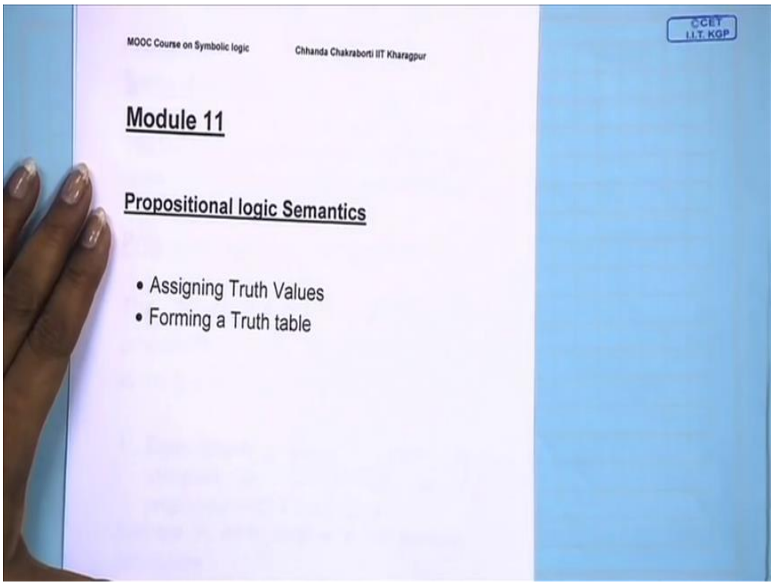
Symbolic Logic
Prof. Chhanda Chakraborti
Department of Humanities and Social Sciences
Indian Institute of Technology, Kharagpur

Lecture No - 11
Propositional Logic: Semantics Basics of a Truth table

Hello. So this is our module 11 of the NOC course on Symbolic Logic. We have just finished, by module 10, the Syntax of PL or propositional logic, where you have learnt how to combine propositions, what are the syntactically correct ways to arrange propositions, to generate, say, compound propositions , and how to use the connectives properly. The symbolization was mainly an exercise to understand the operations of the connectives, where to put, where not to put, and so on and so forth. We have also learnt the scope of the connectives.

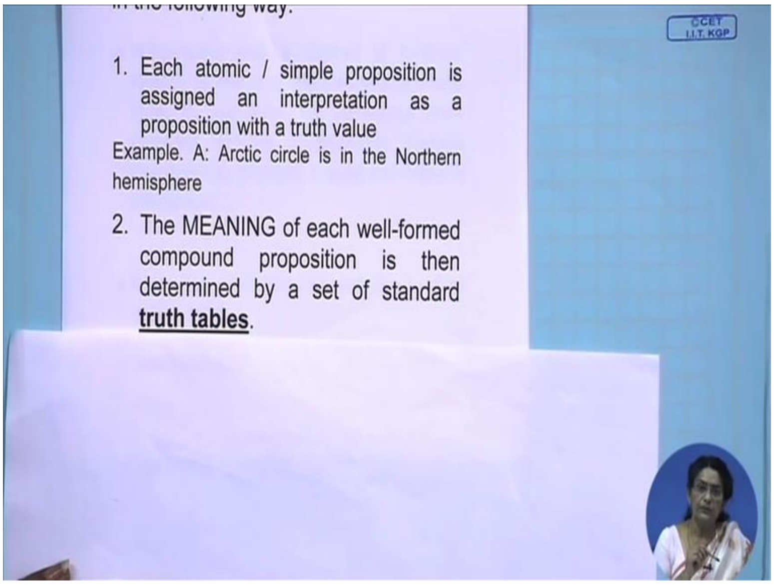
But, today onwards we are starting something called the semantics. If syntax is one of the important components, semantics is also an equally important component for any formal logic system.

(Refer Slide Time: 01:25)



So from module 11 onwards what we are starting is semantics. Propositional logic semantics, I will explain what that is, but mainly what you are going to learn in this module is what does it mean to assign truth-values to propositions, and then also we are going to learn how to construct a truth-table. You may have heard the term *truth table* earlier also. Some of you may have done this truth-table also. But together in this course, we are going to learn the way we intend to do the truth-table. You know, there are more than one ways to do them, but this is our way of doing the truth-table. So this is what we are going to learn today.

