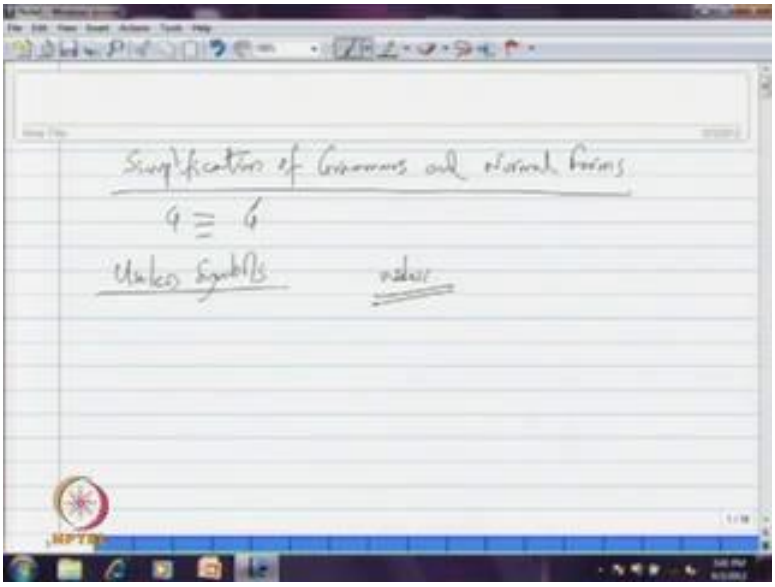
Formal Languages and Automata Theory
Prof. Diganta Goswami
Department of Computer and Engineering
Indian Institute of Technology, Guwahati

Module - 8
Simplification of CFGs
Lecture - 1
Simplification of CFG

In today's lecture, we will discuss Simplification of Context to Grammars and normal forms.

(Refer Slide Time: 00:26)



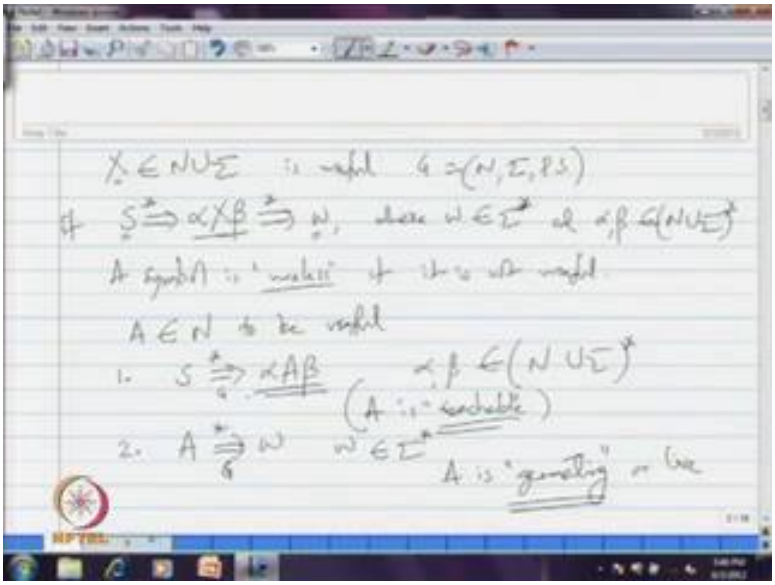
Simplification of grammars and normal forms, so given any CFG, suppose we have a CFG, would liked to find an equivalent CFG G dash, say G and G dash are equivalent. But here G dash should be simpler than G, the grammar on simpler than the original one in a sense that, it is less clumsy for example and easy to understand. Maybe it has less number of symbols or some awkward productions, which are there in G are remove from G dash, but still they are equivalent.

Now, if a grammar is given in simple form or simpler form, it helps us to bring many fraction results about languages, will consider simplification of grammars by removing useless symbols, epsilon production and unit production. So, first we consider useless

symbol, we will first define useless symbol, what is mean by useless symbols and then, see how can I remove useless symbols from grammar creating an equivalent grammar.

Now, a grammar design to generate a language that is what we know, and every non-terminal introduce in the grammar should contribute to the generations of strings in the language, is meaningless to introduce a non-terminal that do not occur in derivation that generate terminal strings in the language. Because, it will make a grammar unnecessarily large and clumsy, the same is true for a terminal as well, we would liked to eliminate all symbols of this kind normally denoted as useless, that means which not record at all. Elimination of these symbols makes the grammar simple, straightforward and if our formally define useless symbols and then, give methods to eliminate them.

(Refer Slide Time: 03:16)



We say that a symbol say x belong to the set of non-terminals or the set of terminals is useful, suppose x is symbol belong to N union sigma, we say that x is useful in the grammar G, N, sigma, P S. N is a set of non terminals, sigma is set of terminals, P is a set of productions and S is the star symbols, we say that x is useful. If there is some derivations of the form say starting the star symbol in zero or more steps in the grammar, we get some sentential form like this say alpha x beta, which in zero or more steps eventually derives w.

