Program Slicing Techniques over Software Version Histories

D.Jayakumar, C.H.Tirupathaiah, R.Kannadasan, Komal venugobal, M.S.SaleemBasha

Abstract: ***Programing is basic knowledge in maintaining software and software engineering. Knowing the program suspected of the output extraction of the program. And relationship carried out through programming. The behaviour of spectating full the involved given program sentences is focusing or unfocusing the value the which the variable given in the statements at some place in the program it is known as program slicing. The approach that extracts the parts of system programs by excluding them by data flow which is similar to items of data in the program is Program slicing technique. This paper explains the different centers of program slicing techniques (which are not executed) like conditional slicing, static slicing, dynamic slicing and, quasi-static slicing. The paper incorporates various techniques, playing with various slicing techniques like, backward slicing, forwarding slicing, semantic slicing and syntactic slicing.***

Keywords: Maintaining Software, Static Slicing, Dynamic Slicing, Quasi-Static Slicing.

I. INTRODUCTION

The software is very difficult in the current situation. Definitely it becomes very complex for programmer to know the data, to maintain, testing and debugging. It will be difficult to search the bug line-by-line in a code. The slicing is introduced to solve this issue. This technique will extract the sub-program in which the whole program is dependent. Program slice can be determined in different ways. We can find it using the dataflow directions. Else the intermediate form is found and an efficient algorithm is implemented to the program. The program slicing is a technique which helps the programmers and coders and employers to reduce the work and saving the time. The program slicing can be applied in many fields like debugging, maintenance of software and refactoring and many.

Revised Manuscript Received on November 12, 2019.

* Correspondence Author
D.Jayakumar, Department of CSE, RMD Engineering College, Chennai, India. Email: djayakumarbe@gmail.com
C.H.Tirupathaiah, Technical Spcialist(Senior), IdentivPrivate Ltd., (“INVE”), Chennai,India . Email: tirupathaiahch@gmail.com
R.Kannadasan, SCOPE, VIT University, Vellore, India. Email: desurkannadasanr@gmail.com
Komal Venugobal SCOPE, VIT University, Vellore, India. Email: komalvenugopal@gmail.com
M.S.Saleem Basha, Head-Research centre, Department of Computer Science, Mazoon University, Sultanate of Oman.
Email: m.s.saleembasha@gmail.com

II. BACKGROUND

Program slicing is a technique it helps in reducing the tasks of the programmers, coders in their work and it may also extend the time of them. The program slicing mainly useful in different tasks such as debugging, refactoring, etc. The slicing technique is mainly used in different software industries. There is a definite need to find the different algorithms so that the efficiency of the system increases a lot to reduce the cost and memory usage as it deletes the repeated and un-wanted variables.

III. MOTIVATION

1. Programmers debug the code and they are mentally executed by training their brain. so, to help programmers from mental debugging it turned into an inspiration driving to the development of spontaneous techniques. Code was equal as the set variable and problem comes from; the sentences may cause the problem to the program. By using this we can make the debugger to find the actual problem or a fault.

2. Cohesion is the application of the program of syntax-preserving for the static-slicing. A program which modularizes program performance accurately is known to be a cohesive-program. A strong cohesive function is to perform the assigned tasks which are relevant to each other.

3. In SDLC, it is easy to understand beginning of maintenance phase. This is fully acceptable by the legacy systems in case of, where the documentation cannot be accessible and it less to the original developers. So, the slicing (conditional) may help the maintenance of the comprehension phase. Here both the slicing algorithms can show their properties as for the condition to build the addition of conventional static slicing criterion. This condition helps us to use and recognize the several cases of the slicing.

IV. RELATED WORK

There are many concepts which has been proposed similarly intermediate representation and there are many steps have been considered to obtain representation. In 1987, ferrante proposed an algorithm for constructing the control dependency graph by using many tools like control flow graph and dominator tree of flow for the input program.

For finding the dominators there is an algorithm used immediate dominators and semi dominators for considering every node. The intermediary step is the semi dominator computation. They have proposed, many certain properties for immediate dominators. Many of them proposed many theories for program slicing. There is a slicing criterion known as <S, V> in which V is the subset variable at the point and S is point of program. The slices which are computed are the primary executable programs where we can obtain zero or many sentences from the extraction of the program. The proposed algorithm the date which we used is for analysis of control flow graph of the program which we can compute procedural process like inter and intra.

