DirectFix: Looking for Simple Program Repairs

Sergey Mechtaev, Jooyong Yi, Abhik Roychoudhury

March 7, 2017

presenter name(s) removed for FERPA considerations
What is DirectFix?

- Automatic software repair program
  - GenProg
  - SemFix
  - PAR
- Why automate software repair?
  - Mozilla, 300 bugs daily
Similarity to existing?

- Test-driven method
- Automated repair
Difference from existing?

● Simplicity thus safer
Key Ideas

- For existing test-driven repair methods except DirectFix,
  - Statistical fault localization.
  - generate possible patches accordingly.

- For DirectFix,
  - fault localization + repair generation in an efficient way.
  - i.e. without enumerating all possible repairs.
  - insight of potential buggy functions.
A Buggy Program

1. \( x = E_1; \) // \( E_1 \) represents an expression.
2. \( y = E_2; \) // \( E_2 \) represents an expression.
3. \( S_1; \) // \( S_1 \) represents a statement. Neither \( x \) nor \( y \) is redefined by \( S_1 \).
4. if \( (x > y) \) // FAULT: the conditional should be \( x \geq y \)
   5. return 0;
6. else
7. return 1;

(a) A buggy program snippet; a bug is in line 4.
Patch Generated by Genprog

(b) A repair that resembles a GenProg repair
Patch Generated by DirectFix

```plaintext
1  x = E1;
2  y = E2;
3  S1;
4  if (x >= y) // SIMPLE FIX: >= is substituted for >
5       return 0;
6  else
7     return 1;

(c) An alternative simpler repair; an operator is replaced.
```
Another Buggy Program

1. if (x > y) // FAULT 1: the conditional should be x >= z
2.   if (x > z) // FAULT 2: the conditional should be x >= y
3.     out = 10;
4. else
5.     out = 20;
6. else out = 30;
7. return out;

(a) A buggy program snippet; bugs are in line 1 and 2.
Patch Generated by SemFix

1 if (x > y)
2   if (x > z)
3       out = 10;
4     else
5       out = 20;
6     else out = 30;
7 return ((x>=z)? ((x>=y)? 10 : 20) : 30); // This line is one possible repair.

(b) A repair that resembles a SemFix repair
Patch Generated by DirectFix

1. if \((x \geq z)\)  // SIMPLE FIX: \(\geq z\) is substituted for \(> y\)
2. if \((x \geq y)\)  // SIMPLE FIX: \(\geq y\) is substituted for \(> z\)
3.  \quad out = 10;
4.  else
5.  \quad out = 20;
6. else out = 30;
7. return out;

(c) An alternative simpler repair; operators and variables are replaced.
Third Buggy Program

1  // FAULT: k is NOT equal to the length of array s.
2  for (i=0; i<k; i++)
3      if (s[i] == c) return TRUE;
4  return FALSE;

(a) A buggy program that checks if the character c is included in string s.
Expected and Actual output

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>expected</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ab?&quot;</td>
<td>'?'</td>
<td>2</td>
<td>TRUE</td>
</tr>
<tr>
<td>&quot;ab?c&quot;</td>
<td>'?'</td>
<td>3</td>
<td>TRUE</td>
</tr>
<tr>
<td>&quot;!ab&quot;</td>
<td>'!'</td>
<td>2</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

(b) Expected input and output
A repair that pass the above tests

1. `for (i=0; i<k; i++)`
2. `// The following line is one possible repair.
3. `if (c == '?' || c == '!' ) return TRUE;`
4. `return FALSE;`

(c) A (buggy) repair that passes the above tests
A simplest repair

1  for (i=0; i<=k; i++)  // SIMPLE FIX: <= is substituted for <
2   if (s[i] == c) return TRUE;
3 return FALSE;

(d) A more reliable repair
Statement Proposal (Hypothesis)

Simple repairs are less likely to change the correct behavior of the original version than more complex repairs. Thus, simple repairs are likely to be less hazardous.
Solution Statement Proposal

- Iteratively generate a repair at each combination of suspicious program locations.
- Select simplest repair.

- Scalability?
Solution Statement Proposal

- Integrate the two phases of program repair
  - Fault localization
  - Repair search

- How to do these?
DirectFix
WorkFlow

Test suite
Buggy C program → VCC, Boogie → RC generator
trace formula
RC (repair condition)
pMaxSMT solver model → Post processor
Repair
## Experiment Result

<table>
<thead>
<tr>
<th>Subject</th>
<th>Repairs</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>Equivalent (E)</td>
<td>Same Loc (S)</td>
<td>Diff (D)</td>
</tr>
<tr>
<td>Tcas</td>
<td></td>
<td>36 (87%)</td>
<td>19 (54%)</td>
<td>33 (91%)</td>
<td>2.28</td>
</tr>
<tr>
<td>Replace</td>
<td></td>
<td>11 (37%)</td>
<td>9 (81%)</td>
<td>10 (91%)</td>
<td>2.54</td>
</tr>
<tr>
<td>Schedule</td>
<td></td>
<td>4 (44%)</td>
<td>4 (100%)</td>
<td>4 (100%)</td>
<td>2.5</td>
</tr>
<tr>
<td>Schedule2</td>
<td></td>
<td>2 (22%)</td>
<td>1 (50%)</td>
<td>2 (100%)</td>
<td>2</td>
</tr>
<tr>
<td>Coreutils</td>
<td></td>
<td>5 (56%)</td>
<td>0 (0%)</td>
<td>3 (60%)</td>
<td>2</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>59%</td>
<td>56%</td>
<td>89%</td>
<td>2.26</td>
</tr>
</tbody>
</table>
## Result Comparison

<table>
<thead>
<tr>
<th>Subject</th>
<th>Total</th>
<th>DirectFix</th>
<th></th>
<th></th>
<th></th>
<th>SemFix</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>E</td>
<td>S</td>
<td>D</td>
<td>R</td>
<td>E</td>
<td>S</td>
<td>D</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Tcas</td>
<td>30</td>
<td>16</td>
<td>29</td>
<td>2.26</td>
<td>12</td>
<td>3</td>
<td>11</td>
<td>4.1</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Replace</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2.8</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>10.2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2.5</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>8.5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Schedule2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Coreutils</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>44</td>
<td>53%</td>
<td>95%</td>
<td>2.31</td>
<td>31%</td>
<td>17%</td>
<td>46%</td>
<td>6.36</td>
<td>54%</td>
<td></td>
</tr>
</tbody>
</table>

*E stands for Equivalent, S stands for Same Loc, D stands for Diff, and R stands for Regression*
Disadvantages

- Slower
  - DirectFix, 3 minutes 20 seconds
  - SemFix, 9 seconds

- More sizeable changes
  - Time would exhausted.

- Incorrect function designated to be suspicious function
  - Fail to find the repairs
Improvement?

- Search based method, Genprog
  - Aggressively narrow down the search space
- Semantic analysis based method, DirectFix
  - Find the smallest patch

To Improve Scalability
Discussion Question?

What about apply DirectFix on non-buggy program?
Discussion Question?

How DirectFix ensure its repaired output is relevant to original buggy version?
Discussion Question?

The runtime of the DirectFix compared to SemFix, which one is faster?
Discussion Question?

What about DirectFix on buggy program that require more sizable changes?
Discussion Question?

How to eliminate DirectFix constraint on more sizeable changes?
Reference

DirectFix: Looking for Simple Program Repairs
Thank You