
Monographs in Computer Science

Editors

David Gries
Fred B. Schneider

Monographs in Computer Science

Abadi and Cardelli, **A Theory of Objects**

Benosman and Kang [editors], **Panoramic Vision: Sensors, Theory, and Applications**

Bhanu, Lin, Krawiec, **Evolutionary Synthesis of Pattern Recognition Systems**

Broy and Stølen, **Specification and Development of Interactive Systems: FOCUS on Streams, Interfaces, and Refinement**

Brzozowski and Seger, **Asynchronous Circuits**

Burgin, **Super-Recursive Algorithms**

Cantone, Omodeo, and Policriti, **Set Theory for Computing: From Decision Procedures to Declarative Programming with Sets**

Castillo, Gutiérrez, and Hadi, **Expert Systems and Probabilistic Network Models**

Downey and Fellows, **Parameterized Complexity**

Feijen and van Gasteren, **On a Method of Multiprogramming**

Grune and Jacobs, **Parsing Techniques: A Practical Guide, Second Edition**

Herbert and Spärck Jones [editors], **Computer Systems: Theory, Technology, and Applications**

Leiss, **Language Equations**

Levin, Heydon, Mann, and Yu, **Software Configuration Management Using VESTA**

McIver and Morgan [editors], **Programming Methodology**

McIver and Morgan [editors], **Abstraction, Refinement and Proof for Probabilistic Systems**

Misra, **A Discipline of Multiprogramming: Programming Theory for Distributed Applications**

Nielson [editor], **ML with Concurrency**

Paton [editor], **Active Rules in Database Systems**

Poernomo, Crossley, and Wirsing, **Adapting Proof-as-Programs: The Curry-Howard Protocol**

Selig, **Geometrical Methods in Robotics**

Selig, **Geometric Fundamentals of Robotics, Second Edition**

Shasha and Zhu, **High Performance Discovery in Time Series: Techniques and Case Studies**

Tonella and Potrich, **Reverse Engineering of Object Oriented Code**

Dick Grune
Ceriël J.H. Jacobs

Parsing Techniques

A Practical Guide

Second Edition

Dick Grune and Cerie J.H. Jacobs
Faculteit Exacte Wetenschappen
Vrije Universiteit
De Boelelaan 1081
1081 HV Amsterdam
The Netherlands

Series Editors

David Gries
Department of Computer Science
Cornell University
4130 Upson Hall
Ithaca, NY 14853-7501
USA

Fred P. Schneider
Department of Computer Science
Cornell University
4130 Upson Hall
Ithaca, NY 14853-7501
USA

ISBN-13: 978-0-387-20248-8

e-ISBN-13: 978-0-387-68954-8

Library of Congress Control Number: 2007936901

©2008 Springer Science+Business Media, LLC

©1990 Ellis Horwood Ltd.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper.

(SB)

9 8 7 6 5 4 3 2 1

springer.com

Preface to the Second Edition

As is fit, this second edition arose out of our readers' demands to read about new developments and our desire to write about them. Although parsing techniques is not a fast moving field, it does move. When the first edition went to press in 1990, there was only one tentative and fairly restrictive algorithm for linear-time substring parsing. Now there are several powerful ones, covering all deterministic languages; we describe them in Chapter 12. In 1990 Theorem 8.1 from a 1961 paper by Bar-Hillel, Perles, and Shamir lay gathering dust; in the last decade it has been used to create new algorithms, and to obtain insight into existing ones. We report on this in Chapter 13.

More and more non-Chomsky systems are used, especially in linguistics. None except two-level grammars had any prominence 20 years ago; we now describe six of them in Chapter 15. Non-canonical parsers were considered oddities for a very long time; now they are among the most powerful linear-time parsers we have; see Chapter 10.

Although still not very practical, marvelous algorithms for parallel parsing have been designed that shed new light on the principles; see Chapter 14. In 1990 a generalized LL parser was deemed impossible; now we describe two in Chapter 11.

Traditionally, and unsurprisingly, parsers have been used for parsing; more recently they are also being used for code generation, data compression and logic language implementation, as shown in Section 17.5. Enough. The reader can find more developments in many places in the book and in the Annotated Bibliography in Chapter 18.

