

Software Testing
Prof. Meenakshi D'Souza
Department of Computer Science and Engineering
Indian Institute of Information Technology, Bangalore



Lecture - 22
Logic Coverage Criteria

Hello again, we are in the fifth week, this is the third lecture. Last class I introduced you to the basics of logic, we saw propositional logic, predicate logic. I gave you propositional logic mainly as a recap and as an entity to introduce the logical connectors; negation, conjunction, disjunction, implication and equivalence. The logic that we will be using throughout logic based testing would be predicate logic. So, today what we will begin with is to understand various coverage criteria in terms of logic like we did for graphs now we will focus on logic.

(Refer Slide Time: 00:44)

Re-cap: Predicates and clauses

- A **predicate** is an expression that evaluates to a Boolean value.
- E.g.: $(x > y) \vee C \vee f(z)$ is a predicate where x, y and z are non-Boolean variables, C is a Boolean variable and $f(z)$ is a function that returns a Boolean constant.
- A **clause** is a predicate that does not contain any logical operators.
- E.g., in the above predicate, there are three clauses:
 $x > y, C$ and $f(z)$.

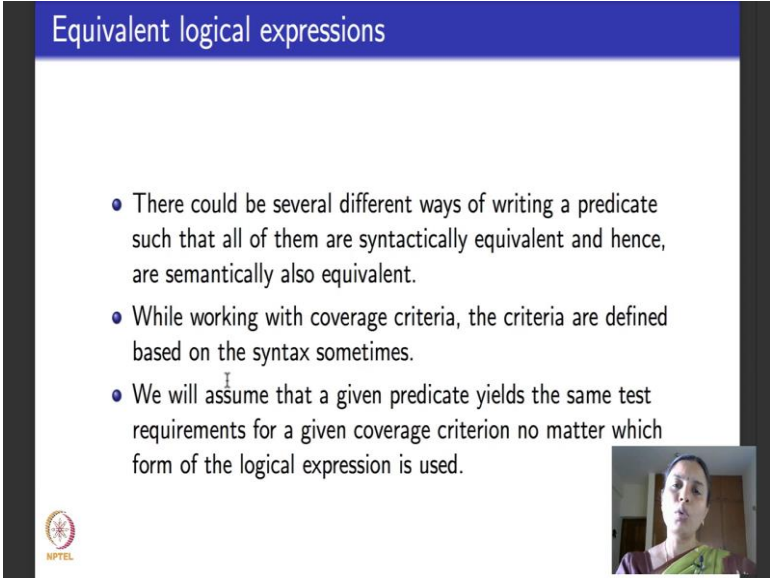


And as I told you, we will work with predicates and clauses. So, I will recap the definition of predicates and clauses that we saw from the last lecture what is a predicate a predicate is any expression any formula that evaluates to a true or a false value.

Here is an example. So, that this is a predicate which has 3 clauses: x is greater than y or C or f of z . x, y and z are some variables, we assume that they are not Boolean. x is greater than y will result in a Boolean value, C is a standalone clause. So, C better be a Boolean entity f of z is directly 'or'ed with these 2 other clauses. So, f of z better be a

function that takes z and return either true or false. So, x is greater than y will be either true or false, C is a Boolean variable, f of z is a function that returns a Boolean value. So, I can 'or' these 3 and get a true or false value for this predicate. Each of these 3 entities in this predicate are what are called clauses. A clause is a predicate that does not contain any logical operators.

(Refer Slide Time: 02:03)



The slide has a blue header with the text "Equivalent logical expressions". Below the header, there are three bullet points:

- There could be several different ways of writing a predicate such that all of them are syntactically equivalent and hence, are semantically also equivalent.
- While working with coverage criteria, the criteria are defined based on the syntax sometimes.
- We will assume that a given predicate yields the same test requirements for a given coverage criterion no matter which form of the logical expression is used.



In the bottom left corner, there is a small circular logo with the text "NPTEL" below it. In the bottom right corner, there is a small video inset showing a woman with dark hair, wearing a green and purple sari, speaking.

So, now before we move on and look at logical coverage criteria, I would like to spend little time making you notice one point. So, as we saw in the last class and even otherwise you might know from logic that the same formula, well formed formula can be written using several different equivalent ways. And when we look at coverage criteria, the clauses and the way the clauses react could be different for each of the equivalent ways, but what we assume throughout this lectures where we deal with coverage criteria is that we assume that the given predicate will yield the same TR for a given criteria no matter which form of the logical expression is used. For example, I could write a implies b directly using implies or I could write a implies b as $\text{not } a$ or b , but I assume that these are semantically equivalent and as for as coverage criteria is concerned I fix one of them and carry on.

(Refer Slide Time: 02:58)

Predicates and clauses: Notations

- Let P be a set of predicates and C be a set of clauses in the predicates in P .
- For each predicate $p \in P$, let C_p be the clauses in p .
 $C_p = \{c \mid c \in p\}$.
- $C = \bigcup_{p \in P} C_p$.




So, here are some notations that we will be using for the rest of this lecture and next lecture. We assume that we are dealing with a particular piece of software artifact, could be code or it could be requirement given as a finite state machine and that software artifact has a set of predicates. So, we are dealing with a universal set of predicates that come from a given software artifact. Like capital P be that set and C is the set of all clauses that come in the predicates in p . For each predicate p in P the clauses specific to the predicate small p we denote it by the set C suffix p C_p , right. So, what is the set C suffix p , it is a set of all clauses that occur in the predicate p C this is a set of all clauses in the set of predicates p is the union of all the clauses as they occur in each of the predicates in p . We deal in several logical coverage criteria. The first one that we deal with is the most elementary one it is called predicate coverage.

(Refer Slide Time: 03:51)

Predicate coverage

- **Predicate coverage (PC):** For each $p \in P$, TR contains two requirements: p evaluates to true and p evaluates to false.
- E.g., for the predicate $(x > y) \vee C \vee f(z)$, the two tests that satisfy predicate coverage will be $x = 5, y = 3, C = \text{true}, f(z) = \text{false}$ (the predicate evaluates to true) and $x = 1, y = 4, C = \text{false}, f(z) = \text{false}$ (the predicate evaluates to false).
- Overlap with graph coverage criteria: For a set of predicates associated with branches, predicate coverage is the same as edge coverage.



So, if you see the English sense of this term what is predicate coverage mean ? It means cover the predicate, what does it mean to cover the predicate in turn ? Predicate, as we saw is any formula that results or evaluates to a Boolean value. So, when I cover the predicate I am insisting that I am giving one set of values that make the predicate true and one set of values that make the predicate false. So, predicate coverage says that for each predicate p in this set of universal predicates P , my TR or test requirement contains 2 requirements: one requirement insists that the predicate p evaluates to true and the other says p evaluates to false.

