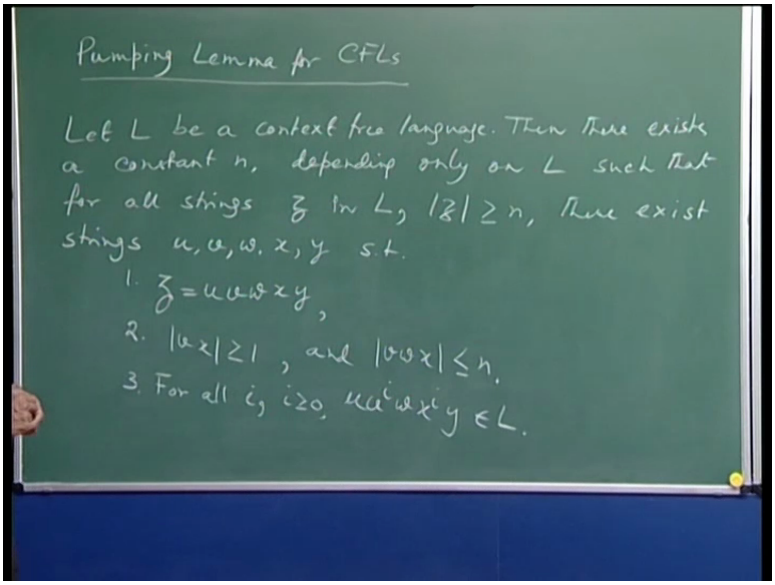
Theory of Computation
Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Lecture 27
Completion of Pumping Lemma Proof
Examples of use of Pumping Lemma
Converse of Lemma does not hold
Closure Properties of CFLS

We will complete the proof of the pumping lemma for context free languages. Recall the statement of the lemma. Basically it says that suppose L is a context free language then there is a constant n which depends only on the language L such that for all strings z in L such thatthe lengths of these strings are greater than equal to n. You should be able to break up this z in five parts u v w x y such that both v and x cannot be empty. In other words length of the string v x that is greater than equal to 1.

Length of v w x all together is less than equal to n where this n is the pumping lemma constant for the language L and then the pumping condition that for all i, i greater than equal to 0. That u v to the power i that means i copies of v followed by w followed by i copies of x followed by y. This string is also in the same language L.

(Refer Slide Time: 01:44)



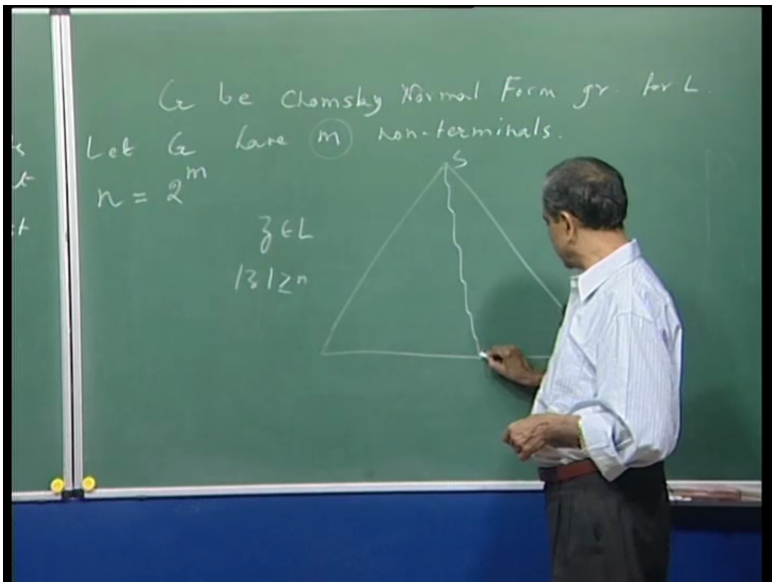
Last time we really completed the main crux of the proof which is showing how this condition 3 we managed to obtain for a certain breakup u v w x y. What was left was to show that this condition2 that is the length of v and x is greater than equal to 1 and v w x together

their lengths,the concatenation of v w x, the length of that string is less than equal to n. This particular part was left.

Otherwise we had completed the proof. Let us complete the proof by showing that the proof that we gave, it is such thatthis condition 2 will also hold. Now very briefly what we did was first of all we started with the assumption of course that is given that L is a context free language and therefore we said that let G be a Chomsky normal form grammar for L. Then we said that let G has orthis grammar G let it have m nonterminals.

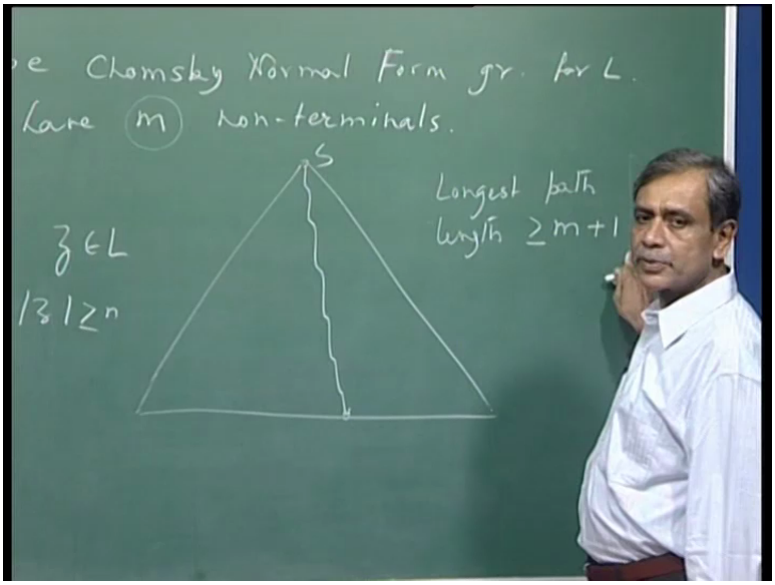
Next what we did was using this value of m which is the number of nonterminalsof the Chomsky normal form grammar G of the language L we defined for us the pumping lemma constant. We said let n be 2 to the power m, okay. Then we said consider any stringin the language z whose length is greater than this number. Soand then we considered the derivation tree for such a string z, right? And z is greater than equal to n. Remember n is 2 power m. Then we said look at the longest path in this derivation tree, okay.

(Refer Slide Time: 04:49)



Now we proved a small fact which showed that thislongest path length is at least m plus 1, right? So once I have the longest path in the tree starting from the root node which is the start symbol S and of course going to a leaf node which is labelled with a terminal, we know that this path has length at least m plus 1. We proved this fact.

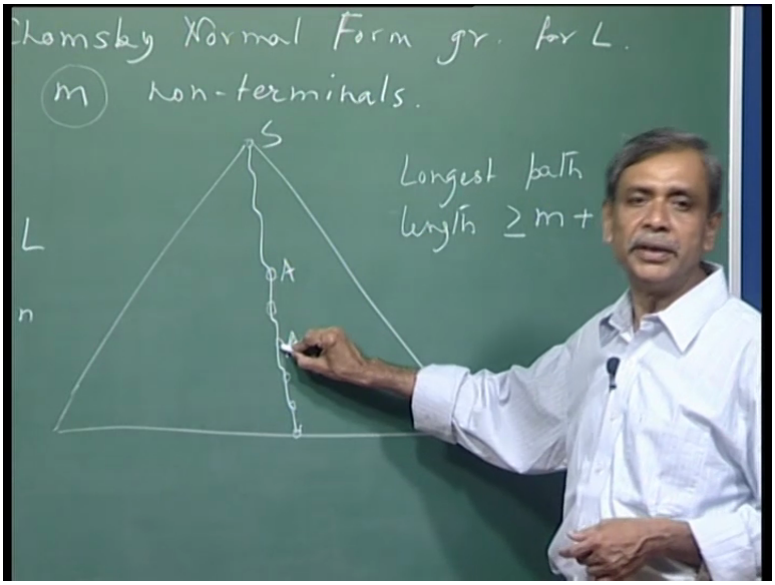
(Refer Slide Time: 05:39)



Then what we did wasin this you know we started from here, the end of this path and going up till we found a nonterminal which got repeated. And again we argued that this case must happen that is a nonterminal must repeat. Furthermore the argument that we gave also clearly showed that by the time you move up m plus 1 edges, one nonterminal must repeat. Solength of this path it cannot be any more than m plus 1, okay. So is this clear?

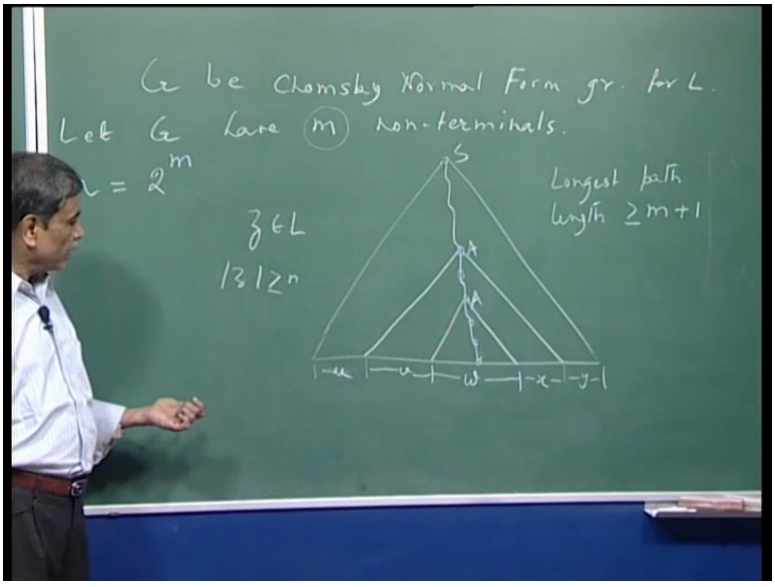
What I am saying is that we started from this point, first of all we said look at the longest path. This path of course has to be of length itself greater than equal to m plus 1. But it could be in general much larger than m plus 1 because you know z can be very large. And then you are considering the longest path. But now you are going up from the bottom of this path till you find the first nonterminal which is occurred earlier in the path as you go from below.

