Theory of Computation

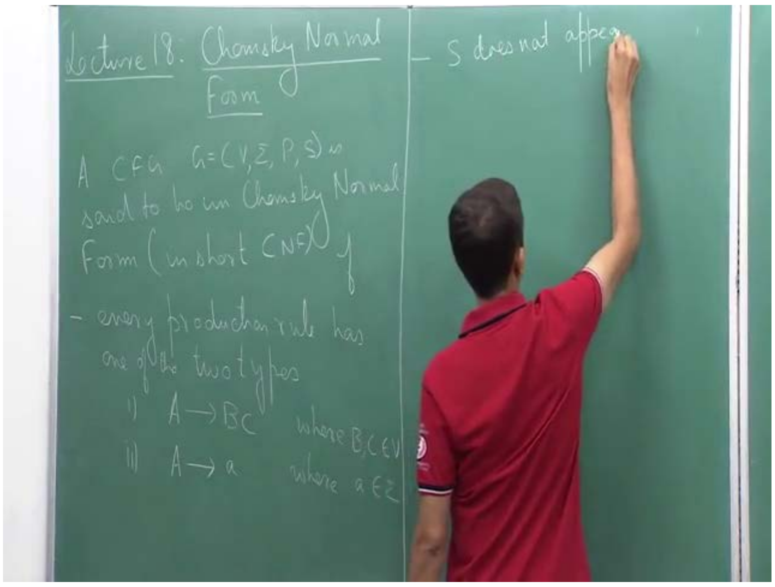
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 18
Normal forms, Chomsky normal form

Welcome to the 18th lecture of this course. Today we are going to talk about Normal Forms of Context Free Grammars. And in particular, we will look at this particular normal form known as the Chomsky normal form. So, this normal form is named after a computer scientist Noam Chomsky.

So, what are Normal forms? So, normal forms of context free grammars are grammars where the production rules have a very specific type of form or they follow a very specific nature. Let me define what Chomsky normal forms are and we will then talk more about it.

(Refer Slide Time: 01:08)



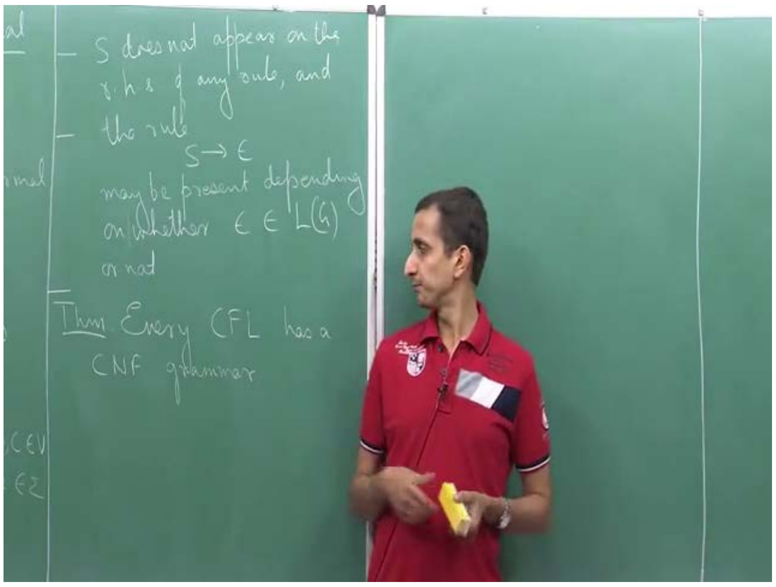
Lecture 18: Chomsky Normal Form

A CFG $G=(V,\Sigma,P,S)$ is said to be in Chomsky Normal Form (in short CNF) if

- every production rule has one of the two types
  - i) $A \to BC$ where $B,C \in V$
  - ii) $A \to a$ where $a \in \Sigma$
- S does not appear

So, a context free grammar G equals V, sigma, P, S is said to be in Chomsky normal form or in short CNF if first of all every production rule has one of the two types. So, firstly, it can be some A going to BC, where B and C are variables, or it can be some A going to small a, where small a is a terminal symbol. So, all the production rules are of the following form. So, you either have a variable going to a concatenation of two variables or a variable going to a terminal; well, almost all production rules there is just one exception, I will come to it. So, this is the first condition that every production rule has one of the two types.

