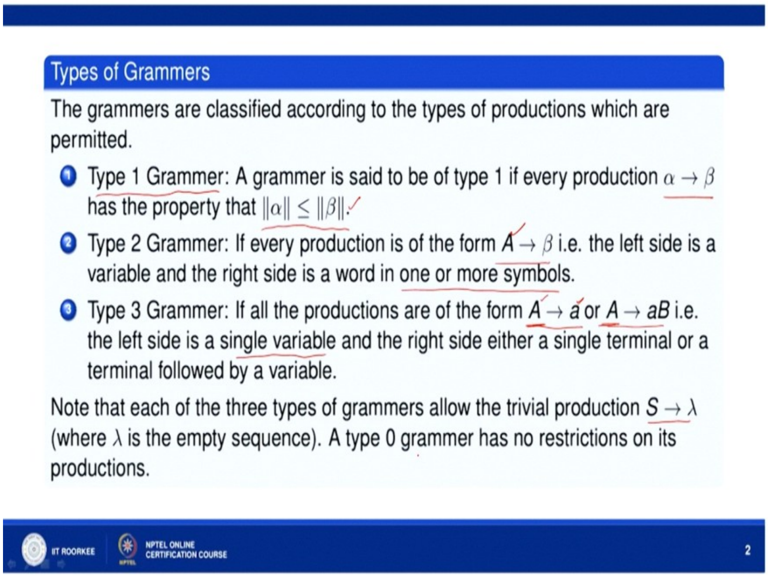
Higher Engineering Mathematics
Prof. P.N. Agrawal
Department of Mathematics
Indian Institute of Technology Roorkee
Language and Grammers - III
Mod02_Lec09

(Refer Slide Time: 0:29)



Hello friends welcome to my third lecture on the Languages and Grammers. In this lecture we shall discuss the various types of the grammer. The grammers are classified according to types of productions which are permitted, type one Grammer, let us see what is type one grammer. A grammer is said to be of type 1 if every production $\propto \to \beta$ has the property that $\| \propto \| \leq \| \beta \|$, let us remember that the $\| \ \|$ of, this notation, I am calling it $\| \propto \|$, a $\| \propto \|$ means length of this string $\propto$ okay.

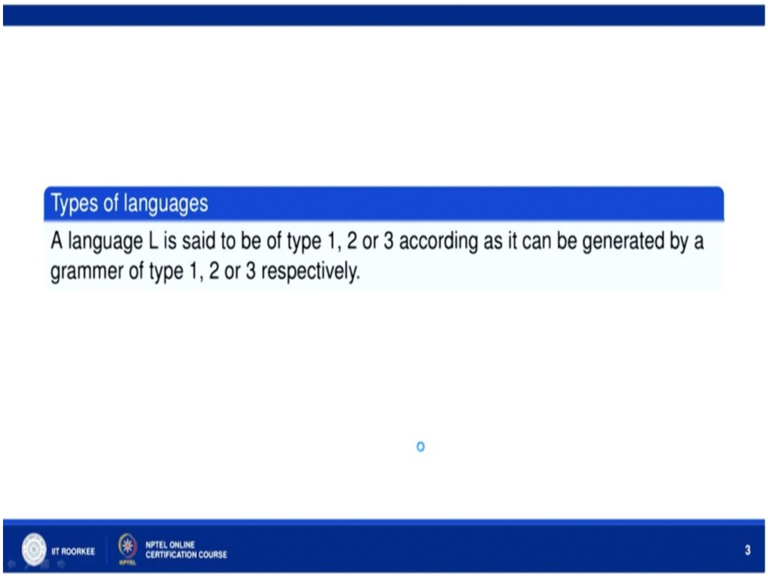
So this is $\| \propto \| \leq \| \beta \|$. Now $\| \ \|$, type 2 grammer, if every production is of the form $A \to \beta$ that is the left side is a variable, A is a variable as we know okay and this the right side β, β is a word, I would consist of one or more symbols, so here the left side is a variable okay, the right side is a word consisting of one or more symbols.