(Refer Slide Time: 07:18)



And the reason that we gave some kind of pigeonhole principal argument that this repeat must occur by the time this path is of length m plus 1 or less, right? And then we said that look remember we said that suppose this particular A derives this part of the string and this second A derives this part of the string. And this part we identified as w. This part we said is v. This part is u. This part is x and this part was y. So this is how we would break up the string.

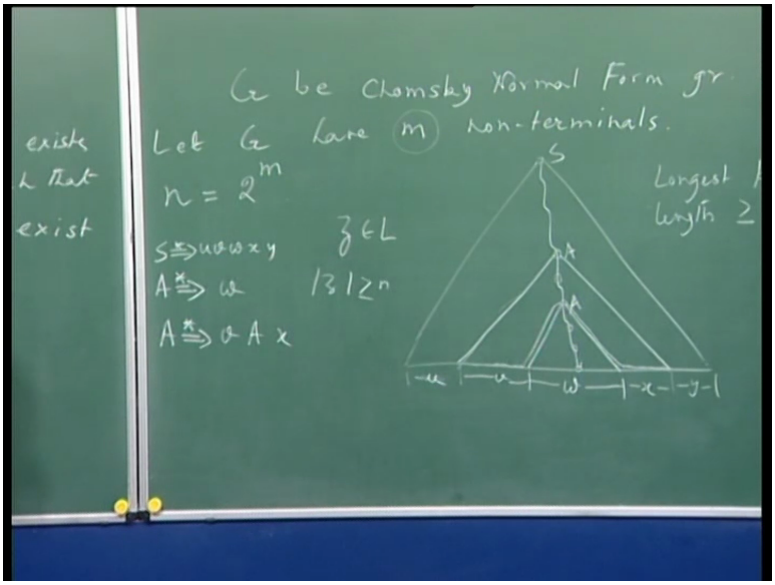
(Refer Slide Time: 08:32)



It is clear, right, that S ofcourse derives the whole string z which is u v w x y. Further A derives this string v w x. First of all I can see that A derives this part. A derives this wpart, so

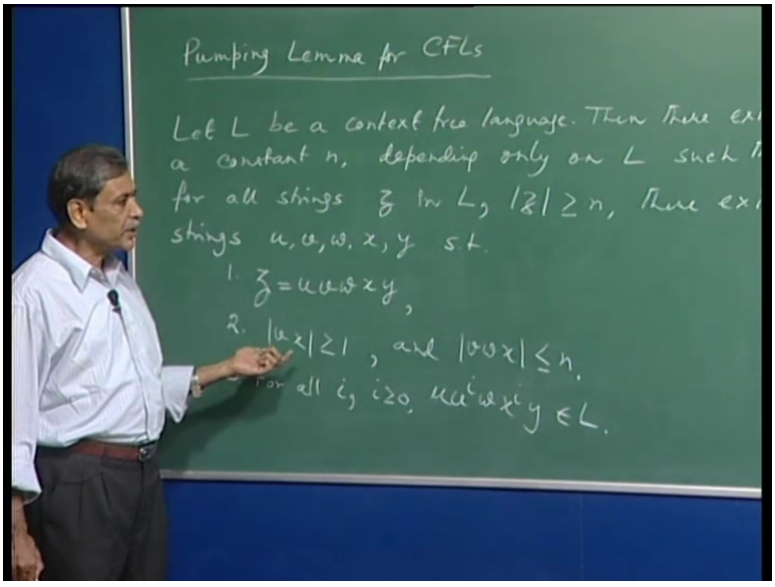
A derives w. And if youthink in this manner that derivation tree of A is you think of this, so it is generating you know we argued all this last time that A I can see to be deriving also v A x.

(Refer Slide Time: 09:36)



And now it is easy to see. And then see thisfact is very clear from the diagram itself and now it is very easy to see suppose this condition is not true. That means length of v x is 0. You know it has to be something. Length cannot be(any) anything negative. So if it is not greater than equal to 1 then this v x length is 0 which means both the strings v and x is empty, right?

(Refer Slide Time: 10:06)

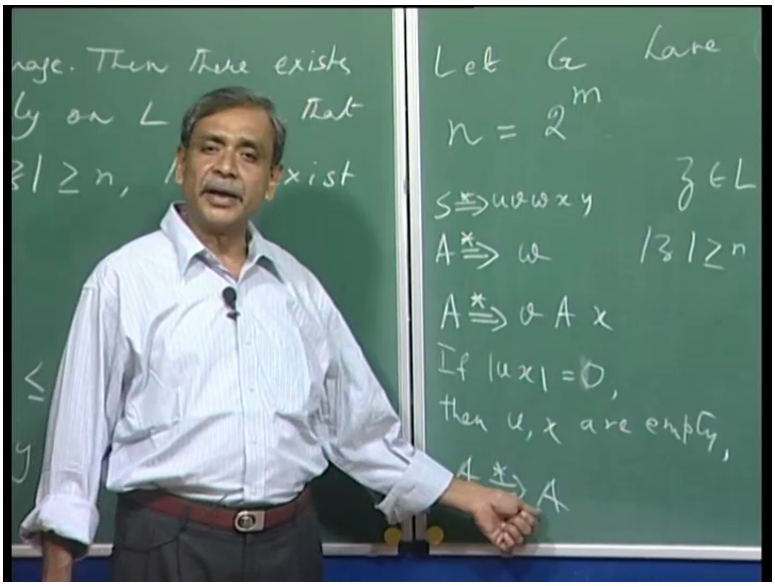


So what it means is if v x length is equal to 0 then this of course implies both v and x are empty and in that case this particular derivation would mean that A derives A, right? But that

is not possible because the grammar is a Chomsky normal form grammar. Remember for this to happen weneed to have either epsilon productions or unit productions or both, right?

I mean either some unit productions such that A derives A or you know some at epsilon productions such that somenonterminals get rewritten as epsilon and starting from A we get only A back.

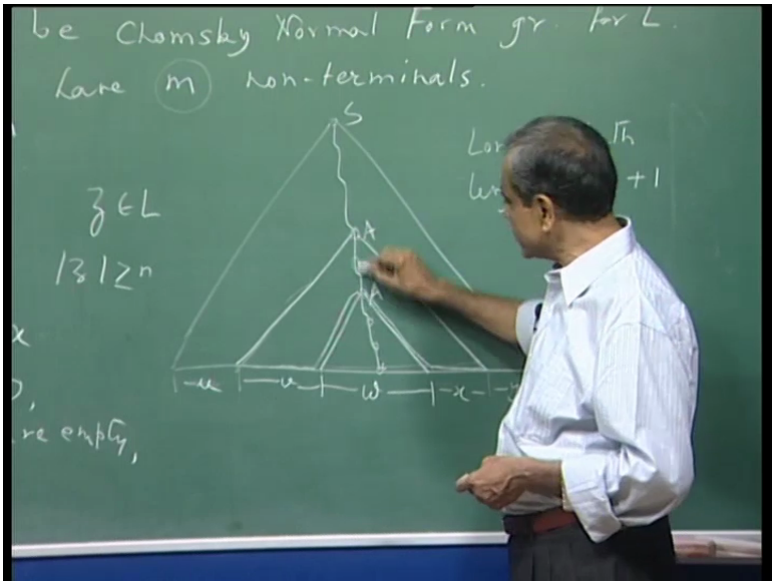
(Refer Slide Time: 11:21)



But these cases are not there because our grammar is Chomsky normal form grammar. So if we assume that both v and x they are empty then we get a contradiction because that cannot happen as the grammar is Chomsky normal form grammar. So therefore we say that we can see that this condition therefore must hold. And the other condition is what? We said that length of v w x is less than equal to n. But what is v w x? Nowwhat is v w x is derived by A, right? And this is the derivation tree. This part is the derivation tree for v w x.

In that the longest path is the path from here to this. See that is clear is not it because here there cannot beany part which is longer than this because then the original path which wesaid is this is the longest path that could not have been the longest path in the big tree. So this is the as larger path as possible in this derivation tree.

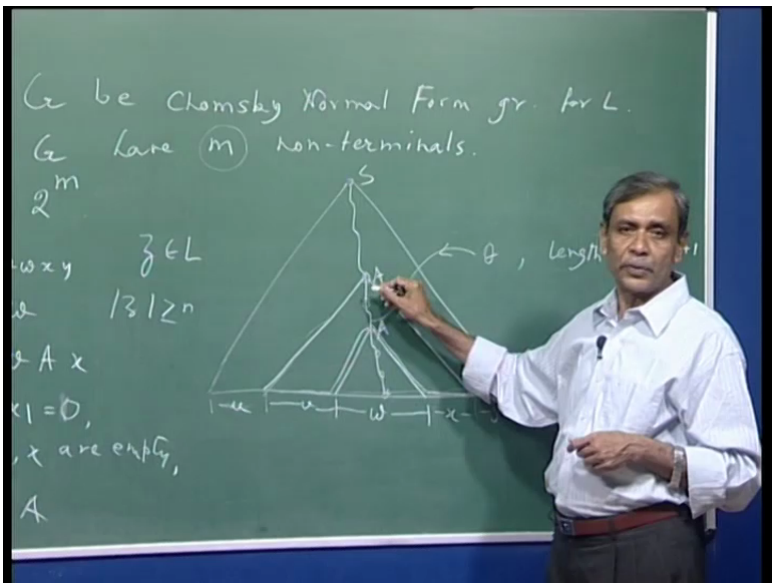
(Refer Slide Time: 12:47)



And we know that is how we are good that this path length isno more than m plus 1. See let us not get confused by this statement. This was about the overall tree and thenwhat we said was that we are going down from here up and by the time we go upby the time we get no more than m plus 1 edges, onenonterminal must repeat. So we were waiting till the first nonterminal.

