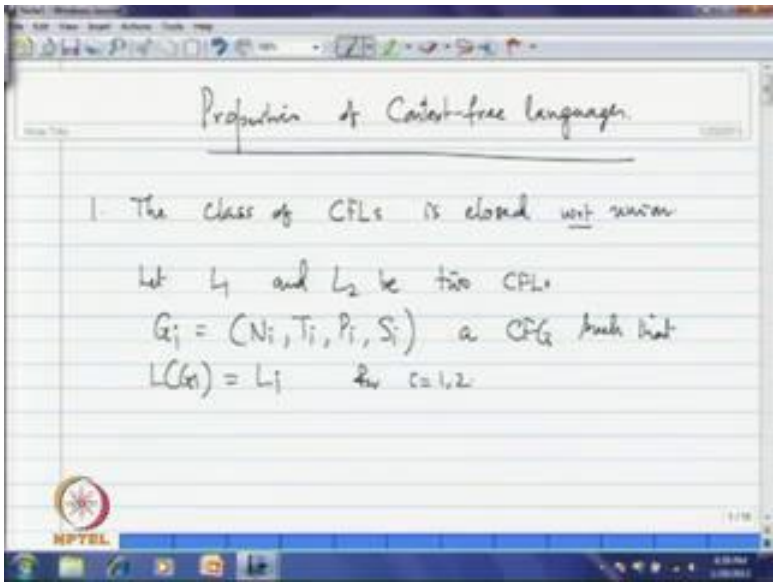
Formal Languages and Automata Theory
Prof. Dr. K. V. Krishna
Department of Mathematics
Indian institute of Technology, Guwahati

Module - 9
Properties of CFLs
Lecture - 1
Properties of CFLs

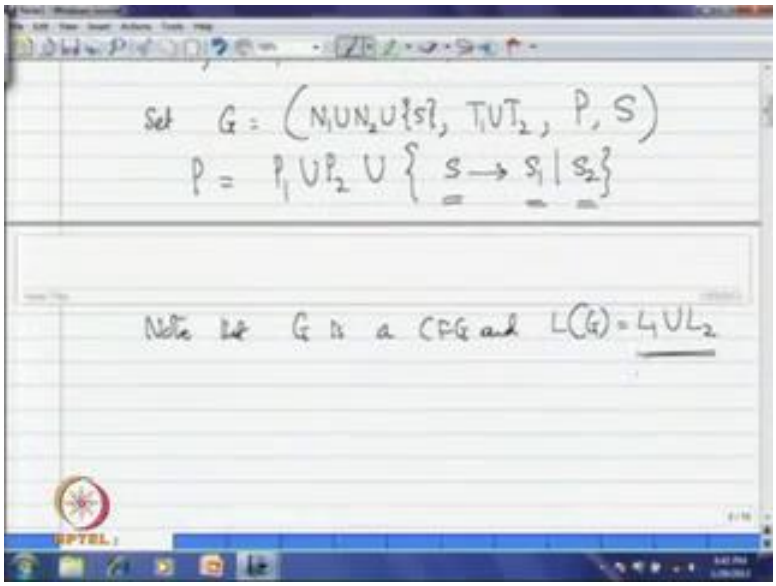
In this lecture, we will discuss properties of context free languages and first you observe certain trivial properties. And then we will ascertain some properties connecting to context free languages which you know you will look at in parallel to what are called properties concerning regular languages.

(Refer Slide Time: 00:52)



So, this properties of context free languages, so 0.1 it is very clear from the definitions of context free grammars you can quickly understand that the class of context free languages is closed with respect to union. This proof is very simple if you take two context free languages, let L 1 and L 2 with 2 CFL, so that means, you have say G 1 is equal to N 1 say T 1, P 1, S 1, a CFG such that language generated by G 1 is L 1 And similar you can have G 2, so let me call L i for i is equal to 1, 2 I have two context free grammar.

(Refer Slide Time: 02:39)



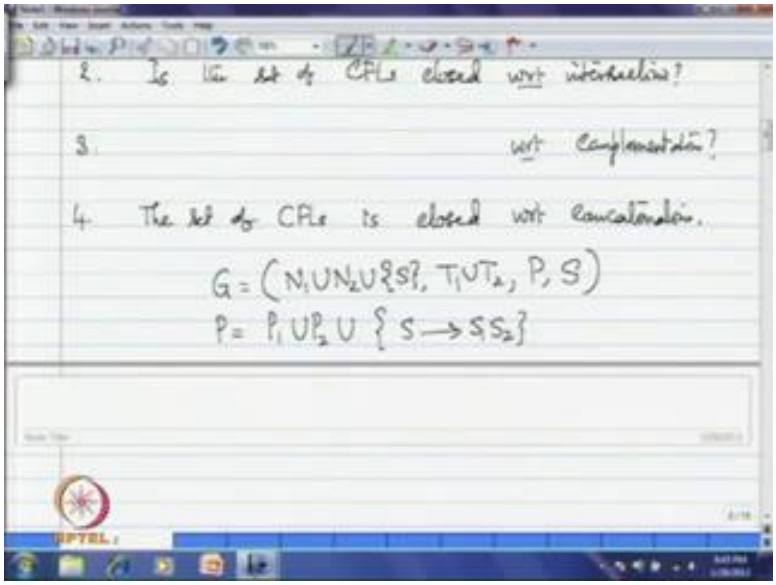
Now, you can design, so set G 2 be you know this N 1 union N 2 union singleton S, you take new not terminal symbol now this language, it is certainly over union of terminal symbols and now production rules P i, let you know and S. So, in this case your production rules you take P 1, union P 2 and you give U rule S goes to S 1 or you know S 2, if you set grammar like this you can clearly see the productions in P 1 are in the form for context free grammar and for P 2 also.

Now, what are the two U rules I have introduced S goes to S 1 and S goes to S 2 in thiS 2 are you know satisfying the conditions for the context free grammar and thus we can see not that this G is a CFG. And now we can ascertain that the language G is the CFG and the language generated by G is simply L 1 union L 2. That one can observe right because if you take any string generated by this grammar you have to start from the start symbol s and you have to take one of the branches S 1 or S 2.

If it is going to S 1, then you can generate the strings of L 1, if it is going to S 2 you can generate the strings of L 2. So, that you know any strings, which is generated by this in L 1 union L 2 and conversely what are the string which is in L 1 and L 2 you know either it will be L 1 or it will be in L 2. So, you know if it is in L 1 you can generate through that to S 1 thus you can generate that through S also, so that it will be in L of G similarly. So, you can observe you can prove that this construction was to show that context free

grammar context free languages are close to with respect to the class of context free language is close with respect to union.

