# CONTENTS

1 Introduction 1

1.1 Categories of Program Slicing 4
  1.1.1 Static and Dynamic Slicing 4
  1.1.2 Backward and Forward Slicing 5
  1.1.3 Intra-procedural and Inter-procedural Slicing 5
  1.1.4 Slicing of Structured vs., Unstructured Programs 6
  1.1.5 Other Slicing Categories 6

1.2 Applications of Program Slicing 8
  1.2.1 Program Debugging 8
  1.2.2 Program Differencing 10
  1.2.3 Software Maintenance 10
  1.2.4 Testing 12
  1.2.5 Parallelization 13
  1.2.6 Code Reusability 14
  1.2.7 Program Integration 14
  1.2.8 Functional Cohesion 15
  1.2.9 Software Quality Assurance 16
  1.2.10 Software Portability 16
  1.2.11 Other Applications of Program Slicing 17

1.3 Basic Object-Oriented Programming Concepts 17

1.4 Object-Oriented Program Representation 19

1.5 Multi-threading 20
  1.5.1 Threaded Objects 22
  1.5.2 Understanding and Debugging Threads 24

1.6 Major Goals of our Work 26

1.7 Organization of this Thesis 27

2 Basic Concepts in Program Slicing 30
### 2.1 Introduction

Overview of Slicing

2.2.1 What is Slicing?

2.3 Types of Slicing

2.4 Static Slicing

2.5 Dynamic Slicing

2.6 Intermediate of Program Slicing---Program Dependency Graph (PDG)

2.7 Slicing from PDG

2.8 OPDG (Object-Oriented Program Dependency Graph)

2.9 Comparison of Different Representations

2.9.1 Call Graph

2.9.2 Program Dependence Graph

2.9.3 System Dependence Graph

2.9.4 OPDG

2.10 Summary

### 3 Previous Work

3.1 Work Done on Slicing

3.1.1 Weiser Slicing

3.1.2 Weiser's Algorithm

3.1.3 Slicing with Program Dependency Graph

3.1.4 Comparison Between Weiser's Method and PDG Based Methods

3.2. Inter-procedural Slicing: System Dependence Graph

3.3. Slicing in the presence of Unstructured Control Flow

3.4. Slicing in the presence of Composite Data Types and Pointers

3.5. Dynamic Slicing

3.6. Slicing of Concurrent Programs

3.6.1 Static Slicing

3.6.2 Dynamic Slicing

3.7 Some Variants of Slicing Methods

3.7.1 Slicing with Relational Equations

3.7.2 Quasi-Static and Conditioned Slicing
9 Reverse Execution of Programs and Bi-directional Debugging

9.1 Related Work

9.2 Review Basic of Debugging Approaches
9.2.1 Output Debugging
9.2.2 Tracing
9.2.3 Breakpoints
9.2.4 Assertion Execution
9.2.5 Controlled Execution
9.2.6 Replay
9.2.7 Monitoring

9.3 Reverse Execution of Programs
9.3.1 Handling sequential Programs
9.3.2 Sequence Statements
9.3.3 Scope Handling
9.3.4 Selection Statement
9.3.5 Iterative Statements
9.3.6 Handling Distributed Programs
9.3.7 Receive Statements
9.3.8 Send Statements

9.4 Handling Object-Oriented Programs
9.4.1 Classes
9.4.2 Inheritance
9.4.3 Method Invocation
9.4.4 Object Instalation
9.4.5 Polymorphism

9.5 Comparative Study with the Unix Deboger
9.5.1 Debugging on Unix

9.6 The Bi-directional Debugger

9.7 Summary
10 Design and Implementation for Bi-directional Debugger

10.1 Introduction

10.2 Design for BDb for Sequential Programs

10.3 Design for BDb for Distributed Programs

10.4 Implementation

10.5 Organization of the Software

10.5.1 The Package bdd.distributedC.gui

10.5.2 The Package bdd.distributedC.parser

10.5.3 The Package bdd.distributedC.interpreter

10.5.4 The Package bdd.distributedC.main

10.5.5 The Package bdd.oops.gui

10.5.6 The Package bdd.oop.parser

10.5.7 The Package bdd.oops.interpreter

10.5.8 The Package bdd.oops.runtime

10.5.9 Interpreter and Parser

10.6 Performance Studies

10.6.1 Results for the C Program Debugging

10.6.2 Analysis for the Results for Sequential Programs

10.6.3 Results for the OOP Debugging

10.6.4 Analysis of the Results for 0-0 Programs

10.6.5 The Performance Metric

10.7 Summary

11 Conclusion and Future Work

11.1 Scope and Future Work

12 Bibliography