RSRE
MEMORANDUM No. 4158

ROYAL SIGNALS & RADAR ESTABLISHMENT

COUNTING PATHS IN COMPUTER PROGRAMS

Author: B D Bramson

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

RSRE MEMORANDUM No. 4158

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 4158

COUNTING PATHS IN COMPUTER PROGRAMS

B D Bramson

May, 1988

Abstract

An algebraic method is presented for assessing the numbers of paths through loop-free portions of computer programs, thus providing a useful measure of complexity. The underlying program model is that of a directed graph, though this does not exclude recursion. The algebraic structure is that of a path algebra but with one axiom relaxed. The theory has been implemented within the Malvern Program Analysis Suite (MALPAS). Experience has shown it to be of particular value in guiding the analysis of large programs.

Contents

1 INTRODUCTION 1

2 THE PATH ASSESSOR 2

3 CONCLUSION 3

1 INTRODUCTION

A recent report [1] summarises the Malvern Program Analysis Suite, MALPAS, a set of tools for the assessment and verification of software that may be used throughout development or post facto. Two of the analysers reveal the flows of control and information through a given program. The resources required for each of these are polynomial functions of the size of the program in terms of numbers of statements and declarations of data. A further tool, the Semantic Analyser, executes the program symbolically and provides a description of each loop-free program path, loops being exercised precisely once. However, semantic analysis is non-polynomial in nature. In particular, the number of paths through a loop-free procedure can depend exponentially on the number of conditional statements.

```
TYPE anytype;
FUNCTION p(integer, anytype): boolean;
FUNCTION f(integer, anytype): anytype;
FUNCTION g(integer, anytype): anytype;

PROCSPEC iffy(INOUT x: anytype);

PROC iffy;
  IF p(1,x) THEN x := f(1,x) ELSE x := g(1,x) ENDIF;
  IF p(2,x) THEN x := f(2,x) ELSE x := g(2,x) ENDIF;
  ..................................................
  IF p(n,x) THEN x := f(n,x) ELSE x := g(n,x) ENDIF
ENDPROC
FINISH
```

Figure 1: The procedure "iffy" has n conditional statements in series and 2^n potential paths.

Given the path-condition,

```
NOT p(1,x) AND p(2, g(1,x)) AND p(3, f(2, g(1,x))) AND
.....................................................
....p(n, f(n-1, f(n-2, ..... f(2, g(1,x))))..........),
```

the corresponding action is

```
x := f(n, f(n-1, f(n-2, ..... f(2, g(1,x))))........).
```

Figure 2: A typical path-condition for the procedure "iffy" and the corresponding input-output relation.

Figure 1, expressed in MALPAS Intermediate Language (IL) [2,3] illustrates the problem. The semantic analysis of the procedure "iffy" yields a path-condition and consequent

action for each path. For example, the path for which all the p s are true on execution, except for the first, is described algebraically in figure 2.

It may be seen that the numbers of ASCII characters contained in the path-condition and action depicted in figure 2 are respectively quadratic and linear in n. Further, there are 2^n program paths. So, ignoring possible algebraic simplifications, the volume of information presented to the analyst is of order $n^2\ 2^n$. Thus, there will be programs for which the unfettered application of semantic analysis is doomed to failure. No matter how big or how fast the computer, n can always be increased to defeat it.

Two techniques have been implemented to circumvent the problem just described and each involves processing a given program prior to semantic analysis. The Partial Programmer [4,5], given a subset of program variables nominated by the user, generates a new program dedicated to their specific calculation by removing irrelevant conditional statements. Reducing n by 1 in figure 1 halves the volume of output presented to the analyst. In practice, more dramatic improvements have been experienced.

Node marking [6], on the other hand, involves partitioning the program by fixing key way-points for retention. Semantic analysis then reveals the behaviour of the program between the retained nodes. A choice of dominator, P, for example, would generate expressions like those depicted in figure 2 for the (symbolic) execution from the start to P and from P to the end, effectively re-introducing an element of sequential logic. If P partitions the program into pieces that each contain $n/2$ conditionals, the volume of output decreases by a factor $2^{1+n/2}$.

In the light of experience, it became clear that an estimate of the number of program paths would be essential if the analysis of large programs were to be tackled. The purpose of the following section is therefore to outline the theory behind the MALPAS Path Assessor.

2 THE PATH ASSESSOR

Consider, first, an arbitrary loop-free procedure, with a single start and single end, that may or may not call other procedures including itself. Suppose the procedure to be modelled as a labelled, directed graph G whose nodes correspond to simple statements or procedure calls and whose arcs are labelled with the distinct elements of an abstract alphabet A. The details of the map from the procedure to the graph need not concern us. The point is that the number of paths through the graph indicates the number of syntactically possible paths through the procedure without expanding any procedure calls. [1]

By employing node reduction [7,8], the graph G may be transformed into a regular expression in A [9] that involves only the operations $\times$ (sequence) and $+$ (alternation). Owing to the loop-free assumption, $*$ is omitted. Let the set of such restricted expressions be $R(A)$.