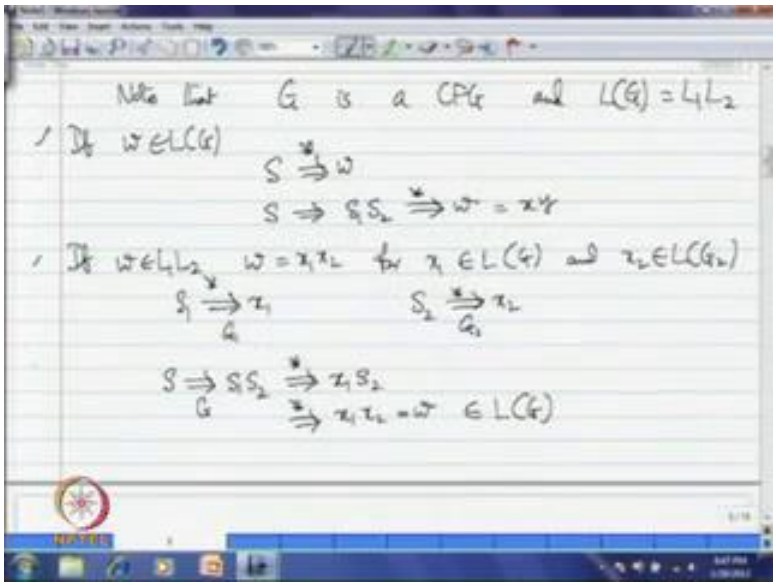
(Refer Slide Time: 05:13)



Now, if you look at intersection that will be a question to us the question is the set the class of CFLs closed with respect to intersections and you can question that is this closed with respect to complementation etcetera? The set theoretic properties related will address these points little. Now, if you ask about these are set theoretic thing, now if you look at the concatenation again, so the result is the set of or the class, whatever the set of CFLs is closed with respect to concatenation.

How as earlier you know you let L 1 and L 2 be 2 CFLs considering the respective contract text free grammar say G 1, G 2. And now when I am setting G here, I consider the protection rules you know of this form thus this is T 1 union T 2 union in this S goes to S 1, S 2, I will consider. So, here you will require the non terminals of the first one of the second one union new symbol S and our concatenation any way this T 1 union T 2 and the P S.

(Refer Slide Time: 07:29)



If you have this again this new production rule I have a single rule here, this acts as the criteria for context free grammar, so that you know can note that the G is a CFG and you can observe that language generated by G is L 1, L 2. How come, if you start from S you have to use this is if I take any word which is in L of G I have to start from S and produce that word w.

If w is in L of G, I require this, but you look at the first step from S, I have only one rule, so that means, I should use this S 1, S 2 and after that infinitely many steps, I might be producing this w. Now, corresponding to the non terminal whatever that we are generating the terminal string, the portion of w that you can generate in L 1, similarly corresponding to S 2 whatever that we are generating portions is w that is in L 2.

So, that this w can be written as x y for x in L 1 y in L 2, so that it is in L 1 and L 2 and similarly you can see the converse; that means, if you take any string in L 1, L 2 you will have to strings say x 1, x 2, x is in L 1, x 1 is in L 1, x 2 is in L 2. Now, since x 1 is in L 1 you can generate that in the grammar G 1 and x 2 is in L 2, you can generate that in the grammar G 2 and thus you have the respective production rules to generate this x 1 and x 2.

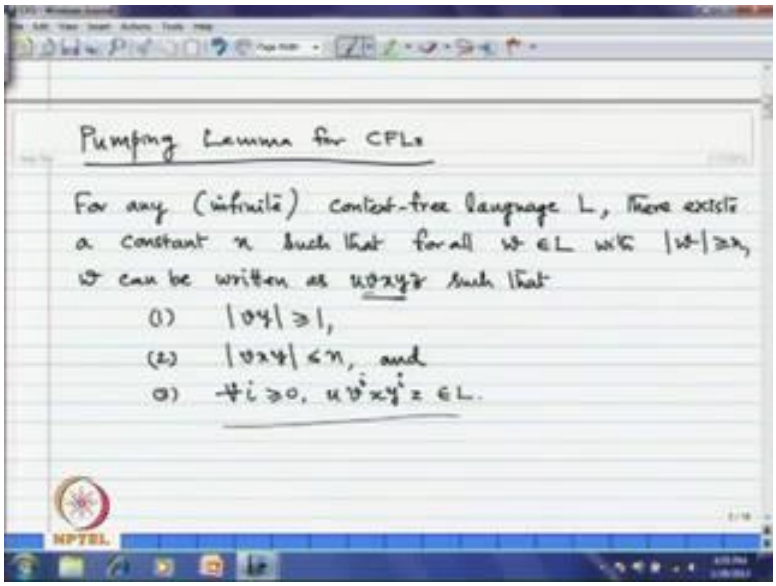
Now, if you start the derivation what are the derivation that we have got from S 1 to x 1 you have got a derivation in G 1 and similarly for x 2 you have got a if you start with a string w is in L 1, L 2 is w is are the form x 1, x 2 and for x 1 is in L of G 1 and x 2 is in

L of G 2. So, I have this derivations and now you just have to start use this production rule S 1, S 2 and then use this derivation and produce from S 1, because the derivation G 1 will be derivation G also.

Because, all the production rules of G 1 P 1, the production rules, they already there in G, so I can make that as a derivation in G also. So, I have x 1 this and after again finitely many steps S 2 can be made in this x 1, x 2 thus you see that you have a derivation this in G, so this is a derivation in G. So, that this x 1, x 2 that is what is w is in L of G, so we can observe these things.

So, this I S 1 side to observe that L of G is continue in L 1, L 2 this is other side to observe L 1, L 2 is continue L of G, so that this grammar what we have constructed here that generates L 1, L 2. So, that you know the class of context free languages is close with respect to concatenation, so regarding this set theoretic questions, whether the class of CFLs is close with respect to intersection complementation, if you are going to such questions and many other properties.

(Refer Slide Time: 11:17)



Let me just give you a pumping lemma for CFLs which you know it is which something parallel to the one we have worked for regular languages. So, let me first state the lemma because, I dint have to give much introduction for this because, you have already worked for such pumping lemma for regular languages. The philosophy is similar, what pumping

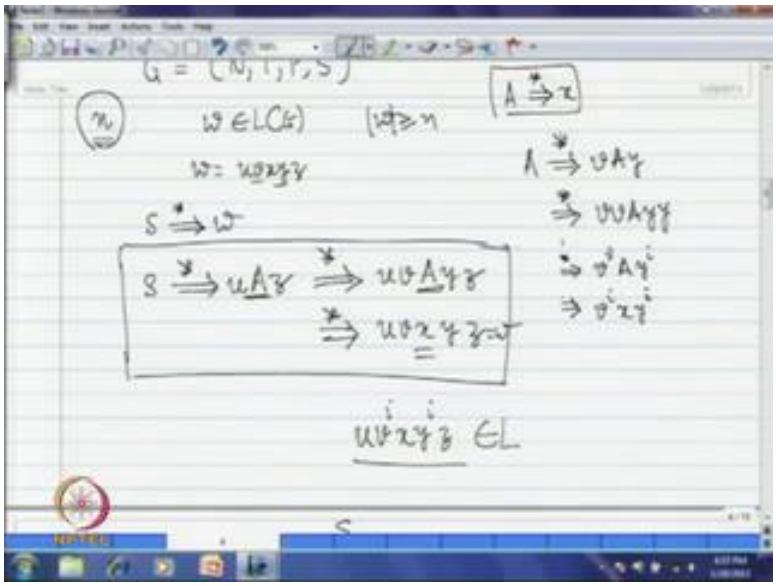
lemma says for any infinite context free language L of course, this we will be observing that it is super finite languages also, but any way I will work for infinite language here.