Kees van Reeuwijk has — only half in jest — called our book “a reservation for endangered parsers”. We agree — partly; it is more than that — and we make no apologies. Several algorithms in this book have very limited or just no practical value. We have included them because we feel they embody interesting ideas and offer food for thought; they might also grow and acquire practical value. But we also include many algorithms that do have practical value but are sorely underused; describing them here might raise their status in the world.

Exercises and Problems

This book is not a textbook in the school sense of the word. Few universities have a course in Parsing Techniques, and, as stated in the Preface to the First Edition, readers will have very different motivations to use this book. We have therefore included hardly any questions or tasks that exercise the material contained within this book; readers can no doubt make up such tasks for themselves. The questions posed in the problem sections at the end of each chapter usually require the reader to step outside the bounds of the covered material. The problems have been divided into three not too well-defined classes:

- not marked — probably doable in a few minutes to a couple of hours.
- marked *Project* — probably a lot of work, but almost certainly doable.
- marked *Research Project* — almost certainly a lot of work, but hopefully doable.

We make no claims as to the relevance of any of these problems; we hope that some readers will find some of them enlightening, interesting, or perhaps even useful. Ideas, hints, and partial or complete solutions to a number of the problems can be found in Chapter A.

There are also a few questions on formal language that were not answered easily in the existing literature but have some importance to parsing. These have been marked accordingly in the problem sections.

Annotated Bibliography

For the first edition, we, the authors, read and summarized all papers on parsing that we could lay our hands on. Seventeen years later, with the increase in publications and easier access thanks to the Internet, that is no longer possible, much to our chagrin. In the first edition we included all relevant summaries. Again that is not possible now, since doing so would have greatly exceeded the number of pages allotted to this book. The printed version of this second edition includes only those references to the literature and their summaries that are actually referred to in this book. The complete bibliography with summaries as far as available can be found on the web site of this book; it includes its own authors index and subject index. This setup also allows us to list without hesitation technical reports and other material of possibly low accessibility. Often references to sections from Chapter 18 refer to the Web version of those sections; attention is drawn to this by calling them “(Web)Sections”.

We do not supply URLs in this book, for two reasons: they are ephemeral and may be incorrect next year, tomorrow, or even before the book is printed; and, especially for software, better URLs may be available by the time you read this book. The best URL is a few well-chosen search terms submitted to a good Web search engine.

Even in the last ten years we have seen a number of Ph.D theses written in languages other than English, specifically German, French, Spanish and Estonian. This choice of language has the regrettable but predictable consequence that their contents have been left out of the main stream of science. This is a loss, both to the authors and to the scientific community. Whether we like it or not, English is the de facto standard language of present-day science. The time that a scientifically in-

terested gentleman of leisure could be expected to read French, German, English, Greek, Latin and a tad of Sanskrit is 150 years in the past; today, students and scientists need the room in their heads and the time in their schedules for the vastly increased amount of knowledge. Although we, the authors, can still read most (but not all) of the above languages and have done our best to represent the contents of the non-English theses adequately, this will not suffice to give them the international attention they deserve.

The Future of Parsing, aka The Crystal Ball

If there will ever be a third edition of this book, we expect it to be substantially thinner (except for the bibliography section!). The reason is that the more parsing algorithms one studies the more they seem similar, and there seems to be great opportunity for unification. Basically almost all parsing is done by top-down search with left-recursion protection; this is true even for traditional bottom-up techniques like LR(1), where the top-down search is built into the LR(1) parse tables. In this respect it is significant that Earley's method is classified as top-down by some and as bottom-up by others. The general memoizing mechanism of tabular parsing takes the exponential sting out of the search. And it seems likely that transforming the usual depth-first search into breadth-first search will yield many of the generalized deterministic algorithms; in this respect we point to Sikkel's Ph.D thesis [158]. Together this seems to cover almost all algorithms in this book, including parsing by intersection. Pure bottom-up parsers without a top-down component are rare and not very powerful.

So in the theoretical future of parsing we see considerable simplification through unification of algorithms; the role that parsing by intersection can play in this is not clear. The simplification does not seem to extend to formal languages: it is still as difficult to prove the intuitively obvious fact that all LL(1) grammars are LR(1) as it was 35 years ago.

The practical future of parsing may lie in advanced pattern recognition, in addition to its traditional tasks; the practical contributions of parsing by intersection are again not clear.

