Perfect Is the Enemy of Test Oracle

Ali Reza Ibrahimzada[1], Yigit Varli[2], Dilara Tekinoglu[3], Reyhaneh Jabbarvand[1]

[1]University of Illinois Urbana-Champaign, [2]Middle East Technical University, [3]University of Massachusetts Amherst




Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)

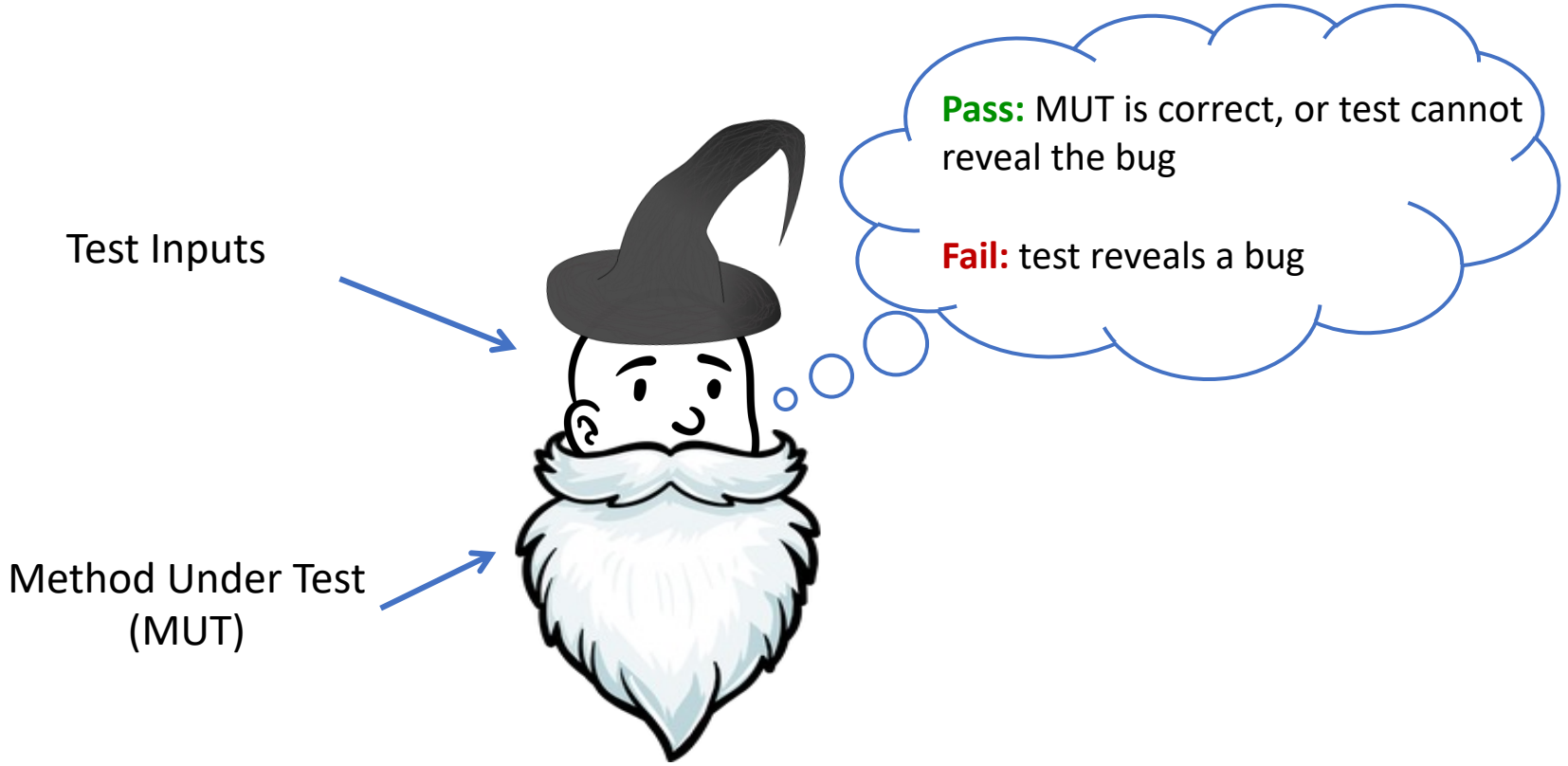
Singapore, November 2022

