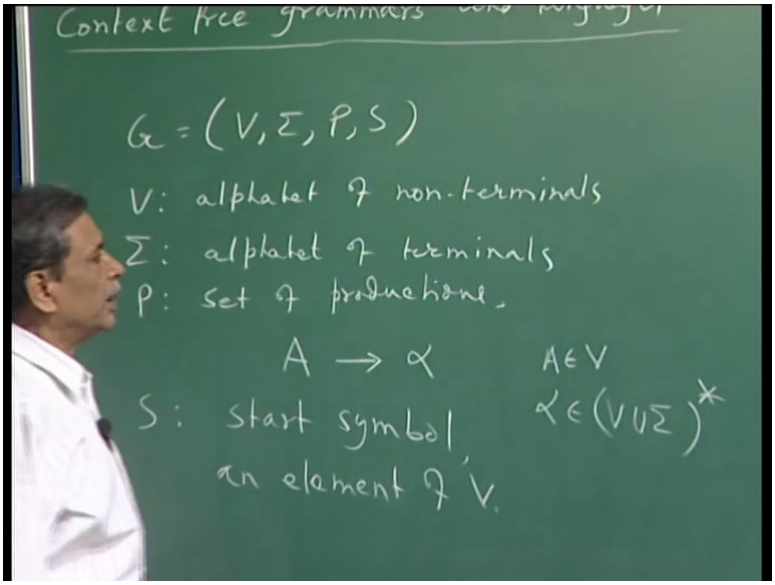
Theory of Computation
Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 21
Languages Generated by a CFG, Leftmost Derivation, More Examples of CFGS and CFLS

We will start with a very quick recap of what we did last time. Recall, we defined first what are context free grammar is and we said a context free grammar G as four components, V, sigma, P and S where V was what we call is a finite alphabet. V was the alphabet of nonterminals, sigma was the alphabet of terminals and P was the set of productions where each production is of the form that there is a left hand side, an arrow and a right hand side where this left hand side has only one symbol and that symbol is an element of V.

So essentially you take anonterminal and you read it as to be replaced with or can be rewritten as alpha where alpha is a string over V union sigma star. What it means is that right hand side can be a string over terminals and nonterminals. And fourth component S is a special symbol from V what we call the start symbol or start nonterminal if you wish and which is an element of V.

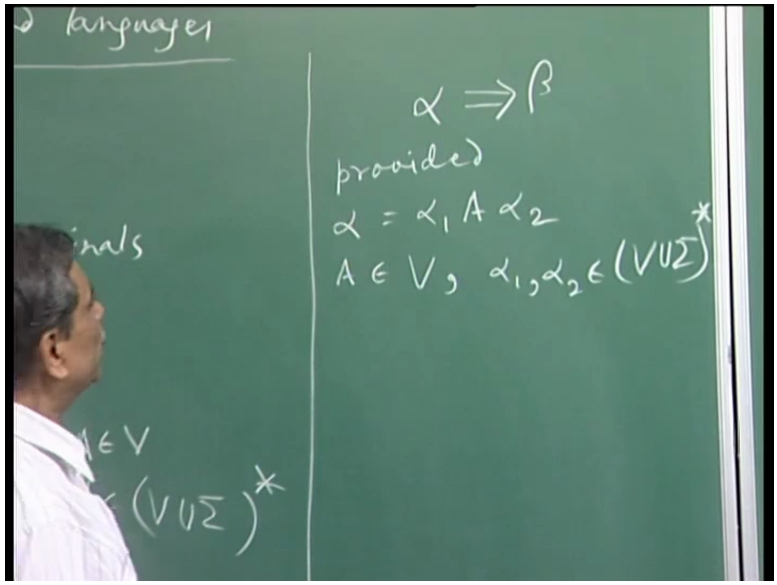
(Refer Slide Time: 02:39)



Then what we defined was a notion of derivation and for that first of all we talked of something we used this notation. We said that alpha derives in one step betaif this will hold

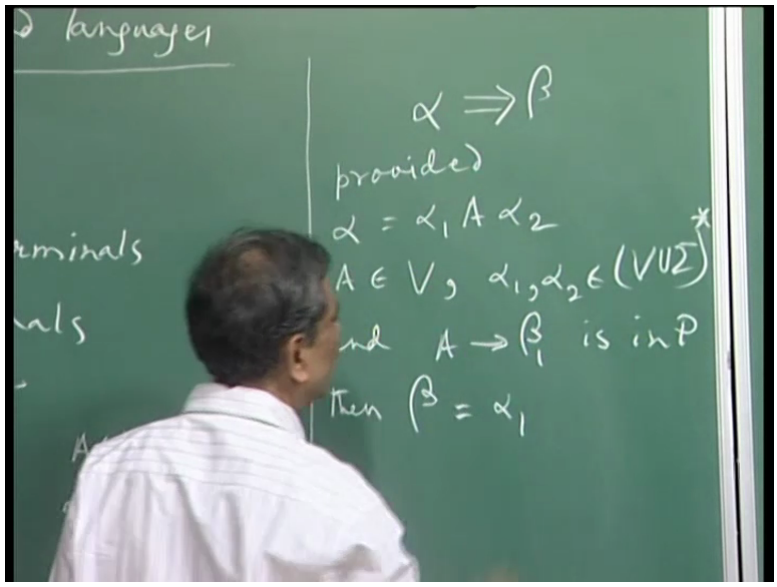
provided alpha is of the form let us say alpha 1 A alpha 2 wereA is a symbol from V, alpha 1 and alpha 2 are strings over V union sigma, right?

(Refer Slide Time: 03:39)



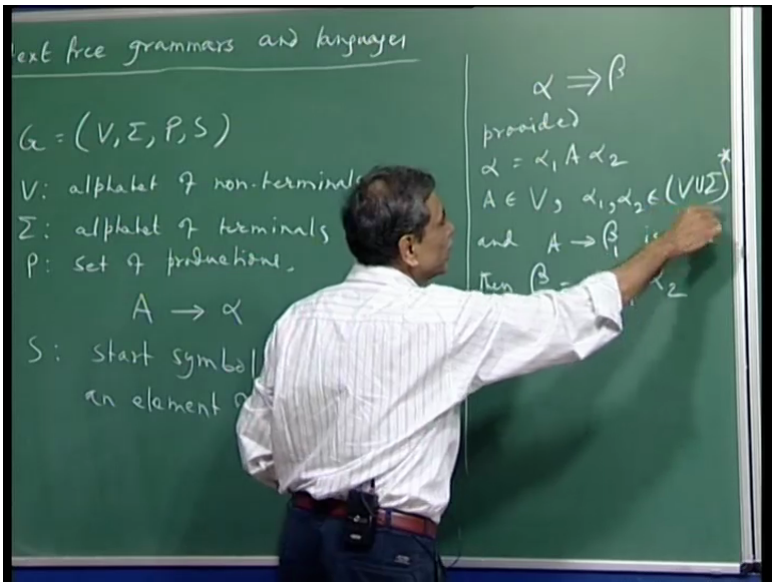
And beta then, so basically this alpha is some string over V union sigma terminals and nonterminals where at least one nonterminal must be there. So suppose call it A and A goes to beta is in P. Supposethis is one first that alpha is of some of this form. If you can write it as alpha 1, alpha 2 and A goes to beta is a production.and then beta is of the form this alpha 1. I should have said beta 1 because this beta and this beta we should not get confused.

(Refer Slide Time: 04:36)



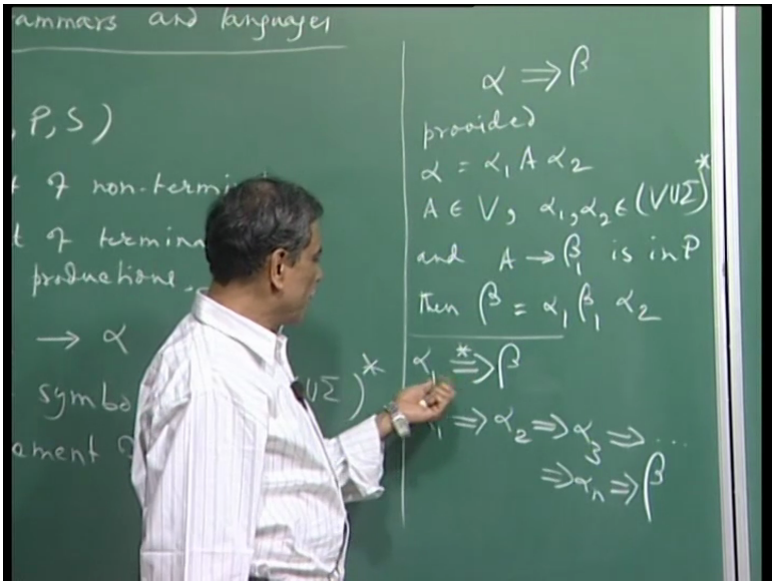
So alpha 1, then this A where this A is replaced by beta 1 and alpha 2. So basically we were saying that these two strings alpha and beta can be related in this manner that alpha in one step derives beta provided alpha has this form where the nonterminalA is you are replacing with the right hand side of the production of that nonterminal. And then the string that you get is obviously the string beta. So notice that this is really a binary relation over this set. Set of all strings over V union sigma.

(Refer Slide Time: 05:26)



And then we said that sigma star will hold between two strings. So let us say alpha 1 and alpha 2 if I can find alpha 1 in one step goes to alpha 2 goes to alpha 3 and so on and maybe up to alpha n and then this alpha 1 in one step, let us say this is beta, okay. And this relation again is over strings over V union sigma. So this is also binary relation which relates to strings.

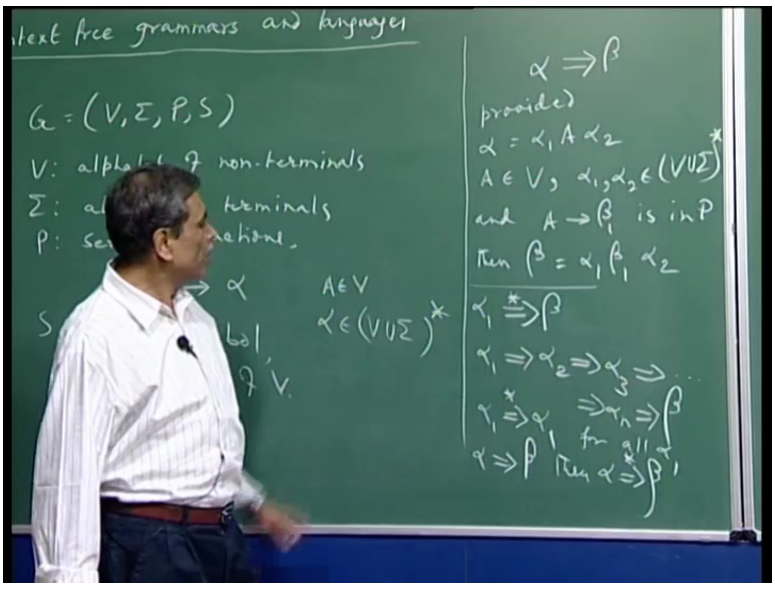
(Refer Slide Time: 06:29)



And such a relation will hold between two strings provided the left hand string after a series of this one step derivations lead to the string which is in the right hand side. So therefore this particular relation is called derivation in zero or more steps. And here of course there it is more than one step that is how I have shown it. But this is a particular case, the other cases have that we will see that alpha 1 also relates to alpha 1, this is a zero step.

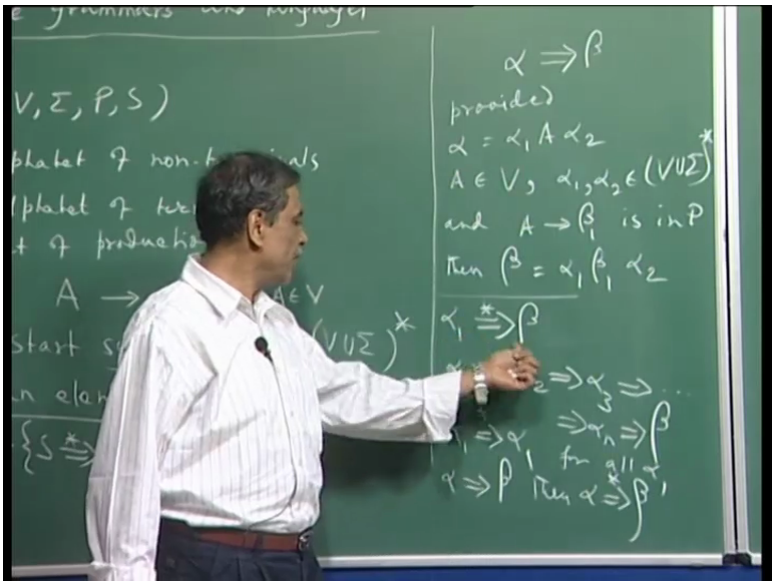
And in one step is this itself. So this holds for all alpha and then if alpha in one step goes to beta then also we say alpha in zero or more steps goes to the same string beta. And this is the more general case that you are using more than one steps to go to the right hand string.

