Indonesian Journal of Electrical Engineering and Computer Science
Vol. 13, No. 1, January 2019, pp. 1~6
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v13.i1.pp1-6

A proposed forward clause slicing application

Khalil Awad[1], Mohammad Abdallah[2], Abdelfatah Tamimi[3], Amir Ngah[4], Hanadi Tamimi[5]
[1,2,3]Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan
[4]School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, Malaysia
[5]EYEVIDO GmbH, Koblenz, Rhineland-Palatinate, Germany

Article Info

Article history:

Received Aug 26, 2018
Revised Nov 19, 2018
Accepted Nov 26, 2018

Keywords:

Clause slicing
Clauser forward slicing
Program analysis
Program slicing

ABSTRACT

The Clause slicing technique is static slicing techniques which also have forward and backward slicing methods. The Clause slice criteria are the clause and the clause number. In this paper, we have discussed the Clauser tool the forward clause slicing tool introduce some improvements to it. The Clauser mechanism divides the program code statement into clauses, depending on clause slicing rules, identifies the variables and built-in functions, then slices the clauses regarding the slice criterion that was entered by the user. Comparing to other static slicing techniques the clause slicing is more accurate and precise because it considers all the code in micro-level, where it focuses on every syntax in the code. The Clauser still needs to be enhanced to slice more code features.

Corresponding Author:

Mohammad Abdallah,
Faculty of Science and Information Technology,
Al-Zaytoonah University of Jordan,
Queen Alia airport road, Amman, Jordan.
Email: m.abdallah@zuj.edu.jo

1. INTRODUCTION

Program slicing was introduced in 1979 by Weiser [1] as "a method used for abstracting from computer programs." Regarding the formula; in Program P, the program slice has criteria which are represented as <s, v>, where s is the statement number and v is the variable. With respecting of slicing criteria, the slice includes only those statements of P needed to capture the behavior of v at s [2].

Clause slicing was introduced in 2012 by Abdallah [3] as a static slicing technique, the clause slicing is concentrating on the code syntax of the program, as part of program robustness measurement technique. The Clauser was built to identify the clauses that can be sliced into the program, give them a number, and allow the user to choose one of them to be sliced and return the formed slice. So far, it only slices the variables and built-in functions. But, in the plan, we are intended to make it slice every single clause in the code. The clause slicing considers most of the code syntax words as a potential slicing criterion, which makes it more useful in testing and measuring the program quality.

This research paper has improved the clause slicing tool that introduced in previous works. The clause slicing for function and variables are not automatically done. Therefore, a new level of Clauser was developed and applied.

This paper is divided into five sections; Section 2 will explore related program slicing and tools that applied them. In Section 3, the Clauser model will be described in details showing how it works regarding the clause chosen rules. Section 4 introduces a case study and the evaluation of Clauser. The conclusions and future work in Section 5.

2. RELATED WORK

Static slicing means that the code syntax will be reserved after slicing. In other words, all possible executions of the program are taking into account [4-5]. Static slicing is built by assign a point of interest and delete all irrelative statements to this point [6-8]. A point of interest is the statement to be sliced; it is signed by the variable and line number (V, L). Which called slicing criteria [9-10]. The static slicing can be executable or non-executable [10]. Executable slice means the code that produced after the slicing operation (the slice) can be compiled and run as a program.

Weiser [1, 7-8, 11] introduced the program slicing which is known later as Executable Backward Static Slicing. It is Executable because the slice produced as an executable program. Backward Slicing is computed by gathering statements and control predicts by way of a backward traversal of the programs starting at the slicing criteria [10]. Backward slicing contains the statements of the program which affect the criteria slice, and it answers the question "what program components might affect a selected computation?" [5]

Another form of static slicing is Forward Slicing. Forward Slicing traverse data and control dependence edges in the forward direction and answers the question "what program components might be affected by a selected computation?" [5].

Forward slice captures the effect of its slicing criteria on the rest of the program, and it is considered a kind of flow effect analysis [4, 12]. It contains the set of statements and control predicts that were affected by the computation of the slicing criterion. The Slicing criteria are the same as in backward slicing (V, L) [9, 10], [13-16].

