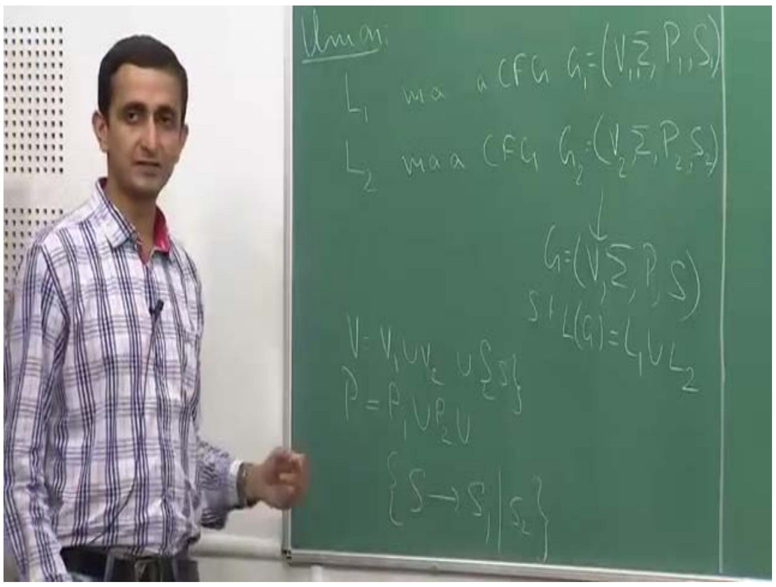
Theory of Computation

Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 24
Closure Properties of CFLs

Welcome to the 24th lecture. Today we are going to study Closure Properties of Context Free Languages.

(Refer Slide Time: 00:14)

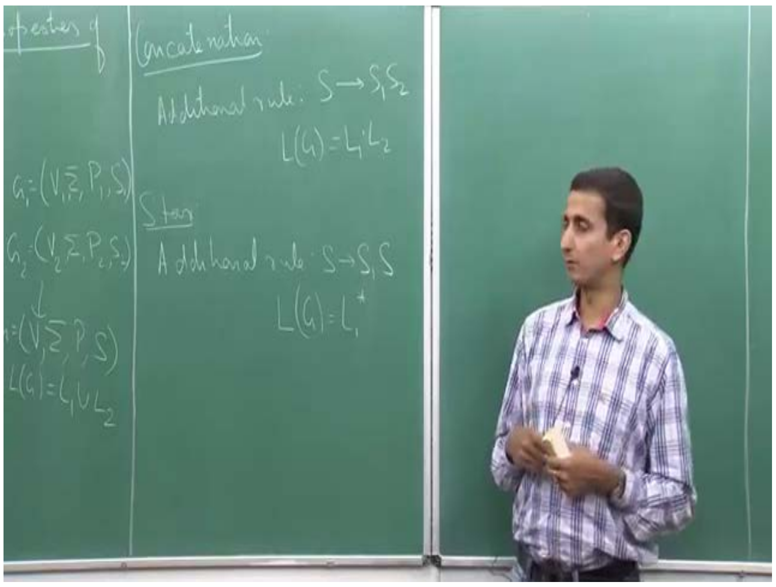


The first closure property that we will see is union and the context free languages are closed under union, which means that if L 1 is a context free language and L 2 is a context free language and L 1 union and L 2 is also a context free language. So, the proof is not very difficult.

Let say that L 1 is a context free language via a context free grammar G; let say G 1 equals V 1 sigma P 1, S 1 and L 2 is a context free language via a context free grammar G 2 which is V 2 sigma P 2, S 2. So, using these 2 we will construct a grammar G, which I will call V sigma P, S such that language of G will be equal to L 1 union L 2.

So, how do we construct this grammar? The thing is that the variables we will just be the union of these 2. So, let we will just assume that the symbols that we use for the variables of G 1 and G 2 are different. This is just for so that we do not mix. So, a used here should not be the same as a used here. So, we can give them separate names, but basically what we will have is V will be equal to V 1 union V 2, production rules also P will be equal to P 1 union P 2. Now, we have a variable S, actually what is this variable S? This is the new variable that I add to V. So, this is the variable, a new variable that I add and in the production rule I will have another rule, which takes S to S 1 or S 2. This is generating, all strings that are generated by S 1 or all strings that are generated by S 2. So, this is why it accepts the union. This is the main construction.

