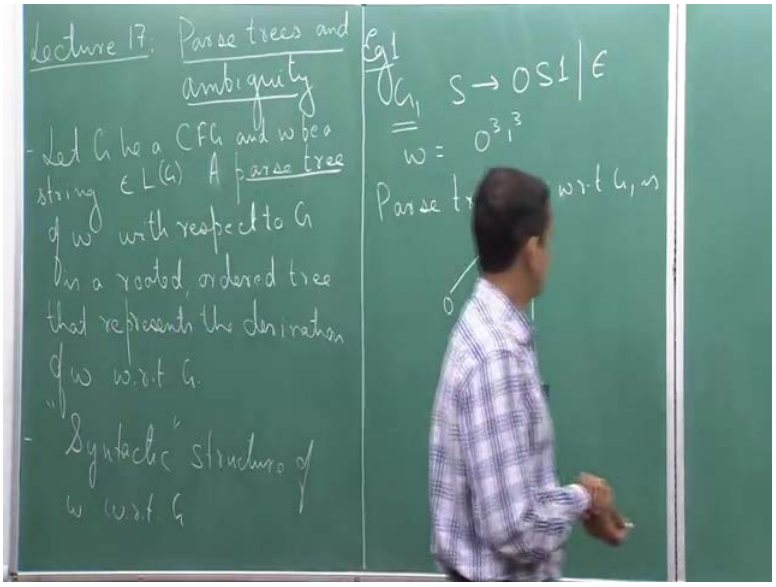
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute Of Technology, Kanpur

Lecture – 17
Parse tree, Derivation, Ambiguity

Welcome to the 17th lecture of this course. Today, we are going to talk about parse trees and ambiguity with respect to context free grammars. First let me talk about parse trees; let me define what parse trees are and see with some examples.

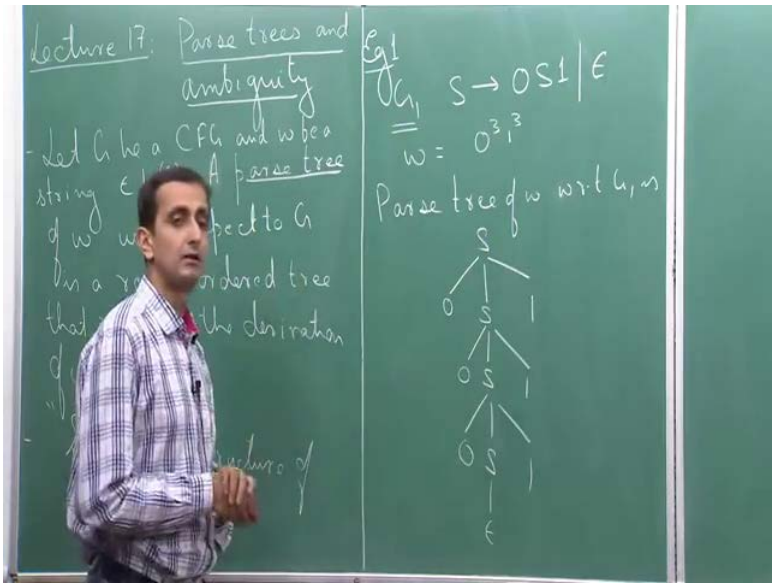
(Refer Slide Time: 00:49)



So, for the context of this lecture let G be a grammar - a context free grammar, and w be a string in the language of the grammar let say. So, a parse tree of w with respect to G is a rooted ordered tree that represents the derivation of w with respect to G. So, I have not formally defined what we mean by derivation, and I am going to do that very soon, but basically when we have a grammar and when we derive a string from that grammar, it happens via a sequence of substitutions.

So, we substitute a variable with one of its production rules and that is how we derive a string. So, a parse tree gives a visual representation or it is a visual representation of how this derivation takes place. It is a rooted ordered tree and it gives what is called a syntactic structure of w with respect to G. So, we will see very soon.

