ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

A New approach of program slicing: Mixed S-D (static & dynamic) slicing

Mrs. Sonam Jain[1], Mr. Sandeep Poonia[2]
M.Tech (CSE) Scholar, Jagannath University, Jaipur[1]
Associate Professor, Department of Computer Science & Engineering, JNIT University, Jaipur[2]

Abstract: Program slicing technique is used for decomposition of a program by analyzing that particular program data and control flow. The main application of program slicing includes various software engineering activities such as program debugging, understanding, program maintenance, and testing and complexity measurement. When a slicing technique gathers information about the data and control flow of the program taking an actual and specific execution (or set of executions) of it, then it is said to be *dynamic slicing*, otherwise it is said to be static slicing. Generally, dynamic slices are smaller than static because the statements of the program that affect by the slicing criterion for a particular execution are contained by dynamic slicing. This paper reports a new approach of program slicing that is a mixed approach of static and dynamic slice (S-D slicing) using Object Oriented Concepts in C++ Language that will reduce the complexity of the program and simplify the program for various software engineering applications like program debubbing.

KEYWORDS: Program-Slicing, Static-Slicing, Dynamic-Slicing-D slicing, Data-Dependency, Control- Dependency.

1. INTRODUCTION

To obtaining subparts of a program with a collective meaning a technique is used called Slicing. Program slicing is one of the debugging methods used to locate the errors in a program originally proposed by Weiser [7]. The idea of program slicing is to focus on the statements that have something to do with a variable of interest (criterion variable), referred as the *slicing criterion* with those statements which are unrelated being omitted. Using the slicing method, one obtains a new program of generally smaller size which still maintains all aspects of the original program behavior with respect to the criterion variable.

Program slicing can be classified into two main categories: *Static slicing and dynamic slicing*.

A *static slicing* uses static analysis to derive slices. i.e., the source code of the program is analyzed and the slices are computed for all possible input values.

Dynamic Slicing makes use of the information about a particular execution of a program. [11] .A dynamic slice preserves the effect of the program for a fixed input. An advantage of dynamic slicing over static slicing is a smaller size program is derived from the dynamic slicing approach, or in the worst case, it will be of equal size.

When the slicing is performed from the starting statement referred to in the criterion and going back to the beginning, then it is said to be backward slicing, or in backward slicing we are interested in all those statements that can influence the slicing criterion. Otherwise it is said to be forward slicing. In forward slicing we are interested in all those statements that could be influenced by the slicing criterion .The main application of forward slicing is that we can determine how a modification in a part of the program will affect other parts of a program.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

1.1 PURPOSE AND SCOPE OF THE STUDY

The main purpose of this work is to implement a mixed slicing approach of static and dynamic slicing i.e. S-D slicing approach in generating a program slice. Specifically, in this study we develop a code that have an Object Oriented approach by using both static and dynamic slicing.

The rest of this paper is organized as follows. Section2 contains a general discussion of static and dynamic slicing and Section 3,4 represents the steps involved in the designing of a new approach is S-D slicing (mixed of static and dynamic slicing) with object oriented concepts.

2. EXAMPLE OF STATIC AND DYNAMIC SLICING

2.1 Process of static slicing

```
1: a:=4;
2: b:=3;
3: readln(c);
4: if c=0 then
5: d:=a ;
6: else
7: d:=a+1;
8: e:=b+a;
9: writeln(d);
10: writeln(e);
```

Fig. 1 Example of program Slicing

The data dependence and control dependence in static slice is denoted using SDG (System Dependence Graph). The SDG for the above lines of code is:

