Overview

CS560: Reasoning About Programs

Roopsha Samanta

PURDUE
UNIVERSITY

Roadmap

Today

- Motivation
- Overview
- Logistics

Roadmap

Today

- Motivation
- Overview
- Logistics

What is this course about?

Logical foundations & algorithmic techniques to ensure program correctness

Specification	Logics to express expected program behavior
Verification	Methods to automatically check if a program satisfies a specification
Repair	Methods to automatically fix an incorrect program
Synthesis	Methods to automatically generate a correct program

Why should you care?

Programmers make mistakes

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

127 little bugs in the code...

Why should you care?

Software bugs can be expensive, or fatal, or both!

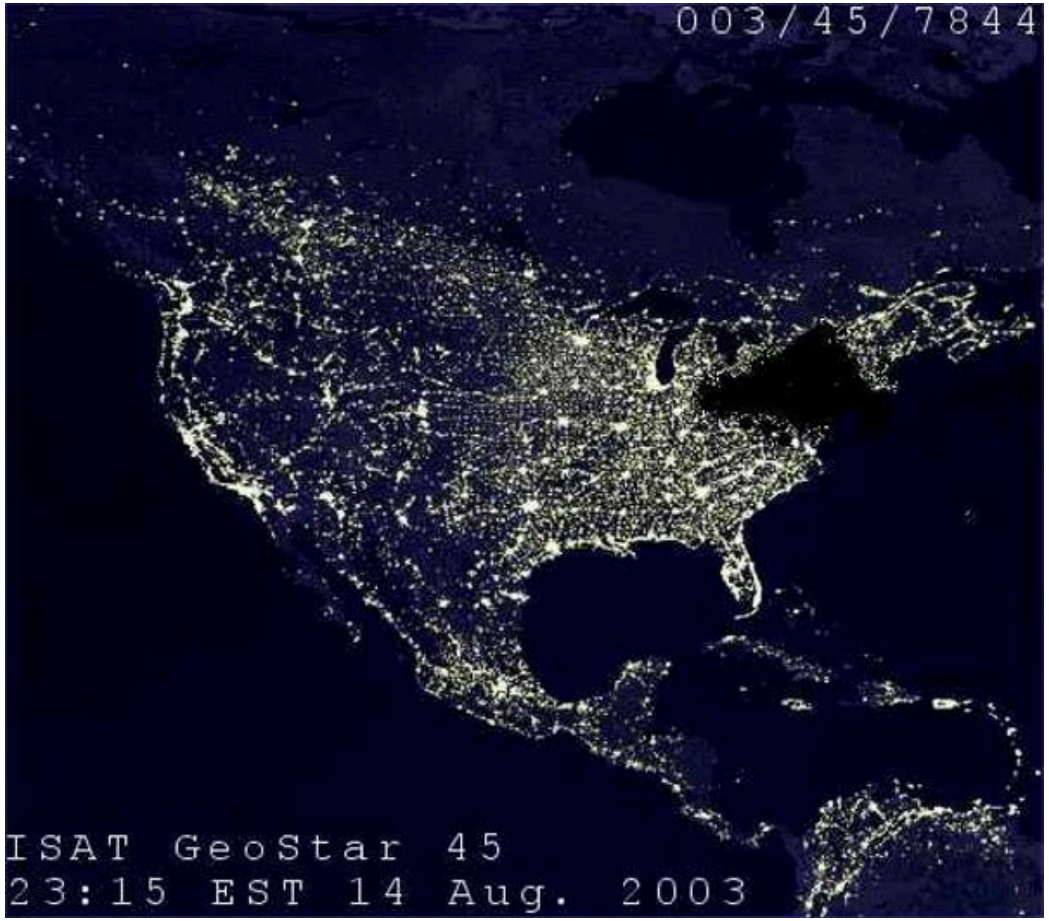
Therac-25 radiotherapy machine overdose, mid-1980s

6 deaths. Overflow error, race conditions

Ariane 5 explosion

$7 billion loss. Overflow error

North American power blackout



11 deaths, $6 billion loss. Race condition

Turing Awards

Dijkstra

Floyd

Hoare

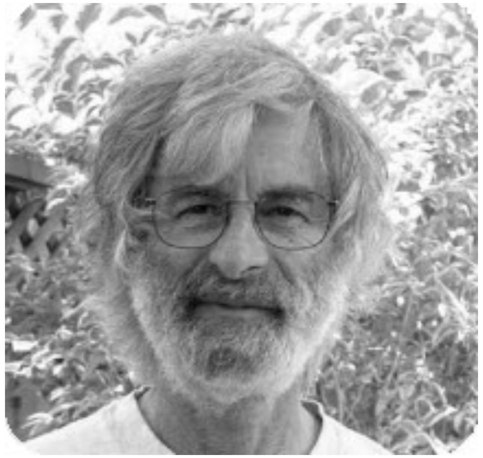
Milner

Pnueli

Clarke

Emerson

Sifakis



Lamport

Success stories

- **Intel** CPU arithmetic and logical operations
- **Microsoft** device drivers
- **Rockwell Collins** AAMP7G microprocessor’s partition management
- **Rolls Royce** Trent Series Health Monitoring Units
- **Lockheed Martin** C130J Mission Computers
- **Boeing** “Little Bird” helicopter (seL4 OS-based mission computer)
- **Royal Navy** Ship/Helicopter Operating Limits Unit
- **Airbus 380** primary flight control software
- **Paris Metro** (RATP)
- **NASA** Mars Rover data management subsystem
- **Bombardier ILLBV950L2** railway interlocking system
- **Apple, ARM/SoftBank, Nvidia, IBM, Oracle** RTL
- **AMD** K5 floating point square root microcode
- **Micrium** OS µC/OS-II real-time kernel

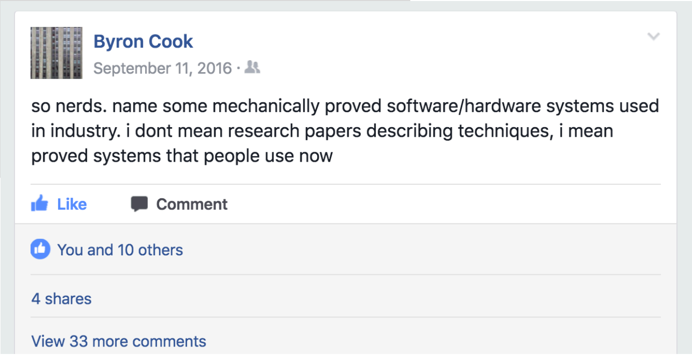
synopsys®

TOYOTA

galois

powered by amazon webservices

Astrée



Byron Cook
September 11, 2016 ·

so nerds. name some mechanically proved software/hardware systems used in industry. i dont mean research papers describing techniques, i mean proved systems that people use now

