ISSN: 0975-833X

Available online at http://www.journalcra.com

International Journal of Current Research
Vol. 8, Issue, 06, pp.32412-32416, June, 2016

INTERNATIONAL JOURNAL OF CURRENT RESEARCH

RESEARCH ARTICLE

APPLICATION OF STATIC SLICER 1.01 FOR STATIC SLICING

***Suchismita Mishra and Sarada Baboo**

Department of Computer Science and Applications, Sambalpur University, Jyoti Vihar, Odisha, India

ARTICLE INFO

Article History:
Received 23rd March, 2016
Received in revised form
29th April, 2016
Accepted 27th May, 2016
Published online 15th June, 2016

Key words:

Static Slicing,
Control Flow Graph,
Static Slicer tool,
Graphviz.

ABSTRACT

The size as well as the complexity of the software are a substantial impact and are increasing with the increase of customer requirements and applications. As a result the maintenance of software becomes more and more tough. So in order to cope up the program complexity, slicing technique is performed. Program slicing is used for disintegration of a program. Slicing is a program analysis technique which is also used for various imperative programming languages. It aids understanding of data flow, control flow and debugging. Program slicing is used in various applications in the field of software engineering such as program debugging, understanding, program maintenance, testing and complexity measurement. Now a days the software size is very large and it is also very difficult to understand, maintain and test. To debug a program, the whole program is checked line by line and find out the error. This procedure is very slow and also time taking. To overcome such issues Weiser introduced program slicing. The process of slicing deletes those parts of the program that can be determined to have no effect upon the semantics of interest. Thus program slices are smaller than the program. The sliced program is easy to understand and maintain. In the past three decades, various notions of program slices have been proposed as well as a number of methods to compute them. Static slicing is the easiest program slicing and it is used to calculate the computation slice. In this thesis static slicer1.01 tool is used. Graphviz tool is used to construct CFG.

Citation: Suchismita Mishra and Sarada Baboo, 2016. "Application of static slicer 1.01 for static slicing", *International Journal of Current Research*, 8, (05), 32412-32416.

INTRODUCTION

The complexity of software application shows significant increase as the size of software increases. As a result the program maintenance activities such as adding new functionality, debugging and testing consume more amounts of available resources for software development. In order to deal with the complexity of software, there is need of computer supported methods for both decomposition and dependence analysis of programs. The Program slicing technique is one such method that carries out decomposition and dependence analysis of programs (Mark Harman and Robert Hierons, 2001). Program slicing was originally introduced by Mark Weiser. It is a decomposition technique. This technique extracts from program statement to a particular computation. Program slicing is referred to as finding all statements in a program that directly or indirectly affect the value of a variable occurrence (Mark Weiser, 1981). There are different types of program slice like Static Slicing, Dynamic Slicing, Forward Slicing, Backward Slicing, Quasi static slicing, Amorphous Slicing, Chopping etc. One of the simplest slicing among all the slicing is static slicing. Program slices are computed by static slicing. Using static data flow and control data flow analysis program slice is computed. Static slicing is computed for those statements of the program which are possible to execute. To understand the program and for maintenance of software static slicing is used. To understand the program execution and its detail process is not required in static slicing. To compute slice, static analysis is used in this technique. It means first analyzed the source code of the program and then compute the slices for input values (Tip Frank, 1995). The advantages of static slicing are that, it is easy to use. It is very fast to identify a slice. In static slicing, slicing is calculated directly from the original source program so it is easier. There are many disadvantages in static slicing. First, in static slicing a very large size of program slice is generated but in dynamic slicing smaller size of program slice is generated. Second, the array elements and fields in dynamic records as individual variables cannot treat by static slicing. The parts of a program that contribute to the computation to the selected function are understood by the help of static slicing (David Binkley and Keith Brian Gallagher, 1996).

****Corresponding author: Suchismita Mishra,***
Department of Computer Science and Applications, Sambalpur University, Jyoti Vihar, Odisha, India.

