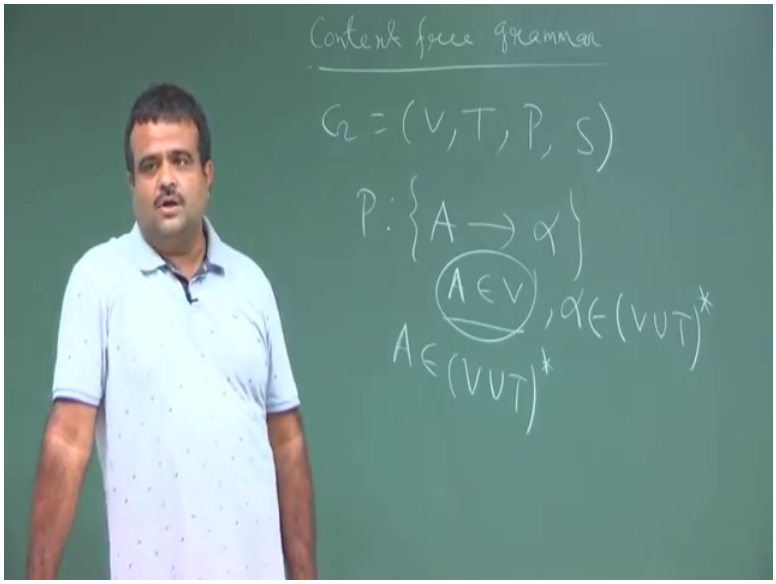
Introduction to Automata, languages and Computation
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 39
More on CFG

So we are talking about context free grammar and context free language.

(Refer Slide Time: 00:23)



So, just to recap; so, it is a 4 tuple, where V is the set of variable finite set of variable and T is the finite set of terminals, P is the productions rules productions and S is the starting variable which is belongs to V, it is a special variable which is called starting variable.

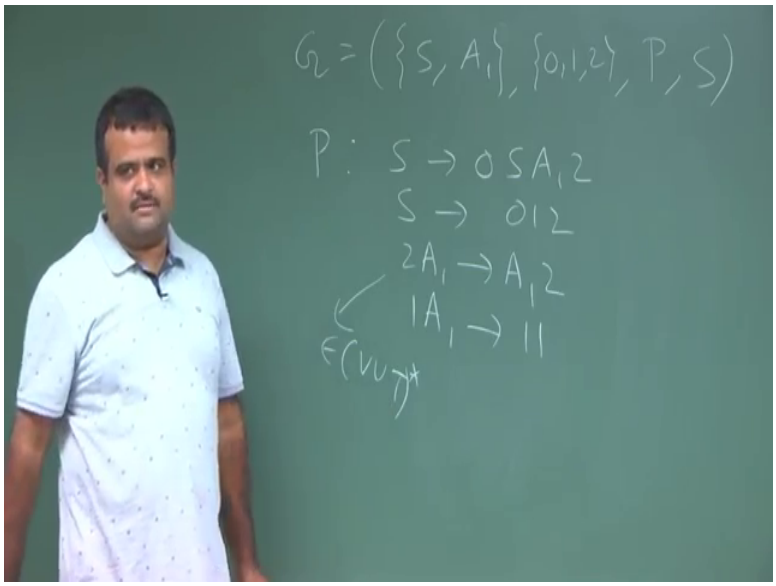
And then we have a derivation based on the production. So, if we say S. So, usually this V this derivation the production consists of this type of rules where A is a variable and we have one variable here in the left hand side and right hand side it is a string of either variable or terminals ok. So, production consists of such rules. So, we have many such rules are there in the in the set P ok.

And so, far we have talked about this A is a variable. Now we will take an example where A can be I mean this is the different type of grammars which we will not often refer where A consists of string also. So, instead of variable a consist A is also a string, A

belongs to consists of string. So, this is this is different type of grammar, but we will not often discuss about it I mean, but this has some applications also.

But for our case we will strict on A is a variable, but let us talk about beta on this.

