CS 5154: Software Testing

Testing with Determination

Instructor: Owolabi Legunsen

Fall 2021
Recall the four software models in this course

- **Input Domains**
  - A: \{0, 1, >1\}
  - B: \{600, 700, 800\}
  - C: \{cs, ece, is, sds\}

- **Graphs**

- **Logic Expressions**
  - \((\neg x \mid \neg y) \& a \& b\)

- **Syntax**
  - if \((x > y)\)
    - \(z = x - y;\)
  - else
    - \(z = 2 \ast x;\)
We need criteria that are not as costly as CoC

• The general idea is quite simple:

  Test each clause independently from the other clauses

• But, getting the details right is hard
  • e.g., what exactly does “independently” mean?

• The book presents this idea as “making clauses active” ...
Active Clauses

• A **weakness of Clause Coverage**: values do not always make a difference

• **Values** \(((5 < 10) \lor true) \land (1 \geq 1 \times 1)\) for \(((a < b) \lor D) \land (m \geq n \times o)\)
  • Only the last clause counts!

• To really test the results of a clause, the clause should be the **determining factor** in what the predicate evaluates to
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 of $p$.

- Making $c_i$ determine $p$ is said to make the clause active.
CS 5154: Software Testing

Testing with Determination
(Active Clause Criteria)

Instructor: Owolabi Legunsen

Fall 2021
Recall: predicates and clauses

\[(a < b) \lor D) \land (m \geq n \times o)\]
Why do we care about clause coverage?

