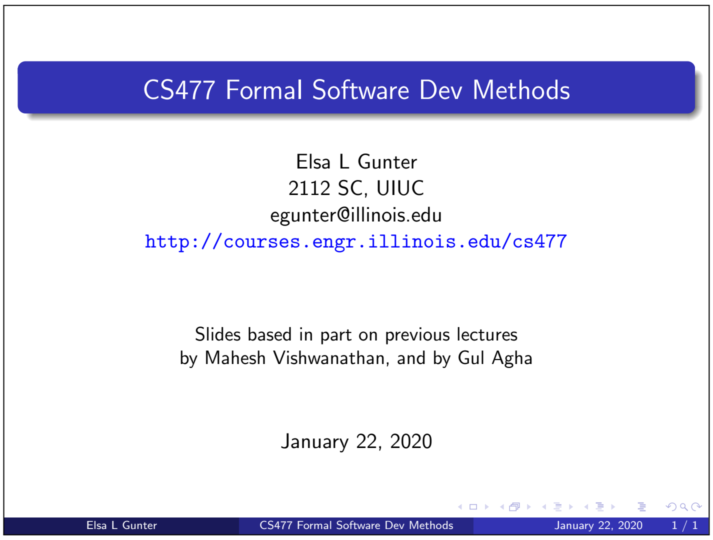
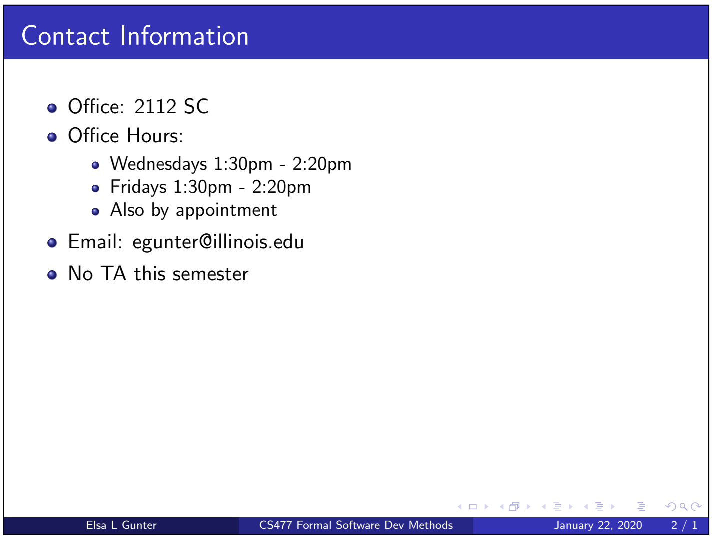


CS477 Formal Software Dev Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

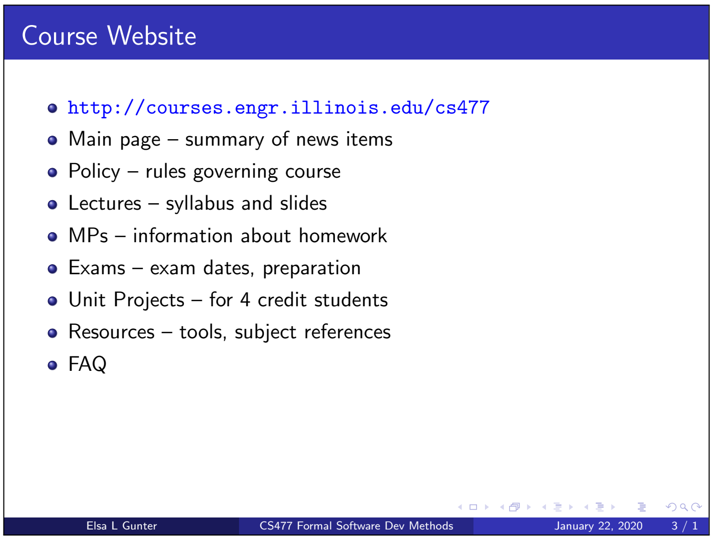
Slides based in part on previous lectures
by Mahesh Vishwanathan, and by Gul Agha

