Take nothing on its looks; take everything on evidence.
There's no better rule.

– Charles Dickens, "Great Expectations."

Nopol:
Repairing Bugs in Conditional Expressions

Favio DeMarco

Universidad de Buenos Aires - INRIA

February 10, 2017

Bugs?

The difference between the right word and the almost right word is the difference between lightning and a lightning bug.

– Mark Twain

Motivation

The Six Stages of Debugging

1. That can’t happen.
2. That doesn’t happen on my machine.
3. That shouldn’t happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work?

What are conditional expression bugs?

```
boolean expression ? someValue : someOtherValue;

if (boolean expression) {
...
}
```

Change of If Condition Expression (IF-CC)

Kai Pan et al.[1]:

> *This bug fix change fixes the bug by changing the condition expression of an if condition. The previous code has a bug in the if condition logic.*

```
- if (getView().countSelected() == 0) {
+ if (getView().countSelected() <= 1) {
```

[1] Toward an understanding of bug fix patterns

What are conditional expression bugs?

Commons Math - MathUtils class

```
public static int gcd(int u, int v) {
  if (u * v == 0) {
    return (Math.abs(u) + Math.abs(v));
  }
...
```

What about `u=0x00110000` and `v=0x01100000`?

Problem I

How does the tool know something is *wrong*?

404

Problem I

How does the tool know something is *wrong*?

Some kind of specification:

- Model
- Contracts
- **Unit tests**
- ...

How does the tool know something is *wrong*?

At least one failing test

```
assertEquals(3 * (1<<15)
            , gcd(3 * (1<<20), 9 * (1<<15)));
```

No-Pol input

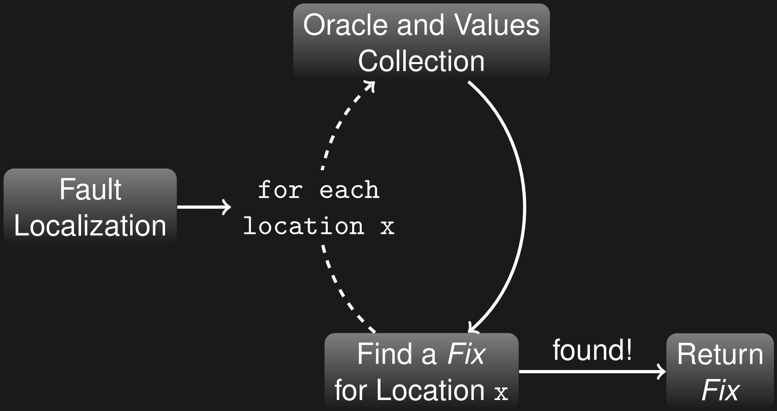
- Java source code.
- Unit tests with at least one failing test case.
- Dependencies (*classpath*).

No-Pol output

Patched Java source file.

Overview

Trial and error



Problem II

Where is the bug?

Fault Localization (statement ranking)

GZoltar - Ochiai coefficient

The suspiciousness s_j of a statement j depends on:

- The number of **failing** test cases **executing** statement j
- The number of **failing** test cases **not executing** statement j
- The number of **successful** tests **executing** statement j

Fault Localization (statement ranking)

GZoltar - Ochiai coefficient

```
MathUtils:413 - Suspiciousness: 0.23570226039551587
MathUtils:431 - Suspiciousness: 0.1543033499620919

...

MathUtils:460 - Suspiciousness: 0.11322770341445956
MathUtils:412 - Suspiciousness: 0.11180339887498948

...
```

Fault Localization (statement ranking)

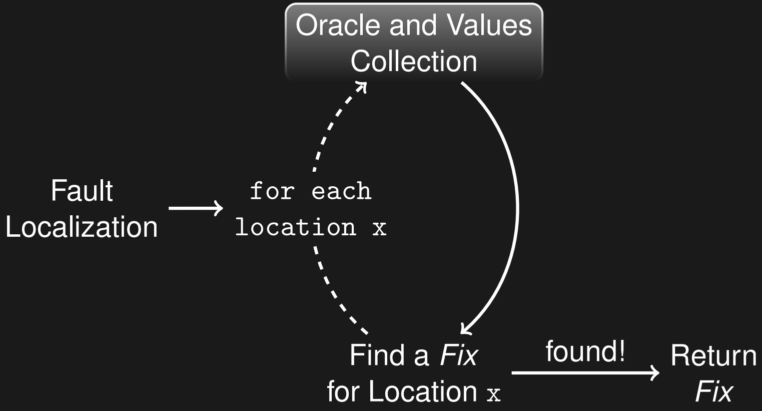
GZoltar - Ochiai coefficient

```
...
MathUtils:460 - Suspiciousness: 0.11322770341445956
MathUtils:412 - Suspiciousness: 0.11180339887498948
...
```

```
411: public static int gcd(int u, int v) {
412:   if (u * v == 0) {
413:     return (Math.abs(u) + Math.abs(v));
414:   }
```