Now, in the type 3 grammer, all the productions are of the type $A \to a$ or $A \to aB$, A you know small a, it is a terminal symbol okay, so either $A \to a$ okay, left side, you can see left side is a variable, right side we have a terminal symbols, so $A \to a$ or $A \to$ a terminal symbol A followed by a variable okay, so we can see the left side is a single variable okay, the left side is a single variable A here, here also A, so left side is a single variable, right side is either a single terminal, like here it is a single terminal or a terminal followed by a variable, here we have a terminal A followed by a variable B.

Now, we can see each of these types of grammers okay, they are allow the trivial production $S \to \lambda$, how, you can see $\| S \|$ are sorry, $\| S \| = 1$ because it is one symbol, $\| S \| = 1$, $\| \lambda \| = 1$, so it satisfies the inequality, $\| \propto \| \leq \| \beta \|$., further it also there in type 2 grammer because it is of the form $A \to \beta$, A here is replaced by S here and in place of β we have λ okay.

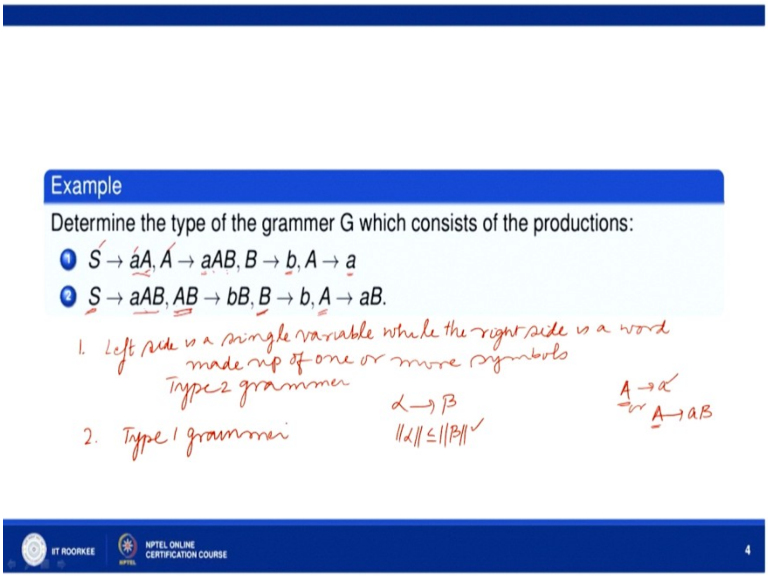
Now, it also belongs to type 3 grammer because it is of the type A → A okay, AS → λ, so these S → λ is there in all the 3 types of grammer okay. A type 0 grammer has no restrictions okay, on its production, so if we do not, if we have a case where type 1 grammer, type 2 grammer, type 3 grammer none of the 3 apply then that case will be the case of type 0 grammer, there is no restriction on this kind of a grammer.

(Refer Slide Time: 3:43)



So, let us see a language, types of language, a language L is called to be of type 1, 2 or 3 according as it can be generated by a grammer of type 1, 2 or 3 respectively.

(Refer Slide Time: 3:52)

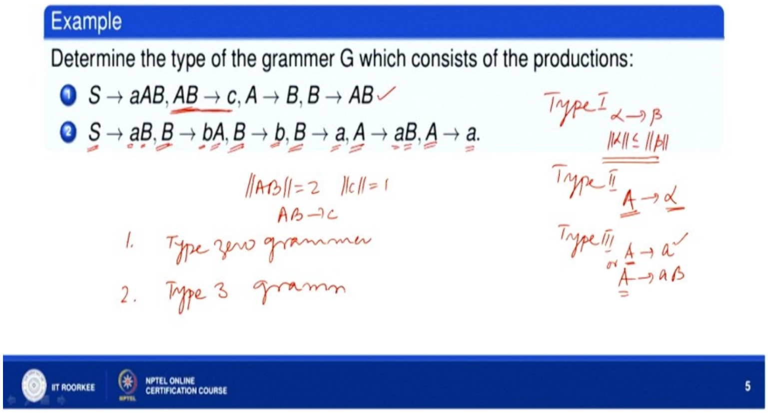


Now, let us look at this example, determine the type of the grammer G, which consists of the productions, S → aA okay, A → aAB, B → b A → a, now here you can see left side is a variable, here is single variable, S here, A here, B here, A here okay. Right side is a word okay, which consist of one or more symbols, here we have a terminal A and a symbol A, here we have a terminal A there are two symbols, two variables, A and B, here we have an terminal B and a terminal A here, so left side is a single variable okay, in the case 1, left side is a single variable, while the right side is a word made up of, consisting of one or more symbols, made up of, there are two symbols here A and A, there are three symbols here small a, capital AB, there is one symbol here, there is one symbol here, so the first case, in the first case, the grammer generated by these production rules will be of type 2. Okay, so type 2 grammer.

