CS 745 / ECE 725
Computer Aided Verification

Lecture 1: Overview, Motivation, and Outline

Jo Atlee

DC 2337, jmatlee@uwaterloo.ca

Office Hours: Mon 1:00-2:00, Wed 1:00-2:00

http://www.student.cs.uwaterloo.ca/~cs745

Today's Agenda

- Motivation
- Computer-Aided Verification
- Course Outline
- Evaluation
- Class Schedule
- Questions

Motivation

Motivation

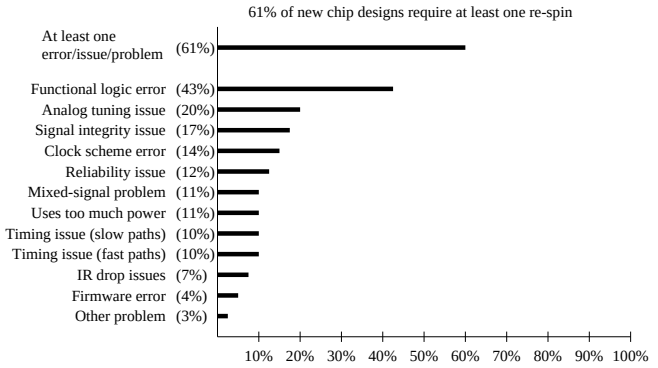
It is widely agreed that the main obstacle to "help computers help us more" and relegate to these helpful partners even more complex and sensitive tasks is not inadequate speed and unsatisfactory raw computing power in the existing machines, but our limited ability to design and implement complex systems with sufficiently high degree of confidence in their correctness under all circumstances.
– Amir Pnueli, Turing Award Winner, in forward to [CGP99]

Software Hall of Shame

Robert Charette, "Why Software Fails", IEEE Spectrum, 2005

YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.
2003–04	AT&T Wireless	Customer relations management (CRM) upgrade problems lead to revenue loss of $100 million.
2002	McDonald's Corp.	The Innovate information-purchasing system canceled after $170 million is spent.
2002	Sydney Water Corp. [Australia]	Billing system canceled after $33.2 million† is spent.
2002	CIGNA Corp.	Problems with CRM system contribute to $445 million loss.
2001	Nike Inc.	Problems with supply-chain management system contribute to $100 million loss.
2001	Kmart Corp.	Supply-chain management system canceled after $130 million is spent.
2000	Washington, D.C.	City payroll system abandoned after deployment costing $25 million.
1999	United Way	Administrative processing system canceled after $12 million is spent.
1999	State of Mississippi	Tax system canceled after $11.2 million is spent; state receives $185 million damages.
1999	Hershey Foods Corp.	Problems with ERP system contribute to $151 million loss.
1998	Snap-on Inc.	Problems with order-entry system contribute to revenue loss of $50 million.
1997	U.S. Internal Revenue Service	Tax modernization effort canceled after $4 billion is spent.
1997	State of Washington	Department of Motor Vehicle (DMV) system canceled after $40 million is spent.
1997	Oxford Health Plans Inc.	Billing and claims system problems contribute to quarterly loss; stock plummets, leading to $3.4 billion loss in corporate value.
1996	Arianespace [France]	Software specification and design errors cause $350 million Ariane 5 rocket to explode.
1996	FoxMeyer Drug Co.	$40 million ERP system abandoned after deployment, forcing company into bankruptcy.
1995	Toronto Stock Exchange [Canada]	Electronic trading system canceled after $25.5 million** is spent.
1994	U.S. Federal Aviation Administration	Advanced Automation System canceled after $2.6 billion is spent.

Motivation



Source: Aart de Geus, Chairman and CEO of Synopsys. Keynote address. Synopsys Users' Group Meeting, Sep 9 2003, Boston USA.

Problems

The development of computer-based systems:

- often run behind schedule or get cancelled
- may result in products that don't do what you expect them to do
- often have subtle bugs. When used in safety-critical applications, these bugs can cost lives or huge amounts of money.

