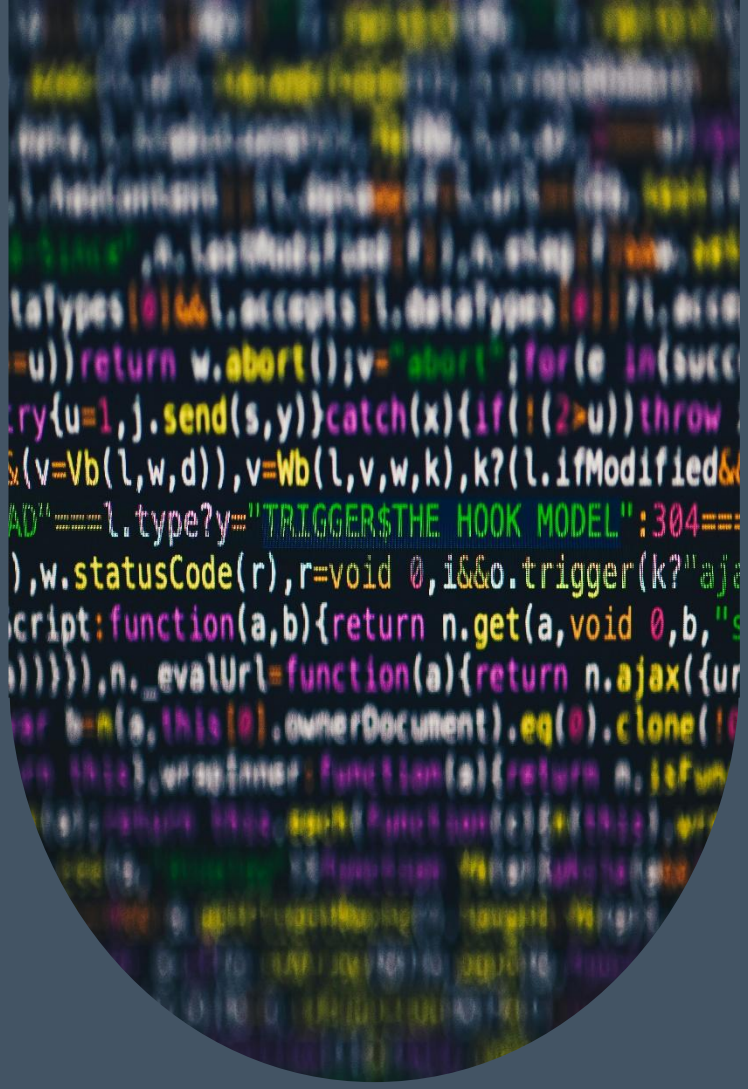
Games ENG II
Chapter 3,
Logic Coverage
Determination



Dr. Omer Ali

Adapted from
Introduction to Software Testing (Ammann & Offutt)

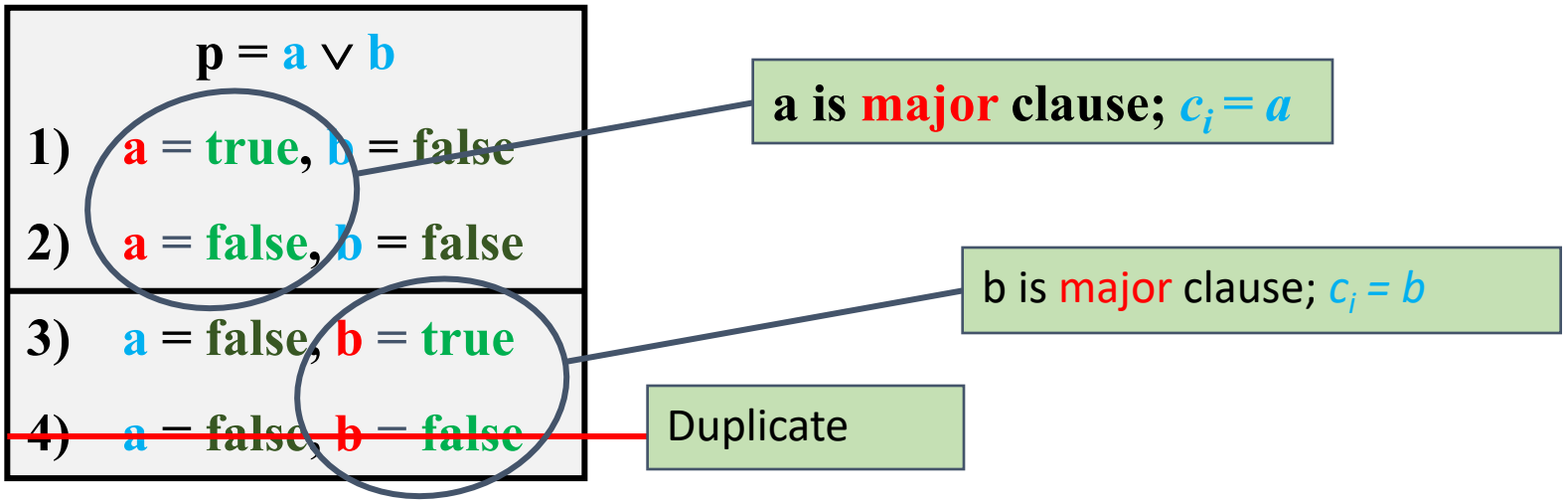
SETU
Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

