

Advanced Software Testing and Debugging (CS598)

Intro

Fall 2020

Lingming Zhang



Course info

- Instructor: Lingming Zhang
 - Homepage: <http://lingming.cs.illinois.edu/>
 - Email: lingming@illinois.edu
 - Class Time: Tues/Thur 09:30am -- 10:45am (Central Time)
 - Office hours: Tues/Thur 10:45am – 11:45am (Central Time)
- Course:
 - Webpage: <http://lingming.cs.illinois.edu/courses/cs598ast-f20.html>
 - Forum/notifications: piazza.com/illinois/fall2020/cs598ast
 - Zoom link: <https://illinois.zoom.us/j/91263560130>
 - Social events: ???

About me

- This is my first class in Illinois!
- I work on Software Engineering, as well as its synergy with Machine Learning, Formal Methods, and Programming Languages
 - Simply put, **I love building practical systems to deal with all types of software bugs**
- I got my PhD from UT Austin in 2014
- I worked at UT Dallas for 6 years before joining Illinois
- I applied to Illinois for grad school
 - **Obviously I failed:(**

About you

- Who are you (and where are you now)?
- What are you working on or interested in?
- What do you want to learn/obtain from the class?
- Anything else you'd like to share?
 - E.g., what's your story with software bugs? 😊

About the class



Textbook

Class organization

- Discuss two research papers each class
 - They usually belong to the same topic
 - The *primary* paper will be formally presented and discussed
 - The *optional* paper will usually be briefly discussed
- You are required to
 - **Read** both papers before each class
 - **Write** review for the primary paper before each class
 - **Due: 11:59pm before each class day** (submission links on course website)
 - **Participate** in the classroom discussions
 - I will randomly choose students to answer questions
 - **Lead** the discussion for one class
 - **Make your choice before 11:59pm Sept. 5th** (submission link on course website)

Goal of the course

- Get you exposed to real-world software testing and debugging problems
- Get you interested in SE/PL/FM research (if possible)
- Get your feet wet in SE/PL/FM research (through course project)
- Get you familiar with the typical research process (if you are junior PhD students)

Grading

Paper review	20%
Paper presentation	10%
Class participation	10%
Course project	60%

- **No exam!**



Basic questions to ask on a research paper

- Why is the targeted problem important?
- What is the proposed technique and why does it work?
 - Does the proposed technique have enough technical contribution?
- How is the proposed technique evaluated?
 - Are the evaluation benchmarks/subjects real-world systems?
 - Are the used metrics reasonable?
 - Is the experimental procedure replicable?
 - Is it compared against state-of-the-art techniques?
- How are the experimental results?
 - Does it outperform prior work marginally or substantially?
- What are the impacts of this work?
 - Is it working on a rather specific problem or impacting a larger area?
- What are the strengths/limitations for this work?
- What are your suggestions/proposals to further advance this work?

Reading papers

- “How to Read a Research Paper”, by Michael Mitzenmacher
 - <http://www.eecs.harvard.edu/~michaelm/postscripts/ReadPaper.pdf>
- “How to Read an Engineering Research Paper”, by William Griswold
 - <http://cseweb.ucsd.edu/~wgg/CSE210/howtoread.html>
- Advice compiled by Tao Xie:
 - <http://taoxie.cs.illinois.edu/advice.htm#review>

Writing reviews

- “The Task of the Referee”, by Allan Smith
 - <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.177.3844>
- “Constructive and Positive Reviewing” by Mark Hill and Kathryn McKinley
 - <http://www.cs.utexas.edu/users/mckinley/notes/reviewing.html>

Presenting papers

- “How to give strong technical presentations” by Markus Püschel
 - <http://users.ece.cmu.edu/~pueschel/teaching/guides/guide-presentations.pdf>
- Patrick Winston’s talk @ MIT:
 - <https://www.youtube.com/playlist?list=PL9F536001A3C605FC>
- Jean Luc Doumont’s talk
 - <https://www.youtube.com/watch?v=meBXuTIPJQk>

The way to **learn** software engineering is to go out there and **do** software engineering

Course project: group

- The course project will be group-based
 - 1-2 students in each group
 - Feel free to use the “Search for Teammates” thread on Piazza
 - Let me know if you need help to find teammate(s)
- Suggestions for finding your teammate
 - Find someone with **common** interest but **complementary** expertise!

Course project: topic

- A list of example topics on software testing&debugging will be available for you to choose from on Piazza
 - **Research-based ones:** recommended for PhD students
 - **Engineering-based ones:** recommended for MS/MCS students
- **You are encouraged to propose your own topics (subject to my approval)**
 - Especially the ones related to your own research/experience/interest
- Most research-based projects will fall into the following categories:
 - **Bug study:** study a specific (and interesting) type of bugs (>100), discuss the potentially implications for future bug detection, diagnosis, and fixing
 - **Technique study:** empirically study and compare a set of state-of-the-art bug detection, diagnosis, and fixing techniques on new and larger datasets
 - **New technique:** design and build new techniques for better testing and/or debugging of certain types of bugs

Course project: topic selection

- Is this topic an impactful problem?
- Is this topic related to my own research?
- Am I really passionate about this topic?
- More importantly, can I finish this on time and in good shape?
 - Detect unknown bugs, or
 - Outperform state of the art on real-world benchmarks, or
 - Provide practical guidelines for future testing and debugging
- Don't know what to work on yet?
 - Read the course project document and the papers in our schedule!
 - Read more related papers (e.g., ICSE, FSE, ISSTA, PLDI, SOSP/OSDI, Oakland)
 - **Discuss with me!**

Course project: deadlines

- **Proposal (due 9/28)**

- What is the targeted problem
- Why is it important
- How you will do it
- How you will evaluate it
- What is your plan and expected outcome

- **Deliverables**

- 1-page .txt proposal
- 5min presentation (9/24)

- **Midterm (due 11/15)**

- What have you done
- Any challenges you have faced
- Any changes you have made since proposal
- Concrete plan for final report

- **Deliverables**

- 3-4 page PDF report
- 10min presentation (11/10)

- **Final (due 12/15)**

- What is the targeted problem
- Why is it important
- How you have done it
- How you have evaluated it
- What is your outcome

- **Deliverables**

- 5-6 page PDF report
- 15min presentation (12/08)

The final report/presentation will be evaluated based on real research paper standards (e.g., the ones you are going to read) 18

Why this course?

