Jennifer Niederländer

504: Machine Learning meets Program Analysis
Assignment 1

Problems that occurred to me while programming where the following:

1. **Debugging** can be really hard. I had several cases, where I just could not find the error and debugged for hours without a concrete result. Especially if your program has an error that only occurs in some runs of the program, this could be hard. Therefore, automated debugging can really improve everyones programming experience by taking really hard and time consuming work from us. As far as I know, there has already been some work in automated debugging although it is not used regularly [1]. Basically debugging consists of three phases: Fault localization, understanding and solving. For fault localization, there are already many techniques available such as slicing although they are barely used in practice. This is due to them only being effective in small pieces of code. Another problem with this techniques is that they only point out where an error might occur and the user itself needs to find out whether there is a fault in this particular piece of code and how to remove the error. So even if the user localizes an error, he may need to take a look at the other pieces of code that may contain an error as well. I guess this problem is not solved yet, because detecting errors can be really challenging especially since there are so many parts of code that could contain the same error but look totally different depending on the programming style of its programmer. Therefore, I guess the interesting part for a possible project would be helping the user to find errors by improving his understanding and pointing out possible faults more precisely. To accomplish this task, we would need to collect data of as many errors and failures possible and then highlight statements that are likely to have the same issues. In order to do so, the first step would be to develop certain error patterns, we could use in the automated debugging tool, and find out how to apply these to the code that needs to be debugged.

2. **Performance** issues. During writing my bachelor thesis, I had some hard problems with performance. To my knowledge, there was already some work regarding performance testing. These kind of work can tell you whether or not your program has a good performance or not [2]. However, most of the time this is not a problem I have. When my program has bad performance, sooner or later I notice it by simply running it. However, I think some help about finding the performance problem could be very beneficial. It would be really good to have something that points out parts of your code that could be improved regarding performance and also gives hint about how to improve it. The challenge is to not point out every piece of code that may be slow, since sometimes there may not be room for improvement and pointing out to many irrelevant parts will cost the programmer a lot of time, but pointing out the right pieces of the code and deliver information about what to change in order to make the program faster. Maybe it would also be beneficial to have a tool that proposes packages to you that could improve your performance and also help you write your code faster. To accomplish this task, we would need to collect data about performance issues and how they are solved. The next step would be generating patterns and applying them to other programs. Again a main challenge would be the different programming style of different persons. Maybe transferring the code into LTL or boolean algebra could help to solve this problem.

3. **Security** is a relevant topic for every programmer. If your program is not safe, people are not likely to use it and when working for a company, security issues can harm you, your company and your customers. Therefore, some level of security against well known attacks should be considered while writing a program. However, to really ensure security, you will have to test your program against these harmful attacks. This can be challenging because more and more security threads get known and writing tests for all of them can be very hard without the knowledge as well as very time consuming. But since security nowadays becomes more and more important, you cannot skip this part. Therefore, it could be beneficial to have an automated security test suite that checks certain types of programs against well known attacks. Although it is impossible to collect an exhaustive list of security relevant issues that can be

tested against, it would still be possible to detect the most common mistakes. First, we would need to collect security relevant errors and learn patterns from it that can be matched against. Thereby, it can be a real challenge that people have different programming styles and thus the same errors can look different when written by different people. Another question would be whether to learn patterns on the program dependency graph or the control flow graph of programs that contain errors. As far as I know, such tools already exist for web applications, but I think it can be interesting to explore them for a wider range of programs.

Sources

[1] „Are Automated Debugging TechniquesActually Helping Programmers?“ *by Chris Parnin and Alessandro Orso*

[2] „DiPerF: an automated Distributed Performance testing Framework“ *by Catalin Dumitrescu, Ioan Raicu, Matei Ripeanu and Ian Foster*