EE360T/382V Software Testing

khurshid@ece.utexas.edu

February 15, 2020

Overview

Now – Logic coverage

Last time – Graph coverage for designs and specs

Next time – Continue with logic coverage

EE360T/382V Software Testing

khurshid@ece.utexas.edu

Chapter 3*: Logic Coverage

*Introduction to Software Testing by Ammann and Offutt

Criteria based on structures

The textbook focuses on four kinds of structures to define criteria:

- Graphs
  - E.g., control-flow graphs (CFGs)
- Logical expressions
  - E.g., if-conditions
- Input domain characterization
  - E.g., sorted array
- Syntactic structures
  - E.g., mutation

3.1 Overview: logical expressions

Predicate – expression that evaluates to a boolean value, e.g., "(($a > b$) || C) && p(x)"

- May contain variables (boolean or non-boolean) and methods
- Internal structure defined by logical operators, e.g., "!", "&&", "||"

Clause – predicate that contains no logical operator, e.g., "$a > b$", "C", and "p(x)"

Logical expressions come from various sources, e.g., program source-code

3.2 Logic expression coverage criteria

P – set of predicates

C – set of clauses in predicates in *P*

C3.12 Predicate coverage (*PC*) – for each *p* ϵ *P*, *TR* contains two requirements:

- *p* evaluates to *true*; and
- *p* evaluated to *false*

C3.13 Clause coverage (CC) -- for each *c* ϵ *C*, *TR* contains two requirements:

- *c* evaluates to *true*; and
- *c* evaluated to *false*

PC and CC relation

Consider the predicate "((a > b) || C) && $p(x)$"

Predicate coverage satisfied by two tests:

1. $a = 5, b = 4, C = true, p(x) = true$
2. $a = 5, b = 4, C = true, p(x) = false$

- The two tests do not satisfy clause coverage

Clause coverage satisfied by two tests:

1. $a = 5, b = 5, C = false, p(x) = true$
2. $a = 5, b = 4, C = true, p(x) = false$

- The two tests do not satisfy predicate coverage!

PC does **not** subsume CC and CC does **not** subsume *PC*

Combinatorial coverage

C_p – set of clauses in predicate p

C3.14 Combinatorial coverage (CoC) – for each $p \in P$, *TR* has test requirements for the clauses in C_p to evaluate to each possible combination of truth values

- Also called multiple condition coverage

Example: "!a || b"

a	*b*	*!a || b*
F	*F*	*T*
F	*T*	*T*
T	*F*	*F*
T	*T*	*T*

Predicate with *n* clauses has 2^n possible assignments

Often impractical for predicates with > a few clauses

Determination

Motivation – need criteria that capture the effect of each clause using a reasonable number of tests

- Would like to exercise conditions where flipping a clause flips the predicate

Major clause – clause c_i that is our focus

Minor clause – clause c_j where $j \neq i$

D3.42 Determination – major clause c_i *determines* predicate p if the minor clauses have values so that changing the truth value of c_i changes the value of p

Active clause coverage

D3.43 Active clause coverage (*ACC*)* – *TR* has two requirements for each active clause $c_i \in C_p$ for each $p \in P$: c_i evaluates to *true*; and c_i evaluates to *false*

Example – consider predicate $p = a \,||\, b$

- *a* determines *p* iff *b* is false; likewise for *b*
- 4 requirements:
  - $c_i = a$: { <a = ***T***, b = *f*>, <a = ***F***, b = *f*> }
  - $c_i = b$: { <a = *f*, b = ***T***>, <a = *f*, b = ***F***> }
    - 2 of these are identical, so 3 in total

Key question in *ACC* – do minor clauses have constant values when major clause c_i is and when c_i is false?