- Software bugs are inevitable!
 - Programming still mainly a manual process
 - Software systems can be rather complicated
 - Software systems can be evolving
 - Interaction between software systems
 - Dependence on hardware supports
 - ...



The first “bug”

“You were partly correct, I did find a ‘bug’ in my apparatus, but it was not in the telephone proper...”

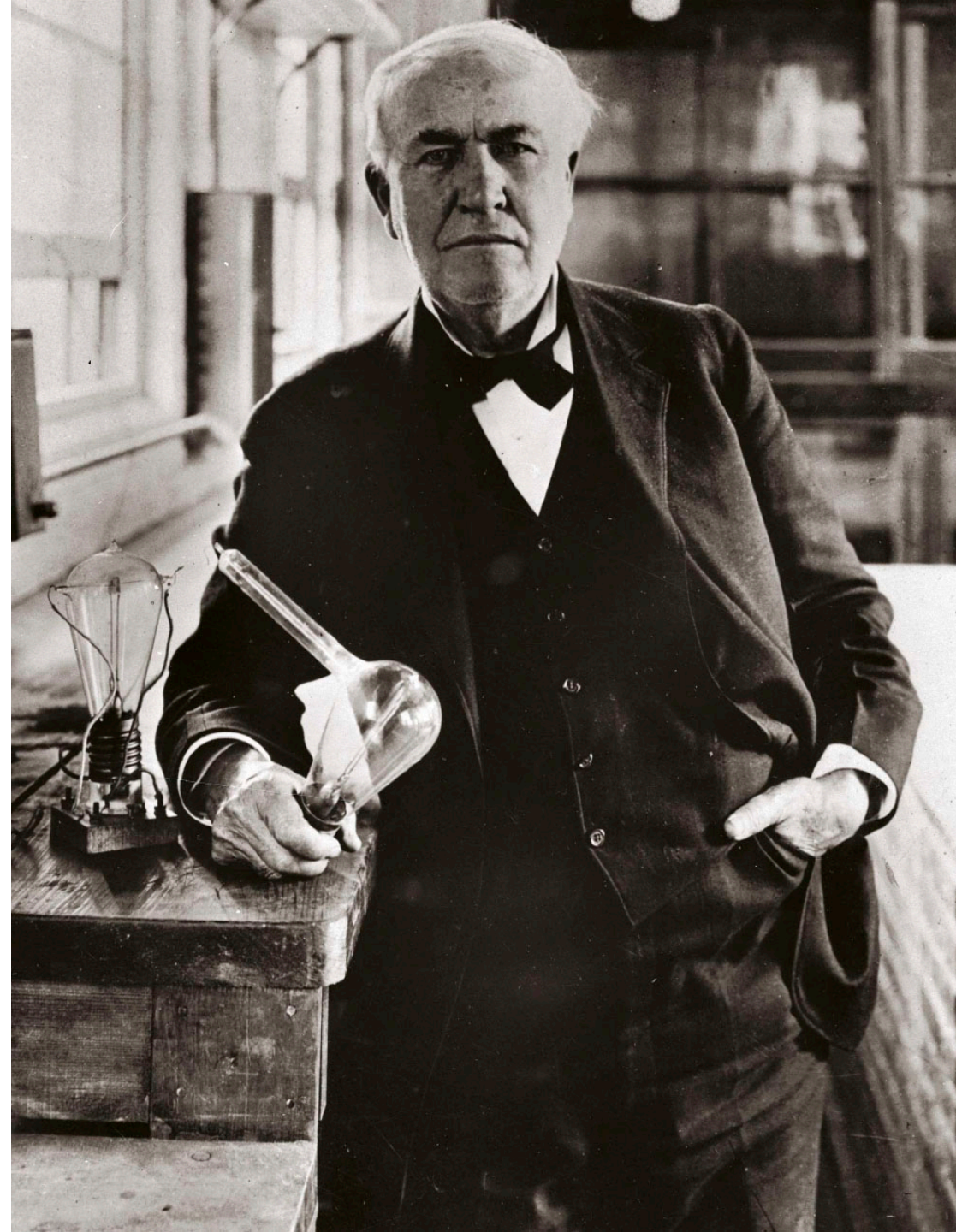
Thomas Edison (early 1880s)

Menlo Park Mich 3rd 78

W^m Otton Esqr

Dear Sir

You were partly correct, I did find a “bug” in my apparatus, but it was not in the telephone proper. It was of the genus “callbellum”. The insect appears to find conditions for its existence in all call apparatus of telephones. Another delay was the sickness of Adams wife.



The first computer "bug"

"First actual case of bug being found."

Grace Hopper (1947)

Photo # NH 96566-KN (Color) First Computer "Bug", 1947

9/2

9/9

0800 Antan started
1000 " stopped - antan ✓

13 ⁰⁰ MC (032) MP-MC	1.582169000	1.2700	9.037 847 025
(033) PRO 2	2.130476415	2.130476415	9.037 846 995 com
conv	2.130476415		4.615925059(-2)


Relays 6-2 in 033 failed special speed test
in relay " 11.000 test.

Relays changed

1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.

1545

Relay #70 Panel F
(moth) in relay.



First actual case of bug being found.

1630 Antan started.
1700 closed down.





Nowadays,
software is
everywhere!

So are software bugs...



THE SPOKESMAN-REVIEW

NEWS > BUSINESS

British Airways computer problem strands 20,000

Wed., Aug. 7, 2019



This Jan. 10, 2017 file photo, British Airways planes are parked at Heathrow Airport in London. British Airways said Wednesday Aug. 7, 2019, it has canceled some dozens of flights from London airports after its check-in systems were hit by a computer glitch. (Frank Augstein / AP)



Technology

Tesla sued by family of Apple engineer killed in Autopilot crash

"Tesla's Autopilot feature was defective and caused Huang's death," attorneys for Walter Huang said.