Amsterdam, Amstelveen
June 2007

Dick Grune
Ceriel J.H. Jacobs

Acknowledgments

We thank Manuel E. Bermudez, Stuart Broad, Peter Bumbulis, Salvador Cavadini, Carl Cerecke, Julia Dain, Akim Demaille, Matthew Estes, Wan Fokkink, Brian Ford, Richard Frost, Clemens Grabmayer, Robert Grimm, Karin Harbusch, Stephen Horne, Jaco Imthorn, Quinn Tyler Jackson, Adrian Johnstone, Michiel Koens, Jaroslav Král, Olivier Lecarme, Lillian Lee, Olivier Lefevre, Joop Leo, JianHua Li, Neil Mitchell, Peter Pepper, Wim Pijls, José F. Quesada, Kees van Reeuwijk, Walter L. Ruzzo, Lothar Schmitz, Sylvain Schmitz, Thomas Schoebel-Theuer, Klaas Sikkels, Michael Sperberg-McQueen, Michal Žemlička, Hans Åberg, and many others, for helpful correspondence, comments on and errata to the First Edition, and support for the Second Edition. In particular we want to thank Kees van Reeuwijk and Sylvain Schmitz for their extensive “beta reading”, which greatly helped the book — and us.

We thank the Faculteit Exacte Wetenschappen of the Vrije Universiteit for the use of their equipment.

In a wider sense, we extend our thanks to the close to 1500 authors listed in the (Web)Authors Index, who have been so kind as to invent scores of clever and elegant algorithms and techniques for us to exhibit. Every page of this book leans on them.

Preface to the First Edition

Parsing (syntactic analysis) is one of the best understood branches of computer science. Parsers are already being used extensively in a number of disciplines: in computer science (for compiler construction, database interfaces, self-describing databases, artificial intelligence), in linguistics (for text analysis, corpora analysis, machine translation, textual analysis of biblical texts), in document preparation and conversion, in typesetting chemical formulae and in chromosome recognition, to name a few; they can be used (and perhaps are) in a far larger number of disciplines. It is therefore surprising that there is no book which collects the knowledge about parsing and explains it to the non-specialist. Part of the reason may be that parsing has a name for being “difficult”. In discussing the Amsterdam Compiler Kit and in teaching compiler construction, it has, however, been our experience that seemingly difficult parsing techniques can be explained in simple terms, given the right approach. The present book is the result of these considerations.

This book does not address a strictly uniform audience. On the contrary, while writing this book, we have consistently tried to imagine giving a course on the subject to a diffuse mixture of students and faculty members of assorted faculties, sophisticated laymen, the avid readers of the science supplement of the large newspapers, etc. Such a course was never given; a diverse audience like that would be too uncoordinated to convene at regular intervals, which is why we wrote this book, to be read, studied, perused or consulted wherever or whenever desired.

Addressing such a varied audience has its own difficulties (and rewards). Although no explicit math was used, it could not be avoided that an amount of mathematical thinking should pervade this book. Technical terms pertaining to parsing have of course been explained in the book, but sometimes a term on the fringe of the subject has been used without definition. Any reader who has ever attended a lecture on a non-familiar subject knows the phenomenon. He skips the term, assumes it refers to something reasonable and hopes it will not recur too often. And then there will be passages where the reader will think we are elaborating the obvious (this paragraph may be one such place). The reader may find solace in the fact that he does not have to doodle his time away or stare out of the window until the lecturer progresses.

On the positive side, and that is the main purpose of this enterprise, we hope that by means of a book with this approach we can reach those who were dimly aware of the existence and perhaps of the usefulness of parsing but who thought it would forever be hidden behind phrases like:

Let \mathfrak{P} be a mapping $V_N \xrightarrow{\Phi} 2^{(V_N \cup V_T)^*}$ and \mathfrak{H} a homomorphism . . .

No knowledge of any particular programming language is required. The book contains two or three programs in Pascal, which serve as actualizations only and play a minor role in the explanation. What is required, though, is an understanding of algorithmic thinking, especially of recursion. Books like *Learning to program* by Howard Johnston (Prentice-Hall, 1985) or *Programming from first principles* by Richard Bor-nat (Prentice-Hall 1987) provide an adequate background (but supply more detail than required). Pascal was chosen because it is about the only programming language more or less widely available outside computer science environments.

