Theory of Computation

Prof.Kamala Krithivasan
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

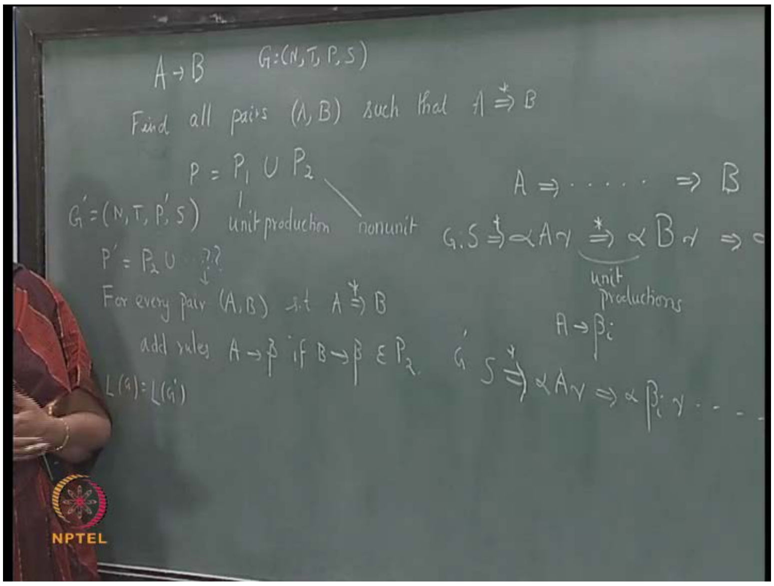
Module No.# 01

Lecture No. # 06

Removal of Unit Productions, Chomsky Normal Form for CFG

So, in the lastclass we have seen how to remove the useless symbols from a context free grammar,we have also seen how to remove the epsilon productions from a context free grammar.But,if the language contains epsilon you must have a rule of the form S goes to epsilonand you must make sure that S this not appear on the right hand side of any production.The next result we will see is that we can remove the unit productions also what is a unit production?

(Refer Slide Time: 00:43)



A unit production is of the formAgoes to B. A non terminal going intoanother non terminalset aproduction is called aunit production.

So, you may want to get rid of these rules,say mentioned.If you introduce too many of this unit production in acompiler if youwant to represent aprogramming language by agrammar and use acompiler for thatthen if you have to many of this unit production. The compile time will be moreso, you want to avoid these unit productions.

So, how to remove these unit productions? First of all you must see from which non terminalsfindall pairs A B.Such that A,fromAyou can go to B need not be in onestep but, in many steps.That is your starting with thegrammar G is equal to N T P S and in this grammar you want toremove the unit production without affecting the language generative.

Now,you split P intotwo sets P 1 union P 2,where P 1 isaset ofunit productionsand P 2 is the other set non unit.Now, you construct an equivalent grammar G dash is equal toN T P dash same non terminals same terminals. But,the production rules are different.What is P dash? P dash consists of all P 2,all P 1 will be removed but, insteadyou will add some more rules and what are these rules you are going to add.

Now, for every pair A B,such thatfromAyou can go to B by unit productions, it may not be in onestep. IfAgoes to B is A unit production you can go in onestepor if you have A goes to c, c goes to B then you will go to B from A in twosteps.So, for every pair such that A a goes to B add rulesAgoes to betaif B goes to betabelongs to P 2.If you havearule B goes to beta and P 2 then you add the ruleAgoes tobeta to this.

So, P dash consists of P 2 plus all such rules added.Then you can show that L G is equal to L G dash,againaformal proof on the number of steps and socan be given in inductionon the number of steps. But,informally you can see like this, you can see that suppose,you haveastepfrom which you derive B.And inaderivation from S you may haveasentential form say alphaAgamma,somewhere it get gamma.Then after some stepsmaking use of unit productions onlyyou get alpha B gammabecause, from A you are able to derive B.

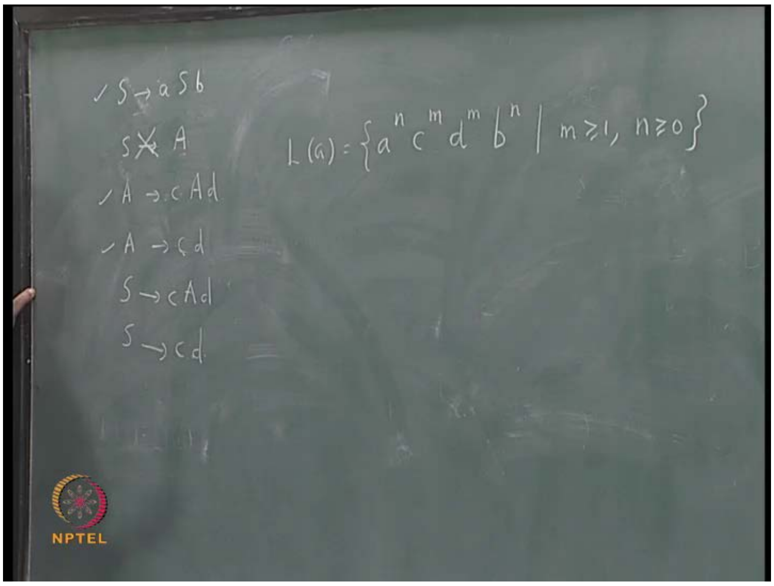
And these steps are using only unit productions,then after some stepyou may be using some rule B goes to beta i.So, after in the next step may be alpha beta i gammathenetcetera. But, in the grammar G dash which we have constructedyou have added the rule A goes tobeta i,you have added the rule this.So, instead of going through all these steps startthis is happening in G and in G dash what will happen is you are getting alphaAgamma.Andinstead of applying these unit productions and then applying B goes to beta i you straight away apply A goes to beta i.So, alphabeta i gamma you get in onestep,then the derivation in G dash can proceed.

So, the effect is the same instead of applying A derives B and then B derives beta i straight away you are using the rule A goes to beta i and deriving this inonestep.So, by removing the unit productionsand adding the rules of the formAgoes to beta, the resultant language is not changed, the language generated is the same.So, it is advantageous to do this rather than useAunit productionsbut, onething you must be careful, the number of rules here will be enormously increased depends on.Suppose, I havesuch pairs A B 5 6 or7 pairs like that. The number of rules is going to increase very much but, if Ihave just onepair you can just remove it and so on.

So,the number of rules in P dash will be much more than the number of rules in P and when you useaparser to reduceastringyou at every point you may be able, you may have to check, which rule is can be used for reduction.Then if the number of rules is too much every rule you must try.

