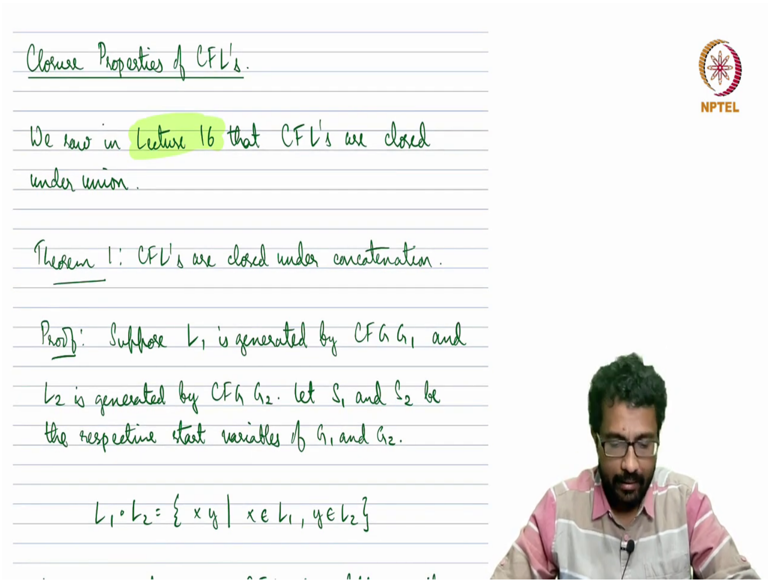
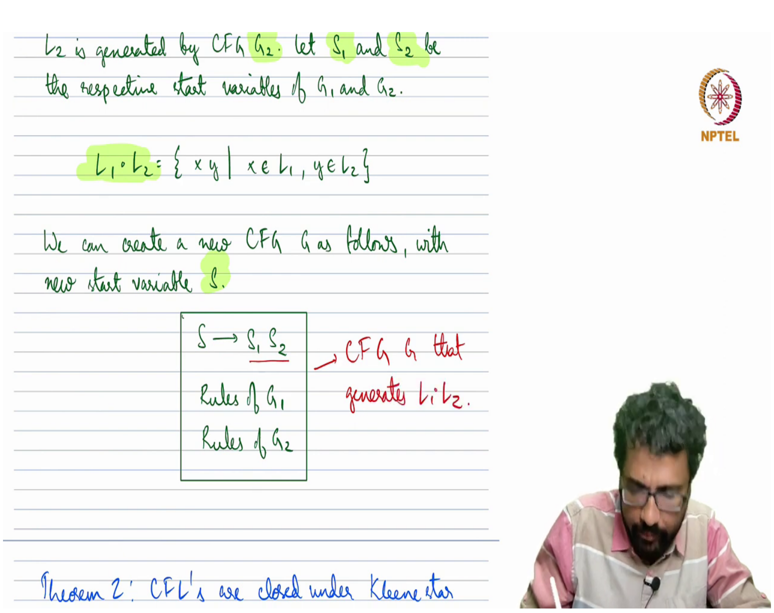
Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Closure Properties of Context Free Languages

(Refer Slide Time: 0:16)





Hello and welcome to the 19 of the course Theory of Computation. In this rather short lecture, we will see the closure properties of context-free languages. So, in the previous couple of lectures we saw what context-free languages are, we saw the definition, we saw examples, then we saw the Chomsky normal form, and then we saw the CYK algorithm.

In this lecture, we will see some of the closure properties. So, in lecture 16 which is the first lecture where we saw context-free languages, we already saw that context-free languages are closed under the union. So, in this lecture we will see a couple of more closure properties. So, the first thing that we will see is that context-free languages are closed under concatenation.

So, let us try to recollect what concatenation was, so if L1 is a language and L2 is a language, the concatenation of $L1 \circ L2$, which is defined like here is the set of all strings x y, the concatenation of all strings x y, where x is a string from L1 and y is a string from L2. So, basically it is a concatenation, it is a set of all concatenations where the first string comes from L1 and the second string comes from L2.