The book features an extensive annotated bibliography. The user of the bibliography is expected to be more than casually interested in parsing and to possess already a reasonable knowledge of it, either through this book or otherwise. The bibliography as a list serves to open up the more accessible part of the literature on the subject to the reader; the annotations are in terse technical prose and we hope they will be useful as stepping stones to reading the actual articles.

On the subject of applications of parsers, this book is vague. Although we suggest a number of applications in Chapter 1, we lack the expertise to supply details. It is obvious that musical compositions possess a structure which can largely be described by a grammar and thus is amenable to parsing, but we shall have to leave it to the musicologists to implement the idea. It was less obvious to us that behaviour at corporate meetings proceeds according to a grammar, but we are told that this is so and that it is a subject of socio-psychological research.

Acknowledgements

We thank the people who helped us in writing this book. Marion de Krieger has retrieved innumerable books and copies of journal articles for us and without her effort the annotated bibliography would be much further from completeness. Ed Keizer has patiently restored peace between us and the `pic|tbl|eqn|psfig|troff` pipeline, on the many occasions when we abused, overloaded or just plainly misunderstood the latter. Leo van Moergestel has made the hardware do things for us that it would not do for the uninitiated. We also thank Erik Baalbergen, Frans Kaashoek, Erik Groeneveld, Gerco Ballintijn, Jaco Imthorn, and Egon Amada for their critical remarks and contributions. The rose at the end of Chapter 2 is by Arwen Grune. Ilana and Lily Grune typed parts of the text on various occasions.

We thank the Faculteit Wiskunde en Informatica of the Vrije Universiteit for the use of the equipment.

In a wider sense, we extend our thanks to the hundreds of authors who have been so kind as to invent scores of clever and elegant algorithms and techniques for us to exhibit. We hope we have named them all in our bibliography.

Amsterdam, Amstelveen
July 1990

Dick Grune
Ceriel J.H. Jacobs

Contents

Preface to the Second Edition	v
Preface to the First Edition	xi
1 Introduction	1
1.1 Parsing as a Craft	2
1.2 The Approach Used	2
1.3 Outline of the Contents	3
1.4 The Annotated Bibliography	4
2 Grammars as a Generating Device	5
2.1 Languages as Infinite Sets	5
2.1.1 Language	5
2.1.2 Grammars	7
2.1.3 Problems with Infinite Sets	8
2.1.4 Describing a Language through a Finite Recipe	12
2.2 Formal Grammars	14
2.2.1 The Formalism of Formal Grammars	14
2.2.2 Generating Sentences from a Formal Grammar	15
2.2.3 The Expressive Power of Formal Grammars	17
2.3 The Chomsky Hierarchy of Grammars and Languages	19
2.3.1 Type 1 Grammars	19
2.3.2 Type 2 Grammars	23
2.3.3 Type 3 Grammars	30
2.3.4 Type 4 Grammars	33
2.3.5 Conclusion	34
2.4 Actually Generating Sentences from a Grammar	34
2.4.1 The Phrase-Structure Case	34
2.4.2 The CS Case	36
2.4.3 The CF Case	36
2.5 To Shrink or Not To Shrink	38

2.6	Grammars that Produce the Empty Language	41
2.7	The Limitations of CF and FS Grammars	42
2.7.1	The $uvwx$ y Theorem	42
2.7.2	The uvw Theorem	45
2.8	CF and FS Grammars as Transition Graphs	45
2.9	Hygiene in Context-Free Grammars	47
2.9.1	Undefined Non-Terminals	48
2.9.2	Unreachable Non-Terminals	48
2.9.3	Non-Productive Rules and Non-Terminals	48
2.9.4	Loops	48
2.9.5	Cleaning up a Context-Free Grammar	49
2.10	Set Properties of Context-Free and Regular Languages	52
2.11	The Semantic Connection	54
2.11.1	Attribute Grammars	54
2.11.2	Transduction Grammars	55
2.11.3	Augmented Transition Networks	56
2.12	A Metaphorical Comparison of Grammar Types	56
2.13	Conclusion	59
3	Introduction to Parsing	61
3.1	The Parse Tree	61
3.1.1	The Size of a Parse Tree	62
3.1.2	Various Kinds of Ambiguity	63
3.1.3	Linearization of the Parse Tree	65
3.2	Two Ways to Parse a Sentence	65
3.2.1	Top-Down Parsing	66
3.2.2	Bottom-Up Parsing	67
3.2.3	Applicability	68
3.3	Non-Deterministic Automata	69
3.3.1	Constructing the NDA	70
3.3.2	Constructing the Control Mechanism	70
3.4	Recognition and Parsing for Type 0 to Type 4 Grammars	71
3.4.1	Time Requirements	71
3.4.2	Type 0 and Type 1 Grammars	72
3.4.3	Type 2 Grammars	73
3.4.4	Type 3 Grammars	75
3.4.5	Type 4 Grammars	75
3.5	An Overview of Context-Free Parsing Methods	76
3.5.1	Directionality	76
3.5.2	Search Techniques	77
3.5.3	General Directional Methods	78
3.5.4	Linear Methods	80
3.5.5	Deterministic Top-Down and Bottom-Up Methods	82
3.5.6	Non-Canonical Methods	83
3.5.7	Generalized Linear Methods	84

