Introductory Comments on Program Verification

Program verification approach:

- Proof-based: Involves a proof-calculus that avoids searching over all states (which is infeasible)
- Semi-automatic: Some proof-steps are automatic, others not so
- Property-oriented: Correctness established for certain properties
- For sequential transformational programs: Runs on single processor and transforms inputs into outputs.

Need for Verification:

- Documentation: Specification is a good form of documentation
- Time-saving: Debugging/Testing a big program is time-consuming an introduces local fixes. Verification done in "planning" phase reduces time for development and maintenance.
- Refactoring: formally verified software is easier for reuse, maintenance, et
- Certification audits: Safety critical software such as, flight control, nuclear power control, banking software require certification.

Microsoft's emergent technology combines program verification, testing and model-checking in an integral environment.

Framework for verification

1) Represent informal requirements R in a logic as formula ϕ_R.
2) Write a program P which is meant to realize R in a chosen prog. env
3) Prove that P satisfies ϕ_R.

Step 1 is not always obvious; Step 3 is what we will learn.