Now, let us go to the second case S → aAB okay, AB, now here you can see, here there is one variable on the left side, here there are two variables on the left side okay, here there is one variable on the left side, here there is one variable on the left side, so this grammer, the grammer generated by these production rules cannot be of type 2. Okay, it cannot be of type 3 also why? Because in the case of type 3 we have S → A or we had, A → a or A → aB, that is a terminal followed by a variable okay, so either A should go to, either we should be having a terminal okay or we should have terminal followed by a symbol.

Now, but left hand side should be having a single variable, here there are two variables here aB, so it cannot be of type 3 also but it can be of type 1, in the case of type 1 we have α → β and || α || ≤|| β || so here || S || =1, length of this aAB = 3. Okay, length of this is 2, length of this is 2, okay, length of this is 1, length of this is 1, || A || = 1, || AB ||= 2, so each of this production rules satisfy the inequality, || α || ≤|| β || okay, so in the case 2 we have type 1 grammer okay, so because each of the production rules satisfy the inequality || α || ≤|| β ||.

(Refer Slide Time: 7:47)



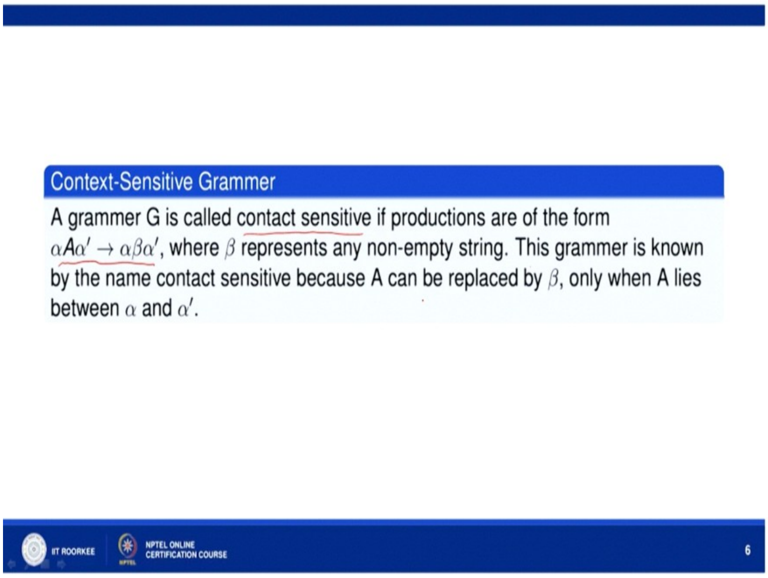
Now, let go to the second example, determine the type of the grammer G, which consists of the productions S → aAB, AB → c, A → B, B → AB, let us look at this first, let us see whether it is of type 1 okay, in the case of type I grammer, we have α →β, it is, it consist of the production of the type, α → β where || α || ≤|| β ||, type 1 grammer is generated by such kind of productions rules okay.

So, here you can see we have this AB → c, so || AB || = 2, and || C || = 1 okay, so AB → C does not follow this rule okay, || α || ≤|| β ||, so this is not, this first case is not of type 1 grammer, is it of type 2 grammer! In type 2 grammer okay, we have rules of the type production rules of the type S → α okay, S → α, there no, A → α, where A is a variable, A → α, A, so left side is a single variable okay.

In the type 2 grammer left side is a single variable, right side is a word okay, which consist of one or more symbols, so here left side should have only single variable, why here we see left side has two variables, so it is not of type 2. Okay, now it is not of type 3 also, why? Because, type 3 grammer is generated by production rules of the type A → a or A → aB okay, so left side here again is a single variable, here left side is a consisting of 2 variables okay, so it is not of type 3 okay.

