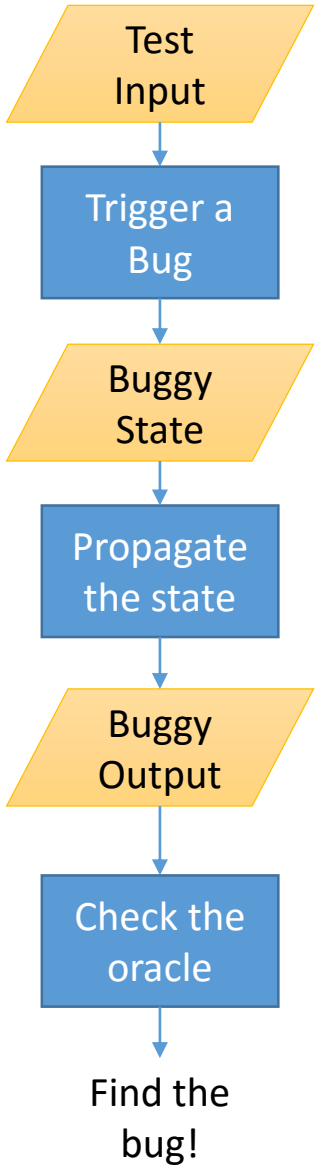
Inner Oracles: Input-Specific Assertions on Internal States