(Refer Slide Time: 07:59)



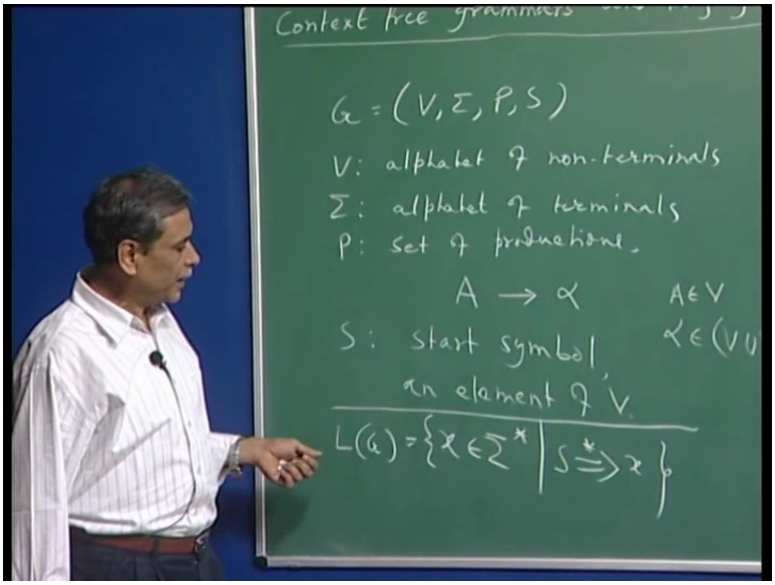
And once I have this notion then it was very easy for me to define what the language generated by a grammar G was. So which was L G for this grammar is going to be set of all terminal strings which can be derived. In general which is say this is derived.alpha 1 derives beta.

(Refer Slide Time: 08:27)



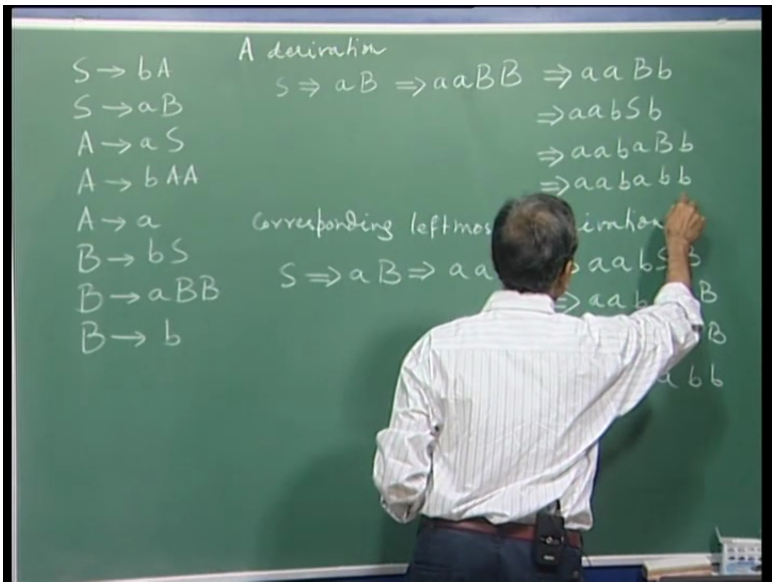
S derives I should have write it this way that L G is the set of all strings x in sigma star such that S derives this string x. In other words what we say? The language generated by the grammar G consists of all terminal strings where you see x in sigma star. Sigma is the set of terminal. So all terminal strings which can be derived from the start symbol S. So that is notion of the language generated with the grammar.You see once I define a grammar then that uniquely defines a languagewhich is over sigma and that is this language.

(Refer Slide Time: 09:32)



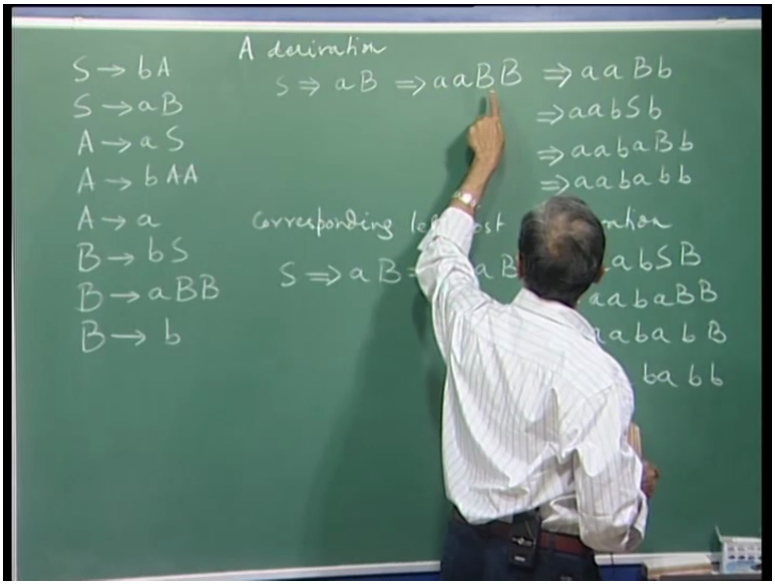
Last time we gave examples of three grammars and in the last grammar was you recall this was the grammar and I showedat least one example of a derivation. Now here is an example of a derivation using this grammar. So you can see that from S I am using this first rule to compare and then this B is rewritten as with this particular production and so on. And finally I am getting this string.

(Refer Slide Time: 10:15)



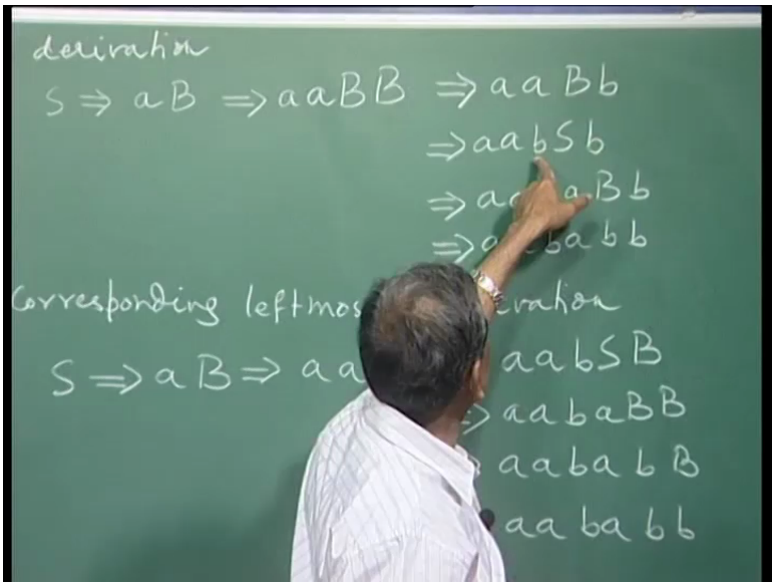
So therefore this string is in the language of this grammar. Nowand I told you that time thatwhen we have a stringstarting from S when you get a string something like this then consider this.

(Refer Slide Time: 10:38)



If you have two nonterminals in here, right? Here first I chose to rewritethis particular nonterminalB, you know you can see that this B is rewritten as small b and thenthis particular B was rewritten as b S.

(Refer Slide Time: 11:06)

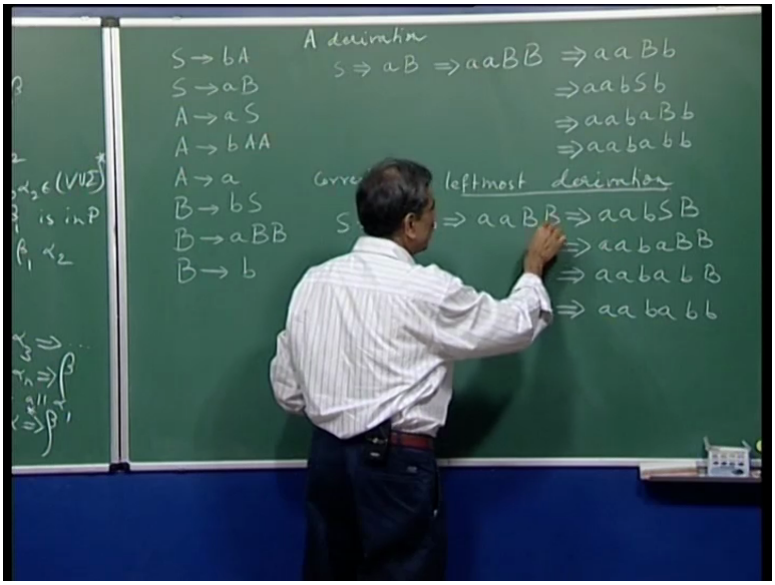


I can derive the same string by always following one particular convention that whenever I have a number of nonterminals in my string that I derived so far, there I will always choose to rewrite the leftmost nonterminal. Such a derivation is called leftmost derivation, okay.

And this same string using the same production, but if I now use this convention that whenever there are more than onenonterminals, the first I will always rewrite the nonterminal

which is leftmost in the string that I have. So in this case I have two nonterminals. I will rewrite this particular B and not this particular B.

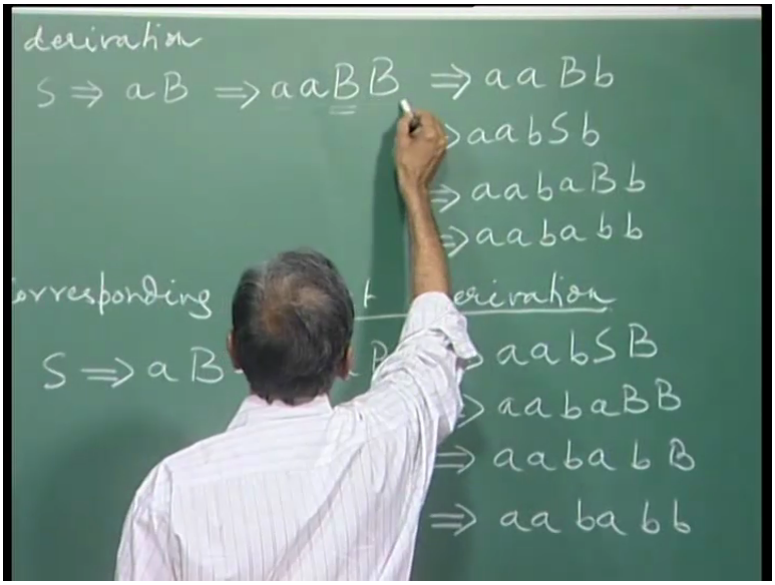
(Refer Slide Time: 12:19)



And it is not too difficult to see if you think about it that if I can derive a string at all then that string will have a leftmost derivation. Now why it is happening? The main reason for that is the reason why thisgrammar is called context free grammar. Whywe call it context free? We call this grammar context free because you see when I choose to rewrite a particular nonterminal, the production that I use need not consider at all the context in which that particular nonterminal is occurring.

So every nonterminal is rewritten independent of the context. Now what is meant by context? For example this particular is occurring in this context in thatthere is you know some a in front of a and something after it or you know whatever you can see this is, the left I have this, the right I have this.

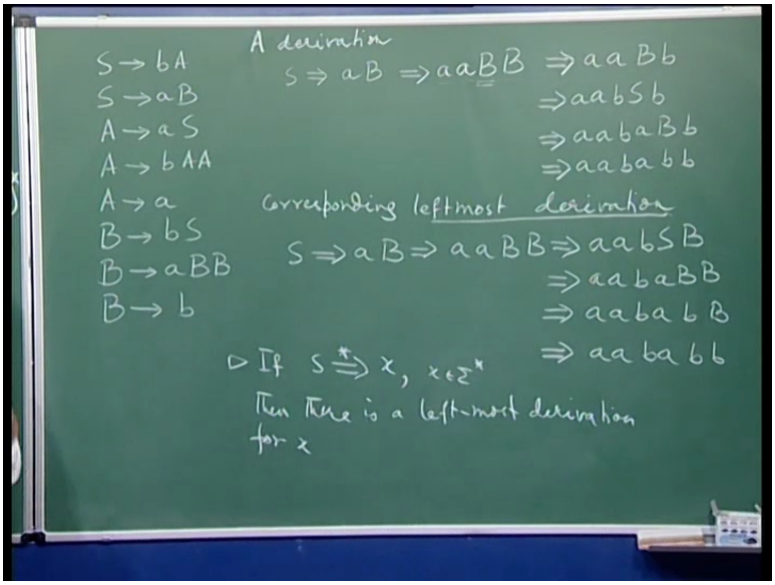
(Refer Slide Time: 13:44)



So that is the context in which this particular B is occurring in the string. Now clearly the way we have considered our grammar, my production has rules of this kind, right? Nonterminal to be rewritten by something else. That is the right hand side of the production. It does not say anything. It does not constrain me in any mannerhow that nonterminal is occurring, right? And that is why such a grammar is called a context free grammar.

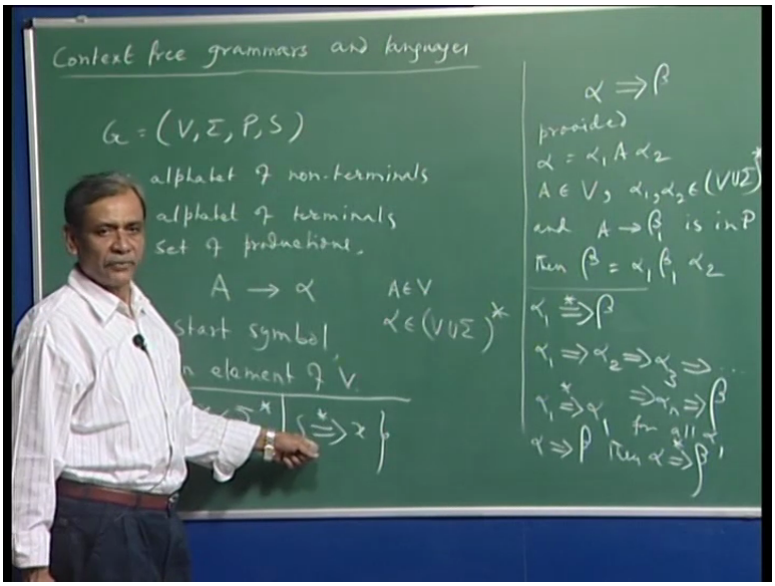
And when this particular liberty is not there then we get a grammar called context sensitive grammar. That is a separate topic altogether. And as I said because of this nature of our productions that left hand side is always a single nonterminal and that is the main reason why I can claim , let me write it, that ifS derives x thenwhat I mean is this xis in sigma star then there is leftmost derivation for x. This is true of all context free grammar. Not particular to this exampleas you can see.

