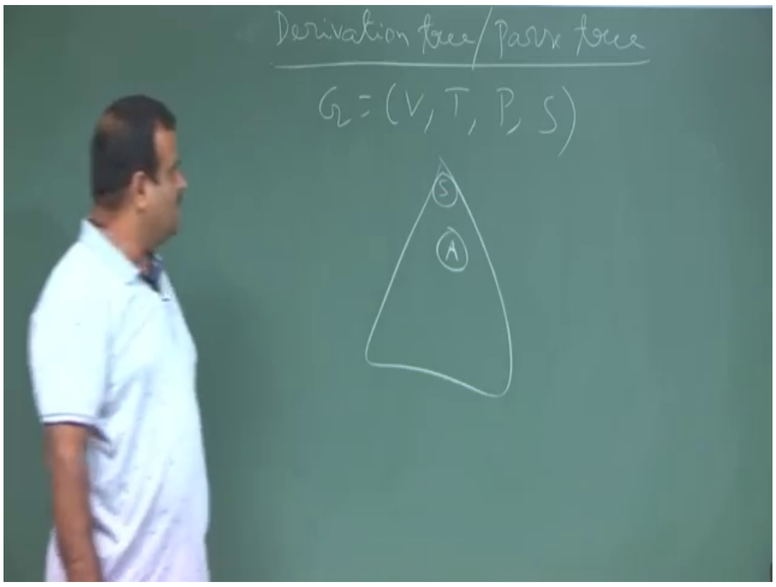
Introduction to Automata, Languages and Computation

Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

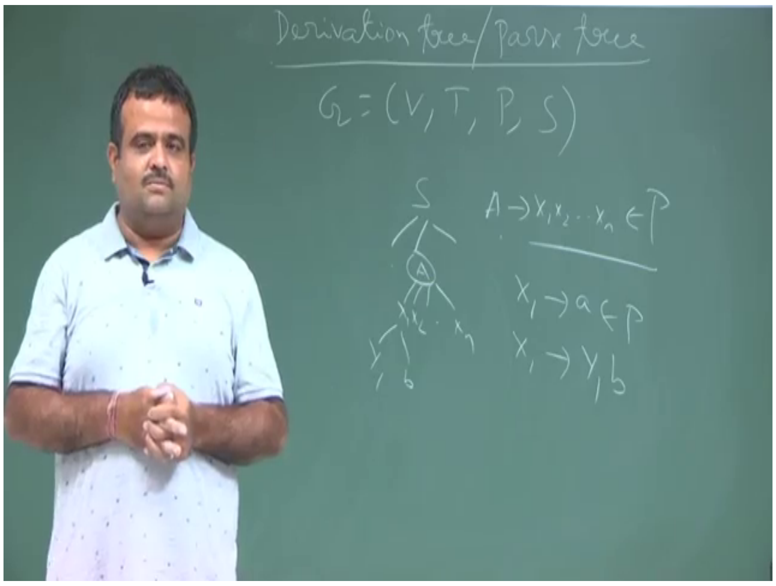
Lecture – 40
Derivation Tree / Parse Tree

(Refer Slide Time: 00:17)



So, we are talking about Derivation Tree or the Parse Tree of grammar. So, just to recap so, we have given a grammar V T P S or V is the set of vertices a set of variables consists of S also; S is a special variable which is called start variable. And T is the set of terminals or the alphabets finite set, P is the set of rules that is also finite. Now, we draw a tree; a tree is called derivation tree if is consists of the internal nodes are all the variables like A. And the branches of A the leafs this is and I need to start with the vertex S, S is the starting vertex.

