ming decisions that we have expressed at a higher level. Moreover, by using a single formalism in which both specifications and algorithms can be expressed, we are in effect using a calculus for deriving programs (*cf.* Dijkstra, 1976).

For the sort program synthesis the obvious choices and programming strategies have led to known algorithms. However, there remains the hope that by gaining experience about what choices are crucial for the derivation of known algorithms, new algorithms can be discovered by systematically investigating alternative derivation paths.

4.1. *Related work*

A similar description and synthesis exercise for sort programs has been done by Green and Barstow (1977). The major difference of our approach is that the sort programs are derived within a formal deductive system each step in the derivation being a deductive step. For them, the actual program transformations are performed by programs that embody both the definition that we would make explicit in the specification, and the 'rules of programming' of the kind that we have used to guide the derivations.

Hogger (1977) shows how the four sort programs we have considered can be derived in predicate logic from a formal specification, However he does not view the specification as a high level program, with the derivations a symbolic execution of this program, an approach which we think pays great dividends. He has a more syntactically formal approach.

The work reported in Darlington (1978) approaches a similar task to the one reported here in a different way. There six sorts are synthesised from a top level definition. First the set of all permutations of a list is defined and three different algorithms for its computation synthesised. The sort function is then defined using a filter that rejects all but the ordered permutation. Different ways of combining this filter with each of the permutation algorithms, results in versions of six sorting algorithms, Quick Sort, Merge Sort, Insertion Sort, Selection Sort, Exchange Sort and Bubble Sort.

Lothar Schmitz (1978) has performed a similar exercise to ours, this time deriving a family tree of transitive closure algorithms. In a similar manner to ourselves, he has attempted to do as much manipulation as possible on a high level specification written in formal language.

Acknowledgements

Special thanks are due to R. M. Burstall and R. Kowalski for their encouragement and help.

The form and content of this paper have been much influenced by the constructive comments of the referee. We should like to thank him for his efforts on our behalf.

The work was supported by the Science Research Council.

References

BURSTALL, R. M. (1977). Design considerations for a functional programming language, Proc. Infotech State of the Art Conference, Copenhagen.

BURSTALL, R. M. and DARLINGTON, J. (1975). A transformation system for developing recursive programmes, JACM, Vol. 24 No. 1 pp. 44–67.

CLARK, K. L. and SICKEL, S. (1977). Predicate logic: a calculus for the derivation of programs, IJCAI5, 1977.

CLARK, K. L. and TÄRNLUND, S. A. (1977). A first order theory of Data and Programs. *Proc. IFIPS Conference*, Toronto, Canada.

CLARK, K. L. (1977). Synthesis and verification of logic programs, Research report, Dept. of Computing and Control, Imperial College, London.

DARLINGTON, J. (1975). Application of program transformation to program synthesis, *Proc. IRIA Symposium on Proving and Improving Programs*, Arc-et-Senans, France, pp. 133–144.

DARLINGTON, J. (1978). A synthesis of several sort programs, *Acta Informatica*, Vol. 11 No. 1, pp. 1–30.

DIJKSTRA, E. W. (1976). *A discipline of programming*, Prentice-Hall, Englewood Cliffs, NJ.

GREEN, C. and BARSTOW, D. (1977). *Program synthesis for efficient sorting*, A.I. Lab, Computer Science Dept., Stanford University.

HOARE, C. A. R. (1962). Quicksort, *The Computer Journal*, Vol. 5 No. 1, pp. 10–15.

HOGGER, C. J. (1977). *Deductive synthesis of logic programs*, Research report, Theory of Computing Research Group, Dept. of Computing and Control, Imperial College, London.

MANNA, Z. and WALDINGER, R. J. (1975). Knowledge and reasoning in program synthesis, *Artificial Intelligence*, Vol. 6 No. 2, pp. 175–208.

SCHMITZ, L. (1978). An exercise in program synthesis: algorithms for computing the transitive closure of a relation, Draft report, Hochschule der Bundeswehr, Munich.

Book review

The Social Impact of Computers by G. A. Silver, 1979; 341 pages. (*Harcourt Brace Jovanovich*, £6·45)

The title of this book is perhaps misleading for a broad introduction to computers and computing, incorporating as a final theme a discussion of the social implications. As the author states in his preface, instruction in computers has been virtually ignored for 'students in the social sciences or liberal arts, to say nothing of the layperson', and he enthusiastically sets out to provide something to fill the gap.

The book commences with a section covering basic definitions; the history of computer development; computer hardware and software, including the concepts of batch and interactive programming and a summary of the main computer languages. A summary of occupational trends and employment in computing is followed by descriptions of computer applications in industry, government, etc. The final sections (about 40% of the material) deal with the social or 'people' aspects, including broad social issues (particularly privacy), computers in education and effects on the business world (including crime).

A final conclusion is reached that there is a danger of developing an overdependence on computers and technology and that 'means will have to be developed to moderate the influx of new technology to give people time to adjust to the changes about them'. The format and style are particularly noteworthy. There are clear and interesting diagrams and photographs and the material is interspersed with extracts from newspaper reports and some amusing cartoons. The style is clear, attractive and highly readable. Definitions are simple and down-to-earth, and examples are up-to-date (1978). The author finds it possible to cover a broad area in a relatively short book without becoming superficial.

All-in-all, a well written, helpful and interesting book for those outside the computer profession (and perhaps a refreshing second look for the insider too). Students, particularly, would make use of the exercises at the end of each chapter. A frustration for the British reader may be that the example applications, statistics and legislation quoted are American.

M. J. BLYTHE (Haywards Heath)