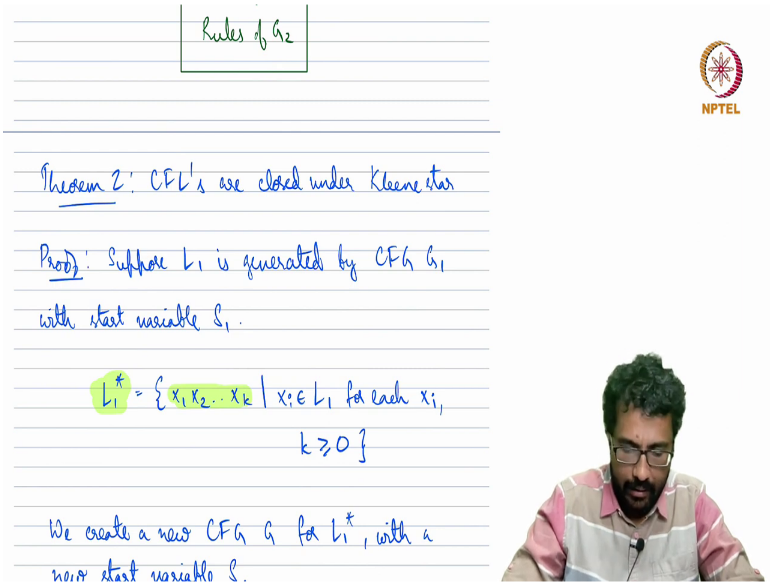
So, we want to show that if L1 and L2 are context-free languages, then this context free, this language that is defined here, the concatenation of L1 with L2 that is also context free. This turns out to be somewhat straightforward. So, we proceed the normal way. So, if L1 and L2 are context-free languages. So, we may assume that L1 is generated by some context-free grammar G1 and L2 is generated by some other context-free grammar G2 and S1 be the start variable of L G1 and S2 be the start variable of G2.

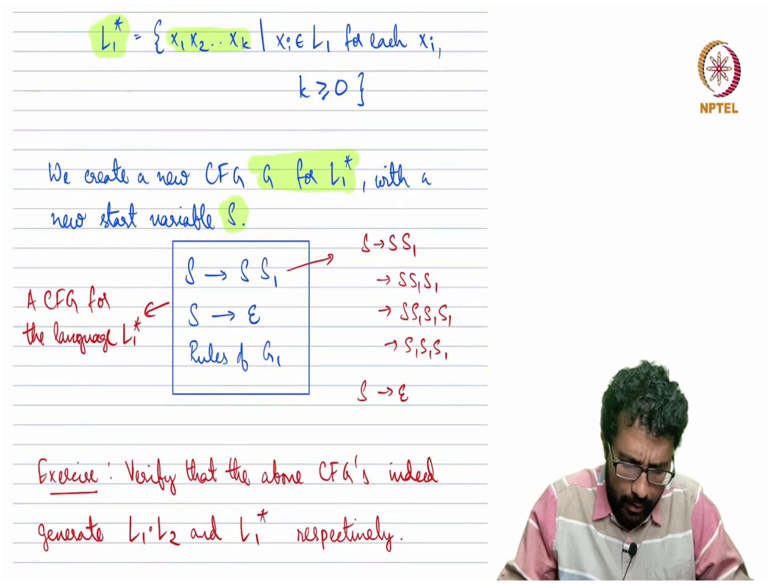
So, all we need to do. So, we need to produce all the strings x y where x comes from L1 and y comes from L2. So, which means x is generated by the context free grammar G1 and y is generated by the context free grammar G2. So, all that we have to do is to create a new start variable S, create a new start variable S and have that S generate S1S2.

