6. Acknowledgements

The CLIP project is supported by the Science Research Council and by the Department of Physics and Astronomy at University College London. The authors are grateful to the technical staff of the Image Processing Group who have collaborated in this project (particularly to T. J. Fountain, M. Postranecky and G. K. Shaw).

References

BADEN FULLER, A. J. (1973). *Engineering Field Theory*, Pergamon Press.

DUFF, M. J. B., WATSON, D. M., FOUNTAIN, T. J. and SHAW, G. K. (1973). A cellular logic array for image processing, *Pattern Recognition*, Vol. 5, pp. 229-247.

KRUSE, B. (1973). A parallel picture processing machine, *Trans. IEEE*, Vol. C-22, No. 12, p. 1075.

MCCORMICK, B. H. (1963). The Illinois pattern recognition computer-ILLIAC III, *Trans. IEEE*, Vol. EC-12, No. 6, p. 791.

PRESTON, K. Jr. (1971). Feature extraction by Golay hexagonal pattern transforms, *Trans. IEEE*, Vol. C-20, p. 1007.

UNGER, S. H. (1958). A computer orientated toward spatial problems, *Proc. IRE*, Vol. 46, No. 10, p. 1744.

UNGER, S. H. (1959). Pattern detection and recognition, *Proc. IRE*, Vol. 47, No. 10, p. 1737.

WATSON, D. M. (1974). The application of cellular logic to image processing, Ph.D. Thesis, University of London.

Book review

Programs, Machines and Computation, by K. L. Clark and D. F. Cowell, 1976; (*McGraw-Hill*)

Theoretical options are not generally popular with computer science students. Various reasons can be adduced to explain this. Firstly theoretical courses are considered to be more difficult than nontheoretical courses. Secondly they often appear to have little relevance in real situations, and finally what relevance they do have is often obscured by vast amounts of meaningless definitions and symbolic notation. I do not believe that it is possible to avoid the first objection—a good theory should have predictive as well as explanatory qualities and will inevitably be more difficult to understand than an empirically observed fact. However textbooks on the theory of computer science should always attempt to avoid the second and third criticisms.

The book by K. L. Clark and D. F. Cowell can be split into two parts—the first four chapters in which the material is well motivated and well presented, and the final three chapters in which the authors succumb to illusions of mathematical elegance and, consequently, lose all sight of the practical relevance of their subject.

The first three chapters lead up to the standard results on the undecidability of the halting problem of Turing machines (presented in chapter four) and are concerned with demonstrating the Church-Turing thesis—that any computable function is computable on a Turing machine. The choice of material here would appear to owe a lot to Minsky's book *Computation: Finite and Infinite Machines* and includes proofs of the equivalence of Turing machines with unlimited register machines and recursive functions. Yet the order and manner of presentation improves, I feel, on Minsky's and should appeal to computer science students. One feature of the book, which is very much to be applauded, is the addition of assertions establishing 'correctness' to all their programs. Exercises, at this stage of the book, are also well-chosen to illustrate and add to the material in the text.

Chapter 5, entitled 'Machines with input and output streams' is, in my opinion, the worst of the book. It is this chapter which the authors would probably regard as the most novel. So far as I can discern the main result in this chapter is that the set of all strings generable by a class of (deterministic) machines is identical to the set of strings accepted by the corresponding class of nondeterministic machines. However this result is obscured by the authors efforts to accommodate the more usual definitions of pushdown automata and finite-state machines into the formalism which they established in chapters 1 to 4. This predilection for mathematical elegance is continued in chapters 6 and 7 where proofs of a number of solvability results are chosen for their brevity and aesthetic (?) value rather than their practical value. For instance to test whether a finite-state machine having m modes accepts the empty set we are told to try all inputs of length $\leq m$! Similar algorithms are given for constructing a deterministic finite state machine from a nondeterministic one, testing two regular languages for equality and testing whether a context free grammar generates the empty set The combined effect of all these brief but impractical proofs is that when the authors prove an important result which the students can relate to their own experience—that the equivalence problem for (monadic) program schemas is solvable—the authors are unable to provide exercises illustrating the result, the simple reason being that it would take a very patient and careful student a very long time to follow through the steps in their proof on even a small example. In fact all the problems referred to above can be solved using variants of a simple graph searching technique. Had the authors included a short section on graph searching and then used the technique in their proofs it would have improved the last three chapters considerably.

On the whole the book contains few typographical errors or errors of fact. I shall be communicating errors I have found to the authors but the following more major errors are worth noting. An error on the last line of page 31 is amusing. The line reads

$$R_{x_i \leftarrow x_j}: \{n_i\}_{i=1}^{\infty} \mid \rightarrow \{n_i'\}_{i=1}^{\infty} \text{ where } n_i' = n_j$$
$$\text{and for all } i \neq j,\ n_i' = n_i\ .$$

There is clearly a typographical error here since the final j should in fact be i. Correcting this we obtain

$$R_{x_i \leftarrow x_j}: \{n_i\}_{i=1}^{\infty} \mid \rightarrow \{n_i'\}_{i=1}^{\infty} \text{ where } n_i' = n_j$$
$$\text{and for all } i \neq i,\ n_i' = n_i\ .$$

What a howler! Clearly there is a clash between the free variable i and the bound variable i. Clearly also the authors never expected anyone to read this line of text! The proof of theorem 1.6.3 is false—a proof must surely involve a graph searching technique (yet another reason for including graph searching in the text). In addition the proof of theorem 2.2.6 is incomplete—the authors prove a sufficient condition for one machine to simulate another, but by the time they reach theorem 2.2.6 they seem to have forgotten the existence of the condition. In fact their proof is correct but it assumes a less stringent notion of 'stepwise simulation'. Finally on page 52 they suggest methods of simulating $x \leftarrow x + 1$ and $y \leftarrow y + 1$ on a Turing machine which are unnecessarily complicated—as was pointed out to me by one of my students in the middle of a lecture!

In spite of these criticisms I feel that the book can be recommended to computer science students. The first four chapters cover basic material on unsolvable problems which should form a compulsory element of all computer science degrees. The material is extremely well presented and indeed, in my opinion, better presented than in any other text I know. The final three chapters mainly cover 'optional' material and hence one's appreciation of the choice of material will be very much more subjective. My criticisms reflect my own views on the importance of the relevance of theory, but it may be that others will find the last half of the book as much to their liking as the first.

R. C. BACKHOUSE (Edinburgh)