INTELLIGENT Code Analysis Testing

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

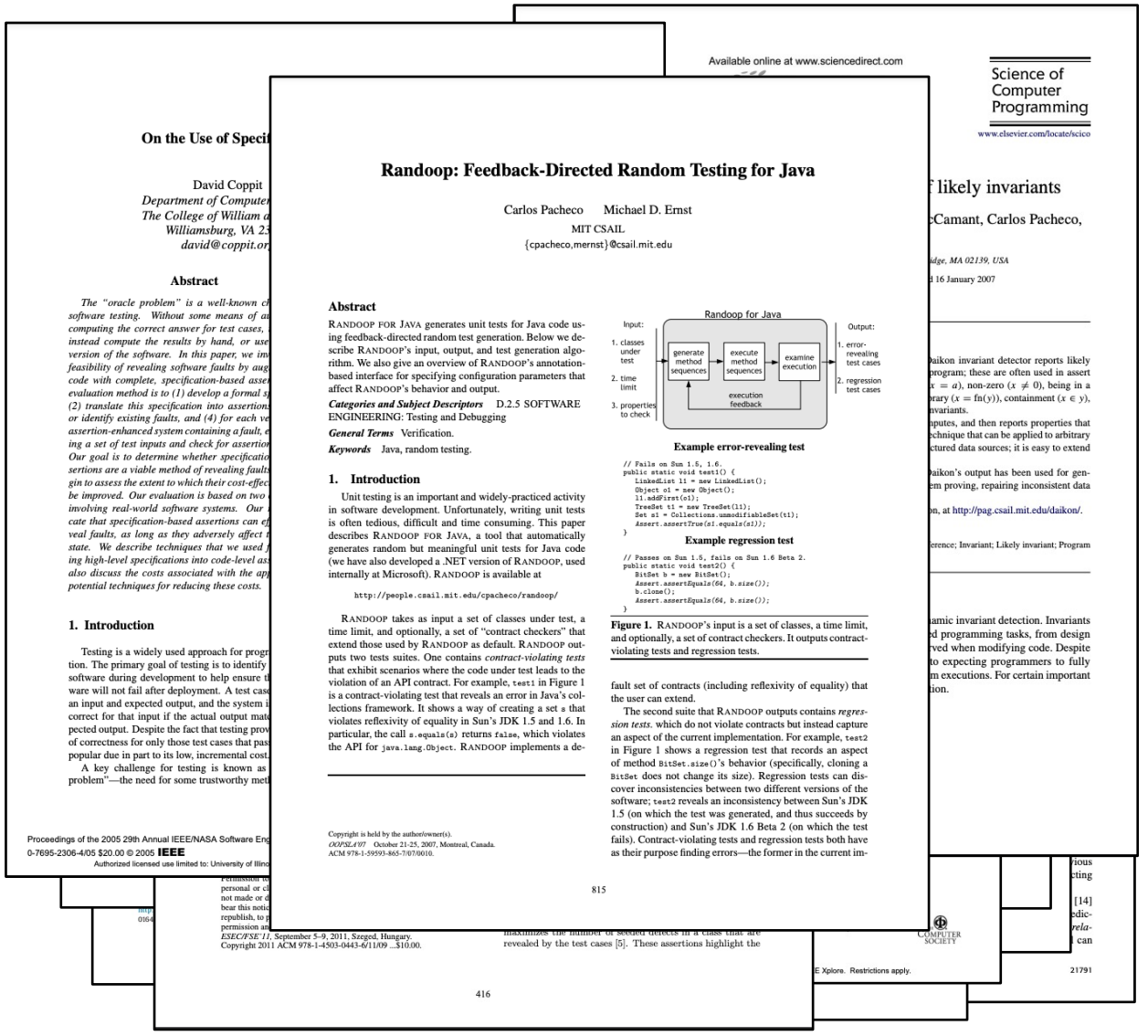
What is a Test Oracle?

Does execution of a test reveal a bug in the code?



Automated Test Oracle Construction

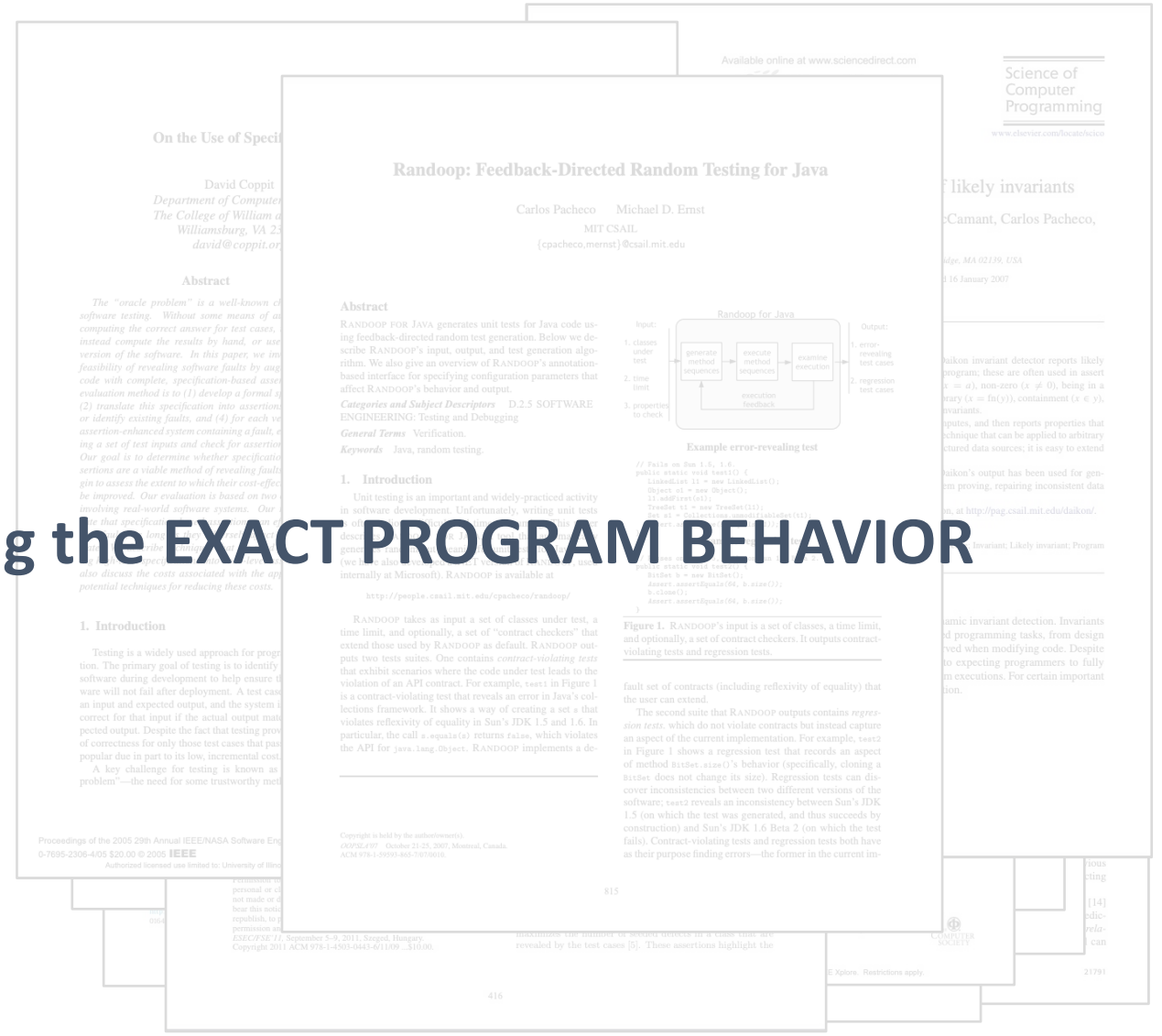
- Formal Specifications
- Assertions
- Program Invariants
- Metamorphic Relations



