IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727,
PP 37-44
www.iosrjournals.org

Generating &Prioritization of test cases using Static Program Slices

Ms. JyotiArora, Ms, RadhikaThapar
Rukmini Devi Institute of Advanced Studies
(2A & 2B, Phase-1, MadhubanChowk, Outer Ring Road, Rohini, Delhi-110085)
[1]jyoti.arora@rdias.ac.in,[2]radhika.thapar@rdias.ac.in

Abstract*: *Program slicing is a process to classify the program into number of parts based on various types of dependencies between program statements.This paper presents an overview of basic concept of generating static program slices and on the basis of these program slices generating and prioritization of test cases.
Keywords: *Program dependency, static program slices, test cases prioritization and execution history*

I. INTRODUCTION

Program Slicing enables programmers to view subsets of a program by filtering out code that is not relevant to the computation of interest. It was originally introduced by Mark Weiser. A slice consists of all statements of a program that might affect the value of a variable at a program point of interest for every possible input to the program. [1]

A concept of Program Slicing is introduced to aid developer debugging, testing and program comprehension by reducing the complexityof the program. With the help of these static program slices, we can easily generating and prioritizing test cases.

II. PROGRAM SLICING

Software Testing is the process of executing a program or system with the intent of finding errors. There are two methods for software testing: - Black box testing (Functional Testing) and White box testing (Structural Testing). The term ‘Black Box’ refers to the software which is treated as a black box. By treating it as a black box, we mean that the system or source code is not checked at all. It is done from customer’s viewpoint. White box testing is a way of testing the external functionality of the code by examining and testing the program code that realizes the external functionality. Full white-box testing is a large task. However, for many properties, only a small portion of the program is relevant.

Hence property-based testing has the potential to substantially simplify much of the testing work. The portion of a program that relates to a given property can be identified through slicing.

Slicing is an important technique which has a wide range of applications in software testing. Basically, slicing is a technique for simplifying programs by focusing on selected aspects of semantics. It is method of program analysis which is used to extract a set of statements in a program which is relevant for a particular computation. This set of statements is called a program slice. Various type of slicing strategies exist such as forward slicing, backward slicing, static slicing, dynamic slicing, etc. These different slicing techniques have different application domains such as software maintenance, software optimization, program analysis, information flow control, etc.

Program slicing refers to finding all statements in a program that directly or indirectly affect the value of a variable occurrence. The selected statements constitute a slice.

Depending on the run-time environment, Program Slicing classified as static slicing and dynamic slicing. In the case stacte slicing, a slice S consists of all statements in program P that may affect the value of variable v at some point p. The slice is defined for a slicing criterion C=(x, V), where x is a statement in program P and V is a subset of variables in P. A static slice preserves the program’s behavior (value of variable v) for all possible program executions.Static Slicing is used to identify those parts of the program that potentially contribute to the computation of the selected functions for all possible programs inputs. In the case of dynamic slicing, a dynamic program slice is part of a program that “affects” the computation of a variable of interest during program execution on a specific program input. This paper focus on static program slicing.

2.1 Terminology used in Static Program Slicing:-

2.1.1Program Slice:- A program slice is a subset of a program.

2.1.2Slicing Criteria :- Assume that: P is a program and V is the set of variables at a program location (line number) n. A slice S(V,n) produces the portions of the program that contribute to the value of V just before or after the statement at location n is executed.

2.1.3Slicing Variable:- Slicing variable is variable specified in the slicing criteria , used to find out directly or indirectly affected lines from the program at the slicing Point.

2.1.4Slicing Point:- Slicing Point is point from where programmer wants to slice the program that contribute to the value of slicing variable.

2.1.5Slicing Direction:- Slicing direction used to exact the slice from the program in the forward or backward manner.

2.1.6Type of the Slicing result:- The result of slicing may be equivalent to the program or few set of statements from the program.

A Program slice Must Satisfy the following Conditions:-

1. Slices created with respect to one variable not a set of variables.
2. Creating slices for all nodes/program statements whose focus is on the variables in the right hand side of an assignment.
3. Slices should be executable.

For Example Figure 1. shows C' Program

```
1. void main()

2. {

3.  inti,sum,mul;

4.  i=1;

5.  sum=0;

6.  mul=1;

7.  while(i<=10)

8.  {

9.  sum=sum+i;

10. mul=mul*i;

11. i=i+1;

12. }

13. printf("%d",i);

14. printf("%d",mul);

15. printf("%d",sum);

16. getch();

17. }
```

Fig.1 C’ Program Example

Figure-2 shows the demonstration for various program slices of program -1.