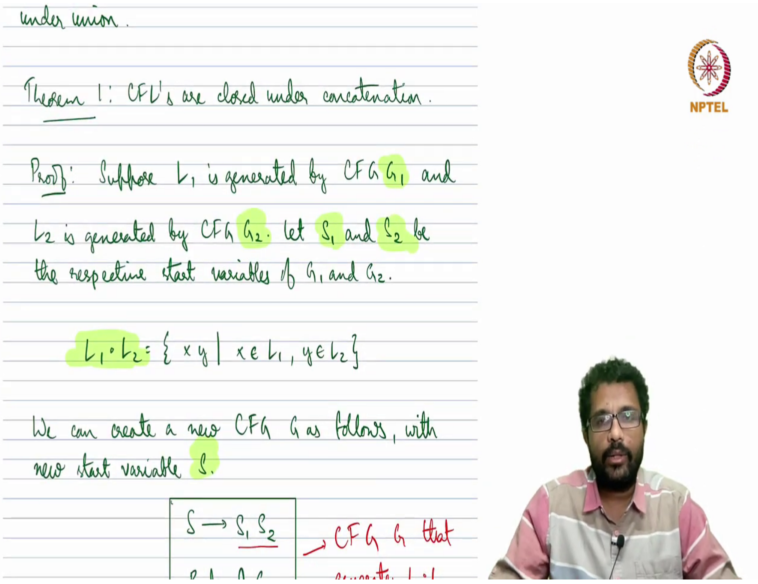
So, add a rule where S yields S1S2 and then along with that, we add all the rules of the grammar 1 we add all the rules of grammar 1 and then all the rules of grammar 2. So, which means S gives S1 S2 and this S1 is a start variable for grammar 1. So, we are also adding all the rules of grammar 1. So, using those rules S1 will generate some string in L1 which is generated by the grammar 1 and using the rules of G2 S2 will generate some string in L2.

So, as a result this will exactly generate, S will exactly generate all the strings that are on the form x y then x is from L1 and y is from L2. So, this is the context-free grammar for this is the context free grammar G, so the context free grammar G that generates L1 L2. So, that is how context-free languages are closed under concatenation.

(Refer Slide Time: 4:14)







So, the next one is closed under Kleene star. So, remember a Kleene star. So, given a language L1, the star of L1 also called the Kleene star is just strings of the form x_1 x_2 up to x_k, where each of these x 1s, so each x1 x2 each of these xi's are part of the language L1. So, each, so basically it is a k string from the same language one after the other, where k could be anything, k could be 0, k could be 1, k could be 2 and so on.

So, you could have one string from the language, two strings from the language, one after the other or three strings from the language, one after the other, after the other and so on. It could also be 0 strings in which case we get the empty string. So, suppose, now we want to show that given L1 is a context-free language, which means if L1 has a context program or G1 with the start variable S1 can we build a context free grammar for the $L1^*$.

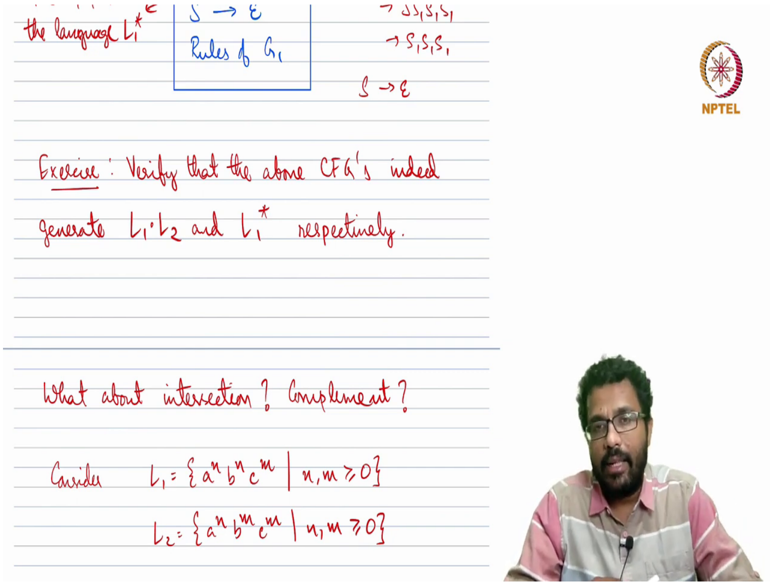
So, we will build in context free grammar for L1 star we will call it G and with the new start variable. So, the new start variable is S and we add two new rules along with the rules of G1. So, we made two new rules and now the start variable is S which is the start variable of the new grammar and not S1, S1 was the start variable of G1. So, now S is the new start variable. So, the first rule is that S gives S S1. So, S gives S S1. So, I am referring to this rule.