(Refer Slide Time: 03:18)

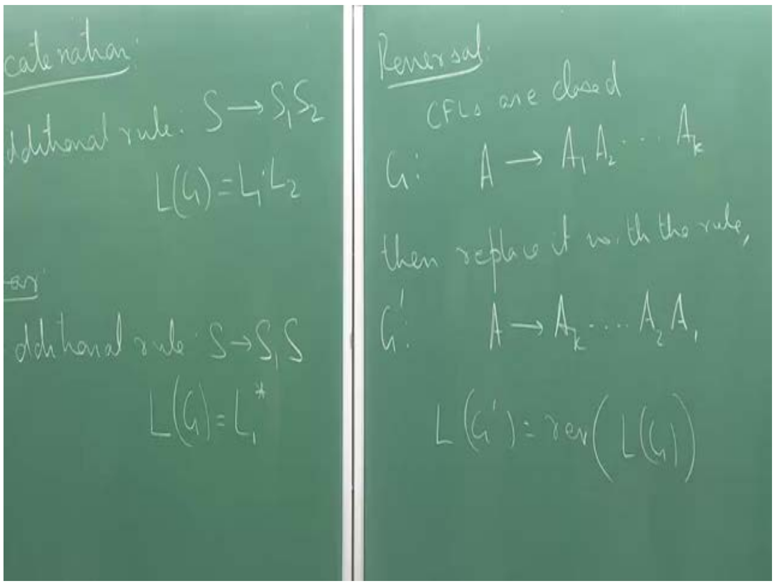

operties of
Concatenation
Additional rule: $S \rightarrow S_1S_2$
$L(G)=L_1 \cdot L_2$
Star
Additional rule: $S \rightarrow S_1S$
$L(G)=L_1^*$
$G_1=(V_1,\Sigma,P_1,S_1)$
$G_2=(V_2,\Sigma,P_2,S_2)$
$=(V,\Sigma,P,S)$
$L(G)=L_1 \cup L_2$

Similarly, if we have concatenation, how do we do? So, the entire construction remains the same. Again we take the variables as the union of the 2 variable sets, the productions as the union of the 2 production sets and we have new start variable S, but this time we have additional rule. We will have an additional rule of the form S going to; let me write this as S going to S 1, S 2. So, it generates a string generated by S 1. So, it generates a string over the language L 1 first and then concatenates with the string generated over the language L 2. This will generate by this construction L of G will be L 1 concatenated with L 2.

The next operation that we will see is the star operation. Again, in the case of star operation we only have a single grammar. Let say we have a single grammar G 1, which is a grammar for L 1 and I want to construct the a grammar for L 1 star. So, I have a grammar where I have a new production; I have a new variable S and I have a new rule S going to S 1, S and by this language of G will be language of L 1 star. This is for star. These are the three basic closure properties and we have seen that regular languages are also closed under these three properties, but there are other closure properties as well, for example, intersection complement reversal homomorphism inverse homomorphism and so on.

Let us look at some of those properties, and see what happens when we try to study the closure properties of context free languages under these properties. So, what about reversal?

(Refer Slide Time: 06:18)



catenation:

dditional rule: $S \rightarrow S_1 S_2$

$L(G) = L_1 \cdot L_2$

ar

dditional rule: $S \rightarrow S_1 S$

$L(G) = L_1^*$

Reversal:

CFLs are closed

$G: \quad A \rightarrow A_1 A_2 \dots A_k$

then replace it with the rule,

$G': \quad A \rightarrow A_k \dots A_2 A_1$

$L(G') = rev(L(G))$

Again context free languages are closed under reversal and the idea is suppose, I have a rule of the form sum a goes to A 1, A 2 up to some A k then replace it with the rule A going to A k dot, dot, dot A 2, A 1. Basically look at the right hand side of A, every production rule and reverse it. So, whatever string is being generated by a will get reversed and by this construction we get a new grammar, let say from a G grammar I get another grammar prime. So, L of G prime will be the reverse of the language of G. Let say this was grammar G and this is a grammar G prime then L of G prime is reverse of L of G. Similarly, under homomorphism and inverse homomorphism also context free languages are closed.