Figure 2. Shows executable program slices of all variables of program -1, every program slice includes statements from program-1 that directly or indirectly affects the value of that program slice variable.

With the help of program slices, we decompose our program -1 into number of slices containing statements that directly or indirectly affect the slice variable in program-1. Finally for slice S1, count the value of i variable, for slice S2, compute the multiplication function and for slice S3 compute the sum function.

S1(i, 13)	S2(mul,14)	S3(sum,15)
void main()	void main()	void main()
{	{	{
inti,sum,mul;	inti,sum,mul;	inti,sum,mul;
i=1;	mul=1;	sum=0;
while(i<=10)	i=1;	i=1;
{	while(i<=10)	while(i<=10)
i=i+1;	{	{
}	mul=mul*i;	sum=sum+I;
Printf(“%d”,i);	i=i+1;	i=i+1;
getch();	}	}
}	printf(“%d”,mul);	printf(“%d”,sum);
	getch();	getch();
	}	}

Fig. 2 Executable Program Slices for above C’ language Program

III. PROGRAM SLICING PROCESS

The step by step activities for generating static program slicing are:-

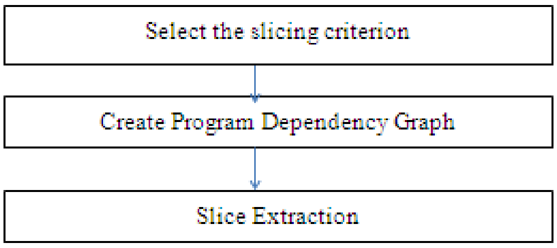
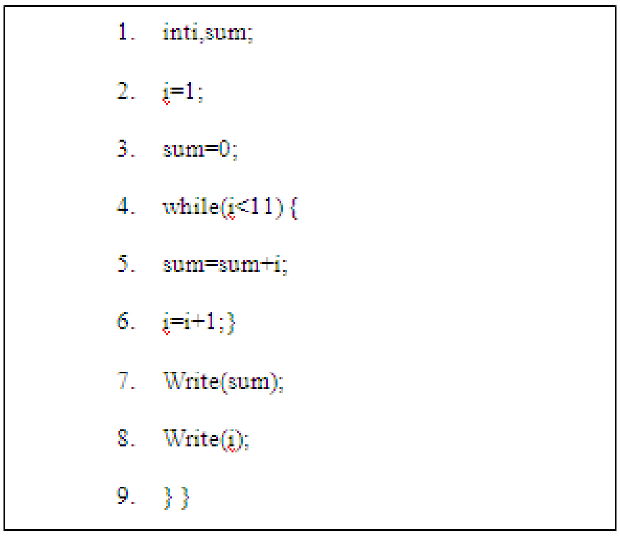


Fig. 3 Static program slicing process



```
1. inti,sum;
2. i=1;
3. sum=0;
4. while(i<11) {
5. sum=sum+i;
6. i=i+1;}
7. Write(sum);
8. Write(i);
9. } }
```

Fig. 4 Example Program

3.1.1Select the slicing criterion:- The slicing criterion specifies a point of interest (a statement in the program to be sliced) and interested variable. For static program slices, criterion is expressed as (n,V)- where n is point of interest and V is interested variable. For dynamic program slices, criterion is expressed as (value,n,V)- where value is input variable value, n is point of interest and V is interested variable. For example Slicing criterion for above program is (i,8)[2]

3.1.2Create Program Dependency Graph:- The second step for program slicing process is to create program dependency graph on the basis of various types of dependencies between the statements[4]. For example dependency can be data, control, function call etc. The Program Dependency Graph for above program is as follow:

