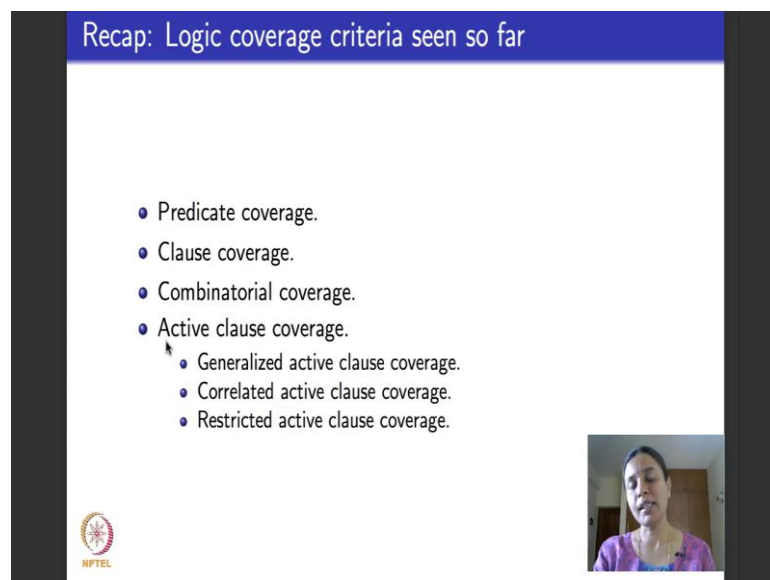


**Software Testing**  
**Prof. Meenakshi D'Souza**  
**Department of Computer Science and Engineering**  
**International Institute of Information Technology, Bangalore**

**Lecture - 23**  
**Logics: Coverage Criteria, contd**

Hello everyone. We are in the middle of week 5. What I will be doing today is the 4th lecture of week 5. We did assignments solving for the last week then we began with logic coverage criteria, I introduced to you the basics of logic in the second lecture of this week. Then last lecture we started seeing coverage criteria. Today we will continue to see coverage criteria.

(Refer Slide Time: 00:42)



The slide is titled "Recap: Logic coverage criteria seen so far" in a blue header. It contains a bulleted list of coverage criteria. The first four are: Predicate coverage, Clause coverage, Combinatorial coverage, and Active clause coverage. The last one, Active clause coverage, has three sub-bullets: Generalized active clause coverage, Correlated active clause coverage, and Restricted active clause coverage. In the bottom left corner is the NPTEL logo, and in the bottom right corner is a small video feed of the professor.

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

We will finish doing logic coverage criteria. In the next class, I will tell you about algorithms for logic coverage criteria. So, what were the coverage criteria that we have seen till now? We saw what were predicates and clauses. Predicate is like any expression that evaluates to true or false that labels decision statements that you find in programs and guards that you could find in design or requirements.

So, the predicate has one or more clauses which are atomic entities in predicates. Clauses do not have logical connectors. So, we saw predicate coverage which tested the predicate to be true or false, clause coverage which tested each clause in the predicate to be true or false in turn. Then we saw what was called all combinations coverage, sometimes it is

also called complete clause coverage where they test the entire predicate for every possible combination of true false values in clauses. This will result in exponential number of test cases, usually not desirable.



So, what we try to want to test is how each clause affects a predicate in turn and, which are the important clauses, which will affect the predicate in the sense that the truth or falsity of the clause clearly determines the truth or falsity of the predicate. So that gave rise to what we call as active clause coverage, and we realize that active clause coverage did not really tell you what the other clauses would do. We fix the clause called the major clause see how the major clause determines the predicate, but we did not really specify how the other clauses put together called as minor clauses affect the truth or falsity of the predicate. What we did was we went ahead and refined active clause coverage into 3 specific coverage criteria called generalized active class coverage, correlated active clause coverage and restricted active clause coverage.

So, in generalized, we do not have any conditions that tell you what are the values that the minor clauses take. In correlated, we say that the minor clauses take values such that they aid the major clause to completely determine the predicate and in restricted, we say the minor clauses all have to take the same value as the predicate is true and as and when the predicate is false.

(Refer Slide Time: 02:52)

### Inactive clause coverage

- Complementary criterion to active clause criteria, ensures that the major clause *does not* affect the predicate.
- **Inactive Clause Coverage (ICC)**: For each  $p \in P$  and each major clause  $c_p \in C_p$ , choose minor clauses  $c_j, j \neq i$  so that  $c_i$  *does not* determine  $p$ .  
TR has four requirements for  $c_i$ :
  - 1  $c_i$  evaluates to true with  $p$  true.
  - 2  $c_i$  evaluates to false with  $p$  true.
  - 3  $c_i$  evaluates to true with  $p$  false.
  - 4  $c_i$  evaluates to false with  $p$  false.



