CS 477: Formal Methods in Software Development

Mahesh Viswanathan
vmahesh@cs.uiuc.edu
3232 Siebel Center

University of Illinois, Urbana-Champaign

Spring 2009
Part I

Administrivia
Instructional Staff

- **Instructor**: Mahesh Viswanathan (vmahehs)
- **Office Hours**: 2:30—3:30 on Thursdays, and by appointment
Instructional Staff

- **Instructor:** Mahesh Viswanathan (vmaheesh)
- **Office Hours:** 2:30—3:30 on Thursdays, and by appointment
- And you?
Electronic Bulletin Boards

- **Webpage:** http://www.cs.uiuc.edu/class/sp09/cs477
- **Newsgroup:** uiuc.class.cs477
Textbooks

- Textbook: None!
- Lecture Notes: Available on the webpage after every class
- Additional References: See papers and books mentioned on the website
Textbooks

- **Text-book**: None!
Textbooks

- **Text-book**: None!
- **Lecture Notes**: Available on the web-page after every class
Textbooks

- **Text-book**: None!
- **Lecture Notes**: Available on the web-page after every class
- **Additional References**: See papers and books mentioned on the website
Homeworks and Grading

- Roughly 4 to 8 homeworks.
Homeworks and Grading

- Roughly 4 to 8 homeworks. Will be typically assigned on Tuesday and due the following Tuesday.
Homeworks and Grading

- Roughly 4 to 8 homeworks. Will be typically assigned on Tuesday and due the following Tuesday.
- Will include both machine problems and problems requiring mathematical proofs.
Homeworks and Grading

- Roughly 4 to 8 homeworks. Will be typically assigned on Tuesday and due the following Tuesday.
- Will include both machine problems and problems requiring mathematical proofs.
- Homework grading: Philosophy—Constructive comments on fixing the solution and pointing out flaws in reasoning.
Homeworks and Grading

- Roughly 4 to 8 homeworks. Will be typically assigned on Tuesday and due the following Tuesday.
- Will include both machine problems and problems requiring mathematical proofs.
- Homework grading done by me and you!
Homeworks and Grading

- Roughly 4 to 8 homeworks. Will be typically, assigned on Tuesday and due the following Tuesday.
- Will include both machine problems and problems requiring mathematical proofs.
- Homework grading done by me and you!
  - Philosophy: Constructive comments on fixing the solution and pointing out flaws in reasoning.
Grading Policy: Overview

Total Grade and Weight

- **Homeworks:** 50%
- **Grading:** 15%
- **Finals:** 35%
Part II

Formal Methods: An Overview
What are Formal Methods?

It is about finding bugs.

Viswanathan

CS477
What are Formal Methods?

It is about finding bugs.
A fatal exception OE has occurred at 8020:C0011E36 in UBX 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...
Intel released Pentium in March 1993
Intel released Pentium in March 1993

In October 1994, Prof. Thomas Nicely discovers that certain floating point divisions produce errors
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
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 (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 its 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 its 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
NASA’s Mars Exploration

- Mars Climate Orbiter is lost in September 1999, due to an error in conversion from English to Metric units in navigation software.
NASA’s Mars Exploration

- Mars Climate Orbiter is lost in September 1999, due to an error in conversion from English to Metric units in navigation software.
- Mars Polar Lander’s engine shuts down, in December 1999, due to spurious signals that gave false indication that the vehicle had landed.
NASA’s Mars Exploration

- Mars Climate Orbiter is lost in September 1999, due to an error in conversion from English to Metric units in navigation software.
- Mars Polar Lander’s engine shuts down, in December 1999, due to spurious signals that gave false indication that the vehicle had landed.
- In 2004, Mars Rover freezes due to software failure.
Boeing 777

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

Cause: Defective software provided incorrect data about the plane’s speed and acceleration.
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 acceleration.
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.”
```
“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.”
```
“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.”
Modern Automobiles