The New York Times Airline Blames Bad Software in San Francisco Crash



Software testing and debugging cycle

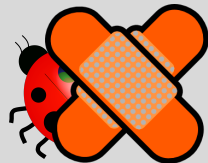
Detect bugs!



Localize bugs!



Fix bugs!



- Build cycles per day:
 - Google: 17K*
 - Facebook: 60K
 - Microsoft: 30K



- ❖ Google: <https://bit.ly/2SYY4rR>
- ❖ Facebook: <https://bit.ly/2CAPvN9> (Android only)
- ❖ Microsoft: <https://bit.ly/2HgjUpw>

Course topics (tentative)

Bug detection

Guided Unit Test Gen
Symbolic Execution
Spec-based Test Gen
Fuzz Testing
Human-assisted Bug Detection
Oracle Inference
Regression Testing

} White-box testing
} Black-box testing

Bug fixing

Search-based Program Repair
Semantics-based Program Repair
Faster Program Repair
Unified Debugging

Bug diagnosis&localization

Failure Analysis and Cause Reduction
Fault Localization

Testing&debugging for more

Flaky Tests
ML: Deep Learning Models/Libs
FM: SMT Solvers
DB: DB Engines

This is tentative, let me know your thoughts!

Guided unit test generation

```
public class HashSet extends Set{  
    public boolean add(Object o){...}  
    public boolean remove(Object o){...}  
    public boolean isEmpty(){...}  
    public boolean equals(Object o){...}  
    ...  
}
```

Program under test

- Feedback guide:
 - Discarding illegal/redundant tests
 - Covering more object states
 - Closer to coverage targets
 - ...

Generation

```
Set s = new HashSet();  
s.add("hi");
```

Generated test *t1*

```
Set s = new HashSet();  
s.add("hi");  
s.remove(null);
```

Generated test *t2*

```
Set s = new HashSet();  
s.isEmpty();  
s.remove("no");  
s.isEmpty();  
s.add("no");  
s.isEmpty();  
s.isEmpty();  
...
```

Generated test *t3*

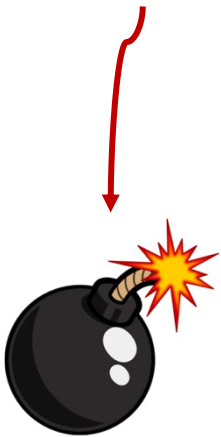
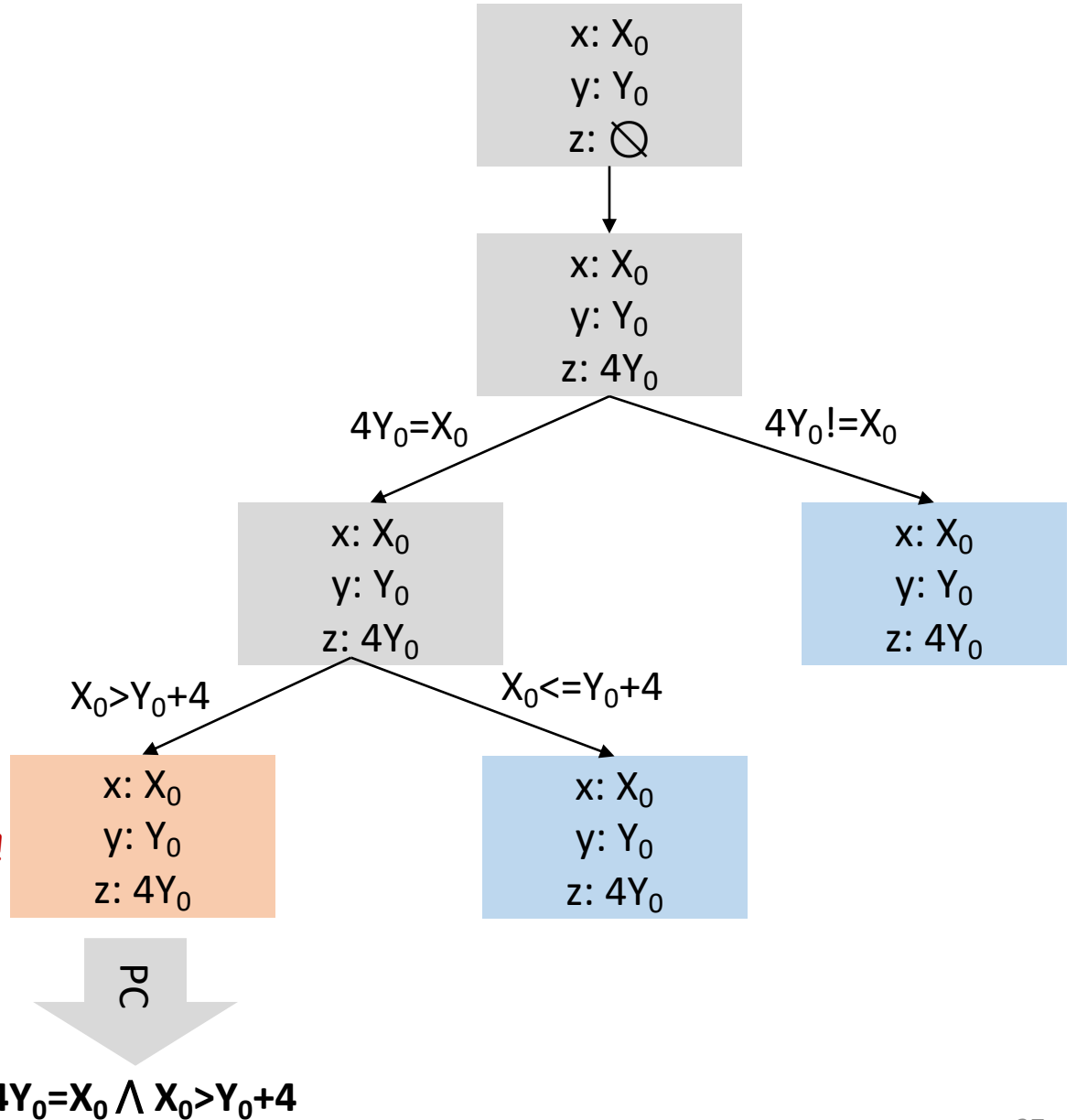
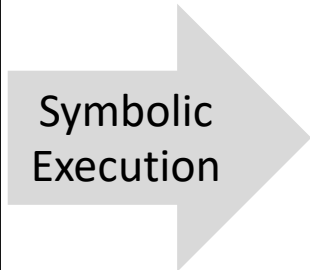
...