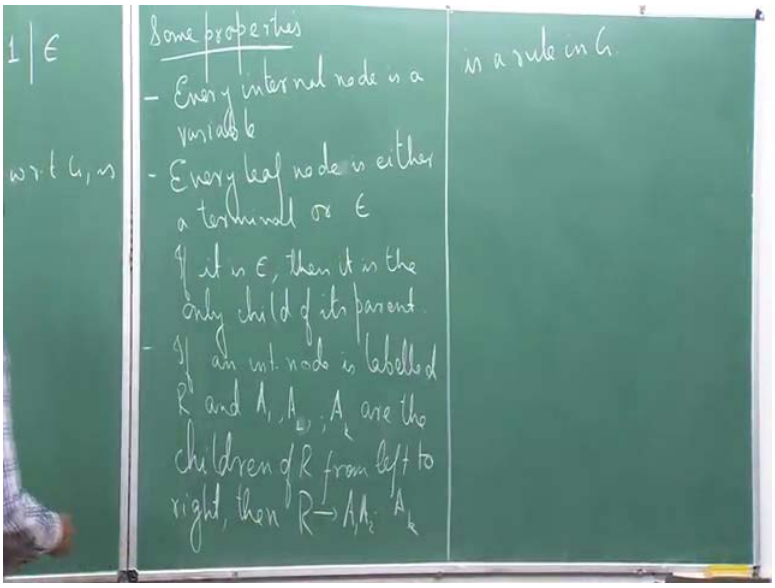
(Refer Slide Time: 03:24)



Let us look at some examples. We will describe two grammars. So, the first one, so the first example is so let say we have a grammar G 1 which is basically the grammar for the language 0 to the power n, 1 to the power n. G 1 is described as follows; so G 1 has the rules S going to 0 S 1 or epsilon. And let w be the string 0 cube 1 cube in the language of G 1. So, the parse tree of w with respect to G 1 is the following tree. So, we have a node S first, and then we have so basically the root node represents the start variable and each level would represent what a variable in the layer just above it gets replaced with the production rule that is used to replace the variable just above it.

So, S gets replaced with 0 S 1 first, then this S gets replaced again with 0 S 1, again this gets replaced with 0 S 1. Now, we have a string consisting of three 0s and three 1s. So, this last S gets replaced with epsilon. So, if I basically look at the leaves from left to right if I concatenate the leaves of a parse tree from left to right, it will give me the string w. So, this is the parse tree for our string w.

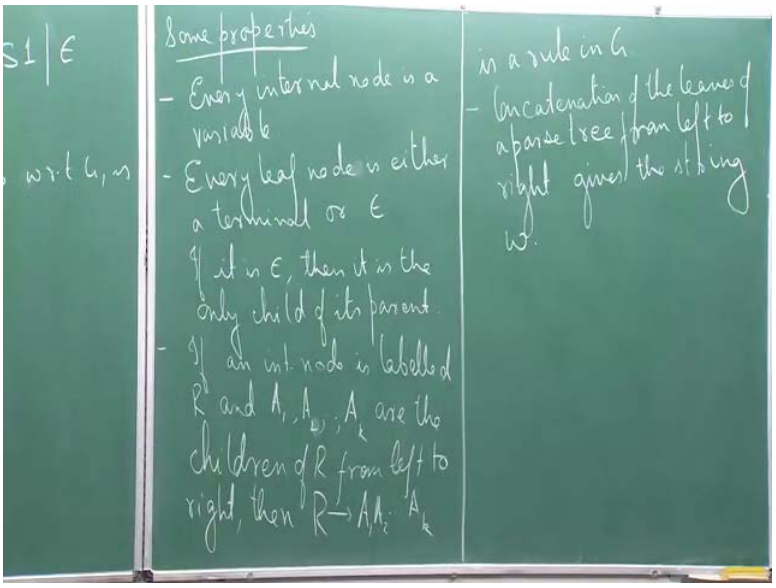
(Refer Slide Time: 05:37)



So, a couple of properties about parse trees. So, firstly, every internal node is a variable, so internal or non-leaf nodes, so nodes which are not leaf or in other words nodes which have some child.