(Refer Slide Time: 15:46)



So essentially here I could have said that and in particular I could have even constrained it saying that this derivation is a leftmost derivation. I will not change the language in any manner.

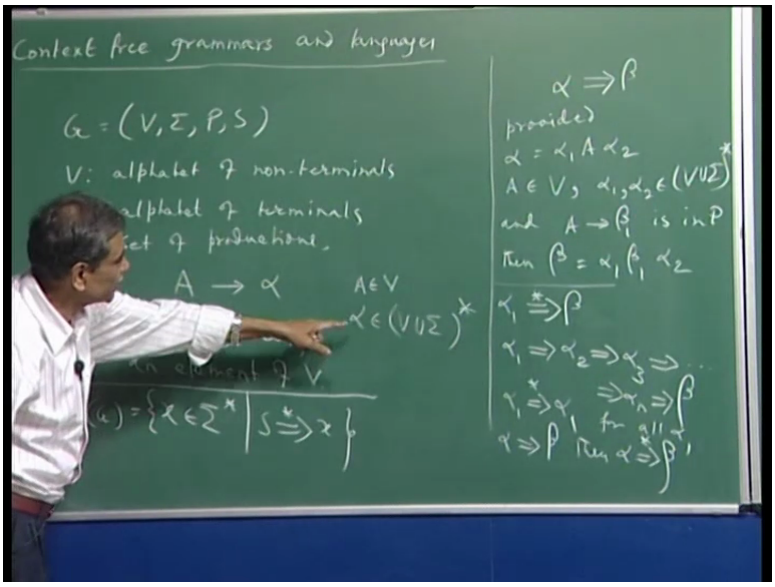
(Refer Slide Time: 15:55)



Now one more thing we should notice or now I have been using convention which is always generally followed that capital letters likecapital A, capital B, capital S, etc. these are used for nonterminals, right? And small letters from the beginning of the alphabet they are usually used as terminal symbol, elements of sigma. So capital letters are elements of V and small letters from the beginning of the English alphabetusually these are considered to be elements of the set sigma which is the set of terminal.

And also I am using some Greek letters alpha, beta, smallcase Greek letters. These are strings which are over V union sigma. If you notice this is the kind of convention that I am following, right?

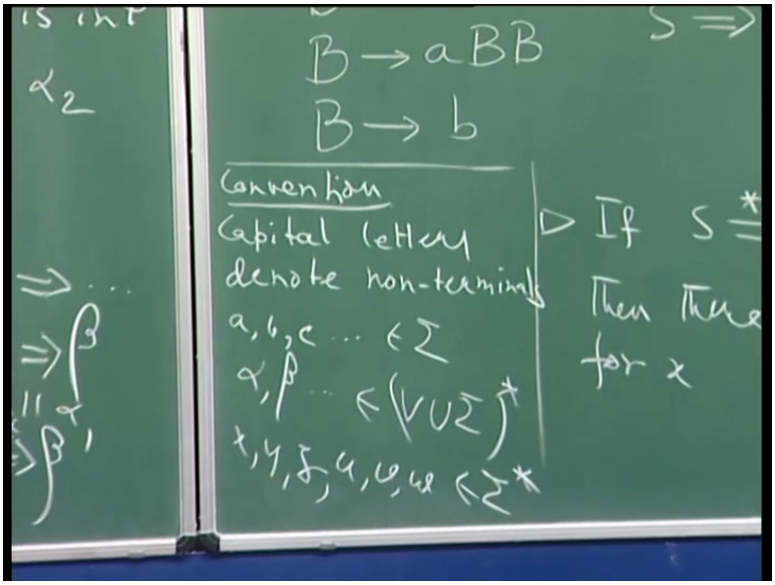
(Refer Slide Time: 17:09)



So letme write this convention here. Capital letters denote nonterminals that is these are elements of V. Second convention is letters like a, b, c, you know 0, 1, these are elements of sigma. Alpha, beta, etc. you knowlowercase (Gee) Greek lettersthese are strings and the strings are over V union sigma.

So that meansalpha is a string which can have both terminal and nonterminal and finally(fa) usually we say x, y, z, u, v, w, so small letters from the end of the English alphabet, these are typically used as terminal strings. So these are elements of sigma star, okay.

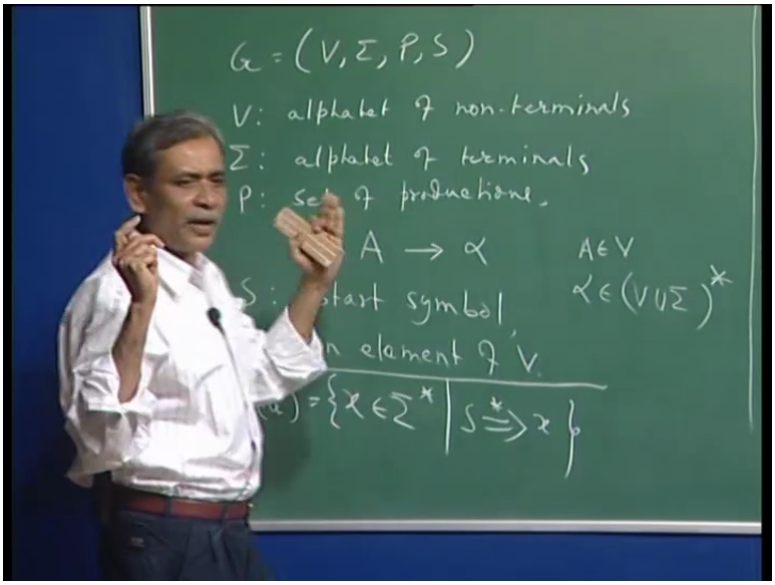
(Refer Slide Time: 18:50)



And that is the reason I am saying these arenot really very important. You need not follow but usually in the literature, in books this convention is there.So every time I write something I do not have to say whether it is a terminal and nonterminal. Unless otherwise specified we use this convention, okay. Now there are several things we need to you know go beyond this. We have defined something called a grammar. We have associated a language with a set of grammar.

Now one more thing you notice that this grammar is a finite object. In the sense V is a finite set, sigma is a finite set, P is a, I have not written it explicitly but these are again it is a finite set of this kinds of production rules and S is of course it is just an element.

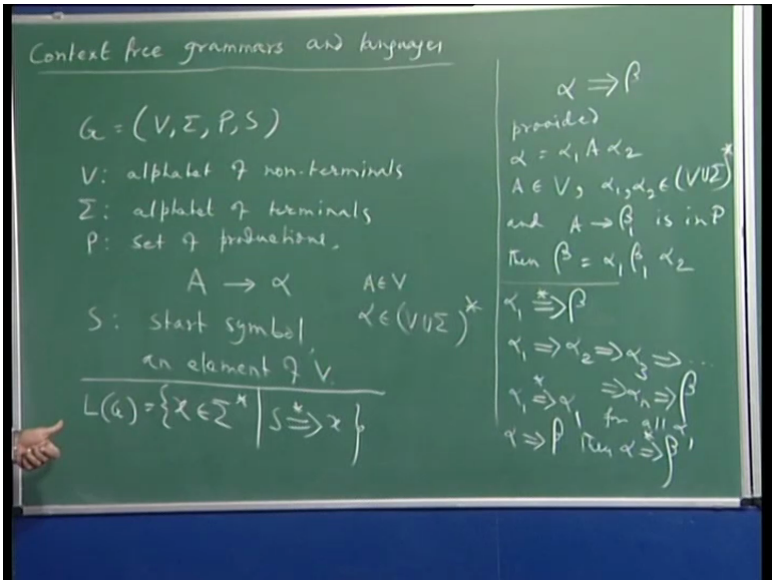
(Refer Slide Time: 19:55)



So G in itselftaking all its four components in account, is a finite object. However the language associated with G of course need not be finite. In fact in general L G language associated with a general context in grammar is going to be infinite. So we are back to our old concern. If you see what was our old concern when we talked of regular languages that I have this infinite set? How do I represent it finitely, right? And there we used several notions like we used a notion of a machine.

Machine again so finite strip machine was a finite object andwith such a finite strip machine we associated a language, right? And uniquely like here also if you see with a grammar G, what is the language associated or the language generated by the grammar G? That is unique once you specify what the grammar G is, okay.

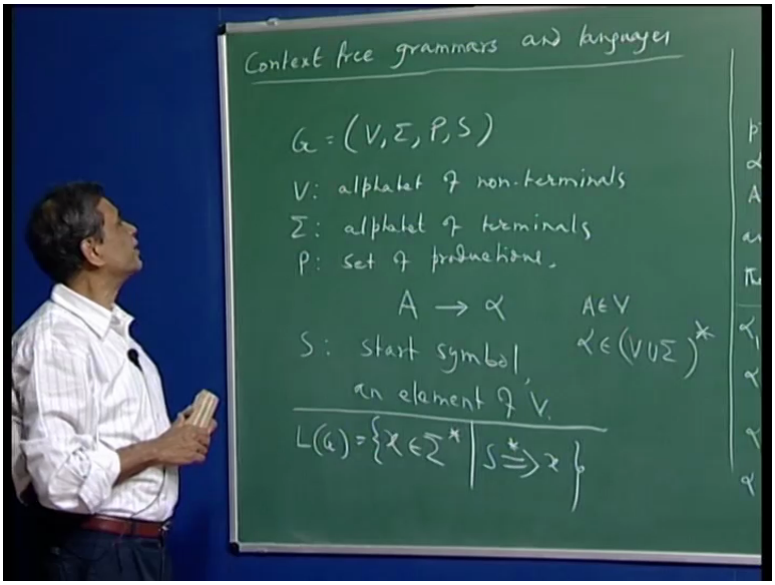
(Refer Slide Time: 21:14)



So instead of describing this language in whatever way you have, by in English or whatever, I give you this finite object G. And I said look, the language that I am talking of isa language generated by the grammar G. The way for a regular language I would have given you again a finite descriptionwhich could have been a machine, which could have been a regular expression, right?

So these are the ways we talked of or finitely specify that infinite object like a typical regular language. So in that sense again these object, these grammars, they are finite objects but they help us specify languages which in general could be infinite.

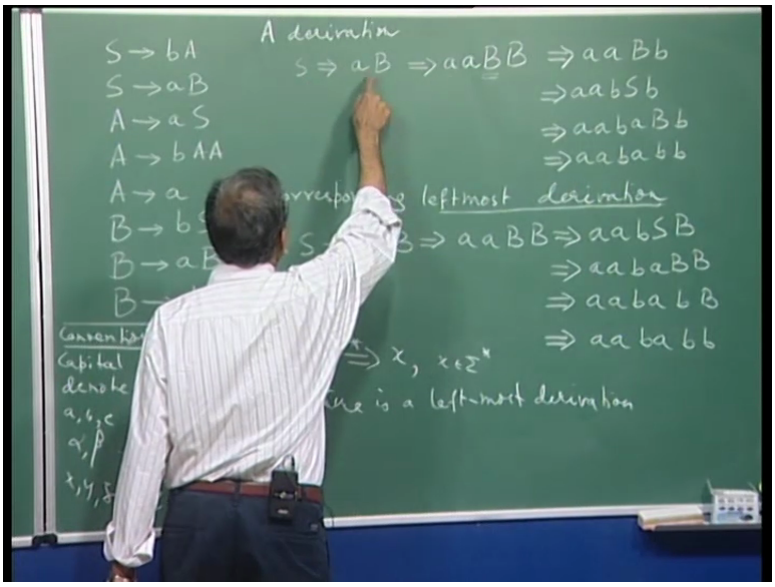
(Refer Slide Time: 22:01)



Now coming back to this (kno)derivation of course I can talk of a derivation or you know specify a derivation like this. There is a more convenient way of specifying or seeing how a string is derived in a context free grammar. And that is through what is known as parse trees. So let us now see what parse treesare? First of all parse trees is a graphical way of seeinghow a particular string is derived? In fact let us take this example.

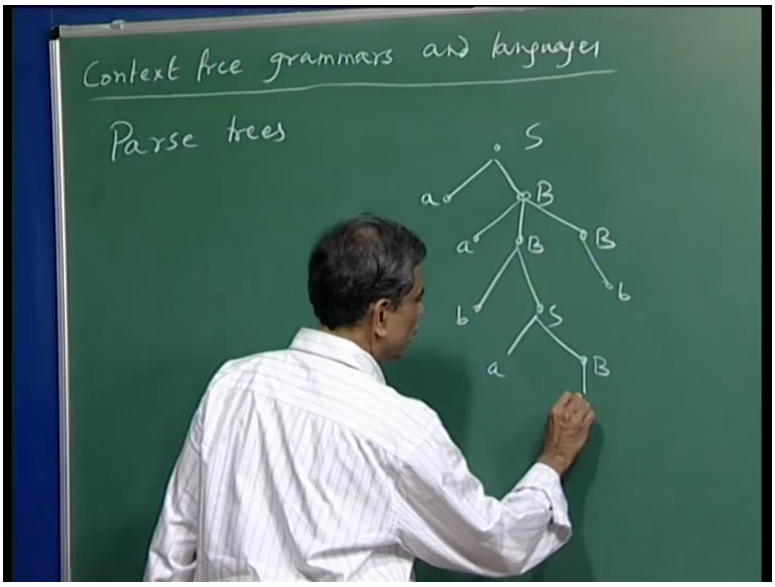
What we will do is let us start with S, right? And I look at it as the root of a particular tree which we are going to see. And then if you see in this derivation S was rewritten as a B.

