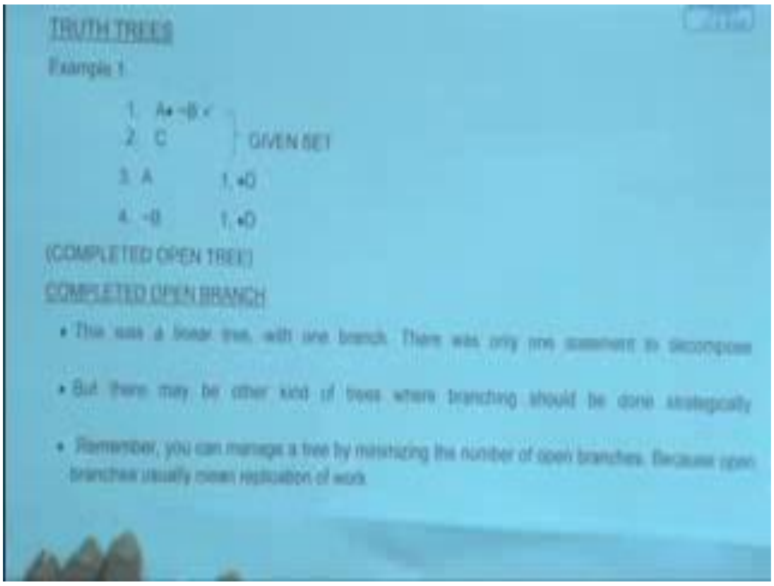
Symbolic Logic
Prof. Chhanda Chakraborti
Department of Humanities and Social Sciences
Indian Institute of Technology, Kharagpur

Lecture - 18
More on Truth-Tree
Recovery of Partial Truth-Values

Hello! And we are now back into this truth tree and there is more to learn in this. So, earlier in the earlier modules, what we have done I hope you remember that, and that it has given you some platform to start doing the truth tree on your own.

So, we are going to learn a little bit more on how to do the trees and what are the things to see. But this is the agenda for our module 18 today : That we are going to learn more about the truth tree; how to construct, how what are the ways to do it. And then there is something else that we are going to specially looking into : that is called *recovery of partial truth values*. Recovery of partial truth values. And this is something that we will learn how to do it from the truth trees and you will see that this is going to go a long way when we start posing it the questions that we did to our truth tables, for example. This would become a major way to recover information from truth trees. So, first we are… right now we are going to learn how to do the trees better, and how to do the trees actually and then… then the point will come how to utilize the information from the truth trees. So, first construction, and then how to make it talk to us properly. So, this is what we are slowly going to.

(Refer Slide Time: 01:54)



In this segment we are going to look into various worked out trees. Worked out trees as in I am working them out. What I would request you is to do it along with me. As you are understanding, also you try to replicate that on your own, in your time, but in a piece of paper and so on. That, that itself is a learning process, you know, because I am doing it and then you are doing it together, and that is how you gain the way to do it into this.

So, here is an example of a truth tree. There are two members in the root. A • ~B and there is C. And this is your given set and you are asked to do the tree of this. Fine?

So, what you do? Remember the starting point is always the root and you take a good look at the root. Can you do any branching on C? The answer is no. Why not? Because it is a literal by itself. There is nothing to decompose here. So, your obvious choice for decomposition in the root will go into 1. When you have the root, the only way the tree can be done is through decomposition on the root. And, in this case the only candidate for decomposition is line number 1. Am I correct? So, this compound then needs to be broken. What is the decomposition rule to be applied here? The answer is : Look at the main connective. And, line number one the main connective is a dot. So, now, you know that you need to apply the *dot decomposition rule* (•D) on this line to get the branches growing. And if you have done it, I will give you just half of a second to do that before we show the result so that we know what to compare.

So, here is your tree. If you have done it correctly, then this is how it should look like. I will remind you once more that it is very important that you number the lines that you are generating. Of course, the root has to be numbered also. That is how we knew; there are two propositions in here. But this is a new line, this is a new line and each of these new line also will have to be numbered.

What is this? This is the justification panel. So, every line that you have generated, I said earlier must be justified. How do you justify? By referring to the line that you have decomposed. In this case that is line number 1, and see, we have mentioned 1 here, 1 here. Then comes : Which rule? By which rule you have decomposed this? The answer is : By dot decomposition rule. The separator between the line number and the rule name is a comma. So, 1 comma (•D) is the justification for 3. And you know this is how it has been… line has been obtained. Same goes for line four. Fine? And as soon as you have decomposed, you I told you to remind yourself so that this is checking the compound. That is your tick mark for telling yourself that I have finished the decomposition on this line.

