Coincidental Correctness: an Interference or Interface to Successful Fault Localization?

Zheng Zheng, Yichao Gao, Peng Hao, Zhenyu Zhang

1School of Automation Science and Electrical Engineering
2Institute of Software, Chinese Academy of Sciences

zhengz@buaa.edu.cn, zhangzy@ios.ac.cn
Contents

• 1. Introduction
• 2. Motivation
• 3. Our Technique
• 4. Empirical Evaluation
• 5. Conclusion
1. Introduction

- Statistical Fault Localization (SFL)
  - Compare spectra abstracted from both failed runs and non-failed runs
  - Predicate-based SFL techniques (CBI)
  - Statement-level SFL techniques (Tarantula)
1. Introduce

- Coincidental Correctness
  - No failure is detected, though the fault has been exercised
  - Negative impact on the accuracy of SFL Techniques

- Previous Solution
  - Recognize and remove the coincidental correct runs
  - Low precision of the recognition of coincidental correctness

Can we allow the existence of coincidental correctness and locate a fault with the precedence of it?
Contents

• 1. Introduction
• 2. Motivation
• 3. Our Technique
• 4. Empirical Evaluation
• 5. Conclusion
2. Motivation

Program:

```c
int max(int a, int b, int c) {
    int x = a;
    if (a > b) {
        x = a;
        if (x > c) {
            return x;
        }
    } else {
        x = a; // x = b
    }
    return c;
}
```

<table>
<thead>
<tr>
<th>L1: int x = a;</th>
<th>P1: x=a</th>
<th>T1: (2,1,3)</th>
<th>T2: (2,1,2)</th>
<th>T3: (1,1,3)</th>
<th>T4: (1,2,3)</th>
<th>T5: (2,3,1)</th>
<th>T6: (1,2,0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2: if (a &gt; b)</td>
<td>P2: a&gt;b</td>
<td>1:0</td>
<td>1:0</td>
<td>≠ 0:1</td>
<td>0:1</td>
<td>≈ 0:1</td>
<td>0:1</td>
</tr>
<tr>
<td>L3: x = a;</td>
<td>P3: x=a</td>
<td>0:1</td>
<td>0:1</td>
<td>≠ 0:0</td>
<td>0:0</td>
<td>≈ 0:0</td>
<td>0:0</td>
</tr>
<tr>
<td>L4: else</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5: x = a; // x = b</td>
<td>P4: x=a</td>
<td>0:0</td>
<td>0:0</td>
<td>≠ 0:1</td>
<td>0:1</td>
<td>≈ 0:1</td>
<td>0:1</td>
</tr>
<tr>
<td>L6:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L7: if (x &gt; c)</td>
<td>P5: x&gt;c</td>
<td>0:1</td>
<td>0:1</td>
<td>≈ 0:1</td>
<td>0:1</td>
<td>≠ 1:0</td>
<td>1:0</td>
</tr>
<tr>
<td>L8: return x;</td>
<td>P6: x=0</td>
<td>0:0</td>
<td>0:0</td>
<td>≈ 0:0</td>
<td>0:0</td>
<td>≠ 0:1</td>
<td>0:1</td>
</tr>
<tr>
<td>L9: else</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10: return c;</td>
<td>P7: c=0</td>
<td>0:1</td>
<td>0:1</td>
<td>≈ 0:1</td>
<td>0:1</td>
<td>≠ 0:0</td>
<td>0:0</td>
</tr>
</tbody>
</table>

Observations

| Neutral predicates: Similar spectra in all runs. |
| Fault-leading predicates: Spectra in {T3,T4} resemble those in {T5,T6}, because both of them lead to trigger the fault; spectra in {T3,T4} vary from those in {T1,T2}, because the latter lead to trigger no fault. |
| Fault-led predicates: Spectra in {T3,T4} vary from those in {T5,T6}, because the latter reveal failures; spectra in {T3,T4} resemble those in {T1,T2}, because both of them reveal no failure. |

Fig. 1. Motivating example (predicates may behave differently according to their relative position to fault)

- **Neutral Predicate** does not correlated with the existing status of fault
- **Fault-leading Predicate** leads to trigger or skip the fault
- Triggering fault determines to exercise or skip the **Fault-led Predicates**
2. Motivation

- On a Neutral Predicate

- Its dynamic spectra in every runs resemble each other
  - Less relation with the fault
  - Less difference in behavior

Program:

```c
int max(int a, int b, int c) {
    int x = a;
}
```

<table>
<thead>
<tr>
<th></th>
<th>Dynamic spectra of predicates</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1:</td>
<td>T1: (2, 1, 3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T2: (2, 1, 2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3: (1, 1, 3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T4: (1, 2, 3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T5: (2, 3, 1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T6: (1, 2, 0)</td>
<td></td>
</tr>
<tr>
<td>P1:</td>
<td>x==a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0:1</td>
<td></td>
</tr>
</tbody>
</table>

Neutral predicates: Similar spectra in all runs.