So, for any context free language L there exist a constant n such that for all w in L with length of w is greater than or equal to n, this w can be written as u v x y z such that the length of this v y a second component and the fourth one. Length of v y is greater than or equal to 1 and length of v x y the middle 1, this v x y these 3 components that is less than or equal to n. And if you pump the strings v and y simultaneously for i number of times; that means, for all i greater than or equal to 0, u v power i x y power i z is in L all these strings will be in L.

So, now you look at in case of regular languages every string you will be able to split into three parts such that the middle string is non empty and if you pump that string for any number of times the resultant strings will be within the language. That is what we have observed in case of pumping lemma for regular languages, here what is happening the string can be split it in to 5 parts and among these 5 parts say u v x y z, when we have splitted.

This v and y at which one of them is non empty when I am saying this carnality of v y is greater than or equal to 1; that means, at least one of them has to be a non empty string and the middle portion that v x y. These 3 strings together you can always produce such a way that this length is less than or equal to n, what are the number n is existing and now if you pump v and y simultaneously for any number of times the resultant strings will be in L. So, the philosophy is similar and the applications are also similar now I will give you an idea that how you will produce this result.

(Refer Slide Time: 14:10)



Now, you look at if L is a context free language you have a grammar corresponding to that say some N,T,P,S and if you now we have to identify the number and I have to come up with the number n I tell you what should be that number. And now what how this number should be if you take any string w in L of G whose length is greater than or equal to L i should be able to write this w in 5 parts u v x y z, so this v y at least one of them is non empty.

So, that is what is the break up we require and if you pump this two portions any number of times, simultaneously the resultant string should be in the language. Now, w is an L of G you have a derivation for w, what I will show here is I can always have this derivation of the form. Because, I will find the number n such that this derivation can be for this w, the derivation I will get things likes this say u some non terminal A z, I can always produce things like this or this u and z you may be producing later also does not matter.

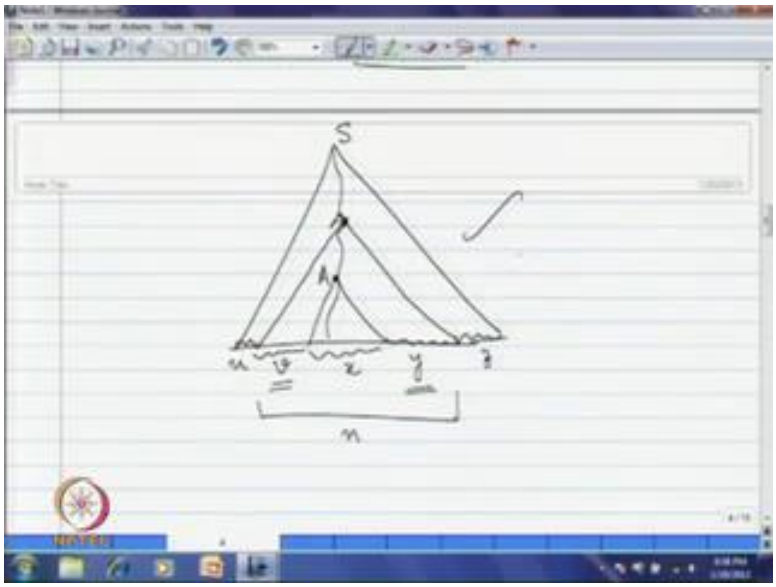
So, I will have a non terminal symbol this non terminal symbol will occur after you know after production of some strings, that is you know u v A y z and then this A I will be able to terminate u v x y z. So, if I can show that I will have a derivation that the given context of this form, then I am through this is what is w why because, once I have situation like this you see what is happening this non terminal A is producing the string v A y.

Now, this A once again we can produce v what is called v A y after finitely many steps once again, so that we will have w y and if you once again, if you use you are getting triple v A triple y and also you see that A is producing x infinitely many steps. So, I have this also a produces x infinitely many steps, so this is also there in the part of the derivation.

Thus after finitely many times, if you use you will get you know v power i A y power i you can get like this and then, if I use this I will get v power i x y power i because A can be terminated with the terminal string x. So, I will get always like this thus you see I have a derivation in this grammar of this sort, now you look at what is happening here in this derivation if you observe.

If I can have such a derivation what is here a non terminal symbol which as occurred once is a pairing once again in the derivation of a string, whose length is bigger than or equal to that n. So, that is what is essential the fundamental idea, once a non terminal symbol reoccurs, I would be able to produce derivations for v power i x y power i within the portion of that and then you know I can produce the strings in the required as mentioned in the statement, in this is clear we can get the idea how we have to proceed.

(Refer Slide Time: 18:37)



Suppose, if I draw the derivation tree for this, this is a start symbol let us assume this way and what I have here I am getting say for example, once again if I am getting A here within the per view of this A, whatever is because you see context free grammar. So,

every non terminal symbol will have production rules, there is nothing to do with other symbols addition to that, for A if it is producing x and this A if it is this tree, if it is producing the portion v y this is the leaf of this derivation tree.

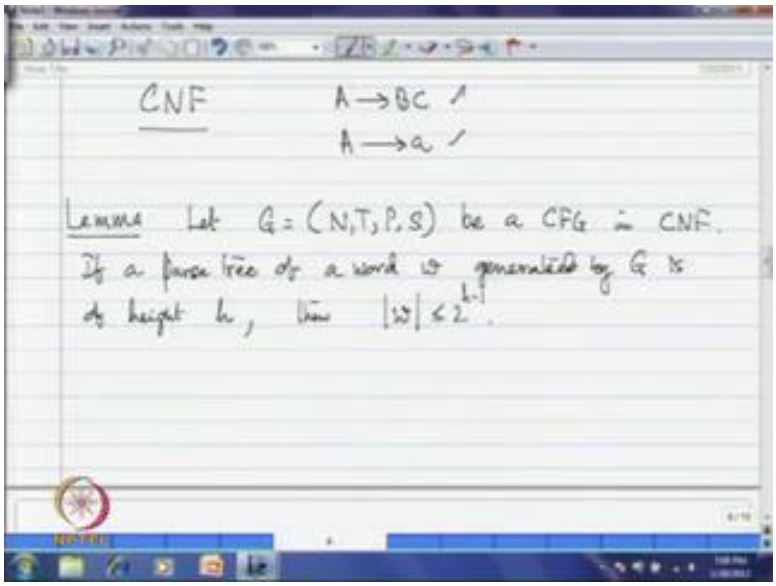
And now what are the remaining portion here that is u z this portion, now the derivation tree if a can have this A reoccurring, now connecting this sub trees will be having this way. And thus whatever the way that I have described here you know this such A derivation A as occurred once again and thus I will be able to produce the strings of the form u v power i, x y power i, z in the language the language generated by j.