So, that look up procedure will be more it may take more time.So, you have to see thatthere isabalance that in while removing the unit production you are not adding too manyof other rules. But,any way this result tells you that you can rid of unit productions it is not necessary to use the unit productions.Let us take an example.

(Refer Slide Time: 08:34)



A simple examplelet me take (No audio from 08:25 to 28:31) S goes to a S b,S goes to A,A goes to c A d, A goes to c d.If I havethisgrammar,what is the language generated?The language generatedisfirst you will use this rule many times then from S you go to A, then use this to generate equal number of c’s and d’s.

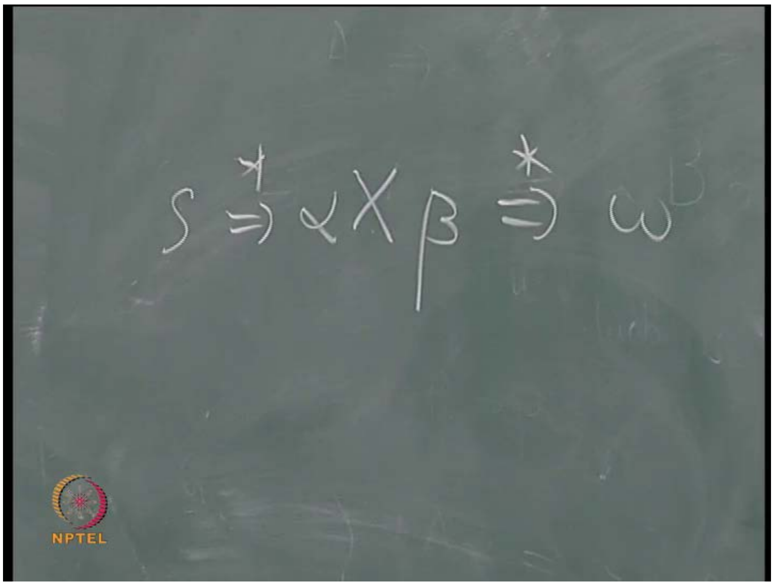
So, it will be a power n,c power m,d power m,b power n,m greater than or equal to 1 but, n can be 0 also.This willbe the language generated and there is a unit production.Now, we want to get rid of this unit production.So, by this procedure you split it into P 1 and P 2, P 2 will consist of thisthreerules, P 1 will consist of this.

So, remove this you have to remove this but, P 2 you have to keepas it is.So, these threerules you keep,then you have a pair S A.Then whenever there is a pair S A,the right side hand you replace with the right hand side for every rule with A on the left hand side.So, what are the rules you will be adding now, the rules you will be adding will be S goes toc A d, S goes toc d.So, instead of applying S goes to A, then A goes to c A d you can straight away apply this rule,instead of applying S goes to A,A goes to c d you can straight away use this rule.

Now, look atthe grammar without this rule but, addingthesetworules you will see that any number of equal number a’s and b’s can be generatedusing this rule.Then you can go from S to thisor S to this, you see that at least 1 c and 1 d must be generated,use this many times then use this 1 c d will be generated,use this many times as many times you want.And then use this once and then thisyou will get c square d square.If you want to get more use this many times for a power n b power n.

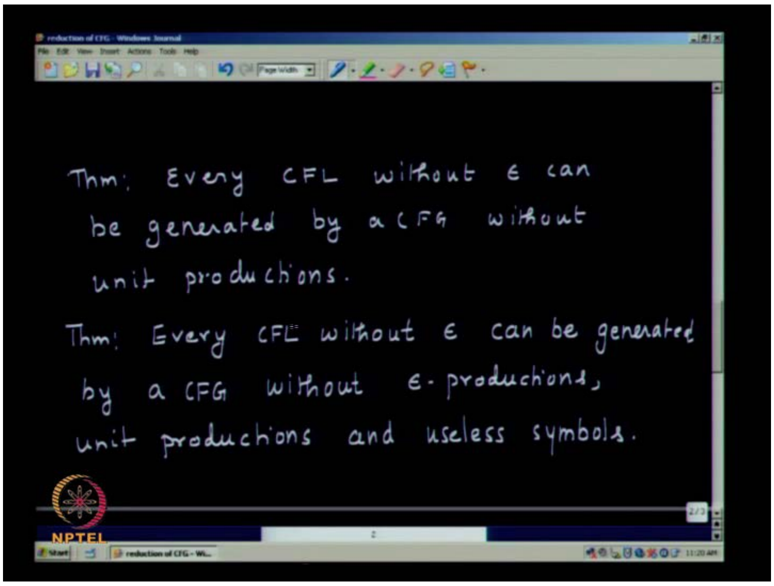
Then c cube d cube if you want use this once, use this once,use this once then any number of c power m, b power m you cangenerate d power m you can generate with propercombination of thesethreerules. Just 1 c d alone you want means you use this rule after applying this ruleyou need not have to apply this rule at all because, n can be 0 also just c d also belongs to the language.You can start with S and derive c d there n can take the value 0 1 2 etcetera m takes a value 1 2 3 that should be at least 1 c and 1 d.So, this is the way you get rid of the unit productions.

(Refer Slide Time: 12:50)



So, we have seen how to remove the useless symbols that is form every symbol should occur in a derivation like,every symbolXmust occur in a derivation like this,from S you should beable to reach x and then from X you should be able derive a terminal string.So, for that we have seen that how to applytwoLemmasLemma 1 and Lemma 2 and you have to apply them in that particular order.The first make sure that forfrom every non terminal a terminal string is derived with this againonemake sure that every symbol is reachable from S.

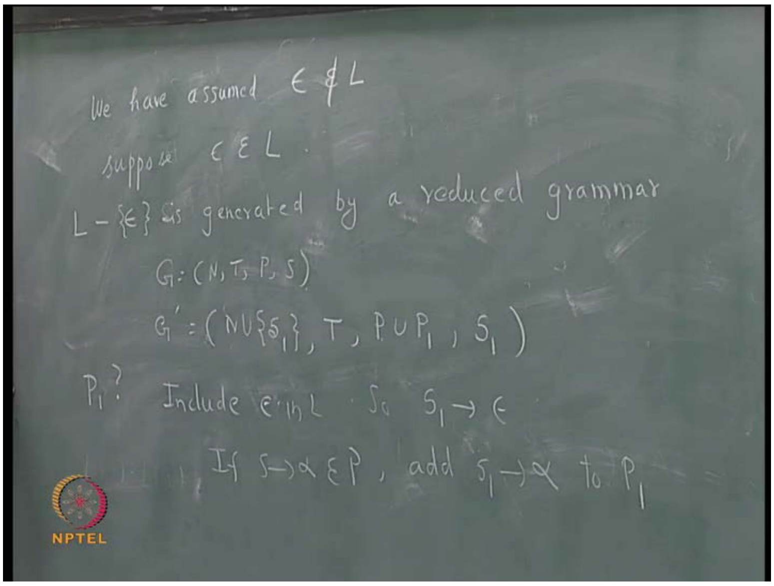
(Refer Slide Time: 13:32)



So, now, you have this result every CF L without epsilon can be generated by a C F G without epsilon productions, unit productions and useless symbols.So, in which order you go aboutbecause,we are without loss of generality,we are assuming that the language S 0 contain epsilon.So, the first step will be remove the epsilon productions because, when you remove the epsilon productions you mayintroduce unit productions.