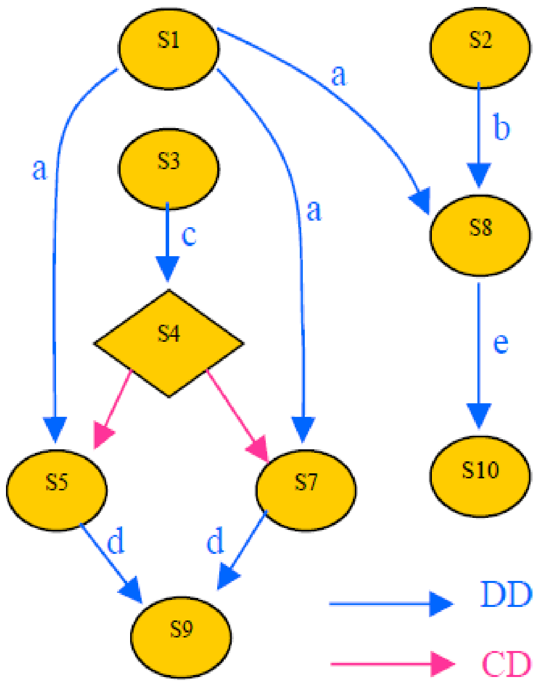


Fig. 2.1 System Dependence Graph

2.1.1 Process of Static slicing

Consider the procedural C++ example program given in Figure 2.1.a. The static slice with respect to the slicing criterion <11; add > is the set of statements {4, 5, 6, 8, 9}. Consider a particular execution of the program with the input value x = 15. The dynamic slice with respect to the slicing criterion < 11, add > for the particular execution of the program is {5}.

```
1 main( )
2 {
3 int x, add;
4 cin>> x;
5 add = 0;
6 while(x <= 10)
7 {
8 add=add+x;
9 ++ x;
10 }
11 cout<<add ;
12 cout<< x;
13 }
```

Fig. 2.1.a An example program

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

```
1 cin>> x;
2 add = 0;
3 while(x <= 10)
4 add=add+x;
5 ++ x;
```

Static slicing: criterion <11; add >
Fig 2.1.b An example Program on Static Slicing

2.2 Process of dynamic slicing

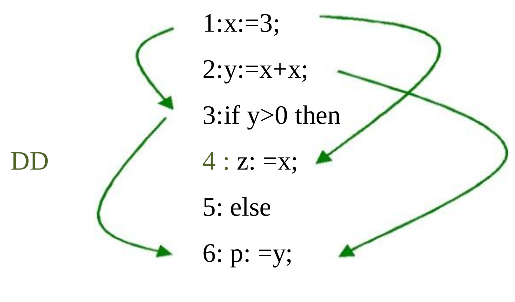
To understand the concept of dynamic slicing to have to know about dependency i. e. when each statement of a code is dependent on other statement in some way, this is known as dependency. Generally, it is categorized into two types:

(1)*Data Dependency*: In this dependency scheme a statement or a variable is dependent on some other statement for *some data* then it is known as data dependency.

(2)*Control Dependency*: In this dependency scheme the execution of a statement is dependent on some other statement it is called as control dependency.

2.2.1 Data -Dependence: DD (s1, v1, t1):

- Variable v1 is defined in a statement s1.
- v1 is referred in a statement t1, and
- At least one execution path without re-definition between s1 and t1 exists.



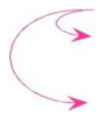
```
1:x:=3;
2:y:=x+x;
3:if y>0 then
4 : z: =x;
5: else
6: p: =y;
```

Fig. 2.2.a An example program on Data Dependence

In the above figure 2.2.a the curve lines shows *data dependen*ce.

2.2.2 Control -Dependence: CD (s1, t1):

- s1 is a conditional predicate, and
- the result of s1 determines whether statement t1 is executed or not



```
1: x: =3;
2:  y:=x+x;
3:  if y>0 then
4:  z:=x
5:  else
6:  p:=y;
```

Fig. 2.2.b An example program shows Control Dependence

In the above figure 2.2.b the curve lines shows *control dependen*ce.

2.2.3 Process of Dynamic slicing

Let us consider an example:

```
1: x:=5;
2: y:=3;
3: readln(z);
4: if z=0 then
5: p:=x
6: else
7: p:=x+1;
8: q:=y+x;
9: writeln(p);
10: writeln(q);
```

Execution trace with input z=0 is as follows:

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 5, May 2013

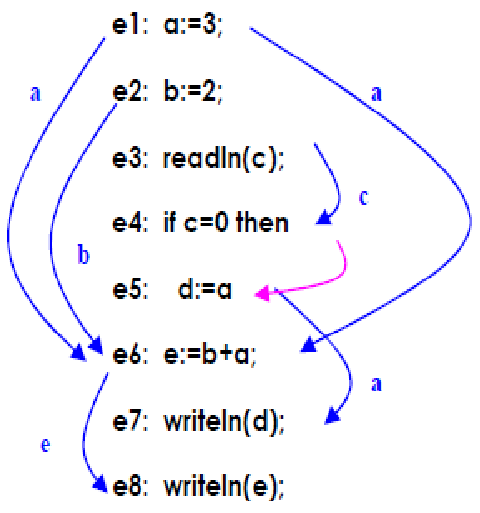


Fig. 2.2.c An example program Dynamic Slicing

3. MIXED APPROACH OF STATIC AND DYNAMIC SLICING (S-D SLICING) APPROACH

When we study both approach about static and dynamic slice some advantage and disadvantage of both approaches are seen that the advantage of static slicing it is easier and faster to identify a static slice. This is because computations for generating a static slice are done directly from the original source program. Static slicing does have the d*i*sadvantages. First, a larger size program slice is generated by static slicing than that of dynamic slicing. Second, the array elements and fields in dynamic records as individual variables cannot treat by static slicing.