Where, w is basically a string of terminals, eventually we derive a string of terminals and alpha and beta maybe any again of terminal and non-terminals; so alpha is any string of

our terminals non-terminals and w is a string of terminals. So, starting with S eventually we derive a string of terminals and we get a sentential form where we have the symbol x, that occurs over here, in such a case we see that x is useful.

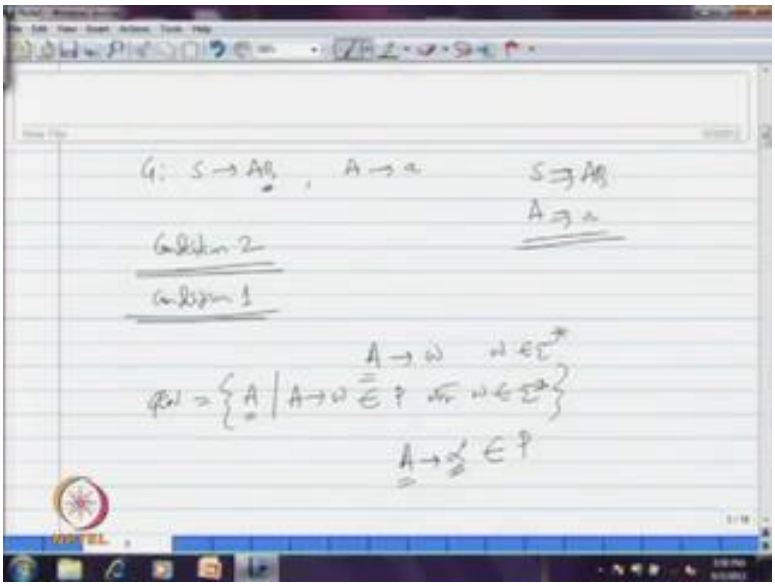
And a symbol is useless if it is not useful, so this is the case if useful, if this ((Refer Time: 05:07)) and a symbol is useless if it is not useful, so that is the definition of useless symbols. That means, a symbol is useless if it is not used in derivation of any string w, in the language generated by the grammar, so this is what we mean by an useless symbol. Now, a terminals symbol is useful if it occurs in a string of the language, that is what we know.

And similarly, a non-terminal is useful if it occurs in a derivation that begins with the star symbol of the grammar that eventually generates a terminal string, so that is what we know about the usefulness of a non terminal symbol. That is for a non terminal suppose A, belong to N the set of non terminals, say suppose A a symbol that belongs to the set of non-terminals.

So, for non-terminal A to be useful it need to satisfy the following two conditions, so starting with S in zero or more steps under the grammar G, we should have this sentential form, say alpha A beta, where alpha beta belongs to N union sigma star, it maybe string of terminals and non-terminals. So, N union sigma star, in such a case we see that the symbol A is reachable, that means starting the star symbol we can eventually reach the symbol A, A reachable.

And the second point is that eventually starting with A in zero or more steps in the grammar G, we should be able to derive a string w for some w belong to sigma star, that means a string of terminals. In such a case we say that the symbol A is generating or live, we used a term generating or live to indicate such kind of symbols and we say that, A is reachable if starting with S, we can get this kind of sentential form. But, a converse does not hold that means, even if both the conditions satisfied, in such a case we cannot say that A is useful, A maybe useless.

(Refer Slide Time: 08:17)



Now, an example if we say given a grammar G like this, so S goes to A B and A goes to say small a, where small a is non-terminal symbol and A, B and S are non-terminals. So, in this case even both the conditions satisfied, because starting with S, we can get S derives A, B, so first condition satisfied and the second condition also A derives eventually A using the second production.

For in this case A is, the non-terminal A is not useful, because eventually we cannot derive the string A starting in the star symbol S, this because of the second non-terminal that we have B along with A in this production. So, because of this non-terminal, presents in non-terminal creates the problem, so what we do in our procedure for elimination useful symbol, we start which applying the condition 2.

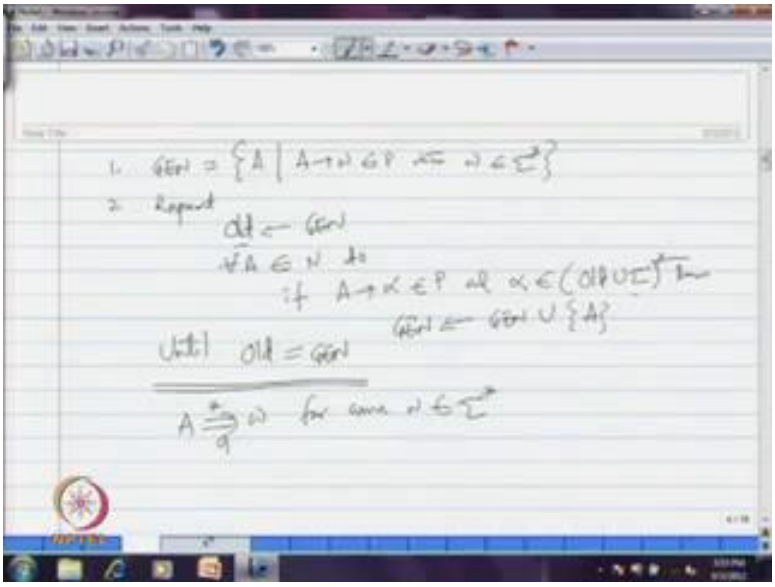
We first applying condition 2, that means we will first eliminate all non terminal symbols that are not generating and non-terminal that are not generating and we applying condition 1 to remove symbols that are not reachable. So, given a context to grammar G, we first used algorithm to remove all non generating symbols from the grammar G and then, construct equivalent grammar G dash containing only generating symbols.