So, now you look at whenever you know if I am looking forward or repetition of a non terminal symbol, now you put a cap on it; that means, you consider number such a way that if you take any string. Whose length is bigger than that number you needs derivation tree you know you require this non terminal symbol to be repeated, there should be a repetition of non terminal symbol, this is the fundamental idea that I have just discussed.

Accordingly you know, so what essentially we have to set the number any string if we look at the derivation tree in a branch I should have a non terminal symbol repeated taking this in to count. If you consider that number then what are the string in the language whose length is bigger than that particular number, you take the derivation tree of this form then you can manage to show that you know it is splitted into u v x y z.

And of course, we have to observe that we have to have this split in such a way that at least this v r y should be non empty and of course, the extra criteria we had put the length of this should be less than or equal to that number all these things we require, these are all extra conditions. But as a you can understand that it will be splitted into 5 parts in which this v and y portions are at least non empty and u v power i x y power i z for all i this will be inside L that is what we have to observe. So, what I do in this connection to a get this extra properties instead of considering an arbitrary context free grammar, I will consider a context free grammar which is in CNF.

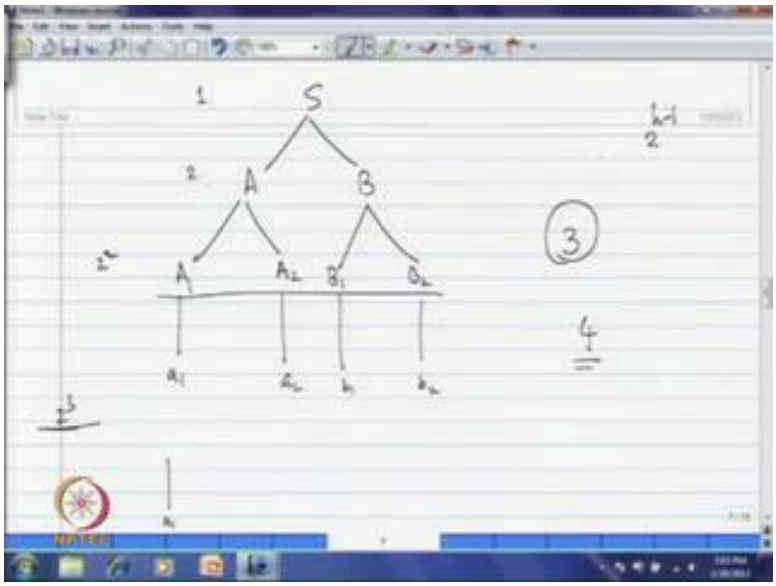
(Refer Slide Time: 21:56)



You know Chomsky normal form, so in Chomsky normal form every production rule is of the form you see that A goes to BC, A goes to a, of course, if connecting grammar, if it is in Chomsky normal form in the language. If you have empty string except empty string everything else you can generate you know that, so these are the types of production rules that you have and now if a grammar is in Chomsky normal form, you know this you can quickly understand this property.

So, let me write this lemma, it is not difficult for you to understand let G is equal to N, T, P, S be a CFG which in CNF Chomsky normal form, you are generating CNF. If the parse tree of a word w generated by G is of height h, then you can observe that the length of this is less than or equal to 2 power h minus1. You see what happens I just of course, this you can prove by induction, I will just demonstrate illustrate this example.

(Refer Slide Time: 23:56)



For example I have like this, so since it is CNF I can have say like this if the situation is once again I have non terminals, I will have like this then say for example, a 1, a 2 and if in the case you know a 3 or B whatever. Now, you look at because it is in CNF, you know the length of what happens now if I consider the complete binary tree here. So, the possibility is I can in fact, extend if I am looking for the height a 3 say b 1, b 2.

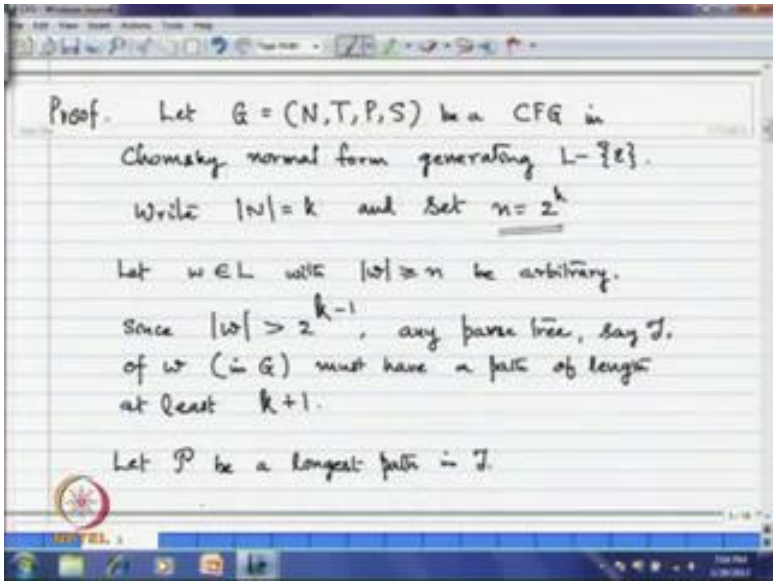
Now, height of this is 1, 2, 3, is the height and now you see the maximum length of because I have filled all the nodes in this complete this thing, here I can get maximum length is 4, because you see at this level you have node 1, as you have in complete this thing you have 2 nodes here, here 2 square nodes will be there. If you go one more level 2 cube nodes will be there and now once you have all the levels all the nodes are available; that means, in this 2 power you know 2 cube nodes are there.

And all of them are getting terminated say for example, a 1, a 2 and so on, a, a 8 and now you see the length of this thing is 8 that is what is the maximum. So, to this level if you come now you understand now at any level suppose if you are you are terminating at this level itself, now the lesser number but the maximum you look at that if it is of height you know here 3 you see that 2 square here. So, if it is of height 3 you know 2 power h minus 1, that is the maximum length string that you can produce.

If it is of height h any parse tree of height h can produce a string of length maximum 2 power h minus 1 maximum this is the length. Now, In fact, you can observe this result by

induction the by induction on this binary trees you can observe this is an I will use this result.

