Logic Coverage

CS 4501 / 6501
Software Testing

[Ammann and Offutt, “Introduction to Software Testing,” Ch. 8]
Structures for Criteria-Based Testing

Four structures for modeling software

- **Input space**
  - Source
  - Design
  - Specs
  - Use cases
  - Applied to: R--R

- **Graph**
  - Source
  - Design
  - Specs
  - Use cases
  - Applied to: ---R

- **Logic**
  - Source
  - Specs
  - FSMs
  - DNF
  - Applied to: RI-R

- **Syntax**
  - Source
  - Models
  - Integration
  - Inputs
  - Applied to: RIPR
Overview

• Logic coverage ensures that tests not only reach certain locations, but the internal state is infected by trying multiple combinations of truth assignments to the expressions.

• Covering logic expressions is required by the US Federal Aviation Administration for safety critical avionics software.

• Logical expressions can come from many sources:
  • Decisions in programs
  • FSMs and statecharts
  • Requirements

• Tests are intended to choose some subset of the total number of truth assignments to the expressions.
Logic Predicates and Clauses

- **Predicate**: An expression that evaluates to a boolean value
  - May contain
    - Boolean variable
    - Non-boolean variables that contain >, <, ==, >=, <=, !=
    - Boolean function calls
  - Created by the logical operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬</td>
<td>negation operator</td>
</tr>
<tr>
<td>∧</td>
<td>and operator</td>
</tr>
<tr>
<td>∨</td>
<td>or operator</td>
</tr>
<tr>
<td>→</td>
<td>implication operator</td>
</tr>
<tr>
<td>⊕</td>
<td>exclusive or operator</td>
</tr>
<tr>
<td>↔</td>
<td>equivalence operator</td>
</tr>
</tbody>
</table>

- **Clause**: A predicate with no logical operators
Example

Three clauses
A relational expression \((a = b)\)
A boolean variable \(C\)
A boolean-valued function \(p(x)\)

Logically equivalent

\[(a = b) \lor C \land f(x)\]

A predicate with logical operators

Three clauses
A relational expression \((a = b)\)
A boolean variable \(C\)
A boolean-valued function \(p(x)\)

Logically equivalent

\[((a = b) \lor C) \land ((a = b) \lor f(x))\]

A predicate with logical operators
Note on Predicates

- Most predicates have few clauses
- Sources of predicates
  - Decisions in program source code
  - Guards in finite state machines
  - Precondition in specifications

```
public boolean isSatisfactory() {
    if ((good && fast) || (good && cheap) || (fast && cheap))
        return true;
    else
        return false;
}
```

(good ∧ fast) ∨ (good ∧ cheap) ∨ (fast ∧ cheap)

gear = park ∧ button2 = true

pre: stack not full AND object reference parameter not null
¬ stackFull() ∧ newObj ≠ null
Note on Predicates

- Be careful when translating from English
  “I am interested in CS6501 and CS4501”
  \[(\text{Course} = \text{CS6501}) \text{ OR } (\text{course} = \text{CS4501})\]

From a study of 63 open source programs (>400,000 predicates), most predicates have few clauses [Ammann and Offutt]
- 88.5% have 1 clauses
- 9.5% have 2 clauses
- 1.35% have 3 clauses
- Only .65% have 4 or more

Try to keep the predicate simple and short
How? Refactor it
Short Circuit Evaluation

- Impacted by the order of operation
- Evaluate an expression or predicate until an outcome is known

\[((a = b) \lor C) \land f(x)\]

If \(f(x)\) is evaluated to \(T\), we evaluate \((a = b) \lor C\) which can be \(T\) or \(F\).

If \(f(x)\) is evaluated to \(F\), we stop. The outcome of the predicate is \(F\).
Short Circuit Evaluation

If `isHungry` is evaluated to T, we evaluate `(time == f(time))` which can be T or F.

If `isHungry` is evaluated to F, we stop. The outcome of the predicate is F.

