A LITERATURE SURVEY ON PROGRAM SLICING

S.Mishra*

S.Panda**

S.Baboo***

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. In this review, it provides an overall review from different literature on program slicing and to ascertain a simplest approach for static slicing technique to reduce the time and cost issues.

Keywords

Program slicing, chopping, Program Integration, Partial Evaluation, Complexity Measure Based on Program Slicing (CMBPS), Constraint Logic Programming (CLP), System Dependence Graph

* M.Phil. CSA

** M.Phil. CSA

*** Reader

P.G. Department of Computer Science and Applications, Sambalpur University, Odisha, INDIA

1. 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.

In program slicing technique, the program elements related to a particular computation are extracted from the program. . A program slice consists of those parts of a program that may directly or indirectly affect the values computed as per the program point of interest. Now a day it is a challenging research problem to find satisfactory and effective slicing algorithm [1].

An application or software undergoes various changes during its lifecycle. In order to eliminate the bug one might come across during the testing period. For smooth working of software, every time the software undergoes some modification. After modification again it undergoes testing. The modification and testing is completed when the modified code behaves as expected. To get the correct output, again reset the application from the beginning .It would take considerable amount of time and the process is very easy. Various testing approaches are proposed for selective retest problem and it is based on code analysis. But all these approaches are expensive to implement. A better approach is intermediate representation of the program. This approach is less complex than the code analysis. Another advantage of the intermediate code representation is that we can have a better understanding of the program which is very useful for computing slices [2].

Program slicing is used to finding the errors in the program. It may affect the entire software .There are many software used for these purposes .Now a days it is more preferred due to time and cost issues. Slicing is used in regression testing and has many other applications in software maintenance activities. A program slice consists of the parts of a program that affect the values of computation at some point of interest. For slicing a program several aspects are to be considered. These are

- **Slicing variable:** It may be based on the variables specified in the criteria or it may be on all variables.

- **Type of the result:** The result of slicing may be equivalent to the program of few set of statements from the program.
- **Scope:** The scope of the slice may be interprocedural or intraprocedural [3].If the slicing algorithm can handle slicing across procedure boundaries it is called interprocedural.
- **Slicing point:** considering the slicing point,programmer's interest may be before or after a particular statement [4].
- **Slicing direction:** The expected slice from the program may be in the forward or backward direction.
- **Abstraction level:** It is either statement or in procedure. If an algorithm works at the statement level it means that the units that are considered for inclusion in the slice are statements. Procedure level abstraction means that the algorithm will decide if whole procedures should be included in the slice or not.
- **Type of information:** Another aspect is whether the slice can be computed at compile time, i.e. that the slicing algorithm only uses statically available information or if it also uses run-time information from the actual execution. The first kind is called static slicing, the second one dynamic slicing. The information will be either static or dynamic.
- **Computational method:** The method of computing the slice is by solving the data flow equation. It may be done through graph reachability in various dependence graphs [5].

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 [6].Program slicing is a program analysis and reverse engineering technique that reduces a program to those statements that are relevant for a particular computation. It is used to assist the programmer in a lot of tedious and error prone tasks such as debugging, program integration, software maintenance, testing and software quality assurance. A program slice is a subset of a program. It provides a convenient way of filtering out irrelevant code.

Slicing Criterion: The pair <s,v> is known as slicing criterion where s is a program point of interest and v is a variable used or defined at s[6].

2. Issues in Program Slicing

The main issues of static program slicing are

- It has to be extended for object oriented programming.
- It requires more memory than the dynamic slicing algorithm i.e. the space complexity of static slicing is more comparatively dynamic slicing.
- The time complexity of the slicing algorithm is also high.
- The static slicing algorithm is limited for particular criteria and cannot be used during the process of program execution.

3. Program Slicing Tools

Different algorithms have been developed to implement program slicing. Each algorithm is language independent. Based on similar or different algorithms many tools have been developed and program slicing techniques to demonstrate its usefulness.