Automated Test Oracle Construction

- Formal Specifications
- Assertions
- Program Invariants
- Metamorphic Relations

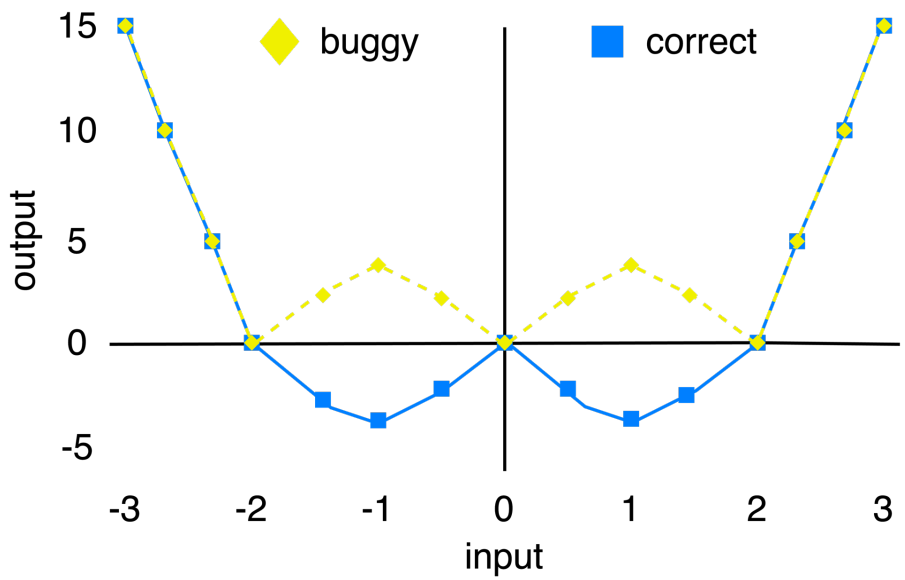
Commonality -> determining the EXACT PROGRAM BEHAVIOR



Research Gap

```java
// Correct implementation f(x) = |x| * (x + 2) * (x - 2)
public double example(double x) {
    double output;
    if (x >= 0)
        output = x * (x + 2) * (x - 2);
    else
        output = Math.abs(x) * (x + 2) * (x - 2);
    return output;
}
```

```java
// Buggy implementation f(x) = |x * (x + 2) * (x - 2)|
public double example_buggy(double x) {
    double output;
    output = Math.abs(x * (x + 2) * (x - 2));
    return output;
}
```




```java
@Test
public void test1() {
    double o1 = example_buggy(0.5);
    Assert.assertTrue("msg", o1 >= 0);
}
```

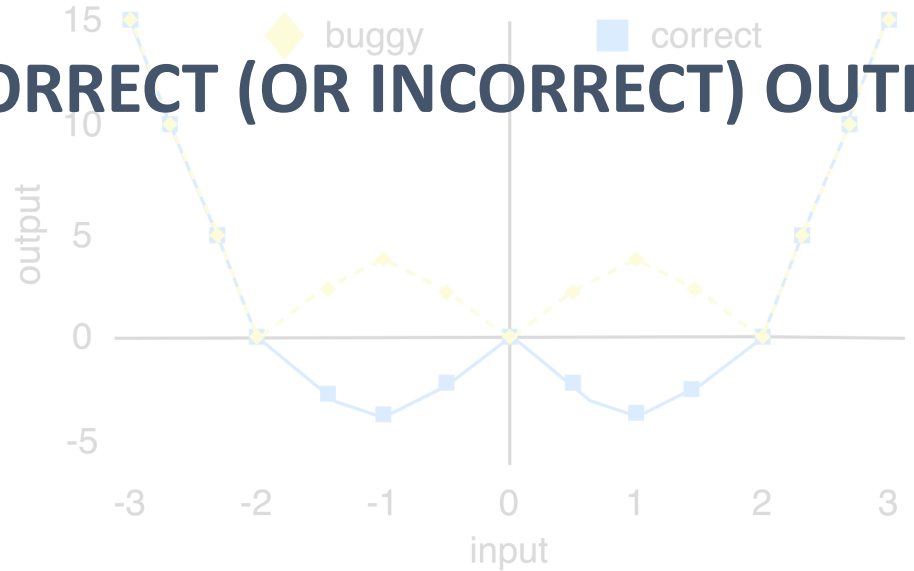
```java
@Test
public void test2() {
    double o2 = example_buggy(-1);
    double o3 = example_buggy(1);
    Assert.assertEquals(o2, o3);
}
```

Research Gap

```
// Correct implementation f(x) = |x| * (x + 2) * (x - 2)
public double example(double x) {
    double output;
    if (x >= 0)
        output = x * (x + 2) * (x - 2);
    else
        output = Math.abs(x) * (x + 2) * (x - 2);
    return output;
}
```

```
// Buggy implementation f(x) = |x * (x + 2) * (x - 2)|
public double example_buggy(double x) {
    double output;
    output = Math.abs(x * (x + 2) * (x - 2));
    return output;
}
```

NEED TO KNOW THE CORRECT (OR INCORRECT) OUTPUT FOR GIVEN INPUTS




```
@Test
public void test1() {
    double o1 = example_buggy(0.5);
    Assert.assertTrue("msg", o1 >= 0);
}
```

```
@Test
public void test2() {
    double o2 = example_buggy(-1);
    double o3 = example_buggy(1);
    Assert.assertEquals(o2, o3);
}
```