setu.ie
INSPIRING FUTURES

Lecture 5 summary

Active Clause Coverage

Active Clause Coverage (ACC) : For each p in P and each major clause c_i in C_p, choose minor clauses c_j, j != i, so that c_i determines p. TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false.



- This is a form of MCDC (*Modified Condition/Decision Coverage*), which is required by the FAA for safety critical software
- **Ambiguity** : Do the minor clauses have to have the same values when the major clause is true and false?

General Active Clause Coverage

General Active Clause Coverage (GACC) : For each p in P and each major clause c_i in C_p, choose minor clauses c_j, j != i, so that c_i determines p. TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false.

Condition: 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,
$c_j(c_i = true) = c_j(c_i = false)$ for all c_j OR
$c_j(c_i = true)$!= $c_j(c_i = false)$ for all c_j

- This is complicated !
- It is possible to satisfy GACC without satisfying predicate coverage (PC) . i.e. p = a ↔ b
- We really want to cause predicates to be both true and false !

Restricted Active Clause Coverage

Restricted Active Clause Coverage (RACC) : For each p in P and each major clause c_i in C_p, choose minor clauses c_j, j != i, so that c_i determines p. TR has two requirements for each c_i: c_i evaluates to true and c_i evaluates to false.

Condition: 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 $c_j(c_i = true) = c_j(c_i = false)$ for all c_j.

- This has been a common interpretation by aviation developers
- RACC often leads to infeasible test requirements
- There is no logical reason for such a restriction

Correlated Active Clause Coverage

Correlated Active Clause Coverage (CACC) : For each p in P and each major clause c_i in C_p, choose minor clauses c_j, j != i, so that c_i determines p. TR has two requirements for each c_i: c_i evaluates to true and c_i evaluates to false.

Condition: 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 $p(c_i = true)$!= $p(c_i = false)$.

- A more recent interpretation
- Implicitly allows minor clauses to have different values
- Explicitly satisfies (subsumes) predicate coverage (PC)

Simple Example #1: GACC, RACC, CACC

	a	b	p = a ↔ b
1	T	T	T
2	T	F	F
3	F	T	F
4	F	F	T

NOTE: For simple predicates, RACC and CACC might be the same. But, they sure differ for complex (bigger) predicates

GACC:

Remember: 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

GACC for $c_i = a$: $T_{14} = \{TT, FF\}$;

GACC for $c_i = b$: $T_{14} = \{TT, FF\}$,

GACC = $T_{14} \cup T_{14} = \{TT, FF\}$; This satisfies CC but not PC

RACC:

Remember: The values chosen for the minor clauses c_j must be the same when c_i is true as when c_i is false

RACC for $c_i = a$: $T_{13} = \{TT, FT\}$

RACC for $c_i = b$: $T_{12} = \{TT, TF\}$

RACC = $T_{13} \cup T_{12} = \{TT, FT, TF\}$; This satisfies CC & PC

CACC:

Remember: 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

CACC for $c_i = a$: $T_{13} = \{TT, FT\}$;

CACC for $c_i = b$: $T_{12} = \{TT, TF\}$

CACC = $T_{13} \cup T_{12} = \{TT, FT, TF\}$; This satisfies CC & PC

Example #2 : CACC *vs.* RACC

	a	b	c	p = a ∧ (b ∨ c)
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F

major clause

CACC can be satisfied by choosing any of rows 1, 2, 3 AND any of rows 5, 6, 7 – a total of nine pairs

