

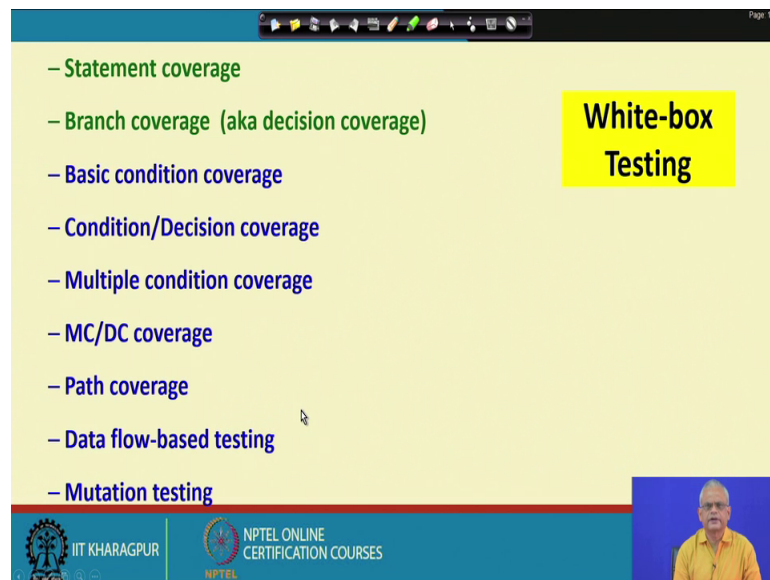
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 56
Condition Testing

Welcome to this lecture. In the last couple of lectures, we have been discussing about the white box testing. And, we had said that white box testing has to be carried out on a unit, even though we are also performing black box testing. And, we had said that there are several types of white box tests strategies. And, we had discussed about statement coverage based testing and then the branch coverage based testing.

If you remember said that branch coverage is a stronger form of testing, if we are doing branch coverage testing, then we need not do the statement coverage testing. But is branch coverage testing good enough or in other words are there stronger testing strategies, which we need to carry out and if we carry out those branch or decision testing becomes not necessary. Let us see the other types of testing.

(Refer Slide Time: 01:34)



White-box Testing

- Statement coverage
- Branch coverage (aka decision coverage)
- Basic condition coverage
- Condition/Decision coverage
- Multiple condition coverage
- MC/DC coverage
- Path coverage
- Data flow-based testing
- Mutation testing

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

In the last few lectures discussed about statement coverage based testing and then we had discussed about branch coverage testing, which is also popularly known as the decision coverage testing. Now, we will look at the basic condition coverage testing. And, we will later after basic coverage condition testing. We will look at the condition decision

coverage based testing. And, then we will look at the multiple condition coverage testing, MC DC coverage testing, path coverage testing, and then the data flow coverage testing, and finally, the mutation testing.

(Refer Slide Time: 02:27)

The slide is titled "All Branches can still miss testing specific conditions". It contains the following text:

- Assume failure occurs when `c==DIGIT`
- `if((c == ALPHABET) || (c == DIGIT))` (The expression `(c == DIGIT)` is circled in red, and a handwritten note `c = splchar` is written next to it. To the right, there are handwritten letters 'T' and 'F' with arrows pointing to the condition.)
- Branch adequacy criterion can be satisfied by `c==alphabet` and `c==splchar`
- **The faulty sub-expression might not be tested!**
- Even though we test both outcomes of the branch

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a speaker is visible in the bottom right corner.

First, let us address the question that each is the branch coverage testing good enough or to tell that in other words. If, all branches have been tested can we miss some specific test conditions or can bugs remain, even after we have completed branch coverage testing or decision coverage testing.