So, first remove the epsilon productions after removing them remove the unit productions, then after doing thisremove the useless symbols use Lemma 1 and Lemma 2 in that order.So, ultimately you will end up with a grammar which does not have epsilon productions, unit productions and useless symbols.Such a grammar is called a reduced grammar. (No audio from 14:30 to 14:39)So, we are interested in finding out the reduce to grammar for a language.Now,oneassumption we have made is thatit hasit does not have epsilon.

(Refer Slide Time: 15:07)



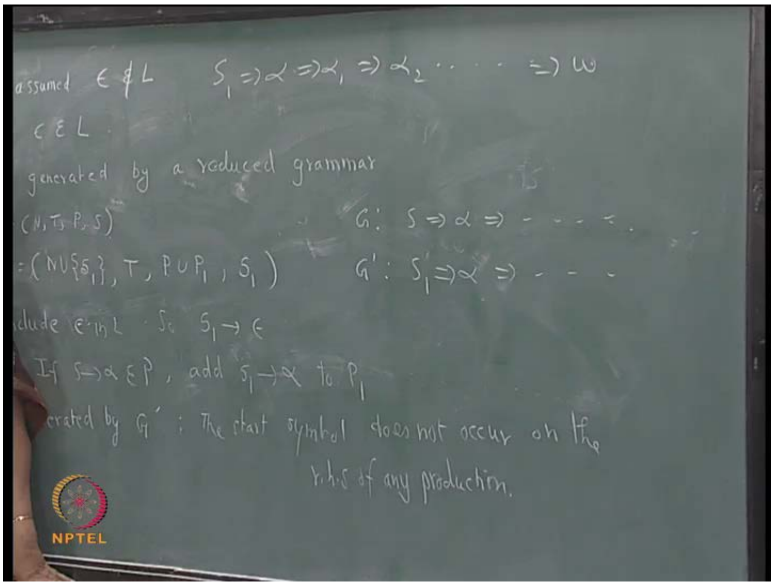
So,we have assumedepsilon does not belong to Lthe languagewith which we started.Suppose,what do we do supposeepsilon belongs to L,then how do we go about it.In this case L minus epsiloncan be generatedby a reduced grammar (No audio from 15:42 to 15:49) L minus epsilon will be reduced by a grammar.

Now, we have to include epsilonfor that what we do?So, this is suppose, this is generate by G isequal toN T P S then you have G dash,where you addonemore symbol S 1 same terminalsP union say P 1 I will put comma S the new symbol will be the startsymbol you addonemore symbol S 1 till set of non terminals and make it the start symbol.And all these productions will be there but, you may add a few more productions and what is the set of productions P 1 you are going to add.First of all you want to include inepsilonis not ityou want to include epsilon in the language.

So,you have a rule S 1 goes to epsilon.So, if you want to derive epsilon you will use this rule alone S 1 goes to epsilon and then derive epsilonno other steps involved.Other rules will not have epsilon on the right hand side. But,starting from S 1 you also have if S goes to alpha belongs to P add S 1 goes to alphato P 1.If S goes to alpha belongs to P the earlier start symbol going into alpha,then you also add S 1 goes to alpha.

So, P 1 will consist of rules of the form S 1 goes to epsilon and S 1 goes to alpha, where S goes to alpha belongs to P.This makes sure that the start symbol does not occur on the right hand side.Now,alpha cannot contain alpha belongs to P the earlier grammar the earlier grammar didnot have S 1.So, alpha cannot contain S 1.So, S 1 will not occur on the right hand side of any productionand if you want to derive epsilon you was you have to use this rule and this rule only and you will able to drive epsilon.

(Refer Slide Time: 18:54)



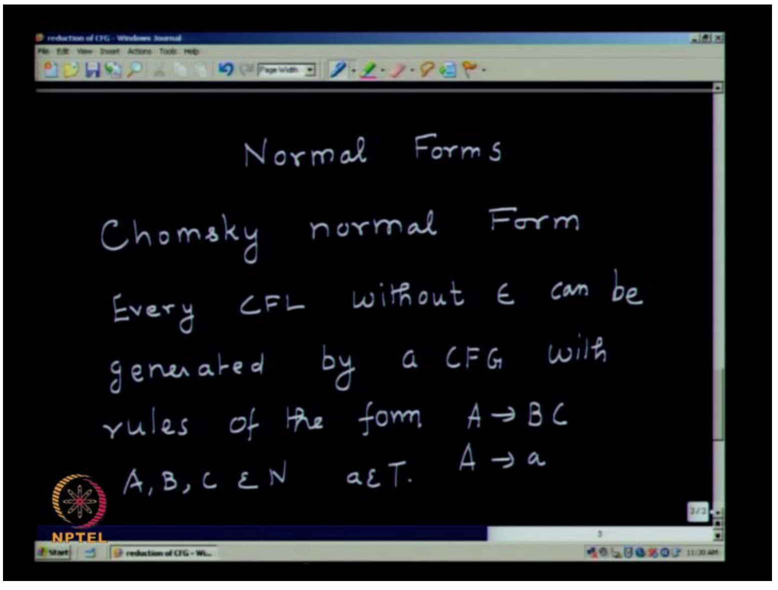
Now, if you have derivation in G of the form S goes to some alpha then somethingetcetera.In G dashyou will have S dash goes to alphaI am sorry S 1 goes to alphasame derivation you can have the first step alone will be changed.So, whenever a string is derivable here it is derivable here and whenever a string is derivable here, it is derivable here, apart from that you are also deriving epsilon.

So, L minus epsilon is generate by a reduced grammar L is generated by G dashand in G dash you have the property that,the start symboldoes not occuron ther h S ofany production. (No audio from 20:06 to 20:14)The reason for this is if you have a derivation S 1 goes tothissome alpha,alpha 1 or alpha 2 whatever it is.If we have a derivationthe successive sentential forms will be non decreasing in length.