For example, if we take this predicate, x greater than y or C or f of z that we saw in the previous slide this whole thing is one predicate p . I want to be able to do predicate coverage for this predicate p which means my TR says that there are 2 test requirements: one that makes this whole predicate true and one that makes this whole predicate false. Now this is my TR. So, what would be the set of test cases that will satisfy this test requirement. Set of test cases that will satisfy this test requirement will be one for each kind of TR. So, I need to write a set of test cases that will make the predicate true and I will write a set of test cases that make the predicate false. So, remember this is a 'or' predicate. So, it is true if any one of these clauses in this predicate become true and it is false if each of the clauses in this predicate become false.

So, to make it true I can assign values for any of these variables such that one of the clauses become true. For example, here is one assignment of value that will make the predicate true. I have assigned x to be 5 y as 3. So, 5 is greater than 3, x is greater than y, this predicate value is to be true. Remember this is enough to make the entire predicate true, but for complete test case we need to go ahead and assign values for all the other variables also. So, here I have assigned C to be true and f of z to be false. So, combining and substituting back these values you realize that 5 is greater than 3. So, x greater than y will be true; true will be 'or'ed with C which is true which in turn will be 'or'ed with f of z which is false. So, it will be true or true or false which will make the entire predicate p to be true.

Now, I need another test case that will make this predicate false because my goal is to be able to do predicate coverage. So, here is such a test case. I assign x as 1, y as 4. So, x is not greater than y. So, this predicate this clause evaluates to be false and I assign C to be false. So, the second clause also becomes false and I make f such that f of z returns false. So, the third clause also evaluates to false. Since all 3 are false the entire predicate becomes false.

Here is a small observation. Now if you go back and look at the graph coverage criteria that we saw predicate coverage has an overlap with graph coverage criteria. If you remember we saw this edge coverage criteria when we did graph coverage which was also called branch coverage. Assuming that every branch in the graph every node corresponding to a branch in a graph is labeled by a predicate, predicate coverage for that predicate will be the same as doing edge coverage for that branch because there will be one set of test cases that will make the predicate true making it take the then part of the branch and there will be another set of test values that will make the predicate false making it take the else part of the branch. So, there will be 2 edges coming out of the node that corresponds to this predicate and it will mean edge coverage for each of these 2 nodes. So, predicate coverage is the same as edge coverage.

Now what is the disadvantage in predicate coverage? If you go back and see this example x greater than y or C or f of z this predicate coverage tests this entire predicate as a whole. So, and if we look at the test values that we assigned to achieve predicate coverage for that predicate in both these set of test cases in the first one and in the second one we had assigned f of z to be false. So, it so, happens that we have tested predicate

coverage, but we did not really exercise the clause f of z being true. It could have been the case that there could have been an error in the software because the clause f of z was true. So, it is not wise to just do predicate coverage and say I have given true and false values to make the entire predicate true once and to make the entire predicate false once and leave it that because we can do it as we did it in this example without exercising each clause. And you never know, may be the clause that wasn't exercised for that particular value had an error in it.

In this case, it could be the case that if f of z was true may be the software had an error. So its not wise just to do predicate coverage. What we would to move on is the next coverage criteria which is called clause coverage. What does clause coverage say? Clause coverage is abbreviated as CC, it says for each clause in the set of all clauses TR contains 2 requirements--- that clause evaluates to true that clause evaluates to false. So, it says exercise each clause and make it true once make it false once. That is what clause coverage is said to do. So, if you see this example, it had 3 clauses: x greater than y , C and f of z . So, what does clause coverage mean? Clause coverage will mean that you make the first clause x greater than y true once and false once. So, I have made this true once because 5, x I have assigned x to be 5, y to be 3 and x greater than y is true and here we have assigned x to be 5, y to be 6, so, x greater than y is false. So, I made the first clause true once and false once, C I have made true once false once, f of z I have made true once and false once. So, this is a set of test cases that achieve clause coverage criteria on a predicate.

So, just to recap predicate coverage says make the predicate true once false once. Predicate coverage need not exercise all the clauses in the predicate. That is not desirable from the point of view of testing because a particular clause that is not exercised for a particular value could contain an error. So, we define another coverage criteria called clause coverage which says you exercise each clause to be true once and to be false once. So, that is what we did for this example.

Now, just for now let us say, let us try to understand what will happen to the predicate for the first assignment of clauses. If you see first assignment of clauses makes all the 3 predicates true. So the entire, I mean, makes all the 3 clauses true. So, the entire predicate will be true second one makes all the 3 clauses false. So, the entire predicate is false. So, in this case clause coverage also happened to achieve predicate coverage, the

set of test cases that we wrote for clause coverage also happened to satisfy predicate coverage.


(Refer Slide Time: 11:48)

Predicate coverage vs. clause coverage

- Clause coverage does not subsume predicate coverage.
- Consider $p = a \vee b$. The clauses are a and b . There are four test inputs that consider all combinations of true/false values for a and b .

	a	b	$a \vee b$
1	T	T	T
2	T	F	T
3	F	T	T
4	F	F	F

- The test set $\{2, 3\}$ above satisfies clause coverage but the predicate is true in both the cases. Hence, predicate coverage is not satisfied.



But that need not be the case in general. Clause coverage in general need not subsume predicate coverage. The notion of subsumption is the same as that we saw for graph coverage criteria. We say one coverage criteria subsumes the other if every set of test cases that satisfy coverage criteria 1 also satisfies coverage criteria 2.

So, the claim now is that clause coverage does not subsume predicate coverage. Why is that so? Here is an example. Consider the predicate p is equal to a or b it has 2 clauses a and b and here is the entire truth table for that predicate what I have done is I have assigned true false values to a and b and in the last column we have tabulated when a or b will become true. This just happens to be directly the truth table for or because we do not have anything else. Now let us say I want to achieve clause coverage for this predicate a or b . So, I make each clause a and clause b true once and false once. So, I chose rows 2 and 3 for that. If you see in row 2 a is made true and in row 3 a is made false, in row 2 b is made false and in row 3 b is made true. So, between rows 2 and 3, I have made a true and false once and I have made b also true and false once.

So, I have satisfied clause coverage by picking up these 2 values for a and b . Now look at the entire predicate a or b . For both row 2 and row 3 the predicate evaluates to true. So, the predicate, I have not given a test case where the predicate has evaluated to false,

right. So, predicate coverage is not satisfied by choose choosing these 2. So, I can say by using this as an example or as a counter example to say that clause coverage does not subsume predicate coverage. In fact, for the same example, if I had chosen rows 1 and 4 which makes a true once false once, b true once false, then I would have achieved predicate coverage. I would have made the whole predicate true once and false once, but the purpose of this example is not to illustrate that. The purpose of this example is to illustrate that it is possible to have combination of values such that clause coverage is satisfied.


