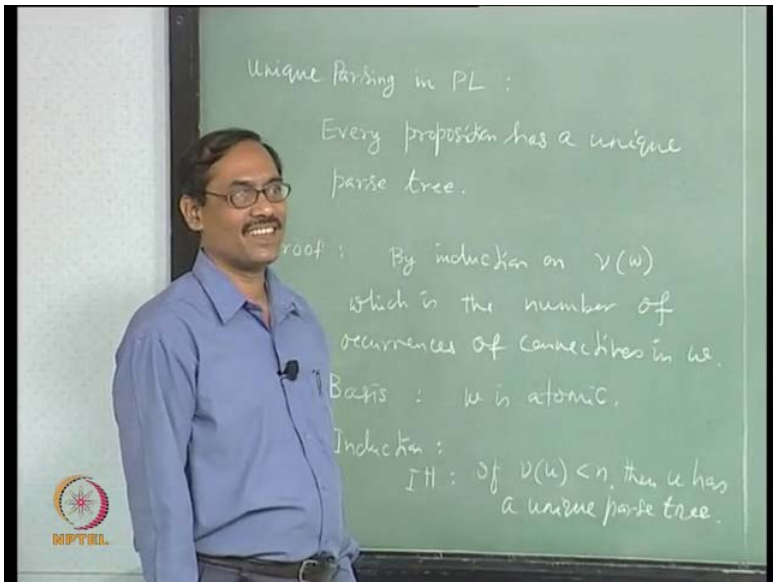
Mathematical Logic
Prof. Arindama Singh
Department of Mathematics
Indian Institute of Technology, Madras

Lecture - 3
Unique Parsing

(Refer Slide Time: 00:09)



You are writing PL for proportional language, but not yet it is a propositional language, it is not a logical language till now, it is only a formal language, the syntax part of it which is our PROP, PROP. How do we state it? Every proposition is uniquely parsed, every proposition has a unique parse tree. Now the question is how the proof should go. Usually the key is how the concepts are defined that will yield the proof method. How the things were defined there in PROP, they define recursively or you could have defined them iteratively or inductively, by introducing PROP0, then PROPi, PROPi+1, and so on.

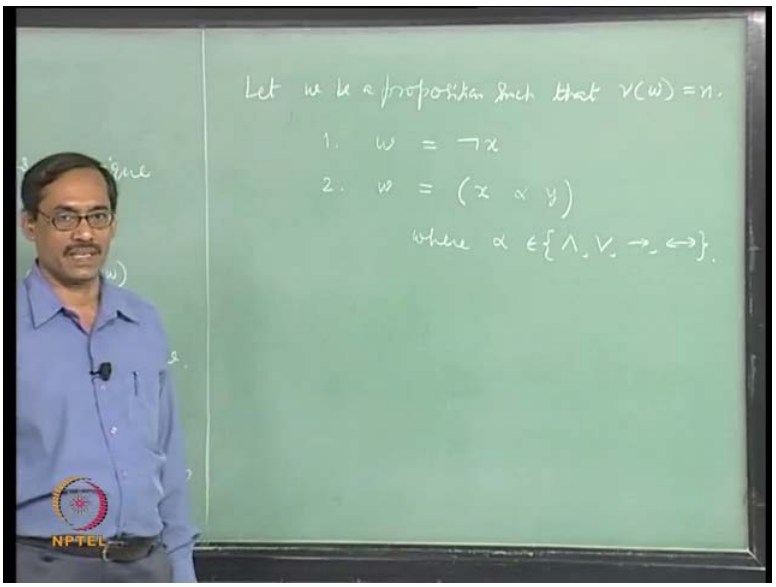
This suggests that we should go for an inductive proof basing on the length of the propositions or on something else. That is usually called the structural induction basing on the structure of what is formed you try to have the induction method. Here the structure means, number of occurrences of connectives, because each stage you might be introducing another connective, taking clue from the earlier generated propositions. Then introduce another connective, which is the main one for the current.

Let us have induction on the number of occurrences of connectives in a given proposition. We suggest that we prove it by induction on say, let us give a formula w, a number, nu of w, which is the number of occurrences of connectives in w, where w is any arbitrary proposition. The basis case is when, what is the basis case? nu of w equal to 0, there is no connective, but w is a proposition therefore, w is atomic.

So, w is atomic, is there a unique parse tree? Yes, that itself. There is the only route, it has no child and that becomes a tree; that is the parse tree. Only possible parsed tree there. Nothing more to do there, then let us take the induction step. The assumption is that if nu of w is say less than n for any arbitrary proposition w, then unique parsing holds. If you have to construct the parse tree, it will be unique and then we go for a new proposition or any proposition, which is having number of occurrences of connectives in it as n.

You are using the strong induction here. First we have to take the induction hypothesis. Induction hypothesis is if nu of u is less than n, then u has a unique parse tree. To make it specific you also should have written u is a proposition, unless it is a proposition all these things becomes empty.

(Refer Slide Time: 04:59)



Then we start with, let w be a proposition such that number of occurrences of connectives in w is n.

Now you must show that, there is a unique parse tree for w, probably basing on your induction hypothesis. Now if w has number of occurrence of connectives as n, which is not 0 of course, that is covered in the basis step. Then how could it look? Well there is at least one connective in w. How does it look like?

Student: w1 connective w 2 connective

w1

Student: some connective

w1 some connective w2, but you are missing the best part of it.