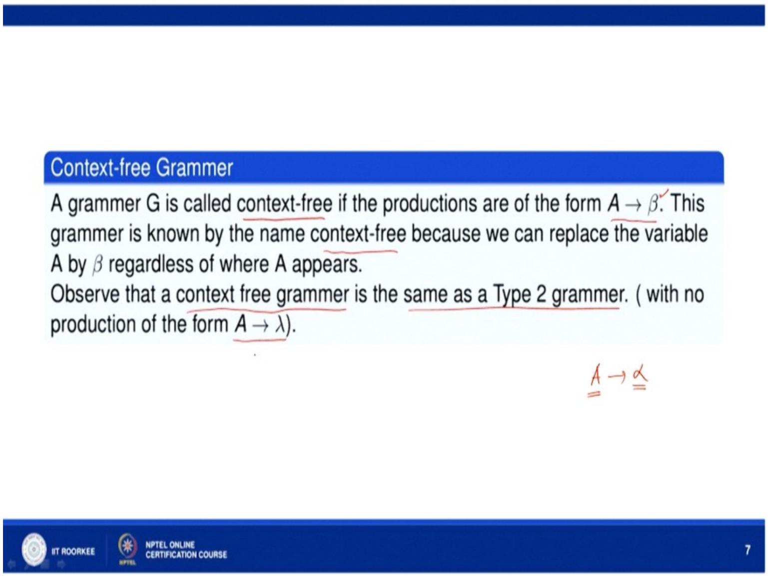
When the grammer is not generated by type 1, type 2, type 3 we call it as type 0 okay, so the first case is of type 0 grammer because it does not follow any rules okay, type 0 grammer. Now let us go to 2, S as → aB okay, so here there is a single variable, here there is a single variable, here, here, here and here, so all are single variables. Now let us see here we have a terminal followed by a variable, here we have a terminal followed by a variable, here we have a terminal, here we have a terminal, here we have a terminal followed by a variable and here we have terminal, so all these production rules are of the type, variable → a terminal or a variable → a terminal followed by a variable okay, so the second case is of type 3 grammer.

(Refer Slide Time: 11:02)



Now, let us discuss context sensitive grammer, a Grammer G is called contact sensitive if productions are of the form α A $\alpha^{'} \rightarrow \alpha\beta\alpha^{'}$, where β represents any non-empty string, empty string as we know it denoted by λ, so this grammer is called by the name context sensitive because A can be replaced with β only $\alpha \leq A \leq \alpha^{'}$

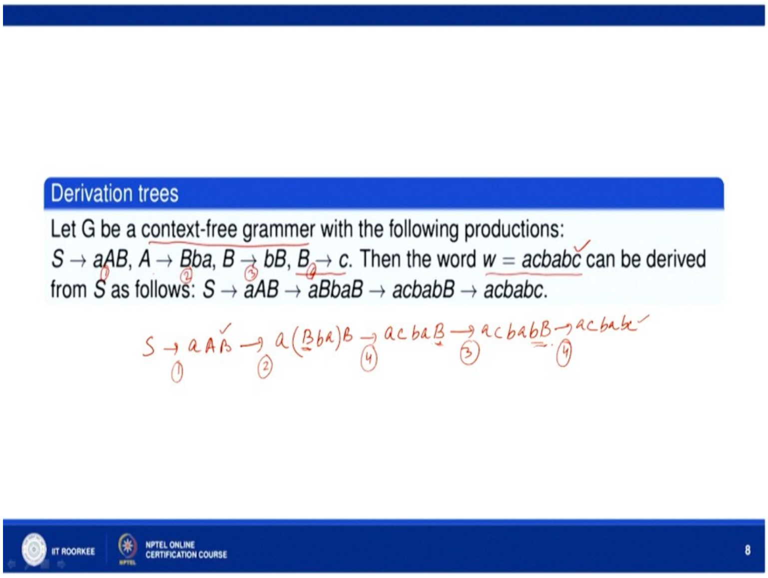
(Refer Slide Time: 11:29)



Now, a grammer G is called context, there is another type of grammer which is called as context free, if the productions are of the form A → β. This grammer is known by the name context free because we can replace A by β, regardless of where A appears. Now let us know that context free grammer is the same as type 2 grammer, in the type 2 grammer if you do not consider the production of the form A → λ, A → λ we are, if you do not consider in type 2 grammer, then the grammer, type 2 grammer is same as context free grammer because in the context free grammer this β is a nonempty string okay.

