Generating structured flow diagrams: the nature of unstructuredness

M. H. Williams

Department of Computer Science, Rhodes University, Grahamstown 6140, South Africa

Various methods have been put forward by different authors for converting an 'unstructured flow diagram' into an equivalent 'structured' one. This paper examines the basic flow diagram substructures, the presence of which in a flow diagram causes the flow diagram to be unstructured, and proves that these are the only structures which lead to unstructuredness.

(Received March 1975)

1. Introduction

In the last decade there has been a growing interest in flow diagrams and program schemata, and, in particular, in the problem of determining the equivalence of flow diagrams (Ershov, 1972). Böhm and Jacopini (1966) introduced the idea of using a small set of base diagrams and showed how more general flow diagrams could be decomposed into these base diagrams. However, since it is not possible to decompose all flow diagrams into 'a finite number of given base diagrams', they proposed two normalisation methods for converting diagrams which could not be decomposed into a finite number of base diagrams, into equivalent diagrams which can be decomposed thus. This idea, viewed from the point of view of synthesis instead of analysis, namely that flow diagrams should be constructed initially from a small set of three base diagrams, has been termed structured programming and considerable success has been ascribed to this technique (Yourdon, 1974; Baker and Mills, 1973; Miller and Lindamood, 1973).

Since the terminology associated with structured programming appears to vary somewhat, for the purposes of this paper the term *structured flow diagram* will be used to refer to a flow diagram that can be decomposed completely in terms of the three base diagrams shown in **Fig. 1.** The selection mechanism generally used (Fig. 1(*b*)) is a two-way branch; however, one can extend this to an *n*-way conditional branch (Wulf, 1972) if desired. Contrariwise, an *unstructured flow diagram* is one that cannot be decomposed completely in these terms.

The notion of a method for converting an unstructured flow diagram into a structured one has attracted some interest since Böhm and Jacopini's original paper, and several methods for doing this have been put forward (Cooper, 1967; Ashcroft and Manna, 1972; Wulf, 1972). In each case the method was illustrated by one or two examples.

This paper examines the general nature of 'unstructuredness', i.e. those structures within flow diagrams which cannot be directly resolved in terms of the three base diagrams.

2. The basic causes of unstructuredness

There are five basic structures in flow diagrams which lead to nondecomposability in terms of the three base diagrams. These may be classified as:

(*a*) abnormal selection path

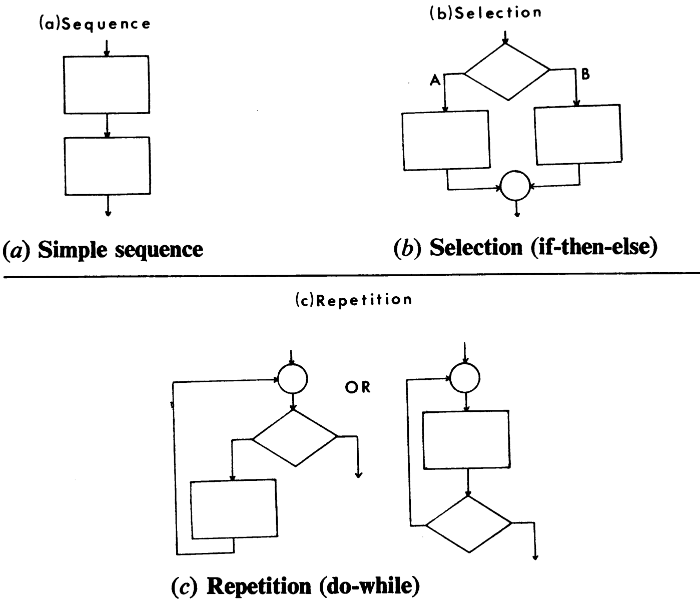
(*b*) loop with multiple exit points

(*c*) loop with multiple entry points

(*d*) overlapping loops, and

(*e*) parallel loops.

These five basic constructs are illustrated in **Fig. 2.** Cases (*d*) and (*e*) are very similar and could be combined into a single category; however, it is convenient to treat them as separate cases. If a flow diagram is unstructured, it must contain at least one (or possibly a combination) of these basic structures. In the next section it is proved that these are the only structures which cannot be decomposed in terms of the three basic diagrams in Fig. 1.