Student: With parenthesis

With parenthesis or there is another form which is

Student: Negation of it

Negation of some other proposition; w may be, w is equal to not of x or w is equal to a parenthesis, x, some connective, y and a parenthesis, where alpha is some connective, it is a binary connective now. Let us take the cases individually. In case one, how do you show that it has a unique parse tree? First look at your parameter, your parameter is nu of w which is n, what about nu of x,

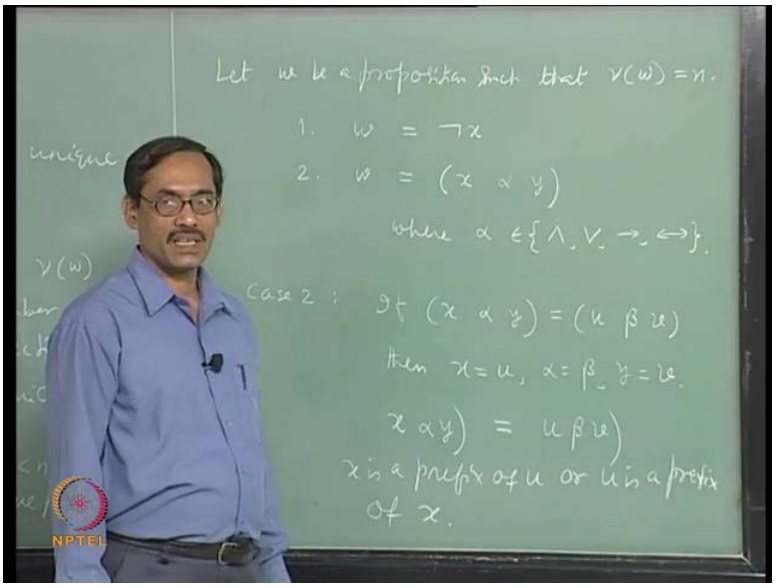
Student: (Refer time: 07:08) n-1.

Less than n. Now you can use the induction hypothesis, that x has a unique parse tree, will not x have a unique parse tree?

Student: Yes

Yes, because if not x equal to not y then x has to be equal to y. Recollect the problem in parsing.

(Refer Slide Time: 07:36)



The problem in parsing was this, you add some proposition of this form let us say, now you could not parse it properly, it give rise to two types of parsing. One can be that, first one is this and the next one is this, another could be, first one is this next one is this. That means if x alpha y is equal to u alpha or beta v, we cannot say that x is equal to u that was the problem in parsing.

If that is always equal, then automatically unique parsing will hold, the problem is clear. But here what happens if not x equal to not y, then x has to be equal to y as strings. There has to be unique parse tree, there is no other way of parsing it, you have parsed through the connectives; not x becomes the father and x becomes its son or it is a parent node and the child node is x, parsed tree goes. We are not going to discuss it once more, this case might be difficult to do because of that reason.

Let us see the case 2. Now our first responsibility to show that if w is in this form, then x is unique, alpha is unique, y is unique. Which means if x alpha y is equal to u beta v then x must be equal to u, alpha must be equal to beta, y must be equal to v. One step of parsing will be alright. There is no way of parsing or starting the parsing any other way, this is a unique way in the first step. Then, our plan is to use the unique parsing for x and y they will give two different trees, they will be joined.

Unique parsing will be proved, this is the crucial step to be seen. Now how do we proceed?

Student: x and y are less than n

We cannot approach x and y directly because, we have the parenthesis before it, it is a block. Just compare the strings, comparing the strings it starts with the first symbol as left parenthesis, this also starts with the symbol as left parenthesis. You can cancel them you get the other things as equals, as strings not as propositions, we are concerned about strings now. Deleting these left parenthesis from the assumption, you get x alpha y is equal to u beta v.

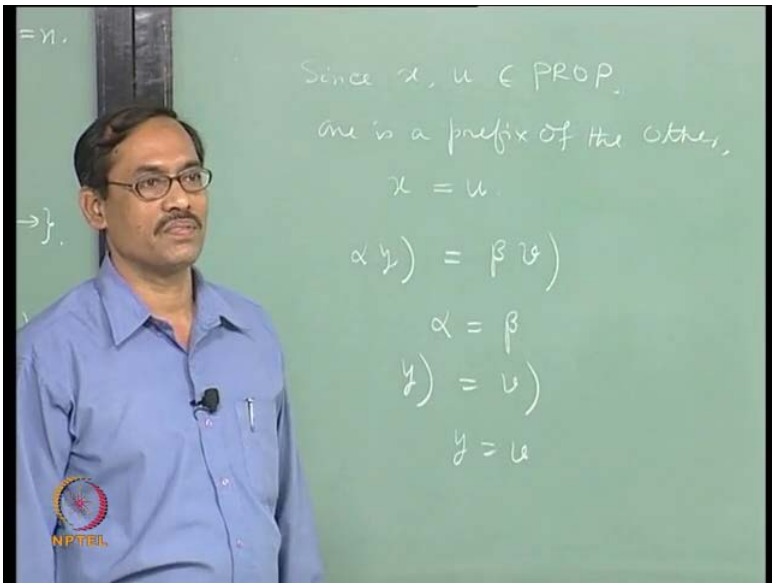
We are becoming very formal, no heuristics now, then what happens? As strings you can match their symbols, problem is we do not know their length. If length of x is equal to length of u, they will become same, because they become the prefix of the same string. But then it reminds you something, once you think of prefix; since w is a proposition and this is expressed in this form with a binary connective alpha, x and y are also propositions. Now look at these two strings, x is a prefix of the string, u is also a prefix of the string and u and v are also propositions. Now x is a prefix u is a prefix, what could be x and u, prefix lemma.

Student: One has to be the prefix of the other.

