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

Charles Dickens, "Great Expectations."

Repairing Bugs in Conditional Expressions

Favio DeMarco

Universidad de Buenos Aires - INRIA

August 28th, 2013

Context

So far, the Universe is winning.

Bug fixing continues to be a mostly manual, time consuming, and therefore expensive activity in software development.

Hoang Duong Thien Nguyen et al, “SemFix: Program Repair via Semantic Analysis”

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));
  }
...
```

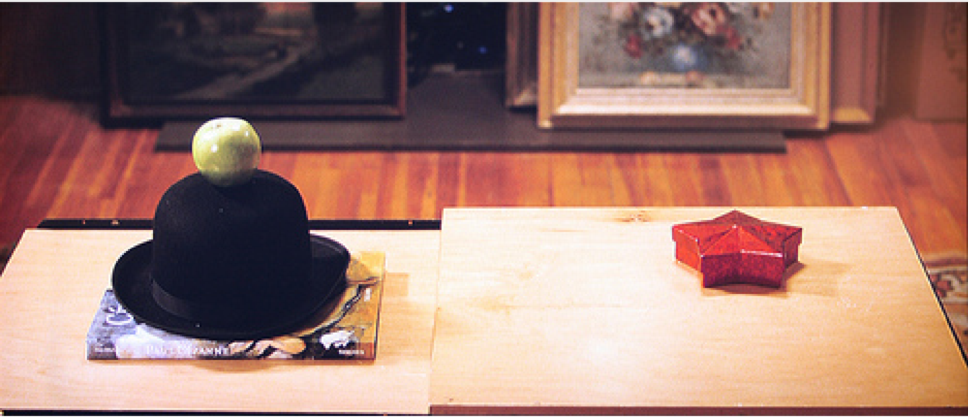
Case study
Commons Math

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

Case study
Commons Math

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

State of the art



State of the art
Trial and error

```
“Fault Localization” (aka Statement Ranking)
while it doesn’t pass all tests
→ generate a candidate and apply it
end while
```

State of the art

SemFix: Program Repair via Semantic Analysis

- **Statement ranking (Tarantula) $\rightarrow$**
- **Symbolic Execution (KLEE) $\rightarrow$**
- **Repair Constraint $\rightarrow$**
- **Program Synthesis (SOLFP[1] -paper-)**

[1] Synthesis of Loop-free Programs

State of the art
GenProg: A Generic Method for Automatic Software Repair

- Statement ranking $\rightarrow$
- Genetic Algorithm

State of the art

ClearView, AutoFix-E, Gopinath et al, Pachika.

Problems

- 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 (OGCBPS[2] -paper-) $\rightarrow$
- Repair Constraint $\rightarrow$
- Program Synthesis (SOLFP[3] -paper-)

[2] Oracle-Guided Component-Based Program Synthesis

[3] Synthesis of Loop-free Programs

Experimental methodology

Seeded and wild bugs.

Evaluation / Validation

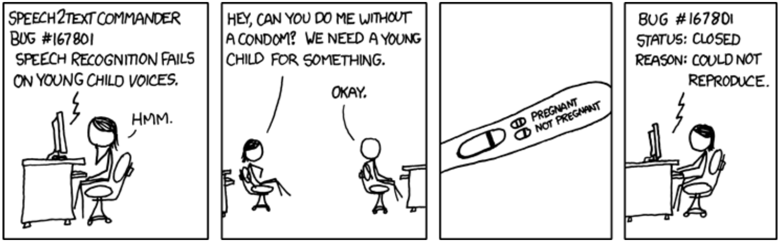
Generated patches vs. reality.

Perspectives

Conclusion

Contribution

Can't and shouldn't.


SPEECH2TEXT COMMANDER
BUG #167801
SPEECH RECOGNITION FAILS ON YOUNG CHILD VOICES.
HMM.
HEY, CAN YOU DO ME WITHOUT A CONDOM? WE NEED A YOUNG CHILD FOR SOMETHING.
OKAY.
PREGNANT
NOT PREGNANT
BUG #167801
STATUS: CLOSED
REASON: COULD NOT REPRODUCE.

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 0.154303349962091

...

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));
  }
...
```

Case study
Repair Constraint

$$(u = 0 \land v = 0 \Rightarrow \mathit{output} = \mathit{true}) \land$$
$$(u = 0 \land v = 55 \Rightarrow \mathit{output} = \mathit{true}) \land$$
$$\dots$$
$$(u = 77 \land v = 55 \Rightarrow \mathit{output} = \mathit{false})$$

Case study
Commons Math

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