BECKLEY, D. F. (1967). An Optimum System with Modulus 11. *The Computer Bulletin*, Vol. 11, No. 3, pp. 213-215.
BELL, D. A. (1972). Decimal Numbers. *The Computer Bulletin*, Vol. 16, No. 8, p. 373.
BERLEKAMP, E. R. (1968). *Algebraic Coding Theory*. McGraw Hill.
BRIGGS, T. (1970). Modulus 11 Check Digit Systems. *The Computer Bulletin*, Vol. 14, No. 8, pp. 266-269.
BRIGGS, T. (1971). Weights for a Modulus 97 System. *The Computer Bulletin*, Vol. 15, No. 2, p. 79.
CAMPBELL, D. V. A. (1970). A Modulus 11 Check Digit System for a Given System of Codes. *The Computer Bulletin*, Vol. 14, No. 1, pp. 12-13.
PETERSON, W. W., and WELDON, E. J. (1972). *Error Correcting Codes* (2nd Edition). MIT Press.
REID, C. J. (1970). Modulus 11 Check Digits. *The Computer Bulletin*, Vol. 14, No. 4, p. 122.
TANG, D. T., and LUM, V. Y. (1970). Error Control for Terminals with Human Operators. *IBM Jour Res and Devel.* Vol. 14, No. 4, pp. 409-416.
WILD, W. G. (1968). Theory of Modulus N Check Digit Systems. *The Computer Bulletin*, Vol. 12, No. 12, pp. 309-311.

Book review

The Theory of Parsing, Translation and Compiling, by Alfred V. Aho and Jeffrey D. Ullman; Volume 1: Parsing, 1972, 541 pages, £8·75, Volume 2: Compiling, 1973, 460 pages, £8·60. (*Prentice Hall*)

These two volumes bring together much of the substantial body of theory accumulated over the last decade from studies of the many models introduced to formalise various aspects of compilers.

Volume 1 commences with a review of mathematical concepts and a short chapter providing both an overview of compilers and a brief review of methods for specifying the syntax and semantics of programming languages. Chapter 2 completes the preliminaries with a thorough survey of regular sets, context free languages and the related finite and pushdown automata; this chapter might well serve as a text in a course covering these topics. Chapter 3 introduces formalisms for translation which are elaborated subsequently. Here we meet syntax directed translation schemata, finite and pushdown transducers; the relatively simple lexical analysis phase of compilation is treated in terms of regular expressions and finite transducers and then the subject of parsing is introduced. The intuitive notions of top down and bottom up parsing and their connection with left and right parses is discussed.

The considerable attention which the parsing problem has received in the literature is reflected in the fact that the next five chapters, rather less than half of the total work, are devoted to it; there is very little missing here. Chapter 4 deals with general parsing methods for context free grammars and includes the Cocke-Younger-Kasami method and that of Earley. Chapter 5, on one pass methods without backtracking, treats all of the main models which have been used in compilers, *LL*(*k*), *LR*(*k*), the many variants of precedence parsing and Floyd-Evans productions. Chapter 6 covers limited backtrack methods of both top down and bottom up varieties.

Starting Volume 2, Chapter 7 is concerned with some techniques for improving time and space requirements of various parsing methods and Chapter 8 develops the theory of deterministic parsing, establishing inclusion and equivalence relations between the language classes recognised by different deterministic parsers.

Chapter 9 returns to the subject of translation, dealing with intermediate representations of programs, models for code generation and syntax directed translation methods in the contexts of deterministic and backtracking parsing algorithms. Chapter 10 deals with the problems of storage and retrieval of semantic information for tokens, such as identifiers. In addition to the conventional solution using symbol tables and hashing functions, the theoretically interesting but (impractically?) expensive property grammars of Stearns and Lewis are examined. The final chapter presents the emerging theory underlying machine independent aspects of code optimisation. Program transformations eliminating both useless assignment statements and redundant computations are considered in the context of increasingly general environments—first within sequences of assignment statements then utilising algebraic properties, commutativity, associativity, etc. of certain operators, and finally, using flow analysis techniques, in the context of program loops. Other optimisations, code motion from within loops, reduction in strength of operators, efficient allocation of registers, all receive attention.

The presentation is formal, proofs are presented for the major theorems and lemmas but details are occasionally left and included in the many exercises at the end of chapter subsections. These exercises also serve to amplify, or to introduce additional, ideas. Bibliographical notes at the end of these subsections refer to the original papers listed in an extensive bibliography. (Volume 2 contains a composite bibliography for both volumes.) Great care has been taken in proofreading; for books of this typographical complexity, errors are few in number.

This is undoubtedly a valuable reference work for those committed to improving compiler technology and for those interested in formal languages and it is very welcome. It contains a wealth of material, examples and exercises which would prove useful in a course on compilers, but one might quarrel with the authors' recommendations on its use as a textbook in such courses. A number of topics which can have far reaching effects on the overall design of a compiler are not mentioned at all; examples are runtime diagnostics, runtime storage administration and the treatment of procedures and parameters. Their omission in a work on the theory of compilers is not surprising; currently there is little of mathematical significance to say. Their omission from a course on compilers, or their relegation to a laboratory course as matters of implementation detail, is a different issue.

Similarly one might also question the value of studying so many parsers in the detail suggested. The parsing problem, notwithstanding the attention it has received, has never been large in relation to the total problems of constructing a compiler. The *LR* methods of parsing, which have not been so fully discussed elsewhere, have considerable attractions from a practical viewpoint. They will surely completely displace several of the methods recommended for detailed study. It would be quite reasonable in a course simply to present the relatively straightforward reasoning which leads to (but tends to be obscured by) the formal descriptions of the algorithms which produce parsing tables. This would allow those students who wished to do so to pursue the details of the formal descriptions more readily and it would provide sufficient background for understanding more readily and it would provide sufficient background for understanding the simple algorithms which interpret parsing tables to produce a parse. It is certainly not necessary that a compiler designer be conversant with the details of the space-time optimisations implemented in his parsing table generating program any more than it is necessary to comprehend the details of implementation in compilers to design good programs. Familiarity with general principles will help and suffice for both.

In summary, I liked these books, they contain much that is not available elsewhere. They will certainly influence my teaching but I would find it necessary both to supplement and to prune vigorously to provide a balanced view in a course on compilers.

J. EVE (Newcastle)

Downloaded from https://academic.oup.com/comjnl/article/17/1/12/441286 by guest on 19 April 2024