# More Course Overview:
# Models, Tests, Bugs, and Symbols

1

## But first

- Homework 1 and 2
- Yi and Jingbo’s lectures
- What’s coming up
- Idea proposal assignment
- Overview of the final topics in this class

2

## Homework 1

- Posted on class website
- Due Tuesday March 2, 9 AM on gradescope
  - Everyone in the class should have gotten a notification from gradescope about being added to the class. If you didn’t get it, let me know!

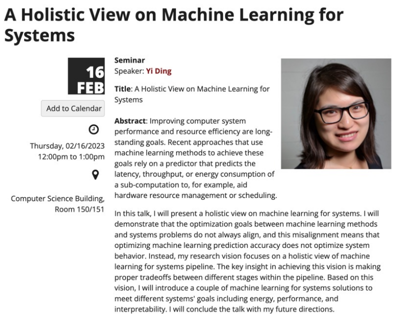
3

## Homework 2

- A little different.
- In-person, 2.5-hour session.
- Sign up for a slot soon.
- Slots will take place between March 6 and March 28
- More info soon!

4

## Yi Ding



**A Holistic View on Machine Learning for Systems**

16 FEB

Add to Calendar

Thursday, 02/16/2023
12:00pm to 1:00pm

Computer Science Building,
Room 150/151

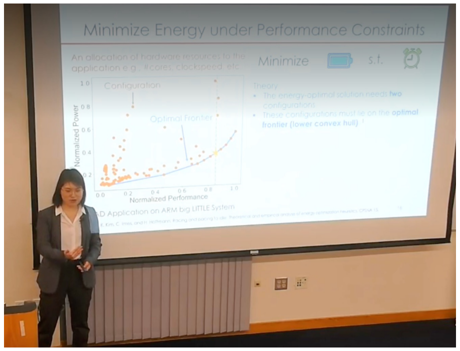
**Seminar**
Speaker: Yi Ding

**Title**: A Holistic View on Machine Learning for Systems

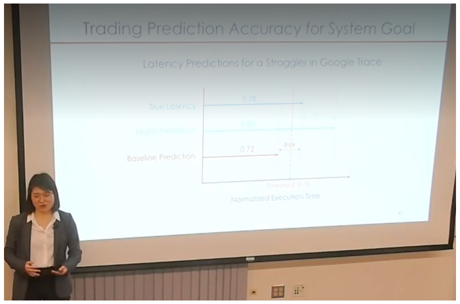
**Abstract**: Improving computer system performance and resource efficiency are long-standing goals. Recent approaches that use machine learning methods to achieve these goals rely on a predictor that predicts the latency, throughput, or energy consumption of a sub-computation to, for example, aid hardware resource management or scheduling.

In this talk, I will present a holistic view on machine learning for systems. I will demonstrate that the optimization goals between machine learning methods and systems problems do not always align, and this misalignment means that optimizing machine learning prediction accuracy does not optimize system behavior. Instead, my research vision focuses on a holistic view of machine learning for systems pipeline. The key insight in achieving this vision is making proper tradeoffs between different stages within the pipeline. Based on this vision, I will introduce a couple of machine learning for systems solutions to meet different systems' goals including energy, performance, and interpretability. I will conclude the talk with my future directions.

5


Minimize Energy under Performance Constraints
An allocation of hardware resources to the application e.g., #cores, clockspeed, etc.
Minimize
s.t.
Configuration
Optimal Frontier
Normalized Power
Normalized Performance
0.0
0.2
0.4
0.6
0.8
1.0
Theory
The energy-optimal solution needs two configurations
These configurations must lie on the optimal frontier (lower convex hull)
AD Application on ARM big LITTLE System

6

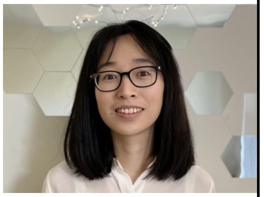

Trading Prediction Accuracy for System Goal
Latency Predictions for a Straggler in Google Trace
True Latency
0.78
Baseline Prediction
0.72
0.06
Threshold: 0.75
Normalized Execution Time

