CS 5154: Software Testing

Applying Logic Coverage to Source Code

Instructor: Owolabi Legunsen

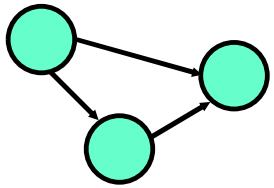
Fall 2021

Recall the four software models in this course

Input Domains

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

Graphs



Logic Expressions

(!x | !y) & a & b

Syntax

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

Steps in Logic-based MDTD

- Develop a model of the software as a set of predicates ✓
  - That's it!
  - But how?

- Require tests to satisfy some combination of clauses ✓
  - We learned some criteria and their strengths/weaknesses

Predicates: logic expressions in source code

- Predicates are derived from decision statements
  - if, while, for, switch, do-while
- In programs, most predicates have less than four clauses
  - In fact, most have just one clause
- With one clause, CoC, ACC, and CC collapse to predicate coverage (PC)
  - ACC is only useful with three or more clauses

Finding values for variables in predicates

```
public int checkVal(int x) {
    y = x*2;
    if (x>0)
        if ((x>10 && x<20) || y==50)
            return 1;
    else
        if ((x<-10 && x>-20) || y<-60)
            return 2;
}
```

Some things to consider when finding values

- *Reachability* : tests must reach the predicate
- *Controllability* : tests must cause the (clauses in a) predicate to have the truth assignment that we want
- *Internal variables : reachability* and *controllability* require reasoning about variables that are not inputs

Finding values for variables in predicates (2)

```
1. public int checkVal(int x) {
2.     y = x*2;
3.     if (x>0)
4.         if ((x>10 && x<20) || y==50)
5.             return 1;
6.     else
7.         if ((x<-10 && x>-20) || y<-60)
8.           return 2;
9. }
```

What internal variables do we need to think about?

y

What values of x do we need to <u>reach</u> the predicate on line 4?

x > 0

<u>Control</u>: what values of x will satisfy the truth assignment TFT in the predicate on line 4?

x == 25

Another issue: beware of code transformations

With one clause, CoC, ACC, and CC collapse to predicate coverage (PC). So, why not just transform all predicates to have only one clause?

Why not just do this?

```
if ((a && b) || c)
{
  S1;
} else
{
  S2;
}
```

Transformation 1

```
if (a) {
  if (b)
    S1;
  else {
    if (c)
      S1;
    else
      S2;
    }
} else {
    if (c)
      S1;
    else
      S2;
}
```

Problems with Transformation 1

1. We trade one problem for two problems :
   - Maintenance becomes harder
   - Reachability can be harder to compute

```
if (a) {
  if (b)
    S1;
  else {
    if (c)
      S1;
    else
      S2;
    }
} else {
    if (c)
      S1;
    else
      S2;
}
```

More problems with Transformation 1

2. Consider coverage :
   - CACC on original code requires four rows
   - PC on transformed code requires five rows
   - Testing transformed code is more costly!
   - Tests that satisfy PC on transformed code do not satisfy CACC on the original code

a	b	c	(a∧b)∨c	CACC	PC_T
T	T	T	T		X
T	T	F	T	X	
T	F	T	T	X	X
T	F	F	F	X	X
F	T	T	T		X
F	T	F	F	X	
F	F	T	T		
F	F	F	F		X

Okay, but maybe I can just do this?

```
if ((a && b) || c)
{
  S1;
} else
{
  S2;
}
```

Transformation 2

```
d = a && b;
e = d || c;
if (e)
{
  S1;
} else
{
  S2;
}
```

Problems with Transformation 2

1. We move the complexity into computations :
   - Logic criteria are not effective at testing computations

```
d = a && b;
e = d || c;
if (e)
{
  S1;
} else
{
  S2;
}
```

More problems with Transformation 2

2. Consider coverage :
   - CACC on original code requires four rows
   - PC on transformed code requires two rows

   - PC on transformed code is equivalent to clause coverage (CC) on original code
     - CC is not effective for testing

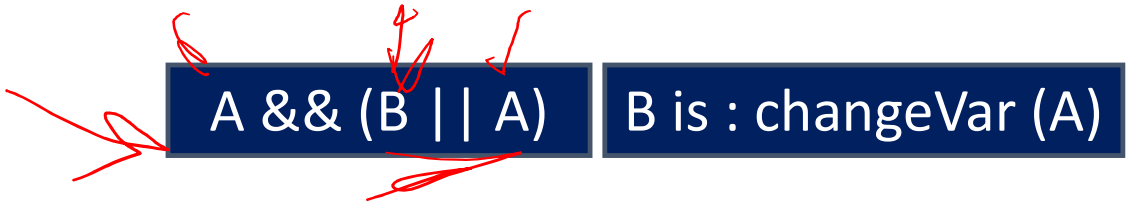
a	b	c	(a∧b)∨c	CACC	PC_T
T	T	T	T		X
T	T	F	T	X	
T	F	T	T	X	
T	F	F	F	X	
F	T	T	T		
F	T	F	F	X	
F	F	T	T		
F	F	F	F		X

The moral of the transformation story

Don’t

- Logic criteria exist to help us design better software
- Circumventing logic criteria via program transformations is unsafe

One last issue: side effects in predicates



- Runtime system checks *A*, then *B*, if *B* is false, check *A* again
- But now *A* has a different value!
- How to write a test that has two different values for A?

- There are no clear answers to this controllability problem!

We suggest a social solution : ask your team!

Summary: Logic Coverage and Source Code

- Predicates come from decision expressions (while, if, do-while), etc
- To find values for testing, reachability, controllability, and internal variables must be considered
- Using program transformations to sidestep logic criteria is a bad idea

Next

- Practicing logic coverage concepts on the next homework
- Syntax-based testing