So this path length. So how shall I write it? So let us just say this path is theta. So length of theta is less than equal to m plus 1. And theta is the longest path in this derivation tree starting from A, right?

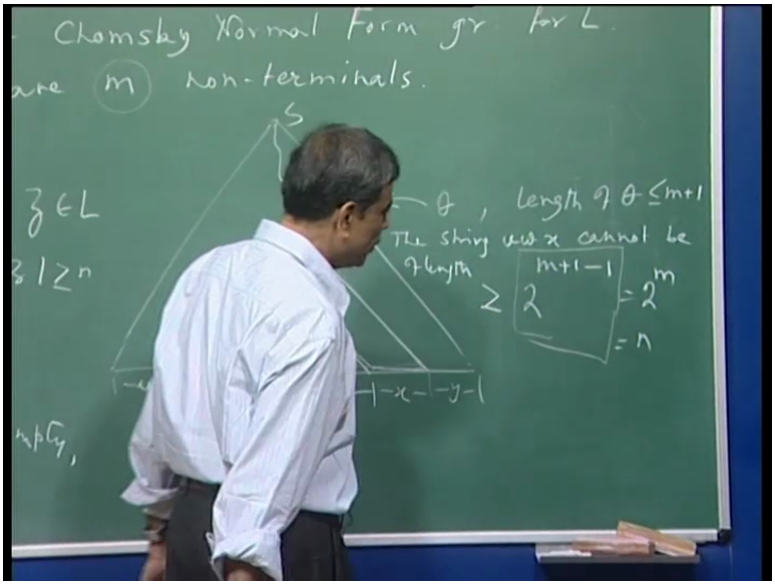
(Refer Slide Time: 13:59)



That means what? We go back to old fact that suppose I have a derivation tree where no path is greater than m plus 1 then the string that is generated cannot be of length greater than 2 to the power m plus 1 minus 1 which is equal to 2 to the power m which is of course equal to the pumping lemma constant, right?

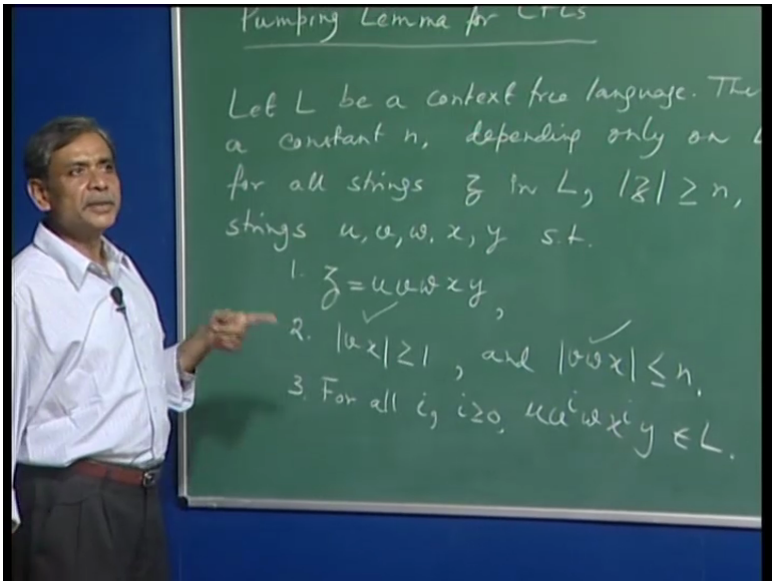
So let me write it so that this is clear. Length of theta is m plus 1 so the string v w xcannot be of length greater than this, which is equal to 2 to the power m which is equal to n where n is the pumping lemma constant, right?

(Refer Slide Time: 15:03)



So we have said the length of the string v w xcannot be of lengthstrictly greater than this. Therefore length is less than equal to n. And that is the other condition. Sowe have proved this also and this and therefore the proof of the lemma is complete.

(Refer Slide Time: 15:41)

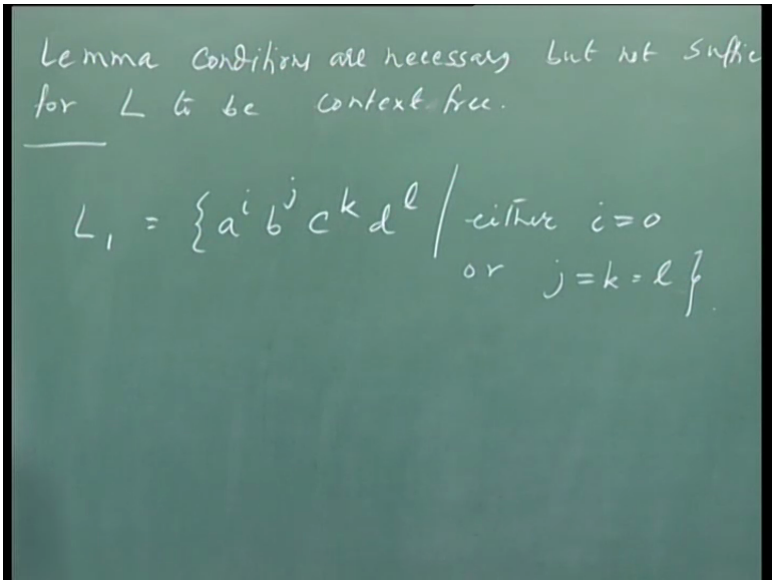


See this is something we must appreciate andthis kind of thing was there for our pumping lemma for regular languages as well. What we are saying is that this lemma conditions they are necessary butnot sufficient to prove a language to be context free. In other words that suppose you find a language L which satisfies all these, from there you cannot conclude that the language is context free. In other wordswhat does it mean?

That it is possible that some non-context free languages also to satisfy all these conditions. That is what we mean when we say that these conditions are necessary. In other words of course as the lemma states, every context free language L they will satisfy these pumping lemma conditions but the other wayis not true. Just because for a language all these conditions hold you cannot includethat language L to be context free.

In factlet me make this point once more that what it means is that it is not always possible to make use of this lemma to prove a non-context free language not to be context free. So such an example let me give right away. It is a well-known example. L 1 is a i, b j, c k, d L such that either i equal to 0 or j is equal to k is equal to L.

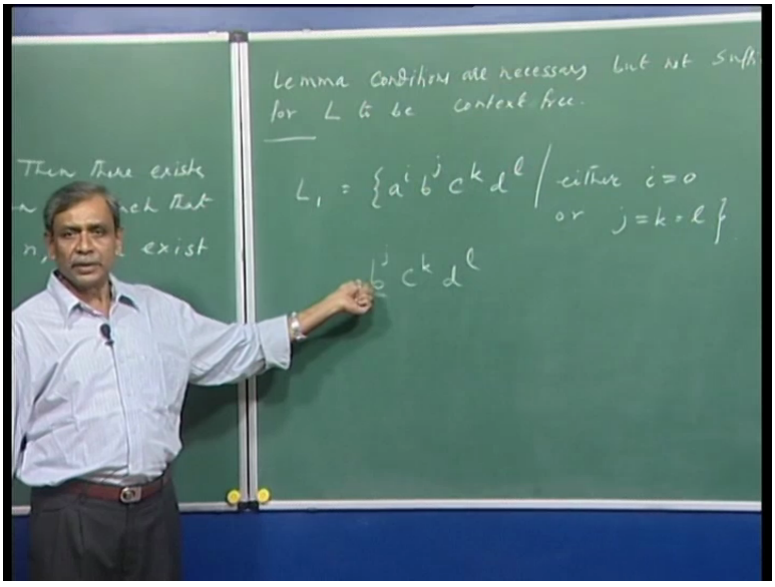
(Refer Slide Time: 17:59)



So what it means is that when you know firstly this language L 1 as a string of a's followed by b's followed by c's followed by d's where either there are no a's or if you have a number of a's then it must be the case that the number of b's is equal to the number of c's is equal to the number of d's. Now this language is not a context free language but you see that it will satisfy all these conditions. Why? You seethere are two cases, either i equal to 0 then there is no really restriction on j k L, right?

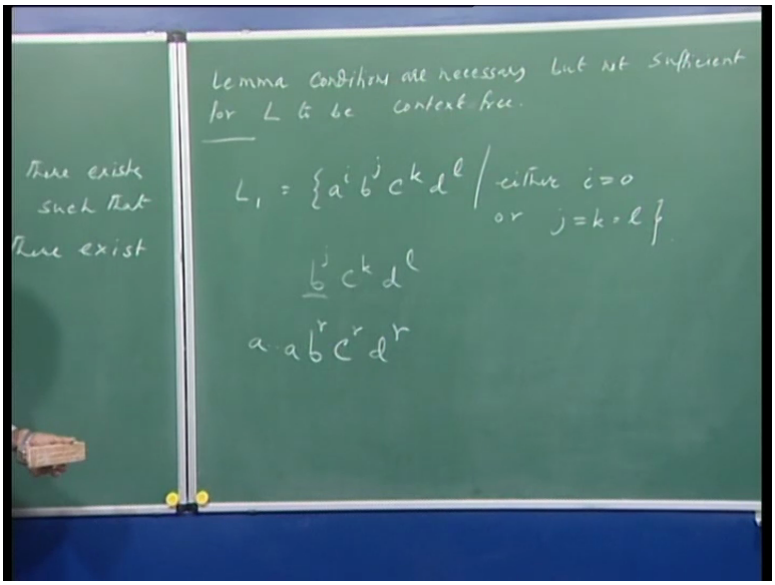
So basically we have consider a string like b j, c k, d L and let us say the entire v w x part occurs withinand then you pump so the number of b's go up or one b goes down. So that is okay, right? It is still in that form becausewe cannot add any a's in front by pumping up or down, right?

(Refer Slide Time: 19:30)



So we will remain within the language L. On the other hand suppose I have some a's followed by b's. So let us say this is you know r, then it must be the case if since I have the number of a's, c r and d r. These number of occurrences of b'smust be equal to number of occurrences of c's and as well as for d's.