7

## Tuesday, 2/21, noon in CS151

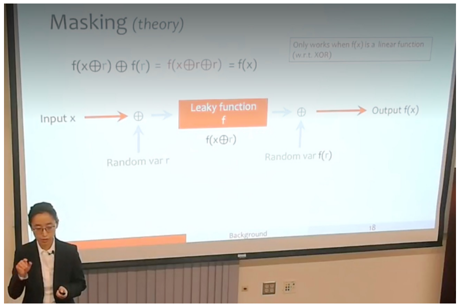
Trustworthy Software Enabled by
Program Analysis and Synthesis

**Abstract:**
Security, robustness, and fairness are all important non-functional properties of critical systems, such as software applications in autonomous driving, healthcare, and finance. Unlike functional correctness, which has been the subject of extensive research, techniques that can formally guarantee these non-functional properties are still severely lacking. In this talk, I will present two techniques based on data-driven synthesis and program analysis for improving security and fairness. The first one is a new method for detecting side-channel leaks, which are a class of emerging security threats that exploit the statistical dependencies between secret data and the physical characteristics of the computing devices. Instead of hand-crafting the analysis algorithm and then proving its soundness, our technique synthesizes the algorithm automatically while guaranteeing that it is sound by construction. The second technique is for synthesizing a machine learning model with a fairness guarantee, which is important for applications that are increasingly used to make socially sensitive decisions. Finally, I will talk about my research in the future, which will focus on providing formal guarantees of security, robustness, and fairness to other emerging applications.



Jingbo Wang

8



9

## Coming Up: 3 guest lectures and a day off

- Tuesday, February 28: guest lecture
- Thursday, March 2: Use class slot to discuss project ideas and form teams. No lecture.
- Tuesday, March 7: guest lecture
- Thursday, March 9: guest lecture

- … no more guest lectures after!

10

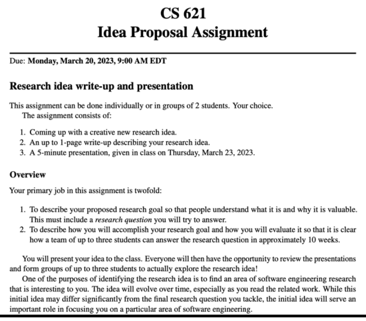
## Tuesday, 2/28, noon CS 151

From Barriers to Bridges:
Designing Processes and Tools for
Inclusive Open-Source Communities.

Diversity and Inclusion in Open-Source Software (OSS) has a significant impact on the OSS ecosystem and society. The low state of diversity and inclusion in OSS (e.g., women participation ranging from 1.5% to 11%) has unfortunate effects on OSS projects, individual contributors, and society. In this talk, I will present my findings from three research projects: (1) a conceptual model of the challenges faced by OSS contributors in a mature OSS organization, (2) a systematic inclusivity debugging process "Why/ Where/ Fix" based on this conceptual model to help project leaders find and fix inclusivity bugs and (3) the automation of a vertical slice of the inclusivity debugging process. Our results showed that the "Why/ Where/ Fix" inclusivity debugging process reduced the number of inclusivity bugs by 90%, produced positive effects across diverse cognitive styles, and made the project more equitable. These results provide encouraging evidence that the Why/Where/Fix process may provide an effective way to increase the equity and inclusion of information-rich environments like OSS projects.

11

## Idea Proposal Assignment



**CS 621**
**Idea Proposal Assignment**

Due: **Monday, March 20, 2023, 9:00 AM EDT**

**Research idea write-up and presentation**

This assignment can be done individually or in groups of 2 students. Your choice.

The assignment consists of:

1. Coming up with a creative new research idea.
2. An up to 1-page write-up describing your research idea.
3. A 5-minute presentation, given in class on Thursday, March 23, 2023.

**Overview**

Your primary job in this assignment is twofold:

1. To describe your proposed research goal so that people understand what it is and why it is valuable. This must include a *research question* you will try to answer.
2. To describe how you will accomplish your research goal and how you will evaluate it so that it is clear how a team of up to three students can answer the research question in approximately 10 weeks.

You will present your idea to the class. Everyone will then have the opportunity to review the presentations and form groups of up to three students to actually explore the research idea!

One of the purposes of identifying the research idea is to find an area of software engineering research that is interesting to you. The idea will evolve over time, especially as you read the related work. While this initial idea may differ significantly from the final research question you tackle, the initial idea will serve an important role in focusing you on a particular area of software engineering.