3.5.8	Conclusion	84
3.6	The “Strength” of a Parsing Technique	84
3.7	Representations of Parse Trees	85
3.7.1	Parse Trees in the Producer-Consumer Model	86
3.7.2	Parse Trees in the Data Structure Model	87
3.7.3	Parse Forests	87
3.7.4	Parse-Forest Grammars	91
3.8	When are we done Parsing?	93
3.9	Transitive Closure	95
3.10	The Relation between Parsing and Boolean Matrix Multiplication ..	97
3.11	Conclusion	100
4	General Non-Directional Parsing	103
4.1	Unger’s Parsing Method	104
4.1.1	Unger’s Method without ϵ -Rules or Loops	104
4.1.2	Unger’s Method with ϵ -Rules	107
4.1.3	Getting Parse-Forest Grammars from Unger Parsing	110
4.2	The CYK Parsing Method	112
4.2.1	CYK Recognition with General CF Grammars	112
4.2.2	CYK Recognition with a Grammar in Chomsky Normal Form	116
4.2.3	Transforming a CF Grammar into Chomsky Normal Form ..	119
4.2.4	The Example Revisited	122
4.2.5	CYK Parsing with Chomsky Normal Form	124
4.2.6	Undoing the Effect of the CNF Transformation	125
4.2.7	A Short Retrospective of CYK	128
4.2.8	Getting Parse-Forest Grammars from CYK Parsing	129
4.3	Tabular Parsing	129
4.3.1	Top-Down Tabular Parsing	131
4.3.2	Bottom-Up Tabular Parsing	133
4.4	Conclusion	134
5	Regular Grammars and Finite-State Automata	137
5.1	Applications of Regular Grammars	137
5.1.1	Regular Languages in CF Parsing	137
5.1.2	Systems with Finite Memory	139
5.1.3	Pattern Searching	141
5.1.4	SGML and XML Validation	141
5.2	Producing from a Regular Grammar	141
5.3	Parsing with a Regular Grammar	143
5.3.1	Replacing Sets by States	144
5.3.2	ϵ -Transitions and Non-Standard Notation	147
5.4	Manipulating Regular Grammars and Regular Expressions	148
5.4.1	Regular Grammars from Regular Expressions	149
5.4.2	Regular Expressions from Regular Grammars	151
5.5	Manipulating Regular Languages	152

5.6	Left-Regular Grammars	154
5.7	Minimizing Finite-State Automata	156
5.8	Top-Down Regular Expression Recognition	158
5.8.1	The Recognizer	158
5.8.2	Evaluation	159
5.9	Semantics in FS Systems	160
5.10	Fast Text Search Using Finite-State Automata	161
5.11	Conclusion	162
6	General Directional Top-Down Parsing	165
6.1	Imitating Leftmost Derivations	165
6.2	The Pushdown Automaton	167
6.3	Breadth-First Top-Down Parsing	171
6.3.1	An Example	173
6.3.2	A Counterexample: Left Recursion	173
6.4	Eliminating Left Recursion	175
6.5	Depth-First (Backtracking) Parsers	176
6.6	Recursive Descent	177
6.6.1	A Naive Approach	179
6.6.2	Exhaustive Backtracking Recursive Descent	183
6.6.3	Breadth-First Recursive Descent	185
6.7	Definite Clause Grammars	188
6.7.1	Prolog	188
6.7.2	The DCG Format	189
6.7.3	Getting Parse Tree Information	190
6.7.4	Running Definite Clause Grammar Programs	190
6.8	Cancellation Parsing	192
6.8.1	Cancellation Sets	192
6.8.2	The Transformation Scheme	193
6.8.3	Cancellation Parsing with ϵ -Rules	196
6.9	Conclusion	197
7	General Directional Bottom-Up Parsing	199
7.1	Parsing by Searching	201
7.1.1	Depth-First (Backtracking) Parsing	201
7.1.2	Breadth-First (On-Line) Parsing	202
7.1.3	A Combined Representation	203
7.1.4	A Slightly More Realistic Example	204
7.2	The Earley Parser	206
7.2.1	The Basic Earley Parser	206
7.2.2	The Relation between the Earley and CYK Algorithms	212
7.2.3	Handling ϵ -Rules	214
7.2.4	Exploiting Look-Ahead	219
7.2.5	Left and Right Recursion	224
7.3	Chart Parsing	226