Literature Review

In this part various existing method and techniques of different slicing techniques are reviewed like mixed approach of static and dynamic slicing (S - D Slicing), Partial Evaluation And Program Slicing, Modular Monadic Slicing Approach, Complexity Measure Based on Program Slicing (CMBPS), Constraint logic Programming (CLP), Forward static slicing etc. From different paper it is found that slice computation is difficult for I/O statements, logical statements, control statements, looping statements etc. So in this paper the work is extended to calculate slicing point using Static Slicer 1.01tool.

Experimental Designs

Here 2 types of tools are used known as Graphviz and Static Slicer 1.01. Graphviz is also known as Graph Visualization Software. It is a program for drawing graphs specified in DOT (Diagrammatic Objective Text) language scripts. DOT language is a graph description language to draw the data and control flow. It is a plain text graph description language. Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks (https://en.wikipedia.org/wiki/Graphviz). Static Slicer is a small, java based application. It is simple to use. It is specially designed to calculate static slice (http://sourceforge.net/p/someslice/wiki/Home;http://www.downloadcollection.com/staticslicer.htm). The proposed algorithm is described below:

Algorithm

Step 1: Input File
Step 2: Create CFG nodes and set line numbers
Step 3: Initialize root node and slicing criterion
Step 4: Initialize variable of criterion
Step 5: Initialize root node
Step 6: Count number of nodes and set criterion to start node
Step 7: Store node in CFG as BST (Binary Search Tree)
Step 8: Create vector
Step 9: Find child node and its predecessor and successor node
Step 10: Add variables
Step 11: if variable! = null
Get slice criterion and calculate the relevant variable and store the variable
else
No slice point
end
Step 12: if criterion! = null
Find node using line number
else
end
Step 13: Get slice number from CFG
Step 14: Get child node
Step 15: if child node > 0
Get slicing criterion
Get line numbers or node
If node is in branch
Find line number
else
Display line number or slicing points
end
(Check influence variable)
Step 16: if influence variable= = criterion and variable vector
Get line number
Go to step 13

Description of the Algorithm

The above algorithm describes the overall flow of the slicing algorithm technique. Initially a file is input and creates control flow graph nodes. After creating the nodes set the line numbers of file to node. The root node and slicing criterion C= (L, V) where L is the line number and V is the slicing variable for example (10, {x}) is initialized. Then initialize the variable of criterion i.e. x. In the file, the first line is initialized as start node. The number of nodes is count and set the criterion to start node. The nodes are stored in control flow graph as binary search tree. To store slice point a vector is created. Then find the child node and its predecessor and successor node. In the algorithm add variables. If in the algorithm, variable is not there then there is no slice point and the process is end there. If variable is there, then get the slice criterion and calculate the relevant variable and store the variable in a vector. It is based on the criterion. If the criterion is null then the process is end there. If the criterion is there then find the node by using the line number. After finding the node get slice number from control flow graph by using recursion. Child node is derived from the root node. If the child node is greater than zero then get slicing criterion. If the child node is less than zero then displays the line numbers or slicing point and the process is end there. Slicing criterion is found out and then gets the line numbers or node. If node is in branch then find the line number and check the influence variable. If influence variable is equal to criterion and variable vector then find the line number and the process is repeated.

RESULTS

Here in this paper, Static Slicer 1.01 tool is used for computing the slicing points and has got the results as shown in the following screenshots. Here the algorithm has been taken as input to the slicer tool and the respective slicing points have been determined and shown in the screen shots. Graphviz tool is used to display the control and dataflow dependency. Here programs of different sequential structured program, selective program using if statement, iterative program using while loop, looping and selective statement etc has been taken. In this paper we take a Program to calculate factorial using while loop to show the slicing point in the screen shots. Slicing point is depending on the slicing criterion. So when slicing criterion is changed then slicing point also changed. How the slicing points are changed it is shown in below screen shots.

Dot program to generate the data flow

Here the slicing points that were found out using static slicer are (Mark Weiser, 1981; Tip Frank, 1995; David Binkley and Keith Brian Gallagher, 1996; https://en.wikipedia.org/wiki/Graphviz; http://www.downloadcollection.com/staticslicer.htm). The slicing criterion for this program was taken as (11, {fact}).