Feedback

Symbolic execution

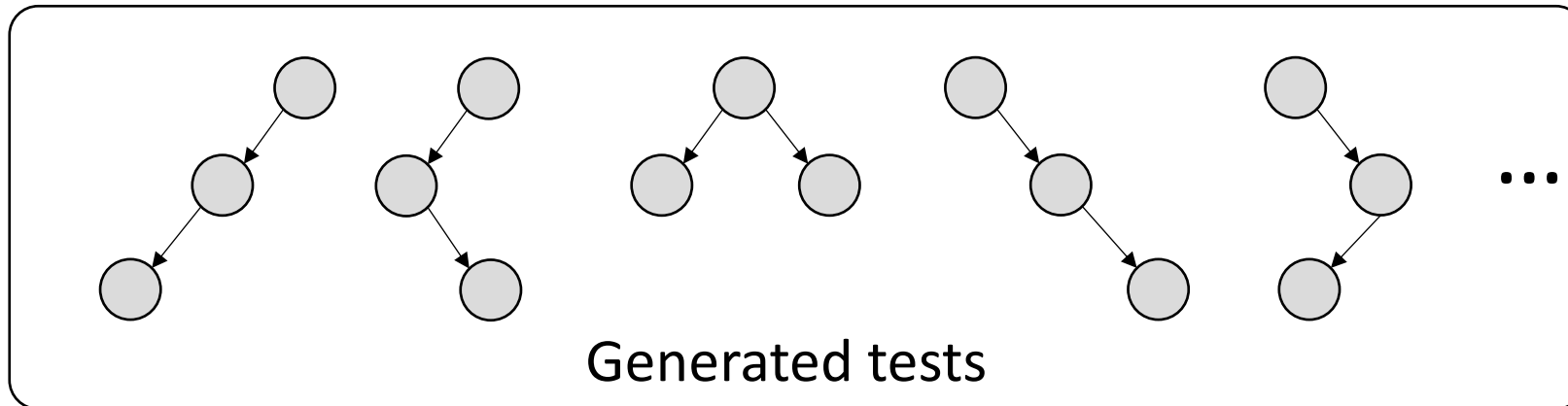
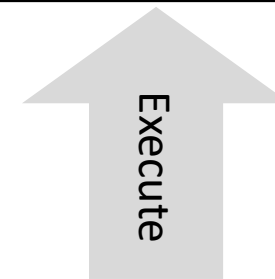
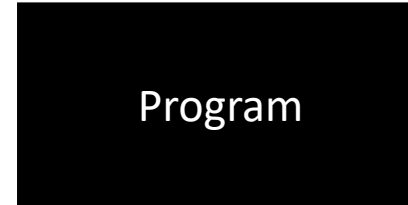
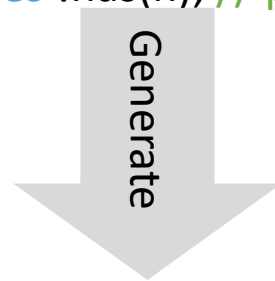
```
void testme (int x, int y) {  
1 z = 4 * y;  
2 if (z == x) {  
3   if (x > y+4) {  
4     ERROR;  
5   }  
6 }  
}
```

Program under test



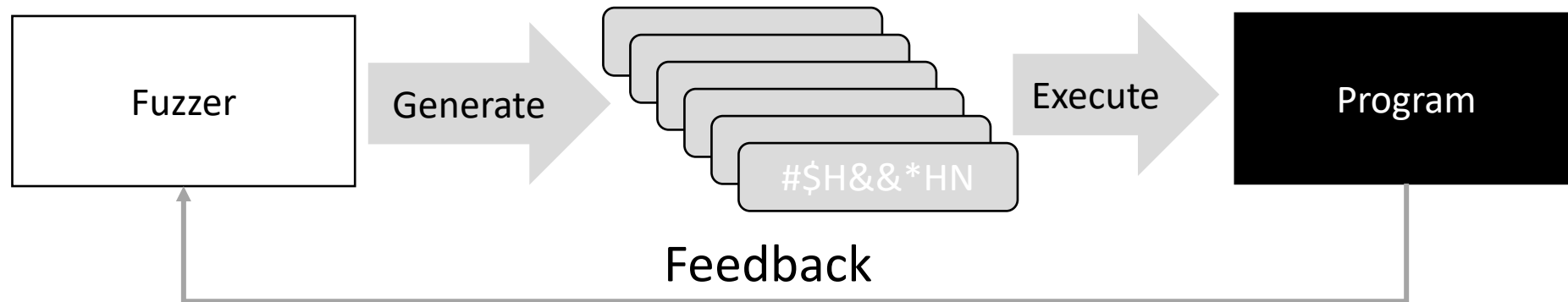
Spec-based testing

```
// specification for removing from binary tree  
/*@ public normal_behavior  
@ requires has(n); // precondition  
@ ensures !has(n); // postcondition @*/
```