But predicate coverage is not met and these 2 values for rows 2 and 3 for the clauses a and b happen to be that. So, in general I cannot say the clause coverage always subsumes predicate coverage, it is a wrong to sayso.

(Refer Slide Time: 14:35)

Combinatorial coverage

- **Combinatorial coverage (CoC):** For each $p \in P$, TR contains test requirements for the clauses in C_p to evaluate to each possible combination of truth values.
- Combinatorial coverage is commonly called as **multiple condition coverage**.
- For the predicate $p = a \vee b$, combinatorial coverage will test all the values in the truth table below.

	a	b	$a \vee b$
1	T	T	T
2	T	F	T
3	F	T	T
4	F	F	F

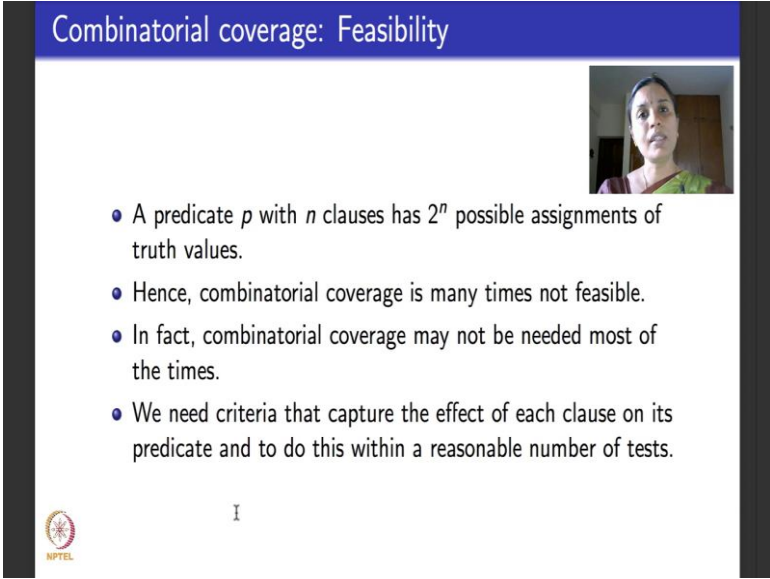


So, the next one would be because there are these combinations where some combination of clauses do not subsume predicates the next one would be to give up and say you test for every possible combination of clause that can occur in the truth table of the given predicate. So, that leads to a coverage criteria called combinatorial coverage abbreviated as CoC.

So, what is combinatorial coverage define? It says for each predicate p in P, TR contains test requirements for the clauses in the C_p to evaluate to each possible combination of truth values. So, if I have, basically what it says is given a predicate p you write out the

truth table for p for each clause in p evaluating to be true or false and then you test for the entire truth table of p . So, if I have p is equal to a or b , there are 4 test cases that I give I met a ; a to be true along with b to be true, then a true b false, a false b true, a false and b false again right, and I test for all possible true false values of the clauses. Now it is not too surprising to see that this might lead to a lot of test cases.

(Refer Slide Time: 15:50)



Combinatorial coverage: Feasibility

- A predicate p with n clauses has 2^n possible assignments of truth values.
- Hence, combinatorial coverage is many times not feasible.
- In fact, combinatorial coverage may not be needed most of the times.
- We need criteria that capture the effect of each clause on its predicate and to do this within a reasonable number of tests.

I

NPTL

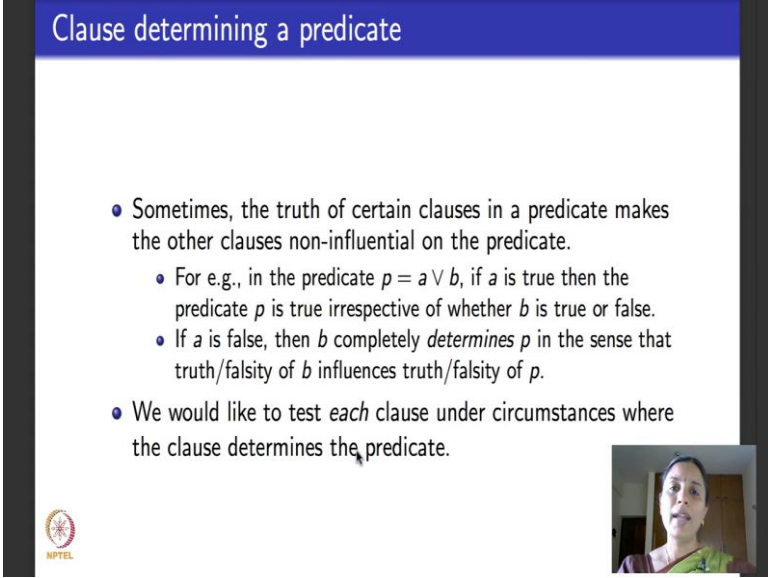
In fact, if I have a predicate with n clauses right and because I am testing against the entire truth table and the truth table has 2 power n rows in the worst case, I will end up getting 2 power n test cases for a predicate with n clauses. That is an exponential number of test cases for a predicate with n clauses. In testing ideally we would like to the test cases to be as minimum in number as possible, but effective in finding faults if they exist. There is no point in testing it for exponential number of combination of clauses and be exhaustive. Of course, this is like exhaustive testing, but it may not be possible to do exhaustive testing all the time and it is not necessary also.

What we are looking for are some kind of criteria that capture the effect of each clause on the predicate and to be able to do this without considering all the combinations of clauses right. Like for example, if you go back to this it so happens that if I choose values for clauses a and b from rows 2 and 3, that combination of values did not really affect the truth or falsity of the predicate. The predicate happened to be true in both the cases. But if I had chosen values 1 and 4 then I was effective in not only achieving clause

coverage, but I was also effective in achieving predicate coverage. I chose different true false values for each of the clauses and as a result I made the entire predicate true once and I made the entire predicate false once. So, that is our ideal goal.

Our ideal goal is to be able to get criteria that capture the effect of each clause on the predicate and to be able to do it without considering an exponential number of combinations. So, the rest of coverage criteria that we will be seeing in this lecture and in part of next lecture, goal is to achieve this particular point.


(Refer Slide Time: 17:48)



Clause determining a predicate

- Sometimes, the truth of certain clauses in a predicate makes the other clauses non-influential on the predicate.
 - For e.g., in the predicate $p = a \vee b$, if a is true then the predicate p is true irrespective of whether b is true or false.
 - If a is false, then b completely *determines* p in the sense that truth/falsity of b influences truth/falsity of p .
- We would like to test *each* clause under circumstances where the clause determines the predicate.