You and 10 others

4 shares

View 33 more comments

Roadmap

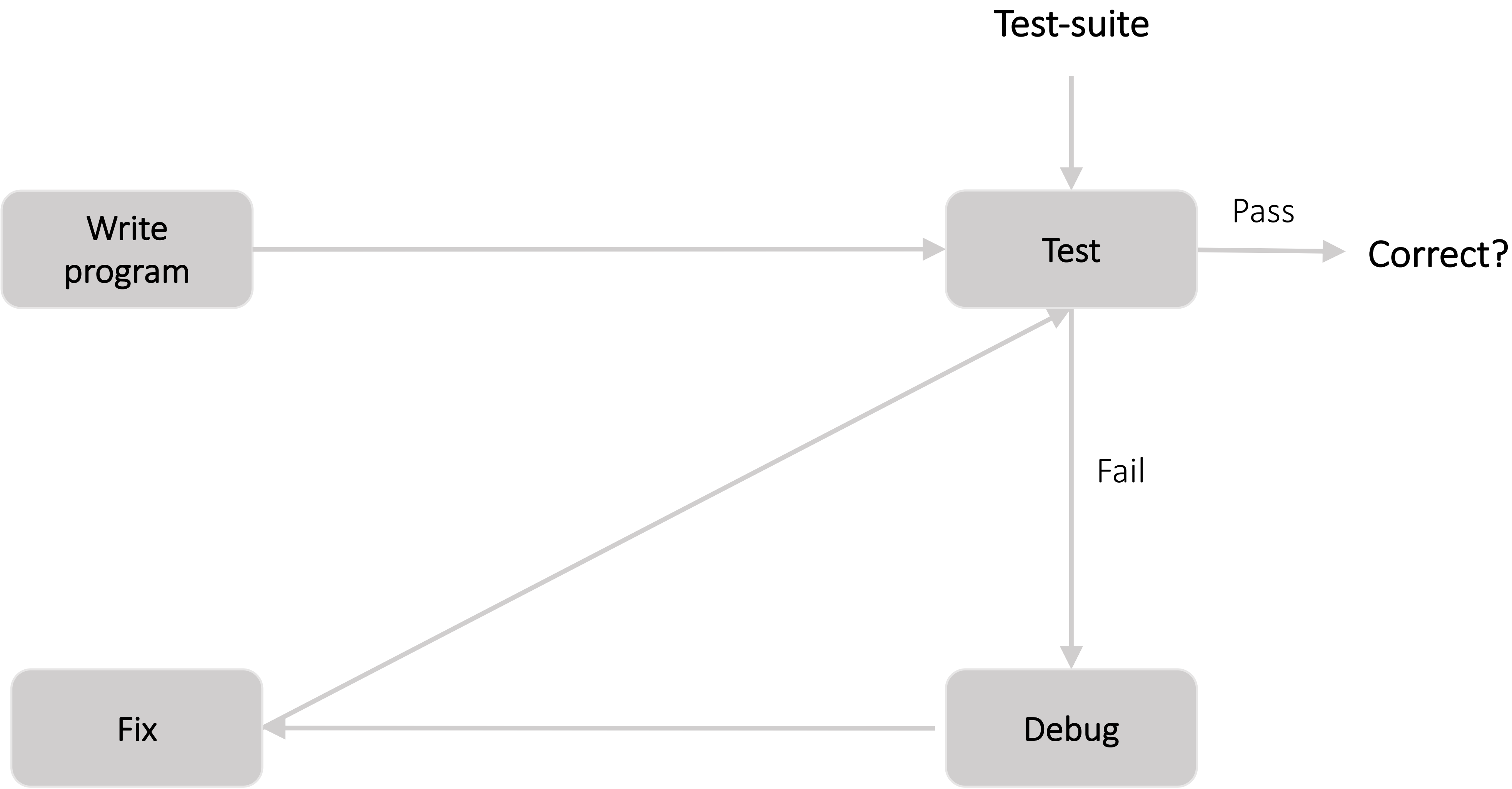
Today

- Motivation
- Overview
- Logistics

Dijkstra

Testing shows the presence,
not the absence of bugs.

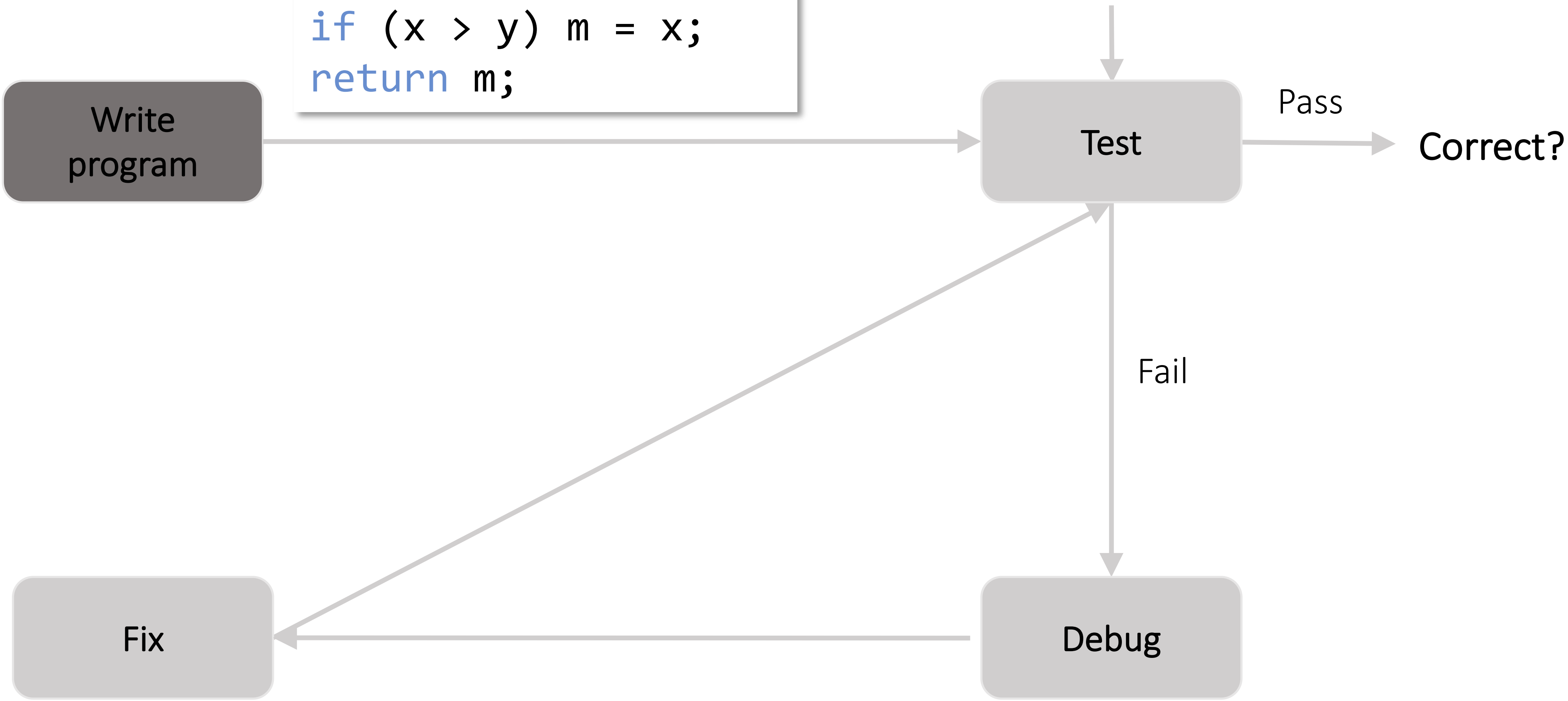
Ergo, testing can fail to show the presence of some bugs!



