Structural Testing

Supplement to Notes 3
J. Bieman
CS314

Structural (White Box) Test Coverage Criteria.

- Statement or node coverage.
- Branch coverage, edge coverage, or decision coverage.
- Condition coverage.
- Variable definition-use coverage.

Test Coverage Strength

- Branch coverage is stronger than statement coverage,
- Condition coverage is stronger than branch coverage, and
- Definition/Use coverage is stronger than branch coverage.

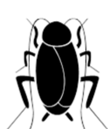
If tests satisfy a coverage criteria, they also satisfy all weaker ones.

Example

- Look at the code:
  ```
  if (A) S1;
  S2;
  ```
- We can cover both S1 and S2 with 1 test.
  - Just set A=true.
- To cover all branches, we must also test the path that skips S1.
  - We need another test case where A=false.

Sometimes Stronger Coverage is Needed

- Buggy code:
  ```
  i = 0;
  if (A) i = 1;
  x = y/i;
  ```



- No error when you test with A=true.
- Bombs if you test with A=false.

Branch coverage reveals the error, but statement coverage may not!

An Error, Not Detected by Branch Coverage

```
…
if (A() && B()) x = y + z;
…
boolean A() {
 if (F1) {
   q = 0;
   return true;
 }
 else return false;
}
```

```
…
boolean B() {
 if (F2) {
   q = 0;
   return true;
 }
 else {
   x = 10/q;
   return false;
 }
}
```

We Test the Code

- Branch coverage is satisfied with 2 tests:
  - F1==true and F2==true: takes the true path.
  - F1==false and F2==false: takes the false path.
- The error occurs when F1==true & F2==false.
- Condition coverage would require this test.

Another Example Program

```
while (notDone) do {
 if (A) x = f(x);
 else   x = g(x);
 ...
}
```

- **then** branch 1st references the prior value of x (a use of x) & then redefines x (a definition of x).
- The **else** branch does the same thing.

Test paths required by the all-uses (all DU-Pairs) criterion:
- Loop through the **then** branch twice in a row.
- Loop through the **else** branch twice in a row.
- 1 cycle through the **then** branch followed by a cycle through the **else** branch.
- A cycle through the **else** branch followed by a cycle through the **then** branch.

Slides from Prof. Ghosh and Prof. France

White box testing criteria

- **Statement coverage criterion**: Select a test set T such that executing program P for each t in T results in each elementary statement of P being executed at least once.
- **Edge-coverage criterion**: Select a test set T such that executing P for each t in T results in each edge of P's control graph being traversed at least once.
- **Condition-coverage criterion**: Select a test set T such that executing P for each t in T results in each edge of P's control graph being traversed at least once and all possible values of the constituents of compound conditions being exercised at least once.
- **Path-coverage criterion**: Select a test set T such that executing P for each t in T results in all paths leading from the initial to the final node of P's control graph being traversed.

Statement coverage example

1. read(x);
2. read(y);
3. if x > 0 then
4. write("1");
5. else
6. write("2");
7. end if;
8. If y > 0 then
9. write("3");
10. else
11. write("4");
12. end if;

Input domains for statement coverage
D1: {x>0}
D2:{x<=0}
D3: {y>0}
D4:{y<=0}

How did we get these domains?
Ans: from the branch conditions.

Statement coverage weakness

1. if x < 0 then
2. x := -x;
3. end if;
4. z:=x;

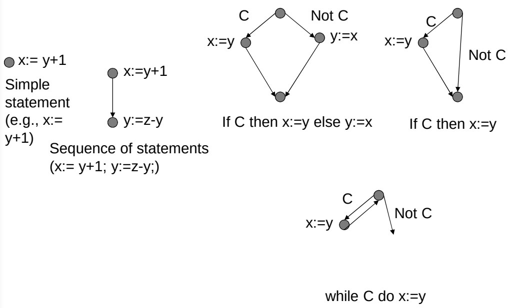
Program is intended to change negative numbers to positive number and leave positive numbers unchanged, and then assign to z

Input domains for statement coverage
D1: {x<0}

Weakness: does not cover the case when x >= 0.

