Welcome to CS 477

Formal Methods in Software Development

Spring 2019

Madhusudan Parthasarathy (Madhu)
madhu@cs.uiuc.edu

What is this course about?

- Course on formal ways of
  - Proving programs correct
  - Developing reliable software
  - Analyzing programs for correctness
  - Finding bugs in software
- Formal ←→ Mathematical (provable/rigorous)
- Informal methods are also useful, but they are not covered in this course; see Soft. Engg courses
  - Eg. Random testing; Software management planning

Aims of this course

Theoretical:

- The fundamental mathematics behind proving a program correct by reducing it to logic

  Floyd-Hoare logic;
  contracts; pre/post conditions; inductive invariants
  verification conditions, strongest post, weakest pre

- Formal logic (FOL); to understand proof systems and automatic theorem proving, some decidable theories
- Contract-based programming for both sequential and concurrent programs; developing software using contracts.
- Static analysis using abstraction; abstract interpretations, overview of predicate abstraction.
- Finding test inputs formally using logic solvers

Aims of this course

Practical:

- Proving small programs correct using a modern program verification tool (Floyd-style)

- Use SMT solvers to solve logical constraints; understand how program verification can be done using these solvers.

- Build static analysis algorithms for some analysis problems using abstraction, and learn to use some abstract-interpretation tools

- Learn contract based programming using Dafny; use to generate unit tests and proofs

Aims of this course

The course is hence:

Formal-development of programs using contracts
+
Foundations of proving programs correct
+
Verification tools for proving programs using abstraction and automatic theorem proving.

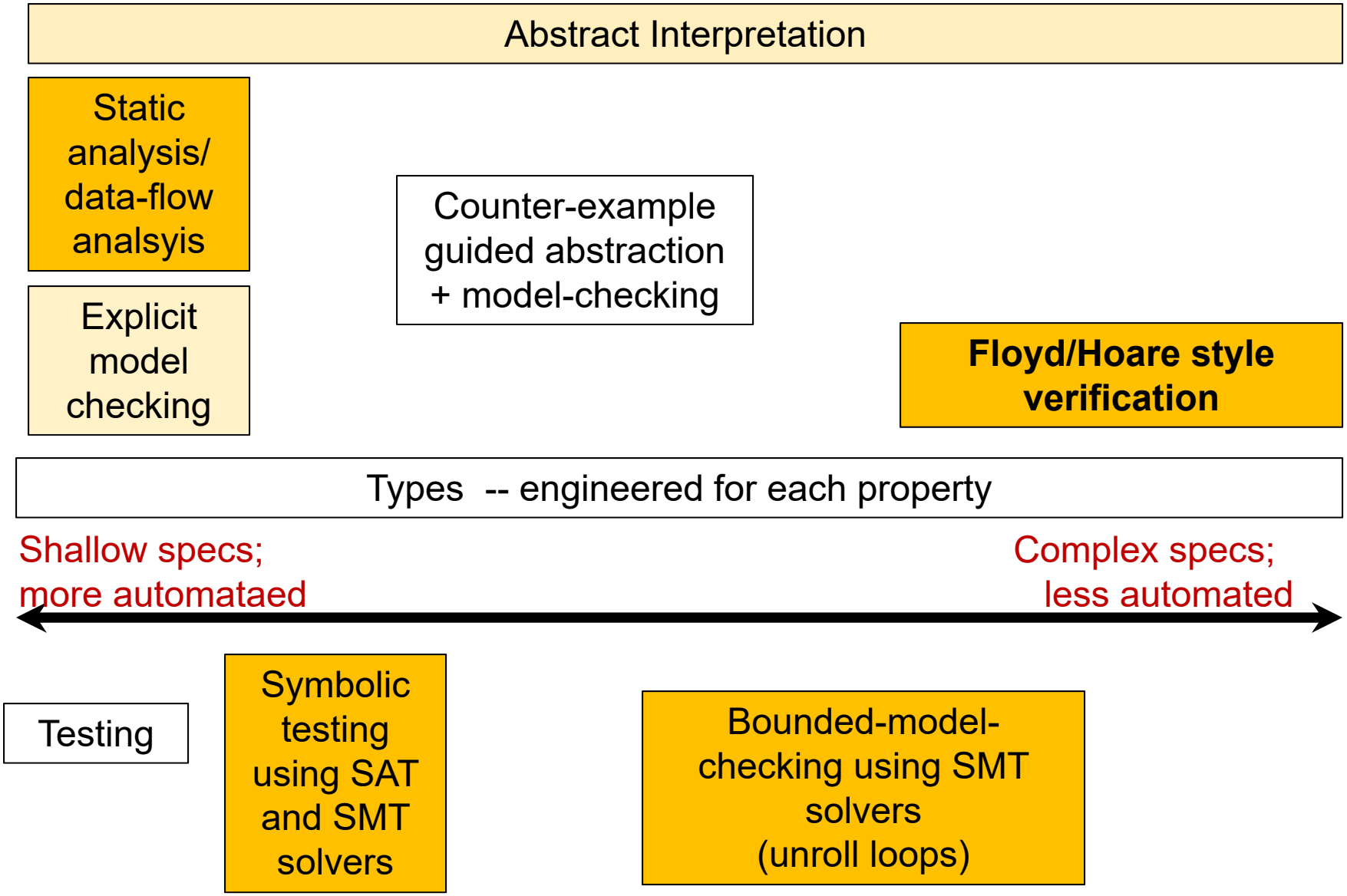
There are other formal software development methods that we will probably not cover:

