My Info

Send me email: azadeh@cs.toronto.edu

Stop by my office: BA 3252

More on: www.cs.toronto.edu/~azadeh
Motivation

Software Validation: one of the toughest open problems in Computer Science.

Verification has always been derived by academia

very rich theoretical basis

logics, algorithms, calculi, ...

a lot of room for pragmatism

theoretically-motivated heuristics
Outdated List of Bugs

- Northeast blackout
- data race error
- Ariane V Crash (1996):
  - 64 bit to 16 bit conversion
- Pentium FDIV bug (1997):
  - lookup table had mistakes
- Mars Orbiter
  - feet-per-second vs. Newtons-per-second
Therac-25

radiation therapy over-radiated patients

Windows crashed during Gate's presentation in 2006

windows is used to control highly sensitive army carriers (including those that carry thermo-nuclear intercontinental ballistic missiles).
Newer Bugs

- The Heartbleed bug
- Random generator error
- Loads of airline outages
- British Airways (the most recent)
- Loads of news about security breeches
What kind of certification do we get for software these days?
11. EXCLUSION OF INCIDENTAL, CONSEQUENTIAL AND CERTAIN OTHER DAMAGES. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL MICROSOFT OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, BUT NOT LIMITED TO, DAMAGES FOR LOSS OF PROFITS OR CONFIDENTIAL OR OTHER INFORMATION, FOR BUSINESS INTERRUPTION, FOR PERSONAL INJURY, FOR LOSS OF PRIVACY, FOR FAILURE TO MEET ANY DUTY INCLUDING OF GOOD FAITH OR OF REASONABLE CARE, FOR NEGLIGENCE, AND FOR ANY OTHER PECUNIARY OR OTHER LOSS WHATSOEVER) ARISING OUT OF OR IN ANY WAY RELATED TO THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT, THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, OR OTHERWISE UNDER OR IN CONNECTION WITH ANY PROVISION OF THIS EULA, EVEN IN THE EVENT OF THE FAULT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, BREACH OF CONTRACT OR BREACH OF WARRANTY OF MICROSOFT OR ANY SUPPLIER, AND EVEN IF MICROSOFT OR ANY SUPPLIER HAS BEEN ADVISED OF THE POSSIBILITY OF
The GPL

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
My favorite part of “The Good Omens”
... along with the standard computer warranty agreement which said that if the machine 1) didn't work, 2) didn't do what the expensive advertisements said, 3) electrocuted the immediate neighborhood, 4) and in fact failed entirely to be inside the expensive box when you opened it, this was expressly, absolutely, implicitly and in no event the fault or responsibility of the manufacturer, that the purchaser should consider himself lucky to be allowed to give his money to the manufacturer, and that any attempt to treat what had just been paid for as the purchaser's own property would result in the attentions of serious men with menacing briefcases and very thin watches. Crowley had been extremely impressed with the warranties offered by the computer industry, and had in fact sent a bundle below to the department that drew up the Immortal Soul agreements, with a yellow memo form attached just saying: "Learn, guys..."
Opinions?
Our Holy Grail

Make software (more) reliable

Software is a product!

needs industry standards.

A notion of certification for Software

Meanwhile ... make it more reliable

partial validation, intelligent testing, ...

Next generation languages with better validation support.
What is Verification Anyway?

Proving (in a formal way) that program satisfies a specification written in a logical language.

- Formal models for programs.
- Logics for specifications.
- Algorithms for checking the model against the specification.
Example
Example

Initially: \[ \phi(x) : x_1 > 0 \land x_2 > 0 \]

At each iteration of the loop:
\[ I(x', y') : x_1 > 0 \land x_2 > 0 \land y_1 > 0 \land y_2 > 0 \]
\[ \land \gcd(y_1, y_2) = \gcd(x_1, x_2) \]

When done: \[ \psi(x', z) : z = \gcd(x_1, x_2) \]

Use induction to prove the invariant and the specification.
Verification in the Past

In 70s

Proving programs Correct

Floyd, Hoare, Dijkstra, ...

Philosophy: programmers write programs and prove them correct with a prover.