For example, we have this as an internal node this, this, and this all of contain variables. Every leaf node is either a terminal or it is the empty string epsilon. If it is epsilon then it is the only child of its parent, for example, here we have leaf nodes, which are symbols like 0 or 1, and there are actually other child also for the parent of any leaf node. But if you look at epsilon the parent of epsilon which is this node has only one child, and then when we have if an internal node is labeled R and A 1 to A 2 up to A k are the children of R from left to right, then R going to A 1, A 2 up to A k is a rule in the grammar.

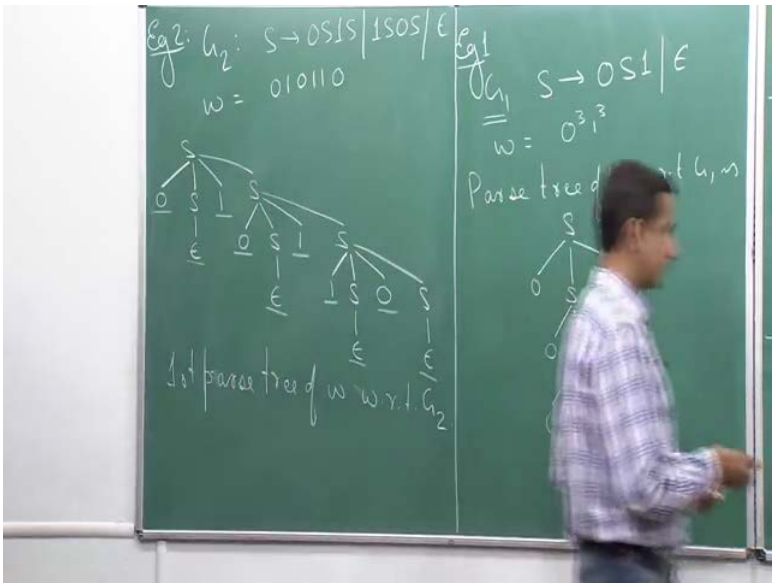
(Refer Slide Time: 08:48)



For example, if I have an internal node, let us look at this internal node. So, we have some variable on it, S in this case. And if we look at the children of that internal node from left to right 0 S 1, then S going to 0 S 1 is a rule in the grammar which is what it is. And finally, concatenation of the leaves of a parse tree from left to right gives the string w. So, if I concatenate the leaves 0 0 0 epsilon 1 1 1 from left to right, it gives the string w, which is 0 to the power 3, 1 to the power 3.

So, one important point about this parse tree is that observe that if we take any string w that belongs to the language of G 1 0 to the power k 1 to the power k for any k, observe that you were only get one parse tree for that string. So, there will only be one way in which 0 to the power k 1 to the power k can be represented in a parse tree format, but this is not always the case.

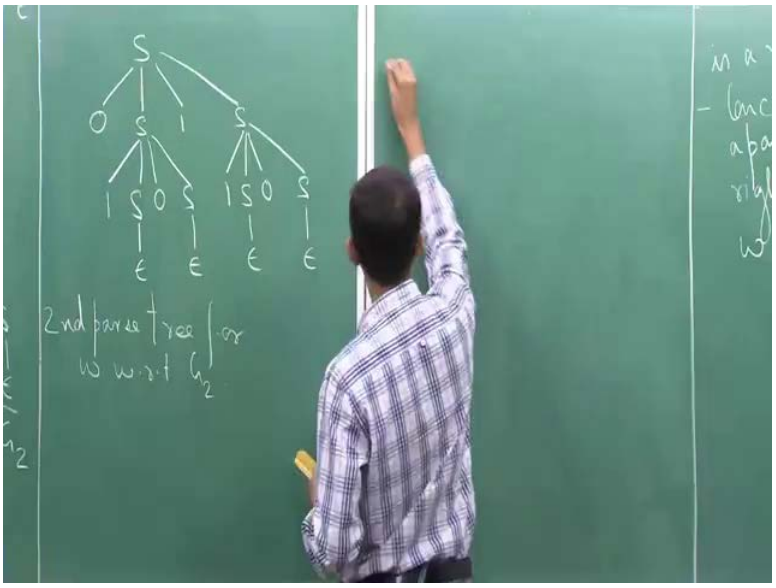
(Refer Slide Time: 10:52)