Fuzz testing

./Program < /dev/random

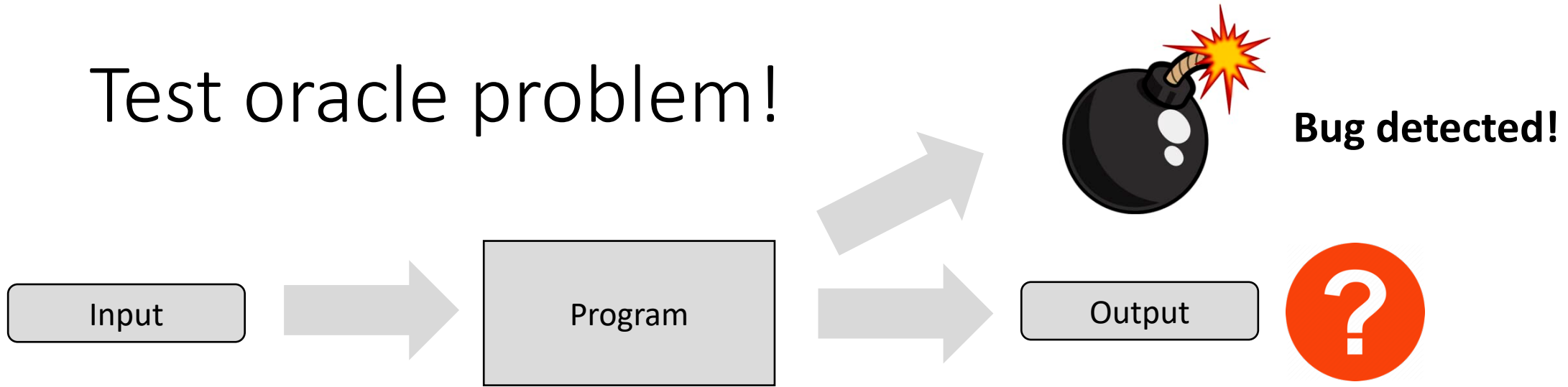


- Fuzzing strategies
 - Mutation-based
 - Grammar-based
 - Learning-based

- Feedback guide
 - New coverage?
 - Longer execution?
 - Valid input?

- Targeted programs
 - Binaries
 - Compilers
 - Browsers
 - Deep learning systems
 - ...

Test oracle problem!



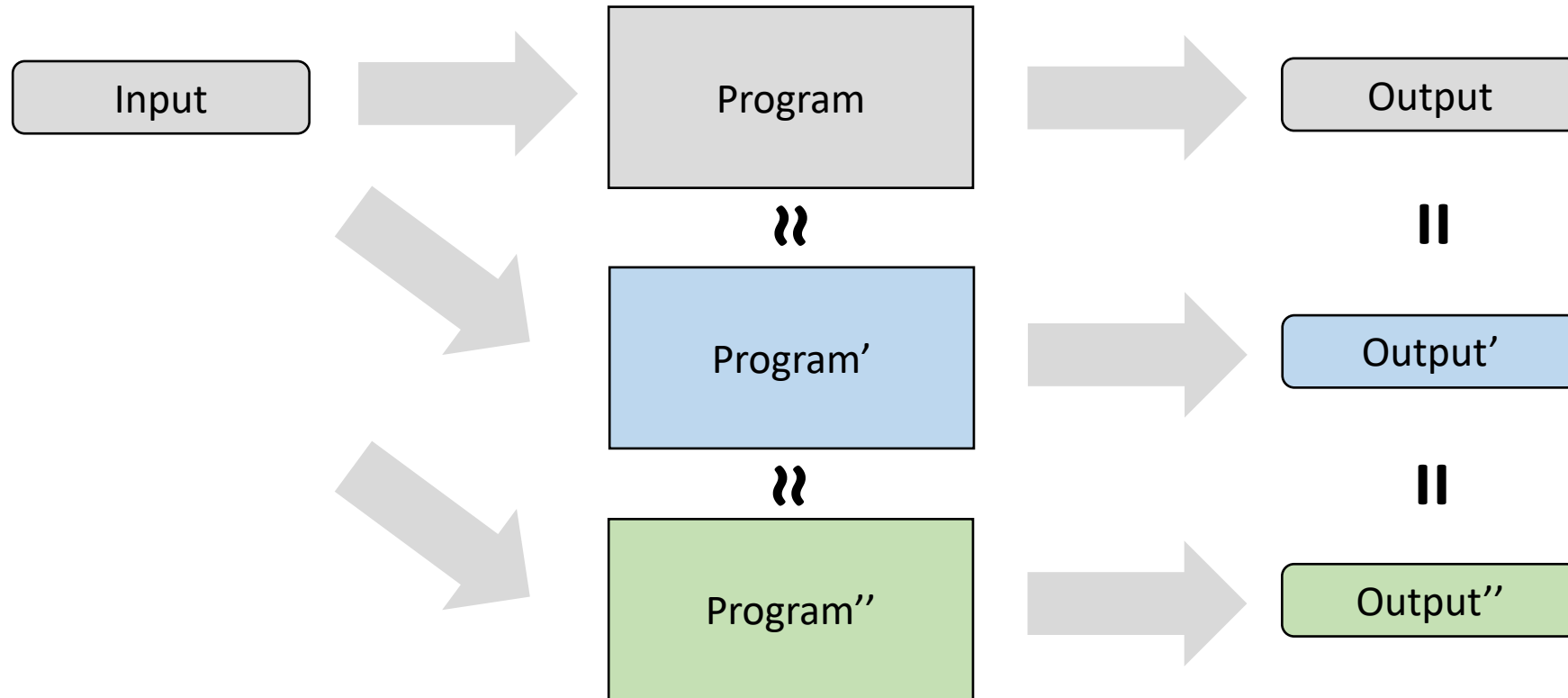
Test oracle: a mechanism for determining whether software executed correctly for a test¹.

One of the hardest problem in Software Engineering!

How to mitigate it?