Now, moving on we will today look at what is called inactive clause coverage criteria. So, here again we want to know, we want to know per clause in a predicate how that clause influences the predicate in active clause coverage criteria. In inactive clause coverage criteria, we fix a clause again call it the major clause, but our interest is knowing how the major clause does not affect the predicate. This might be a little counter intuitive. Very soon I will tell you an example that motivates for the need for a clause criteria like inactive clause criteria. So, what is the basis with which we start inactive clause coverage criteria? So, there is a predicate, there are several clauses in the predicate, each clause in the predicate takes turn to become what is called the major clause.

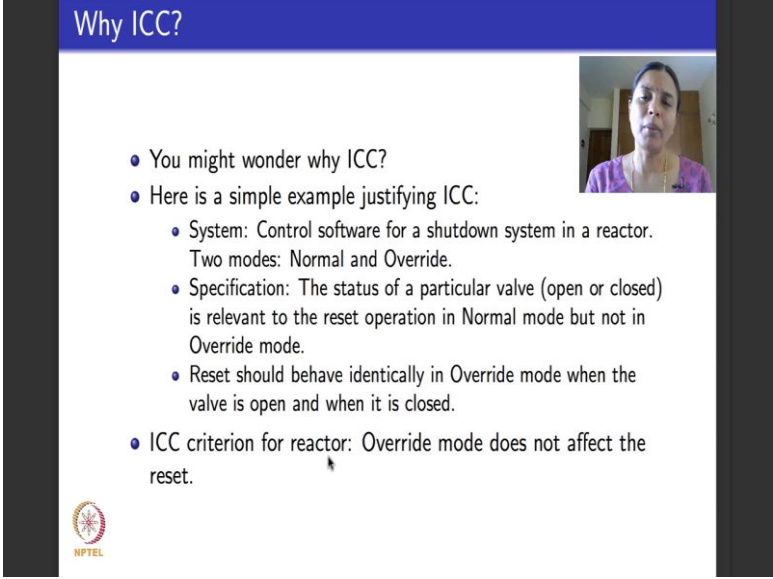
So, let us fix one clause in the predicate, call it a major clause. I want the other clauses minor clauses to take values in such a way that the major clause does not affect the predicate. So, that is what is called inactive clause coverage criteria. So, what is inactive clause coverage abbreviated as ICC? It says for every predicate  $p$  in the set of predicates  $P$ , each clause takes turns to be a major clause let us say  $c_i$  is a major clause we chose minor clause as  $c_j$  such that  $c_i$  does not determine  $p$ . The test requirement has 4 requirements for  $c_i$ : the predicate  $p$  can evaluate to be true in the first 2 cases in which case the  $c_i$  major clause evaluates to true once and to false once. In the second case the predicate  $p$  evaluates to false and the major clause evaluates to true and evaluates to false.

So, put together if you read all these 4 TRs, it says the predicate  $p$  can evaluate to true once and predicate  $p$  evaluates to false once and for each of these combinations of the predicate  $p$  evaluating to true and false, the major clause  $c_i$  evaluates to true once and false once each time. So, the major clause has no influence on  $p$ . In other words the major clause does not determine  $p$ . So, we say the major clause is inactive as far as the truth or falsity of  $p$  is concerned and this is what is called inactive clause coverage. Another thing that I would like you all to observe before we move on is look at these 4 test requirements: it says  $p$  evaluates to true in the first two  $p$  evaluates to false in the first two.

So, in both the cases between both the cases predicate coverage is true. So, inactive clause coverage criteria subsume predicate coverage. So, towards the end of this module we will see logic coverage criteria subsumption. There we will tell you what happens


right. So, just to repeat and make sure you understand what inactive clause coverage is. So, I fix one clause, call it the major clause and my goal is to make sure that the major clause has no influence on the truth or falsity of  $p$ . In other words the major clause does not determine the predicate  $p$ . So, my TR under inactive clause coverage has four requirements:  $p$  can become true in the first two,  $p$  can become false in the last two. Whenever  $p$  becomes true the major clause becomes true once and false once for  $p$  becoming false the major clause becomes true once and false once.

(Refer Slide Time: 06:30)



**Why ICC?**

- You might wonder why ICC?
- Here is a simple example justifying ICC:
  - System: Control software for a shutdown system in a reactor. Two modes: Normal and Override.
  - Specification: The status of a particular valve (open or closed) is relevant to the reset operation in Normal mode but not in Override mode.
  - Reset should behave identically in Override mode when the valve is open and when it is closed.
- ICC criterion for reactor: Override mode does not affect the reset.



So, the major clause that I fix has no influence on the value of  $p$ . So, as I told you might wonder why I want to have inactive clause coverage criteria. Why do I want a major clause such that it does not determine  $p$ ? It so happens that it is very useful when I consider practical applications of logic coverage criteria. So, here is a simple example that motivates and justifies the need for inactive coverage clause criteria. So, let us take an example system. The example system that we consider is a shut down system of a reactor. So, there is a particular reactor, it could be any kind of reactor let us say nuclear reactor or any other reactor and sometimes I may want to shut down the working of the reactor. How is the shutdown done? Shutdown is done with the help of a software that controls the working and shutting down of a reactor.