In the second step we eliminate all non terminals from G dash which are not reachable from the star symbol of the grammar and then, construct an equivalent CFG, say G double dash that contains no useless symbols. So, first let us construct or let us discuss a procedure to find an equivalent grammar G dash from G, which do not counting any non

generating symbol. So, our idea is that suppose A goes to w, where w belongs to sigma star that means, from A in one step we can derive a string of terminals, so therefore A must be generating.

So, we create a set say GEN, where we keep all non-terminals A such that, A goes to w belongs to the set of productions of the grammar, which say w belong to sigma star. So, if w is a belong to sigma star and A goes to w is in the grammar, then we say that A is generating. So, we keep all symbols in the set of in this genset, generating and then, suppose we have already included some symbols in this genset and then, suppose we have some productions like this A goes alpha that belongs to P. And already the symbols of alpha or non terminals set is there in alpha is already included in genset, in such a case if we also include this A in the genset, we continue this process until no more symbols can be added.

(Refer Slide Time: 12:20)



So, therefore, we can write it in the form algorithm, so first we include in genset all those symbol A such that, A goes to w belongs to P with the condition that w belongs to sigma star, in a string of terminals. And then we repeat this process, we first say create a sets whole where we put this genset, then for all A belong to a set of non terminals, we do the following.

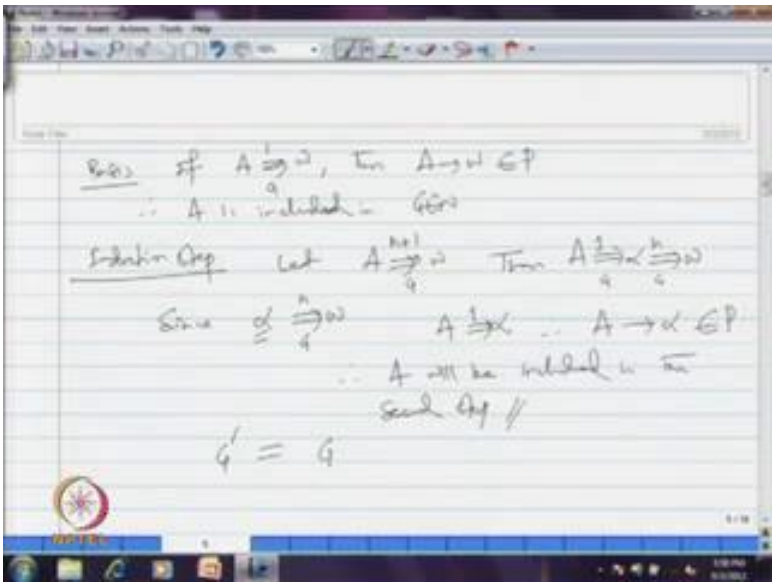
If A goes to alpha in P in the set of productions and alpha is already there in this set all or it may be, basically alpha is a string of, maybe string of terminals, non-terminals. So,

alpha may belong to all union sigma star, it may be any string over the symbol that are already there in this set that means, already included in genset or it may be string of terminals. Then we include in GEN this new symbol, GEN union A we keep on adding like this, until this all set and genset are identical.

So, in such a case what we can compute is that no more symbols can be added, so it is easy to see that this algorithm will terminate, because there finitely many productions in the grammar. Upon termination GEN contains only those and all those symbols or all those non-terminals and in those non terminals and only those on terminals, which are generating.

We can prove it easily by applying in that sum, suppose A is a generating non terminal and A eventually derives w under the grammar G, for some w belong to sigma star, that means it generates the string of terminals. Now, we can apply in that sum on a numbers of steps, needed to generate w, to show that this GEN contains only those symbols and on those symbols, which are generating.

