Typkonvertierung
- 6.5.3 Überladen von Funktionen und Operatoren
- 6.5.4 Typinferenz und polymorphe Funktionen
- 6.5.5 Ein Unifikationsalgorithmus

Übungen zu Abschnitt 6.5

6.6 Kontrollfluss

- 6.6.1 Boolesche Ausdrücke
- 6.6.2 Short-Circuit-Code
- 6.6.3 Kontrollflussanweisungen
- 6.6.4 Kontrollflussübersetzung von booleschen Ausdrücken
- 6.6.5 Vermeiden redundanter Goto-Befehle
- 6.6.6 Boolesche Werte und Sprungcode

Übungen zu Abschnitt 6.6

6.7 Backpatching

- 6.7.1 Einpass-Codeerzeugung mit Backpatching
- 6.7.2 Backpatching für boolesche Ausdrücke
- 6.7.3 Steuerungsflussanweisungen
- 6.7.4 Break-, continue- und goto-Anweisungen

Übungen zu Abschnitt 6.7

6.8 Switch-Anweisungen

- 6.8.1 Übersetzung von switch-Anweisungen
- 6.8.2 Syntaxgerichtete Übersetzung von switch-Anweisungen .

Übung zu Abschnitt 6.8

6.9 Zwischencode für Prozeduren



Inhaltsverzeichnis

Zusammenfassung

Literatur zu Kapitel 6

Kapitel 7 Laufzeitumgebungen

7.1 Speicheraufbau

 7.1.1 Statische und dynamische Speicherzuweisung

7.2 Speicherzuweisung auf dem Stack

 7.2.1 Aktivierungsbäume

 7.2.2 Aktivierungseinträge

 7.2.3 Aufrufsequenzen

 7.2.4 Daten variabler Länge auf dem Stack

Übungen zu Abschnitt 7.2

7.3 Zugriff auf nichtlokale Daten auf dem Stack

 7.3.1 Datenzugriff ohne verschachtelte Prozeduren

 7.3.2 Probleme bei verschachtelten Prozeduren

 7.3.3 Eine Sprache mit Deklarationen für verschachtelte Prozeduren

 7.3.4 Verschachtelungstiefe

 7.3.5 Zugriffslinks

 7.3.6 Bearbeiten von Zugriffslinks

 7.3.7 Zugriffslinks für Prozedurparameter

 7.3.8 Displays

Übungen zu Abschnitt 7.3

7.4 Heap-Verwaltung

 7.4.1 Der Speichermanager

 7.4.2 Die Speicherhierarchie eines Computers

 7.4.3 Lokalität in Programmen

 7.4.4 Verringern der Fragmentierung

 7.4.5 Manuelle Speicherfreigabe

Übung zu Abschnitt 7.4

7.5 Einführung in die Garbage Collection

 7.5.1 Entwurfsziele für Garbage Collectors

 7.5.2 Erreichbarkeit

 7.5.3 Garbage Collectors mit Referenzzählern

Übungen zu Abschnitt 7.5



Inhaltsverzeichnis

7.6 Einführung in die Garbage Collection mit Zeigerverfolgung

- 7.6.1 Ein einfacher Mark-and-Sweep-Collector
- 7.6.2 Grundlegende Abstraktion
- 7.6.3 Optimieren der Mark-and-Sweep-Collection
- 7.6.4 Mark-and-Compact-Collectors
- 7.6.5 Kopier-Collectors
- 7.6.6 Kostenvergleich

Übungen zu Abschnitt 7.6

7.7 Garbage Collection mit kurzen Pausen

- 7.7.1 Inkrementelle Garbage Collection
- 7.7.2 Inkrementelle Erreichbarkeitsanalyse
- 7.7.3 Grundlagen der teilweisen Garbage Collection
- 7.7.4 Garbage Collection nach Generationen
- 7.7.5 Der Zugalgorithmus

Übungen zu Abschnitt 7.7

7.8 Fortgeschrittene Themen der Garbage Collection

- 7.8.1 Parallele und gleichzeitige Garbage Collection
- 7.8.2 Teilweise Objektverschiebung
- 7.8.3 Konservative Garbage Collection für unsichere Sprachen
- 7.8.4 Schwache Referenzen

Übung zu Abschnitt 7.8

Zusammenfassung

Literatur zu Kapitel 7

Kapitel 8 Codeerzeugung

8.1 Aspekte für den Entwurf eines Codegenerators

- 8.1.1 Eingabe in den Codegenerator
- 8.1.2 Das Zielprogramm
- 8.1.3 Befehlsauswahl
- 8.1.4 Registervergabe
- 8.1.5 Auswertungsreihenfolge

8.2 Die Zielsprache

- 8.2.1 Ein einfaches Modell der Zielmaschine
- 8.2.2 Programm- und Befehlskosten

Übungen zu Abschnitt 8.2



Inhaltsverzeichnis

8.3 Adressen im Zielcode

8.3.1 Statische Zuweisung

8.3.2 Stackzuweisung

8.3.3 Laufzeitadressen für Namen

Übungen zu Abschnitt 8.3

8.4 Grundblöcke und Flussgraphen

8.4.1 Grundblöcke

8.4.2 Informationen über die nächste Verwendung

8.4.3 Flussgraphen

8.4.4 Darstellung von Flussgraphen

8.4.5 Schleifen

Übungen zu Abschnitt 8.4

8.5 Optimierung von Grundblöcken

8.5.1 Die DAG-Darstellung von Grundblöcken

8.5.2 Suche nach lokalen gemeinsamen Teilausdrücken.

8.5.3 Entfernen von totem Code

8.5.4 Algebraische Identitäten

8.5.5 Darstellung von Arrayreferenzen

8.5.6 Zeigerzuweisung und Prozeduraufrufe

8.5.7 Reassemblierung von Grundblöcken aus DAGs

Übungen zu Abschnitt 8.5

8.6 Ein einfacher Codegenerator

8.6.1 Register- und Adressdeskriptoren

8.6.2 Der Algorithmus zur Codeerzeugung

8.6.3 Entwurf der Funktion getReg

Übungen zu Abschnitt 8.6

8.7 Peephole-Optimierung

8.7.1 Entfernen redundanter Lade- und Speichervorgänge

8.7.2 Entfernen unerreichbaren Codes

8.7.3 Optimierung des Kontrollflusses

8.7.4 Algebraische Vereinfachung und Kostenreduzierung.

8.7.5 Verwenden von Maschinenidiomen

Übungen zu Abschnitt 8.7

8.8 Registervergabe und -zuweisung



Inhaltsverzeichnis

- 8.8.1 Globale Registervergabe
- 8.8.2 Verwendungszähler
- 8.8.3 Registerzuweisung für äußere Schleifen
- 8.8.4 Registervergabedurch Graphfärbung

Übungen zu Abschnitt 8.8

8.9 Befehlsauswahl durch Baumersetzung

- 8.9.1 Baumübersetzungsverfahren
- 8.9.2 Codeerzeugung durch Zerlegung/Kachelung eines Eingabebaumes
- 8.9.3 Mustererkennung durch Syntaxanalyse
- 8.9.4 Routinen für die semantische Prüfung
- 8.9.5 Allgemeine Mustererkennung für Bäume

Übungen zu Abschnitt 8.9

8.10 Optimale Codeerzeugung für Ausdrücke

- 8.10.1 Ershov-Zahlen
- 8.10.2 Codeerzeugung aus Ausdrucksbäumenmit Kennzeichnungen
- 8.10.3 Auswerten von Ausdrücken ohne ausreichenden Vorrat an Registern

Übungen zu Abschnitt 8.10

8.11 Codeerzeugungmit dynamischer Programmierung

- 8.11.1 Zusammenhängende Auswertung
- 8.11.2 Der dynamische Programmieralgorithmus

Übungen zu Abschnitt 8.11

Zusammenfassung

Literatur zu Kapitel 8 .