Forward Slicing usually does not produce an executable slice, unlike the backward slicing. Because the challenge caused by Forward Slicing is defining the semantics captured by a forward slice [6, 10, 17]. A decomposition slicing is a slice used to decompose the program into different components. Decomposition slicing is a union of certain slices taken at certain line numbers on a given variable [8, 18].

Decomposition slicing has two parts: The slice, which is the slice criteria, and the complement. The slice "captures all relevant computations involving a given variable" [18], where decomposition slice depends on the variable name only and does not depends on statement number. The complement is the rest of the program code; it also can be considered as a slice that corresponds to the rest of slicing criteria [18].

There are also many other programs slicing techniques. However, in this research we only interested in forward and backward static slicing techniques. Program slicing is widely used for many purposes: debugging [8], maintenance [18], testing [19, 20], detecting dead code [21], measuring program robustness [22-24] and quality[25-27] and many other applications [26, 28]. Therefore, Researchers have tried applying their ideas of using program slicing and advanced tools. Tools of program slicing are developed to slice different programming languages [29]. As we focused in this research on C language slicing, we only listed some slicing tools for C language: CSurf, frama-C, and Wisconsin Program-Slicing.

CodeSurfer [30] is part of CodeSonar technology, GrammaTech's, the development company, aims to automate a source-code analysis tool that finds bugs. CodeSurfer is a program-understanding tool that makes a manual review of code easier and faster. CodeSonar is an automated bug finder that generates a report of defects in the code.

Many programs understanding tools interpret code loosely. In contrast, CodeSurfer does a precise analysis. Program constructs, including pre-processor directives, macros, and C++ templates, are analyzed correctly. CodeSurfer calculates a variety of representations that can be explored through the graphical user interface or accessed through the optional programming API [30].

Frama-C [31] is a code analysis tool which is used for programs written in C programming language only. It supports static slicing techniques; Forward and backward slicing. It also provides a dependency analysis.

Frama-C comes with plugins such as Slicing and Value analysis. Frama-C allows these plugins to collaborate. It also enables the users to insert and run their plugins and connect them with other plugins in Frama-C. However, it still needs to be improved to support other types of slicing such as dynamic slicing.

There is also a program slicing tool called Wisconsin Program-Slicing tool [32]. It can do a Forward Slicing, backward slicing, and chopping. Also, it consists a package for building and manipulating control-flow graphs and program dependence graphs. The Wisconsin Program-Slicing tool is only developed and tested on Sun OS 5.5.1 which make it a less known than previous tools [32].

The srcML [32-33] program is a command line application for the conversion source code to srcML, an interface for the exploration, analysis, and manipulation of source code in this form, and the conversion of srcML back to source code. The current parsing technologies support C/C++, C#, and Java [16, 34-35].