(*a*) **Simple sequence**

(*b*) **Selection (if-then-else)**

(*c*) **Repetition (do-while)**

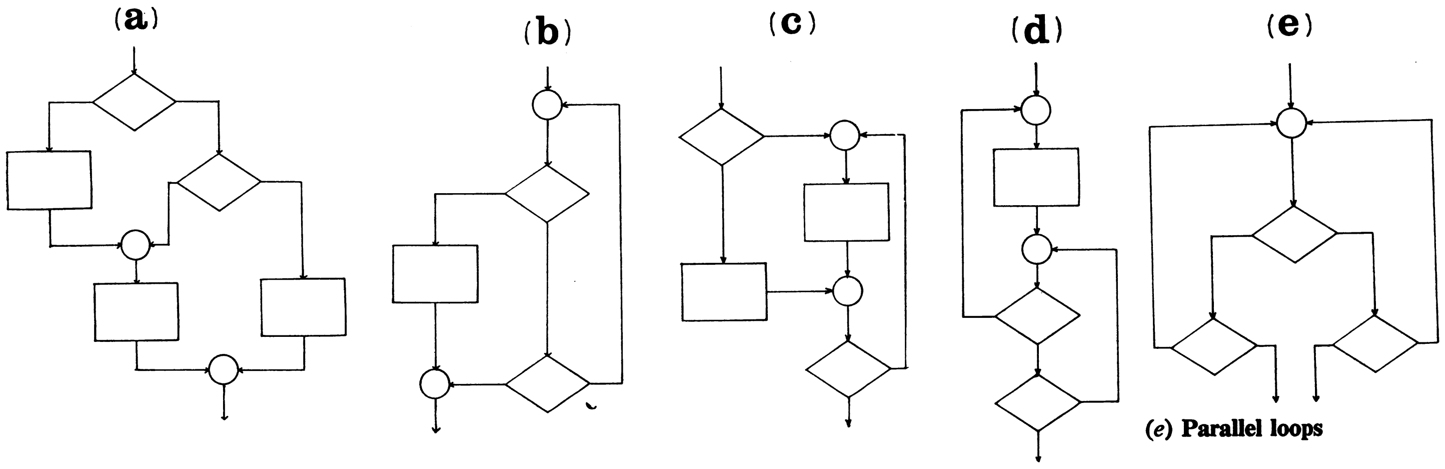
Fig. 1 The three basic structures of structured programming

Consider some of the examples used by other authors and how these all reduce to one or more of these five structures:

1. **Fig.** 3(*a*) shows an example used by Wulf in which the line marked *x* is an abnormal exit from the selection path of *B* and the line *y* an abnormal exit from the selection path of *A*, while both selection paths of *C* have abnormal entries.
2. Fig. 3(*b*) contains an example given by Yourdon in which all four selection structures have abnormal entries or exits.
3. **Fig.** 4(*a*) demonstrates an example from Ashcroft and Manna which can be reduced to the form given in Fig. 4(*b*), a loop with multiple exit points.
4. **Fig.** 5(*a*) is an example from the paper by Böhm and Jacopini (Ω_5). This is rearranged without alteration in Fig. 5(*b*) to show clearly that it is simply a loop with multiple exit points.
5. **Fig.** 6(*a*) displays an example taken from Mullins *et al.* (1974). Fig. 6(*b*) contains a similar example taken from the paper by Böhm and Jacopini (their Type II). Both examples correspond to the parallel loops class of structure.
6. **Fig.** 7 is an example used by Yourdon which illustrates overlapping loops.

3. Proof of existence of five unstructured forms

Consider any flow diagram consisting of a collection of boxes and lines. Assume that there are five different types of boxes