Kapitel 9 Maschinunabhängige Optimierungen

9.1 Hauptmöglichkeiten der Optimierung

- 9.1.1 Ursachen der Redundanz
- 9.1.2 Praxisbeispiel:Quicksort
- 9.1.3 Semantikerhaltende Transformationen
- 9.1.4 Globale gemeinsame Teilausdrücke
- 9.1.5 Kopiepropagation (Copy Propagation)
- 9.1.6 Eliminieren von totem Code(Dead-Code Elimination)
- 9.1.7 Codeverschiebung
- 9.1.8 Induktionsvariablen und Kostenreduzierung

Übungen zu Abschnitt 9

Inhaltsverzeichnis

9.2 Einführung in die Datenflussanalyse

- 9.2.1 Die Datenflussabstraktion
- 9.2.2 Das Datenflussanalyseschema
- 9.2.3 Datenflussschema für Grundblöcke
- 9.2.4 Erreichende Definitionen
- 9.2.5 Analyse lebendiger Variablen
- 9.2.6 Verfügbare Ausdrücke
- 9.2.7 Zusammenfassung

Übungen zu Abschnitt 9.2

9.3 Grundlagen der Datenflussanalyse

- 9.3.1 Halbverbände
- 9.3.2 Transferfunktionen
- 9.3.3 Der iterative Algorithmus für allgemeine Frameworks
- 9.3.4 Bedeutung einer Datenflusslösung

Übungen zu Abschnitt 9.3

9.4 Konstantenpropagation

- 9.4.1 Datenflusswerte für das Framework der Konstantenpropagation
- 9.4.2 Durchschnitt für das Framework der Konstantenpropagation
- 9.4.3 Transferfunktionen für das Framework der Konstantenpropagation
- 9.4.4 Monotonie im Framework der Konstantenpropagation
- 9.4.5 Nichtdistributivität im Framework der Konstantenpropagation
- 9.4.6 Interpretation der Ergebnisse

Übungen zu Abschnitt 9.4

9.5 Eliminierung teilweiser Redundanz

- 9.5.1 Quellen der Redundanz
- 9.5.2 Lässt sich Redundanz ganz entfernen?
- 9.5.3 Das Problem der verzögerten Codeverschiebung
- 9.5.4 Vorhersehen von Ausdrücken
- 9.5.5 Algorithmus für die verzögerte Codeverschiebung

Übungen zu Abschnitt 9.5

9.6 Schleifen in Flussgraphen

- 9.6.1 Dominatoren
- 9.6.2 Depth-First-Anordnung
- 9.6.3 Kanten in einem Depth-First-Spannbaum



Inhaltsverzeichnis

- 9.6.4 Rückkanten und Reduzierbarkeit
- 9.6.5 Tiefe eines Flussgraphen
- 9.6.6 Natürliche Schleifen
- 9.6.7 Konvergenzgeschwindigkeit von iterativen Datenflussalgorithmen

Übungen zu Abschnitt 9.6

9.7 Bereichsbasierte Analyse

- 9.7.1 Bereiche
- 9.7.2 Bereichshierarchien für reduzierbare Flussgraphen
- 9.7.3 Überblick über eine bereichsbasierte Analyse
- 9.7.4 Notwendige Annahmen über Transferfunktionen
- 9.7.5 Algorithmus für die bereichsbasierte Analyse
- 9.7.6 Umgang mit nicht reduzierbaren Flussgraphen

Übungen zu Abschnitt 9.7

9.8 Symbolische Analyse

- 9.8.1 Affine Ausdrücke von Referenzvariablen
- 9.8.2 Formulieren von Datenflussproblemen
- 9.8.3 Bereichsbasierte symbolische Analyse

Übungen zu Abschnitt 9.8

Zusammenfassung

Literatur zu Kapitel 9

Kapitel 10 Parallelität auf Befehlsebene

10.1 Prozessorarchitekturen

- 10.1.1 Befehlspipelines und Verzweigungsverzögerung
- 10.1.2 Ausführung in der Pipeline
- 10.1.3 Ausgabemehrerer Befehle

10.2 Einschränkungen für die Codeplanung

- 10.2.1 Datenabhängigkeit
- 10.2.2 Ermitteln von Abhängigkeiten zwischen Speicherzugriffen
- 10.2.3 Kompromiss zwischen Registernutzung und Parallelität
- 10.2.4 Phasenanordnung zwischen Registervergabe und Codeplanung
- 10.2.5 Steuerungsabhängigkeit
- 10.2.6 Unterstützung der spekulativen Ausführung
- 10.2.7 Ein einfaches Modell eines Computers

Übungen zu Abschnitt 10.2



Inhaltsverzeichnis

10.3 Ablaufplanung für Grundblöcke

- 10.3.1 Datenabhängigkeitsgraphen
- 10.3.2 Listenplanung von Grundblöcken
- 10.3.3 Topologische Anordnungen mit Prioritäten

Übungen zu Abschnitt 10.3

10.4 Globale Codep

- 10.4.1 Elementare Codeverschiebung
- 10.4.2 Codeverschiebung aufwärts
- 10.4.3 Codeverschiebung abwärts
- 10.4.4 Aktualisieren von Datenabhängigkeiten
- 10.4.5 Algorithmus zur globalen Ablaufplanung
- 10.4.6 Fortgeschrittene Techniken zur Codeverschiebung
- 10.4.7 Interaktion mit dynamischen Ablaufplanern

Übungen zu Abschnitt 10.4

10.5 Softwarepipelines

- 10.5.1 Einführung
- 10.5.2 Softwarepipelines für Schleifen
- 10.5.3 Registervergabe und Codeerzeugung
- 10.5.4 Do-Across-Schleifen
- 10.5.5 Ziele und Einschränkungen von Softwarepipelines
- 10.5.6 Algorithmus für Softwarepipelines
- 10.5.7 Ablaufplanung für azyklische Datenabhängigkeitsgraphen
- 10.5.8 Ablaufplanung für zyklische Abhängigkeitsgraphen
- 10.5.9 Verbesserungen des Pipelinealgorithmus
- 10.5.10 Modulare Variablen erweiterung
- 10.5.11 Bedingte Anweisungen
- 10.5.12 Hardwareunterstützung für Softwarepipelines