(Refer Slide Time: 02:16)



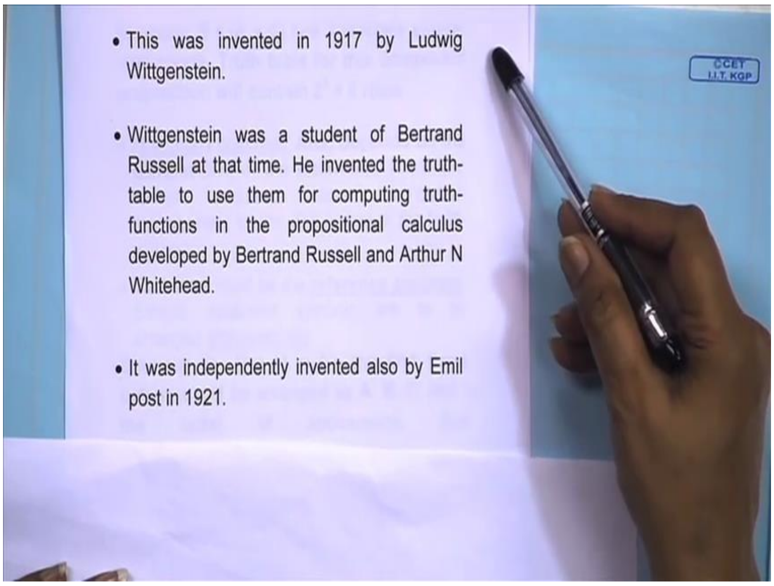
So first thing is, what is semantics and what do we mean by the propositional logic semantics? See, syntax, as I told you, gives us rules, gives us rules to form *well-formed* sentences in a given language. It is the grammatical rules that tell you how to generate *well-formed* propositions in a given language. What is the job of semantics? That once this well-formed syntactically correct propositions or statements are done, how to assign meaning to them. So meaning assignment , or interpreting the well-formed formulas, or well-formed statements of a given language; that is the task of semantics. So semantics teaches us how to assign meaning to the well-formed statements.

If that is what semantics is, then what does propositional logic semantics do? And this is where we need to come through like this. It also assigns meaning to the well-formed propositional logic statements. But the way to assign the meaning is like this, that each simple proposition is assigned an interpretation with a truth-value. Alright? Remember what I said earlier is that you cannot have in PL any proposition that does not have a truth-value.

So here you are. The beginning is that the simple proposition is assigned an interpretation with a truth-value. For example, this is what we have been doing in the symbolization. So this is your prepositional constant, and we give it, we assign it some sort of a propositional statement with the truth-value. For example, ‘arctic circle is in the northern hemisphere.’ That’s a statement which has a truth-value. So that first thing is that. We are taking the elementary units, basic units and we are trying to assign truth-values to them. We are assigning them propositions which have truth-value.

And then, compound propositions are born by the use of the connectives. They are meaning, the meaning of the well-formed compound propositions, are done by processes, semantic processes. One of them is the truth-table. There are other procedures also. So the same way does not work for the compound proposition. Why? Because, the compound propositions here in this PL, they are all truth functional. So the knowing the truth-value of the components somehow is going to reflect back on the truth-value of the compound proposition. So one of the ways to assign the meaning to the compound proposition, as I said, is through truth-tables.

(Refer Slide Time: 05:45)