(Refer Slide Time: 23:27)



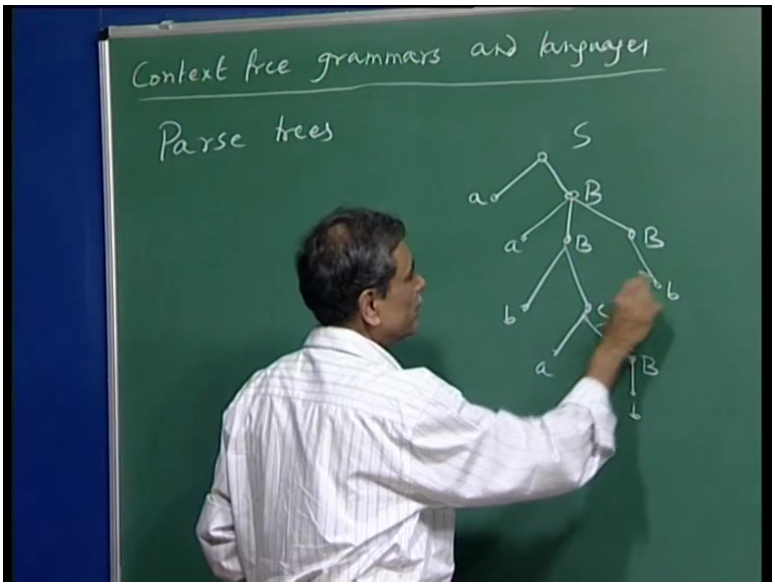
So what we will do is let me complete this and then we will discuss. So then this B was rewritten as a B B and then the last B, this B was written as b andthis B was rewritten as b S and S this was rewritten as a B and then this particular B was rewritten as b.

(Refer Slide Time: 24:44)



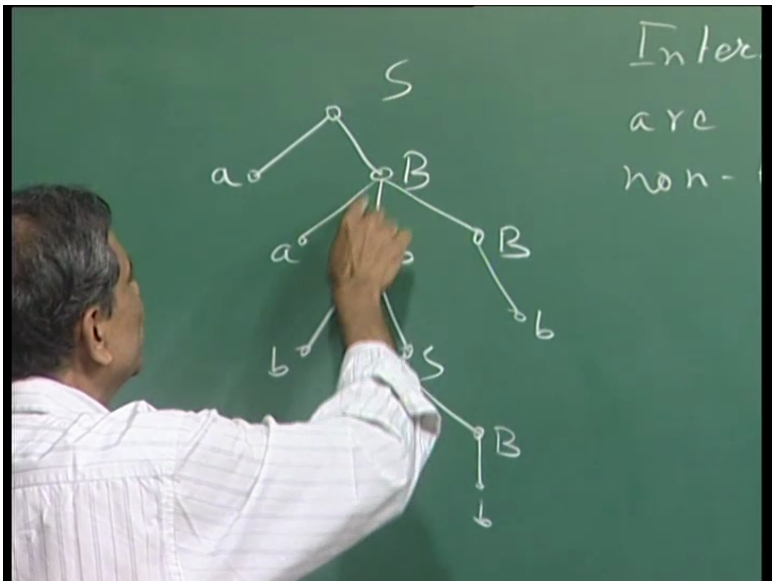
So this is the rooted tree. And notice this is, each nodeis level with a symbol. That symbol could either be a nonterminal or a terminal. Howeverif a node is labelled with a terminal symbol then it is a leaf node, right? In this tree the leaf nodes are this, this, this, this, this and this, right?

(Refer Slide Time: 25:25)



So this is a derivation tree or parse tree and in this parse tree the internal nodes are alwayslabelled with nonterminals. So because a node is internal, it is anonterminal and its children spell out a rewrite ruleor a production. So here you see that B this nonterminal, its children are if you look at it from left to right, a B B.

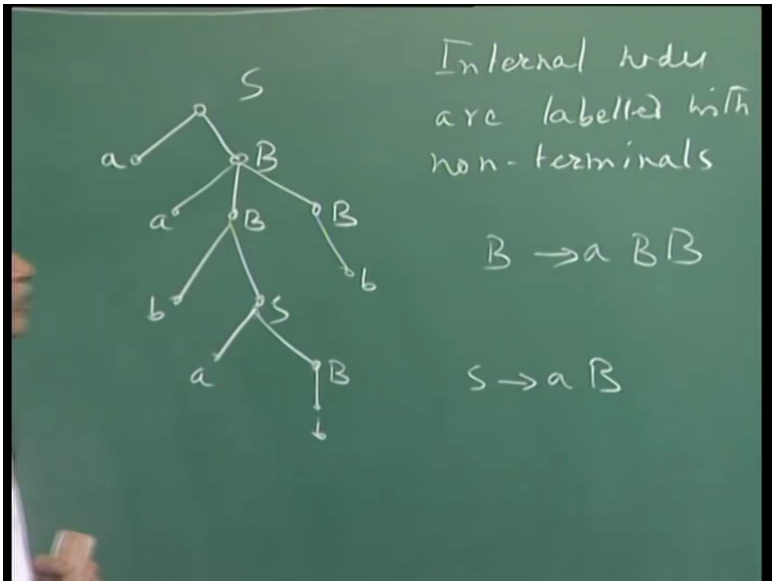
(Refer Slide Time: 26:24)



This nonterminal is B and this children are a B B, right? And noticed that this was one of the productions, okay. So another way of saying this thing would be that internal node will have a children. Thesymbols of the right hand side of a production whose left hand side is that symbol, okay. See that is what is happening. This is an internal node which is labelled with S. So this internal node it is labelled with S.

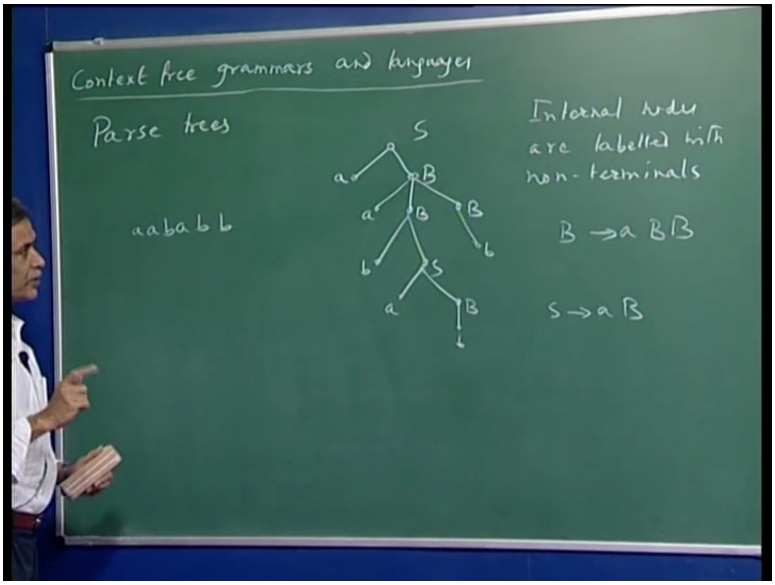
Its children as you read it from left to right or the labels of its children as you read these labels from left to right they must spell out the right hand side of a production whose left hand side is S. So here S goes to a B.

(Refer Slide Time: 27:32)



Indeed this is one of the productions that we have, alright? And what node the frontier of the tree? What is the frontier of the tree? The way you see and the way we know is basically readall the leaf nodes from left to right, right? Sowhen you read the leaf nodes from left to right they are going to be a a b a b b, right?

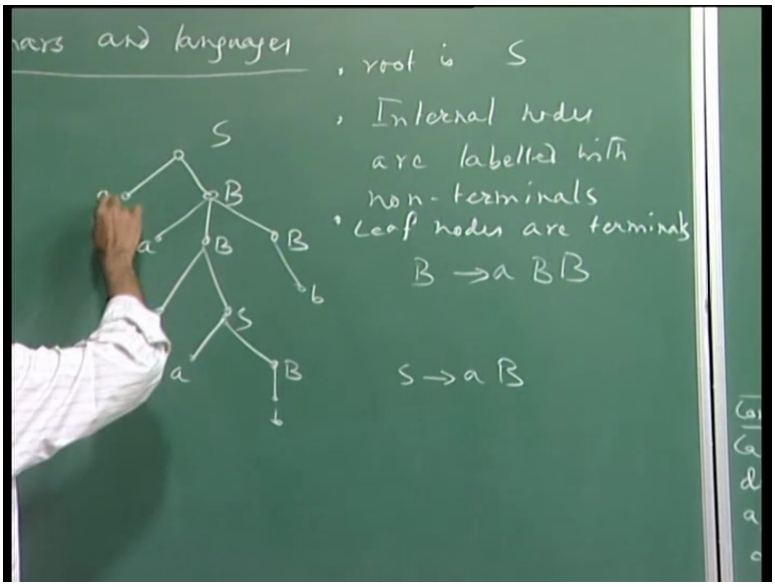
(Refer Slide Time: 28:08)



So first of all remember in a parse tree every leaf has to be labelled with a terminal, right? And an internal node has to be actuallyone of the nonterminals because its children will depend on the right hand side of a production starting with left hand side whose level was that particular node, right?

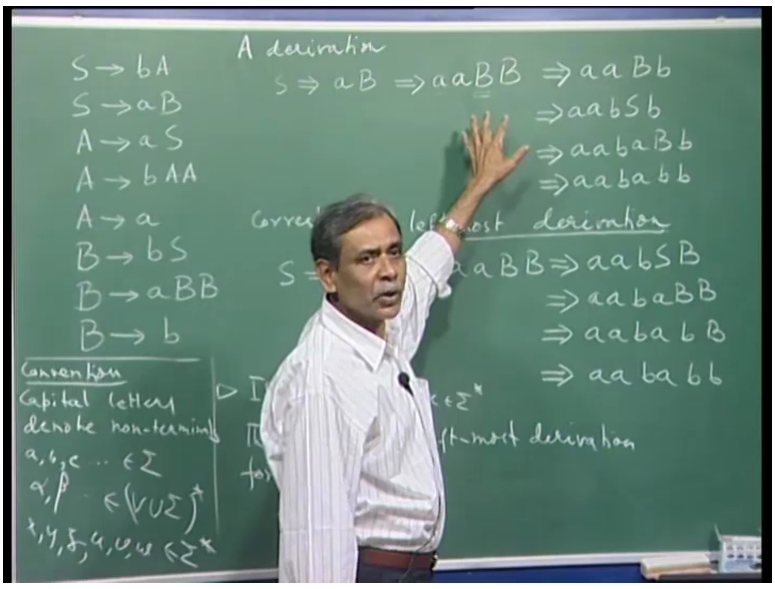
So, clearly as I said internal nodes are labelled with nonterminals. (Extern)or leaf nodes areterminals, right? The root is the start symbol S and I explain to you, let me not to write it that what is the relation between an internal node and its children?

(Refer Slide Time: 29:10)



Now if yousee this is clearly a graphical descriptionof a derivation. Now what about the leftmost derivation? Okay andyou agreed that this or this derivation, this was thecorresponding leftmost derivation. Corresponding in the sense that we use the same productions however whenever there was a choice of expanding one of the two or more nonterminals we always choose to rewrite the leftmost nonterminal.

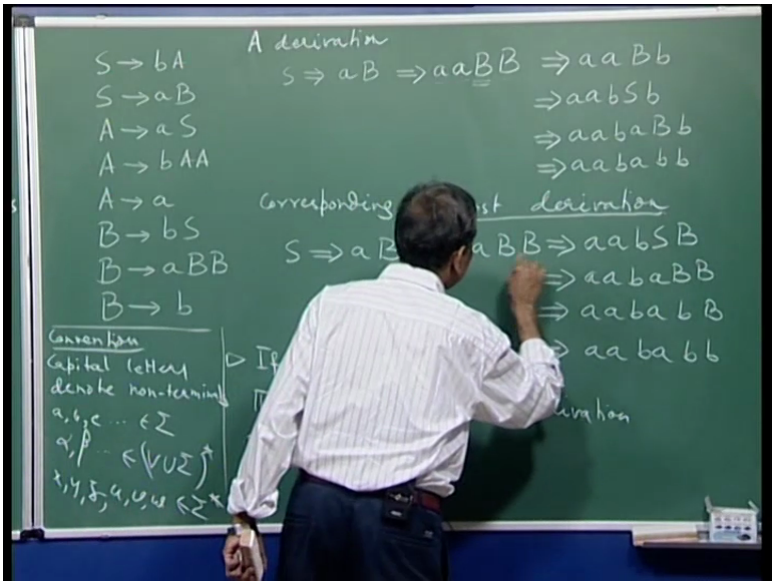
(Refer Slide Time: 29:53)



But we would have rewritten here the same way as that particular nonterminal was (re) rewritten in the derivation, okay. So in that sense it is a corresponding leftmost derivation and as I said that it is not too difficult to see thatif a terminal string can be derived at all, there it can be derived through a leftmost derivation.

