coordinated publication than a single author would. No fault is found with the technical content, only with the number of words used to present the essential information.

L. C. Hobbs
Hobbs Associates, Inc.
Corona del Mar, Calif. 92625


This is the Proceedings of the International Symposium on the Theory of Machines and Computations held at the Technion—Israel Institute of Technology, Haifa, Israel, during August 1971. The 38 papers presented are all included in this volume; six are represented only by abstracts. They are divided as follows:

Section I: Computability Theory
Section II: Formal and Stochastic Languages
   1) Formal Languages
   2) Stochastic Automata and Languages
Section III: Finite Automata
   1) Algorithms and Bounds
   2) Linear Machines
   3) Algebraic Theory of Automata
Section IV: Fault Detection Experiments
Section V: Switching Theory
   1) Combinational Circuits
   2) Sequential Circuits

The papers in Section I are primarily concerned with “low-order” computability. Included are papers by A. Borodin (“Horner’s Rule is Uniquely Optimal”), S. Even and A. Atai (“Queues, Stacks, and Graphs”), and D. Tischritzis (“IFF Programs”). Tischritzis discusses programs containing “only if ... then ... else ... control statements and polynomial assignment statements.” Optimization procedures are presented for certain specific cases.

Section III.1 includes “An n log n Algorithm for Minimizing States in a Finite Automaton,” by J. Hopcroft. Also in this section, Z. Kohavi and J. Winograd (“Bounds on the Length of Synchronizing Sequences and the Order of Information Losslessness”) exhibit a four-input seven-output n-state machine that is information lossless of maximal order, for any number of states n. The authors state that “it may be possible to find an information lossless machine of maximal order with fewer inputs or outputs.” Z. Bavel, in “Connectivity and Separation of Automata” (Section III.3), presents “a method for finding the minimal separated subautomata of a finite automaton” and shows their use in finding the morphisms of finite automata.

In the section on Fault Detection Experiments, C. Seitz (“An Approach to Designing Checking Experiments Based on a Dynamic Model”) presents a heuristic technique for the design of checking experiments, based on the maintenance of a dynamic model during test generation. Direct sum machines and homing sequences are used, but distinguishing sequences and characterizing sequences are not used. The author states that the experiments generated “tend to be shorter or about the same in length as those designed by other techniques” [1]. However, a worst case or average case analysis of the length of the experiments is not presented. Therefore, it is not possible to compare the effort required or the length of the experiments with those generated using the subsequently published techniques of Hsieh [2].

The book contains a photocopy of each manuscript, as prepared by its author. The reproduction is of high quality and the presentation in no way hinders the reading of the material. The papers are reports of current research and results, rather than tutorial essays. This book is a worthwhile addition to any library, personal or organizational, that seeks to be complete in this area. It also can serve as a valuable aid to the researcher or student seeking a sample of some of the current research interests of, and problems being examined by, leading researchers in the theory of languages and computations.

References

A. C. ROOCHVAR
Univac, Div, Sperry Rand Blue Bell, Pa. 19422


This book is a major addition to the well-known McGraw-Hill Computer Science Series. Systems Programming, as viewed by the author, encompasses a vast array of programming areas, as can be seen from the chapter titles: Chapter 1: Background; Chapter 2: Machine Structure and Machine Language and Assembly Language; Chapter 3: Assemblers; Chapter 4: Macro Language and the Macro Processor; Chapter 5: Loaders; Chapter 6: Programming Languages: Chapter 7: Formal Systems and Programming Languages; Chapter 8: Compilers; Chapter 9: Operating Systems; and Chapter 10: Bibliography.

The preface observes that this book covers material in six courses of ACM’s Curriculum 68 (Commun. Ass. Comput. Mach., Mar. 1968). The book has been used for: a first course in undergraduate computer science, an advanced software course, a review or introduction for computer science graduate students, a service course for non-computer science students, and a software engineering course. It has been extensively used at M.I.T. and other universities, and at several industrial installations. Course videotapes are available, and so is a Teachers’ Manual, both of which should be useful to instructors.

Chapter 1 is a short outline of the material to follow, containing one-line definitions of terms. Most of these are so abbreviated as to be of value only to someone who already knows the meaning.

Chapter 2 discusses machine structure, machine language, and assembly language, primarily that of the IBM 360. There is a brief section on how index registers and loops are used for repetitive array operations. A set of 360 data and instruction formats are shown, and five pages discuss 360 assembly language.

In its attempt to describe the 360 in detail, this chapter skimps on insight into machine structure in general, and will probably be difficult going for those unfamiliar with the subject. (The preface does say that it is helpful if the reader has experience with assembly language, which would presumably include familiarity with machine architecture.)

Chapter 3 is an extended discussion of a two-pass assembler (evidently for a 360), including reasonably detailed flow charts and database layout. An attempt has been made to show good software design style here: the design flow chart starts with: “specify the problem; specify the data structures,” and a search is made for modularity—functions common to several phases.

There is also a good section on table searching and sorting, and hash functions, giving a spectrum of useful techniques, including the simple Shell sort. Unfortunately the programming style in this section is appallingly bad—the linear search on p. 81 goes into an infinite loop; the binary search on p. 84 works only on tables of size 2**N-1, although advertised as working on tables of size 2**N; and the sort programs on pp. 85 and 110 blow up on various short tables.

Chapter 4 contains almost 40 pages on macroprocessors and languages. Again the treatment is reasonably detailed, including more complex issues, like nested macro calls and definitions.

Chapter 5 discusses conventional loaders for resolving symbolic cross