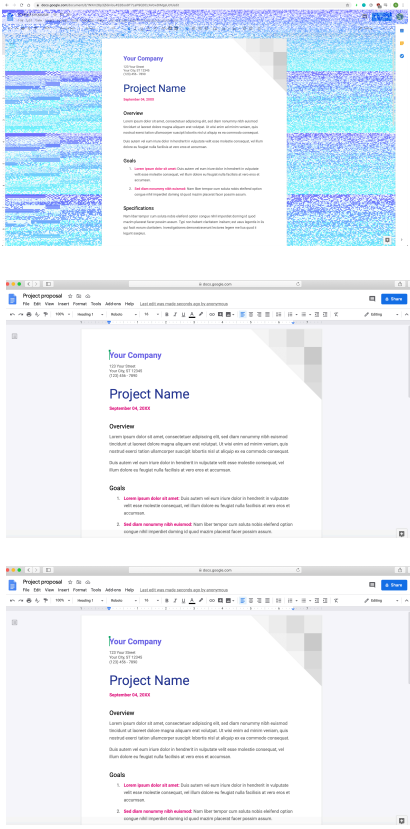
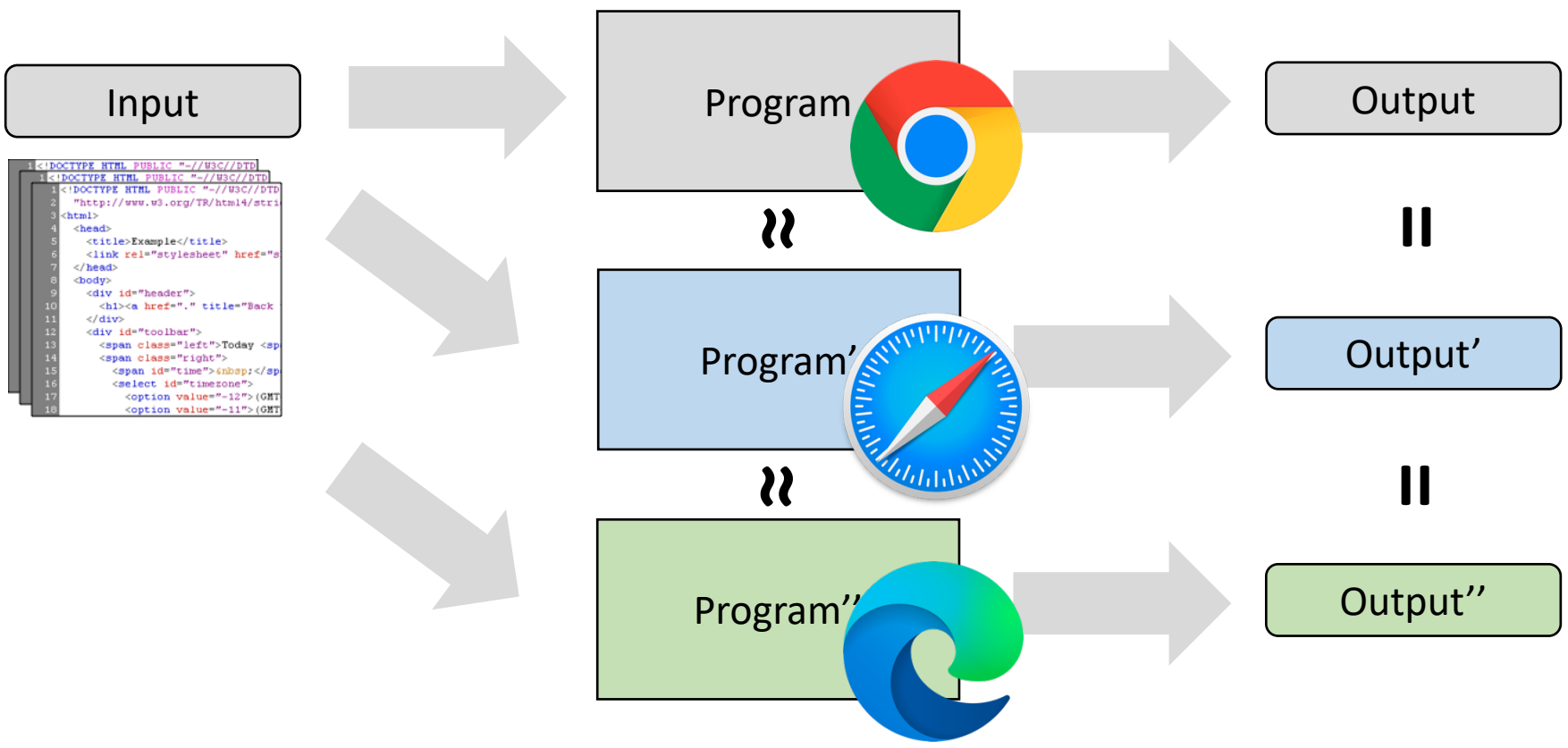
¹https://en.wikipedia.org/wiki/Test_oracle

Differential testing



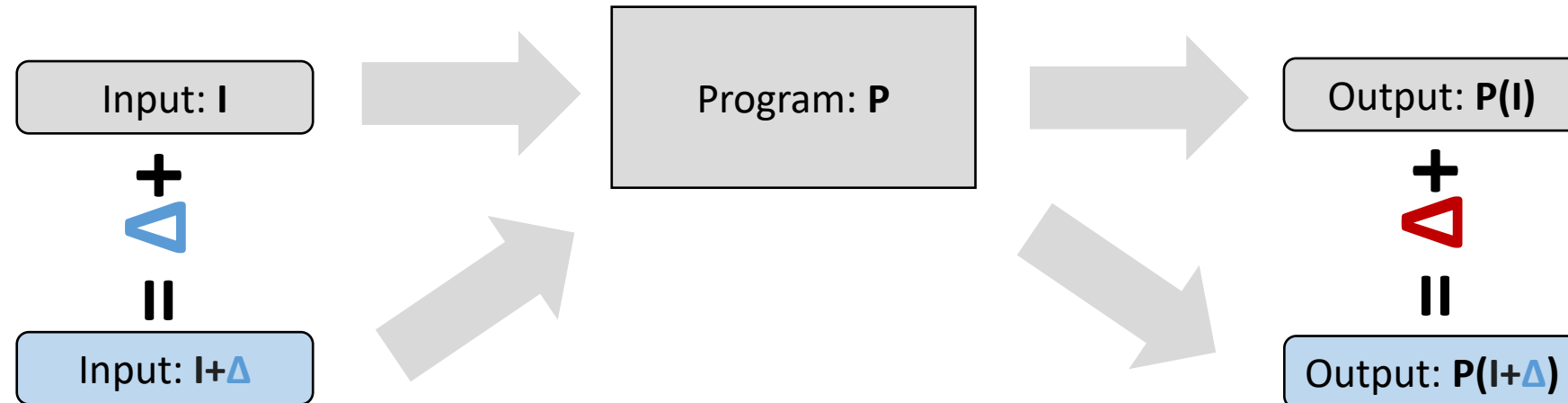
Provide the same input to **similar** applications, and observe output **differences**