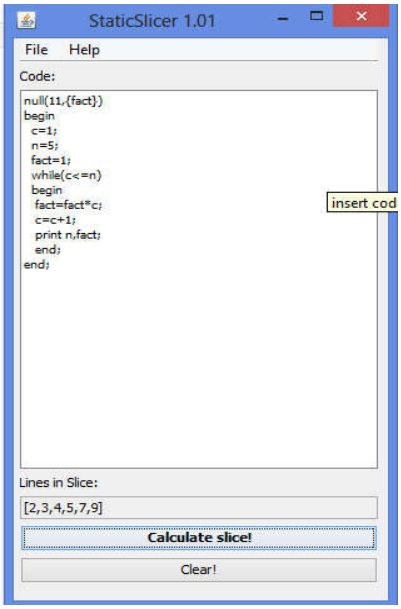

Fig. 1. Slicing point calculation using static slicer

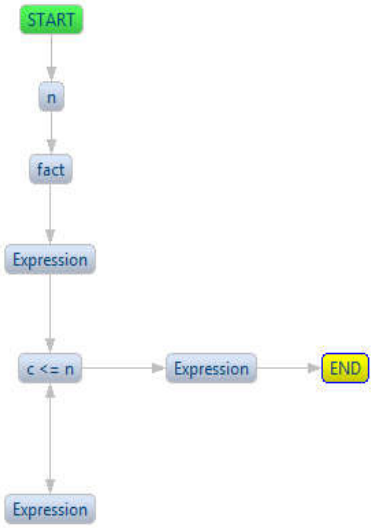

Fig. 2. CFG of factorial number

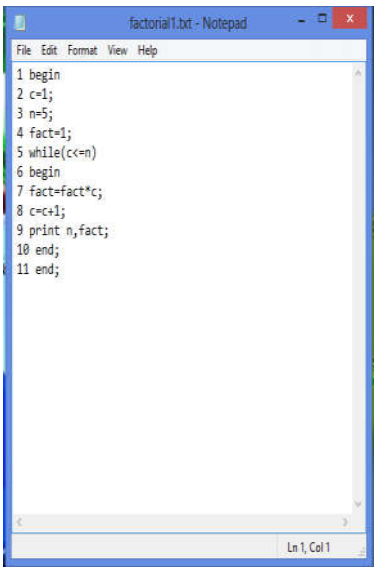

Fig. 3. Program with Line number

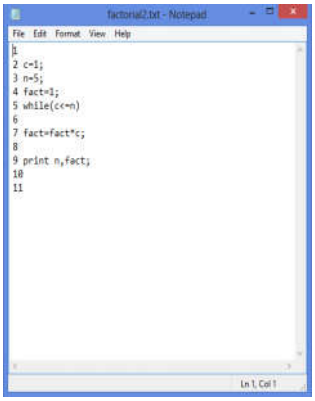

Fig. 4. Line number with slice point

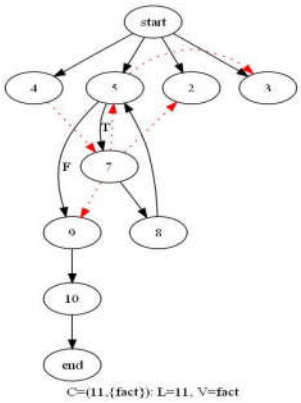

Fig. 5. Data flow with influence variable dependency

Dot program to generate the data flow

```
digraph{
label=" C=(11,{fact}): L=11, V=fact"
start->2
start->3
start->4
start->5
subgraph{
rank=same; 2;3;4;5;
}
5->7[label="T"]
5->9[label="F"]
7->8
8->5
9->10
10->end
4->7[style=dotted, color=red]
7->2[style=dotted, color=red]
7->5[style=dotted, color=red]
5->3[style=dotted, color=red]
7->9[style=dotted, color=red]
}
```

Here line number L is 11 and the slicing variable is fact. According to slicing criterion and the slicing variable fact, the line number 4, 7 and 9 are the slicing point. In the line number 7 c is the influence variable. Hence the slicing point occurred at line number 2 and 5. In line number 5, n is the influence variable for variable c. So line number 3 is the slicing point.

