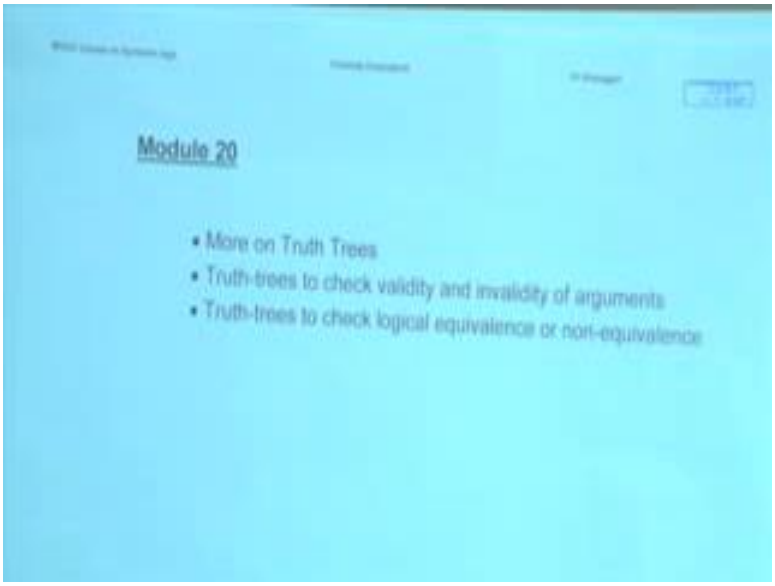
Symbolic Logic
Prof. Chhanda Chakraborti
Department of Humanities and Social Sciences
Indian Institute of Technology, Kharagpur

Lecture - 20
More on Truth - Tress
Truth – Trees to check Validity and Invalidity of Arguments
Truth – Trees to check Logical Equivalence or Non – Equivalence

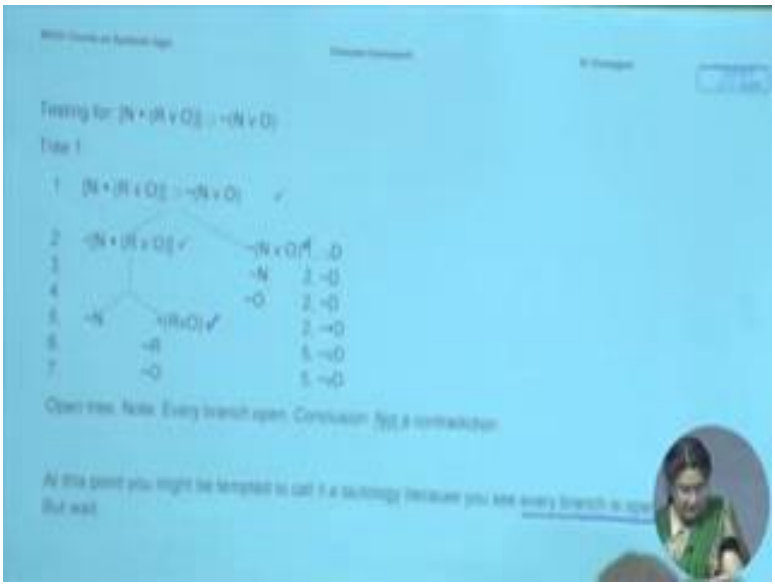
Hello! We are doing truth-trees these days. So we are back again with little bit more on the truth-trees.

(Refer Slide Time: 00:30)



And this would be our last lesson on truth-trees; so we are trying to finish the tasks that we can ask it to do. In the last module we were looking into the categorization of propositions. So, we will be speaking a little bit on that also by truth-trees. And then today we are specifically going to look into the validity and invalidity determination. So, arguments : Are they valid or invalid? Can we find that out by using truth-trees? And the last task would be whether given two propositions the truth-tree is able to tell us whether this is logically equivalent or not. So, our task for Module 20 is like this : That we will be doing follow up on the earlier module a little bit and then go into two more tasks.

(Refer Slide Time: 01:25)



I said that we need to do a little bit more on the categorization, because I don't know whether you have captured or you have grasped the truth-tree procedure completely by now. And it may be very tempting for you to infer what it can do or how to read the tree. In order to correct any kind of misconception that there might be in your reading of the truth-tree, I thought that this kind of an example or this discussion would help you a little bit. This is connected to our module 19 discussion on the categorization propositions into tautology, contradiction and contingent.