Test-suite
Write program
Test
Pass
Correct?
Fail
Fix
Debug

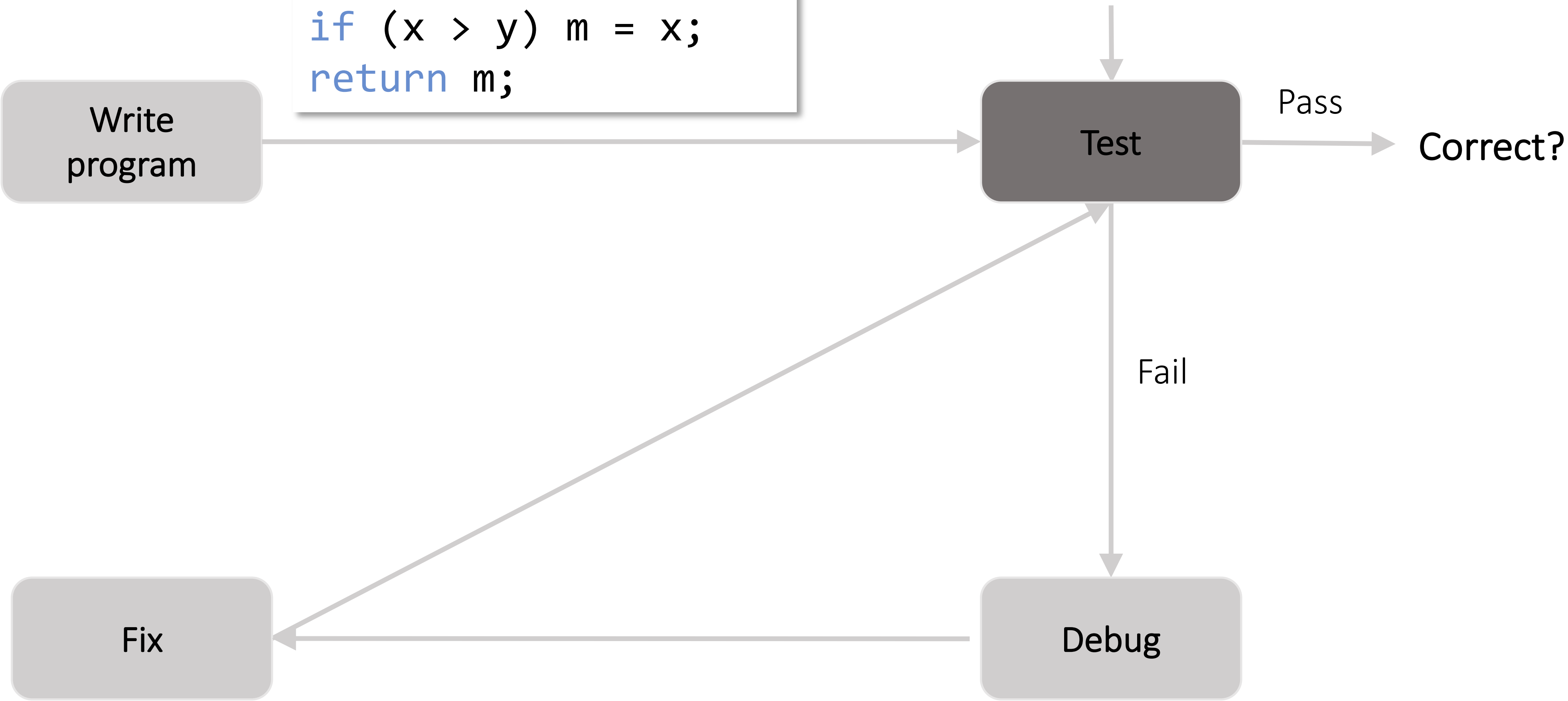
x	y	m	Pass?
3	0	3	
100	99	100	
5	5	5	

```
int max (int x,int y
m = 0;
if (x > y) m = x;
return m;
```

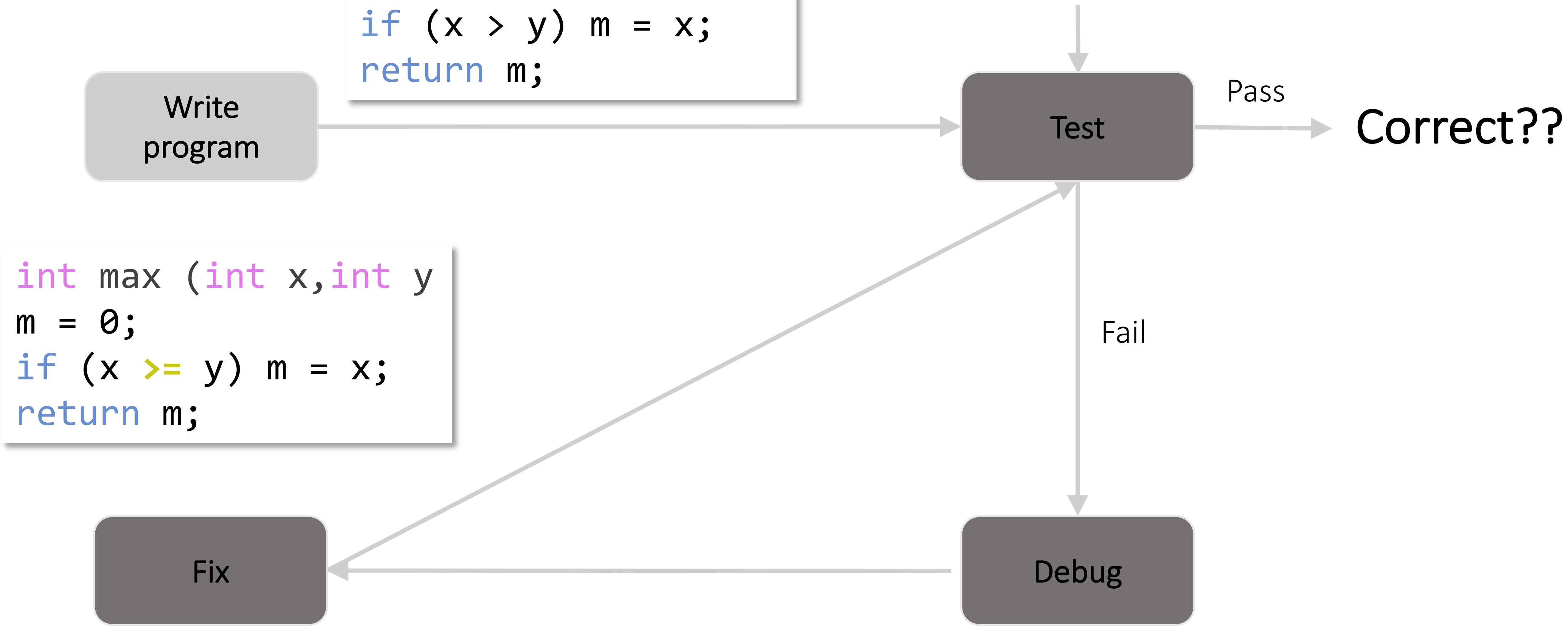


x	y	m	Pass?
3	0	3	✓
100	99	100	✓
5	5	5	✗

```
int max (int x,int y
m = 0;
if (x > y) m = x;
return m;
```



x	y	m	Pass?
3	0	3	√
100	99	100	√
5	5	5	√



```
int max (int x,int y
m = 0;
if (x > y) m = x;
return m;
```

Write program

Test

Pass

Correct??

