doc2.pdf

Daniel Lin

May 2016

1 Introduction

I implemented a better selection criteria for the testing algorithm. Specifically, for each coverage percentage, we determine if the coverage is below a certain threshold. If it is, we re-run the statement with low coverage and see if we can get more coverages out of it. This is a useful test because some statement's coverages are acceptable to some users. The advantage of this testing method is that it can more thoroughly test statements that are below certain coverages and obtain better test results.

The algorithm is as follows: First, we identify the statement with the lowest coverage. Then, we check if the coverage percentage is lower than the user's desired threshold. If it is, then we re-run that branch to see if we have better coverage. To determine the best coverage threshold, we set a global threshold percentage. This threshold percentage is 100. Then, we assign weights to each of the sorted coverage that we obtain by running TSTL until the time limit hits. Then, for each of the coverage, we subtract the global threshold by the coverage percentage. The greater the covered percentage, the smaller the difference. Then, we multiply the difference with the coverage percentage, so that the smallest coverage would get larger weight. Afterwards, we check if the multiplied weighted coverage is greater than the coverage threshold determined by the user. If it is, then the statement is rerunned. If not, then we skip the statement.

2 Motivation

Why is this algorithm a good idea? The reason why this algorithm is a good idea is because it gives users more control as to which statement should be covered more and which statement should be covered less. This is an advantage in testing because this allows users a much more fined grained control over the testing speed, which statement needs to be tested, and faster test and fault generation. For example, if the user sets the user tolerance value as 30 percent, any statements that are above 30 percent are ignored. This is useful for tests that don't need to go into too much detail and if the tester is under time constraint to finish the test quickly. Another useful feature of this type of test is that users might determine that 50 percent coverage is enough for testing. However, current TSTL implementations doesn't allow users to select which statement to cover more and which statement to cover less. Thus, by adding this feature, we allow users to be able to select which statements to cover more. This again gives users a greater flexibility in test generation.

Finally, the new feature allow users to re-test statements that are below coverage. For example, if a statement is below user threshold, the algorithm will re-test that statement and give users better test coverage. This is very useful in test generation because one would always want to identify the branch that is least covered.

3 Improvements

We used difference to assign weight. This time, to account for large statistical errors and to obtain more accurate weight, we use statistical distance to calculate the weight of each coverage and introduce a minimium weight average to account for statistical errors.