--- Model-based software development

--- Z-notation; B method, etc. (?)

--- UML, etc.

Landscape of program verification



Contracts

- First proposed by Bertrand Meyer (Eiffel)
  called ‘Design by Contract’™
- Inspired by Hoare-style program verification

- Writing specifications *with* the code that formally specifies:
  - Preconditions of methods
  - Postconditions of methods
  - Class invariants

Contracts

- A compelling way to build develop programs
  - Specifications give *formal* documentation (not English comments); helps in communication between developers
  - Specifications can be used to do *unit testing*
  - Faster and more effective debugging by checking contracts at runtime; leads to finding bugs earlier
  - And…... can be used for program verification
    (with lots of manual help:
    loop invariants/thmprovers/patience!)

Contracts

- Impressive uses:
  - E.g. Buffer-overflow errors were eradicated from MS Windows kernel using contract-based programming where contracts described the ranges of variables to index arrays.
  - Huge effort; tremendous gain;
  - Satisfaction of programmers: bug localization

Techniques: Logic, Logic, Logic

- Logic!!
  - Program analysis of all kinds requires reasoning

    (E.g. x>y & x'=x+1 => x'>y;

    adding x larger to the end of a sorted list is still sorted if x is larger than all elements in the list)
  - Advent of SMT solvers:
    - Constraint solvers for particular theories
    - Engineering abstraction of logical reasoning that any program analysis tool can use
    - Completely automated
    - Boolean logic: SAT
    - Other theories: linear arithmetic, arrays, heaps, etc.

Techniques: Logic

- Use of logic
  - Formal specification logic (for contracts/invariants)
  - Separation logic
  - Hoare-style verification: Verification conditions
  - Abstraction: finding the abstract transitions
  - Symbolic execution: solving path constraints to generate input

SMT solvers enable all these technologies!

So you will learn logic:

Prop. Logic, FOL, FO theories like arithmetic, reals, arrays, etc., and decidable fragments

Successful tools

- Testing by Symbolic executions
  - PEX (http://research.microsoft.com/en-us/projects/pex/)
    Whitebox testing
    (internal to Microsoft; available in Visual Studio for .NET)
    PEX-for-fun website
  - SAGE
    Checks for security vulnerabilities in Windows code
    stems from DART/CUTE : ``concolic testing''
  - VeriSol (NEC) for Verilog
  - CBMC for C