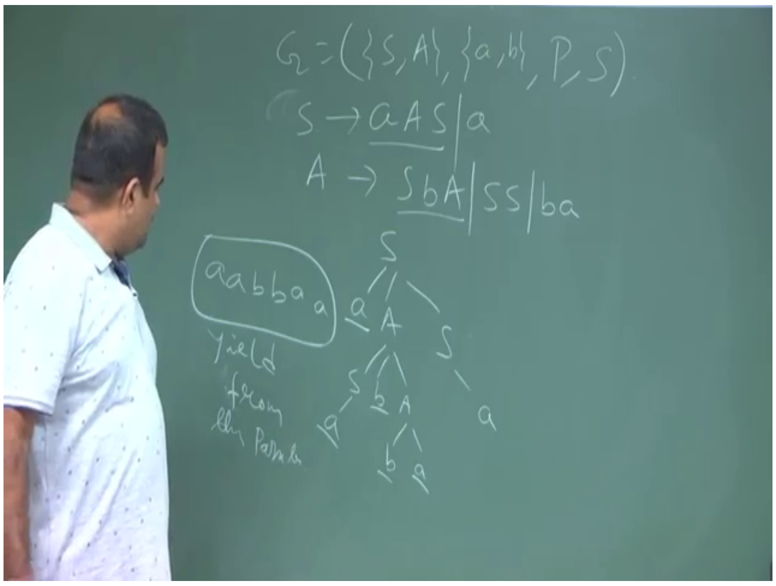
(Refer Slide Time: 01:26)


Derivation tree/Parse tree
$G = (V, T, P, S)$
$A \rightarrow X_1 X_2 \cdots X_n \in P$
$X_1 \rightarrow a \in P$
$X_1 \rightarrow Y_1 b$

So, it will start with the vertex S and we take a rule form P with S that will be the branches. So, like this we are going; so, suppose A X 1 X 2 X n; that means, A X 1 X 2 X n is a production in P. So, that is the way we derived this and finally, when we stop if it is reaching to some epsilon if there is a rule that alphabet is going to that variable is going to epsilon. Or, we stop if it is reaching to a terminals suppose this is going to say if this X 1 is going to say some if there is a rule X 1 is going to a; a is terminal see.

Then we stop this branch at a, if there is another rule X 1 is going to say Y 1 b then we can have this also we can write Y 1 b. So, these are all coming from the this tree is drive from this production in P. And the all the internal nodes of the trees are the variables other than the leaf nodes and the leaf nodes are only the terminals or it is epsilon; if there is a epsilon production in the in P. So, we have taken an example; so, this is called parse tree or the derivation tree.

(Refer Slide Time: 03:11)

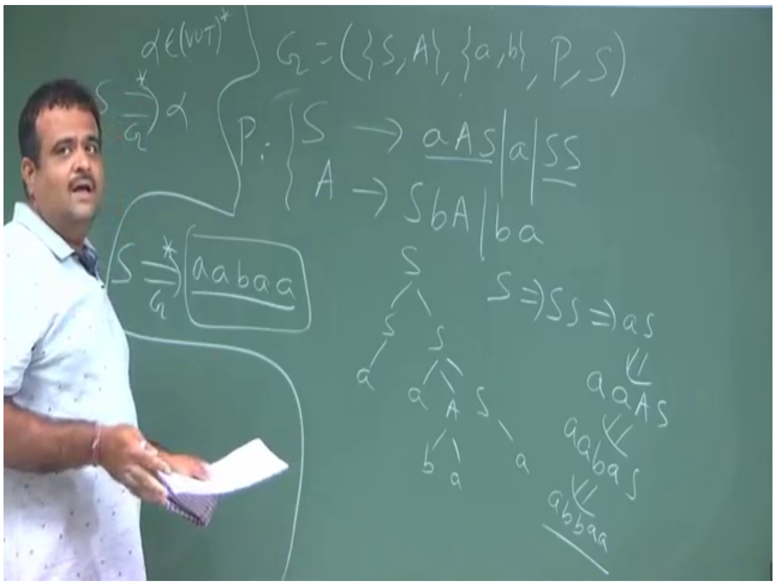

$G_2 = (\{S, A\}, \{a, b\}, P, S)$
$S \to aAS \mid a$
$A \to SbA \mid SS \mid ba$
aabbaa
yield from the Parse tree

So, this is the just a last class we have taken this example a b P S. So, now P consists of this a is A S or a and a is going to so, this is S is going to and A is going to S b A or SS or ba. Now, in the last class we have derived this so, S we can take either if we take a only which stop; that means, a is yield by this. So, this is also a parse tree or we can if you take this rule this is a S S. Now, this is know yet to go this already a terminal which stop this branch over here. And this A we can go we can take this one S b A.