Point I am trying to make is if youtry to draw the parse tree corresponding to the leftmost derivation we will get the same thing because S goes to a Bthat is fine. From this B was rewritten as a B B.

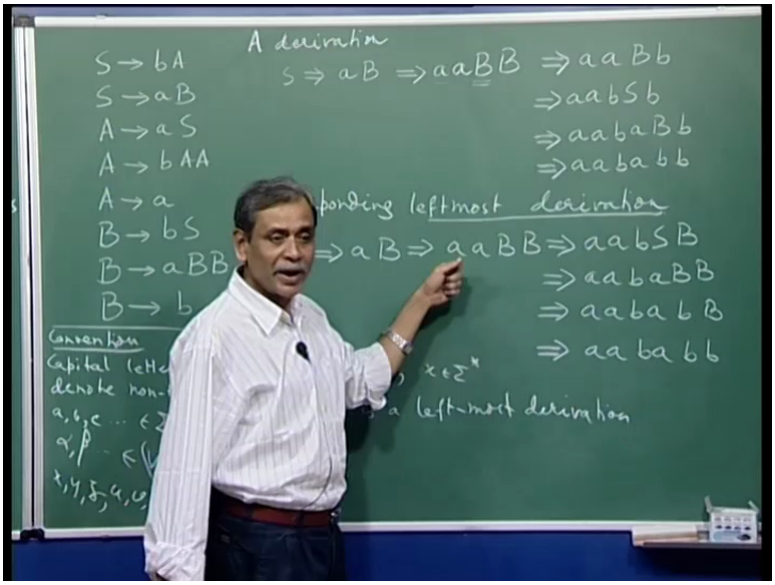
(Refer Slide Time: 30:40)



That is what you did here and so on. And then when you read out after all thenonterminals have been rewritten you will be left with the terminal string and if you read that I mean the frontier of the tree will be all terminals. And if you read that frontier that is the terminal string generated. So in a way you can see like there can be many derivations. Corresponding to this there is a uniqueleftmost derivation for a number of derivations and left most derivation and parse tree they are again kind of one to one, right?

If you give me a leftmost derivation I can give you a parse tree and if you give me a parse tree I can of course justgo through this parse tree and tell you what the leftmost derivation was. But you see given a parse tree I may not be able to say whether this was the derivation you actually did this derivation you did.

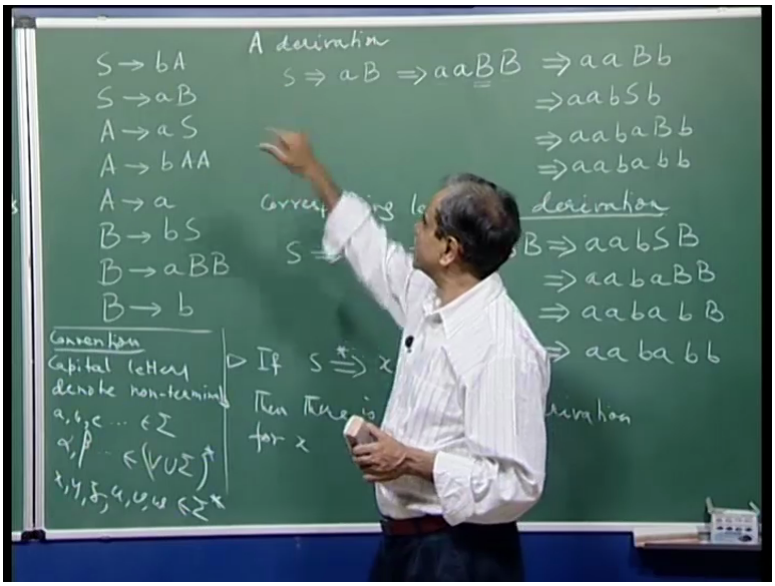
(Refer Slide Time: 31:54)



But both generate the same string and I will give you the leftmost derivation given the parse tree. So parse trees, leftmost derivations they are kind of equivalent. Either you give a parse tree or a leftmost derivation. We will most of the time deal with parse trees because somehowyou will see the intuition about context free language come out better once you considerthis.the derivation of a string can be seen as a tree.

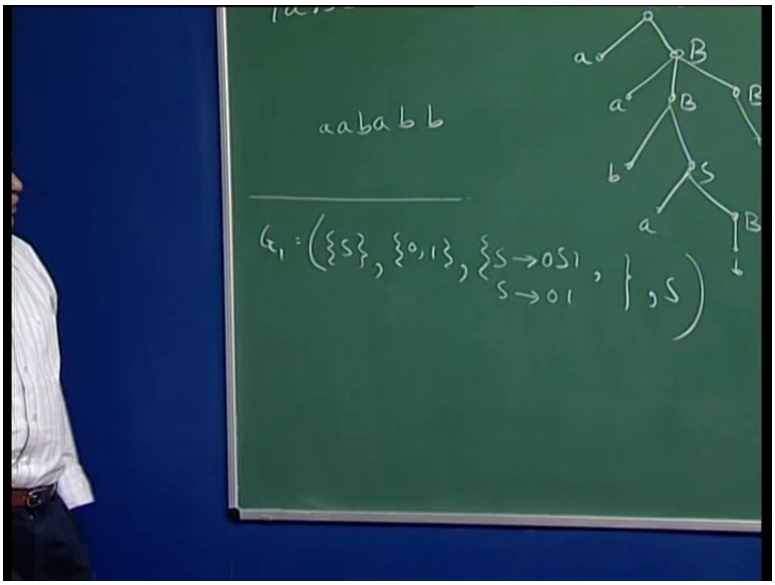
Now I want to spend a little time about the languages that we have generated so far using the grammars, right? One example was this.

(Refer Slide Time: 32:52)



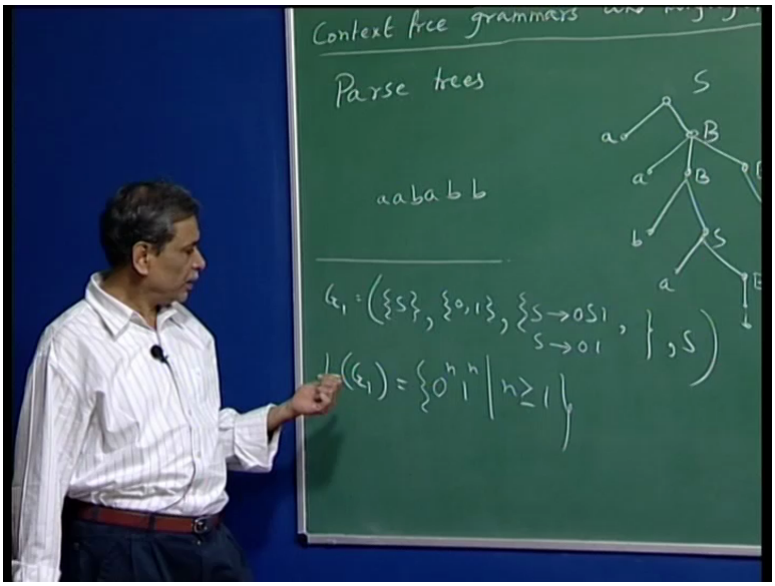
there were two other examples, the first one was the simplest and the (())(33:04) thatG 1 if I call it, that had only one nonterminal and it had terminals 0 and 1 let us say, right? And the P was simply consisting of S goes to 0 S 1 or S goes to 01. These are the only two productions that I had and of course there is only one nonterminal and that is S, right? The start symbol.

(Refer Slide Time: 33:49)



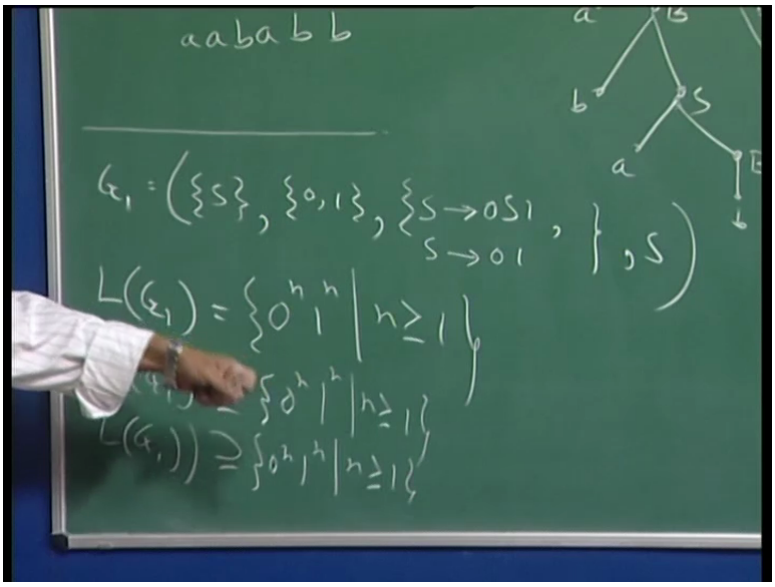
Now I even claimed that look this grammar generates this language. So all strings consisting of0s followed by 1s where the number of 0s is equal to number of 1s. Can we prove this? How will I prove this?

(Refer Slide Time: 34:31)



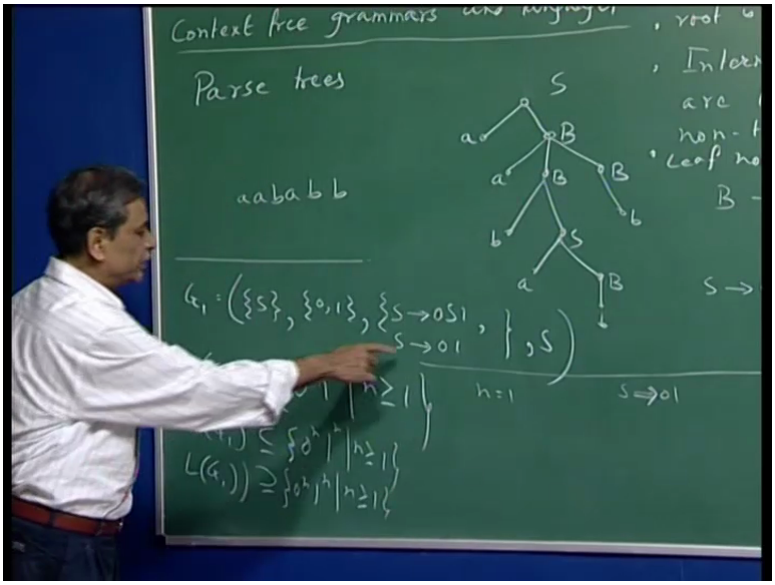
Remember thatif this is again equality of two sets. Herethis set is the set of all strings which are generatedthrough this grammar G and the right hand side is this side. So I should be able to show that equality of two sets.Againthe similar things I have said before that I should show this as well asthis L G 1. Usually that is how (())(35:21) is approved. Now that means I would like to show that. Solet us just show that every string of this form 0 and 1 n can be generated through the grammar G 1.

(Refer Slide Time: 35:36)



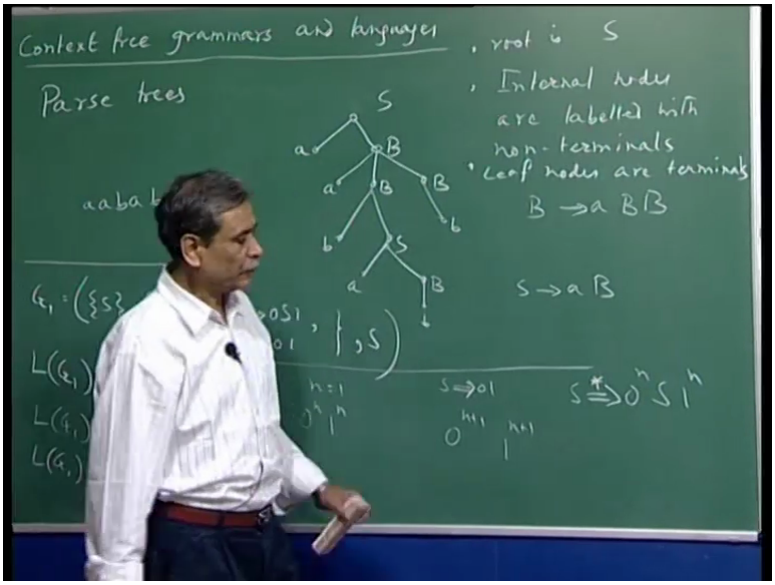
How do I show that? So which of these I am proving? I want to prove that every string of this type can be generated by the grammar G 1. So essentially I am trying to prove this one, okay. So you seesuch a proof will be through induction. That will be the most straight forward way of doing it. So what is the base case? Base case is when n is equal to 1 then the simpleone step derivation that S just rewrite as 01, so S derives 0 1 which is a rule which I use and I get. So this is a base case.