(Refer Slide Time: 02:37)

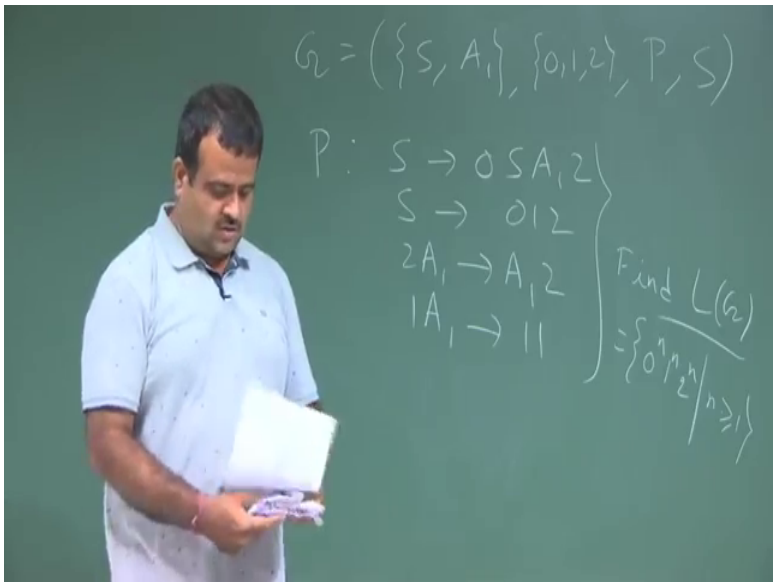


Let us take an example of this variant of context free grammar. So, suppose we consider this grammar G say we have 2 variable S and A 1. And we have this terminals 0 1 2 we have the productions and we have S. Now P consists of what? P consists of say S is rule is S is going to 0 S A 1 2.

No problem again another rule is no issue, but here this is 1 rule A 1 2; and 1 A 1 is going to 1 1. So, this is which we have not seen yet, this type of rules this type of productions. Because so, far we have seen the products on the left hand side is a single variable, but here left hand side is coming from is belongs to it is also a string consists of variables and the terminals.

So, this is a different type of grammar which we do not often discuss, which we which is, but which may occur I mean, but for our case we will only consider those which is this left hand side is only available. But anyway they just talk about which type of language it is generating.

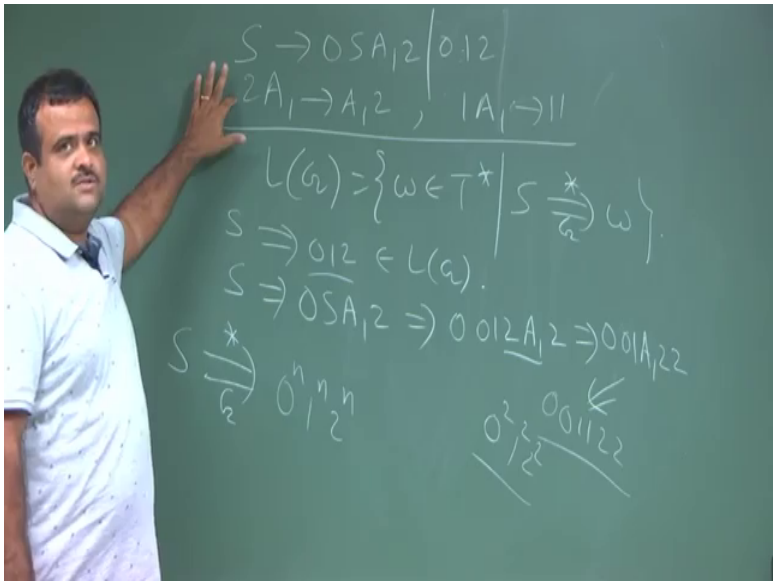
(Refer Slide Time: 04:19)



So, suppose this is our grammar, then if we ask find the language generated by this G ok. So, now we can see what type of language generated by G.

So, we will; so this language is nothing but 0 to the power n, 1 to power n, 2 to the power n where n is greater than equal to 1. So, it is basically all the strings of any number of 0s, any number of 1s and any number of 2s; so, that we have to show it has just ok.

