Active Clauses — Determination

Clause $c_i$ determines the value of its predicate when the other clauses have certain values.

If $c_i$ is changed, the value of the predicate changes.

$c_i$ is called the major clause.

Other clauses are minor clauses.

This is called making the clause active.
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 \( p \).

\[ P = A \land B \]
- If \( B = \text{false} \), \( p \) is always false.
- So if \( B = \text{true} \), \( A \) determines \( p \).
- If \( A = \text{true} \), \( B \) determines \( p \).

**Goal**: Find tests for each clause when the clause determines the value of the predicate.
Infeasibility & Subsumption (8.1.4)

Consider the predicate:

\[(a > b \land b > c)\]

Realize the abstract test tt into a concrete test by finding values for a, b, and c that create the truth assignments tt

\[a=9, b=7, c=5\]

Now consider the predicate:

\[(a > b \land b > c) \lor c > a\]

Realize the abstract test ttt into a concrete test by finding values for a, b, and c that create the truth assignments ttt

Impossible!

**Infeasible test requirements are recognized and ignored.**

Recognizing infeasible test requirements is generally **undecidable.**

(Thus usually done by hand)
Logic Criteria Subsumption

- Combinatorial Clause Coverage (COC)
  - Restricted Active Clause Coverage (RACC)
    - Correlated Active Clause Coverage (CACC)
  - Restricted Inactive Clause Coverage (RICC)
    - General Inactive Clause Coverage (GICC)
- General Active Clause Coverage (GACC)
- Clause Coverage (CC)
- Predicate Coverage (PC)
Determination Techniques

1. Informal by inspection
   This is what we’ve been doing
   Fast, but mistake-prone and does not scale–for experts

2. Tabular method
   Very simple by hand
   Few mistakes, slower, scales well to 5 or 6 clauses

3. Definitional method
   More mathematical
   Scales arbitrarily
### Tabular Method

Find pairs of rows in the truth table.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>P=a \land b</th>
<th>p_a</th>
<th>p_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For Pa, find a **pair** of rows where
- **b** is the same in both
- **a** is different
- **P** is different

For Pb, find a **pair** of rows where
- **a** is the same in both
- **b** is different
- **P** is different
Tabular Method

Find pairs of rows in the truth table.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>P=a ∧ b</th>
<th>p_a</th>
<th>p_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

For Pa, find a **pair** of rows where
- **b is the same** in both
- **a is different**
- **P is different**

For Pb, find a **pair** of rows where
- **a is the same** in both
- **b is different**
- **P is different**

Now do the same for “or”

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>P=a ∨ b</th>
<th>p_a</th>
<th>p_b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>✓</td>
</tr>
</tbody>
</table>
**In-class Exercise**

**Tabular method**

Use the tabular method to solve for $P_a$, $P_b$, and $P_c$. Give solutions as pairs of rows.

<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>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</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>T</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td>F</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>F</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>
### In-class Exercise

#### Tabular method

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>a ∧ (b ∨ c)</th>
<th>p&lt;sub&gt;a&lt;/sub&gt;</th>
<th>p&lt;sub&gt;b&lt;/sub&gt;</th>
<th>p&lt;sub&gt;c&lt;/sub&gt;</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>

*b & c are the same, a differs, and p differs … thus TTT and FTT cause a to determine the value of p*

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

Again, b & c are the same, so TTF and FTF cause a to determine the value of p

Finally, this third pair, TFT and FFT, also cause a to determine the value of p

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

Three separate pairs of rows can cause a to determine the predicate.

