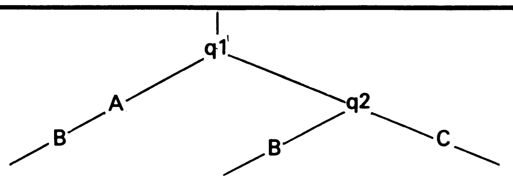

Fig. 2(*a*)

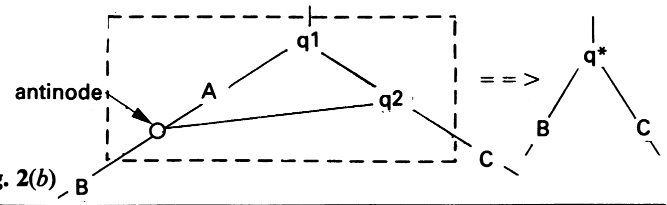

Fig. 2(*b*)

each GOTO statement is replaced by the text to which it points then the nested form of the program emerges automatically. This kind of GOTO may be called 'virtual' because it affects neither logic nor structure.

In my view the key to true unstructuredness lies in information, one bit of which is released by each conditional element (node) in a flow tree. If information is subsequently absorbed (at an 'antinode', where two branches merge), then the flow diagram is no longer a tree and unstructuredness occurs. This condition must exist if a loop is present, but it may be possible to 'internalise' the unstructuredness to a subdiagram, which can then be represented as a single conditional or procedural element.

The WHILE-DO loop is a good example of an unstructured construct which, because it has one entry point and one exit, may be replaced by a procedure (branch) in a flow tree.

Reducibility of a task in a two-valued logic environment thus depends on the availability of such constructs for internalising processes which absorb information. This is a function of human language conventions which appear ill equiped to deal with anything more complex than IF-THEN, WHILE-DO and similar constructs, as illustrated above.

Unstructuredness is not an inherent feature of any problem since a flow tree, albeit infinite, can always be constructed. In the interests of economy, however, it is felt desirable to identify similar conditional and procedural elements in the tree.

In the example Fig. 1(*a*) of Williams and Ossher (1978), whose flow tree is shown in **Fig. 2**(*a*), the urge arises to identify the two procedures *B*, which happen to be identical. This cannot be done without losing structure unless the unstructuredness is internalised to a complex conditional element as in Fig. 2(*b*). Although *q** is a feasible elementary construct words fail to describe it concisely and the diagram remains, for practical purposes, unstructured.

It may be observed that the arc in **Fig. 2**(*b*) linking *q*2 and the antinode represents a 'real' GOTO, which is quite distinct from the benign 'virtual' kind. It is unfortunate that these two constructs, share the same name and are thus tarred with the same brush. If, for example, the 'virtual' GOTO is renamed REFER then the confusion, and much of the acrimony, may be avoided since it is possible to detect and flag the case where a label is referred to by more than one REFER statement.

Yours faithfully
N. B. Taylor

'Hook-a-gate'
Eversley Road
Yately, Surrey
26 September 1979

References

Arblaster, A. T., Sime, M. E. and Green, T. R. G. (1979) Jumping to some purpose, *The Computer Journal*, Vol 22 No. 2, pp. 105-109.

Williams, M. H. and Ossher, H. L. (1978). Conversion of unstructured flow diagrams to structured form, *The Computer Journal*, Vol 21 No 2 pp. 161-167.

To the Editor
The Computer Journal

Sir,

Points and n-sided irregular figures

I have just noticed your correspondence about determining whether a point is inside or outside the given n-sided irregular figure and I would draw your attention to the great many solutions to this problem that have been produced in the field of Urban and Regional studies over the past seventeen years.

When the Department of the Environment considered this subject in 1975 in their Research Report 2 (Point-in-Polygon Project Stage I) they identified fourteen algorithms that solved the problem and of these nine had been published.

Yours faithfully,
Mervyn Bryn-Jones

Borough of Haringey
Hornsey Town Hall
The Broadway
Crouch End
London N8 9JJ
24 October 1979

Book review

Mini/Microcomputer Hardware Design by G. D. Kraft and W. N. Toy, 1979; 514 pages. (*Prentice-Hall*, £12·80)

This book analyses different approaches to small computer design. By concentrating on conventional small machines however, the treatment must be incomplete. Important architectural features that are traditionally software, but which increasingly affect the hardware, e.g. semaphore control, are left out. And large machine features such as virtual memory, which are becoming available on micros, are given little attention.

Chapters 1 to 6 review the structure of microcomputers and some of the earlier microprocessors (up to 8080, 6800). It shows how fundamental ideas, for example unified bus structures, and instruction code formats, have been incorporated into different machines. The text demands a basic understanding of computer systems by the reader, for ideas and words are often used without introduction. Chapter 7 describes the design of microcontrol units for microprogram sequencing of instruction fetch and execution cycles. In comparison with the rest of the book, this material is excessively detailed, and could usefully have been restricted to allow discussion of ROM-based microprogram techniques, which are excluded. Chapters 8 and 9 discuss program controlled input/output and direct memory access. The treatment of interrupts, for example, is comprehensive, but there is no discussion of modern programmable I/O controllers or special purpose I/O processors. The book's five year gestation clearly shows.

The useful and interesting examples are peculiar in that many cannot be tackled using this book. For example distributed computing is recognised as important, and several examples deal with it. However, there is only the briefest treatment of the subject in the text itself. Likewise error control and testing are practically ignored except in the examples. Perhaps there is a second book under way. Hence, while its choice of material is too uneven to provide a complete study of the subject, the book reviews and compares many approaches to hardware design. It is clearly written, and would make a useful back-up reference for system designers and students.

R. W. Prowse (Uxbridge)