No Guarantees

Because of these problems, most commercial software comes with no guarantees.

Two good references on the challenges of using computers in safety or mission-critical applications:

- Risks Forum moderated by Peter G. Neumann and his book "Computer-Related Risks", Addison-Wesley, 1995.

  `http://catless.ncl.ac.uk/Risks/`
- "Fatal Defect: Chasing Killer Computer Bugs" by Ivars Peterson, Vintage Books, 1996.

Loss of Life: Therac-25

A computer-controlled radiation therapy machine called the Therac-25 made by Atomic Energy of Canada overdosed six people between June, 1985 and January 1987.

In some cases the only indication that something was wrong was the cryptic message:

malfunction 54

The error was a timing problem on data entry.

Loss of Life: Therac-25 (cont'd)

The Therac-25 could deliver radiation as either a beam of electrons or a beam of X-rays. If the operator entered "x" for x rays, the setting of the magnets took 8 seconds. If the operator discovered she had made a mistake and fixed the entry to be "e" within that 8 seconds, even though the screen reflected the change, the change did not affect a part of the program.

See: Leveson and Turner, "An investigation of the Therac-25 accidents" [LT93]

Pentium FDIV bug

The floating point processor of Intel Pentium chips manufactured before a certain date had a bug in the FDIV operation. Approximately 2 million chips had the flaw. The FDIV operation didn't always give precise results. According to sources on the web, the bug could be seen by entering the following formula in the Windows calculator:

$$(4195835/3145727) * 3145727 - 4195835)$$

It produced 512 instead of 0.

Pentium FDIV bug (con'd)

Intel identified the problem in testing, summer, 1994.

Prof. Thomas Nicely, Lynchburg College, first identified in email to colleagues Oct, 1994.

EE Times had an article on it in Nov, 1994.

> "This was a very rare condition that happened once every 9 to 10 billion operand pairs," said Steve Smith, a Pentium engineering manager at Intel.
>
> Intel's Smith emphasized that the anomaly would not affect the average user. Speaking of Nicely, Smith said: "He's the most extreme user. He spends round-the-clock time calculating reciprocals. What he observed after running this for months is an instance where we have eight decimal points correct, and the ninth not showing up correctly. So you get an error in the ninth decimal digit to the right of the mantissa. I think even if you're an engineer, you're not going to see this."
> – EE Times, November 7, 1994, Issue 822, page 1

Pentium FDIV bug (con'd)

Intel offers to replace it for those customers based on "need".

Stock prices fell, great concern.

Dec, 1994, Intel offers to exchange the processor for anyone who asks.

Cost to Intel estimated at $475 million.

Intel hired many people who did formal verification after this incident and they continue to carry out research in hardware verification and are applying these techniques in practice.
See: The Pentium FDIV bug - a Picture
http://www.ipsr.ku.edu/stafffil/hoyle/pentium_fdiv/pentgrph.html The Pentium Papers
http://web.ccr.jussieu.fr/ccr/Documentation/Calcul/matlab5v11/docs/00000/0002d.htm

The Ghost of the Pentium FDIV Bug

From The Risks Digest, Vol. 19, Issue 4, 4 April 1997
(http://catless.ncl.ac.uk/Risks/19.04.html#subj3)

> The ghost of the Pentium FDIV bug
>
> Frank Solomon Fri, 04 Apr 1997 09:09:41 -0500
>
> It seems that the ghost of the FDIV bug lives on in Excel spreadsheets created using Pentiums affected by the problem.
>
> I finally got rid of my Intel Pentium computer with the FDIV bug. Last night, while checking out my new Pentium Pro computer with Microsoft Excel 97 I decided to open up the spreadsheet which demonstrates the Pentium Floating Point divide bug. I was surprised to find that that the calculation of:
>
> 4195835 - (4195835/3145727)*3145727
>
> within the spreadsheet still showed the answer 512 instead of zero.