Length of alpha 1 will be more or equal to alpha length of alpha 2 will be more or equal to alpha and so on.It will not decrease but, if you have S 1 on the right hand side the start symbol.Because, of this rule any time you will be able to apply this rule,the success sentential form can reduce in length youwant to avoid that,in anyway derivation you want to make sure that the successive sentential forms are non decreasing in length.And that is why you want to avoid having S 1 on the right hand side of any production since you are adding this rule S 1 goes to epsilon that is a reason.

So, whenever we have this problem with epsilon, epsilon has slight whenever you want to have the empty word there will, in the proofs and all there will be a slight problem.So, you want to avoidwriting a very lengthy proof.So, you make sure that the grammar generator has this format.So, the next thing is we shall studytwo normal forms.

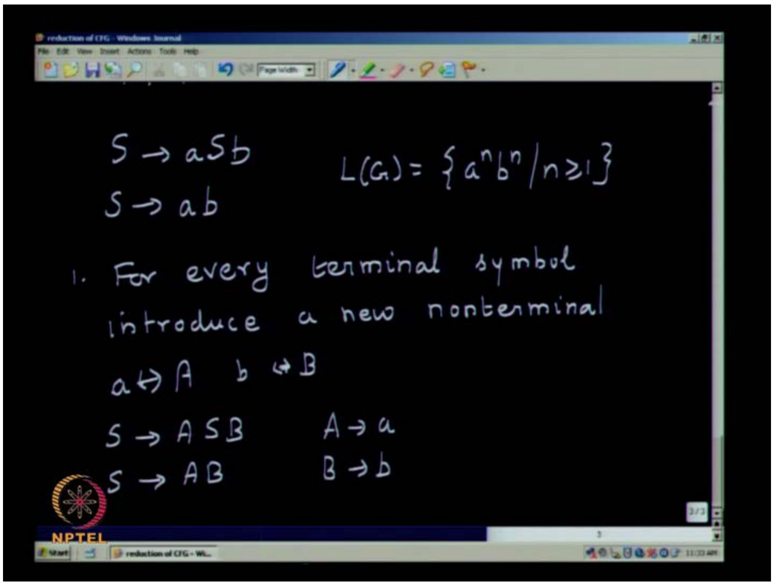
(Refer Slide Time: 21:54)



What are the normal forms?One is called this Chomsky normal form. (No audio from 21:59 to 22:13)It says that every C F L without epsiloncan be (No audio from 22:25 to 22:33)generatedby a C F G withrulesof the formA goes to B C or A goes to a,where A B C are non terminals and a isa terminal. (No audio from 23:08 to 23:22)So, without loss of generality,we can assume that the language, S not contain epsilon and we know how to accommodate for epsilon now.So, if a C F L without epsilon, then it can be generated by thegrammar with rules or of the form A goes to B C and A goes to a. That is on the left hand side as usual for any C F G you how only a single non terminalon the right hand sideyou havetwonon terminals or a single terminal on the right hand side. You can just havetwonon terminals or asingle terminal.

So, every C F G you can bring in to this form this is called Chomsky normal form.How can you bring it? Let us, take an exampleby considering an example it is much more easy.

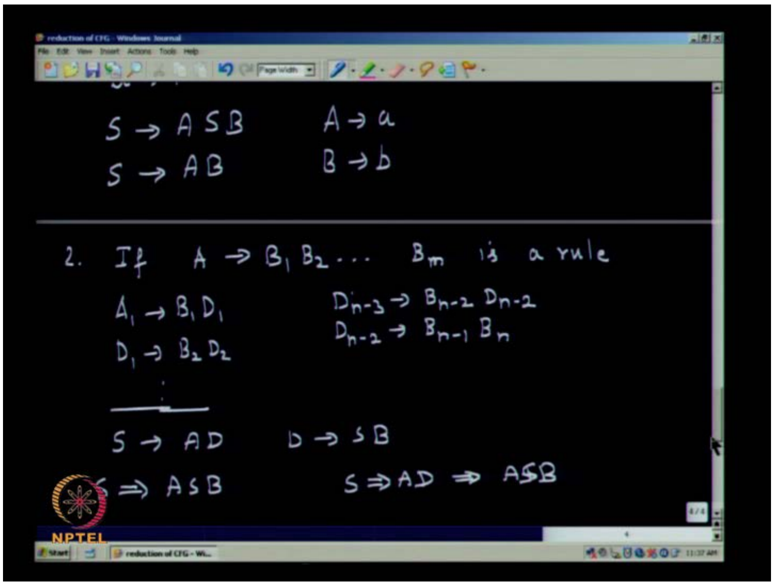
(Refer Slide Time: 24:22)



Let us, takethegrammar. (No audio from 24:23 to 24:34)This you know that the language generatedisa power n b power nn greater than or equal to 1.Now, we want to bring this to Chomskynormal form.How do we go about doing this, first step will befor every terminal symbolintroducea new non terminal.So, for small a introduce A non terminalit shouldnot.For small a you use Aand small b usecapital B.Now, the rules will become in this example the rules will become S goes to A S B, S goes to AB,A goes to a,B goes to b (No audio from 26:11 to 26:22).

Now, we can see that,this rule is in Chomsky normal form, this rule is also in Chomsky normal form, this rule is also in Chomsky normal form. But,this is not in a Chomsky normal form.So, you have to do something about it.

(Refer Slide Time: 26:41)



So, the second step is if A goes to B 1 B 2 B mis a rulenote that now you have made all the symbols on the right hand side as non terminals.