Distinguishing Between Correct and Buggy MUTS

$$f(x) = |x| * (x + 2) * (x - 2)$$

```
@Test Expects o1 = -1.875
public void test1() {
    double o1 = example_buggy(0.5);
    Assert.assertTrue("msg", o1 >= 0);
}
```

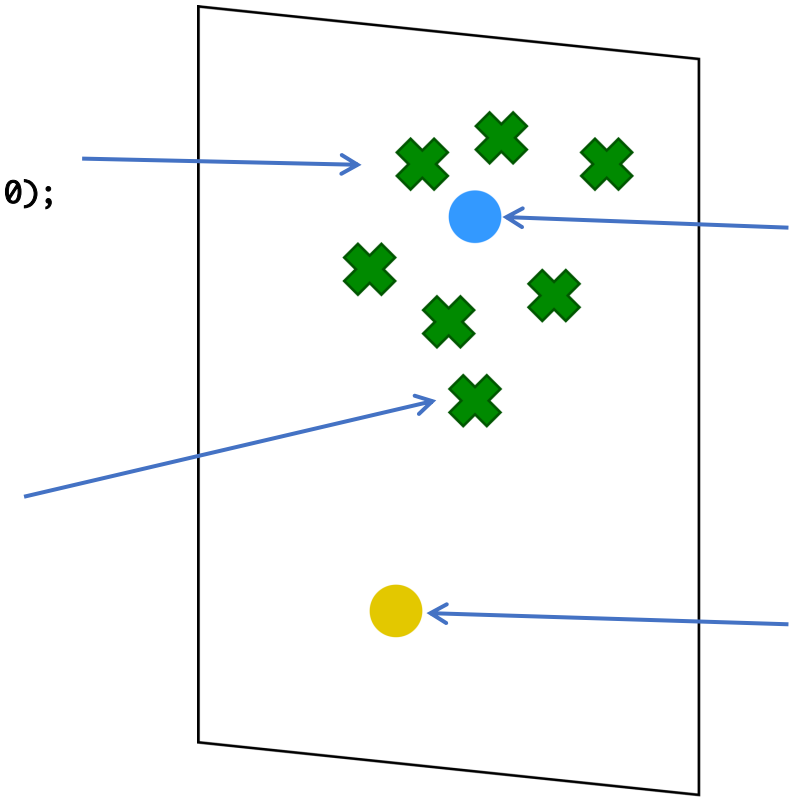
Passes on correct code

Fails on buggy code

```
@Test Expects o2 = o3
public void test2() {
    double o2 = example_buggy(-1);
    double o3 = example_buggy(1);
    Assert.assertEquals(o2, o3);
}
```

Passes on correct code

Passes on buggy code



```
// Correct implementation f(x) = |x| * (x + 2) * (x - 2)
public double example(double x) {
    double output;
    if (x >= 0)
        output = x * (x + 2) * (x - 2);
    else
        output = Math.abs(x) * (x + 2) * (x - 2);
    return output;
}
```

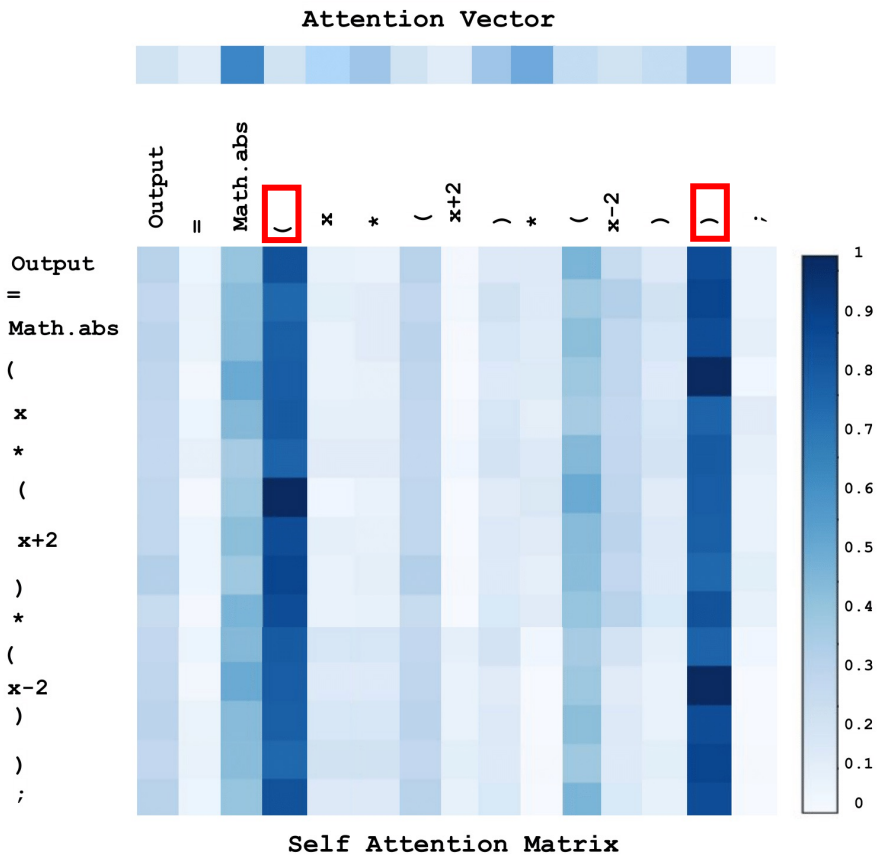
```
// Buggy implementation f(x) = |x * (x + 2) * (x - 2)|
public double example_buggy(double x) {
    double output;
    output = Math.abs(x * (x + 2) * (x - 2));
    return output;
}
```

