Precise Program Repair

Yingfei Xiong

Peking University
Background of Yingfei Xiong

- 2000~2004, UESTC, Undergraduate
- 2004~2006, Peking University, Graduate
  - Advisor: Hong Mei, Fuqing Yang
- 2006~2009, The University of Tokyo, Ph.D.
  - Advisor: Zhenjiang Hu, Masato Takeichi
- 2009~2011, University of Waterloo, PostDoc
  - Supervisor: Krzysztof Czarnecki
- 2012~, Peking University, Assistant Professor under Young Talents Plan

Research Interests
- Software Analysis, Program Language Design
Origin

• “War. War never changes” – Fallout series
  • The war between developers and bugs never changes

• Fault Detection: Is there a bug?
  • Since 60s
  • Example Techniques: Testing, Verification

• Fault Localization: Where is the bug?
  • Since 90s
  • Example Techniques: Spectrum-based fault localization

• Fault Repair: How to fix the program?
  • Since 00s
  • Example Techniques: Test-based Program Repair
“Generate-Validate”
Program Repair

Input: a program and a set of tests, where the program fails at least one test
Output: a patch that makes the program pass all the tests
Existing Work

• GenProg
  • [Weimer et al.: ICSE’09, GECCO’09, CACM’10, ICSE’12]
  • Approach
    • Replace the potentially faulty code with code pieces elsewhere
    • Use search algorithm to find an optimal combination
  • Results: 55/105, 8$/bug

• Inspired a wave of program repair research
  • AutoFix, Nopol, RSRepair, MintHint, AutoRepair, SemFix, DirectFix, SPR...
A Turning Point

• [Qi-ISSTA’15]
  • Only 2 among the 55 defects were correctly fixed by GenProg
  • Reason: passing the test does not guarantee correctness

• [Le Goues-FSE’15]
  • Extensive experiments on more methods, datasets, test suites
  • The finding still holds

• Other work
  • Prophet, Angelix
  • The precision (proportion of correct patches) is lower than 40%
Our Work

High Precision Defect Repair

Learn from QA sites [ASE15]

Precise Condition Repair [ICSE17]

[ASE15] Qing Gao, Hansheng Zhang, Jie Wang, Yingfei Xiong, Lu Zhang, Hong Mei. Fixing Recurring Crash Bugs via Analyzing Q&A Sites. ASE'15

[ICSE17] Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, Lu Zhang. Precise Condition Synthesis for Program Repair. ICSE'17
Fixing from QA Sites

• How do developers get their experience?

```java
29    public void onReceive (final Context context, final Intent intent) {
30    final int action = intent.getExtras().getInt(KEY_ACTION, -1);
31    final float bl = BatteryHelper.level(context);
32    LOG.i("AlarmReceiver invoked: action=%s bl=%s.", action, bl);
33    switch (action) {
34        ...
35    }
36
37}
```

java.lang.RuntimeException: Unable to start receiver
com.vaguehope.onosendai.update.AlarmReceiver:
Fixing from QA Sites

Stack Overflow is a community of 4.7 million programmers, just like you, helping each other only takes a minute:

"IntentReceiver components are not allowed to register to receive inter determine Battery level

android - "IntentReceiver components are not allowed to register to receive inter determine Battery level

android - Battery changed broadcast receiver crashing app ... stackoverflow.com/...battery-changed-broadcast-receiver-crashing-app-... 2 Feb 27, 2013 - Battery changed broadcast receiver crashing app on some phones. No ... PowerConnectionReceiver > <intent-filter> <action android:name="android.intent.action.BATTERY_CHANGED"> ... RuntimeException: Unable to start receiver com.doubelwakexy. ... ReceiverCallNotAllowedException: IntentReceiver components are not ...

android - Want app to execute some code when phone is ... stackoverflow.com/...want-app-to-execute-some-code-when-phone-is-pl... 2 Jun 29, 2012 - ACTION_BATTERY_CHANGED): int plugged = intent ... The code errors out with "FATAL EXCEPTION: main: java.lang.RuntimeException: Unable to start receiver com.example.CharrainOnReceiver: android.content.IntentReceiver

public void sendBatteryInfoMessage() {

    IntentFilter iFilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
    Intent batteryStatus = c.registerReceiver(null, iFilter);
Challenge of Analyzing QA Sites

• It is hard to understand natural languages

Instead of:

```java
4
custom.registerReceiver(null, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
```

use:

```java
context.getApplicationContext().registerReceiver(null, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
```

This is annoying -- `registerReceiver()` should be smarter than this -- but it's the workaround for this particular case.

