Program slicing has many applications including program understanding, debugging, testing, maintenance, complexity measurement, software reengineering etc. The static slice is the set of all statements that might affect the value of a given variable at a particular statement. We propose a novel static slicing algorithm for intra-procedural programs. We use the program dependence graph (PDG) as the intermediate program representation. We store the PDG in the form of an adjacency matrix. We have implemented our algorithm by using file structures to find out all the statements that are included in the static slice.

1. Introduction

Program slicing is a program analysis technique [1][2]. Program slicing is a decomposition technique that extracts statements from a program relevant to a particular computation [1]. The main application of program slicing are debugging [3] and testing [4] of softwares. The other applications of program slicing includes various software engineering activities such as program understanding, software maintenance, complexity measurement, software reengineering etc. [5][2]. It can also be used to extract the statements of a program that are relevant to a given computation. A program slice consists of the parts or components of a program that (potentially) affect the values computed at some point of interest. Program slices are computed with respect to a slicing criterion. Typically, a slicing criterion consists of a pair \(< s, V >\) where \(s\) is a statement number and \(V\) is a variable at statement \(s\). A static slice of a program \(P\) with respect to a slicing criterion \(< s, V >\) is the set of all the statements of the program \(P\) that might affect the slicing criterion for every possible input to the program [6].

Consider the example program given in Figure 1 to illustrate the concept of program slicing. The static slice of the example program given in Figure 1 with respect to the slicing criterion \(< 12, \text{count} >\) is shown in Figure 2. Several categories of program slicing as well as methods to compute them are found in literature [7][8]. The main reason for the existence of so many categories of slicing is the fact that different applications require different types of slices. Slices can be backward or forward, static or dynamic, intra-procedural or inter-procedural. In a recently reported form of slicing called amorphous slicing, slices are not necessarily produced by deleting statements and may not necessarily even be made from components of the original program being sliced.

A static slice [1] contains all statements that may affect the value of a variable at a program point for every possible execution. But a dynamic slice [10] contains all statements that actually affect the value of a variable at a program point for that particular execution. Thus, the size of dynamic slices are much smaller than that of the static slices.

Intra-procedural slicing [6] computes slices within one procedure. Calls to other procedures are either not handled at all or handled conservatively. If the program consists of more than one procedure, inter-procedural slicing [9][4] can be used to derive slices that span multiple procedures. In this paper, we propose a novel static slicing algorithm for intra-procedural programs.
structures to handle all the statements that compute the static slice. The time complexity of our algorithm is $O(Vn^2)$, where $V$ is the number of variables and $n$ is the number of statements in the program. Currently, we are working on static slicing of inter-procedural programs and object-oriented programs.

Figure 6. Contents of file "pdg.txt"

Figure 7. PDG of the sample program given in Figure 3

Figure 8. Static slice of the program given in Figure 3 w.r.t. slicing criterion $< 15, a >$

Figure 9. PDG containing the static slice of the program given in Figure 3 w.r.t. slicing criterion $< 15, a >$

REFERENCES


**Durga Prasad Mohapatra**

received his Ph.D from Indian Institute of Technology Kharagpur and M.E. from Regional Engineering College (now NIT), Rourkela. He joined the faculty of the Department of Computer Science and Engineering at the National Institute of Technology, Rourkela in 1996, where he is now Assistant Professor. His research interests include discrete mathematics, software engineering, real-time systems and distributed computing. He has published more than thirty papers in these fields. Dr. Mohapatra has been teaching discrete mathematics at NIT Rourkela for the past ten years. He has received Young Scientist Award for the year 2006 by Orissa Bigyan Academy. Currently, he is a member of IEEE. Dr. Mohapatra has co-authored the book *Elements of Discrete Mathematics: A computer Oriented Approach* published by Tata McGraw Hill.