One has to be the prefix of the other. From this you conclude that x is a prefix of u or u is a prefix of x; is this step clear? We are comparing them as strings, then we find that x is a prefix, u is a prefix of the same the string therefore, one has to be a prefix of the other. Now then, we have proved some property, Property 2? It says that if there are two propositions, one is a prefix of the other then the propositions have to be same, they cannot be different. No proper prefix of a proposition can become a proposition, that was our Property 2. This says x must be equal to u.

This means since x and u are propositions, one is a prefix of the other, x must be equal to u. Now, once x is equal to u, take them away, then what remains is alpha y right parenthesis beta v right parenthesis, then as beginning of the strings, first symbols of the strings alpha must be equal to beta.

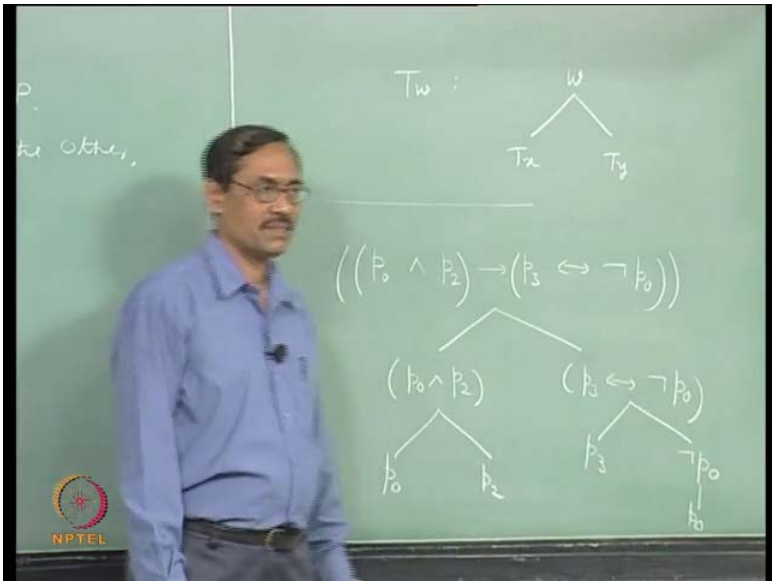
(Refer Slide Time: 13:21)



Then what remains is y right parenthesis and v right parenthesis; cancel the last symbols, you get y equal to v. This proposition what you wrote here is proved; once these two are same x must be u, alpha must be beta, u must be v.

Now you look at w, w is x alpha y so? x, y, alpha are uniquely determined. Now nu of x, nu of y are less than n they have parse trees, if the trees are Tx and Ty then the unique parsed tree for w is Tw.

(Refer Slide Time: 15:21)



We would get Tw, as w then you have Tx, you have Ty that is the unique parse tree. There is no other way of parsing it. This consists of the syntax of propositional logic, of PROP. We know that no proposition is ambiguous. And why you are worried about your ambiguity? If it is ambiguous, where is the problem?

Student: Multiple meanings.

Multiple meanings may become assigned. It is because we want to give the meaning compositionally, starting from the smaller ones you want to give the meanings to the bigger ones. We need that the construction of a proposition should be unambiguous. What happens in a parse tree is, if you want to give meaning; let us say you want to assign values to a function on a proposition, which constituents as the atomic propositions.

Suppose I have a proposition of this form say p0. Now we have a parse tree, which looks this way. Now, I assign some value to p0, I assign some value to p2, same way to p3, I do not know what do they mean, some values. Now then, I say how this and, the connective `and' operates with these two values, I define only for `and', then you see that it is defining uniquely a value for p0 and p2, once these rules are fixed. The same way if you go along the parse tree, from the leafs to the root, you will see that, there is a unique value for the proposition and this is coming because there is unique parsing.

If it is not uniquely parsed, there is another parse tree, then for the same assignment of values to p0, p2 and p3, you might end at another value for the root proposition. That is the reason we have to do parsing first and see that it is unambiguous so that, we can give, give meaning, or assigne values to the atomic propositions; and then evaluate the proposition accordingly in a unique manner. We will come to it shortly, that will form our semantics or giving meaning to the propositions. Till now whatever we have done is the syntax of PROP, only grammatically we are defining it, as strings; what do they do?

Now, there are some other ways of looking at unique parsing and its propositions. For example, you can find out what is the main connective in a proposition because, it is uniquely parsed, there is a particular alpha you have seen here, there is a particular connective as can be found out, the way it has been formed. That corresponds to the main connective of the proposition. Main connectives can be found out, but how do define a main connective?

Student: By the definition.

Which definition?

Student: w is equal to parenthesis or w equal to negations.

Right.

Student: So in the second case alpha is the main connective

Alpha is the main connective, in the first case?

Student: Negation.

Negation is the main connective, that in the first case you say negation is the main connective, in the second case you say that binary connective alpha is the main connective. There is a third case which is trivial, there is no connective, there is no main connective. Now, once you know the main connective, you can use it to determine whether some expression is a proposition or not, can you do that? For example, let us take this proposition.

We say it is a proposition because it has a parse tree which ends in atomic propositions, the leaves are the atomic propositions. Now if it is not a proposition, you may not be able to do it, can you see that? For example, to find the main connective you should have a way, otherwise, you cannot parse it, start the parse tree, to form it is easier. Because, it starts with the leaves and then proceeds in a bottom to top way, but when giving an expression or a string, the problem is different, you have to start the parsing. How do you parse, unless it is in proposition at all? your rules do not permit.

