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

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

<table>
<thead>
<tr>
<th>P = A ∨ B</th>
<th>P = A ∧ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>if ( B = true ), ( p ) is always true.</td>
<td>if ( B = false ), ( p ) is always false.</td>
</tr>
<tr>
<td>so if ( B = false ), ( A ) determines ( p ).</td>
<td>so if ( B = true ), ( A ) determines ( p ).</td>
</tr>
<tr>
<td>if ( A = false ), ( B ) determines ( p ).</td>
<td>if ( A = true ), ( B ) determines ( p ).</td>
</tr>
</tbody>
</table>
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 \(ttt\)

- 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)
  - General Active Clause Coverage (GACC)

- Restricted Inactive Clause Coverage (RICC)
  - General Inactive Clause Coverage (GICC)

- Clause Coverage (CC)
- Predicate Coverage (PC)
Making Clauses Determine a Predicate

Three 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>a</th>
<th>b</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>T</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For \( P_a \), find a **pair** of rows where
- \( b \) is the same in both
- \( a \) is different
- \( P \) is different

For \( P_b \), 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></th>
<th>a</th>
<th>b</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>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>F</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></th>
<th>a</th>
<th>b</th>
<th>(p = a \lor 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>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></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>$a \wedge (b \lor c)$</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-class Exercise

Tabular method

\( b \& c \) are the same, \( a \) differs, and \( p \) differs ... thus TTT and FTT cause \( a \) 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 \( p \)

Likewise, for clause \( c \), only one pair, TFT and TFF, cause \( c \) 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 \( c \).

<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>
Definitional Method

Scales better (more clauses), requires more math

Definitional approach:

• $p_{c=\text{true}}$ is predicate $p$ with every occurrence of $c$ replaced by $true$
• $p_{c=\text{false}}$ is predicate $p$ with every occurrence of $c$ replaced by $false$

To find values for the minor clauses, connect $p_{c=\text{true}}$ and $p_{c=\text{false}}$ with exclusive OR

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

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

\[ p = a \lor b \]

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

\[ p = a \land b \]

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

### \( p = a \lor b \)

<table>
<thead>
<tr>
<th>( p_a )</th>
<th>( p_{a=true} \oplus p_{a=false} )</th>
<th>( = (true \lor b) \text{ XOR } (false \lor b) )</th>
<th>( = true \text{ XOR } b )</th>
<th>( = ! b )</th>
</tr>
</thead>
</table>

### \( p = a \land b \)

| \( p_a \) | \( p_{a=true} \oplus p_{a=false} \) | \( = (true \land b) \oplus (false \land b) \) | \( = b \oplus false \) | \( = b \) |

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

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

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

Use the definitional approach to solve for \( p_a \)

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

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

\[ p = a \lor b \]

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

\[ p = a \land b \]

\[ p_a = p_a=true \oplus p_a=false \]
\[ = (true \land b) \oplus (false \land b) \]
\[ = b \oplus false \]
\[ = b \]

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

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

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

Exclusive-OR (xor, ⊕) means both cannot be true
That is, A xor B means
"A or B is true, but not both"

\[
p = A \oplus A \land b = A \land \neg b
\]

\[
p = A \oplus A \lor b = \neg A \land b
\]

with fewer symbols ...

\[
p = A \text{ xor } (A \text{ and } b) = A \text{ and } \neg b
\]

\[
p = A \text{ xor } (A \text{ or } b) = \neg A \text{ and } b
\]
Repeated Variables

The definitions in this chapter yield 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

Use the simplest form of the predicate, and ignore contradictory truth table assignments
A More Subtle Example

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

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

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

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

Definitional method

\[ P = (a \lor b) \land (a \lor c) \land d \]

Use the definitional method to solve for \( Pa \)
First step: \(((T \mid b) \land (T \mid c) \land d) \text{ xor } ((F \mid b) \land (F \mid c) \land d)\)

\[
Pa = ((T \mid b) \land (T \mid c) \land d) \text{ xor } ((F \mid b) \land (F \mid c) \land d) \\
= (T \land T \land d) \text{ xor } (b \land c \land d) \\
= d \text{ xor } (b \land c \land d)
\]

Using the identity: \( A \text{ xor } (A \land b) = A \land \neg b \)
\[
= d \land \neg (b \land c) \\
= d \land (\neg b \lor \neg c)
\]