Conditional Entropy and Failed Error Propagation in Software Testing

Rob Hierons
Brunel University London

Joint work with: Kelly Androutsopoulos, David Clark, Haitao Dan, Mark Harman

COW, Jan 2018
Coverage criteria

• The most widely used type of test criterion.

• Mandated in some standards.

• Failing to achieve coverage clearly demonstrates that testing is weak.

• Syntactic: what does achieving coverage tell us?
Finding Faults

• To find a fault in statement $s$ a test case must:
  – Execute $s$.
  – Infect $s$.
  – Propagate this to output.

• (The PIE framework.)

• Propagation is not just data dependence.
  – Consider e.g. statement $y = y \mod 2$;
Failed Error Propagation (FEP)

• This occurs when:
  – A test case leads to execution and infection but not propagation.

• Makes testing less effective.

• Empirical evidence suggests:
  – Affects approximately 10% of test cases but this can be as high as 60% for some programs.

COW, Jan 2018
FEP and Coverage

• The ‘hope’ in coverage is that:
  – If a test case executes e.g. a statement s and this contains a fault then the test case will find this fault.

• This already looks weak (need ‘infection’).
  – Also need to avoid FEP.

• Could help explain evidence of limited effectiveness of coverage.
Failed Error Propagation (FEP)
The basic idea

• In test execution FEP occurs through the following:
  – The program state at statement s should be $\sigma$ but is $\sigma'$.
  – The code after this maps $\sigma$ and $\sigma'$ to the same output.

• There has been a loss of information.

• Underlying assumption: only one fault.
Squeeziness

• This is the loss of entropy (uncertainty) during computation.
• For function $f$ with input domain $I$ this is:

$$Sq(f, I) = H(I) - H(O)$$

• Where

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$
Estimating the probability of FEP

• Using test case t, FEP is caused by a lack of information flow after a fault (in statement s).

• We could use:
  – The Squeeziness of the code that follows s.
    • The QIF of Q; or
    • The QIF of the path.

• The former captures the computation; the latter might approximate this.

• Should we consider the code before s?
Possible measures

• M1: Squeeziness of Q (on the states at pp’)
• M2: M1 + Squeeziness of R (code before)
• M3: Squeeziness of Q on states reachable via a given upper path $\pi$
• M4: M3 + Squeeziness of (upper/initial) path $\pi_u$
• M5: Squeeziness of (lower/final) path $\pi_l$
Experimental study

• For a program p we:
  – Randomly generated a sample T of 5,000 inputs from a suitable domain.
  – Generated mutants of p.
  – For mutant m (mutated statement s), input t in T:
    • Determine whether m and p have the same state after s.
    • Determine whether m and p have the same output.
  – A different ‘outcome’ denotes FEP.
Comparison made

• We compared our measures with the true (for the sample) probability of FEP:

\[ p(FEP) = \frac{\text{#tests with different state after } s \text{ but same output}}{\text{#number of tests with different state after } s} \]
Experimental subjects

• Three groups, all written in C:
  – 17 toy programs.
  – 10 functions from R.
  – 3 functions from GRETL (Gnu Regression, Econometrics and Time-Series Library).

• R functions: between 137 and 2397 LOC.

• GRETL functions: between 270 and 688 LOC.
Results: all programs

• Rank correlations:

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP1</td>
<td>0.715267</td>
</tr>
<tr>
<td>EXP2</td>
<td>0.699165</td>
</tr>
<tr>
<td>EXP3</td>
<td>0.955647</td>
</tr>
<tr>
<td>EXP4</td>
<td>0.948299</td>
</tr>
<tr>
<td>EXP5</td>
<td>0.031510</td>
</tr>
</tbody>
</table>
Results – real programs

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXP1</td>
<td>0.974459</td>
</tr>
<tr>
<td>EXP2</td>
<td>0.974459</td>
</tr>
<tr>
<td>EXP3</td>
<td>0.998526</td>
</tr>
<tr>
<td>EXP4</td>
<td>0.998526</td>
</tr>
<tr>
<td>EXP5</td>
<td>-0.001361</td>
</tr>
</tbody>
</table>
Toy programs (M2)
Real programs (M2)
Consequences

• Potential to use Information Theory based measures to predict the likelihood of FEP.

• In practice might:
  – Use as measure of testability (help us to decide e.g. how many tests cases to use?).
  – Try to cover e.g. a statement $s$ with a test that follows it with code that has low FEP.
  – Have more test cases for ‘hard to test’ parts of the code.

COW, Jan 2018
References

• The work is contained in:

• Other work includes:
Questions?