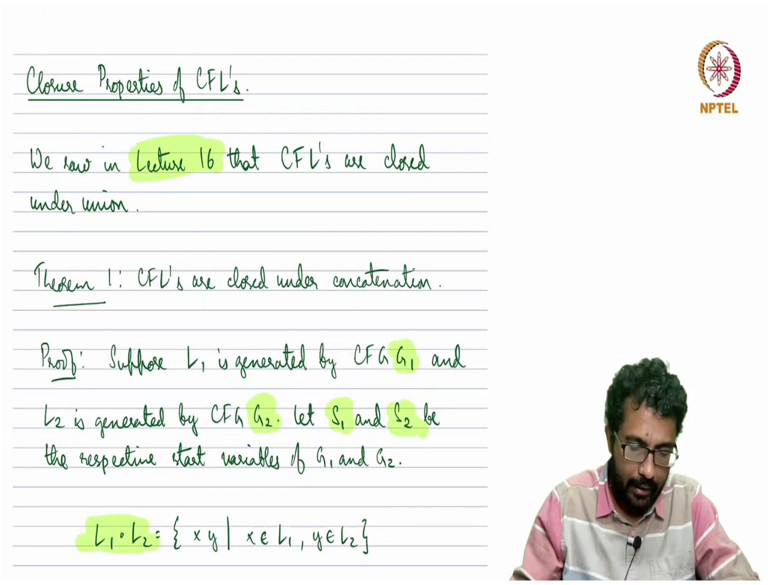
So, now S gives S S1, now this S1 will generate some string in the language L1 and again this S. So, we may have things like we may again have a rule or again we may apply this S may be again giving S S1. So, which means I will get two strings from L1 and maybe let us say maybe let us say this S, now we use a second rule which is S gives an empty string. So, which means I get S1 S1.

So, now I use in this manner now each of the S1s can generate some string from L1. Consequently, we get some like x1 x2, where both x1 and x2 come from L1. Now we may we can also get, sorry, we can also get this S to again get S S1 and then go empty, in which case you get some string or three strings from L1, one after the other, after the other. We may also get, let we may also directly use a second rule where S gives empty string, which gives us the empty string which is also part of $L1^{*}$.

So, that way this construction gives us a new grammar or a CFG for the language L1 star. So, this is how we get context free grammar for the language L1 star. So, an exercise is to just verify that these two grammars indeed generate the languages that they were supposed to generate.

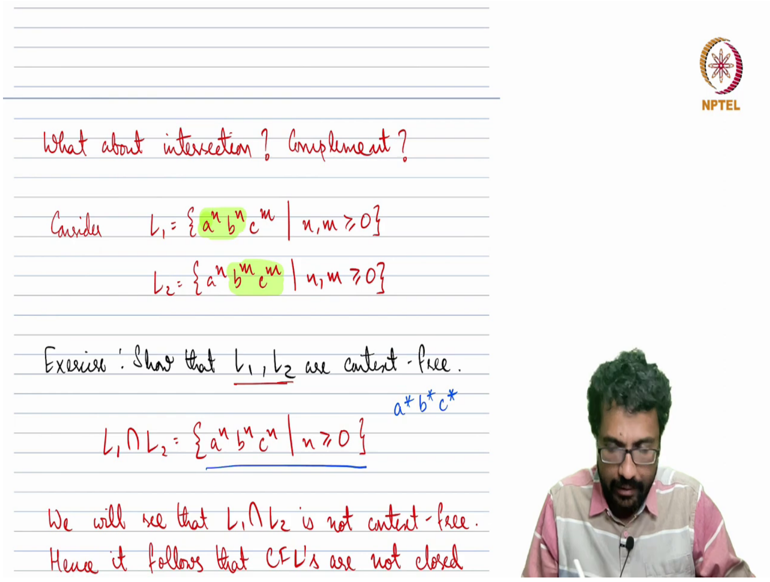
(Refer Slide Time: 8:23)