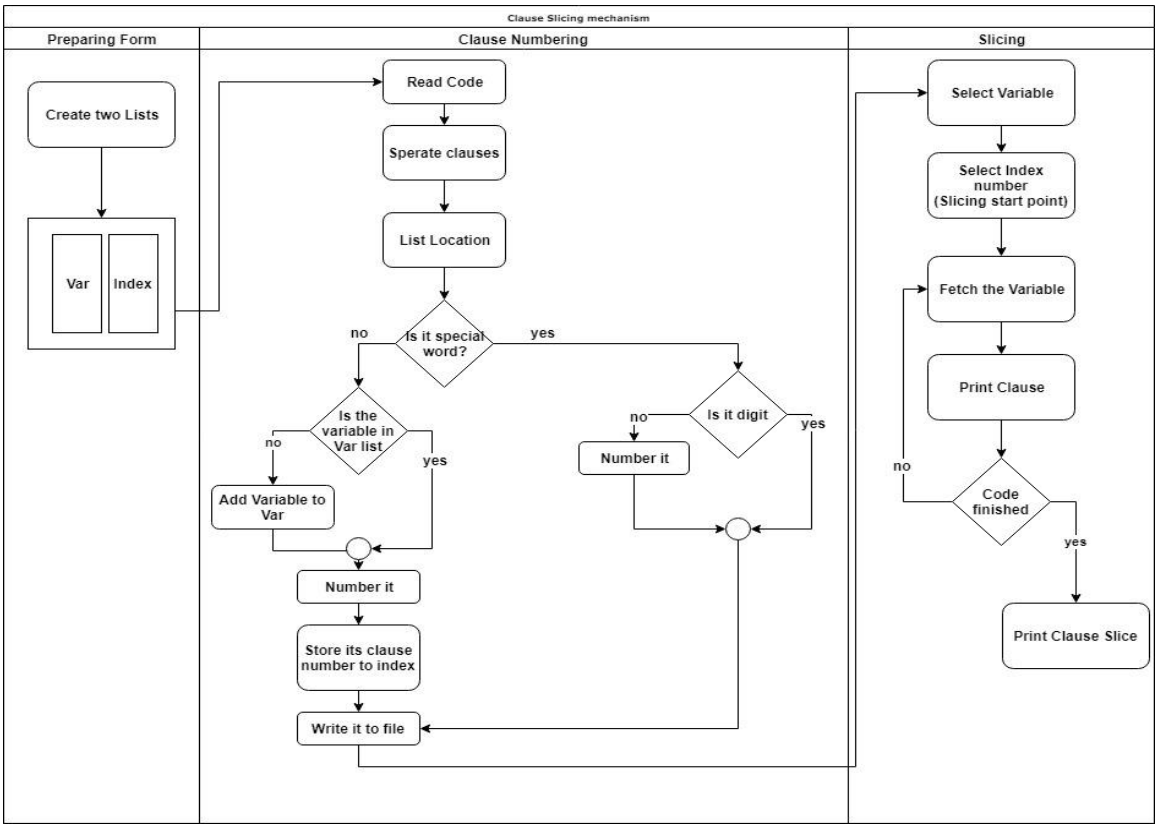

Figure 1. The proposed forward clause slicing model

3. CLAUSER: THE PROPOSED APPLICATION

Clause slicing was introduced in [3, 36] as a slicing technique that interest in a part of the statement that may affect the rest of the slice. The main purpose of Clause slicing was mainly to enhance the software robustness measurement of C programs.

The Clause slicing is a special type of static slicing technique; it is reserved syntax technique. Clause Slicing has the same types of static slicing. It can be forward clause slicing, backward clause slicing, or decomposition clause slicing. In this paper, only the forward clause slice will be discussed.

A Clause is defined as the minimum piece of code that can be sliced [3]. Some clauses that are not sliceable, i.e., #include, and break that called the un-sliceable clauses.

The slicing criteria for the Clause slicing are <C, n>, where C is the Clause, and n is the Clause Number. The Clause slicing (Cn) is all clauses in the program that depends on clause slicing criteria <C, n>.

The first step in the Clause slicing technique is the Clause numbering. It is different than the statement numbering, where not every statement is a clause and vice versa. Therefore, the clauses must be defined depending on some rules that identified in [3, 36]. Then depending on the slicing criteria that identified previously, the slicing will be applied, and the Clause slice will be produced as shown in Figure 1.

In Figure 1, the Forward Clause slicing model is presented. In the first stage, Preparing Form, two lists are created once the Clauser runs: Sliceable; which will be used to store all the sliceable clauses in the target code (the code to be sliced), and Index; which will be used store the all variable clause numbers.

Moving to the stage two, Clause Numbering, where it starts with reading the target code, it must be stored in a text file. Then the clauses will be separated using the rules [3, 36] into keywords, variables, numbers …etc. Then the model starts the following procedure:

List for the location for each index
For each item in the source code
#include → numbering it and write it to a new file
Return → numbering it and write it to a new file
Functions (main, scanf, printf, sqrt) → numbering it and write it to a new file
Digital → write it to a new file
Special character → write it to a new file
Otherwise, the item will be Variables

Check if the variable exists in variable list → numbering it and write it to a new file
If not exists
Add a new variable to variable List
Store its number into variable indices list
Numbering it
Write it to a new file

The third stage, Clause Slicing, starts using the data that was collected previously in stage two.

