Richard White

CS 6315 – Automated Verification – Vanderbilt

Project Proposal

Intro:   
For the course project, I would like to do deeper/further research into SMT solvers, such as Z3, and their applications in domains such as test case generation and bug finding. The challenge in test case generation is to sample a relatively small sub-set of the programs’ overall state space, while ensuring that all ‘interesting’ behaviors are explored from inputs.

Currently I’d like to pursue something in between Categories 1 and 4: Category 1 is “Case Study”, while

Category 4 is “implement/develop novel algorithm applicable to an interesting class of systems.

Details

Test Case Generation:

At a high level, a given function is parsed, and inputs are generated based on ‘symbolic simulation’, or representation on an execution path. Test cases are constructed by satisfying conditions which steer the code execution along the different branches in function (such as while, if, else blocks).

I would like to write/generate a few example functions, likely in python programming language, and apply an SMT/tool to analyze the source code for using formal methods/reduction to pure logic in aim of identifying test cases that cover all relevant behavior.

If all goes well, I’d like to dissect what’s happening in building the abstract/symbolic model, and attempt to replicate by designing a light-weight, (toy) algorithm, and compare and contrast aspects of the home-made implementation vs. the more formal existing solver.

In addition to Z3, I am considering applying another tool, such as

Additional Areas of Research Interest:

I’m also interested in finding other software focused use-cases for symbolic model checking, including in Security. It seems this approach is similar to the one stated above and can help identify inputs to guide programs/function execution towards bugs.

I’ve looked ahead a little bit and see some of this will be covered in regard to Dynamic Analysis/Specification Inference. So being able to generate a formal model from existing code to determine if the code is valid.

Other Thoughts:

I’d like to get started with some sample/toy code, and be able to demonstrate usage with an existing tool such as Z3, and then extend to the home-grown implementation.

From there, it may be fun to attempt to apply to methods to a project such as the linux kernel, or some other large open source project.

Non-course References:

<https://fm.csl.sri.com/SSFT14/smt-application-chapter.pdf>