COMP 426:Automated Reasoning
Advanced undergraduate course

Brigitte Pientka

School of Computer Science

McGill University

Outline

- Motivation
- Administrative issues

Motivation

- Hardware and software is pervasively used in many safety-critical applications.

Motivation

- Hardware and software is pervasively used in many safety-critical applications.
- We need to understand how to
  - formally characterize important invariants
  - prove important invariants
  - increase the confidence in software

Motivation

- Hardware and software is pervasively used in many safety-critical applications.
- We need to understand how to
  - formally characterize important invariants
  - prove important invariants
  - increase the confidence in software
- We need tools to
  - specify and implement formal statements
  - automatically verify formal statements
  - catch flaws early in the development

Human costs of bugs

- Software failure can cause loss of lifes.

Human costs of bugs

- Software failure can cause loss of lifes.
- Bugs in medical software
  - 1985: Software-design flaws lead to radiation overdosis in US. and Canadian patients (Therac-25)
  - 1997: Software-logic error causes infusion pump to deliver lethal doses of morphine sulfate
  - 2001: Panamian cancer patients die following overdosis of radiation due to faulty software

Economic costs of bugs

- Software bugs cost north-american economy over 10 billion US Dollars per year.

Economic costs of bugs

- Software bugs cost north-american economy over 10 billion US Dollars per year.
- Bugs in hardware and software systems
  - 1994: FDIV bug (floating point arithmetic) in Intel Pentium processor

    Cost: US 500 million

Economic costs of bugs

- Software bugs cost north-american economy over 10 billion US Dollars per year.
- Bugs in hardware and software systems
  - 1994: FDIV bug (floating point arithmetic) in Intel Pentium processor
    Cost: US 500 million
  - 1996: Ariane 5 space craft crashes 40 sec after take-off due to floating point conversion error
    Cost: US 500 million

Economic costs of bugs

- Software bugs cost north-american economy over 10 billion US Dollars per year.
- Bugs in hardware and software systems
  - 1994: FDIV bug (floating point arithmetic) in Intel Pentium processor
    Cost: US 500 million
  - 1996: Ariane 5 space craft crashes 40 sec after take-off due to floating point conversion error
    Cost: US 500 million
  - 2004: Mars Rover is paralyzed for 5 days. After communication breakdown, it starts continuously rebooting.

Limits of testing

- Testing is good, but not good enough.

Limits of testing

- Testing is good, but not good enough.
- Slow
- Too many possibilities
  2^{160} possible input pairs for floating point adder
  even higher number of states for more complex architectures
- Sometimes even a huge weight of empirical evidence is misleading.

Industrial efforts in automated reasoning

- Harware verification: IBM, Intel, Motorolla, Bell Laboratories, Saab Ericsson, etc
- Software verification examples:
  - Active Missile Decoy 'Nulka' system
  - Hong Kong Mass Transit Railway Corporation
  - Safety analysis for Queensland Rail.
  - Smart Card Verification

This course

- A thorough introduction to modern constructive logic, its numerous applications in computer science, and its mathematical properties.
- Centerpiece : Design and implementation of a succession of theorem provers
- Fundamental principles which lead to effi cient implementations

Warning:

- Speed is not everything!
- I am not interested in performance tuning.
- I am interested in
  - transferrable techniques.
  - correctness.
  - completeness.

Learning outcome

- Good working knowledge of logic
  - Connection between logic and computation
  - Translate informal problems into formal statements
  - Recognize a proof and know how to do it
- Understand the fundamental techniques for designing and implementing automated theorem provers
- Be able to transfer techniques to different logics

Topics

- Natural deduction
- Sequent calculus
- Backwards theorem prover
- Forward theorem prover
- High-level optimizations
- . . .

Homeworks (30%)

- Every thursday – due the following thursday.
- Handwritten assignments
  - Correctness
  - Elegance
- Implementation assignment
  - Functional language: SML
  - I will give a brief tutorial and provide sample code.
  - Books:
    1. L.C. Paulson: ML for the Working Programmer. Cambridge University Press, 1996.
    2. R.W. Harper: Programming in Standard ML (online)

Project (30%)

- After midterm, instead of weekly homeworks students will work on a project.
- You can work alone or in pairs.
- Task: Building a theorem prover
- Requirements:
  1. Project proposal including timeline and goal
  2. Implementation
  3. Experimental evaluation
  4. Weekly progress report
  5. Final project report

Exams (40%)

- Two exams
  1. Midterm (20%)
  2. Final (20%)
- Closed book, one sheet of notes permitted
- Dates to be announced in class

Coordinates

Prof	Brigitte Pientka
e-mail	bp@cs.mcgill.ca
office	205N McConnell
office hours	Wed 2:00pm – 3:00pm

TA	Ahmer Ahmedani
e-mail	ahmer.ahmedani@mail.mcgill.ca
office	234 McConnell
office hours	Tue 10:00am – 11:00am

Course web-page

http://www.cs.mcgill.ca/~bpientka/courses/atp-04

Acknowledgements

This course material was developed by

Frank Pfenning, Carnegie Mellon University