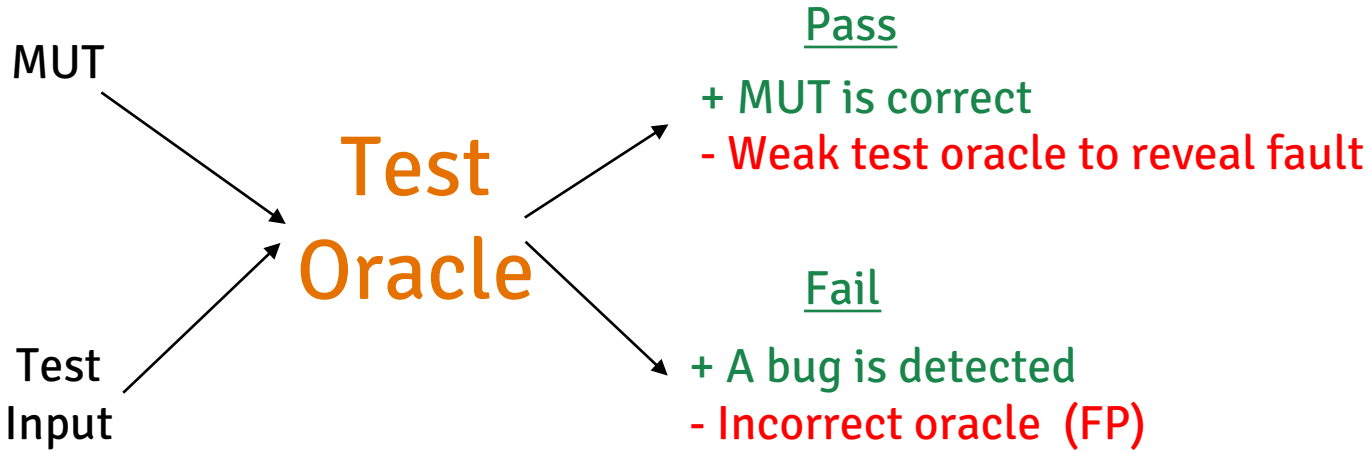
Neural-Based Test Oracle Generation: A Large-Scale Evaluation and Lessons Learned

Soneya Binta Hossain (UVA)
Antonio Filieri (AWS)
Matthew Dwyer (UVA)
Sebastian Elbaum (UVA)
Willem Visser (AWS)



Automated test oracle generation, while advancing, still faces significant challenges

EvoSuite

- An automated unit test generation method for Java
- Produces test inputs to achieve high code coverage
- Suggests assertion or exception oracles based on observed behavior

Test Prefix + Assertion Oracle

Test Prefix

```
public void test00() throws Throwable  {
    Stack<Integer> s0 = new Stack<Integer>();
    Integer int0 = new Integer(0);
    s0.push(int0);
    assertEquals(1, s0.size());
}
```

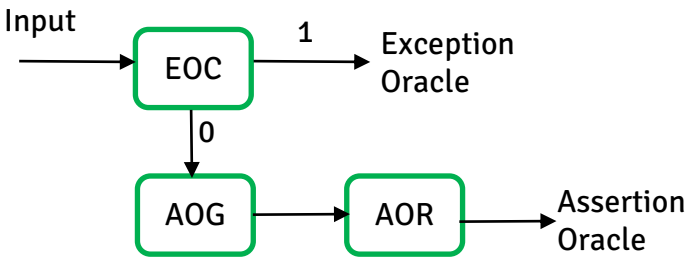
Test Prefix + Exception Oracle

```
public void test11() throws Throwable  {
   Stack<Integer> s0 = new Stack<Integer>();
    try {
      s0.pop();
      fail();
     } catch(EmptyStackException e) {
        verifyException("Stack", e);
    }
}
```

Learning-based Method

- Learns from large-scale training data
- Understands both code and natural language document
- Detects bugs in the current program version

TOGA [1], a state-of-the-art method for test oracle generation



1. *Elizabeth Dinella et al. (2022). TOGA: Neural Method for Test Oracle Generation. ICSE '22, ACM, pp. 2130–2141. https://doi.org/10.1145/3510003.3510141*

Example of Learning-based Oracles

Test Prefix

```
public void test03() throws Throwable {
  Stack<Object> s0 = new Stack<Object>();
  boolean b0 = s0.isEmpty();
}
```

Test Prefix With Assertion Oracle

```
public void test03() throws Throwable {
  Stack<Object> s0 = new Stack<Object>();
  boolean b0 = s0.isEmpty();
  assertTrue(b0)
}
```