NPTEL

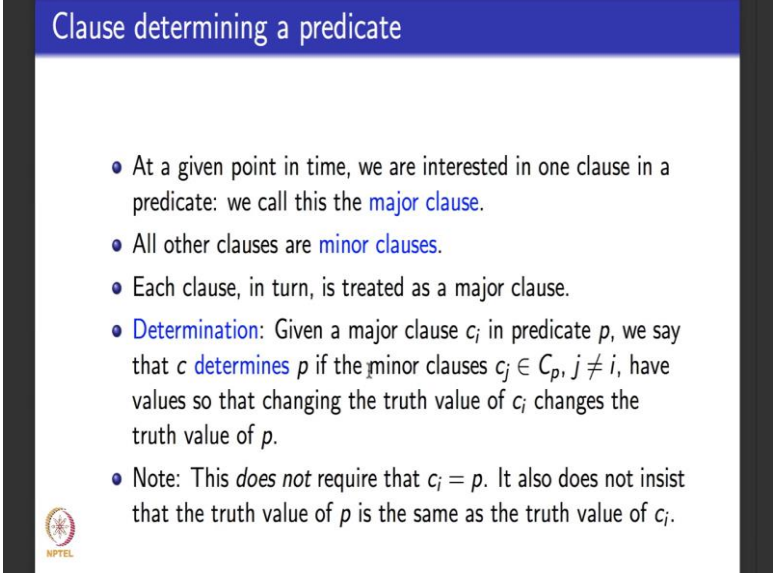


So, for that we need to be able to understand the notion of a clause determining a particular predicate. So, if you take this example p is equal to a or b , let us say a is true right. In this example a happens to be true. If a happens to be true because p is a or b then no matter what the value of b is the entire predicate p will evaluate to be true. Why, because it is an or and there is one for or to be true one of the operands it is enough to be true.

So, we can say that when a is true b does not determine p , which means irrespective of the value that b takes p will become true. But let us say we consider the other case where a happens to be false. If a is false, now whether b is true or not will completely determine whether p is true or not. Let us say a is false, we parked it with that let us say b becomes true then false or true p will become true. Now let us say a being false b becomes false then p will become false. So, if a is false then we say that the clause b completely


determines p in the sense that the truth or falsity of b influences the truth or falsity of the predicate p . So, this is the kind of combination that we would like to test. We would like to test each clause under circumstances where that clause determines the predicate.

(Refer Slide Time: 19:25)



Clause determining a predicate

- At a given point in time, we are interested in one clause in a predicate: we call this the **major clause**.
- All other clauses are **minor clauses**.
- Each clause, in turn, is treated as a major clause.
- **Determination:** Given a major clause c_i in predicate p , we say that c_i **determines** p if the minor clauses $c_j \in C_p, j \neq i$, have values so that changing the truth value of c_i changes the truth value of p .
- **Note:** This *does not* require that $c_i = p$. It also does not insist that the truth value of p is the same as the truth value of c_i .

 NPTEL

So, how do we do it ? We have to define what the notion of the clause determining a predicate case first. So, we say at any point in time given a predicate with several clauses I focus my attention on one particular clause. The clause on which I focus my attention on is called the major clause at that point in time. All the other clauses are called minor clauses.

So, if I take for example, this one, the first what I have made is that I say I want to focus my attention on the clause a . So, when I want to focus my attention on the clause a , I make a the major clause. When I make a the major clause, b becomes the minor clause and when I make a the major clause, I want a to determine p . For a to determine p b should be made false. So, at any given point in time we are interested in one clause, we call it the major clause and the rest of the clauses are called minor clauses and then we test how this major clause that we have fixed determines the predicate p .

After finishing this test we pick up another clause from the predicate p , call that other clause that we have picked as the major clause and the remaining clauses including the one we picked up first become the minor clause. So, each clause in turn will be treated as a major clause. So, now, we define the notion of determination. What is determination?

Given a major clause C_i , I fix one clause to be the major clause C_i . I say that C_i determines p if the minor clause is C_j which means all the other clauses are such that changing the value of C_i changes the value of p . So, if we go back to this example, if a is false, if a is false and a is a minor clause and let us say b is a major clause then b completely determines p whenever a is false. So, that is the definition here.

We say a clause that we fix call it the major clause determines the predicate p if the values of the minor clauses are such that changing the value of the major clause changes the value of p . Please note that it does not mean that the major clause is equal to p , neither does it mean that the truth value that the p takes is the same as the truth value of C_i . Let us say the major clause is true, it does not mean that the p predicate p also becomes true exactly when the major clause is true. It could be the case that when the major clause is true the predicate is false and when the major clause is false the predicate is true. Or, it could be the case that when the major clause is true and the predicate is also true and if the major clause is false and the predicate is also false.

But what is important is that I fix one clause, call it the major clause and say this clause has to determine p , means whenever this clause changes its truth value from true or false or false to true, it will change the truth or falsity of p . All the other clauses called minor clauses take values such that this happens and when this happens I say that the major clause determines p .



(Refer Slide Time: 22:53)

Active clause coverage

- **Active Clause Coverage (ACC):** For each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses $c_j, j \neq i$, so that c_i determines p .
TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false.
- For e.g., if $p = a \vee b$, we get four requirements in TR, two for clause a and two for clause b . b needs to be false for a to determine p and vice versa.

	a	b
$c_i = a$	T	F
	F	F
$c_i = b$	F	T
	F	F

- There are three distinct TRs above:
 $\{(a = T, b = F), (a = F, b = T), (a = F, b = F)\}$.

So, using this we define a few other coverage criteria. The first coverage criteria that we will be seeing is what is called active clause coverage abbreviated as ACC. So, what does it say? It says for each predicate p in the set of all predicates P , fix a clause in p , call it the major clause let it be clause C_i and you choose minor clauses C_j 's, all the other clauses such that C_i determines p . Active clause coverage says then there are 2 requirements in my test requirement for active clause coverage. It says one requirement says that the major clause C_i should evaluate to true and the other requirement says that the major clause C_i should evaluate to false. Just to repeat what I told you in a different style, I have a predicate p . The predicate p has several clauses. I fix one clause that I want to focus my attention on let us say clause C_i . I call that the major clause now rest of the clauses are all minor clauses, I make them take values in such a way that this major clause C_i determines p .