(Refer Slide Time: 36:24)



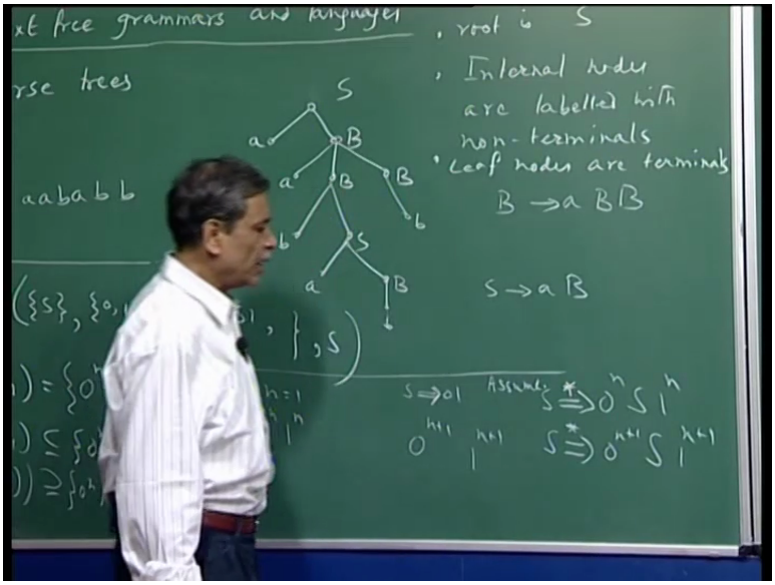
What is the (induct) induction? That suppose I assume that strings of this kind, 0 n 1n all strings up to some n can be generated of this form. Then I should be able to show that 0 n plus 1 and 1 n plus 1 also can be generated, right? So what I canprove a little stronger thing that from S I can derive 0 nS 1 n always for every n, okay right.

(Refer Slide Time: 37:15)



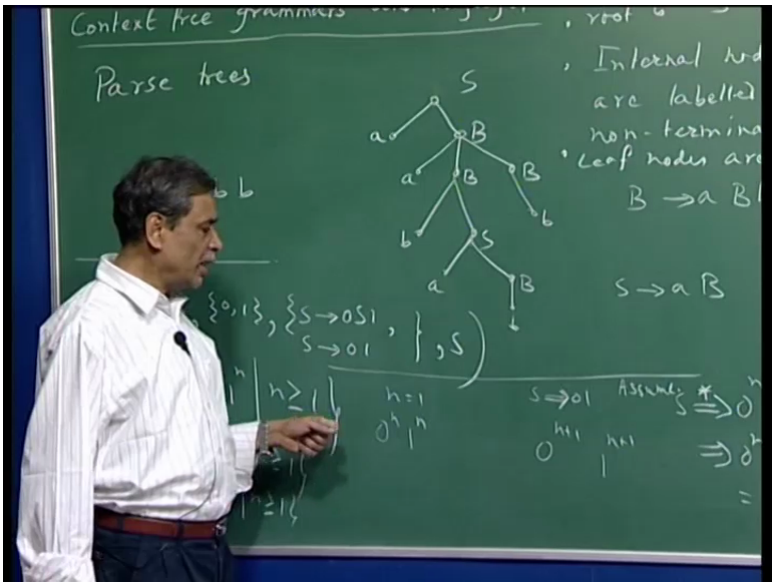
So againthis particular part we can do through induction and it is fairly clear. Now suppose I go to the induction step that I can derive this and then I would like,assuming this to prove that S can also derive, okay.

(Refer Slide Time: 38:00)



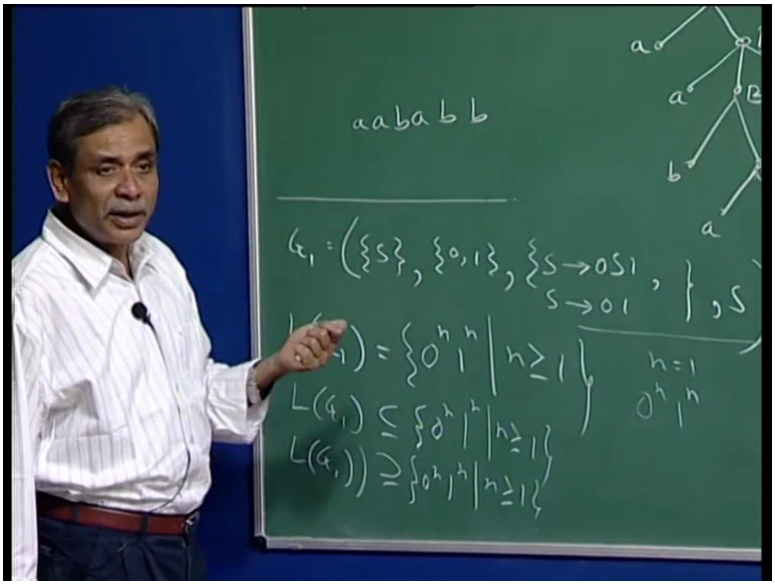
But that is easy, is not it? So once you assume this one then in another step this S you can rewrite as 0 S 1 and of course this 1 n which is nothing but 0 n plus 1 S 1 n plus 1, right? And then this particular S finally you can rewrite as 0 1 and this way you can see that you know this particular proof essentially (showe) shows that all strings of this kind 0 n 1 n where n is greater than equal to 1 can be derived byour grammar.

(Refer Slide Time: 38:42)



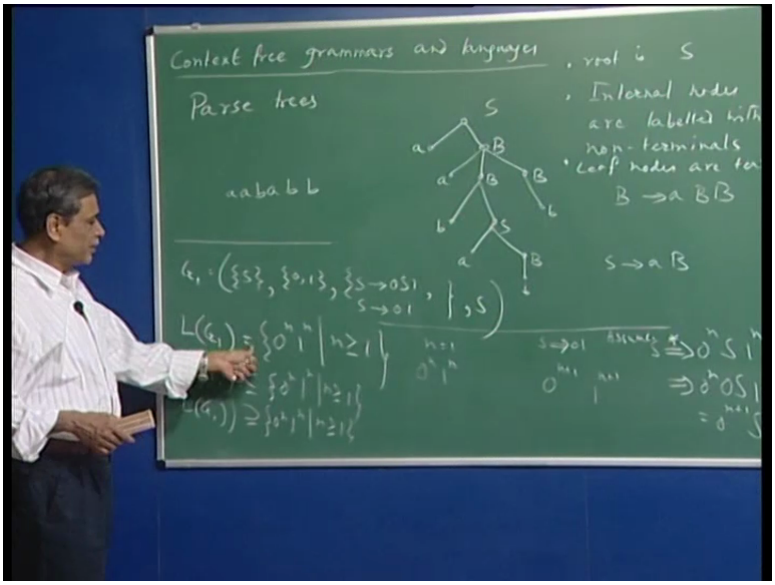
And this part is basically saying that this grammar does not derive anything other than strings of the form 0 n 1 n. But that is again the same thing you see, is not it? That in one step it just drives 0 1.

(Refer Slide Time: 39:04)



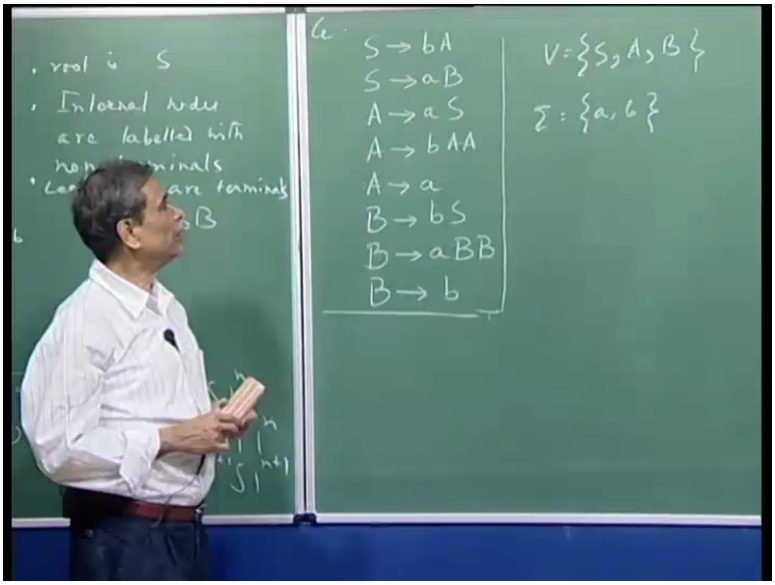
Now assuming you have carried out many steps and then derived this and one more further (se) step it will derive only 0 n plus 1, 1 n plus 1 or 0 n plus 1, S 1 n plus 1, right? So thereforeeasy to see this grammar will generate only strings of this kind. So put together this grammar generate this language.

(Refer Slide Time: 39:34)



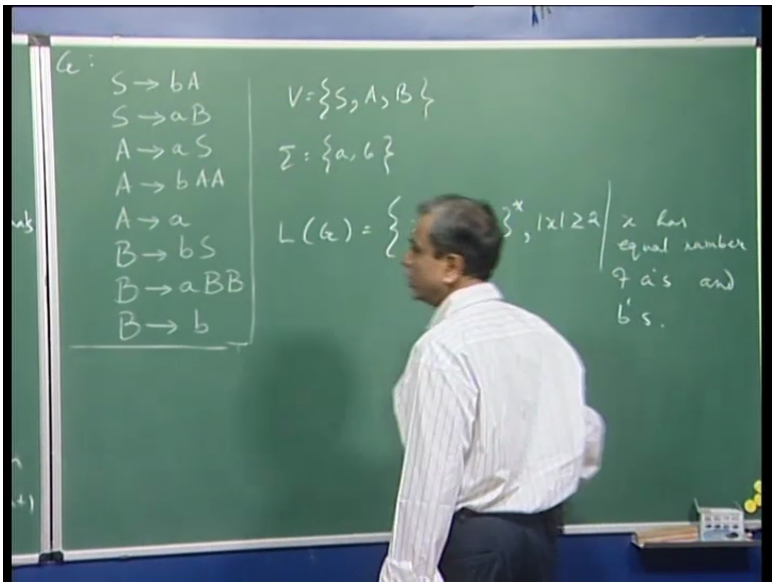
Let us take this example which is a little more interesting. See this isthe grammar G where the set V is nonterminal. There we had only one nonterminal buthere we have three, S, A, B and terminals are a and b and S of course is the start symbol and these are the productions.

(Refer Slide Time: 40:10)



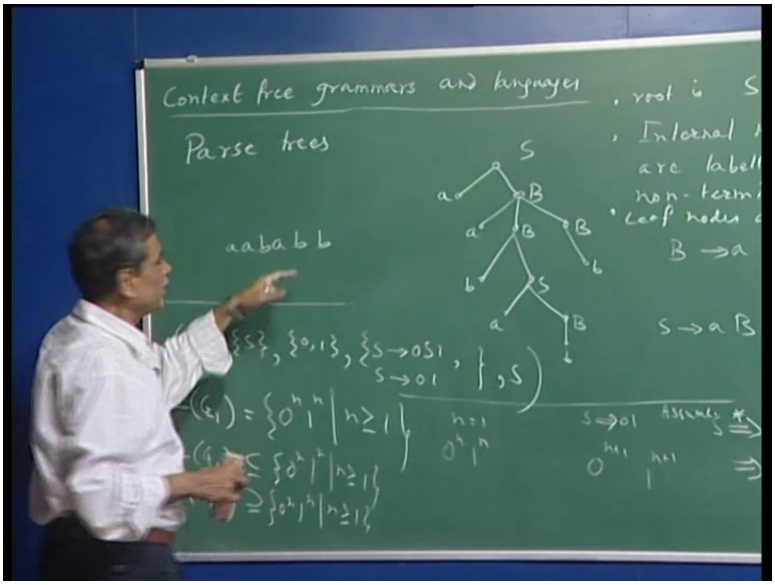
Now what is the language generated by this grammar? Let me write it what all it generates. Basically this is all strings over a, b star and link of x is greater than or equal to 2 such that x has equal number of a's and b's.

(Refer Slide Time: 41:00)



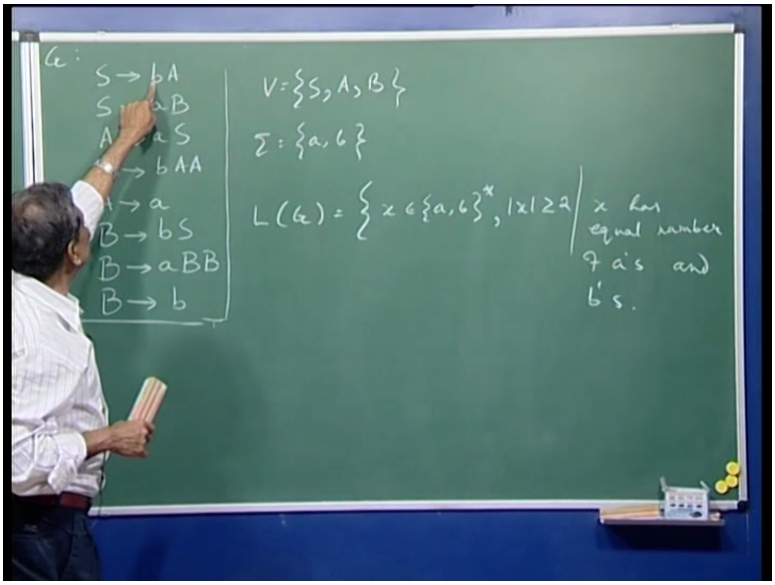
See for example this was the string that we generated. In this one particular string that we generated a a b a b b, okay. So this particular string has three a's and three b's. So (num) number of a's is same as number of b's.

(Refer Slide Time: 41:12)



What I want to claim is that this grammar generates all strings with equal number of a's and b's withat least one a and therefore one b. So it will generate a b, b a. Can you see how will it generate b a? For example this S will start with b. Use this as the first production and then rewrite this a as small a, okay. So b a you can generate.