Yingfei Xiong, Dan Hao, Lu Zhang,
Tao Zhu, Muyao Zhu, Tian Lan

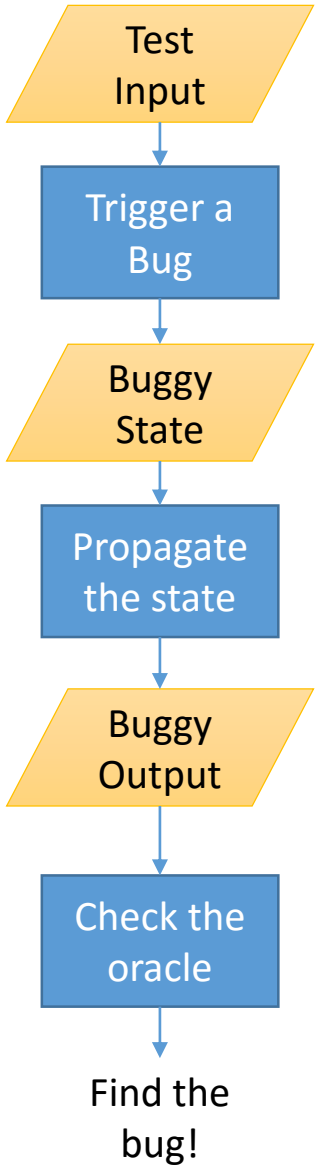
Peking University, China

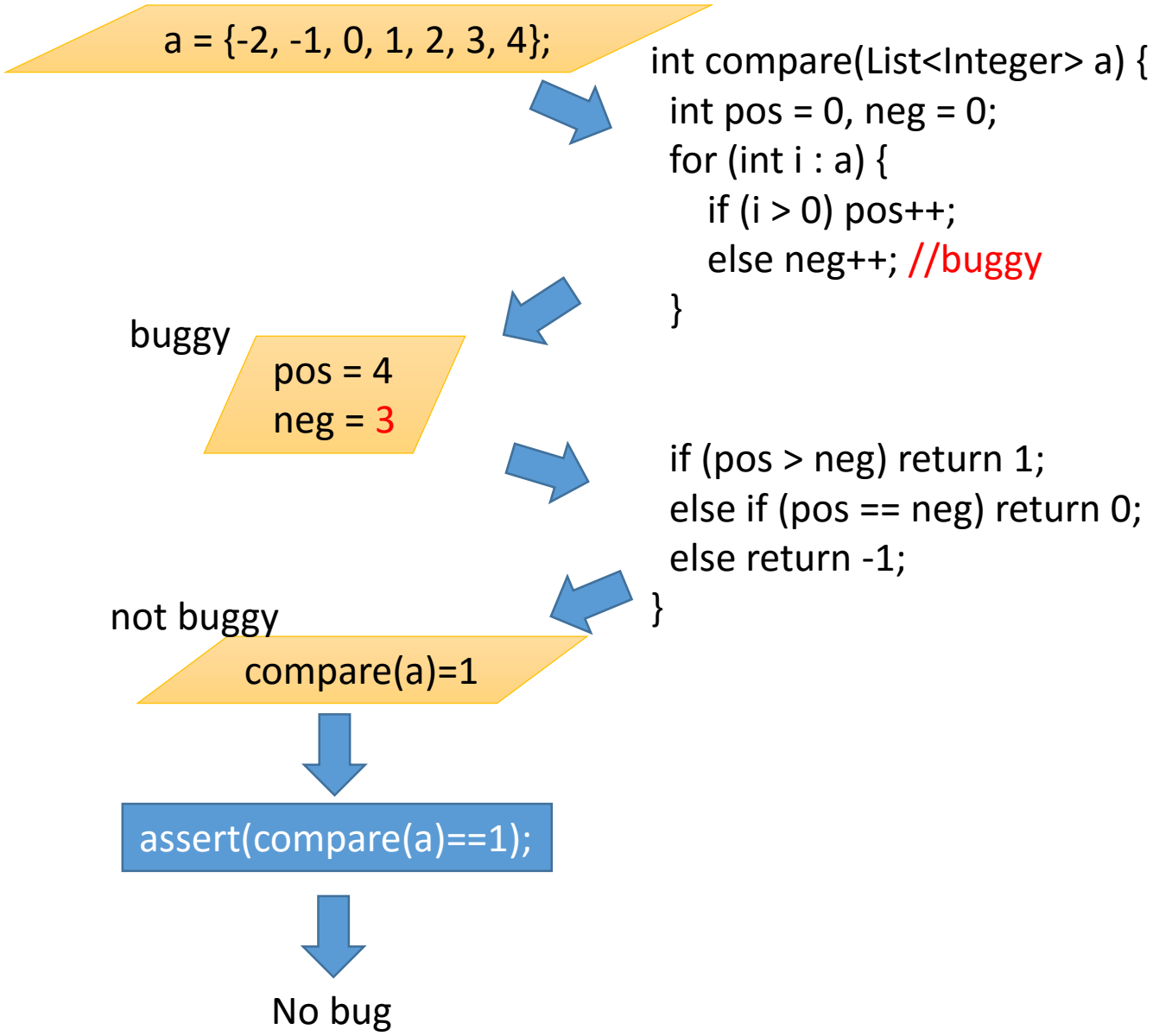
2015

How a Test Detects a Bug

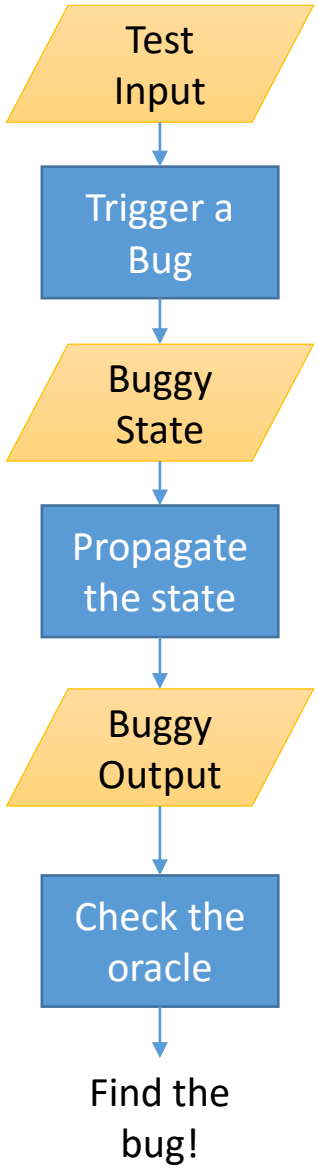


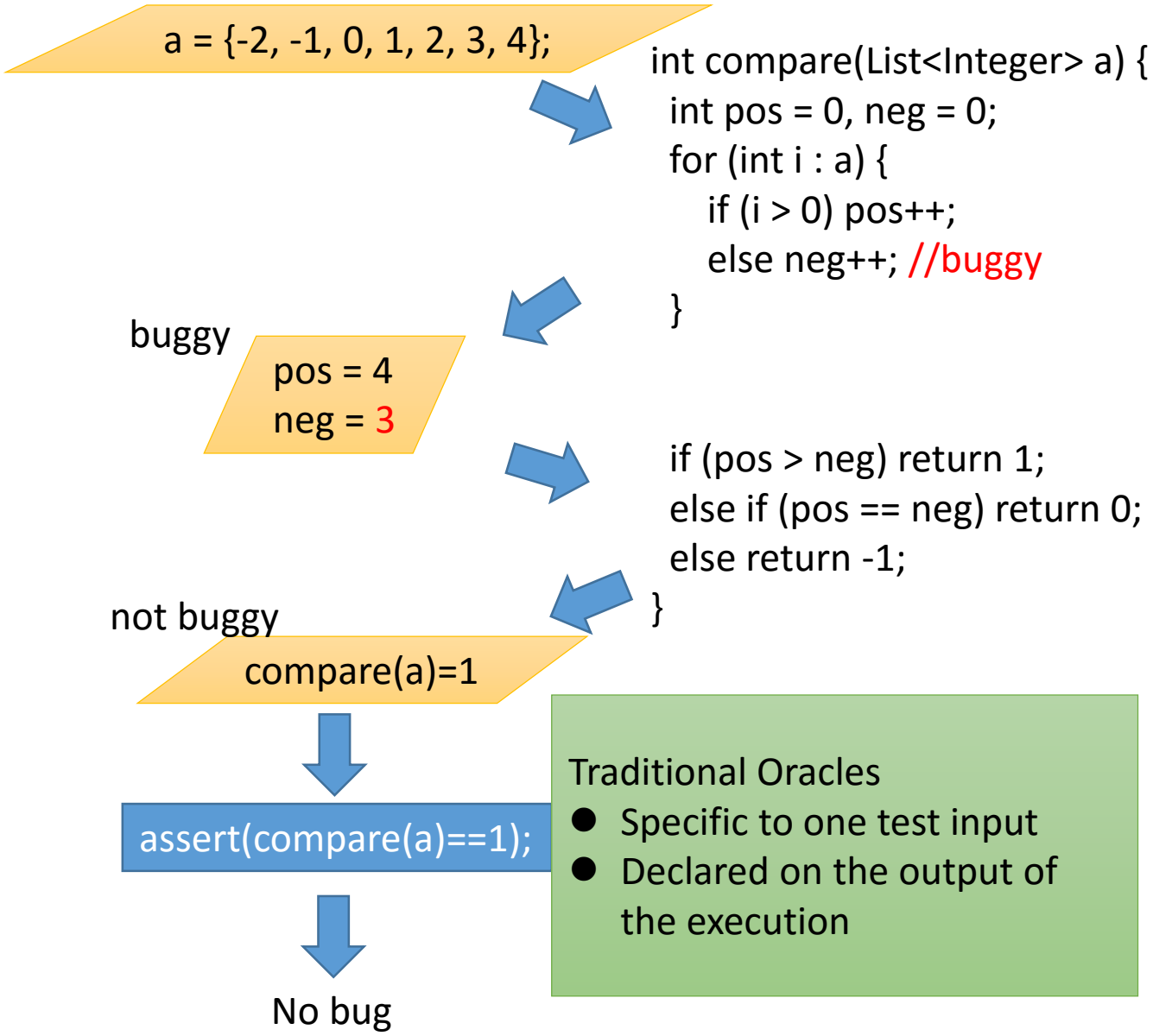
How a Test Not Detects a Bug





Traditional Oracles

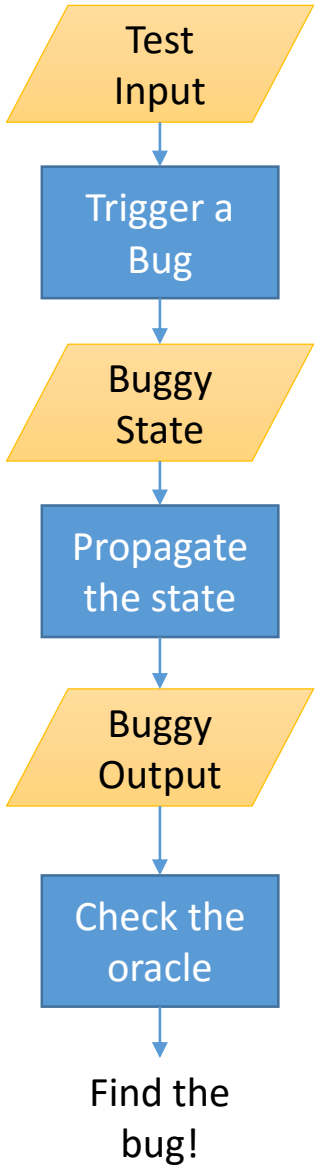




Traditional Oracles

- Specific to one test input
- Declared on the output of the execution

Assertions on Internal State



a = {-2, -1, 0, 1, 2, 3, 4};

```
int compare(List<Integer> a) {
  int pos = 0, neg = 0;
  for (int i : a) {