The Clause Slicing procedure is executed by the user. Where the user is responsible for selecting the slicing criteria (the variable to be sliced and the clause number where to start slicing), then Fetch the selected variable from the start point to the end of the program. If the clauses are sliceable, regarding the rules in [2, 24], then it will be printed to the output files. Otherwise, it will be skipped and not shown in it.

4. EVALUATION

In the previous work [36] the Clauser was introduced as the first tool that applies the Clause slicing technique. It was in the early stages of development. It was not a user friendly tool, where the user has to choose the variable and its number before slicing, and if the user wants to see another slice, he or she has to rerun the program and select the new slicing criteria. The main contribution of the new version of Clauser that introduced Function slicing, where now the user can slice variables and the built-in functions such as scanf and printf.

Moreover, it is more user-friendly. Where the user can select the variable and its index number to see its slice. The user then can change the slicing criteria without reloading the program file, which reduces the execution time and improves the usability.

In addition, the Clauser in the first version was producing two text file, one after the clause numbering that contains clauses and their numbers and the second text file that contains the slicing results. In the new version the Clauser shows the programmed with clause numbered to the user, and on another screen, it shows the results of slicing depending on the chosen slicing criteria.

In the earlier Clauser version, the Clauser only numbered the variables, and the developer has to have renumbered the code taking into account the special words. But in this version, all types of words in the code are numbered. However, in the two version on Clauser, it still slices the variables.

Comparing to another C programs slicing tools such as frama-C and CSurf. The new version of Clauser, in addition to the previous feature, can slice part of the program, with no need to compile or run the program. Moreover, Clauser can number the code even if there are no variables, with the ability to skip the comments. Therefore, it can give an accurate number of clauses in the program.

Clauser as the only slicing tool that runs the clause slicing techniques that makes it useful to be part of measuring the program robustness. So, using Clauser will make the program robustness measuring is fully automated. In addition, the Clauser can be used to measure the quality of the code regarding some standard criteria such as MISRA C [3] that consider every single word in the code syntax. Or can be used to identify the dead code easily.

The Clauser has an advantage that the developer can only number the code without the need to slice it, which can be useful for further practice such as code analysis.

For now, our tool can slice variables in all different positions it could occur at, such as in an initialization statement, as a parameter in a function, in an assignment statement as in the use or the definition condition. In addition, to the built-in functions such as scanf and printf. Thus, the Clauser tool still needs more work to be able to slice more language features.

5. CONCLUSION

Clause slicing is a static slicing technique that was introduced as a slicing technique for part of the code statement or line. The Clauser is the tool that slices a C program using Clause Slicing techniques. Clauser mechanism depends on the Clause Slicing rules of clause slicing ability and numbering. Then, it allows the user to choose the clause or the function to be sliced, then Clauser applies the clause slicing technique and returns the slice of it.

The Clauser can be used to number the code clauses that can be useful in applying rules on these clauses coding standards such as MISRA C. It also helps to analyze the code which has further in code maintenance, regression testing and debugging.

Clauser still needs to be upgraded to apply to all C program features. Now, it only slices the variables and built-in functions. In the future work, the Clauser will be developed to slice all clauses in a C program.