(Refer Slide Time: 41:45)

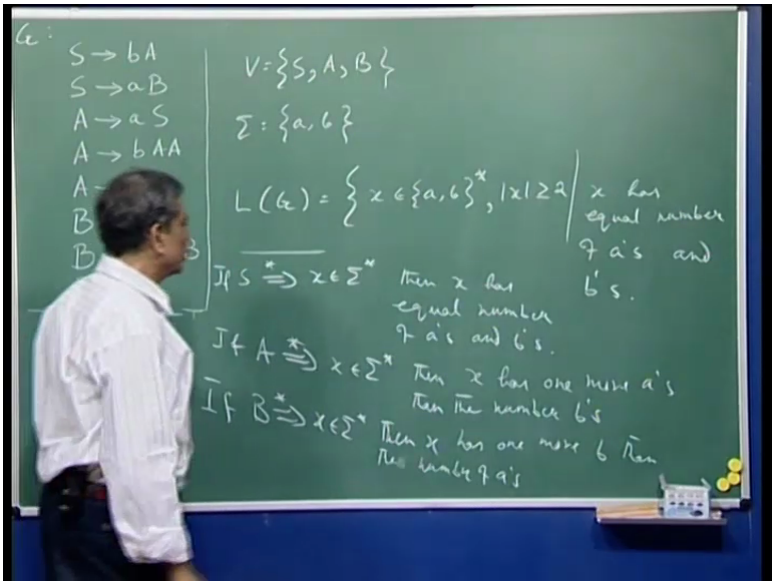


If youplay the setyou will convince yourself that you can generate, indeed looks like I mean you can generate all strings. But how do I prove that this is indeed the case? Let memake one point. I will not spend too much time on this. You see I will understand what are the terminal strings that you can generate starting from S?

But thenwhy did I also understand what are the terminal strings generated by other nonterminals a and b? Andlet me make this claim. So here of course this is stating thatif S generates x in sigma star, right? If this is the case then x has equal number of a's and b's. To consider starting from A and deriving strings. Sostarting from A and after some steps you get x which is again a terminal string.

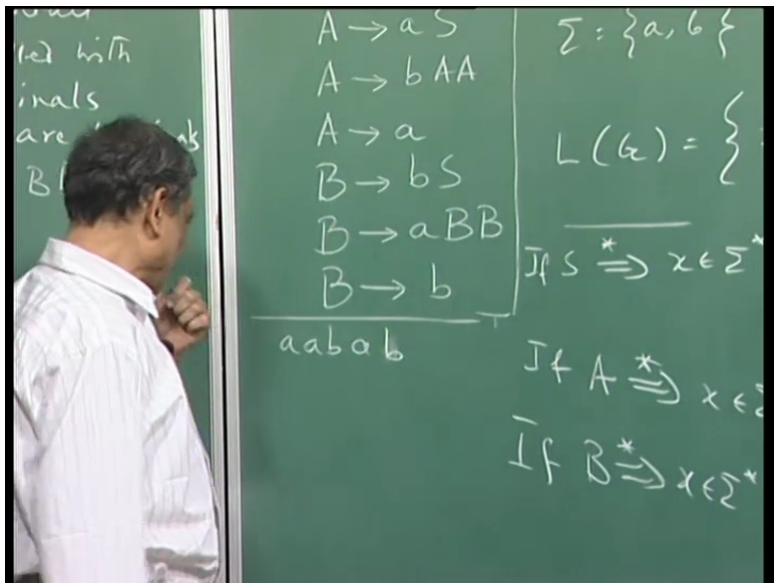
Then this x has one more a's than the number of b's, okay. So A generate all strings of a's and b's where the number of a's is one more than the number of b's. And similarlycapital B case that if B generates a string which is a terminal string then that string has one more b than the number of a's it has, okay.

(Refer Slide Time: 44:48)



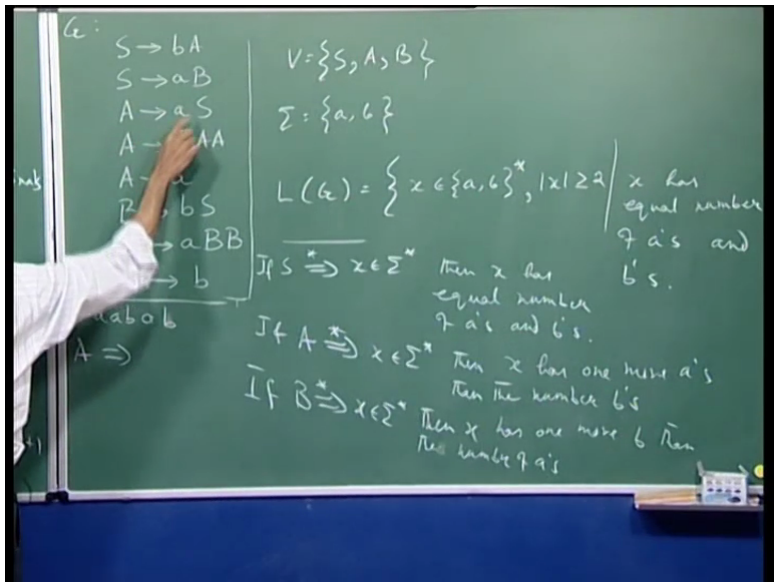
So for example from A you should be able to derive a a b a, this as two more a's. So you needone more b. So this string has three a's and two b's. So the difference one more a than the number of b's. So I am claiming that A should be able to derive it, okay.

(Refer Slide Time: 45:15)



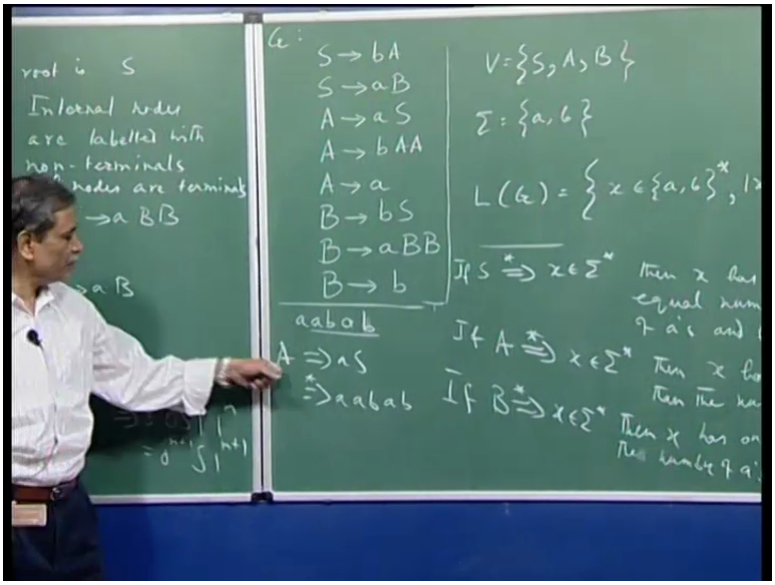
Can you see that? You see if you believe this is true thenthis is easy to show because from A I use this production,this A I use the first this production a S.

(Refer Slide Time: 45:35)



And now what I have is equal number of a's and b's. And using this trackI believe that I should be able to claim that this S can be rewritten as a b a b, okay. So therefore this string can be generated through A.

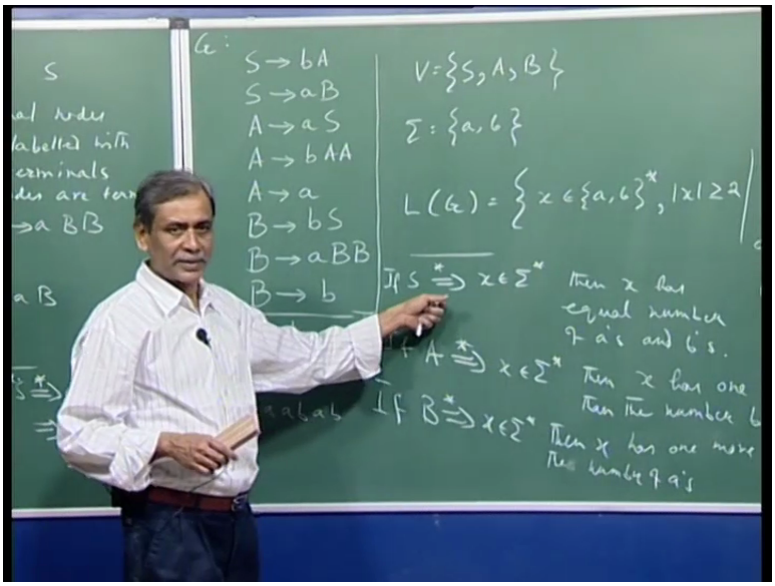
(Refer Slide Time: 45:56)



Now the point is to prove any one of these. I request to use, it should not be too difficult to see this that I request the use of the other two possibly, okay. And how do I prove such things? So basically through simultaneous induction. All these three statements I proved using induction but simultaneously. So what are the base cases? Base cases is for this the base case will be a b, right? Thusfrom S can you generate a b.

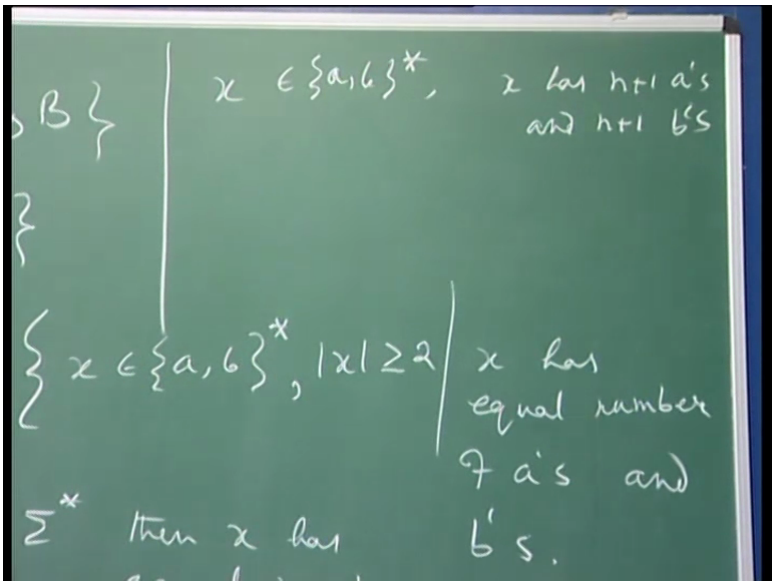
That is simple right?That you can just see that from S I go to this and then this I write that or and ofcourse S generates b a. This is the two base case strings for this, right?

(Refer Slide Time: 46:58)



And now consider the basic idea is something like thisof this proof through simultaneous induction, okay. Now consider a string which has n a's and n b's where n is larger than 1.so consider x, and x is in a b star and x has n plus 1 a's and n plus 1 b's. I am just trying tosketch a proof very quickly.

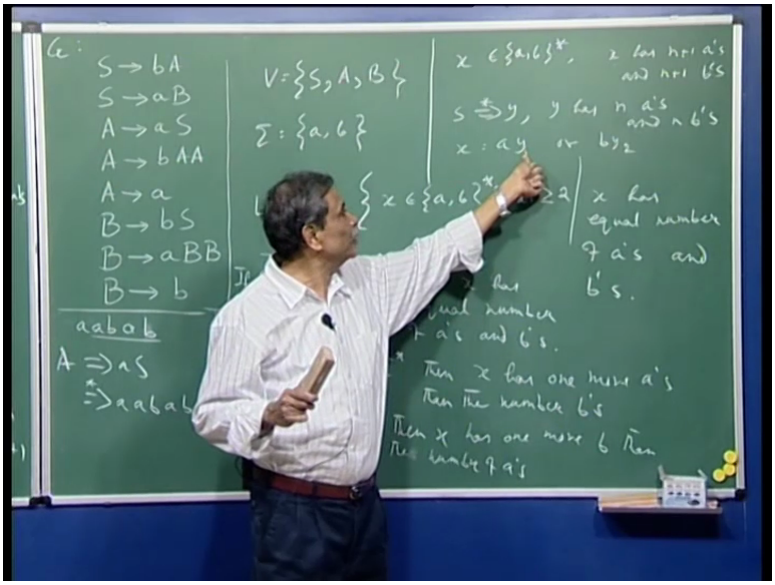
(Refer Slide Time: 47:51)



Through induction hypothesis I can say that S can generate strings of this kind.Y, whereany y which has equal number of n a's and n b's. That is my and n b's, right? Now you see suppose in x I would like to show that then S can also generate this x which has n plus 1 a's and n plus 1 b's.

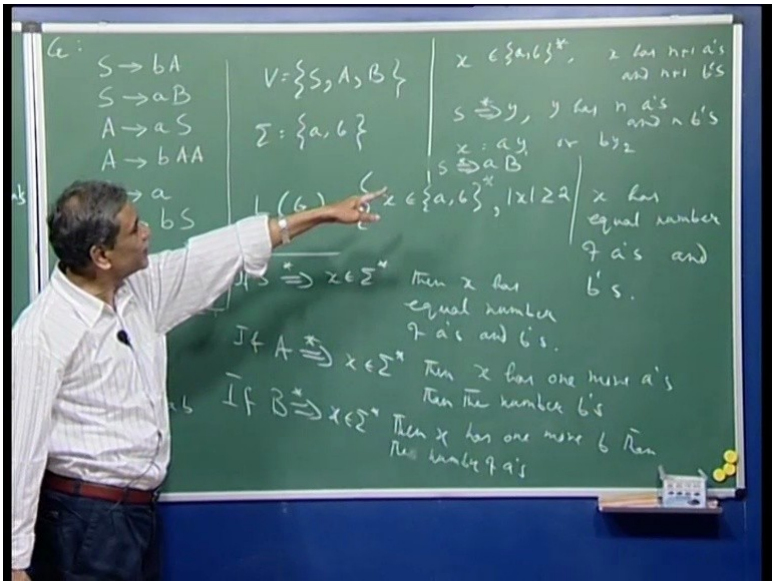
So that string x is either of the form a y 1 or that means either it starts with a or it starts with b. If it starts with a then what is this y 1? Y 1 is a string with one more b. Seexhas n plus 1 a's and n plus 1 b's, right? Take out the one a so thenthis string y 1 will have n plus 1 b's and n a.