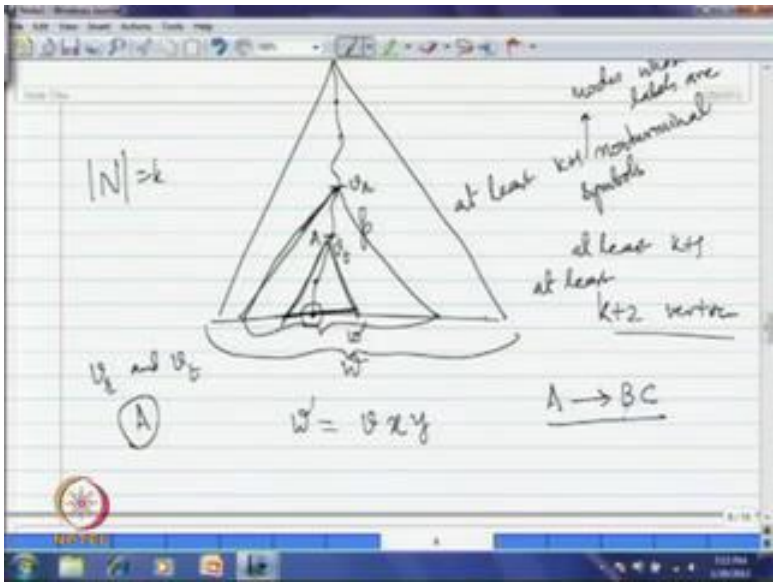
(Refer Slide Time: 26:41)



Now, So, let me consider this Chomsky normal form in the present context, so for proof of this result let G equal to N, T, P, S be a context free grammar in Chomsky normal form, generating l minus singleton Epsilon. The non terminal sets suppose the size is k and set the number n to be 2 power k, now you will understand because just now the result what we have discussed is if it is of height h the length of you know the string will be maximum 2 power h minus 1.

So, I will consider this n equal to 2 power k, now you consider string w in L of length greater than or equal to n, take any string whose length is greater than equal to n. Now, since it is the length of this is greater than two power k minus 1. Any parse tree T of w in G must have a path of length at least k plus 1, because the length of this thing is greater than 2 power k minus 1. So, you should have at least one path of length at least k plus 1, now let me draw a picture in that in that is sense.

(Refer Slide Time: 28:06)

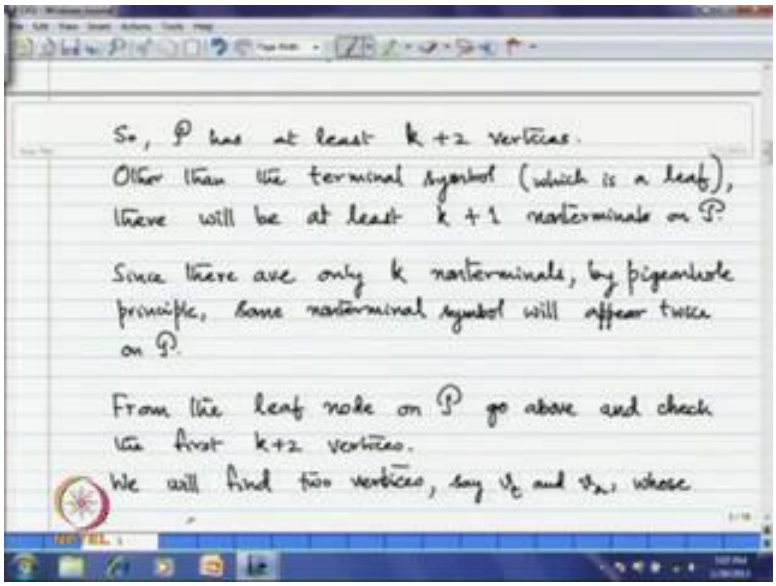


So, you consider a parse tree of this w, so you should have a path at least k plus 1, because whose length is bigger than or equal to n, since it is greater than 2 power k minus 1, I should have a path of length at least at least k plus 1. Now, the last node is any way terminal node, some terminal node will be there and then here before that a non terminal nodes will be there, so these are all non terminal nodes on this of length at least k plus 1 length.

Now, you look at so let me say that path be P, so how many vertices will be there, if it is of length k plus 1, there are k plus 2 vertices will be there on this path, there are at least k plus 2 vertices. Now, you know the last node is a terminal node and at least then k plus 1 non terminal nodes will be there on this path P, on this path P at least k plus 1 non terminal.

So, whose labels are non terminal symbols, so at least k plus 1 node whose labels are non terminal symbols, so you will have like this and thus you see if there are at least you know k plus 1 nodes whose labels are non terminal. But how many non terminal symbols we have there are only k non terminal symbols, therefore, at least one of them by pigeonhole principle at least one of them should be repeated.

(Refer Slide Time: 30:20)

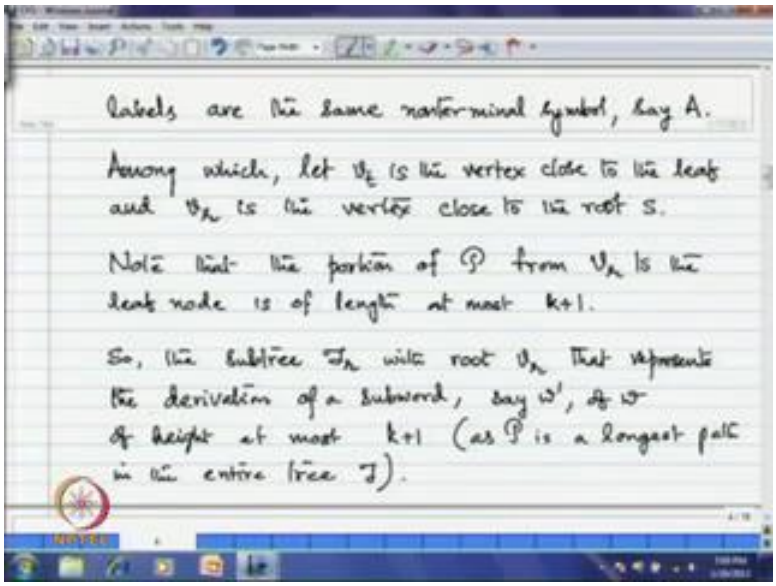


Now, you see Since, there are only k non terminal symbols by pigeonhole principle, some non terminal symbol you know will appear twice on P, now what I will ask you to do you just go from you know this terminal node to above and check you know k plus 2 nodes. Other, than this suppose, if you check k plus 1 nodes, you can certainly get one node repeated, so whatever that it is getting repeated for example, this node label and this node label assume they are same.

For example, say you know the label is A, if I am writing and the node say it is close to this terminal node; that means, let me call v t and the node which is close to the root and I may call v r whose labels are same. The label is the label of v r and v t assume it is A, labels are same, so what I asked you to do from the terminal node you go above till say for example, k plus 2 nodes I means.

Other, than this terminal nodes if you just visit k plus 1 nodes, before that you can realize that 2 nodes will have the same label, before you reach two k plus 2 nodes from the bottom, so let me assume the first node which is getting repeated. So, v t label and v r label I am just writing because v r which is close to root and v 2 which is close to this whose labels are A. So, from the leaf node on t go above and check the first k plus 2 vertices will find two vertices v t, v r whose labels are same non terminal symbol say A, this how we have taken a label.