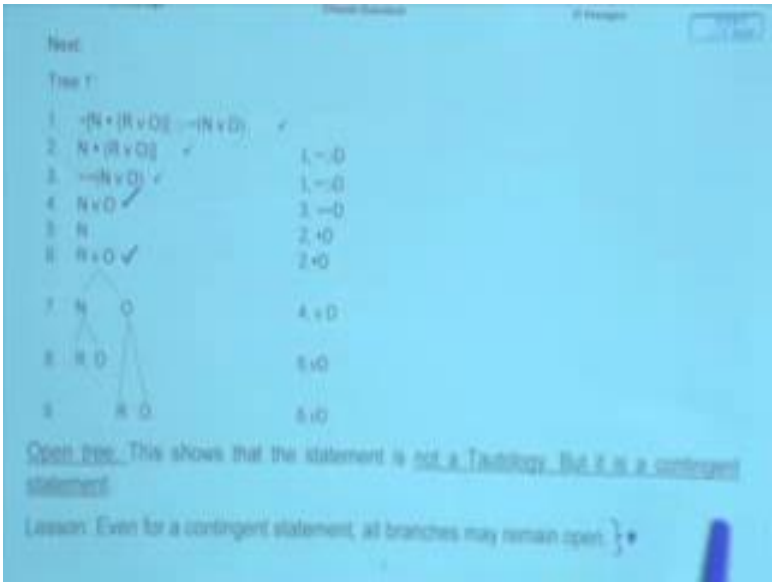
Here is an example where we are doing… trying to do a truth-tree determination what kind of a proposition this is. So, here is [N • (R ∨ O)] ⊃ ~(N ∨ O). And if you take this proposition and try to do a tree right on that. Right? So, remember to tick it and it will give you this kind of a result, because it is a horse shoe (⊃) proposition. You have two branches; again I will remind you that you need to hold one of the branches while you are doing the operation on the other. For me I always work on the left hand side branch first. Its upto you. But if you are doing that, if you want to reverse the process then let's do the right hand branch this time.

So here is this. What we have done is this one we have the decomposed into this. And here is the result at the left hand side and so on. So, this one is decomposed and we have resulted in this, and this is the result of decomposition of this one, and this is the result of decomposition of this one. What do you see? You see completed open branches

everywhere. Every single branch of this tree is a completed open branch. Fine? So, you could have stopped right here also and it will give you an answer. But what is that answer that you have obtained at this point? The answer is, the logically correct answer is that it's not a contradiction, because the tree has not closed down. Even just the small branch would show that.

Are you entitled from this to jump and infer that since every branch is open, it has got to be a tautology? Because what we have said is that if it is a contradiction, all the branches are going to close down and it is going to result into a closed tree. Fine? Now, with that it might seem to you that therefore in case of tautologies what will happen? Every single branch is going to remain open. Is that… how safe is that? How correct are we in inferring this? And therefore, it's important to see that what would be the result if we negate this proposition and do the tree.

(Refer Slide Time: 04:58)



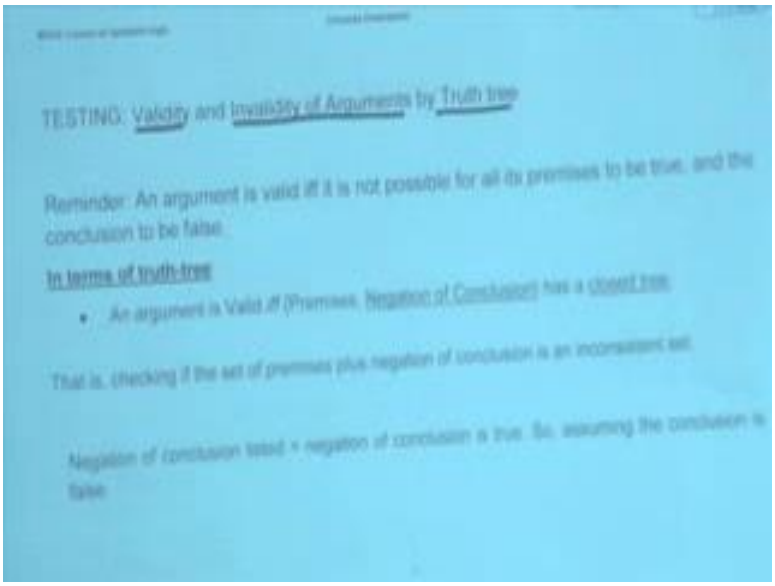
So, here is the negated proposition. Ok? There is a tilde in front of this and you can now decompose it like this. Ok? Remember to put a tick on the ones that you are decomposing. Here comes slowly the decompositions of the other things as we are going along. So, this is your on the …sorry, ~~D decomposition and then comes the second line decomposition. This is the line that you are decomposing. After that comes the necessary branching. And here is your line number 4, ∨D wedge decomposition. Then

comes the 6; namely, this is line number 4, this is line number 6. This is what you have decomposed and the result looks like this.