Here comes the question. So, at line four are you done ? Is the tree complete or not? And remember the way we have defined the completed tree. So, have you reduced the tree into the level of literals only? All the decompositions that were to be done, is it completed? Done? Finished? What do you think? And the answer is that everything has been finished on this. Because there was only one candidate for decomposition here, and that has been taken care of. And what you see here is a branch. It does not look like a branch to you, but this is a single branch; and on it there are only literals now available. This one is already checked compound. Fine?

So, what you have in front of you can be called a completed open tree. Completed because all decompositions here have been finished and the tree is now at the level of literals. Open tree , completed open tree, because everything has been finished and still the tree branch has not closed down. Why not? Because, if you follow this branch, there is no literal and its negation both appearing on the branch. Am I clear? So, this is what you have and this is what you have finished doing a small tree with me. Fine? Is this alright?

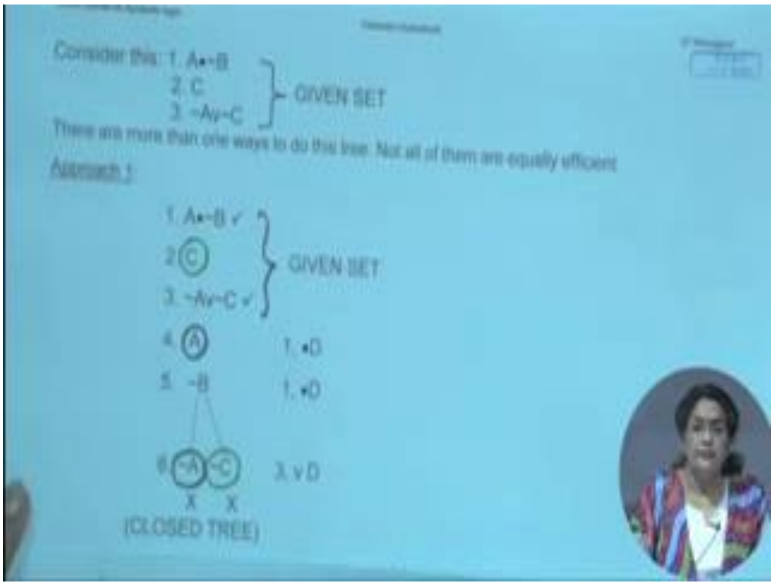
So, this is now. How to read it etc. will come there. Just now we are learning the procedure : How to do this on your own. Ok? So, we'll do more of this. But before that few points for me to mention. Notice, that it does not look like a tree, but it is a tree and you have generated a branch. It's only one branch and it is a linear branch. So, it does not look like a bifurcated, very rather… showy kind of tree, but it is a tree nonetheless. There is only one statement to decompose and we finished it. So, that tree is completely done.

In here, you had very little choice, right? In here means when you had the root, you had very little choice how to do the tree. The only answer is only line number 1 is the point to start or decomposition on. But note, that there can be other situations; other trees where it is not very clear which one to start with and why. Ok? So, in those cases, what I am trying to point out is that the tree may be needed to be done *strategically.* What is the strategy? The strategy is to see that the tree remains manageable. One of the ways to manage the tree is to see how quickly, say for example, you can generate closed branches, or how quickly you can have completed open branches. Remember these are the indications that the tree is about to… about to be over. The doing of the tree has to be over. How to minimize the replication of work in every branch, these are the perceptions, these are the insights that is behind the tree-doing strategy. When I started out I said the truth tables are very mechanical, but trees, truth trees are not that mechanical. You can manage them, you can you can sort of make it go in the way you want and so on. So, one of the reason is that it is a sort of strategized operation. I will show you some examples of that.

So, one of the ways in which you can minimize your work or manage the tree is to keep a track of the number of open branches. Why? Because the number of open branches, if it is large, then remember we have a rule that every open branch that runs through a compound gets the result of the decomposition replicated. So, which means in every branch that is open under a certain compound, you have to repeat the decomposition results, right? So, if you are trying to minimize, that is, you… the number of open branches that is, you can… you try to see how quickly can you generate some closed branches. That means that on those branches you do not have to work on anymore.

So, minimizing the number of open branches is one kind of a strategic advice that I can give you. How do you know which branches are going to open, remain open and which ones is going to soon close down? Well, that's where your grasp of the decomposition rules come. You need to understand this decomposition rules and sort of approach the tree with an open eye. Look at the root and you know what the decomposition rules can give you. So, you can soon see, it is like a chess game, two-three steps down, you can see that if I do this, the branches going to close down fast, right? So, that is the kind of work, that is the kind of perception that you are going to gain. The more you do trees the more you will have better grasp on this. But right now I will just mention this: That some trees will give you no choices, some trees will give you choices, and if you use these choices intelligently, the truth-tree really can be an efficient tree to show you the results. Ok? So, let's see other examples as we go along.