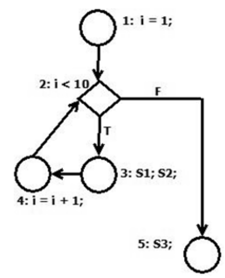
A test set that satisfies the **edge-coverage criterion** will cover the case when x>=0.

Control Flow Graph examples



For Loop Control Flow Graph

```
for (int i = 1;  i < 10;  i = i + 1) {
   s1;
   s2;
}
s3;
```



Edge coverage

1. if x < 0 then
2.   x := -x;
3. end if;
4. z:=x;

Input domains for edge coverage
D1: {x<0}
D2: {x >= 0}

Condition coverage vs. edge coverage criterion

```
found:= false; counter:= 1;
while (not found) and (counter < num_items) loop
   if table(counter) = desired_elem then
      found := true;
   end if;
   counter := counter + 1;
end loop;
If found then
   write("element found");
else
   write("element does not exist");
end if;
```


Edge criterion test set weakness

A test set for the program on previous slide:
- A table with no items
- A table with three items, the second being the desired element.

The above satisfies the edge coverage criterion but fails to uncover the error in the condition of the while loop (< instead of <=)

The coverage criterion can be used to uncover this error.

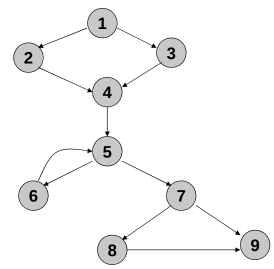
Checking condition coverage using control graphs

- You can use control graphs to check condition coverage if you can rewrite as an equivalent program that uses only conditions with single clauses
- Not as straightforward as you may think to do this!

Code finds the value of x^y

```
scanf(x, y); if(y < 0)
    pow = 0 – y;
else pow = y;
z = 1.0;
while(pow != 0)
  { z = z * x; pow = pow – 1;}
if ( y < 0 )
   z = 1.0/z;
printf(z);
```


Control Flow Graph



Problems with branch coverage

- What if a decision has many conditions (using *and*, *or*)
- Decision may evaluate to true or false without actually exercising all the conditions

```
int check (int x) {
   if ((x >= 5) && (x <= 200))
      return TRUE;
   return FALSE;
}
```

Test inputs:
x = 5:
x = -5:

Error (should be 100)

Rewrite of search program?

```
found:= false; counter:= 1;
while (not found) loop
  if (counter < num_items) then
     if table(counter) = desired_elem then
          found := true;
     end if;
     counter := counter + 1;
  else
     break;
end loop;
If found then
     write("element found");
else
     write("element does not exist");
end if;
```


Edge coverage weakness

```
if x not = 0 then
   y:=5;
else
   z:=z-x;
end if;
if z > 1 then
   z:=z/x;
else
   z:=0;
end if;
```

The following test set satisfies the edge criterion:
{<x=0,z=1>,<x=1,z=3>}

It does not uncover the division by 0 fault.

A test set that uncovers the fault is given below:
{<x=0,z=3>,<x=1,z=1>}

Solution?

- Require all individual conditions to evaluate to true and false
- Problem:
  - Even if individual conditions evaluate to true and false, the decision may not get both true and false values
- Solution:
  - Require both decision / condition coverage!!

White-box testing summary

- Tests what a program does
- Can catch only “commission” faults; cannot catch omission faults
  - Black box testing can be used to catch omission faults.
- It is not always possible to select test sets that satisfy criterion
  - E.g., unreachable statements in code makes it impossible to satisfy statement coverage criterion

Testing Limitations

- If our testing results in:
  - 100% statement coverage,
  - 100% branch coverage,
  - 100% condition coverage,

The program may still have hidden faults.

Why?