And this S we can again stop at a and this S we can stop at a this b is already stopped and this A we can stop ba. So, what is the; what is the string it is yielding? So, we have to write from left to right; we take the only the leafs we read the leafs from left to right. So, a then a then b then b then a then a. So, this is the string yield by this tree yield from this parse tree parse tree. Now, depending on the production we use in the tree it will give us different different string ok.

So, this is the one stream which is yielding by this tree. So, it is the concatenation of the terminals from left to right. Now, if we want to we can have another tree like this say, we want to have this the same grammar or we can have some different grammar.

(Refer Slide Time: 06:00)

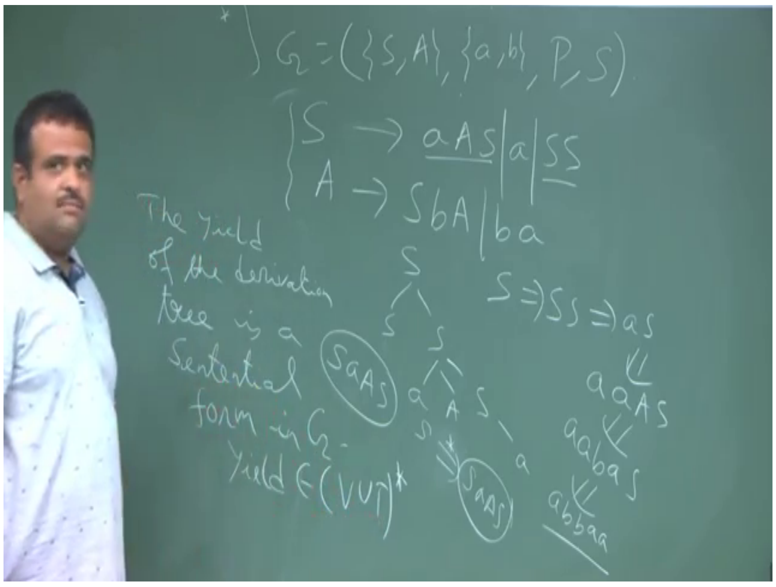

α∈(V∪T)*
S ⇒*_G α
G_2 = ({S, A}, {a, b}, P, S)
P: S → aAS | a | SS
A → SbA | ba
S ⇒*_G aabaa
S ⇒ SS ⇒ aS ⇒ aaAS ⇒ aabaS ⇒ abbaa

So, say S is going to S is going to a A S a we put SS over here and A is going to S b A b a this is our P. Now, let us consider a parse tree. So, S we can take this one say S S, this rule we can take then this S is going to a and this S we can take this rule A S this. And then this A we can take b a and this S we can go for a; this is also an example of a parse tree. Now which what it is yielding? It is yielding so, you just read left right the terminals.

So, a a b a a this is the string yield by from this derivation tree ok. Now, this we can show this is basically S is S this is derived from this. How to show this? So, if you start from S we take this rule S S the way we build the this parse tree; then from S S this S we put this. And then this S we use that rule this a S S S. Now, this a we are using this b a S; now this S we are going to a this rule a b b a a so; that means, this is the string is derived from this; that is the way we develop the parse tree ok.

So, basically this is a sentential form, we know the sentential form of the grammar is any string alpha which is belongs to it could be any string alpha is called sentential form; E base this alpha can be divided some S using the production of G any it is sentential form. And, here alpha could be variable also, but here since it is a generating grammar generating the string of terminals so, this is also a sentential form. So, any so, whatever we are getting from the whatever is yielding from the parse tree is of sentential form.

