Lecture # 01

Introduction to Formal Methods

Instructor: Saima Zareen
Assistant Professor
Department of Software Engineering

saima.zareen@uettaxila.edu.pk
Introduction
Objectives

- In this course students will learn how to represent computing systems with both state-based and process based algebra models.
- They specify computing systems formally,
  - reason about specifications,
  - and verify their properties.
- They connect specifications to programmes through
  - Refinement and
  - decomposition.
- They use theorem proving and model checking tools.
Grading Policy

- Quizes: 10%
- Assignments: 5%
- Class Participation: 5%
- Mid Term: 20%
- Final Exam: 40%
Rules/pattern of class

- Be punctual
- Switch off your mobile phones
- Class participation is must
- Maintain discipline in class
- Dress well yourself
- Hair comb/ trim/ shave if required
- Revise your moral/ ethical values and follow them
Formal Method

The Encyclopedia of Software Engineering defines formal methods in the following manner:

- Formal methods used in developing computer systems are:
- mathematically based techniques for describing system properties.
- Such formal methods provide frameworks within which people can
  - specify,
  - develop, and
  - verify systems in a systematic, rather than ad hoc manner.
Formal Method definition

- A method is formal if it has a sound mathematical basis, typically given by a formal specification language.
- This basis provides a means of precisely defining notions like:
  - consistency,
  - completeness, and
  - more relevantly specification,
  - implementation and
  - correctness.
Formal Method definition contd..

- **Correctness:**
  - the property that an abstract model fulfills a set of well-defined requirements.

- **Consistency:**
  - to be consistent, facts stated in one place in a specification should not be contradicted in another place.

- Used to specify programs, what the system is suppose to do.
- Used for constructing programs.
- Used to verify the program.
Why Formal Methods are required?

- History of software
  - Softwares encountered notorious bugs that were the cause of financial lose and deaths of many people.
  - Famous bugs are
Famous bugs contd..

i. **Therac-25**
   - computerized radiation therapy machine called the Therac-25.
   - killed many people, controller could not stop radiation due to software bug, later on the problem was fixed, after killing many people life.
Famous bugs contd..

ii. AT & T long distance breakdown bug 1990

- ill placed break statement in the code,
- caused the 1/3rd of entire American network to go down for 9 hours.
- Loses to banks, for ATM and transactions
Why Formal Methods are required? Contd..

iii. Patriot Missile Failure Gulf war

- this missile hit the own American troops a software defect in 1991.
- Killing 28 people and many injured
iv. **Pentium bug**

- software error in microcode of Pentium microprocessor, which resulted in error of floating point calculations problems.
- Intel had to take back all the Pentiums, and it caused huge loss of dollars.
Importance of Software

- Software is present everywhere microwave oven, cars and software use is expanded.
- That means a small software error can cause your microwave oven to explode, where system failure can cause losses more than the system itself.
- A software problem can cause life loses.
- We have to be careful for the use of such systems, where loss of life is a bigger loss.
Importance of formal methods in Software

- This is what we are going to study in formal methods.
- Methods to ensure that software is
  - Correct
  - Reliable
Importance of formal methods in Software contd..

- These two attributes deal with the software quality.
- To achieve software quality, we apply different techniques.
  - Testing
  - Verification
  - Validation
Testing

- Black box testing
  - Test input versus output
  - Input
    - Two numbers
  - Output
    - average

- White box testing
  - Test the structure of program.
    - For loops testing, condition testing
Can we test every system?
"Testing can show the presence of errors, but not their absence."

- Edsger Dijkstra
Testing contd..

- In testing we develop test cases and define scenarios, it is not possible to have all scenarios.
- Program to show equality of two strings.
  - `isequal("cat", "dog")` expected false
  - `isequal("testing", "testing")` expected true
  - `isequal("house", "home")` expected false
  - `isequal("house","mouse")` expected false.
- No number of test cases assure this.
Testing Contd..

- Same is case with structural testing, white box testing.
- What is wrong with the following code?

```c
equal = strlen(string1) == strlen(string2);
if (equal)
    for (i = 0; i < strlen(string1); i++)
        equal = string1[i] == string2[i];
return equal;
```
Reliability

- Obviously there is no guarantee in life but every one wants to have reliable software.
- A report was presented to President Bill Clinton in 1996 by committee of IT, in which they stated
  - we know to write a software that works 95% of the time, but do not know how to write software that works for 100 % of the time, it is used.
If you fly in a plane 100 times plane crashes 5 times due to software error then, will you travel through plane?
How Formal Methods are applied?

• We develop models of system.
• With the help of models we will argue and prove correctness of models.
• Formal Method has 2 parts:
  • Logical Theory:
    • Means by which one reasons about specifications, properties and programs predicate calculus, (logic, propositions).
  • Structuring Theory:
    • Defines elements being reasoned about.
How Formal Methods are applied contd..?

- What is a model?
  - An item, a structure, a style,....
  - Lets play Dijkstra’s game to define a model.
Dijkstra’s game

stock of black balls

jar of black and white balls

Rules
Consider the following game to be played by a single person with an urn/jar and as many \( w \) white balls and \( b \) black balls as he needs.

To begin with, an arbitrary positive number of balls is put into the urn and as long as the urn contains two or more balls, the player repeats the following moves:

- he shakes the urn and, without looking, he takes two balls from the urn;
Dijkstra’s game contd..