*Almost identical to *MCDC* in literature

CACC and *RACC*

C3.16 Correlated active clause coverage (*CACC*) – *TR* has two requirements for each major clause $c_i \in C_p$ for each $p \in P$: c_i evaluates to *true*; and c_i evaluates to *false*. **The values chosen for minor clauses c_j ($j \neq i$) must cause *p* to be true for one value of c_i, and false for the other**

C3.17 Restricted active clause coverage (*RACC*) – *TR* has two requirements for each major clause $c_i \in C_p$ for each $p \in P$: c_i evaluates to *true*; and c_i evaluates to *false*. **The values chosen for minor clauses c_j ($j \neq i$) must be the same when c_i is *true* as when c_i is false**

CACC and *RACC* Example

a determines the predicate "*a* && (*b* || *c*)" when "(*b* || *c*)" is true

- <b = T, c = T>
- <b = T, c = F>
- <b = F, c = T>

CACC – pick one of { 1, 2, 3 } and one of { 5, 6, 7 }

- 9 choices

RACC – pick one of <1, 5>, <2, 6>, and <3, 7>

- 3 choices

	a	*b*	*c*	*a* && (*b* || *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*

Inactive clause coverage

Basis – check that changing a clause that should not affect the predicate does not, in fact, affect it

D3.44 Inactive clause coverage (*ICC*) – for each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses c_j ($j \neq i$) so that c_i does **not** determine p. *TR* has four test requirements for c_i:

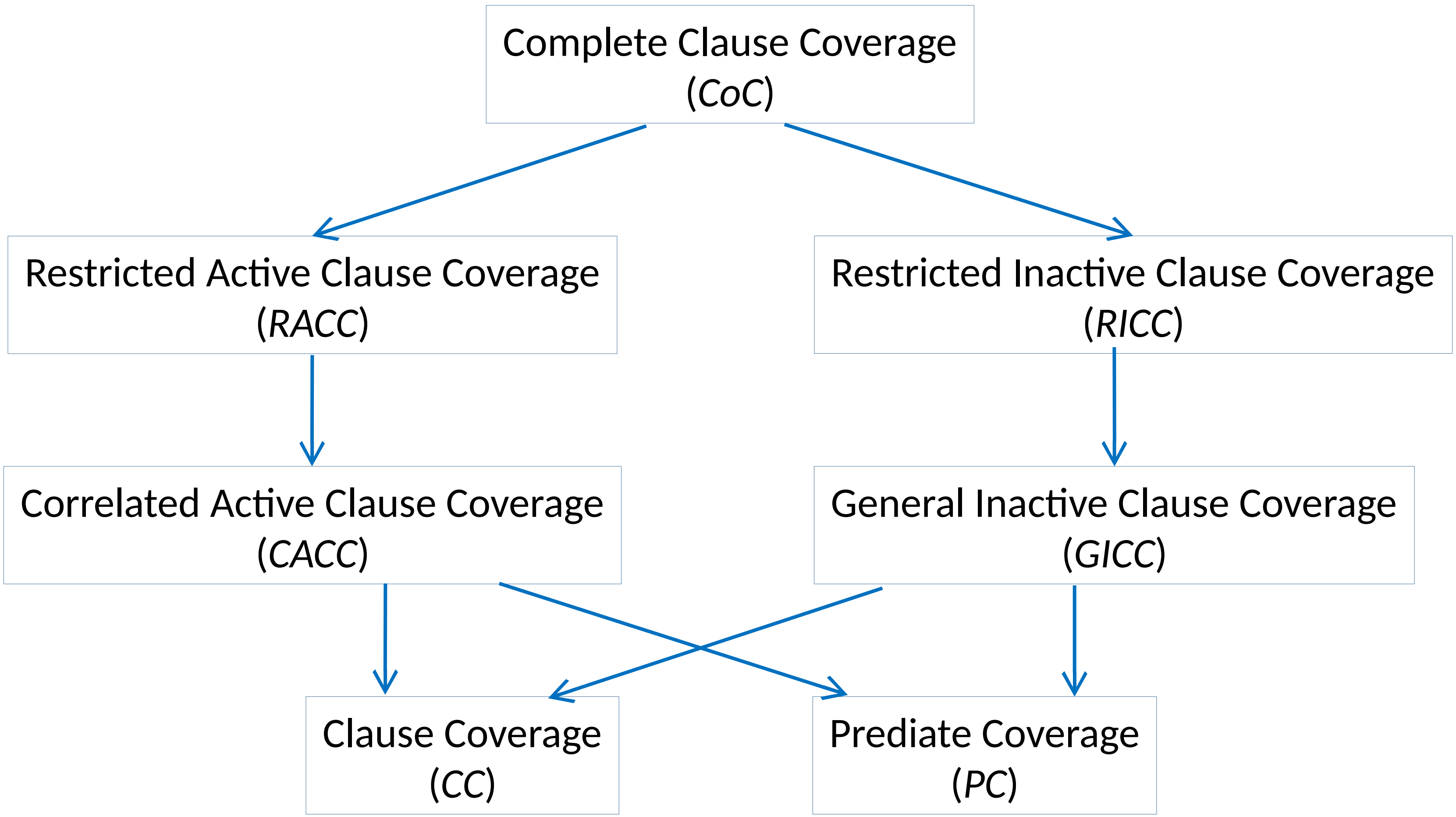
1. c_i evaluates to *true* with *p true*;
2. c_i evaluates to *false* with *p true*;
3. c_i evaluates to *true* with *p false*; and
4. c_i evaluates to *false* with *p false*;