TR = any pair of {1,2,3} X {5,6,7}

	a	b	c	p = a ∧ (b ∨ c)
1	T	T	T	T
5	F	T	T	F
2	T	T	F	T
6	F	T	F	F
3	T	F	T	T
7	F	F	T	F

major clause

RACC can only be satisfied by one of the three pairs above

TR = any pair of { (1,5), (2,6), (3,7) }

Example #3 : CACC *vs.* RACC

Predicate: (**A** ∧ **B**) ∨ **C**

•For **A** to matter: *B must be TRUE, C must be FALSE*
Because then the decision depends on whether A is true.
•For **B** to matter: *A must be TRUE, C must be FALSE*
•For **C** to matter: *both A and B must be FALSE*
Because then the A && B part is false, and only C can change the outcome.

Active Clause	Minor Clause	Test 1 (TRUE test case)	Test 2 (False test case)
A	B = TRUE, C = FALSE	(A = TRUE, B= TRUE, C= FALSE) → p (TRUE)	(A = FALSE, B= TRUE, C= FALSE) → p (FALSE)
B	A = TRUE, C = FALSE	(A = TRUE, B= TRUE, C= FALSE) → p (TRUE)	(A = TRUE, B= FALSE, C= FALSE) → p (FALSE)
C	A = FALSE, B = FALSE	(A = FALSE, B= FALSE, C= TRUE) → p (TRUE)	(A = FALSE, B= FALSE, C= FALSE) → p (FALSE)

These satisfy both CACC and RACC, but how?

Example #4 : Real-world Payment authorization

An online payment gateway authorizes payment iff:

1. Card is valid (number & expiry & not blocked)
2. Strong authentication passed (not blocked) ,
3. And either **risk is low** or **manual override** flag is set by operations.

Let the clauses be:

C1 : cardValid

C2: authPassed

C3: riskLow (e.g. riskScore < 75)

C4: override (manual flag) – usually overridden for small purchases or frequent purchases with the same vendor etc.

Predicate: (**c1** $\wedge$ **c2**) $\wedge$ (**c3** $\vee$ **c4**) $\xrightarrow{\text{TRUE}}$ Payment approved

Example #4 : Real-world Payment authorization

C1 : cardValid
C2: authPassed
C3: riskLow (e.g. riskScore < 75)
C4: override (manual flag)

Predicate: (**c1** ∧ **c2**) ∧ (**c3** ∨ **c4**) —TRUE→ Payment approved

Predicate Coverage (required predicate to be TRUE and FALSE at least once)
Test 1 (**True** case): (c1,c2,c3,c4) = (T,T,T,F) → approve (True)
Test 2 (**False** case): (c1,c2,c3,c4) = (F,F,F,F) → approve (False)

Clause Coverage (requires the clause to flip across set)
Test 1 (**True** case): (c1,c2,c3,c4) = (T,T,T,F) → approve (True)
Test 2 (**False** case): (c1,c2,c3,c4) = (F,T,T,F) → approve (False)

Test 1 (**True** case): (c1,c2,c3,c4) = (T,T,T,F) → approve (True)
Test 2 (**False** case): (c1,c2,c3,c4) = (T,F,T,F) → approve (False)

Test 1 (**True** case): (c1,c2,c3,c4) = (T,T,T,F) → approve (True)
Test 2 (**False** case): (c1,c2,c3,c4) = (T,T,F,F) → approve (False)

Test 1 (**True** case): (c1,c2,c3,c4) = (T,T,F,T) → approve (True)
Test 2 (**False** case): (c1,c2,c3,c4) = (T,T,F,F) → approve (False)

Example #4 : Real-world Payment authorization

Predicate: (**c1** $\wedge$ **c2**) $\wedge$ (**c3** $\vee$ **c4**) —TRUE→ Payment approved

Now determining Active Clauses

Active Clause	Minor Clause	Minor Clause	Minor Clause	Predicate
C1	C2 = TRUE	C3 = FALSE	C4 = TRUE	TRUE
C2	C1 = TRUE	C3 = FALSE	C4 = TRUE	TRUE
C3	C1 = TRUE	C2 = TRUE	C4 = FALSE	TRUE
C4	C1 = TRUE	C2 = TRUE	C3 = FALSE	TRUE

Example #4 : Real-world Payment authorization

Predicate: (**c1** $\wedge$ **c2**) $\wedge$ (**c3** $\vee$ **c4**) —TRUE→ Payment approved

GACC Pairs (where minor clauses ***may*** differ)

