Formal Verification - Course Introduction[1]

Paul Jackson
Paul.Jackson@ed.ac.uk

University of Edinburgh

Formal Verification
Autumn 2023

[1] Including contributions by Elizabeth Polgreen

Overview

- **Lecturer**: Paul Jackson
- **Lab demonstrators**: Mohan Dantam & Paul Jackson
- **Lecture Schedule**:
  Weeks 1-11, Mondays and Thursdays 15:10-16:00
  - *Monday*: Room 2.12, Appleton Tower
  - *Thursday*: Lecture Theatre 3, Appleton Tower
- **Lab Schedule**:
  Weeks 3-10, Fridays: 11:10-13:00, Room 4.12, Appleton Tower
- **Discussion Forum**: Piazza

Prerequisites for Course

- Students are expected to be familiar with discrete maths at a level similar to our Year 2 Undergraduate *Discrete Mathematics and Probability* course (INFR08031).
- Prior exposure to first-order logic is expected.
- Programming experience in an imperative language such as Java, C or C++ is also essential for handling the material related to software verification.
- Familiarity with Finite-State Automata concepts will be helpful

Assessment

- There are two assessed courseworks, each worth 15% of the overall course mark:
  - **Coursework 1**:
    Handout: Mon Week 4 (9 Oct)
    Due: 12 noon, Mon Week 6 (23 Oct)
  - **Coursework 2**:
    Handout: Mon Week 8 (6 Nov)
    Due: 12 noon, Mon Week 10 (20 Nov)

  Courseworks will largely involve practical work with FV tools

  Additional unassessed exercises will introduce several of the tools. Sample solutions will be provided.
- There is a final exam in Dec 2023, worth 70% of the overall course mark.

  The exam will cover all material from lectures, exercises and courseworks.

How do you know your code is correct?

What is Formal Verification?

- FV is the use of mathematical techniques to verify the correctness of various kinds of engineering systems; software systems and digital hardware systems, for example.

- FV techniques are exhaustive and provide much stronger guarantees of correctness than testing or simulation-based approaches.

- FV is particularly useful
  - for safety, security, and mission critical systems,
  - when failure is very costly,
  - when failure can damage reputation,
  - when system behaviour is highly complex and hard to understand

Software Bugs in the real world - Therac-25 (1980s)

- Radiation machine for cancer treatment
- At least 6 cases of overdoses ($\sim$ 100 times dose)
- 3 patients died
- Source: design error in the control software (race condition)
- Software written in assembly language

Software Bugs in the real world - Ariane-5 (1996)

- Rocket flipped 90 degrees in wrong direction shortly after launch
- Caused by overflow on floating-point to integer conversion
- One of the most expensive software failures ever

Industrial Examples of Formal Verification

- Intel: FV now largely-replaces simulation when verifying microprocessor designs
- Microsoft: 3rd party drivers are must pass FV checks of the absence of concurrency bugs
- Toyota: verification of automotive source code using bounded model checking
- Amazon Web Services: big push on FV. FV used to verify boot-code for EC2, policies for S3 buckets + more

Syllabus

Topics covered will include

- CTL and LTL model checking
- Use of SAT & SMT solvers as reasoning engines
- the BDD data-structure used by many model checkers
- Formal models of software based on operational semantics
- Assertion-based software verification using verification condition generation and SMT solvers
- Take-up of FV by industry and the challenges to its wider adoption

Course Approach

- Practical focus on tools and techniques used today in industry or likely to be used in future

- Introduces the underlying mathematical and automated-reasoning techniques

Course should be of interest to both

- those planning a career in industry areas (e.g. software engineering, security) where FV could be useful, and
- those interested in research in formal verification and automated reasoning.

Tools

- NuSMV, NuXmv model checkers
- MiniSAT, Z3 SMT solver and Z3 python API
- SPARK and Why3 assertion-based software verification tools
- CBMC bounded model checker for C programs
- ...