Test Prefix

```
public void test05() throws Throwable {
  Stack<Object> s0 = new Stack<Object>();
  s0.peek();
}
```

Test Prefix With Exception Oracle

```
public void test05() throws Throwable {
  Stack<Object> s0 = new Stack<Object>();
    try {
      s0.peek();
      fail();
    } catch (Exception e){
      verifyException("Stack", e);
    }
}
```

Overview

- Validating prior results
  - revealed several issues with the original study setup
- Investigating precision
  - revealed a very high false positive rates
- Investigating bug detection effectiveness
  - revealed limited bug detection effectiveness

Validating Prior Results, Findings and Lesson

TOGA Defects4J Study

Original Study:

- Generated test cases on fixed programs
- Considered bug reaching tests (tests that fail on the buggy version)
- Generated oracles for the bug reaching prefixes
- A bug is detected if a test passed fixed version and failed on the buggy version
- Detected 57 bugs, outperforming other methods (Randoop, seq2seq, JDoctor, AthenaTest)

Our Findings:

Confirmed original results

Most bugs (67%) were detected by implicit oracles when executing EvoSuite test prefixes

TOGA Defects4J Study

Original Study:

- Generated test cases on fixed programs
- Considered bug-reaching tests (tests that fail on

- Detected 57 bugs, outperforming other methods (Randoop, seq2seq, JDoctor, AthenaTest)

Our Findings:

Implicit oracles should be used as a baseline to report fault-detection improvement