It might fail but failure means what, how do you determine that it is a failure? Because for failure we have not given any rule, for success only we have rules here. That happens in life also, here also its happening; how to proceed then? There is a clue, our Property 1 says, if it is a proposition the number of parenthesis will be matching, that is left parenthesis must be equal to the number of right parenthesis. Then what you do with, if it is starting with a negation symbol, just delete the negation symbol because, the rest should be a proposition. If it is starting with the left parenthesis, then delete that left parenthesis, there must be one right parenthesis at the end, delete that also.

Then, start matching the number of parenthesis. Wherever it is matched that is your first proposition. Next one should be a connective, next one should be second proposition because of the second property, that, once it is matching, it has to be a proposition provided the original string is a proposition.

If it is a proposition, then no sub-string of it, starting from the beginning, that is no prefix will be a proposition again. Once the matching of parenthesis done it should end only at that, the whole of it will be covered with the matching of the parenthesis, that is how the language is styled. Do you see the problem and its solution? Parsing will be easier now, it is not difficult. Start from the given expression, look for the not symbols, if there is no not then there can be propositional variables only, it is only a propositional variable or it might start with a left parenthesis; first case is clear.

If it starts with the left parenthesis, delete that, find out the right one, right most, there should be one, delete that then start matching the parenthesis from the rest of it, from the left, wherever matching occurs take that as the first one x. Next one should be one connective, binary connective, which was your alpha, next one whatever given should be y continue whenever it fails, it is not a proposition. If it succeeds means, if it ends at only leaves which are atomic, then it is a proposition. You have another alternate algorithm for determining the propositions.

This is the second one, third thing is you can define something more from the parsed trees like, your sub-propositions or immediate sub-propositions. Once you know what is the main connective, then the left side of it that string is your x, that is your immediate sub-proposition of w. And the right one which is y, that is again another immediate sub-proposition of w, if it is a binary connective, there will be two immediate sub-propositions. If it is unary connective, that is a negation symbol, there is only one then. In the parse tree whichever things occur, all those propositions that occur they are the sub- propositions; all of them are sub-propositions; immediate sub propositions are the children of the roots.

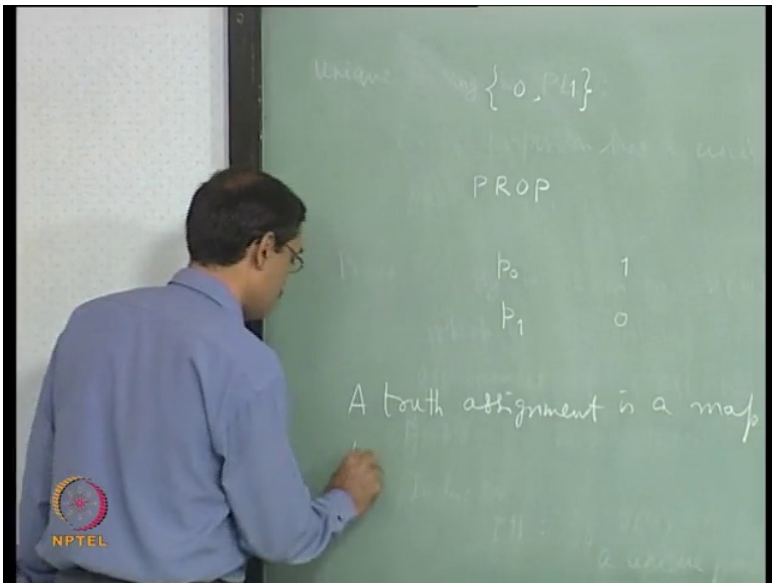
But these things can also be defined alternatively, is it? You take any proposition w, then you take any substring of that, if it is a proposition it is a sub-proposition of w, that is the declarative version of it, this is the algorithmic version. With these, we close our chapter

on syntax, we will go to the semantics now, try to see how the functions can be evaluated starting from the atomic propositions.

Semantics concerns the meaning of something, now we want to connect our thought with reality. Reality only gives us the meaning, or meaning here is, by reference? In usual ordinary language our meaning is by reference. But if you have some abstract concepts you do not have any reference, we still accept that we know the meaning of them. But anyway, it starts from the concrete objects, which are taken as the meaning of that what is. For example, when you say chalk, c h a l k, you just show it, but this is a chalk, now you know what is the meaning of a chalk. You want to say tiger then you take the child to the zoo or give a rough idea by drawing, or something.

If you only draw and the child is very sharp, she will always think tiger as that which is drawn on the paper, not in the zoo. There is a problem in sharpness here; anyway, our meaning is a very simpl compared to this ordinary language usage. This logic will be very simple for you, because the meaning is simple.

(Refer Slide Time: 27:04)

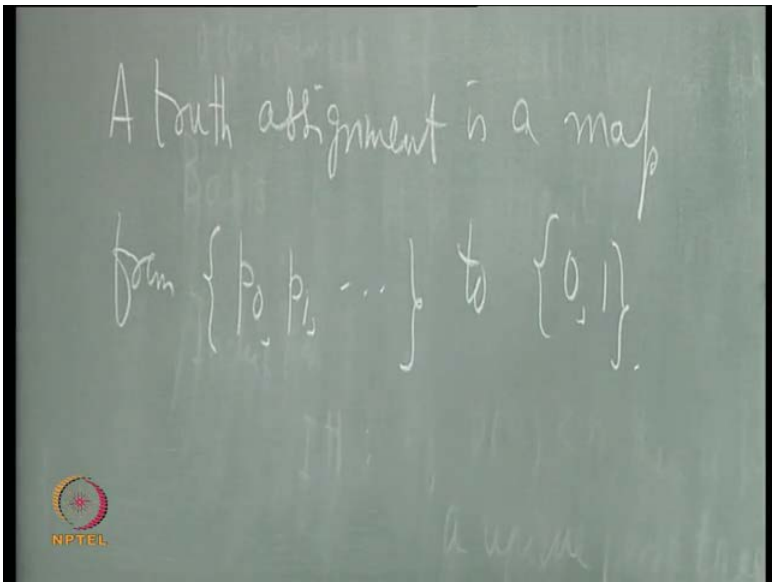


Our world only consists of two elements not chalks and tigers or chairs, only two elements are there, that is easier to construct because of that. Now what we do, when we have PROP, the set of all propositions, we want to give meaning to each element here, by connecting to these two elements in our world 0 or 1.