So, that software we call it control software right. The control software typically operates in 2 modes or in 2 states. So, it operates in normal mode or it operates in what is called in

override mode. You can interpret normal mode to be normal operations of the control software as it is working with control reactions on the reactor. Override mode is control software is trying to take over may be there is an emergency or something like that, it is trying to take over and do certain actions on the reactor system. Now what are specifications of such a control software. There could be several specifications, but here is a small example that will motivate why we need inactive clause coverage criteria.

So, specification says that now, remember we are going to shut down we are looking at control software that shuts down the nuclear reactor. Let us say shutdown happens through a particular valve. The valve could be open when you want to you want the nuclear reactor to work normally and the valve needs to be closed when you want to shut down the nuclear reactor.

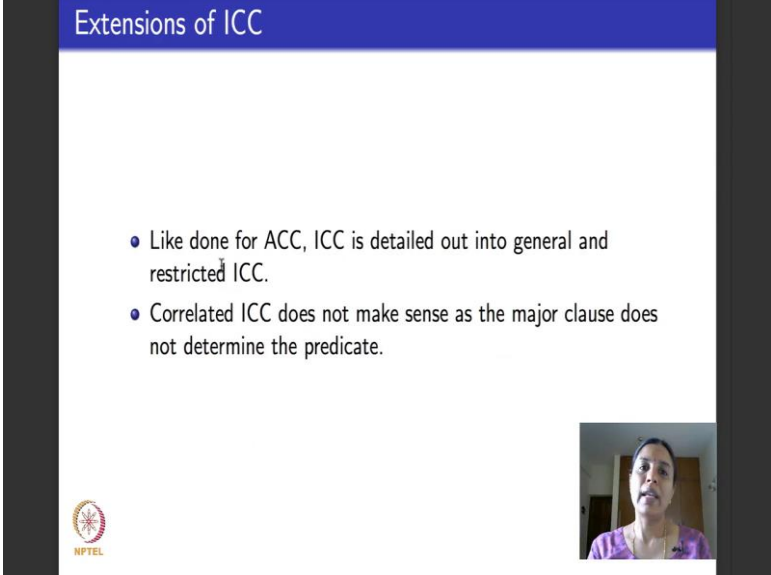
So, the control software actually sends activates the command to open or close this valve. So, specification could be the following it says that a particular valve which could be open or closed is relevant, the status of a particular valve, is relevant to the reset operation in normal mode, but not in override mode. So, what it says is that this reactor could be in normal mode or override mode. So, the shutdown operation, which means opening or closing the valve the status of the valve whether it was already open or already closed, what do I want to do with it, that is relevant to the reset operation in normal mode, but when the control software is working with the reactor in override mode irrespective of what the status of the valve is it will go ahead and shut down the reactor.

So, what it says is that the reset operation, if I consider it as an action, that should behave identically in override mode when the valve is open and when the valve is closed. In other words, stating it in terms of logical coverage criteria, if you think of this as a predicate, it says the clause which says reset is true or false should not influence the predicate, the mode of the predicate. So, it should be the same as an override mode when the valve is open and when the valve is closed. So, it says the particular operation should not influence the state of a valve when the nuclear reactor is in override mode.

So, the particular operation should be inactive if it is modeled as a clause when the reactor is in override mode, inactive to the status of the valve. So, this is a natural case

where you say that if this is modeled as a clause, if it is a major clause it should never influence what happens to the status of the valve.

(Refer Slide Time: 10:16)



The slide is titled "Extensions of ICC" in a blue header. It contains two bullet points: "• Like done for ACC, ICC is detailed out into general and restricted ICC." and "• Correlated ICC does not make sense as the major clause does not determine the predicate." In the bottom right corner, there is a small video inset showing a woman speaking. In the bottom left corner, there is a small circular logo with the text "NPTEL" below it.

- Like done for ACC, ICC is detailed out into general and restricted ICC.
- Correlated ICC does not make sense as the major clause does not determine the predicate.


So, here is the good reason to consider things like inactive clause coverage criteria. Like we did for active clause coverage, if you go back to the first slide where we recapped what we did in the last class we looked at active clause coverage we looked at 3 specific ways of working with active clause coverage. One was general active clause coverage, correlated active clause coverage and restricted active clause coverage. Today we went ahead and defined inactive clause coverage. So, like we did for general active clause coverage you could define extends inactive clause coverage into various specific things.

So, what are the 3 extensions possible? Like we did for active clause coverage you could have generalized correlated and inactive. But if you pause and think for a minute correlated inactive clause coverage criteria does not make sense. Why does it not make sense, because in inactive coverage clause criteria, we say that the major clause does not determine the truth or falsity of the predicate. So, when it does not determine the truth or the falsity of the predicate the minor clauses cannot correlate with each other and assume values that aid in the major clause determining the predicate. So, we do not define correlated inactive clause coverage criteria. It does not make sense. We define only restricted ICC and generalized ICC.