C1 major (fix c2 = TRUE), make right TRUE via **override flag c4**
(T, T, -, T) **Vs** (F, T, -, T) → p flips

C2 major (fix c1 = TRUE), make right TRUE via **override flag c4**
(T, T, -, T) **Vs** (T, F, -, T) → p flips

C3 major (fix c1, c2 = TRUE), now right side is handled by c3, so set c4 = FALSE
(T, T, T, F) **Vs** (T, T, F, F) → p flips

C4 major (fix c1, c2 = TRUE), now right side is handled by c4, so set c3 = FALSE
(T, T, F, T) **Vs** (T, T, F, F) → p flips

Example #4 : Real-world Payment authorization

Predicate: (**c1** ∧ **c2**) ∧ (**c3** ∨ **c4**) —TRUE→ Payment approved

RACC Pairs (where minor clauses ***are identical***)

C1 major (fix c2 = TRUE), make right TRUE via **override flag c4**
(T, T, *, T) **Vs** (F, T, *, T) → p flips

C2 major (fix c1 = TRUE), make right TRUE via **override flag c4**
(T, T, *, T) **Vs** (T, F, *, T) → p flips

C3 major (fix c1, c2 = TRUE), now right side is handled by c3, so set c4 = FALSE
(T, T, T, F) **Vs** (T, T, F, F) → p flips

C4 major (fix c1, c2 = TRUE), now right side is handled by c4, so set c3 = FALSE
(T, T, F, T) **Vs** (T, T, F, F) → p flips

Bonus Points (+3) – Can we also call this CACC, if so, why?

Logic Coverage Summary

Coverage Type	Ensures	Goals	Remarks
PC	Predicate is TRUE & FALSE	Very coarse	May miss logic errors
CC	Each clause TRUE & FALSE	Better	Doesn't ensure influence
GACC	Each clause determines predicate (loosely)	Easy to achieve	Others can vary
CACC	Clause determines predicate, result changes	Best practical coverage	Used in MC/DC
RACC	Only major clause changes	Very strict	Sometimes infeasible

Calculating Determination

Calculating Determination

- We have established that ***major clauses*** impact ***predicates*** (TRUE / FALSE).
- We have also seen how to set ***minor clause*** values for different coverage conditions (some will affect predicate, others won't).
- Finding values for minor clauses is easy for simple predicates, but can be very difficult for complex problems.
- Therefore, a ***Definitional*** approach (based on algebra) is used for calculating determinants.

  P_c = **true** → is predicate ***p*** with every occurrence of clause ***c*** replaced by **true**
- P_c = **false** → is predicate ***p*** with every occurrence of clause ***c*** replaced by **false**
- To find values for minor clauses, p_c = pc=true **XOR** pc=false

Determination Example 1

- Consider p = a ∨ b
- $P_a = p_a$ = true ⊕ p_a = false
- P_a = (true ∨ b) ⊕ (false ∨ b)
- P_a = true ⊕ b
- P_a = ¬b

Thus ***a*** determines ***p*** when ***b = false***

Since XOR will only give output when both of its inputs are different. So, a and b values must be opposite.

Domination Identity used: (T ∨ x)=T; *Identity* $(F \vee x) = x$.

Determination Example 2

- Consider p = a ∧ b
- $P_a = p_a$ = true ⊕ p_a = false
- P_a = (true ∧ b) ⊕ (false ∧ b)
- P_a = b ⊕ false
- P_a = b

Thus ***a*** determines ***p*** when ***b = true***

In this case, XOR is already satisfied , as we have a true on left hand side and false on the right hand side of XOR.

Domination Identity used: (T ∨ x)=T; *Identity* $(F \vee x) = x$.

Determination Example 3

- Consider p = a ∨ (b ∧ c)
- $P_a = p_a$ = true $\oplus$ p_a = false
- P_a = (true ∨ (b ∧ c)) $\oplus$ (false ∨ (b ∧ c))
- P_a = true $\oplus$ b ∧ c
- P_a = ¬ (b ∧ c)
- P_a = ¬ b ∨ ¬ c → De Morgan's Law used.

Thus ***a*** determines ***p*** when ***b = false*** or **c=false**

Domination Identity used: (T ∨ x)=T; *Identity* $(F \vee x) = x$.