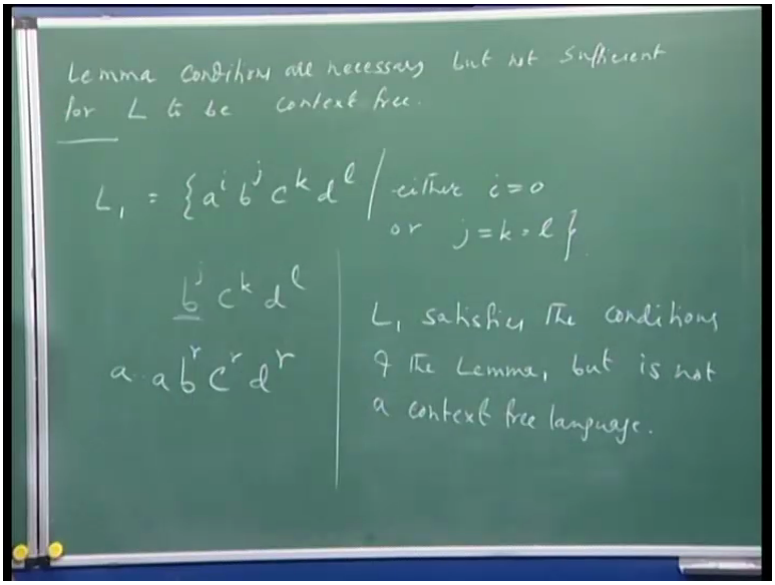
(Refer Slide Time: 20:08)



Now again it is possible if you take such a string from the language then your v w x can be very well be within this, right? You can choose your u v w x y such thatv w x part entirely is within a's, so when you pump up or down all that happens is that you change the number of a's, the number of b's, c's and d's do not change and therefore on pumping the string that you would get will be also in the same language L 1.

So the point I am making is that this is a language L 1 which satisfies the conditions of the pumping lemma and yet L 1 is not a context free language. So let me write it down. L 1 satisfies the conditions of the lemma but is not the context free.

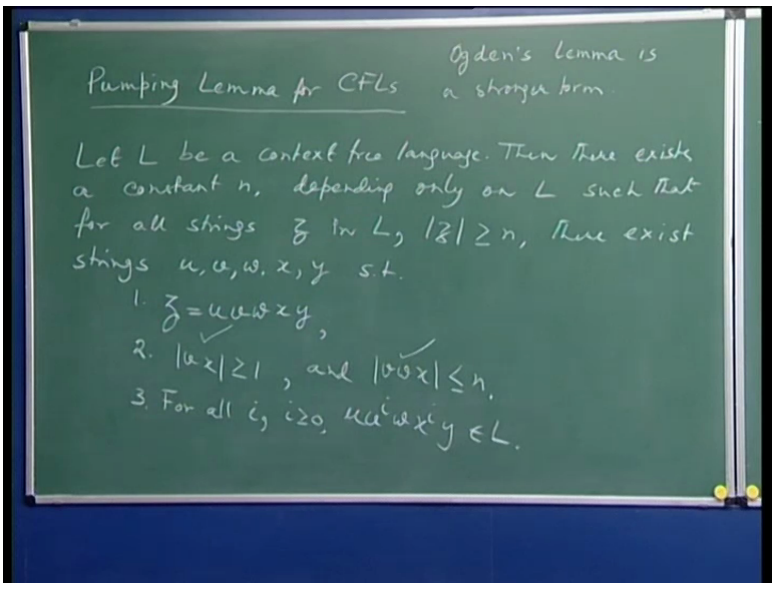
(Refer Slide Time: 21:43)



And we will be able to prove that this language L 1 is not context free language by some other means if not by using this form of pumping lemma. There are stronger forms of pumping lemma. In particular one stronger form which you can use which is fairly well known and used is called Ogden's lemma is a stronger form but we will not discuss Ogden's lemma here or there are other versions.

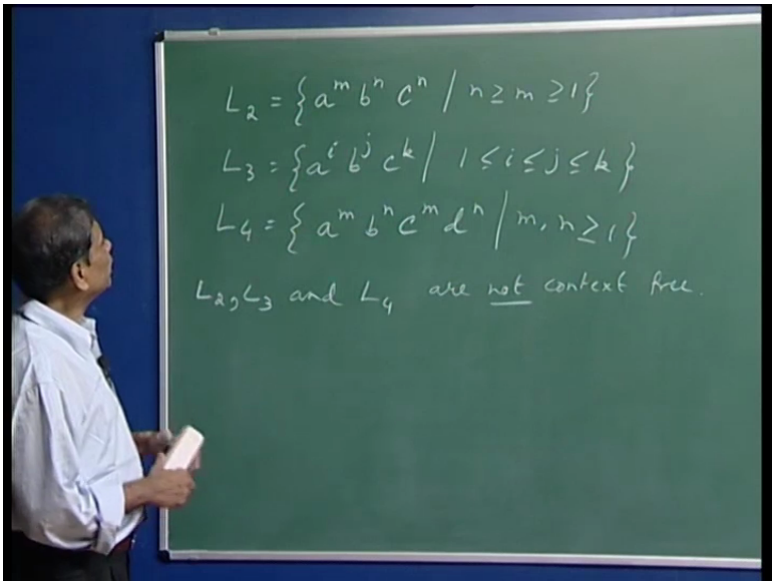
Even Ogden's lemma will not be able to prove some languages to be context free. Then in fact we give one reference whilediscussing for regular languages andthat paper contains the most general form of pumping lemma for context free languages.

(Refer Slide Time: 22:58)



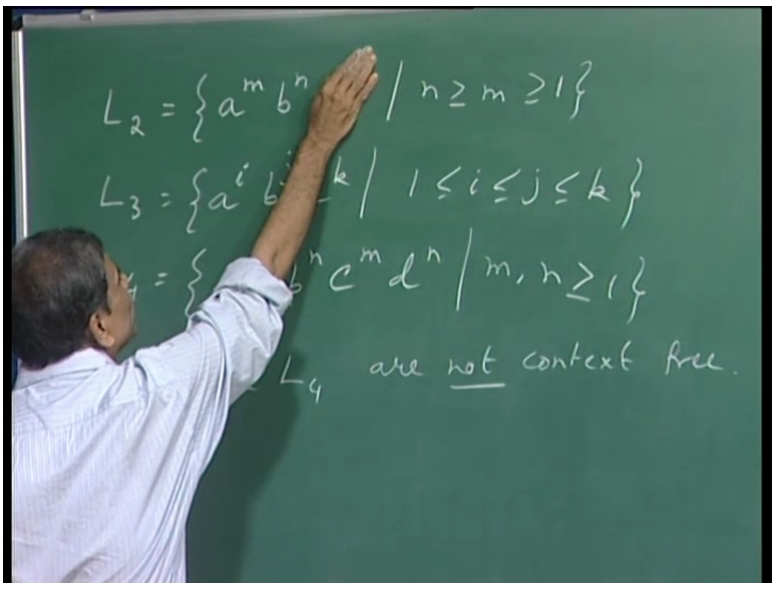
Here are some more examples of languages which are not context free and all these three languages you can prove that they are not context free but by using the pumping lemma that we have seen. Now just a little discussion on the form of these kinds of languages and at least intuitively you should be able to appreciate certain things or certain properties which context free language cannot ensure.

(Refer Slide Time: 23:37)



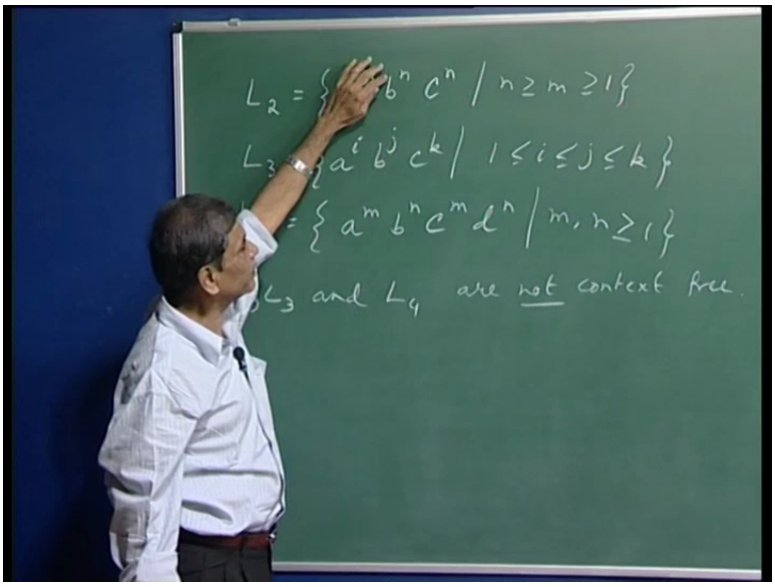
See here what is happening there are three strings a, b and c. What you want is first of all that b and c they must be equal in length and length of a is less than equal to length of either of these. Nowlet us say remove this, right?

(Refer Slide Time: 24:03)



Then it will be a context free language. But adding this is the problem. See what you will see later when we discuss the machine version that it is easy to ensure this or this butit will not be possible to ensure both these constraints at the same time. These are you know intuitively I am trying to explain to you. And similarly if this is not that this condition of course we can ensure, we can write a context free grammar which will generate this language b n, c n such that n is greater than equal to 1. There is no m.