Let us look at another example where this property is not true. Let us look at the grammar for the language of all strings over 0 1 that have an equal number of 0s and an equal number of 1s. So, last time, we saw this example and we constructed a grammar for this language. So, the grammar was so we have the production rule S going to 0 S 1 S or it goes to 1 S 0 S or to epsilon. So, we remember that this takes care of all strings that begin with the 0; this takes care of all strings that begin with the 1 and so on. Now consider a string w in this language which is let say 0 1 0 1 1 0, so this string has equal number of 0s and 1s, so three in each case.

So, what would be a parse tree of this? So, we can have a parse tree where I replace, so we have S being replaced with 0 S 1 S then this S is replaced with epsilon, this S is replaced with 0 S 1 S. So and then we replace this S with epsilon, and then this S gets replaced with the production rule 1 S 0 S. We replace this with epsilon and this with epsilon. Now if I look at the concatenation of the leafs from left to right, I have 0 epsilon 1 epsilon sorry 0 epsilon 1 then 0 then epsilon then 1 and then again we have 1 epsilon 0 epsilon. So, if I concatenate them, I get the string 0 1 0 1 1 0. So, this is let say first parse tree of w with respect to the grammar G 2.

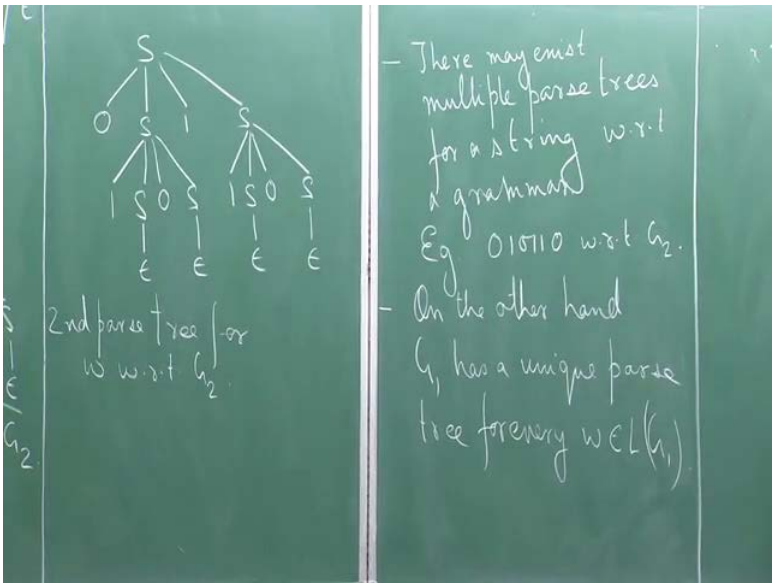
(Refer Slide Time: 14:23)



We can also have another parse tree for w. So, first I replace S with 0 S 1 S; I replace this S with 1 S 0 S; and I replace this S with again 1 S 0 S; and now I replace this S with epsilon S going to epsilon, S going to epsilon, S going to epsilon, and S going to epsilon. So, once again, if I look at the concatenation of the symbols, so I get 0 1 0 1 1 0. So, this is our second parse tree for w with respect to the grammar G 2. So, the moral of the story here is two points, so there are two important takeaways.