(Refer Slide Time: 11:34)

### General Inactive Clause Coverage

- **General Inactive Clause Coverage (GICC):** For each  $p \in P$  and each major clause  $c_p \in C_p$ , choose minor clauses  $c_j, j \neq i$  so that  $c_i$  *does not* determine  $p$ .  
TR has four requirements for  $c_j$ :
  - 1  $c_j$  evaluates to true with  $p$  true.
  - 2  $c_j$  evaluates to false with  $p$  true.
  - 3  $c_j$  evaluates to true with  $p$  false.
  - 4  $c_j$  evaluates to false with  $p$  false.The values chosen for the minor clauses  $c_j$  may vary amongst the four cases.





So, here is how generalized ICC is defined. Like we did for active clause coverage criteria you retain the main part of the definition. You retain the main part of the definition of inactive clause coverage criteria then you add a few extra condition which define what the specific extension is. So, in the first part of generalized inactive clause coverage criteria we repeat the definition of ICC that we gave in this slide right. So, I pick up a clause, call that the major clause and choose minor clauses such that major clause does not determine  $p$ . TR has four requirements,  $p$  evaluates to true and false with major clauses evaluating to true and false in each of the cases.

So, that part is fully repeated, there are no changes. Now what are the extra conditions that we add ? We say that the values chosen for the minor clause is  $c_j$  can vary in the 4 cases. So, it is as general as possible, exactly like we defined for active clause coverage. So, the minor clauses values have no conditions, no restrictions associated with it they can vary amongst the 4 cases.

(Refer Slide Time: 12:48)

### Restricted Inactive Clause Coverage

- **Restricted Inactive Clause Coverage (RICC):** For each  $p \in P$  and each major clause  $c_p \in C_p$ , choose minor clauses  $c_j, j \neq i$  so that  $c_i$  does not determine  $p$ .  
TR has four requirements for  $c_j$ :
  - 1  $c_j$  evaluates to true with  $p$  true.
  - 2  $c_j$  evaluates to false with  $p$  true.
  - 3  $c_j$  evaluates to true with  $p$  false.
  - 4  $c_j$  evaluates to false with  $p$  false.The values chosen for the minor clauses  $c_j$  must be the same in cases (1) and (2), and the values chosen for the minor clauses  $c_j$  must be the same in cases (3) and (4).

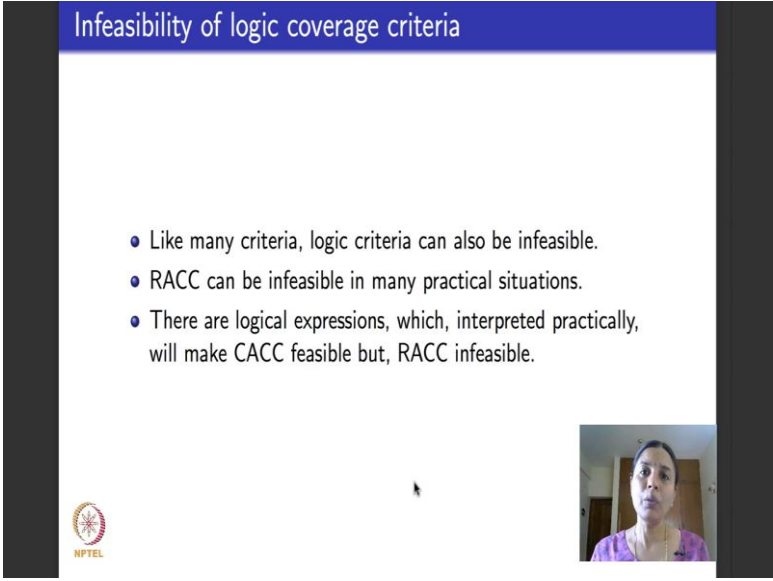


Now what is restricted inactive clause coverage criteria ? Again you repeat the entire definition of inactive clause coverage first. So, this part is the same. So, you choose a major clause, choose minor clauses such that the major clause does not determine  $p$ . The 4 requirements in TR therefore,  $p$  evaluating to true the major clause evaluates to true once false once, for  $p$  evaluating to false the major clause evaluates to true once and false once. Now come the extra conditions here below. What are the extra conditions, what do they say? It says the following it says that the value chosen for the minor clauses  $c_j$  must be the same for cases one and two. What are cases 1 and 2 ? They correspond to, if you go up and look, they correspond to predicate being true. So, the minor clauses should take the same values when the predicate is true and when the predicate is false, which corresponds to cases 3 and 4, the minor clauses should again take the same value.