Übungen zu Abschnitt 10.5

Zusammenfassung

Literatur zu Kapitel 10

Kapitel 11 Optimierungen für Parallelität und Lokalität

11.1 Grundkonzepte

- 11.1.1 Multiprozessorarchitektur
- 11.1.2 Parallelität in Anwendungen



Inhaltsverzeichnis

11.1.3 Parallelität auf Schleifenebene

11.1.4 Datenlokalität

11.1.5 Einführung in die Theorie affiner Transformationen

11.2 Die Matrizenmultiplikation als ausführliches Beispiel

11.2.1 Algorithmus für die Matrizenmultiplikation

11.2.2 Optimierungen

11.2.3 Cacheinterferenz

Übung zu Abschnitt 11.2

11.3 Iterationsräume

11.3.1 Konstruktion von Iterationsräumen aus verschachtelten Schleifen

11.3.2 Ausführungsreihenfolge für verschachtelte Schleifen

11.3.3 Matrixformulierung von Ungleichungen

11.3.4 Aufnehmen symbolischer Konstanten

11.3.5 Steuern der Ausführungsreihenfolge

11.3.6 Ändern der Achsen

Übungen zu Abschnitt 11.3

11.4 Affine Arrayindizes

11.4.1 Affiner Zugriff

11.4.2 Affine und nicht affine Zugriffe in der Praxis

Übung zu Abschnitt 11.4

11.5 Wiederverwendung von Daten

11.5.1 Arten der Wiederverwendung

11.5.2 Selbstwiederverwendung

11.5.3 Räumliche Selbstwiederverwendung

11.5.4 Gruppenwiederverwendung

Übungen zu Abschnitt 11.5

11.6 Analyse der Arraydatenabhängigkeiten

11.6.1 Definition der Datenabhängigkeit beim Arrayzugriff

11.6.2 Ganzzahlige lineare Programmierung

11.6.3 Der ggT-Test

11.6.4 Heuristiken für die Lösung ganzzahliger linearer Programme

11.6.5 Lösen allgemeiner ganzzahliger linearer Programme

11.6.6 Zusammenfassung

Übungen zu Abschnitt 11.6



Inhaltsverzeichnis

11.7 Ermitteln synchronisierungsfreier Parallelität

- 11.7.1 Ein einführendes Beispiel
- 11.7.2 Affine Raumpartitionen
- 11.7.3 Einschränkungen für Raumpartitionen
- 11.7.4 Lösen von Einschränkungen für Raumpartitionen
- 11.7.5 Ein einfacher Algorithmus zur Codeerzeugung
- 11.7.6 Eliminieren leerer Iterationen
- 11.7.7 Eliminieren von Tests aus den innersten Schleifen
- 11.7.8 Quellcode-Transformationen

Übungen zu Abschnitt 11.7

11.8 Synchronisierung zwischen parallelen Schleifen

- 11.8.1 Konstante Anzahl von Synchronisierungen
- 11.8.2 Programmabhängigkeitsgraphen
- 11.8.3 Hierarchische Zeit
- 11.8.4 Der Parallelisierungsalgorithmus

Übungen zu Abschnitt 11.8

11.9 Pipelines

- 11.9.1 Was sind Pipelines?
- 11.9.2 Sukzessive Überrelaxation als Beispiel
- 11.9.3 Vollständig permutierbare Schleifen
- 11.9.4 Vollständig permutierbare Schleifen in der Pipeline
- 11.9.5 Allgemeine Theorie
- 11.9.6 Einschränkungen für Zeitpartitionen
- 11.9.7 Lösen von Einschränkungen für Zeitpartitionen mit dem Lemma von Farkas
- 11.9.8 Codetransformationen
- 11.9.9 Parallelität mit minimaler Synchronisierung

Übungen zu Abschnitt 11.9 .

11.10 Optimierungen der Lokalität

- 11.10.1 Zeitliche Lokalität berechneter Daten
- 11.10.2 Arraykontraktion
- 11.10.3 Verschachteln von Partitionen
- 11.10.4 Die Algorithmen im Ganzen

Übungen zu Abschnitt 11.10

11.11 Andere Verwendungen für affine Transformationen



Inhaltsverzeichnis

- 11.11.1 Verteilte Speichermaschinen
- 11.11.2 Prozessoren mit mehrfacher Befehlsausgabe
- 11.11.3 Vektor- und SIMD-Befehle
- 11.11.4 Vorabruf

Zusammenfassung

Literatur zu Kapitel 11

Kapitel 12 Interprozedurale Analyse

12.1 Grundkonzepte

- 12.1.1 Aufrufgraphen
- 12.1.2 Kontextsensitivität
- 12.1.3 Aufrufstrings
- 12.1.4 Kontextsensitive Analyse auf Klonbasis
- 12.1.5 Kontextsensitive Analyse mit Zusammenfassungen

Übungen zu Abschnitt 12.1

12.2 Gründe für die interprozedurale Analyse

- 12.2.1 Virtueller Methodenaufruf
- 12.2.2 Zeigeralias-Analyse
- 12.2.3 Parallelisierung
- 12.2.4 Ermitteln von Softwarefehlern und Schwachstellen
- 12.2.5 SQL-Injektion
- 12.2.6 Pufferüberlauf

12.3 Logische Darstellung des Datenflusses

- 12.3.1 Einführung in Datalog
- 12.3.2 Regeln in Datalog
- 12.3.3 Intensionale und extensionale Prädikate
- 12.3.4 Ausführung von Datalog-Programmen
- 12.3.5 Inkrementelle Auswertung von Datalog-Programmen
- 12.3.6 Problematische Datalog-Regeln

Übungen zu Abschnitt 12.3

12.4 Einfacher Algorithmus zur Zeigeranalyse

- 12.4.1 Schwierigkeiten der Zeigeranalyse
- 12.4.2 Modell für Zeiger und Referenzen
- 12.4.3 Flussinsensitivität
- 12.4.4 Formulierung in Datalog



Inhaltsverzeichnis

12.4.5 Verwenden von Typinformationen

Übungen zu Abschnitt 12.4

12.5 Kontextinsensitive interprozedurale Analyse

12.5.1 Auswirkungen eines Methodenaufrufes

12.5.2 Ermitteln von Aufrufgraphen in Datalog

12.5.3 Dynamisches Laden und Reflexion

Übungen zu Abschnitt 12.5

12.6 Kontextsensitive Zeigeranalyse

12.6.1 Kontexte und Aufrufstrings

12.6.2 Hinzufügen von Kontext zu Datalog-Regeln

12.6.3 Weitere Aspekte der Sensitivität

Übungen zu Abschnitt 12.6

12.7 Datalog-Implementierungen durch BDDs

12.7.1 Binäre Entscheidungsdiagramme

12.7.2 Transformationen an BDDs

12.7.3 Darstellen von Relationen in BDDs

12.7.4 Relationale Operationen als BDD-Operationen

Übungen zu Abschnitt 12.7

Zusammenfassung

Literatur zu Kapitel 12

Anhang

A Ein vollständiges Front-End

A.1 Die Quellsprache

A.2 Main

A.3 Der Lexer

A.4 Symboltabellen und Typen

A.5 Zwischencode für Ausdrücke

A.6 Sprungcode für Boolesche Ausdrücke

A.7 Zwischencode für Anweisungen

A.8 Parser

A.9 Erstellen des Front-Ends

B Ermittlung linear unabhängiger Lösungen

C Lexer- und Parsergeneration in Java



Inhaltsverzeichnis

Übungen zu Anhang C

Literatur zu Anhang C

Liste mit englischen Begriffen und deren Übersetzung

Liste mit deutschen Begriffen und deren Übersetzung

Index

Die Autoren

Vorwort

Verwendung dieses Buches

Voraussetzungen

Übungen

Unterstützung im Web

Danksagung

Zur deutschen Ausgabe

Englische Fachbegriffe und Übersetzung

Danksagung

Empfehlungen zum Gebrauch des Buches

Verwendung mit Java

Dozent

Student

CWS zum Buch

1 Einleitung

1.1 Sprachprozessoren

Übungen zu Abschnitt 1.1

1.2 Die Struktur eines Compilers

1.2.1 Lexikalische Analyse

1.2.2 Syntaxanalyse

1.2.3 Semantische Analyse

1.2.4 Zwischencodeerzeugung

1.2.5 Codeoptimierung



Inhaltsverzeichnis

- 1.2.6 Codeerzeugung
- 1.2.7 Umgang mit Symboltabellen
- 1.2.8 Gruppieren von Phasen in Läufe
- 1.2.9 Werkzeuge zum Compilerbau