So, we cannot include A → λ, so from the type 2 grammer, if we do not consider the production of the form A → λ than it is same as the context free grammer, so here in the context free grammer A replaced with β, regardless of β where A actually appears. In the type 2 grammer if you recall what we had said, type 2 grammer consist of productions of the type A → α, where, on the left side we have a single variable, on the right side α is a word, consisting of one or more symbols and type 2 grammer includes A → λ, so if you remove A → λ from this type 2 grammer, than it is same as context free grammer.

(Refer Slide Time: 13:07)



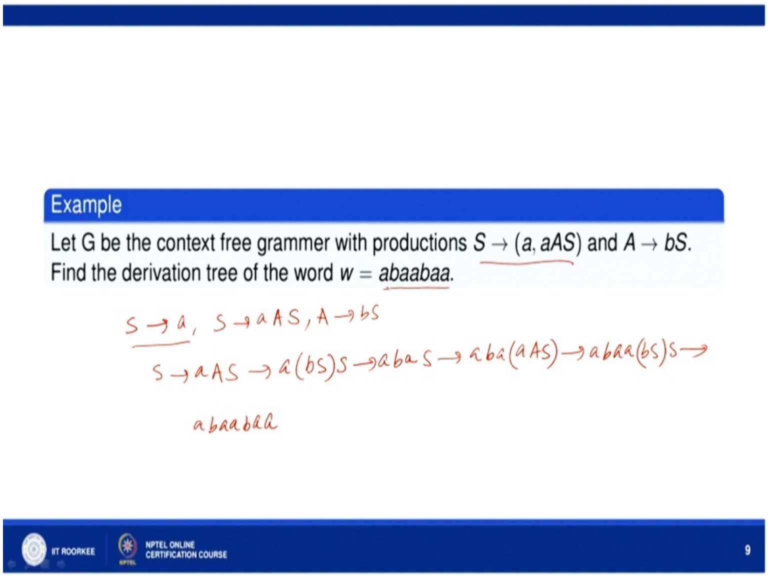
Now, derivation trees, let G be a context free grammer with the following productions, S → aAB, A → Bba, B → bB, B → c. Then the word W = acbabc can be derived from S as follows, you can see we want to drive W = acbabc okay, this word we want to drive, so what we should do, first of all, we should start with S, S → aAB okay, now we want ACBABC, so A, A → Bba okay.

So we write A → Bba and we have B here. Okay, now if you use the production rule B → c, than we will have acbac, we will not reach here okay, so we should not use B → c here. Okay, what we should use? Because we want ac, we want ac okay, yes for this we can use this B → c, so this →, for this we, we write, we use B → c, so we have ACBA and B okay and for this B, now if you use B → c than we will not reach here, so for this B we use bB.

So, acbabB okay, so we have acba okay, now we want bc, so we can go to now acbabc okay, S as → aAB be used first then we use the second production rule A → bBA to reach this, so this is first production rule, let me say this is first, then this is 2, this is 3, this is 4, so let us see in what sequence we are using okay, first we are using production rule 1 okay, then we are using production rule 2 and then we are using production rule 4 okay to come to this place and then B → bB, production rule 3 we are using here okay and then we are using 4.

So, 1, 2, 4, 3, 4, if we use like this, so S → aAB then we have aBbaB and then we have acbabB okay, this B, I have not written this step here, so ACB this is implicit here, acbaB and then B is replaced with bB here and then we ultimately replace B with c, so we reach here, so you can see, when you allow context free grammer, then B → c wherever B occurs can be place with c okay.

