Correspondence

Dear Sir,

Decomposition of Flowchart Schemata

We are writing with reference to the paper 'Decomposition of flowchart schemata' by Prather and Giulieri.[8]

(1) Their paper offers:
  (1.1) a survey of earlier publications on the subject of structured programming;
  (1.2) a model of program flowcharts in two forms.
    (a) a list of 'elementary operations' and program decisions;
    (b) a dominance tree;
  (1.3) a method of decomposing flowcharts.

(2) We wish to summarize criticism, detailed in Ref. 9, of these aspects of the paper, as follows:
  (2.1) the survey is incomplete: since Urschler's paper,[10] a number of other relevant papers have appeared, including those by Williams in 1977[11] (not 1974 as stated in Ref. 8 and consequently not in Urschler's 'comprehensive summary') and 1978[12] (with Ossher, criticized by Kaposi, Gillies and Cowell in Ref. 5), by Paige,[6,7] by Elgot[3] and, following these, by Cowell, Gillies and Kaposi[1,2] and Gillies, Cowell and Kaposi.[4] In view of this insufficiency, such phrases as 'in almost every instance' (section 1) and 'most flowchart restructuring algorithms' (section 5) are meaningless and the need, established by the authors, for their method, becomes questionable;
  (2.2) the model is not optimally concise because:
    (a) P & G's dominance tree cannot entirely replace the flowchart. This makes the model unwieldy in decomposition. In Ref. 9, we have given a simpler algorithm using only the flowchart list of 1.2(a);
    (b) a 'flowgraph'[2] containing only the program decisions of 1.2(a) can be used, with no information loss, if relationships between elementary operations and decisions are stored in a 'transition table. CGK[2] use this model, by which Fig. 1(b) of Ref. 8 reduces to a graph of only six nodes; consists of recursively removing subflowcharts entered (maybe more than once) at exactly one point and with a single exit. Prather and Giulieri's aim is 'a more thorough preprocessing', giving 'a canonical form, where a program's independent processes are most clearly delineated' and an advantage over 'most existing restructuring algorithms' by reducing the size of subflowcharts to be restructured. These aims are more adequately met simply by additionally unfolding subflowcharts with entries at many points to give copies of identical subflowcharts, each with a single entry point. This was demonstrated in 1978.[1,4] Used in conjunction with the matching proposed by Prather and Giulieri in section 5 of Ref. 8, preprocessing to irreducibility under strong equivalence achieved the best possible decomposition under preservation of the formal language describing the flowchart. This is far more precise, conceptually, than the 'preservation of topology' which Prather and Giulieri seek to clarify.

Yours faithfully,
A. SHAFIBEGLY-GRAY
R. W. WHITTY
Department of Electrical and Electronic Engineering,
Polytechnic of the South Bank,
Borough Road,
London SE1 0AA,
UK
April 1982

References

1. Cowell, Gillies and Kaposi, Introduction to flowgraph schemas. *Proc. CISS, March 78*, John Hopkins University, Baltimore, USA (1978).
2. Cowell, Gillies and Kaposi, Synthesis and structural analysis of abstract programs. *Computer Journal* **23** (No. 3), (1980).
3. Elgot, Structured With and Without GOTO statements. *IEEE, SE-2*, No. 1 (1979).
4. Gillies, Cowell and Kaposi, Theory of flowgraph schemas, in Ref. 1 (1978).
5. Kaposi, Gillies and Cowell, Letter to the editor. *Computer Journal* **22** (No. 3) (1979).
6. Paige, Program graphs, an algebra and their implications for programming. *IEEE, SE-1*, No. 3 (1975).
7. Paige, On partitioning program graphs. *IEEE, SE-3*, No. 6 (1977).
8. Prather and Giulieri, Decomposition of flowchart schemata. *Computer Journal* **24** (No. 3) (1981).
9. Shafibegly-Gray and Whitty, Review of recent publications on program restructuring and decomposition. Internal Report, Department of Electrical and Electronic Engineering, Polytechnic of the South Bank, London (1981).
10. Urschler, Automatic restructuring of programs. *IBM J. Res. Dev.* **19** (No. 2) (1975).
11. Williams, Generating structured flow diagrams—the nature of unstructuredness. *Computer Journal* **20** (No. 1) (1977).
12. Williams and Ossher, Conversion of unstructured flow diagrams to structured form. *Computer Journal* **21** (No. 2) (1978).

Dear Sir,

Jumping to Some Purpose

The continuing correspondence under the above heading takes as its starting-point a problem posed by Knuth,[1] who in the same paper recommends, *on efficiency grounds*, the 'sentinel' solution now reiterated by Missala and Rudnicki.[2] It can be important to tune critical parts of a program to gain efficiency in execution (usually at the expense of introducing complexity), but the issue being addressed in this correspondence is the more generally important one of comprehensibility.

While attacking Robinson's program for its failure to handle the case where array A is initially empty, Missala and Rudnicki ignore the implications of their own program in the case where A is initially *full*. If the array is of fixed length and is full, there is no 'first free element'; their program ceases to be usable as soon as the array is full, rather than (as on any reasonable assumption) at the next attempt to insert a new item. Thus, unless we adopt the unsatisfactory convention that an array of size a can accommodate a maximum of $a-1$ items, we are able to insert an item for which we cannot later search.

Reference to Wirth notwithstanding, gratuitous introduction of complexity is not a feature of good programming style. Missala and Rudnicki describe Hill's program as 'not very subtle'; we should therefore not be surprised that, of the three programs under consideration, Hill's is the only one which correctly searches both an empty array and a full array.

Yours faithfully,
JIM INGLIS
Birkbeck College,
University of London,
Malet Street,
London WC1E 7HX,
UK
June 1982

References

1. D. E. Knuth, Structured programming with GOTO statements, Stanford University Computer Science Department Memo No. STAN-CS-74-416 (1974).
2. M. Missala and P. Rudnicki, Jumping to some purpose (letter). *Computer Journal* **25** (No. 2), 286 (1982).