1.3 Die Evolution der Programmiersprachen

- 1.3.1 Der Weg zu höheren Programmiersprachen
- 1.3.2 Einfluss auf Compiler

Übung zu Abschnitt 1.3

1.4 Die Wissenschaft des Compilerbaus

- 1.4.1 Modellierung bei Compilerdesign und -implementierung
- 1.4.2 Die Wissenschaft der Codeoptimierung

1.5 Anwendungen der Compilertechnologie

- 1.5.1 Implementierung von höheren Programmiersprachen
- 1.5.2 Optimierungen für Computerarchitekturen
- 1.5.3 Entwurf neuer Computerarchitekturen
- 1.5.4 Programmübersetzung
- 1.5.5 Werkzeuge zur Produktivitätssteigerung

1.6 Grundlagen von Programmiersprachen

- 1.6.1 Unterscheidung zwischen statisch und dynamisch
- 1.6.2 Umgebungen und Zustände
- 1.6.3 Statischer Gültigkeitsbereich und Blockstruktur
- 1.6.4 Explizite Zugriffskontrolle
- 1.6.5 Dynamischer Gültigkeitsbereich
- 1.6.6 Mechanismen zur Parameterübergabe
- 1.6.7 Aliasing

Übungen zu Abschnitt 1.6

Zusammenfassung

Literatur zu Kapitel 1

2 Ein einfacher syntaxgerichteter Übersetzer

2.1 Einführung



Inhaltsverzeichnis

2.2 Syntaxdefinition

- 2.2.1 Grammatikdefinition
- 2.2.2 Ableitungen
- 2.2.3 Parse-Bäume
- 2.2.4 Mehrdeutigkeit
- 2.2.5 Assoziativität von Operatoren
- 2.2.6 Operatorenpräzedenz

Übungen zu Abschnitt 2.2

2.3 Syntaxgerichtete Übersetzung

- 2.3.1 Postfixnotation
- 2.3.2 Synthetisierte Attribute
- 2.3.3 Einfache syntaxgerichtete Definitionen
- 2.3.4 Durchlaufen von Bäumen
- 2.3.5 Übersetzungsverfahren

Übungen zu Abschnitt 2.3

2.4 Syntaxanalyse (Parsing)

- 2.4.1 Top-Down-Syntaxanalyse
- 2.4.2 Prädiktive Syntaxanalyse
- 2.4.3 Verwendungszweck von epsilon-Produktionen
- 2.4.4 Entwurf eines prädiktiven Parsers
- 2.4.5 Linksrekursion

Übung zu Abschnitt 2.4

2.5 Übersetzer für einfache Ausdrücke

- 2.5.1 Abstrakte und konkrete Syntax
- 2.5.2 Anpassen des Übersetzungsverfahrens
- 2.5.3 Prozeduren für die Nichtterminale
- 2.5.4 Vereinfachen des Übersetzers
- 2.5.5 Das vollständige Programm

2.6 Lexikalische Analyse

- 2.6.1 Entfernen von Leerzeichen und Kommentaren



Inhaltsverzeichnis

- 2.6.2 Vorausschauendes Lesen
- 2.6.3 Konstanten
- 2.6.4 Erkennen von Schlüsselwörtern und Bezeichnern
- 2.6.5 Ein lexikalischer Scanner (kurz Lexer)

Übungen zu Abschnitt 2.6

2.7 Symboltabellen

- 2.7.1 Symboltabellen nach Gültigkeitsbereich
- 2.7.2 Verwendung von Symboltabellen

2.8 Zwischencodeerzeugung

- 2.8.1 Zwei Arten der Zwischendarstellung
- 2.8.2 Konstruktion von Syntaxbäumen
- 2.8.3 Statische Überprüfung
- 2.8.4 Drei-Adress-Code

Übungen zu Abschnitt 2.8

Zusammenfassung

3 Lexikalische Analyse

3.1 Die Rolle des Lexers

- 3.1.1 Lexikalische Analyse und Syntaxanalyse im Vergleich
- 3.1.2 Token, Muster und Lexeme
- 3.1.3 Attribute für Token
- 3.1.4 Lexikalische Fehler

Übungen zu Abschnitt 3.1

3.2 Eingabepuffer

- 3.2.1 Pufferpaare
- 3.2.2 Wächter

3.3 Spezifikation von Token

- 3.3.1 Strings und Sprachen
- 3.3.2 Operationen an Sprachen
- 3.3.3 Reguläre Ausdrücke
- 3.3.4 Reguläre Definitionen



Inhaltsverzeichnis

3.3.5 Erweiterungen regulärer Ausdrücke

Übungen zu Abschnitt 3.3

3.4 Tokenerkennung

3.4.1 Übergangsdiagramme

3.4.2 Erkennen von reservierten Wörtern und Bezeichnern

3.4.3 Abschluss des Beispiels

3.4.4 Architektur eines Lexers auf der Grundlage von
Übergangsdiagrammen

Übungen zu Abschnitt 3.4

3.5 Der Generator Lex für lexikalische Scanner

3.5.1 Verwendung von Lex

3.5.2 Struktur von Lex-Programmen

3.5.3 Konfliktlösung in Lex

3.5.4 Der Lookahead-Operator

Übungen zu Abschnitt 3.5

3.6 Endliche Automaten

3.6.1 Nichtdeterministische endliche Automaten

3.6.2 Übergangstabellen

3.6.3 Akzeptieren von Eingabestrings durch Automaten

3.6.4 Deterministische endliche Automaten

Übungen zu Abschnitt 3.6

3.7 Von regulären Ausdrücken zu Automaten

3.7.1 Umwandlung eines NFA in einen DFA

3.7.2 Simulation eines NFA

3.7.3 Effizienz der NFA-Simulation

3.7.4 Aufbau eines NFA aus einem regulären Ausdruck

3.7.5 Effizienz von stringverarbeitenden Algorithmen

Übungen zu Abschnitt 3.7

3.8 Entwurf eines Generators für lexikalische Scanner

3.8.1 Die Struktur des generierten Scanners



Inhaltsverzeichnis

3.8.2 Pattern Matching auf der Grundlage von NFAs

3.8.3 DFAs für lexikalische Scanner

3.8.4 Implementieren des Lookahead-Operators

Übungen zu Abschnitt 3.8

3.9 Optimierung des Pattern Matching auf DFA-Grundlage

3.9.1 Wichtige Zustände eines NFA

3.9.2 Aus dem Syntaxbaum berechnete Funktionen

3.9.3 Berechnen von nullable, firstpos und lastpos

3.9.4 Berechnen von followpos

3.9.5 Direkte Konvertierung eines regulären Ausdruckes in einen DFA

3.9.6 Minimierung der Anzahl von Zuständen eines DFA

3.9.7 Zustandsminimierung in lexikalischen Scannern

3.9.8 Kompromisse zwischen Raum und Zeit bei der DFA-Simulation

Übungen zu Abschnitt 3.9

Zusammenfassung

Literatur zu Kapitel 3

4 Syntaktische Analyse

4.1 Einführung

4.1.1 Die Rolle des Parsers

4.1.2 Repräsentative Grammatiken

4.1.3 Behandlung von Syntaxfehlern

4.1.4 Strategien für die Fehlerbehebung

4.2 Kontextfreie Grammatiken

4.2.1 Formale Definition einer kontextfreien Grammatik

4.2.2 Konventionen für die Notation

4.2.3 Ableitungen

4.2.4 Parse-Bäume und Ableitungen

4.2.5 Mehrdeutigkeit

4.2.6 Verifizieren der von einer Grammatik generierten Sprache

4.2.7 Kontextfreie Grammatiken und reguläre Ausdrücke im Vergleich



Inhaltsverzeichnis

Übungen zu Abschnitt 4.2

4.3 Schreiben einer Grammatik

- 4.3.1 Lexikalische und syntaktische Analyse
- 4.3.2 Eliminieren von Mehrdeutigkeiten
- 4.3.3 Eliminieren der Linksrekursion
- 4.3.4 Linksfaktorisierung
- 4.3.5 Nicht kontextfreie Sprachkonstrukte

