Information Theory and Software Testing

David Clark


Problems

- What is the test execution order that locates a software fault as quickly as possible?
- How can we choose tests that don’t suffer from coincidental correctness?
- How do we know that we have enough tests?
- How do we know that our test suite is sufficiently diverse?
- How can we measure how much a real oracle deviates from an ideal oracle?
Shannon Entropy

The randomness of a random variable $X$ is given by the Shannon entropy:

$$\mathcal{H}(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

randomness of a random variable
The length of the shortest program that can produce a given string from no inputs

randomness of a string
Program with \( m \) statements, \( S = \{ s_0, s_1, \ldots, s_{m-1} \} \)
Test suite with \( n \) tests, \( T = \{ t_0, t_1, \ldots, t_{n-1} \} \)
\( S \) contains a single fault
Random variable \( X \) models fault locality
\( p(X = s_j) \) is the probability that \( s_j \) is the faulty statement
\( \mathcal{H}(X) \to 0 \) as fast as possible
Estimate the change in entropy due to each test
Employ a greedy algorithm to select the next test
AKA “suspicousness” metrics: likelihood of statement containing the fault
- Tarantula, Ochiai, Jaccard etc.

\[
\tau(s) = \frac{\text{fail}(s)}{\text{total fail}} + \frac{\text{pass}(s)}{\text{total pass}} + \frac{\text{fail}(s)}{\text{total fail}}
\]
that some statements other than the faulty statement get a higher $\tau$ value than $s_0$. Suppose that $s_0$ causes a failure only for certain input values, whereas an error handling routine $s_{00}$ is executed whenever $s_0$ fails: $s_{00}$ will get assigned $\tau = 1$, whereas $s_0$ might get assigned $\tau$ less than 1 depending on the test input.

<table>
<thead>
<tr>
<th>Structural Elements</th>
<th>Test $t_1$</th>
<th>Test $t_2$</th>
<th>Test $t_3$</th>
<th>Tarantula Metric($\tau$)</th>
<th>Test $t_4$</th>
<th>Tarantula Metric($\tau$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>$s_2$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>$s_3$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>$s_4$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>$s_5$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.00</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>$s_6$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>1.00</td>
<td>(\bullet)</td>
<td>1.00</td>
</tr>
<tr>
<td>$s_7$ (faulty)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.67</td>
<td>(\bullet)</td>
<td>1.00</td>
</tr>
<tr>
<td>$s_8$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>1.00</td>
<td>(\bullet)</td>
<td>1.00</td>
</tr>
<tr>
<td>$s_9$</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>(\bullet)</td>
<td>0.67</td>
<td>(\bullet)</td>
<td>0.50</td>
</tr>
</tbody>
</table>

Result: P F P - F -
- \( B(s_j) \) is the event that \( s_j \) is faulty
- \( T_i = T_{i-1} \cup \{ t_i \} \) is a set of tests
- \( \tau(s|T_i) \) is the suspiciousness of \( s \) after executing \( T_i \)

**Tarantula induced Probability Distribution**

\[
P_{T_i}(B(s_j)) = \frac{\tau(s_j|T_i)}{\sum_{j=1}^{m} \tau(s_j|T_i)}
\]

**Tarantula induced Entropy**

\[
H_{T_i}(S) = - \sum_{j=1}^{m} P_{T_i}(B(s_j)) \cdot \log P_{T_i}(B(s_j))
\]
Entropy Lookahead

Lookahead Probability Distribution on Failure

\[ \alpha = P_{T_{i+1}}(F(t_{i+1})) \approx \frac{TF_i}{TP_i + TF_i} \]

Lookahead Probability Distribution on Fault location

\[ P_{T_{i+1}}(B(s_j)) = P_{T_{i+1}}(B(s_j)|F(t_{i+1})) \cdot \alpha + P_{T_{i+1}}(B(s_j)|\neg F(t_{i+1})) \cdot (1 - \alpha) \]

- \( F(t_i) \) is the event that \( t_i \) is identified as a failing test
- Use \( P_{T_{i+1}}(B(s_j)) \) to calculate \( H_{T_{i+1}}(S) \), the estimated entropy of \( B \) that results from adding the execution of \( t_{i+1} \)
Outcomes