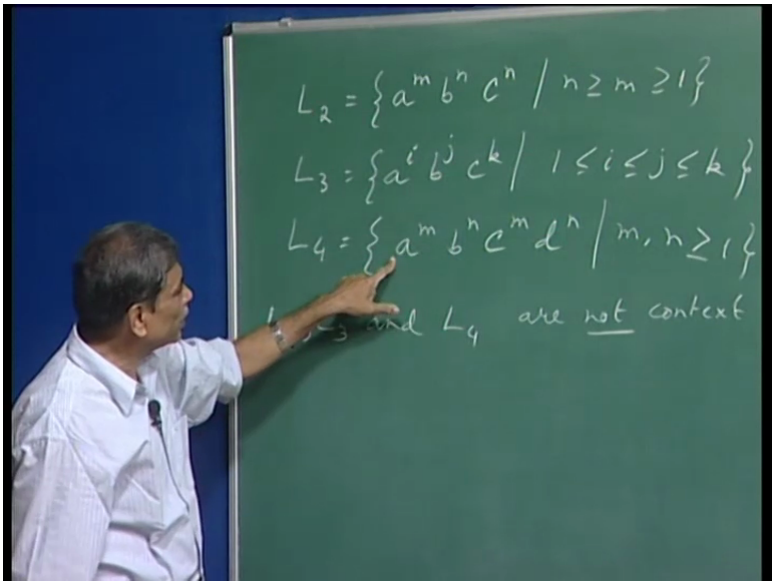
(Refer Slide Time: 24:48)



But together both these conditions we cannot handle using the context free nature of the grammar that we have. Take another kind ofsimilar things you see. Again there are these two condition business that i is less than equal to j and j is less than equal to k. See the problem is that once you ensure this again you seeone of these was not there, either a or you know this oreither of these letters. Then again we can ensure that by means of a context free language.

In other words supposing a was not there. B j, c k such that j is less than equal to k. This is something we can handle. We canwrite a CFG context free grammar to generate such strings. But these two conditions together we cannot handle. Now another version of the same two condition business is now Ishould say that you should also realize that, in fact let mesay what I want to say after I discuss this. What is this?

(Refer Slide Time: 26:21)

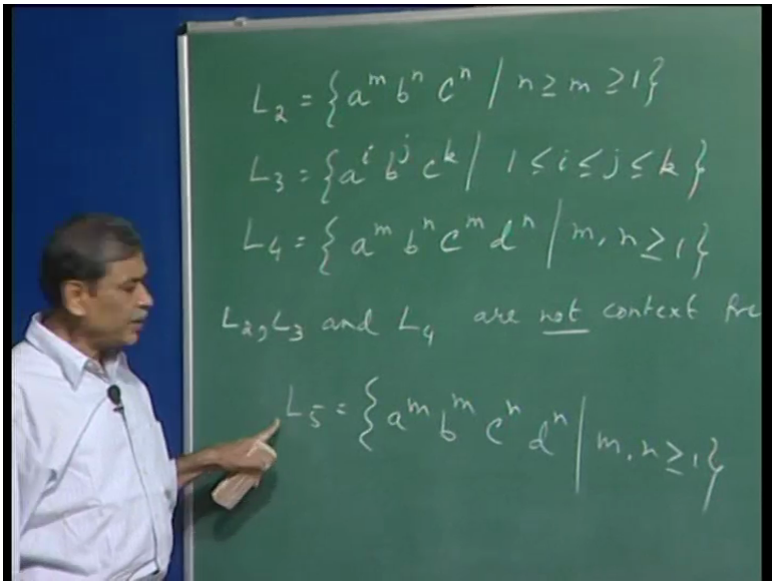


Again I have two conditions, right? That number of a's must be equal to number of c's and number of b's must be equal to number of d's, two conditions. Againthis is something we cannot take care. And if you look at pumping lemma why that is the case? Whereas you know we can say pumping lemma constant is let us say k. Then you take a string a k, b k, c k, d k, right? And then what happens is that the window v w x will be either entirely within of one of these you know letters, strings of one letter.

Like entirely within a's or b's or c's or d's or can straddle two consecutive words. So what will happen is it will change the lengths of two consecutive ones at most. But you know here we need to make sure that the lengths of the first and the thirdthey match and that is where the problem will happen the same way that we have proved for you know that language a n, b n, c n not to be context free language. Same kind of proof will ideal go through here too.

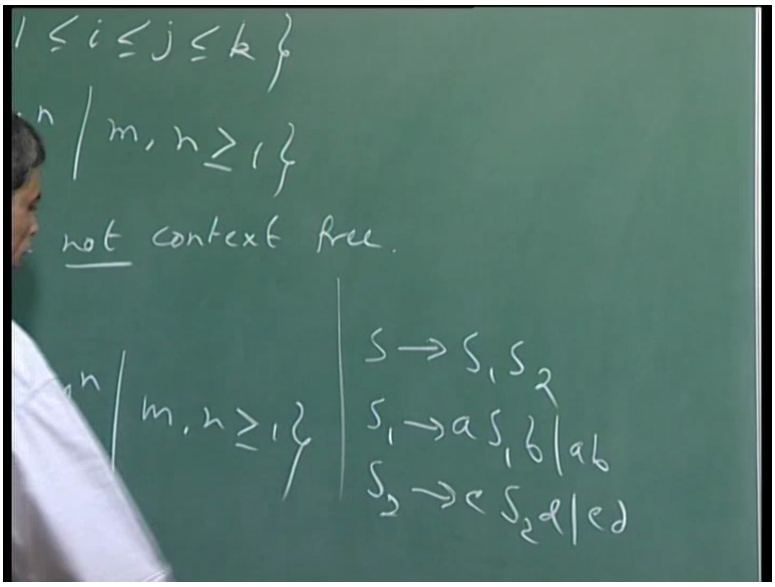
But you know I amsomewhat incorrect by saying that we cannot handle two conditions. We can handle two conditions so long they are not kind of inter mixed together. So for example if youtake another language L 5. Let us say I write it slightly differently, a m, b m, c n, d n. This is context free, right?

(Refer Slide Time: 28:13)



How will you do that? Remember I can separately generate this from some S 1, separately generate this and thisare just are two concatenations. In other words what will be that grammar? So we can start with S, S goes to S 1 S 2, right? And then S 1 is a S 1 b or a b and S 2 is c S 2 d or c d. And you can see that this grammar is going to generate this language, right?

(Refer Slide Time: 28:50)

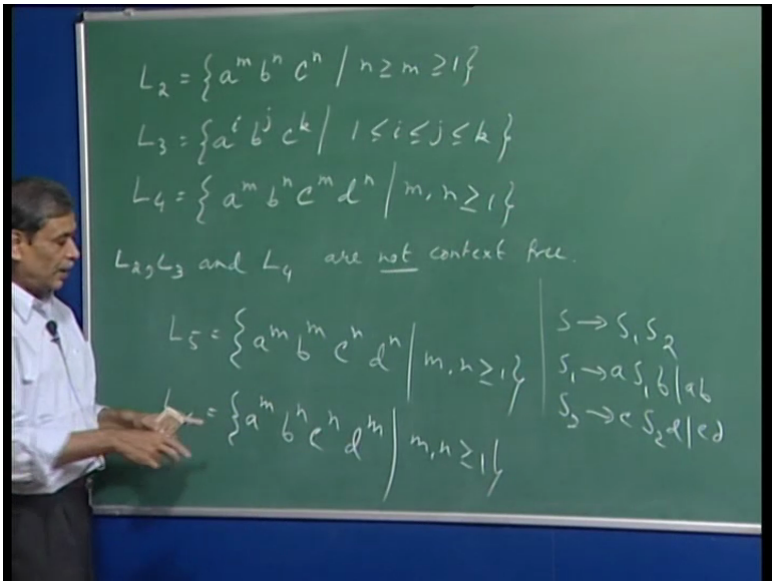


So we can handle two conditions but not together, is that clear? Or you know this is really intuitive way of what I am explaining and whenever you see that the conditions have to be kind of they aremixing up orboth the conditions you need to ensure together, that is where the

problem arises, right? So youshould be able to at least intuitively see that the situation with L 2, L 3 and L 4 is different from L 5 or for that matter L 6 which is a m, b n, c n, d m, right?

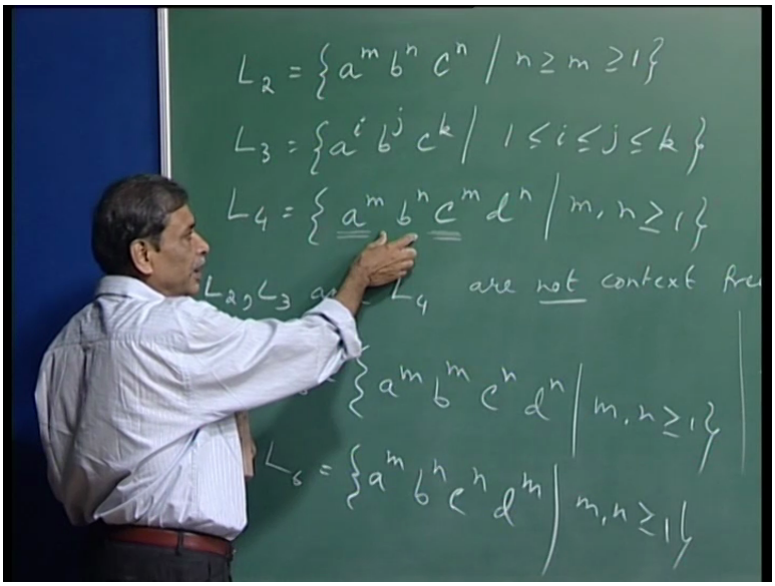
Here I need to keep track of correspondence between a's and d's separately from the correspondence between b's and c's. This is something we can handle.