GICC and *RICC*

C 3.18 General inactive clause coverage (*GICC*) – *ICC* such that the values chosen for the minor clauses for a given major clause may vary among the four cases

C 3.19 Restricted inactive clause coverage (*RICC*) – *ICC* such that the values chosen for the minor clauses for a given major clause must be the same in cases (1) and (2), and also be the same in cases (3) and (4)

Subsumption



Making a clause determine a predicate

Let p be a predicate and c be a clause in p

Let $p_{c=true}$ be p with c set to *true*

Let $p_{c=false}$ be p with c set to *false*

Then, *solutions* to formula $\boldsymbol{p_{c=true} \oplus p_{c=false}}$ give values for clauses $\neq c$ such that c determines p

- Each solution is assignment of values to clauses in $p_{c=true} \oplus p_{c=false}$ such that the formula is *true*

Example of making a clause active

$p = a \,||\, b$

To make a active, solve $p_a = p_{a=true} \oplus p_{a=false}$

$= (true \,||\, b) \oplus (false \,||\, b)$

$= true \oplus b$

$= !b$

There is one solution to p_a, which is $b = false$

Thus, setting $b = false$ makes clause a active

By symmetry, $p_b = !a$

Another example

$p = a \;\&\&\; b$

To make a active, solve $p_a = p_{a=true} \oplus p_{a=false}$

$$= (true \;\&\&\; b) \oplus (false \;\&\&\; b)$$

$$= b \oplus false$$

$$= b$$

There is one solution to p_a, which is $b = true$

Thus, setting $b = true$ makes clause a active

By symmetry, $p_b = a$

An example with no constraint

$p = a \Leftrightarrow b$

To make a active, solve $p_a = p_{a=true} \oplus p_{a=false}$

$= (true \Leftrightarrow b) \oplus (false \Leftrightarrow b)$

$= b \oplus !b$

$= true$

Any value of b is a solution

Thus, setting b to any value makes clause a active

By symmetry, $p_b = true$

A degenerate case

$p = a \mathbin{\&\&} b \mid\mid a \mathbin{\&\&} !b$

To make *b* active, solve $p_b = p_{b=true} \oplus p_{b=false}$

$= (a \mathbin{\&\&} true \mid\mid \mathrm{a} \mathbin{\&\&} !\mathrm{true}) \oplus (a \mathbin{\&\&} false \mid\mid a \mathbin{\&\&} !false)$

$= (\mathrm{a} \mid\mid false) \oplus (false \mid\mid a)$

$= a \oplus a$

$= false$

There is no solution to p_b

Thus, there is no value for *a* that makes *b* active

An example with 3 clauses

$p = a \mathbin{\&\&} (b \mid\mid c)$

To make *a* active, solve $p_a = p_{a=true} \oplus p_{a=false}$

$$= (true \mathbin{\&\&} (b \mid\mid c)) \oplus (false \mathbin{\&\&} (b \mid\mid c))$$

$$= (b \mid\mid c) \oplus false$$

$$= b \mid\mid c$$

There are three distinct solutions to p_a:

- $<b = T,\ c = T>$, $<b = T,\ c = F>$, and $<b = F,\ c = T>$

Any of these three pairs makes clause *a* active

Exercise – make *b* active

?/!