(Refer Slide Time: 11:30)



Consider this example for example, that we have the branch right here. Sorry, we have the root as this. Three members in the root. (A • ~B), C, (~A ∨ ~C). Ok? Now, understandably given this, then the job is to decompose. But where do I start? And more importantly is why? That is the question to ask. Clearly 2 is already a literal. You, there is nothing to be done on 2. But 1 is a choice, 3 is a choice, where do I start and why? Now let us see. I will show you parallel ways of doing this trees and may be more than one ways to do this tree; so that you can understand what I was talking about the strategy and sort of becoming… with that I hope the proficiency will increase also.

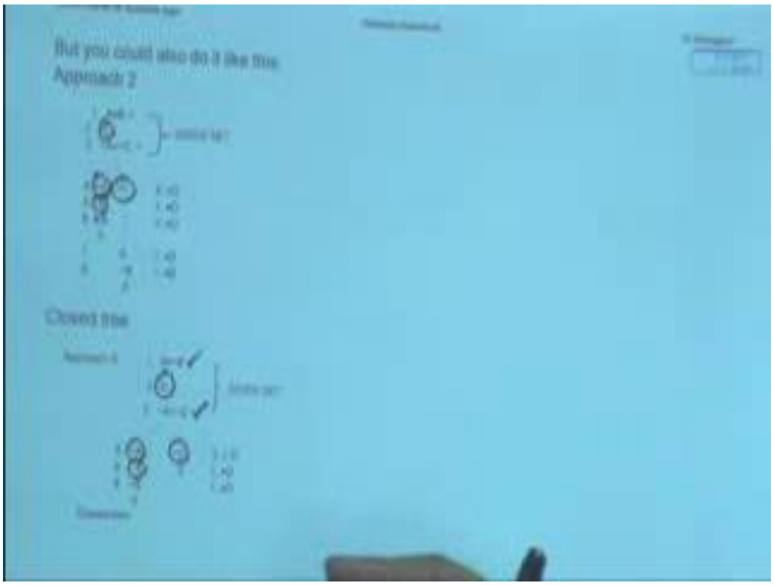
So, more than one ways to do a tree. And my point is not all of them are equally efficient. So, getting that is which one works for you, is going to be practice, matter of practice. But let's see right now. So, I will say I will start with approach one. Suppose, somebody is doing the tree like so. So, this is your given set. That is your given set. And the person said, whosoever it is, that let me start with line number 1. For no apparent reason, but the person probably thinks that's the first which is coming in order, so I will do that. Fine, that's ok. So, here is 4 and 5 obtained from line 1 by •D dot decomposition. Fine, so far?

Now, what? And then immediately this gets a tick, which means that line 1 has been decomposed. But remember line 3 is still there. So, you have open, but not a completed open branch. How do you write the result of 3? Under the open branch. So, this gets bifurcated and here is the result of your writing this. This is ~A or ~C obtained from line 3 ∨D wedge decomposition. Fine? And then you put tick here. Got me? Now is this complete? Is there anything else to be done? Take a look and you will see that all the branches are at the level of literals. When you are at this kind of stage, you need to now pay closer attention to every branch. Take a look into this branch. Is it open or closed? And you follow it through, and here when you reach here, you find *uh-oh*, here is ~A and this is A. Which means what? That this branch is going to close down. Fine? Let me encircle that for you. So, you encircle it, like so. That's a clear marker why this branch has to close down. Let's take a look at this other branch. ~C, ~B, A, but here is C. So, you have ~C and C. Notice how far you have to go. This is all one branch. So, go until up to the root C, whether it is open or closed, and in here you found that here is a contender.

So, here you have ~ C and here you have C. what happens here? The branch closes down. Ok? So, in under each of this branch you are going to put the cross signs. Remember the cross sign is there to indicate that this branch is closed. This branch is closed. What does that make this tree? It's a tree where every branch is closed down. So, it is a *closed tree*. Get it? So, this is how to do the tree. It is again a fairly simple tree, but my idea is to show you how you can approach the tree on your own, and hope you have followed the procedure as I have explained to you.