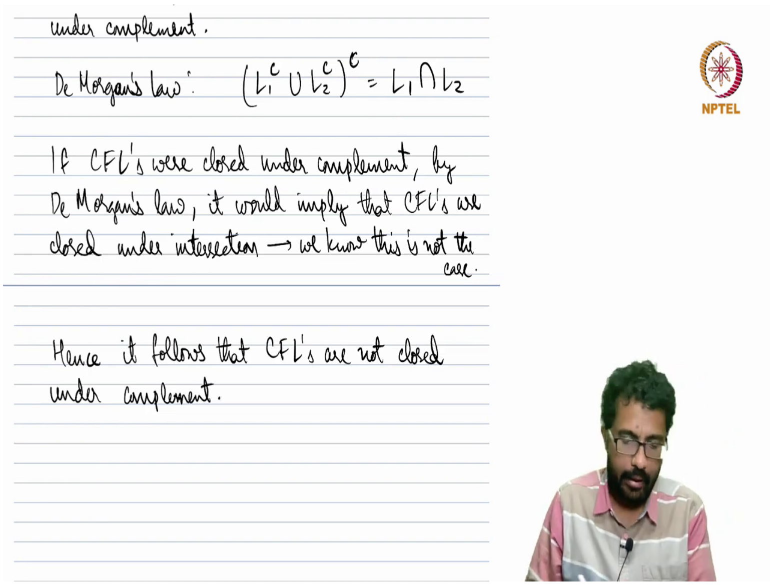


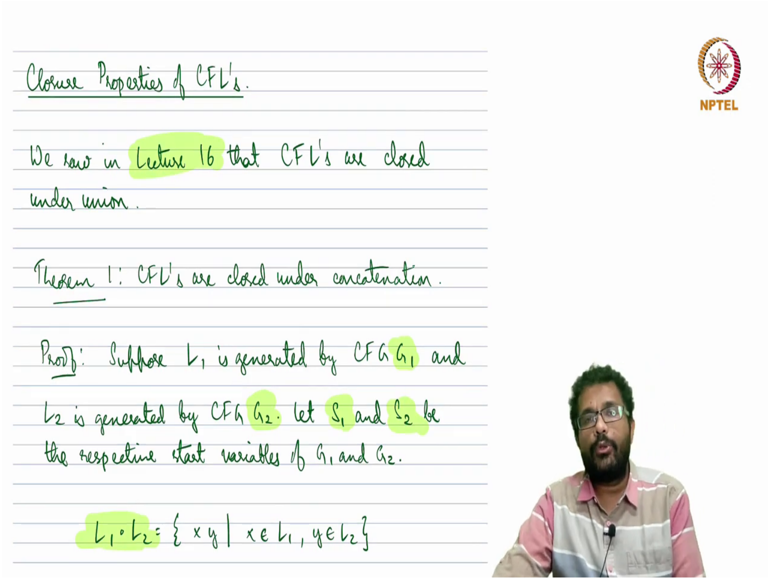


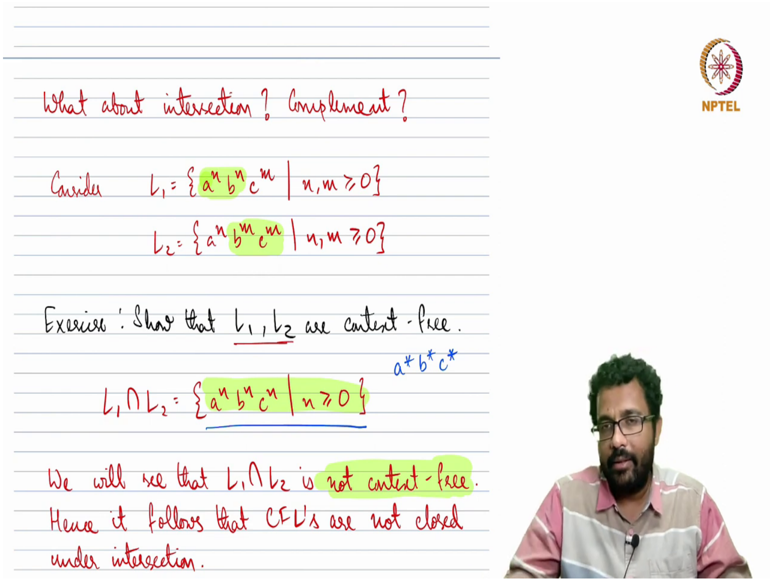
So, now the natural question is, so in the case of regular languages, we saw that regular languages are closed under union intersection complement a star operation and concatenation. So, what about the context-free languages, is it, so we already saw union in the earlier lecture, lecture 16, then now we are seeing a concatenation and Kleene star. So, what about intersection? What about complement?