(Refer Slide Time: 05:09)



So, let me write it here; so rules are S is going to 0 S A 1 2 rather that is 0 1 2 and no more. And another 2 rules are 2 A 1 is A 1 2 and 1 A 1 is going to 1 one now S is going to

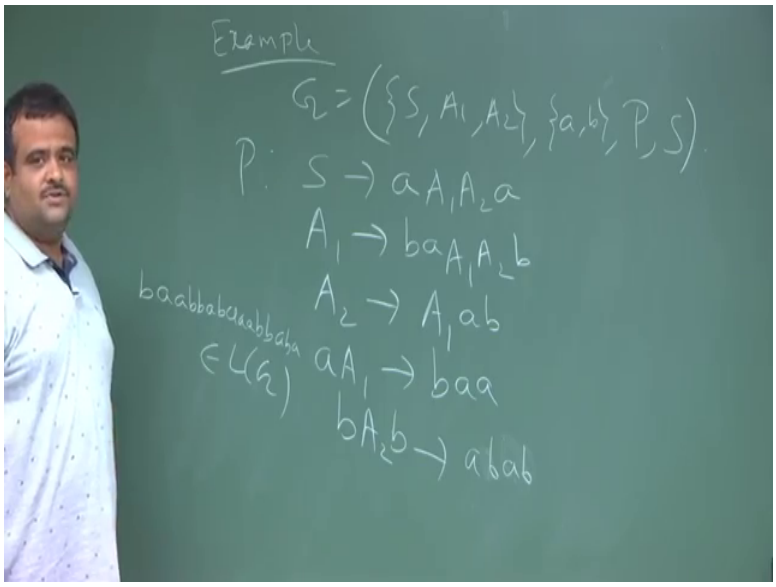
from. So, we are we want to find the string which are derivable from this. So, L of G is basically what? L of G is set of all string of terminals such that which can be derived by 1 or more steps by using the productions of G from S, that is that is our language so, you want to find this.

Now if we use this 0 S is going to 0 1 2. So, 0 1 2 belongs to L of G if we use this now if we use another rule 0 S A 1 2. Now here again if we use the say if we use this rule again 0 S is again going to 0 1 2 say 0 1 2 and then A 1 2 and 2 A 1 2 A 1 is going to say A 1 2.

So, this is again 0 1 no 0 0 1, and 2 A 1 is going to A 1 2 2 now this is now A 1, 1 A 1 1 A 1 is going to A 1 0 0 1 1 2 2. So, this is 0 squared 1 squared 2 squared. So, similarly we can saw any form of 0 to the power n, 1 to the power n, 2 to the power n can be derived from this. So, you can derive from S any form of this.

So, this is one example where we are using the as production rule in this form like we have the mixture like string, mixing with variables and the terminals, but which often we do not use that type of grammar yeah. So, we can take another example we can take another example say example of this type of grammar.

(Refer Slide Time: 08:23)



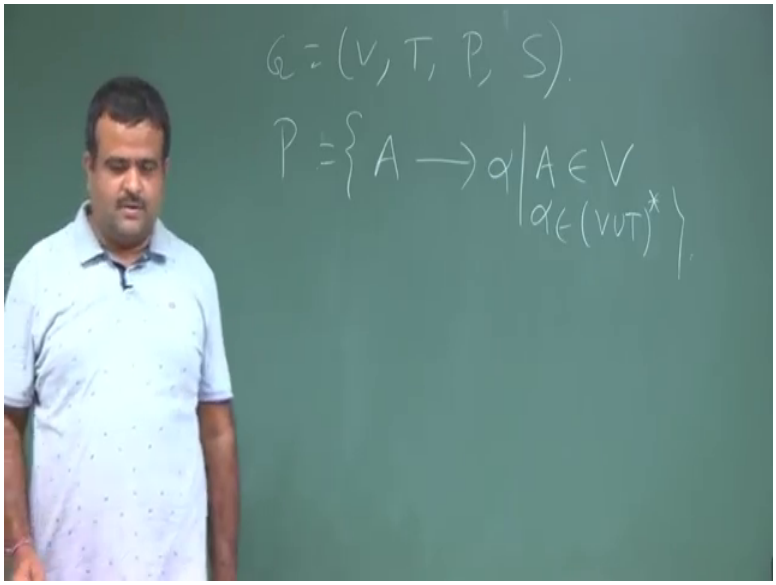
So, G is the variables are S A 1 A 2 (Refer Time: 08:34) and this is a b and this is P this is S. And the P consists of S is going to a A 1 A 2 a no problem so, for. A 1 is going to ba A 1 A 2 b and A 2 is going to A 1 a b and a A 1 is going to b a a and b A 2 b. So, this is the

