# Topic #4 – Software Condition-Based Testing

**Instructor:** Jerry Gao, Ph.D., Professor
San Jose State University


# Topic #3 – Software Condition-Based Testing

What Is Condition-Based Testing?

Why Is Condition-Based Testing Important?

How to Conduct Condition-Based Testing?

A Condition-Based Testing Example

Condition-Based Testing Coverage

# TOPIC #4: SOFTWARE CONDITION-BASED TESTING

## What Is Condition-Based Testing?

**Definition**: Condition-based testing is one program-based software testing strategy in which engineers focus on compound Boolean conditions in predicate nodes.

Its major testing focuses are incorrect logics and implementations in complex Boolean expressions for predicate nodes. They include:

- Boolean variable errors
- Boolean parenthesis errors
- Boolean operator errors
- Relational operator errors
- Arithmetic expression errors

Test model: Program flow graph model

# Why Do We Need Condition-Based Testing?

- **Software programs consist of many logic decisions (in Boolean expressions). Some of them implemented with compound Boolean Conditions.**

- **Many incorrect implementations of compound Boolean conditions lead to software decision errors**

- **The program code coverage is not enough to reach to the decision coverage (or the branch coverage)**

- **Software branch testing can't assure the adequate test coverage for each combined Boolean conditions and its outcomes for a predicate node in a program flow graph.**

# TOPIC #3 – SOFTWARE CONDITION-BASED TESTING

## How to Conduct Software Condition-Based Testing?

**Step #1: Come out a program flow graph as a test model for a given program (i.e. a function in C++/Java).**

**Step #2: Identify predicate nodes with a compound Boolean condition in a program flow graph. Identify an independent path for each predicate node and related branches.**

**Step #3: Create a T/F condition table for each compound Boolean condition, including all of possible outcomes with diverse combinational inputs.**

A compound condition:
→ A AND B or C

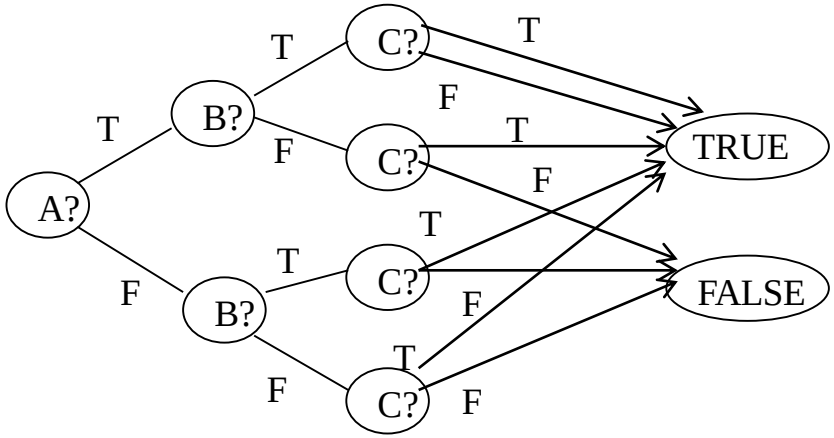
| A | B | C | A AND B or C |
|---|---|---|---|
| TRUE | TRUE | TRUE | TRUE |
| TRUE | TRUE | FALSE | TRUE |
| TRUE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | FALSE |
| FALSE | TRUE | TRUE | TRUE |
| FALSE | TRUE | FALSE | FALSE |
| FALSE | FALSE | TRUE | TRUE |
| FALSE | FALSE | FALSE | FALSE |

# TOPIC #3 – SOFTWARE BRANCH TESTING

## How to Conduct Software Branch Testing?

| A | B | C | A&B or C |
|---|---|---|---|
| TRUE | TRUE | TRUE | TRUE |
| TRUE | TRUE | FALSE | TRUE |
| TRUE | FALSE | TRUE | TRUE |
| TRUE | FALSE | FALSE | FALSE |
| FALSE | TRUE | TRUE | TRUE |
| FALSE | TRUE | FALSE | FALSE |
| FALSE | FALSE | TRUE | TRUE |
| FALSE | FALSE | FALSE | FALSE |