Fail

Debug

Fix

```
int max (int x,int y
m = 0;
if (x >= y) m = x;
return m;
```

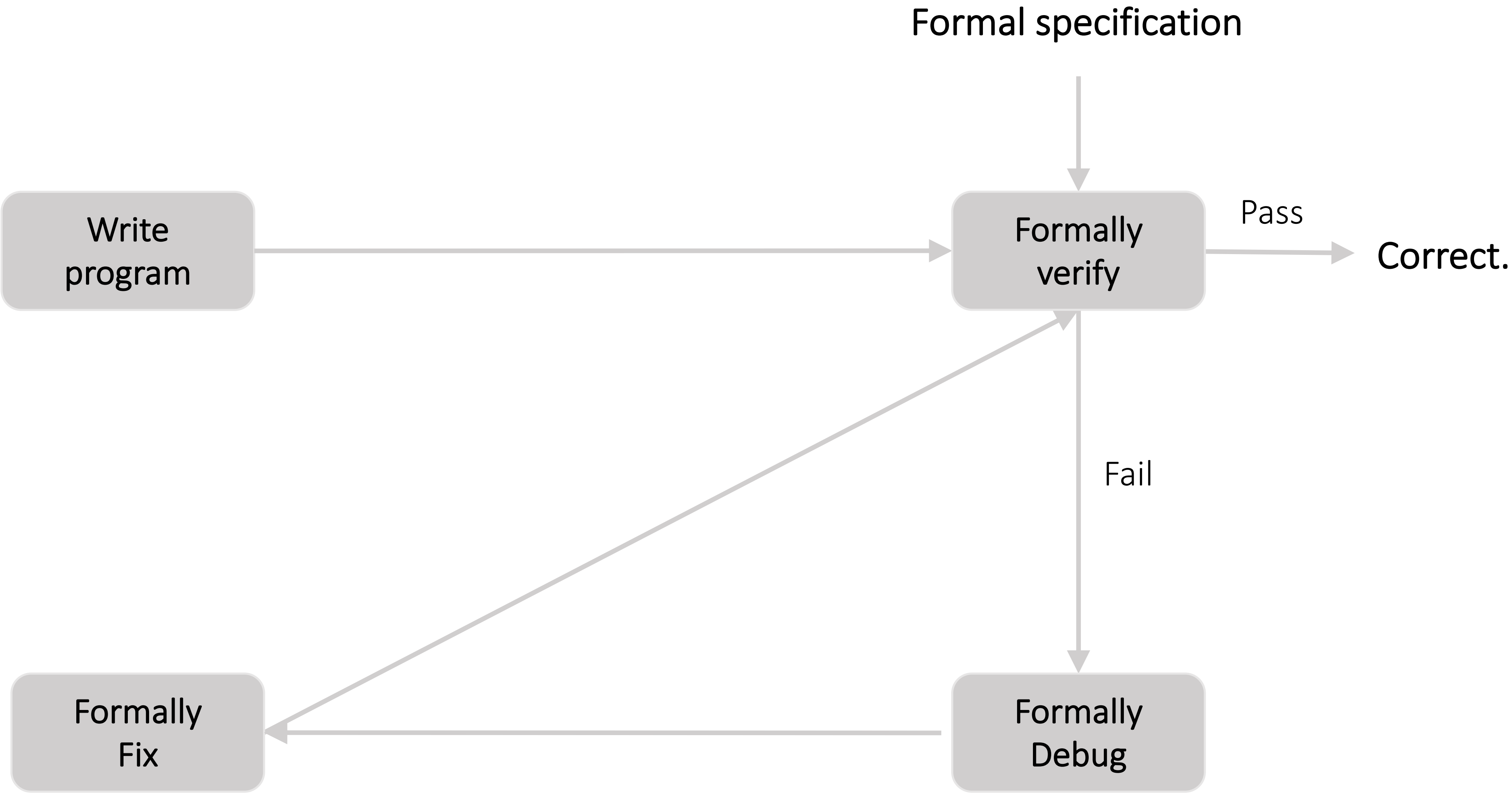
Dijkstra

Testing shows the presence,
not the absence of bugs.

Formal specifications can precisely capture correctness requirements.

Formal verification can **prove** the absence of bugs!

Formal repair can ensure the absence of bugs for programs with bugs!



Formal specification

Write program

Formally verify

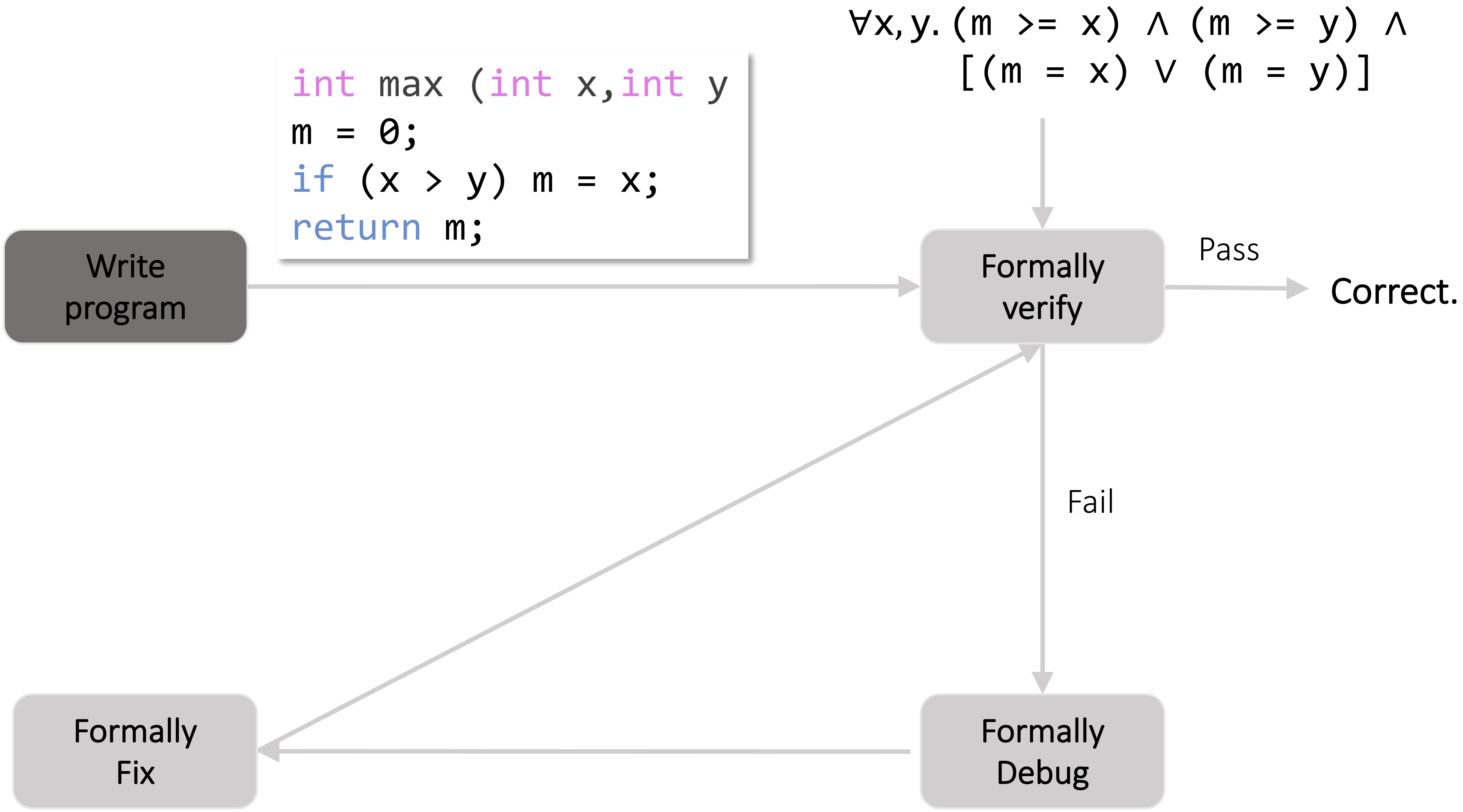
Pass

Correct.