That means suppose I take a propositional variable here, I connect this p0 to 1, again connect my p1 to 0, some arbitrary way, I am giving. Now give meaning this way, after you give meaning you have to give meaning to the set or to all propositions in the set PROP,, not only to these atomics ones. You have to give a rule through its formation. How do you take care of the connectives? Such an assignment which gives meaning in our world 0, 1 is called a truth assignment, instead of true and false we are writing 0 and 1, it may not be in that order, we do not know which one is true and which one is false till now. We have just two elements 0 and 1.

Now, you want to assign them to the propositional variables and then the atomic propositions and then, the set of all propositions, you have to slowly extend it. You start with any such mapping or a truth assignment, from the set of propositional variables to 0 and 1.

(Refer Slide Time: 29:13)



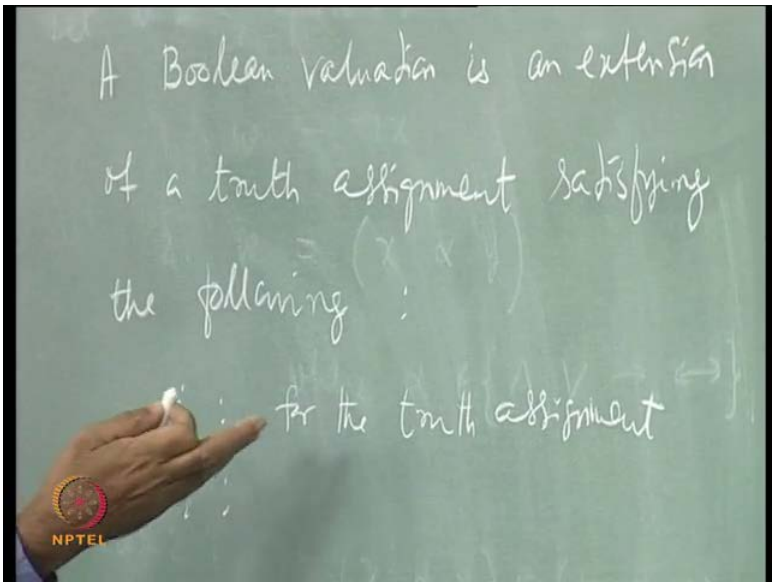
A truth assignment is a map from the set of atomic variables to 0, 1.

Student: What about top and bottom?

We will assign them, we are starting with the propositional variables, next will go to possibly atomic propositions and the set of all propositions. Now that extension will be compositionally defined, by defining the connectives; how do they work, that is how we

are going to do. Starting from any such truth assignment, whatever you get through that is called a Boolean valuation.

(Refer Slide Time: 29:54)



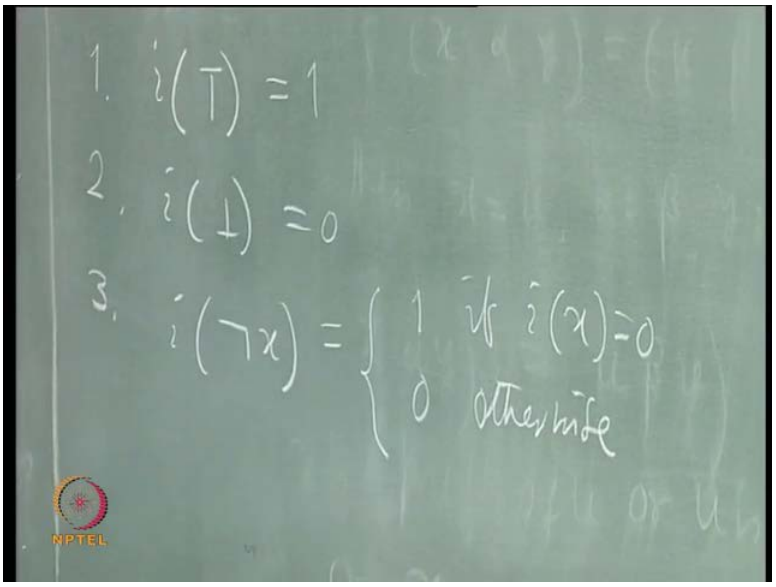
We will say a Boolean valuation is an extension of a truth assignment satisfying the following properties.

Now I want to answer your question what to do for top and bottom? Now suppose I write i for the truth assignment, I will also write i for the same extension, its extension because, anyhow it is same for the base ones p0, p1 and so on.

(Refer Slide Time: 31:03)

Also we will write it for the Boolean valuation. What are the properties that satisfy? i of top is fixed to be 1, whatever be the truth assignment, it always fixes top to 1. Now see, there is justification in it for our giving the name to top and bottom, they are propositional constants. These are propositional constants; they always have the same meaning whatever context they appear, does not matter, whatever truth assignment it is, top is always given the value 1.