So, that is why it is a restricted. It says there is 2 TR s for the predicate becoming true: major clause does not determine the predicate the minor clauses all assume the same value. Now, there are 2 more TRs for the predicate becoming false major clause again does not determine the predicate, minor clauses again take the same set of values. So, this is restricted inactive clause coverage. As I told you we do not define correlated inactive clause coverage because it does not make sense as the major clause does not determine the predicate.



(Refer Slide Time: 14:18)



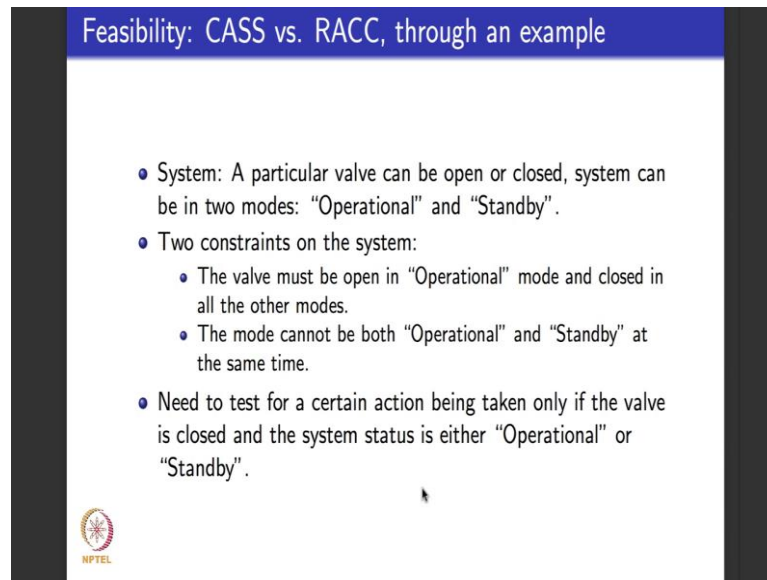
The slide has a blue header with the title "Infeasibility of logic coverage criteria". Below the header, there are three bullet points. In the bottom left corner, there is a small circular logo with the text "NPTEL" below it. In the bottom right corner, there is a small video inset showing a person speaking.

- Like many criteria, logic criteria can also be infeasible.
- RACC can be infeasible in many practical situations.
- There are logical expressions, which, interpreted practically, will make CACC feasible but, RACC infeasible.

So, we move on. Like we did for graph coverage criteria, it is the case that logic coverage criteria also suffer from several of these TRs corresponding to the various coverage criteria being infeasible. Infeasible means there is practically impossible to write a set of test cases that will actually satisfy the test requirement. So, what do we do? If that is the case then what we do is we simply ignore the coverage criteria or we look at other coverage criteria. I will tell you through an example how infeasibility comes when it comes to logical coverage criteria and how to deal with infeasibility.


It so happens that in many cases restricted active clause coverage criteria or restricted inactive clause coverage criteria turn out to be infeasible. In which case we resort to correlated or general active or inactive clause coverage criteria.

(Refer Slide Time: 15:14)



Feasibility: CASS vs. RACC, through an example

- System: A particular valve can be open or closed, system can be in two modes: "Operational" and "Standby".
- Two constraints on the system:
  - The valve must be open in "Operational" mode and closed in all the other modes.
  - The mode cannot be both "Operational" and "Standby" at the same time.
- Need to test for a certain action being taken only if the valve is closed and the system status is either "Operational" or "Standby".



So, here is another example very similar to the nuclear example that we saw, nuclear reactor example that we saw few slides ago. So, here again there is a system, some system may be a reactor we do not know what it is, it is not needed for , it is controlled by software and then there is a particular valve that this software tries to control through an actuator. That valve as always can be open or can be closed and the system can be in 2 modes. The system can be operational or working and the system can be in standby mode. There are 2 constraints on the requirements of the system, 2 design constraints on the requirements of the system that are listed here. It says that the valve must be open when the system is in operational mode and the valve must be closed in all other modes.

In particular the valve must be closed when system is in standby mode the second constraint says that system cannot be both operational and standby mode that makes sense right you might wonder it makes obvious sense why would I want to write it I would want to write it because I have to be able to be complete while writing my constraints.

