# E0227: Program Analysis and Verification

August - December 2021

3:30pm to 5 pm, M, W

K. V. Raghavan & Deepak D'souza

# Program Verification

"The algorithmic discovery of properties of a program by inspection of it's source text"

— Manna & Pnueli

- Also known as static analysis
- As opposed to discovering properties by testing

# Applications of program analysis

- To identify certain classes of errors in programs:
  + null pointer dereferences, array out of bounds access, reading a file after closing it, using an unitialized variable

  + synchronization violations in concurrent code

  + Violations of data structure properties, e.g., acyclicity of a linked list

# Applications — continued

- In compilers, to generate and optimize code
  - To generate optimized single-thread code
  - To generate parallel code, e.g., for multicore processors
  - To estimate performance, e.g., worst-case execution time analysis
- In refactoring tools, e.g., Eclipse
- In "white box" testing tools
  - they generate test cases to exercise various paths in the program, to see how it behaves

# An example: analyzing interference for parallelization

```
main() {
  LinkedList * l1 = ... create
                     new list..;
  LinkedList * l2 = ... create
                     new list..;
  foo(l1, l2);   // 1
  foo(l1, l1);   // 2
}
```

```
foo(lx, ly){
   insert(lx, n);
                     } Can run in parallel?
   lookup(ly, m);
}
```

Q: Can insert and lookup be made to run in parallel?
I.e., are they uninterfering?

# An example: analyzing interference for parallelization

```
main() {
  LinkedList * l1 = ... create
                        new list..;
  LinkedList * l2 = ... create
                        new list..;
  foo(l1, l2);   // 1
  foo(l1, l1);   // 2
}
```

```
foo(lx, ly){
    insert(lx, n);    } Can
                      } run in
    lookup(ly, m);    } parallel?
}
```

Q: Can insert and lookup be made to run in parallel?
I.e., are they uninterfering?

A. YES, when called from 1, NO when called from 2.

# Another example: program slicing

```
t = a
if (x < 50)
    t = b
x = x - 1
if (x > 120)
    z = t
print (z)
```

Can any statement be removed from the program without affecting the final output ?

# Another example: program slicing

```
t = a
if (x < 50)
    ~~t = b~~
x = x - 1
if (x > 120)
    z = t
print (z)
```

Can any statement be removed from the program without affecting the final output?

# PAV Course contents

- Data flow analysis / abstract interpretation
- Analysis of multi-procedure programs
- Pointer analysis
- Program slicing
- Type systems
- Program analysis using Floyd-Hoare logic

# E0 272

# Formal Methods in Software Engineering

January – May 2022

Deepak D'souza & K.V. Raghavan

# Motivation

- There are many cutting-edge tools available for the various phases in the s/w Development Life Cycle
- Knowledge of these tools gives
  - + Exposure to practical uses of various analysis techniques
  - + Prepares one for career in research as well as industry
- Logistics of course
  - + Assignments involving hands-on usage of the tools
  - + No prerequisites!

# Course Contents

- Capturing and analyzing requirements (Alloy)
- Software design
  - Designing state transition systems with Spin
  - Designing data structures with Rodin
- Code verification and validation with VCC
- Automated testing of programs using JPF and AFL

# Principles of Distributed Software

E0 209,January-May 2022
Komondoor V. Raghavan
IISc

# Distributed computing and cloud computing

What is distributed computing?

- A single application runs across multiple nodes/computers, which may be geographically dispersed, and which are connected to each other via networking

Why distributed computing?

- Scalability: Application may be able to scale up to using more nodes or down to using fewer nodes based on real-time load
- Availability: If a node goes down, application  still runs (with reduced scale or functionality)
- Latency: Each user could be served by a node that is closer to them
- Cost: A set of smaller/commodity nodes may be cheaper than a single powerful one

*Contrast with high-performance computing*

What is cloud computing? Application runs on remotely based computers

# Example domains where distributed computing is used

- Online e-commerce, travel booking, banking, etc.
- Mobile-based taxi hailing
- Social-media apps
- Multi player online games
- Web-based collaboration software (e.g., web-based document or spreadsheet editing)
- Large-scale data analytics, machine learning

Note, in some of the examples above the nodes are servers (i.e., very little client-side computing), while in other examples even clients play a role in computing and thus serve as nodes.

# Focus of this course

Core content: Concepts, technologies, and frameworks, for developing and deploying distributed software.

We will focus primarily on techniques useful for database-oriented enterprise applications (e.g., e-commerce, travel booking, social media, banking, etc.)

Topics we will cover:

- Containers and virtualization
- Services, microservices, architectural patterns for developing microservices
- Microservice development and deployment using SpringBoot, Java, and Docker
- Cluster management using Kubernetes
- Event-based, actor-based programming using Akka
- Eventual consistency of data in the presence of distributed updates
- Programming for data analytics (Spark)

# My research interests

Automated tools for all aspects of software development life cycle:

- Automated techniques for finding bugs in programs & verifying correctness of programs
- Tool support to modify programs, for improving code structure, or to add new features.
- Automated correctness testing of reactive programs that use adaptive learning (e.g. autonomous vehicle software)