Unlike static slicing, dynamic slicing is defined on the basis of one computation rather than all computations and a dynamic program slice is computed only by the executable part of the original source program. So, a smaller size program is generated by the approach of dynamic slicing. In addition, it also enables one to treat array elements and fields in dynamic records as individual variables. By applying dynamic slicing, one finds it easier to identify the statements in the program that do not influence the variables of interest. Disadvantage of dynamic slicing is that it is slower and more difficult to compute than static slicing. This is because in this slicing approach we have to determined the executable part of the original source program before computations of slices.

So by designing a new approach mixed approach of static and dynamic slicing with object oriented concept we can overcome some disadvantage of dynamic slicing like we can speedup the execution of program slice and reduce he complexity of program slice.

4. AN EXAMPLE OF MIXED S-D SLICING APPORACH

```
int Parts=1;
class Individual
{
public:
virtual void output()
{
Parts = Parts+10000;
cout << "Individual is the CEO";
}
};
class Worker: public Individual
{
public:
int pay;
Worker()
{
pay = 1000;
}
using Individual::output;
void output(int years)
{
for (int i=1;i < years; i++)
{
Parts++;
pay = pay + years;
pay= pay * 1.01;
}
}
};
class Executive: public Worker
 {
public:
int years;
using Worker::output;
void output(char position)
{
Parts = Parts+100;
cout <<"Executive"<<position;
}
};
void main()
 {
Worker our_Worker;
Individual our_Individual;
Executive our_Executive[100];
int i;
for (i=1; i<100; i++)
{
our_Executive[i].years=50;
if (i < 80)
our_Executive[i].output(50);
else
if(i<99)
our_Executive[i].output('M');
else
our_Executive[i].output();
cout << our_Executive[i].years;
cout <<Parts<<endl;
}
}
```

Fig. 3.1 An example program S-D Slicing]

In the above Figure 3.1 we implemented a new approach of dynamic program slicing is S-D Slicing approach with Object oriented Concepts in C++ Language. In this example we use the concept of Classes and Polymorphism and also inheritance in terms of Object oriented Programming in C++ languages.

CONCLUSION

We have designed a new approach of program slicing i.e. mixed static and dynamic slicing (S-D slicing). The main feature of this approach is that we can generate dynamic slices in a faster way by using Object Oriented Concepts like classes and inheritance etc.. As we know that by using Object Oriented concepts in C++ language we can reduce the size or length of a program. Here we presented the approach in terms of C++; other versions of this approach for object oriented programming languages such as Java can be easily adaptable. In this paper we develop a mixed slicing approach based on the mixture of both static and dynamic slicing which generates dynamic slices. This approach is intended for reduce the complexity and debugging environment for Object Oriented C++ programs in which the static and dynamic slicing technique is being used.

REFERENCES

[1] S. Horwitz. Reps, “Interprocedural slicing using dependence graphs”, Programming Languages and Systems, 1990.
[2] L. D. Larson and M. J. Harrold. “Slicing Object Oriented software”, German, March 1996.
[3] M.Shara Lydia, Jaydev Gyani, “Dynamic Attribute Slicing Technique for Object-oriented programs”, In Proceedings of the National Conference on Informatics , NCI-2008, Nellore,AP.
[4] Y. Song and D. Huynh. “Forward Dynamic Object-Oriented Program Slicing”, Application Specific Systems and Software Engineering and Technology (ASSET'99). IEEE CS Press, 1999.
[5] I.Srinath, Jaydev Gyani, “Static Attribute Slicing Technique for Object-oriented programs”, In Proceedings of the National Conference on Informatics , NCI-2008, Nellore, AP.
[6] M. Weiser. “Programmers use slices when debugging”, Communications of the ACM, 25(7):446 – 452, 1982.
[7] M. Weiser. “Program slicing”. IEEE Transactions on SE 10(4), 1984.
[8] X. Zhang and Y. Zhang. “Forward computation of dynamic slices”, International Conference on SE, 2004.
[9] J. Zhao. “Dynamic slicing of object-oriented programs”, Technical report, Information Processing Society of Japan, May 1998
[10] 10.G. B. Mund, R. Mall, S. Sarkar "Computation of intraprocedural dynamic program slices”, Department of CSE, IIT Kharagpur, Information and Software Technology (45).
[11] B. Korel, J. Laski, "Dynamic program slicing," Information Processing Letters, vol. 29, 1988.
[12] Weiser, Mark, “Program Slicing,” Proceedings of the Fifth International Conference on Software Engineering, March 1981, [13] Weiser, Mark, “Programmers Use Slices When Debugging,” Communications of the ACM, Vol. 25, No. 7, July 1982.
[14] Weiser, Mark, “Program Slicing,” IEEE Transactions on Software Engineering, July 1984.