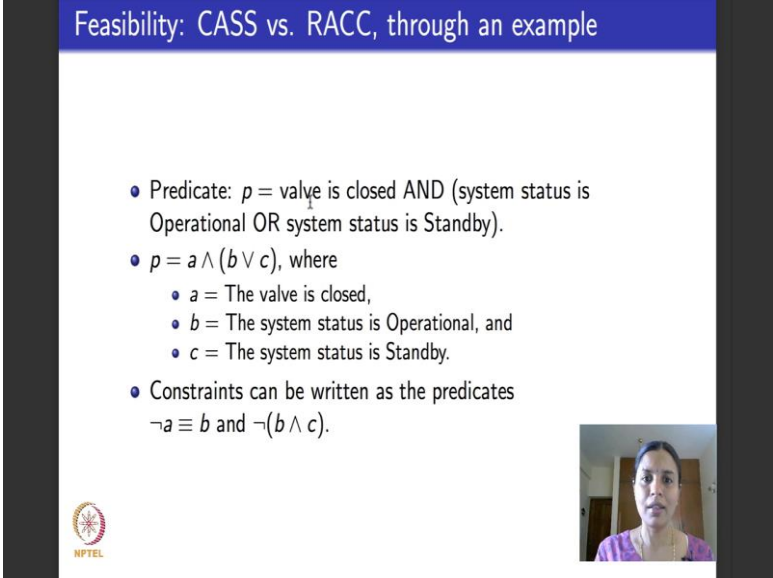
So, just to repeat what is the system? It is some system being controlled by software the software controls the system by opening and closing a valve. There are 2 modes or states in which the system can be in: normal mode, operational mode also called as operational mode and standby mode. The valve can be open or closed. There are 2 constraints on the requirements and design of the system. The first constraint says that the valve must be

open when the system is operational and closed when the system is in standby and it says the system can never be both in the operational state and in the standby state at the same time.

Now, what we want to do is let us say this is our requirement that is given. Somebody comes and tells you test for a certain action being taken only if the valve is closed and the system is either operational or standby. There is some requirement which tells you test this system for the following requirement: the valve is closed and the system status is either operational or standby. So, I take. So, there are 3 constraints here the 2 constraints come as these 2 that are listed here and this last part which gives me a test requirement written in English becomes the main predicate that I have to specify in logic and may be try to use logic coverage criteria to be able to test it.



So, let us read out the predicate once again it says test for what, test for a certain action being taken only if this condition is met. What is the condition? The valve is closed and the system status is either operational or standby.

(Refer Slide Time: 18:05)



Feasibility: CASS vs. RACC, through an example

- Predicate:  $p = \text{valve is closed AND (system status is Operational OR system status is Standby)}$ .
- $p = a \wedge (b \vee c)$ , where
  - $a = \text{The valve is closed,}$
  - $b = \text{The system status is Operational, and}$
  - $c = \text{The system status is Standby.}$
- Constraints can be written as the predicates  $\neg a \equiv b$  and  $\neg(b \wedge c)$ .

So, I write it again the valve is closed and the system status is operational or system status is standby. That is the predicate  $p$  that I want to focus that I want to be able to test on. So, my  $p$  is valve is closed written as 'a', system status is operational written as 'b', system status is standby written as 'c'. So, what is the predicate if I translate; if I substitute back for a, b and c in the formula then what is the predicate that I get I will get

the following. I will say  $a \wedge b \vee c$ , means the valve was closed and the system is either operational or the system is in standby mode.

If you remember this predicate  $a \wedge b \vee c$  is precisely the predicate that we looked at in the last lecture when we worked with examples of combinational coverage criteria, generalized active coverage criteria and restricted active coverage criteria. I had given you the truth table for this predicate and told you which rows to choose from and do what in terms of meeting the TRs for the various RACC and CACC coverage criteria.

Now, this is the predicate go back to this example it also had 2 constraints. What were the constraints? The first constraint said that the valve must be open in operational mode and closed in all other modes. So, the valve is closed is modeled as 'a' for me the system status is operational is modeled as 'b'. So, how will I write the first constraint? I will write the first constraint using this formula it reads as  $\neg a \supset b$ , not a is equivalent to b. What does 'not a' mean?  $\neg a$  means the negation of 'a', 'a' says the valve is closed. So, the valve is not closed which means the valve is open. So, it says the valve is open if and only if the system is operational. That is what is the first constraint right? Second constraint says the mode cannot be operational and standby at the same time.

So, there is a clause that we have already designated calling b for the system status being operational and there is another clause, call it c, for the system status being standby. So, I write it like this I say it is not the case that b and c holds together. It is not the case that the system is operational and the system status is standby together  $\neg(b \wedge c)$ . So, what have I done? My goal is what my goal is to illustrate the difference between CACC and RACC through an example.

So, what have I done? I have taken this example written 2 constraints on this example and the predicate. So, in this slide we have expressed the predicate as a logical formula and we have expressed the constraints as these extra conditions that the predicate, any test requirement on the predicate needs to satisfy before achieving a coverage criteria on the predicate.



(Refer Slide Time: 21:03)

Feasibility: CASS vs. RACC, through an example

Constraints limit the feasible values in the truth table.

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

Only feasible rows are 3, 4 and 6.



So, let us go back and look at this predicate. If you remember this is the truth table for the predicate that we had written in the last lecture. I have re-copied the entire truth table here in the left hand side and I have added one more column here. What does this extra column say? This extra column populates details about if this was the assignment of truth values of the clauses a, b and c and let us say I am considering this as a potential test case value then can I consider it or can I not? I can consider it provided it does not violate any of the given constraints. So, for every combination of truth values for a, b and c, in the last column, I have taken the truth table and added the last column and I have documented about whether the particular combination of a values for a, b and c violates any of the constraints. So, for the first row, a comment is written here saying that the value of a, b and c being true violates both the constraints one and two. Why is that so?

