



*World Headquarters*

Jones & Bartlett  
Learning  
40 Tall Pine Drive  
Sudbury, MA 01776  
978-443-5000  
info@jblearning.com  
www.jblearning.com

Jones & Bartlett  
Learning Canada  
6339 Ormindale Way  
Mississauga, Ontario  
L5V 1J2  
Canada

Jones & Bartlett  
Learning International  
Barb House, Barb Mews  
London W6 7PA  
United Kingdom

Jones & Bartlett Learning books and products are available through most bookstores and online booksellers. To contact Jones & Bartlett Learning directly, call 800-832-0034, fax 978-443-8000, or visit our website, [www.jblearning.com](http://www.jblearning.com).

Substantial discounts on bulk quantities of Jones & Bartlett Learning publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones & Bartlett Learning via the above contact information or send an email to [specialsales@jblearning.com](mailto:specialsales@jblearning.com).

Copyright © 2012 by Jones & Bartlett Learning, LLC

All rights reserved. No part of the material protected by this copyright may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

**Production Credits**

Publisher: Cathleen Sether  
Senior Acquisitions Editor: Timothy Anderson  
Senior Editorial Assistant: Stephanie Sguigna  
Production Director: Amy Rose  
Senior Marketing Manager: Andrea DeFronzo  
Composition: Northeast Compositors, Inc.  
Title Page Design: Kristin E. Parker

6048

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

# Preface

**T**he aim of this manual is to provide assistance to instructors using my book *An Introduction to Formal Languages and Automata*, Fifth Edition. Since this text was organized on the principle of learning by problem solving, much of my advice relates to the exercises at the end of each section.

It is my contention that this abstract and often difficult subject matter can be made interesting and enjoyable to the average undergraduate student, if mathematical formalism is downplayed and problem solving is made the focus. This means that students learn the material and strengthen their mathematical skills primarily by doing problems. Now this may seem rather obvious; all textbooks contain exercises that are routinely assigned to test and improve the students' understanding, but what I have in mind goes a little deeper. I consider exercises not just a supplement to the lectures, but that to a large extent, the lectures should be a preparation for the exercises. This implies that one needs to emphasize those issues that will help the student to solve challenging problems, with the basic ideas presented as simply

## iv Preface

as possible with many illustrative examples. Lengthy proofs, unnecessary detail, or excessive mathematical rigor have no place in this approach. This is not to say that correct arguments are irrelevant, but rather that they should be made in connection with specific, concrete examples. Therefore, homework has to be tightly integrated into the lectures and each exercise should have a specific pedagogical purpose. Assignments need to be composed as carefully and thoughtfully as the lectures. This is a difficult task, but in my experience, the success of a course depends critically on how well this is done.

There are several types of exercises, each with a particular purpose and flavor. Some of them are straightforward drill exercises. Any student with a basic understanding should be able to handle them. They are not always very interesting, but they test the student's grasp of the material, uncover possible misunderstandings, and give everyone the satisfaction of being able to do something.

A second type of exercise in the manual, I call "fill-in-the-details." These are usually omitted parts of proofs or examples whose broad outlines are sketched in the text. Most of them are not overly difficult since all the non-obvious points have been spelled out. For mathematically well-trained students these exercises tend to be simple, but for those not in this category (e.g., many computer science undergraduates) they may be a little more difficult and are likely to be unpopular. They are useful primarily in sharpening mathematical reasoning and formalizing skills.

The prevalent and most satisfying type of exercise involves both an understanding of the material and an ability to carry it a step further. These exercises are a little like puzzles whose solution involves inventiveness, ranging from the fairly easy to the very challenging. Some of the more difficult ones require tricks that are not easy to discover, so an occasional hint may be in order. I have identified some of the harder problems with a star, but this classification is highly subjective and may not be shared by others. The best way to judge the difficulty of any problem is to look at the discussion of the solution.

Finally, there are some exercises that take the student beyond the scope of this course, to do some additional reading or implement a method on the computer. These are normally quite time consuming and are suitable only for extra-credit assignments. These exercises are identified by a double star.

For the actual solutions, I have done what I think is most helpful. When a problem has a simple and concise answer, I give it. But there are many cases where the solution is lengthy and uninformative. I often omit the details on these, because I think it is easier to make up one's own answer than to check someone else's. In difficult problems I outline a possible approach, giving varying degrees of detail that I see necessary for following

Preface **v**

the argument. There are also some quite general and open-ended problems where no particular answer can be given. In these instances, I simply tell you why I think that such an exercise is useful.