(Refer Slide Time: 03:43)

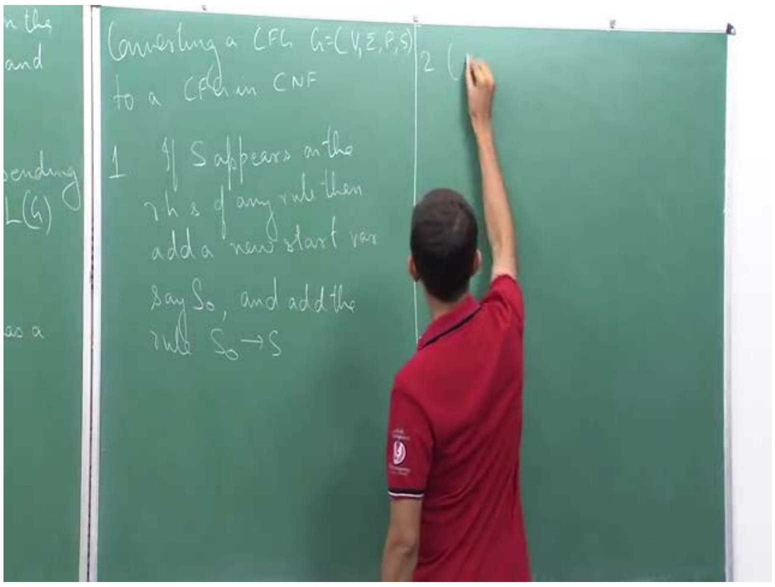

– S does not appear on the r.h.s of any rule, and
– the rule
S → ε
may be present depending on whether ε ∈ L(G) or not
Thm. Every CFL has a CNF grammar

The second is S does not appear on the right hand side of any rule. So, our start variable, it does not appear on the right hand side of any rule; in the sense that it does not appear on the right hand side. And thirdly, the rule S going to epsilon may be present depending on whether L of G contains epsilon or not. So, the first here is epsilon. So, depending on whether epsilon is contained in L of G or not. So, if the language of the grammar contains epsilon then we will have one exception to this rule. So, we will have a production rule of the type S going to epsilon otherwise we do not this, so that is only exception.

If a context free grammar has this kind of a form, it is said to be in Chomsky normal form. So, why normal forms so important the reason is so let me write this as a theorem every context free language has a Chomsky normal form grammar. So, no matter what context free language you take, there is some context free grammar for that language which is in Chomsky normal form. So, this is a very important thing to say.

So, what we will do today is, we will prove this; and our proof will again be a constructive proof in the sense that we will take a context free language, and we will take any grammar for that language, it can be any arbitrary grammar. And we will show how to convert that grammar to a grammar in Chomsky normal form. So, we will show how to convert any arbitrary context free grammar to a grammar in Chomsky normal form.

(Refer Slide Time: 07:05)



Converting a CFG $G=(V,\Sigma,P,S)$ to a CFG in CNF

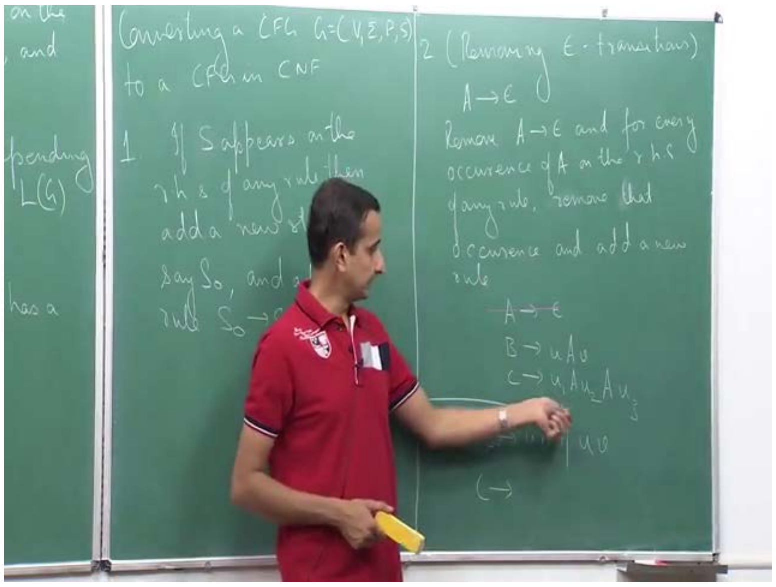
1 If S appears on the r h s of any rule then add a new start var say S_0, and add the rule $S_0 \to S$

2 (

So, converting a CFG to a CFG in Chomsky normal form; of course, when we say we convert it must both the grammars must accept the same language. So, how do we do? So this will give will device an algorithm and the algorithm will follow a step-by-step process.