(Refer Slide Time: 16:06)

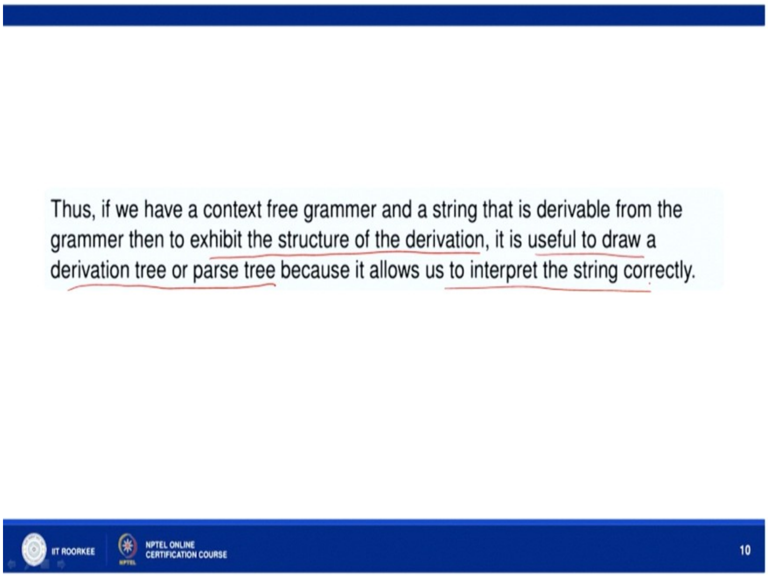


Now derivation trees we will come to that before, let us me also say, let G be the context free grammer with production S → a, S → aAS, A → bS, find the derivation tree of the word W = abaabaa, first let me write the sequence of, sequence in which we can use the production rules, so S →, now S →, here S → aAS, and S → aAS okay, because we have this production means S → A, S → aAS and we also have A → bS okay.

Now, let us see in what sequence we shall use this production rules to reach the word W = abaabaa okay, so S →, first we shall use this S → aAS okay, S → aAS, now A → bS we have to use that because we want AB okay, so A → bS okay and we have S here okay, now what we do if you write S → a, so we shall have aba okay, aba and this S we should then replace by aAS.

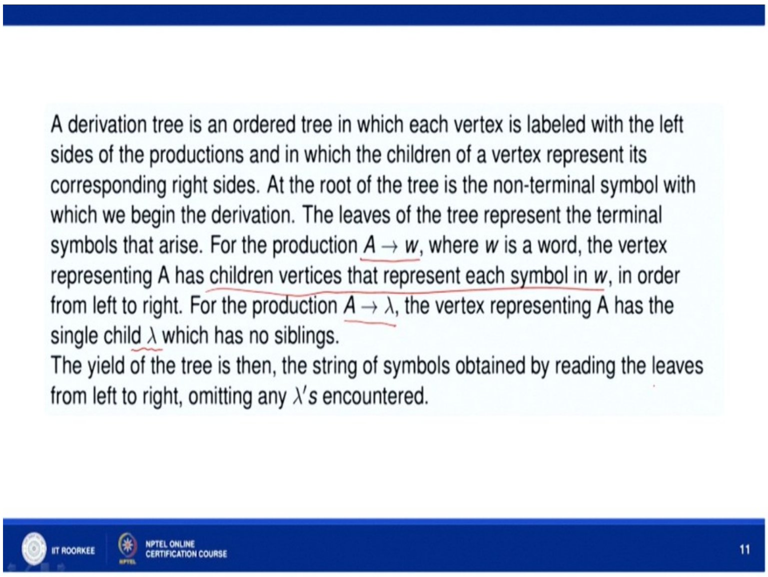
So this is, this now → aB, this S can be replaced with A here okay and then we have S, S now → aAS, so aba, aAs okay, so we have aba, AB AB have, now what we will have AS, A → bS because we want now BAA, so we want this abaa and A → bS okay, so this now → abaa and bs → aa, we want, yes S → a, so we will have baa okay, by using this production rule S → a, so S → a, A will then bring lead us to abaabaa so by using the production rules in this sequence we arrive to the word W = abaabaa, now this sequence can be presented by a derivation tree.

