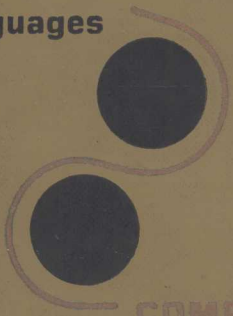
Automata Theory: Machines and Languages



AUTOMATA THEORY: MACHINES AND LANGUAGES

RICHARD Y. KAIN
Associate Professor of Electrical Engineering
University of Minnesota

McGraw-Hill Book Company

New York St. Louis San Francisco Düsseldorf Johannesburg
Kuala Lumpur London Mexico Montreal New Delhi
Panama Rio de Janeiro Singapore Sydney Toronto

AUTOMATA THEORY: MACHINES AND LANGUAGES

Library of Congress Catalog Card Number 71–168453

07–033195–2

234567890DODO798765

This book was set in Times Roman, and printed and bound by R. R. Donnelley & Sons, Company. The designer was Richard Paul Kluga; the drawings were done by John Cordes, J. & R. Technical Services, Inc. The editors were Richard F. Dojny and Madelaine Eichberg. Matt Martino supervised production.

AUTOMATA THEORY: MACHINES AND LANGUAGES

McGraw-Hill computer science series

RICHARD W. HAMMING
Bell Telephone Laboratories

EDWARD A. FEIGENBAUM
Stanford University

Bell and Newell *Computer Structures: Readings and Examples*
Cole *Introduction to Computing*
Donovan *Systems Programming*
Gear *Computer Organization and Programming*
Givone *Introduction to Switching Circuit Theory*
Hamming *Computers and Society*
Hamming *Introduction to Applied Numerical Analysis*
Hellerman *Digital Computer System Principles*
Kain *Automata Theory: Machines and Languages*
Kohavi *Switching and Finite Automata Theory*
Liu *Introduction to Combinatorial Mathematics*
Nilsson *Artificial Intelligence*
Ralston *Introduction to Programming and Computer Science*
Rosen *Programming Systems and Languages*
Salton *Automatic Information Organization and Retrieval*
Stone *Introduction to Computer Organization and Data Structures*
Watson *Timesharing System Design Concepts*
Wegner *Programming Languages, Information Structures, and Machine Organization*

to Helen

Preface

The two extremes in the spectrum of approaches to mathematical topics are: (1) Proceed from definition to theorem with all the details written out and a few examples interspersed for "motivation." (2) Use discussion and some arguments about the plausibility of the results. The former might be called "exact mathematics," the latter, "descriptive mathematics." The latter approach has the advantage that one can omit the excruciating details of complex proofs, on the assumption that the reader can fill in those details.

Most writing about automata theory has adopted the "exact" approach. This approach tends to limit the audience to those having advanced mathematical maturity. In those fields where a theory has not been widely applied, practitioners who are neither extremely curious nor mathematically mature tend to avoid taking the time to work through the exact descriptions. Thus one consequence of the exact approach might be that applications of the theory are not discovered as quickly as they would be if the theory were more accessible to "outsiders."

Three years ago the author taught some of the concepts of automata theory to electrical engineering graduate students. This text evolved from

a set of notes written for that course. Because the majority of these students were interested in practical applications, an exact mathematics approach to the subject did not seem appropriate. Thus this text lies toward the descriptive end of the spectrum. But a book cannot be completely descriptive. Exactness is required especially in definitions and statements of the questions being studied. Exact definitions can be made formally or informally. In technical papers the formal approach is generally used. Therefore we often state formal definitions. However, we sometimes give informal definitions and leave the formulation of exact definitions to the problems.

In discussing some results we will omit tedious details. However, we will include statements about the types of steps which are omitted from the discussion but would be required in an exact proof. The discussion of each result will develop the arguments until the result is intuitively plausible. Readers are strongly encouraged to test each result with a simple example. This exercise should help develop their intuition about why the result holds. Some of the omitted steps are to be provided by the student as exercises in understanding detailed proofs.

The major results are collected in Appendixes 2 and 3. Appendix 2 summarizes the results relating sets of languages to each other. The answers to some unsolvability questions are summarized in Appendix 3. An annotated Bibliography, which includes some recent papers not specifically discussed in the text, is provided. Original sources are cited. References to the Bibliography are cited at the end of each chapter, in the section titled Comments. The reader is encouraged to read some of these papers. The annotations should be useful if one is looking for a specific result. Several comprehensive bibliographies have appeared recently [Rahimi (1970) and Wood (1970)]. They should be consulted if a more complete listing of the literature is required.