(Refer Slide Time: 15:19)



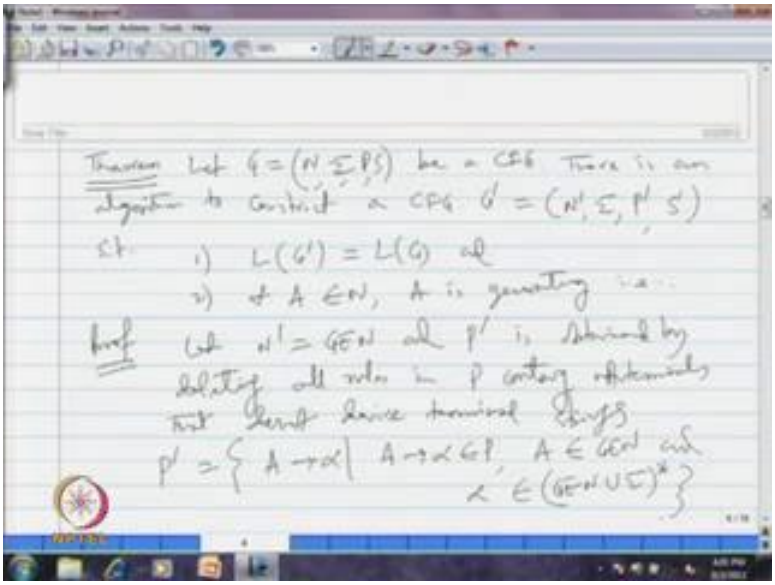
Now, the basic case is that, if A derives w in 1 step under grammar G, then A goes to w must belong to P, otherwise it cannot derive the string w in 1 step, therefore A is included in GEN, hence A is included in genset that we have already seen in first step. Now, in that some step what you assume is that, let A derives w in n plus 1 step under G,

then we can write say A derives alpha in say 1 step under G and in say n steps eventually it derives w.

Now, since alpha derives w in n step under G, we can apply that induction hypothesis to calculate that all the non terminals in G are already included in GEN, so design this is the hypothesis. Again since A derives in 1 step the string alpha, therefore A goes to alpha must belong to P, otherwise in 1 step it cannot derive, therefore according to our rule of the algorithm A will be included in the circum step over algorithm.

So, it is clear that no non generating non terminal will be included in GEN and thus the algorithm generates the desire set, that means the set of all non-terminal s which are generating and only those sets which are generating. Non terminals that are not in genset that we have constructed are useless, since they cannot contribute to the generation of strings in L of G. So, this observation leads us to construct a CFG G dash, which is equivalent to G and eliminates all variables of G that do not derive any string of terminals.

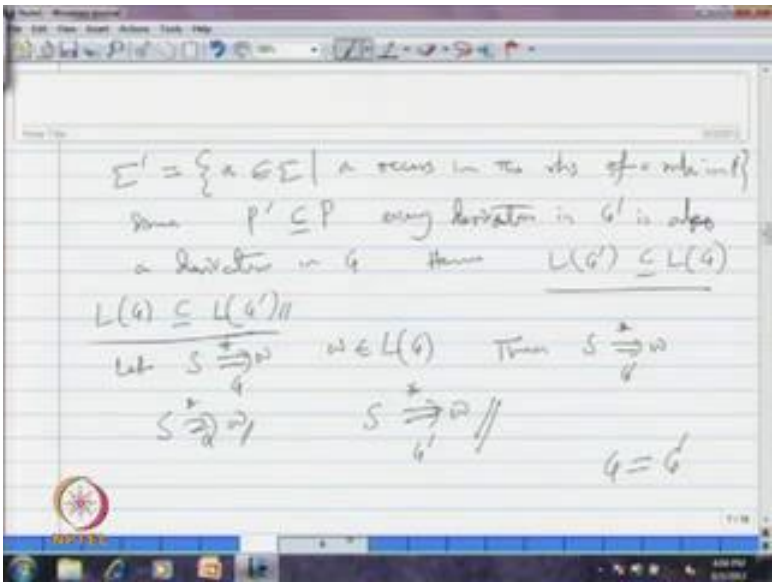
(Refer Slide Time: 18:17)



Now, we can say that in the formal theorem, say let G equal to N, sigma, P, S be a CFG, so there is an algorithm to construct a CFG G dash, which is say N dash, sigma, P dash and S dash such that, number 1 L of G dash equal to L of G. And number 2 for all A non terminal belong to set of non terminals A is generating, that is every non terminal in G dash, G dash is a terminal string in G dash, so you can easily now proof it.

Let N dash equal to the genset, that we have already constructed from G and P dash is obtained by deleting all rules in P. So, from P will delete all those rules, which are not generating, that means containing non terminals that do not derive terminals string that is P dash, specifically contains all those productions like this A goes to alpha. Such that A goes to alpha belongs to P and A belongs to genset and alpha is basically GEN union sigma star.

(Refer Slide Time: 20:57)



Also sigma dash equal to all those terminal symbols A belong to sigma such that, A occurs in the right hand side of a rule in P dash, that means here you will have sigma dash of course. Now, since P dash is purpose set of P, because from P we have eliminated some rules not containing, which contains the non deterministic symbol, since P dash is purpose of the P.