Some successful tools

- Explicit model-checking (we probably won't cover this)
  - Verisoft (http://cm.bell-labs.com/who/god/verisoft/)
    - Fully automatic tool;systematic state-space exploration; 1996; Bell-labs
  - SPIN (http://spinroot.com/spin/whatispin.html)
    - Checks software models
  - CHESS
    - Concurrent programs with bounded preemptions
- Partially symbolic approaches
  - Java PathFinder (NASA): (http://javapathfinder.sourceforge.net/)

Some successful tools

- Abstraction based tools
  - ASTREE – abstract-interpretation (http://www.astree.ens.fr/)
    For flight control software
  - SLAM /SDV – Microsoft (http://www.microsoft.com/whdc/devtools/tools/sdv.mspx)
    For device drivers
  - FSoft – NEC
    (http://www.nec-labs.com/research/system/systems_SAV-website/index.php)
  - TVLA (http://www.math.tau.ac.il/~tvla/)
    Abstractions for heaps using shape analysis
  - Yogi – MSR
    http://research.microsoft.com/en-us/projects/yogi/
    Combines static verification with testing

Some successful tools

- Deductive Floyd-Hoare style verification
  - ESC-Java
  - DAFNY (https://github.com/Microsoft/dafny)
    and Boogie (MSR) (http://boogie.codeplex.com/)
    and VCC (http://research.microsoft.com/en-us/projects/vcc/)
    (use Z3 SMT solver)
  - STORM (http://stormchecker.codeplex.com/)
    - Unsound analysis for finding bugs (uses Z3)
  - FUSION (from NEC)

Some successful tools

Contract-programming languages

- EIFFEL
- CodeContracts from MS for .NET (see also Spec#)
- JML (Java Modeling languages)

SMT (logic) solvers

- A plethora of satisfiability-modulo-theory solvers
  - Simplify, Yices, Z3, CVC, UCLID
  - SAT solvers: zChaff, MiniSAT,…

  - Core technology in several engines
  - Eg. Z3 is used in SDV,PREfix, PEX, SAGE, Yogi, Spec#, VCC, HAVOC, SpecExplorer, FORMULA, F7, M3, VS3, …

Course topics

- Floyd-style verification (motivatating need for logic)
- Prop. Logic, Predicate logic; Theories
- Soundness/completeness/Godel's theorem. Proof systems
- Hoare logic and axiomatic semantics
- Basic paths; weakest pre; strongest post; partial correctness
- Decidable theories; SAT and SMT solvers
- Design by contract; code contracts
- Symbolic test input generation; bounded model-checking
- Logics for reasoning with heap
- Abstract Interpretation: Dataflow analysis, static analysis for certain abstract domains.
- Invariant synthesis techniques

Logistics

- Course website: + Piazza newsgroup
- HWs (about once in two weeks on avg; more in the beginning; less near the end; 5-6 sets)
- Grades will be curved (curve *may* be separate for undergrads and grads)

HW: 30
Midterm: 30
Final exam: 40
Project: 50

- 3 credits: HW + Final (out of 100)
- 4 credits: HW + Final + Project (out of 150)

Homework sets

- Homework can be in groups of two
  - You can work on problems in a group of two
  - ~~But you must submit homework write-ups individually, written by yourself~~.
  - You may submit homework with the person you work with as well.
  - Indicate clearly who you worked with.

Project

4 credits requires a project.

Involves either

- Reading up a set of papers, and writing a report, or
- Programming a particular technique or developing software using contracts, and submitting a write-up

Groups of 3 or less; (ask for exceptions)

More details later…

A sample project

Develop a memory management routing that hands out chunks of memory to processes ensuring no overlap.

Clear simple specification.

Implementation using linked lists.

Specification using separation logic or FO+recursion.

Prove correct using VCC/VCDryad/DAFNY.

Course resources

- No textbook; online handouts
  (accessible from UIUC net domain)
- Software:
  Many; need access to a MS Windows machine

– See course website for info:
  http://www.cs.uiuc.edu/classs/cs477

– Enroll in Piazza

– Teaching Assistant: John Lee

Questions?