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

// Tests: ("happy", 2), (null, 0)
```
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 of $p$.

• Making $c_i$ determine $p$ is said to make the clause active.

• Condition under which $c_i$ determines $p$

\[
\forall c_j \in c \setminus c_i \exists \text{ assignment } (c_j) \text{ s.t. } p(c_i = \text{true}) \neq p(c_i = \text{false})
\]

where a assignment $(c_j)$ is $c_j = \text{true}$ or $c_j = \text{false}$.
The essence of testing with determination

1. Pick one clause in predicate $p$ to be the major clause $c_i$

2. Find conditions under which $c_i$ determines $p$

3. Find a test that makes $c_i$ true and a test that makes $c_i$ false

4. Repeat steps 1 to 3 for all other clauses in $p$

5. Eliminate redundant tests
Examples: determining predicates

**P = A \lor B**
- if \( B = \text{true} \), \( p \) is always true.
- so if \( B = \text{false} \), \( A \) determines \( p \).
- if \( A = \text{false} \), \( B \) determines \( p \).

<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>F</td>
<td>T</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>

**P = A \land B**
- if \( B = \text{false} \), \( p \) is always false.
- so if \( B = \text{true} \), \( A \) determines \( p \).
- if \( A = \text{true} \), \( B \) determines \( p \).

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a \land 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>F</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
More examples: determining predicates

\[ P = A \oplus B \]

if \( B = true \), \( A \) determines \( p \).

if \( B = false \), \( A \) determines \( p \).

so, \( A \) determines \( p \) for any \( B \).

\[ P = A \leftrightarrow B \]

if \( B = true \), \( A \) determines \( p \).

if \( B = false \), \( A \) determines \( p \).

so, \( A \) determines \( p \) for any \( B \).

| 1 | T | T | F |
| 2 | T | F | T |
| 3 | F | T | T |
| 4 | F | F | F |

| 1 | T | T | T |
| 2 | T | F | F |
| 3 | F | T | F |
| 4 | F | F | T |
Testing with determination 😊

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

- This goal is formalized in a **family of criteria** that have subtle, but very important, differences
Active Clause Coverage

• Step 1: For each $p$ in $P$ and each major clause $c_i$ in $Cp$, choose minor clauses $c_j, j \neq i$, so that $c_i$ determines $p$.

Active Clause Coverage (ACC): TR has two requirements for each $c_i: c_i$ evaluates to true and $c_i$ evaluates to false.

• ACC is a form of Multiple Condition Decision Coverage (MCDC)
• MCDC is required by the FAA for safety-critical software
Example on Active Clause Coverage

$$p = a \lor b$$

1) $a = \text{true}, b = \text{false}$
2) $a = \text{false}, b = \text{false}$
3) $a = \text{false}, b = \text{true}$
4) $a = \text{false}, b = \text{false}$

Duplicate

a is major clause

b is major clause

Duplicate
A formulaic way of determining predicates

• Finding values for minor clauses $c_j$ is easy for simple predicates

• How to find values for more complicated predicates?

• We need some “formula” that is easy to apply
A definitional way: when does $c$ determine $p$?

• Let $p_{c=true}$ be predicate $p$ with every occurrence of $c$ replaced by $true$

• Let $p_{c=false}$ be predicate $p$ with every occurrence of $c$ replaced by $false$

• To find values for the minor clauses, connect $p_{c=true}$ and $p_{c=false}$ with XOR

\[ p_c = p_{c=true} \oplus p_{c=false} \]

• After solving, $p_c$ describes exactly the values needed for $c$ to determine $p$
An example using the definitional way

• Let $p = a \lor (b \land c)$. What values of $b$ and $c$ will cause $a$ to determine $p$?

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

• “$!b \lor !c$” means $a$ determines $p$ when either $b$ or $c$ is false
Exercise 1: using the definitional way

• Let $p = a \lor b$. What values of $b$ will cause $a$ to determine $p$?

$$p = a \lor b$$

$p_a = p_{a=true} \oplus p_{a=false}$

$= (true \lor b) \oplus (false \lor b)$

$= true \oplus b$

$= \neg b$

<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>

• “$\neg b$” means $a$ determines $p$ when $b$ is false
• We obtained the same result from reasoning about the truth table
Exercise 2: using the definitional way

• Let \( p = a \leftrightarrow b \). What values of \( b \) will cause \( a \) to determine \( p \)?

\[
p = a \leftrightarrow b
\]

\[
p_a = p_{a=true} \oplus p_{a=false}
\]

\[
= (\text{true} \leftrightarrow b) \oplus (\text{false} \leftrightarrow b)
\]

\[
= b \oplus !b
\]

\[
= \text{true}
\]

• "true" means that \( a \) always determines \( p \)

• We obtained the same result from reasoning about the truth table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>a ( \leftrightarrow 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>F</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Is there a problem with Active Clause Coverage?

• Step 1: For each \( p \) in \( P \) and each major clause \( c_i \) in \( C_p \), choose minor clauses \( c_j, j \neq i \), so that \( c_i \) determines \( p \).

Active Clause Coverage (ACC) : TR has two requirements for each \( c_i : c_i \) evaluates to true and \( c_i \) evaluates to false.

• **Ambiguity** : Must minor clauses have the same values when the major clause is true and when the major clause is false?
Illustrating the ambiguity in ACC

- Recall: a determines p when \( \neg b \lor \neg c \), i.e., when either b or c is false

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

Major clause: a

- a = true, b = false, c = true
- a = false, b = false, c = false

Is this allowed?
Three options for resolving ACC ambiguity

• Minor clauses do not need to be the same

• Minor clauses must be the same

• Minor clauses allow the predicate to become both true and false
Option 1: minor clauses don’t need to be the same

• Step 1: For each \( p \) in \( P \) and each major clause \( c_i \) in \( C_p \), choose minor clauses \( c_j, j \neq i \), so that \( c_i \) determines \( p \).

• Step 2 (ACC): TR has two requirements for each \( c_i : c_i \) evaluates to true and \( c_i \) evaluates to false.