History Mathematical theories are often developed without regard for any applications they may have to problems which concern scientists and engineers. Other mathematical theories developed as the need arose in practical problems. Automata theory developed in both ways. Some of the theory was developed in the 1930s, well before electronic digital computers were built (in the 1940s). Most of the theory has been developed since 1953. One possible application—attempting to explain natural languages (French, English, etc.)—is a very complex problem. The motivation usually given for a study of automata theory is that its problems relate to the problems of translating the languages used to express algorithms for computing machines. Perhaps people in other disciplines will develop new applications of automata theory when they learn more about the ideas and results.

In this text we will emphasize the connection between machines and (mathematical) linguistic models because it is my conviction that most results

can be discovered from a machine interpretation of the problem. Also, many people have more intuition about the behavior of a machine than they have when they think in mathematical abstractions.

Some machine models are closely related to the data structures used in software systems. There complex data structures can simplify the programmers' ways of thinking about the problem. The criterion of simplicity is very complex. A description may be simple if it is phrased in terms of complex structures. If a person has a repertoire of structures to consider, he may be able to find simpler descriptions of his problem by using one of the more complex structures for the statement. In our discussion we will develop a repertoire of machine structures.

We assume that the reader is not familiar with linguistic models but that he does have some knowledge of finite-state machines. This background can be provided by a course covering sequential circuit synthesis, or by material supplementing Chapter 2. In Sections 2.1 and 2.2 we briefly review some of the concepts from that area. We do not assume that the reader knows anything about the relationships between machine models and linguistic models. A knowledge of this relationship is important for intuition. We will proceed somewhat slowly in the first six chapters, introducing linguistic and machine models and proving results about their interrelationships. Occasionally we prove other results, either in the course of becoming familiar with the operation of a machine model or to relate the differing capabilities of the models. As we introduce new models, we discuss some reasons why they might be interesting, often from outside the context of automata theory. Except for these insertions, the structure of each of Chapters 2 to 5 is similar: A discussion of a deterministic machine model is followed by a definition of acceptance. A result concerning the languages accepted by the deterministic machines is proved. Then the nondeterministic model is introduced and its relationship to the deterministic model is discussed. Finally, the relationship between the nondeterministic model and a linguistic model is discussed.

After the relationship between machines and languages is fully developed, we proceed to results which are usually phrased in terms of the languages alone. Even here we will use the machine formulations of the problems to prove most of the results. This approach is taken, not to fulfill our prediction that machine models are useful, but because the proofs of many of the results can be obtained by drawing the proper picture of a machine structure.

Notice that we say that a proof is obtained by drawing a picture. This again emphasizes that our approach is not to provide exact proofs in all the detail which some might consider desirable. Rather, we proceed with the exact proof until a point is reached when the reader should realize the structure of the proof, so that he could complete it if he wanted to. In Chapter 2 we discuss exactly what types of detail are omitted, and occasionally we include an exercise in which the reader should either provide a statement of

what is missing or complete the missing parts. The latter type of exercise is not illuminating in most cases; therefore solving one or two such exercises should be sufficient for any reader.

Philosophy of proofs Most of the results which relate machines to languages are proved by simulations. The machine "mimics" the language, and vice versa. Often some encoding of information is required. When the simulations are complex, we describe them by first describing the coding to be used and then showing a flow chart for the simulation algorithm. Often, if the simulation is simple, we dispense with the flow chart and use a word description of the algorithm. The reader is encouraged to try the simulations on simple examples so that he can fully understand the reasons for the particular result obtained.

Chapter summaries Since the concepts of mathematical linguistics are probably unfamiliar to most readers, we begin in Chapter 1 with a description of the basic concepts, rules, and problems of mathematical linguistics. The reader who is familiar with the work of Chomsky could skim this chapter.

In succeeding chapters we will develop models of differing complexity and capability. We will show the relationship between "hardware" descriptions and linguistic descriptions of computations. Our hardware descriptions will be limited to descriptions of structures and types of components. We will not discuss the detailed interconnections of any logical elements which might be used in a particular realization of any machine. Transition tables, flow charts, and similar techniques are used to describe the behavior of a machine. We will show how the structural details of a machine can be found from the description of any particular related language, and how the language can be found from a knowledge of the structural details of the related machine.

The simplest machine structure is the finite-state machine. In Chapter 2, we will see how these machines are related to particularly simple languages. The relationships that we will consider later are easier to understand in this simple context. Therefore we discuss some concepts in this familiar context, even though they might have been introduced later.

Turing machines can perform complex calculations. In Chapter 3, we will discuss the Turing machine model for computation and show how its computations are related to some very general languages and mathematical functions.

Many interesting machine models can be derived from the Turing model by restricting the amounts of time or space used during the computation. In Chapter 4, we discuss the linear-bounded automaton, the simplest of the restricted Turing machines, and relate it to a class of languages.