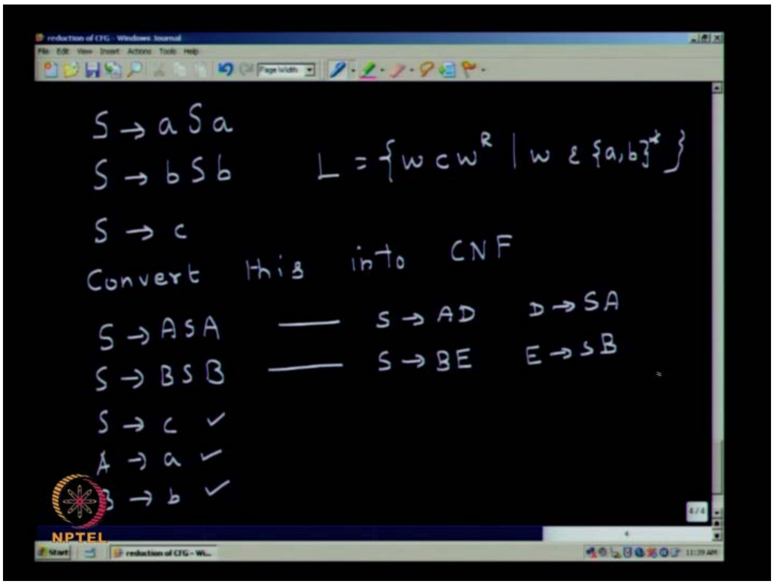
So, B 1 B 2 B m are all non terminals now.So, this rule you can split like thisA 1 goes to B 1 D 1 D 1 goes to B 2 D 2 like that upto D n minus 2 goes to B n minus 1 B n.The previousonewill be D n minus 3 goes to B n minus 2 D n minus 2.So, at the end of the first step after we have replaced everyterminal with a non terminal,the rules will be either in this form Chomsky normal form,terminal rules or rules will be of the form on the left hand side you have anon terminal.

And the right hand side you have a string of non terminals.So, when you have this when you have a string of non terminals on the right hand sideyou would not have onlytwoof them.So, you have to split this rule by introducing new non terminals D 1 D 2 D 3 etcetera up D n minus 2 and split this rule. But,you must be careful that if you havetwosuch rules when you want to split this rules you are introducing the non terminals D 1 D2 etcetera.Another rule is there then you must introduce some other non terminal E 1 E 2 etcetera,you should not use the same non terminals that because they will mix up and then create problem.

So,in this example which we have considered thesetwoare in Chomskynormal form, this is no problem it is in Chomskynormal form.The first rule alone is in not in Chomsky normal form.So, you introduce a new non terminal say D, S goes to Dand then D goes to S B you introduce a new non terminaland have S goes to AD and D goes to S B.

So, instead of applying like thisthe same result will be achieved intwosteps by applying AD first and thenD goes to S B when you apply it will be A S B.So, without any problem,any grammar you can convert into Chomsky normal form.Let us takeonemore example.

(Refer Slide Time: 30:14)

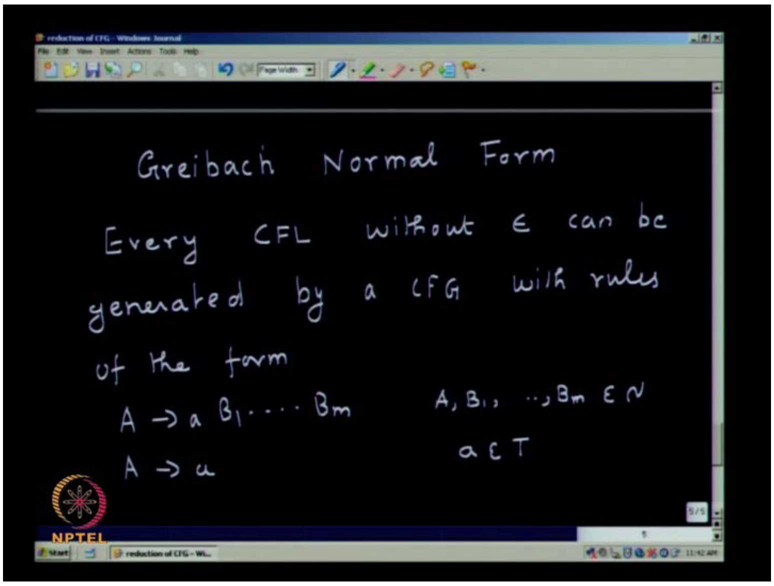


S goes to a S a, S goes tobSb,S goes to c. What is thelanguage generated? The language generated consists of strings of the form w c w R.Where w belongs toabstart any string of a’s and b’s.Now, convert this into Chomsky normal form,convert this grammarintoshortened form is CNF,Chomsky normal form.The first step is introduce a new non terminal for every terminal symbol, if you do that you will get S goes toASA, S goes to B S B,S goes to c,A goes to a B goes tob.

These are in Chomsky normal form no problem.So, the othertworules you have to convert into Chomsky normal form.So, when you want to convert this, what do you do you introduce a new non terminal. And instead of this you will have S goes to say AD introduce a new non terminal Dand have it as S goes to AD,D goes to S A.

Now, when you want to convert this rule, introduce another non terminal E not the same you should not use the same D, S goes to B E,E goes to S B.So, the effect of applying this rule is achieved intwostepsby applying this first and then this.The effect of applying this rule is achieved by applying this and this.Now, all thesefourrules are in Chomsky normal form.So, every C F G can be converted into Chomsky normal form. (No audio from 32:19 to 32:31)

(Refer Slide Time: 32:39)

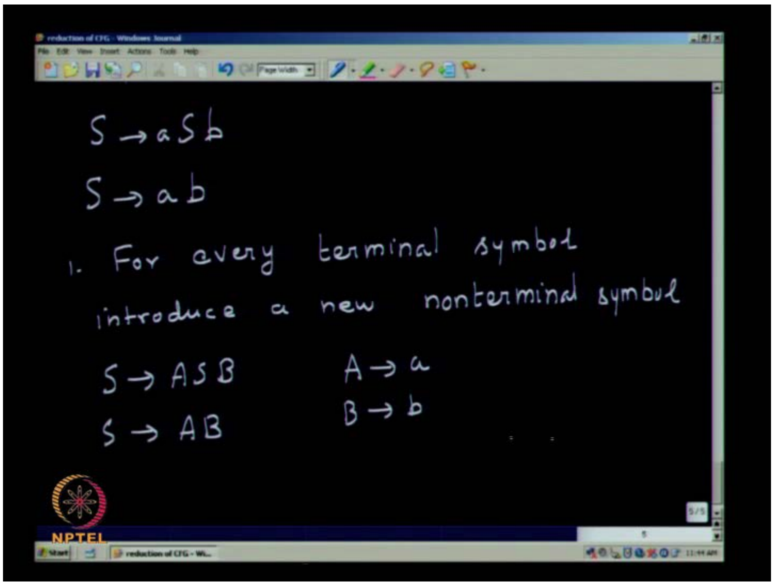


There is another normal formedcalled Greibachnormal form.There are several normal forms for C F G but, these are thetwowhich are mainly used.The Chomsky normal form is very useful in proving some results about context free grammars.Greibach normal form is useful for provingthe equivalence within push down automata and context free grammarsand also sometimes in parsing purposes if the grammar is in Greibach normal form it is easy to parse.

