```
*MAES-FIG-1. NOTE TABLE.
*
*    1 1 1 2 2 2 2 2 2 2 2 2 3 3   GROUP.
*    1 2 3 - - - - - - - - - - -   IS I1 .. IN.    = < >.
*    - - - 1 1 1 2 3 3 3 3 - -     IS I1 .. I2.    > = <.
*    - - - 1 2 3 - - - - - - -     IS I2 .. UGR.   > < =.
*    - - - - - - - 1 1 2 3 - -     IS I2 .. OGR.   = < >.
*    - - - - - - - N Y - - - -     IS A(I2) = A(IN).
*    - - - - - - - - - - N Y       IS A(I1) = A(IN).
*    - - X - - - - - - - - - -     PERFORM A.
*    - X X - - - - - - - - - -     PERFORM B.
*    - - - - - - - - - - X - -     PERFORM C.
*    - - - - - - - - - - X - -     PERFORM D.
*    X - - - - X X X - - - - -     PERFORM F.
*    - - - X - - - - - - - - -     PERFORM G.
*    - - - - X - - - - - - - -     PERFORM H.
*    - - - - X - - - - - - - -     PERFORM J.
*    - - - X X - - - - X - - -     PERFORM E.
*    - 2 2 3 3 - - - - 3 2 2 -     NEXT GROUP.
```

Fig. 2

more than one exit, the table must stipulate which exit is being followed. This is done by writing 'Y' or 'N' for two-way decisions. The more general case of multi-way decisions is done by numbering the exit paths. Such a 'multi-choice' condition is expressed by a generalised stub, followed by a list of parameters which apply to the specifically numbered exits. (This notation is a useful compromise between limited—entry tables, which tend to be cumbersome and extended—entry tables, which tend to fit too few rules on to a coding sheet.)

The technique of describing a flowchart by its paths breaks down when there are loops. But such flowcharts can always be described by dissecting them into components which are loop-free. This is done by assigning lables to selected points in the flowchart. There must be a label at the entry point of the flowchart, and at least one label in each loop. Beyond this, the choice is arbitrary.

The flowchart can now be documented by tabulating the flowpaths as follows:

1. The starting label (referred to as the GROUP)
2. The list of conditions and actions on the path
3. The terminating label (referred to as the NEXT GROUP).

To apply this technique to Maes' Fig. 1 four points can be labelled. The entry point can be labelled '1'; the conditions I1::I2 labelled '2'; and A(I4)::A(IN), '3'. By convention the exit point is always labelled '–'. The result of this labelling is shown in **Table 1.**

There is an interesting analogy with the standard technique used to convert an arbitrary program into a 'structured' program. This technique introduces an auxiliary variable to enable and program to be reduced to a single loop containing a single case statement.

We can, if we wish, imagine the GROUP to be just such an auxiliary variable. The first action in each rule tests it, and the last action either exits the program, or sets a new value in the variable and continues the loop. The conversion of a flowchart to a COPE decision table, and its 'structuring' with an auxiliary variable are closely analogous. (COPE does not actually use auxiliary variables. It will produce a COBOL program with the same flowchart use auxiliary variables. It will produce a COBOL program with the same flowchart as the original problem. The method is explained in Dwyer and Hutchings (1977.)

Because there is freedom in choosing the points to be labelled, different tables can be derived from a given flowchart. The various techniques reviewed by Maes correspond to COPE tables with different labellings. Thus Maes' Fig. 5 is the case of labelling all conditions. Table 5 is similar, except that the single auxiliary variable representing the label has been replaced by a set of two-way switches. Table 6 corresponds to labelling the entry point of each loop.

COPE can therefore be used in ways analogous to each of these examples. This flexibility may be exploited to choose a labelling which best seems to document the problem.

Contrary to Maes' general conclusions, COPE has proved to be an excellent COBOL programming tool (Dwyer, 1978). It is claimed to save about 50% of PROCEDURE DIVISION coding.

COPE is written in ANS COBOL. A source tape is available from the writer for a moderate charge. The program can be expected to work on most COBOL systems.

Yours faithfully,

BARRY DWYER
School of Mathematics and Computer Science
South Australian Institute of Technology
North Terrace
Adelaide, South Australia
2 February 1979

References

DWYER, B. and HUTCHINGS, K. (1977). Flowchart optimisation in COPE, a multi-choice decision table, *Australian Computer Journal*, Vol. 9 No. 3, p. 92.

DWYER, B. (1978). Experience with COPE, a Multi-choice decision table processor, *Proceedings of 8th Australian Computer Society Conference*, Vol. 1, p. 302.

MAES, R. (1978). On the representation of program structures by decision tables: a critical assessment, *The Computer Journal*, Vol. 21 No. 4, p. 290.

To The Editor
The Computer Journal

Sir

Pseudo-random sequences of Mersenne prime residues

The paper by G. W. Hill (*The Computer Journal*, Vol. 22, pp. 80-85) includes description of a FORTRAN function RESIDU(K,KN) 'which replaces the value of KN by K*KN(mod M), leaving K unchanged'. The author then, however, uses X=RESIDU(I(J), I(J)).

The 1966 Fortran Standard (clause 8.3.2) says 'If a function reference causes a dummy argument in the referenced function to become associated with another dummy argument in the same function . . . a definition of either within the function is prohibited'.

The 1978 Fortran Standard repeats this restriction, in slightly modified words, in clause 15.9.3.6.

Yours faithfully,

I. D. HILL
MRC Clinical Research Centre
Watford Road
Harrow, Middlesex HA1 3UJ

Book review

Computers and Commonsense (2nd Edition) by R. Hunt and J. Shelley, 1979; 166 pages. (*Prentice-Hall*, £3·50)

This new edition of a book first published four years ago has been updated by new technology and by the mention of new applications and social issues. As computers influence the life of the ordinary man more and more it becomes much more important to deal with these issues. Five pages may be better than nothing but it is very little compared to the space allocated to the arithmetic and logical unit with which few users of computers are directly concerned. The social issues at work, where top management still has a tendency to lay down the 'requirements' of a new system without consulting those who actively operate the former clerical system, has only been touched upon. The position of the Trade Unions and the welfare of the individual worker as a human being are not really investigated. Is compulsory idleness for the unskilled going to be an inescapable future? Trying to train them to higher levels of skill is an idealistic aim, especially if it involves a loss of personal identification.

This book is well put together, with its cross referencing and its updated bibliography, and such criticism only points towards areas where commonsense is still to be developed. The need for a new edition within four years shows the demand for this paperback which provides nine pages of introduction to BASIC in addition to the wide picture required by everyone who wants an introduction to computers or wishes to develop a sense of proportion in these confusing developments.

PHILIP GILES (Stirling)