Johns Hopkins University

EN.601.422 / EN.601.622

Software Testing & Debugging

Logic Coverage

- With *equivalence partitioning*, we make certain we adequately cover the domain of input/output values
- With Whitebox coverages (statement- & branch-coverages) we make sure we reach all different parts and branches of the code
- With *logic coverage*, we focus on the combinations of truth assignments of the logical expressions in loops and conditionals

Logic Coverage Terminology

- **Predicate:** an expression that evaluates to either true or false
  - Predicates contain Boolean variables, relational operators (e.g., <,>, !=, == etc.), Boolean function calls, logical operators (e.g., logical AND, logical OR etc.) or Boolean values (i.e., true and false)
- **Clause:** A predicate with no logical operators
- **Logical Operators:**
  - $\neg$ – the *negation* operator
  - $\wedge$ – the *and* operator
  - $\vee$ – the *or* operator
  - $\rightarrow$ – the *implication* operator
  - $\oplus$ – the *exclusive or* operator
  - $\leftrightarrow$ – the *equivalence* operator

Logic Expression

- Logic expressions may be derived from variety of artifacts:
  - **Source code:** e.g., `if ((x > 10 && y < -22) || !(z == null))`
  - **Specifications:** *"to be able to push a new item onto stack, the stack should not be full and stack object not null"*
  - **FSMs:** "state = card_inserted and action = PIN_ok → state = show_accounts"
  - **SQL queries:** "SELECT AVG(Salary), Emp_Age FROM Employee WHERE Salary > 100,000 and Emp_Age < 35"
  - etc.

Logic Expression

- Example: $(a < b) \lor f(z) \land D \land TRUE$
  - `(a < b)` is relational Boolean expression
  - `f (z)` is a Boolean function call (function **f** returns true or false)
  - D is a Boolean variable
  - TRUE is a Boolean value
- The predicate is: "$(a < b) \lor f(z) \land D \land TRUE$"
- The four clauses are: "`(a < b)`", "`f(z)`", "D", and "TRUE"

Logic Expression

- Human Language can be vague sometimes:
  - Example: *“I am interested in EN.601.622 and EN.601.682”*
    - Course = *EN.601.622* OR Course = *EN.601.682*
- Especially important when applying logic coverage on informal specifications, user manual, API docs, etc.

Logic Coverage Criteria

- Assume **P** is the set of all predicates and **C** is the set of all clauses in P

<u>Predicate Coverage (PC)</u> : For each $p \in P$, *TR* contains two requirements: *p* evaluates to true, and *p* evaluates to false.

<u>Clause Coverage (CC)</u> : For each $c \in C$, *TR* contains two requirements: *c* evaluates to true, and *c* evaluates to false.

Example

- $(x > y \lor f(z)) \land w$

  C1: $x > y$ C2: $f(z)$ C3: w

- **Predicate Coverage:**
  - x = 10, y = 9, f(z) = true, w = true the predicate evaluates to true
  - x = 5, y = 5 , f(z) = false, w = true the predicate evaluates to false
- **Clause Coverage:**
  - x = 10, y = 9, f(z) = true, w = true C1 is true, C2 is true, C3 is true
  - x = 5, y = 5, f(z) = false, w = false C1 is false, C2 is false, C3 is false

Logic Coverage Criteria

- *PC* does not fully exercise all the clauses, especially in the presence of short circuit evaluation
- Short circuit: Not all clauses may be evaluated
  - Example 1: $x > y \vee f(z)$ — $f(z)$ will be ignored if $x > y$
  - Example 2: $w \wedge f(z)$ — $f(z)$ will be ignored if w is false
- CC does not subsume PC, i.e., we may satisfy CC without causing the predicate to be both true and false
- The most comprehensive solution is to try all possible combinations of the clauses

Logic Coverage Criteria

- Assume **P** is the set of all predicates and **C** is the set of all clauses in P

<u>Combinatorial Coverage (CoC)</u> : For each $p \in P$, TR has test requirements for the clauses in C_p to evaluate to each possible combination of truth values.

Example

$$(x > y \lor f(z)) \land w$$

C1 **C2** **C3**

C1	C2	C3	P
T	T	T	**T**
T	T	F	**F**
T	F	T	**T**
T	F	F	**F**
F	T	T	**T**
F	T	F	**F**
F	F	T	**F**
F	F	F	**F**

C1	C2	C3	P
x = 1, y = 0	f(z) = true	w = true	**T**
x = 1, y = 0	f(z) = true	w = false	**F**
x = 1, y = 0	f(z) = false	w = true	**T**
x = 1, y = 0	f(z) = false	w = false	**F**
x = 1, y = 1	f(z) = true	w = true	**T**
x = 1, y = 1	f(z) = true	w = false	**F**
x = 1, y = 1	f(z) = false	w = true	**F**
x = 1, y = 1	f(z) = false	w = false	**F**

Combinatorial Coverage Criterion

- For a predicate with **n** clauses, there are 2^n possible truth assignments
- Not scalable, maybe even impractical if number of clauses is very large

Is there a way to capture the *effect* of each and every clause while not exhausting all combinations?