The Ghost of the Pentium FDIV Bug

> I pressed the recalculate key (F9) to no avail. So then, I retyped the formulas in neighboring cells. Where I had retyped the formulas the answer(s) were correct, but the same formulas that had been saved from when I had done the calculation on the defective Pentium still showed the wrong answer! In other words, here, on the same spreadsheet was the same problem with two different answers, one correct and one incorrect.
>
> The only way I could find to "correct" the incorrect answers was to retype the formulas over the originals.
>
> It seems to me that this "ghost" could represent some risk since it is logical to assume that if you're no longer using a defective Pentium, you needn't be concerned about wrong answers on the spreadsheets you've moved to a new machine. This obviously is not so.
>
> I've sent in a bug report to Microsoft as of this morning.
>
> Frank Solomon

Banking

... February 1994, automated teller machines (ATMs) at Chemical Bank in New York City mistakenly deducted a total of approximately $15 million dollars from about a hundred thousand customer accounts. Until the problem was discovered, any customers making withdrawal were charged double the withdrawal's actual amount on their accounts, although the printed transaction slip showed the correct amount. Only those people who later check their balance – and knew what it should have been – realized there had been an error. The culprit proved to be a flawed instruction – a single line in updated computer program the company had installed day before the problem surfaced.
– Peterson [Pet96], p. 17

Windows XP

Microsoft released Windows XP on Oct. 25, 2001. That same day, in what may be a record, the company posted 18 megabytes of patches on its Web site: bug fixes, compatibility updates, and enhancements. Two patches fixed important security holes. Or rather, one of them did; the other patch didn't work. Microsoft advised (and still advises) users to back up critical files before installing the patches. Buyers of the home version of Windows XP, however, discovered that the system provided no way to restore these backup files if things went awry. As Microsoft's online Knowledge Base blandly explained, the special backup floppy disks created by Windows XP Home "do not work with Windows XP Home".

Originally from:
http://msnbc.com/news/768401.asp?cp1=1#BODY (link no longer works)

Do you trust your computer system?

The real wonder is that the system works as well as it does.

(Peterson [Pet96], p. 8)

Consider the future applications of computers:

- free flight
- smart homes
- patient monitoring
- automated highways
- etc.

Smart Fridges

- Watch television (remote control)
- Play music, download MP3s
- E-mail or web surfing
- Built-in digital camera
- Leave messages in video or audio
- Keep track of birthdays
- Organize recipes
- Keep track of stored foods and monitor expiration dates
- Self-diagnostics

Originally from:
http://www.lgappliances.com/product.asp?category_id=1 (link no longer works)

Security: SmartCards

Smart cards, the size of a credit card, have a microprocessor and memory, along with a mini operating system. They can run multiple applications, which may be downloaded after the card is in use. (This feature is currently disabled.) These "applets" can carry out various functions such as being an "electronic wallet", carry health information, etc.

Because of the high security considerations, a European project is attempting to verify the code and operating system of these cards for "non-interference" between applications. For these companies "security is their product".

See:
Verificard: `http://isabelle.in.tum.de/verificard/`

Computer-Aided Verification

What are "Formal Methods" (FM)?

In a nutshell, formal methods strive to provide for computer-based systems (digital-hardware / software) what engineering mathematics has provided in other fields of engineering:

- the ability to express specifications precisely,
- the ability to clearly define when an implementation meets the specification ("correctness"), and
- the ability to understand the specification and the implementation better through "calculations".

Definition adapted from: course notes, G. Gopalakrishnan.

What is computer-aided verification (CAV)?

My definition: computer-aided verification is

1. the application of logical reasoning to the development computer-based systems (software, hardware, protocols, web-based distributed applications, . . .), and
2. the use of computer-based tools to aid this logical reasoning.

The goal of CAV is to show by calculation (proof) that a system has certain desirable functional properties such as lack of bugs, safety properties, liveness properties.