Now tell me are we complete? Are these first of all open branches? Are these completed open branches? And we see that they are. Ok? Each one of this is also open, completed open branches. What does that tell you? Remember the same one when you did directly on the tree, it has not closed down, every branch was open. Had it been a tautology, what would you have seen? The negated proposition would also result into a closed tree. It hasn't. It is therefore, you are entitled… See, it is a contingent statement. Because neither the tree on the proposition itself has resulted in a closed tree, nor the tree on the negated proposition has resulted into a closed tree. So, this is a legitimate contingent statement.

But what is the lesson to learn here? That even for a contingent statement as you can see every single branch may remain open. Right? As you can see in the first one that when we saw this is the proposition, this was the scenario that you had earlier. So, given this let us not quickly jump into any conclusion until you have done the further test. At this juncture all you are entitled to say is that it is not a contradiction. This test further proves whatever result you want to give. In this case in turns out be a contingent statement.

(Refer Slide Time: 07:54)



So, this I hope will be a helpful lesson to remember by. And then now we are going to move into our new task for today; namely, validity and invalidity of arguments by truth-

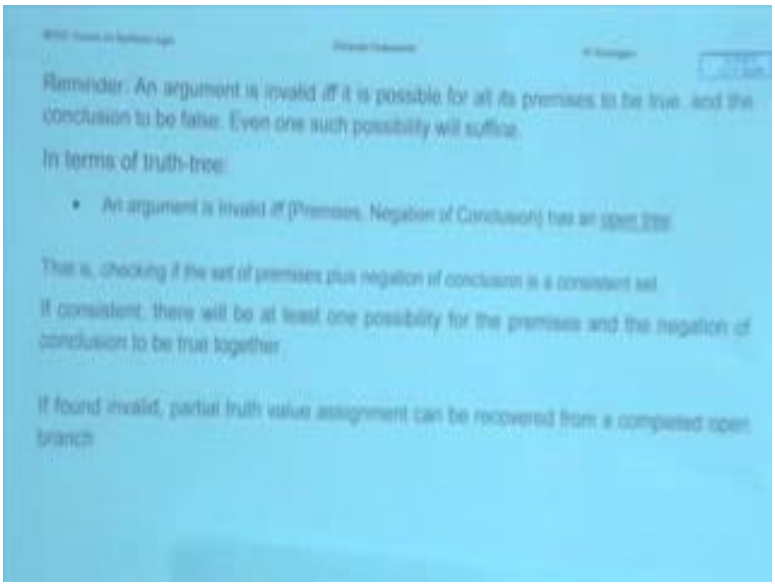
tree. We are going to solve the problem of whether an argument is valid or invalid by doing the truth-tree on the argument itself. Let us remind ourselves when do we call an argument valid? When it is not possible all its premises to be true, and its conclusion to be false.

So, in terms of the truth-tree, what we will do, we'll somehow will reduce this task into a consistency checking. What we will do is this : That we will try to form a set with the premises and *negation of the conclusion*. Once more, the premises are given, the conclusion is given. For the truth-tree what you need to do is to form a set where the premises are the members and the *negation of the conclusion* is also a member. What are you saying? What are you checking here? Whether the premises' truth is consistent with the truth of the negation of the conclusion.

What does that mean? What we are trying to see is whether when the premises are all true, whether the conclusion can be false or not. So, whether it is consistent to claim the premises are true, conclusion is false. If that is not consistent, then you are going to have a closed tree. Which means that there will not be even one situation when the premises are true and the conclusion would be false, which is exactly what we need to establish when the argument is valid. If it is invalid, there will be at least one occasion when the truth of the premises and the falsity of the conclusion will be consistent. Ok? This is going to be our modus operandi for this. Only thing that you need to, sort of, get hold of is that when you are listing the negation of the conclusion in your tree, you are claiming that the negation of the conclusion is true. Which means what? The conclusion is false.