Failed

All or nothing approach: no way to find bugs.

heavily manual ... non–appealing!
Success Stories

- SPIN (Holzmann)
  - Explicit-state model checker
  - Heuristics to control state-space explosion
  - partial order reduction
  - hashing and approximate search
  - specification: LTL/automata
Success Stories

SMV (Started by McMillan), later NuSMV

Symbolic model checker using binary decision diagrams (BDD)

handles large state spaces

heuristics to handle search spaces well

specification: CTL (and later LTL)

by far the most useful for hardware
Success Stories

Big advances in SAT solvers

- zChaff (Princeton)
- can handle formulas with 100000 variables and millions of clauses!

- Boosted the idea of Bounded Model Checking (BMC)

- NuSMV, and other more contemporary model checkers
Success Stories

- The SLAM tool from Microsoft Research (Ball and Rajamani)

- Static Driver Verifier: big breakthrough
  - model checker that validates device drivers against formal spec.

- Key ideas: predicate abstraction, algorithms for pushdown automata, BDDs for boolean programs.
Success Stories

The Metal project from Stanford (Dawson Engler)

Static analysis to find patterns of bad programming in system code

very successful in error finding: hundreds of bugs found in Linux/BSD.
Success Stories

- Light-weight static analysis
- Pointer chasing, data dependency analysis, ...
- Codesurfer by GrammaTech
- Testing tools: PEX by Microsoft, DART, etc.
Correct by Construction

- **Program Synthesis**: produce a program that satisfies a specification
- **Specifications**: logical or examples
- **Algorithms** for performing the synthesis
- **Formal models**: to define state space of viable candidates.
Program Synthesis

End user programming: for those who know zero programming

Example: Excel's Flashfill

Menial Programming Tasks: saving precious programmer time

The reverse Von Neumann

Removing Human Error: removing human error
You will learn ...

- Introduction to Systematic Testing techniques based on program reasoning.
- Become familiar with formal models
  - CFGs, state transition systems, symbolic representations, ...
- Specification of properties
  - Temporal logics (LTL, CTL), assertions, pre-post conditions
Algorithms/techniques for reasoning

- Static analyses: data flow analyses
- Inductive Assertion method
- Model checking for finite state systems

You will teach yourself tools such as:

- NuSMV: a model checker
- SOOT or PAG/WWW: a framework for defining data flow analyses
- Dafny (a theorem prover)
- An SMT solver
A rough outline to the course
End of Intro. Questions?
Your part

- Participate in class and tutorials
  - This means both showing up and getting involved in discussions.

- Do the work
  - It is cliche but: you will get as much as you put into the course.
Text Books, Aids, ...

-No official Text

-A list of helpful references are posted on the course webpage

-Others will follow throughout the term: books, papers, theses, ...

-Three TAs

-They will do most tutorials for you and partially help you use the tools and help you with problem solving.
Prerequisites

Basic knowledge of Automata and Languages, Theory of Computation, Propositional (boolean) logic, First Order Logic, set theory, algorithms, data structures, and programming (in Java)
Now, a word of advice ...
Don’t take this course if ...

- You don’t like logic
- You don’t like theory
- Your knowledge of logic/theory is shaky
- You want to an easy course to satisfy a breath/depth requirement
- You think this is a systems course
Don’t take this course if ...

- You want to an easy course to satisfy a breadth/depth requirement
- You are not self-sufficient at learning new things quickly
- You are bad at working in a team
- It will basically be assumed that you can dig yourself out of a hole with the help of your peers!
The course is adversarial
By Design
and, out of necessity
What does adversarial mean?

- This is an elective 4th year course.
- It will not be as cleanly streamlined as your 1st/2nd year courses.
- There are lectures, but you are meant to learn a lot on your own.
- You are meant to learn to use new tools on your own with shady online documentation.
What does adversarial mean?

Outside lectures, I will be your first manager.

Outside tutorials, your TAs will be like your senior colleagues.

You will be in an environment similar to your future working environment.

It is your 4th year; it really is the time to be comfortable with something resembling the real world!
You should just do your best and not worry about grades!
As in the real word, you will be only compared to your peers
How are you graded?

- No exams.
- A few problem solving assignments and few mini-projects (automatically graded) → 60%
- One group big project → 40%
No Tutorial
On Monday