CAV provides an analytic side to computer system development.

Why study CAV?

CAV is an area of research whose goal is to create techniques for analyzing systems to find subtle, critical logic and safety errors. As computer-based systems:

- become more complex, and
- are used in mission-critical applications (e.g., space shuttle, radiation treatment, stock market),
- the effort and money spent on testing and maintenance takes up more of the system development cycle,
- many bugs can be traced to faults in the requirements/models,

these techniques are necessary to ensure the development of reliable and safe systems.

These techniques are gradually being incorporated into CASE and CAD tools.

What is logical reasoning?

According to Webster's [Web84], logic is "the science of correct reasoning".

According to the Free On-Line Dictionary of Computing [FrO], "logic is concerned with what is true and how we can know whether something is true".

Using logic, we can reason about a system.

> Reasoning about situations means constructing arguments about them: we want to do this formally, so that the arguments are valid and can be defended rigorously, or executed on a machine
> executed on a machine.
> – Huth and Ryan [HR00], p. 1

Trivial Example of Logical Reasoning

Example:

- If the train arrives late and there are no taxis at the station, then John is late for his meeting.
- John is not late for his meeting.
- The train did arrive late.

Were there taxis at the station or not? Therefore, there were taxis at the station.
– Huth and Ryan [HR00], p. 1

This argument has a structure of facts and a conclusion. The conclusion logically follows from the facts.

Logical Reasoning and Verification

Verification involves checking a satisfaction relation, usually of the form of a sequent:

where $\mathcal{M} \models \phi$

- $\mathcal{M}$ is a model (or implementation) describing the possible behaviours of the system
- ϕ is a property (or specification)
- $\models$ is a relationship that should hold between $\mathcal{M}$ and ϕ – what does it mean for the system to be "correct"?

Logic can be used to express the model, property, and relation, and valid arguments of the logic are used to deduce whether the relation holds for the particular model and particular property.

Verifying Software Models

Using logical reasoning, we can calculate/reason about computer-based systems to determine whether the system has certain desirable properties.

$$\mathcal{M} \models \phi$$

- Model Checking: Given a model $\mathcal{M}$ of software behaviour, do executions of $\mathcal{M}$ satisfy properties ϕ?
- Model Finding: Given a collection of properties ϕ, do there exist models $\mathcal{M}$ that satisfy ϕ. Are all such models acceptable?
- Theorem Proving: Given a model $\mathcal{M}$ of properties (not including ϕ), does the model entail property ϕ?

Verifying Software Models

In formal verification, we check these properties of all possible behaviours of our model of the system. Thus, formal verification is able to do an exhaustive analysis of the model of the system, whereas testing (or simulation) is only able to check a finite number of behaviours. By reasoning symbolically, exhaustive analysis becomes possible.

Note: in formal verification we are checking a model of the system, not necessarily the real system. For example, formal verification doesn't help us find physical defects resulting from the manufacturing of a chip.

Formal Verification

Verification is:

- formal: the correctness claim is a precise mathematical statement – the model and properties are unambiguous
- definitive: verification either proves or disproves the correctness claim; helps identify the errors in the system

as opposed to:

- testing the actual system on selected inputs
- simulating a model of the system on selected inputs
- manually inspecting the system or model of the system

Computer-Aided Verification

Computer-based tools assist us in this task by:

- automating many of the tedious parts of the proof (e.g., decision procedures),
- identifying all cases to be checked (in a case-based proof),
- doing many of the bookkeeping tasks such as keeping track of what still needs to be proved, and
- checking our proof steps.

Mechanized proofs are those that have been carried out in a computer-based tool.

Levels of Formalization

There are various degrees of using formal methods. In order of increased formality:

1. Using formal notations (those with a precise semantics)
2. Using mechanized support for formal notations, such as syntax checkers, and typecheckers
3. Using formal proofs usually with mechanization. This mechanization may be fully automated.