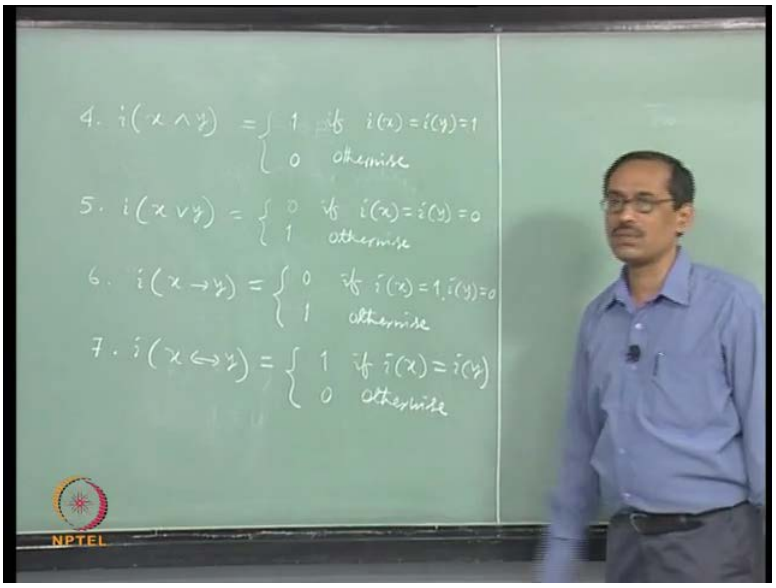
(Refer Slide Time: 31:45)



Similarly, bottom is always given the value 0. If you think of this one as true, then top mean something which is always true and bottom means which is always false. Then third, you have to go for the connectives slowly because, p0, p1, …, top, bottom, for all the atomic propositions, we have fixed till now, you go for the connectives. Now you say i of not x equal to, 1 if i of x is 0, and it is 0, otherwise.

This is a structural definition because the propositions themselves were defined structurally, recursively or inductively. We are doing a similar one here, once you know what is i of x, you can only determine i of not x, x might be very complex, some other connectives might be there and so on. They will be defined later.

(Refer Slide Time: 33:11)



Next i of x and y equal to 0, if well, we can take 1, that will be here, if i of x equal to i of y equal to 1, and 0 otherwise. That complies with our usual meaning of and, a statement join with `and' becomes true when both of them are true, interpreting 1 as true here. Next we write i of x or y which is 0, if i of x equal to i of y equal to 0, and it is 1 otherwise. You must see intuitively what does that mean, it becomes false only when both its components are false, otherwise it is true. `or' is true when at least one of them is true, both also included.

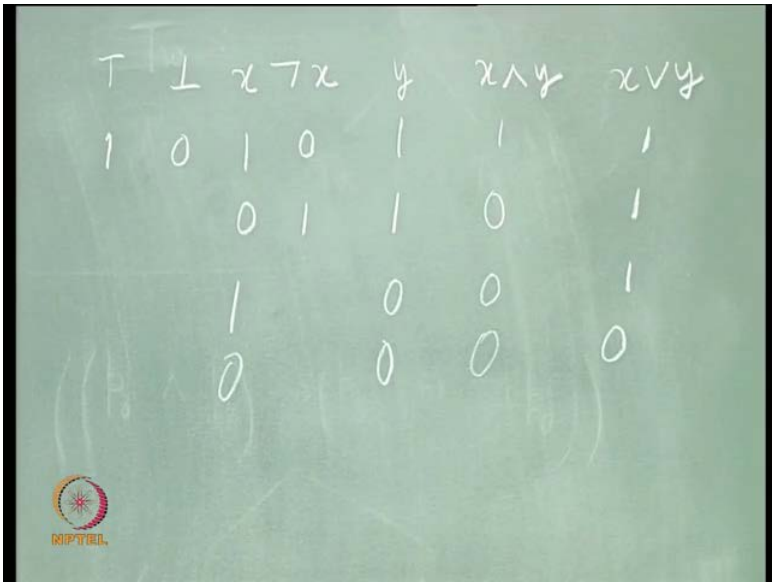
Then x implies y. This is 0 if i of x is 1, i of y is 0, in other words, it is false. This says that the implication is false when its assumption part, or the antecedent, is true and the consequence is false. In all other cases it is taken to be true. This is slightly problematic; but this is the sense we follow in mathematics.

In natural language we may not be able to follow this always. For example, you go back to your school days when you learnt about Buddha, he was telling or supposed to be telling that if you have desires, then you will get pain. Therefore, if you have no desires you get no pain. So is an if then statement or imply statement; you say that if you have desire then you have pain. Suppose you interpret this way, from this you can never conclude that if you do not have desire, you do not have pain. All that you can say if you do not have pain, then you do not have desire; that you can follow?

This not really this implies; it is different. But we can give examples for this implies also in a natural language, it is not difficult. For example, your brother says that “if you get ten points or an S grade in logic course, then I will give you one blackberry or i-phone 4”. Now what happens, you have not made it; at the end it, found that you have not got S grade, still he gave you a blackberry, very nice situation. But then let us see whether you are happy or unhappy, did he contradict himself or not? He din't contradict. But the case when you have really got S grade and he did not give you a blackberry, you are really unhappy, he has not kept his word. That is the sense of implication we are taking here. It can be true even if the antecedent is false and the consequent is true. Even if both are false it is also allowed, that the whole thing will be true, these are all the cases here.