(Refer Slide Time: 08:10)

$S \to S_1 S_2$
$L(G) = L_1 \cdot L_2$
$S \to S_1 S$
$L(G) = L_1^+$
Reversal
CFLs are closed
$G: A \to A_1 A_2 \cdots A_k$
then replace each the rule,
G': $A \to \cdots A_1$
- CFLs are closed under homomorphisms & inverse homomorphisms
- If L_1 is a CFL & L_2 is reg then $L_1 \cap L_2$ is CFL
DFA $M = (Q_M, \Sigma, \delta_M, q_M, F_M)$
PDA $N = (Q_N, \Sigma, \Gamma, \delta_N, q_N, z, F_N)$

So, context languages are closed under homomorphism and inverse homomorphism. So, for homomorphism, actually we can start with the context free grammar for the language and construct a context free grammar for the homomorphism of that language and for inverse homomorphism, we can actually do it using pushdown automaton. In our last lecture, we saw that push down automaton and context free grammars, both have equal expressive power. In other words, even push down automaton accepts the same class of languages as context free languages.

To prove inverse homomorphism, we take a language; we assume a push down automaton for that language and then we using that we construct the push down automaton for the inverse homomorphism of that language. I would not give the proof of this thing, I will just leave them. You should try them as an exercise, but let us just remember this as a fact.

Now, what about intersection? For intersection, there are 2 things. So, we can study 2 properties or 2 intersection properties of context free languages. One is of course; if I have a language L 1 which is context free and I have a language L 2 which is context free what happens to L 1 union L 2. Also, I can study the following, if I have a language L 1 which is context free and I have a language which is L 2, language which is regular then what happens to the intersection of these 2 languages.

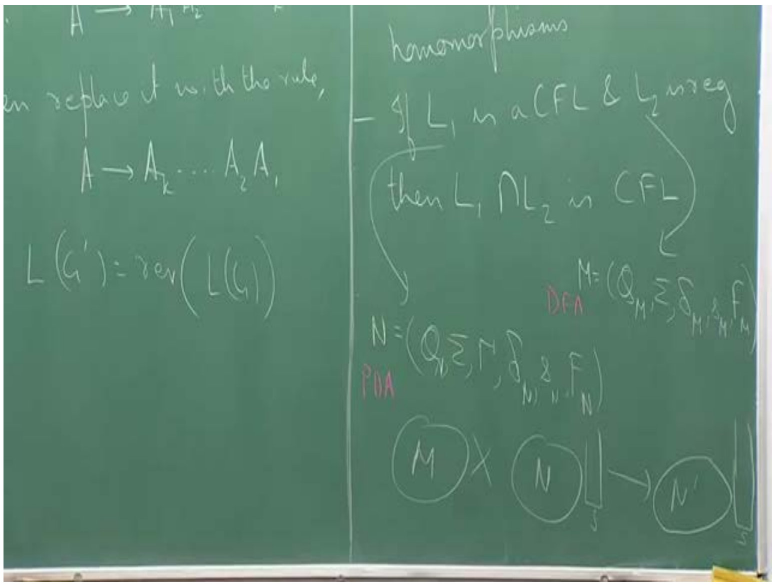
So, intersection of regular with context free and of course, context free with context free. We will study both these. If L 1 is a context free language and L 2 is regular then L 1 intersection L 2 is a context free language. Let us try to show this or in fact, even if I before giving the construction the idea is the following. Suppose I have a PDA for L 1. So, suppose there is a PDA N for L 1 which has states, let say Q 1, sigma, gamma, delta.

Let us call it Q N instead of 1, delta N, S and F, N. So, this is the push down automata for L 1 and because L 2 is a regular I can assume a DFA for L 2. Let say there is a DFA for L 2 M which is equal to Q M sigma delta M, S M, F M. So, this guy is a PDA and this guy is a DFA. So, what we will do is we will construct the product automaton of M and N. So, essentially the product automaton is the following. I look at the cross product of Q N and Q M. So, all tuples where 1 pair, where 1 element comes from Q N element and the other elements comes from Q M and that is going to be my set up states of the product automaton and from there I will make transitions simultaneously according to N and M.

So, the first element of that pair will make a transition according to Q N and the second element of the pair will make a transition according to, sorry delta M. The first will make according to delta N, second will make according to delta M. So, that is how it goes and in the end I will check if both N and M accepts then I will basically accept.

Now, comes the question of the stack because I need to prove that L 1 union intersection L 2 is a context free language, it is sufficient for me to construct a push down automata for L 1 intersection L 2 and in the push down automaton I am allowed a stack. So, I will use the stack of the push down automata to simulate the stack of N. So, I have this additional stack that is given to me.