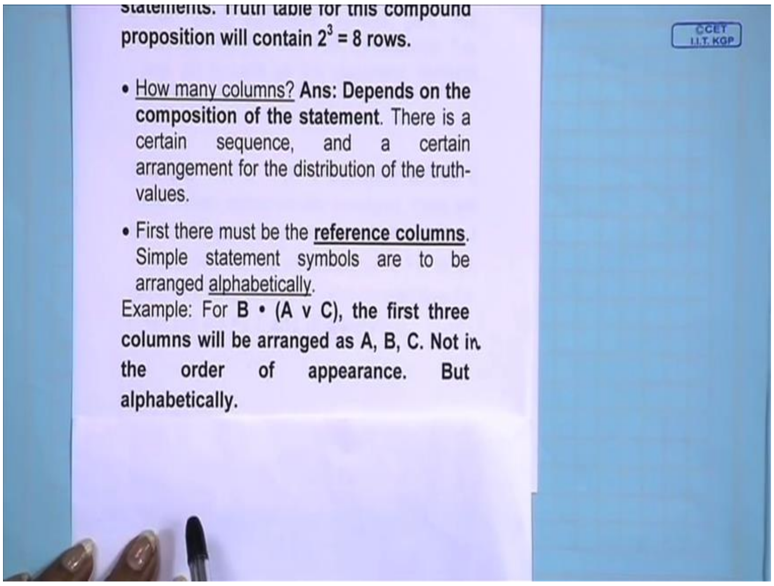
And so now, our next stop would be to learn how to do these truth-tables.

What truth-tables are and how to construct the truth-table? That is the goal of this module specifically. Truth tables. What truth tables are actually a mechanical method. It’s a method that is sort of blind. Mechanical as in the machines also can do it. It doesn’t require much. If you know and you can follow an algorithm and the method will be completed. What is its goal? The goal is to compute the truth-values of the truth functional compounds. That is the goal. Ok?

Typically, it’s a table, unlike other tables, this has the closer connection with the mathematical tables, and we will show you examples, we will show you in a second. Who brought it in is an interesting answer. In 1917, the truth-table procedure was invented by Ludwig Wittgenstein. He was an engineer who converted into philosophy. So he did his Ph.D in philosophy after doing his engineering degree. As part of his dissertation in Ph.D, actually this truth-table was devised. This Wittgenstein, if you recall my lecture on the history of symbolic logic development, then you will recall that Bertrand Russell is the founder of symbolic logic and under him Wittgenstein was one of the students; Ph.D or doctoral student. And he invented the truth-table to enable Russell's propositional calculus to mechanically compute the truth-value of the compound propositions. This was the goal and this is how we got it.

Truth-table was also independently invented by Emil post in 1921. So 1917 Ludwig Wittgenstein did it, but independently Emil post also did that. And this somehow changed the contour of mathematical logic. So we are going to take a look into this procedure closely, and we are going to learn it. So, how to form a truth-table? We know now what the truth-table is, who invented this, and so on and so forth. But how to form the truth-table, where do we start, what do we do?

(Refer Slide Time: 08:25)



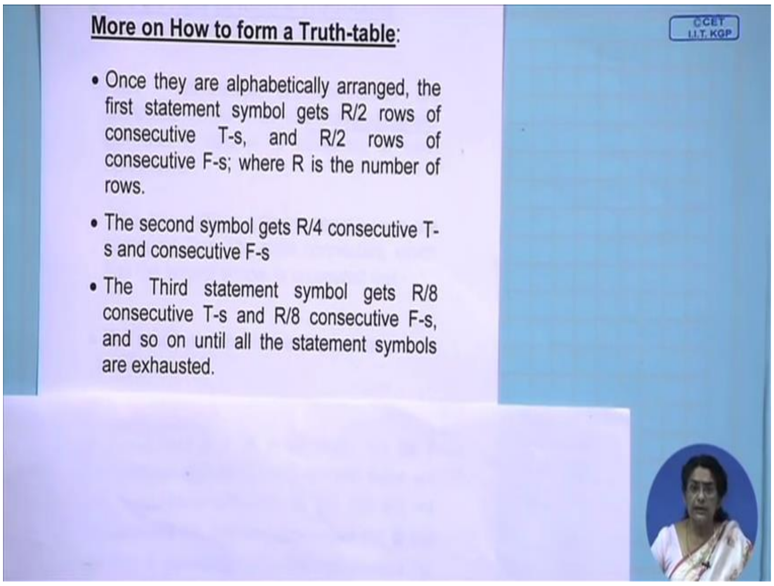
First note that the truth table is composed of rows and columns. So you are going to have certain number of rows, certain number of columns. If you are asking how many rows? How many rows are going to be there in the truth-table? The answer is there is a formula for that. The formula is 2^n, 2^n where the 2 stands for the number of truth-values allowed in your system. Remember, propositional logic domain believes in only 2 truth-values, truth and falsity. Hence, you are going to have 2. And the ‘n’ stands, the ‘n’ stands for the number of discrete simple propositions in the compound proposition. So, together here is the formula, 2^n, where 2 is the number of values and n is the discrete, the number of discrete simple propositions present in the compound proposition. Let’s take an example, so that we clearly understand. Suppose we have B ● (A∨ C). These are all propositional constants. How many discrete simple propositions we have? 1, 2, 3. So how many rows, if I am going to do construct a truth-table for this, how many rows am I going to have? The answer is 2^3, which is 8 rows. Does it help? So if you had, on the other hand, 4 discrete simple propositions, how many rows you are going to have ? 2^4 , namely 16. So this is how to compute how many rows you are going to have in a truth table. It depends on the number of simple propositions. It also depends on number of truth-values that your system allows. 2^n in our case. Ok?