The earliest developed tool was Wisconsin Program Slicer (“Wisconsin Program Slicing”, 1996). It was based on the system Dependence graph (Horwitz, Reps and Binkley, 1988) for inter-procedural slicing. The tool can be used to perform forwards and backwards slicing of C programs however it only supports static slicing. The initial version of Wisconsin Program Slicer was distributed freely. Later it was taken over by Gramma Tech (“static Analysis”2000), Inc. The Slicing Tool consists of a package for building and manipulating control-flow graphs and program dependence graphs. There are two versions of the Wisconsin Program-Slicing Tool: an *interactive version*, which allows a user to invoke various operations and view the results; and a *batch version*, which can be used to report statistics about a program.

Another tool is Unravel ("The Unravel, 1988").It supports only static backward slicing of C programs. It runs under a Unix/Linux environment. Unravel is a prototype "program slicing" tool that can be used to statically evaluate ANSI C source code.Unravel should compile and run on most UNIX systems that have the MIT X Window System.

Kansas State University had developed a slicing tool for JAVA program. It was published as Indus (Ranganath and Hatcliff, 2007). Indus program slicer has been presented as an Eclipse plugin called Kaveri (Jayaraman, et al, 2005). This slicer supports static forward and backward slicing. It can handle concurrent program codes.

The modularity attribute of Indus allows it to be utilized as command line program, or embedded inside another Java program as sub component. To utilize its power Kaveri has been developed as Eclipse plugin. Eclipse is well known and widely used program development, deployment and analysis tool. Kaveri contributes several features to Eclipse.

i. Viewing the program slice in the Java editor.
ii. Choosing slice criteria.
iii. Chasing dependencies to support program comprehension.
iv. Performing context-sensitive slicing.

4. <u>Types of Program Slicing</u>

There are different types of program slice as described below:

4.1. Static slicing:

Static slicing computes program slices through static data flow and control flow analysis. It is valid for all possible executions of the program. Static slicing is mainly used in program understanding and software maintenance. Static slicing helps to understand those parts of the program that contribute to the computation to the selected function. Static slices cannot be useful in the process of understanding of program execution. This technique uses static analysis to derive slice. That is the source code of the program is analyzed and the slices are computed for all possible input values [7].The advantage of static slicing is that it is easier and also faster to identify a static slice. Static slicing is easier because slicing is done directly from the original source program. Static slicing has also some disadvantages. 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.

4.2 Dynamic slicing

Dynamic slicing was first introduced by Korel and Laski in 1988. It consists of triplet i.e. input, occurrence of a statement, variable. Dynamic slicing assumes fixed input for a program. It is used to identify those parts of the program that contribute to the computation of the selected function for a given program execution (program input). Dynamic slices are much smaller than static slices. Dynamic slicing may be used to understand program execution. The process of understanding of large software slicing tool is important. An intermediate representation of a program is used to understand large software system [8]. A dynamic slice contains all statements that actually affect the value of a variable at a program point for that particular execution. Advantage of dynamic slicing is the run time handling of arrays and pointer variables. Dynamic slicing treats each element of an array handling. Disadvantage of dynamic slicing is that it is slower and more difficult to compute than static slicing.

4.3 Forward slicing

It computes all those parts that might be affected by the slicing criterion, using their dependence on the slicing criterion [7]. Forward slicing computes the code-segment that is affected by the slicing criterion.

4.4 Backward slicing

It contains all the statements of the program those might have affected the variable at the statement given in the slicing criterion [7]. Backward slicing computes the code-segment that affects the slicing criterion.

4.5 Quasi static slicing

Quasi static slicing is the hybrid of static and dynamic slicing. Static slicing is examined during compile time. In static slicing no information is required about the input variables of the program. Dynamic slicing analyses the code and gives input to the program. It is constructed at run time at a particular input. Dynamic slices are smaller than static slice. The quasi static slicing was first introduced by Venkatesh [4]. The value of some variables is fixed in quasi slicing and the value of other variables varies when the program is analyzed. With respect to the slicing criterion the behavior of the original program is not changed. Slicing criteria includes the set of variables of interest and initial conditions. Hence quasi slicing is called as conditioned slicing [9].