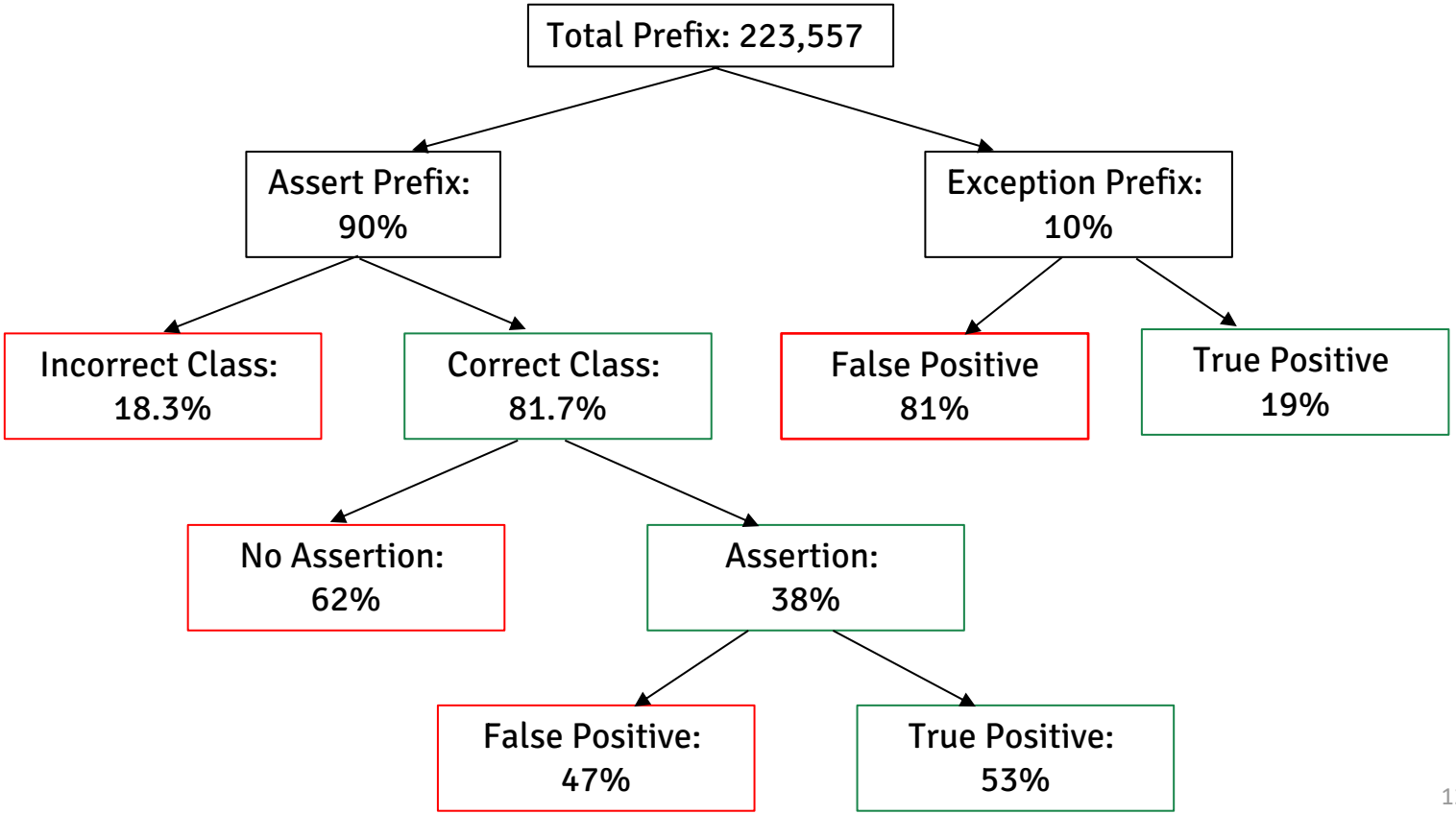
Precision Study, Findings and Lesson

Precision of Learning-based Method

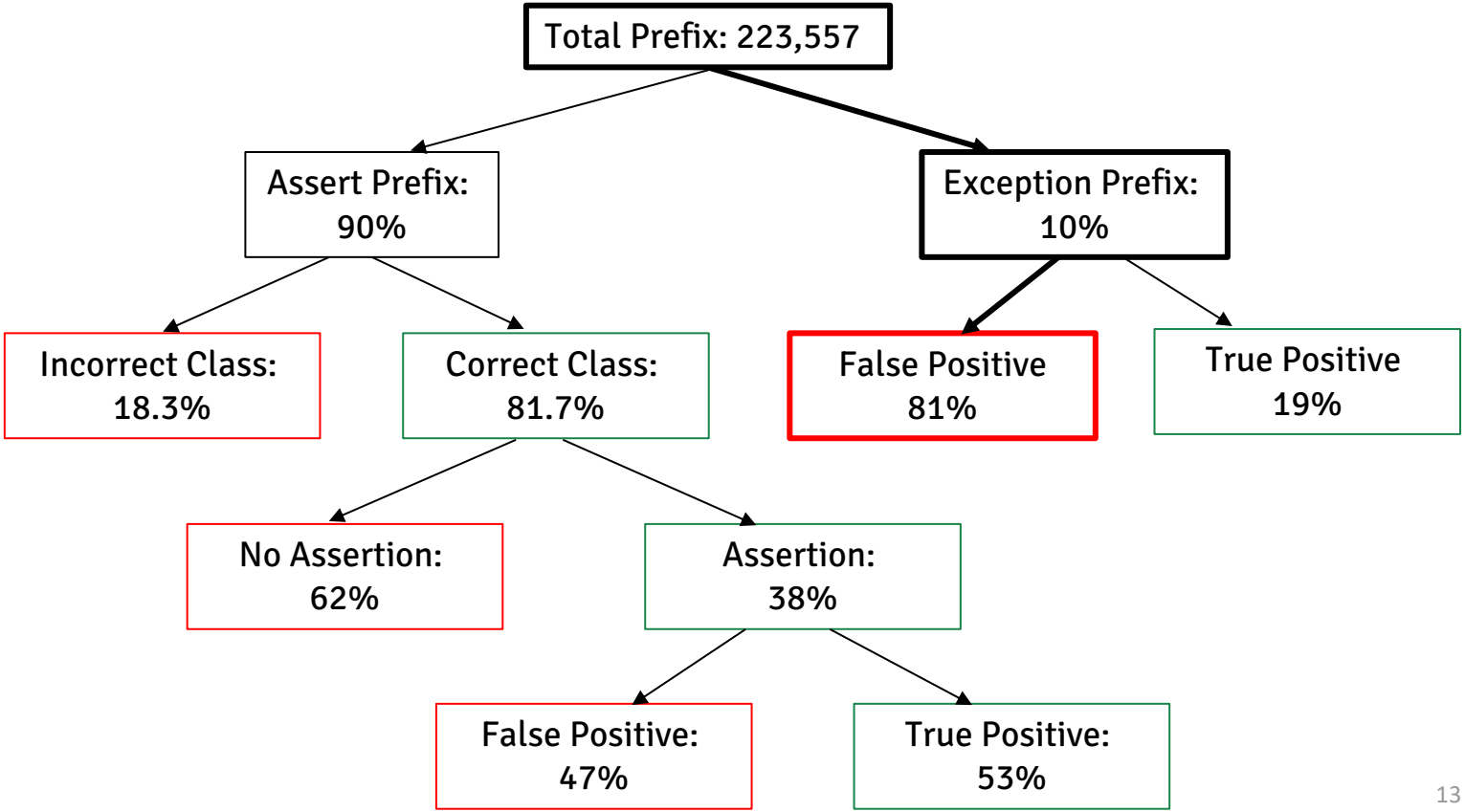
Study Setup:

- Prepared a large-scale dataset from 25 Java applications, consisting of 223.5K test cases
- Generated ground truth oracles using EvoSuite
- Prepared inputs for TOGA to generate oracle
- Ran the integrated tests for validation

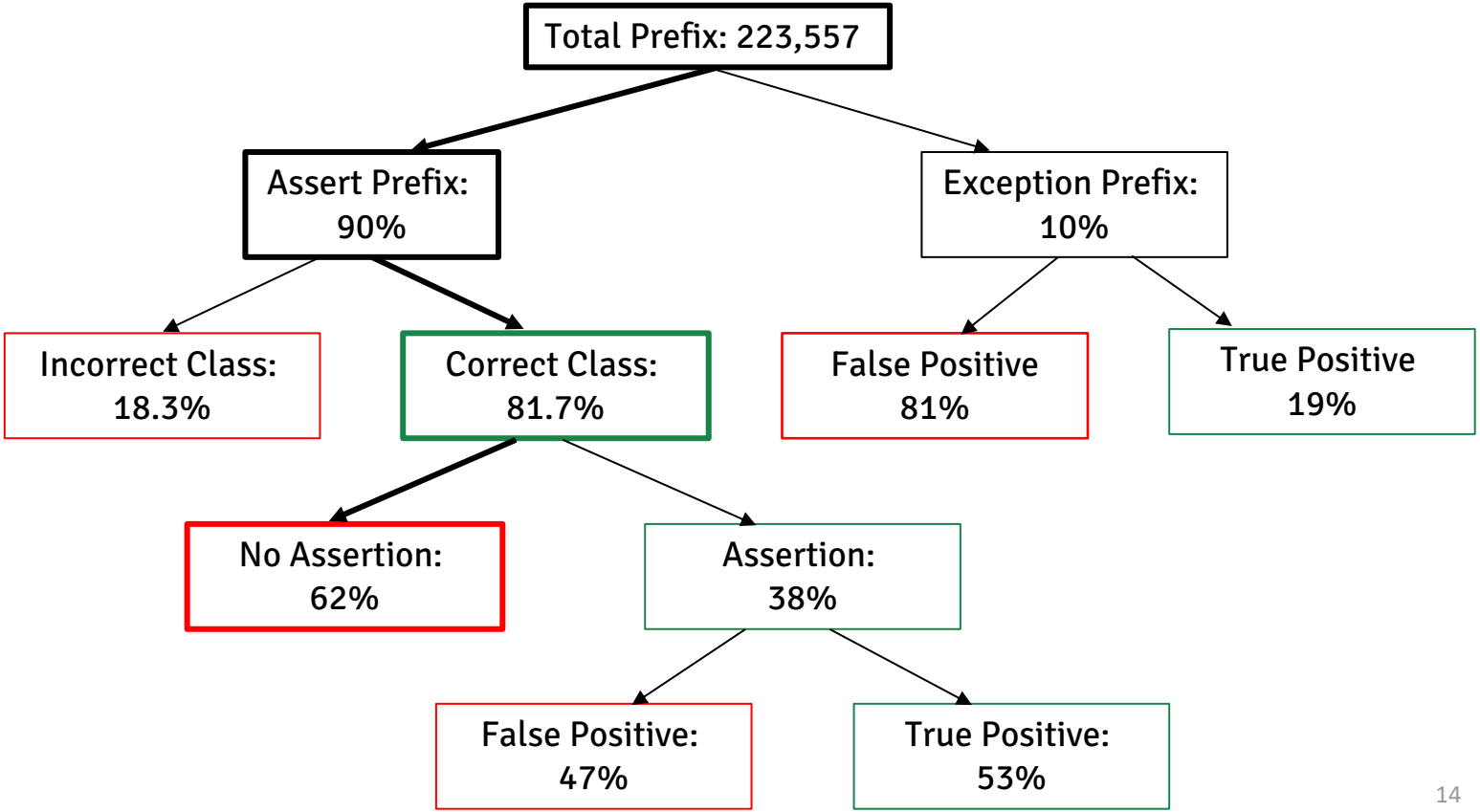
Our Findings



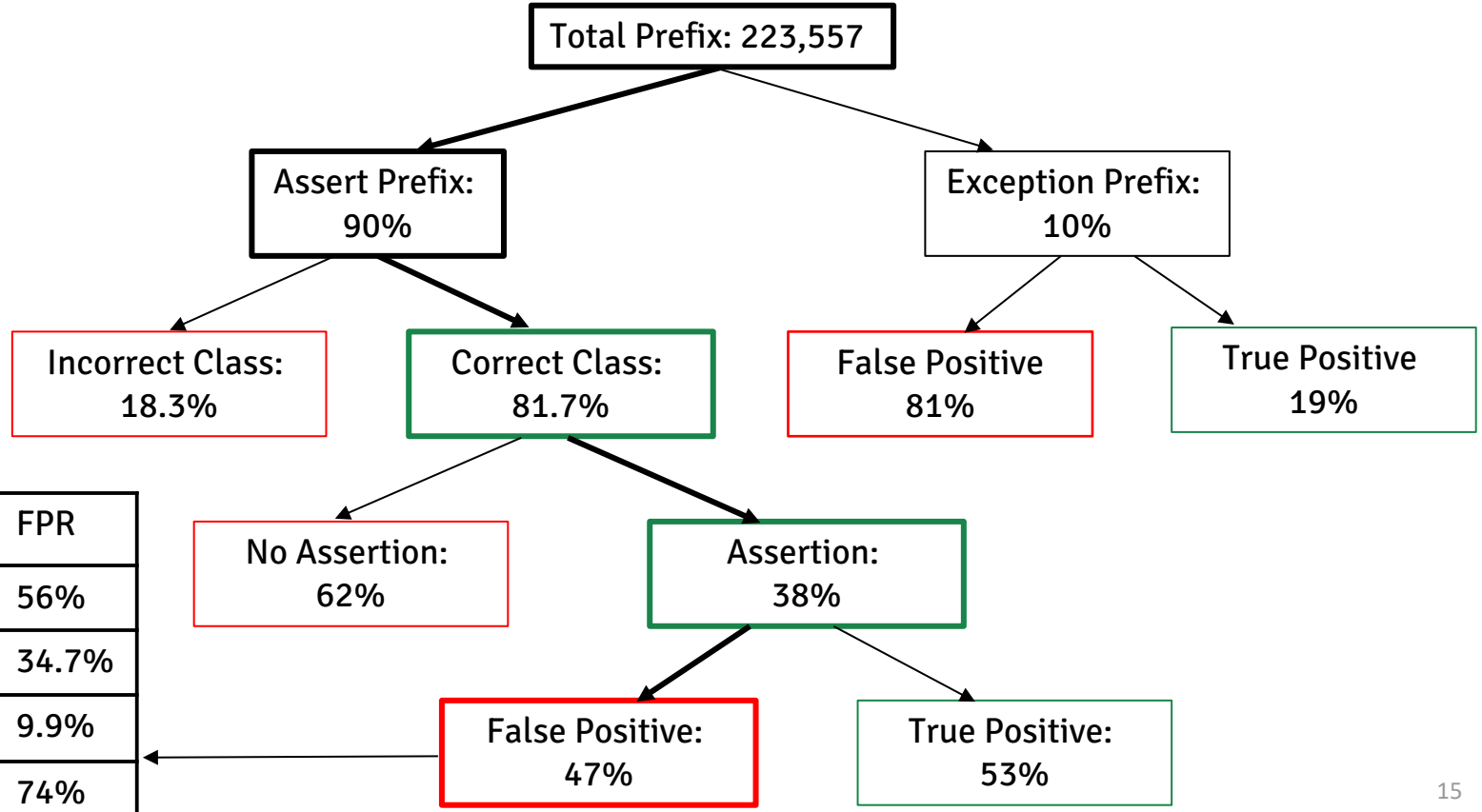
Our Findings



Our Findings

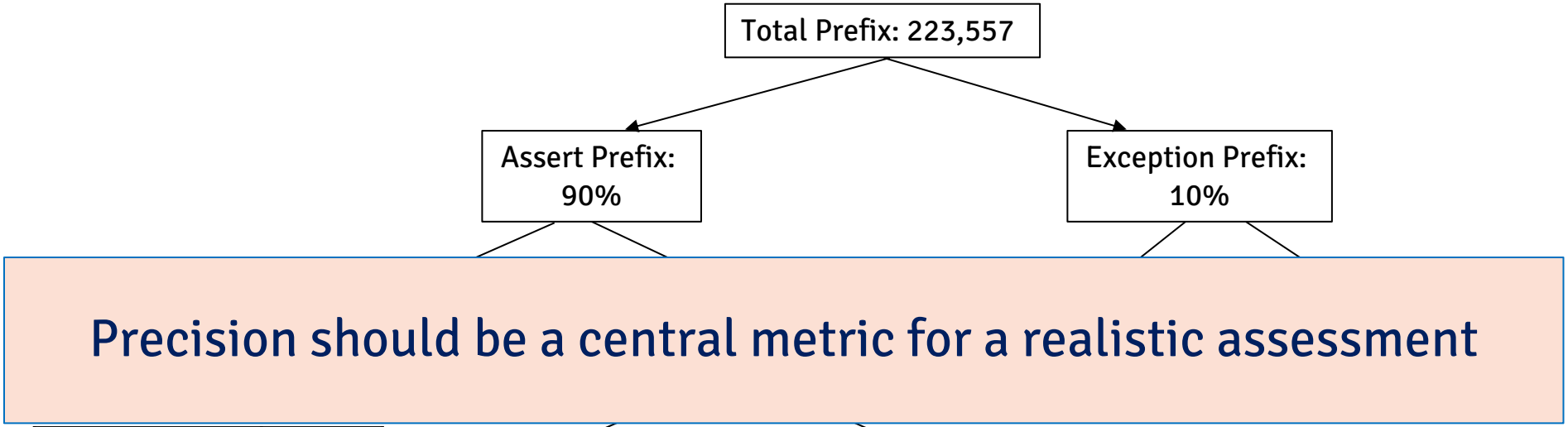


Our Findings



Assertion Type	FPR
assertTrue	56%
assertFalse	34.7%
assertNotNull	9.9%
assertEquals	74%

Our Findings



Assertion Type	FPR
assertTrue	56%
assertFalse	34.7%
assertNotNull	9.9%
assertEquals	74%

Bug Detection Study, Findings and Lesson

Bug Detection Effectiveness of Learning-based Method

Study Setup:

- Considered only true positive assertions

- Prepared three test suites with identical prefixes but with different type of assertions: implicit assertions, EvoSuite assertions, learning-based assertions