Model Interpretation

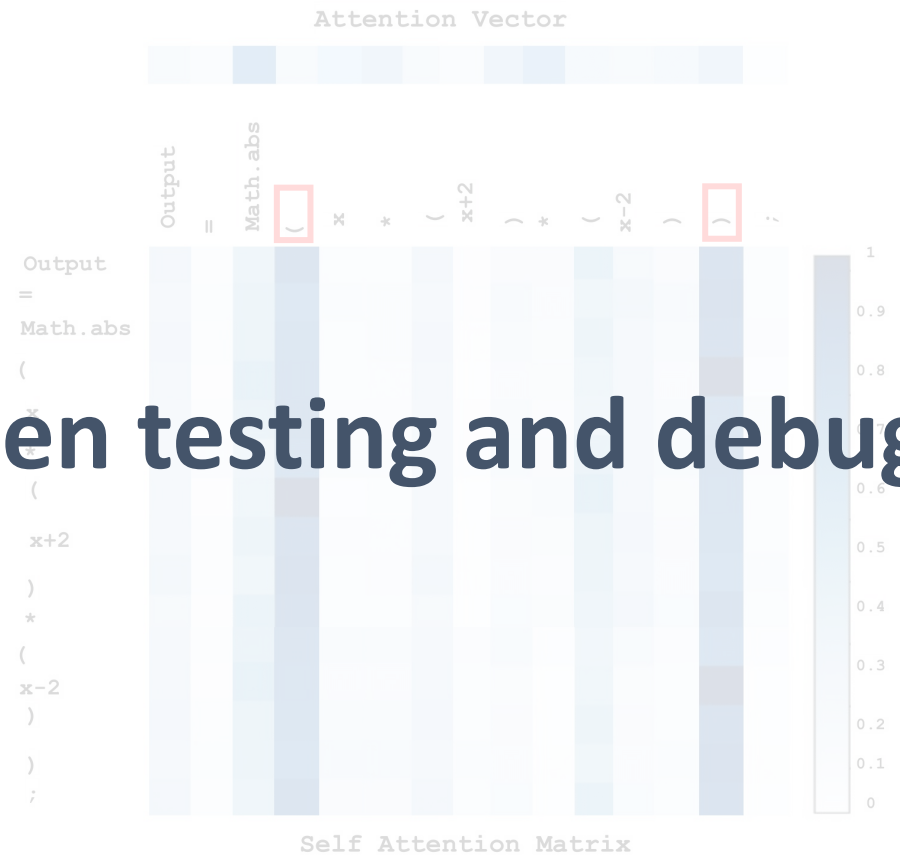
SEER is beyond solving the **ever-challenging** test oracle problem

Model Interpretation (Cont'd)

```java
// Buggy implementation f(x) = |x * (x + 2) * (x - 2)|
public double example_buggy(double x) {
    double output;
    output = Math.abs(x * (x + 2) * (x - 2));
    return output;
}
```



Model Interpretation (Cont'd)

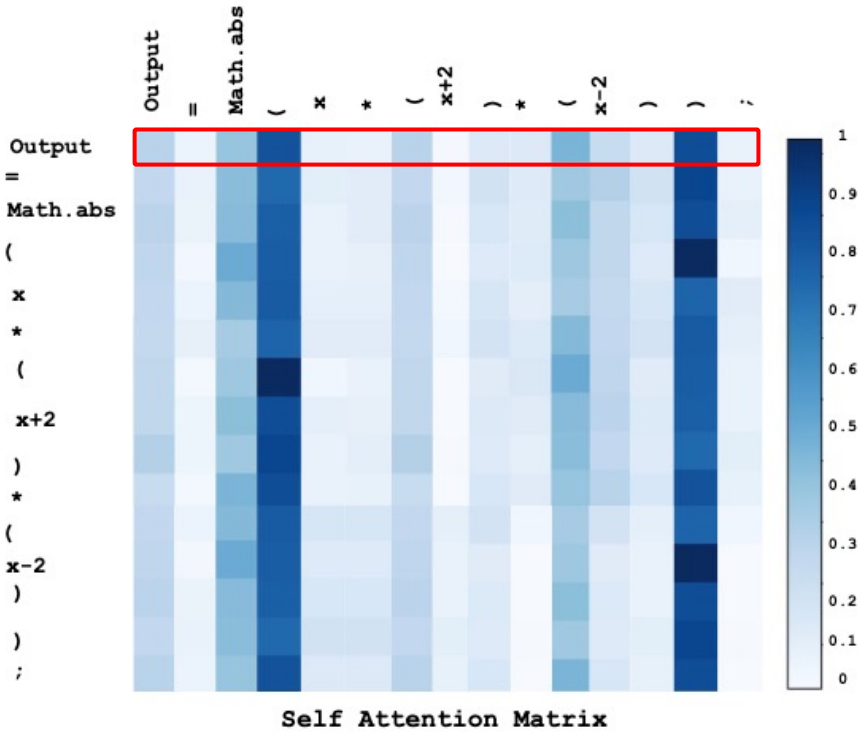


SEER bridges the gap between testing and debugging

Attention Analysis

Input: MUT's tokens $Tkn = \{c_0, \ldots, c_n\}$
Input: MUT's statements $Smt = \{s_0, \ldots, s_m\}$
Input: $SA = [[w0_0, \ldots, w0_n], \ldots, [wn_0, \ldots, wn_n]]$
Input: Attention threshold k
Output: Attended tokens $ATkn$, Attended statements $ASmt$

1 $ATkn \leftarrow \emptyset$
2 $ASmt \leftarrow \emptyset$
3 **foreach** row = $[wi_0, \ldots, wi_n] \in SA$ **do**
  // $localATkn = \{\langle c_i, ind_i \rangle \mid ind_i$ `is index of` c_i in $Tkn\}$
4 $localATkn \leftarrow getMostAttended(row, Tkn, k)$
5 **foreach** $\langle c_i, ind_i \rangle \in localATkn$ **do**
6 **if** $\neg ATkn.contains(\langle c_i, ind_i \rangle)$ **then**
7 $ATkn \leftarrow ATkn \cup c_i$
8 **foreach** $s_i \in Smt$ **do**
9 **if** $| s_i \cap ATkn | > k$ **then**
10 $ASmt \leftarrow ASmt \cup s_i$




```
// Buggy implementation f(x) = |x * (x + 2) * (x - 2)|
public double example_buggy(double x) {
    double output;
    output = Math.abs(x * (x + 2) * (x - 2));
    return output;
}
```