(*e*) **Parallel loops**

(*a*) **Abnormal selection path**

(*b*) **Loop with multiple exit points**

(*c*) **Loop with multiple entry points**

(*d*) **Overlapping loops**

Fig. 2 The five basic structures which cause unstructured flow diagrams

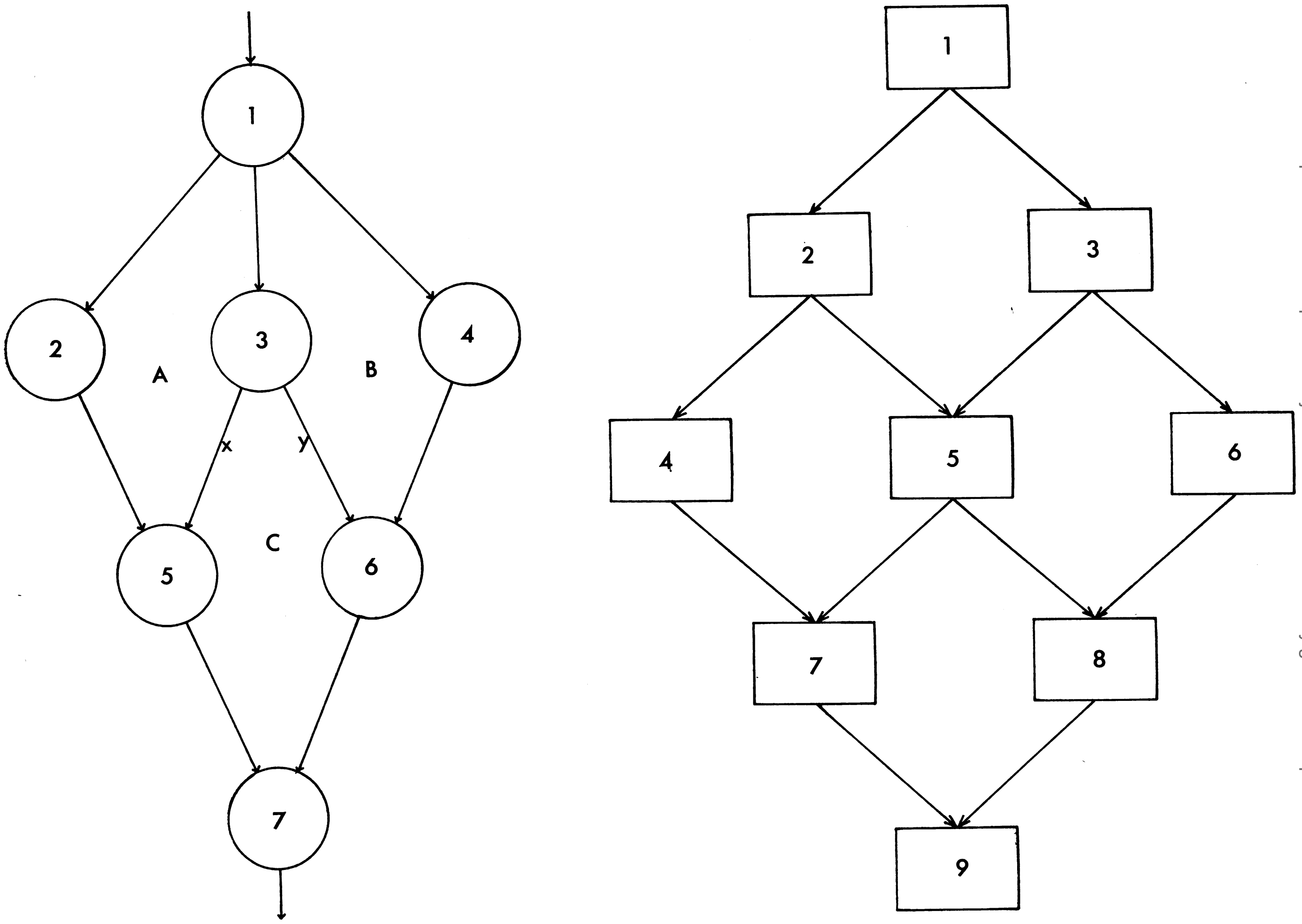


Fig. 3 Examples of abnormal selection paths given by: (*a*) Wulf (1972) (*b*) Yourdon (1974)

(see **Fig. 8**), viz.

(*a*) decision box

(*b*) process box

(*c*) junction box

(*d*) START box, and

(*e*) FINISH box

and that the flow diagram has only one entry point, the START box, and one exit point, the FINISH box.

Define a *path* as a sequence of successive directed lines through the diagram. The *length* of the path is the number of lines in the sequence. For each line (or box) in the diagram, there are a set of paths through the diagram from the START box to the line (or box) in question. These will be called *origin paths.* The *level* of a line or box is the length of its shortest origin path (the shorter the length, the lower the level).

A line is said to lie within a *loop* if there exist for that line two origin paths, A with path length n and B with path length m $(>n)$, such that the first n lines of the sequence B are identical to those of the sequence A.

