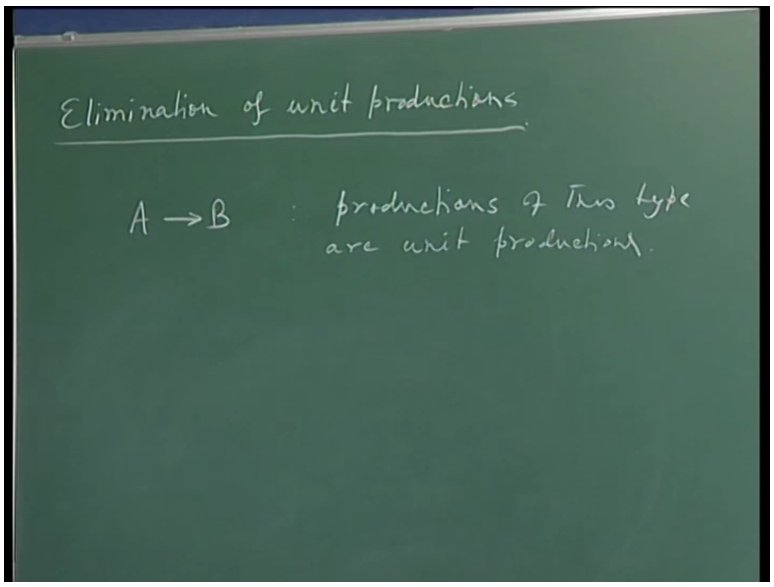
Theory of Computation
Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 25
Elimination of Unit Productions
Converting a CFG into Chomsky Normal Form
Towards Pumping Lemma for CFLS

For simplifying context free grammars we carry out three processes, one is removal of useless symbols, the second is removal of epsilon productions and today we will look at elimination of unit production. The first two processes we have already covered in the earlier lectures. Unit productions are the productions of the kind A goes to B. In other words the right hand side consist of exactly one nonterminal.

So production of the kind where the left hand side as usual is a nonterminal and the right hand side consists of just one nonterminal such things are called unit productions. Soproductions of this type are unit productions.

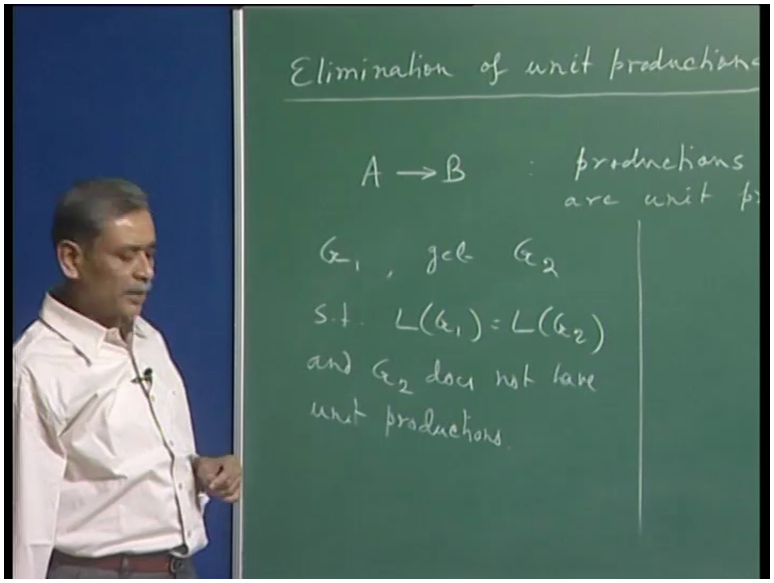
(Refer Slide Time: 01:27)



And we would like to eliminate all such productions from the grammar without affecting the language generated by the grammar. In other words as before we are given one context free grammar G 1. From here we would like to get another context free grammar G 2 such that the two grammars generate the same language and G 2 does not have unit productions. So recall that suchpairs of grammars are equivalent grammars because they generate the same language.

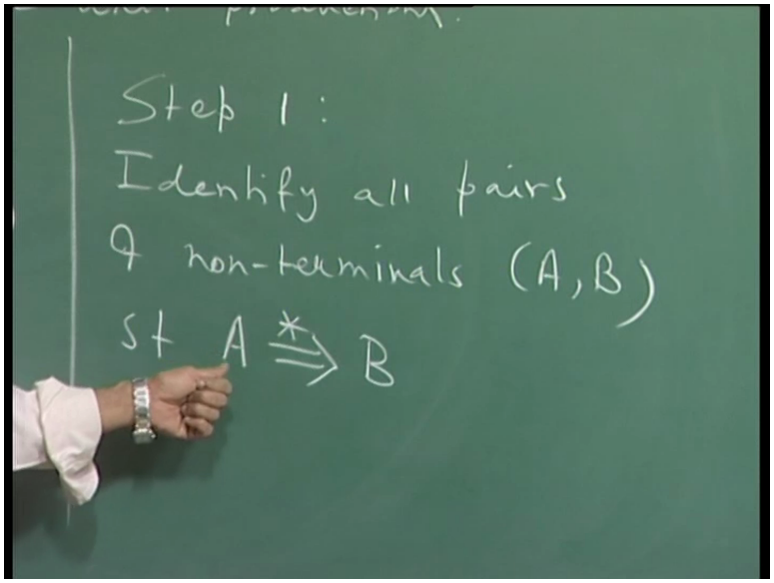
And the way we eliminate all unit productions is very similar to the way we eliminate epsilon productions. Remember productions were the production where the right hand side contains just the empty string epsilon.

(Refer Slide Time: 02:41)



Now what we do is to remove all unit productions, step 1 is to identifyall pairs of nonterminals A B such that A derivesB. Recall this means in number of steps, one or more stepsfrom A you can go to B. Actually this we have used for zero or more steps that is if you can go in zero or more steps from A to B then we say that A derives B.

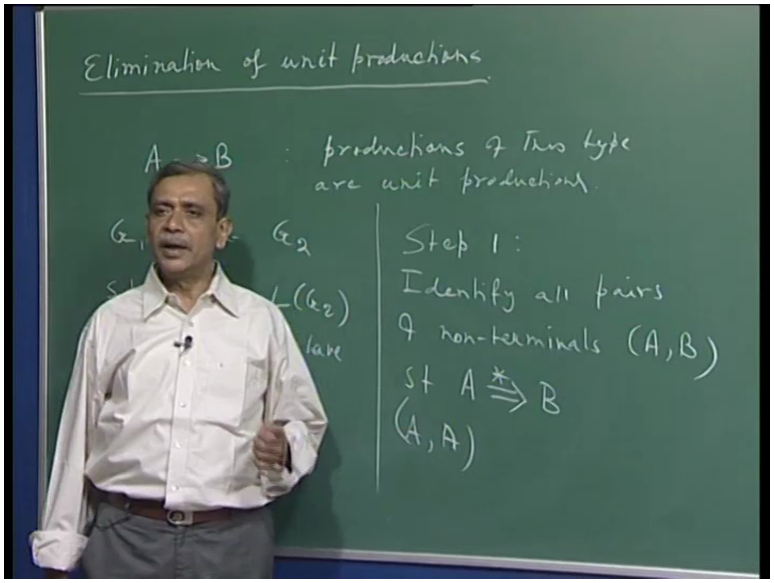
(Refer Slide Time: 03:51)



Now how do we do this? How do we find out all such pairs?Of course trivially every such pair will satisfy this. That of course you can go from A to A using no production at all. Using

zero production you can go from A to A. But this is the trivial case. More interesting case is when you can go from A to B in exactly one step. That is exactly in one step of derivation you can go from A to B.

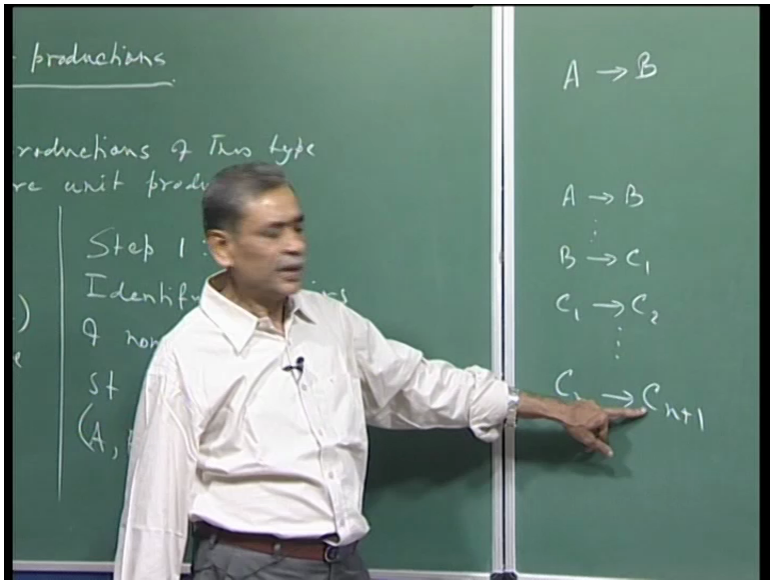
(Refer Slide Time: 04:32)



And when is that possible? Clearly that will be possible when you have a production of this kind. So what will do is as we did for productions we will start with a base and from there we can generate the rest of pairs such that A derives B. Sothe basesetwill consist of A B where A goes to B is a production.

And you can now see inductively what we are going to do that if you have already defined let us say A goes to B and let us sayB goes to C 1, C 1 goes to C 2 and so on. And let us say C n goes to C n plus 1. Then clearly from all this you can see that A also will derive this nonterminal C n plus 1.