REFERENCES

[1] M. Weiser, "Program slices: formal, psychological, and practical investigations of an automatic program abstraction method," PhD, The University of Michigan, Michigan, 1979.
[2] D. Binkley and K. Gallagher, "Program Slicing," in *Advances in Computers*. vol. Volume 43, V. Z. Marvin, Ed., ed: Elsevier, 1996, pp. 1-50.
[3] M. Abdallah, "A Weighted Grid for Measuring Program Robustness," PhD, *Computer Science*, Durham University, 2012.
[4] X. Baowen, Q. Ju, Z. Xiaofang, W. Zhongqiang, and C. Lin, "A brief survey of program slicing," vol. 30, pp. 1-36, 2005.
[5] K. Gallagher and D. Binkley, "Program slicing," in *Frontiers of Software Maintenance*, 2008. FoSM 2008., 2008, pp. 58-67.
[6] S. Mitra, D. Kim, and M. Fong, "Program Slicing," 2007.
[7] M. Weiser, "Programmers use slices when debugging," *Communications of the ACM*, vol. 25, pp. 446-452, 1982.
[8] M. Weiser, "Program Slicing," *IEEE Transactions on Software Engineering*, vol. 10, pp. 352-357, 1984.
[9] M. Harman and R. Hierons, "An Overview of program slicing," *software focus*, vol. 2, pp. 85-92, 2001.
[10] F. Tip, "A survay of Program Slicing Techniques," *Journal of Programming Languages*, vol. 3, pp. 121-189, 1995.
[11] M. Weiser, "Program slicing," presented at the *Proceedings of the 5th international conference on Software engineering*, San Diego, California, United States, 1981.
[12] B. Sue, "Computing ripple effect for software maintenance," *Journal of Software Maintenance*, vol. 13, p. 263, 2001.
[13] A. d. Lucia, "Program Slicing: Methods and Applications," presented at the *IEEE International Workshop on Source Code Analysis and Manipulation*, 2001.
[14] S. Horwitz, T. Reps, and D. Binkley, "Interprocedural slicing using dependence graphs," *ACM Transaction of Program Language Systms*, vol. 12, pp. 26–60, 1990.
[15] H. Mumtaz, M. Alshayeb, S. Mahmood, and M. Niazi, "An empirical study to improve software security through the application of code refactoring," *Information and Software Technology*, vol. 96, pp. 112-125, 2018/04/01/ 2018.
[16] B. Alokush, M. Abdallah, M. Alrifaee, and M. Salah, "A Proposed Java Static Slicing Approach," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 11, pp. 308-317, 2018.
[17] D. Binkley, S. Danicic, T. Gyimóthy, and M. Harman, "Theoretical foundations of dynamic program slicing," *Theoretical Computer Science*, vol. 360, pp. 23 – 41, 2006.
[18] K. Gallagher and J. R. Lyle, "Using program slicing in software maintenance," *Software Engineering, IEEE Transactions* on, vol. 17, pp. 751-761, 1991.
[19] A. Ngah, M. Munro, and M. Abdallah, "An Overview of Regression Testing," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, pp. 45-49, 2017.
[20] A. Ngah, M. Munro, Z. Abdullah, M. A. Jalil, and M. Abdallah, "Regression Test Selection Model: A Comparison between ReTSE and Pythia," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, 2018.
[21] N. AlAbwaini, A. Aldaàje, T. Jaber, M. Abdallah, and A. Tamimi, "Using Program Slicing to Detect the Dead Code," in *2018 8th International Conference on Computer Science and Information Technology (CSIT)*, 2018, pp. 230-233.
[22] A. Abdalla, M. Abdallah, and M. Salah, "ABrief PROGRAM ROBUSTNESS SURVEY," *International Journal of Software Engineering & Applications*, vol. 8, pp. 1-10, 2017.
[23] M. Abdallah, M. Munro, and K. Gallagher, "Certifying software robustness using program slicing," in *2010 IEEE International Conference on Software Maintenance*, 2010, pp. 1-2.
[24] M. Abdallah, M. Munro, and K. Gallagher, "A Static Robustness Grid Using MISRA C2 Language Rules," presented at the The *6th International Conference on Software Engineering Advances*, Barcelona, Spain, 2011.
[25] K. S. Patnaik and P. Jha, "Proposed Metrics for Process Capability Analysis in Improving Software Quality: An Empirical Study," *International Journal of Software Engineering and Technology (IJSET)*, vol. 1, pp. 152-164, 2016.
[26] M. M. A. Abdallah and M. Alrifaee, "Towards a new framework of program quality measurement based on programming language standards," *International Journal of Engineering & Technology*, vol. 7, pp. 1-3, 2018.
[27] M. M. Abdallah and M. M. Al-Rifaee, "Java Standards: A Comparative Study," *International Journal of Computer Science and Software Engineering*, vol. 6, p. 146, 2017.
[28] A. D. Lucia, "Program slicing: methods and applications," in *Proceedings First IEEE International Workshop on Source Code Analysis and Manipulation*, 2001, pp. 142-149.
[29] T. Hoffner, "Evaluation and comparison of program slicing tools," *Department of Computer and Information Science*, Linkping University, Sweden1995.
[30] GrammaTech. (2009, 12/1/2018). CodeSurfer. Available: http://www.grammatech.com/products/codesurfer/overview.html
[31] P. Baudin, F. Bobot, R. Bonichon, L. Correnson, P. Cuoq, Z. Dargaye, et al., "Frama-C," Frama-C 16 - Sulfur ed: Informations légales et droit de diffusion, 2007.
[32] M.-C. Lee and T. Chang, "Software Measurement and Software Metrics in Software Quality," *International Journal of Software Engineering and Its Applications*, vol. 7, pp. 15-34, 2013.
[33] C. D. Newman, T. Sage, M. L. Collard, H. W. Alomari, and J. I. Maletic, "srcSlice: A Tool for Efficient Static Forward Slicing," in 2016 *IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 2016, pp. 621-624.