Fail

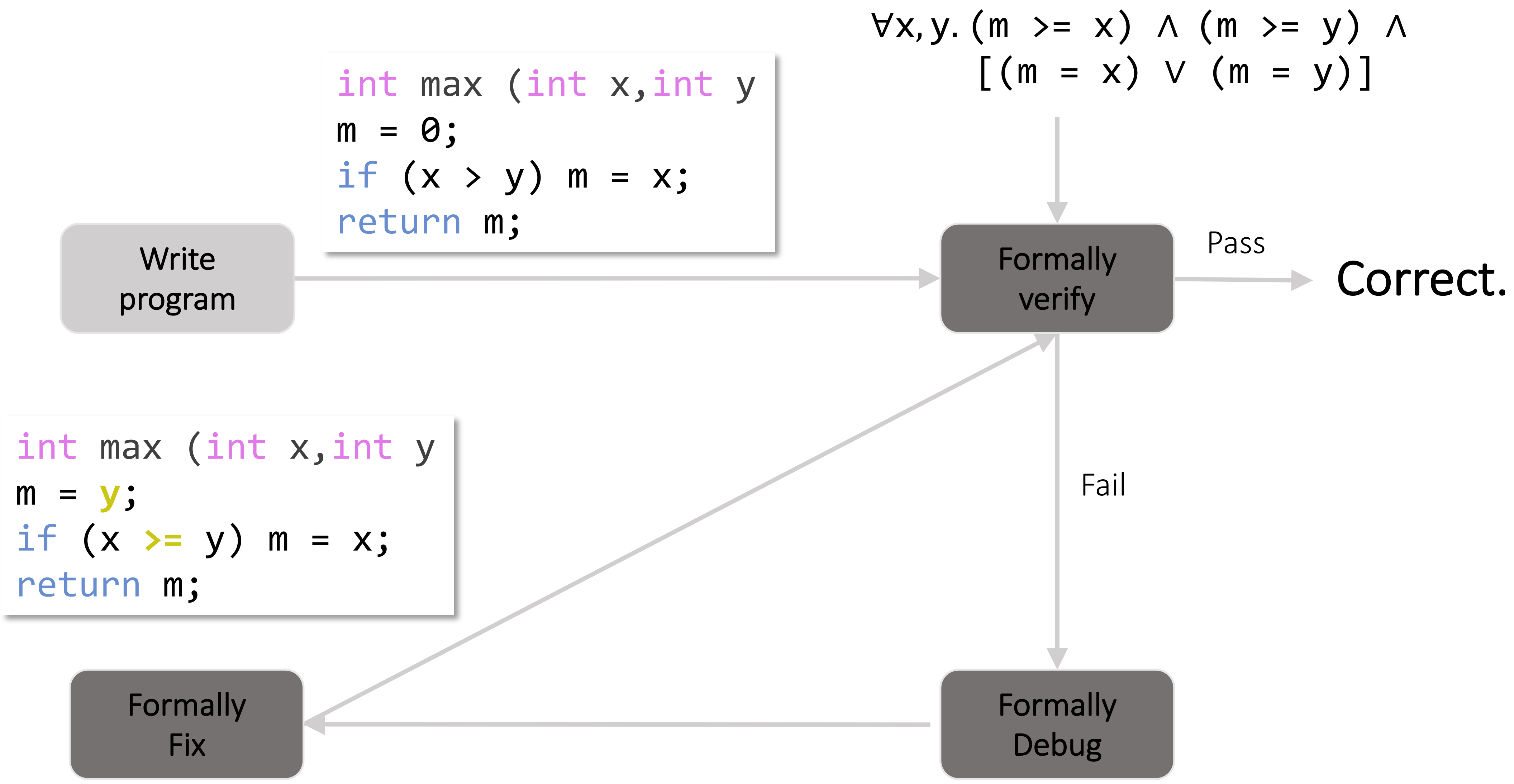
Formally Debug

Formally Fix



$$\forall x,y.\,(m >= x) \land (m >= y) \land [(m = x) \lor (m = y)]$$

```
int max (int x,int y
m = 0;
if (x > y) m = x;
return m;
```



$$\forall x, y.\ (m >= x) \wedge (m >= y) \wedge [(m = x) \vee (m = y)]$$

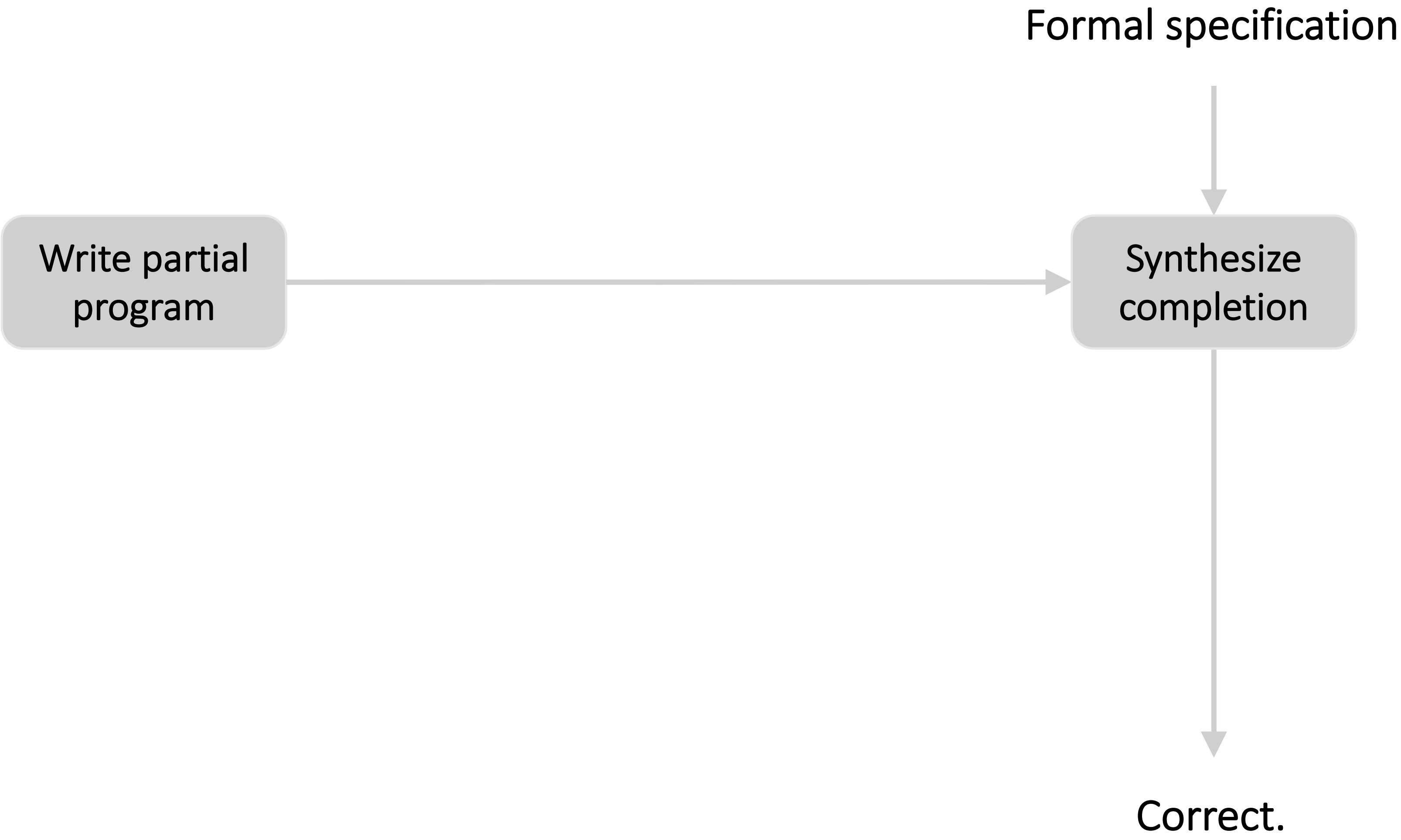
```
int max (int x,int y
m = 0;
if (x > y) m = x;
return m;
```

```
int max (int x,int y
m = y;
if (x >= y) m = x;
return m;
```

Dijkstra

Testing shows the presence,
not the absence of bugs.

Program synthesis can generate programs that are **correct-by-construction**!