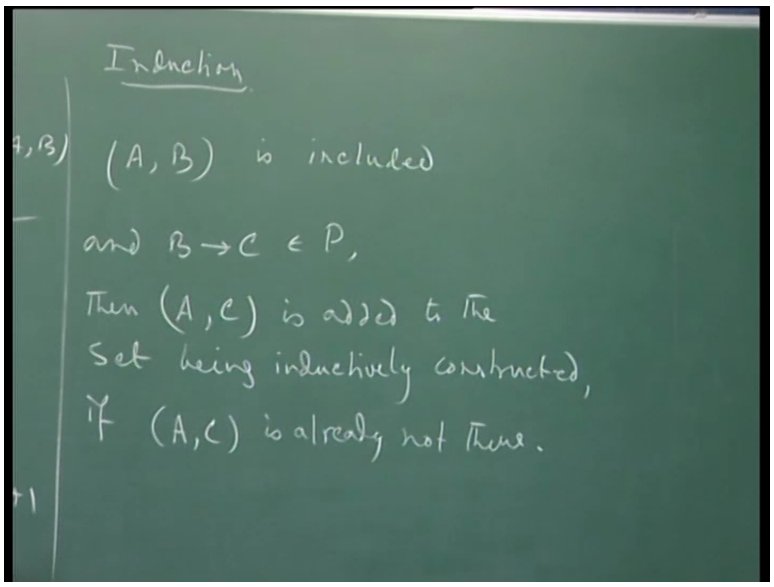
(Refer Slide Time: 05:39)



So therefore A is C n plus 1 is going to be such a pair, right? So actually we can do it in a more straight forward manner which is a little more intuitively clear but even let me complete what I was saying here that we start with the base case. So base consists of identifying all pairs A B such that A goes to B is a production, right? So let me write it down. Base consists of all pairs A B satisfying above. And induction is going to be that suppose I have created a number of such pairs.

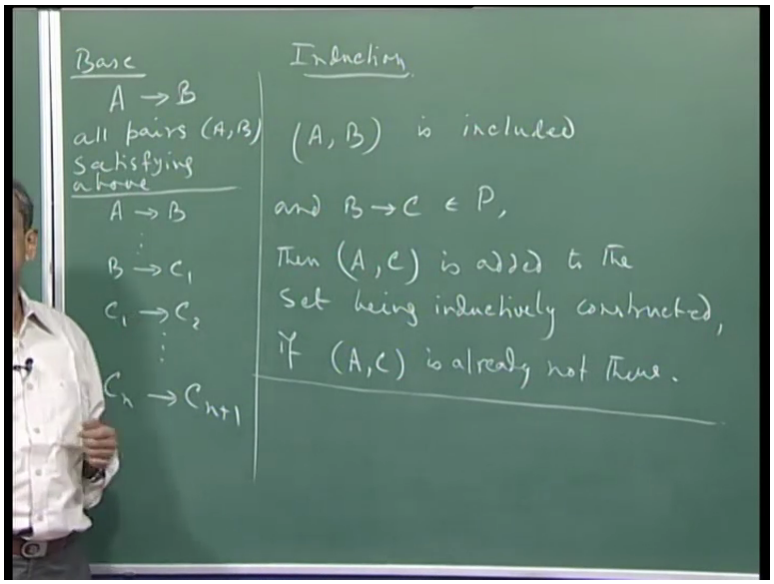
You know starting with the base following whatever we are suggesting we have already have a number of pairs satisfying this. And then if I find that in my set A B is already included and B goes to C is a production is in P, then(go) A C pair is added to the setbeing inductively constructed if A C is already not there, right?

(Refer Slide Time: 08:03)



Now this way in other words I have created some amount of these pairs and then I looked at all unit productions again and see that if I have any situation where B is the right hand side of something that I already have, in that case I will put the left hand side pair with C. This is the new pair that I am adding provided it is already not there. So this way wecarry on till we find that the set that we are constructing cannot be made any larger. This is the usual inductive construction that we did for eliminating epsilon productions.

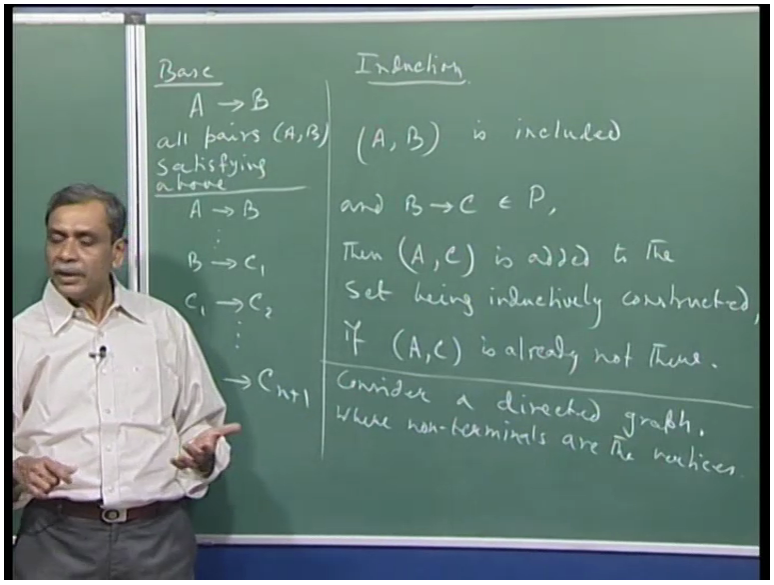
(Refer Slide Time: 09:02)



I just remark that we can see this process of identifying all pairs such that A derivesB is the case in the grammar. A little more simply and clearly and that is if you think of a graph, consider a digraph directed graph, okay, where nonterminals are the vertices, alright. So

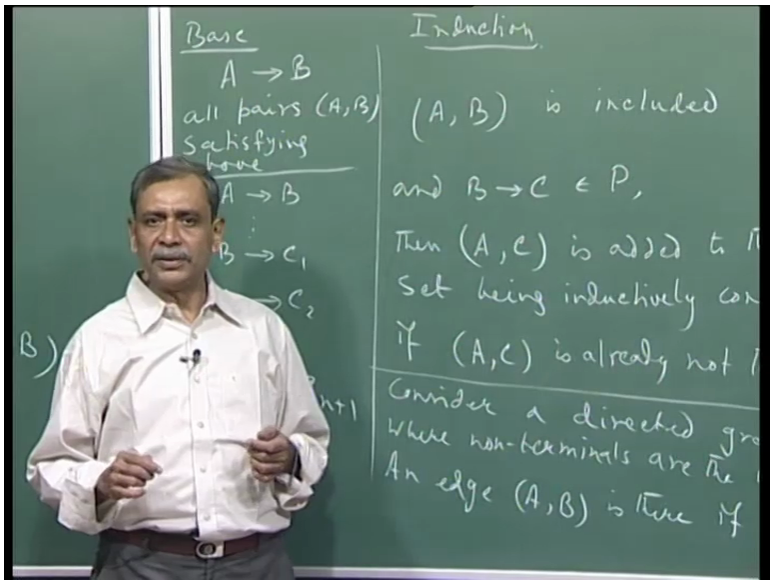
consider a directed graph. Weare defining a directed graph in which first of all I define the vertices of the graph are the nonterminals in the grammar.

(Refer Slide Time: 10:04)



And then we will put an edge.An edge A to B is there if A goes to B is one of the productions in the grammar, right? So it is very simple that you start with the set of all nonterminals as vertices of digraph and add this edge. And this is the directed edge A to B if A goes to B is a production, right?

(Refer Slide Time: 10:43)

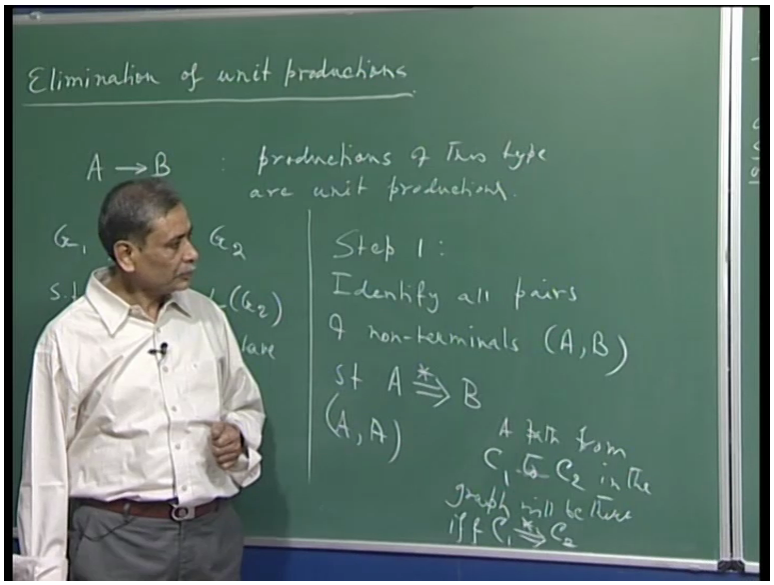


And now in that graph if you find that there is a path from C 1 to C 2, okay, in the graph. So a path from C 1 to C 2 in the graph will be there if and only C 1 derives C 2, okay. This is not difficult to see why this is happening. So start with a digraph whose vertices are the

nonterminals and the edges are all those unit productions in the graph. And then to check whether A derivesB or not all we need to see if there is a path.

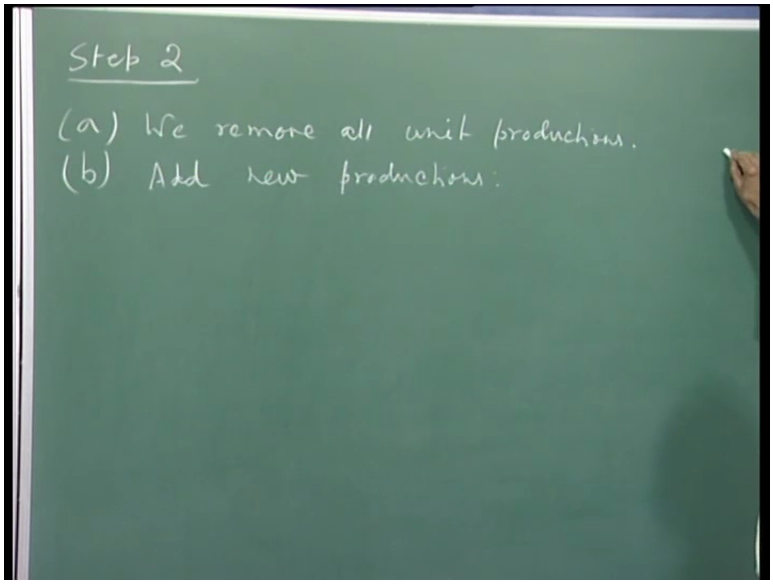
Now remember in this case the path is going to be a sequence of directed edges. So if there is a path from A to B in the graph that means A derives B. And that is fairly easy to see why that is happening.

