Codeflaws: A Programming Competition Benchmark for Evaluating Automated Program Repair Tools

Shin Hwei Tan*, Jooyong Yi*, Yulis*, Sergey Mechtiaev*, Abhik Roychoudhury*
*National University of Singapore, †Innopolis University

Abstract
Several automated program repair techniques have been proposed to reduce the time and effort spent in bug-fixing. While these repair tools are designed to be generic such that they could address many software faults, different repair tools may fix certain types of faults more effectively than other tools. Therefore, it is important to compare more objectively the effectiveness of different repair tools on various fault types. However, existing benchmarks on automated program repairs do not allow thorough investigation of the relationship between fault types and the effectiveness of repair tools. We present Codeflaws, a set of 3902 defects from 7436 programs automatically classified across 39 defect classes (we refer to different types of fault as defect classes derived from the syntactic differences between a buggy program and a patched program).

Our defect classes and example of each defect class

- **Control flow**
  - $	ext{Control flow}$: Replace if, else, for or while
  - Example: if (x > 10) ... else if (x < 5) ...

- **Data flow**
  - $	ext{Data flow}$: Replace expression
  - Example: $x = x + y$ (instead of $x = x + y * z$)

- **Arithmetic**
  - $	ext{Arithmetic}$: Replace arithmetic operator
  - Example: $x = x + 1$ (instead of $x = x * 1$)

- **Operator**
  - $	ext{Operator}$: Replace operator name
  - Example: $= >$ (instead of $= >$)

- **Variable**
  - $	ext{Variable}$: Replace with variable or constant
  - Example: $x = x + 1$ (instead of $x = x + 1$)

- **Array**
  - $	ext{Array}$: Replace array size
  - Example: $[0 ... n-1]$ (instead of $[0 ... n-1]$)

- **Higher-order**
  - $	ext{Higher-order}$: Replace higher-order defect
  - Example: $f(x, y) = f(x, y)$ (instead of $f(x, y) = f(x, y)$)

- **Instruction**
  - $	ext{Instruction}$: Replace instruction
  - Example: $x = x + 1$ (instead of $x = x + 1$)

- **Expression**
  - $	ext{Expression}$: Replace expression
  - Example: $x = x + 1$ (instead of $x = x + 1$)

- **Other**
  - $	ext{Other}$: Other higher-order defect classes
  - Example: $f(x, y) = f(x, y)$ (instead of $f(x, y) = f(x, y)$)

---

The Basic Statistics of Subject Programs in Codeflaws

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Total/Range</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Programming Contest</td>
<td>548</td>
<td>-</td>
</tr>
<tr>
<td># of Programming Problems</td>
<td>1284</td>
<td>-</td>
</tr>
<tr>
<td># of Programs</td>
<td>7436</td>
<td>-</td>
</tr>
<tr>
<td># of Defects</td>
<td>3902</td>
<td>-</td>
</tr>
<tr>
<td>Size of Repair Test Suite</td>
<td>2-8</td>
<td>3</td>
</tr>
<tr>
<td>Size of Held-out Test Suite</td>
<td>5-350</td>
<td>40</td>
</tr>
<tr>
<td>Source Lines of Codes</td>
<td>1-322</td>
<td>36</td>
</tr>
</tbody>
</table>

---

Our Criteria for Automated Program Repair Benchmark

- C1: Diverse types of real defects.
- C2: Large number of defects.
- C3: Large number of programs.
- C4: Programs that are algorithmically complex
- C5: Large held-out test suite for patch correctness verification

- Defect class classification based on the syntactic differences between the buggy program and the patched program.
  1) Allows automatic classification of defect classes
  2) Enables extensive evaluation of different repair tools
  3) Commonly deployed in the literature

---

Example Usage In Intelligent Tutoring

- What kind of programming mistakes have I made?
  - **Buggy Program**
  123: if(sum < n)

- Automated Program Repair
  - Hint: Wrong relational operator at line 123
  - Repair: if(sum>=n) + if(sum>n)

---

Conclusion

- Our Codeflaws benchmark aim to facilitate future empirical study in automated program repair.
- A step towards the evaluation of program repair tools against multiple dimensions with defect classes being one such dimension.
- Publicly available for download at: https://codeflaws.github.io/