Techniques for Searching, Parsing, and Matching

Alberto Pettorossi

Techniques for Searching, Parsing, and Matching

Springer

Alberto Pettorossi
DICII
University of Rome “Tor Vergata”
Rome, Italy

ISBN 978-3-030-63188-8 ISBN 978-3-030-63189-5 (eBook)
https://doi.org/10.1007/978-3-030-63189-5

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

In this book we present some techniques for exploring trees and graphs. We illustrate the linear search technique and the backtracking technique, and as instances of tree exploration methods, we present various algorithms for parsing subclasses of context-free languages. They include: (i) the chop-and-expand parsers for $LL(k)$ languages, (ii) the shift-and-reduce parsers for $LR(k)$ languages and, among them, the $LR(0)$, the $SLR(1)$, the $LR(1)$, and the $LALR(1)$, and (iii) the operator-precedence parsers. We illustrate the use of the parser generators Bison and Yacc, and the lexical analyzer generator Flex.

We also illustrate some tree exploration and manipulation methods by presenting algorithms for visiting trees, evaluating boolean expressions, proving propositional formulas, and encoding trees. We consider the minimal spanning tree problem in undirected graphs and the shortest path problem in directed graphs. For the latter problem we present the solutions based on boolean matrix multiplication, semirings, and dynamic programming.

Finally, we consider the pattern-matching problem and we analyze the Knuth-Morris-Pratt algorithm. In Chapter 10 we present some parsing programs written in Prolog, and we briefly recall some decidability results concerning the $LL(k)$ languages and the $LR(k)$ languages.

This book was written for a course on Automata, Languages, and Translators, taught at the University of Rome "Tor Vergata". We assume that the reader is familiar with the basic notions of Automata Theory and Formal Languages.

Some of the algorithms we have presented are written in Java 1.5 and some others in Prolog. For the Java language the reader may refer to the Java Tutorial at http://java.sun.com/docs/books/tutorial/. All Java programs have been compiled using the Java compiler 1.8.0_25 running under Mac OS X 10.15.4 Darwin 19.4.0. For the Prolog language the reader may refer to http://lpn.swi-prolog.org/. The Prolog language incorporates a backtracking mechanism that can be used for exploring search spaces and solving parsing and matching problems.

I am grateful to Professor Leslie Valiant for teaching me some of the techniques presented in Chapter 8 while I was a student at the University of Edinburgh in 1979.

Many thanks to my colleagues of the Department of Civil Engineering and Informatics of the University of Rome "Tor Vergata" and the IASI Institute of the National Research Council of Italy. I am also grateful to all my students and co-workers, and in particular to Lorenzo Clemente, Emanuele De Angelis, Corrado Di Pietro, Fabio Fioravanti, Fulvio Forni, Fabio Lecca, Maurizio Proietti, and Valerio Senni for their support and encouragement.

My warmest thanks go also to my student Alessandro Cacciotti for building a tool for the manipulation of context-free grammars and the construction of $LR(1)$ and $LALR(1)$ parsers. That tool was very useful for checking many of the parsing examples presented in the book.

Thanks also to Mr. Ronan Nugent of Springer for his most appreciated cooperation and help.

Previous editions of this book were published by the Aracne Publishing Company, Ariccia (RM), Italy.

Roma, October 2021 Alberto Pettorossi

Contents

Preface v

Chapter 1. Preliminary Definitions on Languages and Grammars 1
1.1. Free Monoids and Languages 1
1.2. Formal Grammars 2

Chapter 2. Exploring Search Spaces 11
2.1. Exploring Linear Search Spaces 11
2.2. Backtracking Algorithms 15
2.2.1. Dispositions 15
2.2.2. Combinations 19
2.2.3. n-Queens 21
2.3. Visiting Trees While Looking for Good Nodes 24
2.3.1. Depth First Visit of Trees: Basic Version 24
2.3.2. Depth First Visit of Trees: Burstall's Version 26
2.3.3. Breadth First Visit of Trees 29