(Refer Slide Time: 30:00)



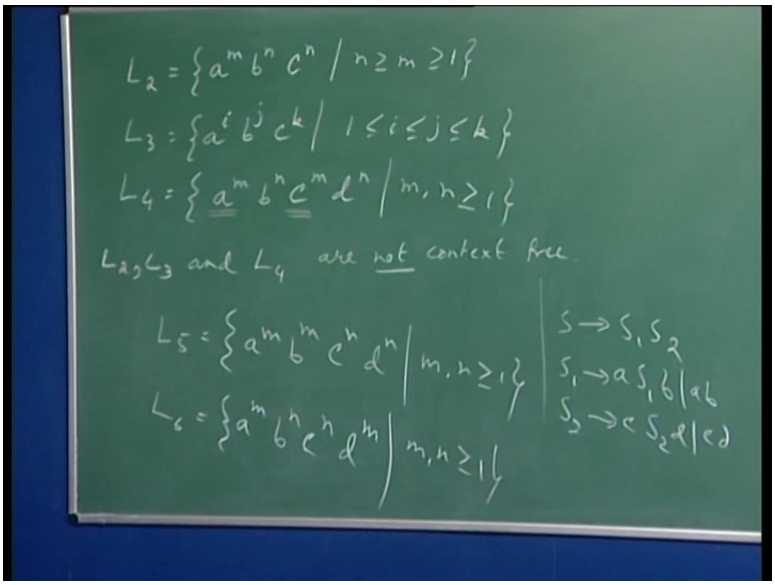
Sowhat is the difference between this and this? As you handle this m is this is m you know (num) number of a's here is equal to number of c's. There I am getting some b's inside. And you see that is where the problem will happen.

(Refer Slide Time: 30:18)



We willkind of be able to appreciate this a little more whereas here you know this I can handleyou know I had generated some a's and this is taken care of(sepa) I mean best is you should yourself write the grammar for L 6. This is a context free language so you will be able to again very simply write a context free grammar which will generate this language.

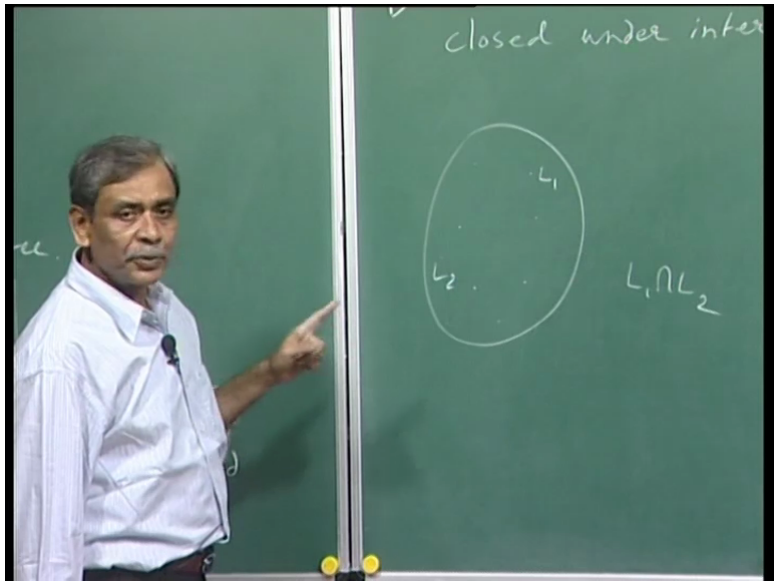
(Refer Slide Time: 31:00)



Well now that we have the pumping lemma andnow we know in particular one language you know several languages which are not context free. Now usingthat fact I can come to a very strong conclusion, right? So let me show what thatconclusion is that the class of context free languages, this class, is not closed under intersection. We willlittle later we will talk about the other closure properties but you know thiskind ofalmost immediately follows from whatever we have been discussing.

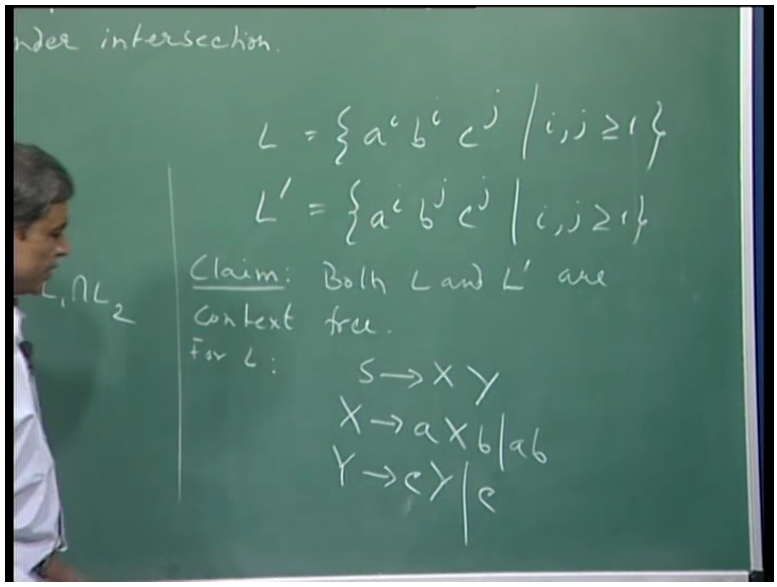
Sorecall what does it meanfor a class to be closed under intersection? We are talking about classes of languages so each language is a set. You know I take two elements, so one language and another language L 2 and I create the intersection of these two. We say the class is closed under intersection if whenever you pick up two languagesfrom within the class, take their intersection, the result is also within that class.

(Refer Slide Time: 32:50)



But we will be able to show very easily that this is not the case with context free languages. So let us take two languages L a i, b i, c j and L dash a i, b j, c j. So claim which we can easily prove that both L and L dash are context free. Let mequickly give a grammar forL. You can this let me show you the productions. S goes to let us say X Y, okay. X goes to a X b a b and Y goes to c Y or c.

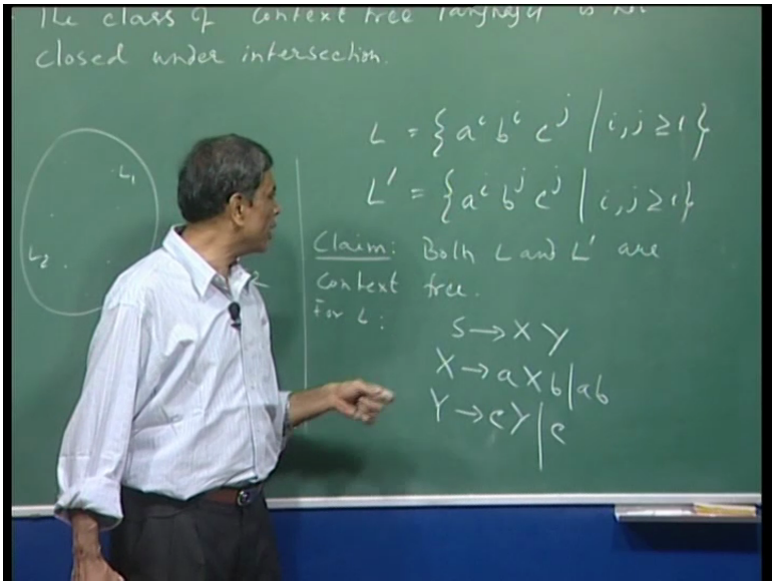
(Refer Slide Time: 34:48)



So you see basically Y can generate any number of c's and X will generate equal number of a's followed by b's, right? So X will generate a string of a'sand b's where number of a's will be equal to number of b's and Y will generate just any number of c's. And your string S isthe

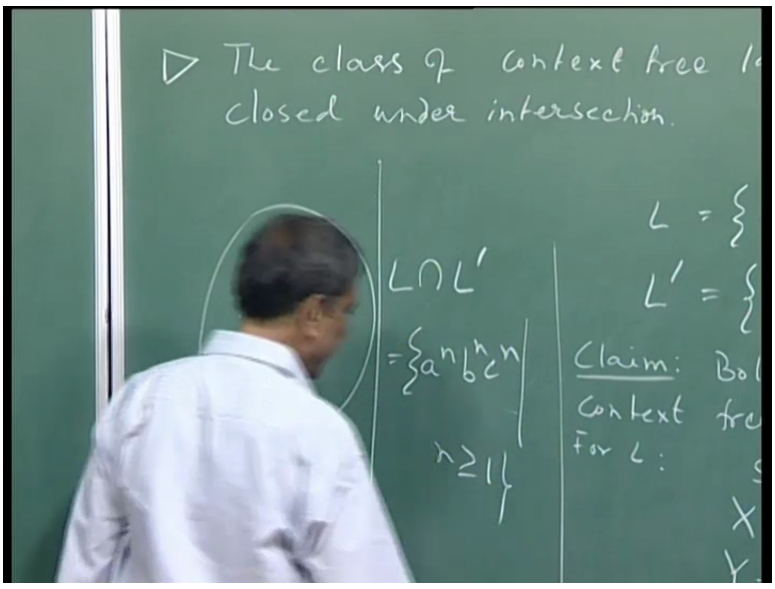
concatenation of these two. So therefore we will generate all strings of the kind which is mentioned here. And a verysimilar manner you can also give a grammar for L dash.

(Refer Slide Time: 35:24)



But what is the intersection of L and L dash? So obviously they have to be again of the form a's followed by b's followed by c's since there in L all the strings in the intersection strings are in L so thenumber of a's must be equal to number of b's. Since the same string is also in L dash that means number of b's also must be equal to number of c's. So therefore a's is equal to number of b's is equal to number of c's. So this is in fact this language a n, b n, c n, n is greater than equal to 1.

(Refer Slide Time: 36:17)

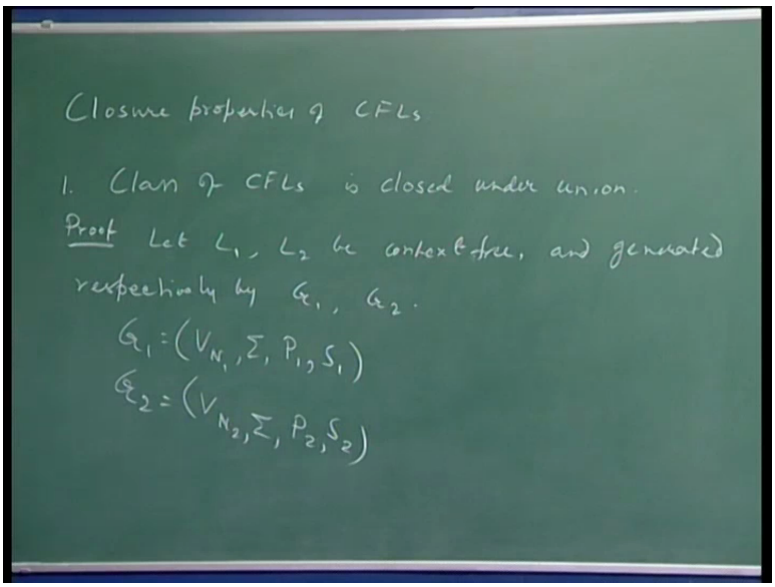


So as you see a very important result we proved quite easily from the fact that this language is not context free and this particular language is the intersection of two very simple context free languages. So the class of context free languages, this class is not closed under intersection because you know I could find intersection of two context free languages which is a n, b n, c n whichwe proved by pumping lemma which is this language is not context free.