type which we are which is mean to ask this is different type of grammar which is using the not only the variable is a using the string of variables and the terminals a b.

So, a b sorry ab ab ab ab; so these are our rules now we can easily check that this b a a, then b b a b then triple a a a a then b b a b a we can check this belongs to L of G; we can just follow this rule and we can derive. This is another example where we have this left hand side of the production is not only a variable, but it is also a string of variable and the terminal.

But this is different type of grammar we will not go on that. So, we will just strict ourselves on the grammar, where this we have only the variables in the production in the left hand side. So, like V T P S or productions are of the form.

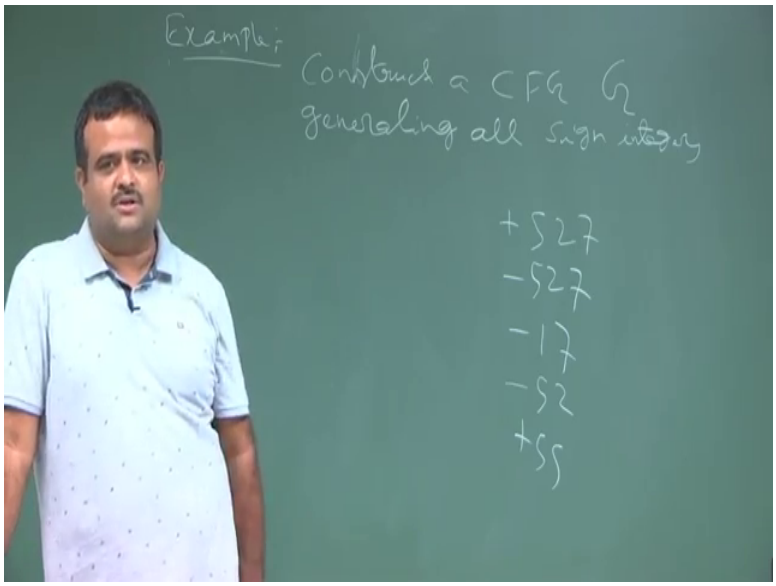
(Refer Slide Time: 10:47)



All the rules like A is going to alpha A is variable its not a stream of variable and terminals and, but alpha is a string of variables terminals, we will will we will consider not this type of grammar ok.

So, now let us talk about more on this context free grammar. So, suppose we want to construct a grammar.

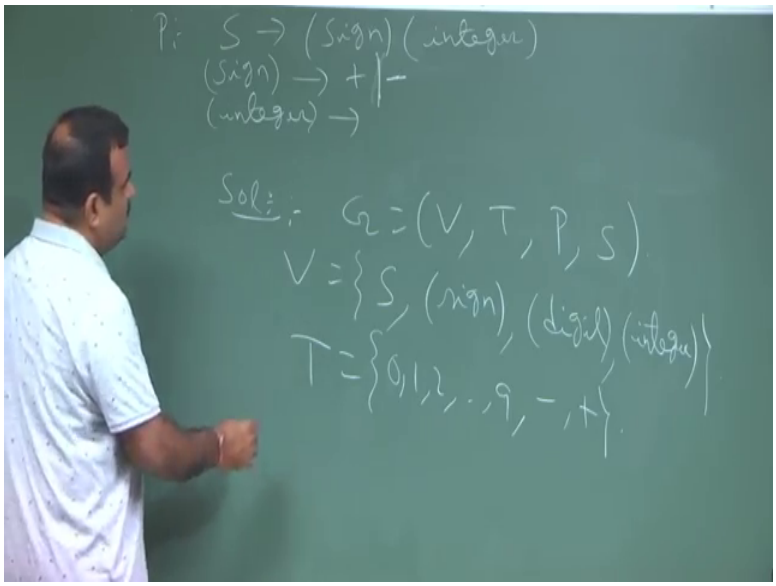
(Refer Slide Time: 11:31)