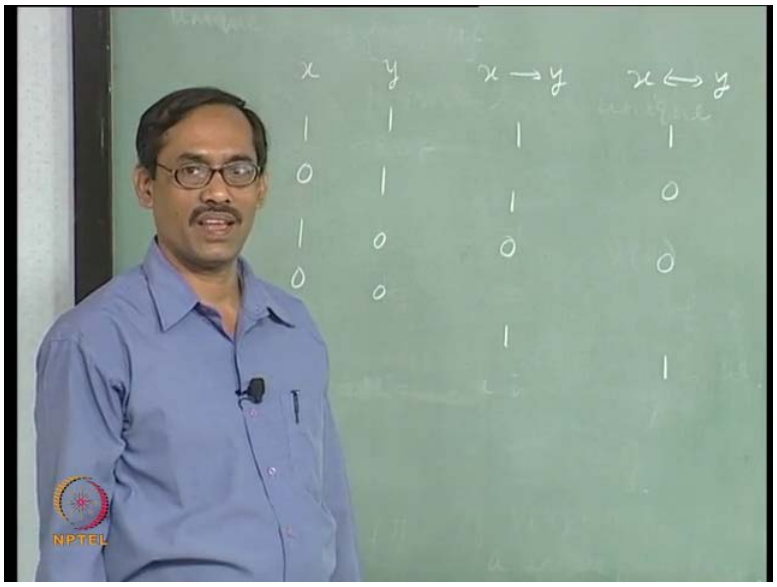
Then, let us see the biconditional; this is 1 if both are same, otherwise it is 0. It is easy to see this, the cause of another fact then you usually take this one, that is why Buddha may be correct, he is thinking of desire as a cause of pen. Once a cause happened the effect has to happen and the once the effect has happened, the cause must have happened, that is how these cause is related; causal connection. But it is a difficult topic, there is much philosophy behind causal connection on how it is exactly formulated in logic. Buddha’s idea was this, he was using if and only if, not if then. Now all these things can be shown in a truth table in an easier way. How do you show it, it is usual.

(Refer Slide Time: 38:58)



You have top, you have bottom, they always take the values 1 and 0, they are the constants. Then you have not x and you have x, if you have x which is 1 or 0, not x will be changing the values. Similarly, you have y also another then you can think of x and y. Now only in this case it will be 1, all the other cases it will be 0, x or y is 0, only when both of them are 0, all other cases it is 1. Other two you can find out; just complete them.

(Refer Slide Time: 40:28)



Look at the table for implies. It says that, x implies y is true when its antecedent is 0, these two cases I am taking now. It is true when its antecedent is 0 or it is false, if x is false then x implies y is true and if y is true, then also x implies y is true, one case is repeated in both the things.

But that is how it can be covered, whenever x is false x implies y is true, whenever y is true x implies y is true and these are all the cases. The remaining case is when x is true y is false, x implies y is false, sometimes we describe x implies y using this three conditions. Instead of all the four conditions, we say it is true when one of this things happens: x is false or y is true. That gives some kind of equivalence between the x implies y and x is true x is false y is true. When x is false that means not x is true or y is true, that gives x implies y is true. Probably x implies y is equivalent to not x or y do you see the connection? How do you define equivalence?

Student: All the four should be.

Yeah

Student: Truth table should be same.

Truth table should be identical, now in terms of Boolean value and.

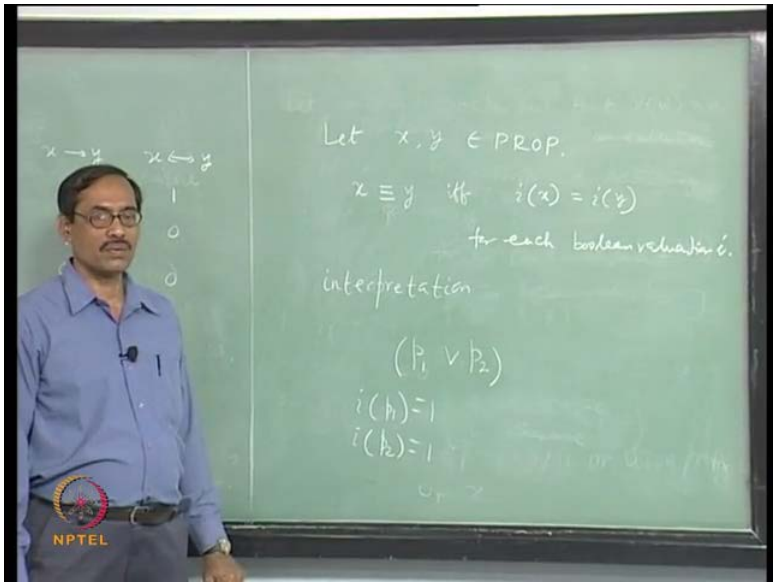
Student: (Refer time: 43:04)

Suppose you have to define x is equivalent to y.

Student: i of x is equivalent to y.

i of x equal to i of y, for which i is? all Boolean values and y, all Boolean values and y then they will be equivalent, x and y will be equivalent. In that sense x implies y, not x or y, are equivalent.

(Refer Slide Time: 43:39)



You may say that, let x and y be propositions, we say x is equivalent to y, if, what happens, i of x equal to i of y, for each Boolean valuation i. Now Boolean valuations are also called interpretations, you are interpreting the syntactic entities through truth and falsity.