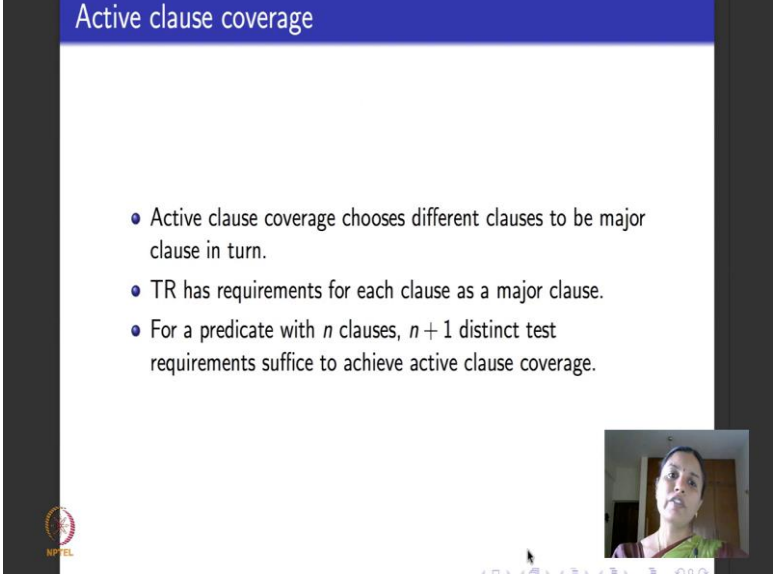
Now, my TR for active coverage criteria says that you make this major clause C_i true once and then you make the major clause C_i false once. Is it clear? So, here is the same example: p is equal to a or b , there are 2 clauses here clause a and clause b . As I told you before we take each clause in turn and make it the major clause. Let us say, to start with I make a the major clause. So, C_i in this table happens to be a , read it as a is the major clause I want a to determine p . So, if a has to determine p then b must become false. So, b is assigned false here in the second column, third column of the table and now my test requirement for active clause coverage says you make a true once, a false once. So, I have given that in bold.

Now, I am done with a being the major clause. It is b 's turn, b 's turn to be the major clause. So, I say C_i is b now for b to determine p a should be false and to achieve active coverage criteria TR, I made b true once and false once right. So, this table just to summarize it, each clause in the predicate takes turns be the major clause. Here there are only 2 clauses a and b . To start with a is the major clause.

So, TR for active clause coverage says make a true once false once. So, I have made a true once false once and I want a to determine p . If I want a to determine p , I have to make b false. Similarly when b is the major clause, a has to be made false and b has to be made true once and false once. If you see these 4 rows, they are not different there is a repeat. Second row and fourth row are repeats of each other. Both in the second row and the fourth row, both a and b are false.

So, I remove it. If I remove it, I only get 3 distinct tests one that makes a true b false which is the first row, the second one makes a false b true which is the third row here and the third one which stands for the combined second and fourth one makes both a and b false.

(Refer Slide Time: 26:37)



The slide is titled "Active clause coverage" in a blue header. It contains three bullet points:

- Active clause coverage chooses different clauses to be major clause in turn.
- TR has requirements for each clause as a major clause.
- For a predicate with n clauses, $n + 1$ distinct test requirements suffice to achieve active clause coverage.

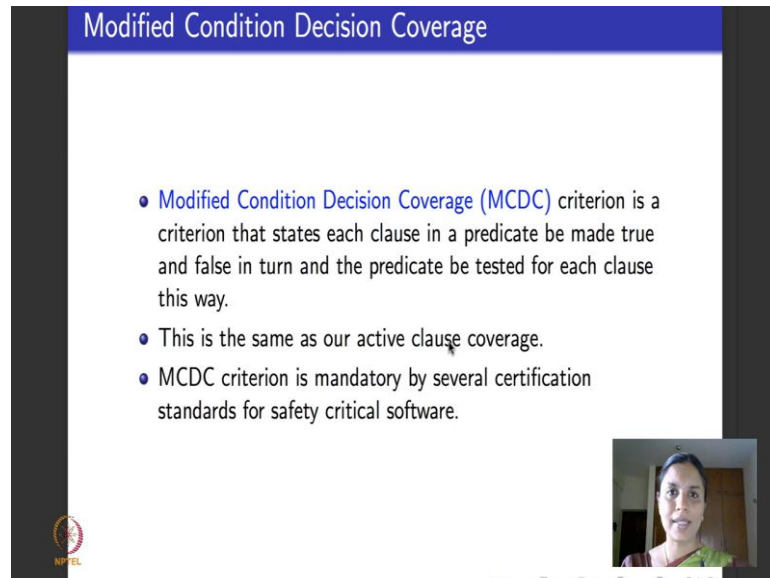
In the bottom right corner, there is a small video inset showing a woman speaking. At the bottom left of the slide, there is a small NPTEL logo.

So, what does active clause coverage do? Active clause coverage chooses each clause in the predicate p to be a major clause and then it writes TR, makes the major clause determine the predicate p and writes TR which says now make the major clause true once make the minor clause true once. So, that is the term active clause coverage.

So, let us say a predicate has several clauses. Each clause takes turn to be major and the clause that is major becomes active in the sense that determines the value of the predicate p . So, I make it true once and I make it false once and because I make it true once and I make it false once, it would completely test the predicate p because that determines the predicate p . The predicate also would become true or false once and true or false once again right. It so happens that for a predicate with n clauses active coverage can be achieved by writing n plus one test requirements. Here it looks like we had $2n$ test requirements right, where n is the number of clauses. So, 2 requirements for a, 2 requirements for b, but as this example illustrates one of it was a repeat requirement. So, we merged them and we happened to get 3 requirements. So, in general for a predicate

with n clauses to get active clause coverage it is enough to have n plus one distinct test requirements.

(Refer Slide Time: 28:06)

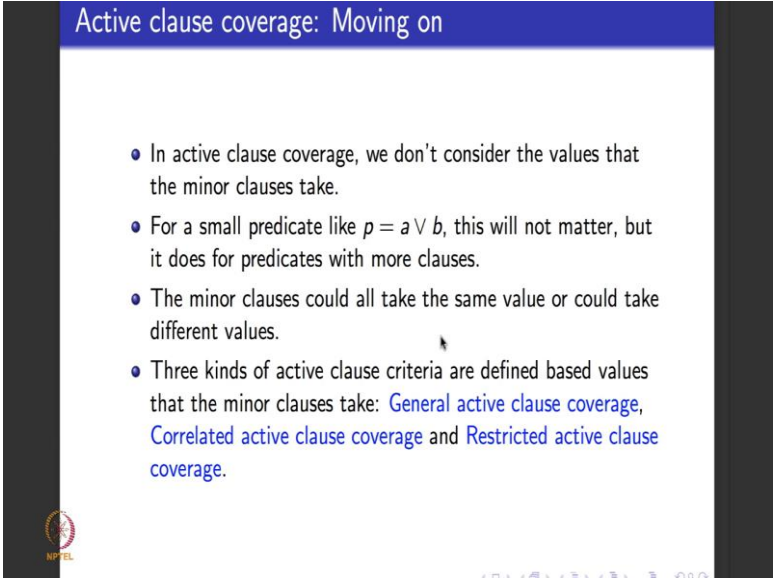


Modified Condition Decision Coverage

- **Modified Condition Decision Coverage (MCDC)** criterion is a criterion that states each clause in a predicate be made true and false in turn and the predicate be tested for each clause this way.
- This is the same as our active clause coverage.
- MCDC criterion is mandatory by several certification standards for safety critical software.