So, one example, suppose you want to construct a context free grammar G, which is generating all the integers all sign integers what do you mean by sign integers? Sign integers means it is say 5 2 7 integers, this is no sign is there it is means plus, then minus 5 2 7 this is another integer then minus 17, minus 52 plus 55 this is one integer even I can have the symbol plus.

So, this type of integer we want to get, we want to have a grammar which in which can whose language is which can generate this type of integers I mean the language is it can generate all the strings of this form. So, how to generate that? So, you have to have a rules. So, you have to have a grammar to do that.

(Refer Slide Time: 13:03)

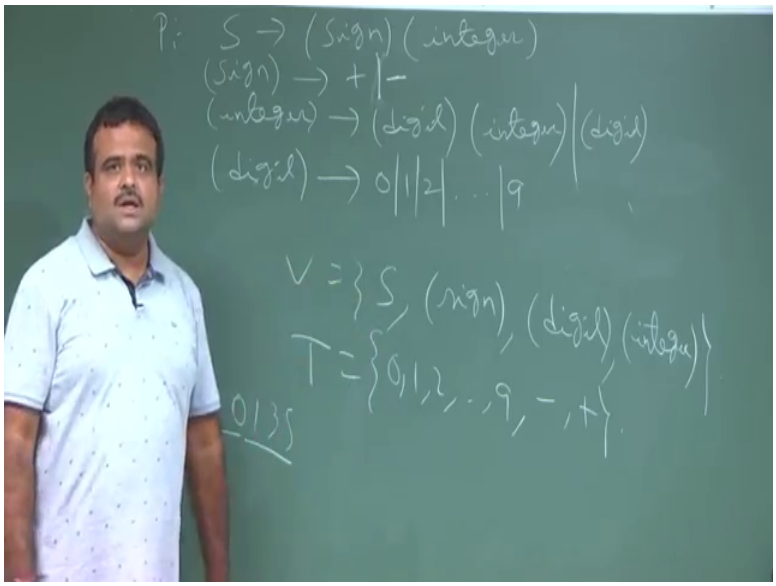


So, let us try to construct a grammar. So, let G V V T P S, where V is S is the starting variable.

And we have another variable called sign and we have another variable called digit and we have another variable called integer ok. And the terminals are basically what? It will eventually go to all the combinations of the digits and the and the this plus or minus in the prefix. So, it is all the digit 0 1 2 up to 9 and then minus 1 plus these are all terminals and the rules the productions P.

So, P consists of like this, S is going to sign or an integer and the sign is going to sign is either plus or it will go to minus there are 2 rules over here plus and minus and the integer is going to. So, integer is either a integer; that means, it is a positive integer or it have a digit then integer sorry it is digit or it can have say if it is single digit then it will go to a digit or if it is a more digit say 0 1 3 5.

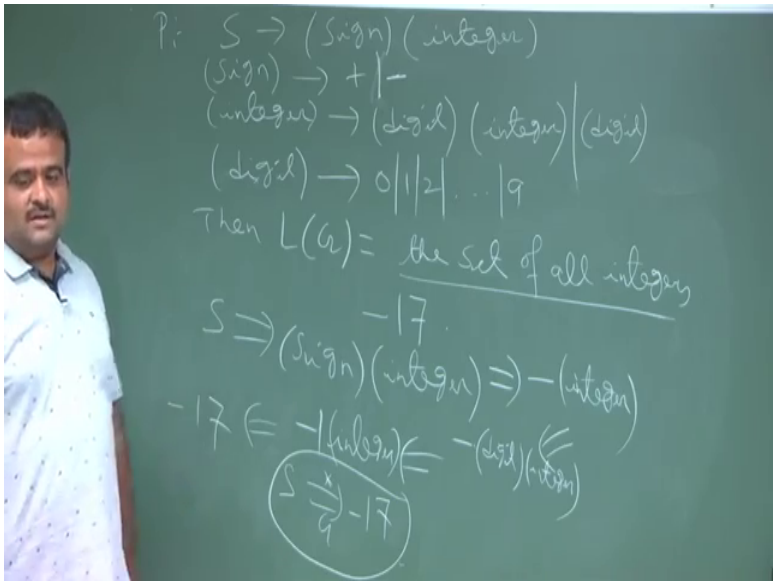
(Refer Slide Time: 15:15)