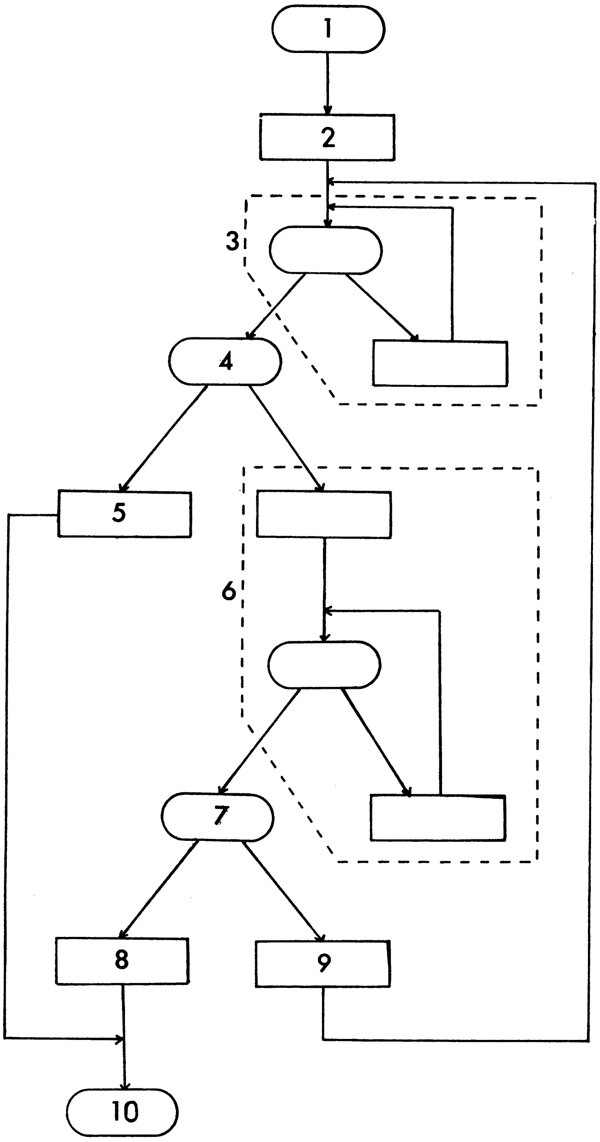
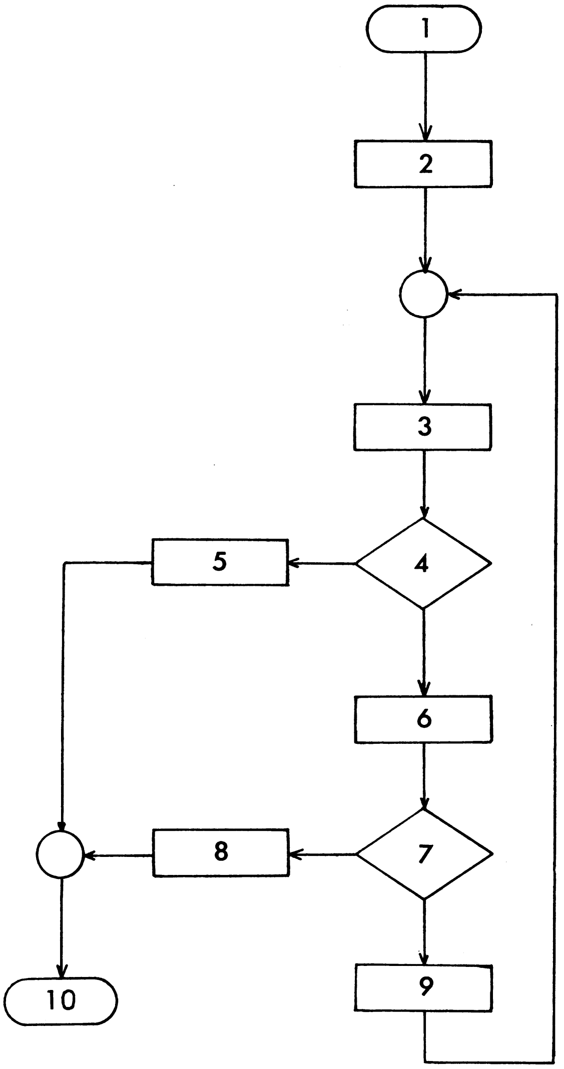


Fig. 4 (*a*) **Example used by Ashcroft and Manna (1972)**



(*b*) **The same example after reduction and rearrangement—an obvious example of a loop with multiple exit points**

The two paths of the selection process shown in Fig. 1(*b*) (labelled *A* and *B* in the diagram) are referred to as *selection paths.*

The process of *reducing* a flow diagram consists of replacing each occurrence of a base diagram (i.e. one of the three shown in Fig. 1) within the flow diagram by a single process box (Wulf, 1972). This is repeated until no further replacement is possible.

A flow diagram which reduces to a single process box (together with one START box and one FINISH box) is a structured flow diagram.

A further process, called *compression*, involves reducing a flow diagram and then, if any loops remain (i.e. loops which are not of the form given in Fig. 1(*c*)), replacing them by single boxes with the appropriate entry and exit lines (as shown in **Fig. 9**).

Theorem 1:
A flow diagram which contains any of the five structures in Fig. 2 is unstructured.

Proof:
By inspection it is obvious that a flow diagram which contains any of these five structures cannot be reduced to a single process box.

Theorem 2:
Given transformations to convert each of these five structures to structured form, then any flow diagram can be transformed to a structured flow diagram.

Proof:
Consider any line connecting two boxes in a *reduced* unstructured flow diagram. It can either lie within at least one loop, or it can lie outside any loops.

Case I. Outside any loops
If the line lies outside any loops, consider the set of origin paths for this line in the *compressed* flow diagram. There are two possibilities: either there is a single origin path for this line in the compressed flow diagram, or there is more than one such path.

1. *Single origin path*
This corresponds to a normal structured flow diagram path. For example, see **Fig. 10**(*a*).