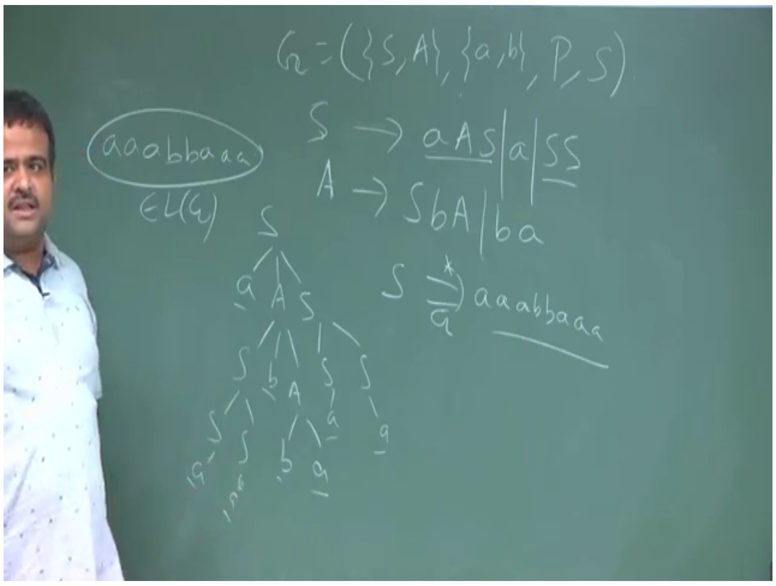
(Refer Slide Time: 09:10)


G₂ = ({S, A}, {a, b}, P, S)
S → aAS | a | SS
A → SbA | ba
The yield of the derivation tree is a sentential form in G₂.
Yield ∈ (V ∪ T)*
SaAS
S ⇒ SS ⇒ aS ⇒ aaAS ⇒ aabaS ⇒ aabbaa

So, let us write this the yield of the derivation tree derivation tree is a sentential form from in G ok. So, that is whatever yield is coming belongs to V. So, yield we can stop like for example, this is also yield because yield need not be a always the terminal whatever we stop we read this left to right. So, this is what it is S a even we do not we cannot go further S a A S; this is also yield because this is coming from S is S a A S. This consists of variables and the terminals that is why we are writing this, but eventually we want to reached to a holy the string of terminals because, that that consists of the grammar a language sorry that consists of the language.

But, yield may not be a only the string of terminals, it could be a variables also, if we stop; if you do not apply any rules after this then it is just S a A S if we do not apply this. So, that is also yielding by this parse tree; parse tree is we may not stop at the end. If you are stopping at the terminals those are basically the leaf nodes. So, those are basically our terminals then that will give us a grammar I am in the language. So, this is one example of the parse tree.

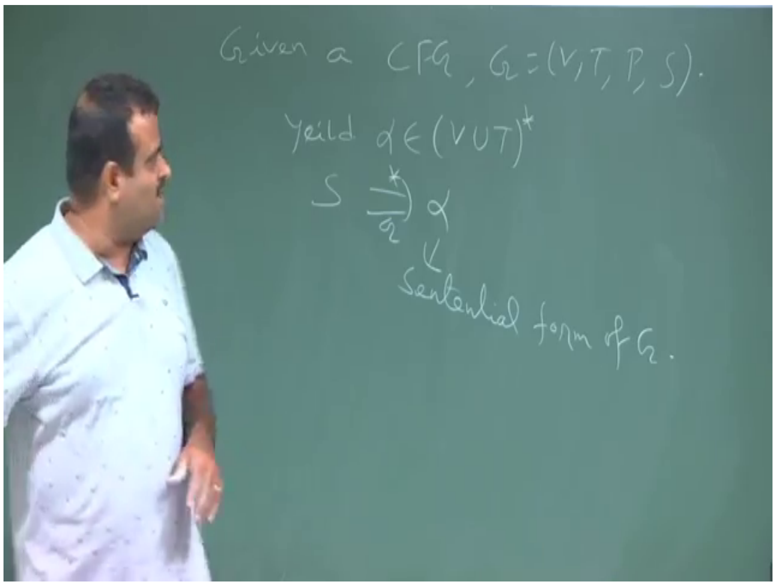
(Refer Slide Time: 1 1:29)