(Refer Slide Time: 8:55)









So, the **answer is that, it is not a closed under intersection**, it is also not closed under the complement. So, let us see why that is the case. So, consider these two languages that I have written here L1 is $a^n \ b^n \ c^m$. So, it is of the form $a^* \ b^* \ c^*$ some A is followed by some B's followed by some C's, where the number of A's and B's are the same, number of A's and B's are the same, that is L1. So, C is repeated m times where m is different from n.

And L2 is $a^n \ b^m \ c^m$, where the number of B's and C's are the same which is m and A is repeated some other number of time. So, n is different from m. So, L1 has the same number of S's and B's, L2 has the same number of B's and C's and it is not that difficult. So, already we have seen other languages which are context-free. So, one exercise is to show that L1 and L2 are context free. So, the L1 and L2 as written here are context free, it is not that difficult to show, you can try to show that. But let us see what happens when we take the intersection of these languages.

So, L1 and L2 both are of, both give a strings of the form A star B star C star, it is some A is followed by some C's followed by some C's, both maintain this structure. Now L1 intersection L2 also means since both the language maintain their structure L1 intersection L2 also maintains the structure. However, L1 consists of all the strings where the number of A's is equal to the number of B's and L2 consists of all the strings with a number of B's equal to number of C's.

So, L1 intersection L2 will have to satisfy both, meaning the number of A's must equal the number of B's and number of bees must equal the number of C's. So, L1 intersection L2 are this all the strings of the form A star B star C star, where the number of A's B's and C's are all the same. So, in other words we get this language L1 intersection L2 is A power n B power n C power n.

And later in the like maybe couple of lectures down the line, we will see that, this A power n B power n C power n, for the same n, this is not a context-free language. So, this language is not context-free. So, which means context free languages are not closed under intersection. So, this is something that we have not proved yet like we have we are not even discussed, how to show that something is not context free.

So, the way we will show it is, we saw pumping lemma for regular languages. Similarly, we will see pumping lemma for context-free languages and using that we will be able to show that this language is not context-free. So, here we have two languages L1 and L2 which are both context free but the intersection is not context free, which means they are not closed under intersection.

And because they are not closer to intersection this also implies that, they are not closed under the complement, why is that? This is because of De Morgan's law. So, this says that A1 complement, sorry, it is L1 instead L1’ ∪ L2’ and the whole complement, this is one of de Morgan's law is L1 ∩ L2. So, if context-free languages are closed under complement then L1 complement would have been a context-free language L2 complement would have been a context-free language. And the union we already saw that context-free language is closed under the union.

So, the union also would be a context-free language and by assumption the complement is again context-free. So, the L1 complement union L2 complement the whole complement is also going to be context free which means L1 intersection L2 is context-free. So, if CFLs were closed under complement by De Morgan's law, it would imply that CFLs are closed under intersection because basically by combining union and complement you can get intersection.

So, if they were close we already know it is closed under union, if they were closed under complement we would get a complement under intersection. We know this is not the case, we know they are not closed under intersection, this is not the case. Hence it follows that, sorry, it follows that CFLs are not closed under, sorry, complement. Because if it is closed under complement it would imply that they were closed under intersection which we know is not the case.

So, that is all that I had in this particular lecture. So, we saw that we already, we recall that CFLs are closed under union, we saw that it is closed under concatenation by making a grammar, we saw that it is closed under star, again by making grammar and we saw by an example that they are not closed under intersection, context free language is not closed under intersection.

So here we took for granted that this language this, this language that I am highlighting right now, maybe I will use a this language that I am highlighting right now, sorry, this does not look nice, maybe I will use a different color for highlight, maybe this language that I am highlighting right now this is assuming that this language is not context-free.

So, later we in the course maybe in 3 4 lectures down the line, we will see that this is not context free and hence they are not closed under intersection and by De Morgan's law if it is closed under complement and union it follows that it is closed under intersection, we know it is closed under union. So, if it is closer to complement, it implies that it is closed under intersection which we know is not the case. Hence, it follows that context-free languages are not closed under complement. So, not closed under complement.

So, context-free languages are not closed under complement intersection but are closed into the regular operations, which are union, concatenation, and start. And that is all I have for you in this lecture, see you in the next lecture.