One is that whenever we talk about the parse tree of a string it is always with respect to a fixed grammar; parse tree of a string independent of a grammar does not make any sense, because the same string can be accepted in different manner in different grammar. So, it is always a string with respect to a grammar that is point number one. The second key point is that we can have grammars particularly the grammar G 2, where there is multiple parse trees for the same string. So, if you look at these two parse trees the one on the left, and the one on the right, they have different structure, but they are the parse tree for the represent the same string.

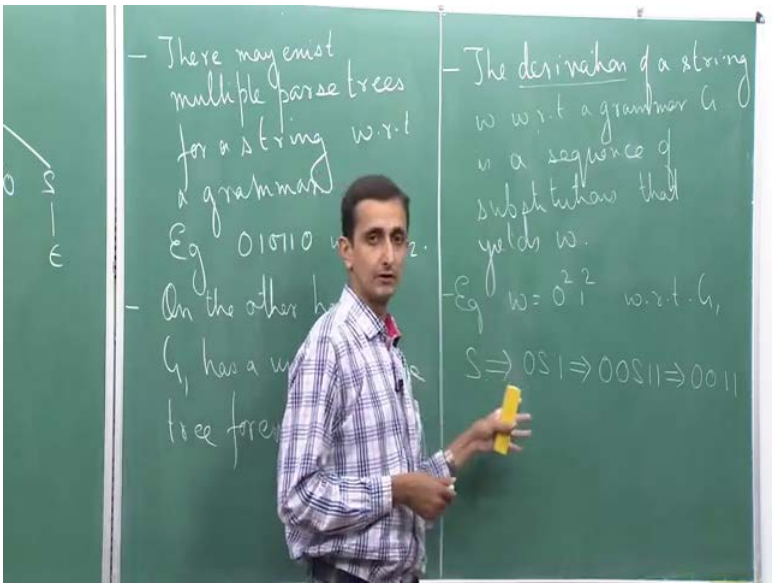
(Refer Slide Time: 16:28)



So, there may exist multiple parse trees for a string with respect to a grammar; an example is 0 1 0 1 1 0 with respect to the grammar G 2. What about the grammar G 1, the grammar G 1 that we had earlier for the language 0 to the power of n, 1 to the power n. Can there be a string which has multiple parse trees. So, the answer is no, because if you look at the rules of that grammar the production rules of that grammar. in every step of the derivation, there is a unique variable.

So, whenever I replace S with a production rule I replace it with 0 S 1, so there is exactly one variable that I get in the current string. So, we do not have any choice as to which rule to apply in order to get let say different parse tree. In this case, that is not true, because here I have two variables that are there in this current string 0 S 1 S. So, either I can replace the first string, if the first S or I can replace the second S with some other production rule. So this is what gives raise to ambiguity. So, on the other hand G 1 has a unique parse tree for every w l in the language of G 1.

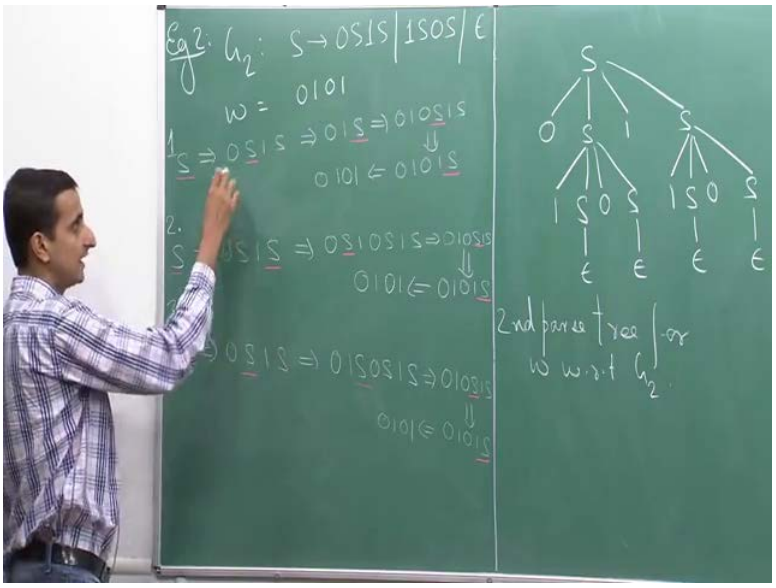
(Refer Slide Time: 19:09)