When slicing criterion is changed:

Fig. 6. Slicing point calculation using static slicer

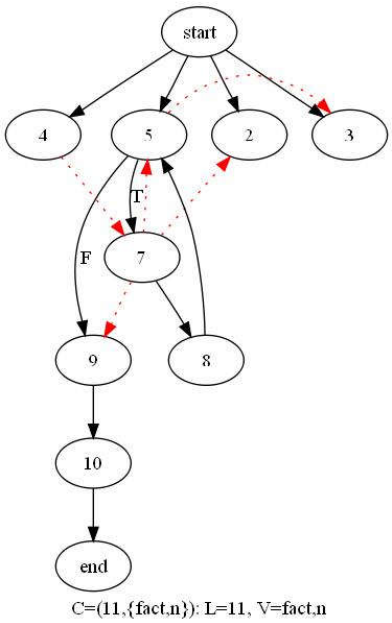

Fig. 7. Data flow with influence variable dependency

Dot program to generate the data flow

Here the slicing points that were found out using static slicer are (Mark Weiser, 1981; Tip Frank, 1995; David Binkley and Keith Brian Gallagher, 1996; https://en.wikipedia.org/wiki/Graphviz; http://www.downloadcollection.com/staticslicer.htm). The slicing criterion for this program was taken as (11, {fact, n}). Here line number L is 11 and the slicing variable is fact, n. According to the slicing variable fact, line number 4, 7 and 9 are slicing points. According to the slicing variable n, line number 3 is the slicing variable. In line number 7, c is the influence variable. So line number 2 and 5 are the slicing points. In this way we changed the slicing criterion and got the different slicing point.

```
digraph{
label=" C=(11,{fact,n}): L=11, V=fact,n"
start->2
start->3
start->4
start->5
subgraph{
rank=same; 2;3;4;5;
}
5->7[label="T"]
5->9[label="F"]
7->8
8->5
9->10
10->end
4->7[style=dotted, color=red]
7->2[style=dotted, color=red]
7->5[style=dotted, color=red]
5->3[style=dotted, color=red]
7->9[style=dotted, color=red]
}
```

Conclusion

Program slicing is a technique to find essential program instructions for a specific result in the execution using a particular condition. The advantage of using this technique is that the portions of the program can be left out that does not influence the result. It is an automated technique to extract part of a program that affect a set of variables. It is used for easy identification of an error for a program having large Lines of Code. It is also used to find the dependency that exists between direct and indirect related instructions basing on data and control flow analysis. Program slicing technique is not only used for Procedure Oriented Programs but also for Object Oriented Programs. Reusability is the vital concept of Object Oriented Programming. However in some situations the code that needs to be reutilized sometimes combined with other kind of code, which is not needed. The solution for this type of problem is Static slicing technique which can minimize the

needed code for execution and ignore the irrelevant one. In this thesis, programs having static contents are taken into consideration for analysis of Static Slicing. A set of programs are analyzed using the tool basing on both Data Flow Graph (DFG) and Control Flow Graph (CFG). The tool can be used to find the Slicing points basing on an input Slicing criterion. It is concluded from the experiment of this paper for a considerable set of programs, that the tool can be used for Static Slicing for programs having control and looping types of statements. In future, new algorithms and approaches to Static Slicing can be implemented for the used tool to extend the work of Weiser. The research work shall be extended to slice programs for jumping type of statements, slicing of array elements using Program Dependency Graph (PDG) that may be applicable in various real and industrial applications. Taking the advantage of Static Slicing, the tool Static Slicer 1.01 shall also be extended to identify semantics of a program that can be applied for program verification.

REFERENCES

David Binkley and Keith Brian Gallagher, 1996. "Program slicing", Advances in Computers, *Academic Press*, Volume 43, pages 1–50.

http://sourceforge.net/p/someslice/wiki/Home/

http://www.downloadcollection.com/staticslicer.htm

https://en.wikipedia.org/wiki/Graphviz

Mark Harman and Robert Hierons, 2001. "An overview of program slicing", *Software Focus*, Volume 2, Issue 3, pages 85–92.

Mark Weiser, 1981. "Program slicing", Proceedings of the 5th International Conference on Software Engineering, *IEEE Computer Society Press*, pages 439–449.

Tip Frank, 1995. “A Survey of program slicing techniques”, *Journal of programming languages*, 3.3121-189.

****