The first step is to make sure that the second conditions hold that is the start variable does not appear on the right hand side of any rule. So, if appears on the right hand side of any rule, then add a new start variable say S naught and add the rule S naught going to S. So, we first step, if S appears on the right hand side, we add the rule S not going to S; and this becomes our new start variable. So, this is removing start variable from the right hand side.

(Refer Slide Time: 09:01)

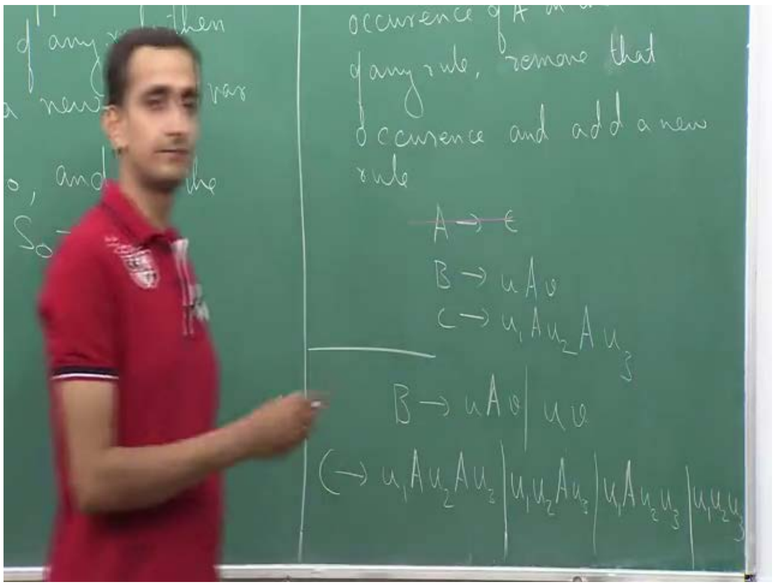

Converting a CFG G=(V,Σ,P,S) to a CFG in CNF
1. If S appears on the rhs of any rule then add a new st say S0, and a rule S0 →
2 (Removing ε-transitions)
A→ε
Remove A→ε and for every occurrence of A on the rhs of any rule, remove that occurrence and add a new rule
~~A→ε~~
B→uAv
C→u1Au2Au3
| uv
C→

The second is removing epsilon transitions. So, suppose if we have a rule of the form a going to epsilon, so suppose in the grammar we have some ‘A’ going to epsilon then we remove that rule a going to epsilon. And for every occurrence of ‘A’ on the right hand side of a rule, we add a new rule by removing that. So, say that the grammar contains a rule a going to epsilon then remove a going to epsilon; and for every occurrence of ‘A’ on the right hand side of any rule remove that occurrence and add a new rule. So, in addition let us say that by, yeah so we do this.

So, now let us look at an example to motivate this. So, let us say that we have something like A going to epsilon, and B going to u A v, and we have C going to let say u 1 A u 2 A u 3. So, let us say that we have a grammar where these three rules are present. So, then what we do is we first remove A going to epsilon. So, we remove A going to epsilon. And what we do is we have now B going to u A v.

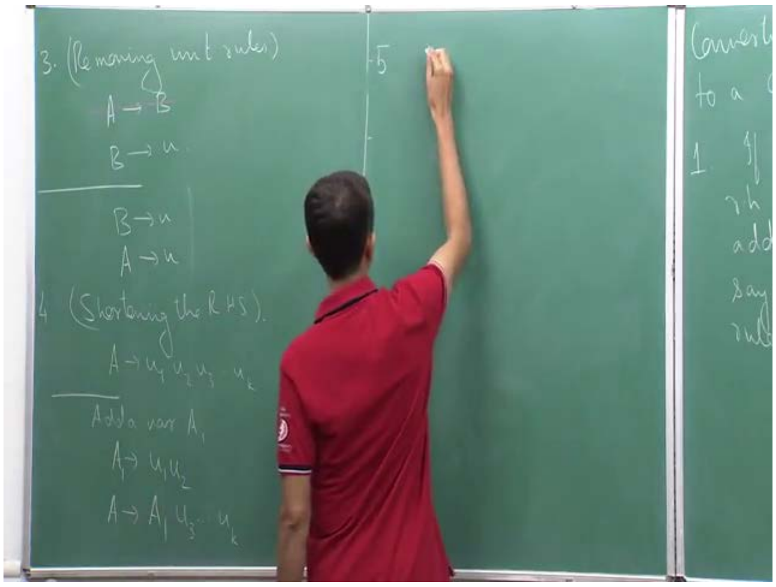
We add another rule by removing this occurrence of A, we do not remove the earlier occurrence, so we keep the earlier occurrence. So, we add another rule B going to u v. So, earlier we had B going to u A v now in addition we have B going to u v. Now what about C. Observe that in this rule there are multiple occurrences of A on the right hand side; so when we have multiple occurrences of A, we will add a new rule for each such occurrences.