January 22, 2020



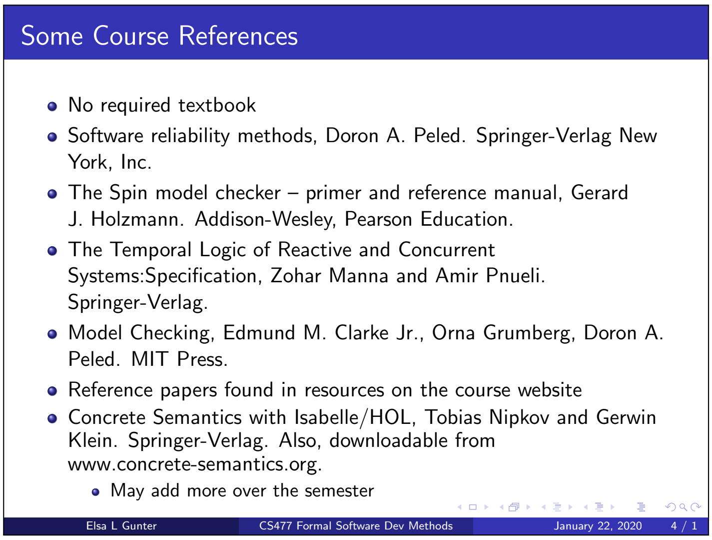
Contact Information

- Office: 2112 SC
- Office Hours:
  - Wednesdays 1:30pm - 2:20pm
  - Fridays 1:30pm - 2:20pm
  - Also by appointment
- Email: egunter@illinois.edu
- No TA this semester



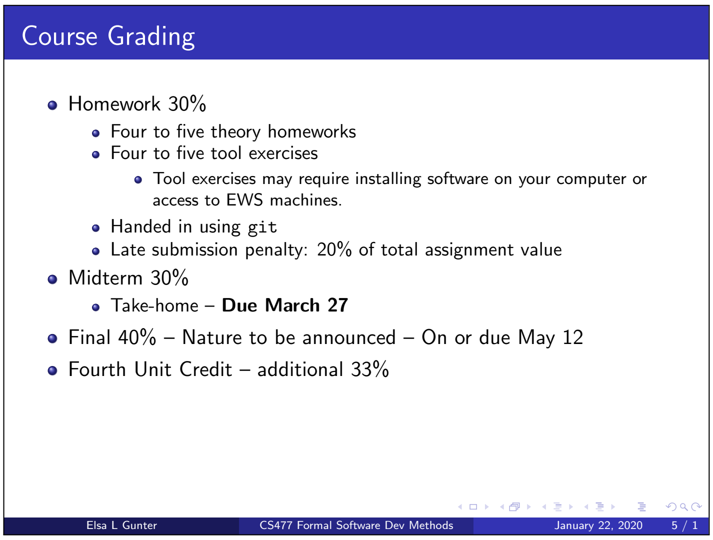
Course Website

- `http://courses.engr.illinois.edu/cs477`
- Main page – summary of news items
- Policy – rules governing course
- Lectures – syllabus and slides
- MPs – information about homework
- Exams – exam dates, preparation
- Unit Projects – for 4 credit students
- Resources – tools, subject references
- FAQ