Übungen zu Abschnitt 4.3

4.4 Top-Down-Parsing

- 4.4.1 Rekursiv absteigendes Parsing
- 4.4.2 FIRST und FOLLOW
- 4.4.3 LL(1)-Grammatiken
- 4.4.4 Nichtrekursive prädiktive Syntaxanalyse
- 4.4.5 Fehlerbehebung bei der prädiktiven Syntaxanalyse

Übungen zu Abschnitt 4.4

4.5 Bottom-Up-Parsing

- 4.5.1 Reduktionen
- 4.5.2 Handle-Stzung
- 4.5.3 Shift-Reduce-Syntaxanalyse
- 4.5.4 Konflikte bei der Shift-Reduce-Syntaxanalyse

Übungen zu Abschnitt 4.5

4.6 Einführung in die LR-Syntaxanalyse: einfaches LR

- 4.6.1 Warum LR-Parser?
- 4.6.2 Items und der LR(0)-Automat
- 4.6.3 Der LR-Parsealgorithmus
- 4.6.4 Aufbau von SLR-Parsertabellen
- 4.6.5 Sinnvolle Präfixe

Übungen zu Abschnitt 4.6

4.7 Leistungsfähigere LR-Parser

- 4.7.1 Kanonische LR(1)-Items

Inhaltsverzeichnis

- 4.7.2 Aufbau von LR(1)-Item-Mengen
- 4.7.3 Kanonische LR(1)-Parsertabellen
- 4.7.4 Aufbau von LALR-Parsertabellen
- 4.7.5 Effizienter Aufbau von LALR-Parsertabellen
- 4.7.6 Komprimierung von LR-Parsertabellen

Übungen zu Abschnitt 4.7

4.8 Mehrdeutige Grammatiken

- 4.8.1 Präzedenz und Assoziativität zur Konfliktlösung
- 4.8.2 Mehrdeutigkeit durch ein hängendes else
- 4.8.3 Fehlerbehebung beim LR-Parsing

Übungen zu Abschnitt 4.8

4.9 Parsergeneratoren

- 4.9.1 Der Parsergenerator Yacc
- 4.9.2 Einsatz von Yacc bei mehrdeutigen Grammatiken
- 4.9.3 Erstellen von Yacc-Lexern mit Lex
- 4.9.4 Fehlerbehebung bei Yacc

Übungen zu Abschnitt 4.9

Zusammenfassung

Literatur zu Kapitel 4

5 Syntaxgerichtete Übersetzung

5.1 Syntaxgerichtete Definitionen

- 5.1.1 Ererbt und synthetisierte Attribute
- 5.1.2 Auswerten einer syntaxgerichteten Definition an den Knoten eines Parse-Baumes

Übungen zu Abschnitt 5.1

5.2 Auswertungsreihenfolge für syntaxgerichtete Definitionen

- 5.2.1 Abhängigkeitsgraphen
- 5.2.2 Reihenfolge der Auswertung von Attributen
- 5.2.3 S-attributierte Definitionen
- 5.2.4 L-attributierte Definitionen

Inhaltsverzeichnis

5.2.5 Semantische Regeln mit kontrollierten Nebenwirkungen

Übungen zu Abschnitt 5.2

5.3 Anwendungen der syntaxgerichteten Übersetzung

 5.3.1 Aufbau von Syntaxbäumen

 5.3.2 Die Struktur eines Typs

Übungen zu Abschnitt 5.3

5.4 Verfahren zur syntaxgerichteten Übersetzung

 5.4.1 Postfix-Übersetzungsverfahren

 5.4.2 Parserstack-Implementierungen von syntaxgerichteten
 Postfix-Übersetzungen

 5.4.3 Syntaxgerichtete Übersetzungen mit Aktionen innerhalb von
 Produktionen

 5.4.4 Eliminieren der Linksrekursion aus syntaxgerichteten
 Übersetzungen

 5.4.5 Syntaxgerichtete Übersetzungen für L-attributierte Definitionen

Übungen zu Abschnitt 5.4

5.5 Implementieren von L-attributierten syntaxgerichteten
 Definitionen

 5.5.1 Übersetzung bei der rekursiv absteigenden Syntaxanalyse

 5.5.2 Codeerzeugung im laufenden Betrieb

 5.5.3 L-attributierte syntaxgerichtete Definitionen und LL-Syntaxanalyse

 5.5.4 Bottom-Up-Syntaxanalyse von L-attributierten syntaxgerichteten
 Definitionen

Übungen zu Abschnitt 5.5

Zusammenfassung

Literatur zu Kapitel 5

6 Zwischencodeerzeugung

6.1 Varianten von Syntaxbäumen

 6.1.1 Gerichtete azyklische Graphen für Ausdrücke

 6.1.2 Die Wertenummermethode für die Konstruktion von DAGs



Inhaltsverzeichnis

Übungen zu Abschnitt 6.1

6.2 Drei-Adress-Code

 6.2.1 Adressen und Befehle

 6.2.2 Quadrupel

 6.2.3 Tripel

 6.2.4 Statische Einzelzuweisungsform

Übungen zu Abschnitt 6.2

6.3 Typen und Deklarationen

 6.3.1 Typausdrücke

 6.3.2 Typäquivalenz

 6.3.3 Deklarationen

 6.3.4 Speicherlayout für lokale Namen

 6.3.5 Sequenzen aus Deklarationen

 6.3.6 Felder in Strukturen und Klassen

Übungen zu Abschnitt 6.3

6.4 Übersetzung von Ausdrücken

 6.4.1 Operationen in Ausdrücken

 6.4.2 Inkrementelle Übersetzung

 6.4.3 Adressieren von Arrayelementen

 6.4.4 Übersetzung von Arrayreferenzen

Übungen zu Abschnitt 6.4

6.5 Typüberprüfung

 6.5.1 Regeln für die Typüberprüfung

 6.5.2 Typkonvertierung

 6.5.3 Überladen von Funktionen und Operatoren

 6.5.4 Typinferenz und polymorphe Funktionen

 6.5.5 Ein Unifikationsalgorithmus

Übungen zu Abschnitt 6.5

6.6 Kontrollfluss

 6.6.1 Boolesche Ausdrücke



Inhaltsverzeichnis

- 6.6.2 Short-Circuit-Code
- 6.6.3 Kontrollflussanweisungen
- 6.6.4 Kontrollflussübersetzung von booleschen Ausdrücken
- 6.6.5 Vermeiden redundanter Goto-Befehle
- 6.6.6 Boolesche Werte und Sprungcode