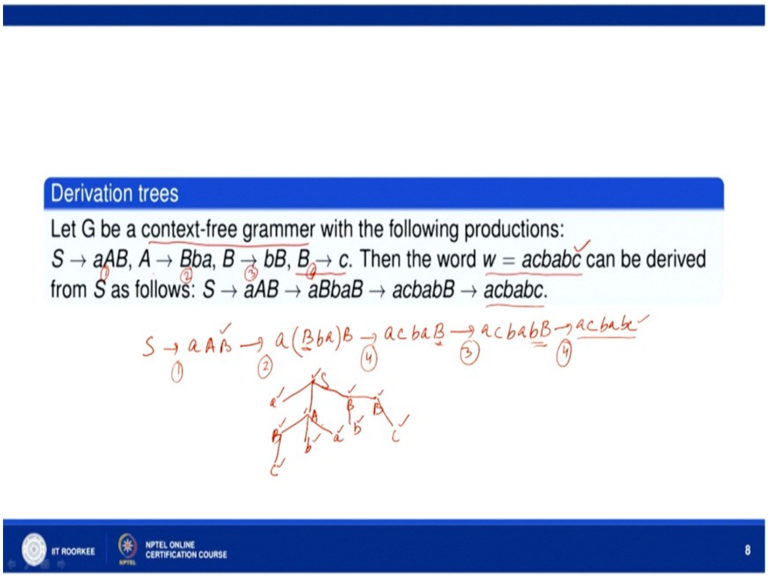
(Refer Slide Time: 18:51)



Let us define what we mean by derivation tree, before that. Thus if we have context free grammer and a string that is derivable from the grammer then to exhibit the structure of a derivation, it is useful to draw a derivation tree or parse tree okay, because it is allows us to interpret the string correctly.

(Refer Slide Time: 19:15)





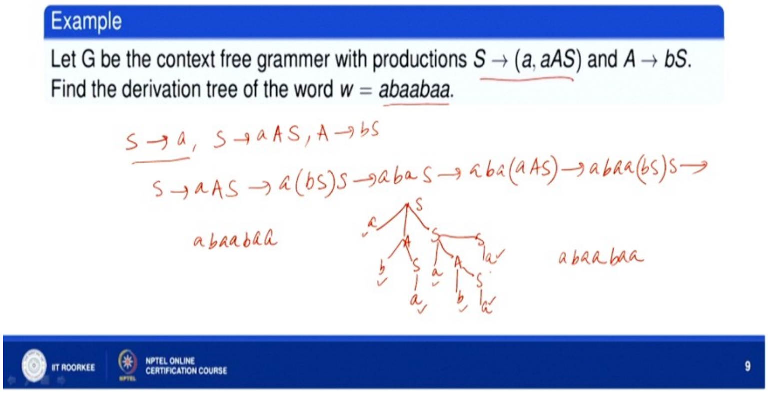
So, let us see what is a derivation tree? A derivation tree is an ordered tree in which each vertex is labelled with the left sides of the productions and in which the children of a vertex represent its corresponding right sides. The root of the tree is the non-terminal symbol with which we begin the derivation okay. The leaves of the tree represent the terminal symbols that arises, for the production A → W, where W is a word, the vertex representing A has children vertices that represent each symbol in W, in order from left to right.

For the production A → λ come the λ is the empty string, the vertex representing A has the single child λ which has no siblings. The yield of the tree is then, the string of symbols obtain by reading the leaves from left to right, omitting any λ's encountered. Now, let us see how we get the tree here in this case, say for example derivation tree if you want to write, we will have S here okay, let say this is my S, S → aAB okay, this is the first production rule.

Now will replace A with Bba okay, so A with Bba okay, Bba okay, Bba than second one is, but we did A replace with Bba, now what we did was B we replace with c okay, so B with c okay and then BAB okay, so acbab okay and then what we did in the next step ACBA, B we replace with bB okay, so B we replace with this B okay, this B we replace with bB okay, B we replace with bB and then B we replace with c okay, now you can see this is, these are vertices, this is vertex, this is vertex okay, this is vertex okay, this is vertex and this is vertex, they are vertices and these are childrens okay.