12

## On to semester overview

13

## Static analysis

- Using the source code to improve a program
- Manual code reviews and inspections
- Automatic inference of properties

Improve the software quality

14

## Dynamic analysis

- Using the program executions to improve the program
- Manual with debuggers, etc.
- Automatic inference over logged behavior
- Does not need source code or even binaries

Improve the software quality

15

## Areas we will cover in this course

- Static analysis
- Dynamic analysis
- Model checking
- Mutation testing
- Bug localization
- Symbolic execution

areas for your projects

16

## As we go over each topic…

- Think whether this sounds interesting
- Think about what kind of a tool you could make that uses this

- You are all programmers:
  think about things you've done while programming that were hard, and how these kinds of analysis might make it easier

17

## Model checking

- I actually meant:
  - Model checking
  - Model inference
  - Model simulation

18

## Model inference

problem:

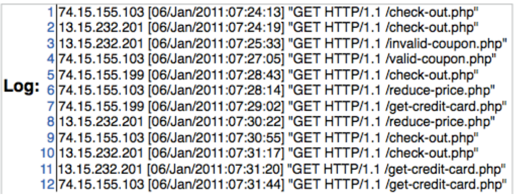
I have a system (or a log of executions).

I want a small, descriptive model of what the system does.

Model can be used to **understand** the system, **debug**, detect **anomalies**, **document**.

19

## Logs are hard to read



```
Log:  1 74.15.155.103 [06/Jan/2011:07:24:13] "GET HTTP/1.1 /check-out.php"
      2 13.15.232.201 [06/Jan/2011:07:24:19] "GET HTTP/1.1 /check-out.php"
      3 13.15.232.201 [06/Jan/2011:07:25:33] "GET HTTP/1.1 /invalid-coupon.php"
      4 74.15.155.103 [06/Jan/2011:07:27:05] "GET HTTP/1.1 /valid-coupon.php"
      5 74.15.155.199 [06/Jan/2011:07:28:43] "GET HTTP/1.1 /check-out.php"
      6 74.15.155.103 [06/Jan/2011:07:28:14] "GET HTTP/1.1 /reduce-price.php"
      7 74.15.155.199 [06/Jan/2011:07:29:02] "GET HTTP/1.1 /get-credit-card.php"
      8 13.15.232.201 [06/Jan/2011:07:30:22] "GET HTTP/1.1 /reduce-price.php"
      9 74.15.155.103 [06/Jan/2011:07:30:55] "GET HTTP/1.1 /check-out.php"
     10 13.15.232.201 [06/Jan/2011:07:31:17] "GET HTTP/1.1 /check-out.php"
     11 13.15.232.201 [06/Jan/2011:07:31:20] "GET HTTP/1.1 /get-credit-card.php"
     12 74.15.155.103 [06/Jan/2011:07:31:44] "GET HTTP/1.1 /get-credit-card.php"
```

20

## Model inference

- First, parse out the executions

check-out → valid-coupon → check-out → reduce-price → get-credit-card

check-out → invalid-coupon → check-out → reduce-price → get-credit-card

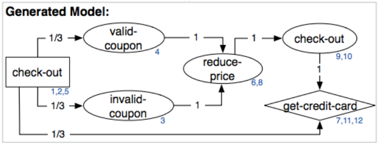
check-out → get-credit-card

- …hard to understand

21

## Infer the model

- Magic!



22

## So what's the magic?

- Lots of ways to do it:
  - Try merging the executions into a small model
  - Mine properties then build a model from the properties alone
  - Use static or dynamic analysis to determine what events can legally take place after others

23

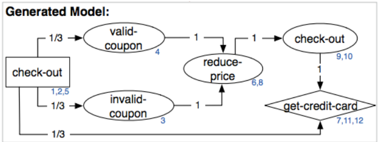
## K-Tails

- let's use k=1 as an example
- merge two states if their name is the same
- (k=2 means merge two states if their name, and all the states to which they have transitions are "the same")
- and so on for larger k

24

## Model checking

- Given a property and a model, check if the model satisfies that property

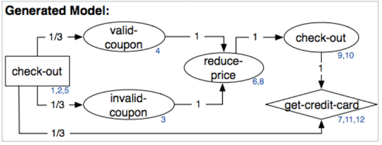


- Reduce-price always followed by get-credit-card?

