CS 5154: Software Testing

Coverage Criteria and Input Space Partitioning

Instructor: Owolabi Legunsen

Fall 2021
The next step in ISP require coverage criteria

• Step 1: Identify testable functions in your program
• Step 2: Find all input parameters
• Step 3: Model the input domain
• Step 4: Use a criterion to choose combination of values
• Step 5: Refine combinations of blocks into test inputs
But what is a coverage criterion?
Example 1: statement coverage criterion

• What elements of software should tests exercise?

loc

• What rule do we want to impose on those elements?

Cover all lines

• How do we check if the rule is satisfied?

did you cover them all?
Example 2: branch coverage criterion

• What elements of software should tests exercise?
  
  Control branches (if, while)

• What rule do we want to impose on those elements?
  
  Each branch must eval to T & F

• How do we check if the rule is satisfied?
  
  How many branches satisfy the rule?
  OK: How many test requirements are satisfied.
These questions point to general concepts

• What elements of software do should tests exercise?
  • Test requirements

• What rule(s) do we want to impose on those elements?
  • Coverage criteria

• How do we measure the degree to which the rules are met?
  • Coverage
Defining these three concepts generally

- **Test Requirement**: A software element that a test must satisfy or cover.

- **Coverage Criterion**: A rule or collection of rules that impose test requirements on a set of tests.

- **Coverage**: Given a set of test requirements \( TR \) for coverage criterion \( C \), a test set \( T \) satisfies \( C \) coverage if and only if for every test requirement \( tr \) in \( TR \), there is at least one test \( t \) in \( T \) such that \( t \) satisfies \( tr \).
We saw these concepts in CS5154 (indexOf)

Graph: abstract version

Initial Node: 1
Final Nodes: 4, 5

Edges
1 2
2 3
3 2
3 4
2 5

6 requirements for Edge-Pair Coverage
1. [1, 2, 3]
2. [1, 2, 5]
3. [2, 3, 4]
4. [2, 3, 2]
5. [3, 2, 3]
6. [3, 2, 5]

Test Paths
[1, 2, 5]
[1, 2, 3, 2, 5]
[1, 2, 3, 2, 3, 4]

Conventional:

\((\{1,2\}, \{2,3\}) = \{1,2,3\}\)
Question after last class

• Program P has six if statements. How many test requirements does the branch coverage criteria impose on tests for P?

- $2 \times 6$
- $2^6$

Do we need $2^6$ tests to satisfy branch coverage?
Question for you

Why do we need these general and fairly abstract definitions?
Do we always want 100% coverage?

• **Coverage Level**: The ratio of the number of test requirements satisfied by $T$ to the size of $TR$

• What if
  • we just started programming?
  • 100% coverage is too expensive?
  • we just want to get a sense of how we are doing?

• It makes sense to measure the degree of coverage
Is 100% coverage **always** possible?

- **Coverage**: Given a set of test requirements $TR$ for coverage criterion $C$, a test set $T$ satisfies $C$ coverage if and only if for every test requirement $tr$ in $TR$, there is at least one test $t$ in $T$ such that $t$ satisfies $tr$.

- What if some $tr$ is impossible to satisfy?
  - Example: dead code

- **An infeasible** test requirement is one that cannot be satisfied.
How to handle infeasible test requirements?

• Drop infeasible \( tr \) from TR

• Replace infeasible \( tr \) with less stringent TR

• Thoughts?
Quiz: Who said it?

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Coverage Criteria?
Are all criteria born equal?

• These tests satisfy 100% statement coverage but miss a fault

```java
int stringFactor(String i, int n) {
    if (i != null || n != 0)
        return i.length() / n;
    else
        return -1;
}
```

// Tests: ("happy", 2), (null, 0)

• Trick question: Will tests that satisfy 100% branch coverage find the fault?

• Teaser: “stronger” criteria can help, e.g., Multiple Condition Decision Coverage
Subsumption: comparing criteria “strength”

• **Criteria Subsumption** : Test criterion $C_1$ subsumes $C_2$ if and only if every set of test cases that satisfies $C_1$ also satisfies $C_2$

• Examples that we have seen in CS 5154:
  
  • Branch coverage subsumes statement coverage
  
  • Edge-Pair coverage subsumes edge coverage
Homework: Set relationships in subsumption

• Let C1 and C2 be two distinct coverage criteria whose sets of test requirements are TR(C1) and TR(C2), respectively. If C1 subsumes C2, which of the following is correct?

  - TR(C1) is a superset of TR(C2)
  - There is a many-to-one relation between TR(C1) and TR(C2)
  - There is a one-to-many relation between TR(C1) and TR(C2)
Questions about coverage criteria
So, how can criteria help us with ISP?

- triang() characteristic: relation of each side to 0

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b₁</th>
<th>b₂</th>
<th>b₃</th>
<th>b₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>greater than 1</td>
<td>equal to 1</td>
<td>equal to 0</td>
<td>less than 0</td>
</tr>
<tr>
<td>q₂</td>
<td>greater than 1</td>
<td>equal to 1</td>
<td>equal to 0</td>
<td>less than 0</td>
</tr>
<tr>
<td>q₃</td>
<td>greater than 1</td>
<td>equal to 1</td>
<td>equal to 0</td>
<td>less than 0</td>
</tr>
</tbody>
</table>

- How should we consider multiple partitions at the same time?
- What combination of blocks should we choose values from?
Idea 1: choose all combinations

- **All Combinations Coverage (ACoC) Criterion**: All combinations of blocks from all characteristics must be used.