Research Questions

- ❖ Effectiveness in predicting test labels
- ❖ Generalization to unseen projects
- ❖ Interpretation analysis
- ❖ Performance

Experimental Setup

SEER's dataset is built on top of Defects4J + Augmentations

Projects	# Bugs	# Mutants	#Tests #Pass, #Fail (Contribution%)	Dataset							
				Defects4J				Higher-Order Mutants			
				Developer Tests		Randoop Tests		Developer Tests		Randoop Tests	
				# Pass	# Fail	# Pass	# Fail	# Pass	# Fail	# Pass	# Fail
Compress	47	24,322	30,753 (30.26%)	1,214	22	2,965	2,230	385	265	192	23,480
Lang	64	12,824	15,130 (14.89%)	702	59	1,249	296	397	509	1,124	10,794
Chart	26	4,313	14,901 (14.66%)	248	37	5,316	4,987	154	109	229	3,821
Math	106	9,488	13,580 (13.36%)	1,685	91	1,584	732	732	814	546	7,396
Codec	18	8,973	10,210 (10.05%)	271	11	648	307	375	54	93	8,451
Closure	174	1,615	3,170 (3.12%)	1,431	8	113	3	768	76	49	722
JacksonDatabind	112	581	2,818 (2.77%)	2,133	27	77	0	410	94	42	35
Time	26	203	2,415 (2.38%)	1,765	39	255	153	84	70	18	31
Jsoup	93	677	2,134 (2.1%)	925	57	458	17	125	234	277	41
Cli	39	913	1,612 (1.59%)	348	10	292	49	106	75	371	361
Csv	16	1,166	1,582 (1.56%)	327	6	83	0	297	2	23	844
JacksonCore	26	941	1,420 (1.4%)	331	15	105	28	320	163	136	322
Gson	18	531	1,386 (1.36%)	678	15	93	69	295	92	144	0
JxPath	22	143	314 (0.31%)	93	1	74	3	0	1	105	37
Mockito	38	0	100 (0.1%)	56	39	5	0	0	0	0	0
JacksonXml	6	49	81 (0.08%)	19	1	9	3	28	12	2	7
Collections	4	0	7 (0.01%)	4	1	2	0	0	0	0	0
Total	835	66,739	101,613 (100%)	12,230	439	13,328	8,877	4,476	2,570	3,351	56,342

Can SEER Effectively Predict Test Labels?