Stop evaluating the predicate when we know the outcome.
Logic Coverage Criteria

• We use predicates in testing as follows:
  • Developing a model of the software as one or more predicates
  • Requiring tests to satisfy some combination of clauses

• 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)

- For each $p$ in $P$, TR contains two requirements:
  - $p$ evaluates to true
  - $p$ evaluates to false

```
p = ((a = b) \lor C) \land f(x)
```

Need 2 test cases to satisfy PC

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>C</th>
<th>f(x)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>T</td>
<td>3</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>F</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

- PC does **not** evaluate all the clauses, especially in the presence of short circuit evaluation

“Decision coverage”
Clause Coverage (CC)

- For each $c$ in $C$, TR contains two requirements:
  - $c$ evaluates to true
  - $c$ evaluates to false

```
p = ((a = b) \lor C) \land f(x)
```

(a = b) evaluates to $T, F$

C evaluates to $T, F$

f(x) evaluates to $T, F$

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>C</th>
<th>f(x)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Need 2 test cases to satisfy CC

- CC does **not** always ensure PC
- The simplest solution is to test all combinations
Combinatorial Coverage (CoC)

- Evaluate all possible combination of truth values

"Multiple Condition coverage"

\[ p = ((a = b) \lor C) \land f(x) \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>C</th>
<th>f(x)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</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>3</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Need \(2^N\) test cases to satisfy CoC, where \(N = \text{number of clauses}\)
Note on CoC

- Coc is simple and comprehensive
- But quite expensive
- $2^N$ tests, where $N$ is the number of clauses
  - Impractical for predicates with more than 3 or 4 clauses
- The literature has lots of suggestions – some confusing
- The general idea is simple:

  Test each clause that makes a big difference ...
  "active clause"
Revisit Coc Example

- Which clause makes a big difference

\[ p = ((a = b) \lor C) \land f(x) \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>C</th>
<th>f(x)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Active Clauses

• To really test the results of a clause, the clause should be the determining factor in the value of the predicate

• 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 $c_j$ are such that changing $c_i$ changes the value $p$
  • That is:
    • Major clause – the clause (being considered) that determines the predicate
    • Minor clause – all other clauses in the predicate

• This is considered to make the clause active
Determination

• **Goal**: Find tests for each clause when the clause determines the value of the predicate

• Determination: the conditions under which a clause solely determines the outcome of a predicate
  
  • Given a **major clause** $c_i$ in a predicate $p$, $c_i$ determines $p$ if the **minor clauses** $c_j (j \neq i)$
  
  • Major clause – “**active clause**” – controls the behavior

• Consider $p = a \lor b$
  
  • If $a = \text{true}$, the value of $b$ does not matter
  
  • If $b = \text{false}$, the value of $a$ is the determining factor in the value of the predicate
Revisit Coc Example (again)

- Which clause determines the predicate

\[ p = ((a = b) \lor C) \land f(x) \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>C</th>
<th>f(x)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>3</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>3</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Deriving Determination Predicates

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

\[ p_a = p_{a=true} \oplus p_{a=false} \]
\[ = (true \land (b \lor c)) \oplus (false \land (b \lor c)) \]
\[ = (b \lor c) \oplus false \]
\[ = b \lor c \]

\[ p_b = p_{b=true} \oplus p_{b=false} \]
\[ = (a \land (true \lor c)) \oplus (a \land (false \lor c)) \]
\[ = (a \land true) \oplus (a \land c) \]
\[ = a \oplus (a \land c) \]
\[ = a \land \neg c \]

\[ p_c = p_{c=true} \oplus p_{c=false} \]
\[ = (a \land (b \lor true)) \oplus (a \land (b \lor false)) \]
\[ = (a \land true) \oplus (a \land b) \]
\[ = a \oplus (a \land b) \]
\[ = a \land \neg b \]
Identifying Determination Using Truth Table

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

<table>
<thead>
<tr>
<th>row</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>p</th>
<th>( p_a )</th>
<th>( p_b )</th>
<th>( p_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td></td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Blank indicates \( F \)

Major clause: \( a \)
Identifying Determination Using Truth Table

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

### Major clause: \( b \)

<table>
<thead>
<tr>
<th>row</th>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( p )</th>
<th>( p_a )</th>
<th>( p_b )</th>
<th>( p_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td></td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
<td></td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Blank indicates \( F \)
Identifying Determination Using Truth Table

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

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

Blank indicates F

Major clause: c
What’s next?

• Use determination
• Apply logic coverage criteria to derive test requirements and design test cases
  • Active Clause Coverage (ACC)
  • General Active Clause Coverage (GACC)
  • Correlated Active Clause Coverage (CACC)
  • Restricted Active Clause Coverage (RACC)