(Refer Slide Time: 12:05)



Now this is how first of all we identify all pairs of nonterminals such that A derivesB. So after carrying out the step 1, we carry out the step 2. And in step 2 first of all we remove all unit productions. And in the second part of step 2 we add some productions so that the removal of this unit productions will not affect the language which is being generated. So add new productions.

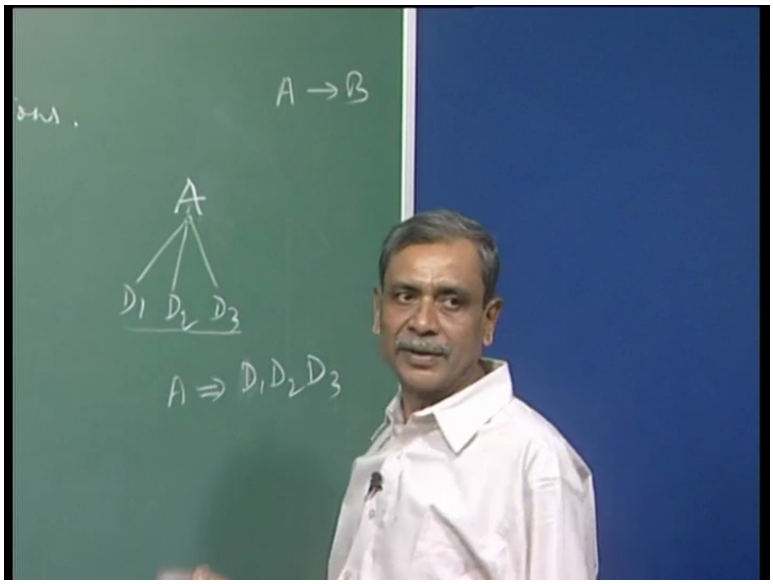
(Refer Slide Time: 13:14)



And which are these new productions? The idea let me explain first.Say suppose in the old grammar you hadA goes to B is a production, so it is possiblein a derivation you replace A with B and then apply one productionwhich is not a unit production. So let us say this one was D 1, D 2, D 3. So in affect what is happening is A is deriving D 1, D 2, D 3.

So now if this unit production is not there, right. This unit production A goes to B is not there then what I want is that from A I should be able to directly (diff) derive or I should have a production so that from A, D 1, D 2, D 3 can be obtained. One can replace A by D 1, D 2, D 3.

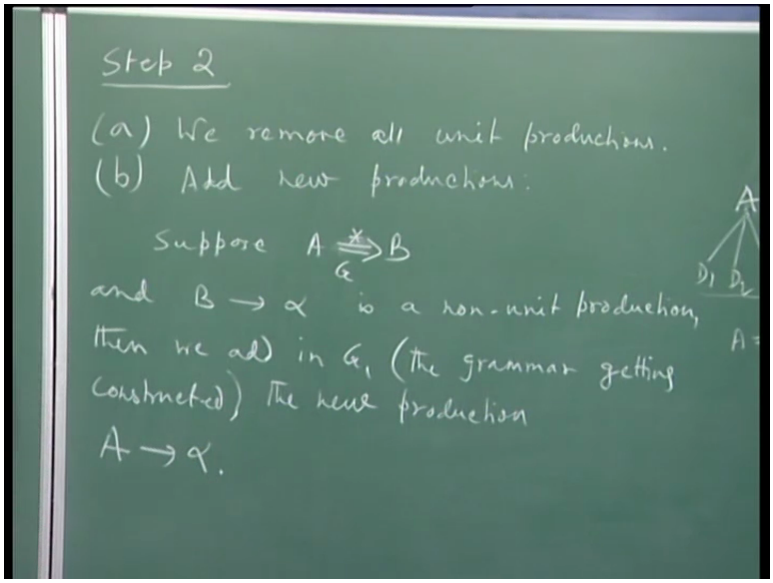
(Refer Slide Time: 14:21)



So this is the basic idea, right? Thereforethe language which has been generated will not be affected. So let us seehow we will see this. We say that suppose A derives B in the old grammar G. Now you should say the old grammar G and the new grammar is the G 1which is what we are defining.

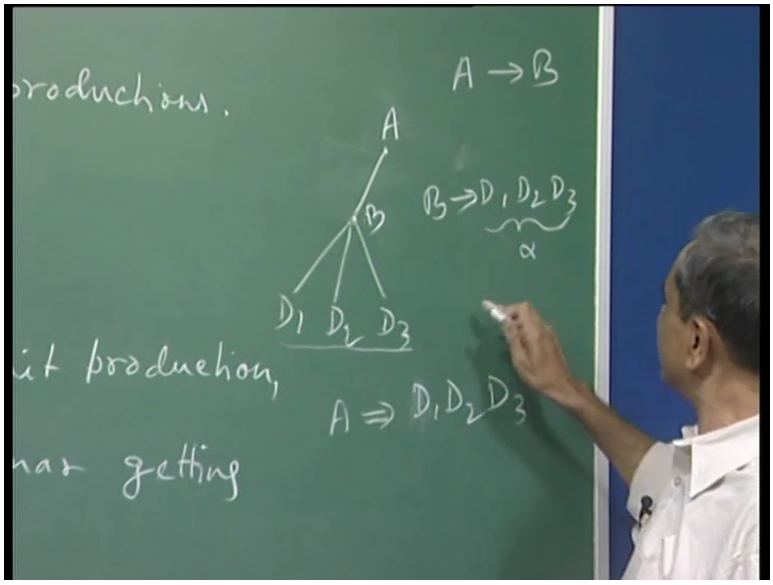
So if in the old grammar from A you could derive Band let us sayB goes to alpha is a non-unit production then we add in G 1, the grammar getting constructed, thenew production A goes to alpha, okay.

(Refer Slide Time: 16:10)



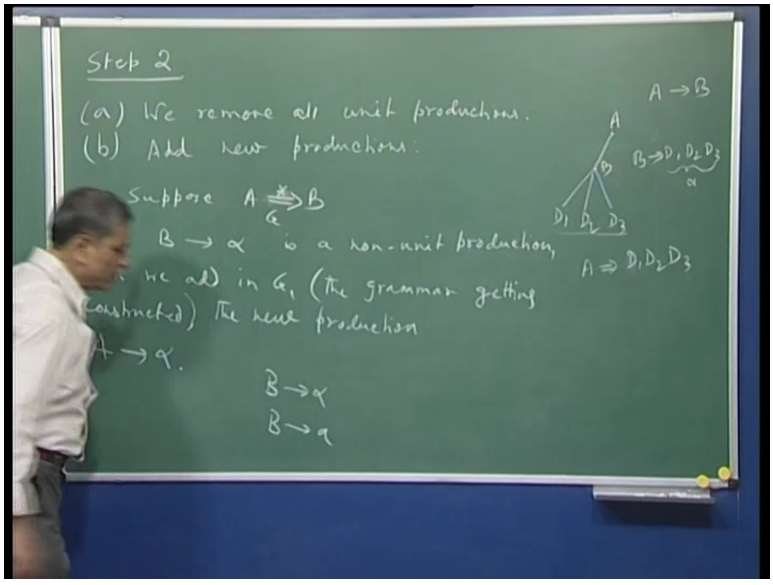
Now in the old example of ours this was A, this was B. So in the old grammar G we could have derived from A, this D 1, D 2, D 3. Now what is happening? We said soif you look at what we have said, so here alpha is D 1, D 2, D 3. This is alpha and B derives alpha and A goes to B is a production. So we are saying add the production A, D 1, D 2, D 3.

(Refer Slide Time: 16:47)



So then you would be able to generate the same string which the derivation of that tree was carrying out. So it is not difficult to see why this should ensure that the new grammar G 1 will generate the same language. At the same time the new grammar G 1 will not have any unit productions because we have already removed them. Just one point here, see recall that B goes to alpha we said is not a unit production. Is this a unit production?

(Refer Slide Time: 17:37)



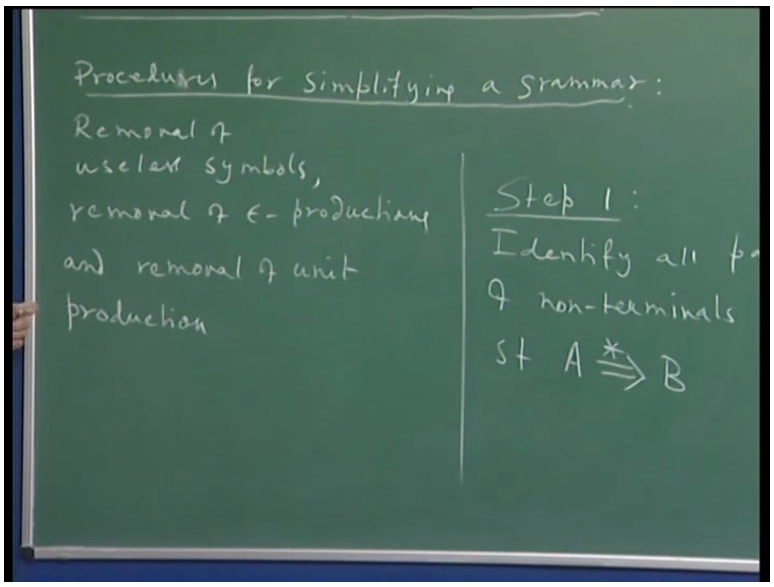
It is not a unit production because remember all the unit in the sense one, although the right hand side consists of only one symbol but because this is a terminal it is not a unit production. In case of a unit production it has to be that B goes to something which isone nonterminal. So this takes care of the process of elimination of unit productions from a

grammar G to obtain a new grammar G 1. And the fact that the two grammars are equivalent that means they generate the same language that can also be proved.

And that proof is very similar. Again in all this proofsare by induction and that proof is very similar to the proof that we had given when we constructed a new grammar from an old grammar having removed all epsilon productions. Of course there we had to say that the new grammar will generate all strings as the old grammar did except epsilon, the empty string.