What is Greibach normal form? EveryCF L withoutepsiloncan begeneratedbya C F G with rulesof the formA goes to a B 1 B 2 B m,A goes to a.Where A B 1 etceteraare non terminals and a is a terminal.That is on the left hand sideyou have a single non terminalon the right hand side you have a single terminalor a single terminal followed by a string of non terminals you may haveonenon terminal, twonon terminal, threenon terminals any number you can have.

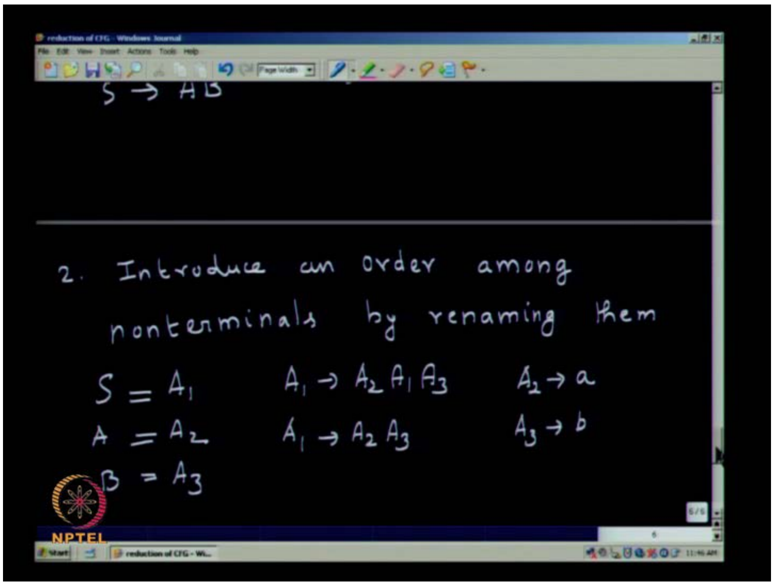
So, the rules are of the form suchthat isset on the left hand side you have a single non terminal, on the right hand side you can have a single terminalora single terminal followed by a string of non terminals.So, you can bring any C F G to this form.Let us see how we can do that.

(Refer Slide Time: 35:12)



Again take the sameSimple examples if S goes to a Sb,S goes to abthis we have already considered.The first step isfor everyterminal symbol;introducea new non terminal symbol.So, in letus, let me illustrate after every step, how for this example.There arefivesteps in the conversion but, thisexample may not use all thefivesteps.So, let me seehow we can do this.So, S goes to A S B, S goes toAB. Then you have A goes to athis is step is a same as the previousconversion to Chomsky normal form.Now, you find that because of this the language generator is not affected, the language generated is still going to be the samefirst you will generate capital a power n, b power n convert all this capital A’s to small a’s and capital B’s to small b’s.Now, thesetwoare already in Greibach normal form.

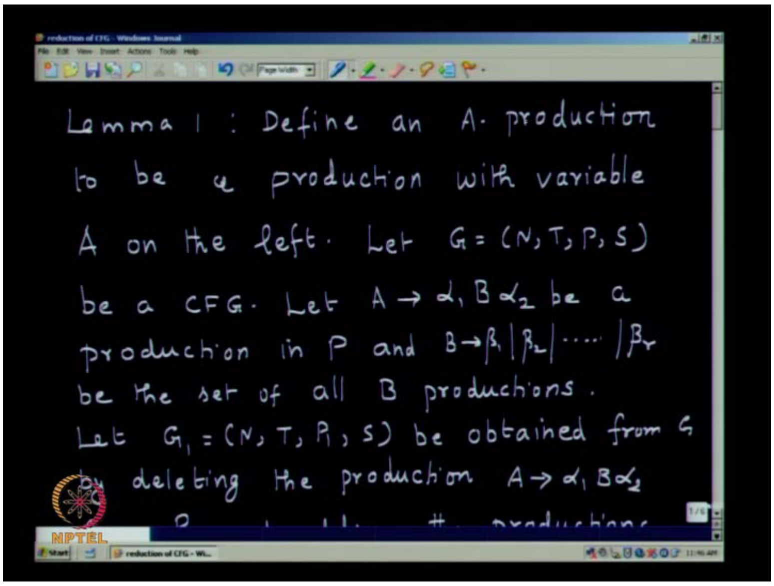
(Refer Slide Time: 37:21)



The second step herewill be (No audio from 37:07 to 37:18)introducean orderamongnon terminalsbyrenaming them. (No audio from 37:41 to 37:50)So, the grammar which we consider earlier (No audio from 37:55 to 38:01)you cancall them as A 1A 2 A3.So, make S as A 1, A as A 2,B as A 3 there are non terminals here.

So, make S (No audio from 38:20 to 38:28) S as A 1, A as A 2,B as A 3 you are renaming them. So,now the rules will become A 1 goes to A 2 A 1 A 3, A 1 goes to A 2 A 3, A 2 goesa,A 3 goes to b.So, you arehaving an order among the non terminals.This is also very simple it is justathe language generator will not be affected because just you are renaming the non terminals and calling them as A 1 A 2 A 3.

(Refer Slide Time: 39:33)

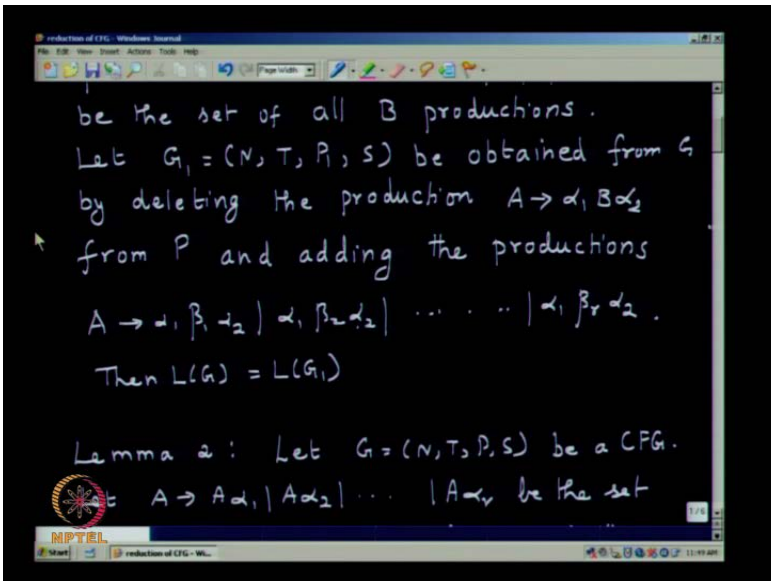


The third stepis involves the use oftwoLemmas,we will see what are thetwoLemmas. (No audio from 39:25 to 39:31)Lemma 1 is thisdefine an A production to beaproduction with variable A on the left, an A production isaproduction with variable A on the left.Let G is equal to N T P S beaC F G,then there is an A production A goes to alpha 1 b alpha 2 for some reason you want to get rid of this rule.