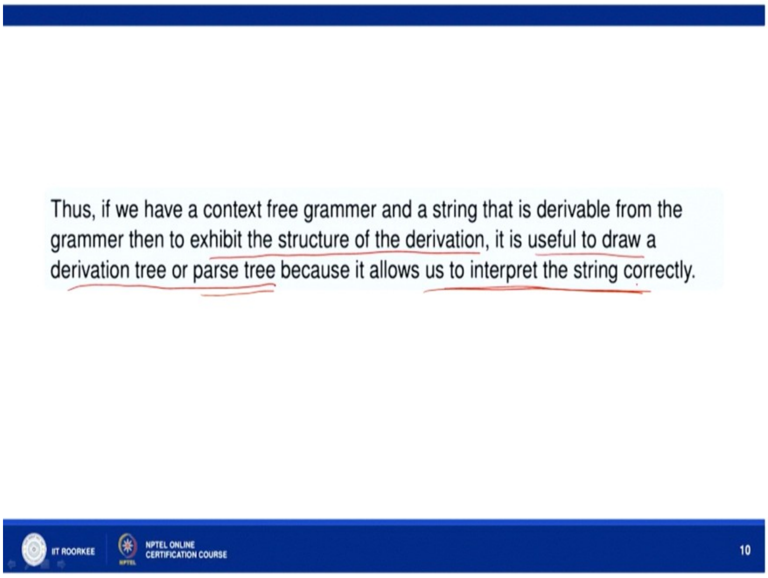
So acbabc, so we read them from left to right, you can see we start from here, so acbabc okay and we get this acbabc okay, so this is the derivation tree in this case, so I am in this production rules which, sequence of production rules when we apply in this manner will lead us to the word acbabc, so this how we reach the word acbabc can be exhibited by means of this derivation tree.

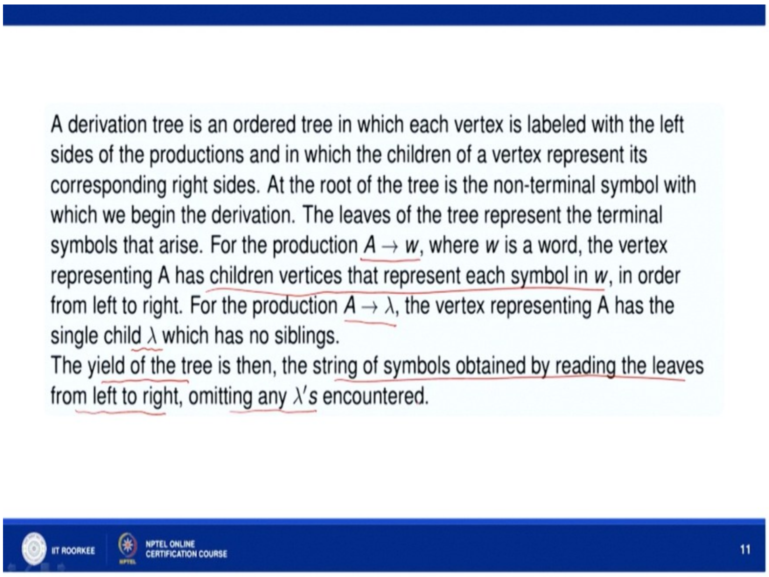
(Refer Slide Time: 22:35)



Now, let us look at in this case of what we will have, so S, this is my S, start simple S, then S → aA and S. Okay and then we use A → bS, so bS okay, A → bS and then we write S → a okay, so S → (a, bS), this S, this S → a okay and this S, this S now aAS, aAS okay, the right S, the rightmost S. Okay, this S → aAS okay and then A → this a, this A → bS okay and then the two S okay, this S and this S both →a okay, so now we can read the from left to right abaabaa and we have ab okay, aa b ab aa ba aa, so this is the derivation tree in this case.

→ (Refer Slide Time: 24:19)





So the derivation tree are also called as parse tree, sorry, derivation tree or parse tree okay, it allows us to interpret the string correctly, we have to simply read it from left to right as we have seen here, the yield of the tree is the string of symbols obtained by reading the leaves from left to right. If there is any λ in between the λ we omit okay, omitting any λ encountered that is not read, when we read the leaves from left to right, so this is how we draw a derivation tree, with this, I would like to end my lecture. Thank you very much for your attention.