Active Clause

Active Clause

- Can we control for all the clauses in a predicate except one such that that one clause (i.e., active/major clause) would decide the outcome of the predicate?
  - **Example:** $(a \vee b) \wedge c$ **a** is "active" if **b** is false and **c** true
    c is active if $(a \vee b)$ is true

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 ,(j != i), are such that changing C_i changes the value of *p*

Determination

- Our goal is to find tests for each clause when the clause determines the value of the predicate
- An important thing to note about “determination”
  - the definition does not require $c_i = p$ (c_i can be negation of p)
    - *Example:* a $\leftrightarrow$ b   a is active if b is false

      a = false, b = false → P = true

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, $j \mathrel{!}= i$, 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.

$p = a \vee b$

1) **a = true, b = false**
2) **a = false, b = false**
3) **a = false, b = true**
4) ~~**a = false, b = false**~~

a is major clause (1, 2)

b is major clause (3, 4)

Duplicate (4)

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. 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.

Example

$$p = a \leftrightarrow b$$

- **a** determines **p** no matter what **b**'s value is
  - Assume b is false. Now, if a is true → p = false and if a is false → p = true
  - Assume b is true. Now, if a = true → p = true and if a is false → p = false
- Likewise**,** **b** determines **p** no matter what **a**'s value is
  - Assume a is false. Now, if b is true → p = false and if b is false → p = true
  - Assume a is true. Now, if b is true → p = true and if b is false → p = false
- TR contains both ***a*** and ***b*** evaluate to both true and false
  - Can be achieved by {TT, FF}: TT→ p = true
FF → p = true

General Active Clause Coverage

- It is possible to satisfy GACC without satisfying predicate coverage
- GACC does not subsume predicate coverage (PC)
- We really want to cause predicates to be both true and false!

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

Example

$p = a \leftrightarrow b$

- **a** determines **p** no matter what **b**’s value is
  - Assume b = false. Then a = true → p = false and a = false → p = true
  - Assume b = true. Then a = true → p = true and a = false → p = false
- **b** determines **p** no matter what **a**’s value is
  - Assume a = false. Then b = true → p = false and b = false → p = true
  - Assume a = true. Then b = true → p = true and b = false → p = false
- TR should contain both ***a*** and ***b*** evaluate to both true and false **and *p* must evaluate to true and false for each truth assignment of either *a* or *b*.**
  - **CACC CanNOT** be achieved by {TT, FF}: TT→ p = true
    FF → p = true

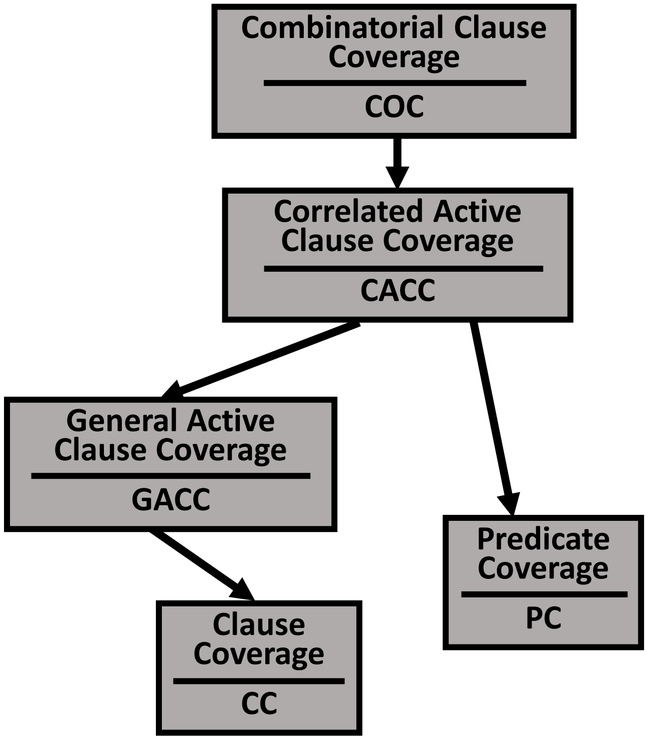
Example

$p = a \leftrightarrow b$

- CACC can now be achieved by the combinations in the table:
  - a is the major clause:
    - a = true, p = true (1st row)
    - a = false, p = false (3rd row)
  - b is the major clause:
    - b = true, p = true (1st row)
    - b = false, p = false (2nd row)

a	b	p
T	T	T
T	F	F
F	T	F

Logic Criteria Subsumption



Exercise

- First, formulate the following sentence as a logic predicate:
  - *"List all wireless printers in the store with a price of greater than $300 or for which the store has more than 100 items. Also, list all non-wireless printers with price less than $200"*
- Next, write truth assignments to achieve:
  - *Clause Coverage*
  - *CACC*

`wireless(e) ∧ (price(e) > 300 ∨ count(e) > 100) ∨ (non-wireless(e) ∧ price(e) < 200)`

Clause Coverage

C1	**C2**	**C3**	**C4**	**C5**	**P**
T	T	T	F	T	T
F	F	F	T	F	F

Constraint: C1 ⊕ C4

CACC

C1	**C2**	**C3**	**C4**	**C5**	**P**
T	T	-	F	-	T
F	T	-	T	F	F
T	**T**	F	F	F	T
T	**F**	F	F	F	F
T	F	**T**	F	F	T
T	F	**F**	F	F	F
F	-	-	**T**	T	T
T	F	F	**F**	T	F
F	-	-	T	**T**	T
F	-	-	T	**F**	F