(Refer Slide Time: 12:34)



So, firstly, we have the rule c going to u 1 A u 2 A u 3 some space. So, first we have C going to u 3. Now addition to this, I remove the first occurrence of A, and I get the rule u 1 u 2 A u 3. Now I remove the second occurrence of A is to get the rule u 1 A u 2 u 3. Now I have these two rules again which have a single occurrence of A. So, I remove that occurrence of A from these two rules to get a third rule which is just u 1 u 2 u 3. So, these are the three new rules that I have added from this old rule. So, for every occurrence, I remove that occurrence to get a new rule. So, this will be the new grammar by removing it.

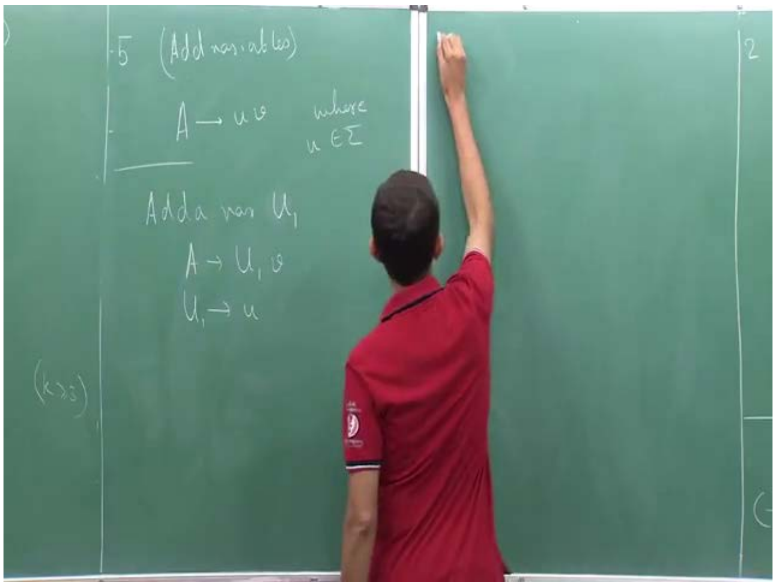
(Refer Slide Time: 13:56)



So, the third type is removing unit rules. So, if we have something of the form A going to B, what we do is and let us say we have something like B going to u, so where u is some string. We remove A going to B, so we remove this. And now in addition to B going to u I add a new rule. So, I have B going to u, I also add A going to u. So, we do not want units. So, these types of rules are known as unit rules.

The fourth is shortening the right hand side. So, suppose I have something like A going to u 1 u 2 u 3 up to sum u k where let us say that k is greater than or equal to 3. So, if I have a long right hand side, which is 3 or more, what I do is that I just shorten it in steps. So, first I add a variable let say A 1 and I add the rule A going to so basically A 1 let us say is going to u 1 u 2. So, A is going to u 1 u 2. And now I add the rule capital A going to A 1 followed by u 3 up to u k. Now note that this rule has only two symbols on the right hand side, and this rule has k minus 1 symbol on the right hand side. So, I have reduced it by one by adding one extra variable and I keep on doing this until and unless k becomes equal to 2.