Pushdown automata use a data structure very similar to some data

structures used in compilers, some user programs, and one family of computers. In Chapter 5, we discuss this model, relating it to languages similar to ALGOL.

The machine models introduced in Chapters 2 to 5 are not sufficient to model certain aspects of compilation problems. Modifications of the structure of a machine or the imposition of restrictions upon their operation can change the capabilities of the machine. In Chapter 6, we discuss some of these changes, which happen to be closely related to solutions for some compilation problems. We also discuss some which fit into the hierarchy of machine models in interesting ways.

The last three chapters discuss questions which are often phrased in linguistic terms, though the proofs of the results can usually be made by finding machine structures which are relevant to the question. For example, in Chapter 7, we discuss certain mappings which can be applied to the sentences of a language. By finding a machine which performs the mapping it is easy to prove some properties of the results of the mapping. In Chapters 8 and 9, we discuss some questions that might be asked about languages, such as "Does this language contain any sentences?" After discussing the simple cases—those in which the questions can be answered—in Chapter 8, we turn, in Chapter 9, to those cases where an algorithm to answer the question in all cases cannot exist. We close by showing the theoretically curious result that there is an infinite hierarchy of classes of machines (and corresponding languages) about which many questions not only cannot be answered, but also become increasingly difficult for the machines higher in the hierarchy.

Comments to readers The importance of working the problems and carefully examining the examples cannot be overemphasized. It is very easy to sit back and nod your head "yes" when proofs are discussed, but unless you try the proof or try to perform the construction in an example, you may not understand why some details are necessary. There are suggested problems at the end of almost every section of the book. The difficulty of these problems varies greatly. Some are simple exercises in executing algorithms discussed in the section. These problems are placed toward the beginning of each set of problems. Other problems are statements of questions that we believe are open at the present time. The latter problems are marked (R). Some problems discuss results that will be cited in a later section of the text. These problems are marked (P).

Comments on ordering The ordering of Chapters 1 to 5 should not be changed, except that Chapter 5 can be moved to any point after Chapter 2 (except for the material in Sec. 5.7). The materials in Sections 5.8 and 6.4 can be omitted without loss of continuity. Chapters 6 and 7 should be discussed only after the first five chapters.

The material in Chapters 8 and 9 may be interspersed with the material from Chapters 3, 4, and 5. For example, the definition and unsolvability of the correspondence problem (Section 9.1) can be discussed after the halting problem (Section 3.6). Then some of the unsolvable linguistic problems in Chapter 9 can be discussed after Chapter 4 and the others after Chapter 5. The solvable cases of these problems are discussed in Chapter 8, which can be left for the mature student to read without class discussion.

Audience The more mathematical maturity the student has, the faster can the material in this book be covered. At the graduate level for electrical engineering students, this text can require one semester, or two quarters if the students have previously studied the synthesis of sequential circuits. At the junior or senior level, a year would be required, and the instructor should supplement the text with some background material from the theory of finite-state machines.

The Association for Computing Machinery has published some curricular suggestions for computer science programs. This text was planned before these proposals were made, and does not exactly match any one of them. This text material is covered in courses A1, A7, and I7 of ACM (1968) and in courses 4, 5, and 6 of McNaughton (1968).

Acknowledgments It is difficult to single out those persons who deserve the most thanks for helping me, either directly or indirectly, with this effort. I will cite a few individuals, knowing that others who have contributed must be omitted. What follows is a chronological listing of some of my associates, because this avoids the difficult task of ranking the persons, either by the magnitude of their influence or alphabetically. I begin with Prof. David A. Huffman, who had the patience to advise me during my thesis research (on a problem unrelated to the present text) and caused me to pay more attention to my writing. Next, my colleagues, past and present, Profs. F. C. Hennie, D. J. Kuck, and C. L. Liu (alphabetically!) introduced me to much of this material. Professor R. J. Collins, as department chairman, has provided a congenial atmosphere in Minneapolis in which I could teach the course and write the notes that formed the backbone of this text. My colleagues at the University of Minnesota contributed to that atmosphere. One of them, Prof. O. H. Ibarra, has checked my accuracy in many places, though any errors are my own responsibility. My students have used class-note versions of this material and have pointed out places where the discussions required changes, contractions, or expansions.

Mrs. Sharon Nelson has typed several drafts of this manuscript with extremely high accuracy. Without this help this book would not have been completed as quickly or as easily.

My wife has been helpful with editing the text, but editing errors are

my responsibility. Last, but far from least, my wife and children have put up not only with my moments at the desk with pencil and paper, but also with my daydreaming about how best to approach some of these topics. Their only compensation has been to share some elation when a thought in the shower clarified the problem.

Richard Y. Kain

Contents