There are also semi-formal notations.

Opinion Slide

This research area is an elegant mix of theory and practise.

The use of formal methods is a key component in the evolution of computer- and software-engineering as a collection of best practices that result in predictable, reliable computer-based systems.

Companies using Formal Methods

- Intel
- Siemens
- BT
- AT&T
- Lucent
- Nvidia
- Nortel
- IBM
- NASA
- Motorola
- Cadence
- Synopsis
- etc.

See:

- Craigen, Gerhart, and Ralston, “Formal methods reality check: Industrial Usage” [CGR95].

Caveats

The use of formal methods does NOT solve all problems.

Formal methods form a part of a methodology that involves:

- “best practice” system development
- consideration of safety from the beginning,
- adequate testing and inspection processes,
- appropriate assessment of the context of the system and,
- the suitability of human interfaces and practises.

Also note that formal methods tools and users can make mistakes.

Course Outline

Course Organization

This course will cover an introduction to computer-aided verification.

- logics,
- logical representations of systems, and
- computer-based support for reasoning in these logics.

We will focus on highly-automated verification techniques, and move towards more general, less-automated techniques, covering

- modelling notations
- automated tools
- how and why the automation works

Course Goals

- To introduce you to the fundamental concepts of formal methods
- To introduce you to the range of topics within formal methods and to provide a map to guide you toward applications and future research in this area.
- To give you hands-on experience with formal methods tools. FM is an area that combines both theory and practise.

Course Goals

To achieve these goals we will:

- cover fundamental topics in detail
- talk a little bit about many other topics
- demonstrate and use formal methods tools

Course Outline

- Outline, Motivation, Overview (today)
- Foundations : propositional and predicate logic
- Major Topics:
  - Temporal Logic and Model Checking
  - Object Models and Model Finding
- Applications: software requirements, software designs, hardware designs

. . . continued

Course Outline (con'd)

- Detailed list of advanced topics
  - Additional model checkers and model finders
  - Decision Procedures for Propositional Logic (SAT, Binary Decision Diagrams, etc.)
  - Decomposition Strategies, Abstraction, and Symmetry Reductions
  - Theorem Proving

Course Outline

In addition there will be:

- Paper Presentations (30 min each): These will be scattered throughout the course.
- Project Proposals: These will be about half way through the course.
- Project Presentations (30 min per group): These will be at the end of the course.

Evaluation (Preliminary)

Assignment 1 (model checking)		15%
Assignment 2 (model finding)		15%
Paper presentation		20%
Project		40%
Presentation	10%	
Report	30%	
Participation		10%

There will not be a midterm or a final exam.

Paper Presentation

You will give a 30min presentation on a research article (25min talk; 5 min for questions).

Evaluation is based on an evaluation form (available on course web page).

- 65% by the instructor
- 35% by your classmates

Next week there will be a list of possible research articles for the presentations on the course web page.

The presentations will be throughout the course.

Project

In the project, you (working in pairs or alone) will verify a small system of your choice using existing tools. (The system should be approximately 1.5 to 2 times the size of the assignments.)

Think of yourself as a formal methods consultant who has the job of verifying a system

- Choose how the system (implementation) should be modelled.
- Choose how the properties (specification) should be described.

Project (con'd)

- Present your recommended approach to solving this problem in your project proposal around the middle of the term. Feedback from me and the rest of class.
- Carry out the specification and verification effort.

Project (con'd)

- Evaluate what you did. (Should they pay you? Did you get results for them?) Rate your solution on:
  - were the specification and implementation notations suitable, sufficiently expressive?
  - how easy was the verification from a user's point of view?
  - how long did it take you to set up the problem?
  - how long did it take learn the tool?
  - how long did it take use the tool for the problem?
  - would you recommend this solution again?
  - any other factors.