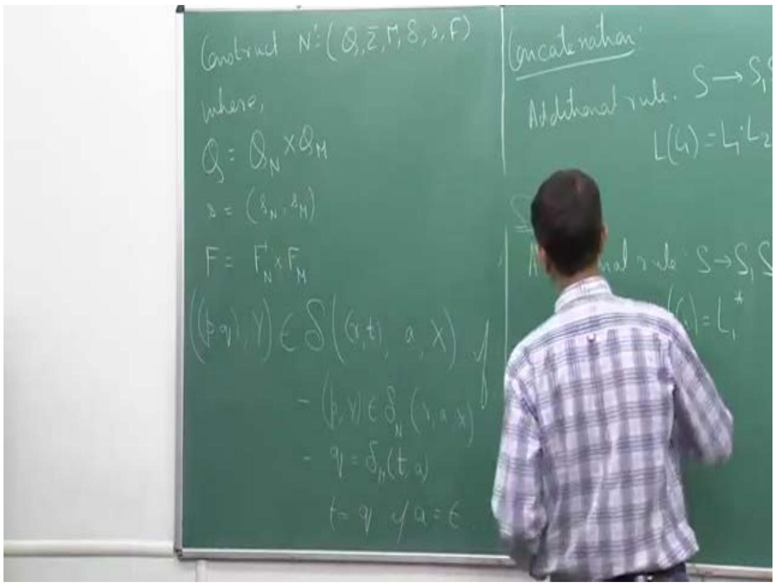
(Refer Slide Time: 13:46)



en replace it with the rule,
$A \rightarrow A_k \ldots A_2 A_1$
$L(G') = rev(L(G))$
homomorphisms
– If L_1 is a CFL & L_2 is reg
then $L_1 \cap L_2$ is CFL
DFA $M = (Q_M, \Sigma, \delta_M, s_M, F_M)$
PDA $N = (Q_N, \Sigma, \Gamma, \delta_N, s_N, F_N)$
M
N
N'
S

I have, this is my M. So, I create a cross product with N who has this stack S given to it. So, what I do is I will create an automaton, let us call it N prime and I will use the stack of N prime to simulate the stack of S. So, the formal description is as follows.

(Refer Slide Time: 14:16)



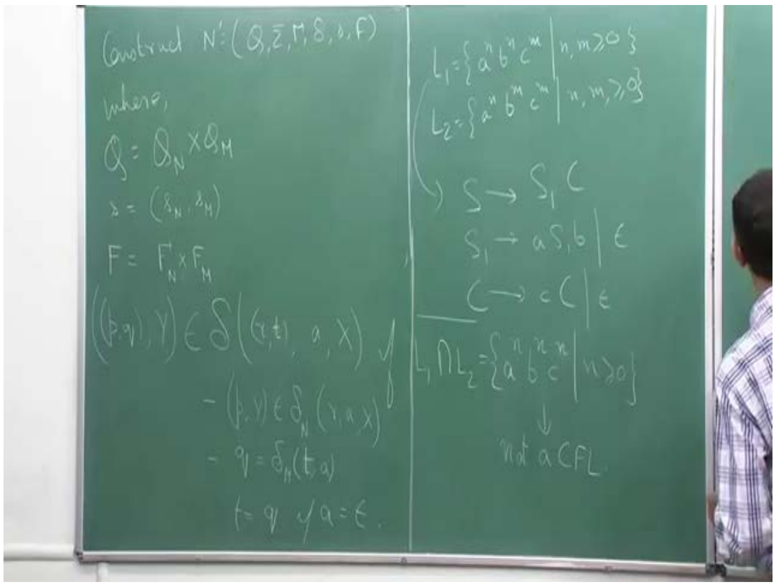
Construct $N' = (Q, \Sigma, \Gamma, \delta, s, F)$
where,
$Q = Q_N \times Q_M$
$s = (s_N, s_M)$
$F = F_N \times F_M$
$((p, q), Y) \in \delta((r, t), a, X)$ if
– $(p, Y) \in \delta_N(r, a, X)$
– $q = \delta_M(t, a)$
$t = q$ if $a = \epsilon$
Concatenation
Additional rule. $S \rightarrow S_1 S_2$
$L(G) = L_1 \cdot L_2$
$S \rightarrow S_1 S$
$L(G) = L_1^*$

So, we construct N prime which is equal to Q sigma, gamma, delta S and F, where Q is the cross product of Q N and Q M. S is basically the pair S N comma S M. So, I take the start state of both these, what about f when do I accept. So, I basically accept if both delta N and delta M make a string accept. So, basically I have I will take the cross product of F N and F M. So, both states must I look at all those pairs a comma b, where a is an accept state and b is an accept state of N and M respectively. Now, what about delta? I say that, note that this is a PDA. So, I say that some P comma Q comma Y is an element of delta of r comma S comma a comma x.

