Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 111 (2017) 355–360

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Advances in Information Technology, IAIT2016, 19-22 December 2016, Macau, China

Application of dynamic program slicing technique in test data generation

Mao yang hong[a]*, Lin ruo qin[b]

[a]*South China Institute of Software Engineering.GU,No 548,GuangCong south road, High tech Industrial Park, Conghua Economic Development Zone,GuangZhou,510990,China*

[b]*South China Institute of Software Engineering.GU,No 548,GuangCong south road, High tech Industrial Park, Conghua Economic Development Zone,GuangZhou,510990,China*

Abstract

The core and key of software testing is test data generation. In the process of generating test data automatically, if the dynamic program slicing technique is used, the efficiency of generating test data can be improved. for generating test data, The main algorithm is as follows： First, in the program, we calculate the dynamic slice of the interest point's variable, and get the current value of the interest point's variable; Then In the branch function, we use the method of minimization, and guide the adjustment of program input. Through practical examples verify that, it is feasible for us to use dynamic program slicing technique in test data generation.

Peer-review under responsibility of the organizing committee of the 8th International Conference on Advances in Information Technology

Keywords: Dynamic program slicing;interest point;test data generation; software testing

1. Introduction

In the source program, the program slicing is composed of some statements, directly or indirectly, the statement may affect the value of a interest point's variable, it contains the static and dynamic slice. In all possible input values, the static slice contains the statement which has an effect on the interest point. In a particular input, the dynamic

* Corresponding author. Tel.: 13926195641.
E-mail address: ccxthxy@126.com

1877-0509
Peer-review under responsibility of the organizing committee of the 8th International Conference on Advances in Information Technology
10.1016/j.procs.2017.06.034

slice contains all the statements which may affect the interest point's variable, It only considers a particular execution path in the program.

Dynamic slicing is widely used, for example, it can be used in the process of debugging, testing, maintenance, etc. The dynamic program slicing technique can be introduced into the software test data generation, because it can improve the efficiency of the test data generation. For dynamic program slicing, it reduces the time of the executing the program, At the same time, it maintains the dynamic behavior of the program executing, According to the dynamic behavior, we can determine the current value of the corresponding branch function, and to guide the test data's adjustment based on value. Through practical examples verify that, it is feasible for us to use dynamic program slicing technique in test data generation.

2. Test data generation process based on dynamic program slicing

Test path. A kind of important software testing method is white box testing, it also called structure testing, it is a kind of test case design method, the box is refers to the testing software, white-box refers to box is visible, the contents and the operation Inside the box is clear. This approach requires a comprehensive understanding of the internal logic structure, testing all logical path, therefore, it also known as exhaustive path testing. When using this scheme, the tester must check the internal structure of the program, starting from the logic of inspection program and test data are obtained. The number of independent paths through the program is very much. In practical testing, even small programs, it is difficult to do. During the test, therefore, We must choose the execution path for testing, the execution path is Representative and most likely to find errors, this is the testing path.

Program path and path set. Suppose path=（$a_s a_{s+1}\ldots a_t$）, it is a sequence of statements in the program P, in the process of program execution, if you meet the execution statement 1, the statement 2 can be run immediately, among them, i=s+ 1,…,t-1. A path from a_s to a_t is the path in the program P, in the program P, we write a collection of all the paths to $path_g$.

Conditional path.In the program P, appear conditional statements if - then - else sequence（$a_s\ldots a_t\ldots a_u$）, and $s\le t\le u$, In order to illustrate the convenience, we assume that a_s is a if conditional statement. If a_s conditions was established, the last statement is a_t of then branch, At this time we call path（$a_s a_{s+1}\ldots a_t$）a conditional path, make a note of $conpath_s$.

Cycling path.In the program P, existence loop statement while (conditional) do or for (conditional) do, The sequence of statements is（$a_s\ldots a_t\ldots a_u$）,and $s\le t\le u$. If a_s is a loop statement, a_t is the last statement in the loop; if exist, a_u is the first statement in the loop in vitro, we call （$a_s a_{s+1}\ldots a_t$）a cycling path, and make a note of $testpath_{st}$.

Test node and non - test node. Program P contains the longest test path $testpath_{st}$. It also has a test path $tsetpath_{ef}$ belong to the path set $path_g$, and it is included in the $testpath_{st}$, and testing path is more than one, that is $tsetpath_{ef} \in path_g$, $tsetpath_{ef} \subseteq testpath_{st}$, $testpath_{st} - tsetpath_{ef} \neq \emptyset$. We called the statement a_i corresponding node test nodes, and statement a_i in the longest path $testpath_{st}$. and make a note of $testnode_{ai}$. Otherwise it is not a test node, it exist relation of more than one to one between test node $testnode_{ai}$ and statement a_i, non test node a_j and statement a_j exist one to one relationship. In general, statements and nodes do not distinguish, Only when there exist a test path.

In the search for a test path, If there exist two or more than two longest test paths, and they have the same start and end nodes, Attach small marks at the $testpath_{st}$, and make a note of $testpath_{st1}$ and $testpath_{st2}$. Here, we need a special note, The test path defined here is different from the traditional sense of "executable path". Such as loop subroutine, Cycle conditions are established and not established there is a test path, And the traditional sense of "executable path" is determined by the number of cycles, there may be numerous. First, we select the partial path in the program as the test path, and then analyze the dependency between nodes. From the above theory, as long as there is a loop or branch structure of the program P, there is a corresponding test path and test node, We can know from the definition of 3 and 2, If the program contains a branch condition or loop condition a_i, then exist two test nodes $testnode_{ai}$ and $testnode_{ak}$ at least.

First of all, in the input field, we enter a data d, and according to the data, then execute the program waiting for the test. We find out the difference about the program's specific path and the actual execution path. Because the predicate's current value is different, the program will be executed along a different path, so the difference will appear in a statement, and the statement is within the scope of the predicate, and the predicate exists in the program. We put the predicate as a interest point s, program input is d, in the program, the variable's subset is v. We construct