(Refer Slide Time: 32:12)

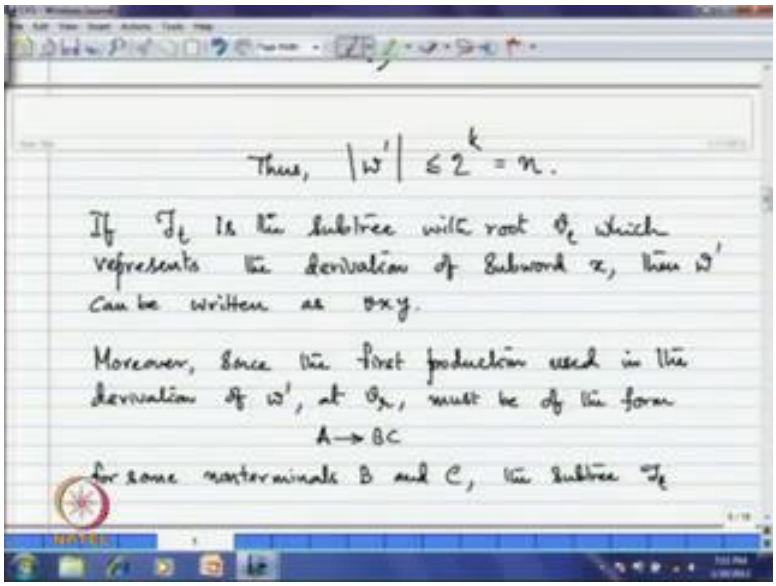


Now, note that the portion of P from v r 2 leaf node is of length at most k plus 1. So, the sub tree t r with root v r, that represent a derivation of sub word say w dash of w of height at most k plus 1 from here whatever is this w is the entire word. Of course, this sub tree let me call it has w dash, now you look at how many nodes I have visited k plus 2, set the maximum k plus 2 nodes I have visited and therefore, the length of this path is you know the length of this path is k plus 1.

Now, you see the height of this tree is maximum k plus 1 why is because, the height of this is maximum because P, what we have consider that is the longest path in the entire tree, therefore you know here this portion should be longer. Then, any other path within this sub tree also, if I have something else then it cannot be longer and therefore, the height of this thing is at most k plus 1 thus what you have the length of this word is less than or equal to 2 power k.

Length of the word because whose height is maximum k plus 1 and from the result what I have just discussed, the length of the yield. This from lemma, you can observe that it is less than or equal to 2 power height minus 1 and therefore, the length of this w dash should be less than or equal to n.

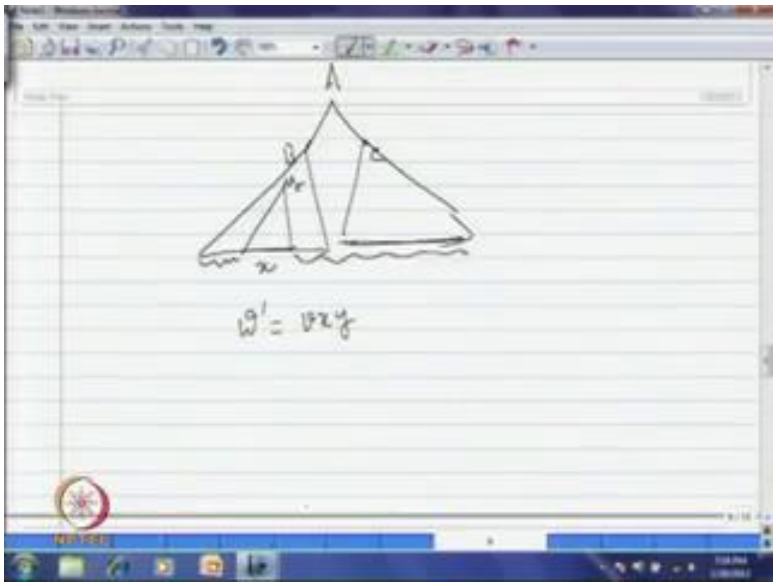
(Refer Slide Time: 34:01)



Now, you see if you consider the sub tree root at this v t, if t is a sub tree with root v t which represents the derivation of sub word x, then w dash can be written as v x y. Because, this w dash is the yield of the sub tree who rooted at v r, if you consider the sub tree rooted at v t, the yield of this sub tree if I call it has x, then this w dash can be written as v x y the centre portion w dash.

So, this portion is x, now you understand this point because I have considered CNF, this is the node A and this A goes to say for example, BC because, it is in CNF, I have you know this sub tree here which is marked rooted at this v t. This should be a sub tree of you know which is rooted at B, because I am having you know two branches here from this node. So, the sub tree will be entirely within the sub tree rooted at B or it should be within the sub tree rooted at C.

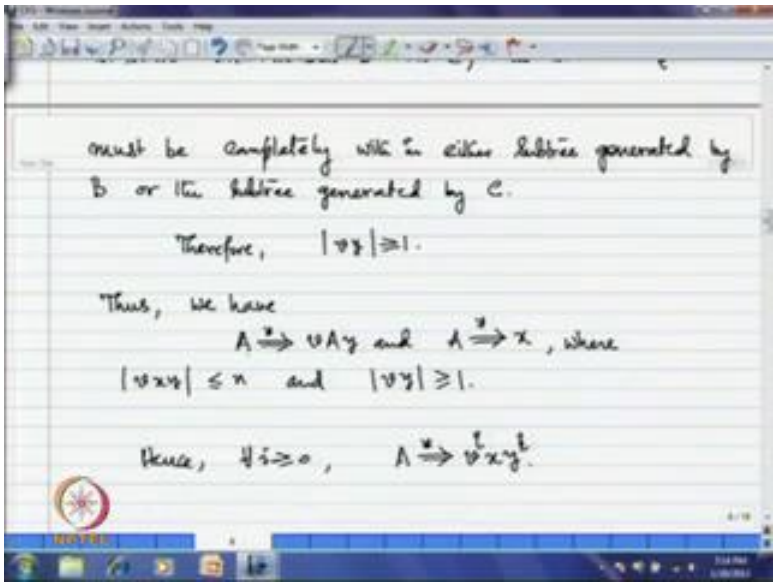
(Refer Slide Time: 35:38)



So, you look at that say A now, the rule say for example, BC is like this, now you have a tree connecting to this B and you will have a tree connecting to this, there will be not anything common. Now, what is the sub tree, I am showing rooted as v t, this v t will be therefore, within this or it cannot be you know this v t cannot be in the common of these two.

And therefore, from this you can observe that other than this portion x you know when I have C here, suppose it is within B this C should get terminated and at least you know some string some yield you will have so when I am writing w dash is equal to say v x y. So, the remaining portion other than x should be at least you know you should have at least one symbol and thus it has to be non empty the length. So, this is the argument I am placing here, since the first production used in the derivation of w dash, let v r must be of the form A goes to BC for some non terminal symbols B and C.

