PROJECT PROPOSAL

Junjie Pan

Before I specifically discuss the project plan, it is worth of noting some important concepts and my understanding about them so that I can detail and construct the project plan based on them.

First, we need to clear about the concepts of test generation and techniques related to it. Test generation usually refer to automated test generation, which is the process of automated generation of test cases. And by testing[1], it means the observation of testing processing when by given conditions. Though nowadays most testing processing can be automated in many modern development environment, like visual studio for c, c++ and c# on windows, and g++ for c++ on linux, in most case, it still need manually construct test case to verify and testing many software. Therefore, test generation, or more specifically, automated test generation is proposed to relieve some test case that need large manual processing.

Actually, till now, there are many method to dealt with automated test generation problem, such as model checking method. Based on different application scenarios, different model checking should be applied. The traditional model checking need to abstract the program first, though this is not hard work, the abstraction always have fewer behaviors than the original program itself[1]. For more complex case, more specialize model checking should be applied. However, the most important problem is that even we have many existed methods for test generation, and different strategies of model checking corresponding to different applications, there is one more fundamental thing is the construction of test for system. Yes, even we have test cases but we still can't do anything if we don't have the "machine" to run of test cases. Therefore, we need valid tests for the system we want to test.

Actually, it is well-known obstacle [2] for the adoption of automated test generation and model checking: wring the test generation. However, as it is stated in the paper [2], there are many problems about using traditional method for constructing test harness. Therefore, the authors in this paper [2] present a novel solution: template scripting test language. This is a domain-specific language based test harness, which can make generic test generation and manipulation tools to apply to any SUT. TSTL is based on the understanding of the possible actions the SUT would take. However the most innovative concept is the template scripting. As we know, in in c++, the templates of c++ are the foundation of generic programming [3], which enable many function used in program can be independent from any particular variable type, therefore, can provide functional behavior for any type of inputs. I think the idea of template of TSTL somewhat similar to the spirit of template in c++. In the case of TSTL, the template indicate any testing processing, like writing test generation,or manipulation tool to generate, execute or replay test are independent from the knowledge of details of any SUT themselves, which make the generic testing possible.

It is really great idea.

Therefore, one of my project plan is to how to implement or base on the existing TSTL to many SUT problem, like avltree and so on. What's most important, how to check the TSTL with many model checking problems in [1]. For example, there is one of model checking methods that must be applied to one SUT, how can we write model checking using TSTL to test the SUT without knowing any detail of the SUT. Also, I also have the plan to verify if the TSTL really works for some kind of test generation strategies or automated debugging methods. These are all the questions, I want to explore in the near future.

[1] Automated Test Generation and Verified Software, "Verified Software: Theories, Tools, Experiments", 2005.
[2] TSTL: The Template Scripting Testing Language, ISSTA15
[3] http://www.cplusplus.com/