A technique for computer flow chart generation

By F. O'Brien and R. C. Beckwith*

This paper describes an approach to the problem of computer flow chart generation. The resulting ICFLOW program is a simple and easily used aid which produces unannotated linear flow charts. The flow charts are not annotated as it has been found desirable to leave this phase to the programmer concerned.

(First received 1 December 1967)

One of the concerns of the Centre for Computing and Automation at Imperial College is the teaching of good programming practice. To this end new courses have been devised, and tested in practice, with the object of giving students a good grasp of programming techniques, and not just a knowledge of, say, FORTRAN coding.

Flow charting is taught as an integral part of the overall problem-solving effort required to implement any algorithm on a computer; as an aid to be used throughout the design and development stages of a program, as well as for final documentation. Flow charts are also extensively used as lecturing aids, illustrating the logical facilities of the programming language in a graphic manner, and also demonstrating good program design features.

There is some tedium in producing flow charts for our lecture notes, which are distributed, and also in the general production of flow charts in a neat form for final documentation. It was decided, therefore, to investigate the possibility of producing flow charts automatically, and various approaches were examined, notably those described in Sherman (1966), and Roberts (1967).

These approaches suffer from the main drawback that it is assumed that the text of the program contains sufficient information to describe the function of the program completely and clearly. We have found that, in general, production programs do not contain enough relevant information to enable a new reader to understand their function. Neither the variable names used, nor the usually sparse comments in the text, bear any close relationship to the function of the program.

For instance the FORTRAN variable XT may refer to 'Time' in one section of a program and 'Distance' in another, and we have also found that programmers are reluctant to insert comments into the text in other than a cryptic manner.

The flow chart generator that resulted from our study, called ICFLOW,† produces a conventional layout of boxes and arrows for any number of program routines at one time, within machine storage limitations, with no attempt made to insert text into the boxes, for the reasons given above. The programmer is left to insert, either by hand or typewriter, the information describing the function of each box in his own words, plus any additional information, such as labels, that he feels desirable for increased clarity. This has not proved to be an onerous task, and has the side effect of forcing the programmer to perform a rigorous check on the function of the program, which has proved very enlightening in some cases.

It has also been found that the proliferation of various types of box shapes, such as card shapes or magnetic reel shapes, is not necessary, and that a set of four simple shapes is satisfactory. These shapes are shown in Fig. 1.

A typical layout for a simple routine is shown in Fig. 2. The layout is such that a single box of type 1 is drawn for any set of instructions not containing a transfer, without any cognisance being taken of the number of instructions in the set, since it is not possible to relate the information content of the set to its length. It would, however, be a simple matter to relate the depth of the box to the number of the instructions in a set, if desired.

The mode of operation of the ICFLOW program is simply to place the routines to be flowcharted behind the ICFLOW card deck and run as a complete program. Our present version runs on the IBM 7090/IBSYS system and handles routines written in FORTRAN IV, and MAP, or their equivalent binary decks.

The ICFLOW program itself consists of two stages, a trace stage and a graphic output stage, which deal with one routine at a time. The trace stage, written in

† Pronounced 'ICEFLOW'.

* Centre for Computing and Automation, R.S.M. Building, Imperial College of Science and Technology, Prince Consort Road, London, S.W.7.
machine code, scans a routine and produces three lists describing the structure of the routine, these lists holding the initial address of a box, the type of box and the transfer address if appropriate. These lists are processed by the graphic stage to give an output suitable for a plotter (the Calcomp incremental digital plotter in our case).

To assist in producing an efficient layout of a set of flowcharts on the plotter paper, and to enable a programmer to obtain boxes of a suitable size to suit his verbosity, the box dimensions have to be specified at the input stage. This specification also determines the channel spacing between adjoining arrows.

Arrows joining non-sequential boxes are placed to the right of the main sequence if pointing downwards, to the left if pointing upwards. Channel allocation for such arrows is determined so that no two arrows are superimposed, though channels may be shared at various points in a flow chart, and so that arrows of shorter length tend to be nested within arrows of greater length. These points are illustrated by the set of flowcharts in Fig. 3, and it is worth noting that there is no restriction in the number of channels used nor in the length of a flowchart other than those due to the plotter size.

A reordering algorithm is used at the graphic stage to select a sequence of plotting actions in order to minimise the distance the plotter head has to move.

Using this algorithm the lists of box addresses, types, and transfer addresses are reordered such that any given box is followed by the box indicated by the transfer address, if appropriate, or by the next sequential box if no transfer is indicated. If the following box has already been reordered a search is made in both directions, from this box, to find the nearest unallocated box and the process continued from this new box until all the boxes have been reordered.

The flowchart is then drawn by sequentially processing the reordered lists, making references to a set of library routines containing descriptions of each box type and arrows, scaling the output drawing as specified by the programmer.

Using ICFLOW a large number of routines can be handled at one time, the greatest number to date being a complete program of 17 routines, while the largest routine produced 540 boxes stretching over 20 ft of Calcomp paper. It was decided that this routine was overlong.

References
Computer flow chart generation

Fig. 3