Predicate Testing
CSE 5324
Summer 2017
Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Summary
Motivation

- **Predicates** are expressions that can be evaluated to a boolean value, i.e., true or false.
- Many decision points can be encoded as a predicate, i.e., which action should be taken under what condition?
- **Predicate-based testing** is about ensuring those predicates are implemented correctly.
Applications

• **Program-based**: Predicates can be identified from the branching points of the source code
  – e.g.: if ((a > b) || c) { ... } else { ... }

• **Specification-based**: Predicates can be identified from both formal and informal requirements as well as behavioral models such as FSM
  – “if the printer is ON and has paper then send the document for printing”
Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Summary
A **predicate** is an expression that evaluates to a Boolean value.

Predicates may contain:
- **Boolean** variables
- **Non-boolean** variables that are compared with the relational operators \(\{>, <, =, \geq, \leq, \neq\}\)
- **Boolean** function calls

The internal structure is created by **logical operators**:
- \(-, \land, \lor, \rightarrow, \oplus, \leftrightarrow\)
Logical operators

¬ – the negation operator
∧ – the and operator
∨ – the or operator
→ – the implication operator
⊕ – the exclusive or operator
⇔ – the equivalence operator
A **clause** is a predicate that does not contain any of the logical operators.

Example: \((a = b) \lor C \land p(x)\) has three clauses:
- a relational expression \((a = b)\),
- a boolean variable \(C\),
- a boolean function call \(p(x)\).
Predicate Faults

• An incorrect *Boolean* *operator* is used
• An incorrect *Boolean* *variable* is used
• *Missing* or *extra* *Boolean* *variables*
• An incorrect relational operator is used
• *Parentheses* *are* *used* *incorrectly*
Example

• Assume that \((a < b) \lor (c > d) \land e\) is a correct Boolean expression:
  
  – \((a < b) \land (c > d) \land e\)
  – \((a < b) \lor (c > d) \land f\)
  – \((a < b) \lor (c > d)\)
  – \((a = b) \lor (c > d) \land e\)
  – \((a = b) \lor (c \leq d) \land e\)
  – \((a < b \lor c > d) \land e\)
Predicate Testing

• Introduction
• Basic Concepts
• Predicate Coverage
• Program-Based Predicate Testing
• Summary
Abbreviations

- \( P \) is the set of predicates
- \( p \) is a single predicate in \( P \)
- \( C \) is the set of clauses in \( P \)
- \( C_p \) is the set of clauses in predicate \( p \)
- \( c \) is a single clause in \( C \)
Predicate Coverage (PC)

- The first (and simplest) two criteria require that each predicate and each clause be evaluated to both true and false and each clause be evaluated to both true and false.

- For each predicate \( p \), TR contains two requirements: \( p \) evaluates to true, and \( p \) evaluates to false.

- Example: \( p = ((a > b) \lor C) \land p(x) \)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>C</th>
<th>p(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

"decision coverage” in literature"
Predicate Coverage Example

\[ p = ((a < b) \lor D) \land (m \geq n \times o) \]

**Predicate = true**

- \( a = 5, \ b = 10, \ D = true, \ m = 1, \ n = 1, \ o = 1 \)
- \( (5 < 10) \lor true \land (1 \geq 1 \times 1) \)
- \( true \lor true \land TRUE \)
- \( true \)

**Predicate = false**

- \( a = 10, \ b = 5, \ D = false, \ m = 1, \ n = 1, \ o = 1 \)
- \( (10 < 5) \lor false \land (1 \geq 1 \times 1) \)
- \( false \lor false \land TRUE \)
- \( false \)
Clause Coverage (CC)

• For each clause $c$, TR contains two requirements: $c$ evaluates to true, and $c$ evaluates to false.

• Example: $((a > b) \lor C) \land p(x))$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>C</th>
<th>p(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>false</td>
</tr>
</tbody>
</table>

“condition coverage” in literature
Clause Coverage Example

\[ P = ((a < b) \lor D) \land (m \geq n \cdot o) \]

- **(a < b) = true**
  - \(a = 5, b = 10\)
  - \(a = 10, b = 5\)

- **D = true**
  - \(D = true\)
  - \(D = false\)

- **m \geq n \cdot o = true**
  - \(m = 1, n = 1, o = 1\)

- **m \geq n \cdot o = false**
  - \(m = 1, n = 2, o = 2\)

**Two tests**

1. \(a = 5, b = 10, D = true, m = 1, n = 1, o = 1\)
2. \(a = 10, b = 5, D = false, m = 1, n = 2, o = 2\)
Predicate vs. Clause Coverage

• Does predicate coverage subsume clause coverage? Does clause coverage subsume predicate coverage?

• Example: \( p = a \lor b \)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>( a \lor b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Naturally, we want to test both the predicate and individual clauses.
Predicate and Clause Coverage

• CC does not always ensure PC

• This is, we can satisfy CC without causing the predicate to be both true or false

• This is definitely not what we want!!
  – We need to come up with other approaches
Combinatorial Coverage (CoC)

• For each predicate $p$, TR has test requirements for the clauses in $p$ to evaluate to each possible combination of truth values.