A Compound Condition: ( A AND B Or C )



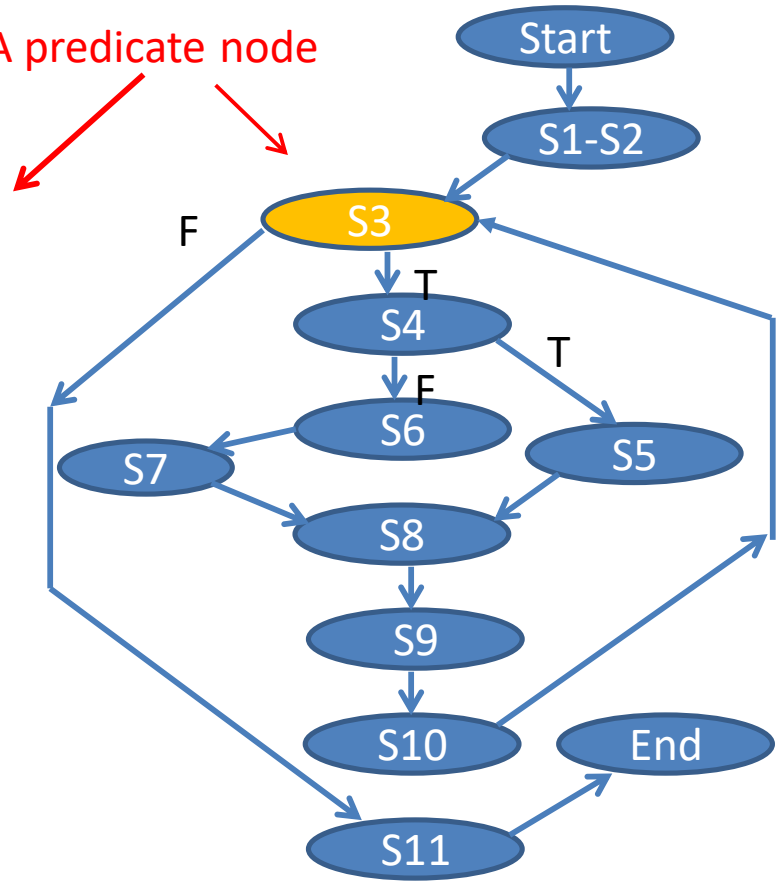

**Step #4: Identify one independent executable path to cover one target predicate node and its branch to cover one condition table entry.**

**Step #5: Find test data and expected test result for this path. Continue Step #4 until covering the rest of condition table entries.**

# A Condition-Based Testing Example

**Step #1: Create Program Flow Graph**

```
/* Condition Testing Example*/
S1   READ Length;
S2   READ Count;
S3   WHILE (Count < 6)AND (Length <200) LOOP
S4         IF (Length > 100) THEN
S5            Length = Length - 2;
S6         ELSE
S7            Length = Count * Length;
S8         END IF
S9      Count = Count + 1;
S10  END LOOP;
S11  PRINT Length;
```