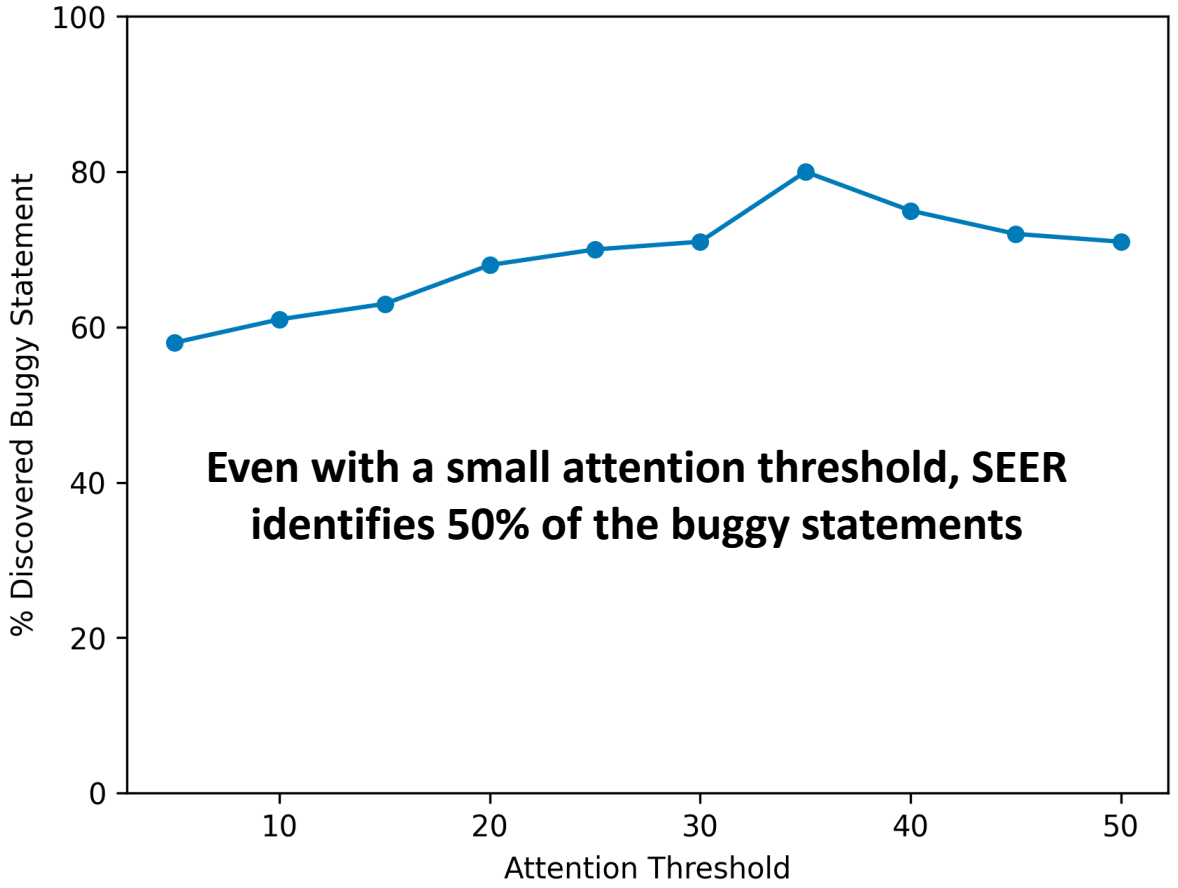
SEER's Evaluation

Accuracy	Precision	Recall	F1 Score
93%	86%	94%	90%

The longer the MUTs and test, the better the performance of SEER

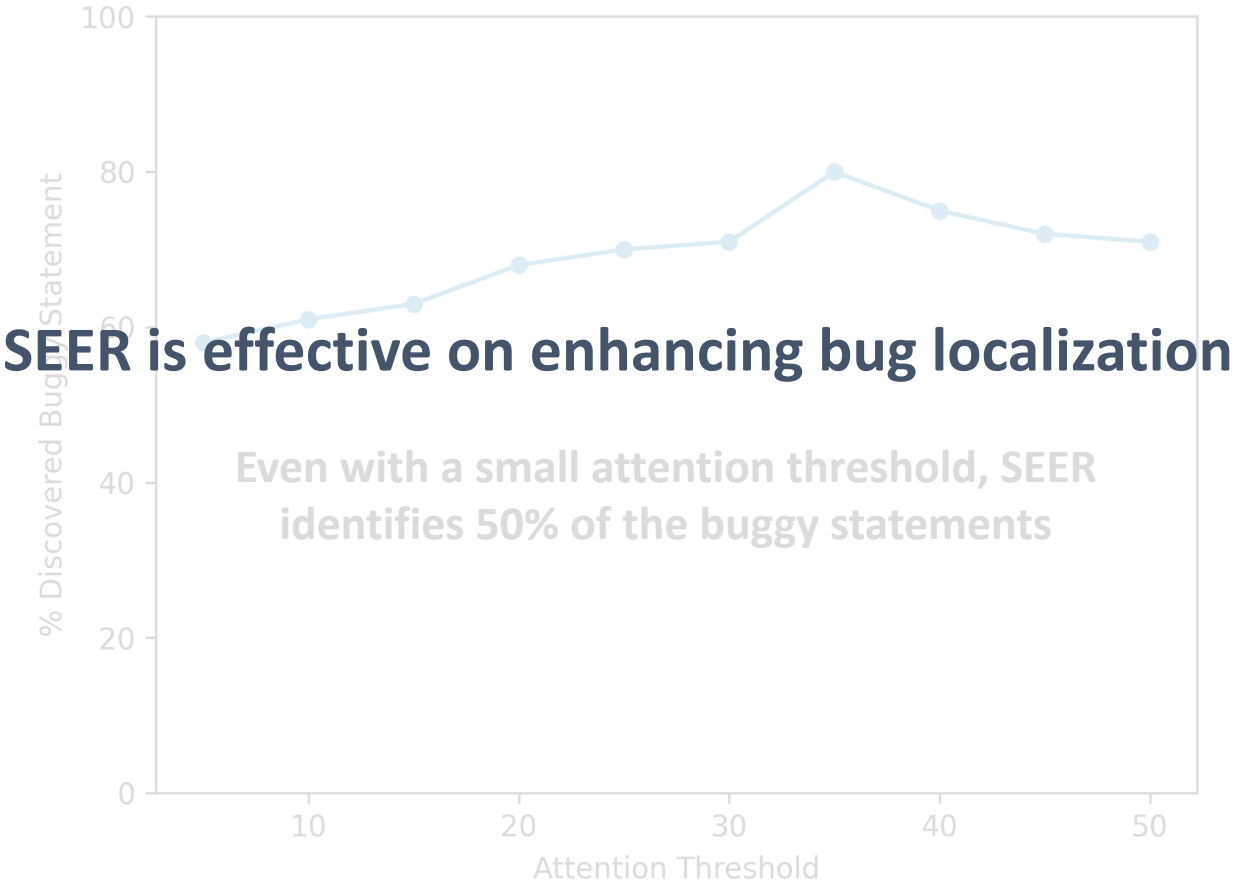
Can SEER find buggy statements?

$$\%\ Discoverd\ Buggy\ Statements = \frac{Attended\ Buggy\ Statements}{Buggy\ Statements}\ x\ 100$$



Can SEER find buggy statements?

$$\% \, Discoverd \, Buggy \, Statements = \frac{Attended \, Buggy \, Statements}{Buggy \, Statements} \, x \, 100$$