Übungen zu Abschnitt 6.6

6.7 Backpatching

- 6.7.1 Einpass-Codeerzeugung mit Backpatching
- 6.7.2 Backpatching für boolesche Ausdrücke
- 6.7.3 Steuerungsflussanweisungen
- 6.7.4 Break-, continue- und goto-Anweisungen

Übungen zu Abschnitt 6.7

6.8 Switch-Anweisungen

- 6.8.1 Übersetzung von switch-Anweisungen
- 6.8.2 Syntaxgerichtete Übersetzung von switch-Anweisungen

Übung zu Abschnitt 6.8

6.9 Zwischencode für Prozeduren

Zusammenfassung

Literatur zu Kapitel 6

7 Laufzeitumgebungen

7.1 Speicheraufbau

- 7.1.1 Statische und dynamische Speicherzuweisung

7.2 Speicherzuweisung auf dem Stack

- 7.2.1 Aktivierungsbäume
- 7.2.2 Aktivierungseinträge
- 7.2.3 Aufrufsequenzen
- 7.2.4 Daten variabler Länge auf dem Stack

Übungen zu Abschnitt 7.2

7.3 Zugriff auf nichtlokale Daten auf dem Stack

Inhaltsverzeichnis

- 7.3.1 Datenzugriff ohne verschachtelte Prozeduren
- 7.3.2 Probleme bei verschachtelten Prozeduren
- 7.3.3 Eine Sprache mit Deklarationen für verschachtelte Prozeduren
- 7.3.4 Verschachtelungstiefe
- 7.3.5 Zugriffslinks
- 7.3.6 Bearbeiten von Zugriffslinks
- 7.3.7 Zugriffslinks für Prozedurparameter
- 7.3.8 Displays

Übungen zu Abschnitt 7.3

7.4 Heap-Verwaltung

- 7.4.1 Der Speichermanager
- 7.4.2 Die Speicherhierarchie eines Computers
- 7.4.3 Lokalität in Programmen
- 7.4.4 Verringern der Fragmentierung
- 7.4.5 Manuelle Speicherfreigabe

Übung zu Abschnitt 7.4

7.5 Einführung in die Garbage Collection

- 7.5.1 Entwurfsziele für Garbage Collectors
- 7.5.2 Erreichbarkeit
- 7.5.3 Garbage Collectors mit Referenzzählern

Übungen zu Abschnitt 7.5

7.6 Einführung in die Garbage Collection mit Zeigerverfolgung

- 7.6.1 Ein einfacher Mark-and-Sweep-Collector
- 7.6.2 Grundlegende Abstraktion
- 7.6.3 Optimieren der Mark-and-Sweep-Collection
- 7.6.4 Mark-and-Compact-Collectors
- 7.6.5 Kopier-Collectors
- 7.6.6 Kostenvergleich

Übungen zu Abschnitt 7.6

7.7 Garbage Collection mit kurzen Pausen



Inhaltsverzeichnis

- 7.7.1 Inkrementelle Garbage Collection
- 7.7.2 Inkrementelle Erreichbarkeitsanalyse
- 7.7.3 Grundlagen der teilweisen Garbage Collection
- 7.7.4 Garbage Collection nach Generationen
- 7.7.5 Der Zugalgorithmus

Übungen zu Abschnitt 7.7

- 7.8 Fortgeschrittene Themen der Garbage Collection
 - 7.8.1 Parallelle und gleichzeitige Garbage Collection
 - 7.8.2 Teilweise Objektverschiebung
 - 7.8.3 Konservative Garbage Collection für unsichere Sprachen
 - 7.8.4 Schwache Referenzen

Übung zu Abschnitt 7.8

Zusammenfassung

Literatur zu Kapitel 7

8 Codeerzeugung

- 8.1 Aspekte für den Entwurf eines Codegenerators
 - 8.1.1 Eingabe in den Codegenerator
 - 8.1.2 Das Zielprogramm
 - 8.1.3 Befehlsauswahl
 - 8.1.4 Registervergabe
 - 8.1.5 Auswertungsreihenfolge

8.2 Die Zielsprache

- 8.2.1 Ein einfaches Modell der Zielmaschine
- 8.2.2 Programm- und Befehlskosten

Übungen zu Abschnitt 8.2

8.3 Adressen im Zielcode

- 8.3.1 Statische Zuweisung
- 8.3.2 Stackzuweisung
- 8.3.3 Laufzeitadressen für Namen

Übungen zu Abschnitt 8.3



Inhaltsverzeichnis

8.4 Grundblöcke und Flussgraphen

- 8.4.1 Grundblöcke
- 8.4.2 Informationen über die nächste Verwendung
- 8.4.3 Flussgraphen
- 8.4.4 Darstellung von Flussgraphen
- 8.4.5 Schleifen

Übungen zu Abschnitt 8.4

8.5 Optimierung von Grundblöcken

- 8.5.1 Die DAG-Darstellung von Grundblöcken
- 8.5.2 Suche nach lokalen gemeinsamen Teilausdrücken
- 8.5.3 Entfernen von totem Code
- 8.5.4 Algebraische Identitäten
- 8.5.5 Darstellung von Arrayreferenzen
- 8.5.6 Zeigerzuweisung und Prozeduraufälle
- 8.5.7 Reassemblyierung von Grundblöcken aus DAGs

Übungen zu Abschnitt 8.5

8.6 Ein einfacher Codegenerator

- 8.6.1 Register- und Adressdeskriptoren
- 8.6.2 Der Algorithmus zur Codeerzeugung
- 8.6.3 Entwurf der Funktion getReg

Übungen zu Abschnitt 8.6

8.7 Peephole-Optimierung

- 8.7.1 Entfernen redundanter Lade- und Speichervorgänge
- 8.7.2 Entfernen unerreichbaren Codes
- 8.7.3 Optimierung des Kontrollflusses
- 8.7.4 Algebraische Vereinfachung und Kostenreduzierung
- 8.7.5 Verwenden von Maschinenidiomen

Übungen zu Abschnitt 8.7

8.8 Registervergabe und -zuweisung

- 8.8.1 Globale Registervergabe



Inhaltsverzeichnis

- 8.8.2 Verwendungszähler
- 8.8.3 Registerzuweisung für äußere Schleifen
- 8.8.4 Registervergabe durch Graphfärbung

Übungen zu Abschnitt 8.8

8.9 Befehlsauswahl durch Baumersetzung

- 8.9.1 Baumübersetzungsverfahren
- 8.9.2 Codeerzeugung durch Zerlegung/Kachelung eines Eingabebaumes
- 8.9.3 Mustererkennung durch Syntaxanalyse
- 8.9.4 Routinen für die semantische Prüfung
- 8.9.5 Allgemeine Mustererkennung für Bäume

Übungen zu Abschnitt 8.9

8.10 Optimale Codeerzeugung für Ausdrücke

- 8.10.1 Ershov-Zahlen
- 8.10.2 Codeerzeugung aus Ausdrucksbäumen mit Kennzeichnungen
- 8.10.3 Auswerten von Ausdrücken ohne ausreichenden Vorrat an Registern

Übungen zu Abschnitt 8.10

8.11 Codeerzeugung mit dynamischer Programmierung

- 8.11.1 Zusammenhängende Auswertung
- 8.11.2 Der dynamische Programmieralgorithmus

