Outline for Hoare Logic Proof Assistant

Bowen Zhang and Zhechen Li

Key Laboratory of High Confidence Software Technologies (MOE),
School of Electronics Engineering and Computer Science,
Peking University, Beijing 100871, China

Abstract. Using these methods, the partial correctness of BCSSs management programs can be proved.

Keywords: Separation Logic, Modeling Language, Formal Verification, Cloud Storage Systems

1 Induction

Computer programs tend to become involved in a growing number of devices. Moreover, the complexity of those programs keeps increasing. Nowadays, even mobile software needs thousands of lines of code to be implemented. One major concern is whether programs behave as they are intended to. Writing a program that appears to work is not so hard. Producting an absolutely correct program is vary difficult. Even one single character wrong among millions of lines of code may cause a bug. If you are playing a game on your cell phone and a bug causes the game to crash suddenly, or even causes the phone to reboot, it will be upsetting but it will not have serious consequences. However, think of the consequences of a bug occurring in the control program of an airplan, or a bug in the control system of a nuclear plant.

Testing is often used to expose bugs, enough test cases can guarantee the one hundred percent code coverage. Yet, there is no guarantee that it will expose all the bugs. For example, testing may not expose the bugs like reading a value in memory at an address that has never been allocated by the program. More advanced approaches are needed to prove that programs always behave as intended. Among them, the program verification is the most frequently used method, which can guarantee the correctness of the programs from the mathematical point.

Program Verification In the context of program verification, a program is said to be correct if it satisfies its specification. The specification of a program is a description of what a program is intended to compute, regardless of how it computes it.

[Some introduction of research on logical verification system result.Focus on verification by paper-and-pen. Describe the rigorous and general of program verification. Meanwhile, express implicitly that the reasoning process is complex.]

While several decades of research on this topic have brought many useful ingredients for building verification systems, no tool has yet succeeded in making program verification as an easy thing. Using an interactive theorem prover is an approach to transferring information from the user to the verification tool.

Interactive Verification An interactive theorem prover, also called a proof assistant, allows one to state a theorem, for instance a mathematical result from group theory, and to prove it interactively.In an interactive proof, the user writes a sequence of tactics and statements for describing the reasoning steps in the proofs, while the proof assistant verifies each of those steps is legitimate.There is no way to fool the theorem prover:if the proof of a theorem is accepted by the system, then the theorem is certainly true.The statement ”this program admits that specification” can be viewed as a theorem. So, one could naturally attempt to prove statements of this form using a proof assistant.

Interactive theorem provers can satisfy not only the rigorous but the convenience of verification. It thus have been significantly improved in the last decade.

[Some researchs on ITP] While most results in ITP need users to describe every step of specificaton, it doesn't seem to make verification much easier.
I have re-discovered Hoare Logic while looking for a way to simplify the verification, in which some redundant specification can be skipped by tactics in the theorem prover. In a sense, my approach may be viewed as removing some unnecessary and obvious factors during verification, hiding the technical details while retaining its benefits.

2 The Modeling Language using Coq

3 The Specification Language using Coq

4 An example

5 Conclusion