Therefore, every derivation in G dash is also a derivation in G, therefore is clear that L of G dash is a subset of L of G, now we understood the other side, that means L of G is a subset of L of G dash. So, that will prove that the resulting grammar G dash is equivalent to G, to show this we assume that, say let S derives is string w in grammar G, that means w is a string of L G and starting with G a S, we can derive the string w in grammar G.

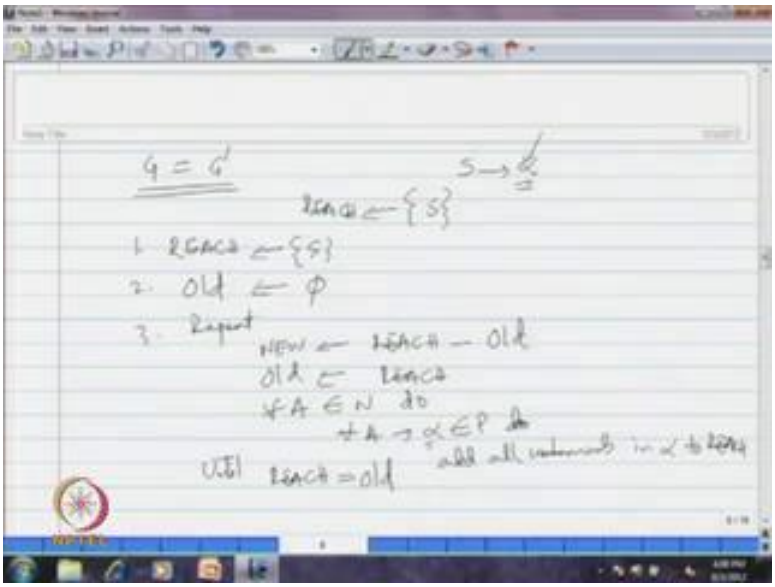
Then, what we can say if that is the case, then we want to show that, then S derives w in grammar G dash as well, then only you can say that L G is a subset of L of G dash. Suppose, this is not a case, then in non terminal which is non generating must occur in

and in terminal step in the derivation, that is what the case may be otherwise it is not possible.

But, a derivation from a sentential form containing a non generating symbol, cannot generate the terminal string, hence all the rules used in S derives w, so all the rules which are used in the corresponding derivation S dash w under G must also use in P dash. So, therefore since we are using all those rules which are generating this, and those rules must be in P dash, therefore S derives w under G dash also in zero or more steps.

So, therefore, as it is a subset of L of G dash and because of both of this, we can say that G is equivalent to G dash, that means we can now construct from G and equivalent grammar G dash, which do not contain any non generating symbols. Now, we will construct the set of all variables or all non-terminals, which are reachable.

(Refer Slide Time: 24:49)



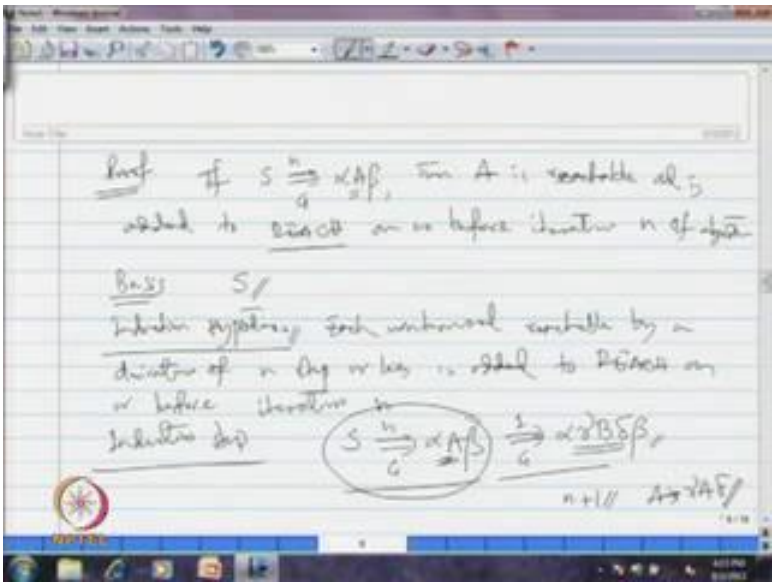
And then we will see how to construct the equivalent grammar containing only reachable non terminals, suppose given a grammar G by constructing G dash as we have already described equivalent grammar G dash from G, we have removed all non generating symbols. So, we must now remove variables that are not reachable that is what our aim, now we will use the following process. So, it is start with the first reachable non-terminal, which is basically the star symbol, star symbol is a first non terminal symbol, which is reachable.

And then, starting with this if there is a production like say S goes to some alpha, where alpha contains some non terminals, then that non terminal will also be reachable. So, therefore, initially we construct a set which is called reach, which contains only the star symbol S and then, for every production the forms S goes to alpha all the non terminals, which is there in alpha will be included in the set reach.