• Example: $(a \lor b) \land c$

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>$(a \lor b) \land c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

CoC requires every possible combination.
Write all the clauses and the CoC of the given predicate:

\[ P = ((a > b) \lor C) \land p(x) \]
Combinatorial Coverage (CoC) – Exercise -1

P = ((a \succ b) \lor C) \land p(x)

P = (X \lor Y) \land Z

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Z is more important clause in this predicate than the others
What is the problem with combinatorial coverage?

*Combinatorial coverage is very expensive if we have multiple clauses in the predicate.*

- $2^n$ possibilities, which $n$ is number of independent clauses.
Active Clause

• Major clause
  – The clause which is being focused upon;

• Minor clause
  – All other clauses in the predicate (everything else).

• Determination:
  – A clause $c_i$ in predicate $p$, called the major clause, determines $p$ if and only if the values of the remaining minor clauses $c_i$ are such that changing $c_i$ changes the value of $p$ ($c_i$ Controls the behavior)

• In the previous example, if we chose $Z$ as Major clause, when it has value of “False”, it doesn’t matter what the other clauses are, but when it is “True”, it does matter what other clauses are
Determining Predicates

\( P = A \lor B \)

if \( B = true \), \( p \) is always true.
so if \( B = false \), \( A \) determines \( p \).
if \( A = false \), \( B \) determines \( p \).

\( P = A \land B \)

if \( B = false \), \( p \) is always false.
so if \( B = true \), \( A \) determines \( p \).
if \( A = true \), \( B \) determines \( p \).
Active Clause Coverage (ACC)

- For each predicate \( p \) and each major clause \( c \) of \( p \), choose minor clauses so that \( c \) determines \( p \). TR has two requirements for each \( c \): \( c \) evaluates to true and \( c \) evaluates to false.

Two of these requirements are identical, so we end up with three distinct test requirements for active clause coverage for the predicate \( a \lor b \), namely, \{\((a = \text{true}, b = \text{false})\), \((a = \text{false}, b = \text{true})\), \((a = \text{false}, b = \text{false})\)\}
Active Clause Coverage (ACC) – Example (1)

```java
public static void printHonorRollStatus(double cumulativeGPA,
    double termGPA, int creditsCompleted, boolean fullTimeStatus) {
    // Determine if the student is on the deans list.
    if ((creditsCompleted > 30) && (cumulativeGPA > 3.20)
        && (fullTimeStatus == true) && (termGPA > 2.0)) {
        System.out.println("You are on the dean's list.");
    } else if ((creditsCompleted > 30) && (cumulativeGPA > 3.70)
        && (fullTimeStatus == true) && (termGPA > 2.0)) {
        System.out.println("You are on the high honors dean's list.");
    } else if ((creditsCompleted > 30) && (cumulativeGPA > 2.0)
        && (fullTimeStatus == true) && (termGPA > 3.2)) {
        System.out.println("You are on the honor list.");
    } else {
```
### Active Clause Coverage (ACC) – Example (2)

<table>
<thead>
<tr>
<th>CC</th>
<th>CGPA</th>
<th>FT</th>
<th>TGPA</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>3.3</td>
<td>True</td>
<td>2.4</td>
<td>False</td>
</tr>
<tr>
<td>31</td>
<td>3.3</td>
<td>True</td>
<td>2.4</td>
<td>True</td>
</tr>
<tr>
<td>31</td>
<td>3.0</td>
<td>True</td>
<td>2.4</td>
<td>False</td>
</tr>
<tr>
<td>31</td>
<td>3.3</td>
<td>True</td>
<td>2.4</td>
<td>True</td>
</tr>
<tr>
<td>31</td>
<td>3.3</td>
<td>False</td>
<td>2.4</td>
<td>False</td>
</tr>
<tr>
<td>31</td>
<td>3.3</td>
<td>True</td>
<td>2.4</td>
<td>True</td>
</tr>
<tr>
<td>31</td>
<td>3.3</td>
<td>True</td>
<td>1.9</td>
<td>False</td>
</tr>
</tbody>
</table>

- The Green cells indicate active clauses,
- The Orange color cells indicate minor clauses
- We can test this with only 5 test cases
Finding Satisfying Values

How to choose values that satisfy a given coverage goal?
Example (1)

• Consider $p = (a \lor b) \land c$:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x &lt; y</td>
</tr>
<tr>
<td>b</td>
<td>done</td>
</tr>
<tr>
<td>c</td>
<td>List.contains(str)</td>
</tr>
</tbody>
</table>

How to choose values to satisfy predicate coverage?
Example (2)

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

\[ p = (a \lor b) \land c: \]

\[ \{1, 3, 5\} \times \{2, 4, 6, 7, 8\} \]
Suppose we choose \{1, 2\} to create test data.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 3  y = 5</td>
<td>done = true</td>
<td>List = [“Rat”, “cat”, “dog”] str = “cat”</td>
</tr>
<tr>
<td>x = 0, y = 7</td>
<td>done = true</td>
<td>List = [“Red”, “White”] str = “Blue”</td>
</tr>
</tbody>
</table>
Recap

• **Predicate testing** is about ensuring that each decision point is implemented correctly.
• If we flip the value of an active clause, we will change the value of the entire predicate.