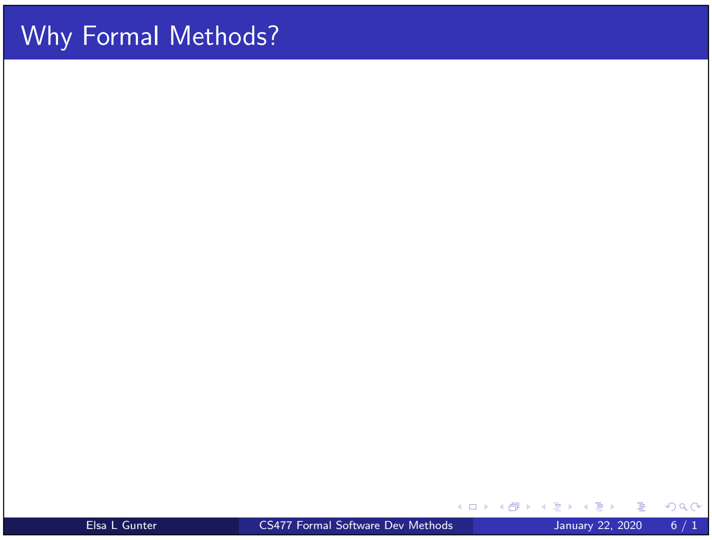
Some Course References

- No required textbook
- Software reliability methods, Doron A. Peled. Springer-Verlag New York, Inc.
- The Spin model checker – primer and reference manual, Gerard J. Holzmann. Addison-Wesley, Pearson Education.
- The Temporal Logic of Reactive and Concurrent Systems:Specification, Zohar Manna and Amir Pnueli. Springer-Verlag.
- Model Checking, Edmund M. Clarke Jr., Orna Grumberg, Doron A. Peled. MIT Press.
- Reference papers found in resources on the course website
- Concrete Semantics with Isabelle/HOL, Tobias Nipkov and Gerwin Klein. Springer-Verlag. Also, downloadable from www.concrete-semantics.org.
  - May add more over the semester



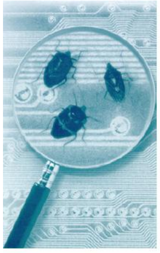
Course Grading

- Homework 30%
  - Four to five theory homeworks
  - Four to five tool exercises
    - Tool exercises may require installing software on your computer or access to EWS machines.
  - Handed in using `git`
  - Late submission penalty: 20% of total assignment value
- Midterm 30%
  - Take-home – **Due March 27**
- Final 40% – Nature to be announced – On or due May 12
- Fourth Unit Credit – additional 33%

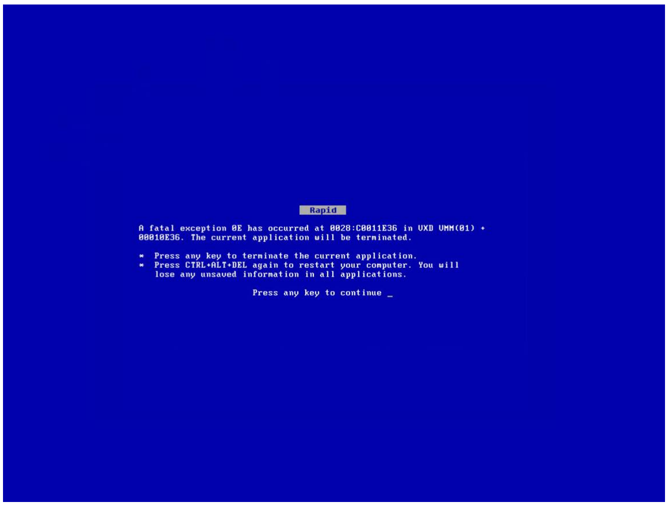


Why Formal Methods?

Why Formal Methods?



- To find bugs.



```
                        Rapid

A fatal exception 0E has occurred at 0028:C0011E36 in UXD UMM(01) +
00010E36. The current application will be terminated.

*  Press any key to terminate the current application.
*  Press CTRL+ALT+DEL again to restart your computer. You will
   lose any unsaved information in all applications.

                 Press any key to continue _
```

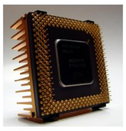
AT&T Network Outage

- 1990: AT&T # 4ESS long distance switch carried all long distance calls in USA, including for Air Traffic Control
- Jan 15, 1990 switch in New York crashes; reboot causes neighboring switches to crash, reboot
- 114 switches caught in oscillating crash - reboot cycle
- Over 60,000 people with no phone service
- No inter-airport ATC communication
  - eventually amateur ham radio help with volunteer network

