Abstract

Program slicing is a program analysis technique. It can be used to extract the statements of a program that are relevant to a given computation. Some of the major applications of program slicing include program understanding, debugging, testing, program comprehension, program maintenance and modification, reusable component generation, parallelization, dead code elimination etc.

A survey of the dynamic slicing techniques shows that the existing dynamic slicing techniques are not very efficient. We have the following observations: (i) the space complexity and time complexity (the time to compute a slice after execution of a statement of interest) of each of the existing intraprocedural dynamic slicing algorithms for structured sequential programs are either exponential in the number of statements in the program or unbounded, (ii) The space complexity and time complexity of each of the existing intraprocedural dynamic slicing algorithms for unstructured sequential programs are unbounded, (iii) The space complexity and time complexity of each of the existing interprocedural dynamic slicing algorithms are unbounded, (iv)the existing dynamic slicing algorithms for parallel, distributed and object-oriented programs are extensions of these basic algorithms for sequential programs and thus, display similar characteristics.

The high space and time requirements of the existing dynamic slicing algorithms make them unsuitable for use in interactive applications such as program debugging and testing. Therefore, there is a pressing necessity to devise efficient dynamic slicing algorithms. With this moti-

vation, the thesis attempts to develop suitable frameworks and efficient dynamic slicing algorithms for sequential programs.

We first introduce the concepts of *stable* and *unstable* edges, and present an efficient intraprocedural dynamic slicing algorithm called the *edge-marking* algorithm. The space complexity and time complexity of the edge-marking algorithm are quadratic in the number of statements in the program. Then we explore the possibility of reducing the time complexity and present two efficient intraprocedural dynamic slicing algorithms called *refined-edge marking* algorithm and *node-marking* algorithm. The refined-edge marking and node-marking algorithms have space complexity quadratic and time complexity linear in the number of statements. We show that these algorithms are increasingly efficient in the order of their presentation in the thesis.

We extend our framework to handle unstructured programs, and introduce the concept of *Unstructured Program Dependence Graph.* Then we present an efficient dynamic slicing algorithm for unstructured programs using the $UPDG$ as the intermediate program representation. The space complexity and time complexity of the algorithm are quadratic and linear in the number of statements in the program respectively.

For interprocedural dynamic slicing we develop a suitable intermediate representation, and then present the dynamic slicing algorithm. We have named the algorithm as $InterSlice$ algorithm. The $InterSlice$ algorithm has space complexity quadratic and time complexity linear in the number of statements in the program. We also discuss how our

slicing algorithms can be extended to efficiently handle arrays, structures, pointers and recursive procedure calls.

We show that our dynamic slicing algorithms compute precise dynamic slices, and are more efficient than the existing algorithms. We implement our dynamic slicing algorithms to experimentally verify their correctness and preciseness.