Verification of Reactive Systems

Ali Movaghar
Department of Computer Engineering
Sharif University of Technology
Spring 2012
Some related books:


Some Similar Recent courses


- Techniques for Program Analysis and Verification, CS357, Fall 2011, Stanford University, Department of Computer Science. Taught by Dill and Aiken.
Some Similar Recent courses

- **Formal Software Development Methods**, CS477, Spring 2012, University of Illinois at Urbana-Champaign, Department of Computer Science. Being taught by Gul Agha.

- **Computer-Aided Verification**, CS745, Spring 2009, University of Waterloo, School of Computer Science. Taught by Joe Atlee.
Some Similar Recent courses

- Software Verification and Testing, CSC410, Winter 2012, University of Toronto, Department of Computer Science. Being taught by Azadeh Farzan.
Teaching Assistant

To be announced.
Grading Policy

- Programming Assignments: 10%
- Research Presentations: 10%
- Final Research Paper: 20%
- Midterm Exam: 20%
- Final Exam: 40%
A design is a process of getting a (more detailed) realization from a given specification.

A Multi-Level Design

And Validation

An implementation can be viewed as the most detailed realization.
Design and Validation

- **Validation** is a process of ensuring that a realization satisfies its specifications.
- **Validation** is mainly used in system design and implementation.
- Design of a **complex** system may happen in many levels.
- **Implementation** may be viewed as the lowest level of the design.
Verification and Testing

- **Verification** has three main methods: verification, evaluation and testing.
- Verification is a formal mathematical method to prove that a realization satisfies its specifications.
- **Evaluation** is a method for finding how good a system behaves.
- **Testing** is a method of proving that a specification does not satisfy its specifications.
- Testing, verification and evaluation are usually complementary.
So, why not test?

- Testing only shows presence of bugs not their absence!
What are formal methods?

- Techniques for analyzing systems, based on some mathematics.
- This does not mean that the user must be a mathematician.
- Some of the work is done in an informal way, due to complexity.
Formal Methods

Mathematically-based techniques for describing properties of systems

- Provide framework for
  - Specifying systems (and thus notion of correctness)
  - Developing systems
  - Verifying correctness
    - Of implementation w.r.t. the specification
    - Equivalence of different implementations
- Reasoning is based on logic
  - Amenable to machine analysis and manipulation
  - In principle, can verify everything that is true in the system!
    - Given enough time, skill and patience
Formal Verification

- Formal verification seeks to establish a mathematical proof that a system works correctly. A formal approach provides:
  1. A system model (language) to describe the system,
  2. A specification model (language) to describe the correctness requirement,
  3. An analysis technique to verify that the system meets its specification.
Why aren’t FMs used more?

“Formal methods can revolutionize development!”

“Formal methods are difficult, expensive, not widely useful and for safety-critical systems only”
... and one more problem

- Need to know what to build (specification) before you start building
- Unrealistic!
  - May need to discover what to build iteratively
  - Software changes all the time
Formal Methods “Light”

- Partial application of formal methods
  - only parts of systems are specified
- Emphasis on analysis of some properties
  - security, fairness, deadlock freedom, rather than complete verification
- Debugging rather than assurance
- Automation

Most successful lightweight technique:
Model-Checking
Verification Methods

- There are two major methods for verification: **Inductive method** and **Model checking**.

- In *inductive method*, the problem is formulated as proving a theorem in a mathematical proof system.

- In the method of *model checking*, the behavior of the system is checked algorithmically through exhaustive search of all reachable states.
Reactive Systems

- A reactive system is a system whose role is to maintain an ongoing interaction with its environment.
- The family of reactive systems include most of the classes of systems whose correct and dependable construction is to be considered to be particularly challenging, including concurrent and real-time systems, embedded and process control systems, and operating systems.
Reactive Systems Properties

- Reactive systems have usually the following properties:
  - Concurrency
  - Timeliness
  - High dependability requirements
Formal Verification

- Formal verification seeks to establish a mathematical proof that a system works correctly. A formal approach provides:
  1) A system model (language) to describe the system,
  2) A specification model (language) to describe the correctness requirement,
  3) An analysis technique to verify that the system meets its specification.
System Models

- Transition Systems (Automata)
- Process Algebras and their extensions
- Communicating Sequential Processes (CSP)
- Calculus of Communicating Systems (CCS)
- Actors
- Petri Nets and their extensions
- Other more recent models
Specification Models

- Temporal Logics and their extensions
- Linear Temporal Logics (LTLs)
- Computational Tree Logics (CTLs)
- CTL*
- Other more recent models