Separately, consider the set N of natural numbers {0,1,2...}, closed under the arithmetic operations $\times$ and $+$. [2] Then the algebra $\{N, \times, +\}$ satisfies all but one of the axioms of a path algebra [10], $+$ not being idempotent. (For completeness, when applied to N: $+$

[1] In MALPAS IL, the specifications of procedures are called rather than their bodies.

[2] Note that $\times$ and $+$ are defined for both regular expressions and integers.

is commutative and associative; $\times$ is associative and left and right distributive over $+$. There is a multiplicative identity, namely 1, and an element, namely 0, that is both an additive identity and a two-sided multiplicative zero.)

Next, we set up a map, $PATHS$, from $R(A)$ to N, first by assigning 1 to each arc in G and thereafter by natural extension. Thus, for each a in A and for each u, v in $R(A)$:

$$\begin{aligned} PATHS(a) &= 1 \\ PATHS(u \times v) &= PATHS(u) \times PATHS(v) \\ PATHS(u + v) &= PATHS(u) + PATHS(v) \end{aligned} \tag{1}$$

Manifestly, the axioms 1 are sufficient to guarantee that the result of applying $PATHS$ to any restricted regular expression is independent of the sequence of evaluation, modulo standard conventions on removing brackets. Furthermore, the map is useful in so far as it calculates in polynomial time [3] the number of paths through G.

Note, however, that $PATHS$ is not a homomorphism from the restricted algebra of languages over A into N. An element of the former is a set of words with $+$ being set union and idempotent. The axiom of idempotency is not preserved by the map. Nevertheless, because G's arc labels are distinct, sub-expressions of the form $s + s$ never appear.

Finally, for programs with loops, the analysis above may be applied to each loop-free portion of each procedure. One approach is to use node reduction to generate regular expressions while preserving nodes with self-loops as they are encountered.

3 CONCLUSION

The technique of path assessment just described was implemented by RSRE and added to the Malvern Program Analysis Suite where it is now supported commercially. The MALPAS Path Assessor has been used in many real applications. It provides a simple but accurate assessment of the structural complexity of software and can indicate where practical problems may arise in attempting to run the Semantic or Compliance Analysers. [11]

It is interesting to note that the algebra on which path assessment is based is not a path algebra; which seems to be an example of a more general phenomenon [12].

Finally, it gives me great pleasure to thank: J-A Fernandez for his efficient and accurate implementation of the ideas presented in this paper; Dr J M Foster for a correspondence relating to the relaxation of algebraic axioms; and H C Williams for suggestions that have added clarity to the text.

References

[1] Tools for the specification, design, analysis and verification of software:
B D Bramson, RSRE report 87005, 1987.

[3] linear for a "while-structured" program

[2] MALPAS Intermediate Language (Version 3.2):
R J Granville & C M O'Halloran, RSRE memorandum 3731, 1986.

[3] MALPAS Intermediate Language Manual, version 4.0:
Rex, Thompson & Partners, 1987.

[4] An exercise in partial programming: B D Bramson, RSRE report 85002, 1985.

[5] Automatic extraction of partial programs:
R J Granville & C M O'Halloran, RSRE memorandum 3736, 1986.

[6] A note on node marking: B D Bramson, RSRE memorandum 3503, 1982.

[7] Graph theory leads to program visibility
B D Bramson & S J Goodenough, RSRE report 80004, 1980.

[8] The mathematical foundations of node reduction:
B D Bramson, RSRE report 81014, 1981.

[9] Data use analysis for computer programs:
B D Bramson & S J Goodenough, RSRE report 82001, 1982.

[10] Graphs and networks: B A Carré, Oxford University Press, 1979.

[11] MALPAS User Guide, release 4.1: Rex, Thompson & Partners, 1988.

[12] Algebraic specification of a target machine, Ten15:
J M Foster, article in High Integrity Software,
ed C T Sennett, Pitman Publishing, to appear.

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	MEMO 4158		U/C

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
778400	ROYAL SIGNALS & RADAR ESTABLISHMENT ST ANDREWS ROAD, GREAT MALVERN, WORCESTERSHIRE, WR14 3PS

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title Counting paths in computer programs

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
BRAMSON B D			1989.5	4

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement

UNLIMITED

Descriptors (or keywords)

continue on separate piece of paper

Abstract An algebraic method is presented for assessing the numbers of paths through loopfree portions of computer programs, thus providing a useful measure of complexity. The underlying program model is that of a directed graph, though this does not exclude recursion. The algebraic structure is that of a path algebra but with one axiom relaxed. The theory has been implemented within the Malvern Program Analysis Suite (MALPAS). Experience has shown it to be of particular value in guiding the analysis of large programs.