I also said that this is only one way to do this tree. There may be other ways to do it. So, we will take a look into that. The result is not going to vary, result may not change, but the tree may look different. The number of steps may decrease, increase and so on. And then you decide how to proceed with this. See here, if you review this tree, then this approach shows that the person preferred that first to keep it linear as far as possible. When there is a choice between two decompositions and one of them is linear, no branching, only single branch sort of comes out, no bifurcation, that is branching, but no bifurcation; and this one gives you the bifurcation option. Then this approach shows that the person has decided to keep it linear as far as possible, and then only when it comes to compulsion then that bifurcation comes. Ok? Just to keep the matter of replication to the minimum. How many steps here? Six. Ok? Let's take a look into other ways of doing it.

(Refer Slide Time: 17:33)



See this is approach two. So, you can also do it like this. So, here is that set doesn't change. This is your given, and then suppose somebody said that I want to start with 3, line number 3. Ok? Nothing wrong with it as per se. Because, you will have the same result, but the tree is going to look completely different. So, you need to put a tick on line 3, but your line number 1 decomposition is still left. So, how to do that? Well, you have to do it here on this branch once, and again on the right hand branch again. Why? Because, this belongs to both the branches. Remember the parental property analogy that I gave you? So, both the branch deserves their result of the decomposition of that.

And I said when you are working on one branch; you please keep the other branch on hold. Do not… even if it is the same rule application, do not try to, do not attempt to do decomposition on all the open branches at the same time. Because, then you are more prone to error and some of the important information might be overlooked and so on. So, one branch at a time when you are doing decomposition, even if it is the same decomposition rule that you are applying. Ok?

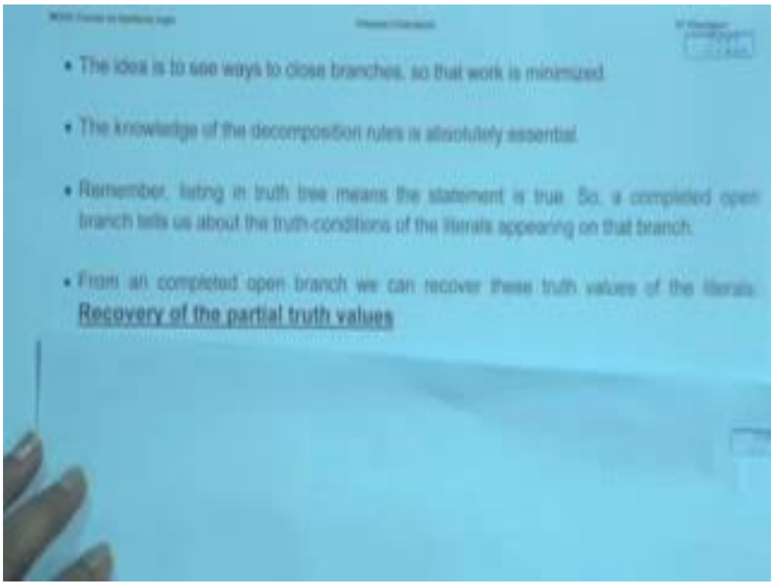
So, after line 4, then this person is doing on this side, line number 1 decomposition and result is A and B. And therefore, you.. the person says we have this… sorry this is not a ~B, this is just B; and then, you get that this is where you need to stop it. Because you have A and ~A. And line 7 and 8, you have A, ~B. And here you are; you need to see that here is ~C and here is C. So, branch is closed. So, in a way you have generated a closed tree. This is also closed tree, but look at… look at the number of steps. So, earlier it had 6 steps and here you have 8 steps. Point is that between these 2 approaches you might prefer doing the first way, though it is this is approach two is not essentially wrong, but it is taking your little bit more time, little bit more time. And it is done rather , in a sort of random way without any specific logic, but just a person is rather impulsively choosing one option over the other. But there is a third way to do it also.

So, take a let us take a look into that. This is approach three. So, we have the same set. This person said : Look, I am doing this one, line number 3. Line number 3 is here and the results are here, but the… immediately the person has spotted that there is this chance to close the branch down. Right? And the branch is closed. Which means that we do not need to replicate the result of decomposition of line number 1 and 2 both branches. See this is line number 3. We need to put a tick here and this is line number 1. But when you have done line number 3 and one side has closed down because there is C and ~C, thenyou write the result only on the left hand side. And left hand side again gives you the same reason to close it down like so, and it is a closed tree. The result does not change, but what changes is your strategy, the number of steps that you going to take and the rule and so on and so forth.

So, this is one way to get into what we call the doing the tree. Doing the tree is not a blind process, which is what I am trying to explain to you. That whenever… because you knew and because you are approaching it with the newly acquired knowledge, so let me tell you that first of all, you need to really understand the decomposition rules. Your understanding of the decomposition rules will depend on how well you understood the truth tables of the connectives. So, each one is built upon the previous one. If you remember the connectives, how they function, what is their truth table, you will have no problem in understanding the decomposition rules. And those are going to be the staples for doing the tree.