So, just a small point of observation before we move on. In testing you might have heard of this term called MCDC criteria or MCDC testing. MCDC stands for modified condition decision coverage. It is a criteria which says that when I have a predicate with several clauses, you make each clause in the predicate become true once false once and test the predicate. Usually you switch off all the compiler optimizations, make every clause important in the predicate and exercise each clause to be true once false once and test and see how it influences the predicate. And you can go on modifying this condition what is popularly called as MCDC is what we call as ACC and MCDC is a mandatory way of testing for certifying several safety critical software.

(Refer Slide Time: 28:59)



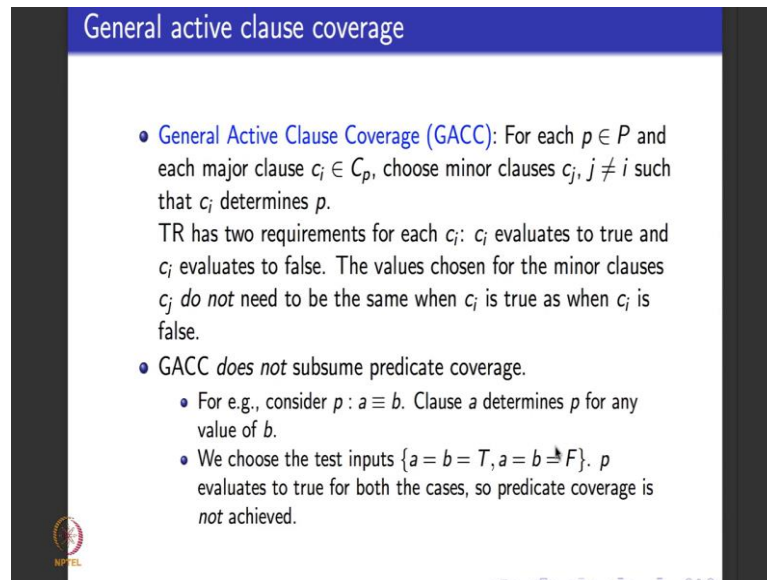
Active clause coverage: Moving on

- In active clause coverage, we don't consider the values that the minor clauses take.
- For a small predicate like $p = a \vee b$, this will not matter, but it does for predicates with more clauses.
- The minor clauses could all take the same value or could take different values.
- Three kinds of active clause criteria are defined based values that the minor clauses take: General active clause coverage, Correlated active clause coverage and Restricted active clause coverage.

NPTEL

So, now we come back to active clause coverage criteria. In active clause coverage criteria we said you check one clause make it the major clause, let the minor clauses take values such that the major clause determines the predicate. We do not really worry about what will be the values that the minor clauses will take right. Will they all take same values, will they all take different values we did not really think about it. If we start thinking about the kind of values that the minor clauses will take, you can further refine active clause coverage criteria into 3 different categories. That is what we are going do now. We will refine it into 3 different categories: one will be called generalized ACC, the other one will be called correlated ACC and the third one will be called restricted ACC.

(Refer Slide Time: 29:51)



General active clause coverage

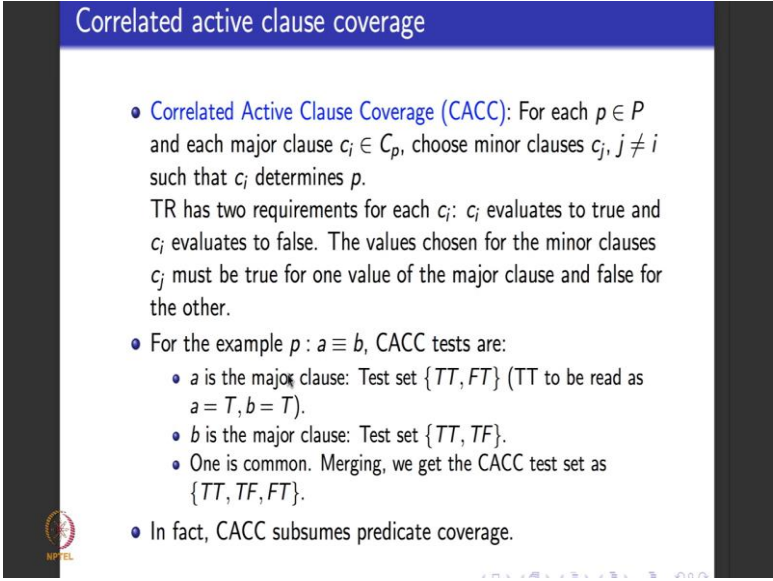
- **General Active Clause Coverage (GACC):** For each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses $c_j, j \neq i$ such that c_i determines p .
TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false. The values chosen for the minor clauses c_j *do not* need to be the same when c_i is true as when c_i is false.
- GACC *does not* subsume predicate coverage.
 - For e.g., consider $p : a \equiv b$. Clause a determines p for any value of b .
 - We choose the test inputs $\{a = b = T, a = b = F\}$. p evaluates to true for both the cases, so predicate coverage is *not* achieved.

So, we will define each one of them one after the other and see what they mean. The first one that we will look at is what is called general active clause criteria GACC. What does it say ? It says the following for each predicate p in P each major clause C_i choose minor clause C_j such that C_i determines p . This is the same active clause coverage I have not changed anything. The next sentence is also the same I have not changed anything TR has 2 requirements for each C_i ; C_i evaluates to true and C_i evaluates to false. Again this is the same as active clause coverage. So far no changes. This is the sentence where we begin to see the change it says the values chosen for minor clauses do not have to be the same as when C_i is true and as when C_i is false. Basically it says do not put any condition on the values that the minor clauses take. Let them be what they are.