- **Rule 1**
  - if those two balls have the same color
    - he throws one black ball into the urn,
- **Rule 2**
  - otherwise he returns one white ball into the urn
  - Because each move decreases the total number of balls into the urn by 1, the game is guaranteed to terminate after a finite number of moves and it is not difficult to see that the game ends with exactly 1 ball in the urn.
Dijkstra’s game contd..

- The question is:
  - What can we say about the color of the final ball when we are given the initial contents of the urn?"

- Difficult to answer
- Lets play the same game with different number of balls.
Dijkstra’s game contd..

- One ball game
  - The game will end without playing

- Two balls game
  - Three different combinations of balls
    - One Black, one white
    - One White, one white
    - One Black, one black
Dijkstra’s game contd..

2 ball game

[Diagram showing a three-ball game with moves indicated]
Dijkstra’s game contd..

- Mathematical models use functions.
- What is a function?
- Function:
  - Putting the balls in jar is a function
Conclusion from 2 and 3 balls game

- Depends on parity of white balls, even or odd parity.
- Even number of white balls, last ball is black color.
- Odd number of balls, last ball is of white.
- If we play with 100 balls, then can we argue or prove our hypothesis?
- What is the color of last ball, given \( w \) white balls and \( b \) black balls?
Mathematical model and its proof

\[ F(b, w) = \]

- 2 black out, 1 black in \( b-2+1, (b-1, w) \)
  We reduce the number of black balls by 1 and we maintain the number of white balls.

- 2 white out, 1 black in \( w-2, b+1 \)
  We reduce the number of white balls by 2 and increase the number of black balls by 1.

- 1 of each out, 1 white in \( b-1, w-1+1=(w) \)
  We reduce the number of black balls by 1 and maintain the number of white balls.
Mathematical model and its proof contd..

- Total number of balls removed in each move is 1.
- Parity(even/odd number) of white balls does not change.
- Yes we will say the parity of white balls determine the outcome of the game.
- Hence hypothesis is correct.
Formal method steps

- We will define state based model for our computer programs using formal methods.
  1. Define the specifications of the system (Formal specification).
  2. Define abstract model of specifications.
     i. Define the states of system (steps of a model)
     ii. Define invariant (condition)
     iii. Define set of operations for model to function.
        • System/model operation is associated with two conditions
           • Pre-condition
           • Post condition
  3. Model verification and Implementation
     • Make formal model and use tools to prove mechanically that formal execution model satisfies formal requirements.
Model types

- Two types of models are defined
  - Abstract model
    - abstraction is the process by which data and programs are defined with a representation similar to its pictorial meaning.
    - High level system design
  - Concrete model
    - Detailed model of the system achieved after refinement of abstract model.
Formalization Spectrum

Less Formal

- Natural language text description
- Mathematical proof
- Specification languages
- Specification + mathematical proof and/or automated tools
- Model checking, automated deduction/theorem proving

More formal
Examples of Formal Methods

- Formal methods can include graphical languages.
- For example,
  - Data Flow Diagrams (DFDs) are the most well-known graphical technique for specifying the function of a system.
  - DFDs can be considered a semi-formal method, and researchers have explored techniques for treating DFDs in a completely formal manner.
Examples of Formal Methods contd..

- Petri nets provide another well-known graphical technique, often used in distributed systems.
- Petri nets are a fully formal technique.
Examples of Formal Methods contd..

- Yet, another formal method is the Finite state machines, which are commonly presented in tabular form.

```
<table>
<thead>
<tr>
<th>Input</th>
<th>State A</th>
<th>State B</th>
<th>State C</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>...</td>
<td>State C</td>
<td>...</td>
</tr>
<tr>
<td>Z</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```
Use of Formal methods

- There is an increasing interest about formal methods and their applications.
- Formal methods have the potential to provide increased confidence in a system by satisfying the standards set by regulatory bodies.
Formal methods are not meant to

- to show correctness" of entire systems
- to replace testing entirely
- Formal methods work on models, on source code, or, at most, on
  - byte code level
  - Many non-formalizable properties
- to replace good design practices
- One can't formally verify messy code with unclear specs.
- We are not developing(programming) a complete system using Formal methods.
Formal methods are meant for

- But ....
  - Formal proof can replace (infinitely) many test cases
  - Formal methods improve the quality of specs (even without formal verification)
  - Formal methods guarantee specific properties of system model
A Fundamental Fact

- Formalization of system requirements is hard.
- But we can formalize critical features/requirements of system.
- Formal methods is a system design based on analysis of Specifications(requirements).
Seven Myths of Formal Methods

1. Formal methods guarantee perfection.
2. They work by proving correctness.
3. They are only good for critical systems.
4. They involve complex mathematics.
5. They increase costs.
6. They are incomprehensible to clients.
7. Nobody uses them for real projects.
Tools for formal Systems

- Pro-B tool
- Atelier B
- RODIN tool
Book

- The b method an introduction by Steve Schneider
Summary

- Formal methods are used to ensure correctness and reliability of software systems.
- Formal methods are based on mathematical models.
- Formal methods are difficult to apply but results are fruitful.
- Formal methods does not mean we are programming a part of the system.
- We are verifying the system correctness using formal methods.
Questions