So now you see that we have learnt three processes or three procedures for simplifyinga context free grammar and these are removal of useless symbols, then removal of epsilon productions and removal of unit productions.So let me give a title to this that it is going to be the (pros) procedures for simplifying a grammar. So these are the three procedures that we have learnt for simplifyinga grammar.

(Refer Slide Time: 20:30)

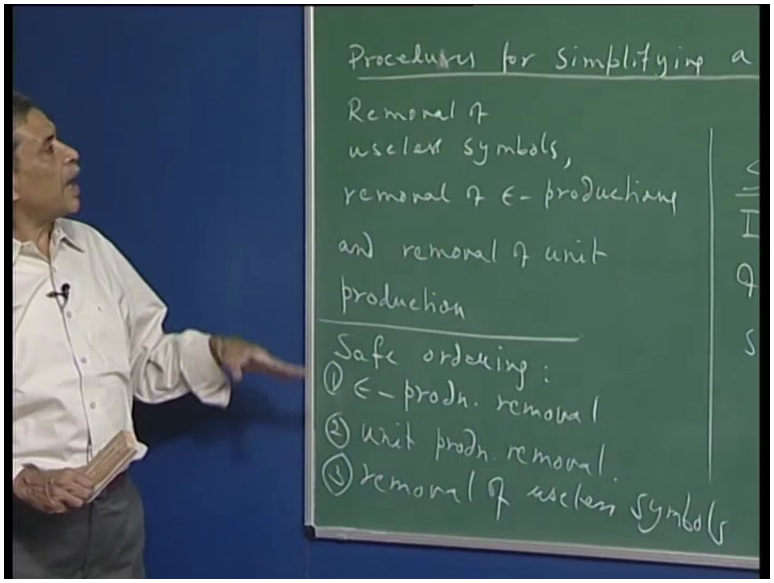


And now a question arises in which order I should carry these out? See the (re)ordering is important because what should not happen is that the effect of carrying out one procedure is destroyed by a subsequent procedure. In other words we should have that order in which whatever I did earlier that effect is not going to be lost by any subsequent procedure.

And again the safe order is, and I am saying again the safe ordering because if you recall for removal of useless symbols again we had two separate sub procedures and there again there was a question of which of these two should be carried out first. And there we said that the safe ordering depends on that principle that the ordering should be such that the effect of the previous procedure should not be destroyed by the subsequent procedure.

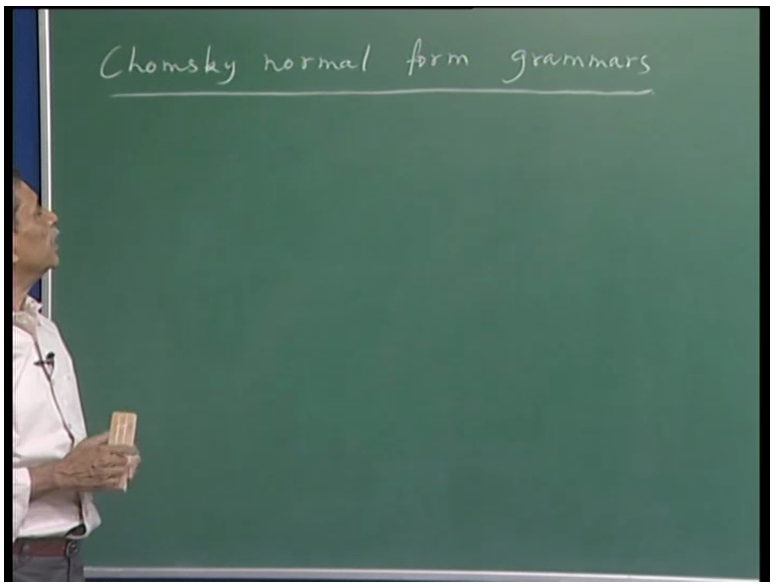
So here too the safe ordering because of that principle is going to be first epsilon production removal. So first is epsilon production removal, then unit production removal and then finally useless symbols removal.

(Refer Slide Time: 22:49)



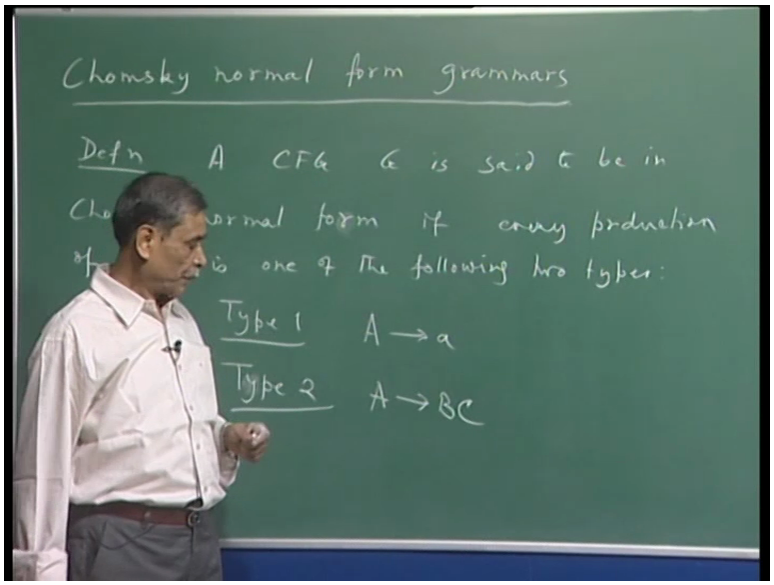
Once we have a simplified form of a grammar then what we can do is to turn the grammar into something canonical. And by that what I mean is the grammar will have a form which is syntactically of the same kind and we will explain that. And such grammars are called normal form grammars and the one that we will study is called Chomsky normal form grammar. And there we will start with a grammar where all these procedures have been carried out. This is one kind of normal form grammars for context free grammars.

(Refer Slide Time: 23:41)



And let me define what Chomsky normal form grammars are? Definition, a context free grammar G is said to be in Chomsky normal form if every production of G is one of the following two types. So type 1 form is a nonterminal followed by a terminal. And type 2 productions are A goes to B C. So in other words type 2 productions as usual ofcourse every production has a nonterminal at the left hand side. The right hand side consists of exactly two nonterminals.

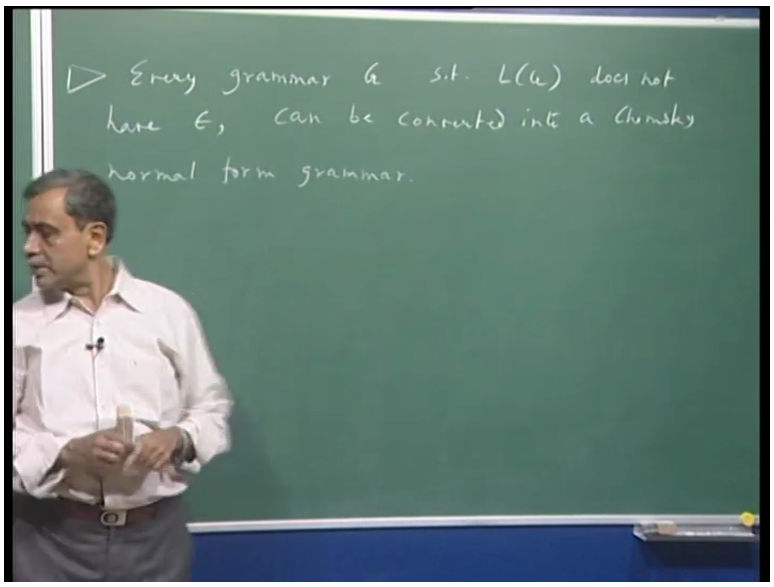
(Refer Slide Time: 25:29)



Now although I am writing A goes to B C,one of Band C or both of B and C could be the nonterminal A itself. So B and C could be any nonterminals, right, including A itself. So these are the only two type of productions which Chomsky normal form grammar will have. And main point isthe following result that every grammar G such that L G does not have epsilon.

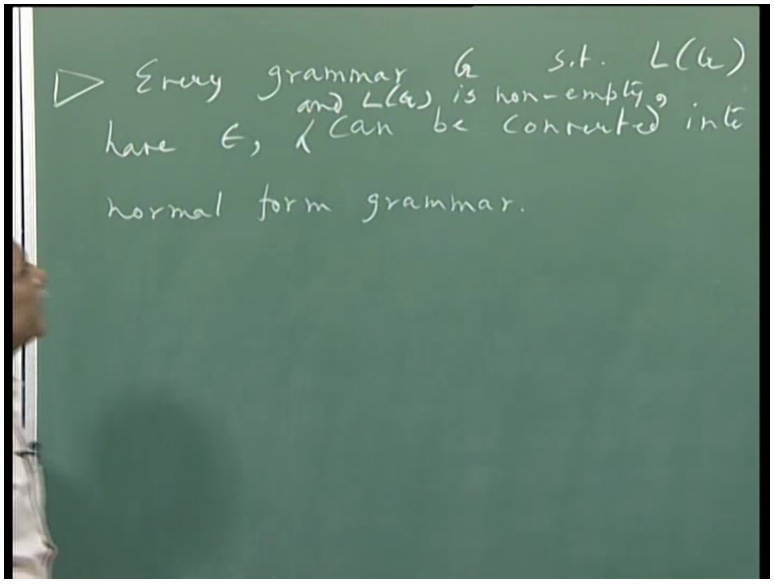
Soany context free grammar G which generates a language without epsilon, such a grammar G can be converted into a Chomsky normal form grammar. I should add that some people would define a grammar to be (Choms) in Chomsky normal form provided it satisfies one more condition. Further G does not have any useless symbols, okay. The result here applies also for this extended definition of Chomsky normal form grammar.

(Refer Slide Time: 28:03)



So what we would like to show is that any grammar G which produces a language which does not have epsilon can be converted into a Chomsky normal form grammar. And one more condition that we should have and that is you will see that is a kind of technicality that the languagewithout of course it does not have epsilon, but also it should be non-empty. In other words L G does not haveepsilon, so let me use this also. L G is non-empty, okay.