7.3.1	Inference Rules	227
7.3.2	A Transitive Closure Algorithm	227
7.3.3	Completion	229
7.3.4	Bottom-Up (Actually Left-Corner)	229
7.3.5	The Agenda	229
7.3.6	Top-Down	231
7.3.7	Conclusion	232
7.4	Conclusion	233
8	Deterministic Top-Down Parsing	235
8.1	Replacing Search by Table Look-Up	236
8.2	LL(1) Parsing	239
8.2.1	LL(1) Parsing without ϵ -Rules	239
8.2.2	LL(1) Parsing with ϵ -Rules	242
8.2.3	LL(1) versus Strong-LL(1)	247
8.2.4	Full LL(1) Parsing	248
8.2.5	Solving LL(1) Conflicts	251
8.2.6	LL(1) and Recursive Descent	253
8.3	Increasing the Power of Deterministic LL Parsing	254
8.3.1	LL(k) Grammars	254
8.3.2	Linear-Approximate LL(k)	256
8.3.3	LL-Regular	257
8.4	Getting a Parse Tree Grammar from LL(1) Parsing	258
8.5	Extended LL(1) Grammars	259
8.6	Conclusion	260
9	Deterministic Bottom-Up Parsing	263
9.1	Simple Handle-Finding Techniques	265
9.2	Precedence Parsing	266
9.2.1	Parenthesis Generators	267
9.2.2	Constructing the Operator-Precedence Table	269
9.2.3	Precedence Functions	271
9.2.4	Further Precedence Methods	272
9.3	Bounded-Right-Context Parsing	275
9.3.1	Bounded-Context Techniques	276
9.3.2	Floyd Productions	277
9.4	LR Methods	278
9.5	LR(0)	280
9.5.1	The LR(0) Automaton	280
9.5.2	Using the LR(0) Automaton	283
9.5.3	LR(0) Conflicts	286
9.5.4	ϵ -LR(0) Parsing	287
9.5.5	Practical LR Parse Table Construction	289
9.6	LR(1)	290
9.6.1	LR(1) with ϵ -Rules	295

9.6.2	LR($k > 1$) Parsing	297
9.6.3	Some Properties of LR(k) Parsing	299
9.7	LALR(1)	300
9.7.1	Constructing the LALR(1) Parsing Tables	302
9.7.2	Identifying LALR(1) Conflicts	314
9.8	SLR(1)	314
9.9	Conflict Resolvers	315
9.10	Further Developments of LR Methods	316
9.10.1	Elimination of Unit Rules	316
9.10.2	Reducing the Stack Activity	317
9.10.3	Regular Right Part Grammars	318
9.10.4	Incremental Parsing	318
9.10.5	Incremental Parser Generation	318
9.10.6	Recursive Ascent	319
9.10.7	Regular Expressions of LR Languages	319
9.11	Getting a Parse Tree Grammar from LR Parsing	319
9.12	Left and Right Contexts of Parsing Decisions	320
9.12.1	The Left Context of a State	321
9.12.2	The Right Context of an Item	322
9.13	Exploiting the Left and Right Contexts	323
9.13.1	Discriminating-Reverse (DR) Parsing	324
9.13.2	LR-Regular	327
9.13.3	LAR(m) Parsing	333
9.14	LR(k) as an Ambiguity Test	338
9.15	Conclusion	338
10	Non-Canonical Parsers	343
10.1	Top-Down Non-Canonical Parsing	344
10.1.1	Left-Corner Parsing	344
10.1.2	Deterministic Cancellation Parsing	353
10.1.3	Partitioned LL	354
10.1.4	Discussion	357
10.2	Bottom-Up Non-Canonical Parsing	357
10.2.1	Total Precedence	358
10.2.2	NSLR(1)	359
10.2.3	LR(k, ∞)	364
10.2.4	Partitioned LR	372
10.3	General Non-Canonical Parsing	377
10.4	Conclusion	379
11	Generalized Deterministic Parsers	381
11.1	Generalized LR Parsing	382
11.1.1	The Basic GLR Parsing Algorithm	382
11.1.2	Necessary Optimizations	383
11.1.3	Hidden Left Recursion and Loops	387