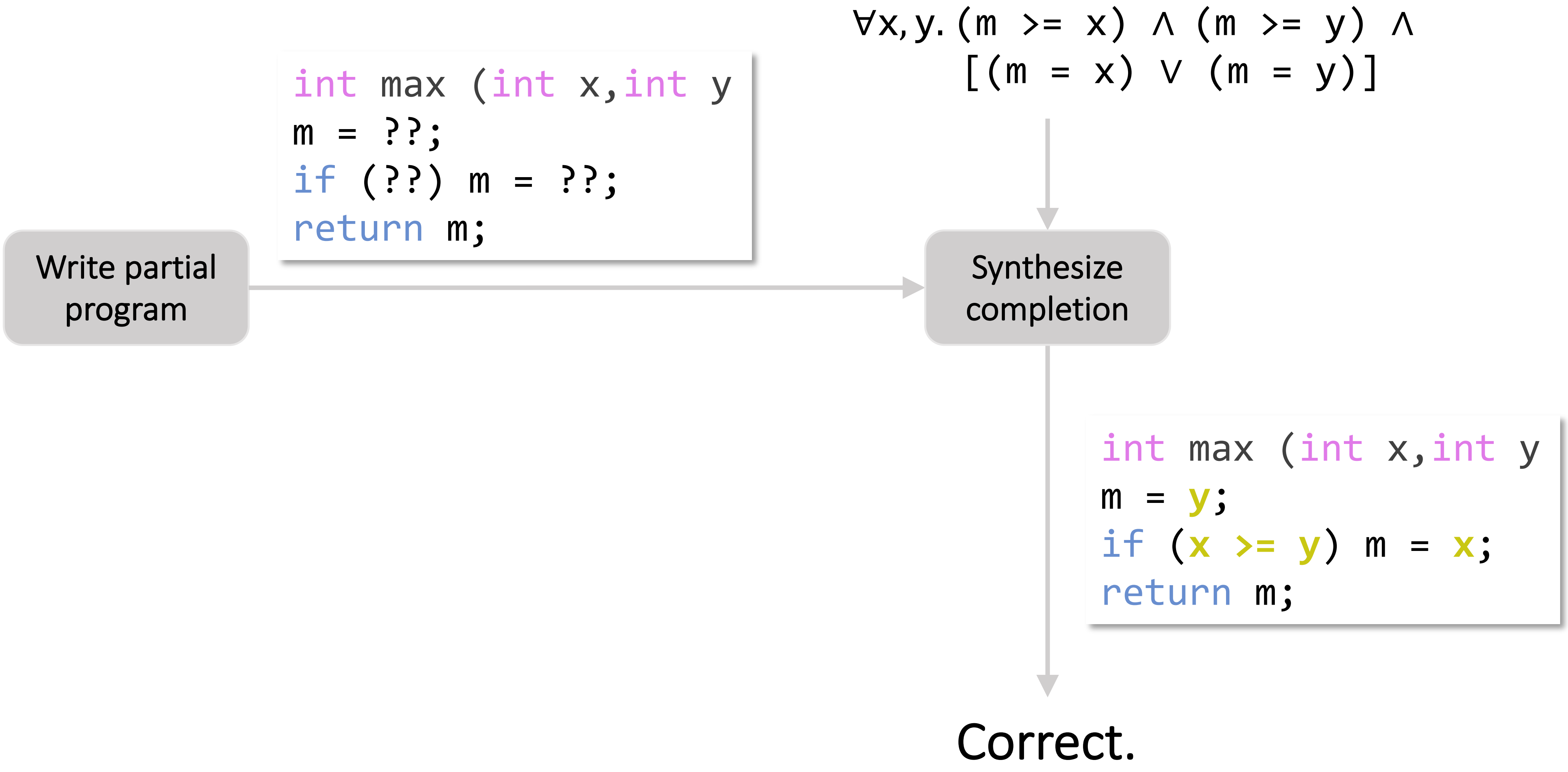


Formal specification

Write partial program

Synthesize completion

Correct.



Write partial program

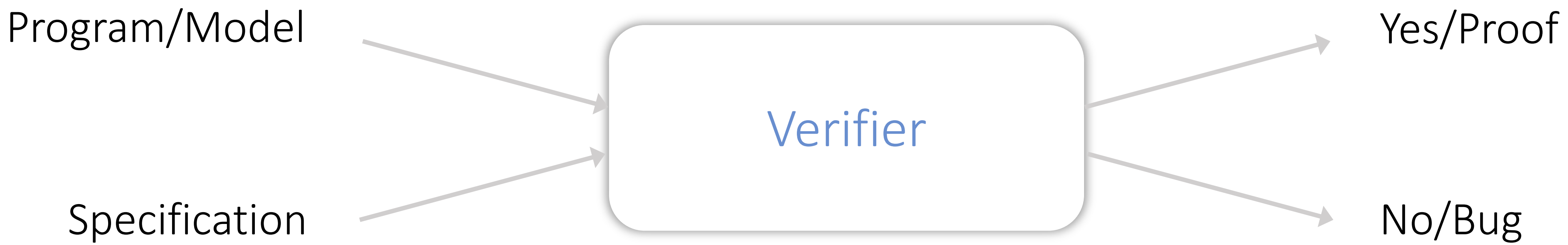
```
int max (int x,int y
m = ??;
if (??) m = ??;
return m;
```

$\forall x, y.\ (m >= x) \wedge (m >= y) \wedge [(m = x) \vee (m = y)]$

Synthesize completion

```
int max (int x,int y
m = y;
if (x >= y) m = x;
return m;
```

Correct.