4.6 Amorphous Slicing

Amorphous slicing is based on preserving the semantics of the program. In amorphous slicing any program transformation technique preserves the semantics of the program with respect to the slicing criterion. Amorphous slicing is used in program comprehension, analysis and reuse [10].

4.7 Chopping

In chopping, two sets of variables source and sink are selected by slicing criterion. After selecting, it computes all the statements in the program. So chopping is a generalization of both forward and backward slicing. Chopping is used to detect those statements that "transmit effects" from one part of the program (source) to another (sink) [11].

5. Applications of Program Slicing

Program slicing technique is used in various applications. Program slicing technique was developed to realize automated static code decomposition tools.

5.1. Debugging:

In a program finding bugs is always a difficult job. To locate a bug in a program the process is to run the program several times, learn more and narrow down the search each time till the bug in the code is finally located. To find a bug in distributed system is more difficult due to control and data dependencies [12]. While debugging code, programmers mentally compute slice [13]. Program debugging is a key application of slicing techniques.

5.2 Software Maintenance:

Software maintenance is a costly process. Many challenges are appear in the software maintenance. The challenges are to understand various dependencies in existing software and to make changes to the existing software without introducing new errors. One problem in software maintenance is that of the ripple effect i.e. whether a code change in a program will affect the behavior of other codes of the program. Solution of this problem is, it is important to know which variable will be affected by a modified variable [14].

5.3. Testing:

In testing program slicing is very useful. It is helpful in reducing the number of test cases required to debug a program. Regression testing is often carried out by software maintainers. It deals with retesting of software after modification. When apiece of code is changed, extensive tests may be necessary which involve running a large number of test cases. This requires generating new test cases along with the existing test cases. To reduce the number of test cases slicing can be used.

5.4. Program Integration:

The relatedness of the code of some component is measured by cohesion [15]. A software component is highly cohesive that has only one function which cannot be further divided in to sub modules. Bieman and Ott [16] define data slices to consist of data tokens. Variables of constant definitions and references may be data tokens. Each output of a procedure is computed by data slices. The connection between the slices are the tokens that are common to more than one data slice, they are called glue. The glue binds the slices together.

6. Literature Reviewof Program Slicing Technique

S Jain et al, 2013proposeda new approach of program slicing is used known as mixed approach of static and dynamic slice(S-D Slicing).Researchers used Object Oriented concepts in C++ language. This concept reduces the complexity of the program and simplifies the program for various software engineering applications like program debugging. The main feature of this approach is to generate dynamic slices in a faster way by using object oriented concepts like classes, inheritance etc. [17].

N. Sasirekha et al, 2011 discussed about various types of slicing techniques like the static slicing, dynamic slicing, quasi static slicing, amorphous slicing, forward and backward slicing. Each category of slicing is explained with separate program which was developed in java language. The applications of slicing in various areas like debugging, cohesion measurement, comprehension, maintenance, re-engineering and testing are highlighted. The analysis can be extended in case of Concurrent Object Oriented programs, Distributed Object Oriented Programs [10].

D W Binkley et al, 2006 proposed a technique known as partial evaluation. It is also known as mixed computation. It was considered as an extensive of compiler theory. It is used to improve efficiency, create compilers and compiler generators from interpreters. Many programs contain computations at compile time. Optimizing compilers exploit such computations by pre computing their values. Partial evaluation takes this process one step further and allows some inputs to a program to be statically bound. According to [18] a residual program produced by partial evaluation is semantically equivalent to a conditioned slice for terminating program.

H Hang et al, 2007 proposed Modular monadic slicing approach. The monadic slicing algorithm allowed that program slices could be computed directly on abstract syntax. Using this algorithm there is no need to construct intermediate structures such as dataflow graphs, dependence graphs etc. Modular monadic slicing has excellent flexibility and reusability properties compared to other slicing algorithms. According to [19] the work can be extended as analyze monadic slicing for programs with special features such as concurrent, object oriented, exceptions etc.

T Hongwei et al, 2014 proposed a slicing known as Complexity Measure Based on Program Slicing (CMBPS).It evaluates factor which affect program complexity such as the length of the program, control flow, data flow and data types of output variables. CMBPS also can give expression the interactive relation between programs. According to [20] CMBPS is a well-structured complexity measure as compared to other popular complexity measure. The work can be extended by measurement theory and the real cases.