Now, it may have another string we may get from this parse tree for example, so, S S can go to say a A S. Now, this a we have know how to go; now this A A we can go to say S b A. And this S again we can go to S S and this S we can go to again SS, b is no had to go and this S we can go to b a and this S we can go to a. This S we can go to a and this S we can go to a this S we can go to a. So, what is that what it is yielding? If we complete this tree we may stop at some point of time that also will be yielding some stream, but that will consist of variable and the terminal. But, here we have we reach to the end there is no how to go now then that is only yield the string of the terminals, that is the generating the language you look.

So, what is the language what is the string? So, we read a then, a then, a then we have a b then b then a a a a a. So, this is the yielding so, this belongs to L of G ok. So, for different different string we have different different a parse tree ok. So, this is yielding means S is so, this string is derived from this a a a b b a a a ok. So, this is a string of terminal this belongs to the language, but we could stop at the middle. Then that also yielding the that also yielding something which is a mixture of variable and the terminals.

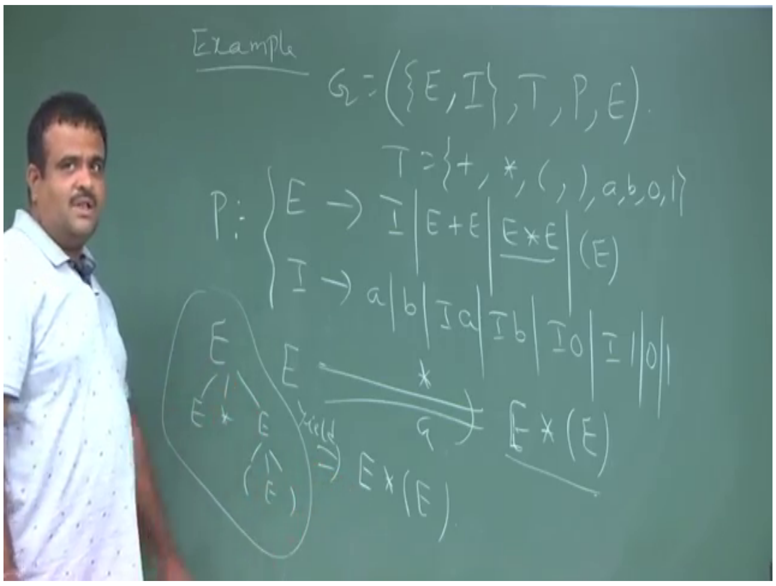
(Refer Slide Time: 1 3:45)


Given a CFG, $G = (V, T, P, S)$.
Yeild $\alpha \in (V \cup T)^*$
$S \overset{*}{\underset{G}{\Rightarrow}} \alpha$
Sentential form of G.

So, this is so; that means, given a language yielding means I mean given a language, given a grammar context free grammar if we construct a tree based on the rules; based on the rules. And suppose a particular tree is yielding something like alpha which is belongs to this; if we are stopping in the middle I mean if you are not reaching to the terminals only or epsilon if there is epsilon rules epsilon production.

Then if yielding this alpha then we can show this then alpha is sentential form of G ; that means, alpha can be derived from S by the repeated application of this; that is the way we from the tree that is our parse tree that is the way from the tree ok. Let us take some more example. So, again alpha is if alpha is a string of terminal then it is a language I mean we get a language out of it ok.