- Approach is independent of the fault localisation method used
- Experimental evidence from four SUTs plus their test suites drawn from the Software Infrastructure Repository (SIR)
- Increased the suspiciousness ranking and decreased the cost of fault localisation for 70% of the faults examined

Paper

Use Conditional Entropy to avoid Coincidental Correctness

intended

input

\[ t_1 : x = 3 \]
\[ t_2 : x = -5 \]

\[
\begin{align*}
x &= x + 2; \\
\text{if}(x > 0) & \quad x = x \% 4; \\
\text{else} & \quad x = x; \\
x &= 3 \times x; \\
\text{if}(x > 0) & \quad x = x \% 4; \\
\text{else} & \quad x = x;
\end{align*}
\]

output

\[ t_1 : x = 1 \]
\[ t_2 : x = -3 \]

unintended

output

\[ t_1 : x = 1 \]
\[ t_2 : x = -15 \]
The Abstract View

Intended

Unintended

Intended

Unintended
Loss of information from running program $P$

$H(I) - H(O) = H(I|O)$

where $[P]I = O$

Conditional entropy of $I$ given $O$:

**Squeeziness**.

$Sq(f) = H(I) - H(O) = \sum_{o \in O} p(o) \ H(f^{-1}o)$

via the partition property
Example Hypothesis

\[ \pi = A' B' \]
\[ \pi_l = B' \]

Intended

Unintended

\[ \begin{bmatrix} \pi \end{bmatrix}^{pa}_{pp'} \]

David Clark  IT and ST
Summary

- 30 SUTS
- 1,408 Mutants
- 7,140,00 test cases
- Five different IT metrics experimentally investigated
- Two metrics showed 0.95 Spearman rank correlation with the probability of failed error propagation
- 10% of all 7,140,000 test inputs suffered from FEP

Paper

Use Kolmogorov Complexity to Measure Input Diversity

### Normalised Information Distance

For two strings $x$ and $y$,

$$\text{NID}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}$$

Enables comparisons between strings of different lengths

### NCD: The Normalised Compression Distance

For two strings $x$ and $y$,

$$\text{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

Computable approximation using compressors such as 7zip, Bzip
Experiments

- Use a version of NCD for multisets – calculate the set “diameter”
- Bigger diameter means more diversity
- Purely consider sets of inputs – no information from executions except in the course of evaluation
- Inputs for three SUTs: JEuclid, NanoXML, ROME
- Controlled for input size
- Compared test sets using three fixed sizes: 10, 25 and 50
Outcomes for Higher Diameter Test Sets

- On average higher code coverage
- Higher code coverage than randomly selected test sets
- Leads to higher code coverage even if we control for the size of test inputs
- May have better fault-finding ability
- Selection scales quadratically in the size of the initial pool of tests and linearly with the average length of the tests

Paper

Oracles may be too strong (false alarms) or too weak (missed faults)
Oracle Improvement Steps

Since $E$ is fixed:
\[a + b = \text{const}\]
\[c + d = \text{const}\]
(repartitioning)

**False negative reduction:**
\[a' = a + \Delta\]
\[b' = b - \Delta\]

**False positive reduction:**
\[c' = c + \Gamma\]
\[d' = d - \Gamma\]
Mutual information

\[ I(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \]

\[ I(\alpha; G) = \begin{cases} 
-(b + c)\log_2(b + c) - (a + d)\log_2(a + d) \\
-(a + b)\log_2(a + b) - (c + d)\log_2(c + d) \\
+a \log_2 a + b \log_2 b + c \log_2 c + d \log_2 d 
\end{cases} \]
A **bad oracle** $\alpha$ is one for which $ac < bd$.

$$\Delta = \frac{bd - ac}{c + d}$$
In Conclusion

- Looked at contributions both theoretical and practical to
  - oracle improvement
  - test set diversity
  - coincidental correctness
  - test set prioritisation
- More to come:
  - InfoTestSS EPSRC funded project
  - Applying information theoretic ideas to test set selection and exploring relationships with coverage and mutation testing
  - EPSRC contribution approx £900,000 shared between UCL and Brunel
  - Industrial contribution approx £230,000 from J.P.Morgan and Berner Mattner
  - Project collaborators include Rob Hierons, Mark Harman, Robert Feldt, Michele Boreale, Paolo Tonella