Oracle and Values Collection

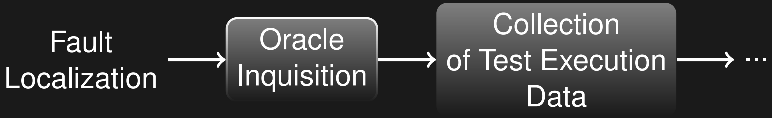
For Location `x`



Oracle and Values Collection

For Location `x`

Two steps:



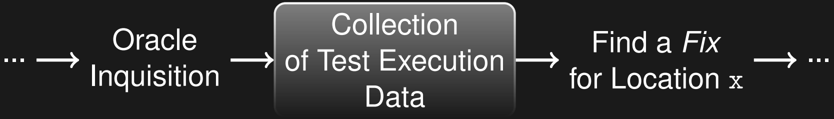
Oracle Inquisition

For Location `x`

$$\texttt{if(}\overbrace{\texttt{a < b}}^{\text{bug}}\texttt{)} \qquad \underbrace{\overset{\texttt{if(true)}}{PASS} \qquad \overset{\texttt{if(false)}}{KO}}_{\text{Oracle}}$$

Collection of Test Execution Data

For Location `x`


...
Oracle Inquisition
Collection of Test Execution Data
Find a *Fix* for Location x
...

Collection of Test Execution Data

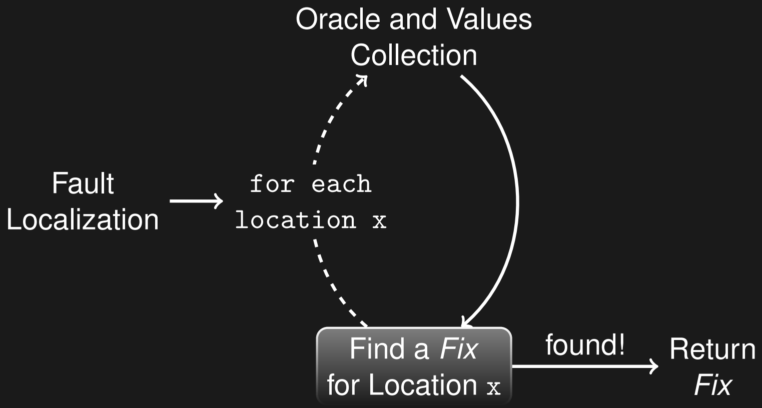
For Location x

```
 42: public static final double TWO_PI = 2 * FastMath.PI;
...
411: public static int gcd(int u, int v) {
412:   if (u * v == 0) {
413:     return (Math.abs(u) + Math.abs(v));
414:   }
```

	TWO_PI	u	v	
testZero	6.283185...	0	0	
testOverflow	6.283185...	0x00110000	0x01100000	

Find a *Fix*

For Location `x`



Constraints Generation (aka secret sauce)

For Location `x`

$$\begin{tikzcd}
\cdots \arrow[r] & \begin{array}{c}\text{Collection}\\ \text{of Test Execution}\\ \text{Data}\end{array} \arrow[r] & \boxed{\begin{array}{c}\text{Constraints}\\ \text{Generation}\end{array}} \arrow[r] & \begin{array}{c}\text{SMT}\\ \text{Solving}\end{array} \arrow[r] & \cdots
\end{tikzcd}$$

Constraints Generation

Oracle-Guided Component-Based Program Synthesis:

Components and line numbers:

$$\text{Constants}\begin{cases} 0 & \texttt{true} \\ 1 & \texttt{false} \\ 2 & \texttt{-1} \\ 3 & \texttt{0} \\ 4 & \texttt{1} \end{cases}$$

$$\text{Input values}\begin{cases} 5 & \texttt{u} & 8 & \texttt{col != null} \\ 6 & \texttt{v} & 9 & \texttt{col.isEmpty()} \\ 7 & \texttt{TWO\_PI} & 10 & \texttt{col.size()} \\ \ldots & & & \end{cases}$$

$$\text{Oracle}\begin{cases} l_O & \text{Expected output} \end{cases}$$

Constraints Generation

Oracle-Guided Component-Based Program Synthesis:

Components and line numbers:

$$
\begin{aligned}
l_{O1} &= l_{I11} < l_{I12} \\
l_{O2} &= l_{I21} <= l_{I22} \\
l_{O3} &= l_{I31} + l_{I32} \\
l_{O4} &= l_{I41} * l_{I42} \\
l_{O5} &= l_{I51} \wedge l_{I52} \\
&\ldots \\
l_{On} &= l_{In1} ? l_{In2} : l_{In3}
\end{aligned}
$$

