Project Progress Report

Robin Zhang, lz2811@columbia.edu

Project Description

Automatic Program Repair (APR) is massively adopted by the developers by automatically generating different patches so that the software could function without eliminating the expected features. But such repair has limitations: their patches is incorrect since the APR is not complete. One significant problem is patch overfitting, which means the APR program generates many patches that exceed the actual test suite. Thus, the usage of formal verification could be a possible way to mitigate such problems. Formal verification is known for its completeness and the adoption of static analysis, which is a great compensation for APR programs. Thus, I am proposing a tool/framework that evaluates the effectiveness of an APR tool for the developers by analyzing its possible of overfitting.

This program is mainly focused on assessing the overfitting problem on APR tools working on Java programs. The Java developers could see the possibility of a certain APR tool, such as NOPOL, overfitting their program using this framework, which takes different modules from the program and verifies the patches the APR tool generates. OpenJML will be used as the verifier of Java programs in this framework. And the testing examples and benchmark standards are from Defects4J and an open repository[1]. The repository provides 537 buggy sample programs with written formal specifications which can be used on different APR tools. They cover a large number of functionalities normally seen in common Java programs. And these sample programs could be extended to be tested on more Java APR tools used in the repository. But I might modify the main functionalities of the framework I proposed as I get more handy on the code and automatic bug fixing theory.

Value to User Community

This project is mainly designed for developers with some independent projects which are written in Java and have needs to be auto repair. Or if the developers need to pick an appropriate APR tool to fix their bugs before they thoroughly look through the code and try to fix them, this framework would be an excellent help for choosing the APR tool. It might not be a very efficient way at the beginning of the development, but it could save times on using those APR tools especially if the developers pick the best suited one at the beginning. For Java programs, there are a lot of different APR tools and it is hard for the developers to learn their pros and cons and decide one to use, so this framework is aimed at helping them finding the probably the best suited one for their own project by providing an assessment of the overfitting problem of each APR tools on the software program.

Research Questions

1. *How efficient of the framework when assessing each APR tool? And how much time does it save comparing to the situation without using this framework?*

[1] https://github.com/Amirfarhad-Nilizadeh/BuggyJavaJML

This framework is supposed to bring efficiency to developers who need to choose one APR tool for their program. If this framework is computationally hard itself, then value it provides is limited. This assessment also considers the situation if the developer picks the worst APR tool comparing to using the framework and choosing relatively the best one.

2. *How does the program complexity affect the APR tools? How does it affect the performance of this framework?*
   Different APR tools might suit different program, but for one program, does this tool still the best suited one to use to fix bugs as the program gets much bigger and complicated. Also, is the complexity the main factor of causing overfitting in APR? Or does the variety of different programs affect the most?
3. *For what features or functions in a program most likely cause each APR tool overfitting problem?*
   At the level of Java APR tools, it could be helpful to the developers if they have a guide or comparison summarization of those tools before choosing. What decides the overfitting results for each APR tool is a clear and precise assessment.

Demo

The elevator pitch would consist of a general overview of APR, then talks about the purpose, the usage, and the target users of the project, and lastly mention my personal motivation.

Then the 5-minute demo will first give an overview of APR overfitting problem and shows how the users could use this framework in their development. Then the slide will show these 3 research questions above along with some tentative data/conclusions in tables/figures, then I will give a brief discussion on those results.

Resource

The testing data are mostly from https://github.com/Amirfarhad-Nilizadeh/BuggyJavaJML and https://github.com/Amirfarhad-Nilizadeh/Java-JML

Benchmark framework: https://github.com/rjust/defects4j

My project framework will be uploaded to my GitHub as a private project.