2. *More than one origin path*
Since there is more than one origin path in the compressed diagram, there must be either:

(*a*) at least one decision box for which both selection paths lead

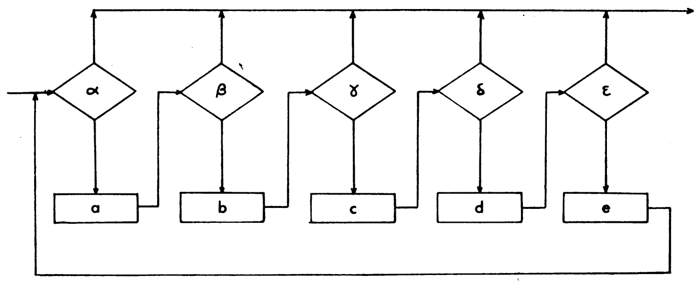
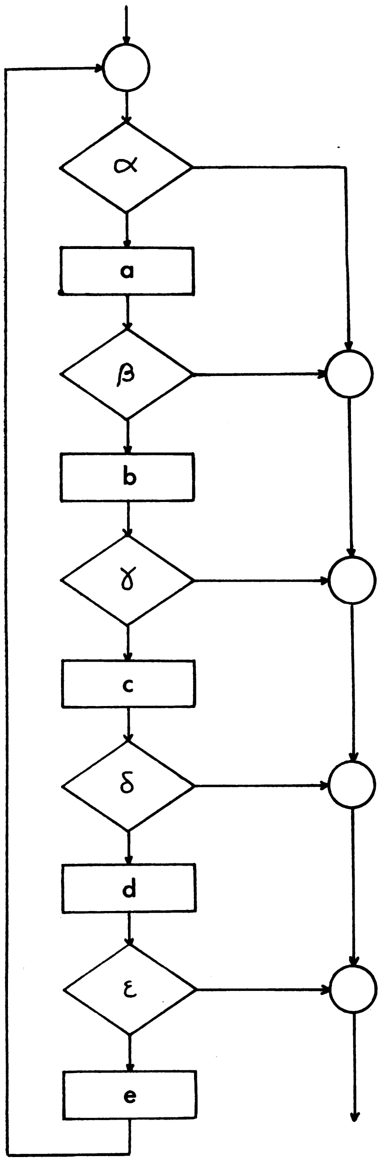


Fig. 5 (*a*) Example (Ω_5) from paper by Böhm and Jacopini (1966)



(*b*) The same flow diagram, rearranged slightly, is obviously a loop with five exit points

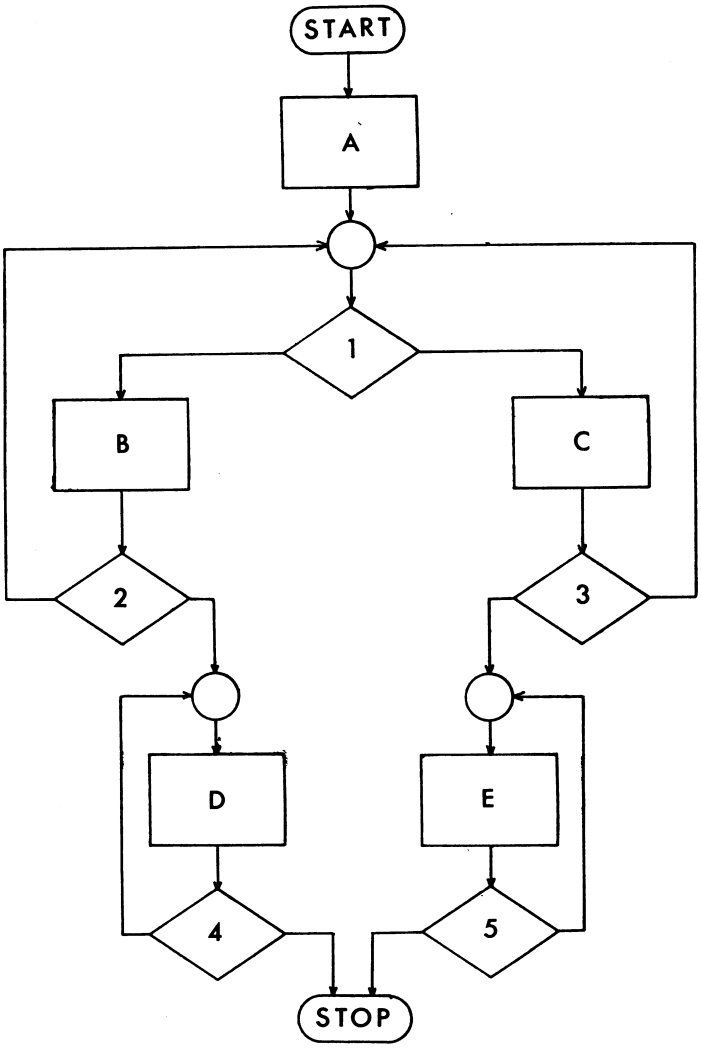
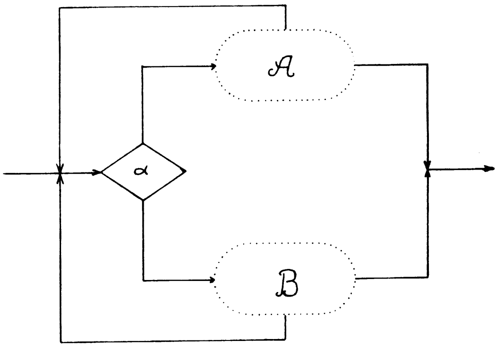


Fig. 6 (*a*) Example taken from Mullins et al. (1974)



(*b*) Similar example (Type II) from Böhm and Jacopini (1966)

to the line under consideration, or

(*b*) at least one loop-replacement box for which at least two of the exit paths lead to the line under consideration.

