Introduction to Software Testing
Logic Coverage (Ch. 8.1.4 & 8.1.5)

Software Testing & Maintenance

SWE 437

http://go.gmu.edu/swe437

Dr. Brittany Johnson-Matthews

(Dr. B for short)

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

$P = A \vee B$	$P = A \wedge B$
if ***B* = *true***, *p* is always true.	if ***B* = *false***, *p* is always false.
so if ***B* = *false***, *A* determines *p*.	so if ***B* = *true*,** *A* determines *p*.
if ***A* = *false*,** *B* determines *p*.	if ***A* = *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:

$(a > b \wedge b > c)$

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:

$(a > b \wedge b > c) \vee c > a$

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

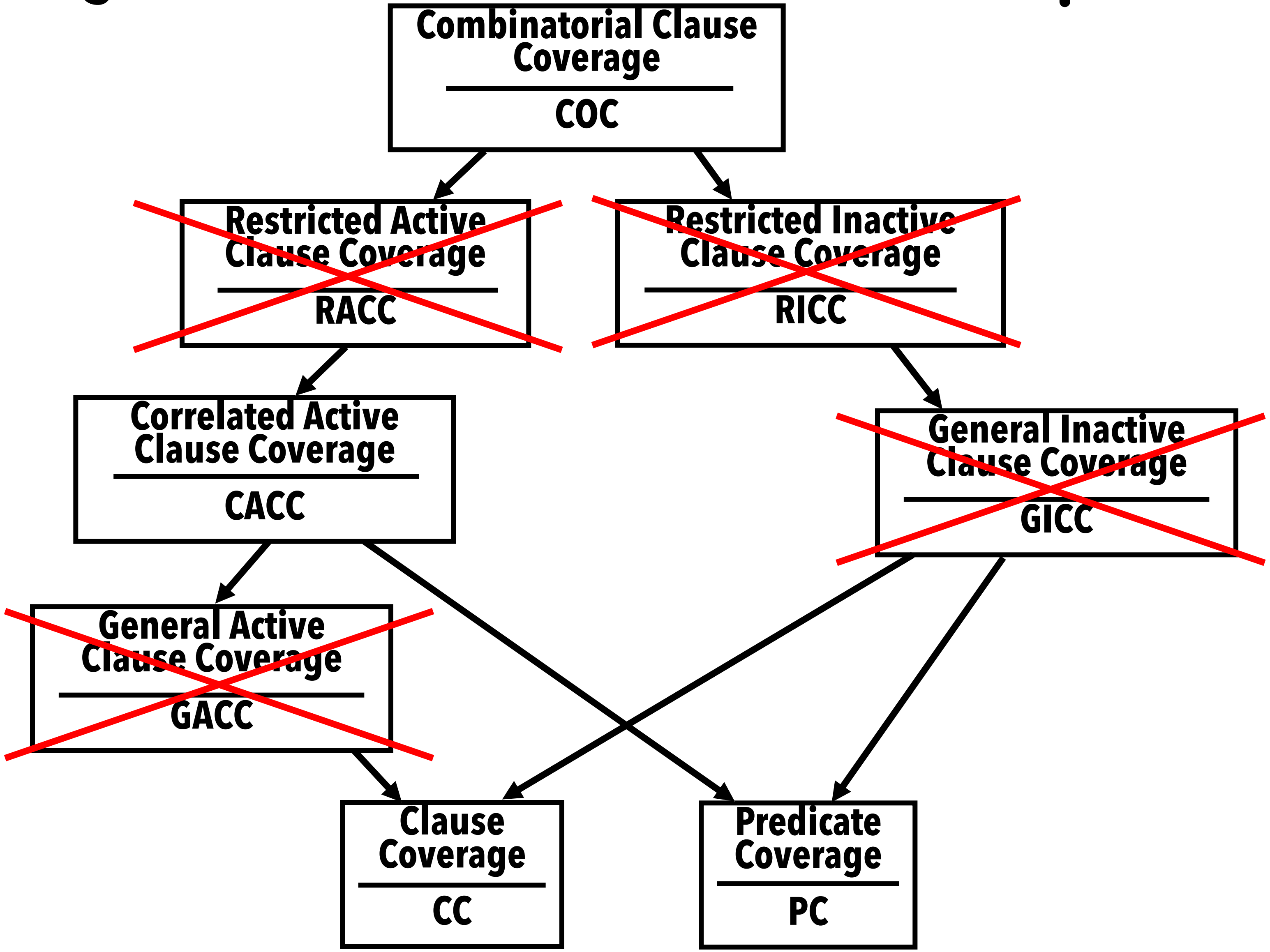
Impossible!