11.1.4 Extensions and Improvements	390
11.2 Generalized LL Parsing	391
11.2.1 Simple Generalized LL Parsing	391
11.2.2 Generalized LL Parsing with Left-Recursion	393
11.2.3 Generalized LL Parsing with ϵ -Rules	395
11.2.4 Generalized Cancellation and LC Parsing	397
11.3 Conclusion	398
12 Substring Parsing	399
12.1 The Suffix Grammar	401
12.2 General (Non-Linear) Methods	402
12.2.1 A Non-Directional Method	403
12.2.2 A Directional Method	407
12.3 Linear-Time Methods for LL and LR Grammars	408
12.3.1 Linear-Time Suffix Parsing for LL(1) Grammars	409
12.3.2 Linear-Time Suffix Parsing for LR(1) Grammars	414
12.3.3 Tabular Methods	418
12.3.4 Discussion	421
12.4 Conclusion	421
13 Parsing as Intersection	425
13.1 The Intersection Algorithm	426
13.1.1 The Rule Sets I_{rules} , I_{rough} , and I	427
13.1.2 The Languages of I_{rules} , I_{rough} , and I	429
13.1.3 An Example: Parsing Arithmetic Expressions	430
13.2 The Parsing of FSAs	431
13.2.1 Unknown Tokens	431
13.2.2 Substring Parsing by Intersection	431
13.2.3 Filtering	435
13.3 Time and Space Requirements	436
13.4 Reducing the Intermediate Size: Earley's Algorithm on FSAs	437
13.5 Error Handling Using Intersection Parsing	439
13.6 Conclusion	441
14 Parallel Parsing	443
14.1 The Reasons for Parallel Parsing	443
14.2 Multiple Serial Parsers	444
14.3 Process-Configuration Parsers	447
14.3.1 A Parallel Bottom-up GLR Parser	448
14.3.2 Some Other Process-Configuration Parsers	452
14.4 Connectionist Parsers	453
14.4.1 Boolean Circuits	453
14.4.2 A CYK Recognizer on a Boolean Circuit	454
14.4.3 Rytter's Algorithm	460
14.5 Conclusion	470

15 Non-Chomsky Grammars and Their Parsers	473
15.1 The Unsuitability of Context-Sensitive Grammars	473
15.1.1 Understanding Context-Sensitive Grammars	474
15.1.2 Parsing with Context-Sensitive Grammars	475
15.1.3 Expressing Semantics in Context-Sensitive Grammars	475
15.1.4 Error Handling in Context-Sensitive Grammars	475
15.1.5 Alternatives	476
15.2 Two-Level Grammars	476
15.2.1 VW Grammars	477
15.2.2 Expressing Semantics in a VW Grammar	480
15.2.3 Parsing with VW Grammars	482
15.2.4 Error Handling in VW Grammars	484
15.2.5 Infinite Symbol Sets	484
15.3 Attribute and Affix Grammars	485
15.3.1 Attribute Grammars	485
15.3.2 Affix Grammars	488
15.4 Tree-Adjoining Grammars	492
15.4.1 Cross-Dependencies	492
15.4.2 Parsing with TAGs	497
15.5 Coupled Grammars	500
15.5.1 Parsing with Coupled Grammars	501
15.6 Ordered Grammars	502
15.6.1 Rule Ordering by Control Grammar	502
15.6.2 Parsing with Rule-Ordered Grammars	503
15.6.3 Marked Ordered Grammars	504
15.6.4 Parsing with Marked Ordered Grammars	505
15.7 Recognition Systems	506
15.7.1 Properties of a Recognition System	507
15.7.2 Implementing a Recognition System	509
15.7.3 Parsing with Recognition Systems	512
15.7.4 Expressing Semantics in Recognition Systems	512
15.7.5 Error Handling in Recognition Systems	513
15.8 Boolean Grammars	514
15.8.1 Expressing Context Checks in Boolean Grammars	514
15.8.2 Parsing with Boolean Grammars	516
15.8.3 \S -Calculus	516
15.9 Conclusion	517
16 Error Handling	521
16.1 Detection versus Recovery versus Correction	521
16.2 Parsing Techniques and Error Detection	523
16.2.1 Error Detection in Non-Directional Parsing Methods	523
16.2.2 Error Detection in Finite-State Automata	524
16.2.3 Error Detection in General Directional Top-Down Parsers	524
16.2.4 Error Detection in General Directional Bottom-Up Parsers	524

