Generating Coverage for Regression Testing

Jianwen Dong (jd48784)
Yizhuo Du (yd4788)

Motivation

Software development and maintenance are two major parts of the software systems evolution. After the software is updated, regression testing is applied to the modified version of the software to ensure that it behaves as intended. However, regression testing is costly because, as software grows in size and complexity, a suite accumulates more tests and therefore takes longer time to run. Therefore, for our project, we want to find out a solution to improve the performance of generating the coverage report of regression testing on time consumption by not running all the test cases. This coverage generating system will significantly increase the efficiency of the regression testing of a software, since we don't need to run the entire test suite to obtain the overall coverage.

Feasibility

Our solution is to generate a new coverage report based on regression testing by combining a regression test selection (RTS) algorithm with a coverage re-computing algorithm. The process is as follows:

1. Run the initial program and get the initial coverage information(A matrix "M1[file][line]=boolean", and a collection of test suites "T1").
2. Use a RTS to select the test suites to rerun(A map "Map" from old (file,line) to new (file,line), and a collection of test suites "T2"), and get the coverage information for them(A matrix "M2[file][line]=boolean").
3. For each element in Map which is in T1 but not in T2, add the coverage information in M1 into M2.
4. Generate coverage report.

More detailed algorithms can be found in the paper [Re-computing Coverage Information to Assist Regression Testing].

Evaluation

As for the criteria of evaluating the performance of our technique, we will implement our technique with Java Code Coverage Library (JaCoCo) and generate a set of empirical studies to test the efficiency and accuracy of the technique. As for the testing subjects, we would use a series of testing modules to test the reliability of our tool (GitHub link). For efficiency, we will compare the time consumed to generate the updated coverage report by utilizing our technique and running the whole test suite. For accuracy, we will calculate the rate of the test cases that do not need to be rerun and would not be selected with accurate, updated coverage data (i.e. false positive rate) and the one of test cases that should have been selected because they go through changes but were not (i.e. false negative rate).