Now, when you want to get rid of this rule,what should you do beaproduction in P and with B on the left here,B isanon terminal with B on the left hand side you haveaset of production B goes to beta 1,B goes to beta 2,B goes to beta r. A set of productions you have with B on the left hand side,with B goes to beta 1 beta 2 beta r be the set of B productionsthen from G you construct G 1.Such that G 1 hasasame non terminal same terminals but, the productions have same.

Let G 1 be obtained from G by deleting the production alpha 1 B alpha 2 for some reason you want to get ridof it,we will see why we need thisLemma later.So, you delete this rule but, when you deletethis rule.

(Refer Slide Time: 41:12)

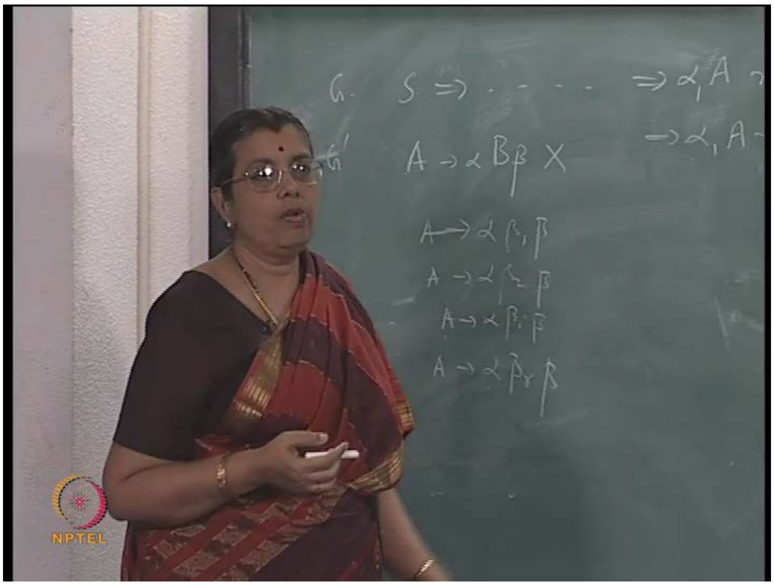


The effect must be obtained by some other rules.So,when you delete this rule what you do is you add the rules A goes to alpha 1 beta 1 alpha 2 alpha 1 beta 2 alpha 2 alpha 1 beta r.So, r rules you add,then the language generated does not change.So, inasense what happens is.

(No audio from 41:31 to 41:58)

In this grammar I haveaderivation.

(Refer Slide Time: 42:06)



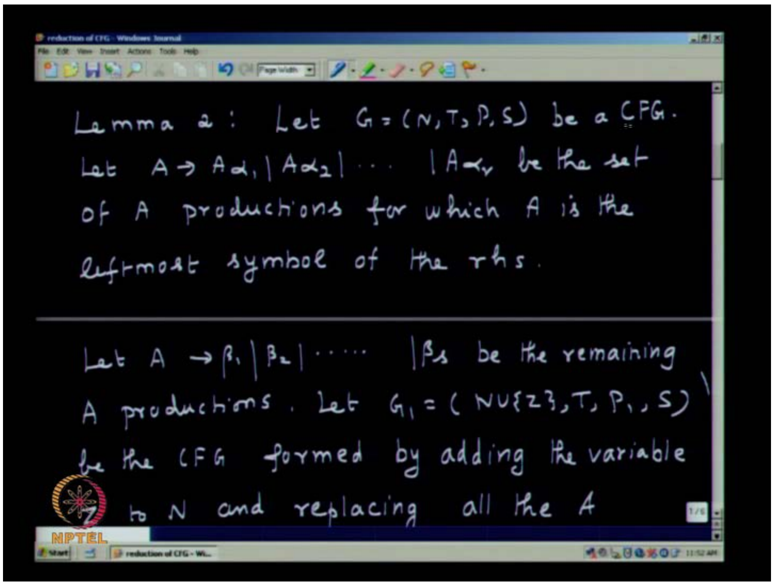
Somewhere I have alpha A betaorprobably alpha A gamma you can write.And the next step I use the rulealpha 1 I will putAgoes toalpha B betaI use this rule.

So,Agoes to alpha Bbeta I applyand the next gamma will be there. And the next step the beta will be replaced by some beta i, alpha 1 alphabeta ibeta gamma,then the derivation proceeds.Now, this ruleyou want to removenow, when you remove this you are addingthe rules of the form A goes toalpha beta 1 beta,A goes toalpha beta 2 beta and so on,a goes toalpha betar betasuch rules you are adding.

So, at this stageinstead of using this rule and then replacing B by beta i straight away you can use the rule A goes to alphabeta i betaand write it as alpha 1 alpha beta i betagamma.In the original grammar G you may have a derivation which a is replaced by alpha B beta first and then B replaced by beta i.By in the new grammar G dash you do not have this production A goes to alpha B beta.

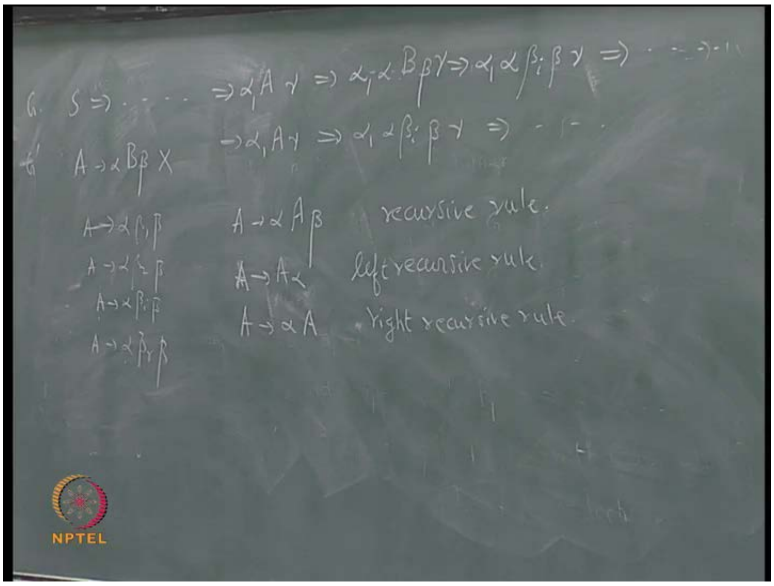
So, but, instead of you have a added these productionsso, instead of getting it intwosteps like this straight away you apply the rule A goes to alpha beta i beta and get this.Send the derivationwill proceedas before.So, the effect is the same so, the language generated will not beaffectivethis is when you want to get rid of a particular rule for some reason.Then there is another Lemma thisalso,we will be making usein stepsthreeand fourof that conversion to Greibach normal form.