In the first case, since the two selection paths of the decision box lead to the same line, the two paths must join at a junction box, forming a closed circuit. However, since it has not been replaced by a single box in the reduction process, at least one of the selection paths must include some peculiarity. This could either be a branch out of the selection path which does not return (see Fig. 10(*b*)) or a branch into the selection path from outside.

In the case of a branch into the selection path from outside (labelled X in Fig. 10(*c*)), there must exist a second decision box (2) at a lower level such that the first decision box (1) lies on one of its selection paths and the line labelled X lies on the other. The result, shown in Fig. 10(*c*), is identical to the structure shown in Fig. 10(*b*).

The remaining possibility is that the line under consideration lies on two of the exit paths emanating from a loop-replacement box. If this is the case, one can replace the loop-replacement box by the original loop. Then there must exist some decision box within the loop such that the two exit paths concerned lie on its two selection paths. Hence a decision box has been found for which the two selection paths both lead to the line

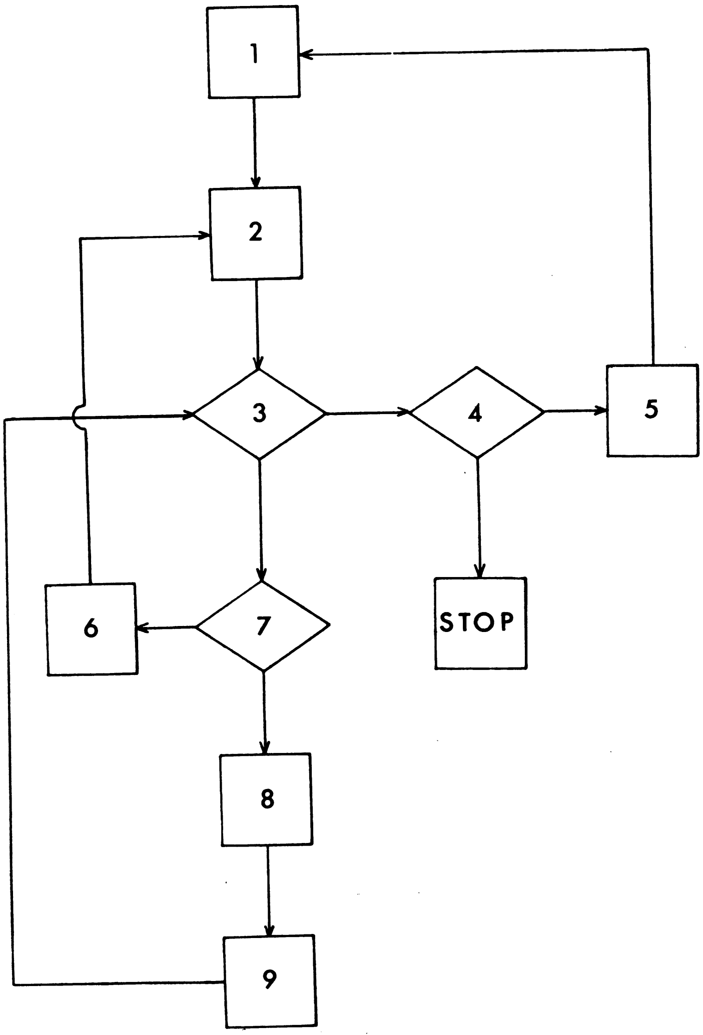


Fig. 7 Example from Yourdon (1974) which illustrates overlapping loops

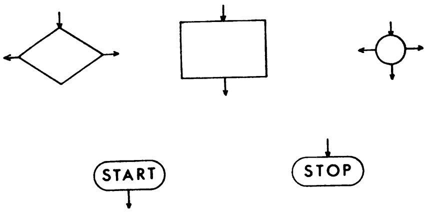


Fig. 8 The five different types of flowchart boxes

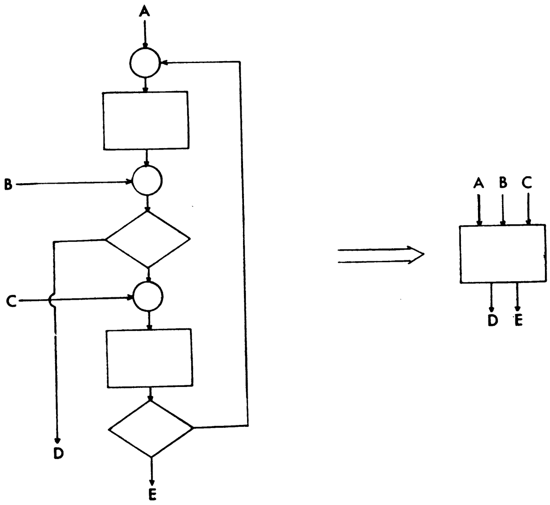


Fig. 9 The process of compression replaces all loops by single boxes. This illustrates how a loop with three entry points and two exit points is replaced by a single box

under consideration, and the argument of the previous two paragraphs applies.

