CS 5154: Software Testing

Logic-Based Coverage

Owolabi Legunsen

Some resources on Graph-Based MDTD

- https://cs.gmu.edu:8443/offutt/coverage/GraphCoverage
- https://primepaths.coecis.cornell.edu

Demo on measuring branch coverage

CS5154 is organized into six themes

1. How to automate the execution of tests?
2. How to design and write high-quality tests?
3. How to measure the quality of tests?
4. How to automate the generation of tests?
5. How to reduce the costs of running existing tests?
6. How to deal with bugs that tests reveal?

Recall the four software models in this course

✓ ✓ ↓

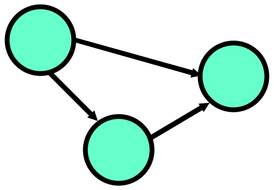
Input Domains

Graphs

Logic Expressions

Syntax

```
A: {0, 1, >1}
B: {600, 700, 800}
C: {cs, ece, is, sds}
```



```
(!x | !y) & a & b
```

```
if (x > y)
    z = x - y;
else
    z = 2 * x;
```

Why learn about logic coverage?

- Tests can miss faults with 100% statement, branch, and prime path coverage
- The “I” in the RIPR model
  - Increase the chance that faults in reached code infect the state
- Required by the US Federal Aviation Administration for safety critical software

Example on the importance of logic coverage

```
int stringFactor(String i, int n) {
  if (i != null || n !=0)
   return i.length()/n;
  else
   return -1;
}
// Tests: ("happy", 2), (null, 0)
```

- Tests achieve 100% statement, branch, prime path coverage but miss faults

Steps in Logic-based MDTD

- Develop a model of the software as a set of predicates
- Require tests to satisfy some combination of clauses

Terminology: predicates and clauses in software

- *Predicate* : an expression that evaluates to a boolean value
- In software, a predicate can be
  - boolean variables or non-boolean variables related by >, <, ==, >=, <=, !=
  - function calls that return boolean values
  - composed from other predicates using logical operators (!, $\wedge$, $\vee$, $\rightarrow$, $\oplus$, $\leftrightarrow$)
- A *clause* is a predicate with no logical operators

Example predicate

- $(a < b) \vee f(z) \wedge D \wedge (m >= n*o)$ has four clauses:
  - $(a < b)$ – relational expression
  - $f(z)$ – boolean-valued function/method
  - D – boolean variable
  - $(m >= n*o)$ – relational expression

A study of predicates in programs

- A study of non-FAA, 63 open-source programs with >400,000 predicates*
- Most predicates have few clauses
  - 88.5% have 1 clause
  - 9.5% have 2 clauses
  - 1.35% have 3 clauses
  - Only 0.65% have 4 or more !

* What to expect of predicates: An empirical analysis of predicates in real world programs. Vinicius H.S. Durelli, Jeff Offutt, Nan Li, Marcio E. Delamaro, Jin Guo, Zengshu Shi, Xinge Ai. Journal of Systems and Software. 2016

Where do software predicates come from

- Decisions in programs
- Guards in finite state machines
- Decisions in UML activity graphs
- Requirements, both formal and informal
- SQL queries

Abbreviations that we will use

- P is the set of predicates
- p is a single predicate in P
- C is the set of clauses in P
- C_p is the set of clauses in predicate p
- c is a single clause in C

Steps in Logic-based MDTD

- Develop a model of the software as a set of predicates ✓
    - That’s it!

- Require tests to satisfy some combination of clauses
    - What concept have we learned in this course that can help us here?

Predicate and Clause Coverage

Predicate Coverage (PC) : For each *p* in *P*, *TR* contains two requirements: *p* evaluates to true, and *p* evaluates to false.

Clause Coverage (CC) : For each *c* in *C*, *TR* contains two requirements: *c* evaluates to true, and *c* evaluates to false.

Illustrating Predicate and Clause Coverage

```
int stringFactor(String i, int n) {
  if (i != null || n !=0)
   return i.length()/n;
  else
   return -1;
}
// Tests: ("happy", 2), (null, 0)
```

Do these tests satisfy Predicate Coverage?

Do these tests satisfy Clause Coverage?

Practice: Predicate Coverage

- Provide sets of variable assignments that satisfy predicate coverage

$$((a < b) \vee D) \wedge (m >= n*o)$$

a, b, m, n, and o are int; D is a Boolean

True: a = 5, b = 10, D = true, m = 1, n = 1, o = 1

False: a = 5, b = 10, D = true, m = 0, n = 1, o = 1

Practice: Clause Coverage

- Provide sets of variable assignments that satisfy clause coverage

$$((a < b) \vee D) \wedge (m >= n*o)$$

True: a = 5, b = 10, D = true, m = 1, n = 1, o = 1

False: a = 10, b = 5, D = false, m = 1, n = 2, o = 2

Any question so far

?

Are Predicate and Clause Coverage Sufficient?

```
int stringFactor(String i, int n) {
  if (i != null || n !=0)
   return i.length()/n;
  else
   return -1;
}
// Tests: ("happy", 2), (null, 0)
```

Tests satisfy Statement, Branch, Prime Path, Predicate, and Clause Coverage

Still, these tests miss the fault

More Predicate and Clause Coverage problems

- Does predicate coverage subsume clause coverage? Why?

No. See earlier example/practice slides. Also, short circuit evaluation.

- Does clause coverage subsume predicate coverage? Why?

No. Example: a ∨ b

a	b	$a \vee b$
T	T	T
T	F	T
F	T	T
F	F	F

A stronger criterion than PC and CC is needed

- Test all combinations of clauses!

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.

- Sometimes called *Multiple Condition Coverage*

Illustrating Combinatorial Coverage

	a < b	D	m >= n*o	((a < b) ∨ D) ∧ (m >= n*o)
1	T	T	T	T
2	T	T	F	F
3	T	F	T	T
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

Would Combinatorial Coverage find the fault?

```
int stringFactor(String i, int n) {
  if (i != null || n !=0)
   return i.length()/n;
  else
   return -1;
}
// Tests: ("happy", 2), (null, 0)
```

More on Combinatorial Coverage

- CoC is a simple, neat, clean, and comprehensive criterion
- But it can be expensive
  - 2^N tests for one predicate, where N is the number of clauses
  - Programs often have several clauses.
  - Impractical if $N > 3$ or $N > 4$

We need criteria that are not as costly as CoC

- The general idea is quite simple:

Test each clause independently from the other clauses

- But, getting the details right is hard
  - e.g., what exactly does "independently" mean ?

- The book presents this idea as "*making clauses active*" …

Logic coverage so far

- Why logic coverage is important
- The steps in logic-based MDTD
- Three logic-based coverage criteria

Next

- Criteria that improve the shortcomings of Combinatorial Coverage