This situation as you know is very different from the situation of a regular languages. Regular languages are closed under intersection, right? So since we are talking of closure you might as well see some closure properties ofcontext free languages. So first point which is again fairly simple to see that class of CFLS is closed under union. Why? You see whatwe are saying that you take two languages which are context free languages then take their union, L 1 union L 2 and we are claiming thatunion language iscontext free.

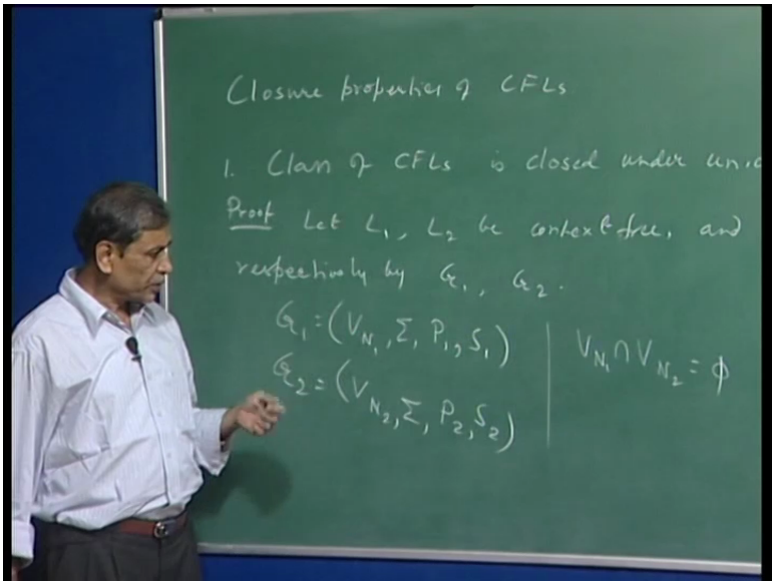
Sothis is very simple. Solet me give the proof that let L 1 L 2 be context free and generated respectively byG 1 and G 2. So in particular let us say G 1is V N 1, P 1, S 1 and G 2 is, correct?

(Refer Slide Time: 39:34)



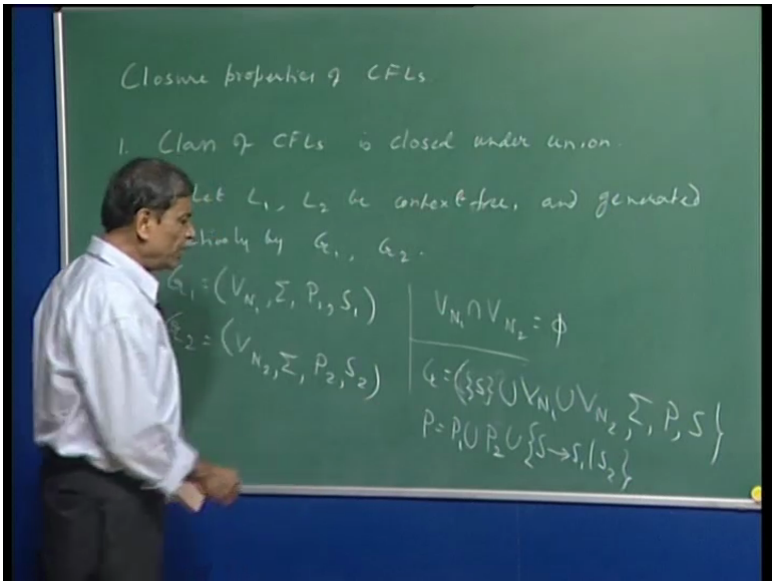
Now assume that you know we canby renaming some of the nonterminals assume that these two sets V N 1 intersection V N 2 is empty. That means wedo not use the same name for two nonterminals fromthese two grammar, right?

(Refer Slide Time: 40:03)



And now you see think of another grammar which is like this G which is a new start symbol S union, so I should put it in places. So basically just take the union of this V N 1 and V N 2 at S sothat make it your new set of nonterminals union V N 1 union V N 1, right? This isthe set of nonterminals, of course sigma,P and this new nonterminal is your start. And basically your P is P 1 union P 2 plus one more. And which is that? P is P 1 union P 2 union just one more production that is S goes to either S 1 or S 2.

(Refer Slide Time: 41:34)

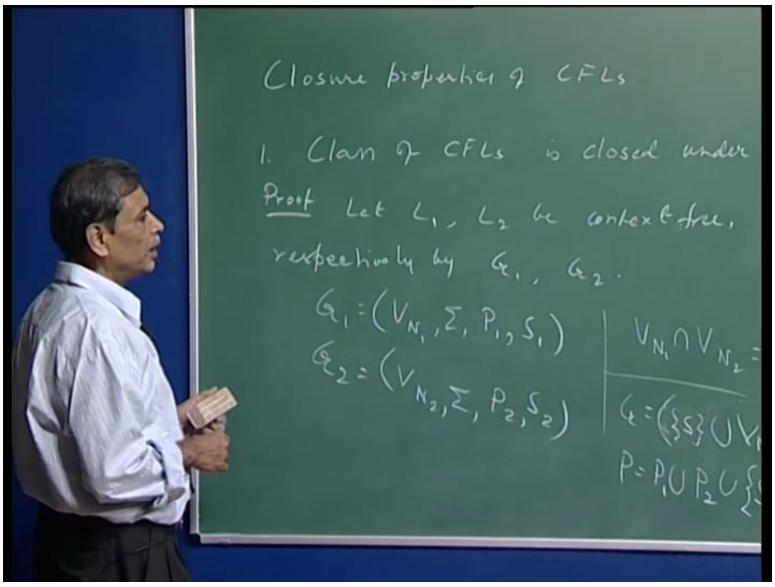


So what is happening you start this grammar G with the symbol S and this can be rewritten as either S 1 or S 2. If you rewrite as S 1 you willuse the grammar G 1 to generateany string of

L 1. Similarly you can generate if you first rewrite S by S 2 then you can generate any string of L 2.

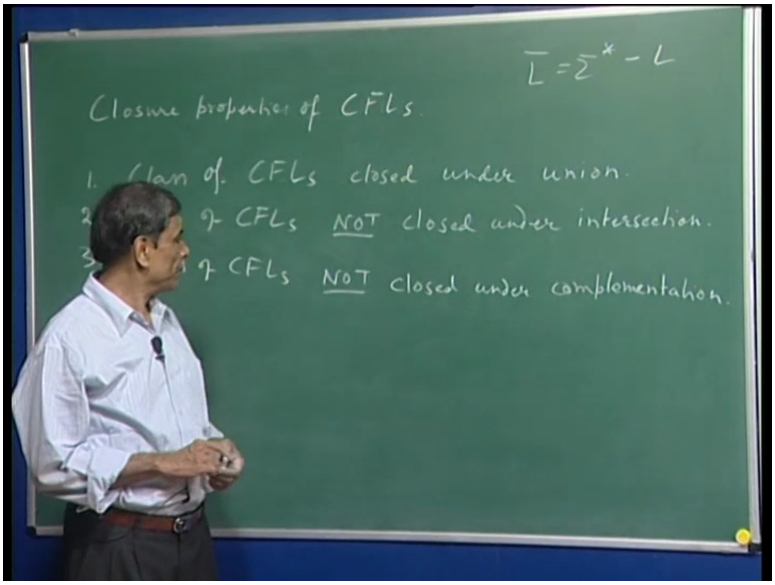
So therefore togetheryou know it means that from S you can generateall strings of language L 1 as well as all strings of language L 2 which is precisely what? The union language. So therefore the class of CFLS, this class is closed under union.

(Refer Slide Time: 42:28)



So from these two which we have just proved we can conclude something very important. Anotherclosure property that class of CFLS not closed under complementation. Remember complementation is that language L is there, you take the complement and the complement language is sigma star that is alphabet minus language L, right? The set of all strings over sigma which is not in the language L. That set is the complement of L.

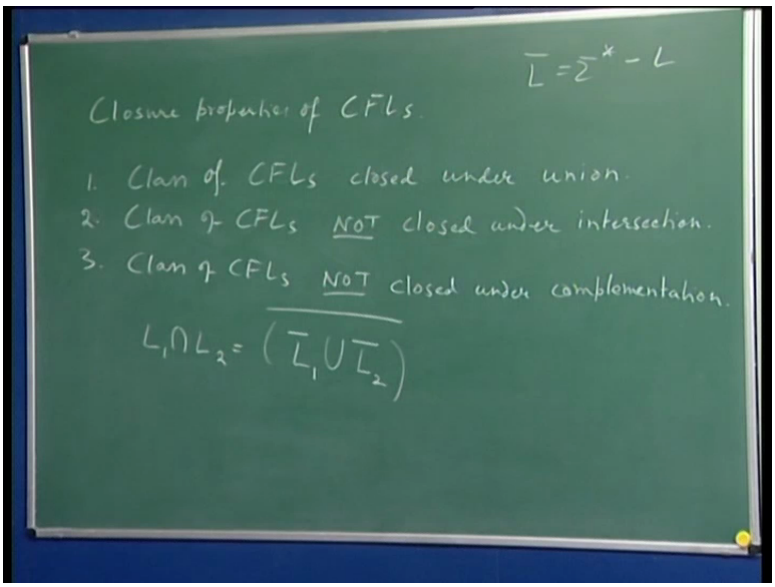
(Refer Slide Time: 43:25)



So what we are saying is it is possible that you take a CFL context free language L, you complement it and you get a language which is not context free. Or similarly it is possible that you take a not context free language, take its complement and itmay turn out to be (con) context free. So you see the point I am making is that this 3 actually follows from 1 and 2.