Next is, how many columns? Now that answer is not going to be obtained through any formula, but you have to understand. So the answer to this question ‘how many columns’, we have to say: Depends. Depends on what? Depends on first of all the composition in the statement. What kind of complexity you have in the compound proposition. You will see that you have to assign columns for each of those little bit of components. Ok? So, that is there. Plus there is a certain sequence in which you are going to build the truth-table and we will show you that. So, *depending upon* all these things, the number of columns you are going to require is going to be fixed.

So let us see first of all, and we will take examples also, but let us just proceed anyway. But doesn't matter how many columns you are going to have, what you must have, in terms of columns, are the reference columns. Reference columns . So these are going to be the very first few columns in your truth-table. What you need to do is to pick up on the simple statement symbols, and arrange them in an *alphabetical order*, in the way they appear in the alphabet series. Remember you are talking about the constants of the variable and they are all alphabets. So you need to follow the alphabetical order in which they come. Let me take an example so that you understand. Here for example, suppose we have the same proposition B ● (A ∨ C). How many distinct simple propositions symbols? B, A, C- three.

So, the first three columns belong to these symbols and there are going to be arranged in alphabetical order, not in the way they appear, but in the alphabetical order. So the first column belongs to A, second one belongs to B, third one belongs to C. So, that's the first thing to understand; that you need this reference column to start with.

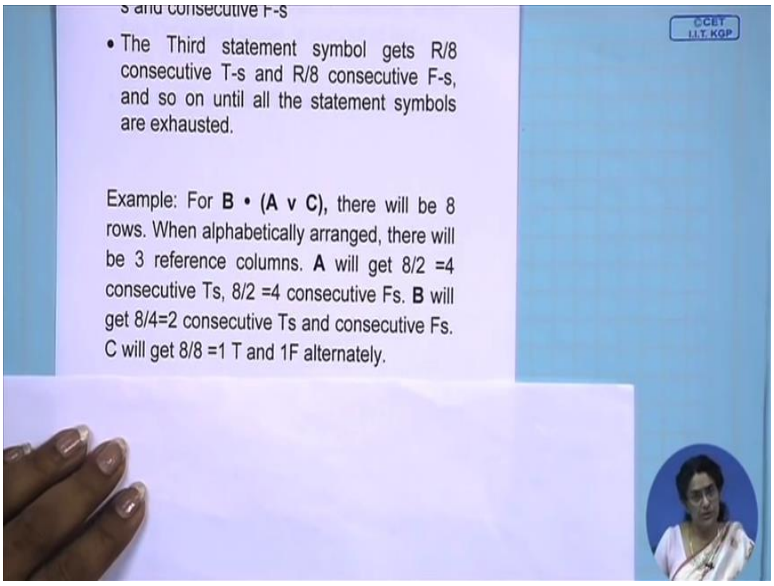
(Refer Slide Time: 12:50)



Once you have them, the reference column, then there is this question about *distribution of truth-values* to these simple statement symbols, and there is a certain way to do that. Once these symbols are alphabetically organized, then the first statement symbol gets R/2 rows of consecutive T-s and R/2 rows of consecutive F-s, where R stands for the number of rows. Remember you already know how many rows there is going to be. That is 2^n.