- Generated 51K mutated programs and ran different test suites to detect them

- Compared and analyzed relative bug detection effectiveness

Findings

Total Tests	Total Mutants	Mutant Detected by				
		Implicit Oracle (#)	EvoSuite Assertion (#)	EvoSuite Unique (#)	TOGA Assertion (#)	TOGA Unique(#)
34,378	51,385	20,597 (40%)	9,814 (19%)	3,026 (5.9%)	6,893 (13.4%)	105 (0.2%)

Findings

Total Tests	Total Mutants	Mutant Detected by				
		Implicit Oracle (#)	EvoSuite Assertion (#)	EvoSuite Unique (#)	TOGA Assertion (#)	TOGA Unique(#)
34,378	51,385	20,597 (40%)	9,814 (19%)	3,026 (5.9%)	6,893 (13.4%)	105 (0.2%)

Findings

Total Tests	Total Mutants	Mutant Detected by				
		Implicit Oracle (#)	EvoSuite Assertion (#)	EvoSuite Unique (#)	TOGA Assertion (#)	TOGA Unique(#)
34,378	51,385	20,597 (40%)	9,814 (19%)	3,026 (5.9%)	6,893 (13.4%)	105 (0.2%)

To avoid bias, a more realistic evaluation should use mutation testing

In Summary …

Finding - 1: 67% of the Defects4J bugs can be detected by implicit oracles
Lesson - 1: Implicit oracles should be used as the baseline

Finding – 2: SOTA learning-based method has a very high false positives rate
Lesson – 2: Precision should be a central evaluation metric for a realistic assessment

Finding - 3: SOTA learning-based method has limited unique bug detection capability
Lesson - 3: To avoid bias, a more realistic evaluation should use mutation testing

Neural-Based Test Oracle Generation: A Large-Scale Evaluation and Lessons Learned

Artifact: https://doi.org/10.6084/m9.figshare.21973091.v4

Acknowledgement: AWS and DARPA ARCOS FA8750-20-C-0507, Air Force Office of Scientific Research FA9550-21-0164, and Lockheed Martin Advanced Technology Laboratories