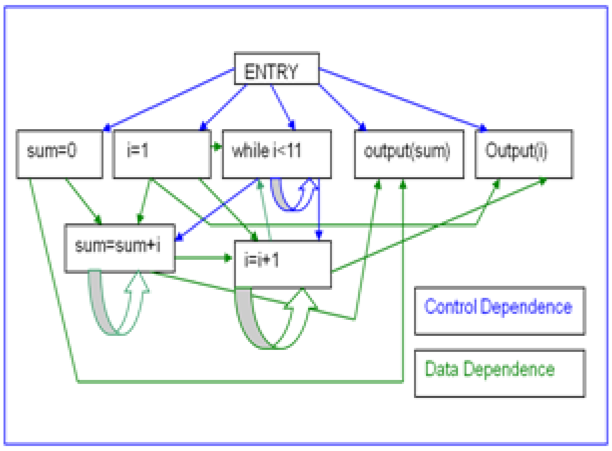


Fig. 5 Program Flow Graph

3.1.3 Slice Extraction: - After creating dependency graph, the next step is to extract slices from program dependency graph using forward or backward approaches. Backward slices contain all parts of the program that might have influenced the variable in the statement under consideration. Here the slice is computed by working backwards from the point of interest finding all statements that can affect the specified variable at the point of interest and discarding the other statements. Forward slices contain all parts of the program that might be influenced by the variable in those statements which are affected by the slicing criterion. Here the slice is computed by working forwards from the point of interest finding all statements that can affect the specified variable at the point of interest and discarding the other statements[3].

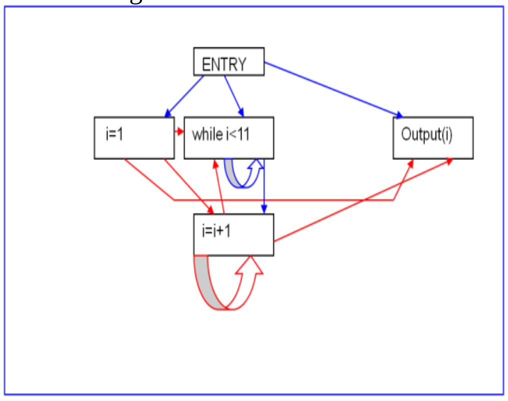


Fig. 6 Program flow graph of variable i

IV. APPLICATION OF PROGRAM SLICING

4.1 DEBUGGING: - PROGRAM SLICING IS HELPFUL IN DEBUGGING FOR:-

4.1.1 Finding errors: Execute the program by a given input. If the result is not coincident with the expected result, it indicates that there are errors in the program. Suppose there may be error when we input the value of a variable V at statement i.

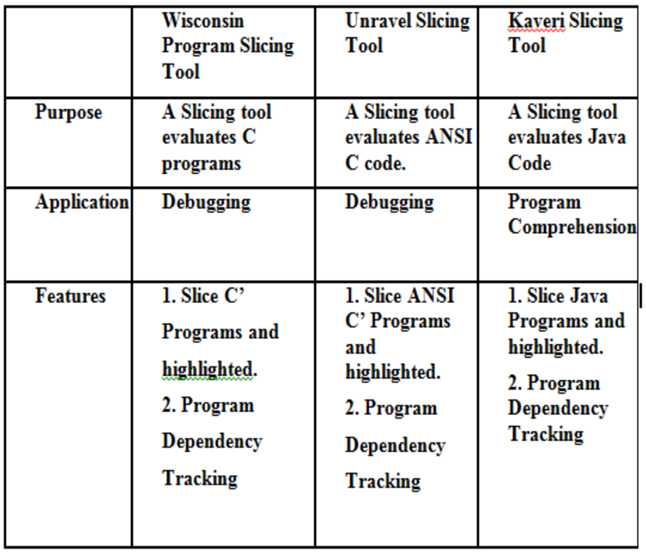
4.1.2 Localizing errors: Localizing errors is a trivial and time-consuming work in debugging. It is difficult to find errors quickly and precisely without tools. In order to find errors, insert breakpoints into the program. The purpose is to narrow the errors to the program section before the breakpoint. The i in a slicing criterion <i, V> is a breakpoint in nature and V is the variable set interesting. But program slice only includes the statements that influence the V at i, not all the statements before i. Thus, the error codes are localized to a less program section. If the slice is executable, the errors are to be localized by executing the slice, not the whole program repeatedly.

4.1.3 Analyzing ripple effort:- Program is a system with interactions among statements. The modification of an error (statement) might cause other unexpected errors. Thus when a statement is modified we must check the statements affected by the modification and modify them if necessary. The process to analyze the affected statements is called ripple analysis. Such statements can be obtained from the slicing[15].

4.2 Program Understanding:- With the help of program slices, understanding the program is more easier than full program. Each program slice having statements relevant to the specific variable.

4.2.1 Testing: - Program slicing play very important role in software testing. At the time of regression testing, it is very difficult to find out number of test cases that need to be re-run again or effected test cases due to the modification of original program. But with the help of program slices, we can easily find out effected test cases after modification of original program. Program slices contains execution statement of the program. [5] With the find out difference between executions history of program and program slices execution statements, we can find out effected and non-effected test cases after the modification of the program.