So then you know how many number of rows you have. But the first symbol gets half of that consecutive T-s and consecutive F-s. Alright? That's the first thing to know. The first statement symbol gets R/2 rows of T-s, R/2 rows of F-s. The second symbol gets R/4 consecutive T-s and R/4 consecutive F-s and the third symbol gets R/8 consecutive T-s and R/8 consecutive F-s, and so on and so forth until all the statement symbols are exhausted. This is how you go.

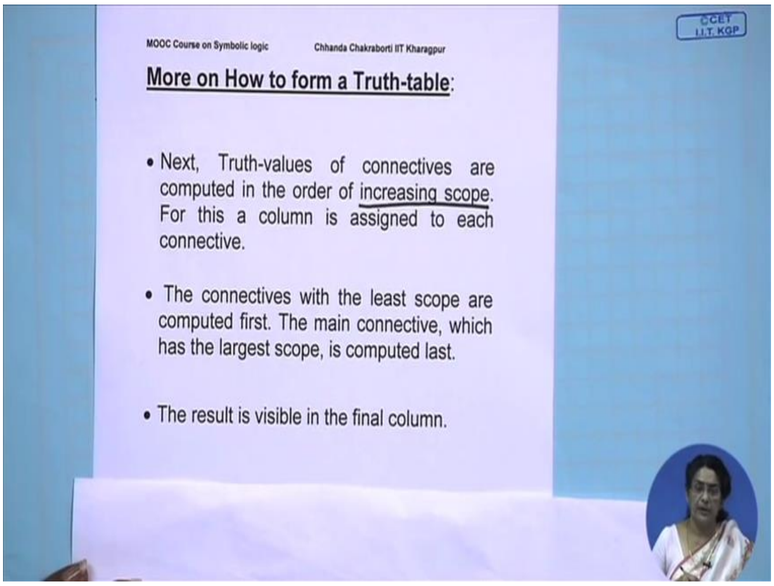
(Refer Slide Time: 14:00)



If that is already clear to you, wonderful, if not, we will take an example. So here again we go back to our example of B ● (A $\vee$ C). You know that there is going to be 8 rows here. So, the first symbol which is your A, right? That is the first statement symbol, it is going to get 8/2 consecutive T-s, 8/2 consecutive F-s. That is 4 T-s and 4 F-s. Second symbol is going to get 8/4. That is, R by 4, that is 2 consecutive T-s, 2 consecutive F-s. So, TT FF TT FF, until you exhaust the all 8 rows. And the last the third one will get 8/8, that is one T one F. So alternately TF, TF until you exhaust the 8 rows. We’ll show you examples. And then comes the remaining part.

So this is your how to construct the reference columns and then what columns do I have to do? The answer is then you slowly take up the connectives that are present, the components that you have inside. And these are to be computed following the truth tables that I have shown you of the connectives in *increasing scope*, increasing scope, and for each of these connectives you need to assign a column so that the connective with the least scope is done first, and the connective with the maximum scope is done at the very end.

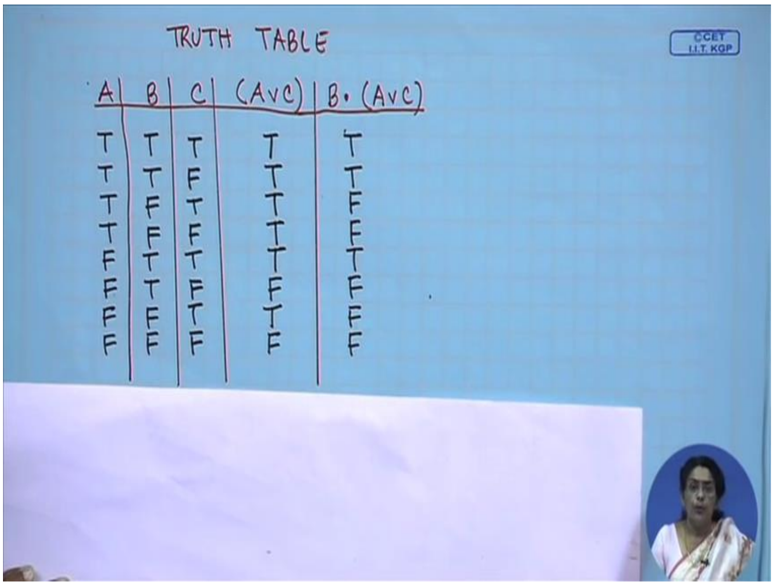
(Refer Slide Time: 15:09)