     if (i > 0) pos++;
     else neg++; //buggy
  }
```

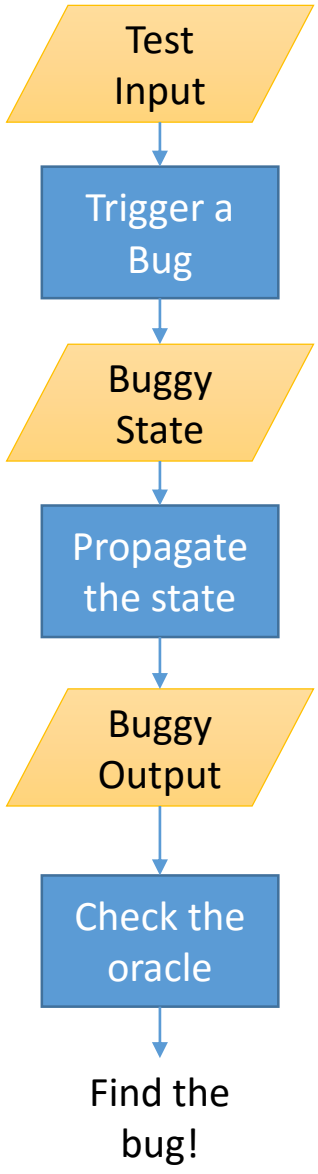
pos = 4
neg = 3

```
assert(neg ==
 /*number of negatives*/);

if (pos > neg) return 1;
== neg) return 0;
-1;
```

Standard Assertions
- on internal states
- common to all input
  - Not easy to write
  - Programmers may make the same mistake

Inner Oracles



a = {-2, -1, 0, 1, 2, 3, 4};

```
int compare(List<Integer> a) {
  int pos = 0, neg = 0;
  for (int i : a) {
     if (i > 0) pos++;
     else neg++; //buggy
  }
```

pos = 4
neg = 3

```
assert_for_this_test(
  neg == 2);

if (pos > neg) return 1;
else if (pos == neg) return 0;
else return -1;
}
```

Inner Oracles
- Declared on internal states
- Specific to a test input

How to write inner oracles

```
class CountTest {
  public static boolean guard = false;
  @Test
  public void test1() {
    List<Integer> a = {-2, -1, 0, 1, 2, 3, 4};
    guard = true;
    compare(a);
    guard = false;
  }
}
```

```
int compare(List<Integer> a) {
  int pos = 0, neg = 0;
  for (int i : a) {
     if (i > 0) pos++;
     else neg++; //buggy
  }
  assert (!CountTest.guard ||
    neg == 2);
  if (pos > neg) return 1;
  else if (pos == neg) return 0;
  else return -1;
}
```