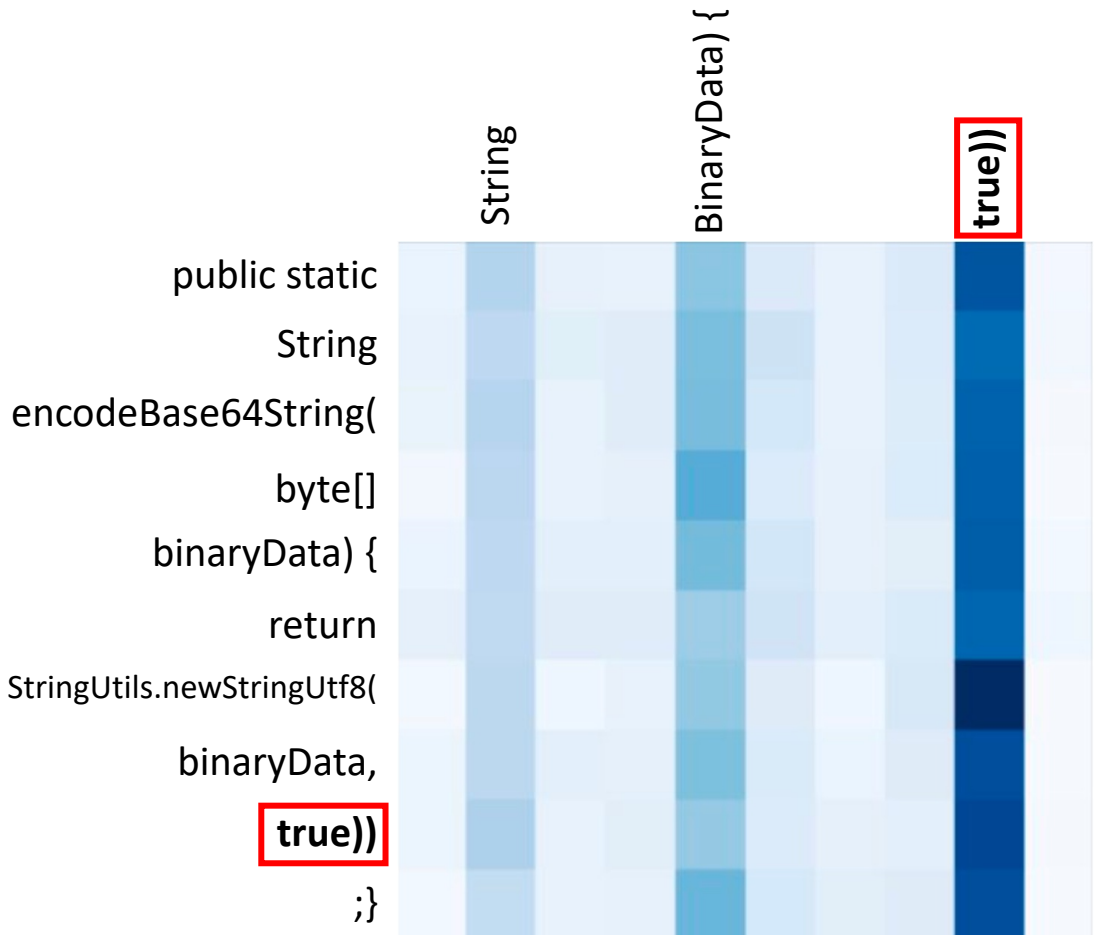


SEER is effective on enhancing bug localization

Even with a small attention threshold, SEER identifies 50% of the buggy statements

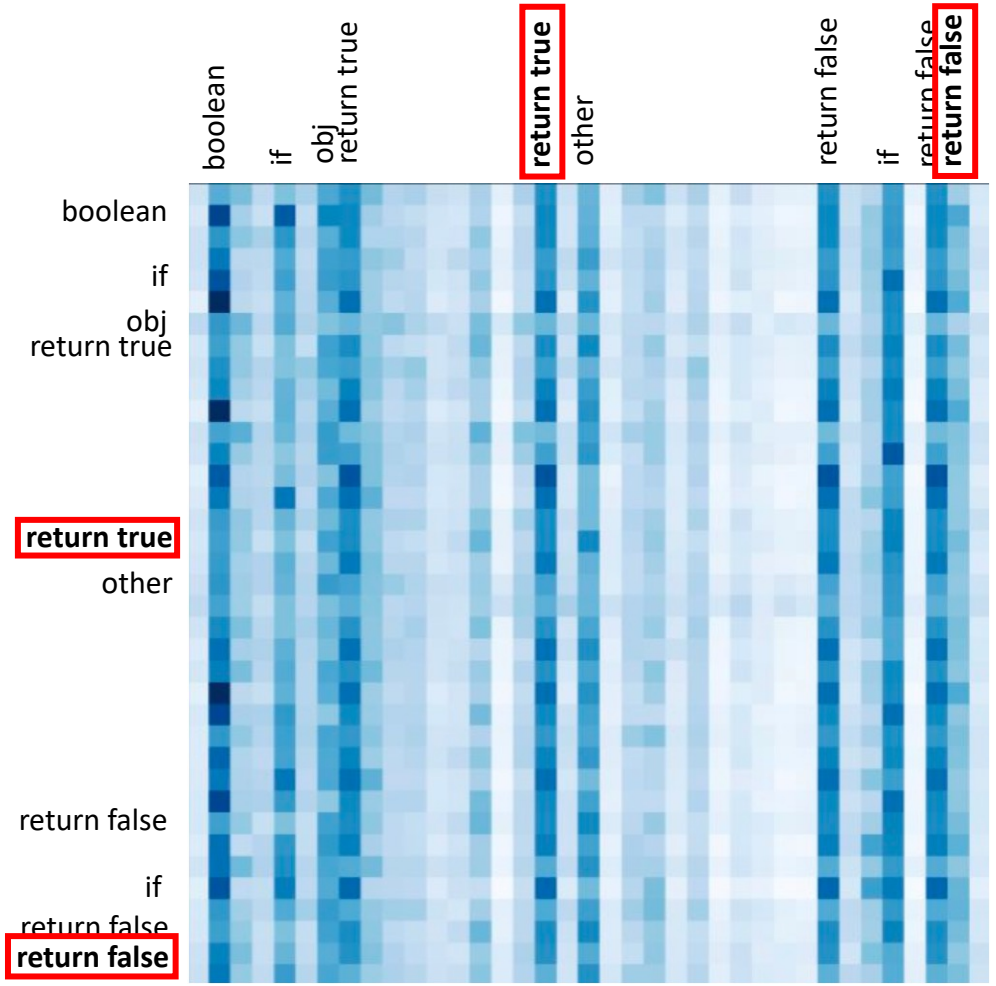
Can SEER find buggy tokens?

```
public static String encodeBase64String(
    byte[] binaryData) {return StringUtils.newStringUtf8(
                          encodeBase64(binaryData, true));
}
```



Can SEER find buggy tokens? (Cont'd)

```
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null || getClass() != obj.getClass())
        return true;
    ZipArchiveEntry other = (ZipArchiveEntry) obj;
    if (name == null) {
        if (other.name != null)
            return false;
    } else if (!name.equals(other.name))
        return false;
    return false;
}
```



Conclusions

- We propose SEER for automated test oracle construction
- SEER learns the correlation between inputs and outputs
- SEER model is interpretable and can enhance bug localization
- SEER is highly accurate and generalizable to unseen projects

https://github.com/Intelligent-CAT-Lab/SEER