So, in some sense generalized active clause coverage is the same as active clause coverage. So, one thing to be noted is that active clause coverage or generalized active clause coverage does not subsume predicate coverage. Why ? Here is an example. Suppose you consider this predicate p which says a is equivalent to b . Now clause a determines p for any value of b right. Suppose b is true then clause a will completely determine p because a could be true or false and change the value of p and suppose b is false then clause a still determines p because a could be true or false and change the value of p right. Suppose we choose the following test inputs. We make a and b both to be true and a and b both to be false right. Then what happens for this predicate p a equivalent to b ? In both the cases the predicate evaluates to true because true is

equivalent to true and false is equivalent to false. So, it again evaluates to true the entire predicate p . But this means that predicate coverage is not achieved right? But I have achieved generalized active clause coverage because I have done a to be the major clause and b also to be the major clause. So, generalized active clause coverage does not subsume predicate coverage.

(Refer Slide Time: 32:23)



Correlated active clause coverage

- **Correlated Active Clause Coverage (CACC):** For each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses $c_j, j \neq i$ such that c_i determines p .
TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false. The values chosen for the minor clauses c_j must be true for one value of the major clause and false for the other.
- For the example $p : a \equiv b$, CACC tests are:
 - a is the major clause: Test set $\{TT, FT\}$ (TT to be read as $a = T, b = T$).
 - b is the major clause: Test set $\{TT, TF\}$.
 - One is common. Merging, we get the CACC test set as $\{TT, TF, FT\}$.
- In fact, CACC subsumes predicate coverage.

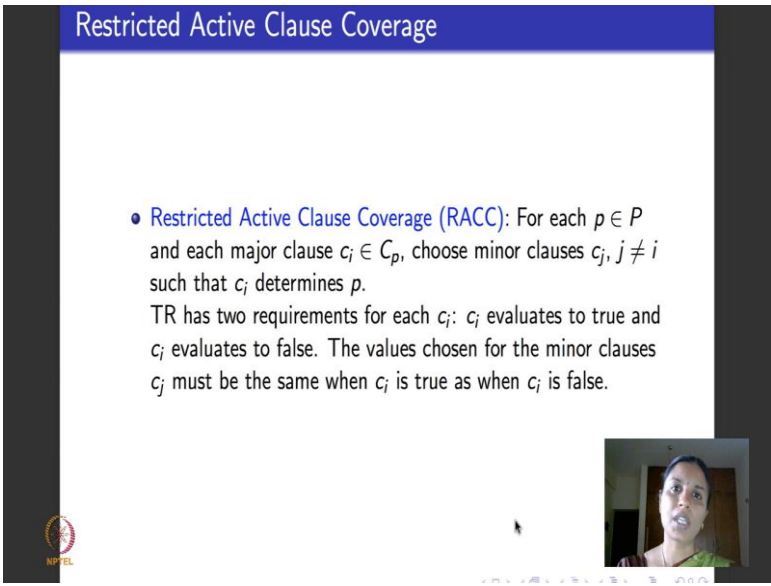
The next condition we will see of ACC criteria is what is called correlated active clause coverage. Now let us focus on the English meaning correlated. What does correlate mean? Correlate means that the minor clauses are such that they correlate with each other. So, the again the first parts of the definition are exactly the same I take a predicate and I fix one clause call it the major clause choose minor clauses such that the major clause determines p no changes so far. Second again no changes the TR has 2 requirements the major clause evaluates to true major clause evaluates to false. The third sentence is again where you begin to see the change. It says the value chosen for minor clauses must be true for one value of major clause and false for one value of the major clause. Go back, what does generalized active coverage criteria say? It says the value chosen for minor clauses need not be the same.

Basically it says do not provide any restriction on the values chosen for minor clauses that is what GACC says. What does CACC say ? It says that the values chosen for the minor clauses must be true for one value of the major clause and false for other value of

the major clause. So we take the same example predicate p which says a is equivalent to b . What are correlated active clause coverage tests for p ? Let's say a is the major clause. Then here is the test set for correlated active clause coverage please read this TT as a is assigning true to a and assigning true to b read FT as assigning false to a and assigning true to b . So, I have made a as my major clause, a is true once false once right. So, I have done this and to make a determine p , I make b true once and true once again. So, this will together determine CACC.

Now, if b is the major clause b becomes true once, false becomes true once and a remains true in both the cases. Here again if you see a is true b is true a is true b is true is common across these things I do not repeat it I put only once. So, I make the 3 tests that put together that satisfy CACC are a true b true, a true b false, a false and b true. In fact, we will show later in the next lecture that CACC does subsume predicate coverage, remember GACC does not subsume predicate coverage and CACC does subsume and I will tell you why in the next lecture.

(Refer Slide Time: 35:15)



Restricted Active Clause Coverage

- **Restricted Active Clause Coverage (RACC):** For each $p \in P$ and each major clause $c_i \in C_p$, choose minor clauses $c_j, j \neq i$ such that c_i determines p . TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false. The values chosen for the minor clauses c_j must be the same when c_i is true as when c_i is false.

The slide features a blue header with the title 'Restricted Active Clause Coverage'. Below the header, a bullet point defines RACC. At the bottom right, there is a small video inset showing a woman speaking. The NPTEL logo is visible in the bottom left corner.

The last active clause coverage criteria that we will be seeing is what is called restricted active coverage clause criteria. Here again the first few parts of the definition are the same: I take a predicate I take a major clause choose minor clauses such that the major clause determines the predicate. We are here now, TR has 2 requirements for the major clause, major clause evaluates to true major clause evaluates to false. The third line is

where the difference begins to show up. It says here for RACC the value is chosen for minor clauses must be the same as when C_i is true and when C_i is false.



So, we will go back. We saw active clause coverage criteria. We are refining active clause coverage criteria into 3 kinds: general active clause coverage criteria, correlated active clause coverage criteria and restricted active clause coverage criteria. What does general say? General says that the major values chosen for the minor values do not have to be the same there is no condition that is why it is called generalized. What does correlated say? Correlated says that the values chosen for minor clauses must be true for one and false for the other that is why the term correlated. What does restricted say? Restricted says that the values for the minor clauses must be the same as when the major clause is true as and when the major clause is false. So, it is restricted because it restricts the truth or falsity of the minor clauses to be the same as that of the major clause.

(Refer Slide Time: 37:02)

ACC Criteria: Example

- We illustrate the various ACC criteria using another example.
- Consider the predicate $p = a \wedge (b \vee c)$.
- Truth table for p .

	a	b	c	$a \wedge (b \vee c)$
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F



So, since this is a little complicate, what I thought I will do is I will take another example and explain active clause coverage criteria with reference to this example to help you understand it well. So, here is the second example. Now you consider this predicate, read it as p is equal to a and b or c . What I have done here is I have given you the entire truth table for the predicate. So, there are 3 clauses in this predicate: a , b and c . Each of these 3 clauses can take 2 false values in combinations. So, if I put 2 values per clause then I totally get eight combinations of true false values as listed here and the final column in

this truth table says what will happen for each of the combinations of true false values of a, b and c. For example, if all 3 are true which is the first row: a, b and c are true then what will happen to a and b or c, it will be true. Why, because b or c will be true and true and true will be true.