Übungen zu Abschnitt 8.11

Zusammenfassung

Literatur zu Kapitel 8

9 Maschinenunabhängige Optimierungen

9.1 Hauptmöglichkeiten der Optimierung

- 9.1.1 Ursachen der Redundanz
- 9.1.2 Praxisbeispiel: Quicksort
- 9.1.3 Semantikerverhaltende Transformationen
- 9.1.4 Globale gemeinsame Teilausdrücke
- 9.1.5 Kopiepropagation (Copy Propagation)



Inhaltsverzeichnis

9.1.6 Eliminieren von totem Code (Dead-Code Elimination)

9.1.7 Codeverschiebung

9.1.8 Induktionsvariablen und Kostenreduzierung

Übungen zu Abschnitt 9.1

9.2 Einführung in die Datenflussanalyse

9.2.1 Die Datenflussabstraktion

9.2.2 Das Datenflussanalyseschema

9.2.3 Datenflussschema für Grundblöcke

9.2.4 Erreichende Definitionen

9.2.5 Analyse lebendiger Variablen

9.2.6 Verfügbare Ausdrücke

9.2.7 Zusammenfassung

Übungen zu Abschnitt 9.2

9.3 Grundlagen der Datenflussanalyse

9.3.1 Halverbände

9.3.2 Transferfunktionen

9.3.3 Der iterative Algorithmus für allgemeine Frameworks

9.3.4 Bedeutung einer Datenflusslösung

Übungen zu Abschnitt 9.3

9.4 Konstantenpropagation

9.4.1 Datenflusswerte für das Framework der Konstantenpropagation

9.4.2 Durchschnitt für das Framework der Konstantenpropagation

9.4.3 Transferfunktionen für das Framework der Konstantenpropagation

9.4.4 Monotonie im Framework der Konstantenpropagation

9.4.5 Nichtdistributivität im Framework der Konstantenpropagation

9.4.6 Interpretation der Ergebnisse

Übungen zu Abschnitt 9.4

9.5 Eliminierung teilweiser Redundanz

9.5.1 Quellen der Redundanz

9.5.2 Lässt sich Redundanz ganz entfernen?



Inhaltsverzeichnis

9.5.3 Das Problem der verzögerten Codeverschiebung

9.5.4 Vorhersehen von Ausdrücken

9.5.5 Algorithmus für die verzögerte Codeverschiebung

Übungen zu Abschnitt 9.5

9.6 Schleifen in Flussgraphen

9.6.1 Dominatoren

9.6.2 Depth-First-Anordnung

9.6.3 Kanten in einem Depth-First-Spannbaum

9.6.4 Rückkanten und Reduzierbarkeit

9.6.5 Tiefe eines Flussgraphen

9.6.6 Natürliche Schleifen

9.6.7 Konvergenzgeschwindigkeit von iterativen Datenflussalgorithmen

Übungen zu Abschnitt 9.6

9.7 Bereichsbasierte Analyse

9.7.1 Bereiche

9.7.2 Bereichshierarchien für reduzierbare Flussgraphen

9.7.3 Überblick über eine bereichsbasierte Analyse

9.7.4 Notwendige Annahmen über Transferfunktionen

9.7.5 Algorithmus für die bereichsbasierte Analyse

9.7.6 Umgang mit nicht reduzierbaren Flussgraphen

Übungen zu Abschnitt 9.7

9.8 Symbolische Analyse

9.8.1 Affine Ausdrücke von Referenzvariablen

9.8.2 Formulieren von Datenflussproblemen

9.8.3 Bereichsbasierte symbolische Analyse

Übungen zu Abschnitt 9.8

Zusammenfassung

Literatur zu Kapitel 9

10 Parallelität auf Befehlsebene



Inhaltsverzeichnis

10.1 Prozessorarchitekturen

- 10.1.1 Befehlspipelines und Verzweigungsverzögerung
- 10.1.2 Ausführung in der Pipeline
- 10.1.3 Ausgabe mehrerer Befehle

10.2 Einschränkungen für die Codeplanung

- 10.2.1 Datenabhängigkeit
- 10.2.2 Ermitteln von Abhängigkeiten zwischen Speicherzugriffen
- 10.2.3 Kompromiss zwischen Registernutzung und Parallelität
- 10.2.4 Phasenanordnung zwischen Registervergabe und Codeplanung
- 10.2.5 Steuerungsabhängigkeit
- 10.2.6 Unterstützung der spekulativen Ausführung
- 10.2.7 Ein einfaches Modell eines Computers

Übungen zu Abschnitt 10.2

10.3 Ablaufplanung für Grundblöcke

- 10.3.1 Datenabhängigkeitsgraphen
- 10.3.2 Listenplanung von Grundblöcken
- 10.3.3 Topologische Anordnungen mit Prioritäten

Übungen zu Abschnitt 10.3

10.4 Globale Codeplanung

- 10.4.1 Elementare Codeverschiebung
- 10.4.2 Codeverschiebung aufwärts
- 10.4.3 Codeverschiebung abwärts
- 10.4.4 Aktualisieren von Datenabhängigkeiten
- 10.4.5 Algorithmus zur globalen Ablaufplanung
- 10.4.6 Fortgeschrittene Techniken zur Codeverschiebung
- 10.4.7 Interaktion mit dynamischen Ablaufplanern

Übungen zu Abschnitt 10.4

10.5 Softwarepipelines

- 10.5.1 Einführung
- 10.5.2 Softwarepipelines für Schleifen



Inhaltsverzeichnis

- 10.5.3 Registervergabe und Codeerzeugung
- 10.5.4 Do-Across-Schleifen
- 10.5.5 Ziele und Einschränkungen von Softwarepipelines
- 10.5.6 Algorithmus für Softwarepipelines
- 10.5.7 Ablaufplanung für azyklische Datenabhängigkeitsgraphen
- 10.5.8 Ablaufplanung für zyklische Abhängigkeitsgraphen
- 10.5.9 Verbesserungen des Pipelinealgorithmus
- 10.5.10 Modulare Variablenweiterleitung
- 10.5.11 Bedingte Anweisungen
- 10.5.12 Hardwareunterstützung für Softwarepipelines

Übungen zu Abschnitt 10.5

Zusammenfassung

Literatur zu Kapitel 10

11 Optimierungen für Parallelität und Lokalität

11.1 Grundkonzepte

- 11.1.1 Multiprozessorarchitektur
- 11.1.2 Parallelität in Anwendungen
- 11.1.3 Parallelität auf Schleifenebene
- 11.1.4 Datenlokalität
- 11.1.5 Einführung in die Theorie affiner Transformationen

11.2 Die Matrizenmultiplikation als ausführliches Beispiel

- 11.2.1 Algorithmus für die Matrizenmultiplikation
- 11.2.2 Optimierungen
- 11.2.3 Cacheinterferenz

Übung zu Abschnitt 11.2

11.3 Iterationsräume

- 11.3.1 Konstruktion von Iterationsräumen us verschachtelten Schleifen
- 11.3.2 Ausführungsreihenfolge für verschachtelte Schleifen
- 11.3.3 Matrixformulierung von Ungleichungen
- 11.3.4 Aufnehmen symbolischer Konstanten



