Modifying Test Suite Composition for Effective Statistical Debugging

Ross Gore
University of Virginia
rjg7v@virginia.edu
Motivation

Efficient debugging is needed:
- Lots of software ships with faults
- Programmers read code to fix faults

Harder for safety-critical:
- Floating-point data types and computations
- Numerical Analysis errors
- Affects millions of lives and billions of $$
Given a failing subject program, rank the likelihood that each predicate ($p$) reflects the fault.

$$\text{Suspiciousness}(p) = \frac{f_p}{f_p + s_p}$$

**Suspiciousness**($p$) based on:
- Test Cases (Program Inputs)
- Execution Profiles
- Status of Test Cases
  - Successful (passing) - $s$
  - Failing - $f$
Elastic Predicates

Static Predicates
• All variables (x) partitioned the same
  • Value is negative, x < 0
  • Value is zero, x = 0
  • Value is positive, x > 0

Elastic Predicates
• Variables (x) partitioned based on observed values
  • Value is a lot higher than average, x > \mu_x + \sigma_x
  • Value matches average, x = \mu_x
  • Value is a lot lower than average, x < \mu_x - \sigma_x
Predicate-level Statistical Debugging

double mean(int x [])
{
    int i = 0;
    double sum = 0;
    while (i < x.length){
        sum = sum + x[i];
        i = i + 1;
    }
    if (x.length >= 0){ /* off by one */
        return (sum / x.length);
    }
    else {
        return (0);
    }
}
Divide by Zero
Predicate-level Statistical Debugging

```
double mean(int x [])
{
    int i = 0;
    double sum = 0;
    while (i > x.length){
        sum = sum + x[i];
        i = i + 1;
    }
    if (x.length >= 0){ /* off by one */
        return (sum / x.length);
    }
    else {
        return (0);
    }
}
```
Elastic Predicate Summary

- Elastic predicates enable improved effectiveness
  - Floating point computations
  - Numerical analysis errors
  - Incurs additional space and time
- Improvements hold in the face of:
  - Sparse Sampling
  - Incomplete Test Suites
Motivating Example

```c
int distance(int x, int y)
{
    int diff = x - y;
    if (!(diff > 1)) /* off by one */
    {
        int dist = 0;
        dist = y - x;
        print(dist);
    }
    int dist = 0;
    dist = x - y;
    return dist;
}
```
## Motivating Example

<table>
<thead>
<tr>
<th>Statement</th>
<th>Predicate</th>
<th>TC 1: {2,2}</th>
<th>TC 2: {5,4}</th>
<th>TC 3: {5,1}</th>
<th>TC 4: {-4,-2}</th>
<th>TC 5: {1,0}</th>
<th>Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>diff = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>diff &gt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>diff &lt; 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>dist = 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>dist = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>dist = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTCOME</th>
<th>PASS</th>
<th>FAIL</th>
<th>PASS</th>
<th>PASS</th>
<th>FAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Motivating Example

<table>
<thead>
<tr>
<th>Statement</th>
<th>Predicate</th>
<th>TC 1: {2,2}</th>
<th>TC 2: {5,4}</th>
<th>TC 3: {5,1}</th>
<th>TC 4: {-4,-2}</th>
<th>TC 5: {1,0}</th>
<th>Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>diff = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>diff &gt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>diff &lt; 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>dist = 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>dist = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>dist = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**OUTCOME**

<table>
<thead>
<tr>
<th>PASS</th>
<th>FAIL</th>
<th>PASS</th>
<th>PASS</th>
<th>FAIL</th>
</tr>
</thead>
</table>

Ranks: 3, 4, 12
## Motivating Example

<table>
<thead>
<tr>
<th>Statement</th>
<th>Predicate</th>
<th>TC 1: {2,2}</th>
<th>TC 2: {5,4}</th>
<th>TC 3: {5,1}</th>
<th>TC 4: {-4,-2}</th>
<th>TC 5: {1,0}</th>
<th>Ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>diff = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>diff &gt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>diff &lt; 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>dist = 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>dist = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>dist = 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>dist &gt; 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>dist &lt; 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTCOME</th>
<th>PASS</th>
<th>FAIL</th>
<th>PASS</th>
<th>PASS</th>
<th>FAIL</th>
</tr>
</thead>
</table>

The highlighted cell in the table indicates a rank of 4.
Confounding Bias

(diff > 0)\textsubscript{5} causes (dist < 0)\textsubscript{7} and (dist < 0)\textsubscript{8} to be true in every failing test case.

This creates bias in the suspiciousness estimate for (dist < 0)\textsubscript{7} and (dist < 0)\textsubscript{8}.
Confounding Bias

There is no difference between \((\text{dist} = 0)_6\) being true vs. Statement 6 being executed.

This creates bias in the suspiciousness estimate for \((\text{dist} = 0)_6\).
Reducing Confounding Bias

- Confounding Biases
  - *Control for* the most immediate cause of a statement being executed
  - *Control for* the immediate cause of a predicate being evaluated
Observational Studies

Create a regression model for each predicate:

• **Treatment Variable** ($T$)
  – $T = 1$ if predicate is true in test case (Treatment)
  – $T = 0$ if predicate is not true in test case (Control)

• **Outcome Variable** ($Y$)
  – $Y = 1$ if test case fails
  – $Y = 0$ if test case is successful (passes)
Controlling for Failure Flow Bias

\[ Y = \alpha_p^{c,f} + \tau_p^{c,f} T_p + \beta_p^{c,f} C_p + \omega_p^{c,f} F_p + \epsilon_p^{c,f} \]

\[ \tau_{ls,p}^{c,f} \] is the least squares suspiciousness estimate
Matching Test Cases

• Same pattern of covariates should exist in Treatment and Control Group
  – control flow predecessor true when predicate is not true?
  – predicate evaluated when predicate is not true?

• Test suite is given
  – Likely to be unbalanced
Estimating Suspiciousness with Matching

\[ Y = \alpha_p + \tau_p T_p + \varepsilon_p \]

\( \tau_{ls,p} \) is the least-squares suspiciousness estimate
Causal Imputation

Algorithm 1 Suspiciousness imputation for predicates with a complete lack of overlap problem.

\begin{algorithm}
\textbf{IMPUTE-SUSP}(p)
\begin{algorithmic}[1]
\State \texttt{matchedTestCases} $\leftarrow$ \texttt{GET-MATCHED-TEST-CASES}(p)
\If{\texttt{matchedTestCases} = $\emptyset$}
\State \texttt{ancestor} $\leftarrow$ \texttt{GET-CFP}(p)
\State \texttt{suspiciousness} $\leftarrow$ \texttt{IMPUTE-SUSP}(\texttt{ancestor})
\EndIf
\State \texttt{suspiciousness} $\leftarrow$ $\tau_{s,p}$
\State \textbf{return} \texttt{suspiciousness}
\end{algorithmic}
\end{algorithm}
\[ d_M(a, b) = \sqrt{(a - b)^T S^{-1} (a - b)} \]
MD Matching

1. For each test case $T_i$ in the treatment group
   a. find the test case $C_{min}$ in the control group that minimizes
   
   $d_M (T_i, C_j)$

2. Move $T_i$ and $C_{min}$ to the set of test cases to fit:

   $Y = \alpha_p + \tau_p T_p + \varepsilon_p$
Results

MaSTRI

[Graph showing improvement percentages for different program versions, comparing Static Predicate and Elastic + Static Predicate]
Conclusion

- Balanced covariate values in test cases improve effectiveness
  - Especially for elastic predicates
- Large cost in terms of efficiency
Questions / Comments