(Refer Slide Time: 29:09)



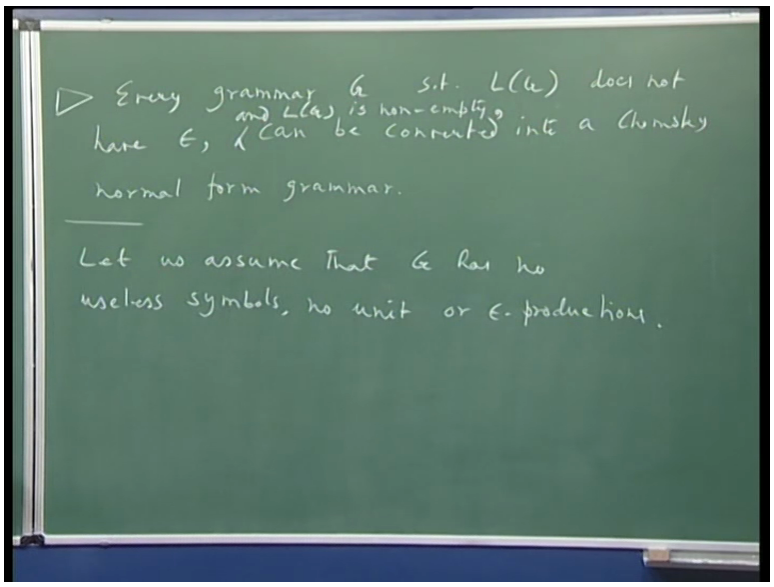
So what is the problemif L G is empty? Because then if you see that the grammar G does not have any useless symbol then you have a problem, is not it? Becauseyou know any grammar has to have a start symbol and if the grammar produces no string at all, right, in that case S

itself is useless because S does not produce any or does not generate any terminal string. So that is a technicality and therefore we are putting that also as the condition.

So therefore what we would like to show is every grammar G such that L G does not have epsilon and L G is non empty, such grammars can be converted into a Chomsky normal form grammar. So now because we have learnt the various (pros) procedures we had outlined for simplifying context free grammars I can right away take G to be in that simplified form. In other words let us assume that G has no useless symbols, no unit or epsilon productions, right?

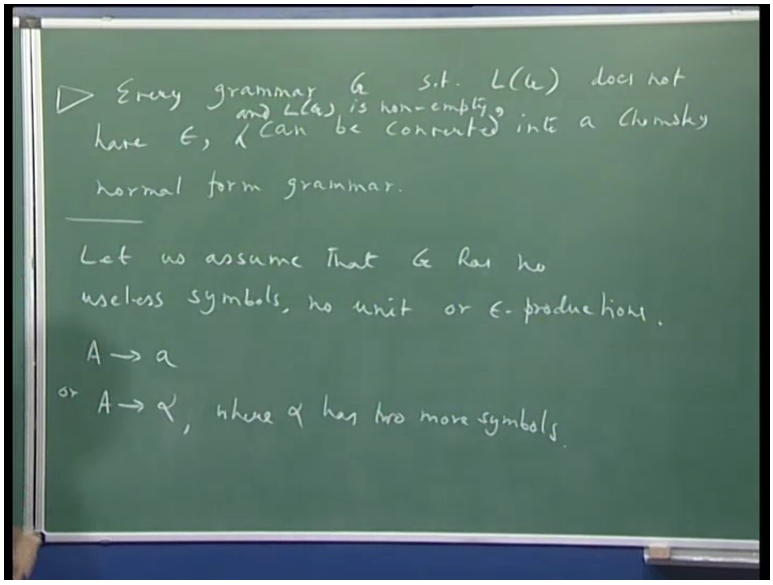
So I can write away assume my G is to satisfy this condition because original G that you gave to me if that did not satisfy these conditions I can turn it into a new grammar satisfying these conditions.

(Refer Slide Time: 31:23)



So let me assume that G has no useless symbols, no unit or epsilon productions. Therefore you see either I have a production of this kind, right? So either the right hand side of a production consists of one nonterminal or A is to alpha where alpha has two or more symbols, right?

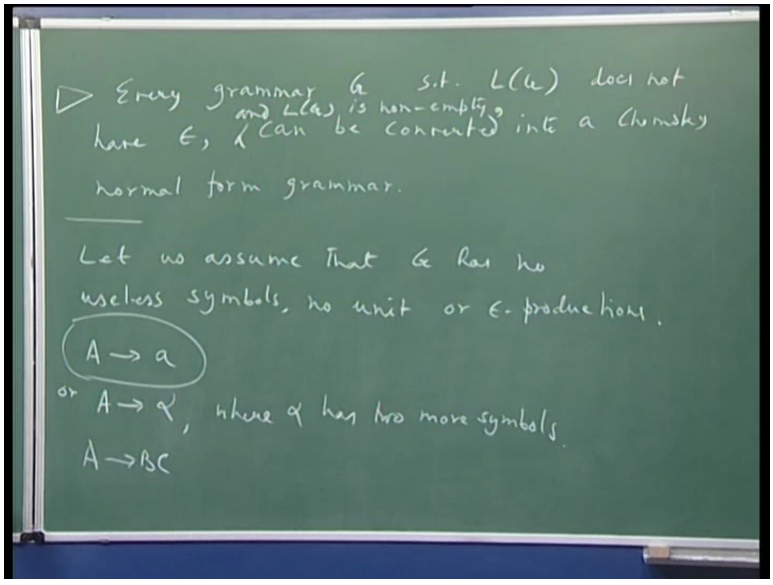
(Refer Slide Time: 32:13)



Because you see we do not have unit or epsilon productions. So these kinds of productions we could have,A goes to a single nonterminal or the other kind of production this G will have, the right hand side will have two or more symbols.By that I mean terminal as well as nonterminal, right? Now these kinds of productions are no problem at all because we allow such productions in Chomsky normal form, right?

Again when alphawith two or more symbols if it is of the kind A goes to B C, again that is not a problem, right? Because these forms are again allowed by our normal form.

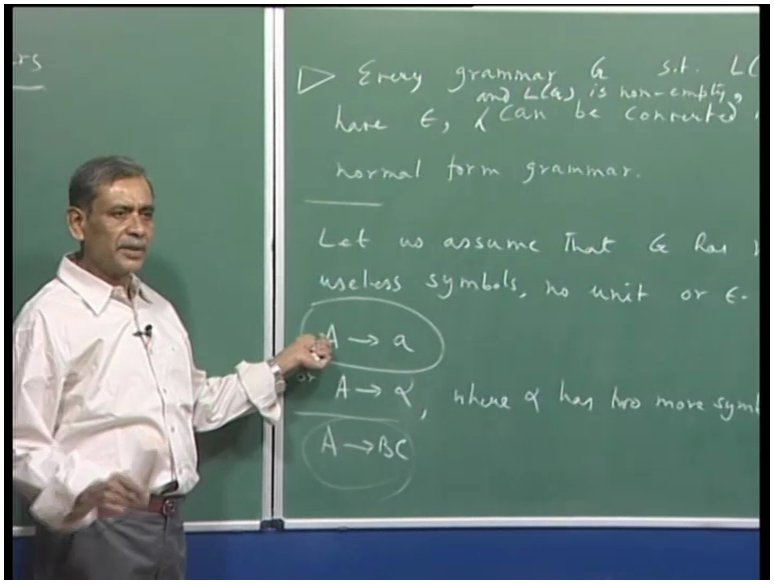
(Refer Slide Time: 33:08)



So the challenge in converting the grammar G into Chomsky normal form is to make sure that all productions A goes to alpha wherealpha has two or more symbols they should look

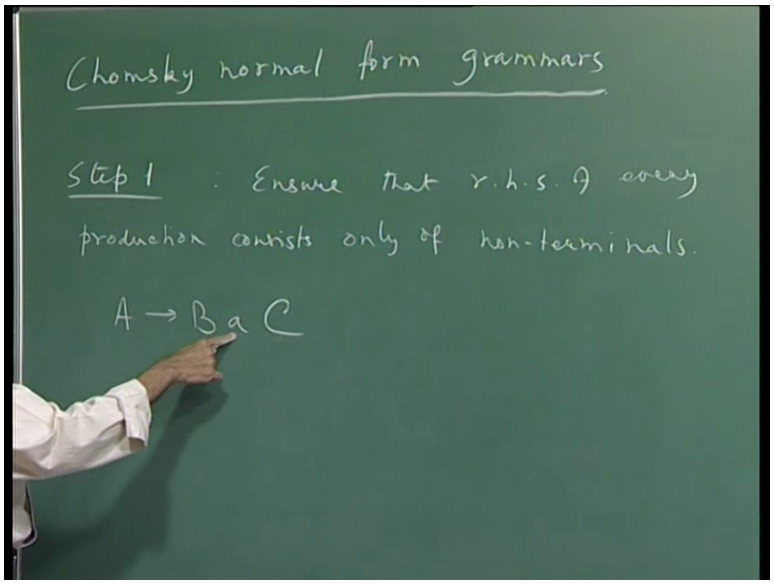
like or you know we should do something to those productions such that ultimately we will have only these kinds of productions in addition to of course these kinds of productions.

(Refer Slide Time: 33:40)



So then the grammar will be in Chomsky normal form. So the step 1 in this conversion is we ensure that right hand side of every production consist only of nonterminals. So you know what we mean is that suppose we have a production of the kind B a C, so these two are of course nonterminals but here is a symbol l which is a terminal.