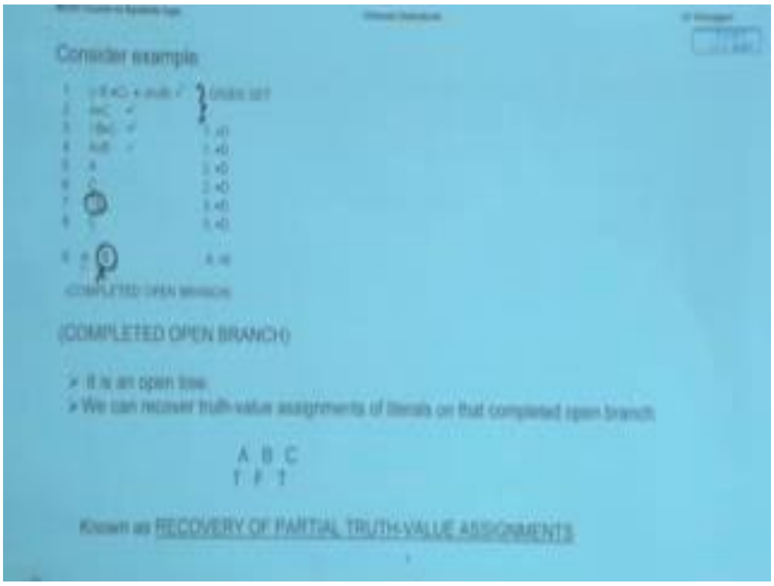
So, say for example, when you see this root here, then unless you know how the dot moves, how the wedge moves and so on, you are stuck at some point. But on top of that what I am saying is that knowing, just memorizing the decomposition rules is never the solution. You need to have a grasp over them, you need to understand why they work in the way they do by paying attention to the…to what the root is talking to you about. And it is important to know the rules, but it is also important to understand the rules.

(Refer Slide Time: 23:04).



Now, remember that I have said that listing in the truth tree means that the statement is true, right? So, whenever you have a completed open branch, I mentioned this earlier, it is telling you something. Namely, about the truth conditions of all the literals appearing in that branch. Right? Because listing means that the statement is true. So, when you have a completed open branch, it means that it is also showing you where the literals are going to be true. When that happens, this is the reason from a completed open branch you can *recover the truth values of the literals*. I will show you, and that process is known as *recovery of the partial truth value assignment* to the literals.

(Refer Slide Time: 24:11)



So, let us take a look, let's take a look and then may be we can understand this better. Suppose we have this as your given set. So, you have (~B • C) • (A ∨ B) as the first proposition. (A • C), ~B • C as your next... sorry, (A • C). Up to here is your given set. This is your given set. Two premises, two statements in your given set and then you are doing the tree as usual. So, you may start.

See both are dots. So, you do not have much of a choice and probably you took the first proposition and here are the results of that dot decomposition. And remember this is why I said you need to check the compounds. So, this is very important that you do this as you go along. So, this is the first one is done. Then you have so many contenders here. Which one do you start? You say : OK I will start at A • C. So, here is your result of A • C. The moment you do that, you tick that, you *check it*. And then there is this ~B • C. So, check it and write the results. Then you please see that there is still line number 4. And this is the reason we need to have this checking in as reminders. And here is the result of the decomposition of this.

When you are at this stage, line number 9, where you have decomposed line number 4, with decomposition your tree is at the level to stop. But this is when, as I said, you need to check which branches are open, which branches are closed. For example, if you follow this through, then you will find that you have a situation here on this right hand branch where you have B appearing here and ~B appearing right here. So, that branch closes down. Ok? But look at what is happening on the left hand side! The left hand side if you follow it through, we have A, C, ~B, C, A and so on and so forth. So, all literals, and none of them are like A and ~A, or B and ~B. So, this remains open. Not only open, it's a completed open branch because every decomposition that was supposed to be done on this branch is finished and everything is reduced to the level of literals. Fine?

This provides us the right opportunity to recover partial truth values. So, from this I can see that A is true, C is true, not-B is true, not-B is true means B is false. And these are the only literals that I have. You get the repetition of this information here: C is true, A is true which is already given here. So, from this we can now form a small, sort of a truth table out of this. See, from this branch we can say, whatwe can now say is that A is true, C is true and B is false. And that would keep this given set the truth conditions. This process is known as the *recovery of truth value assignments*. So, this is something to know from a completed open branch. You can do this process to take care of the partial truth value assignments. We are going to use it in our next modules and so on.