25

## Model simulation

- Given a model, you can generate new executions that have not been observed before!



26

## Mutation testing

- Evaluate the tests
  - not the program!
  - not a type of testing!
  - does not improve a program directly; improves tests!

27

## Mutation

- Take a program
- Create a mutant with one or a few small changes:
  - change a + to a –
  - add/subtract 1 somewhere
  - increment/decrement a loop counter
  - delete a line
  - insert a line from one place in another
- Repeat to create many mutants

28

## Why create mutants?

- Suppose you have a program and a test suite
- All the tests pass
- What does that mean about your program?

1. Program is correct
2. Tests only test parts of the program that are correct and the rest, who knows
3. Tests and program may be written by the same person, using the same *implicit* assumptions

29

## Let's write some tests

```
// returns the factorial of the input n
int factorial (int n) {
  if (n <= 0)
    return 1;
  if (n == 1)
    return 1;
  else
    return n * factorial(n-1);
}
```

30

## OK, so how do we test the tests?

- Run the tests on the main program
- Run the tests on the mutants
  - what if the tests pass?

31

## Mutation testing evaluates the tests

- If a test "kills a mutant" then that's a good test
- If some mutants aren't killed, the test suite is lacking
- Solution: write more tests!
- Is it OK to write more tests until all mutants are killed and then stop?

32

## Consider this mutant

```
// returns the factorial of the input n
int factorial (int n) {
 if (n <= 0)
   return 1;
 if (n == 1)
   return 1;
 else
   return n * factorial(n+1);
}
```

33

## Consider this mutant

```
// returns the factorial of the input n
int factorial (int n) {
 if (n <= 2)
   return 1;
 if (n == 1)
   return 1;
 else
   return n * factorial(n-1);
}
```

34

## Consider this mutant

```
// returns the factorial of the input n
int factorial (int n) {
 if (n == 0)
   return 1;
 if (n == 1)
   return 1;
 else
   return n * factorial(n-1);
}
```

35

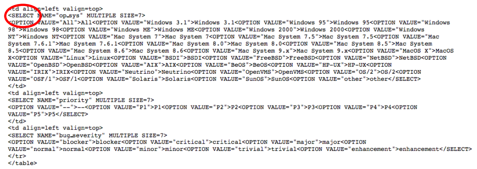
## Bug localization

- Narrowing down the most likely place to contain a bug

36

## Failure-inducing input

- This HTML input makes Mozilla crash (segmentation fault).
- Which portion is the failure-inducing one?



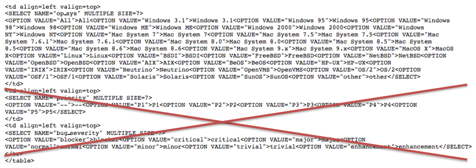
```
<td align=left valign=top>
<SELECT NAME="op_sys" MULTIPLE SIZE=7>
<OPTION VALUE="All">All<OPTION VALUE="Windows 3.1">Windows 3.1<OPTION VALUE="Windows 95">Windows 95<OPTION VALUE="Windows
98">Windows 98<OPTION VALUE="Windows ME">Windows ME<OPTION VALUE="Windows 2000">Windows 2000<OPTION VALUE="Windows
NT">Windows NT<OPTION VALUE="Mac System 7">Mac System 7<OPTION VALUE="Mac System 7.5">Mac System 7.5<OPTION VALUE="Mac
System 7.6.1">Mac System 7.6.1<OPTION VALUE="Mac System 8.0">Mac System 8.0<OPTION VALUE="Mac System 8.5">Mac System
8.5<OPTION VALUE="Mac System 8.6">Mac System 8.6<OPTION VALUE="Mac System 9.x">Mac System 9.x<OPTION VALUE="MacOS X">MacOS
X<OPTION VALUE="Linux">Linux<OPTION VALUE="BSDI">BSDI<OPTION VALUE="FreeBSD">FreeBSD<OPTION VALUE="NetBSD">NetBSD<OPTION
VALUE="OpenBSD">OpenBSD<OPTION VALUE="AIX">AIX<OPTION VALUE="BeOS">BeOS<OPTION VALUE="HP-UX">HP-UX<OPTION
VALUE="IRIX">IRIX<OPTION VALUE="Neutrino">Neutrino<OPTION VALUE="OpenVMS">OpenVMS<OPTION VALUE="OS/2">OS/2<OPTION
VALUE="OSF/1">OSF/1<OPTION VALUE="Solaris">Solaris<OPTION VALUE="SunOS">SunOS<OPTION VALUE="other">other</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<OPTION VALUE="--">--<OPTION VALUE="P1">P1<OPTION VALUE="P2">P2<OPTION VALUE="P3">P3<OPTION VALUE="P4">P4<OPTION
VALUE="P5">P5</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="bug_severity" MULTIPLE SIZE=7>
<OPTION VALUE="blocker">blocker<OPTION VALUE="critical">critical<OPTION VALUE="major">major<OPTION
VALUE="normal">normal<OPTION VALUE="minor">minor<OPTION VALUE="trivial">trivial<OPTION VALUE="enhancement">enhancement</SELECT>
</tr>
</table>
```