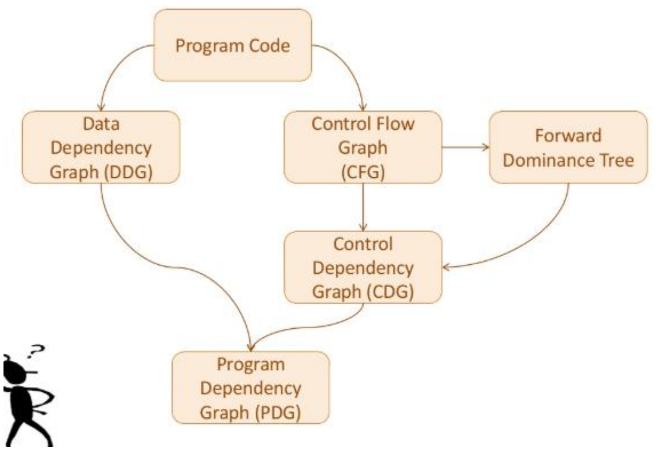
In the algorithm reachability the graph used is to compute the static slice which is the sentences that may affect the program point.

V. LITERATURE SURVEY

1. In [1] there is surveyed on dynamic slice it is discussed that the dynamic slicing is a process that executes the models of program execution which helps in handling the traces of program execution. Here a slicing criterion gives the solution for the execution of the events which are instructed which are changed by the criterion or it may affect, and it reduces the space of search to find the execution of traces. The reduction for the search space is large for many purposes, and they not perfect with the quality of each instruction of event of the execution along with a slice.

2. In [2] they discussed about the software-development. Regularly programming advancement will face many difficult issues of maintaining the key requirements from expense and time. Frequently programming advancement faces difficult issues of meeting key imperatives of expense, time. They are supposed to search the existence of relationship which is similar and between to software development and program slicing phases on basis of the studies which are made in the previous history and they help for making the software systems which saves the cost a time effectively.

3. In [3] the Program Slicing Technique: An Novel Approach to Improve Programming Skills in Novice Learners Kannan M. Moudgalya, Kiran L.N Eranki mentioned that the Conventional classroom the teaching will start with a declarative analysis of the concepts followed by the program finding skills. Program-slicing techniques is used in the process of software assessment, testing, debugging. According to our survey and studies, it is known that the slicing techniques have aided a lot.


Program Code
Data Dependency Graph (DDG)
Control Flow Graph (CFG)
Forward Dominance Tree
Control Dependency Graph (CDG)
Program Dependency Graph (PDG)

VI. PROPOSED MODEL

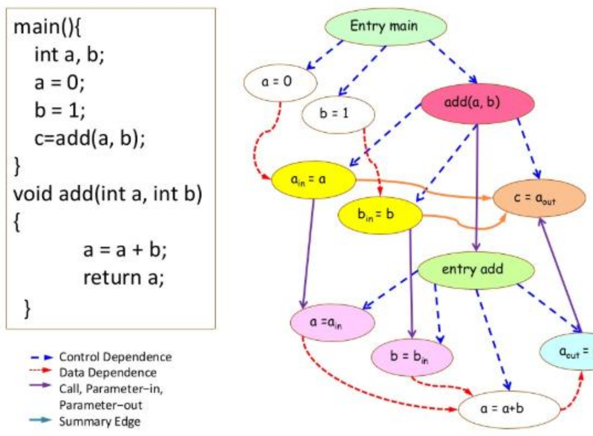
In this paper, we are going to analyze the performance of the System Dependence Graph which maintains the relationship between the sub-programs. The System dependence Graph is considered as an extension of Program Dependence Graph that was discussed earlier.

SDG = PDG + call-graph

It is an inter-procedural slicing algorithm that will work using the SDG.

1) SDG showing the transitive-dependencies.

2) While doing the two-phase traversing on SDG the slices will be computed accordingly.

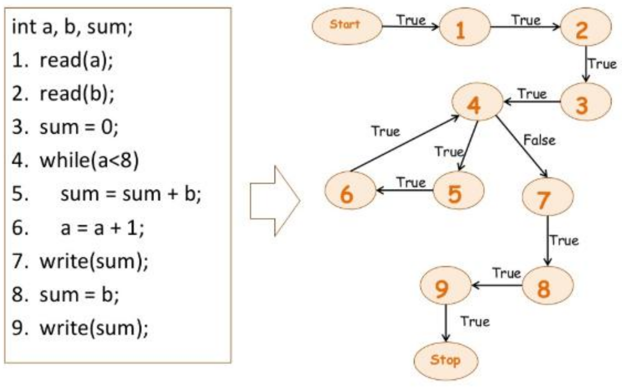


```
main(){
   int a, b;
   a = 0;
   b = 1;
   c=add(a, b);
}
void add(int a, int b)
{
          a = a + b;
          return a;
 }
```