(Refer Slide Time: 34:49)

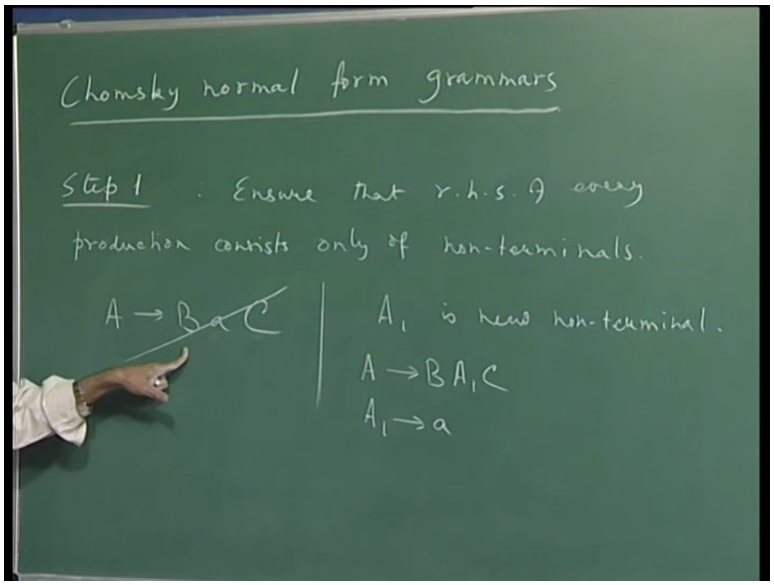


So I would like to ensure that the right hand side of every production consists only of nonterminals, right? Solike consider this example. So how we can do so? Just for this production let us see and that we can do. Wecan ensure this condition by introducing some

extra new nonterminals. So let us say A 1 is a new nonterminal. By that what I mean is A 1 is a symbol for a nonterminal which has not been used in the grammar sofar.

And now what I will do is that instead of this production I will write A goes to B A 1 C and I will add the production A 1 goes to a. So you see in effect therefore what I will have is that A goes to this right hand string, okay.

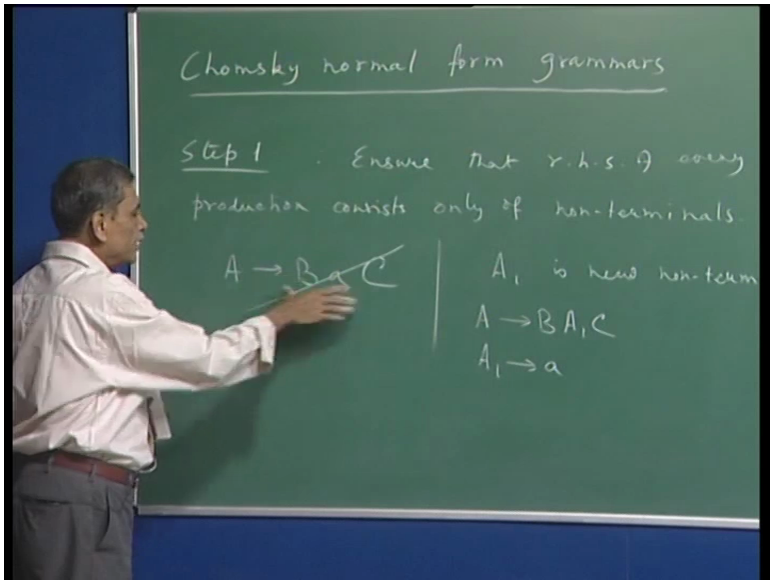
(Refer Slide Time: 36:04)



Sobasically what we are doing is whenever I find the right hand side where a terminal symbol is there what I do is, I introduce a new nonterminal and do thisinstead of that terminal I write in its place the new nonterminal that I have introduced. And I do that for all right hand side because you knowthere can be some other right hand sides also where the terminal a occurs.

So for all that I will introduce this new nonterminal symbol and then the addition of this production will ensure that the semantics will not change. In effect I canhave A generating this string, alright?

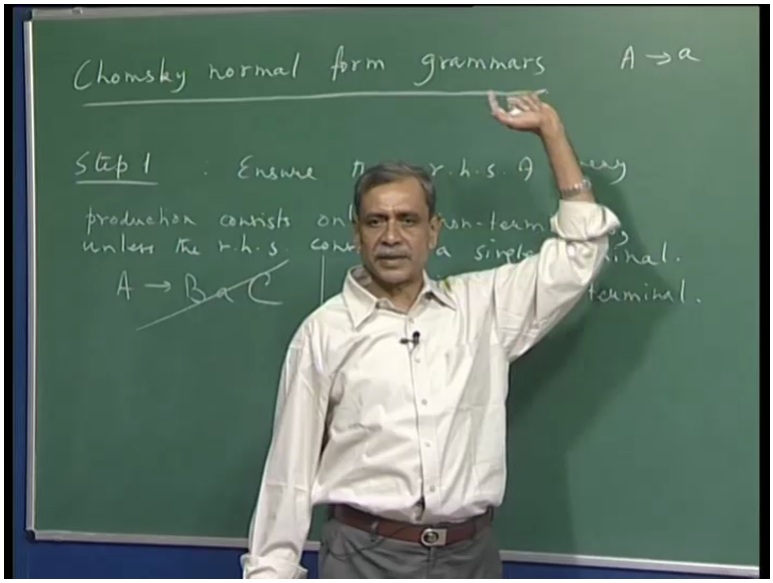
(Refer Slide Time: 36:59)



So this is step 1. Now step 2 is the step in which I will ensure that the right hand side consist precisely of two nonterminals. So how do we do that? Ishould (modi) modifywhat I have written here. Of course we do not mind productions of this form A goes to a single terminal.

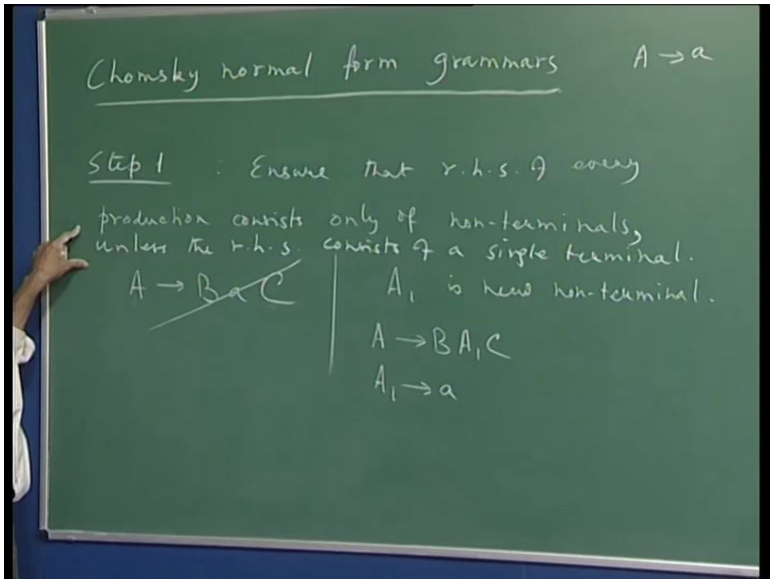
So I should write here, ensure that the right hand side of every production consists only of nonterminals unless the right hand side consist of a single terminal, right?I am saying it for the sake of completeness and being totally correct. Because right in the beginning I said we have no problems with productions of this kind and we leave them as such, right?

(Refer Slide Time: 38:15)



Then whatever the productions that we are left with of course they will have right hand side consisting of not a single nonterminal. But since I am making a statement here I write it in this form, okay.

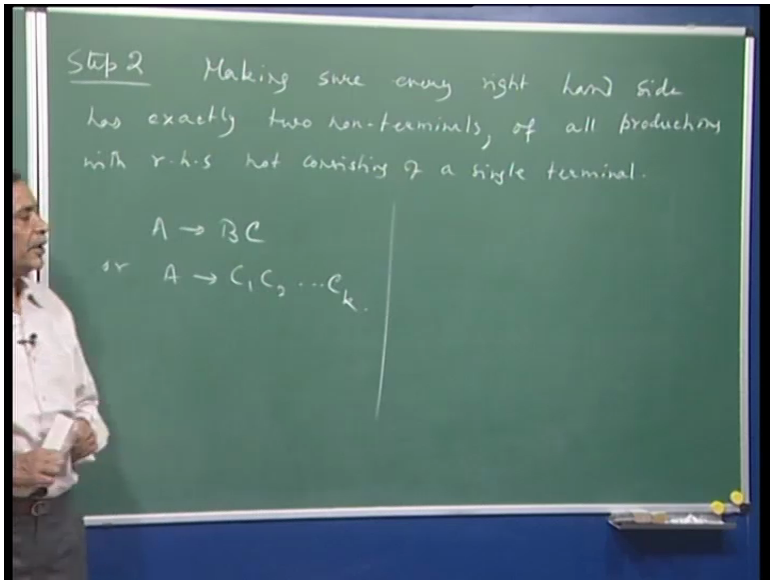
(Refer Slide Time: 38:34)



And now the step 2 is making sure that every right hand side has exactly two nonterminals,again for the sake of correctness I should modify this sentence that of all productionswith RHS not consisting ofa single terminal, right? So after step 1, leave out productions of this kind. Then what are the production that we have? Either they are of this form. That is either the right hand side is of size two which is alright because Chomsky normal form will allow such productions to be there.

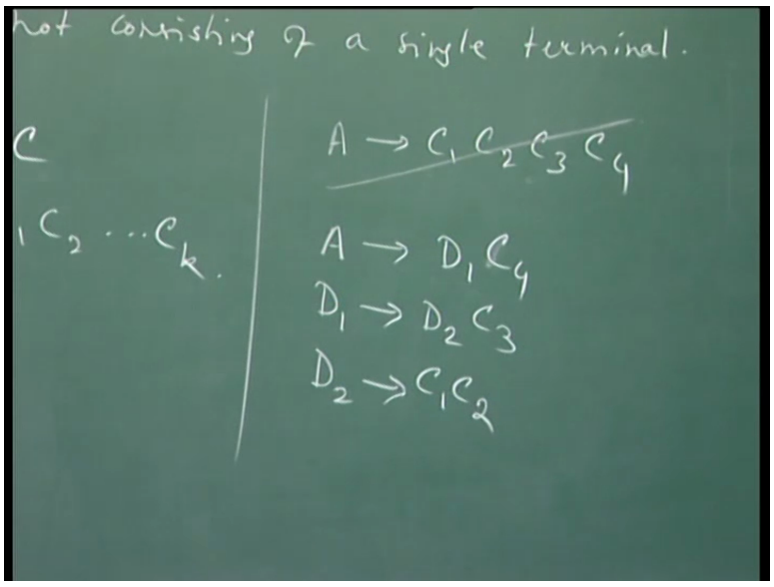
Or the right hand side has more than two nonterminals. So let us sayC 1 C 2 C k. The challenge is to eliminate such things where k is greater than 2 and to have equivalently productions which are of this kind where the right hand side consist exactly of two nonterminals.