Thus if there is more than one origin path to a line (which is outside any loop) in the compressed diagram, this is always due to an abnormal selection path as shown in Fig. 2(*a*). Hence if a transformation exists which converts the structure in Fig. 2(*a*) to structured form, then any portion of a flow diagram which is outside any loop can be converted to structured form by such a transformation. The simplest transformation for this structure is to duplicate boxes. Alternatively one may introduce boolean variables (Wulf).

Case II. Within a loop

A structured loop (Fig. 1(*c*)) should consist of one process box, one junction box and one two-way decision box, arranged in a closed circuit with one entry point and one exit point. Any loop which has not been removed in the reduction process, does not have this form. This can only be due to one or more of the following reasons:

(*a*) the loop has more than one entry point

(*b*) the loop has more than one exit point

(*c*) the loop contains a structure with an abnormal selection path of the type dealt with in Case I, or

(*d*) the loop contains an inner loop with one of these four structures,

for if the loop has only one entry point and one exit point and does not contain any structure with an abnormal selection path or any unreducible loop structures, then the contents of the loop excluding the junction box and the decision box leading to the exit point must be reducible to a single process box.

Case (*c*) has already been dealt with and the method for normalising this to structured form has been mentioned. Thus if one considers the innermost unreducible loop, only cases (*a*) and (*b*) remain.

1. *More than one entry point*

Since a loop should have only one entry point, choose one of the entry points as the main entry point.* Now compress the diagram so that the loop is replaced by a single box (as in Fig. 9). For each entry point besides the main one, there are two possibilities:

(*a*) There exists a decision box at a lower level than the main entry point such that the aforementioned entry point lies on one of its selection paths and the main entry point lies on the other (see **Fig. 11**(*a*)). This corresponds to the category of 'loops with multiple entry points' (Fig. 2(*c*)).

(*b*) There exists a decision box at a higher level having the loop-replacement box on one of its origin paths and the entry point in question on one of its selection paths (see Fig. 11(*b*)). This corresponds to the case of overlapping loops (Fig. 2(*d*)).

By applying transformations for Figure 2(*c*) and (*d*) to each entry point besides the main one, the loop will be replaced by one with a single entry point.

2. *More than one exit point*

As for the case with more than one entry point, choose one exit point as the main exit point and compress the diagram, replacing the loop by a single box. In this case, for each exit point besides the main one, one can distinguish between three possibilities:

*In general this process of choosing the main entry point may have to be defined more rigorously. For the purpose of this paper, however, this process will be assumed to be defined.

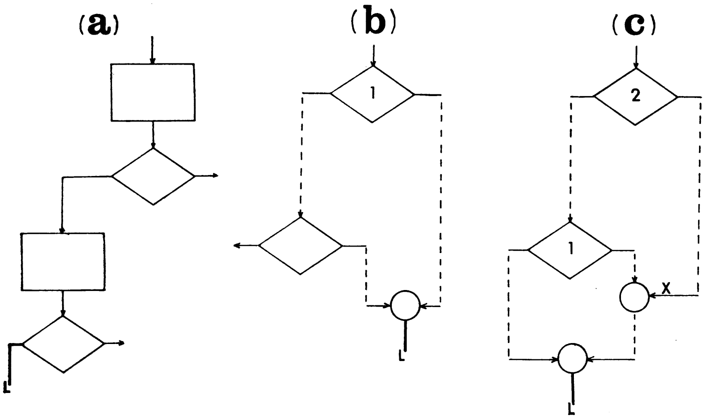


Fig. 10 (*a*) **A normal structured flow diagram path**
(*b*) **A line having two origin paths in a compressed flow diagram, caused by a branch out of the selection path of box 1**
(*c*) **A line having two origin paths in a compressed flow diagram, caused by a branch into the selection path of box 1. In each case the line marked *L* is the one under consideration**

(*a*) There exists a path from the exit point under consideration to one of the entry points other than the main one. This is identical to condition (*b*) of 1. above, viz. an example of overlapping loops (Fig. 2(*d*)).