We keep on continuing process repeating this process until no more symbols can be added, so therefore we can write it in the formal algorithm even the grammar G, first will initialize reach to be the only symbol of this star symbol of the grammar. Then we assign this all set to be phi and this is use for checking the termination criteria of this algorithm, then you repeat this process, we create a new set, which is nothing but, all those symbols in leach. But, which is not there in old, so leach minus old and old is initialized to be now the leach set.

Now, for all non terminal A which belong to N, we do the repeated process for all production of the form A goes to alpha, which is in P do at all non terminals in alpha to leach. So, whatever non terminal we have in alpha must to added to leach, because is already there in leach, we continue this process until no more new non terminal can be added, that means until leach is equal to old. So, by this algorithm we keep on adding a new non-terminal, which is reachable from the star symbol S. Now, prove that a reaching contain only those variables, which are reachable from the star symbol of the grammar G.

(Refer Slide Time: 28:40)



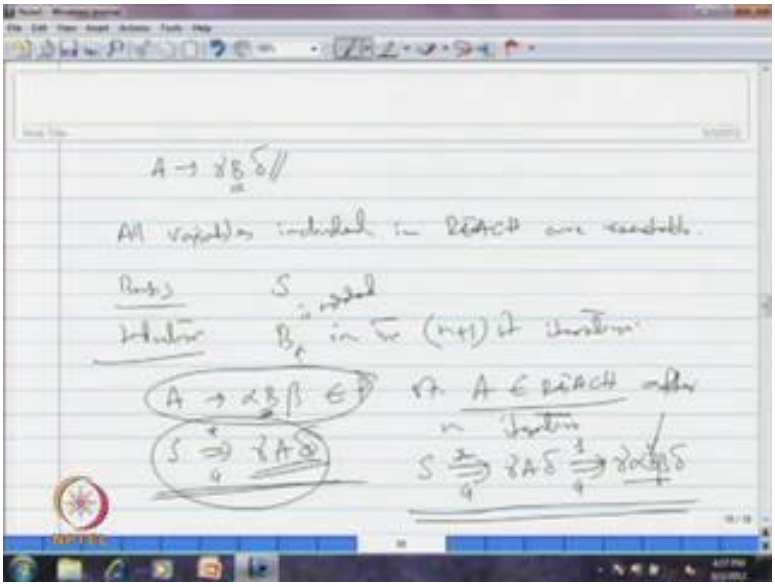
We first show that all reachable variables are added to leach, now if S derives suppose alpha A beta, in suppose n numbers of steps under grammar G, then A is reachable that is quite clear, because we can leach A from the certain symbol S using this derivation. Now, A is reachable and is added to reach on or before, iteration n of the algorithm or before nth iteration of the algorithm of the procedure, will add this symbol not under A which under on right hand side, that will be added to the set reach eventually, that is what we want to show.

That means, all reachable non-terminals will be put in the reach eventually, the basic case is that the star symbol S is the only symbol reachable by derivation of 1 step and it is added to reach in the step one of the algorithm. Because, S is already included in the first step by default, so this is a basic step, then we use this induction hypothesis, so each non-terminal reachable by a derivation n steps or less is added to reach on or before iteration n.

So, every non-terminal to reachable by derivation of n step or less is added to reach before iteration and so this is induction hypothesis. So, each non-terminal reachable by a derivation of n step or less is added to reach set on or before iteration n of the algorithm, so this is what our induction hypothesis. So, in induction step, so that say assume S derives in n step alpha A beta and in 1 step it derives under G say alpha gamma B delta beta.

Suppose these are derivations and length of derivation is clearly n plus 1, because the first step these takers n step and in under 1 step, we get this particular sentential form, so the length of the derivation is exactly n plus 1 step. Now, in the first step you will see that, in the first step we apply, the last step basically we apply a derivation A goes to gamma B delta. Because, this a is replaced by this string means A goes to gamma, A delta must be production in the grammar, they joy in 1 step we apply this and hence, we have got this ((Refer Time: 32:53)) form.

(Refer Slide Time: 32:28)



So, therefore, in the last step we have applied these two A goes to gamma B delta, so by inductive hypothesis A has been included in reach, ((Refer Time: 33:05)) because this takes n steps. So, according to this induction hypothesis, so since it takes n steps A is already there in a reach set. Now, variable B will be added in the next iteration according to our rule, so therefore every variable, which is reachable will be included in the set a reach.

Now, we show that all variables included in reach are reachable, that means we do not yet any variables, which are not reachable in the reachable set by this algorithm. So, the basic case is this, in the first step we include S and S is the already itself it is a star symbol with grammar, in induction step we show that, suppose the variable B is added in the n plus first iteration. So, B is added in the n plus first iteration, then according the