(Refer Slide Time: 40:21)



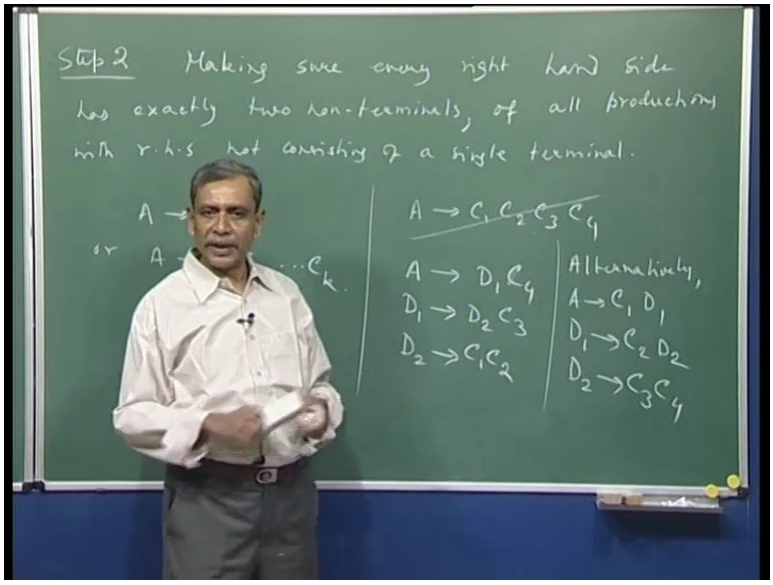
The idea again is fairly simple and you can do so by again introducing new nonterminals. And let me illustrate by taking let us say an example of where k is let us say 4. So let us say I hadA goes to C 1 C 2 C 3 C 4. This I would like to eliminate. In its place what I will have is I will write A goes to D 1C 4 and this D 1 should generate C 1 C 2 C 3. So I would write D 1 goes to D 2 C3 and D 2 goes to C 1 C 2, right?

(Refer Slide Time: 41:21)



Of course it could have done alternatively.A goes to C 1 D 1, D 1goes to C 2 D 2 and D 2 goes to C 3 C 4. Either way you could do. Now it is fairly simple that this idea extends to any k where k is you know 3 4 5 6 7 or 8. So in other words we remove one such production and add a number of productions where you see all these D's are new nonterminals.

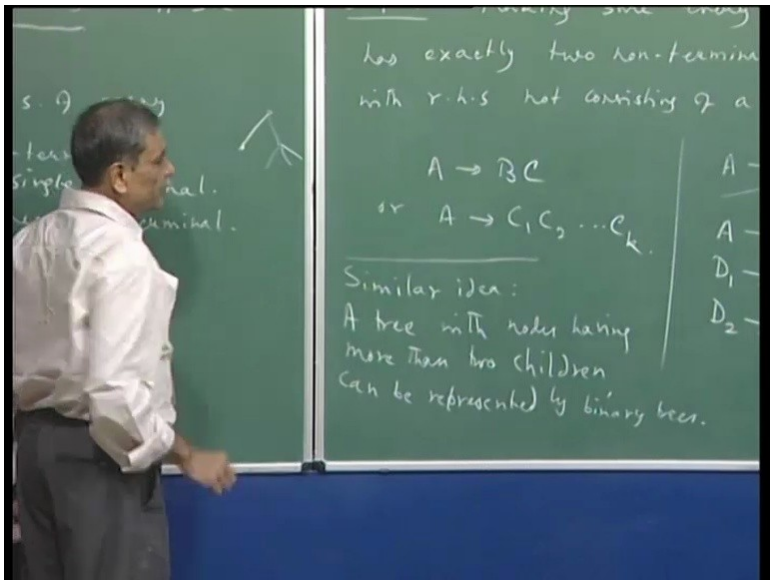
(Refer Slide Time: 42:17)



So again the idea will be to introduce some new nonterminals and making sure that the syntax or the condition on the right hand side is satisfied. I should mention in parsing that this idea is very similar to something you had possibly seen in data structures, right? Similar idea is let me just mention. I will not go into the details. A tree with nodes having more than two children can be converted in some, at least in our data structure we can represent such trees by really trees with exactly two children.

Can be represented by binary trees. Remember that the idea would be that this is the eldest child and then the rest of the children will come here. See this is what we are doing in this.

(Refer Slide Time: 44:02)



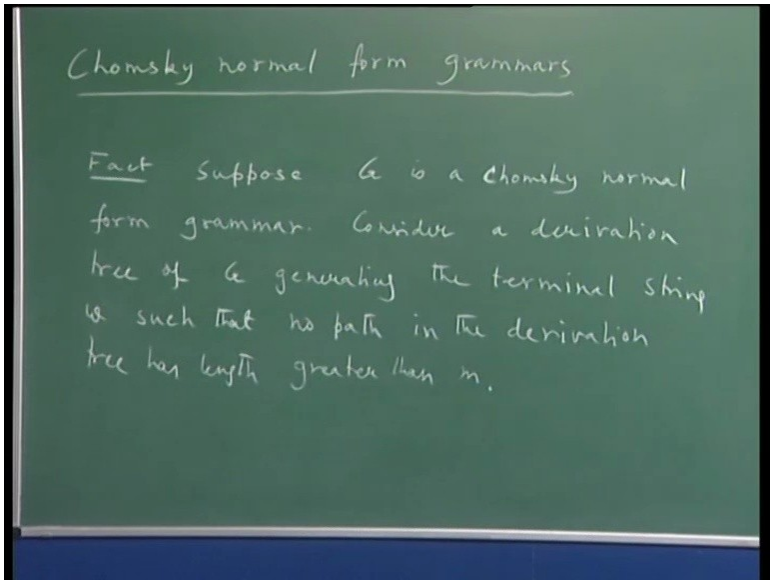
Something similar is what we are doing here. We have outlined a procedure and it will not be difficult to prove formally that ifone follows that procedure one will be able to generate, one will be able to obtain a Chomsky normal form grammar from any grammar satisfying the condition that the grammar generates a non-empty language and that non-empty language has a string other than epsilon and that languagedoes not generate epsilon, right?

So and that can be proved formally using as before induction, we will not do it so. Let me point out that the reason we would like to consider grammars in Chomsky normal form because we will see that later on some results will be fairly easy to prove. We willshow a pumping lemma kind of result. You recall that you are familiar with pumping lemma for regular languages. We will prove a pumping Lemma for context free languages and thereour starting point (goi) will be grammarsin Chomsky normal form.

Also we will use Chomsky normal form grammar to provide you with an efficient algorithm to check whether a string belongs to the language generated by a grammar or not. There again we will consider again without loss of generality that the grammar is in Chomsky normal form. In other words what I am trying to say is from now on without loss of (generity) generality we can always assume a given context free grammar is in Chomsky normal form.

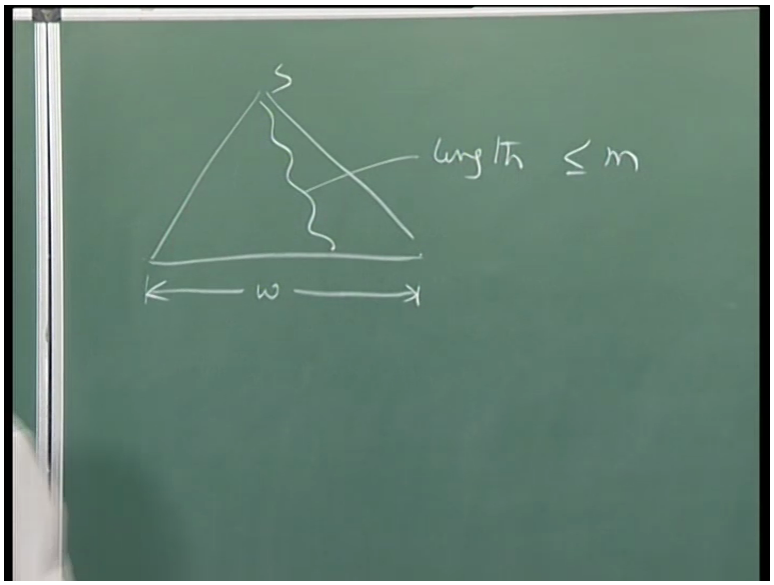
So let me prove to you one simple fact about Chomsky normal form grammars which we will use in the next lecture to prove our pumping lemma. And that fact is, let me say this suppose G is a Chomsky normal form grammar. And let us consider a derivation tree of the grammar G generating the terminal string w such that no path in the derivation tree has length greater than m.

(Refer Slide Time: 48:28)



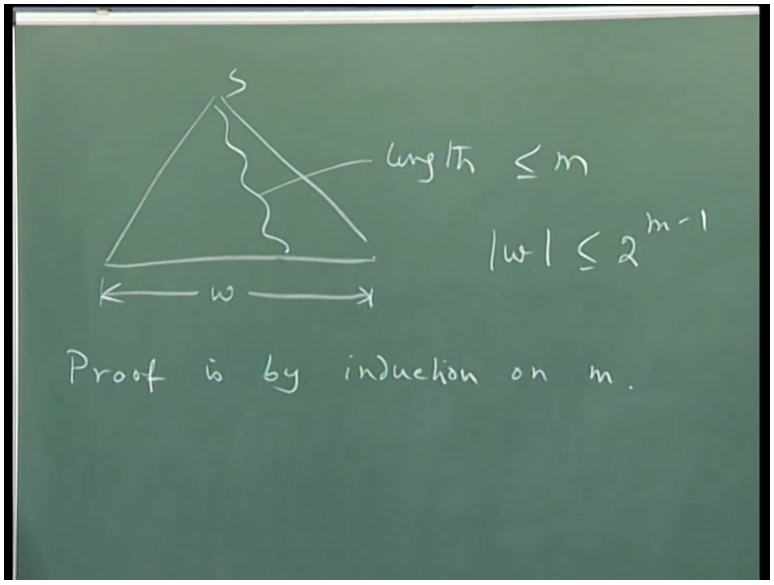
So consider what we are saying. Suppose G is a Chomsky normal form grammar and consider thederivation tree of G generating a terminal string w such that, so let me draw a picture, that this is S and this is w and no path here exceeds m, the length of the path.