Suppose I am at a state r comma S, I read a symbol a on that, a and the top of the stack contains x then I will go to the pair P comma at Q and replace x with y, if the following happens if first of all the second the first transition is according to the transition of the PDA that is P comma y belongs to delta N of r comma a comma x. So, this transition is according to the PDA and second is Q equals delta M of S comma a. So, recall that delta M is a transition function a DFA. Therefore, this should be on the state s. So, this S and this S are not same. So, let me instead of S, let me give this a different name. So, let me call it a r comma t. So, r comma t and here also I will write as t.

So, from state t on input a, the DFA goes to state Q, but now because this is a PDA, a can be the epsilon symbol also. So, t equals Q if a is epsilon. If a is epsilon then basically, t Q is defined to be equal to t. This is the construction, now what about intersection of 2 context free languages.

(Refer Slide Time: 18:26)



Construct $N' = (Q, \Sigma, \Gamma, \delta, s, F)$

where,

$Q = Q_N \times Q_M$

$s = (s_N, s_M)$

$F = F_N \times F_M$

$((p,q), Y) \in \delta((r,t), a, X)$ if

- $(p, Y) \in \delta_N (r, a, X)$
- $q = \delta_M(t, a)$

$t = q$ if $a = \epsilon$.

$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$

$L_2 = \{a^n b^m c^m \mid n, m \geq 0\}$

$S \rightarrow S_1 C$

$S_1 \rightarrow a S_1 b \mid \epsilon$

$C \rightarrow c C \mid \epsilon$

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$

$\downarrow$

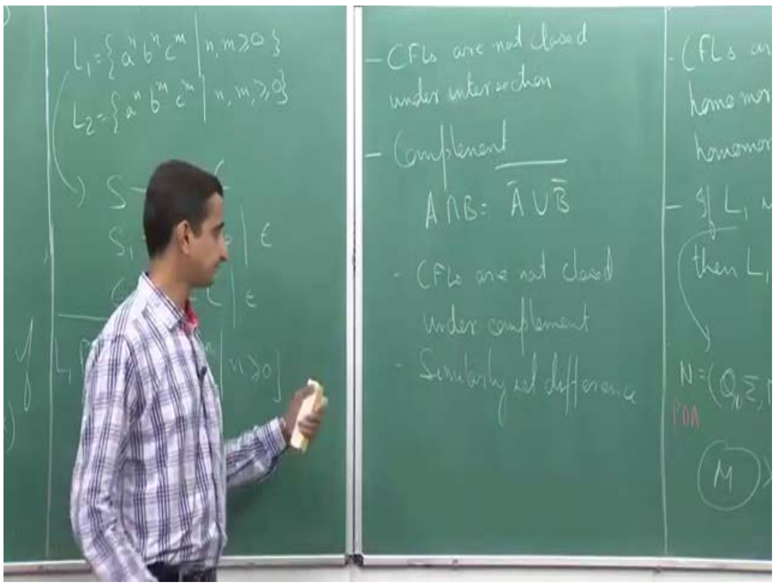
not a CFL.

Let us look at 2 languages; L 1 which is a to the power n, b to the power n, c to the power m, where n comma m is greater than 0. So, this is one language and another language a to the power n, b to the power m, c to the m, where once again n and m are greater than 0. In the first language, the number of a’s is equal to the number of b’s, c can be arbitrary in the second language the number of b’s is equal to the number of c’s a can be arbitrary. So, you can easily show that L 1 and L 2 are context free language.

In fact, I can give context free grammar for L. For L, I can give following grammar. So, I have S which generates strings of the form S 1 c. So, S 1 will generate all strings of the form a S 1 b or epsilon and c will generate strings of the form small c, capital c or epsilon. So, the first part generates all strings are that have equal number of a’s and b’s and c generates some arbitrary number of c’s. So, this generates the language L 1. Similarly, you can construct a grammar which generates the language L 2. So, this proves that L 1 and L 2 are context free.

Now, what about L 1 intersection L 2 what is this language? If you look at this language, this language contains all those strings that have a’s followed b’s followed by c’s, where the number of a’s must be equal to the number of b’s because they are strings in L 1 and the number of b’s must be equal to the number of c’s because they are strings in L 2 as well which implies that a’s, b’s and c’s must be the same number. It has strings of the form a to the power n, b to the power n, c to the power n which we have already shown which we have already seen is not a context free language. Therefore, context free languages are not closed under intersection.