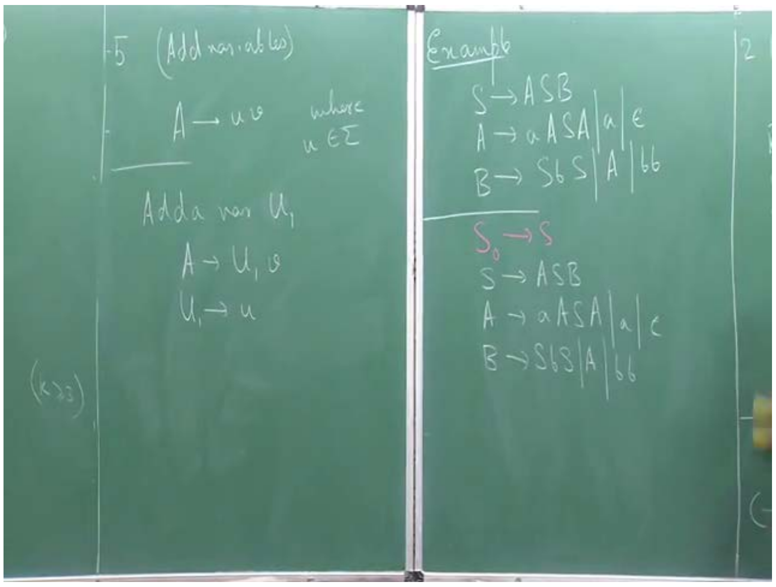
(Refer Slide Time: 16:46)



And the last step of the algorithm is to make sure that there are only so adding necessary variables. For example, let us say that I have a rule of the form A going to some u v, where let us say without loss of generality u is a terminal symbol. So, u is not a variable. What I do is that I add a variable, let us say capital U 1, and I add the rules A going to U 1 v and U 1 going to small u.

And if v is also a terminal, I again do the same thing; I add a new variable and I have A going to U 1 followed by that new variable and that new variable goes to v. Because recall that in the end, I have to ensure that if I have a production rule with two symbols on the right hand side, it must be a concatenation of variables. And if I have one symbol on the right hand side, it must be a terminal. So, this is why we have to ensure this.

(Refer Slide Time: 18:40)



5 (Add variables)

$A \rightarrow uv$ where $u \in \Sigma$

Add a var U_1

$A \rightarrow U_1 v$

$U_1 \rightarrow u$

$(k \geq 3)$

Example

$S \rightarrow ASB$

$A \rightarrow aASA \mid a \mid \epsilon$

$B \rightarrow SbS \mid A \mid bb$

$S_0 \rightarrow S$

$S \rightarrow ASB$

$A \rightarrow aASA \mid a \mid \epsilon$

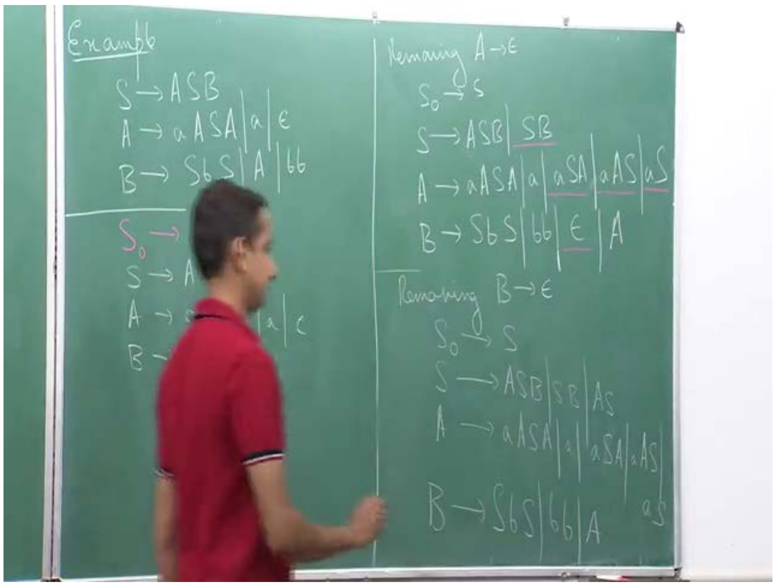
$B \rightarrow SbS \mid A \mid bb$

Let us look at an example to understand this more clearly. So, basically this is the algorithm. Now let us look at an example. So, let us say we start with the context free grammar S going to A S B. A going to small a capital A S capital A or small a or epsilon. And capital B going to S small b capital S or capital A or small bb. So, we have a grammar which has three variables capital S, capital A and capital B and it has two terminals small a and small b.

How do I construct a Chomsky normal form grammar out of this? So, first what we will do is that observe that I have S on the right hand side of several rules, so I have to get rid of that. So, I will add a production rule S naught is going to S. And now I just have the other rules as they are. So, I will have S going to A S B; I have A going to a A S A or a or epsilon; and I have B going to S b S or A or bb. So, this is our new start variable and this does not occur on the right hand side of any rule.