Now, you take some third row here it says a is true b is false c is true, but the result the whole predicate becomes true. Why is that, because even though b is false; false or true b or c will evaluate to be true that ended with a being true will give me true. Let us take another example let us take row number six. It assigns a to be false, b to be true, c to be false and because a is false and a is ended with something the whole predicate becomes false. So, this table gives the entire truth table for this predicate. Now what we are going to do is we are going to understand how the various active clause coverage criteria work for this predicate. We begin, as I told you generalized active clause coverage criteria is the same as ACC nothing interesting.



(Refer Slide Time: 38:50)

CACC on $p = a \wedge (b \vee c)$

- In $p = a \wedge (b \vee c)$, for a to determine p, $b \vee c$ must be true.
- $b \vee c$ can be made true in three ways.
- CACC is satisfied for a being the major clause by choosing one TR from rows 1,2,3 and one from rows 5,6,7.

	a	b	c	$a \wedge (b \vee c)$
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F

- Totally, nine combinations exist.

So, we will see correlated active class coverage criteria for these predicate and restricted active class coverage criteria for the predicate. So, let us start with CACC, correlated active clause coverage criteria. So, now I want to make a the major clause which means what I want a to determine the predicate p. For a to determine the predicate p this must be true right because if b or c is true then if a is true p will become true and if a is false p will become false. Now you look at b or c, b or c can be made true in 3 ways right. You

could make both b and c true, you could make one of b or c true. So, in 3 different possible combinations of true false values for b and c, the predicate b or c will be true.

So, what I have done is I have pulled out rows from this truth table. Only those rows which will help you to get the coverage criteria of CACC satisfied. So, CACC for a being the major clause can be satisfied by making a true once, which means choosing one of rows to one 2 and 3. Please see here, all the 3 rows a is true right and in all the 3 rows there are 3 possible ways of making b or c true. So, a and b or c becomes true. In rows 5, 6 and 7, a is false a is the major clause I made it true in rows 1 2 and 3 now I am making it false in rows 5, 6 and 7. Now again b or c is true 3 different possibilities in all the 3 cases the predicate becomes false. So, to get my test cases for CACC, I pick up one from the top set one 2 3 and one set of values from the bottom set 5, 6, 7. Any of these will make a true once false once and b or c combinations are such that a will determine p and they also satisfy correlated active clause condition right. It says the values chosen for minor clauses must be true for one value for the major clause and false for the other that is also satisfied.



So, to satisfy CACC for the predicate a and b or c, pick up one row from here one row from here. How many different ways of picking up one row from here one row from here exist? 3 into 3 nine ways. So, nine combinations are of nine different test cases can be written to satisfy CACC for this predicate.

(Refer Slide Time: 41:41)

RACC on $p = a \wedge (b \vee c)$

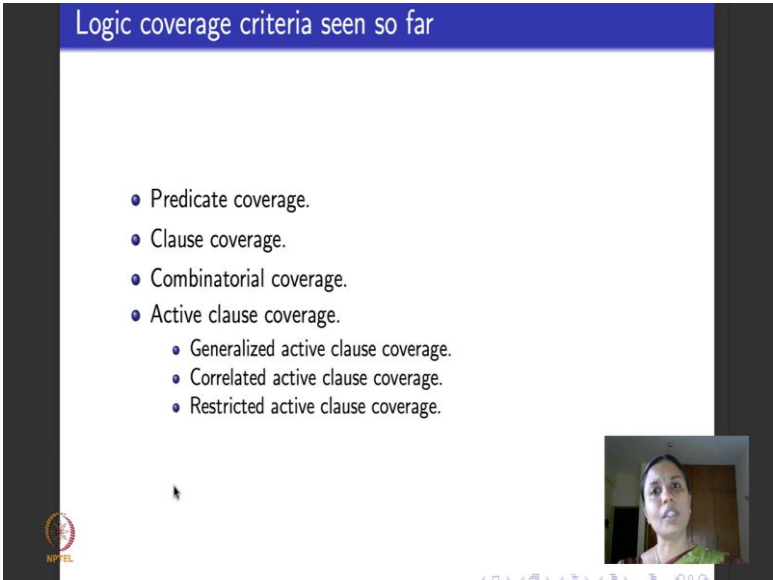
- Clause a is the major clause: Only three of the nine sets of test requirements that satisfy will satisfy RACC.
- Row 1 is paired with row 5, row 2 is paired with row 6 and row 3 is paired with row 7.

	a	b	c	$a \wedge (b \vee c)$
1	T	T	T	T
5	F	T	T	F
2	T	T	F	T
6	F	T	F	F
3	T	F	T	T
7	F	F	T	F

Now for the same predicate if you take a again as the major clause and instead of doing CACC you try to do RACC restricted active clause coverage. In which case what you can do is a is the major clause. So, I make a true once false once, true once false once, true once false once and the values of minor clauses should be such that they take the same value right you remember that they take the same value that is what restricted active class coverage says. So, here if you see b is true and c is also true b is true, c is also true. In which case a determines the whole predicate p the whole predicate becomes true once false once. Similarly here b and c take the same value such that a determines p. Again b and c take the same value such that a determines p. So, here is how I satisfy RACC. So, I any of these 3 combinations 1 with 5, 2 with 6 and 3 with 7 will satisfy restrictive active clause coverage criteria on this predicate.

(Refer Slide Time: 42:50)



Logic coverage criteria seen so far

- Predicate coverage.
- Clause coverage.
- Combinatorial coverage.
- Active clause coverage.
 - Generalized active clause coverage.
 - Correlated active clause coverage.
 - Restricted active clause coverage.

So, what are the logic coverage criteria that we have seen? We started with predicate coverage, most elementary coverage criteria make the whole predicate true once false once. Then we moved on to clause coverage which says for each clause in the predicate make it true once false once. We realize that they do not subsume each other, they can be completely unrelated we saw examples for that. Then we say why bother, you take every combination of true false values of the clauses in the predicate. That will give rise to combinatorial coverage. Then we realized that there were too many combinations. There were in fact, exponential number of combinations it is not worth it.

So, I say I want to look at active class coverage where at any point in time I have focused on one clause in the predicate call it major clause all the other clauses are minor clause and see how these major clause determines p. And there are 3 different ways in which the minor clauses could take values, those give rise to 3 different active class coverage criteria. In the next lecture we will move on and define some more coverage criteria and I will tell you a little bit about how to actually get test cases to satisfy these coverage criteria.

Thank you.