VII. CASE STUDY ASPECT

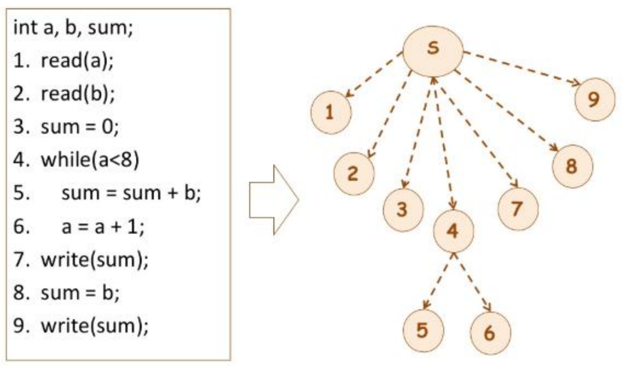
An intra procedural graph will compute the slices within one procedure, whereas an inter-procedural graph computes the slices from the multiple programs.

Control Flowgraph- The Program slicing was proposed initially by Weiser to solve the dataflow problem that was found using CFG. People have tried to find the best slicing algorithms to find the smallest possible program-slice for the given criteria. Some of those problems can be solved using the data flow equations whereas some of the remaining can be solved using the Graph representations. In CFG every line of the code is assigned with a number and the dependency graph is generated for the numbers. The graph generated is an intraprocedural graph. The control flow graph is a data structure. In this, each node is associated with a program statement.



```
int a, b, sum;
1. read(a);
2. read(b);
3. sum = 0;
4. while(a<8)
5.     sum = sum + b;
6.     a = a + 1;
7. write(sum);
8. sum = b;
9. write(sum);
```

Program Dependence Graph- Karl and Linda Ottenstein are have used the PDG initially to compute program slices. The PDG is an inter-procedural graph. The slicing algorithm proposed is mainly focused on the reachability of the graph in PDG. It identifies every variable interdependency in the code. Program Dependence Graph is in brief considered as the CFG annotated with data flow information. Each vertex represents a statement (like CFG). Control dependence (solid lines) – between a predicate and the statements it controls. Data dependence (dotted lines) – between statements modifying a variable and those that may reference it. The Slicing algorithms are classified on a different basis such as
I. Techniques of Slicing- Static vs. Dynamic vs. Conditioned Slicing
II. The direction of Slicing- Forward vs. Backward vs. Chopping Slicing
III. Levels of Slicing- Intraprocedural vs. Interprocedural Slicing
IV. Executability of Slices- Executable vs. Closure



```
int a, b, sum;
1. read(a);
2. read(b);
3. sum = 0;
4. while(a<8)
5.     sum = sum + b;
6.     a = a + 1;
7. write(sum);
8. sum = b;
9. write(sum);
```

Slicing Criterion- <s, v> where s will specify the location of the statement s and v specifies variable (v). Variable V at statement S can be affected by statements S’

Forward Slicing- The statements that are affected by the slicing criterion are included.
Answer the question “what program components might be affected by a selected computation?” Useful in determining which statements in program are affected by a change in v’s value in statement si. The main application is Dead Code Removal. In this type of the slicing, the starting line of the subcode to be checked is taken as line1 initially and the arrows are pointed to the lines in which it is present (its presence of the dependence) and the final graph is generated.

Backward slicing- The traversing happens either forward or in the background direction from the slicing criteria. The statement that influences the slice will be considered.

Answer the question “what program components might affect a selected computation?” The backward slicing is used mainly in debugging. The original method was developed by Weiser: static backward slicing.

Dynamic slicing- It is a non-static slicing technique where the execution is monitored. The slices of the program are computed with respect to the program history. It maintains a trace of the slices.

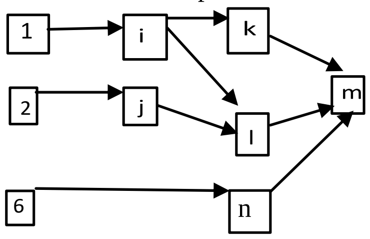
Static Slicing - It consists of all the variables which might effect the variable for all allowable executions. Compared to that of the static slicing, it may lead to big slices. The conditioned slicing is a technique that is the combination of both the static and dynamic slicing techniques.

VIII. DISCUSSION

Let us consider a code when the statements of the code are assigned with some values it looks as below

1	i=10;
2	j=11;
3	k=i+4;
4	l=i-j;
5	m=k-l;
6	n=5
7	m=n+1;

Control Flow Graph-



This is a general graph that shows the flow of variables and their inter-dependency in the program that is mentioned above.

Forward Slicing-<1,{i}>	Backward Slicing-<7, {m}>
1. i=10;	1. i=10;
3. k=i+4;	2. j=11;
4. l=i-j;	3. k=i+4;
5. m=k-l;	4. l=i-j;
	5. m=k-l;
	6. n=5
	7. m=n+1;

The examination of both the forward and the backward slicing using the slicing criterion <1, {i}>, i.e., variable I in line 1, The forward slicing will determine the downstream statements which the statement 1 will affect. In our example, the forward slice includes statements 1, 3, 4, and 5. Similarly, backward program slicing with slicing criterion <7, {m}>, produces a slice that includes statements 1, 2, 3, 4, 5, 6, and 7. sink.

