Program Slicing with Test Data Generation

Punyapich Limsuwan
April 19, 2016

I. Introduction

Program Slicing is one of the static analysis techniques proposed by Weiser [1] in 1984. The main concept of program slicing is to concern only statements of the program that are involved with variables of interest. The main purpose of slicing is to reduce the complexity of the codes and constrain amount of variables, which allows us to find faults on suspicious variables and statements with less effort than analyzing the entire program. Program slicing is well known as generic method for software debugging, and it can be represented by Control Flow Graph (CFG) and Program Dependence Graph (PDG), which enable us to analyze data-flow and dependencies. However, this static analysis method could help us to analyze data to be generated for testing. As we know that using only random test generator is somehow difficult to find faults in the program, since the data is arbitrary generated, we may need large number of data generations to satisfy the goal, which is time and resource consuming. From this aspect, test data generation could be more systematic if we define a scope and constraint to the test suite by implementing program slicing before generating data. This idea has been proposed by Samuel and Surendran in the paper called "Forward Slicing Algorithm based Test Data Generation [2]." Their approach utilizes dynamic slicing in order to narrow search space and decrease test effort. In dynamic slicing, they concern three parameters, which are variable, point of interest and sequence of input values. This makes the slice smaller than the static one. The direction of slicing implemented in this approach is Forward Slicing. Firstly, slicing criteria need to be defined and the program will be executed once to check if variables are affected. Therefore, the variables that are affected will be included in the slice. Then, slice will be used to identify constraints for generating test data. The algorithm for test data generation requires specific goal statement, which will be satisfied by generated random input from the slice's constraints. In my opinion, this method would be effective because faults could be exposed easier if we have obvious goal and smaller search space rather than digging entire codes aimlessly. Program slicing also helps us track an error on set of variables rather than every variable, which could be difficult to check the statement that cause an error. Meanwhile, similar approach proposed by Zhang et. al further improve their approach by applying genetic algorithm to generate testing data along with static slicing to reduce search space [3]. The results from their approach indicate that slicing could reduce the iterative times to generate test.

II. Project Plan

Since, I have not taken previous software engineering course, CS 562, in Winter term, I have no experience with TSTL like other students who have taken before. I vitally have to learn how to work with TSTL at very first stage of the project. Fortunately, I have taken CS 362, which taught me the concept of static analysis including program-slicing method. I hope implementing

program-slicing is compatible with TSTL and it should neither be too simple nor too difficult. After the program slicing is applied, I will then use the constraints from the slice applying with random test generator, and compare how effective the algorithm is comparing to one without slicing. In addition, I will conduct several tests varying in variables, statements and points of interests. The rest of the plans I will follow the requirement mentioned in the project details.

REFERENCES

[1] M. Weiser, "Program Slicing," in *IEEE Transactions on Software Engineering*, vol. SE-10, no. 4, pp. 352-357, July 1984.

[2] P. Samuel and A. Surendran, "Forward slicing algorithm based test data generation," *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, Chengdu, 2010, pp. 270-274.,

[3] Y. Zhang, J. Sun and S. Jiang, "An Application of Program Slicing in Evolutionary Testing," *2009 International Conference on Information Engineering and Computer Science*, Wuhan, 2009, pp. 1-4.