We also call them as interpretations, but interpretation has also a restricted meaning, in the sense that suppose you take the proposition p1 or p2. Now here, there is one Boolean valuation, which assigns p1 to 1, p2 to 1, let us say i of p1 is equal to 1, i of p2 equal to

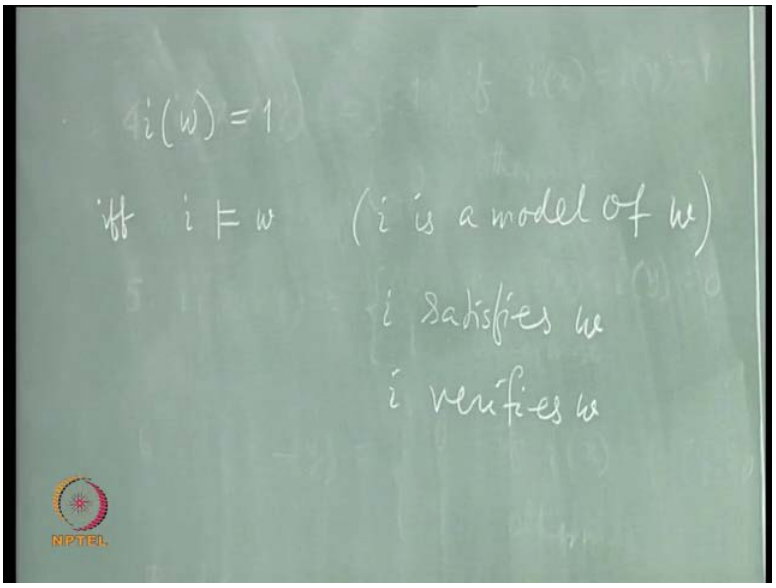
1, but is it a Boolean valuation by definition? Yes? Is it a Boolean valuation by definition? is it a truth assignment? it is not a truth assignment by definition. Because a truth assignment has to assign values to all the propositional variables, it is defining only on p1 and p2, what about the rest of them? There are infinitely many: p0, p3, p4 and so on.

In the strict sense it is not a truth assignment till now, unless you define for others: p0, p3 and so on. Everything should be defined, then only i will become a function from the set of atomic propositions or propositional variables to 0, 1. Till now it is not, it is only a partial function. When you say of interpretation, let us not worry about what other things are, it takes care only of these two. That is a restricted sense of the, what? interpretation, it is a Boolean valuation, but does not bother about all the other propositions, which are all the other propositional variables, which are absent in the context. But then, we say that they are same, the reason is it does not really matter.

Suppose you have a proposition, you have some propositional variables there and there is a propositional variable which is not occurring there, it should really does not matter. And we have to be careful about the meaning, what does it mean by “it does not really matter”, we will come to it shortly. Now, let us take about a Boolean valuation i and how it differs by assigning a 1 or a assigning a 0, you have to name them first. Suppose you have a proposition w and i is a Boolean valuation, you say i of w equal to 1. In that case you say, i is a model of w what will be “the models” mean?

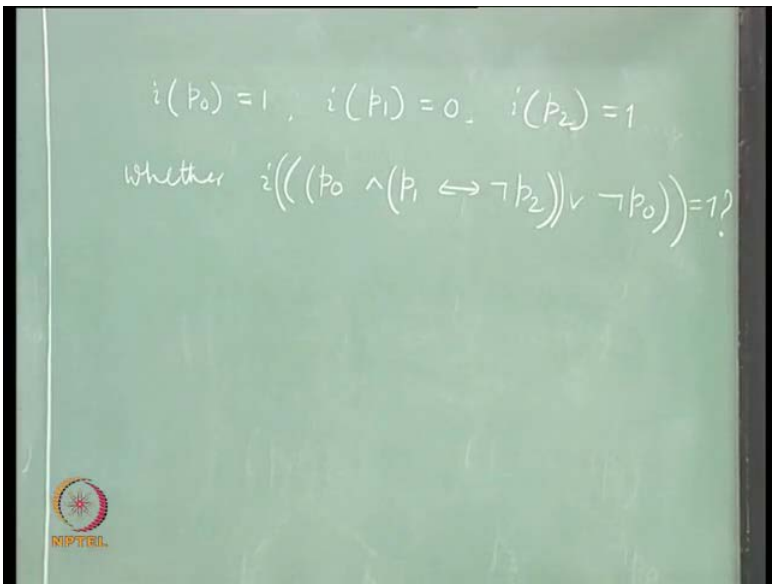
The word model, you have a very big building, you have a model for it. It is replica of that you say miniature or something. Now the thing is, we want to visualize it a world for it, will be true, that is the meaning of a model here. Now i is such a world or truth assignment, which makes the proposition true. We say i is a model of w, that is the terminology followed. Let us go slowly in increasing our vocabulary.

(Refer Slide Time: 48:03)



We will say that i of w equal to 1, is also written as i is a model of w. This is the symbol we are using. Sometimes read it as i satisfies w or i verifies w, all these are used. Now given any proposition, given any truth assignment by following the truth tables or the definition of the interpretation of connectives, you can find out whether that i is a model of proposition w or not. Yes? You can find out? Let us try one, see how does it go.

(Refer Slide Time: 49:15)



Suppose, I want to say whether such an i, which gives p0 1, p1 0, p2 1, such an i whether, i of it is one. Now how do we proceed? We can parse it, take the clue from the

parse tree, which is same thing as how it is formed. You can start giving values and try, it can be done. All that you have to do is, i of p0 you assign 1, i of p1 you assign 0, i of p2 you assign 1, then proceed in the truth table. It can be done. Problem was deciding about the interpretation, if i of p3 is also 1, further its truth value will be changing, in that case, considering i of p3 equal to 0. Well, we will leave it for the next class.

What have we done till now? Unique parsing is done, then unique parsing has been used to define the semantics, giving interpretation to the propositions. Compositionality could be used, starting from any truth assignment for the propositional variables, we now see that there is a scheme to go for defining truth assignments, which are called a Boolean valuations for the set of all propositions.