Our next job is to remove epsilon transitions. So, I have an epsilon transition from a going to epsilon, and I have to remove this. So, then we have to find out every rule such that a occurs on the right hand side of that rule. I have one rule over here, I have one rule over here, and I have another rule over here So, if there is a rule of this form where a occurs on the right hand side of a unit rule I will add. So, if I remove A, basically I will get B going to epsilon. So, I will add B going to epsilon unless I have removed it earlier. So, please keep this in mind. So, let us go through it.

(Refer Slide Time: 21:26)


Example
$S \to ASB$
$A \to aASA \mid a \mid \epsilon$
$B \to SbS \mid A \mid bb$
$S_0 \to$
$S \to A$
$A \to$
$B \to$
Removing $A \to \epsilon$
$S_0 \to S$
$S \to ASB \mid SB$
$A \to aASA \mid a \mid aSA \mid aAS \mid aS$
$B \to SbS \mid bb \mid \epsilon \mid A$
Removing $B \to \epsilon$
$S_0 \to S$
$S \to ASB \mid SB \mid AS$
$A \to aASA \mid a \mid aSA \mid aAS \mid aS$
$B \to SbS \mid bb \mid A$

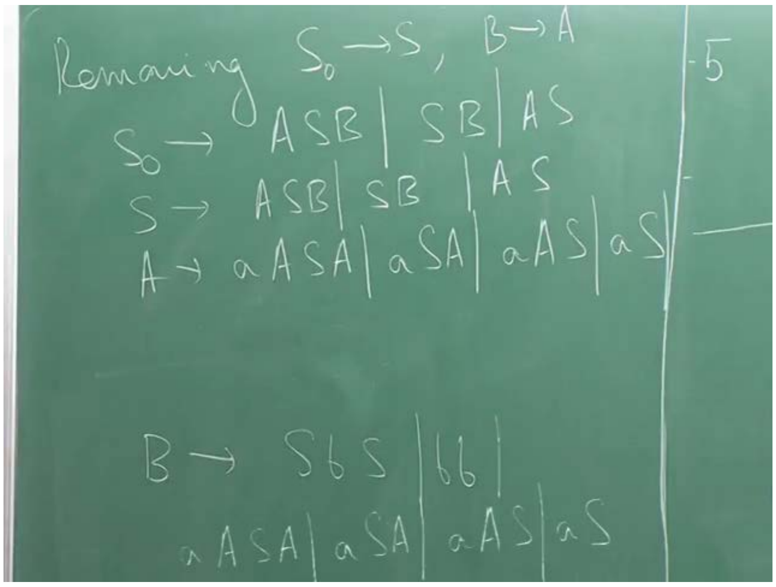
So, first I remove A going to epsilon, I will have S naught going to S, then I will have S going to A S B; I remove A, so I get S B also. Then I have A going to a A S A or a, I have removed epsilon, but I have to add the new rules. So, for this I will get now a S A or a capital A S, or if I remove both the occurrences I get a and capital S. Similarly, for B, I will have S b S or bb. And now if I remove this occurrence of A, I get B going to epsilon; and I add B going to epsilon because I have not removed it so far. So, let me underline the new rules that I have added. So, this is added now, this is added, this is added, this is added, and this gets added.

Now, in the third step I remove B going to epsilon, because this also has to be removed now. So, what we get is S naught going to S, S going to. So, I have a rule A S B where b occurs on the right hand side. So, I will have S going to A S B followed by S B; I remove this occurrence, I get A S; and also I remove this occurrence, so I get just S going to S which is redundant, because S going to S does not make any sense. So, we do not care about it, so that is all.

Now, in the next step, I have A going to a A S A or a or a S A or a capital A S or we have a capital S. And finally, we have capital B going to S small b S or small bb. So, this gets removed. So, now that we have removed all the epsilon rules, I have to remove all the unit rules. So, what are the unit rules, I have a unit rule S naught going to S, I have a unit rule, in fact, that is the only unit rule that I have currently.

So, I have S naught going to S, which I will remove, so I actually missed this one out here. So, when I was removing B going to A, so of course, I keep this B going to A. So, of course, this will be there, and I add the new rule B going to epsilon and the same thing happens here also. So, when I remove a going to epsilon, this is what should be there.