So, this is a digit then we must have a integer to get these digits. So, this is digit followed by integer or just integers sorry just a digit ok. Now where is the digit will go? Digit will go to either one of this 0 1 these are the rules 2 9. So, digit can go to 0 this is 1 production, digit will go to 1 another production is illegal to do like this digit will go to n ok.

So, this is our grammar this is our productions now what is the language exception generating by this grammar? So, we can.

(Refer Slide Time: 16:21)

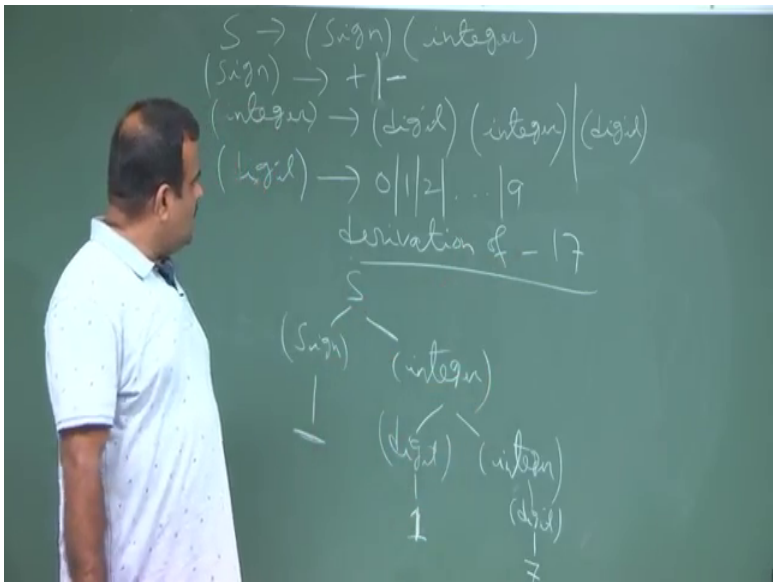


So, L of G is basically all the sign integers the set of all integers positive or negative theta all integers ok. Set of all integer can be generated by this grammar say for example, if we have an integer say minus 1 7, how to generate this from S from S what we can do? From S we can just take the sign and integer.

And now the sign is we can put minus and then an integer, an integer we can go like this digit an integer because we have 2 two digit. So, we have. So, this is 2 step. So, we can take this as integer, then this integer can be minus of digit then another integer, and then this digit may be 1. So, minus 1.

And we have an integer, and now this integer you can have a 7. So, minus 1 7; so this S is. So, S is. So, minus 1 7 is derived by this production of G from S. So, minus once again is an is generating from this grammar. So, this is belongs to L of G. So, not only minus 1 7 any integer positive or negative can be generated by this. So, now, we will draw a sorry this should give integer digit yeah now we will draw what is called derivation tree ok.