# Software Condition-Based Testing Example

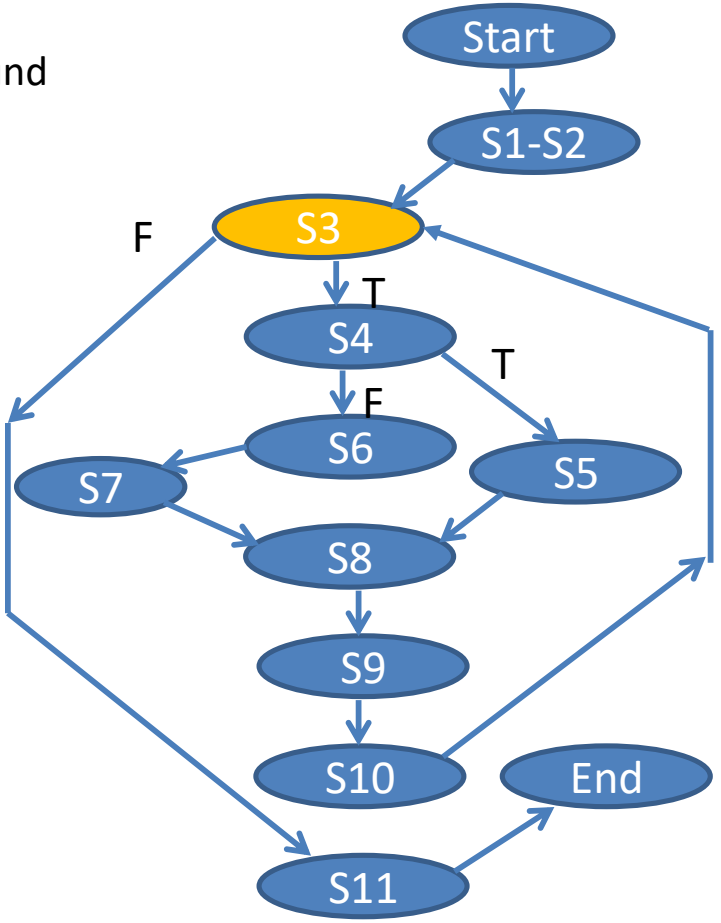
**Step #2:** Identify predicate nodes with a compound Boolean condition in a program flow graph.

Identify an independent path for each predicate node and related branches.

Compound Boolean Condition:
(Count < 6)AND (Length <200)

---

A= (Count < 6)
B= (Length < 200)
C= (Count < 6) AND (Length <200)



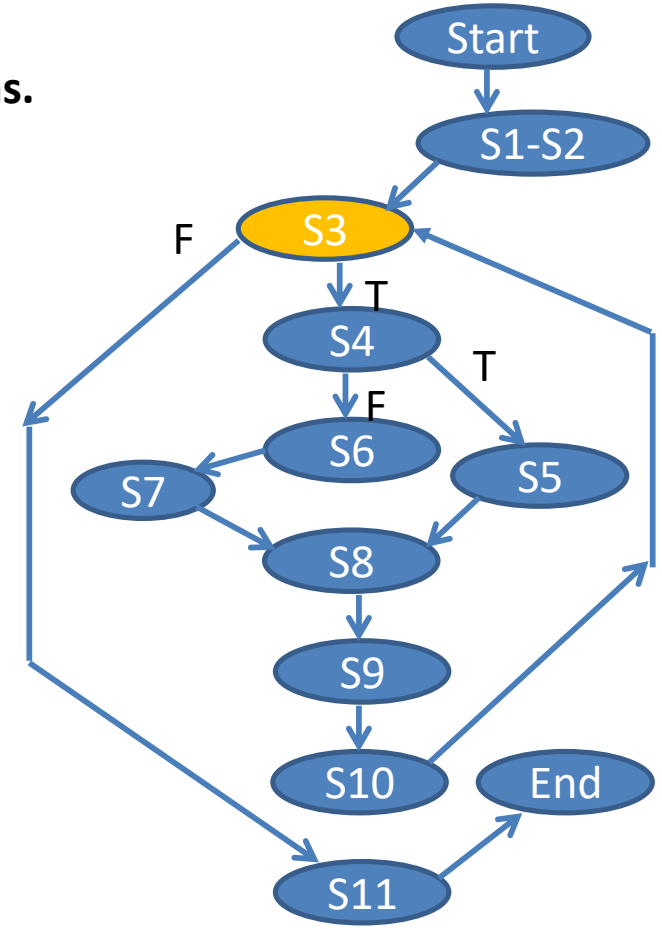

# TOPIC #4 – SOFTWARE CONDITION-BASED TESTING

## Software Condition-Based Testing Example

**Step #3: Create one Independent Path for each predicate node with compound conditions.**

| Predicate Node | Decision | Possible Outcome | Path |
|---|---|---|---|
| S3 | Count < 6 and Length<200 | F | P1, P2 |
| | | T | P2 |