Let us go back to the previous slide look at the constraints. So, first constraint says  $\neg(a \oplus b)$ . So,  $\neg(a \oplus b)$  cannot be true if both 'a' and 'b' are true right, because  $\neg a$  may be false and 'b' will be true. So, that is violated. Second constraint says  $\neg(b \wedge c)$ . So,  $\neg(b \wedge c)$  will be not of true and true, which is not of true which is false right. So, this assignment of truth values for a, b and c all of them being true violates both the constraints. The second assignment where a is true, b is true and c is false violates constraint one.

So, again it violates  $\neg(a \oplus b)$  because both a and b are true. So, negation of a cannot be equivalent to b. So, I have does not violate constraint 2 that is fine, but it violates

constraint one. Third assignment of truth values for a, b and c seems to be fine. So, nothing is written about it violating any of the constraint, fourth assignment is also fine, fifth violates constraint 2 because b and c are true together sixth is fine no violation seventh and eighth both violate constraint one. So, if I consider a test requirement for this entire predicate p, the test requirement basically should not violate the additional constraints imposed on the system. So, totally eight possible test requirements are there because that is the exhaustive combinations of true false values of a and b.

Now, it so happens that 5 of them violate some condition or the other one of them violates both the constraints. So, out of the various test requirements, test values I have for Trs, only 3 combinations of true, false values for a and b can be used as test cases because only those 3 of them do not violate any of the constraints. So, those 3 are given by rows 3, 4 and 6.

(Refer Slide Time: 24:07)

Recap: 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

• CACC is feasible for the predicate p.

Now let us go back and recap from the previous lecture, what was CACC on this predicate, CACC TR and test cases. The predicate is the same. This is what we saw in the last lecture, we had taken the entire truth table and worked out what this restricted table represents as test cases for CACC. If you remember we said any of these combination of test cases, one for the predicate evaluating to true and one for the predicate evaluating to false will be to satisfy CACC requirements. So, you could pick

up either one anyone from rows 1, 2 and 3 or any one from rows 5, 6 and 6; 5, 6 and 7 sorry and that will be a test case combination that will satisfy CACC.

So, now I take this predicate. My goal is to be able to do CACC for this predicate in the presence of these 2 additional constraints. So, these 2 additional constraints mean the only feasible test cases are those from rows 3, 4 and 6 and here, I can pick row 3 and row 6, both of them do not violate the constraint and they end up satisfying the predicate.



(Refer Slide Time: 25:26)

Recap: 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

- RACC is infeasible for  $a$  in the predicate  $p$ .

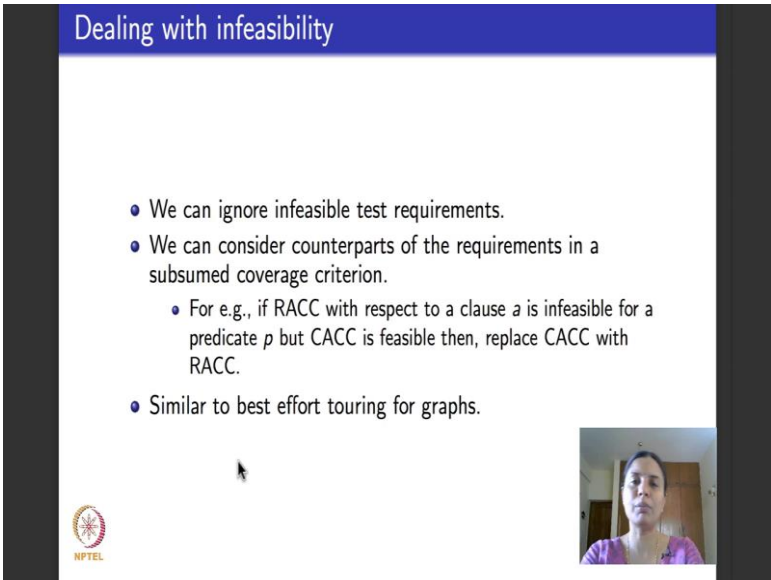



So, CACC is a feasible TR for the predicate  $p$  in the presence of those 2 additional constraints. Similarly now let us go back and look at RACC restricted active clause coverage criteria. Here is what we did in the last lecture to be able to derive at test cases for RACC. So, after working out we realize that one of these combinations: row 1 plus row 5, row 2 plus row 6, row 3 plus row 7 would suffice as a good set of test cases for restricted active clause coverage criteria.

But now if you add these 2 extra constraints that were given here which resulted in only rows 3, 4 and 6 being possible candidates for test cases. You see RACC can never be met because it says one should be paired with 5, both are ruled out because one violates both the constraints, 5 violates constraint 2 and it says the next possible combination you can consider is pairing 2 with 6, 6 is fine because it does not violate any constraint, but 2 violates constraint one.