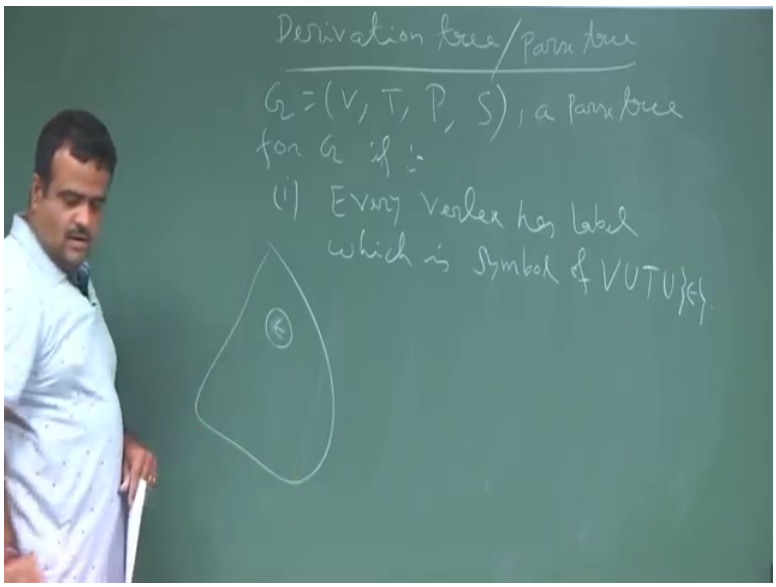
(Refer Slide Time: 18:47)



So, we will just try to draw the derivation of minus 1 7. So, S. S is going to where S has sign and the integer. And this sign is going to either plus we will take the minus 1 and the integer is again going to we will take this rule digit an integer again and then this digit we can go to 1 and this integer again we can go to digit and this digit is going to 7 this digit is going to 7. So, this is the if we read this leaps it is minus 1 7.

So, this is what is called derivation tree we will talk about more on this, derivation tree or it is called parse tree. So, this is the parse tree for the string minus 1 7 ok. So, different string is having different parse tree even for a given string we can have 2 parse tree. So, that is a ambiguities there we will talk about more on this. So, let us just define formally what do you mean by parse tree. So, you can have any say if we have say plus 1 7. So, we will have this symbol to be plus like this. So, any integer can have a can be derived from this S.

(Refer Slide Time: 20:49)

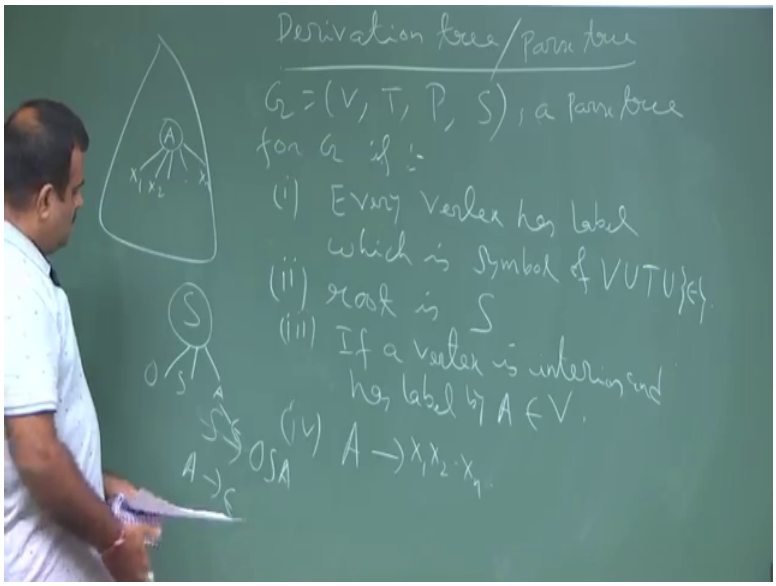


So, let us formally define the derivation tree or the parse tree for a grammar this is also called parse tree.

So, just now the tree we have seen that is called parse tree or the derivation tree for a given (Refer Time: 21:10) for a given the string ok. So, formally the parse tree is a tree, suppose you have given a gummer. A parse tree parse tree for G is a tree if this these are satisfied what is this. So, every node is a every every vertex. So, if your stream is it has a node or vertex is every vertex is either a variable or it is a terminal or epsilon if epsilon is rules is there. So, every vertex every vertex or node of the tree has level, which is symbol of either it is variable or terminals or it is a epsilon ok.

So, it is a tree this is a tree where every vertex is say either a where a is a either a variable or it could be a terminal say alpha or it could be epsilon every vertex of that tree.