500 horses
Modern Automobiles

500 horses, 200 processors!
Modern Automobiles

Saga continues

- Toyota recalls 75,000 Prius cars in May 2005.
Modern Automobiles

Saga continues

- Toyota recalls 75,000 Prius cars in May 2005.
  - Drivers reported sudden stalling at highway speeds
Modern Automobiles

Saga continues

- Toyota recalls 75,000 Prius cars in May 2005.
  - Drivers reported sudden stalling at highway speeds
  - Cause: Software glitch
Modern Automobiles
Saga continues

- Toyota recalls 75,000 Prius cars in May 2005.
  - Drivers reported sudden stalling at highway speeds
  - Cause: Software glitch

- DaimlerChrysler recalls 128,000 Pacifica SUVs because of software problem governing the fuel pump and power train control.
Reliable Engineering
Basis of Reliable Engineering

1. Abstract using suitable mathematics
Basis of Reliable Engineering

1. Abstract using suitable mathematics
2. Over engineering the system to ensure that mathematics predicts reliable behavior in all extreme situations
Mathematics of Software
Mathematics of Software

finite-state control

head

\[ X_1 X_2 \cdots X_n \]

tape
Challenges in Software

Unlike traditional engineering, foundations of software are
Challenges in Software

Unlike traditional engineering, foundations of software are

- not continuous, but discrete
Challenges in Software

Unlike traditional engineering, foundations of software are
- not continuous, but discrete
- sensitive to perturbations
Challenges in Software

Unlike traditional engineering, foundations of software are
- not continuous, but discrete
- sensitive to perturbations
- difficult to over-engineer
Challenges in Software

Unlike traditional engineering, foundations of software are
- not continuous, but discrete
- sensitive to perturbations
- difficult to over-engineer
- difficult to abstract
A program
A program

```c
/* implementation for block/process pic2channels */
void pic2channels(
    realType u, /* formal input */
    pic2channelsContextType* context, /* context */
    realType* y /* formal output */)
{
    /* implied statements */
    context->u = u;
    /* user-defined statements */
    /* LH Controller */
    lh_controller(context->u, &(context->LH_Controller_l),
                   &(context->lh_controller_lcontrol));
    /* RH Controller */
    rh_controller(context->u, &(context->RH_Controller_l),
                   &(context->rh_controller_lcontrol));
    /* sum */
    context->sum_lo = context->lh_controller_lcontrol +
                     context->rh_controller_lcontrol;
    /* copy */
    context->y = context->sum_lo;
    /* implied statements */
    *y = context->y;
}
```
A program

```c
/* implementation for block/process pic2channels */
void pic2channels{
    realType u, /* formal input */
    pic2channelsContextType* context, /* context */
    realType* y /* formal output */
{
    /* implied statements */
    context->u = u;
    /* user-defined statements */
    /* LH Controller_1 */
    lh_controller(context->u, &context->LH_Controller_1),
    &context->lh_controller_lcontrol);
    /* RH_Controller_1 */
    rh_controller(context->u, &context->RH_Controller_1),
    &context->rh_controller_lcontrol);
    /* sum_l */
    context->sum_lo = context->lh_controller_lcontrol +
    context->rn_controller_lcontrol;
    /* copy */
    context->y = context->sum_lo;
    /* implied statements */
    *y = context->y;
}
```
Mathematical Modelling
A Tale of Two Cultures

Engineering
- Differential Equations
- Linear Algebra
- Probability Theory

Computer Science
- Mathematical Logic
- Discrete Structures
- Automata Theory
Formal Methods

Goal

Establish a mathematical proof of system correctness based on:

- Defining a system in a modelling/programming language (possible behaviors)
- Describing requirements in a specification language (desired behaviors)
- Formal analysis techniques to determine if system meets its requirements
Goal

Establish a mathematical proof of system correctness based on
Goal

Establish a mathematical proof of system correctness based on

- Defining a system in a modelling/programming language (possible behaviors)
Formal Methods

