Testing: (A Little) Logic Coverage

Testing, Quality Assurance, and Maintenance
Winter 2018

Prof. Arie Gurfinkel

UNIVERSITY OF
WATERLOO

Why Logic Coverage?

MC/DC (Modified condition/decision coverage)

MC/DC is required by the FAA for safety critical software

- guidelines DO-178B: Software Considerations in Airborne Systems and Equipment Certification.

A Practical Tutorial on Modified Condition/Decision Coverage by NASA

- http://shemesh.larc.nasa.gov/fm/papers/Hayhurst-2001-tm210876-MCDC.pdf

Decisions

Definitions

- A logical expression in a program is called a *decision*.
- The leaves of a decision are called *conditions.*
- Conditions are connected by logical operators to make a decision.

A *decision* is an expression that evaluates to a Boolean value.

- e.g., $a \wedge b \leftrightarrow c$

Decisions can come from code:

- e.g., `b = (visited && x > y || foo(z))`

UNIVERSITY OF WATERLOO

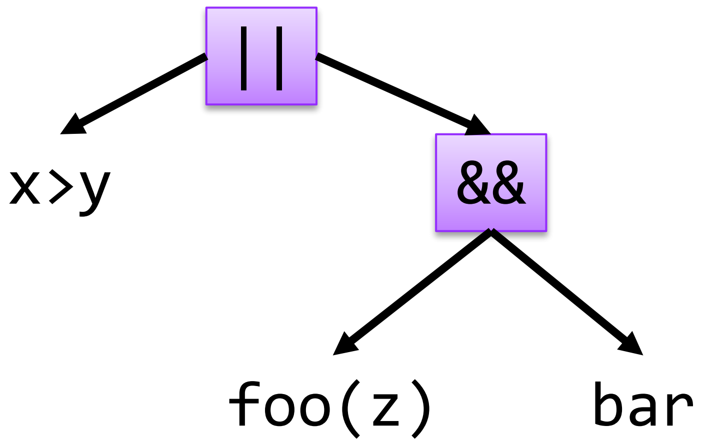
Conditions

Conditions are decisions/predicates without logical operators.

- also called *atomic predicates*

For example, this predicate contains three conditions:

- `(x > y) || (foo(z) && bar)`



Notation

P is a set of decisions (predicates) of the program

For a predicate p, C_p is the set of conditions making up p

C is the set of all conditions in all of the decisions in P

Formally:

$$C_p = \{c \mid c \in p\} \qquad\qquad C = \bigcup_{p \in P} C_p$$

UNIVERSITY OF WATERLOO

Decision Coverage (DC)

Criterion 1: Decision Coverage (DC). For each p ∈ P, TR contains two requirements:

1. *p* evaluates to true; and
2. *p* evaluates to false.

DC is analogous to edge coverage on a CFG. (Let P be the decisions associated with branches.)

- Example:

$$P = \{(x + 1 == y) \wedge z\}$$

predicate coverage:
x = 4, y = 5, z = true -> P=T
x = 3, y = 5, z = false -> P=F

DC is very coarse-grained.

- solution: break up decisions into conditions

Condition Coverage (CC)

Condition Coverage (CC). For each $c \in C$, TR contains two requirements:

1. *c* evaluates to true; and
2. *c* evaluates to false.

Combinatorial Coverage (CoC)

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

This is also known as **multiple condition coverage**.

Unfortunately, the number of test requirements, while finite, grows exponentially and is hence unscalable.

Active Conditions

Focus on each condition and make sure it affects the decision.

- Test each condition while it is active

$$p = x \wedge y \vee (z \wedge w)$$

For example, focus on y – called the major condition.

- Minor conditions: x, z, w.

Determine

$$p = x \wedge y \vee (z \wedge w)$$

y determines *p* with certain minor condition values:

- if we set *y* to true, then *p* evaluates to some value X
  – where X is either True or False
- if we set *y* to false, then *p* must evaluate to ¬ X.

Question: What truth assignment

- x =? z =? w = ?

will make y determine p

- in particular, y true makes p true and y false makes p false.

Active Condition Coverage (ACC)

Active Condition Coverage (ACC). For each $p \in P$ and making each condition $c_i \in c_p$ major, choose assignments for minor conditions c_j , $j \neq i$ such 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 MC/DC, which is required by the FAA for safety critical software.

ACC Example

For p = a v b, make *a* major. We need *b* to be false for a to determine *p*.

- This leads to the TRs: {(a=T, b=F), (a=F, b=F)}

Similarly for *b* to determine *p* we need TRs:

- {(a=F, b=T), (a=F, b=F)}

Note the overlap between TRs; it will always exist, meaning that our set of TRs for active clause coverage is:

- {(a=T, b=F), (a=F, b=T), (a=F, b=F)}

Coverage Criteria

Basic Coverage

- Line coverage
- Statement
- Function/Method coverage
- Branch coverage
- Decision coverage
- Condition coverage
- Condition/decision coverage
- Modified condition/decision coverage
- Path coverage
- Loop coverage
- Mutation adequacy
- …

Advanced Coverage

Strengths and Weaknesses of Graph Coverage

Must create graph

Node coverage is usually easy, but cycles make it hard to get good coverage in general

Incomplete node or edge coverage point to deficiencies in a test set

Summary

Summarizing Structural Coverage:

- Generic; hence broadly applicable
- Uses no domain knowledge

Summarizing Dataflow Coverage:

- Definitions and uses are useful but hard to reason.

Miscellaneous other notes:

- Control-flow graphs are manageable for single methods, but not generally more than that.
- Use call graphs to represent multiple methods, hiding details of each method.
- When we want to test du-paths across multiple elements, use first-use/last-def heuristic.