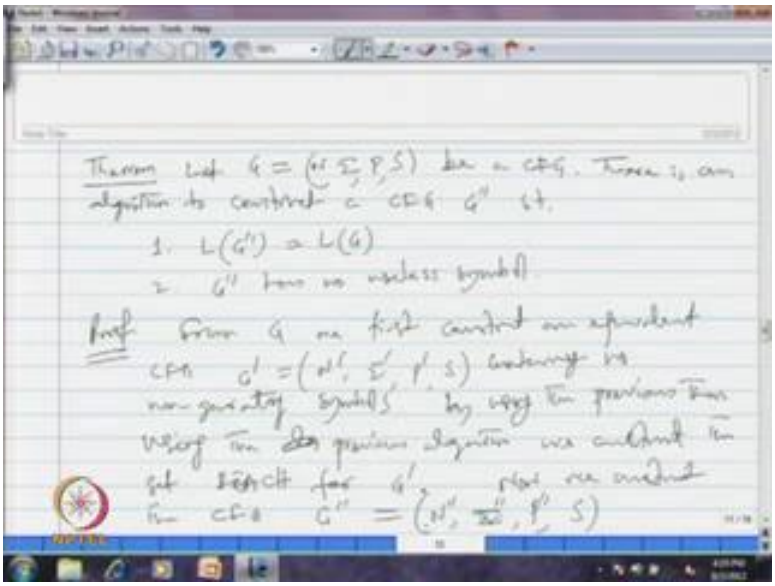
algorithm A goes to alpha B beta must belong to P, then only you can add this kind of production must be there, we applied in 1 step.

And in that step since A is already in the reach we include B in the reachable set, now so A goes to alpha B beta must be a production such that, A already belongs to the set that is what we say after n iteration. Now, by induction hypothesis S must be derives, they must be derivation like this in zero or more steps under grammar G alpha A delta, so there must be derivation like this such S derives gamma A delta, because A is already there in reach.

So, therefore, in zero or more steps S must be derive this sentential form gamma A delta, because A is already there in reach, and in the last step we have added B, hence the derivation S goes to gamma A delta, then in 1 step we have included gamma. So, A can be replace as alpha B beta by this production and an delta, so A is replaced by this, so in the grammar we have derivation like this. Because, A is already included we have this derivation and in the last step we say that we applied this production, so therefore we have the derivation like this.

So, according to this derivation we know that B is reachable and hence, since B is added we know that every such B must be reachable. So, therefore, whatever we add in reach set according to this algorithm must be reachable, therefore it contains the set of reachable variables and all the theory words which are there in it is reachable are reachable.

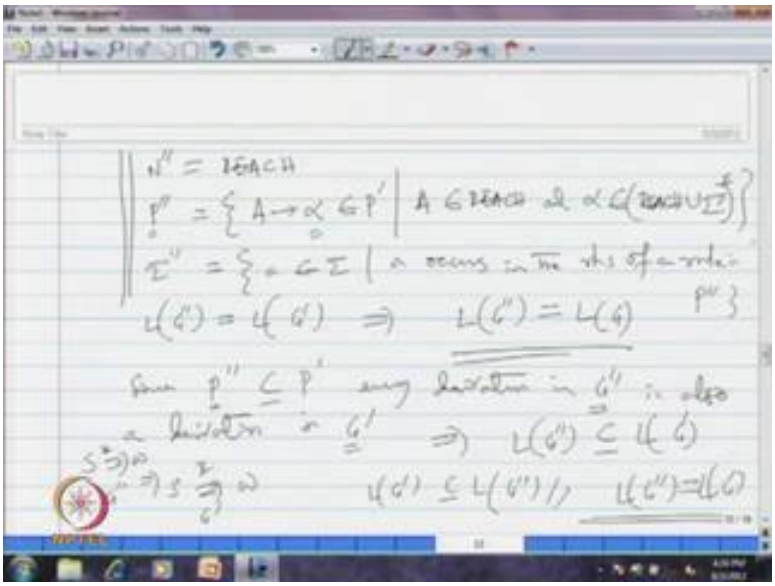
(Refer Slide Time: 37:04)



Once we have this algorithm, so what we can say is that, so that G equal to N, sigma, P, S be is CFG, so there must be an algorithm to construct a CFG G double dash, such that L of G double dash and L of G are equivalent, that means G and G dash are equivalent; and G double dash has no useless symbol. So, what we do from G, we first construct an equivalent CFG G dash, say N dash, sigma dash, P dash and S contain no non generating symbols by using the previous theorem or we have given how to construct it.

Now, using the algorithm just we have described or using the previous algorithm, just we have described we construct the set reach, and this reach for G dash we construct the set reach. Now, we construct the CFG G double dash, which is say N double dash, sigma double dash, P double dash and S according to the following rules.

(Refer Slide Time: 39:42)



So, what we do in the double dash, in the set of terminals in G dash it be basically the all those variables or set non-terminals, which are include in the reach set, because which are basically reachable. We construct all those symbols or non-terminal symbols, which are reachable and thus there will be in the set of non-terminals and P double dash it will contain all those productions that from A goes to alpha that belongs to P dash.