(Refer Slide Time: 26:12)



Removing $S_0 \to S$, $B \to A$

$S_0 \to ASB \mid SB \mid AS$

$S \to ASB \mid SB \mid AS$

$A \to aASA \mid aSA \mid aAS \mid aS$

$B \to SbS \mid bb \mid$

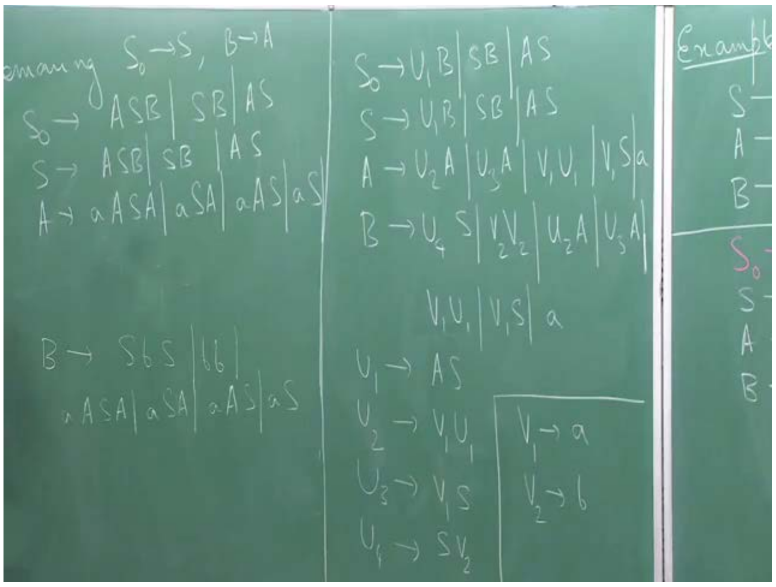
$aASA \mid aSA \mid aAS \mid aS$

-5

Let us removing S naught going to S, so what we get is we remove S naught going to S and we get S. So, every occurrence, if I remove S naught going to S, for every occurrence of S that I have on the right hand side, I will be adding the following rules. So, now, we have to remove unit rules. So, we remove the first unit rule S naught going to S. So, what we get is wherever S occurs, I add this new occurrence of S naught. So, I will have S naught going to A S b or S B. And of course, then I have S going to A S B or S B A going to a A S A or a S A or a A S or a S.

And finally, we have B going to S b S or bb or, so we will also have A S over here, or we have A. And now we also remove the unit rule B going to A and. So, finally, so what we get is if I remove B going to A as well whenever I have A going to something, I add the rules of B. So, if I just write it in one step, I will have a S, and I will have B going to S b S or bb, a A S a or a S A or a A S or a s. So, I have removed both S naught going to S and B going to A. So, now that I have a rule which does not have any epsilon rules or unit rules, I will shorten all the rules on the right hand side, and I will add appropriate variables to complete them.

(Refer Slide Time: 29:40)



$S_0 \to S,\ B \to A$
$S_0 \to ASB \mid SB \mid AS$
$S \to ASB \mid SB \mid AS$
$A \to aASA \mid aSA \mid aAS \mid aS$
$B \to SbS \mid bb \mid$
$aASA \mid aSA \mid aAS \mid aS$
$S_0 \to U_1B \mid SB \mid AS$
$S \to U_1B \mid SB \mid AS$
$A \to U_2A \mid U_3A \mid V_1U_1 \mid V_1S \mid a$
$B \to U_4S \mid V_2V_2 \mid U_2A \mid U_3A \mid$
$V_1U_1 \mid V_1S \mid a$
$U_1 \to AS$
$U_2 \to V_1U_1$
$U_3 \to V_1S$
$U_4 \to SV_2$
$V_1 \to a$
$V_2 \to b$

So, let me write down the final grammar. The final grammar will be something like this. So, I have S naught going to I have a new variable B S B or A S; S going to once again U 1 B S B A S; A going to U 2 A U 3 A V 1 U 1 V 1 S or small a; B going to U 4 S V 2 V 2 or U 2 A or U 3 A or V 1 U 1 or V 1 S or a. Now I have this new rules for U 1, U 1 going to A S; U 2 going to V 1 U 1; U 3 going to V 1 S; U 4 going to S V 2; and V 1 going to a, and V 2 going to b. So, this I get by shortening.

So, I add appropriately new variables and I get this. So, this shows that any grammar, any context free grammar can be converted to a grammar in Chomsky normal form. So, I will stop here.

Thank you.