# AN INTRODUCTION TO FORMAL LANGUAGES AND AUTOMATA

SEVENTH EDITION

PETER LINZ

SUSAN H. RODGER

JONES & BARTLETT LEARNING

*World Headquarters*
Jones & Bartlett Learning
25 Mall Road
Burlington, MA 01803
978-443-5000
info@jblearning.com
www.jblearning.com

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.

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.



**Production Credits**
Vice President, Product Management: Marisa R. Urbano
Vice President, Product Operations: Christine Emerton
Product Manager: Ned Hinman
Director, Content Management: Donna Gridley
Content Strategist: Melissa Duffy
Content Coordinator: Mark Restuccia
Director, Project Management and Content Services: Karen Scott
Manager, Project Management: Jessica deMartin
Project Manager: Roberta Sherman
Senior Digital Project Specialist: Angela Dooley
Content Services Manager: Colleen Lamy
VP, Manufacturing and Inventory Control: Therese Connell
Product Fulfillment Manager: Wendy Kilborn
Composition: Straive
Project Management: Straive
Cover Design: Briana Yates
Rights & Permissions Manager: John Rusk
Rights Specialist: James Fortney
Cover and Title Page: © VS148/Shutterstock.
Printing and Binding: Gasch Printing.

**Library of Congress Cataloging-in-Publication Data**
Names: Linz, Peter, author. | Rodger, Susan H., author.
Title: An introduction to formal languages and automata / Peter Linz, Susan H. Rodger.
Description: Seventh edition. | Burlington, Massachusetts : Jones & Bartlett Learning, [2023] | Includes bibliographical references and index.
Identifiers: LCCN 2021047487 | ISBN 9781284231601 (paperback)
Subjects: LCSH: Formal languages. | Machine theory.
Classification: LCC QA267.3 .L56 2023 | DDC 005.13/1–dc23/eng/20210927
LC record available at https://lccn.loc.gov/2021047487

6048

Printed in the United States of America

26 25 24 23 22 10 9 8 7 6 5 4 3 2 1

*To Thomas and the Memory of My Father*

*— S. Rodger*

# CONTENTS

# PREFACE

This book is designed for an introductory course on formal languages, automata, computability, and related matters. These topics form a major part of what is known as the theory of computation. A course on this subject matter is now standard in the computer science curriculum and is often taught fairly early in the program. Hence, the prospective audience for this book consists primarily of sophomores and juniors majoring in computer science or computer engineering.

Prerequisites for the material in this book are a knowledge of some higher-level programming language (commonly C, C++, Python™, or Java™) and familiarity with the fundamentals of data structures and algorithms. A course in discrete mathematics that includes set theory, functions, relations, logic, and elements of mathematical reasoning is essential. Such a course is part of the standard introductory computer science curriculum.

The study of the theory of computation has several purposes, most importantly (1) to familiarize students with the foundations and principles of computer science, (2) to teach material that is useful in subsequent courses, and (3) to strengthen students' ability to carry out formal and rigorous mathematical arguments. The presentation I have chosen for this text favors the first two purposes, although I would argue that it also serves the third. To present ideas clearly and to give students insight into the material, the text stresses intuitive motivation and illustration of ideas through examples. When there is a choice, I prefer arguments that are easily grasped to those that are concise and elegant but difficult in concept. I state definitions and theorems precisely and give the motivation for proofs but often leave out the routine and tedious details. I believe that this is desirable for pedagogical reasons. Many proofs are unexciting applications of induction or contradiction with differences that are specific to particular problems. Presenting such arguments in full detail is not only unnecessary, but it interferes with the flow of the story. Therefore, quite a few of the proofs are brief, and someone who insists on completeness may consider them lacking in detail. I do not see this as a drawback. Mathematical skills are not the by-product of reading someone else's arguments, but they come from thinking about the essence of a problem, discovering ideas suitable to make the point, then carrying them out in precise detail. The latter skill certainly has to be learned, and I think that the proof sketches in this text provide very appropriate starting points for such a practice.

Computer science students sometimes view a course in the theory of computation as unnecessarily abstract and of no practical consequence. To convince them otherwise, one needs to appeal to their specific interests and strengths, such as tenacity and inventiveness in dealing with hard-to-solve problems. Because of this, my approach emphasizes learning through problem solving.

By a problem-solving approach, I mean that students learn the material primarily through problem-type illustrative examples that show the motivation behind the concepts, as well as their connection to the theorems and definitions. At the same time, the examples may involve a nontrivial aspect, for which students must discover a solution. In such an approach, homework exercises contribute to a major part of the learning process. The exercises at the end of each section are designed to illuminate and illustrate the material and call on students' problem-solving ability at various levels. Some of the exercises are fairly simple, picking up where the discussion in the text leaves off and asking students to carry on for another step or two. Other exercises are very difficult, challenging even the best minds. A good mix of such exercises can be a very effective teaching tool. Students need not be asked to solve all problems, but should be assigned those that support the goals of the course and the viewpoint of the instructor. Computer science curricula differ from institution to institution; while a few emphasize the theoretical side, others are almost entirely oriented toward practical application. I believe that this text can serve either of these extremes, provided that the exercises are selected carefully with the students' background and interests in mind. At the same time, the instructor needs to inform the students about the level of abstraction that is expected of them. This is particularly true of the proof-oriented exercises. When I say "prove that" or "show that," I have in mind that the student should think about how a proof can be constructed and then produce a clear argument. How formal such a proof should be needs to be determined by the instructor, and students should be given guidelines on this early in the course.

The content of the text is appropriate for a one-semester course. Most of the material can be covered, although some choice of emphasis will have to be made. In my classes, I generally gloss over proofs, giving just enough coverage to make the result plausible, and then ask students to read the rest on their own. Overall, though, little can be skipped entirely without potential difficulties later on. A few sections, which are marked with an asterisk, can be omitted without loss to later material. Most of the material, however, is essential and must be covered.

Appendix B briefly introduces JFLAP, an interactive software tool available for free at www.jflap.org that is of great help in both learning and teaching the material in this book. It aids in understanding the concepts and is a great time saver in the actual construction of the solutions to the exercises. I highly recommend incorporating JFLAP into the course.

The seventh edition of this book features two additions. The first is a large number of new exercises, collected at the chapter ends, under the heading Introductory Exercises to distinguish them from the already existing Exercises. These new exercises are largely very simple and require only an understanding of the concepts. They are intended as a bridge to the often much more difficult Exercises. The instructor can decide where and for whom these new exercises can be of help. Chapters 1–14 of the sixth edition, with the new exercises, are now reorganized as Part I: Theory.

Substantial new material comes in Part II: Applications, where we discuss some of the important issues in compiler construction in the context of the material in Part I. This new material comes in response to perennial student questions such as "How does this theory apply to the mostly practical matters of computer science?" or even "Why do we have to know all this abstract stuff?" We hope we have given a satisfactory response to such questions.

How can this material be integrated into an existing course? Instructors who cannot find the time for it might use Part II as suggested reading for curious students. But the material is important and attractive to many students, so we recommend that some attempt to cover it in class be made. Part I introduces the main aspects of the theory of computation, but not everything there is essential. Not only the starred sections, but many sections in the latter part can be treated lightly or skipped altogether without loss of continuity. Doing so might free a week or so for Part II. This coverage can benefit many students. Those who never take a course on compilers will get some understanding of what is involved in compiler design; for others who do, it provides a first look at what will be covered later in more detail.

Peter Linz
Susan H. Rodger