[34] M. Abdallah, B. Alokush, M. Alrefaee, M. Salah, R. Bader, and K. Awad, "JavaBST: Java backward slicing tool," in 2017 *8th International Conference on Information Technology (ICIT)*, 2017, pp. 614-618.
[35] A. Ngah and S. A. Selamat, "Using Object to Slice Java Program," *Journal of Engineering and Applied Sciences*, vol. 13, pp. 1320-1325, 2018.
[36] M. Abdallah and H. Tamimi, "Clauser : Clause Slicing Tool for C Programs," *International Journal of Software Engineering and Its Applications*, vol. 10, pp. 49-56, 03/31 2016.

BIOGRAPHIES OF AUTHORS

Khalil Awad is a lecturer at the Department of Software Engineering in Faculty of Science and Information Technology at the Al-Zaytoonah University of Jordan. He received his BSc in Computer Science from Al-Zaytoonah University, Amman, Jordan in 2003 and His MSc in Computer Science from AlBalqa Applied University, As Salt, Jordan in 2007. His research interests are Program Analysis, AI, and web development.

Mohammad Abdallah is an Assistant Professor at the Department of Software Engineering in Faculty of Science and Information Technology at the Al-Zaytoonah University of Jordan. He received his BSc in Computer Science from Al-Zaytoonah University, Amman, Jordan in 2007 and His MSc in Software Engineering from Bradford University, Bradford, the UK in 2008 and Ph.D. in Software Engineering from Durham University, Durham, the UK in 2012. His research interests are Program Analysis, Software Quality, and Software Testing.

Abdelfatah Tamimi is a Professor at the Department of Software Engineering in Faculty of Science and Information Technology at the Al-Zaytoonah University of Jordan. Prof. Abdelfatah is currently the Dean of at the Faculty of Science and Information Technology at the Al-Zaytoonah University of Jordan. He received his BSc in Math from Jordan University, Amman, Jordan and His MSc and Ph.D. in Computer Science from City of New York University, New York, the USA in 1996.

Amir Ngah is an Assistant Professor at the Department of Computer Science at the Al-Zaytoonah University of Jordan. He received his BSc in Computer Science from Universiti Teknologi Malaysia, Johor, Malaysia and His MSc in Software Engineering from Universiti Putra Malaysia, Selangor, Malaysia and Ph.D. in Software Engineering from Durham University, Durham, the UK in 2012. His research interests are Program Analysis and Software Testing.

Hanadi Tamimi is a Software Developer in EYEVIDO GmbH in Koblenz, Germany. She received her BSc in Computer Science from Jordan University, Amman, Jordan in 2014. MSc in Web Science from Koblenz-Landau, Koblenz, Germany in 2017. She works in Several companies in Jordan and Germany. Her research interests are Web development and Eye tracking systems.