Thank you to Curino and Giusti for contributing to these slides

37

## Delta Debugging: Try half the input

- Will the program still crash?



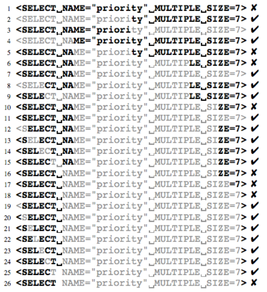
```
<td align=left valign=top>
<SELECT NAME="op_sys" MULTIPLE SIZE=7>
<OPTION VALUE="All">All<OPTION VALUE="Windows 3.1">Windows 3.1<OPTION VALUE="Windows 95">Windows 95<OPTION VALUE="Windows
98">Windows 98<OPTION VALUE="Windows ME">Windows ME<OPTION VALUE="Windows 2000">Windows 2000<OPTION VALUE="Windows
NT">Windows NT<OPTION VALUE="Mac System 7">Mac System 7<OPTION VALUE="Mac System 7.5">Mac System 7.5<OPTION VALUE="Mac
System 7.6.1">Mac System 7.6.1<OPTION VALUE="Mac System 8.0">Mac System 8.0<OPTION VALUE="Mac System 8.5">Mac System
8.5<OPTION VALUE="Mac System 8.6">Mac System 8.6<OPTION VALUE="Mac System 9.x">Mac System 9.x<OPTION VALUE="MacOS X">MacOS
X<OPTION VALUE="Linux">Linux<OPTION VALUE="BSDI">BSDI<OPTION VALUE="FreeBSD">FreeBSD<OPTION VALUE="NetBSD">NetBSD<OPTION
VALUE="OpenBSD">OpenBSD<OPTION VALUE="AIX">AIX<OPTION VALUE="BeOS">BeOS<OPTION VALUE="HP-UX">HP-UX<OPTION
VALUE="IRIX">IRIX<OPTION VALUE="Neutrino">Neutrino<OPTION VALUE="OpenVMS">OpenVMS<OPTION VALUE="OS/2">OS/2<OPTION
VALUE="OSF/1">OSF/1<OPTION VALUE="Solaris">Solaris<OPTION VALUE="SunOS">SunOS<OPTION VALUE="other">other</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="priority" MULTIPLE SIZE=7>
<OPTION VALUE="--">--<OPTION VALUE="P1">P1<OPTION VALUE="P2">P2<OPTION VALUE="P3">P3<OPTION VALUE="P4">P4<OPTION
VALUE="P5">P5</SELECT>
</td>
<td align=left valign=top>
<SELECT NAME="bug_severity" MULTIPLE SIZE=7>
<OPTION VALUE="blocker">blocker<OPTION VALUE="critical">critical<OPTION VALUE="major">major<OPTION
VALUE="normal">normal<OPTION VALUE="minor">minor<OPTION VALUE="trivial">trivial<OPTION VALUE="enhancement">enhancement</SELECT>
</tr>
</table>
```