Chapter 3. Chop-and-Expand Parsers for Context-Free Languages 31
3.1. Chop-and-Expand Context-Free Parser in a Functional Language 31
3.2. Chop-and-Expand Context-Free Parser in Java 35
3.3. Chop-and-Expand Context-Free Parser in a Logic Language 46

Chapter 4. Parsers for Deterministic Context-Free Languages: $LL(k)$ Parsers 47
4.1. Introduction to $LL(k)$ Parsing 47
4.2. $LL(1)$ Parsers 50
4.3. $LL(k)$ Parsers (for $k \geq 1$) 64
4.4. Time Complexity of $LL(k)$ Parsing 80

Chapter 5. Parsers for Deterministic Context-Free Languages: $LR(k)$ Parsers 85
5.1. Introduction to $LR(k)$ Parsing 85
5.2. $LR(0)$ Parsers 87
5.2.1. Avoiding the Powerset Construction for $LR(0)$ Parsing 99
5.2.2. Remarks on the Hypotheses for $LR(k)$ Parsing 101
5.3. $SLR(1)$ Parsers 103
5.4. $LR(1)$ Parsers 108
5.4.1. Avoiding the Powerset Construction for $LR(1)$ Parsing 124
5.4.2. More Remarks on the Hypotheses for $LR(k)$ Parsing 127
5.5. $LALR(1)$ Parsers 132
5.6. Time Complexity of $LR(k)$ Parsing 144
5.7. Complexity of Parsing Subclasses of Context-Free Languages 147

5.8. Subclasses of Context-free Languages 149
5.9. Derivation of Equivalent $LR(1)$ Grammars from $LR(k)$ Grammars 158
5.10. Conventions for $LL(k)$ and $LR(k)$ Parsing 161

Chapter 6. Parsers for Operator Grammars and Parser Generators 163
6.1. Operator-Precedence Parsers 163
6.2. Use of Parser Generators 168
6.2.1. Generation of Parsers Using Bison 168
6.2.2. Generation of Lexical Analyzers Using Flex 183
6.2.3. Suggestions for Constructing Parsers 185
6.3. Summary on Parsers of Context-Free Languages 186
6.3.1. Summary on $LR(0)$ and $LR(1)$ Parsing 188

Chapter 7. Visits of Trees and Graphs and Evaluation of Expressions 193
7.1. Depth First Visit of Trees 193
7.2. Evaluator of Boolean Expressions 203
7.3. A Theorem Prover for the Propositional Calculus 214
7.4. Encoding of n-ary Trees Using Binary Trees 227
7.5. Minimal Spanning Tree of an Undirected Graph 243

Chapter 8. Path Problems in Directed Graphs 255
8.1. Matrix Multiplication Algorithms 255
8.2. Comparing Matrix Multiplication Algorithms 257
8.3. Fast Boolean Matrix Multiplication 258
8.4. IC-Semirings and Path Problems in Directed Graphs 259
8.5. Transitive Closure in Directed Graphs: the Reachability Problem 261
8.6. Reducing Transitive Closure to Boolean Matrix Multiplication 262
8.7. Transitive Closure in IC-Semirings: the Shortest Path Problem 265
8.8. Single Source Shortest Paths in Directed Graphs 267
8.9. From Nondeterministic Finite Automata to Regular Expressions 278

Chapter 9. String Matching 281
9.1. Knuth-Morris-Pratt Pattern Matching 281
9.1.1. Time Complexity Analysis of the Knuth-Morris-Pratt Algorithm 287
9.1.2. Java Implementation of the Knuth-Morris-Pratt Algorithm 289
9.2. String Matching in Prolog 291

Chapter 10. Supplementary Topics 293
10.1. Simple Prolog Programs and Parsing Sentences in Prolog 293
10.2. Decidability Results for $LL(k)$ and $LR(k)$ Grammars and Languages 296

List of Algorithms and Programs 301

Index 303

Bibliography 309