So automatically you know the main connective is going to be computed at the very end. That’s going to be your final column, and the result is to be visible in the final column itself. This is how we construct the truth table. So there are steps and pretty clear steps how to follow them. Well, let me see whether how far we understood this, whether we understood this, so on and so forth.

So we will try to construct truth table together and we will try to see if there are any questions that we can take, that we that may come up in the process. I will try to anticipate them. If not, let us see what to do about them. But here is for example, a sentence like this: the one that we already had B ● (A ∨C).

(Refer Slide Time: 16:23)



Remember, this is our whole compound statement and there are 1 2 3, simple propositions. So there is going to be 8 rows. What did I say? First, we need to organize, or arrange the statement symbols alphabetically; so A, B, C. Take a look what I did next. 8 rows, 1-2-3-4-5-6-7-8 , and look at the ways the truth-values are distributed. So, R/2, T-s consecutive and consecutive F-s. This is R/4 consecutive T-s and Fs and this is R/8, T and F consecutive. So this is the total truth-value distribution. Remember this is the exhaustive truth-value distribution. Can't miss it.

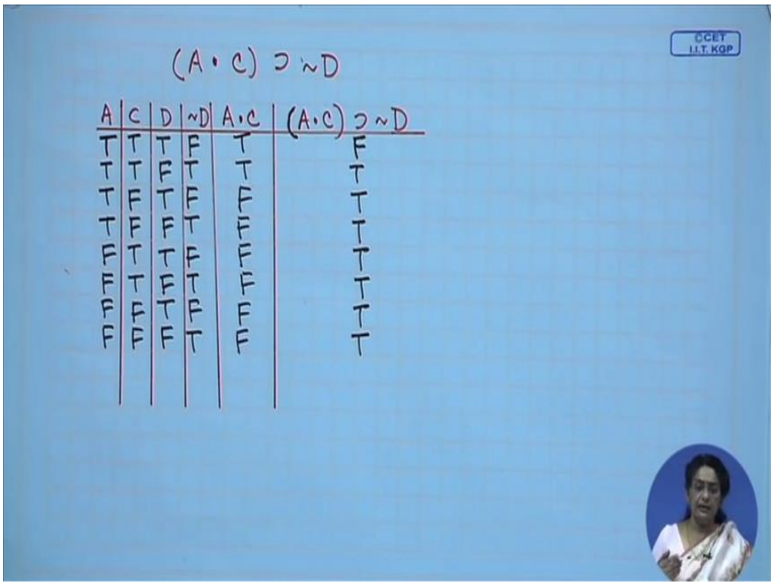
There are other ways to distribute truth-values. But like I said this is our way of distributing the truth-values and this exhausts all possibilities, all possible truth-values of this. Remember, each row represents to you a possibility, a possible situation. For example, the first row represents to you when each of this is true, each of this is holding up. The last row, for example, represents to you when none of these, neither of these, is holding up. Alright? Then you see the other columns. Remember this is our whole proposition. So in here we have a small component here A ∨ C. A separate column has to be assigned to that. How will we figure out the value of this? Well, by looking at this A and this C, and plugging in what we know about the vel (∨) or the disjunction. That's how this has been filled up. This on the other hand, is the whole proposition and which happens to be a conjunction. How are we going to figure this out? Well, we already have the A ∨ C column here and then we look into the B column and we do a conjunction of that. We already know the truth table of conjunction. This is how this column has to be filled out. So if you have done it correctly, this is how your table is going to look like. Remember the disjunction is true, when both of them are true, both disjuncts are true or one of them is true.