Thank you to Curino and Giusti for contributing to these slides

38

## Minimizing via binary search

- 57 test to simplify the 896 line HTML input to the "<SELECT>" tag that causes the crash
- Each character is relevant (as shown from line 20 to 26)
- Only removes deltas from the failing test



```
1  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
2  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
3  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
4  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
5  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
6  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
7  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
8  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
9  <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
10 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
11 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
12 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
13 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
14 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
15 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
16 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
17 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
18 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✗
19 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
20 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
21 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
22 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
23 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
24 <SELECT␣NAME="priority"␣MULTIPLE␣SIZE=7> ✓
25 <SELECT NAME="priority"␣MULTIPLE␣SIZE=7> ✓
26 <SELECT NAME="priority"␣MULTIPLE␣SIZE=7> ✗
```

39

## Impact analysis

- Run the code on passing test cases
- Run the code on failing test cases
- Keep track of which lines execute

- Lines that executes only on passing test cases are OK. So are lines that execute on both.
- Lines that only execute on failing test cases are suspicious.

40

## What else can you do to localize a bug?

Regressions: suppose a test used to pass and now fails.

- consider the latest changes
- do delta debugging on the changes

41

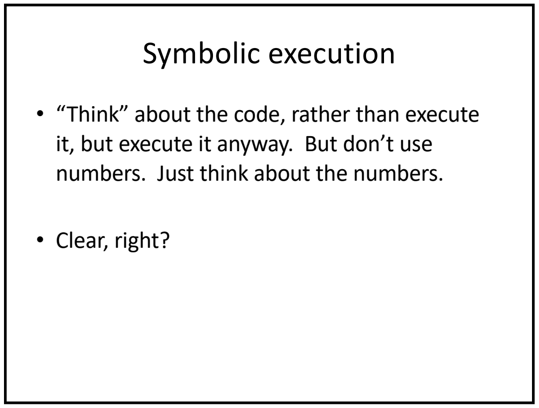
## Can we automatically fix bugs?

Take a program that passes most test cases and fails one or two, and tweak it

- write (tweak) a very similar program (with minimal change) that passes all the tests

[see Weimer et al., Automatically Finding Patches Using Genetic Programming, ICSE 2009]

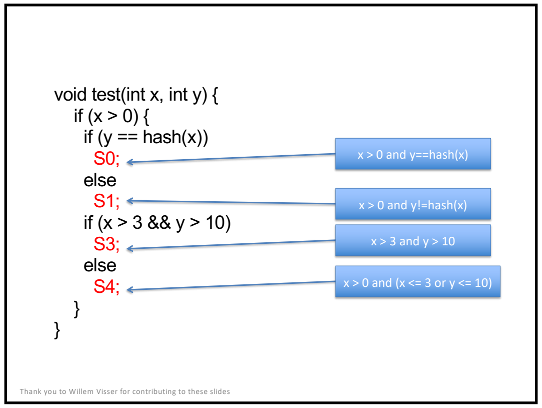
42



## Symbolic execution

- “Think” about the code, rather than execute it, but execute it anyway. But don’t use numbers. Just think about the numbers.
- Clear, right?

43



```
void test(int x, int y) {
   if (x > 0) {
     if (y == hash(x))
       S0;
     else
       S1;
     if (x > 3 && y > 10)
       S3;
     else
       S4;
   }
}
```

x > 0 and y==hash(x)

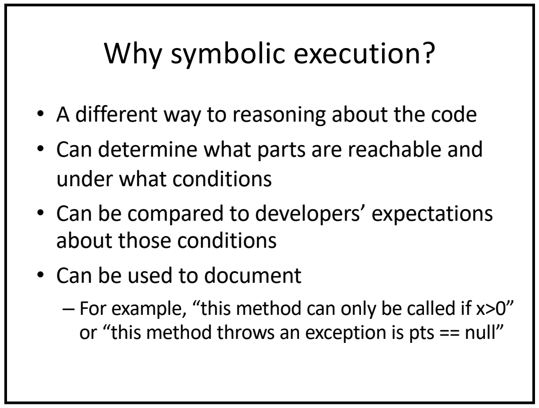
x > 0 and y!=hash(x)

x > 3 and y > 10

x > 0 and (x <= 3 or y <= 10)

Thank you to Willem Visser for contributing to these slides

44



## Why symbolic execution?

- A different way to reasoning about the code
- Can determine what parts are reachable and under what conditions
- Can be compared to developers’ expectations about those conditions
- Can be used to document
  - For example, “this method can only be called if x>0” or “this method throws an exception is pts == null”

45