Goal

Establish a mathematical proof of system correctness based on
- Defining a system in a modelling/programming language (possible behaviors)
- Describing requirements in a specification language (desired behaviors)
Goal
Establish a mathematical proof of system correctness based on

- Defining a system in a modelling/programming language (possible behaviors)
- Describing requirements in a specification language (desired behaviors)
- Formal analysis techniques to determine if system meets its requirements
Holy Grail

Program → Verifier → Yes/No
Property → Verifier

Verifier

Turing: Impossible! Take property to be "does it halt?"
Rice: In fact, take property to be almost anything.
Holy Grail
Impossible Dream

Verifier

Program
Property

Yes/No

Turing: Impossible! Take property to be “does it halt?”
Holy Grail
Impossible Dream

Program → Verifier → Yes/No
Property

**Turing:** Impossible! Take property to be “does it halt?”

**Rice:** In fact, take property to be almost anything.
In a small village, the barber shaves all the men who do not shave themselves (and only them).
Russell’s Paradox

In a small village, the barber shaves all the men who do not shave themselves (and only them). Does the barber shave himself?

Bertrand Russell
Turing’s Proof

Suppose such a verifier exists for property "Halts?"

Program → Verifier → Yes
Property → Verifier → No
Turing’s Proof

Suppose such a verifier exists for property “Halts?”
Suppose such a verifier exists for property “Halts?”
Suppose such a verifier exists for property “Halts?”

Program: foo

Verifier:
- Yes → Loop forever
- No → Halt

Halt?

Does foo halt?
Suppose such a verifier exists for property “Halts?”
Suppose such a verifier exists for property “Halts?”
Does foo halt?
Abstraction

As first step, system model abstracted to yield a simpler model
Abstraction

As first step, system model abstracted to yield a simpler model
Abstraction

As first step, system model abstracted to yield a simpler model
Automated Verification

Abstract Program → Analysis → Yes/No

Property
Challenges in Formal Methods

- What are reasonable abstract models of systems?
Challenges in Formal Methods

- What are reasonable abstract models of systems?
- What are properties of interest?
Challenges in Formal Methods

- What are reasonable abstract models of systems?
- What are properties of interest?
- What are analysis algorithms/heuristics that scale to large systems?
Limitations of Formal Methods

- Correctness is only proved with respect to the requirements and requirements (like programs) could have errors!!
Limitations of Formal Methods

- Correctness is only proved with respect to the requirements and requirements (like programs) could have errors!!
- Correctness of software depends on the correctness of the physical hardware. Errors could appear due to a wide range of accidental causes like defective materials, defects in manufacturing, cosmic rays, changes in weather, physical calamities, etc.
Limitations of Formal Methods

- Correctness is only proved with respect to the requirements and requirements (like programs) could have errors!!
- Correctness of software depends on the correctness of the physical hardware. Errors could appear due to a wide range of accidental causes like defective materials, defects in manufacturing, cosmic rays, changes in weather, physical calamities, etc.
- Complete verification is labor intensive, and requires a lot of computing resources; hence often only the critical modules of the software are checked for correctness.
Course Contents

Exposure to different formal specification languages, analysis methods and tools

- Learn different specification languages: equational/rewriting logic, linear temporal logic, computation tree logic
Exposure to different formal specification languages, analysis methods and tools

- Learn different specification languages: equational/rewriting logic, linear temporal logic, computation tree logic
- Learn different analysis methods: model checking, theorem proving, runtime verification
Course Contents

Exposure to different formal specification languages, analysis methods and tools

- Learn different specification languages: equational/rewriting logic, linear temporal logic, computation tree logic
- Learn different analysis methods: model checking, theorem proving, runtime verification
- Experience using different tools that are either actively used in the industry or have inspired tools that are used in the industry.
Course Goals

Learn to develop specification languages and analysis techniques for new domains.