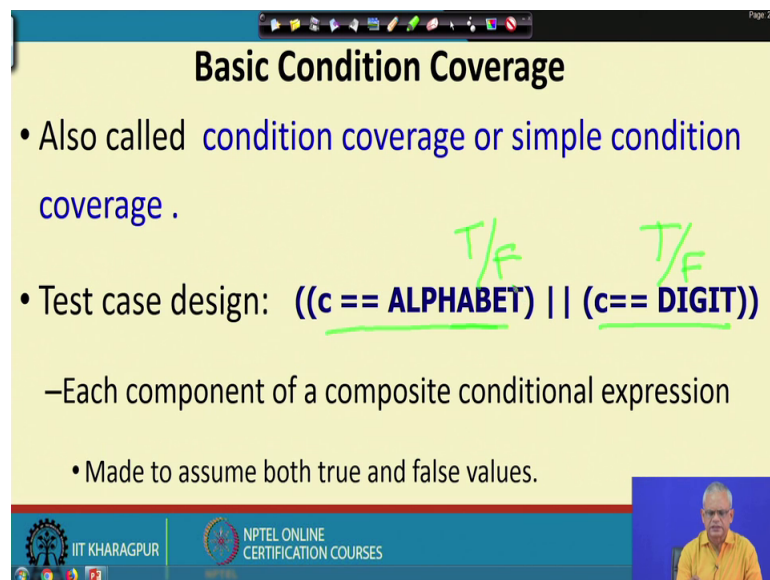
Let us take a simple example; let us assume that we have a conditional statement if c is equal to alphabet or c is equal to digit, where c is a variable character variable and we have read c from the keyboard. And, then we are checking here, if c is either an alphabet or c is a digit we were doing some special processing for that. Maybe we are recovering the integer value of c. If it is a digit we will c is equal to C minus A let us say, which will retrieve the integer value.

But, then if we are doing the branch testing we need to have the branch condition assume true and false values, to make it true if we have c is a alphabet that is capital A to capital Z and small a to small z. Then, if it becomes true this condition does not matter because this is a or condition anyone of that becoming true will make the branch condition true. Now, let us say that the second test case is a special character like a control character or something.

So, now the expression when we evaluate c is not a alphabet and c is not a digit, because it is a special character. And therefore, it becomes false, but the fault is when c is equal to digit we are not doing it properly let us say, we have made it c is equal to c minus small a . So, we made a small mistake here, to get the integer value we made a small mistake, but then our 2 test cases, that is c is a alphabet and c is a special character achieves decision coverage. And, the faulty sub expression, which is c equal to digit that is not tested. And the bug remains, even if we are achieving decision coverage testing both the outcomes of the branch condition, but still we are not able to detect the bug and that the reason for that is that, we are not considering this sub expression becoming true any time.

If, we had a test case which not only turns the first sub expression true and false we will also should have test cases making, the second sub expression true and false and in that case we could have detected the bug. So, that gives us a hint that we should look at the sub expressions. And, we should ensure that the test case makes it sub expression assumed true and false values. And, exactly based on that idea the basic condition coverage testing is designed. Let us look at the basic condition coverage testing.

(Refer Slide Time: 07:04)



Basic Condition Coverage

- Also called **condition coverage** or **simple condition coverage** .
- Test case design: $((c == \text{ALPHABET}) \mid\mid (c == \text{DIGIT}))$
—Each component of a composite conditional expression
 - Made to assume both true and false values.

The slide includes a video inset of a presenter in the bottom right corner. The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

In the basic condition coverage testing for a branch condition having multiple clauses, which we will call as a complex condition; each of the sub expressions or the clauses, we will require them to achieve true and false values. And, you can see that if we make that

then the bug which was earlier not getting caught will get caught, but then is it a good testing is it first of all is it stronger than the statement coverage, is this kind of testing the basic condition coverage testing, is it stronger than simple decision coverage testing. Let us try to answer those questions and is it a good testing strategy let us try to answer those questions.

(Refer Slide Time: 08:11)

Basic Condition Testing

- **Simple or (basic) Condition Testing:**
 - Test cases make each atomic condition assume T and F values
 - Example: if (a > 10 && b < 50) (with handwritten '5' above 'a > 10' and '30' above 'b < 50')
- **Following test inputs would achieve basic condition coverage**
 - a=15, b=30 ✓
 - a=5, b=60 ✓
- Does basic condition coverage subsume decision coverage?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

First let us see, how do we design the basic condition testing strategies? Let us take a example conditional expression, if a is equal to 10 and b is less than 50. And, our objective is that each of the sub expressions will achieve true and false values. Now to make a is greater than 10 to make it true, we will make a is 15. And, to make b is less than 30, we will assign some value b is equal to 30 or something like that. And therefore, our first test case is a is 15 b is equal to 30.