G Szilagyi et al, 2002 proposed a method known as Constraint Logic Programming (CLP). It is an emerging software technology. It has growing number of applications. In constraint programs data flow is not explicit. Due to this reason the concept of slice and slicing techniques of imperative language are not directly applicable. Researchers provide a basis for defining slicing techniques (both dynamic and static) based on variable sharing. Researchers offer a precise declarative formulation of the slicing problem for CLP. Data flow based slicing technique (Gyimothy and Paakki, 1995) was used for the construction of static slices of CLP programs and dynamic slices of proof trees. Two approaches were proposed to reduce the size of the slices: introducing directionality information and handling of the calling context problem. The future work will focus on the application of slicing techniques to constraint programs debugging[21].

R Raphique et al developed a forward static slicing algorithm. Researchers implemented the algorithm by using Java in net beans IDE on windows platform. Researchers got the result that as the number of statements increase the slice computation time also increases reciprocally. Lex, YACC tool was used. Here the algorithm considered only the arithmetic statements for forward static slicing purpose and the work can be extended in I/O statements, logical statements, control statements, looping statements etc [22].

D Kumar et al developed an approach for creating an intermediate representation of a program in the form of System Dependence Graph (SDG). It is again taken as input for computing the slicing of a program with respect to slicing criterion. The objective is to find the static slice of an input program and taking the SDG graph as input to the two phase graph reachability algorithm. Researchers found that complexity for generating the graph increases with addition in number of function. The work can be extended to create intermediate representation for object oriented programs and can apply also to calculate dynamic slices [23].

7. Summary ofvarious Program Slicing Techniques

Sl. No.	Source	Year	Author	Methodology	Conclusions	Future Scope
1	IJARCCE www.ijarcce.com	2013	S Jain et al.	Mixed approach of static and dynamic slice (S-D slicing) for C++.	Speed up the execution of program slice and reduce the complexity of program slicing.	Can be extensible to other OOP like Java.
2	IJSEA	2011	N. Sasirekha et al.	Review on different slicing techniques and implemented on Java Language	Debugging, cohesion measurement, comprehension etc. are discussed	Control Flow Graph (CGF), Program Dependence Graph (PDG)
3	FAOC	2006	D W. Binkley et al.	Partial Evaluation And Program Slicing	Partial evaluation is semantically equivalent to a conditioned slice.	
4	Sci China Ser F-Inf Sci www.scichina.com	2007	Z Hang et al.	Modular Monadic Slicing Approach	Excellent flexibility and reusability properties	Concurrent, object-oriented, exceptions, and side-effects, by combining slice monad trans-former with existing ones.
5	WUJNS	2014	T. Hongwei et al.	CMBPS	well-structured complexity measure compare to other popular complexity measure	Extended by measurement theory and the real cases.
6	ASE	2002	G. Szilagyi et al.	CLP	Offers a Precise declarative formulation and substantial reduction of the program Slice.	Focus on application of slicing techniques to constraint programs debugging

7		2010	R Raphique et al.	Java in Net beans IDE		I/O statements, logical statements, control statements, looping statements
8		2012	D Kumar etal.	System Dependence Graph	Complexity for generating the graph increases with addition in number of function	Extended to create intermediate representation for Object Oriented Programs and apply to calculate Dynamic Slices.

8. Comparison table basing on Advantages and Disadvantages of different techniques

Sl. No.	Technique	Advantages	Disadvantages
1	Mixed approach of static and dynamic slice	Efficient in terms of execution speed and program complexity.	Applied only for OOP as C++
2	Partial Evaluation And Program Slicing	Conditional program semantics can be identified.	Complexity of slicing is increased.
3	Modular Monadic Slicing Approach	Reusability of slice is possible.	Requires handling of concurrency and having many side effects.
4	CMBPS	Well defined complexity measures.	Requires different measurement theories to handle the complexity lacuna.
5	CLP	Uses declarative formula.	Computation time increases when the number of statement increases.
6	System Dependence Graph	Can be extended for dynamic slicing.	Complexity for generating the graph increases with addition in number of functions.

