How to Verify Your Software

Ernie Cohen
Mircosoft Research, Redmond, USA

The primary purpose of these lectures to teach you how to use VCC (a deductive verifier for concurrent C code) to verify the functional correctness of real-world concurrent software – software written to be executed. Functional verification used to be a niche activity, confined to proof-checking experts, and impractical for COTS software. However, thanks to improvements in verification methodology and deduction technology, verifying software is today hardly more expensive than testing it to high industrial quality standards, while conferring other advantages, such as soundness, better scalability, and the ability to check software without an operational executional environment.

There are several reasons to learn how to verify code (rather than learning just the theory of how to verify code):

- Writing verified code is much more fun than writing and thoroughly testing unverified code.
- You might want to write correct code, and verification is today the only practical way to do this.
- The best way to understand the limitations of today's methodology and tools is to try to use them to do useful work.

A secondary purpose of these lectures is to describe some of the design decisions one faces when building an industrial-strength verifier.

Students are encouraged to download VCC from `http://vcc.codeplex.com` and work through the tutorial prior to the summer school. (A second tutorial will hopefully be available in July.) Ambitious students are encouraged to bring small but interesting pieces of C software to verify.

Prerequisites: A working knowledge of C. (Every computing scientist should have this anyway.)

References

[1] `http://vcc.codeplex.com`