AT&T Network Outage

- Short-Term Fix: Reload earlier version of 4ESS OS on all switches
- April 1990: AT&T Bell Labs creates new center Computing Sciences Research Center to try to assure never again
  - I was its first employee
- Bug:
  - Many contributing causes
  - One fatal contribution: a misplaced semicolon
  - Could have been caught by a stronger type system

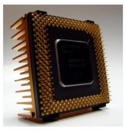
Pentium Chip



- Intel released Pentium in March 1993
- In October 1994, Prof. Thomas Nicely discovers that certain floating point divisions produce errors

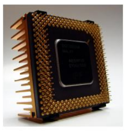
---

Pentium Chip



- Intel released Pentium in March 1993
- In October 1994, Prof. Thomas Nicely discovers that certain floating point divisions produce errors; error in 1 in 9 billion floating point divides with random parameters

Pentium Chip



- Intel released Pentium in March 1993
- In October 1994, Prof. Thomas Nicely discovers that certain floating point divisions produce errors; error in 1 in 9 billion floating point divides with random parameters
- 500 million US dollars + loss of image

Ariane 5 (June 1996)

- Ariane 5 rocket explodes 40 secs into it maiden launch

Ariane 5 (June 1996)

- Ariane 5 rocket explodes 40 secs into it maiden launch due to a software bug!

Ariane 5 (June 1996)

- Ariane 5 rocket explodes 40 secs into it maiden launch due to a software bug!
- A conversion of a 64-bit floating point number to a 16-bit unsigned integer was erroneously applied to a number outside the valid range

---

Ariane 5 (June 1996)

- Ariane 5 rocket explodes 40 secs into it maiden launch due to a software bug!
- A conversion of a 64-bit floating point number to a 16-bit unsigned integer was erroneously applied to a number outside the valid range
- Loss of more than 500 million US dollars

Boeing 777

- Problems with databus and flight management software delay assembly and integration of fly-by-wire system by more than one year

Boeing 777

- Problems with databus and flight management software delay assembly and integration of fly-by-wire system by more than one year
- Certified to be safe in April 1995

Boeing 777

- Problems with databus and flight management software delay assembly and integration of fly-by-wire system by more than one year
- Certified to be safe in April 1995
- Total development cost 3 billion

Boeing 777

- Problems with databus and flight management software delay assembly and integration of fly-by-wire system by more than one year
- Certified to be safe in April 1995
- Total development cost 3 billion; software integration and validation costs were about one-third.

Malaysian Airlines

- A Boeing 777 plane operated by Malaysian Airlines, flying from Perth to Kuala Lumpur in August 2005, experiences problems

Malaysian Airlines

- A Boeing 777 plane operated by Malaysian Airlines, flying from Perth to Kuala Lumpur in August 2005, experiences problems
  - The plane suddenly zoomed up 3000 feet. The pilot's efforts at gaining manual control succeeded after a physical struggle, and the passengers were safely flown back to Australia.

Malaysian Airlines

- A Boeing 777 plane operated by Malaysian Airlines, flying from Perth to Kuala Lumpur in August 2005, experiences problems
  - The plane suddenly zoomed up 3000 feet. The pilot's efforts at gaining manual control succeeded after a physical struggle, and the passengers were safely flown back to Australia.
- Cause: Defective software provided incorrect data about the plane's speed and accelaration.

Malaysian Airlines
Wall Street Journal Analysis

- “Plane makers are accustomed to testing metals and plastics under almost every conceivable kind of extreme stress, but it’s impossible to run a big computer program through every scenario to detect bugs that invariably crop up.”