Differential testing: browsers



Provide the same input to **similar** applications, and observe output **differences**

Metamorphic testing



$$P(I+\Delta) = P(I)+\Delta$$

For example:
 $\sin(I+2\pi) = \sin(I)$
 $\sin(-I) = -\sin(I)$

Provide the manipulated inputs to **same** application,
and observe if output **differences** are as expected

Oracle inference

```
public class HashSet extends Set{
  /*@ public normal_behavior
  @ requires !has(o); // precondition
  @ ensures has(o); // postcondition @*/
  public boolean add(Object o){...}
  /*@ public normal_behavior
  @ requires has(o); // precondition
  @ ensures !has(o); // postcondition @*/
  public boolean remove(Object o){...}
  ...
}
```

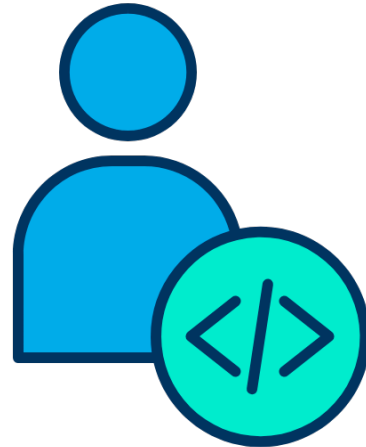
Program under test

- How to automatically obtain such predicates?
 - Manual summarization
 - Automated inference via analyzing the **current project**
 - Learning/mining from **other projects**

Human-assisted bug detection



Historical bugs

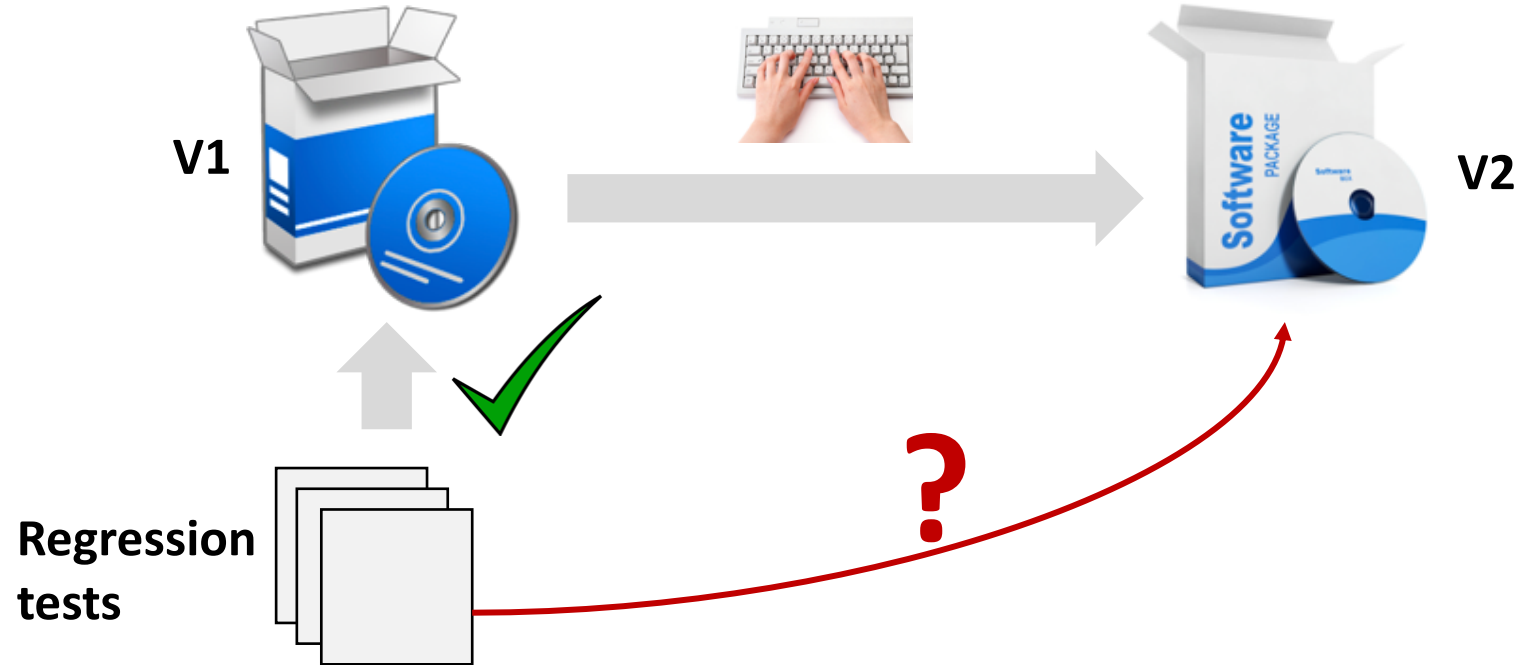


Manual inspection



Derived techniques

Regression testing



- Regression testing is extremely costly in practice
 - Facebook has over 10,000 tests run per change
 - Google has over 150 million test executions per day

