Run #	*# Input records*	*# Duplicates*	*# Errors found*	*E*
—	—	—	—	3,600
1	13	2	0	255
2	37	37	1*	6
3	35	18	0	3†

Notes

*This error was immediately corrected by the program user who changed the documentation to agree with the program action.

†Program accepted by user.

Fig. 5 Statistics of FINBUDD22 testing

program. The full program, called FINBUDD22, scans a file of financial transactions, eliminates all duplicate entries in the file, and prepares a report of all duplicates found and eliminated.

The automatically produced flowchart for a brief excerpt consisting of statements numbered 13, 14 and 15 is shown in **Fig. 2.** Beside each statement is shown the initial values assigned to *a* and *b*, and values obtained for *e* as testing progressed. The frequency information f_i, is shown on the program flow lines. Since statement 15 had been identified as a key statement for FINBUDD22, a pre/posterior analysis was performed on it prior to each test run. The analysis applicable to run 1 is shown in **Fig. 3.** However, since run 1 did not contain inputs to exercise statement 15, the analysis also applies to run 2. The utilities shown in this figure are those established by the program user from a consideration of the losses associated with errors. The values for *a* and *b* from the flowchart have been used in calculating the probabilities of the outcomes on the T1 branch of the analysis. This diagram assumes that the inputs on the run will be such as to exercise the statement once. The expected value of the sample information (EVSI) for such a run is 1 − 0·39996 (thousands of dollars), and justified a test run. This was done and no errors appeared in the statement, so the appropriate posterior probabilities of Fig. 3 become the prior probabilities for **Fig. 4.** In this case, the EVSI is about $270 which did not justify further testing of this statement.

During these runs and additional runs not shown, similar computations were conducted for the other key statement of the program. The error loss estimate, initially artificially high at a $3,600 loss per run, was found to indicate an acceptable program after three runs. These runs are summarised in **Fig. 5.**

Conclusions

The paper has briefly outlined a method to obtain a measure of the reliability of a program. In the example discussed, the outcome was that the program being tested was found to be acceptable. Other possible outcomes are that the program is found to contain so many errors that it is unacceptable, or that the 'costs' of testing to improve a reliability estimate are uneconomic.

The approach is only partially automated at present, but it would not be difficult to automate other portions of it. However, it will always require a skilled programmer for its performance. Although it has only been used for COBOL programs we see no inherent difficulty in extending it to other languages.

One promising observation is that in applying this certification procedure to a library distribution control program, a significant error was discovered which had remained undetected (although, in retrospect, troublesome) through two years of production use of the program.

References

BROWN, J. R., and LIPOW, M. (1975). Testing for Software Reliability, paper presented at the International Conference on Reliable Software, 1975.

FELLER, W. (1957 and 1966). *An Introduction to Probability Theory and its Applications*, Vols. 1 and 2, Wiley.

FISHBURN, P. C. (1968). Utility Theory, *Management Science*, Vol. 14, p. 335.

KITTLER, M. A. R. (1975).Towards Certification of Computer Programs, Ph.D. thesis, Dept. of Computer Science, University of Toronto.

WALD, A. (1950). *Statistical Decision Functions*, Wiley.

Book reviews

Foundations of Secure Computation, edited by R. A. Demillo, D. P. Dobkin, Anita K. Jones and R. J. Lipton, 1978; 404 pages. (*Academic Press*, $19.50)

This is a collection of separate research papers, with the inevitable overlap, disjointedness and lack of solution to problems. Its subject is data access security in computer systems, and it concentrates on online system problems of access. It contains a number of irritating typing errors in the papers. Frequent unfamiliar abbreviations make the book difficult to use. Nevertheless, there is a great deal of realism about the level of security afforded by various methods of protection. Software researchers and developers and computer scientists should find it a useful collection of ideas. Computer managers who selectively scan the book, perhaps avoiding the detailed argument, would make a useful review of the problems.

In the detail there are a number of assumptions which may worry commercial systems users. There are statements such as 'owner controls own data'; 'when an application programmer installs a transaction'; and 'a grantee (of access) passing on his rights to others'. Clearly the computer user experience background of the authors has been in the technical and educational spheres. Much more consideration needs to be given to the complexities, division of duties, and the requirements of business systems.

The papers clearly recognise that a great deal of development work remains to be done, but they have made a useful contribution.

A. J. THOMAS (Sunbury-on-Thames)

Machines, Languages and Computation by P. J. Denning, J. B. Dennis and J. E. Qualitz, 1978; 601 pages. (*Prentice-Hall*, £15·35)

This book based on a course given at MIT, aims to integrate the theories of machines on the one hand (from finite state machines to Turing machines) and on the other hand linguistic theories. It comes closest to the interests of the practical computer scientist when dealing with syntax analysis. The generally terse style of writing which one has come to accept in the work of mathematicians is lightened here and there with discussions of the significance of the concepts and results, and, usefully for the teacher, with many concrete examples of the abstract ideas under discussion.

The first three chapters set the scene and introduce the basic concepts, terminology and notation. The next three chapters deal with finite-state machines and languages and in subsequent chapters these are generalised to deal particularly with pushdown automata and context free languages. After an excellent chapter on syntax analysis (both top-down and bottom-up) the book deals with Turing machines and unsolvable problems. Finally the book relates the foregoing topics to recursive functions and Post systems.

Not many British computing science students will get so thorough a treatment of these topics and at the price of the book few could afford to buy it anyway. However, any lecturer giving a course on these topics will find the book an invaluable source of ideas to enrich his teaching.

P. G. RAYMONT (Manchester)