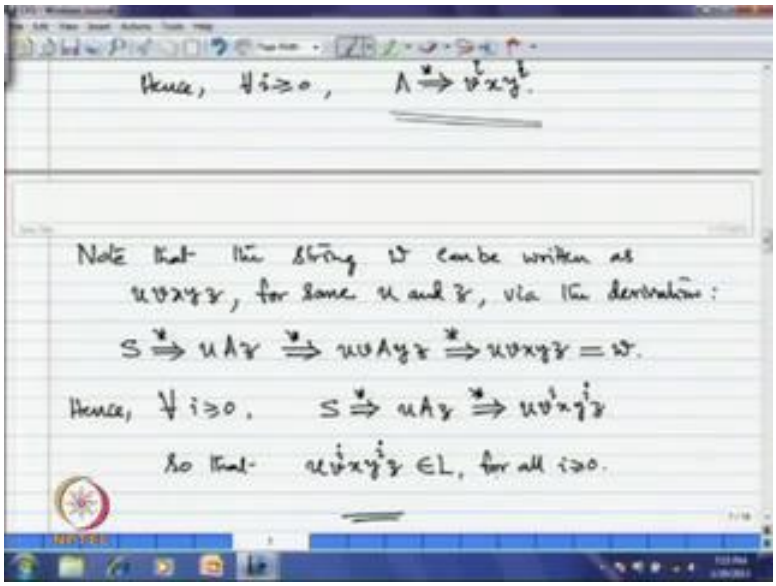
(Refer Slide Time: 36:45)



The sub tree this T t must be completely within either sub tree generated by B or the sub tree generated by C and therefore, you see at least one of this B or y should be non empty. So, the string v y length should be greater than or equal to 1, thus what do I have this a infinitely many steps I am producing v A y and this a infinitely many steps we are producing x.

Where v A y length, that is what is w dash is less than equal to n, that is what we have observed and also with this argument we have observed that v y length should be greater than or equal to 1 .And hence because of these two derivations we have for all I greater than or equal to 0, we have a produces infinitely many step B power i x y power i. So, this is what essentially I have mentioned that, we will give a derivation within the sub derivation that a non terminal symbol will be repeated and it produces strings of this form v power i x y power I, once I have this then I am through. So, I can compute this result.

(Refer Slide Time: 37:56)



Now, note that the string w can be written as u v x y z for some u and z because this is the rest of the portions other than the sub tree what we have discussed. So, the derivation will be of this form this s produces u A z etcetera and now the derivation connecting to this if you place it here you can see that all the strings u v power i, x y power i, z for all i will be in the language L.

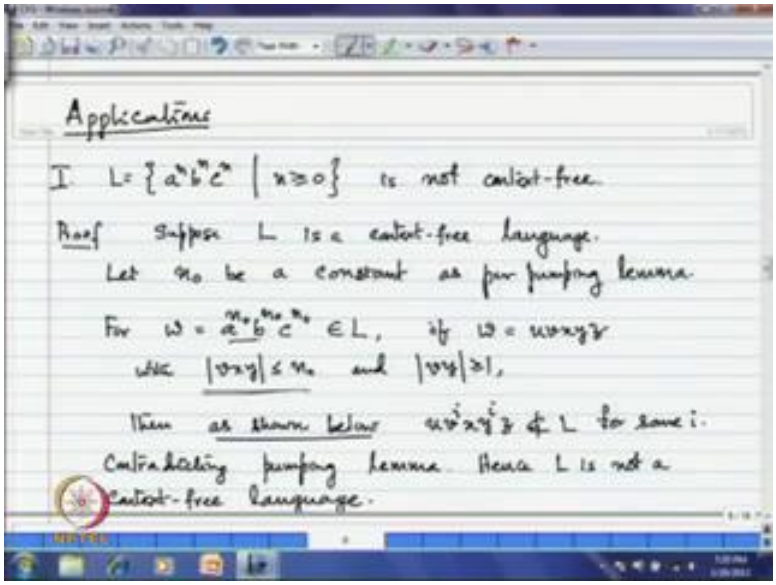
Thus, you see every string w of length greater than or equal to n, what are the constant that we have set here, the constant is in this particular context I have set n equal to 2 power k, because we have considered Chomsky normal form and you can observe this all the strings u v power i x y power i z will be in the language L. So, what is pumping lemma for context free languages and you can compare with pumping lemma for regular languages and understand this better.

Because, the philosophy is similar the way that we are working there we have work with the state, if the proof is through you know finite automata here, we are working through the non terminal symbols here. But, what is the philosophy once again, if you look at you know I have to produce the derivation of this form as I explained here.

Once i have produced derivation like this, I can always produce the strings u v power i, x y power i, z in L for all i, so for that purpose we have consider you know the tree in which we have consider a longest path and such a way that you know the non terminal

symbol is repeated. So, the constant we have to set accordingly, so we had set accordingly and we have produced this result.

(Refer Slide Time: 39:58)



Now, let me talk about applications of this pumping lemma, because you see in case of regular languages the pumping lemma you have used to observe certain languages are not regular. Similarly, here in case of context free languages also use this pumping lemma to observe certain languages are not context free. Let me just give this example, if you consider the language L a power n, b power n, c power n, n greater than or equal to 0, this is not a context free language.

How do I observe this suppose L is a context free language, now as per this pumping lemma you should have a constant let me say n naught be a constant as per pumping lemma for CFL, now as we have worked for regular languages here also same debate. If you choose any string w, whose length is greater than or equal to n naught, what pumping lemma says, if it is a context free language.

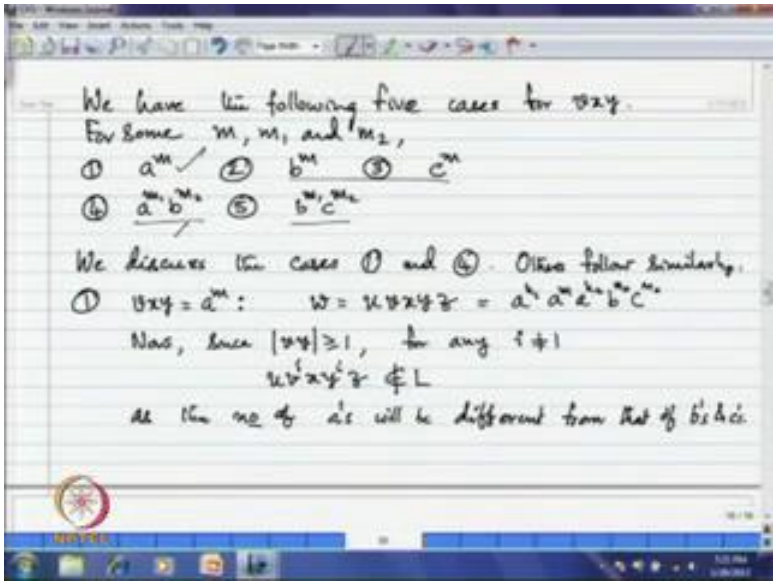
We should be able to split it in to 5 parts in which you know at least one of them is non empty, that v or y such that we should pump string, simultaneously for all the powers the resultant strings should be in the language. Now, I will give you string which face that particular condition and so that we say that our assumption is wrong. So, when I am assuming it is a context free language I get a constant n naught as per the pumping lemma.