So you follow this through. We are comparing this column and this column, T ∨ T is T, T∨ F is also T; T, T; T, F , and but here comes also one of them is F one of them is T. So this is T. But look at this row, where A is F, C is also F, which is why the value returned A ∨ C is F. So this way we have filled out this column, then comes this conjunction and for this conjunction I said you need to compare this column with B. B happens to be the conjunct, one of the conjuncts.

So what is happening and let us remind ourselves how conjunction works. Conjunction is true only when both the conjuncts are true. So for example, here B is true; A∨ C is also true. The value returned under the dot is true. This is also the case where you have both the conjuncts as true. But here B is F, but A ∨ C is true. It doesn't matter which one of them is false. But if you have one false or both false in conjunction you are going to value, the returned value will be F. So this is how this column has been filled out.

I suggest again that you take this proposition and try to do the truth table by yourself because I have explained it and you have it here to compare. But this is way to learn. That you do it hands on. We will do another one and see where we stand, but I want you to sort of do it along with me. So here is the statement.

(Refer Slide Time: 20:52)



This is (A ● C) ⊃ ~D. We are going to do the truth table for this. So what is going to be the first thing to do? How many rows? 2^3, again we are going to have 8 rows. Then, we need the reference column. How many reference columns you need? 1, 2, 3 and in that alphabetical order. First one is A, second one is C; third one is D. Distribute the truth values and then will show the result to you. I will give you just few moments to fill up the reference column.

Check it whether this is how you have it. So 4 T-s, 4 F-s, 2 T-s, 2 F-s and then T F, T F, T F. And then we need to now build up the table, I said by *increasing scope*. Now if you go back to the sentence once more, which connective has the least scope here, the minimum scope? The answer is the *tilde D*. That's what it has. This is the connective. It has the least scope; why? Because, it ranges over only one proposition. This one goes over 2. This is the main connective. So this is going to be at the last, very last. First we are going to do the tilde (~) D. So the next column you might fix it like this, the tilde D column, which is going to be with reference to the D column. Then comes your (A ● C) and then comes the (A● C) ⊃ ~D. And this one as you know will have to be filled up by looking at these 2 columns, and by now we know how to do the conjunctions column.

This one on the other hand is your horseshoe, where am I going to look at? We are going to look at this (A● C) as the antecedent and tilde ~D. Please note this is your antecedent and this is your consequent, because in the case of horseshoe the positioning matters. So this is your first component and this is your second component. Take a look and try to fill up the column for this. I will give you just few moments to fill that up, and then we compare the results. Because we are beginners, so we are just learning how to do the truth table. Many of you may have done it in different ways, but all of you may not have done. So, this is our beginning, therefore we need a little bit of time to do that and this is why I said that you need to know about the connectives. Unless you know enough about the connectives how they work, it is very difficult to do the truth tables.

So there is still time that you can go back and look at the truth table of the connectives, but if you have the ready knowledge, then you can work this with me in real time. So (A ● C) the column looks like this. Ok? The conjunction is true only when both conjuncts are true, otherwise it's going to give you false, return value will be false. So this is where. Now we are comparing these two columns to see what to put under the horseshoe. Let us remind ourselves that the horseshoe is false only when antecedent is true, consequent is false. Ok? So only when if you find a row where A ● C is true, but ~D is false, that is when this horseshoe is going to be false; otherwise in every other case this is going to be true. If you have done it correctly, then this is how this column is going to look like. Ok? This is the only time A ● C is true; ~D is false, and this is going to have the F. Otherwise in every other case it is turned out to be true. Ok?

So this is our first lesson and I think, I have explained it step by step how to construct the truth table. This knowledge we are now going to apply to ask interesting questions. This is just a mechanical method. But it is important that we learn the method first, and that we practice it a little bit, so that when we want to apply to a specific problem, then the method does not become the obstacle. This was the point of doing this module together. I hope you have learnt how to do the, construct the truth table, because now we are going to pose interesting questions to this method. The method itself is mechanical, but we can pose interesting questions to it and we are going to in the next module onwards we are going to find out the answers to those. That's how far I will go with this module.

Thank you.