# Universitext

For other titles published in this series, go to
www.springer.com/series/223

Ian Chiswell

# A Course in Formal Languages, Automata and Groups

Springer

Ian Chiswell
Department of Pure Mathematics
School of Mathematical Sciences
Queen Mary, University of London
London E1 4NS, UK
i.m.chiswell@qmul.ac.uk



ISBN 978-1-84800-939-4 e-ISBN 978-1-84800-940-0
DOI 10.1007/978-1-84800-940-0

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2008939035

Mathematics Subject Classification (2000): 03D10, 03D20, 20F10, 20F65, 68Q05, 68Q42, 68Q45



Printed on acid-free paper

Springer Science+Business Media
springer.com

# Preface

This book is based on notes for a master's course given at Queen Mary, University of London, in the 1998/9 session. Such courses in London are quite short, and the course consisted essentially of the material in the first three chapters, together with a two-hour lecture on connections with group theory. Chapter 5 is a considerably expanded version of this.

For the course, the main sources were the books by Hopcroft and Ullman ([20]), by Cohen ([4]), and by Epstein et al. ([7]). Some use was also made of a later book by Hopcroft and Ullman ([21]). The ulterior motive in the first three chapters is to give a rigorous proof that various notions of recursively enumerable language are equivalent. Three such notions are considered. These are: generated by a type 0 grammar, recognised by a Turing machine (deterministic or not) and defined by means of a Gödel numbering, having defined "recursively enumerable" for sets of natural numbers. It is hoped that this has been achieved without too many arguments using complicated notation. This is a problem with the entire subject, and it is important to understand the idea of the proof, which is often quite simple. Two particular places that are heavy going are the proof at the end of Chapter 1 that a language recognised by a Turing machine is type 0, and the proof in Chapter 2 that a Turing machine computable function is partial recursive.

Chapter 1 begins by discussing grammars and the Chomsky hierarchy, then the notion of machine recognition. It is shown that the class of regular languages coincides with the class recognised by a finite state automaton, whether or not we restrict to deterministic machines, and whether or not blank squares are allowed on the tape. There is also a discussion of Turing machines and the languages they recognise, including the result mentioned above, that a language recognised by a Turing machine is type 0. There are also further characterisations of regular languages, including Kleene's theorem that they are precisely the rational languages. The chapter ends with a brief discussion of machine recognition of context-sensitive languages, which was not included in the course.

Chapter 2 is about computable functions, and begins with a standard discussion of primitive recursive, recursive and partial recursive functions, and of primitive recursive and recursive predicates. Then various precise notions of computability are

considered. These are: computation by register programs, by abacus machines and by Turing machines. In all cases, it is shown that the computable functions are precisely the partial recursive functions. The account follows [4], except that modular machines are not used. This entails giving a direct proof that Turing machine computable implies partial recursive. As mentioned above, this is heavy going, although briefer than if the theory of modular machines had been developed. To ease matters, the proof of a technical lemma has been placed in an appendix.

Chapter 3 begins with an account of recursively enumerable sets of natural numbers. Recursively enumerable languages are defined by means of Gödel numberings, and we then proceed to the proof of the main result, previously mentioned, characterising recursively enumerable languages. The comments on complexity at the end of the chapter were not included in the course, and are intended for use in Chapter 5.

Chapter 4 is about context-free languages and is material not included in the course. It is considerably heavier going than the previous three chapters. Much of the material follows the books of Hopcroft and Ullman, including their more recent one with Motwani ([22]). Some of the results are needed in Chapter 5. However, the ulterior motive for this chapter is to clarify the relationship between $\mathrm{LR}(k)$ languages and deterministic (context-free) languages. Neither [20] nor [21] seems to give a complete account of this.

Chapter 5 is on connections with group theory, which is a subject of great interest to the author, and a primary motivation for studying formal language theory. It begins with the author's philosophical musings on the idea of a group presentation, which are quite elementary. There is a brief discussion of free groups, free products and HNN-extensions. Most of the rest of the chapter is devoted to the word problem for groups. We prove Anisimov's theorem that a group has regular word problem if, and only if, it is finite. The highlight is a reasonably self-contained account of the result of Muller and Schupp. This says that a group has context-free word problem if and only if it is free by finite. It makes use of Dunwoody's result that a finitely presented group is accessible. To give a proof of this would have been too great a digression. A discussion of groups with word problem in other language classes is also given. The chapter ends with a brief discussion of (synchronous) automatic groups, including a proof of the characterisation by means of the fellow traveller property.

Expanding the lectures has given Chapter 5 a theme, which is the interplay between group theory, geometry (specifically, the Cayley graph) and formal language theory. It seems likely that there is a lot more to be said on this subject.

The proofs of several results have been placed in Appendix A, usually to improve the flow of the main text. In some cases, these were given as handouts to the class. Appendices B and C were also handouts, although Appendix B has been expanded to include a brief discussion of universal Turing machines. Appendix D contains solutions to selected exercises. A complete solutions manual, password protected, is available to instructors via the Springer website. To apply for a password, visit the book webpage at *www.springer.com* or email *textbooks@springer.com*. The number of exercises is fairly small, and they vary in difficulty; some of them can be used as

templates for similar exercises (only the exercises in Chapters 1 and 2 were actually used in the course).

The impetus for the development of formal language theory comes from computer science, and as already noted, it can be at times quite complicated. Despite this, it is an elegant part of pure mathematics. The book is written by a mathematician and intended for mathematicians. Nevertheless, it is hoped it may be of some interest to computer scientists.

The book can be viewed only as an introduction to the subject (the audience consisted of graduate students in mathematics). For further reading on formal languages, see, for example, [33] and [34].

The prerequisite for understanding the book is some exposure to abstract mathematics, including an understanding of some basic ideas, such as mapping, Cartesian product and equivalence relation (note that "mapping" and "function" mean the same thing throughout the book). At various points the reader is assumed to be familiar with the combinatorial idea of a graph. This includes both directed and undirected graphs and the idea of a tree. Generally, vertices of a graph are denoted by circles or dots, but in the case of parsing trees (Chapter 4) they are indicated only by their labels. Of course, in Chapter 5, some knowledge of basic group theory is assumed. Also, the reader needs to know at least the definition of a semigroup and a monoid. No advanced mathematical knowledge is needed.

Concerning notation, words in a formal language are elements of a Cartesian product $A^n$, where $n$ is an integer, and in this context are usually written without commas and parentheses. In other cases where Cartesian products are involved, for example the transitions of a machine or the definition of grammars and machines, commas and parentheses are used. The exception is in writing the transitions of a Turing machine, in order to conform with what appears to be the usual practice. Our definitions of grammars and machines are quite formal. This seems the best way to proceed, although it has gone out of fashion when defining basic mathematical objects (such as a group). As usual, $\mathbb{R}$ denotes the set of real numbers, $\mathbb{Q}$ the set of rational numbers, $\mathbb{Z}$ the set of integers and $\mathbb{N}$ the set of natural numbers, which in this book means $\{0,1,2,\ldots\}$.

The author thanks Sarah Rees, Claas Röver and Richard Thomas for their helpful conversations and email messages. In particular, several of the arguments in Chapter 5 were suggested by Richard Thomas. He also warmly thanks Daniel Cohen for his very useful and perceptive comments on the manuscript.

A list of errata will be available on the book webpage at *www.springer.com*.

Queen Mary, University of London *Ian Chiswell*
School of Mathematical Sciences
June 2008

# Contents