Project (con'd)

- Presentation on your results followed by questions
- Write-up (max 10 pages IEEE style file; shorter is fine)

Participation

10% of your final grade will be based on participation. This involves:

- Attendance
- Attention
- Asking questions
- Helpful comments on the presentation evaluation forms.

Web Page and Newsgroup

Lecture notes will be on-line at:
`http://www.student.cs.uwaterloo.ca/~cs745`

Links in the lecture notes are hyperrefs.

This web site will also have links to tools, references, and other course information.

Sources

Course material will be drawn from research articles and the texts. Citations in the notes will point you towards these references. Here are some good references, but you do not need to buy these texts.

- Michael Huth and Mark Ryan. *Logic in Computer Science* Cambridge University Press, 2004. (DC Library: QA76.9.L63 H88 2004 – requested to be on 1 day reserve) (don't forget to check the errata)

  `http://www.cs.bham.ac.uk/research/lics/`
- John Kelly, *The Essence of Logic*. Prentice Hall, 1997.

Sources

- Edmund Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999. (will try to get this on reserve in DC Library)
- Daniel Jackson, *Software Abstractions: Logic, Language, and Analysis*, MIT Press, 2006 (will try to get a copy on reserve in DC library).
- NASA. *Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems Volume II: A Practitioner's Companion.* May 1997. `http://eis.jpl.nasa.gov/quality/Formal_Methods/`

Sources

- Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1996. Second Edition.
- Thomas Kropf, *Introduction to Formal Hardware Verification*. Springer, 1999.
- Carl-Johan Seger, *An Introduction to Formal Hardware Verification*, UBC CS TR 92-13, June 1992.

  `http://ece.uwaterloo.ca/~ece725/Ref`
- Doron Peled, *Software Reliability Methods*, Springer, 2001.

Sources

- Klaus Schneider, *Verification of Reactive Systems*. Springer, 2004.
- B. Berard et al., *Systems and Software Verification*. Springer, 2001.

Equipment

There are several servers available for use in the assignments and projects (all are Intel chips running Debian Linux):

The software used in the assignments will be set up here. Software used in your projects can also be set-up here. The servers are available for remote log-ins.

Formal Methods at Waterloo

If you are interested in computer-aided verification for possible future research, you should consider also attending the Waterloo Formal Methods research group meetings.

Information on these meetings can be found at

`http://www.watform.uwaterloo.ca`

Information on the activities of the research group can be found there also.

To join the WatForm seminar mailing list, see:

`http://www.watform.uwaterloo.ca`

Class Time and Office Hours

- Class Time: 3 hours/wk
  MW 10:30-12:00, MC 2036
- Office Hours: MW 1:00-2:00

Reminder: Ask Questions

ASK QUESTIONS!

Research is about asking questions.
Ask lots of questions.
Ask questions when you don't understand a term.
Ask questions when you don't understand an algorithm or the reasoning leading to a conclusion.
Ask questions when you see a connection between different parts of the course.
Ask questions about anything.

(Of course, you may be asked to answer them yourselves!)

Also, suggestions for how to improve the course are always welcome. Please point out errors in the notes.

References

[CGP99] Edmund Clarke Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[CGR95] Dan Craigen, Susan Gerhart, and Ted Ralston. Formal methods reality check: Industrial usage. *IEEE Transactions on Software Engineering*, 21:90–98, February 1995.

[FrO] Free on-line dictionary of computing. http://foldoc.doc.ic.ac.uk/foldoc/index.html.

[HR00] Michael R. A. Huth and Mark D. Ryan. *Logic in Computer Science*. Cambridge University Press, Cambridge, 2000. DC Library: QA76.9.L63 H88 2000.

[LT93] Nancy G. Leveson and Clark S. Turner. An investigation of the Therac-25 accidents. *Computer*, 26(7):18–41, July 1993.

[Pet96] Ivars Peterson. *Fatal Defect: Chasing Killer Computer Bugs*. Vintage Books, New York, 1996.

[Web84] *Webster's New World Dictionary*. Simon and Schuster, 1984.