(Refer Slide Time: 49:19)



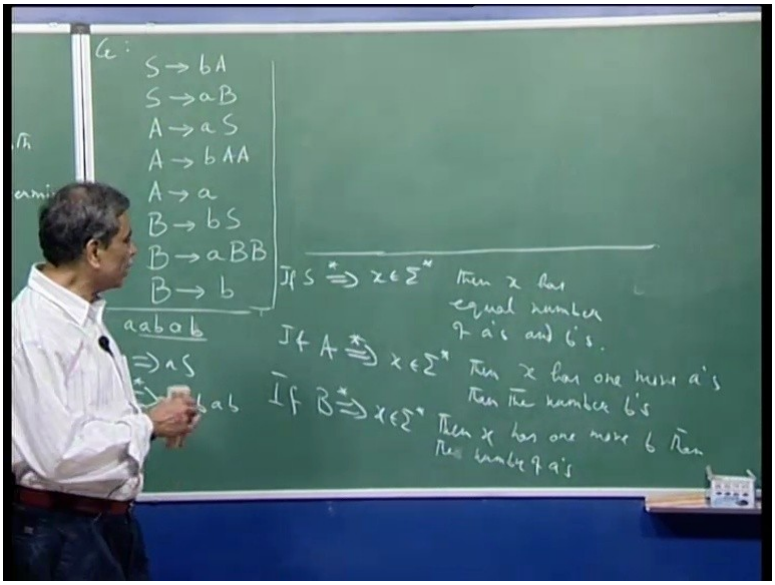
So such a string by induction hypothesis I would claim that can be generated by b, right? So what I am going to do that I will start with S, right? And say S deriving a b and now through b using the induction hypothesis I say(genera)I can generate the string y 1 because it has one more b's than the number of a's, okay.

(Refer Slide Time: 49:50)



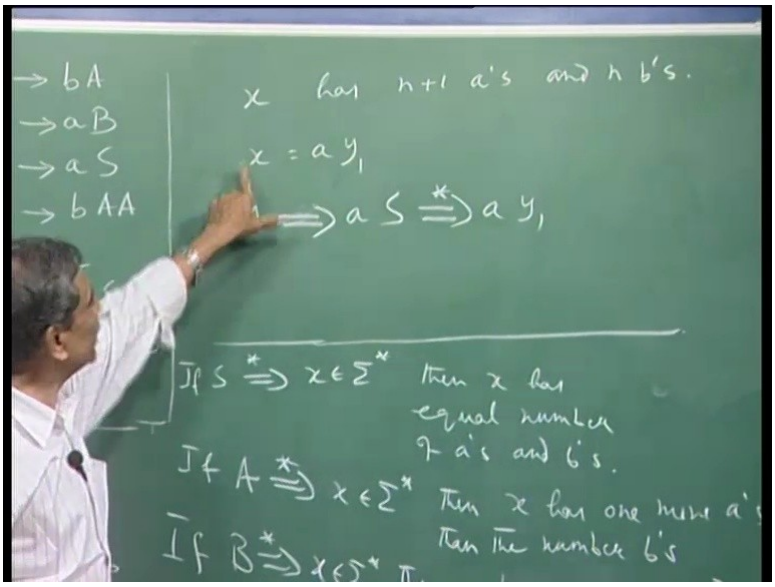
Now similarlyyou will see for example let us take the more interesting case. Now how do I prove these? Again I will of course require the help of other two, let me show that. For example I want to show thathow the induction will go through for a string which has one more a than the number of b's it has, okay.

(Refer Slide Time: 50:27)



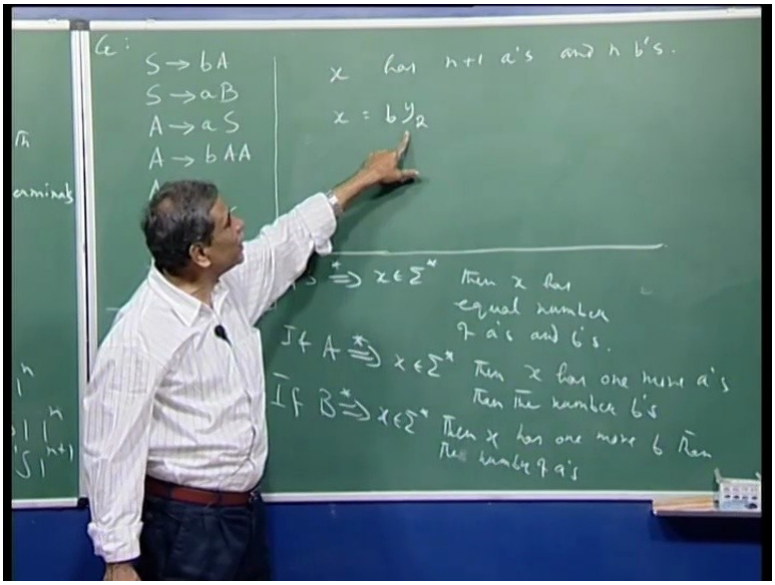
So let us say x has n plus 1 and n b's. Such a string x can start with small a or also can start with small b. Sothis x suppose it starts with a then it is of the form a y 1. So y 1 then has equal number of a's and b's. Therefore it can be generated by S, right? So you see the point is that a I will use this one production to see a S and now I use the induction hypothesis to say that S generatesy 1. So thereforea generates, a will derive x.

(Refer Slide Time: 51:29)



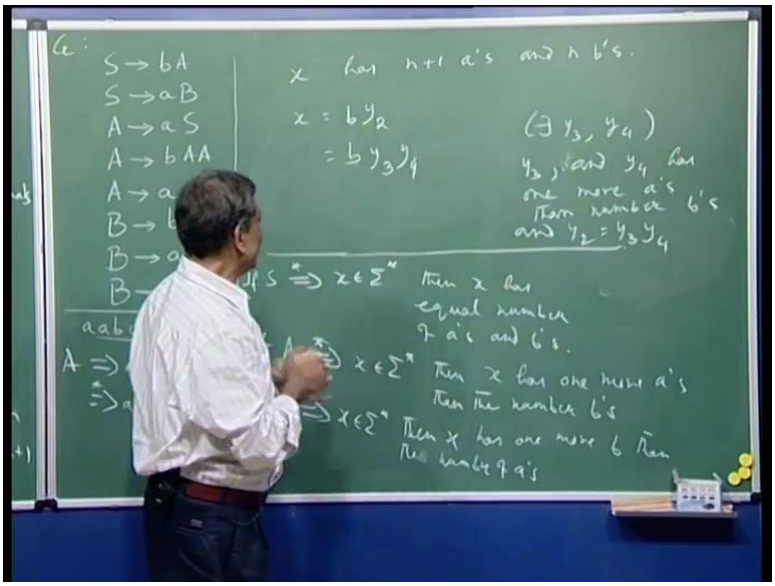
But x could have started with b as well. So let us take that case. That is more interesting. So x has n plus 1 a's and n b's and now x is starting with b, right? What can you say about y 2? Y 2 is a string which has, you seex had n plus 1 a's and n b's and you are taking one of the a's out. So now y 2 will have n plus 1 a's and n minus 1 b's, okay.

(Refer Slide Time: 52:24)



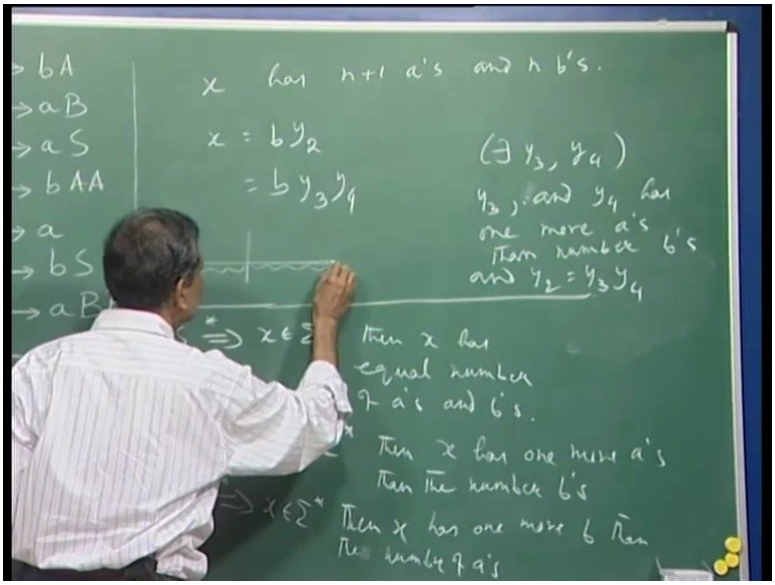
So this y 2 is a string which has two more a's than the number of b's it has, right? Now interestingly clearly such a string y 2 can be rewritten as, rewritten in the sense there must exist. So let me put it this way. There must exist y 3, y 4 such thatboth y 3 and y 4 has one more a's then number of b's. And this y 2 is nothing but y 3, y 4. So what I am saying such a string which has gotexactly two more a's than number of b's it has.

(Refer Slide Time: 53:40)



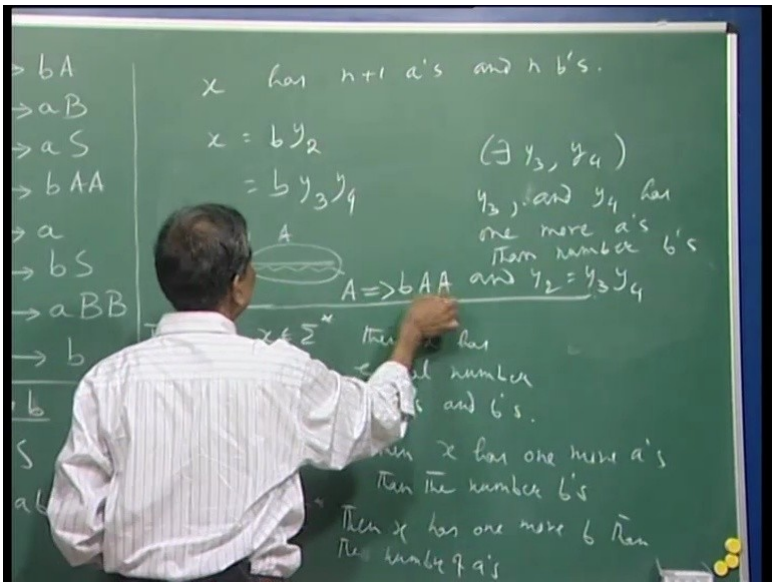
You see what you can do such a string you know you start from left and youshould be able to find. It isnot difficult to argue this that iftotally it has two more a's than number of b's, so there will be a part where it has one more a than the number of b's. And here it will have one more a than the number of b's in this part.

(Refer Slide Time: 54:04)



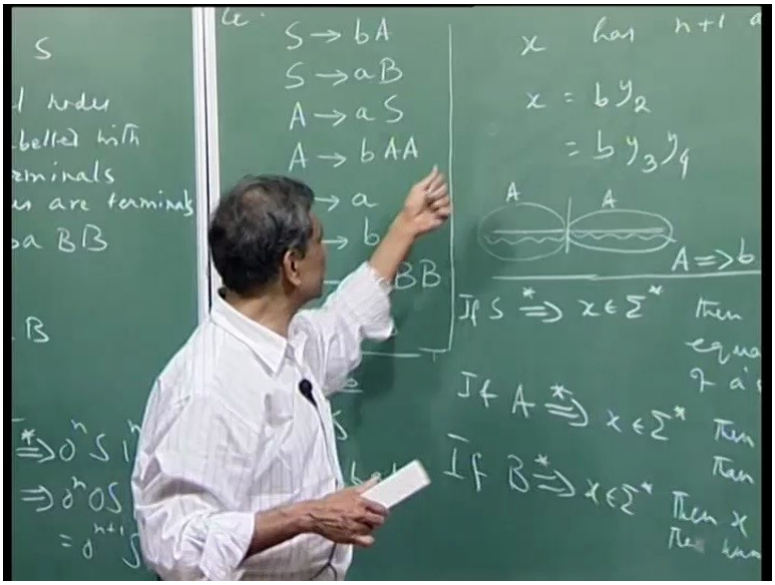
So therefore such a thingcan be generated. This has one more a's than the number of b's here. So that can be generated by a and this part also can be generated by a. And thenso what is happening?Such an xI want to claim that this xcan be derived starting from a. So I will start from a and use this production b A A. And this A I will use to generate y 3 and this A I will use to generate y 4.

(Refer Slide Time: 54:52)



So in this manner I can prove that all strings which has one more a's than the number of b'ssuch strings can be generatedstarting with A.

(Refer Slide Time: 55:15)



You seeI would request you to actually formally use this intuitive idea and prove that indeed this grammar Gprecisely generates all strings with equal number of a's and b's.