(Refer Slide Time: 45:13)



So, let us see what is,let d is equal to n T P S be a C F G and then this is a context free grammar.Then let Agoes to A alpha 1,A alpha 2,Aalpha r be the set of a productions for which A is the left most symbol of the right hand side.Such rules are called left recursive rulesa rule of the form.

(Refer Slide Time: 45:48)



Sum A going intoA alpha A betais calledarecursive rule.A rule of the formAgoes to A alphathere A is the first symbolleft most symbolthis is calledaleft recursive rule.A rule of the formAgoes to alpha A where A is the right most symbolthat is calledaright recursive rule. (No audio from 46:23 to 46:32)In many cases you may want to avoid left recursion,when we learn about parsing,we will see that we have sometimes we will want to avoid left recursion. But,when you want to avoid left recursion you will be introducing right recursion but, that is ok.

So,that is what we are going to do here,we want to avoid the left recursive rules, we do not want haveleft recursion.Then let us see what thelet A goes to A alpha 1 A alpha 2 A alpha be the set of A productions for which A is the left most symbol of the right hand side.Then there are other rules A goes to beta 1 A goes beta 2 A goes to beta S these are remaining A productionsthey are not left recursive rules.

Then you introduce you want to replace all the A rules byaset like this you haveanew grammar,where you introduceanew non terminal Zand instead of having the set of A productions.So, the new grammar let G 1 be equal to N union Z, T P 1 S be the C F G form by adding the variable Zto Nand replacing all the A productions by the productions A goes to beta i, A goes to beta i Z, where I varies from 1 to S and Z goes to alpha i Z goes to alpha i Z,where I varies from 1 to r.

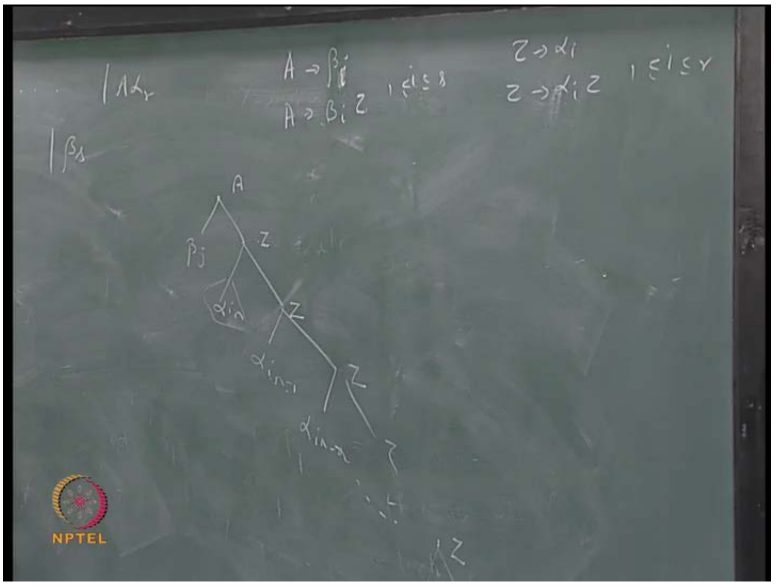
So, the earlier with A you have r leftrecursive rules and S other rules, which are not left recursive. But, even you want to replace you are introducinganew non terminal Zand you are having 2 S rules like this and 2 r rules like this.So, actually what you are trying to do is the r plus S rules you are replacingwith 2 r plus 2 S rules.Now,howdo we justify this?

(Refer Slide Time: 49:16)

So,we you are havingA goes to Aalpha 1 Aalpha 2 Aalpha r,r rules with which are left recursive.And Agoes tobeta 1 beta 2 beta S,s other rules.So, r plus S rules you are having.Now suppose, you are applying starting from sum Athenyou have Aalpha 1 and from this you are deriving something AI would putAalpha i 1.So, 1 of the rules I am applying herethen from this again another left recursive rule Iam applyingAalphai 2 and from this you may derive something.Then again from this you haveAalpha i 3 and from this you may drive something,you can proceed like this until you haveAalpha i n. And then here you use the ruleanon recursive rule betajfrom this again you can derive something.

So, from this Athe string derived isbeta j alpha i n,alpha i n minus 1,alpha i 1 the string derived is this.Now, the same string you must be able to derive with the replaced rules.

(Refer Slide Time: 51:18)



What are the rules which you have replaced A goes to beta i beta i j i, can put A goes toi Z,where i varies from 1 to S then Z goes to alpha i Z goes to alphai Z,where i varies from 1 to r.

Now, how do I get the same effect with these rules start from athen use the rulebeta jZ.Then you use the rulealpha i n Z you have such a rule. From this again you can drive somethingthen from this alpha i n minus 1 Z, then again use the rule alphai n minus 2Z and so on.Until you have Z goes toalpha i 2 Z,thenZ will go toalpha i 1.

So, the string generated here will bebeta j alpha i n alpha i n minus 1 etcetera up to alpha i 1 and from this alpha a’s again something else can be derived.So, the string generated is beta j alpha i n alpha i n minus 1 alpha i 1.Hereyou have the same effect by using this rule firsta goes to beta j is that so.beta j is generated,then you can a use a rule of the form Z goes to alpha i n Z then Z goes to alpha i n minus 1 is that and so on.

So,you will get the same stringbeta j alpha i n etcetera.So, the effect of using such rules you can obtain from this also, the language generated does not change it remains the same. But, instead of r plus S rulesnow, we have 2 r plus 2 S rules and what have we achieved by doing this, we have avoided left recursion left recursion is avoided.

But,what have we done for that,we have introducedright recursion Z occurring as a last symbolintroduces right recursion.So,thesetwoLemmas, we have to use again and again in the step 3 and 4 ofthe conversion to Greibach normal form there are 5 steps there.We shall consider theexamples and the procedure continue with that in the next class. These are not the onlytwonormal forms; there are other normal forms as well.But,these are the main normal forms which are used to especially Greibach normal form is essentially improving the equivalence between push down automataand context free grammars.