(*b*) There exists a path from the exit point under consideration to the main entry point. If the path does not pass through any boxes other than junction boxes at a lower level than the main entry point, this corresponds to the case of parallel loops (Fig. 2(*e*)). Otherwise this corresponds to the case of overlapping loops (Fig. 2(*d*)).

(*c*) No path exists from the exit point in question to any of the entry points. This corresponds to the case of a loop with multiple exit points (Fig. 2(*b*)).

By applying transformations for Figures 2(*b*), 2(*d*) and 2(*e*) to each exit point besides the main one, the loop will be replaced by one with a single exit point.

In the case of the main exit point, there are also three possibilities.

(*a*) A path exists from the main exit point to the main entry point. This corresponds to a normal structured flow diagram nested loop.

(*b*) No path exists from the main exit point to any of the entry points. This corresponds to a normal structured flow diagram without an enclosing loop.

(*c*) A path exists from the main exit point to one of the entry points other than the main one. Such a path may or may not pass through a decision box. If the path does pass through at least one decision box, as shown in Fig. 11(*b*), then the resulting structure is that of overlapping loops (see Fig. 2(*d*)). This case has already been taken into account. However, if the path does not pass through any decision boxes, then this cannot be the main exit point as both paths lead directly back into the loop.

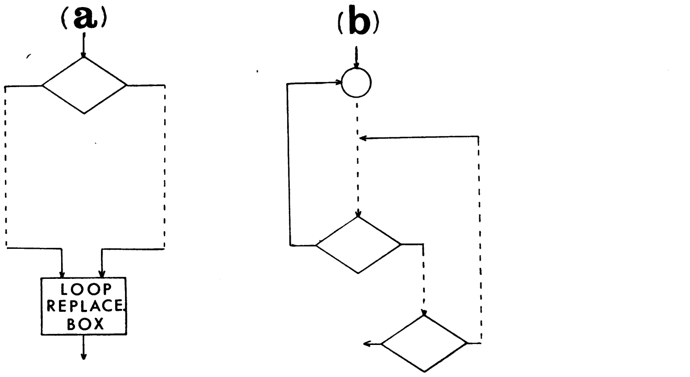


Fig. 11 (*a*) **Case when main entry point and secondary entry point lie on opposite selection paths of a decision box at a lower level**
(*b*) **Case when exit point leads to a decision box at a higher level, which in turn has a secondary entry point on one of its selection paths**

Hence, if transformations exist to convert each of the five structures in Fig. 2 to structured form, then any loop with an arbitrary number of entry and exit points can be converted to one with only one entry point and one exit point. If the boxes within the loop can still not be reduced to a single process box then the loop must either contain an inner unstructured loop or a structure of the type shown in Fig. 2(*a*). In each case by repeated application of the transformations any loop can be converted to structured form.

Thus any flow diagram can be converted to structured form by transformations on these five structures.

Hence the only possible structures which give rise to unstructured flow diagrams are those shown in Fig. 2.

4. Conclusions

There are five basic structures which lead to unstructured flow diagrams, viz.

(*a*) abnormal selection path

(*b*) loop with multiple exit points

(*c*) loop with multiple entry points

(*d*) overlapping loops, and

(*e*) parallel loops.

Examples of unstructured flow diagrams considered by different workers are shown to contain one or more of these basic structures. Thus any method for automatically converting unstructured flow diagrams into equivalent structured forms should be tested on each of these five basic structures.

Acknowledgements

This work was done while on study leave at Cambridge University Computer Laboratory on a British Council Scholarship. The author is grateful to Dr R. M. Needham of the Computer Laboratory for reading and commenting on the paper.

References

Ashcroft, E. and Manna, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress* 71, North-Holland Publishing Co., pp. 250-255.

Baker, F. T. and Mills, H. D. (1973). Chief Programming Teams, *Datamation*, Vol. 19, No. 12, pp. 58-61.

Böhm, C. and Jacopini, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *CACM*, Vol. 9, No. 5, pp. 366-371.

Cooper, D. C. (1967). Böhm and Jacopini's Reduction of Flow Charts, *CACM*, Vol. 10, No. 8, pp. 463-473.

Ershov, A. P. (1972). Theory of Program Schemata, in *Proceedings of IFIP Congress* 71, North-Holland Publishing Co., pp. 28-45.

Miller, E. F. and Lindamood, G. E. (1973). Structured Programming: Top-down Approach, *Datamation*, Vol. 19, No. 12, pp. 55-57.

Mullins, J. M., Radue, J. E., and Webb, G. L. (1974). A Tutorial Note on Functional Programming, *Systems/Stelsels*, Vol. 4, No. 8, pp. 13-14.

Wulf, W. A. (1972). Programming without the GOTO, in *Proceedings of IFIP Congress* 71, North-Holland Publishing Co., pp. 408-413.

Yourdon, E. (1974). A Brief Look at Structured Programming, *Systems/Stelsels*, Vol. 4, No. 5, pp. 7-12.