V. EXISTING PROGRAM SLICING TOOLS

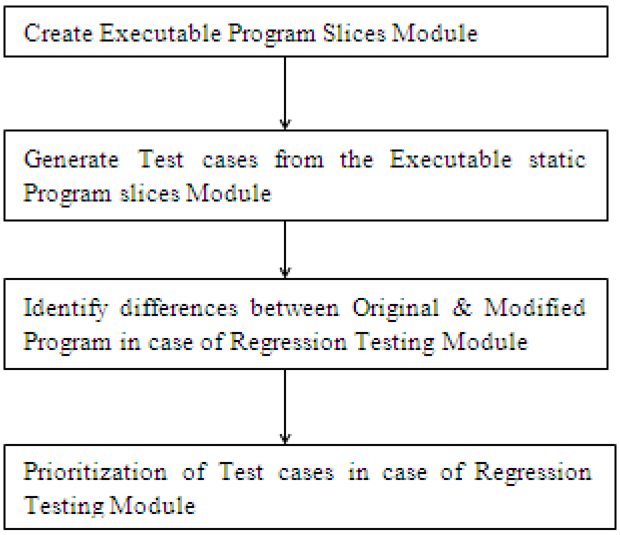


	Wisconsin Program Slicing Tool	Unravel Slicing Tool	Kaveri Slicing Tool
Purpose	A Slicing tool evaluates C programs	A Slicing tool evaluates ANSI C code.	A Slicing tool evaluates Java Code
Application	Debugging	Debugging	Program Comprehension
Features	1. Slice C' Programs and highlighted. 2. Program Dependency Tracking	1. Slice ANSI C' Programs and highlighted. 2. Program Dependency Tracking	1. Slice Java Programs and highlighted. 2. Program Dependency Tracking

Fig. 7 Comparison between static Program Slicing Tools

The above figure shows various existing tools for static program slicing.

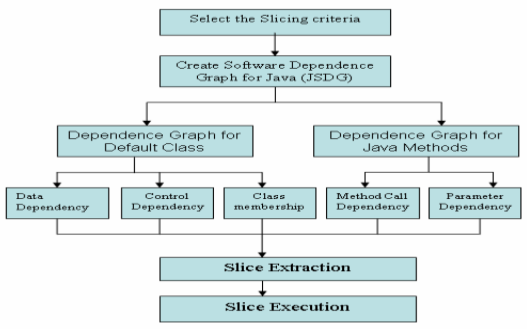
VI. PROPOSED APPROACH


Create Executable Program Slices Module
Generate Test cases from the Executable static Program slices Module
Identify differences between Original & Modified Program in case of Regression Testing Module
Prioritization of Test cases in case of Regression Testing Module
Fig. 8 Proposed Approach

6.1 Create executable Static Program Slices Module

This module includes four steps:-
1. Select the Slicing Criteria.
2. Create Software Dependence Graph.
3. Slicing Extraction
4. Slicing Execution

In the first step of this module, input the slicing criteria from user –slice variable and slicing point.

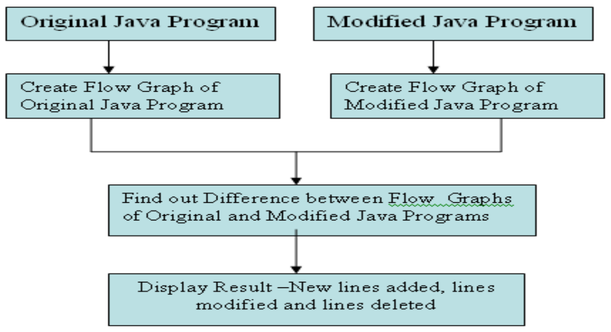

Select the Slicing criteria
Create Software Dependence Graph for Java (JSDG)
Dependence Graph for Default Class
Dependence Graph for Java Methods
Data Dependency
Control Dependency
Class membership
Method Call Dependency
Parameter Dependency
Slice Extraction
Slice Execution

In the second step of this module, create program dependence graph.

6.2 Generate Test Cases from the Executable Static Program Slices Module:- The executable static program slices contained line numbers of original program. On the basis of line numbers of individual executable static program slices generate test cases and maintained its execution history.