Now let me define the concept. Now let me formally define what a derivation is. So, the derivation of a string w with respect to a grammar G is a step-by-step or maybe I should write it as is a sequence of substitutions that yields w. So, it is sequence of substitutions starting from the start variable that yields w in the grammar G. Let us look at some examples, so again with respect to the grammar G 1. So, what is the derivation of a string let say if I take w equal to 0 square 1 square with respect to the grammar G 1.

So, I get a derivation, so I start with S then the first step is S yielding the string 0 S 1; in the second step, I have S yielding this S getting replaced again by 0 S 1, and then I get 0 0 1 1, where S is replaced with epsilon. So, this is a derivation. And again you can easily see that the string w with respect to the grammar G 1 has only one derivation. So, you cannot have multiple derivations. But let us go to our grammar G 2, let us look at the grammar G 2 and let us try to see, if we can have more than one derivation for a string.

(Refer Slide Time: 22:20)



So, this is our grammar G 2, and let me take a simpler w. Let us say I take a w 0 1 0 1. Once again this string belongs to the language of G 2. So, I can have a derivation where I replace S with 0 S 1 S. Then I replace the left most S with epsilon, so I get 0 1 S; then I replace this S with again 0 S 1 S. Now I replace the first S with the epsilon, so I get 0 1 0 1 S. And then I replace the second S with the epsilon and I get 0 1 0 1. So, this is one derivation. I can have another derivation of the string 0 1 0 1 as follows. So, I replace S with let say 0 S 1 S.

Now instead of choosing to replace the first S, I replace the second S. So, I replace the second S with 0 S 1, and I replace the second S with 0 S 1 S. Now I replace the first S with an epsilon, so I get 0 1 0 S 1 S. Then I replace this S with an epsilon, I get 0 1 0 1 S, and finally, I get 0 1 0 1. So, note that these two derivations are not the same, but they give the same string w.

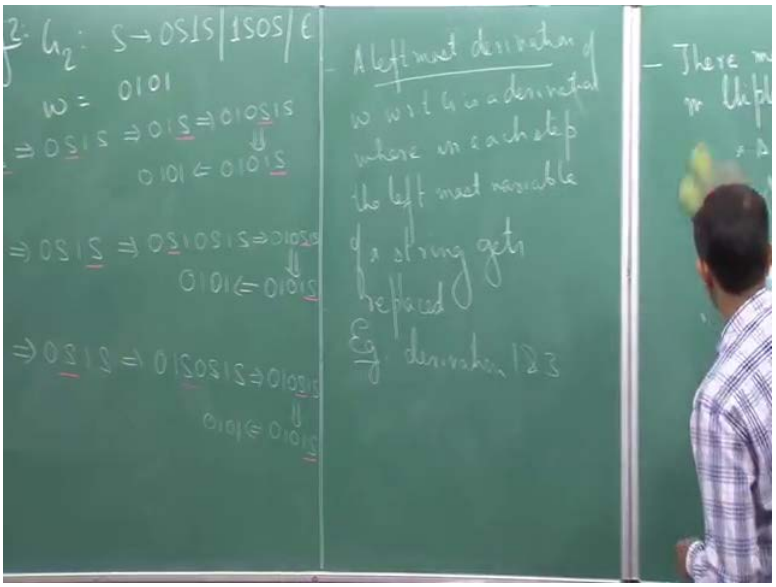
I can have a third derivation where I replace S with 0 S 1 S. Then I replace this first S with the rule 0, so I replace this S with the rule again 1 S 0 S, I replace it with 1 S 0 S and I have the remaining 1 and S. Next, I replace the first S with epsilon, so I get 0 1 0 S 1 S, then I get 0 1 0 1 S and then I get 0 1 0 1. So, if I have to mark which variables get replaced in each step here are the variables.