Peter Linz



# Contents

<b>1</b>	<b>Introduction to the Theory of Computation</b>	<b>1</b>
1.1	Mathematical Preliminaries and Notation . . . . .	1
1.2	Three Basic Concepts . . . . .	2
1.3	Some Applications . . . . .	4
<b>2</b>	<b>Finite Automata</b>	<b>5</b>
2.1	Deterministic Finite Accepters . . . . .	5
2.2	Nondeterministic Finite Accepters . . . . .	8
2.3	Equivalence of Deterministic and Nondeterministic Finite Accepters . . . . .	9
2.4	Reduction of the Number of States in Finite Automata . .	11
<b>3</b>	<b>Regular Languages and Grammars</b>	<b>11</b>
3.1	Regular Expressions . . . . .	11
3.2	Connection Between Regular Expressions and Regular Languages . . . . .	14
3.3	Regular Grammars . . . . .	16
<b>4</b>	<b>Properties of Regular Languages</b>	<b>17</b>
4.1	Closure Properties of Regular Languages . . . . .	17
4.2	Elementary Questions about Regular Languages . . . . .	21
4.3	Identifying Nonregular Languages . . . . .	22

**viii** Contents

<b>5</b>	<b>Context-Free Languages</b>	<b>25</b>
5.1	Context-Free Grammars . . . . .	25
5.2	Parsing and Ambiguity . . . . .	28
5.3	Context-Free Grammars and Programming Languages . . .	29
<b>6</b>	<b>Simplification of Context-Free Grammars and Normal Forms</b>	<b>29</b>
6.1	Methods for Transforming Grammars . . . . .	30
6.2	Two Important Normal Forms . . . . .	32
6.3	A Membership Algorithm for Context-Free Grammars . . .	33
<b>7</b>	<b>Pushdown Automata</b>	<b>33</b>
7.1	Nondeterministic Pushdown Automata . . . . .	33
7.2	Pushdown Automata and Context-Free Languages . . . .	36
7.3	Deterministic Pushdown Automata and Deterministic Context-Free Languages . . . . .	38
7.4	Grammars for Deterministic Context-Free Languages . . .	39
<b>8</b>	<b>Properties of Context-Free Languages</b>	<b>40</b>
8.1	Two Pumping Lemmas . . . . .	40
8.2	Closure Properties and Decision Algorithms for Context-Free Languages . . . . .	43
<b>9</b>	<b>Turing Machines</b>	<b>45</b>
9.1	The Standard Turing Machine . . . . .	45
9.2	Combining Turing Machines for Complicated Tasks . . . .	47
9.3	Turing’s Thesis . . . . .	47
<b>10</b>	<b>Other Models of Turing Machines</b>	<b>47</b>
10.1	Minor Variations on the Turing Machine Theme . . . . .	47
10.2	Turing Machines with More Complex Storage . . . . .	48
10.3	Nondeterministic Turing Machines . . . . .	50
10.4	A Universal Turing Machine . . . . .	50
10.5	Linear Bounded Automata . . . . .	50
<b>11</b>	<b>A Hierarchy of Formal Languages and Automata</b>	<b>51</b>
11.1	Recursive and Recursively Enumerable Languages . . . . .	51
11.2	Unrestricted Grammars . . . . .	53
11.3	Context-Sensitive Grammars and Languages . . . . .	54
11.4	The Chomsky Hierarchy . . . . .	55



<b>12 Limits of Algorithmic Computation</b>	<b>55</b>
12.1 Some Problems That Cannot Be Solved by Turing Machines . . . . .	56
12.2 Undecidable Problems for Recursively Enumerable Languages . . . . .	57
12.3 The Post Correspondence Principle . . . . .	58
12.4 Undecidable Problems for Context-Free Languages . . . . .	59
12.5 A Question of Efficiency . . . . .	59
<b>13 Other Models of Computation</b>	<b>59</b>
13.1 Recursive Functions . . . . .	59
13.2 Post Systems . . . . .	61
13.3 Rewriting Systems . . . . .	62
<b>14 An Overview of Computational Complexity</b>	<b>63</b>
14.1 Efficiency of Computation . . . . .	63
14.2 Turing Machine Models and Complexity . . . . .	63
14.3 Language Families and Complexity Classes . . . . .	64
14.4 Some NP Problems . . . . .	64
14.5 Polynomial-Time Reduction . . . . .	64
14.6 NP-Completeness and an Open Question . . . . .	64

















































































































































**64** Chapter 14

**14.3 Language Families and Complexity Classes**

- 1: Straightforward, fill-in-the-details exercise.
- 2 and 3: Here we continue the idea suggested in Exercise 1, Section 14.2. To divide input into three equal parts, scan it and produce a marker for each third symbol. The number of markers then determines the value of  $|w|/3$ .
- 4: Follows directly from Theorem 14.2.

**14.4 Some NP Problems**

- 1-4: These exercises ask students to provide some omitted detail and require close reasoning.
- 5: A simple induction will work.
- 6: Easy to find a 4-clique. To show that there is no 5-clique, use Exercise 5.
- 7: Again students are asked to provide some omitted details.
- 8: This is not hard if the concepts are well understood. For union, check if  $w \in L_1$ . This can be done in polynomial time. If necessary, check if  $w \in L_2$ .

**14.5 Polynomial-Time Reduction**

The problems in this section are not hard, but require arguments with a lot of detail.

**14.6 NP-Completeness and an Open Question**

- 1: Follows from Exercise 5. Section 14.6.
- 2: Hard for students who have not seen this before, but the answer is easy: each vertex must have an even number of impinging vertices.
- 4: A good problem for finishing the discussion.