CS569 (Spring 2016) --- Project Proposal
Combined Static and Dynamic Automated Test Generation in TSTL

Wen-Yin Wang

04/19/2016

Background

I have searched many papers that described different test generation methods, however, automated test generation is most discussed recently. I believe that the basic and simple principles are the best way to implement complex algorithm in test generation. So, I will tend to combine static test generation and dynamic test generation on automatic code-driven test generation to implement my algorithm in this project.

According to Patrice Godefroid's article, "Higher-Order Test Generation (2011)", he wrote that the most important reason of testing is because large programs are imprecise. For example, static test generation is used to read the program code and simulate abstract program executions without ever executing the program, as we know, program may cause faults when it running; dynamic test generation is used to execute a program with given inputs, and it focus on test all possible paths to detect various errors, however, how to produce an input is also a tricky problem. After I saw Godefroid's article, I noticed that no matter what kind of test generation methods has it own pros and cons, and I am wondering that if combining multiple methods to avoid their drawbacks and enhance their benefits. Godefroid also states that the dynamic test generation is more powerful than static test generation only because it has abilities of observing concrete values and recording path constraints, but concrete values affect the efficiency. He indicated the importance and irreplaceable of dynamic test generation. So, I was thinking why not combine static test generation and dynamic test generation.

After I decided to implement an algorithm in both static and dynamic test generation methods, I found out another article, "Combined Static and Dynamic Automated Test Generation (2011)", published by Zhang, Saff, Bu, and Ernst, they states two objectives of testing are high coverage achieving and unknown bugs finding. Providing a confidence code is necessary, which means static test generation is important to testing. To achieve these two goals, they discussed multiple automated test generation techniques and figured out a best way to improve the effectiveness by using formal specification to guide test generation. Unlike random test generation may produces illegal sequences, their novel approach is focusing on creating legal sequences by dynamic analysis and using information on static analysis, on the other words, their approach can be viewed as fuzzing on specific legal paths. Fuzzing is used widely in test generation. Most of fuzzing using random approach, so I am willing to know if narrowed the instances of testing works.

Implement plan
I will implement my test generation in TSTL based on above academic articles, "Combined Static and Dynamic Automated Test Generation (2011)". I am still thinking of what kind of programs need to be tested, but I have some idea about how to implement my program. First of all, reading papers and get idea for implement. When I starting to programming, I prefer to produce input sequences in a function, like a black box that users will not know how it exactly works. In this function code, I will tend to implement it with random approach at first, then narrow down to legal input generate approach. After I finished coding work, I will revise my code to avoiding potential errors. At last, monitoring the output files and revising it more visible outputs including coverage and bugs. I can expect that my program is more efficient than random test generation.

References
[1] Patrice Godefroid, "Higher-Order Test Generation", Acm Sigplan Notices, 2011 Jun, Vol.46(6), pp.258-269. Retrieve from http://delivery.acm.org.ezproxy.proxy.library.oregonstate.edu/10.1145/2000000/1993529/p258-godefroid.pdf?ip=128.193.164.203&id=1993529&acc=ACTIVE%20SERVICE&key=B63ACEF81C6334F5%2E3B580BAC801349E4%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=772127949&CFTOKEN=73664895&__acm__=1460930231_ea9cf85e38634c01bd6897f8cb610c0b

[2] S. Zhang, D. Saff, Y. Bu, and M. Ernst, "Combined Static and Dynamic Automated Test Generation", Software Testing and Analysis: Proceedings of the 2011 International Symposium, (ISSTA '11), pp.353-363. Retrieve from https://homes.cs.washington.edu/~mernst/pubs/palus-testgen-issta2011.pdf