First, this S gets replaced with this, then I replace the first S with epsilon, then I replace this S, then I replace this S, and finally, I replace this S in this derivation. In the second

derivation, first I replace this, then I replace this S, then I replace this, this and finally this. In the third derivation, I replace this, and then I replace this, and then I replace this, this and finally this. So let us call this my derivation 1, this is 2, and this is 3.

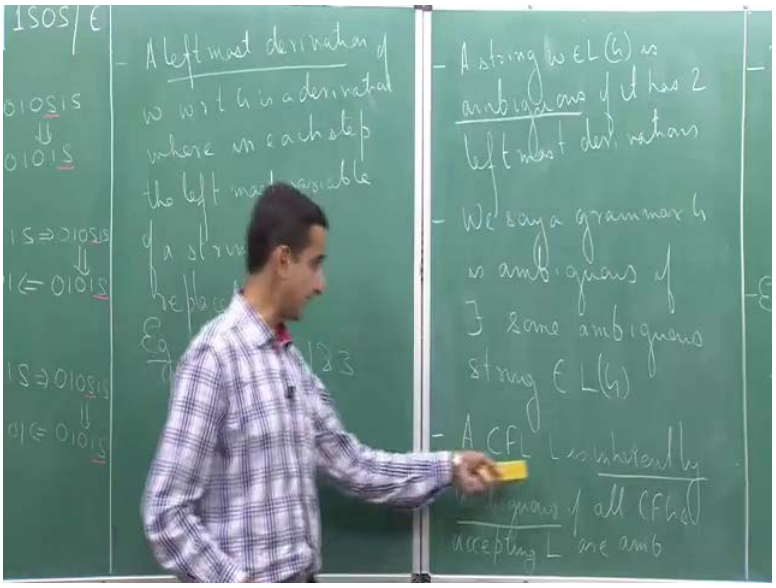
So, note that in my first and in my third derivation in the current string always the left most variable gets replaced. So, this is what is known as left most derivation. So, here also all the left whichever is the left most variable in a string that is getting replaced and the same thing happens here, but that is not so the case with the second derivation, here although the first S is the left most s, but I am replacing another variable to get the next string.

(Refer Slide Time: 26:54)



So, a left most derivation of w with respect to G is a derivation where in each step the left most variable of a string gets replaced, so example derivation 1 and 3 in the earlier example.

(Refer Slide Time: 28:16)



Now we say that a string w belonging to L of G is ambiguous, if it has two left most derivations. So, we say that a string is ambiguous, if it has two left most derivation. You can see that this string is an ambiguous string with respect to this grammar. We say a grammar G is ambiguous, if there exists some ambiguous string in the language of G. So, our grammar G 2 is ambiguous grammar, whereas our grammar G 1 if you remember so that is an unambiguous grammar, if you can check that every string will have a unique left most derivation.

And finally, we say that a context free language L is inherently ambiguous if all CFGs accepting L are ambiguous. So, if we have a language for which every grammar is ambiguous then we say that the corresponding languages inherently ambiguous. So, I will end here with a note that so this is one way of defining ambiguity. So, an equivalent way is that we say that a language or a string is ambiguous with respect to a grammar, if and only if it has more than one parse tree.

So, one way of defining is with respect to derivation if a string has more than one left most derivation we say it is ambiguous; also if it has more than one parse tree, we say that it is ambiguous; and both these notions actually equivalent, one can actually prove that.

Thank you.