Malaysian Airlines
Wall Street Journal Analysis

- “Plane makers are accustomed to testing metals and plastics under almost every conceivable kind of extreme stress, but it’s impossible to run a big computer program through every scenario to detect bugs that invariably crop up.”
- “. . . problems in aviation software stem not from bugs in code of a single program but rather from the interaction between two different parts of a plane’s computer system.”

Malaysian Airlines
Wall Street Journal Analysis

- “Plane makers are accustomed to testing metals and plastics under almost every conceivable kind of extreme stress, but it’s impossible to run a big computer program through every scenario to detect bugs that invariably crop up.”
- “. . . problems in aviation software stem not from bugs in code of a single program but rather from the interaction between two different parts of a plane’s computer system.”
- “. . . Boeing issued a safety alert advising, . . . , pilots should immediately disconnect autopilot and might need to exert an unusually strong force on the controls for as long as two minutes to regain normal flight.”

Why Formal Methods?

- To catch bugs
- To eliminate whole classes of errors
- Contrast: Testing

Testing	**Formal Methods**
Can find errors in systems	Can find errors in systems
Gen works on actual code maybe simulated env	Gen work on abstract model of code and environment
Can’t show errors don’t exist	Can show certain types of errors can’t exist
Can’t show system error-free	Can’t show system error-free

Formal Methods Limitations

- Can be expensive
  - Only used fully on safety-critical system components
- Can only prove model of system satisfies given property (“requirements”)
  - Model may be wrong
  - requirements may be inadequate or wrong

What Are Formal Methods?

- Method of finding errors in
  - Hardware
  - Software
  - Distributed Systems
  - Computer-Human Operator Systems
  - . . .
- Not a way to guarantee nothing will go wrong

What Are Formal Methods?

- Formal Methods are the application of rigorous mathematics to the
    - specification
    - modeling
    - implemetation, and
    - verifcation

  of systems with programmable components
    - Software
    - Hardware
    - Control Systems
    - Combined Computer - Human Operator Systems, . . .
- via computer programs implementing the math

What Types of Maths?

- Sets, Graphs, Trees
- Automata
- Logic and Proof Theory, Temporal Logics
- Process Algebras
- Induction, especially structural induction and well-founded induction, inductive relations
- Category Theory
- Probability
- . . .
- Differential Equations, PDEs
- . . .

What Types of Tools?

- Type Checkers, Type Inference
    - Java, ML (Ocaml, Standard ML), Haskell, . . .
- Model Checkers, SAT solvers
    - SPIN, NuSMV, Mocha, SAL, . . .
- Interactive Theorem Provers
    - Isabelle, Coq, HOL4, PVS, . . .
- Runtime Monitoring
    - JavaMOP

Course Overview

- Review of basic math underlying most formal methods
- Intro to interactive theorem proving
    - Intro to Isabelle/HOL
- Floyd-Hoare Logic (aka Axiomatic Semantics)
    - Verification Conditions
    - Verification Condition Generators (VCGs)
- Operation Semantics
    - Structured Oper. Sem., Transition Sem.
- Models of Concurrency
    - Finite State Automata, Buchi Automata

Course Overview

- Temporal Logics
    - LTL
- Model Checkers
    - Spin
- Abstract Interpretation
- Type Systems
    - Type Systems as Abstract Interpretation

Course Objectives

- How to do proofs in Hoare Logic, and what role a loop invaraint plays
- How to use finite automata to model computer systems
- How to express properties of concurrent systems in a temporal logic
- How to use a model checker to verify / falsify a temporal safety property of a concurrent system
- The connection between types and propgram properties