(Refer Slide Time: 1 5:11)


Example
G = ({E, I}, T, P, E).
T = {+, *, (,), a, b, 0, 1}
P: E → I | E+E | E*E | (E)
I → a | b | Ia | Ib | I0 | I1 | 0 | 1
E ⇒* E*(E)
yield ⇒ E*(E)

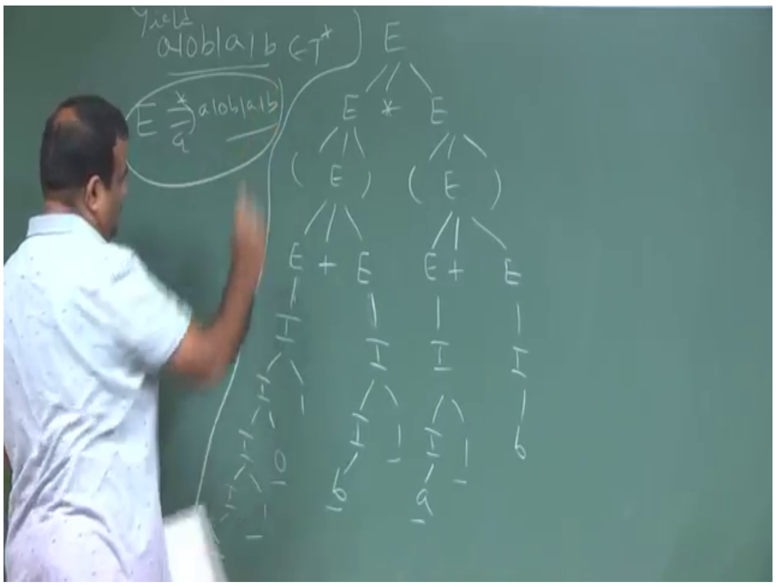
So, let us take another example. We take this G context free grammar which is having two variables E and I. So, you have to take one of them as a starting variable, say you have taking E as a starting variable. And say T is the terminals which are of this form plus star then a b 0 1 ok. And now we have to have a rules productions; suppose E is going to I 1 rule or E is going to this another rule, E is going to this is the all the production from E. And now from I we can go to a or we can go to b these are the different or I a I b or I 0 I 1.

So, then we can have 0 or 1 also. So, this is our P this is our productions set of productions set of rules ok. So now, using these we want to have a parse tree and then we can have a this. So, suppose E is going to where E is so now, let us derive the do the derivation. So, E is can you write E is yielding E is going to E star this. Why? Because, in the first step we can go here say in the first iteration we can go E star E, then in the second step this E we can take this rule E start this.

So, basically this is yielding form this is not yielding this is derived from here this is derived from here. Now, if you want to draw the parse tree; what is the parse tree? For this so, if we have E so now, we are taking this rule. So, it is going to E star E now again this E we are now this is a terminal we have no move, but here we could go. But, here we are not going because you want to we just if we go further until till to the terminus then it will give us a string of terminal which is a belongs to the language generating the language generating language. But, we may stop I mean without going so, that also yielding. So, yielding is a string of variables and the terminals.

So, then this E can go to this rule if we apply bracket E this. Now, if you do not go further suppose, this is our parse tree then what it is yielding? It is yield we read just left right so, E star bracket E bracket which is same as this ok. So, here it is yielding not only the string of terminal it consists of the terminals and the variables also in general we can say. But, we are interested on the terminals because terminal string of terminals are from the language generating language ok. So, we take another example where it is just the string of terminals.