So, that is also ruled out. The next possible combination of test cases for RACC could be pairing of rows 3 and 7. Again if you go back 3 is fine, it does not violate any constraint, but 7 violates constraint one. So, none of the three combinations that we had listed as possible test cases to achieve RACC criteria now for this example, in the presence of constraint are and all 3 of them become unusable. So, we say RACC is infeasible for the predicate for the predicate  $p$  when 'a' is the major clause because there are 2 additional constraints in the system. So, like this when you take logic coverage criteria and try to apply it for practical examples because of the presence of the additional constraints in the system some of the coverage criteria requirements could become infeasible.

(Refer Slide Time: 27:20)



The slide is titled "Dealing with infeasibility" in a blue header. It contains a bulleted list of strategies for handling infeasible test requirements. In the bottom left corner, there is a small NPTEL logo. In the bottom right corner, there is a small video inset showing a person speaking.

- We can ignore infeasible test requirements.
- We can consider counterparts of the requirements in a subsumed coverage criterion.
  - For e.g., if RACC with respect to a clause  $a$  is infeasible for a predicate  $p$  but CACC is feasible then, replace CACC with RACC.
- Similar to best effort touring for graphs.

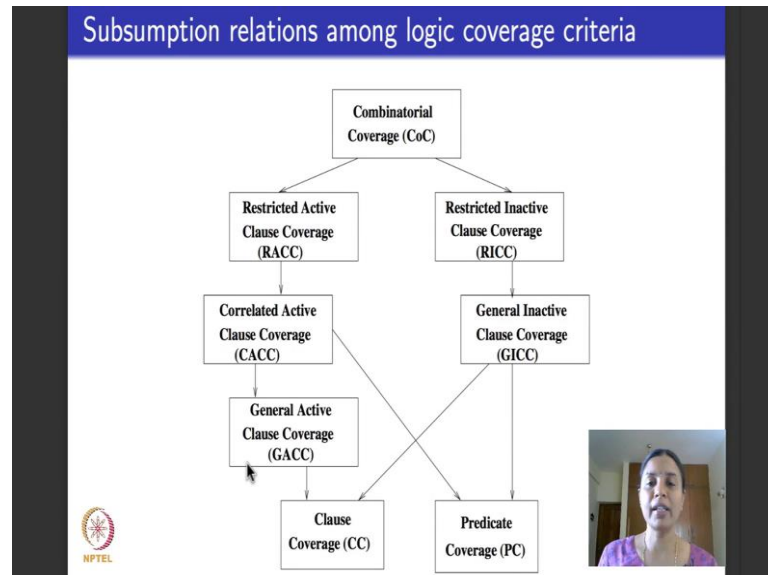
So, hopefully this example would have helped you to understand how infeasibility of these test requirements come into picture. So, how what can we do with infeasible requirements the simplest recommendation that people give as far as testing is concerned is that ignore them and move on ignore the infeasible requirements or we can consider other counterparts of the requirements like for example, if we in this case we realize that RACC is infeasible, but CACC was feasible.

So, you ignore RACC, but consider CACC right. If you remember this is somewhat similar to doing best effort touring graphs. So, whenever we say prime path coverage is infeasible you consider achieving the same thing using side trips and detours. So, you



ignore and you try to replace it with other coverage criteria that would be feasible. That is how you deal with infeasibility in logic coverage.

(Refer Slide Time: 28:16)



So, this is the final slide. What I have done here is I have given you the subsumption relation for coverage criteria. If you remember the definition of what is subsumption, we say one coverage criteria subsumes another coverage criteria if the set of all test cases that satisfy criteria 1 also satisfy criteria 2. So, we realized through an example in the last class that clause coverage and predicate coverage do not subsume each other. There are predicates for which you can achieve predicate coverage, but you may not exercise each clause. There are formulas for which you can achieve clause coverage, but the predicate will be true in both the cases.

So, there is no containment here, they are separate. We know that generalized active clause coverage subsumes clause coverage because it exercises each clause to be true or false and we saw through an example that generalized active clause coverage does not subsume predicate coverage. So, I have not put any arrow here and we also saw that all the ICC criteria, just a little while ago in today's lecture I told you that all the ICC criteria subsume predicate coverage because they test for the predicate to be true and they test the for predicate to be false. And we saw in the last class that CACC subsumes predicate coverage. I know that RACC subsumes CACC, which in turn subsumes generalized ACC this directly follows by definition. Combinatorial coverage is the master of all

coverages because it says you test the predicate for every combination of true false value exponential in number. So, it subsumes all other logical coverage criteria.

So, this is how the various coverage criteria are related amongst each other. So, what we will do in the next module is I will tell you suppose given a predicate, given a set of clauses how to make a clause determine a predicate. Are there algorithms, methodologies to be able to do that. So, that will be the end of week 5 lecture for us.

Thank you.