Tabular Method for Determination

Row	a	b	c	a V (b Λ c)	P_a	P_b	P_c
1	T	T	T	T			
2	T	T	F	T			
3	T	F	T	T			
4	T	F	F	T			
5	F	T	T	T			
6	F	T	F	F			
7	F	F	T	F			
8	F	F	F	F			

Tabular Method for Determination

Row	a	b	c	a V (b Λ c)	P_a	P_b	P_c
1	T	T	T	T			
2	T	T	F	T	✓		
3	T	F	T	T			
4	T	F	F	T			
5	F	T	T	T			
6	F	T	F	F	✓		
7	F	F	T	F			
8	F	F	F	F			

Select ***a*** as the major clause, then choose values for minor clauses ***b*** and ***c*** such that changing only ***a*** changes ***p***, thus ***a*** determines ***p***

Tabular Method for Determination

Row	a	b	c	a V (b Λ c)	P_a	P_b	P_c
1	T	T	T	T			
2	T	T	F	T	✓		
3	T	F	T	T	✓		
4	T	F	F	T			
5	F	T	T	T			
6	F	T	F	F	✓		
7	F	F	T	F	✓		
8	F	F	F	F			

Select ***a*** as the major clause, then choose values for minor clauses ***b*** and ***c*** such that changing only ***a*** changes ***p***, thus ***a*** determines ***p***

Tabular Method for Determination

Row	a	b	c	a V (b Λ c)	P_a	P_b	P_c
1	T	T	T	T			
2	T	T	F	T	✓		
3	T	F	T	T	✓		
4	T	F	F	T	✓		
5	F	T	T	T			
6	F	T	F	F	✓		
7	F	F	T	F	✓		
8	F	F	F	F	✓		

Select ***a*** as the major clause, then choose values for minor clauses ***b*** and ***c*** such that changing only ***a*** changes ***p***, thus ***a*** determines ***p***

Tabular Method for Determination

Row	a	b	c	a V (b Λ c)	P_a	P_b	P_c
1	T	T	T	T			
2	T	T	F	T	✓		
3	T	F	T	T	✓		
4	T	F	F	T	✓		
5	F	T	T	T		✓	
6	F	T	F	F	✓		
7	F	F	T	F	✓	✓	
8	F	F	F	F	✓		

Select ***b*** as the major clause, then choose values for minor clauses ***a*** and ***c*** such that changing only ***b*** changes ***p***, thus ***b*** determines ***p***

Tabular Method for Determination

Row	a	b	c	a V (b Λ c)	P_a	P_b	P_c
1	T	T	T	T			
2	T	T	F	T	✓		
3	T	F	T	T	✓		
4	T	F	F	T	✓		
5	F	T	T	T		✓	✓
6	F	T	F	F	✓		✓
7	F	F	T	F	✓	✓	
8	F	F	F	F	✓		

Select ***c*** as the major clause, then choose values for minor clauses ***a*** and ***b*** such that changing only ***c*** changes ***p***, thus ***c*** determines ***p***

Determination Example 4

- Consider **p** = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)
- $P_a = p_a$ = true ⊕ p_a = false
- P_a = (true ∧ **b**) ∨ **(**¬true ∧ ¬ b ∧ c) ⊕ (false ∧ **b**) ∨ (¬false ∧ ¬ b ∧ c)
- P_a = b ∨ **(**false ∧ ¬b ∧ c) ⊕ false ∨ **(**true ∧ ¬b ∧ c)
- P_a = b ⊕ ¬ b ∧ c
- P_a = b ∨ c

Domination Identity used: (T ∨ x)=T; *Identity* $(F \vee x) = x$.

Identity Proof

$$
\begin{aligned}
b \oplus (\neg b \wedge c) &= \big(b \wedge \neg(\neg b \wedge c)\big) \;\vee\; \big(\neg b \wedge (\neg b \wedge c)\big) \quad \text{(XOR expansion)} \\
&= \big(b \wedge (b \vee \neg c)\big) \;\vee\; \big(\neg b \wedge c\big) \quad \text{(De Morgan, idempotence)} \\
&= b \;\vee\; (\neg b \wedge c) \quad \text{(absorption: } x \wedge (x \vee y) = x\text{)} \\
&= (b \vee \neg b) \wedge (b \vee c) \;=\; \mathrm{T} \wedge (b \vee c) \;=\; b \vee c \quad \text{(distributivity, complement, identity).}
\end{aligned}
$$

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	
3	T	F	T	F	
4	T	F	F	F	
5	F	T	T	F	✓
6	F	T	F	F	
7	F	F	T	T	
8	F	F	F	F	

Select inputs such that a changes, b and c do not change, and p changes. Thus a determines p when b ∧ c