- Successful
- Coincidental
- Failed
2. Motivation

• On a Fault-Leading Predicate

- Its dynamic spectra in coincidental runs are similar to those in failed runs but different from those in successful runs
  - Both coincidental runs and failed runs have fault triggered
  - The execution paths leading to a fault often concentrate into a small cluster*
  - The overlapping of spectrum distribution in failed runs with that in non-failed runs is 100%

<table>
<thead>
<tr>
<th>Program:</th>
<th>Dynamic spectra of predicates</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>int max(int a, int b, int c) {</td>
<td>T1: (2,1,3)</td>
<td>T2: (2,1,2)</td>
</tr>
<tr>
<td>L2: if (a &gt; b) P2: a&gt;b</td>
<td>1:0</td>
<td>1:0</td>
</tr>
<tr>
<td>L3: x = a; P3: x==a</td>
<td>0:1</td>
<td>0:1</td>
</tr>
<tr>
<td>L4: else</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5: x = a; // x = b P4: x==a</td>
<td>0:0</td>
<td>0:0</td>
</tr>
</tbody>
</table>

*W. Dickinson et.al. Pursuing failure: the distribution of program failures in a profile space. FSE, 2001
2. Motivation

• On a Fault-Led Predicate

- Its dynamic spectra in coincidental runs are similar to those in **successful** runs but different from those in **failed** runs
  - The faulty state may be coincidentally not propagated
  - Program executions still behave as normal
  - The overlapping of spectrum distribution in failed runs with that in non-failed runs is 0%

<table>
<thead>
<tr>
<th>Program:</th>
<th>Dynamic spectra of predicates</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>L7: if (x &gt; c)</td>
<td>P5: x&gt;c</td>
<td>0:1 0:1 ≈ 0:1 0:1 ≠ 1:0 1:0</td>
</tr>
<tr>
<td>L8: return x;</td>
<td>P6: x==0</td>
<td>0:0 0:0 ≈ 0:0 0:0 ≠ 0:1 0:1</td>
</tr>
<tr>
<td>L9: else</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L10: return c;</td>
<td>P7: c==0</td>
<td>0:1 0:1 ≈ 0:1 0:1 ≠ 0:0 0:0</td>
</tr>
</tbody>
</table>

**Fault-led predicates:**
- Spectra in {T3,T4} vary from those in {T5,T6}, because the latter reveal failures;
- spectra in {T3,T4} resemble those in {T1,T2}, because both of them reveal no failure.
2. Motivation

• By comparing the extent of overlapping, we can rule out the fault-led predicates

However

• A predicate can be evaluated more than once because of the presence of loops
• The opposite overlapping (spectra in non-failed runs with that in failed runs) can be also a good indicator
• The neutral predicates have high overlapping and are mixed up with the fault-leading predicates.
Contents

• 1. Introduction
• 2. Motivation
• 3. Our Technique
• 4. Empirical Evaluation
• 5. Conclusion
3. Our technique

- Our fault localization technique consists of four steps:

  \( S1: \) Collecting dynamic spectra

  \( S2: \) Calculating the Overlapping of Spectrum Distributions

  \( S3: \) Calculating the Inter- and Intro-class distances

  \( S4: \) Generating a Ranked List of Suspicious Predicates
3. Our technique

- **S2: Calculating the Overlapping of Spectrum Distributions**

Let:

- $y_j$ be a variable of evaluation bias for predicate $p_j$
- $\omega_N$ and $\omega_F$ denote the non-failed runs and failed runs respectively

**Bhattacharyya coefficient:**

$$BC \left( P(y_j|\omega_F), P(y_j|\omega_N) \right) = \sum_{y_j \in D_j} \sqrt{P(y_j|\omega_F) \times P(y_j|\omega_N)}$$

where, $D_j$ is the domain of $y_j$, $P(y_j|\omega_F)$ and $P(y_j|\omega_N)$ are the conditional probabilities of $y_j$ in the set of failed runs and non-failed runs respectively.

The measurement is proved to be the upper bound of Bayes error, which directly related to the overlapping of two models *.

---

3. Our technique

- **S2: Calculating the Overlapping of Spectrum Distributions**

Given a predicate $p_j$, the overlapping $O_j$ of the spectrum distribution in failed runs with that in non-failed runs can be expressed in terms of the Bhattacharyya distance:

$$O_j = - \ln \left[ BC \left( P(y_j|\omega_F), P(y_j|\omega_N) \right) \right]$$

where $BC(\cdot)$ is the Bhattacharyya coefficient. If $BC(\cdot) = 0$, we set $O_j$ to be $+\infty$.