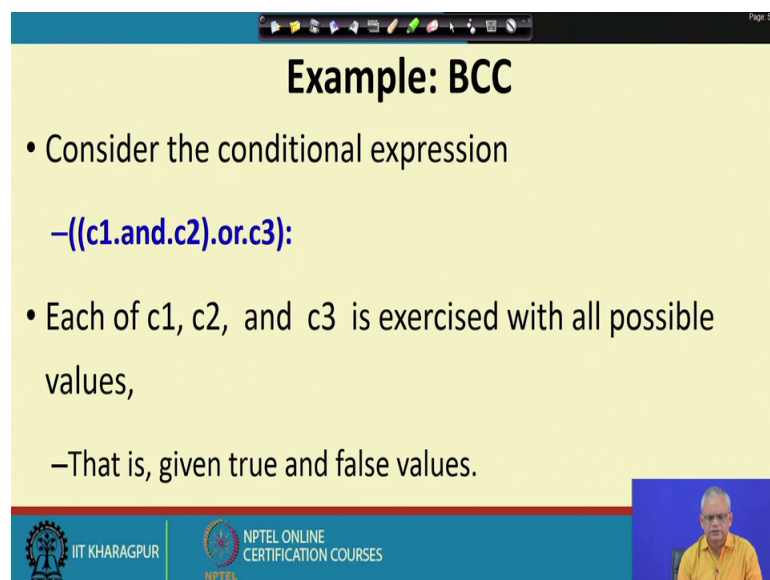
So, this the test case one which will make the 2 sub expressions true. Now, we need another test case which will make both the sub expressions false. So, let us choose a is equal to 5, 5 is less than 10. And therefore, this is false and let us say b is 60 60 is greater than 50. And therefore, the second sub expression is also false. So, these two test cases will achieve the simple condition testing or the basic condition testing, where each sub expression is made into true and false values.

But, can we always perform basic condition testing, yes we can perform basic condition testing and if the 2 sub expressions are independent. If there is a dependency between

this we may or may not achieve basic condition testing. Let us say a is greater than 10 and a is less than 5, that we cannot for that condition there is a dependency between two sub expressions and it becomes difficult to design the test case it cannot be at the same time greater than 10 and a is less than 5.

But, if we have independent clauses, we should be able to design basic condition testing and how many test cases would be needed? In one test case, we can assign true to all and another test case, we can assign false to all. So, 2 test cases would be good enough to achieve basic condition testing, but how does this testing. Compare with the branch coverage and the statement coverage testing; let us try to examine that.

(Refer Slide Time: 11:49)



The slide is titled "Example: BCC" and contains the following text:

- Consider the conditional expression
 $\neg((c1.and.c2).or.c3)$:
- Each of $c1$, $c2$, and $c3$ is exercised with all possible values,
–That is, given true and false values.

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a presenter in the bottom right corner.

There is another example; before we check how effective it is there is one more example let us say we have $c1$ and $c2$ or $c3$. So, there are 3 clauses here and 3 sub expressions or clauses. And, we will make each one of this true and false. Even though there maybe 3 4 5 whatever, if they are independent then we can design test cases such that all are true and all are false. So, each sub expression gets true and false values. So, basic condition coverage is satisfied, but we just use 2 test cases.

(Refer Slide Time: 12:48)

Basic condition testing

- **Adequacy criterion:** each basic condition must be executed at least once
- **Coverage:**
$$\frac{\text{\# truth values taken by all basic conditions}}{2 * \text{\# basic conditions}} = \frac{4}{6} = \frac{2}{3}$$

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, if we use some test cases to test a branch condition, what is the percentage of coverage achieved? Let us define a metric, which we call as the BCC metric, Basic Condition Coverage metric. And, here we have this simple expression, the number of truth values that is true and false taken by all basic conditions and 2 into number of basic conditions.