(Refer Slide Time: 49:10)



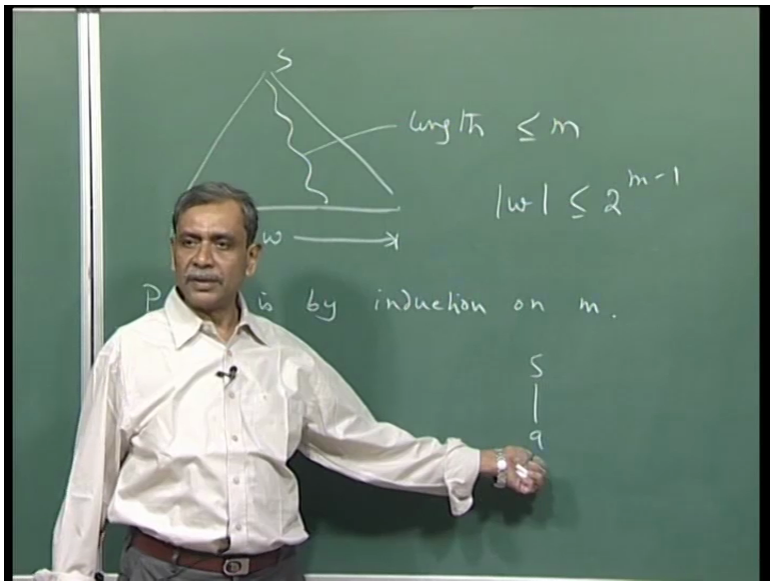
Then length of w is less than equal to 2 to the power m minus 1. So you see what we are saying in effect that if you have a bound on the largest path in the derivation treeof a string in Chomsky normal form grammar then you can provide an upper bound on the length of the string itself, okay. The proof of the fact is by induction on m. Proof of what? Proof of the fact that length of w is bounded by 2 to the power m minus 1, okay.

(Refer Slide Time: 50:27)



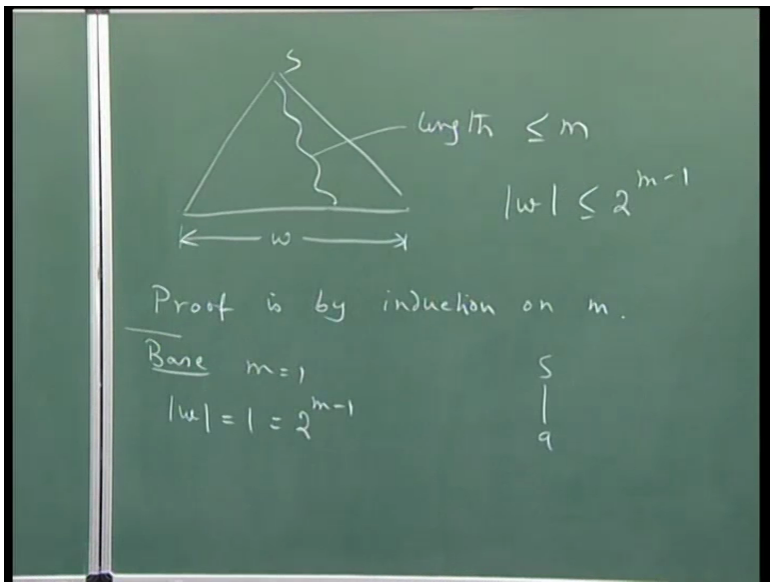
So base case is when m is equal to 1. When m is equal to 1 that means what? That meansno path in the tree has length greater than 1. Actually the only tree is of this kind that S directly deriving the terminal, right?

(Refer Slide Time: 50:58)



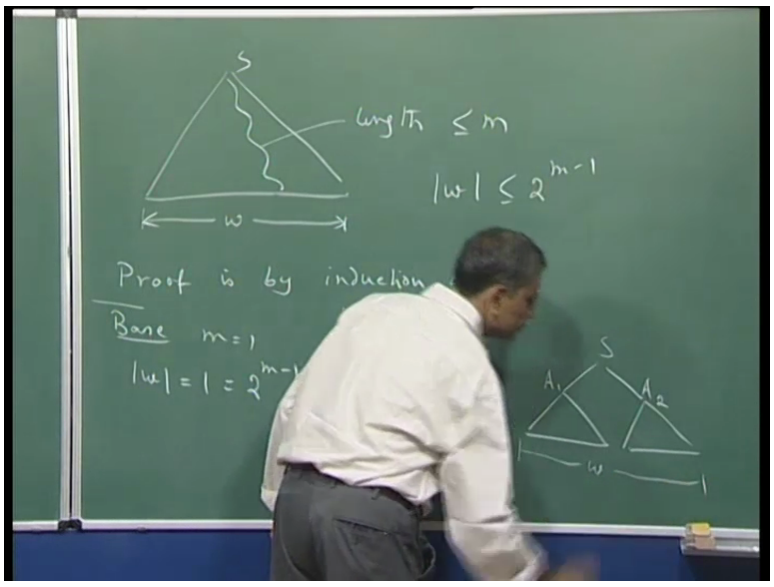
So this is a path of length 1. And trees of this kind will generate only strings of length 1, right? So then we are saying thatin case m is equal to 1 then surely length of w is 1 which is of course 2 to the power m minus 1, right? M is 1 so 1 minus 1 is 0. So 2 to the power 0 is 1, right?

(Refer Slide Time: 51:29)



So the base therefore is proved. And the induction step, so let me write the induction step which is also straight forward. Now consider a tree where no path is of length more than m plus 1 and such a tree is of course not the base case and the induction case. So clearly such a tree will be of this kind, right? So initially S will derive some two nonterminals A 1, A 2 and then some things are happening, right? So basically the total string is w, alright.

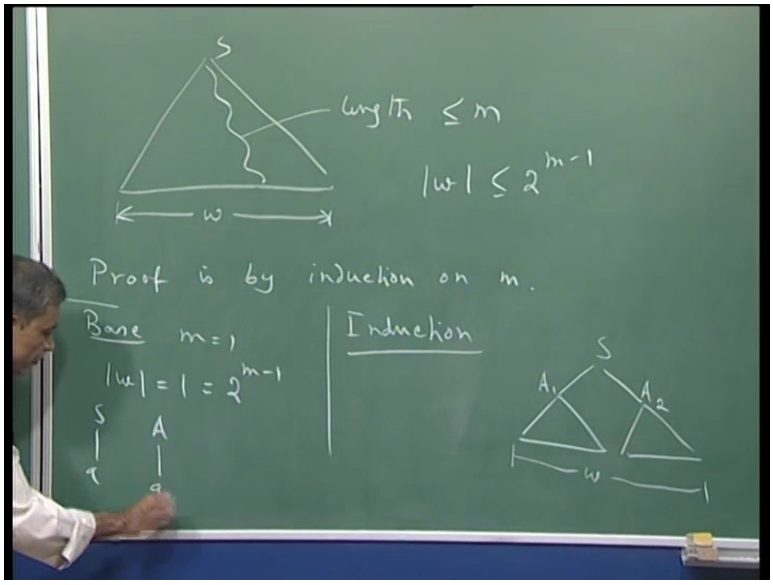
(Refer Slide Time: 52:38)



Now if whatever I am saying is generalized a little bit then the induction is very simple to prove. See I said derivation tree, now derivation tree normally means starting with S. But we can extend this notion of derivation tree to mean derivation starting from any nonterminal,

right? So in that case base again will be not just trees of this kind, also trees of this kind will be allowed.

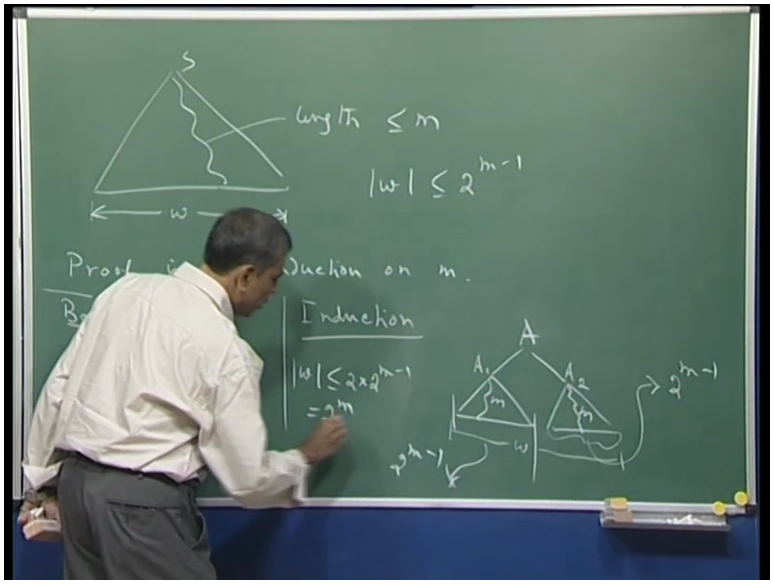
(Refer Slide Time: 53:20)



When I said by derivation tree I mean derivation starting from any nonterminal, right? So in that case for induction in generalinstead of S, I would write some A and then I inductively apply the fact that the (lar) longest path in either of these is of length m or less. So therefore this path has to be of length by induction 2 to the power m minus 1.the length of this path is again bounded by 2 to the power m minus 1 because these subtrees have no path greater than m because in the original tree no path is greater than m plus 1.

So therefore this path is at most has length 2 to the power m minus 1. This path, the right hand path has at most length 2 to the power m minus 1. Sotherefore size of w is bounded by 2 into 2 to the power m minus 1 which is equal to 2 to the power m.

(Refer Slide Time: 54:42)



Which is m is of course one less than the longest path m plus 1. And that proves this fact. And this is one result that we are going to use in the next class to prove our pumping lemma.