The top ranked predicates are supposed to be fault-leading predicates mixing up with the neutral predicates.
3. Our technique

- **S3: Calculating the Inter- and Intro-class distances**

The **inter-class distance** $B_j$ for a predicate $p_j$ is calculated as:

$$B_j = |m_j^F - m_j^N|$$

where $m_j^F$ and $m_j^N$ are the mean value of the evaluation bias for $p_j$ in the failed and the non-failed program runs, respectively.

Here, $m_j^F$ and $m_j^N$ are calculated as follow.

$$m_j^F = \frac{1}{v} \sum_{i=1}^{v} [x(p_j, f_i)], \quad m_j^N = \frac{1}{u} \sum_{i=1}^{u} [x(p_j, n_i)]$$
3. Our technique

• **S3: Calculating the Inter- and Intro-class distances**

The *intra-class distance* $D_j$ for a predicate $P_j$ is calculated as,

$$D_j = \sqrt{\frac{\sum_{i=1}^{v} [(x(p_j, f_i) - m_j^F)^2]}{v}} + \sqrt{\frac{\sum_{i=1}^{u} [(x(p_j, n_i) - m_j^N)^2]}{u}}$$

The normalized inter-class distance $A_j$ for $P_j$ is as follows:

$$A_j = \frac{B_j}{D_j}$$

When $D_j$ is zero and $B_j$ is not zero, we set $A_j$ to be $+\infty$.

When $D_j$ is zero and $B_j$ is also zero, we set $A_j$ to be zero.

This step decreases the ranks of the neutral predicates without affecting the relative order of the fault-leading predicates and the fault-led predicates.
3. Our technique

• Our fault localization technique consist of four steps:

   \textit{S1: Collecting dynamic spectra}
   \textit{S2: Calculating the Overlapping of Spectrum Distributions}
   \textit{S3: Calculating the Inter- and Intro-class distances}
   \textit{S4: Generating a Ranked List of Suspicious Predicates}
Contents

• 1. Introduction
• 2. Motivation
• 3. Our Technique
• 4. Empirical Evaluation
• 5. Conclusion
4. Empirical Evaluation

- **Experiment Setup**
  - Subject programs:

<table>
<thead>
<tr>
<th>Programs</th>
<th># of selected versions</th>
<th># of LOC</th>
<th># of predicates</th>
<th># of runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>print_tokens</td>
<td>4</td>
<td>472</td>
<td>51</td>
<td>4130</td>
</tr>
<tr>
<td>print_tokens2</td>
<td>10</td>
<td>399</td>
<td>116</td>
<td>4115</td>
</tr>
<tr>
<td>schedule</td>
<td>9</td>
<td>292</td>
<td>24</td>
<td>2650</td>
</tr>
<tr>
<td>schedule2</td>
<td>9</td>
<td>301</td>
<td>55</td>
<td>2710</td>
</tr>
<tr>
<td>replace</td>
<td>30</td>
<td>512</td>
<td>63</td>
<td>5542</td>
</tr>
<tr>
<td>tot_info</td>
<td>19</td>
<td>440</td>
<td>47</td>
<td>1052</td>
</tr>
<tr>
<td>tcas</td>
<td>30</td>
<td>141</td>
<td>10</td>
<td>1608</td>
</tr>
<tr>
<td>space</td>
<td>28</td>
<td>6218</td>
<td>914</td>
<td>13585</td>
</tr>
<tr>
<td>flex</td>
<td>20</td>
<td>15297</td>
<td>895</td>
<td>567</td>
</tr>
<tr>
<td>grep</td>
<td>12</td>
<td>15633</td>
<td>1284</td>
<td>809</td>
</tr>
</tbody>
</table>

- Peer techniques: CBI, SOBER, Wilcoxon, and Mann-Whitney
- Effectiveness metrics: P-score
4. Empirical Evaluation

- Experiment Results

![Graph showing the relationship between P-score (Percentage of predicates examined) and the percentage of faults located for different methods: J-B, Mann-Whitney, Wilcoxon, CBI, and SOBER.]
Contents

• 1. Introduction
• 2. Motivation
• 3. Our Technique
• 4. Empirical Evaluation
• 5. Conclusion
5. Conclusion

• Use the presence of coincidental correctness to locate faults.
  – Analyze the phenomenon of coincidental correctness to fault localization
  – Measure the overlapping of spectrum distribution to rule out fault-led predicates.
  – Suppress neutral predicates by assessing the inter-class distribution of spectra in failed runs and non-failed runs.

• Preliminary evaluation shows that our technique is more effective than the other techniques.
Thank you!

Zheng Zheng, Yichao Gao, Peng Hao, Zhenyu Zhang

1School of Automation Science and Electrical Engineering
2Institute of Software, Chinese Academy of Sciences

zhengz@buaa.edu.cn, zhangzy@ios.ac.cn