So, with that we are going to now set up the truth-tree to for this task. Remember that negation of the conclusion is not given. You have to add a negation to the conclusion and list it yourself in the truth-tree. So, that is going to be an addition to your truth-tree. Let's see an example so that we can talk about it, but before that the invalidity let's go over that.

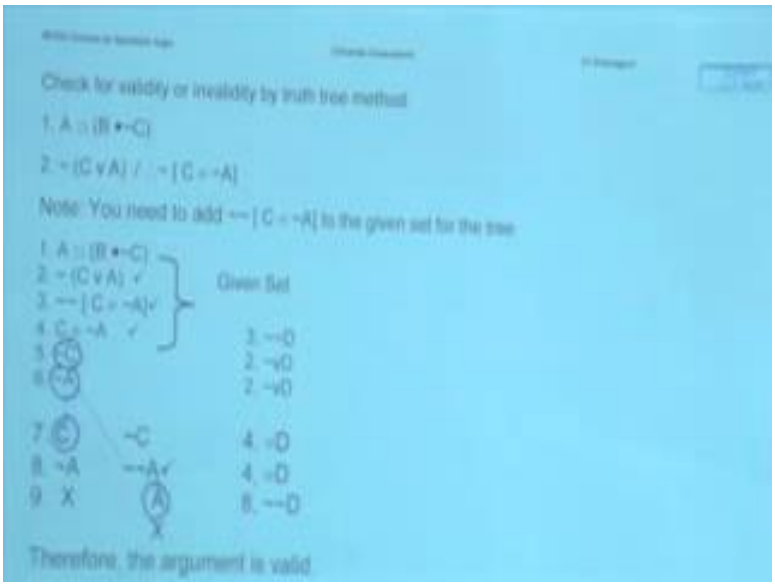
(Refer Slide Time: 10:30)



So argument is invalid when there is one possibility, even one possibility is there where all the premises are true but conclusion is false. Right? So in terms of the truth-tree what will happen? That the same set, the premises with the negation of the conclusion, will be an open tree. There will be at least one completed open branch where the consistency of this set is shown. So, this is our way to show arguments' validity and invalidity. And remember that if you find a completed open branch, from that branch you can recover partial truth value assignments to show when the arguments would be invalid in this case. So, the completed open branch will do the job of that particular row in your truth table where you show the premises are true under this kind of situation but the conclusion is still false. Ok?

. Let's take now the example so that we see this happening.

(Refer Slide Time: 11:40)



Suppose we have this problem in front of us. So, these are two premises and here is the conclusion. So what you are going to do now: Line number 3 in this tree is going to be the negation of the conclusion. This whole thing is your conclusion, ~(C ≡ ~A). When that is negated, what you are going to have is ~~(C ≡ ~A) and that has to be added to the given set for the tree. So now, you are checking whether 1, 2 and this new statement, whether that is going to be a consistent set or inconsistent. Whether it is going to have a closed tree or an opened tree. And accordingly you can tell whether the argument is valid or invalid.

So, here is the new set. These are two premises and here is the negated conclusion. Rest is as usual, you will do the tree as usual. So here is our first decomposition ~~D. The moment you do it, remember to tick it on and then here is the solution of line number 2. We see some potential possibility of closing quickly. So we will follow that. So, this is your 2, line number 2, ~∨D decomposition.

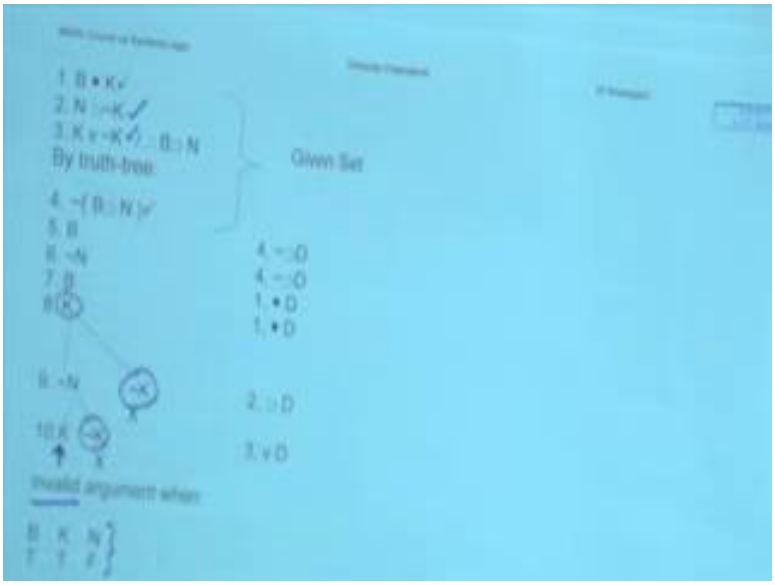
And then comes this, this is your ≡D decomposition and we see that when decomposed it will give me something like this. And right here you can see C and tilde C will result into a closed branch. This is ~C, ~~A which when further decomposed with ~~D decomposition gives you A and here is not-A. So the branches are all closed down. Branches are all closed down; therefore, the argument is valid. Right? You just showed that this set cannot be a consistent set. You have just said that it cannot be when the