**General Active Clause Coverage (GACC)**: The values chosen for the minor clauses \( c_j \) do not need to be the same when \( c_i \) is true as when \( c_i \) is false, that is, \( c_j(c_i = \text{true}) = c_j(c_i = \text{false}) \) for all \( c_j \) OR \( c_j(c_i = \text{true}) \neq c_j(c_i = \text{false}) \) for all \( c_j \).
Problem: GACC doesn’t subsume Predicate Coverage

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>( a \leftrightarrow b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Major clause: a
Option 2: minor clauses do need to be the same

• Step 1: For each $p$ in $P$ and each major clause $c_i$ in $C_p$, choose minor clauses $c_j$, $j \neq i$, so that $c_i$ determines $p$.

• Step 2 (ACC): TR has two requirements for each $c_i$ : $c_i$ evaluates to true and $c_i$ evaluates to false.

Restricted Active Clause Coverage (RACC) : The values chosen for the minor clauses $c_j$ must be the same when $c_i$ is true as when $c_i$ is false, that is, it is required that $c_j(c_i = \text{true}) = c_j(c_i = \text{false})$ for all $c_j$. 
Exercise 3: using the definitional way

• Let $p = a \land (b \lor c)$. What values of $b$ and $c$ will cause $a$ to determine $p$?

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

\[
p_a = p_{a=true} \oplus p_{a=false}
\]

\[
= (true \land (b \lor c)) \oplus (false \land (b \lor c))
\]

\[
= (b \lor c) \oplus false
\]

\[
= (b \lor c)
\]

\[
= b \lor c
\]

• “$b \lor c$” means $a$ determines $p$ when either $b$ or $c$ is true
Example on Restricted Active Clause Coverage

Major clause: \( a, P_a = b \lor c \)

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

RACC (\( c_i = a \)) can only be satisfied by row pairs (1, 5), (2, 6), or (3, 7).

Only three pairs can be used.
Notes on RACC

• Does RACC subsume predicate and clause coverage?

• RACC was a common interpretation by developers for FAA

• Problem: RACC often leads to infeasible test requirements
Option 3: minor clauses **allow** predicate to be true and false

- **Step 1**: For each \( p \) in \( P \) and each major clause \( c_i \) in \( C_p \), choose minor clauses \( c_j \), \( j \neq i \), so that \( c_i \) determines \( p \).

- **Step 2 (ACC)**: TR has two requirements for each \( c_i : c_i \) evaluates to true and \( c_i \) evaluates to false.

**Correlated Active Clause Coverage (CACC)**: The values chosen for the minor clauses \( c_j \) must cause \( p \) to be true for one value of the major clause \( c_i \) and false for the other, that is, it is required that \( p(c_i = true) \neq p(c_i = false) \).
Example on CACC

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>(a \land (b \lor 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>T</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>F</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>

- **a determines P when \((b=true \text{ or } c = true)\)****
- **CACC \(c_i = a\)** can be satisfied by choosing any of rows 1, 2, 3 **AND** any of rows 5, 6, 7 – a total of nine pairs.
Notes on CACC

• CACC *implicitly* allows minor clauses to have different values

• CACC explicitly *subsumes* predicate coverage

• Does CACC subsume clause coverage?
Does CACC subsume clause coverage?

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>$a \land (b \lor c)$</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>T</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>F</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>

A determines $P$ when $(b=true \text{ or } c = true)$

CACC $(c_i = a)$ can be satisfied by choosing any of rows 1, 2, 3 AND any of rows 5, 6, 7 – a total of nine pairs.
Infeasibility

• Consider the predicate: \((a > b \land b > c) \lor c > a\)

• **Infeasible**: \((a > b) = true, (b > c) = true, (c > a) = true\) is infeasible

• As with other criteria, infeasible test requirements must be recognized and dealt with

• Recognizing infeasible test requirements is hard, and in general, undecidable
Subsumption among Logic coverage criteria

- Restricted Active Clause Coverage (RACC)
- Correlated Active Clause Coverage (CACC)
- General Active Clause Coverage (GACC)
- Combinatorial Clause Coverage (CoC)
- Clause Coverage (CC)
- Predicate Coverage (PC)
An end-to-end example with RACC

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

In sum, three separate pairs of rows can cause \(a\) to determine the value of \(p\), and only one pair each for \(b\) and \(c\).

Likewise, for clause \(c\), only one pair, TFT and TFF, cause \(c\) to determine the value of \(p\).

For clause \(b\), only one pair, TTF and TFF, cause \(b\) to determine the value of \(p\).

For clause \(c\), only one pair, TFT and TFF, cause \(c\) to determine the value of \(p\).

How many tests does RACC yield, compared to Combinatorial Clause Coverage?

Number of tests: 8
A more subtle exercise on determination

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

\[
p_a = p_{a=true} \oplus p_{a=false}
\]

\[
= ((true \land b) \lor (true \land \neg b)) \oplus ((false \land b) \lor (false \land \neg b))
\]

\[
= (b \lor \neg b) \oplus false
\]

\[
= true \oplus false
\]

\[
= true
\]

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

\[
p_b = p_{b=true} \oplus p_{b=false}
\]

\[
= ((a \land true) \lor (a \land \neg true)) \oplus ((a \land false) \lor (a \land \neg false))
\]

\[
= (a \lor false) \oplus (false \lor a)
\]

\[
= a \oplus a
\]

\[
= false
\]
A more subtle exercise on determination (2)

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

- \(a\) always determines the value of this predicate

- \(b\) never determines the value – \(b\) is irrelevant!

- So, why would anyone write a predicate like this?
Logic Coverage Summary

• Predicates are often very simple—in practice, most have <3 clauses
  • In fact, most predicates only have one clause!

• Only clause? PC is enough

• 2 or 3 clauses? CoC is practical

• Advantages of ACC criteria can be significant for large (no. of) predicates
  • CoC is impractical for predicates with many clauses
Next

• Applying Logic Coverage to source code