Why? Because I can write intersection of two languages L 1 L 2 as you know take the language L 1, complement it, take the language L 2 complement it, taketheir union, right? And then complement the entire thing. This is just a set theoretic way of expressing intersection in terms of union and complementation, right?

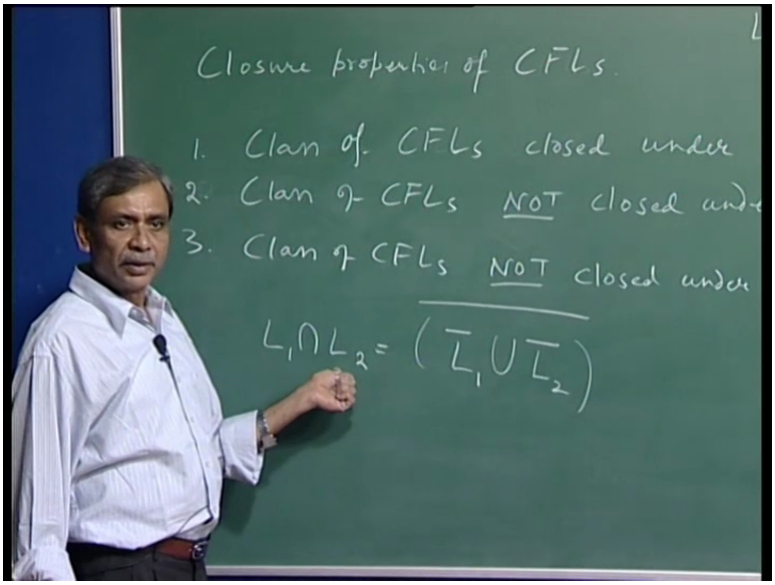
(Refer Slide Time: 44:32)



Essentiallythose of you know it is kind of De Morgan's law. So L1 intersection L 2 isthis. Now suppose the class of CFLS where I mean this class was closed under complementation. So now suppose L1 and L2 are context free languages. If the class was closed under complementation then this will be a context free language L 1 complement. L 2 complement is another context free language. You are taking their union which is union we know isyou know preserves context freeness.

So then this whole thing will be context free. Again you are taking a complementation of a context free language which by assumption would be again context free. So then what would it mean? That L1 intersection L 2 would be also context free.

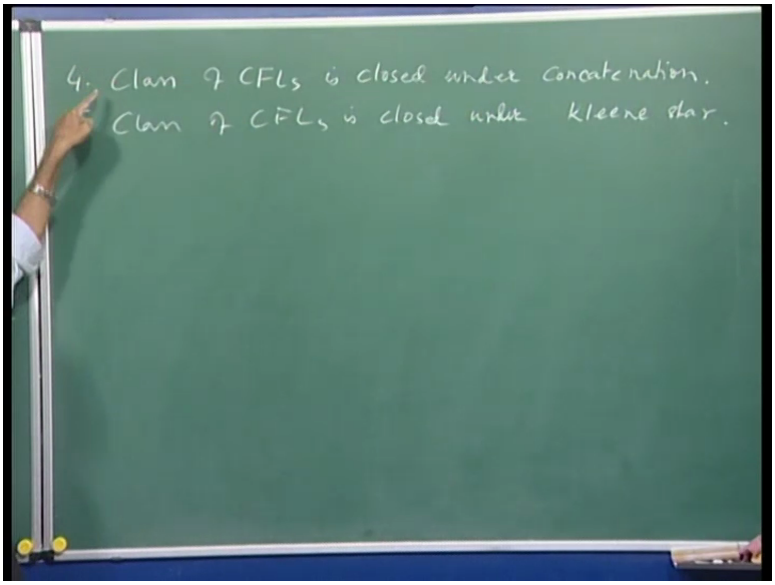
(Refer Slide Time: 45:36)



So in other words what I am saying is ifclass of context free languages, this class was closed under complementation along with the fact thatthis class is closed under union, we would get that the class is closed under intersection. But we know this class is not closed under intersectionand we also know it is closed under union. So what is the possibility left that it is not closed under complementation, right? So that is how we proved 3.

There are other closure properties which we can very easily see. You know we talk of two other things often which is concatenation. So we will say in fact this wecan prove again these two things. Class of CFLS is closed under concatenation and class of CFLS, this class is also closed under Kleene star, okay. Remember thatthis is a unary operator whereas this is a binary operation.

(Refer Slide Time: 47:20)

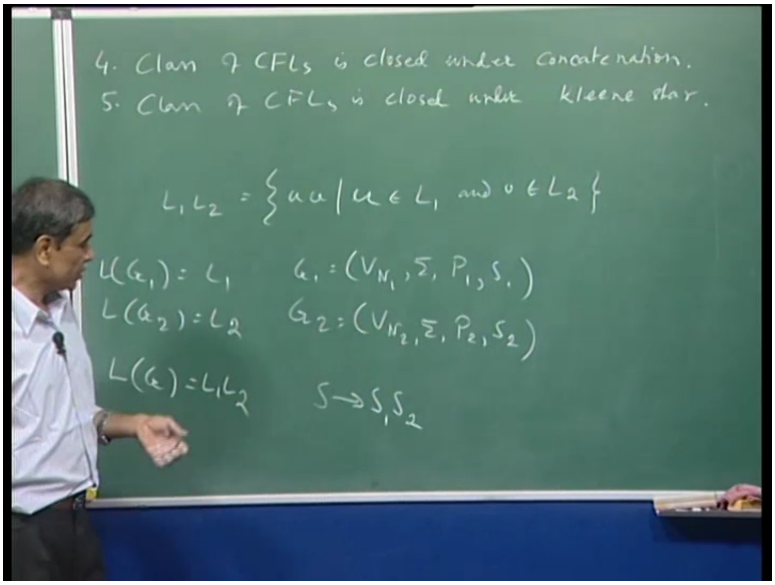


So we take two languages L1 and L2 and this is the concatenation language. Remember that L1 L2 is nothing butconcatenation of u and v such that u is in L 1 and v is in L 2, right? Again this is very easy to see that class of CFLS is closed under concatenation. Why? Because you know for L1 you have a grammar, L 2 you have a grammar. Both are context free grammars.

You know something what we use for union, same ideawe will use that supposethe (langua) language generated by G 1 is L 1 and the language generated by G 2 is L 2, right? And let us say G 1 is V N 1 sigma P 1 S 1. G 2 is V N 2 sigma P 2 S 2. And the newgrammar for L1 L2 will be a G which iswe will obtain by taking the unions of V N 1 and V N 2 assuming that they are disjoint.

You canunderstand why I want them to be disjoint because otherwise the derivation of one will interfere with the other. And we will add a newstart symbol S and that we will write it as, we will just add this production and rest of the productions will remain, G 1 and G 2, right?

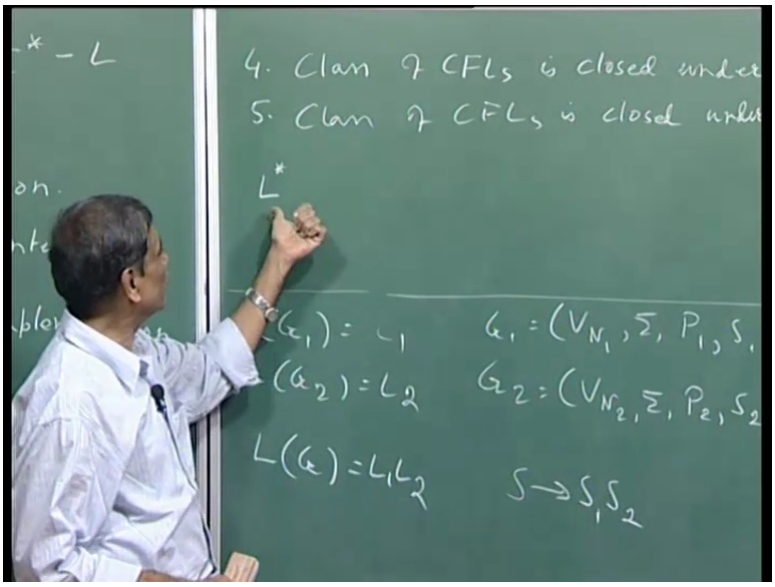
(Refer Slide Time: 49:23)



So you can easily see that if you start with this S, you generate S 1 S 2. S 1 and S 2 are the start symbols of G 1 and G 2. So through S 1 I can generate any string of L 1. With S 2 I can generate any string of L2 and therefore I can generate the concatenation language.Kleene star was if you recall we used it extensively in case of regular languages.

You know there was language L and Kleene star was the star option. So what was it very briefly?A star is a set of all strings obtained by taking 0 orconcatenation of any number of strings from L, okay. That will do.

(Refer Slide Time: 50:21)



You will remember what it was. Now L star, so basically L star is a CFLif L is a CFL. This is the meaning of 5, right? And again that is not very difficult to see that suppose L is generated by G wherethe grammar is V N sigma P S, right? So all we have to do is to just add one more production. So basically sorry I should not have said this way. So I should have saidsuppose G is this such that L G is L then for L star the grammar is G 1, right? Which is just you add one more production P union.

Just add this production. Keep the same start symbol. S goes to S S or epsilon, right? And you can easily see thatL G 1 is L star, right? Because you see then now starting from S you can generate any number of copies of S and from each S you can generate a string of L. And that is what would generate the language L star.