Program/Model

Specification

Verifier

Yes/Proof

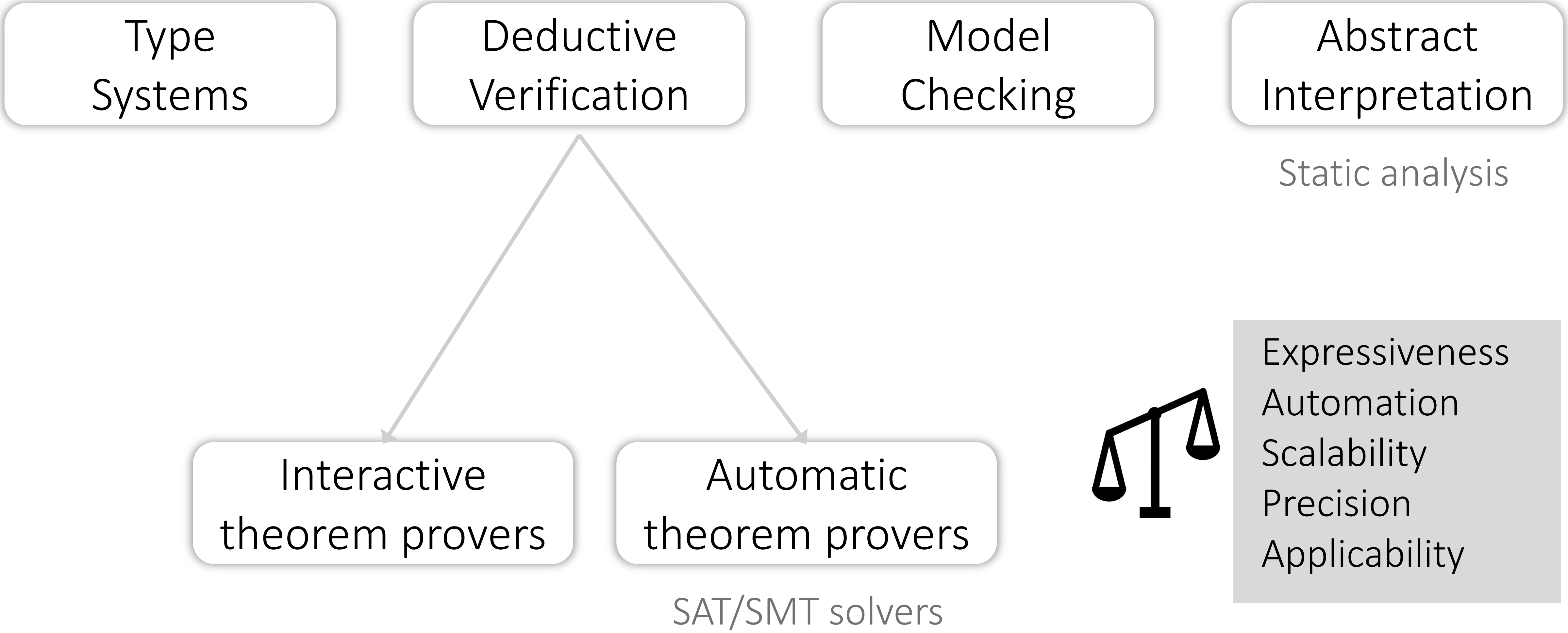
No/Bug

Type Systems

Deductive Verification

Model Checking

Abstract Interpretation



Type Systems

Deductive Verification

Model Checking

Abstract Interpretation

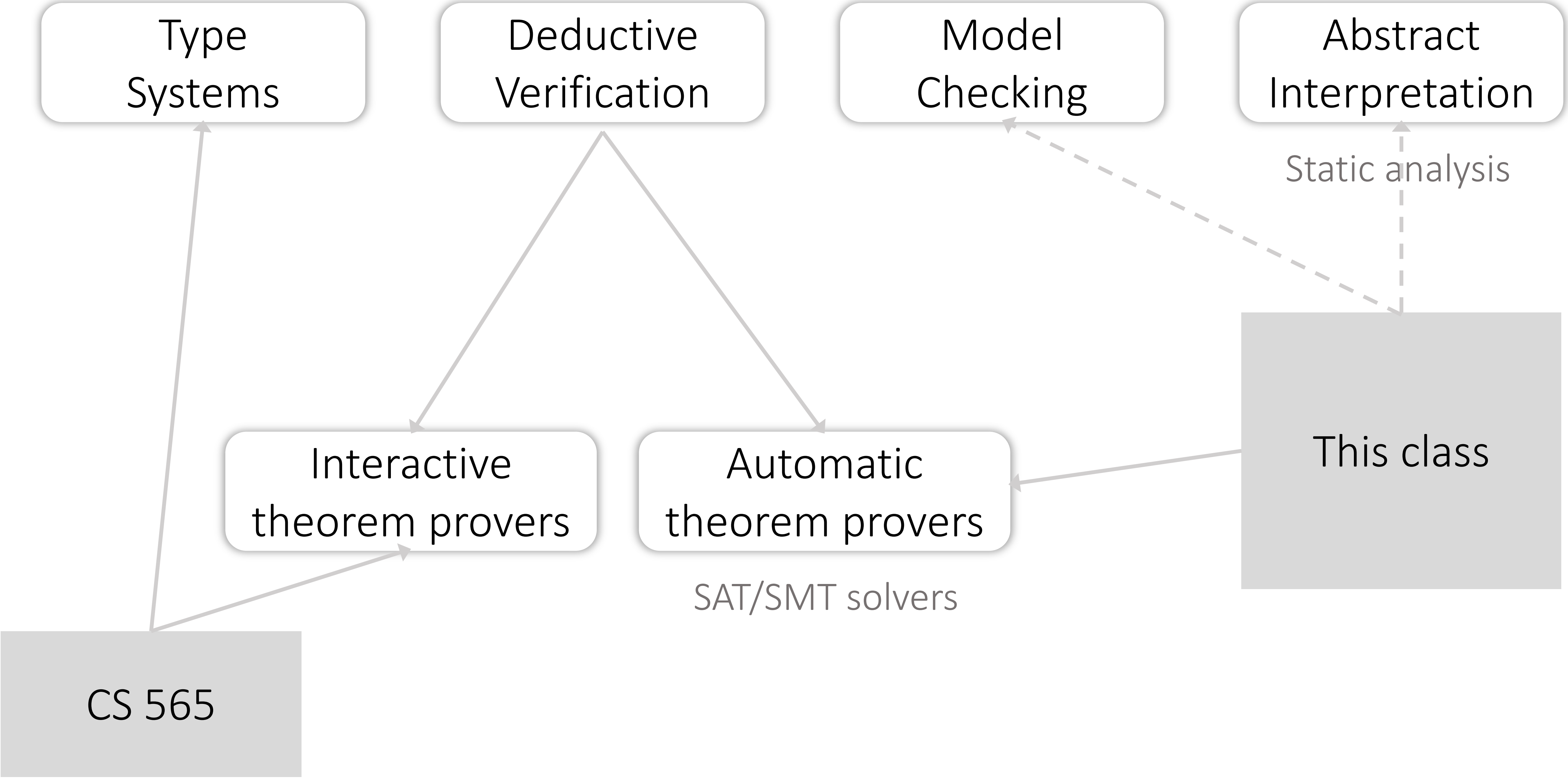
Static analysis

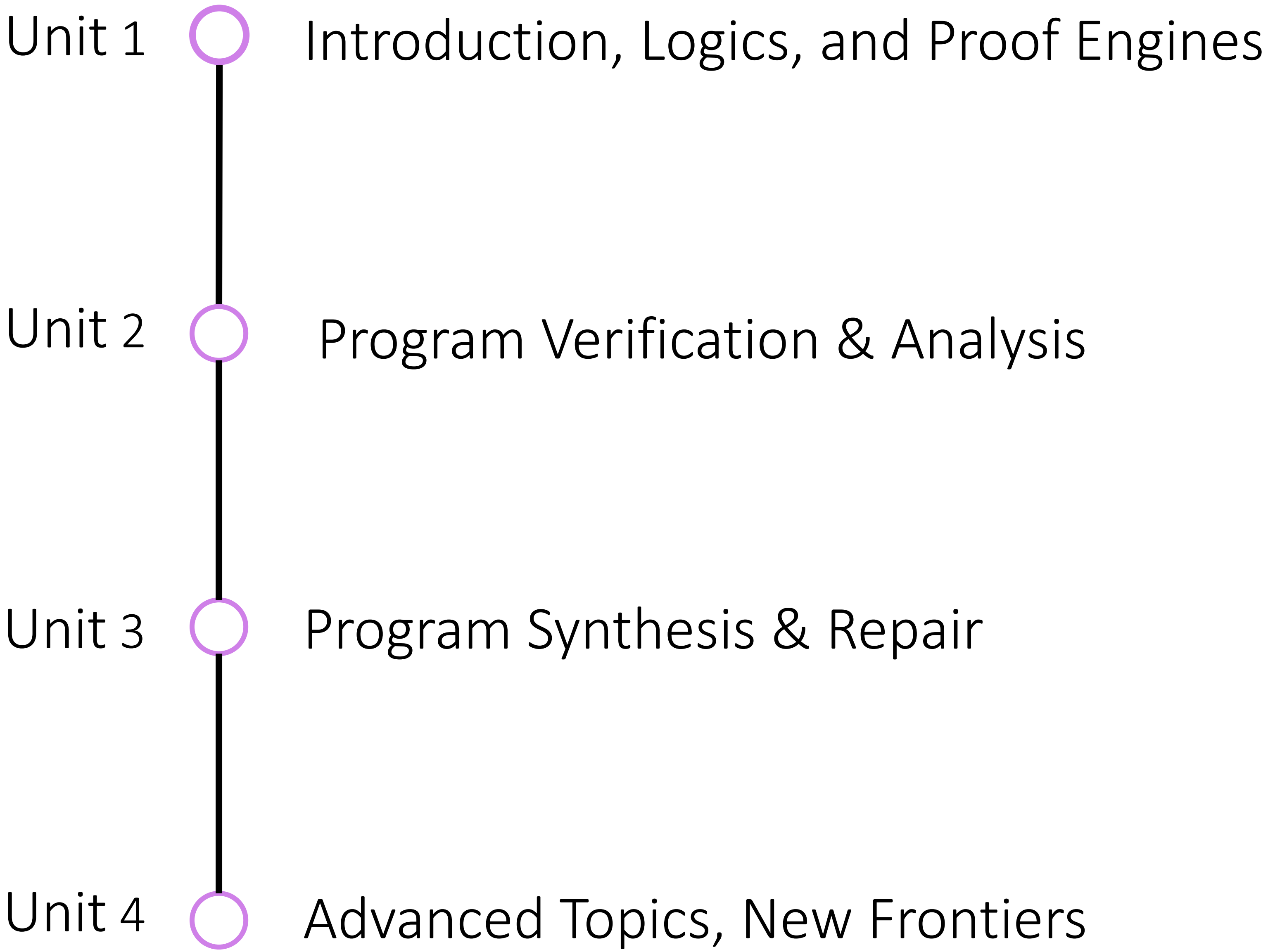
Interactive theorem provers

Automatic theorem provers

SAT/SMT solvers

Expressiveness
Automation
Scalability
Precision
Applicability


Type Systems
Deductive Verification
Model Checking
Abstract Interpretation
Static analysis
Interactive theorem provers
Automatic theorem provers
This class
SAT/SMT solvers
CS 565