9. Result of Comparison

From the comparison table and review from different journal papers, static programming slicing is the simplest and more efficient slicing technique with respect to its slicing point and computational method.

10.Conclusion

This review has described various static program slicing techniques but have some of the disadvantages like slicing of I/O statements, control and looping statements. In the proposed work we are going to implement static slicing technique which has the advantages to slice both control and looping statements. The static slicing technique will be carried out using a free and open source tool, StaticSlicer 1.01 to represent the slicing points and will be represented with algorithm and dependency graph for slicing points using different selective, control and iterative statements.

References

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

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

[3] M. Kamkar, Inter Procedural Dynamic Slicing with Applications to Debugging and Testing. PhD thesis, Linkoping University, Sweden. ISBN 91-7871-065-0, 1993.

[4] G.A. Venkatesh, "The semantic approach to program slicing", ACM SIGPLAN Notices, vol. 26, no. 6, pp. 107-119, 1991.

[5] Susan Horwitz, Thomas Reps, and David Binkley "Interprocedural slicing using dependence graphs", Technical Report 756, Computer Science Department, University of Wisconsin, Madison,1998.

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

[7] Tip Frank, "A Survey of program slicing techniques", Journal of programming languages 3.3121-189,1995

[8] Andrea de Lucia. "Program slicing: Methods and applications", International Workshop on Source Code Analysis and Manipulation, IEEE Computer Society Press, pages 142-149, 2001.

[9] Dave Binkley, Sebastian Danicic, Tibor Gyimothy, Mark Harman, A Kos Kiss, Bogdan Korel ,"Minimal slicing and the relationship between forms of slicing" Processing of the fifth IEEE International Workshop on Source Code Analysis and Manipulation(SCAM05), 2005

[10] N.Sasirekha etal "Program Slicing Techniques And Its Application", International Journal of Software Engineering and Application, vol 2,No 3,July 2011

[11] J Silva," A Vocabulary of Program Slicing-Based Techniques", ACM Journal Name, Vol. V, No. N, Month 20YY.

[12] Weiser, M., "Programmers use slices when debugging," Communications of the ACM, vol. 25, no. 7, pp. 446 – 452, 1982.

[13] Agrawal, H., DeMillo, R. A., and Spafford, E. H., "Debugging with dynamic slicing and backtracking," SoftwarePracticeandExperience, vol.23, no.6, pp.589–616, 1993.

[14] Gallagher, K. and Lyle, J., "Using program slicing in software maintenance,"IEEE Transactions on Software Engineering, vol. 17, no. 8, pp. 751 – 761, 1991.

[15] Cheng, J., "Slicing concurrent programs - A graph theoretical approach in Automated and Algorithmic Debugging", AADEBUG'93 (LNCS, ed.), pp. 223 – 240, Springer-Verlag, 1993.

[16] Bieman, J.M.andOtt, L.M., "Measuring functional cohesion, "IEEE Transactions on Software Engineering, vol. 20, pp. 644 – 657, August 1994.

[17]S Jain etal,"A New approach of program slicing: Mixed S-D (static & dynamic) slicing" International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 5, May 2013

[18]David W. Binkley etal, "A formal relationship between program slicing and partial evaluation",Formal Aspects of Computing 18: 103–119,2006.

[19] ZHang etal, "A novel formal approach to program slicing", Sci China Ser F-Inf Sci | vol. 50 | no. 5 | 657-670| Oct. 2007

[20] T Hongwei etal, "Complexity Measure Based on Program Slicing and Its Validation", Wuhan University Journal Of Natural Sciences, Vol.19 No.6, 512-518,2014

[21] G Szilagyi etal, "Static and Dynamic Slicing of Constraint Logic Programs", Automated Software Engineering, 9, 41–65, 2002

[22] R Raphique etal," Forward Static Program Slicing"B.Tech Thesis,NIT Rourkela, 2010.

[23]D K Kumar etal,"Static Slicing of Interprocedural Programs",B.Tech Thesis, NIT Rourkela, 2012