Structural (White Box) Testing

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.

White box testing criteria

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

- \( D_1: \{ x > 0 \} \)
- \( D_2: \{ x \leq 0 \} \)
- \( D_3: \{ y > 0 \} \)
- \( D_4: \{ y \leq 0 \} \)

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

- \( D_1: \{ x < 0 \} \)
- \( D_2: \{ x \geq 0 \} \)

Weakness: does not cover the case when \( x \geq 0 \).

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**

- \( D_1: \{ x < 0 \} \)
- \( D_2: \{ x \geq 0 \} \)

**Condition coverage vs. edge coverage criterion**

```plaintext
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 \(\leq\)).

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 \( x^y \)**

1. `scanf(x, y); if(y < 0)`
2. `pow = 0 - y;`
3. `else pow = y;`
4. `z = 1.0;`
5. `while(pow != 0)`
6. `{ z = z * x; pow = pow - 1;}`
7. `if (y < 0)`
8. `z = 1.0/z;`
9. `printf(z);`

**Control Flow Graph**

[Diagram of 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

```java
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?

```java
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;
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?