premises are all true, the conclusion can be false, or the negation of the conclusion can be true. Right? So that shows the argument is valid.

This is how we show validity of an argument, not a single branch is open. Now if you closely look into the tree, as you can see, that it has closed down even before you have done line number 1, true? Because we chose those statements first which we thought would result into quick closure of the tree. And line number 1 was not even necessary. So, there is a lesson right there, that when you are doing the tree it may be so that you can get the results. See here the result is already given. Had there been any open situation, then you still need to do number 1, but right here you have closed it all. You have you generated branches that have automatically shut down.

So, when you have obtained the result, already have derived the result, there is no reason why you still have to do line number 1. So there may be such trees where every single given statement need not be decomposed. Ok? That's the further reason why I say that you need to do trees strategically and with the little bit of an open eye.

(Refer Slide Time: 15:22)



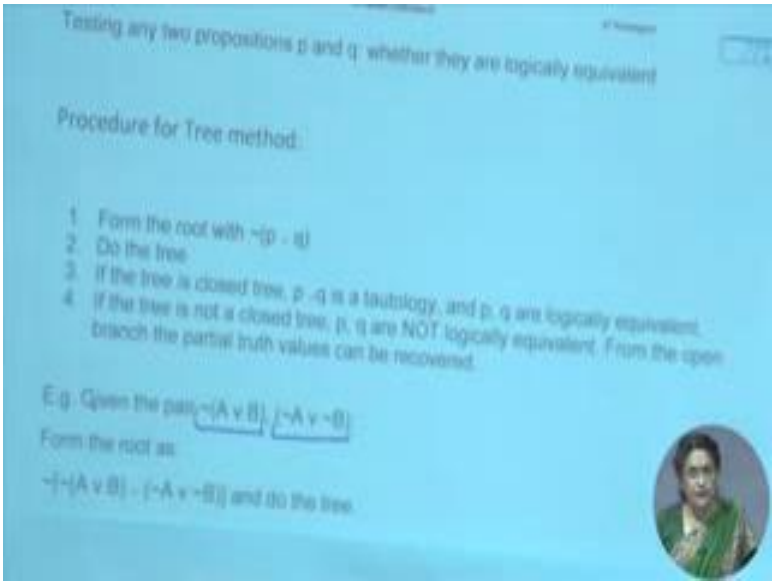
Finally, we come to this one. Let's do another example before we leave the topic. Here is an argument, (B • K), (N ⊃ ~K), (K ∨ ~K) and therefore B ⊃ N. And by now you know how to set the tree. How do you do that? Line number 4 is going to be negation of this conclusion. Right? So this is what you are going to add to the tree. Now this is your given set. You do not include the conclusion. You include only the negation of the

conclusion. And the rest is the tree as usual. So please feel free to do that. You do not have to do it as I have done, but please finish the tree.

This is decomposing on line number 4 and then we do it on line number 1 and then comes the branching. This is line number 2 and we put a tick mark right here, and then already we have generated closed branch. This one is still open so, we do a little bit of decomposition line number 3 on this one. And when you have done that, you find that this one would close down. K and ~K both are here. But this is a completed open branch. And from that we can now say since we have at least one completed open branch in this tree, the argument has to be invalid. And not only that, we'll be able to tell *when* the argument will be invalid. How do we do that? We recover the truth values from here. You know what value K has to be. Because, it is listed, it has to be true; so we can write here true. This is ~N . Because it is listed, therefore N has to be false and then comes B, B has to be true.

So, this small truth table here can show us when the argument is going to be also invalid. This is how we show by truth-tree how the argument's validity and invalidity can be shown.