- The number of resulting tests is the product of the number of blocks in each characteristic:

\[ \prod_{i=1}^{Q} (B_i) \]
ACoC for triang()

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b₁</th>
<th>b₂</th>
<th>b₃</th>
<th>b₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>greater than 1</td>
<td>equal to 1</td>
<td>equal to 0</td>
<td>less than 0</td>
</tr>
<tr>
<td>q₂</td>
<td>greater than 1</td>
<td>equal to 1</td>
<td>equal to 0</td>
<td>less than 0</td>
</tr>
<tr>
<td>q₃</td>
<td>greater than 1</td>
<td>equal to 1</td>
<td>equal to 0</td>
<td>less than 0</td>
</tr>
</tbody>
</table>

- Owolabi relabeled the blocks using same values in corresponding blocks for each characteristic for illustration purposes only:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>b₁</th>
<th>b₂</th>
<th>b₃</th>
<th>b₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>q₁</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>q₂</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>q₃</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>
ACoC tests for triang()

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

ACoC yields $4 \times 4 \times 4 = 64$ tests for triang()!

This is almost certainly more than we need

Only 8 tests have all sides greater than zero
Idea 2: use at least one value from each block

- **Each Choice Coverage (ECC) Criterion**: One value from each block for each characteristic must be used in at least one test case.

- The number of resulting tests is the largest number of blocks among all characteristics:
  \[
  \text{Max}_{i=1}^{Q}(B_i)
  \]
• These three tests satisfy ECC: \((A, 1, x), (B, 2, y), (A, 3, x)\)
• There are many ways to pick tests that satisfy ECC
• Do you see a weakness of ECC?
• ECC doesn’t require using a value with other values
  • e.g., \((A, 2, y)\) may reveal a fault
Idea 3: require pair-wise combinations

- **Pair-Wise Coverage (PWC) Criterion**: A value from each block for each characteristic must be combined with a value from every block for all other characteristics.

- The resulting number of tests is at least the product of the two largest characteristics:

\[
(\text{Max}_{i=1}^{Q}(B_i)) \times (\text{Max}_{j=1, j\neq i}^{Q}(B_j))
\]
PWC Example

- 5 combinations with A: (A, 1), (A, 2), (A, 3), (A, x), (A, y)
- 5 combinations with B: (B, 1), (B, 2), (B, 3), (B, x), (B, y)
- 6 combinations with q2 and q3 values: (1, x), (1, y), (2, x), (2, y), (3, x), (3, y)
- These 16 combinations can be combined in several ways:
  (A, 1, x) (A, 2, x) (A, 3, x) (A, -, y)
  (B, 1, y) (B, 2, y) (B, 3, y) (B, -, x)
  “-” : any value

Can still miss (A, 2, y)
Idea 4: extend pairwise to t-wise

• Problem(?): pair-wise only requires all two-combinations values
  • e.g., we may not choose (A, 2, y) on the previous slide

• The fault may be revealed by checking t-combinations

• t-Wise Coverage (TWC) Criterion: A value from each block for each group of t characteristics must be combined
Some questions about t-wise coverage

• What is the least number of resulting tests?

• What happens if $t$ is equal to the number of characteristics?

• Does t-wise coverage help much more than pair-wise coverage?
A note on the ISP criteria that we saw so far

They are mindless!
Idea 5: use domain knowledge

- **Base Choice Coverage (BCC) Criterion**:
  1. A base choice block is chosen for each characteristic, and a base test is formed by using the base choice for each characteristic.
  2. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other characteristic.

- The resulting number of tests: one base test + one test for each other block

\[ 1 + \sum_{i=1}^{Q} (B_i - 1) \]

- BCC allows using domain knowledge to select the base choice blocks
BCC Example

- Let ‘A’, ‘1’, and ‘x’ be the base choice blocks in $q_1$, $q_2$, and $q_3$ respectively.
- Base choice test: $(A, 1, x)$
- Additional tests: $(B, 1, x)$
  $(A, 2, x)$
  $(A, 3, x)$
  $(A, 1, y)$
Idea 6: what if I cannot choose 1 base choice?

- **Multiple Base Choice Coverage (MBCC) Criterion:**
  - At least one, and possibly more, base choice blocks are chosen for each characteristic, and base tests are formed by using each base choice for each characteristic at least once.
  - Subsequent tests are chosen by holding all but one base choice constant for each base test and using each non-base choice in each other characteristic.

- See textbook for the formula of upper bound of resulting tests
Recap on ISP coverage criteria

- All Combinations Coverage (ACoC)
- T-Wise Coverage (TWC)
- Each Choice Coverage (ECC)
- Base Choice Coverage (BCC)
- Pair-Wise Coverage (PWC)
- Multiple Base Choice Coverage (MBCC)

Which of these criteria subsume the other(s)?
Subsumption among ISP criteria

- **All Combinations Coverage (ACoC)**
- **Multiple Base Choice Coverage (MBCC)**
- **Base Choice Coverage (BCC)**
- **Each Choice Coverage (ECC)**
- **Pair-Wise Coverage (PWC)**
- **T-Wise Coverage (TWC)**

The diagram shows the hierarchy of coverage criteria, with ACoC being the most comprehensive, followed by MBCC, BCC, ECC, PWC, and TWC.
Summary: Input Space Partitioning

- Step 1: Identify testable functions in your program
- Step 2: Find all input parameters
- Step 3: Model the input domain
- Step 4: Use a criterion to choose combination of values
- Step 5: Refine combinations of blocks into test inputs
Next...

- Graph-based Model-Driven Test Design