IX. CONCLUSION

This paper mainly discusses the different slicing techniques such as forwaard slicing, backward slicing, chopping, static and dynamic slicing and their usage in the real-time applications. This paper explains you about the control and the data dependency in a graph. We have compared the different slicing algorithms and found which is efficient in their respective use case.

REFERENCES

1. Vijay Krishna Palepu, James A. Jones, Discriminating influences among instructions in a dynamic slice, Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, September 15-19, 2014, Vasteras, Sweden
2. Muhammad Saleem, Rasheed Hussain, Vasir Ismail, Shaikh Mohsin, Cost-effective software engineering using program slicing techniques, Proceedings of the 2nd International Conference on Interaction Sciences: Informat.
3. ion Technology, Culture and Human, p.768-772, November 24-26, 2009, Seoul, Korea
4. László Vidács, Árpád Beszédes, Tibor Gyimóthy, combining preprocessor slicing with C/C++ language slicing, Science of Computer Programming, v.74 n.7, p.399-413, May 2009
5. Pierre Rousseau, A new approach for concurrent program slicing, Proceedings of the 26th IFIP WG 6.1 international conference on Formal Techniques for Networked and Distributed Systems, September 26-29, 2006, Paris, France
6. Kannan M. Moudgalya, Kiran L.N Eranki, Program Slicing Technique: A Novel Approach to Improve Programming Skills in Novice Learners, Proceedings of the 17th Annual Conference on Information Technology Education, September 28-October 01, 2016, Boston, Massachusetts, USA
7. Sandrine Blazy, Andre Maroneze, David Pichardie, Verified Validation of Program Slicing, Proceedings of the 2015 Conference on Certified Programs and Proofs, January 13-14, 2015, Mumbai, India real-time programs.
8. Christer Sandberg, AndreasErmedahl, Jan Gustafsson, Björn Lisper, Faster WCET flow analysis by program slicing, ACM SIGPLAN Notices, v.41 n.7, July 2006
9. Christopher Pietsch, Udo Kelter, Manuel Ohrndorf, Timo Kehrer, incrementally slicing editable submodels, Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, October 30-November 03, 2017, Urbana-Champaign, IL, USA
10. K. B. Gallagher, "Surgeon's assistant limits side effects", IEEE Software, vol. 7, pp. 64, May 1990.
11. M. Weiser, "Programmers use slicing when debugging", CACM, vol. 25, no. 7, pp. 446-452, July 1982.
12. K. B. Gallagher, J. R. Lyle, "A program decomposition scheme with applications to software modification and testing", Proc. 22nd Int. Conf. System Sciences, vol. II, pp. 479-485, 1989-Jan.
13. R. Grady, "Measuring and managing software maintenance", IEEE Software, vol. 4, Sept. 1987.
14. P. Hausler, "Denotational program slicing", Proc. 22nd Hawuii Int. Conf. System Sciences, vol. II, pp. 486-494, 1989-Jan.
15. S. Horwitz, J. Prins, T. Reps, "Integrating non-interfering versions of programs", Proc. SIGPLAN'88 Symp. Principles of Programming Languages, 1988-Jan.
16. S. Horwitz, J. Prins, T. Reps, "Integrating non-interfering versions of programs", ACM Trans. Programming Languages and Systems, vol. 11, no. 3, pp. 345-387, July 1989.
17. M. Weiser, "Program slicing", Proc. 5th Int. Conf. Software Eng., pp. 439-449, 1981-May.

AUTHORS PROFILE

D.Jayakumar is an Assistant Professor,Department of CSE, RMD Engineering College, Chennai, India. His research activities are carried out in Operation Research,Operating System, Cloud Computing and DBMS computing.

C.H.Tirupathaiah, Technical Spcialist(Senior), IdentivPrivate Ltd., ("INVE"), Chennai,India. His research and technical activities are carried out in language translators and Working in Different Language platform.

R. Kannadasan is an Assistant Professor (Senior) at School of Computer science and Engg., (SCOPE), VIT University, Vellore, India. His research activities are carried out in bio informatics, language translators and DNA computing.

Komal Venugobal, School of Computer science and Engg., (SCOPE), VIT University, Vellore, India. His research activities are carried out in Networks and Distributed computing.

M.S.Saleem Basha is Head-Research centre, Department of Computer Science, Mazoon University, Sultanate of Oman. His research activities are carried out in Service Oriented Architecture, DNA Computing and Software Engineering.