(Refer Slide Time: 17:39)



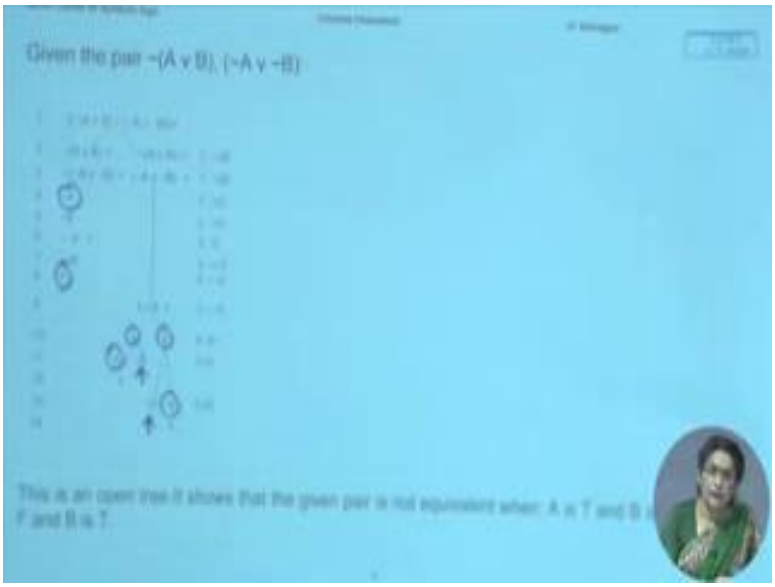
The last remaining task is logical equivalence. When are two propositions logically equivalent? We know in terms of truth table when they have exactly the same truth values. But what about truth-tree? How do we show that? What we try to show is

whether p ≡ q, take any two propositions p, q, whether p ≡ q is a tautology or not. That's what we try to show. So how do you show? Remember tautology proof? Tautology proof is that you take the proposition that you want to test for tautology, add a negation to it and do the tree and see whether it results in a closed tree or not. That's exactly what we will do.

So here, our given propositions are p, q, and we are testing whether they are equivalent or not. So, we formed this statement p ≡ q and add a tilde (~) to it. What are we doing? Once more reminding ourselves, we are testing whether p ≡ q is a tautology or not. So, just regular tree-doing. If it results in a closed tree, you know p ≡ q is a tautology and p, q therefore must be logically equivalent. And if the tree is not a closed tree, then you know p and q are not logically equivalent. There is going to be at least one completed open branch in that tree from which you can tell under which truth values p and q are *not* going to be logically equivalent. Get me? So, this is what it is.

For example, if you have the pair, this whole thing is your p and this is your q. So, this is your p and this whole thing is your q.Right? So, ~(A ∨ B) that is p, and (~A ∨ ~B) that is your q. What you do? You form a proposition like this. What is it? We take p ≡ q and add a tilde to the entire proposition. So this bracket, square bracket, is for that : To designate that inside there is p ≡ q and we add a tilde in front. And then we do the tree to see whether we have logical equivalence or not.

(Refer Slide Time: 20:16)



So, let's do the tree. Here is the pair and here is the root of the tree : The tilde of the two propositions. If we do it correctly, then this is how it is going to look like slowly. This is negation of… ~≡D decomposition, and this is the left hand side and it would result like this. And it will give you a closed branch. Ok? Because you have A and ~A. You also have B, ~B. But we will go take this one and will close it down. Here also you are going to have some closed branches. For example, A and ~A. But this one is open. Here also you are going to have not-B and B, so you close it. But this one is going to be completed open.

So you know, even at this stage you can stop and say that I have found my answer. This one is closed, but here is a completed open branch. And that tells me that the set, this pair is not logical equivalent. So, overall you have… what you have found that if you follow this completed open branch, then A is true and B is false. That's when this not going to be logically equivalent. If you follow this branch you will find when A is false and B is true, these two will not be logically equivalent. Ok? This is how you show… your clue is that it's an open tree and what we are trying to do is to test it whether p ≡ q is a tautology or not. Ok?

With this we will try to finish this module right here. Our discussion on truth-tree is over now, and there may be… you need to practice a little bit on your own just to see where the operational difficulties might be. But conceptually you should not have any difficulty by now, because if you are really familiar with the rules, if you have seen, so many trees have been also done in front of you and I always tell you that you try to do it along with me. If not then afterwards. At least try to do the tree and then you see the result that has been done here so that you get some practice on how to do the trees.

So conceptually you should be clear by now. But then, you know, every time when you do the practice there might be some little bit of an issue. But with practice that should go away also. But now we have finished all the tasks that we wanted the truth tree to do for us. Just like the truth table it is also able to give us clear answers on certain things as I have shown to you. So, here we close module number 20.