P1: Start→S1-S2→S3→S11→End
P2: Start→S1-S2→S3→S4→S5→S8
→S9→S10→S3→S11→End

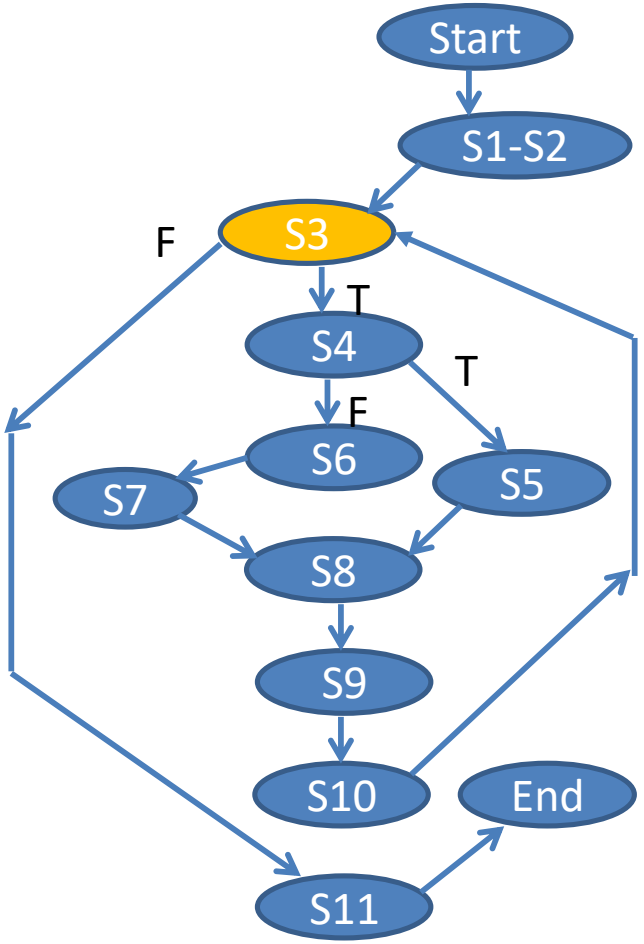



# Software Condition-Based Testing Example

**Step #4: Create a T/F Table for each compound condition**

| Predicate Node | Decision | Possible Outcome | Path |
|---|---|---|---|
| S3 | Count < 6 and Length<200 | F | P1, P2 |
| | | T | P2. |

| A: Count < 6 | B: Length < 200 | C: A AND B |
|---|---|---|
| TRUE | TRUE | TRUE |
| TRUE | TRUE | FALSE |
| TRUE | FALSE | FALSE |
| TRUE | FALSE | FALSE |



# Software Condition-based Testing Example

**Step #5: Create a set of tests for each predicate with component conditions. Each test case covers one entry of the corresponding T/F Table.**

| Path ID | Test ID | Branch Outcome | Inputs | A: Count <6 | B: Length <200 | C: A and B | Expected Outputs |
|---|---|---|---|---|---|---|---|
| Path 1 | T1 | False | Count = 7<br>Length = 10 | F | T | F | Length = 10 |
| Path 1 | T2 | False | Count = 5<br>Length = 10 | T | F | F | Length = 60 |
| Path 2 | T3 | True | Count = 5<br>Length = 10 | T | T | T | Length = 50 |
| Path 1 | T4 | False | Count = 7<br>Length = 200 | F | F | F | Length =200 |

# Software Condition-Based Testing Coverage

**What has been covered by Condition-Based Testing?**

- Cover each predicate node in  a program flow graph.

- Cover each branch link (or edge) in a program flow graph.

- Cover each combinational case in a compound Boolean condition of each predicate node in a program flow graph.

**What has not been covered by Condition-Based Testing?**

- This method can not assure the statement coverage (or known as node coverage) in a program.