• Observation: programmers communicates in programming languages

• Solution: Directly compare the code pieces
Approach Overview

1. Crash trace
2. Q&A pages
3. Patch candidates
4. Fixed source code
Experiments

• 24 Android crash bugs that have answers on StackOverflow
  • Selected out of 161 Android crash bugs

• Correctly Fixed : 8
• Wrongly Fixed : 2
• Precision : 80%
• Recall : 33% (5% among Android crash bugs)
Precise Condition Synthesis

• Targeted defect class: condition bugs

```java
lcm = Math.abs(a+b);
+ if (lcm == Integer.MIN_VALUE)
+   throw new ArithmeticException();
```

- if (hours <= 24)
+ if (hours < 24)
  withinOneDay=true;

Missing boundary checks

Conditions too weak or too strong

Condition bugs are common
ACS System

• ACS = Accurate Condition Synthesis
• Two sets of templates for repair

Oracle Returning

• Inserting one of the following statement before the last executed statement
  • if ($C) throw ${Expected Exception};
  • if ($C) return ${Expected Output};

Condition Modifying

• Changing the condition located by predicate switching
  • if ($D) => if ($D || $C)
  • if ($D) => if ($D && $C)

Need to synthesize condition $C
Challenge – Many incorrect conditions pass the tests

```java
int lcm = Math.abs(
    mulAndCheck(a / gdc(a, b), b));
+ if (lcm == Integer.MIN_VALUE) {
+    throw new ArithmeticException();
+}
return lcm;
```

Test 1 (Passed):
Input: a = 1, b = 50
Oracle: lcm = 50

Test 2 (Failed):
Input: a = Integer.MIN_VALUE, b = 1
Oracle: Expected(ArithmeticException)

Correct condition:
lcm == Integer.MIN_VALUE

Incorrect conditions:
• a != 1
• b == 1
• lcm != 50
• ...

Idea: Rank the Conditions

• Rank potential conditions by their probabilities of being correct
• Validate the conditions one by one
• Stop validating when the probability is too low

Condition1 95%
Validate: fail

Condition2 85%
Validate: pass

Condition3 75%
Idea: Rank the Conditions

• Rank potential conditions by their probabilities of being correct
• Validate the conditions one by one
• Stop validating when the probability is too low
Ranking Conditions is Difficult

• The number of potential conditions is large
  • Cannot enumerate the conditions
  • Difficult to perform statistics: not enough samples for each condition
Solution: Divide-and-Conquer

Variables:
- lcm
- a
- b
- lcm

Predicates:
- \( \text{lcm} = \text{Integer.MIN\_VALUE} \)
- \( a \neq 1 \)
- \( b = 1 \)
- \( \text{lcm} \neq 50 \)

Step 1: Rank variables
Step 2: Rank predicates for each variable

Enumerable

Enables more refined ranking techniques

Allows statistics
Ranking Method 1: Rank Variables by Data-Dependency

• **Locality of variable uses**: recently assigned variables are more likely to be used

• Rank variables by data-dependency
  • \( \text{lcm} = \text{Math.abs}(\text{mulAndCheck}(a/gdc(a, b), b)) \)

• Consider only variables in the first two levels
Ranking Method 2: Filter Variables by JavaDoc

```java
/** ...
 * @throws IllegalArgumentException if initial is not between min and max (even if it is a root)
 ***/

Only variable “initial” is considered when throwing IllegalArgumentException
```
# Ranking Method 3: Rank Predicates by Context

- The predicates tested on the variables are related to its context

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Variable Name</th>
<th>Method Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector v = ...; if (v == null) return 0;</td>
<td>int hours = ...; if (hours &lt; 24) withinOneDay=true;</td>
<td>int factorial()</td>
</tr>
</tbody>
</table>

- Approximate the conditional probabilities by querying GitHub
- Consider only the predicates whose probabilities are larger than a threshold
Evaluation: Performance of ACS

Dataset: Four projects from Defects4J benchmark:
- Time, Lang, Math, Chart
- In total 224 defects

<table>
<thead>
<tr>
<th>Approach</th>
<th>Correct</th>
<th>Incorrect</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS</td>
<td>18</td>
<td>5</td>
<td>78.3%</td>
<td>8.0%</td>
</tr>
<tr>
<td>jGenProg</td>
<td>5</td>
<td>22</td>
<td>18.5%</td>
<td>2.2%</td>
</tr>
<tr>
<td>Nopol</td>
<td>5</td>
<td>30</td>
<td>14.3%</td>
<td>2.2%</td>
</tr>
<tr>
<td>xPAR</td>
<td>3</td>
<td>_4</td>
<td>_4</td>
<td>1.3%(^2)</td>
</tr>
<tr>
<td>HistoricalFix(^1)</td>
<td>10(16)(^3)</td>
<td>_4</td>
<td>_4</td>
<td>4.5%(7.1%)(^2,3)</td>
</tr>
</tbody>
</table>
Vision

• Long-term Goal: automate programming
• Roadmap: deal with more and more difficult issues
  • Issue = bug report + feature requests
Conclusion

• Will program repair be useful in practice?
  • Increasing precision is the key

• Can we improve precision?
  • Yes, at least for incorrect conditions and crashes

• How can we improve precision?
  • By learning from existing resources