We can also identify the pair as (1,5)

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	✓
3	T	F	T	F	
4	T	F	F	F	
5	F	T	T	F	✓
6	F	T	F	F	✓
7	F	F	T	T	
8	F	F	F	F	

Select inputs such that a changes, b and c do not change, and p changes. Thus a determines p when b ∧ ¬c

We can also identify the pair as (2,6)

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	✓
3	T	F	T	F	✓
4	T	F	F	F	
5	F	T	T	F	✓
6	F	T	F	F	✓
7	F	F	T	T	✓
8	F	F	F	F	

Select inputs such that a changes, b and c do not change, and p changes. Thus a determines p when ¬ b ∧ c

We can also identify the pair as (3,7)

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	✓
3	T	F	T	F	✓
4	**T**	F	F	**F**	
5	F	T	T	F	✓
6	F	T	F	F	✓
7	F	F	T	T	✓
8	**F**	F	F	**F**	

Select inputs such that a changes, b and c do not change, but p DOES NOT change. Thus a DOES NOT determines p when ¬ b ∧ ¬ c

Notice the predicate output DOES NOT flips

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

- For GACC, select a pair of conditions (test cases) where the value of major clause changes (the value of minor clause and ***p*** may or may not change)

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	✓
3	T	F	T	F	✓
4	T	F	F	F	
5	F	T	T	F	✓
6	F	T	F	F	✓
7	F	F	T	T	✓
8	F	F	F	F	

GACC Pairs:
(1,5) or (1,6) or (1,7)
(2,5) or (2,6) or (2,7)
(3,5) or (3,6) or (3,7)
because p may or may change

Active Clause Coverage Criteria Summary

Criterion	Major clause determines *p*	Major clause changes *value*	*Change* major clause changes *p*	Minor clauses are held the *same*
GACC	✓	✓		
CACC	✓	✓	✓	
RACC	✓	✓	✓	✓

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

- For CACC, select a pair of conditions (test cases) where the value of major clause changes and the value of ***p*** changes(the value of minor clause may or may not change)

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	✓
3	T	F	T	F	✓
4	T	F	F	F	
5	F	T	T	F	✓
6	F	T	F	F	✓
7	F	F	T	T	✓
8	F	F	F	F	

CACC Pairs:
(1,5) or (1,6)
(2,5) or (2,6) or (2,7)
(3,5) or (3,6) or (3,7)
but not (3,5) or (3,6) because the value of *p* DOES NOT change

Active Clause Coverage Criteria Summary

Criterion	Major clause determines *p*	Major clause changes *value*	*Change* major clause changes *p*	Minor clauses are held the *same*
GACC	✓	✓		
CACC	✓	✓	✓	
RACC	✓	✓	✓	✓

Tabular Method for Determination

In this example ***a*** is considered as the major clause. The task is to determine all the conditions for the ***minor clauses*** where major clause determines the predicate

- For RACC, select a pair of conditions (test cases) where the value of major clause changes and the value of ***p*** changes(and the value of minor clause are held constant)

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_a
1	T	T	T	T	✓
2	T	T	F	T	✓
3	T	F	T	F	✓
4	T	F	F	F	
5	F	T	T	F	✓
6	F	T	F	F	✓
7	F	F	T	T	✓
8	F	F	F	F	

RACC Pairs:
(1,5) or (2,6) or (3,7)

Active Clause Coverage Criteria Summary

Criterion	Major clause determines *p*	Major clause changes *value*	*Change* major clause changes *p*	Minor clauses are held the *same*
GACC	✓	✓		
CACC	✓	✓	✓	
RACC	✓	✓	✓	✓

Inactive Clause Coverage

It is literally taking the ***opposite approach*** where ***major clauses do not affect the predicates***

Why Bother? Sometimes it is useful for testing safety interlock systems (for example, electric grid, reactors, industries with chemical reactions) to ensure that during certain circumstances a control variable ***does not*** have any effect on operation.

DEFINITION

Inactive Clause Coverage (ICC) – For each p in P and each major clause c_i in C_p, choose minor clauses c_j (j!=i) such that c_i *does not* determine p. *TR* has four requirements for c_i: (1) c_i evaluates to true with p true, (2) c_i evaluates to false with p true, (3) c_i evaluates to true with p false, and (4) c_i evaluates to false with p false.

Types of Inactive Clause Coverage

DEFINITION

General Inactive Clause Coverage (GICC) – For each p in P and each major clause c_i in C_p, choose minor clauses c_j (j!=i) such that c_i *does not* determine p. The values chosen for minor clauses c_j *do not* need to be the same when c_i is true as when c_i is false.

