CPSC 538G: Introduction to Formal Verification and Analysis

Mark Greenstreet

CpSc 538G – Term 1, 2016/17
MW 10:30-12:00, DMP 101

- Formal verification uses algorithms to rigorously prove properties of hardware or software design.
- Formal methods are mainstream in hardware design.
- Formal methods are moving into software engineering.
- Robotics and machine learning are creating new challenges for formal verification.

What we do

A few examples:

- Algorithms: better solvers and reachability methods.
- AI: use SMT for probabilistic reasoning.
- ML: learning proof strategies, reasoning about learning algorithms.
- Robotics: is the robot “safe”?
- Software engineering: verify array accesses, exception handling.
- Concurrency: verify race-freedom, proper locking.
- Security: find exploits, track information flow.
- HW: multi-timed circuits, mixed analog/digital designs.

How we do it

SAT solvers

- SAT is NP-complete, but heuristics work for surprisingly large problems.
- This is the key technology that has made formal methods a mainstay of HW design.

SMT solvers

- Add decision procedures for systems of linear inequalities, polynomials, arrays, data structures, and other domains to a SAT solver.
- This is the key technology that is bringing formal methods into software engineering.

Model Checking: reasoning about all possible behaviours of a program, robot, or hardware design.

Symbolic Execution: generalize from specific cases.

Check-out Formal Verification

- If you're interested in AI, software engineering, distributed computing, theory, parallel computing, or anything else: **Formal methods is for you!**
- Classes Monday & Wednesday, 10:30-12:00, DMP 101.
  - One paper per class.
  - I assume that formal methods are new to everyone in class, we take the time to cover it well.
- Homework: get some hands-on experience with verification tools.
- A course project
  - See slide 2 for some ideas.
  - Tell me what you're interested in, we'll find a topic.
- Web page: http://www.cs.ubc.ca/~mrg/cs538g/index.html