Constraints Generation

Example

Components:

```
0   I
lO  oracle
lO1 := f1(lI1);
lO2 := f2(lI2_1, lI2_2);
```

An answer:

```
lI1 = 1    lO1 = 2
lI2_1 = 0  lO2 = 1
lI2_2 = 0  lO  = 2
```

Constraints Generation

Example

Components:

```
0   I
lO  oracle
lO1 := f1(lI1);
lO2 := f2(lI2_1, lI2_2);
```

An answer:

```
lI1 = 1    lO1 = 2
lI2_1 = 0  lO2 = 1
lI2_2 = 0  lO  = 2
```

Another representation:

```
0 I
1 := f2(0, 0);
return f1(1);
```

What it means:

```
f(I) = f1(f2(I, I));
```

Constraints Generation

Example

Well formed program:

```
0   I
lO  oracle
lO1 := f1(lI1);
lO2 := f2(lI2_1, lI2_2);
```

- all line numbers should be between 0 and 3.
- the output lines should be greater than the input lines (acyclicity).
- `lO1` $\neq$ `lO2` (consistency)

Constraints Generation

Example

Library:

$$O_1 = f_1(I_1)$$
$$O_2 = f_2(I_{21}, I_{22})$$

Constraints Generation

Example

Connectivity:

```
0   I
lO  oracle
lO1 := f1(lI1);
lO2 := f2(lI2_1, lI2_2);
```

- if $I_{21} = I$ then $l_{I21} = 0$
- if $I_1 = O_2$ then $l_{I1} = l_{O2}$

Constraints Generation

Oracle-Guided Component-Based Program Synthesis

$$\begin{aligned}\phi_{func}(L, I, O) = \exists P, R \psi_{wfp}(L) \\ \wedge \psi_{lib}(P, R) \\ \wedge \psi_{conn}(L, I, O, P, R)\end{aligned}$$

Constraints Generation

Oracle-Guided Component-Based Program Synthesis

$$\begin{aligned}\psi_{wfp}(L) = &\bigwedge_{x \in P} (0 \leq l_x < M) \\ &\wedge \bigwedge_{x \in R} (|I| \leq l_x < M) \\ &\wedge \psi_{cons}(L) \wedge \psi_{acyc}(L)\end{aligned}$$

Constraints Generation

Oracle-Guided Component-Based Program Synthesis

$$\psi_{lib}(P, R) = \left(\bigwedge_{i=1}^{N} \phi_i(I_i, O_i) \right)$$

$$\psi_{conn}(L, I, O, P, R) = \bigwedge_{x,y \in P \cup R \cup I \cup \{O\}} (l_x = l_y \Rightarrow x = y)$$

Preconditions bugs

Commons Collections - SequencedHashMap class

```
private Entry findEntry(Map.Entry e) {
  if (e == null)
    return null;
  Entry entry = entries.get(e.getKey());
  if (entry.equals(e)) // entry can be null
    return entry;
  else
    return null;
}
```

Addition of Precondition Check (IF-APC)

Kai Pan et al.[2]:

> *This bug fix adds an if predicate to ensure a precondition is met before an object is accessed or an operation is performed.*

```
– lastChunk.init(seg,expander,x,styles,
–   fontRenderContext, context.rules.getDefault());
+ if (!lastChunk.initialized)
+   lastChunk.init(seg,expander,x,styles,
+     fontRenderContext, context.rules.getDefault());
```

[2]Toward an understanding of bug fix patterns

Problems

- It won’t work with infinite loop bugs.
- Can’t automate the testing process.
- It’s not easy to find candidates.

Problems
Test quality

Quality is free, but only to those who are willing to pay heavily for it.

Tom DeMarco, Peopleware

Limitations

Test quality

- Only 1 set of input values.
- Branch coverage.
- A *removed* precondition can generate an infinite loop.
- Tests that exercise both branches.
- Generates *a* fix not **THE** fix.

Contributions

Process

- Statement ranking (GZoltar) $\rightarrow$
- Ad hoc code manipulation and values capturing $\rightarrow$
- Repair Constraint $\rightarrow$
- Program Synthesis (OGCBPS[3] -paper-)

[3] Oracle-Guided Component-Based Program Synthesis

Experimental methodology

Seeded and wild bugs.

Evaluation / Validation

Generated patches vs. reality.

Perspectives

Conclusion

Contribution

Case study

Commons Math - MathUtils class

```
public static int gcd(int u, int v) {
  if (u * v == 0) {
    return (Math.abs(u) + Math.abs(v));
  }
...
```

Case study

Commons Math

```
assertEquals(3 * (1<<15)
        , gcd(3 * (1<<20), 9 * (1<<15)));
```

Case study

Statement ranking (GZoltar)

```
MathUtils:413 Suspiciousness 0.23570226039551587
MathUtils:431 Suspiciousness 0.1543033499620919

...

MathUtils:460 Suspiciousness 0.11322770341445956
MathUtils:412 Suspiciousness 0.11180339887498948
```

Case study

Ad hoc code manipulation and values capturing (OGCBPS -paper-)

```
public static int gcd(int u, int v) {
  if (true) {
    return (Math.abs(u) + Math.abs(v));
  }
...
```

What are conditional bugs?

Commons Math - MathUtils class

```
public static int gcd(int u, int v) {
    if ((u == 0) || (v == 0)) {
        return (Math.abs(u) + Math.abs(v));
    }
    // ...
}
```