Now, let me smartly choose the string say because, here the strings are of the form a power n, b power n, c power n, I am particularly choosing you know this a power n naught, b power naught, c power naught. You see is the length of the string is 3 m naught and therefore, this is greater than or equal to a length n naught, number n naught, length of this string. So, if it is split it in to the any strings of a form u v x y z into 5 portions.

If, this w is splitted like this satisfy the conditions that v x y length is less than or equal to n naught and at least v or y is non empty, once I have take this restriction v x y length is less than or equal to n naught. You see how this portion v x y looks like how does it look this v x y, since it is of length less than or equal to n naught, it will be within a's or within b's or within c's or you know you can have some a's and some b's.

Of course, you cannot have a's, b's, c's because the length is length maximum n naught, so this can have if it is in the border you can have some a's and some b's or you can have some b's and c's, but you cannot have all a's, b'c, c's together.

(Refer Slide Time: 42:52)



Now, you see what we will do as in case of a regular languages I have chosen this string now as shown below this u v power i, x y power i, z is not in L for some i, for some i will produce that which contradicts pumping lemma. So, that the language is context free language that is what is will be the conclusion, if I show that for some i u v power i, x y power i, z is not in L.

So, we have the following 5 cases as I just mentioned for v x y for some m 1 and m 2 at either it can be of the form a power m, this v x y or it is of the form b power m or it is of the form c power m or if it is common to the portion. This a's and b's highly have a power m 1, b power m 2 form or it is of the form, if it is common to b's and c's b power m 1 c power m 2.

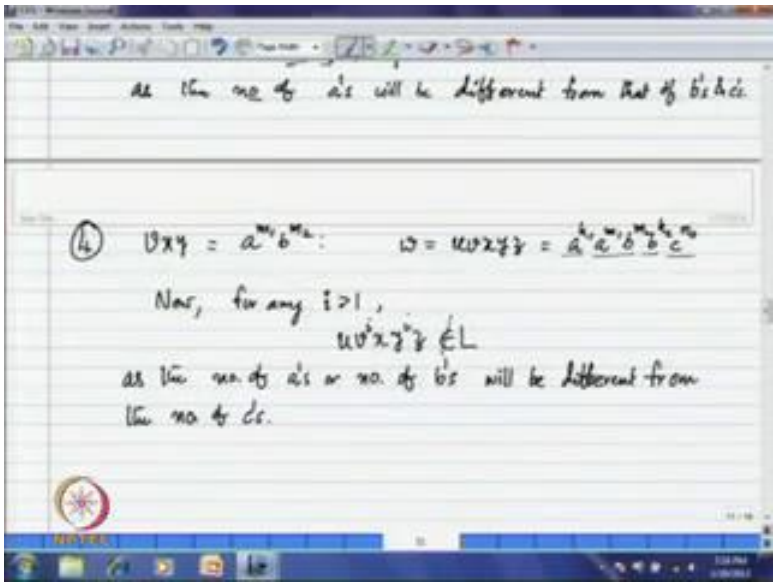
If I argue for a's, then it will be similar for two these cases if I argue for one of these the other thing will be similar, so we discuss the cases 1 and 4, other cases will follow similar manner. Let me consider, the case v x y is of the form a power m and w that is u v x y z, it is of the form a power k 1, because when a power mm when I am saying this side and that side you can have some a's.

Say this k 1 and k 2 you know greater than or equal to 0, because if a power m is equal to a power m naught, then you can you cannot have any number of a's other side, but anyway this is the genetic format, when v x y is of the form a power m. So, this u v x y z if it is in this form since this B or y is non empty that means, within this a's you know at least 1 a.

If, you consider when I am pumping these v's and y's my water is non empty, what will happen the number of a's will increase, but the portions in this jet you know you have b's and c's which remains same. Because i am just pumping the volts v and y the number of a's here for example, here within this v y if 1 a, if it is their some where it has to because the length of v y is greater than or equal to 1.

If i keep pumping the number of v's will increase number and therefore, number of v's will be different from the from that of b's and c's and hence what will happen you know the resultant string will not be in L.

(Refer Slide Time: 45:42)

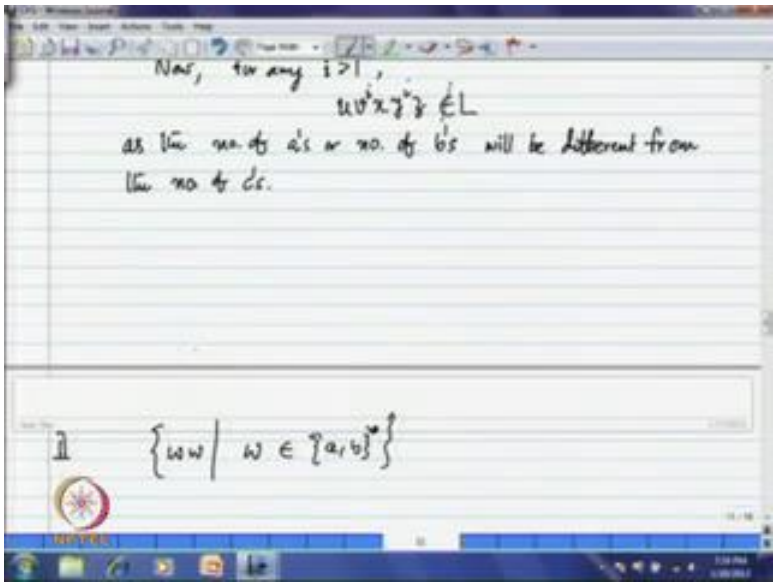


Similarly, if c x y is of the form a power m 1 and b power m 2, if it is like this you know I have some a's here a power m 1, b power m 2 and the rest of the b's, I may call it has b power k 2, c power m naught, iit is of the form..Now, since again v and y's, when I am pumping I do not know v may have some y's, y may have some b's might be pumping them or you know v and y's may have only S.

In which case, when I am pumping them, if it is i greater than 1, if I rise the powers what will happen the number of a's or b's will increase, whereas the c's which are in the portion z, you know that will not increase. And therefore, as a number of a's or b's will be different from number of c's in the string, this cannot be in L.

So, I can produce string for certain powers which are not in L but pumping lemma says that for all I it has to be so since I am producing the strings like this as observed. Now, you have some i such that u v power i, x y power i, z is not in L which is contradict in pumping lemma, thus you can say that this language is not a context free language.

(Refer Slide Time: 47:02)



Similarly, people you know several other languages say for example, w w, w belongs to a b star, we can observe that this is not context free using pumping lemma, only thing is if you say this is context free language, what we have to do you have to there will be a constant you have to now choose some string. And observe that and you divide in to any of the form u v x y z then by pumping v or y, v and y simultaneously you will have certain strings which are going behind the language. So, that is how we have to do now, you can take this as an exercise and ascertain that this language is not context free some more languages I will display now, that they are not context free you can observe them.