16.2.5	Error Detection in Deterministic Top-Down Parsers	525
16.2.6	Error Detection in Deterministic Bottom-Up Parsers	525
16.3	Recovering from Errors	526
16.4	Global Error Handling	526
16.5	Regional Error Handling	530
16.5.1	Backward/Forward Move Error Recovery	530
16.5.2	Error Recovery with Bounded-Context Grammars	532
16.6	Local Error Handling	533
16.6.1	Panic Mode	534
16.6.2	FOLLOW-Set Error Recovery	534
16.6.3	Acceptable-Sets Derived from Continuations	535
16.6.4	Insertion-Only Error Correction	537
16.6.5	Locally Least-Cost Error Recovery	539
16.7	Non-Correcting Error Recovery	540
16.7.1	Detection and Recovery	540
16.7.2	Locating the Error	541
16.8	Ad Hoc Methods	542
16.8.1	Error Productions	542
16.8.2	Empty Table Slots	543
16.8.3	Error Tokens	543
16.9	Conclusion	543
17	Practical Parser Writing and Usage	545
17.1	A Comparative Survey	545
17.1.1	Considerations	545
17.1.2	General Parsers	546
17.1.3	General Substring Parsers	547
17.1.4	Linear-Time Parsers	548
17.1.5	Linear-Time Substring Parsers	549
17.1.6	Obtaining and Using a Parser Generator	549
17.2	Parser Construction	550
17.2.1	Interpretive, Table-Based, and Compiled Parsers	550
17.2.2	Parsing Methods and Implementations	551
17.3	A Simple General Context-Free Parser	553
17.3.1	Principles of the Parser	553
17.3.2	The Program	554
17.3.3	Handling Left Recursion	559
17.3.4	Parsing in Polynomial Time	560
17.4	Programming Language Paradigms	563
17.4.1	Imperative and Object-Oriented Programming	563
17.4.2	Functional Programming	564
17.4.3	Logic Programming	567
17.5	Alternative Uses of Parsing	567
17.5.1	Data Compression	567
17.5.2	Machine Code Generation	570

17.5.3	Support of Logic Languages	573
17.6	Conclusion	573
18	Annotated Bibliography	575
18.1	Major Parsing Subjects	576
18.1.1	Unrestricted PS and CS Grammars	576
18.1.2	General Context-Free Parsing	576
18.1.3	LL Parsing	584
18.1.4	LR Parsing	585
18.1.5	Left-Corner Parsing	592
18.1.6	Precedence and Bounded-Right-Context Parsing	593
18.1.7	Finite-State Automata	596
18.1.8	General Books and Papers on Parsing	599
18.2	Advanced Parsing Subjects	601
18.2.1	Generalized Deterministic Parsing	601
18.2.2	Non-Canonical Parsing	605
18.2.3	Substring Parsing	609
18.2.4	Parsing as Intersection	611
18.2.5	Parallel Parsing Techniques	612
18.2.6	Non-Chomsky Systems	614
18.2.7	Error Handling	623
18.2.8	Incremental Parsing	629
18.3	Parsers and Applications	630
18.3.1	Parser Writing	630
18.3.2	Parser-Generating Systems	634
18.3.3	Applications	634
18.3.4	Parsing and Deduction	635
18.3.5	Parsing Issues in Natural Language Handling	636
18.4	Support Material	638
18.4.1	Formal Languages	638
18.4.2	Approximation Techniques	641
18.4.3	Transformations on Grammars	641
18.4.4	Miscellaneous Literature	642
A	Hints and Solutions to Selected Problems	645
	Author Index	651
	Subject Index	655