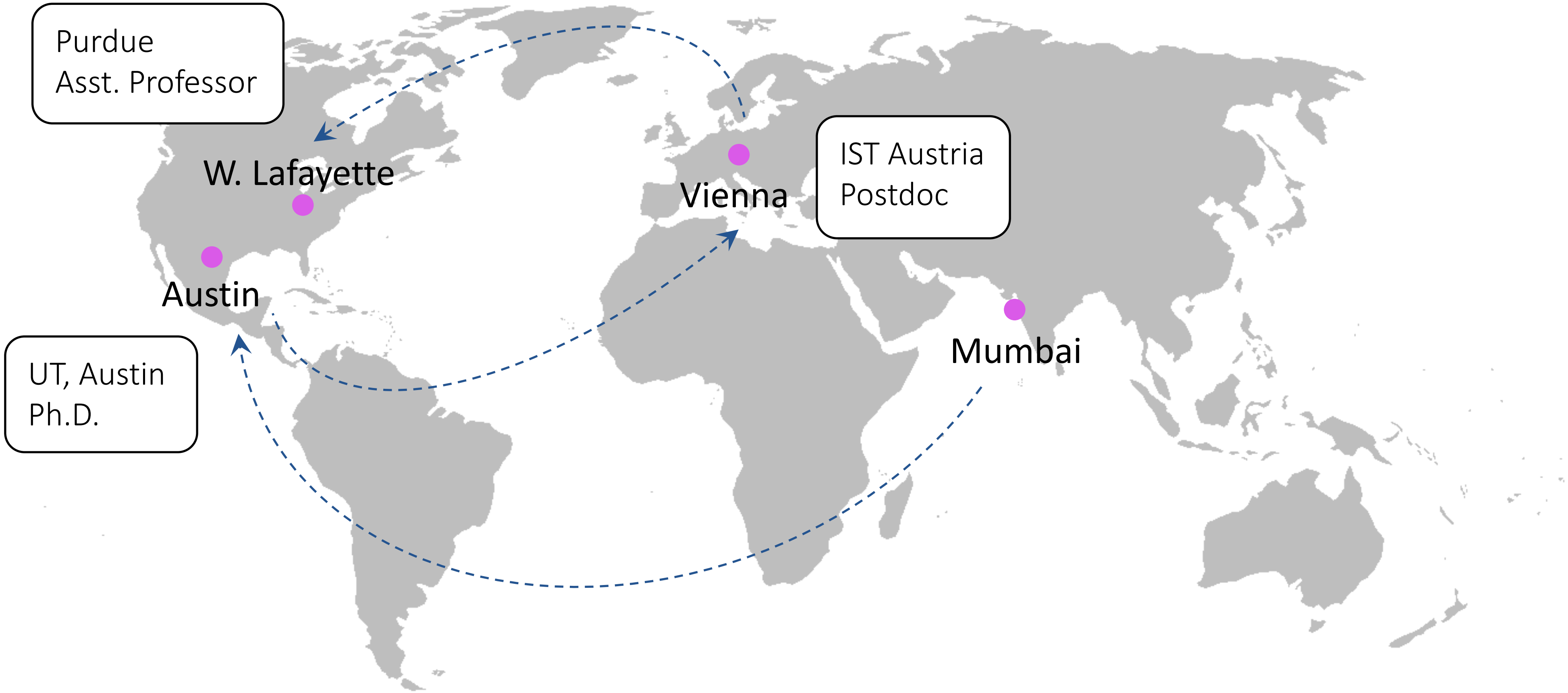
Unit 1	Introduction, Logics, and Proof Engines
Unit 2	Program Verification & Analysis
Unit 3	Program Synthesis & Repair
Unit 4	Advanced Topics, New Frontiers

Roadmap

Today

- Motivation
- Overview
- Logistics

Hi! I am Roopsha.



Purdue
Asst. Professor

IST Austria
Postdoc

UT, Austin
Ph.D.

Developing algorithms/tools to assist programmers in writing reliable programs

DISCOVER[i]

MANTIS

Formal Verification and Synthesis
for Distributed Systems

Semantics-guided
Inductive Program Synthesis

Your turn!

Name?

CS/Math/ECE?

Undergrad/Grad?

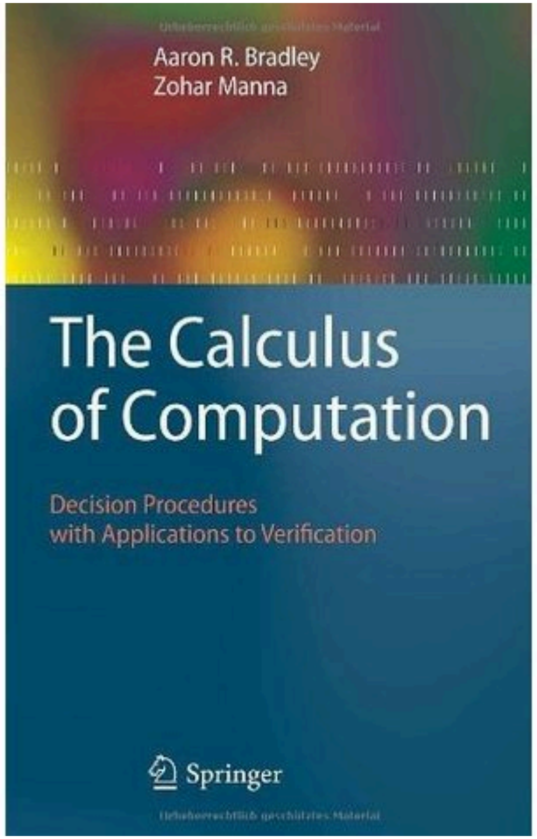
Research Interests?

Why this course?

COVID-19 Impact

Lectures	Zoom SYNC-ONLINE
Syllabus	Course Website
Resources	Brightspace
Discussions	Piazza

If you feel sick, contact Protect Purdue Health Center at 765-496-4636!



Aaron R. Bradley
Zohar Manna

The Calculus of Computation

Decision Procedures
with Applications to Verification

Springer

+

Research papers, survey papers, handbook chapters

Grading

Component	Weight
Class Project	40%
Midterm	20%
Homeworks	35%
Participation	5%

Class project

- You will write a paper and present a talk at our end-of-semester Workshop on Reasoning About Programs (WRAP) 2021!

- You will work in teams of 2-3 students for your project.

  Use Piazza to find teammates.

- Each of you will also *review* your peers’ papers!

- Double-blind reviewing: Reviewer and team identities are concealed.

Class project

- We will do some Semantics-guided Inductive Program Synthesis (MANTIS)!

- You will identify a domain and adapt MANTIS to it.

- *What is MANTIS?* Next class.

Project deliverables

Proposal	Identify team, domain	Feb 11
Partial Paper	Some sections of final paper	Mar 25
WRAP paper	Final paper	Apr 20
WRAP talk	Final presentation	Apr 29

Project grading

- WRAP Paper and Talk: 100% of Project Grade
  Proposal and Partial Paper will not be graded.

- WRAP Paper will be graded by peer reviewers and me.

- WRAP Talk will be graded by me.

Peer Review

- Each of you will serve on the Program Committee (PC) of WRAP 2021!
- Reviewing load: 2-3 papers.
- Each WRAP paper will be reviewed by a subset of your peers and discussed in a PC meeting on Apr 27.
- Goal of PC meeting: Rank papers.
- Reviewing criteria: Contribution, Originality, Presentation

Homeworks

- 5 homeworks
- Theoretical problem sets, programming assignments, paper reviews
- All homeworks will be weighted equally
- Upload to Piazza by 6:00pm on due dates
  Reviews (HW 5) will be due during PC meeting

Policies

- Be honest, reasonable and respectful.
- Presentations, write-ups and homeworks must be your own work.
- Teams are ***not*** allowed to discuss their project specifics with other teams.
- Do not copy text and figures from papers, websites, etc.
  Use your own words. Draw your own figures.
- See course website for all policies.

Summary

Today

- Motivation
- Overview
- Logistics

Next

- Introduction to program synthesis
- Project description