Inner oracles can also be written by weaving, similar to AOP.
Check at: http://ayzk.github.io/InnerTest/

How much can we gain with inner oracles? – Enhancing tests

Subject	KLOC	#Method	#Class	#Test	#Mutant	Pairs
jodatime	25.8	3276	198	3417	26	88842
timeandmoney	1.1	262	30	104	39	4056
barbecue	8.0	283	55	51	50	2550
xmlsec	16.2	1213	181	97	22	2134

In 30.72%-69.65% pairs, fault cannot be captured by traditional oracles on output, but only by inner oracles.

- The buggy state is not propagated into a buggy output (294/1369)
- The buggy part in the output state cannot be accessed by a test, e.g., a private member (1075/1369)

How much can we gain with inner oracles? – Reducing test suites

Subject	*jodatime*	*timeandmoney*	*barbecue*	*xmlsec*
#With traditional oracles	10	9	4	7
#With inner oracles	8	5	2	6

Test suites are further reduced by 14.3%-50.0% with inner oracles.

Applications and Implications --- Testing Optimization

```
int times(int a, int b) {
  if (b % 2 == 0) {
    while (b >>= 1)
      return a << 1;
  }
  else
    return a * b;
}
```

How do we know the first branch is executed when b is 8?

Applications and Implications --- Testing Optimization

```
int times(int a, int b) {
  if (b % 2 == 0) {
    while (b >>= 1)
      return a << 1;
  }
  else {
    assert(!test1);
    return a * b;
}}
```

```
test1 = true;
times(2, 8);
test1 = false;
```

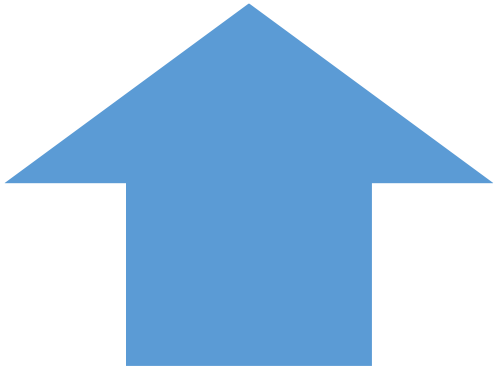
Applications and Implications --- Debugging

Traditional Oracles

- Any executed statements may be buggy

Inner Oracles

- Only the statements executed before the inner oracle may be buggy



Applications and Implications --- Regression Test Generation

```
doStuff(X x, int n, int m) {
  Y y = x.doSth(n);
  Z z = y.doSthElse(m);
  z.field = n+m;
  return;
}
```

```
@Test
…
doStuff(x, 1, 2);
assert(z.filed == 3);
```

- How do we know which object z is?
- How do we access this object?

[Xie et al., ECOOP06], [Taneja et al., ASE08]

Applications and Implications --- Regression Test Generation

```
doStuff(X x, int n, int m) {
  Y y = x.doSth(n);
  Z z = y.doSthElse(m);
  z.field = n+m;
  assert(!test1 || z.field==3);
  return;
}
```

```
@Test
…
test1 = true;
doStuff(x, 1, 2);
test1 = false;
```

Application and Implication --- Invariant Discovery

- Tools like Daikon discovers invariants (oracles on internal states for all inputs)
- Sometimes very few invariants can be discovered if we use too many inputs
- Let Daikon discover inner oracles for some inputs instead

```
test1 = true;
doStuff(1);
test1 = false;
```

```
test1 = true;
doStuff(2);
test1 = false;
```

```
test1 = true;
doStuff(3);
test1 = false;
```

Summary

- Inner Oracles
  - declared on internal states
  - specific to one test input
- Has a lot of applications/implications
- Ignored in existing literatures
- Worth putting more weights on