Only one pair each for b and c.
Definitional Method

Scales better (more clauses), requires more math

Definitional approach:

- $p_{c=true}$ is predicate $p$ with every occurrence of $c$ replaced by $true$
- $p_{c=false}$ is 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 exclusive OR

$$p_c = p_{c=true} \oplus p_{c=false}$$

After solving, $p_c$ describes exactly the values needed for $c$ to determine $p$
Definitional Method Examples

\[ p = a \lor b \]

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

\[ p = a \land b \]

Use the definitional approach to solve for \( p_a \)
Definitional Method Examples

**p = a ∨ b**

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

**p = a ∧ b**

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

*Use the definitional approach to solve for Pa*
# Definitional Method Examples

**Example 1:**

\[ p = a \lor b \]

\[ P_a = P_{a=\text{true}} \oplus P_{a=\text{false}} \]

\[ = (\text{true} \lor b) \text{ XOR } (\text{false} \lor b) \]

\[ = \text{true} \text{ XOR } b \]

\[ = \neg b \]

**Example 2:**

\[ p = a \land b \]

\[ P_a = P_{a=\text{true}} \oplus P_{a=\text{false}} \]

\[ = (\text{true} \land b) \text{ XOR } (\text{false} \land b) \]

\[ = b \oplus \text{false} \]

\[ = b \]

**Example 3:**

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

\[ P_a = P_{a=\text{true}} \oplus P_{a=\text{false}} \]

\[ = (\text{true} \lor (b \land c)) \text{ XOR } (\text{false} \lor (b \land c)) \]

\[ = \text{true} \oplus (b \land c) \]

\[ = \neg (b \land c) \]

\[ = \neg b \lor \neg c \]

"\text{NOT } b \lor \text{NOT } c" \text{ means either } b \text{ or } c \text{ must be false}
XOR Identity Rules

Exclusive-OR (xor, \(\oplus\)) means both cannot be true
That is, \(A\ xor\ B\) means

“\(A\ or\ B\) is true, but not both”

\[
\begin{align*}
p &= A \oplus A \land b \\
   &= A \land \neg b
\end{align*}
\]

\
\[
\begin{align*}
p &= A \oplus A \lor b \\
   &= \neg A \land b
\end{align*}
\]

with fewer symbols …

\[
\begin{align*}
p &= A\ xor\ (A\ and\ b) \\
   &= A\ and\ !b
\end{align*}
\]

\[
\begin{align*}
p &= A\ xor\ (A\ or\ b) \\
   &= !A\ and\ b
\end{align*}
\]
Repeated Variables

Definitional method yields the same tests no matter how the predicate is expressed

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

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

Only has 8 possible tests, not 64

Always use the simplest form of the predicate

and ignore contradictory truth table assignments
A More Subtle Example

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

\[ p_a = p_a=\text{true} \oplus p_a=\text{false} \]
\[ = ((\text{true} \land b) \lor (\text{true} \land \neg b)) \oplus ((\text{false} \land b) \lor (\text{false} \land \neg b)) \]
\[ = (b \lor \neg b) \oplus \text{false} \]
\[ = \text{true} \oplus \text{false} \]
\[ = \text{true} \]

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

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

\[ a \] always determines the value of this predicate
\[ b \] never determines the value - \( b \) is irrelevant!
Logic Coverage Summary

Predicates are often **very simple**—in practice, most have less than 3 clauses

In fact, most predicates only have one clause!

With only clause, PC is enough

With 2 or 3 clauses, CoC is practical

Advantages of ACC and ICC criteria significant for large predicates

CoC is impractical for predicates with many clauses

**Control** software often has many complicated predicates, with lots of clauses
In-class Exercise

Definitional method

\[ P = (a \mid b) \& (a \mid c) \& d \]

Use the definitional method to solve for \( P_a \)

First step: \(((T \mid b) \& (T \mid c) \& d) \text{xor} ((F \mid b) \& (F \mid c) \& d)\)
In-class Exercise

Definitional method

P = (a | b) & (a | c) & d

Use the definitional method to solve for Pa
First step: ((T | b) & (T | c) & d) xor ((F | b) & (F | c) & d)

\[ P_a = ((T | b) & (T | c) & d) \text{ xor } ((F | b) & (F | c) & d) \]
\[ = (T \& T \& d) \text{ xor } (b \& c \& d) \]
\[ = d \text{ xor } (b \& c \& d) \]

Using the identity: \( A \text{ xor } (A \& b) == A \text{ and } !b \)
\[ = d \& !(b \& c) \]
\[ = d \& (!b \mid !c) \]