So, if we have a expression like a is equal to 10 or b greater than 5 and c less than 100, we can see here that there are 3 atomic conditions or the basic conditions and then possible outcomes of each of these is 2 into 3 so that is 6. Now, let us say we designed a test case which made this true false false, the second test case made true true false. So, this is the first test case outcome this is the second test case, and using two test cases we achieved this coverage of the basic conditions.

And, then what is the percentage coverage achieved. So, we achieved true false false and also true value for the second condition. So, we have achieved 4 of the decisions out of the 6. So, the coverage will be 2 by 3.

(Refer Slide Time: 15:12)

The slide is titled "Is BCC Stronger than Decision Coverage?". It contains the following text:

- Consider the conditional statement:
`-If(((a>5).and.(b<3)).or.(c==0)) a=10;`
- Two test cases can achieve basic condition coverage: (a=10, b=2, c=2) and (a=1, b=10, c=0)
- **BCC does not imply Decision coverage and vice versa**

The code snippet has a truth table diagram overlaid on it. The diagram shows the truth values for each sub-expression and the overall condition. For the first test case (a=10, b=2, c=2), (a>5) is true, (b<3) is false, and (c==0) is false. For the second test case (a=1, b=10, c=0), (a>5) is false, (b<3) is false, and (c==0) is true. The overall condition is true for the first test case and false for the second.

The slide footer includes the IIT KHARAGPUR logo and the NPTEL ONLINE CERTIFICATION COURSES logo. A small video inset shows a man in a yellow shirt.

But, is it a strong enough testing how does it compare with the decision coverage and statement coverage. Let us take an example, if a greater than 5 and b less than 3 or c equal to 0 and then we have some action here, a is equal to 10. And we can use 2 test cases to achieve basic condition coverage, let us say a is equal to 10 if this is true b is equal to 2 this is true and c is equal to 2 this is false.

So, the first test case achieves true true and false. The second test case a is equal to 1 which is false, b is equal to 10 which is false, and c equal to 0 which is true. Now, we have achieved basic condition coverage using these 2 test cases. So, each of the sub expressions achieves both true and false values, but then let us see if decision coverage is obtained true and true. So, this clause will become true and irrespective of this even though it is false. So, the branch outcome will be true. Now, let us see the second one false and false. So, the clause here will become false.

But, then the second sub expression becomes true. So, the outcome is again true. So, even though the 2 test cases are able to achieve basic condition coverage, but they are not able to achieve the decision coverage. From this we can conclude that even though basic condition coverage may be satisfied, but still decision coverage may not be obtained. Or in other words we cannot say that basic condition coverage is a stronger form of testing than decision testing.

But is decision coverage testing stronger than, the basic condition coverage no that is also not true, because you had already seen that basic the even though the decision coverage was achieved for c is equal to alphabet or c is equal to digit. We could achieve the decision coverage, but that did not achieve the basic condition coverage. We can therefore, say that the basic condition coverage and decision coverage these are complimentary, they are not really comparable testings one cannot really is stronger than the other testing.

(Refer Slide Time: 18:35)

The slide is titled "Condition/Decision Coverage Testing". It contains two main bullet points:

- Condition/decision coverage:** (with handwritten notes "SC", "BCC", "BC", and "SC" next to it)
 - Each atomic condition made to assume both T and F values
 - Decisions are also made to get T and F values
- Multiple condition coverage (MCC):** (with handwritten notes "2^n" and a truth table for "if(a>5) || (b<20)"):

		if(a>5) (b<20)	
		T	F
T	T	T	F
	F	F	T
F	T	T	F
	F	F	T

 - Atomic conditions made to assume all possible combinations of truth values

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset in the bottom right corner shows a man in a yellow shirt speaking.