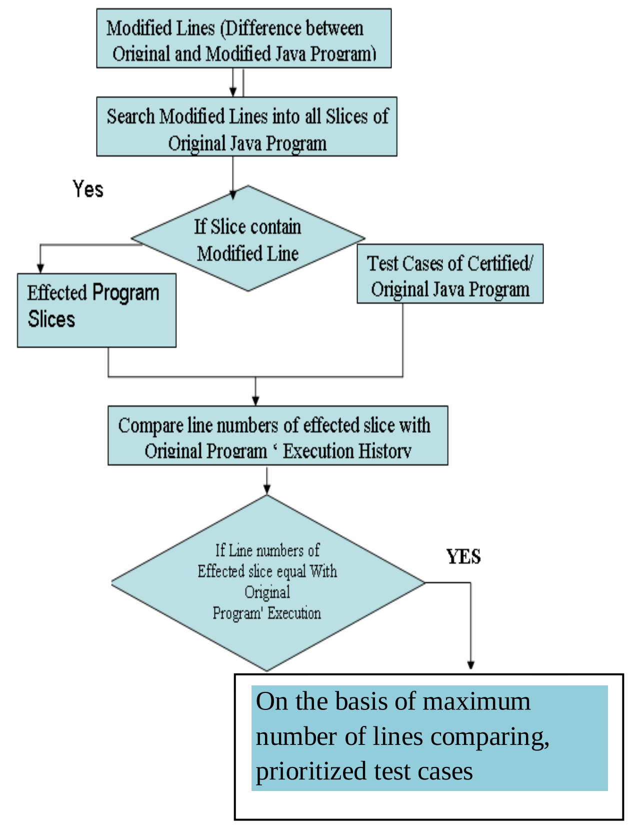
6.3 Identify differences between Original & Modified Program in case of Regression Testing Module:-

This module includes four steps:-
1. Input original and modified java program.
2. Creation of flow graph of original and modified program.
3. Identify differences between flow graph of original and modified java program.
4. Display difference- new lines, deleted lines and modified lines.


Original Java Program
Modified Java Program
Create Flow Graph of Original Java Program
Create Flow Graph of Modified Java Program
Find out Difference between Flow Graphs of Original and Modified Java Programs
Display Result –New lines added, lines modified and lines deleted

6.4 Prioritization of Test Cases in case of Regression Testing Module:-
This module includes two steps:-

1. **Search Modified lines into all slices of original java program & find out effected program slices: -** In this step if slice contain modified lines, then it declared effect program slice.
2. **Compare execution history of original test suite with lines numbers of every effected program slices:-** In this step compare execution history of original test suite with lines numbers of every effected program slices and if both are equals then it declared effected test cases after modification.
3. **Prioritization of Test Cases:-** On the basis of maximum number of matches from the execution history of original test suite with lines numbers of every effected program slices, prioritized the test cases.


Modified Lines (Difference between Original and Modified Java Program)
Search Modified Lines into all Slices of Original Java Program
Yes
If Slice contain Modified Line
Effected Program Slices
Test Cases of Certified/ Original Java Program
Compare line numbers of effected slice with Original Program ' Execution History
If Line numbers of Effected slice equal With Original Program' Execution
YES
On the basis of maximum number of lines comparing, prioritized test cases

VII. CONCLUSIONS

Program slicing is a well-known program analysis and transformation technique that uses program statement dependence information to identify parts of a program that influence or are influenced by an initial set of program points of interest (called the slice criterion). This paper had shown an approach for generated test cases and prioritized test cases using executable static program slices.

REFERENCES

[1]. M.Weiser, Programmers use slices when debugging, Communications of the ACM, vol. 25, 446-452(1982)
[2]. Weiser, Program Slicing, IEEE Transactions on Software Engineering 10(4), 352-357 (1984).
[3]. Tip,F, A survey of program slicing techniques, Journal of programming languages 3, 121-189 (1995).
[4]. Jianjun Zhao, Applying Program Dependence Analysis to Java Software, In Proc. Workshop on Software Engineering and Database Systems, Taiwan 162-169, December 1998.
[5]. Andrea De Lucia, Program Slicing: Methods and Applications, IEEE international workshop,2001
[6]. Indus Project. http://indus.projects.cis.ksu.edu/
[7]. Unravel Project. http://hissa.nist.gov/unravel/
[8]. Wisconsin Program Slicing Project. http://www.cs.wisc.edu/wpis/html/
[9]. Topics in Program Slicing, http://www.cs.drexel.edu/~spiros/teaching/CS576/slides/5.slicing.pdf.
[10]. Ferrante, Ottenstein, The program dependence graph and its use in optimization, et al.-1987,