DEFINITION

Restricted Inactive Clause Coverage (RICC) – For each p in P and each major clause c_i in C_p, choose minor clauses c_j (j!=i) such that c_i *does not* determine p. The values chosen for minor clauses c_j *must be* the same when c_i is true as when c_i is false.

Tabular Method for Determination

In this example ***c*** is considered as the major clause.

- For GICC, select a pair of conditions (test cases) where the major clause DOES NOT determine ***p***, the value of the major clause changes, and the value of ***p*** DOES NOT change

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_c
1	T	T	T	T	
2	T	T	F	T	
3	T	F	T	F	
4	T	F	F	F	
5	F	T	T	F	
6	F	T	F	F	
7	F	F	T	T	✓
8	F	F	F	F	✓

GICC Pairs:
(1,2) or (3,4)
(3,6) or (5,4)
(5,6)

But because we want to choose ICC, we will skip (7,8) as those are ACC

✓ Active coverage when ***c*** is chosen as major clause, because it affects predicate ***p***

Tabular Method for Determination

In this example ***c*** is considered as the major clause.

- For RICC, select a pair of conditions (test cases) where the major clause DOES NOT determine ***p***, the value of the major clause changes, and the value of ***p*** DOES NOT change, and the minor clauses are the same.

Row	a	b	c	p = (a ∧ b) ∨ (¬ a ∧ ¬ b ∧ c)	P_c
1	T	T	T	T	
2	T	T	F	T	
3	T	F	T	F	
4	T	F	F	F	
5	F	T	T	F	
6	F	T	F	F	
7	F	F	T	T	✓
8	F	F	F	F	✓

RICC Pairs:
(1,2) or (3,4)
or
(5,6)

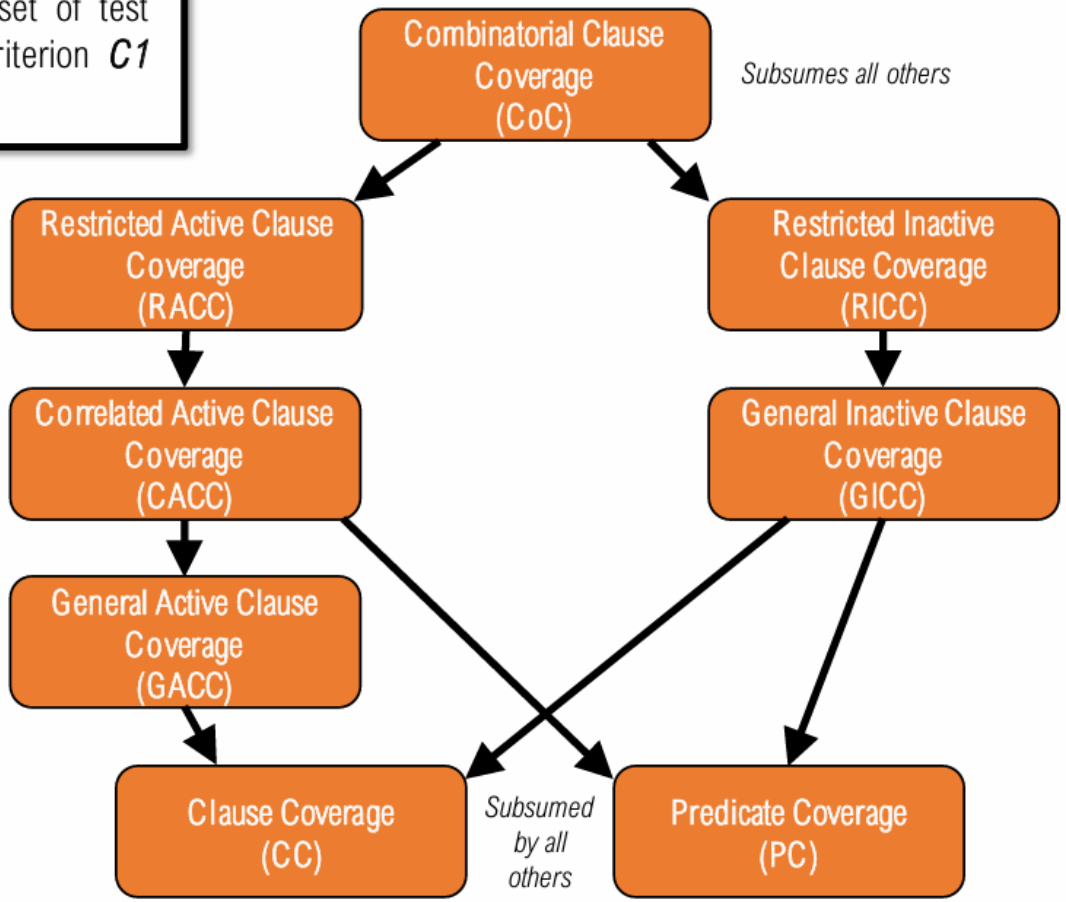
But because we want to choose ICC, we will skip (7,8) as those are ACC