Infeasible test requirements are recognized and ignored.

Recognizing infeasible test requirements is generally undecidable.

(thus usually done by hand)

Logic Criteria Subsumption



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.

	a	b	P=a ∧ b	p_a	p_b
1	T	T	T		
2	T	F	F		
3	F	T	F		
4	F	F	F		

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.

	a	b	$P=a \wedge b$	p_a	p_b
1	T	T	T	●	●
2	T	F	F		●
3	F	T	F	●	
4	F	F	F		

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"

	a	b	$P=a \vee b$	p_a	p_b
1	T	T	T		
2	T	F	T	●	
3	F	T	T		●
4	F	F	F	●	●

In-class Exercise

Tabular method

Use the tabular method to solve for Pa, Pb, and Pc.
Give solutions as pairs of rows.

	a	b	c	$a \wedge (b \vee c)$	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	F			
5	F	T	T	F			
6	F	T	F	F			
7	F	F	T	F			
8	F	F	F	F			

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*

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*

	a	b	c	$a \wedge (b \vee c)$	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	F		●	●
5	F	T	T	F	●		
6	F	T	F	F	●		
7	F	F	T	F	●		
8	F	F	F	F			

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

$p = a \vee b$

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

$= (true \vee b) \text{ XOR } (false \vee b)$

$= true \text{ XOR } b$

$= !\, b$

$p = a \wedge b$

Use the definitional approach to solve for Pa

Definitional Method Examples

$\underline{p = a \lor b}$

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

$\underline{p = a \land b}$

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

Use the definitional approach to solve for Pa

Definitional Method Examples

$\underline{p = a \vee b}$

$p_a = p_{a=true} \oplus p_{a=false}$
$= (true \vee b) \text{ XOR } (false \vee b)$
$= true \text{ XOR } b$
$= !\, b$

$\underline{p = a \wedge b}$

$p_a = p_{a=true} \oplus p_{a=false}$
$= (true \wedge b) \oplus (false \wedge b)$
$= b \oplus false$
$= b$

Use the definitional approach to solve for Pa

$\underline{p = a \vee (b \wedge c)}$

$p_a = p_{a=true} \oplus p_{a=false}$
$= (true \vee (b \wedge c)) \oplus (false \vee (b \wedge c))$
$= true \oplus (b \wedge c)$
$= !\,(b \wedge c)$
$= !\, b \vee !\, c$

Use the definitional approach to solve for Pa

"*NOT b* $\vee$ *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*"

$$p = A \oplus A \wedge b$$
$$= A \wedge \neg b$$

$$p = A \oplus A \vee b$$
$$= \neg A \wedge b$$

with fewer symbols …

p = A xor (A and b)
= A and !b

p = A xor (A or b)
= !A and b

Repeated Variables

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

$(a \vee b) \wedge (c \vee b) == (a \wedge c) \vee b$

$(a \wedge b) \vee (b \wedge c) \vee (a \wedge c)$

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

$$p = (a \land b) \lor (a \land !b)$$

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

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

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

$= true \oplus false$

$= true$

$$p = (a \land b) \lor (a \land \neg b)$$

$p_b = p_{b=true} \oplus p_{b=false}$

$= ((a \land true) \lor (a \land !true)) \oplus ((a \land false) \lor (a \land !false))$

$= (a \lor false) \oplus (false \lor a)$

$= a \oplus a$

$= false$

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 | b) & (a | c) & d

Use the definitional method to solve for P_a

First step: ((T | b) & (T | c) & d) xor ((F | b) & (F | 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) xor ((F | b) & (F | c) & d)
= (T & T & d) xor (b & c & d)
= d xor (b & c & d)

Using the identity: A xor (A & b) == A and !b
= d & !(b & c)
= d & (!b | !c)