(Refer Slide Time: 21:08)

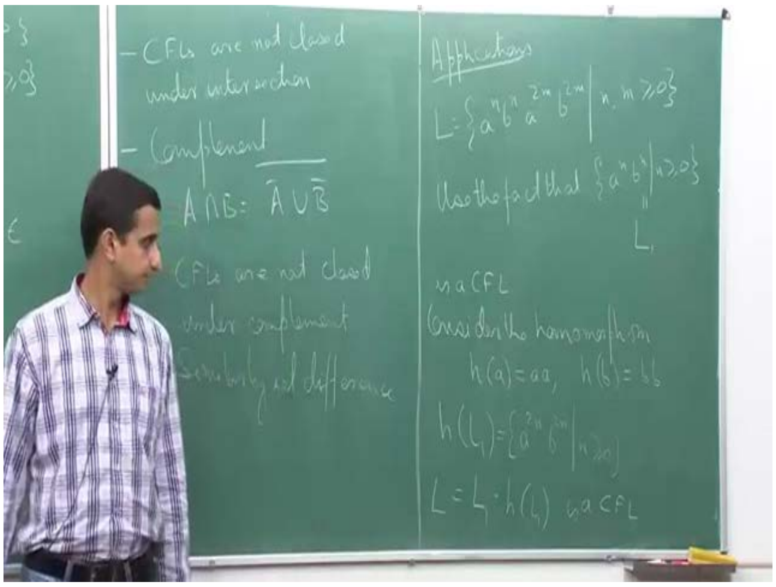


$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$
$L_2 = \{a^n b^m c^m \mid n, m \geq 0\}$
- CFLs are not closed under intersection
- Complement
$A \cap B = \overline{\bar{A} \cup \bar{B}}$
- CFLs are not closed under complement
- Similarly set difference

What about complement? If we want to study complement, I can write A intersection B as A complement union B complement, whole complement using Demorgan's law. So, if context free languages were closed under complement then, if A and B are context free languages; A complement would be a context free language, B complement would be a context free language. We just saw today that union of 2 context free languages, the context free language. So, a complement union b complement is a context free language and again we take whole complement of a context free language.

Hence, the right hand side is a context language, but we just prove that intersection of 2 context free languages is not necessarily context free language hence complement is your context free languages are not closed under complement. Similarly, also under set difference, now let us look at an application of these closer properties.

(Refer Slide Time: 23:29)


- CFLs are not closed under intersection
- Complement
$A \cap B = \overline{\bar{A} \cup \bar{B}}$
CFLs are not closed under complement
Similarly and difference
Applications
$L = \{a^n b^n a^{2m} b^{2m} \mid n, m \geq 0\}$
Use the fact that $\{a^n b^n \mid n \geq 0\} = L_1$ is a CFL
Consider the homomorphism
$h(a) = aa, \quad h(b) = bb$
$h(L_1) = \{a^{2n} b^{2n} \mid n \geq 0\}$
$L = L_1 \cdot h(L_1)$ is a CFL

We have seen several closer properties so far. Let us quickly look at an application. Suppose, I consider the language L equals a to the power n, b to the power n, a to the power 2 m, b to the power 2 m, where n comma m are greater than or equal to 0. How can we prove that this language is a context free language without constructing a grammar for it using closer properties? So, we will use the fact that a to the power n, b to the power n. Let us give this the language a name, let us call this L 1 is a context free language.

Now, consider the homomorphism h, which takes a to a a and b to b b, what does h do to n the language L 1. So, it takes if I have a string of the form a to the power n, b to the power n, it will take that string 2 a to the power 2 n, b to the power 2 n. So, it creates; it has all strings of the form a to the power 2 n, b to the power 2 n, even number of a’s and even number of b’s. So, it has a to the power 2 n, b to the power 2 n for n greater than or equal to 0.

Now, observe that the language L that we started off in the beginning is nothing, but concatenation of L 1 with h of L 1. So, L basically has all strings which have a part which is a to the power n, b to the power n and another apart which is a to the power 2 m, b to the power 2 m. So, it take some string from L l 1 and it take some string from h of L 1 and it is produces a string in l, because regular a context free languages are closed under homomorphism and concatenation. Therefore, L is a context free language. So, I will stop here.

Thank you.