Propositional Logic

The Language of Propositional Logic

- Begins with constants $\{\mathbf{T}, \mathbf{F}\}$
- Assumes countable set AP of propositional variables, a.k.a. propositional atoms, a.k.a. atomic propositions
- Assumes logical connectives: $\wedge$ (and); $\vee$ (or); $\neg$ (not); $\Rightarrow$ (implies); $\Leftrightarrow$ = (if and only if)
- The set of propositional formulae $PROP$ is the inductive closure of these as follows:
  - $\{\mathbf{T}, \mathbf{F}\} \subseteq PROP$
  - $AP \subseteq PROP$
  - if $A \in PROP$ then $(A) \in PROP$ and $\neg A \in PROP$
  - if $A \in PROP$ and $B \in PROP$ then $(A \wedge B) \in PROP$, $(A \vee B) \in PROP$, $(A \Rightarrow B) \in PROP$, $(A \Leftrightarrow B) \in PROP$.
  - Nothing else is in $PROP$
- Informal definition; formal definition requires math foundations, set theory, fixed point theorem ...

Semantics of Propositional Logic: Model Theory

Model for Propositional Logic has three parts

- Mathematical set of values used as meaning of propositions
- Interpretation function giving meaning to props built from logical connectives, via structural recursion

Standard Model of Propositional Logic

- $\mathcal{B} = \{\mathrm{true}, \mathrm{false}\}$ boolean values
- $v : AP \to \mathcal{B}$ a valuation
- Interpretation function ...

Semantics of Propositional Logic: Model Theory

Standard Model of Propositional Logic (cont)

- Standard interpretation $\mathcal{I}_v$ defined by structural induction on formulae:
  - $\mathcal{I}_v(\mathbf{T}) = \mathrm{true}$ and $\mathcal{I}_v(\mathbf{F}) = \mathrm{false}$
  - If $a \in AP$ then $\mathcal{I}_v(a) = v(a)$
  - For $p \in PROP$, if $\mathcal{I}_v(p) = \mathrm{true}$ then $\mathcal{I}_v(\neg p) = \mathrm{false}$, and if $\mathcal{I}_v(p) = \mathrm{false}$ then $\mathcal{I}_v(\neg p) = \mathrm{true}$
  - For $p,\ q \in PROP$
    - If $\mathcal{I}_v(p) = \mathrm{true}$ and $\mathcal{I}_v(q) = \mathrm{true}$, then $\mathcal{I}_v(p \wedge q) = \mathrm{true}$, else $\mathcal{I}_v(p \wedge q) = \mathrm{false}$
    - If $\mathcal{I}_v(p) = \mathrm{true}$ or $\mathcal{I}_v(q) = \mathrm{true}$, then $\mathcal{I}_v(p \vee q) = \mathrm{true}$, else $\mathcal{I}_v(p \vee q) = \mathrm{false}$
    - If $\mathcal{I}_v(q) = \mathrm{true}$ or $\mathcal{I}_v(p) = \mathrm{false}$, then $\mathcal{I}_v(p \Rightarrow q) = \mathrm{true}$, else $\mathcal{I}_v(p \Rightarrow q) = \mathrm{false}$
    - If $\mathcal{I}_v(p) = \mathcal{I}_v(q)$ then $\mathcal{I}_v(p \Leftrightarrow q) = \mathrm{true}$, else $\mathcal{I}_v(p \Leftrightarrow q) = \mathrm{false}$

Truth Tables

Interpretation function often described by truth table

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
true	true					
true	false					
false	true					
false	false					

Truth Tables

Interpretation function often described by truth table

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
true	true	false				
true	false	false				
false	true	true				
false	false	true				

Truth Tables

Interpretation function often described by truth table

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
true	true	false	true			
true	false	false	false			
false	true	true	false			
false	false	true	false			

Truth Tables

Interpretation function often described by truth table

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
true	true	false	true	true		
true	false	false	false	true		
false	true	true	false	true		
false	false	true	false	false		

Truth Tables

Interpretation function often described by truth table

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
true	true	false	true	true	true	
true	false	false	false	true	false	
false	true	true	false	true	true	
false	false	true	false	false	true	

Truth Tables

Interpretation function often described by truth table

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
true	true	false	true	true	true	true
true	false	false	false	true	false	false
false	true	true	false	true	true	false
false	false	true	false	false	true	true