(Refer Slide Time: 19:49)



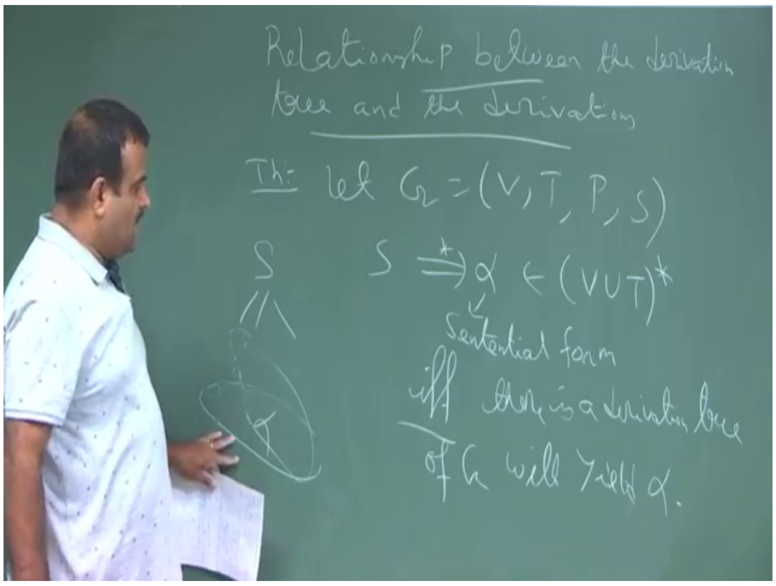
Say E is going to say E start E we are forming a parse tree and say this is going to E this bracket and here know how to go, but this can go bracket E this bracket. Now, again these are stop not have to go, but this E can go E plus E this will also go like this; anyone of the production we can apply and we can keep on going these branches this we can keep on from the tree. Now, suppose this is know how to go suppose this is going to I and this is going to I; here also you go to I here also you go to I.

Now, from I we can go to I 1 these are we can go to I 1 this I we can go to I 1 this is we are going to b. Now, this I 1 is know how to go this I we can go to I 0 again this I we can go to I 1 and this I can go to a which stop there. And this I we can go to just b and this I we can go to just a. Now, we are no we stop no ahead to go because, all are all the leafs are here terminals. Once we reach to a terminals we cannot further proceed, we cannot further have the branch because, branch will come only from the variables branch will come from the productions.

So, production means left hand side is the variable and the right side is the string of variables and the terminal. So, once we reach to a terminal then we have to stop there is no head to go. So, what it is yielding? It is yielding so, you have to read from the leafs you have to read the leafs from the left to right. So, this is a then 1 then 0 a 1 0 b then 1 then a then 1 then b; so, this is it is yielding. So, it yield and this belongs to; this belongs to terminal string, this a terminal string.

So, it is a part of the language generated by this. So, this here it is whatever it is yielding belongs to terminal strings and also we can verify that this S S is basically E E is going to I mean this can be derived from this is a sentential form. So now, we will have a way to formally define the what is called derivation tree and the this sentential form the derivations.

(Refer Slide Time: 23:40)



Relationship between the derivation tree and the derivations

Th: let $G = (V, T, P, S)$

$S \overset{*}{\Rightarrow} \alpha \in (V \cup T)^*$

Sentential form

iff there is a derivation tree of G will yield α.

So, let us just let us just formally define what is the we just state the relationship between the parse tree derivation tree and the derivation. So, this is in a theorem we can state this let b V grammar j sorry G V a grammar which is a context free grammar. And if we have derivation of alpha if say S is going to alpha; alpha could be a string of terminal or in general it could be a consists of variables in terminal. So, alpha is derived from S by using the productions then we have a derivation tree which will yield alpha.

Because so, then if a leaf alpha is a derivation alpha is sentential form; if and only if this is necessary and sufficient. If and only if there is a derivation tree or parse tree derivation tree of G which yield alpha. That means, we start from S and then we should reach to this is the if we read the we should reach to alpha. So, this alpha should be alpha we should have a parse tree which is yield alpha. So, this is necessary and sufficient condition. So, you look at this in more detail; so, we have seen this by the example.

So, given a parse tree; given a parse tree we have the yield sequence. So, that can be derive from the starting symbol because eventually we are applying the production in the parse tree; so, they are basically same. We will talk about more in the next class.

Thank you very much.