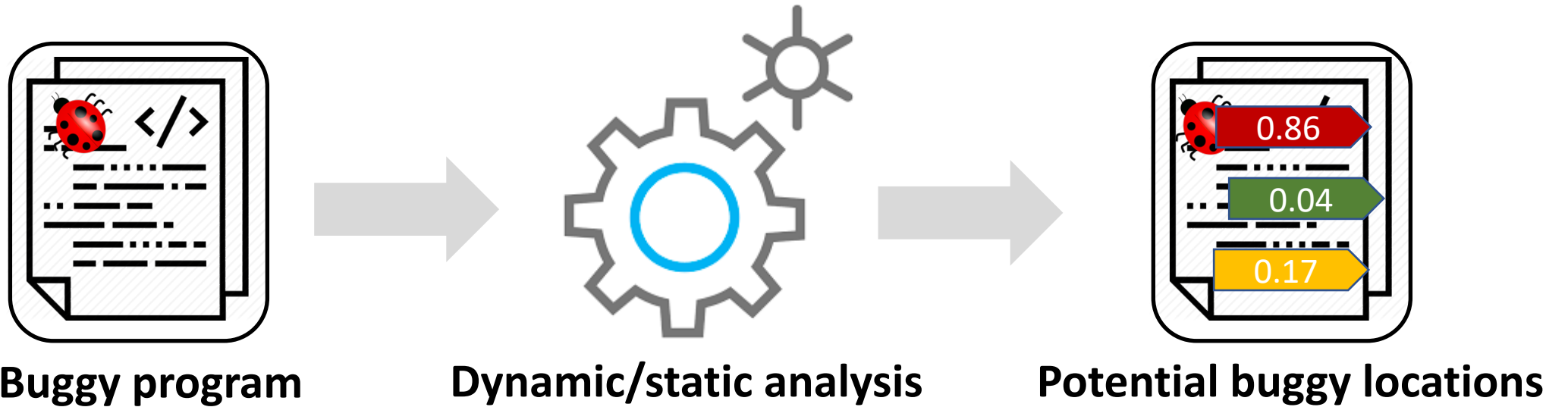


Failure Analysis and Cause Reduction

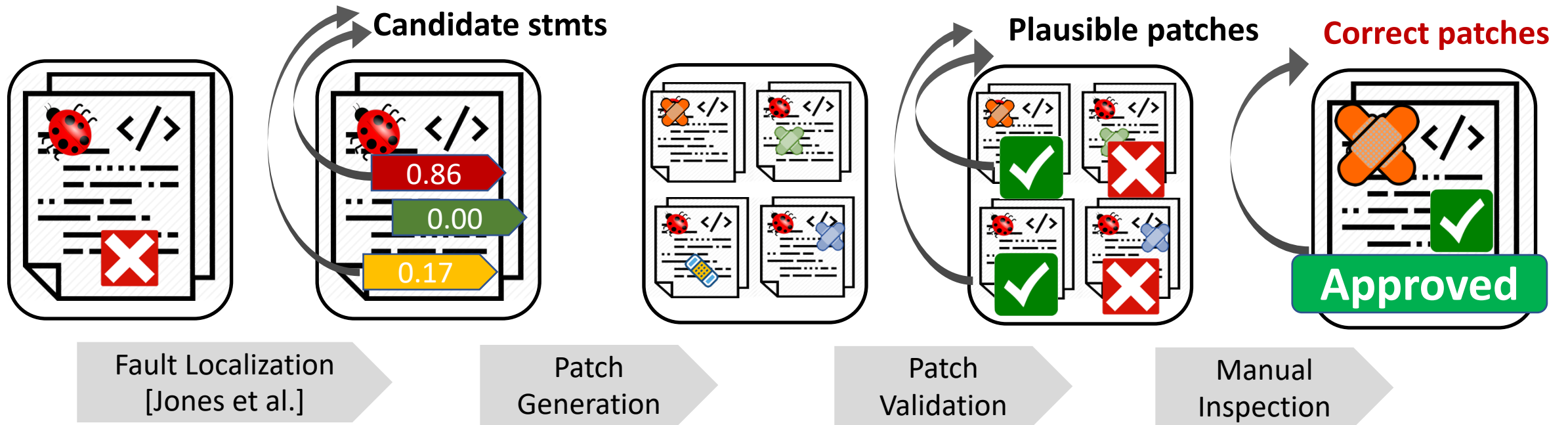


Which page(s) caused Microsoft Word to crash?

Fault localization



Program repair



- Search-based program repair
 - Transforms the repair problem to a search space exploration problem
- Semantics-based program repair
 - Leverages symbolic execution and constraint solving

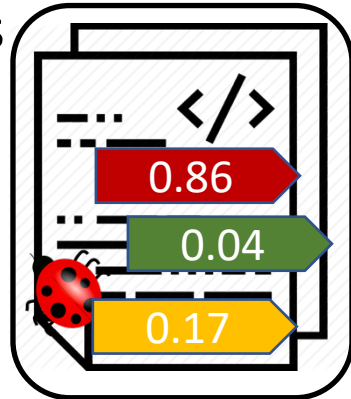
How to perform faster repair?

Unified debugging

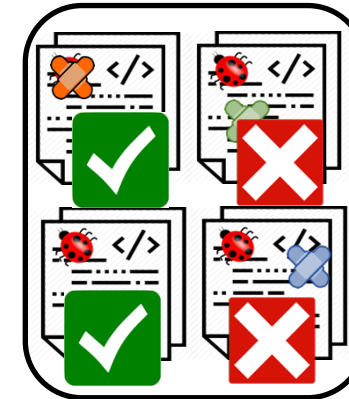
Fault Localization:

👎 Limited effectiveness for manual repair in practice [Parnin et al.]

👍 Largely refined fault localization for manual repair



NOW: Repair for Fault Localization!



Program Repair:

👎 Fixing <20% real-world bugs [Ghanbari et al.]

👍 Making **automated repair** applicable to all bugs!

PAST DECADE: Fault localization for repair

Testing and debugging for more

- Test the tests!
 - Flaky tests
- Machine learning
 - DNN models/libs
- Formal methods
 - SMT solvers
- Database
 - Modern DB engines
- ...

Any question for me?

Thanks and stay safe!