Inhaltsverzeichnis

11.3.5 Steuern der Ausführungsreihenfolge

11.3.6 Ändern der Achsen

Übungen zu Abschnitt 11.3

11.4 Affine Arrayindizes

11.4.1 Affiner Zugriff

11.4.2 Affine und nicht affine Zugriffe in der Praxis

11.5 Wiederverwendung von Daten

11.5.1 Arten der Wiederverwendung

11.5.2 Selbstwiederverwendung

11.5.3 Räumliche Selbstwiederverwendung

11.5.4 Gruppenwiederverwendung

Übungen zu Abschnitt 11.5

11.6 Analyse der Arraydatenabhängigkeiten

11.6.1 Definition der Datenabhängigkeit beim Arrayzugriff

11.6.2 Ganzzahlige lineare Programmierung

11.6.3 Der ggT-Test

11.6.4 Heuristiken für die Lösung ganzzahliger linearer Programme

11.6.5 Lösen allgemeiner ganzzahliger linearer Programme

11.6.6 Zusammenfassung

Übungen zu Abschnitt 11.6

11.7 Ermitteln synchronisierungsfreier Parallelität

11.7.1 Ein einführendes Beispiel

11.7.2 Affine Raumpartitionen

11.7.3 Einschränkungen für Raumpartitionen

11.7.4 Lösen von Einschränkungen für Raumpartitionen

11.7.5 Ein einfacher Algorithmus zur Codeerzeugung

11.7.6 Eliminieren leerer Iterationen

11.7.7 Eliminieren von Tests aus den innersten Schleifen

11.7.8 Quellcode-Transformationen

Übungen zu Abschnitt 11.7

11.8 Synchronisierung zwischen parallelen Schleifen

11.8.1 Konstante Anzahl von Synchronisierungen

11.8.2 Programmabhängigkeitsgraphen



Inhaltsverzeichnis

11.8.3 Hierarchische Zeit

11.8.4 Der Parallelisierungsalgorithmus

Übungen zu Abschnitt 11.8

11.9 Pipelines

11.9.1 Was sind Pipelines?

11.9.2 Sukzessive Überrelaxation als Beispiel

11.9.3 Vollständig permutierbare Schleifen

11.9.4 Vollständig permutierbare Schleifen in der Pipeline

11.9.5 Allgemeine Theorie

11.9.6 Einschränkungen für Zeitpartitionen

11.9.7 Lösen von Einschränkungen für Zeitpartitionen mit dem Lemma von Farkas

11.9.8 Codetransformationen

11.9.9 Parallelität mit minimaler Synchronisierung

Übungen zu Abschnitt 11.9

11.10 Optimierungen der Lokalität

11.10.1 Zeitliche Lokalität berechneter Daten

11.10.2 Arraykontraktion

11.10.3 Verschachteln von Partitionen

11.10.4 Die Algorithmen im Ganzen

Übungen zu Abschnitt 11.10

11.11 Andere Verwendungen für affine Transformationen

11.11.1 Verteilte Speichermaschinen

11.11.2 Prozessoren mit mehrfacher Befehlsausgabe

11.11.3 Vektor- und SIMD-Befehle

11.11.4 Vorabruf

Zusammenfassung

Literatur zu Kapitel 11

Übung zu Abschnitt 11.4

12 Interprozedurale Analyse

12.1 Grundkonzepte

12.1.1 Aufrufgraphen

12.1.2 Kontextsensitivität

12.1.3 Aufrufstrings



Inhaltsverzeichnis

12.1.4 Kontextsensitive Analyse auf Klonbasis

12.1.5 Kontextsensitive Analyse mit Zusammenfassungen

Übungen zu Abschnitt 12.1

12.2 Gründe für die interprozedurale Analyse

12.2.1 Virtueller Methodenaufruf

12.2.2 Zeigeralias-Analyse

12.2.3 Parallelisierung

12.2.4 Ermitteln von Softwarefehlern und Schwachstellen

12.2.5 SQL-Injektion

12.2.6 Pufferüberlauf

12.3 Logische Darstellung des Datenflusses

12.3.1 Einführung in Datalog

12.3.2 Regeln in Datalog

12.3.3 Intensionale und extensionale Prädikate

12.3.4 Ausführung von Datalog-Programmen

12.3.5 Inkrementelle Auswertung von Datalog-Programmen

12.3.6 Problematische Datalog-Regeln

Übungen zu Abschnitt 12.3

12.4 Einfacher Algorithmus zur Zeigeranalyse

12.4.1 Schwierigkeiten der Zeigeranalyse

12.4.2 Modell für Zeiger und Referenzen

12.4.3 Flussinsensitivität

12.4.4 Formulierung in Datalog

12.4.5 Verwenden von Typinformationen

Übungen zu Abschnitt 12.4

12.5 Kontextinsensitive interprozedurale Analyse

12.5.1 Auswirkungen eines Methodenaufrufes

12.5.2 Ermitteln von Aufrufgraphen in Datalog

12.5.3 Dynamisches Laden und Reflexion

Übungen zu Abschnitt 12.5



Inhaltsverzeichnis

12.6 Kontextsensitive Zeigeranalyse

12.6.1 Kontexte und Aufrufstrings

12.6.2 Hinzufügen von Kontext zu Datalog-Regeln

12.6.3 Weitere Aspekte der Sensitivität

Übungen zu Abschnitt 12.6

12.7 Datalog-Implementierungen durch BDDs

12.7.1 Binäre Entscheidungsdiagramme

12.7.2 Transformationen an BDDs

12.7.3 Darstellen von Relationen in BDDs

12.7.4 Relationale Operationen als BDD-Operationen

12.7.5 Verwenden von BDDs für die Zeigt-auf-Analyse

Übungen zu Abschnitt 12.7

Zusammenfassung

Literatur zu Kapitel 12

Anhang

A Ein vollständiges Front-End

A.1 Die Quellsprache

A.2 Main

A.3 Der Lexer

A.4 Symboltabellen und Typen

A.5 Zwischencode für Ausdrücke

A.6 Sprungcode für Boolesche Ausdrücke

A.7 Zwischencode für Anweisungen

A.8 Parser

A.9 Erstellen des Front-Ends

B Ermittlung linear unabhängiger Lösungen

C Lexer- und Parsergeneration in Java

Übungen zu Anhang C

Literatur zu Anhang C

Liste mit englischen Begriffen und deren Übersetzung



Inhaltsverzeichnis

Liste mit deutschen Begriffen und deren Übersetzung

Index

Die Autoren

Ins Internet: Weitere Infos zum Buch, Downloads, etc.

© Copyright



Pearson



Copyright

Daten, Texte, Design und Grafiken dieses eBooks, sowie die eventuell angebotenen eBook-Zusatzdaten sind urheberrechtlich geschützt. Dieses eBook stellen wir lediglich als **persönliche Einzelplatz-Lizenz** zur Verfügung!

Jede andere Verwendung dieses eBooks oder zugehöriger Materialien und Informationen, einschliesslich

- der Reproduktion,
- der Weitergabe,
- des Weitervertriebs,
- der Platzierung im Internet,
in Intranets, in Extranets,
- der Veränderung,
- des Weiterverkaufs
- und der Veröffentlichung

bedarf der schriftlichen Genehmigung des Verlags.

Insbesondere ist die Entfernung oder Änderung des vom Verlag vergebenen Passwortschutzes ausdrücklich untersagt!

Bei Fragen zu diesem Thema wenden Sie sich bitte an: info@pearson.de

Zusatzdaten

Möglicherweise liegt dem gedruckten Buch eine CD-ROM mit Zusatzdaten bei. Die Zurverfügungstellung dieser Daten auf unseren Websites ist eine freiwillige Leistung des Verlags. Der Rechtsweg ist ausgeschlossen.

Hinweis

Dieses und viele weitere eBooks können Sie rund um die Uhr und legal auf unserer Website



<http://www.informit.de>

herunterladen