✓ Active coverage when ***c*** is chosen as major clause, because it affects predicate ***p***

Logic Criteria Subsumption

DEFINITION

A test criterion ***C1* subsumes *C2*** if and only if every set of test cases that satisfies criterion ***C1*** also satisfies ***C2***



Game Level – Logic Coverage Example

Scenario

- Player can use Dash if: in arena, not stunned, and (enough stamina OR has power-up).
- Game-master override can grant Dash regardless of other conditions.
- We use this to show major/minor clauses and determination.

Decision and Clauses

- Decision under test: ***canUseDash***
- Clauses:
- C1: inArena (must be true)
- C2: notStunned (must be true)
- C3: (stamina >= need) OR hasPowerup
- C4: gmOverride (bypass)

Clause Map

Label	Variable(s)	Meaning	Typical role
C1	inArena	Player is in gameplay scene	Minor
C2	notStunned	Player can act	Minor
C3	stamina>=need || hasPowerup	Resource requirement satisfied	Major (focus)
C4	gmOverride	Bypass authorization	Minor/Major (alternate)

C++ (1/3): State and constants

```
struct PlayerState {
    bool inArena;         // C1
    bool notStunned;      // C2
    int  stamina;         // used with staminaNeeded
    bool hasPowerup;      // part of C3
    bool gmOverride;      // C4
};
static const int staminaNeeded = 25;
```

C++ (2/3): Decision under test

```
bool canUseDash(const PlayerState& s) {
    bool c1 = s.inArena;
    bool c2 = s.notStunned;
    bool c3 = (s.stamina >= staminaNeeded) || s.hasPowerup;
    bool c4 = s.gmOverride;
    return ((c1 && c2) && c3) || c4;
}
```

C++ (3/3): Test harness

```
// Pair A: Make C3 the major by fixing the minors
PlayerState minorsFixed{true,true,0,false,false}; // C1=T, C2=T, C4=F

PlayerState A1 = minorsFixed;                  // c3=F  -> expect false
PlayerState A2 = minorsFixed; A2.stamina=25;  // c3=T  -> expect true

// Pair B: Make C4 the major by forcing left side false
PlayerState leftFalse{true,false,0,false,false}; // C2=F ⇒ left side = F

PlayerState B1 = leftFalse; B1.gmOverride=false; // expect false
PlayerState B2 = leftFalse; B2.gmOverride=true;  // expect true
```

How a tester approaches this

- Identify the decision and enumerate clauses.
- Pick a major; set minors supportive and fixed.
- Flip only the major; the outcome should flip (determination).
- Test another major under conditions that make others inert.
- Add an inactive check for a clause that should not matter in a mode.

Active clause demo: C3 determines

Case	C1 inArena	C2 notStunned	stamina	hasPowerup	C4 override	Expected
A1	T	T	0	F	F	FALSE
A2	T	T	25	F	F	TRUE

Active clause demo: C4 determines

Case	C1 inArena	C2 notStunned	stamina	hasPowerup	C4 override	Expected
B1	T	F	0	F	F	FALSE
B2	T	F	0	F	T	TRUE

Inactive check: stamina is inert when hasPowerup = TRUE

Case	C1	C2	stamina	hasPowerup	C4	Expected
I1	T	T	0	T	F	TRUE
I2	T	T	25	T	F	TRUE

Common bug patterns to watch

- Wrong grouping/precedence; missing parentheses.
- Bitwise &/ instead of logical &&.
- Threshold error: >= vs >
- Forgotten reset: gmOverride remains true.
- Short-circuit surprises with side-effecting calls.

Exercise

- Exercise: make hasPowerup the major; find a two-test flip pair.
- Exercise: find minors that make inArena inactive and show both outcomes.

To be submitted on Moodle before 11 November 2025