(Refer Slide Time: 22:53)



Now, it start with the vertex S. So, the route is route is the sim vertex S I mean the starting symbol because we start the tree with S if we take a rule of this. So, it will come to that ok.

So, now this third S. If a vertex is if a vertex is interior and as label by A as label by A then a must be a variable. So, only the interior vertices are the variables and all the leafs vertices are the either epsilon or the terminals label by A then A must be a variable all the leaf vertexes are our terminals because once we reach to the terminal, then we have no more production because there is no rule which is good from terminal to something.

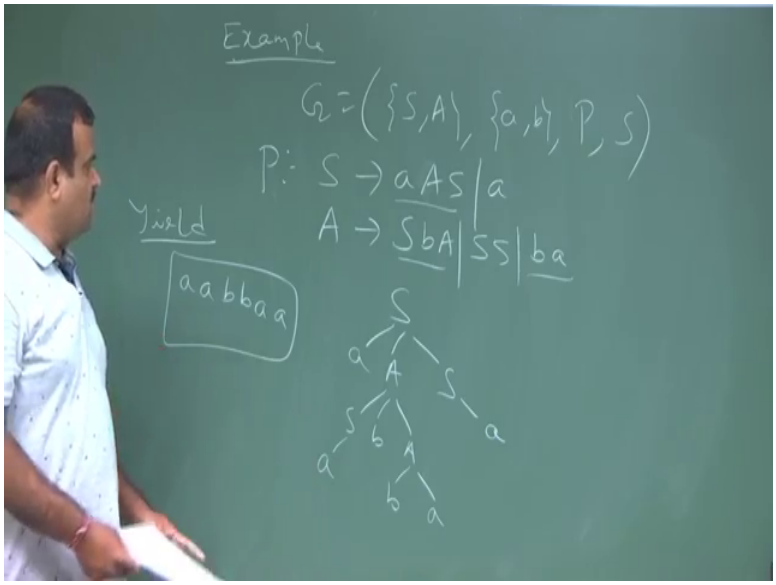
So, we stop at the terminals. So, so that. So, before that the interior vertex all the interior vertex are just label are the variables ok. Now if we have a. So, if we have a rule like this X 1 X 2 X n n and in a tree if a A is there, then the the child of these are basically X 1 X 2 Xn these are the child of this variables because this is coming from the rule if we have a rule then we have the branches of that subtree.

So, this is a subtree rooted by A. So, for S for the starting, but this is our starting vertex. So, what is the we take any rule from S. So, suppose S is going to say AB or say 0 S A then we can have 0 SA. So, these branches are basically the productions.

So, this is the way we construct this and the vertex has level epsilon, and the leaf nodes are once is reached to a terminals or epsilon. Epsilon means if we have a rule say a if we

have a rule, A is going to epsilon then we can have epsilon over here. This is the called parse tree you will take a quick example on this.

(Refer Slide Time: 25:57)



This is called derivation tree on the parse tree ok. So, suppose you have a grammar like this S A there are 2 variables and say a b these are the terminals p and S. And suppose P consists of the rules S is going to a s sorry A or a and A is going to S b A or SS or ba. So, this is the now suppose we want to construct a parse tree or the derivation tree. So, we have to start from the root S then we can take any one of this rules to have the their branches the child.

So, if you take this one, we have a A S ok. Now a A has no had to go because a is a terminal we stop now this can go this can go we can take any one of this which one we take that we are taking say this rules. So, this will go to S b A and suppose this is we are stopping at a small a. And this S is say we are stopping at a, and b we have no had to go and a say we are taking this for this a; so b a.

This is an example of a parse tree or the derivation tree now what it is yield? It yield the string of terminals like this we read like this from the left hand side a a aa b b a a. So, this is yield by this parse tree. So, this is this yields this string ok. So, for this string we have this derivation tree. So, if you have another derivation tree it will yield another string. So, this string is yield by this derivation tree. So, this is an example of derivation tree. So, we will talk on more on the derivation tree or the parse tree in the next class.

Thank you very much.