Now, let us see the condition decision coverage testing, but let me just pose another question to you that the basic condition coverage testing and the branch coverage testing these 2 are not really comparable. We cannot say that any of that any of these 2 is stronger than the other, but what about the statement coverage testing? We know that branch coverage testing is stronger than statement coverage testing, but what about the basic condition coverage and the statement testing. Can we say that basic condition coverage is stronger than statement coverage testing or is it vice versa that statement coverage testing is stronger than basic coverage testing?

Of course, from common sense we can say that statement coverage testing cannot be stronger than basic condition coverage testing, because one which is stronger than the statement coverage that also is not stronger than basic condition testing.

But, what about this basic condition coverage does it achieve statement coverage. Please try to reason it out give examples or prove, what is the relation between basic condition coverage and statement coverage testing? But, our idea while discussing about the branch coverage testing was to come up with a test strategy, which is stronger than branch coverage testing, but at the same time achieves condition coverage. And basic condition coverage could not do that, can we impose the condition that we should achieve the basic condition coverage and also the decision coverage.

So, that is the strategy that we will discuss now, it is called as condition decision coverage. The condition decision coverage strategy here, we achieve basic condition coverage and also the decision coverage. And, here each of the atomic condition is made to assume true and false values, that is the basic condition coverage achieved and also the decision as a whole is also made to achieve true and false values. So, we need test cases for those. And therefore, this is a stronger testing than the basic condition testing or the branch coverage testing because it achieves both of this.

But is it the strongest testing, we can see that this is stronger than both branch testing and the basic condition coverage, but is it the strongest no the strongest is multiple condition coverage. Here all possible atomic conditions, the atomic conditions are made to assume all possible combinations of truth values. If, we have if a greater than 5 or b less than 20, in that case our test cases should make this true this false this true true false true and false false.

In general, if we have n sub expression or clauses in a complex decision statement, then we will need 2 to the power n test cases to achieve the multiple condition coverage. So, the multiple condition coverage testing should ensure that all the sub expressions, they achieve all possible combinations of truth values.

(Refer Slide Time: 23:33)

MCC

- Test cases make Conditions to assume all possible combinations of truth values.
- Consider: **if (a || b && c) then ...**

Test	a	b	c
(1)	T	T	T
(2)	T	T	F
(3)	T	F	T
(4)	T	F	F
(5)	F	T	T
(6)	F	T	F
(7)	F	F	T
(8)	F	F	F

Exponential in the number of basic conditions

Handwritten notes: 20 Test Cases, $2^3 = 8$

Now, let us look at an example, if we have this example a or b and c. we have 3 sub expressions, 3 Boolean values and if we need to achieve the multiple condition coverage how many test cases do we need? We need 2 to the power 3 test cases, which is 8. Let us see what will be the test cases; a b c these are the 3 sub expressions or Boolean variables and then all possible combinations the truth outcomes of these 3 sub expressions or Boolean variables true true true true true false and so on.

And, there are 2 to the power 3; 2 to the power 3 is 8 test cases are needed. We can say that to achieve the multiple condition coverage the number of test cases is required is exponential in the number of the sub expressions or the basic atomic conditions.

But, then this can be huge because in many applications. Especially in the controller embedded control applications in user interfaces etcetera. It is quite common to have expressions involving 10 or 20 sub expressions. And, if we have 20 sub expressions in a expression in in a decision statement, we will need 2 to the power 20 test cases or which is a million test case. Just imagine that just to test one branch we need a million test cases to achieve multiple condition coverage, but if a unit has 10 of that the number of test case will be enormous, even million is a large number a tester would have to spend couple of years to run the million test cases.

We can therefore, say that the multiple condition coverage is a strong form of testing when there are branches, but then it is impractical, because it requires too many test

cases when we have multiple clauses or multiple sub expressions 10 or more or even 8 7 this also lead to large number of test cases.

So, can we have a compromise a test strategy which will achieve as thorough testing as the multiple condition coverage, but then the number of test cases should be few for expression it should be 3 4 5 something like that should not be 1000s. Now, we are almost at the end of the lecture let us stop here and we will continue from here in the next class.

Thank you.