That means, which belongs to this grammar G dash such that, A is a reachable non-terminal, A belongs to reach and this alpha is a string of symbol, which is there in reach and sigma, so it is reaching and sigma star. So, alpha is string of symbols which are here in sigma, sub sigma dash and a reach and sigma double dash is basically all those terminal symbols A belong to P dash belong to sigma such that, A occurs in the right hand side of a rule in P double dash

If that symbol any symbol does not appear in the right hand side of any rule in P double dash that we have constructed, then that symbol is useless, so we can now establish that L of G. So, we use this three rules to construct the grammar G double dash and so that, L G double dash is equivalent to L of G dash and hence, this will imply that L of G dash eventually is equivalent to L of G, which is the original grammar.

Now, since this P double dash is subset of purpose of basically, a subset of P dash, so every derivation in G double dash is also a derivation of also derivation in P dash, so a in derivation of say G S. So, whatever we derive under grammar G dash, can also be

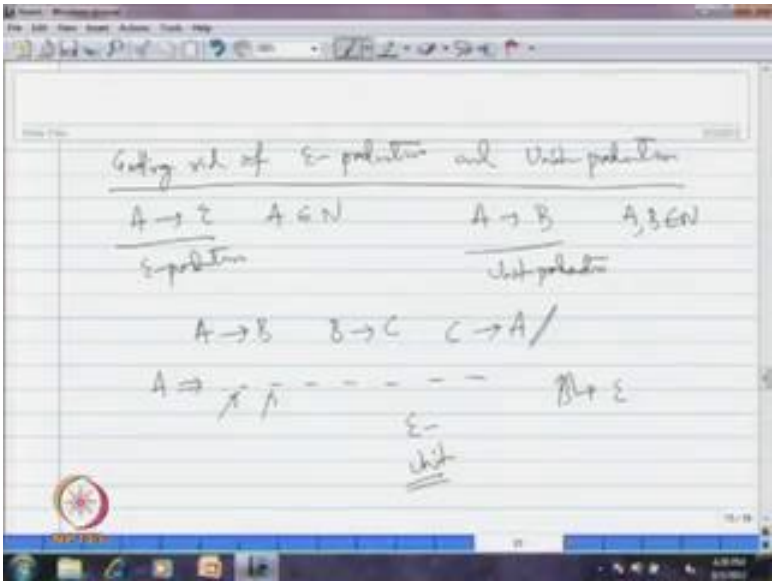
derived under grammar G dash, because P double dash is subset of P dash and therefore, this implies that L of G double dash is a subset of L of G dash.

Now, we need to show the converse, that means every variable in this derivation is reachable and hence, it is in reach and thus and N double dash and it will be N double dash and each tool applied will be in P double dash. So, therefore, S derives w under grammar G dash as well, suppose if S derives w under grammar G double dash we said that, S derives the same string w under grammar G dash as well.

So, therefore, similarly we can show that L of G dash is a subset of is similar to the previous one that we have already discussed, so L of G double dash. So, therefore, L of G double dash is equivalent to L of G dash these are identical and hence, since we have already shown that L of G dash and L of G are same. Therefore, L of G double dash and L of G are same, so this is how we can get rid of all the useless symbols by using the two algorithms, that we have described first to remove or eliminate the non generating symbols.

And next to construct a set of reachable non-terminals and then, eliminate all the productions, not containing the non generating symbols and not reachable symbols. So, we have seen how to remove useless symbol by using these two algorithms, next you will see how to get read of epsilon productions and unit productions.

(Refer Slide Time: 45:44)

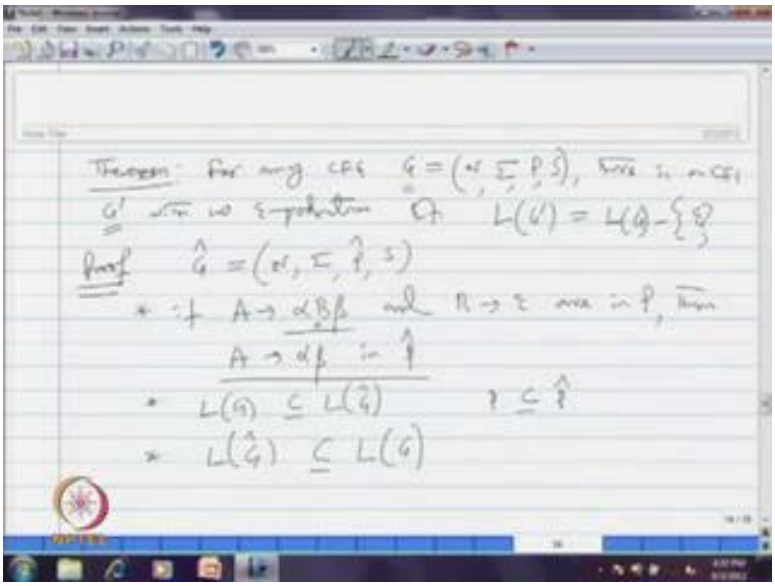


First let us define what is the epsilon production, we know that a production to from A goes to epsilon is said to be an epsilon production, where this A is a non-terminal. Similarly, a production a type A goes to B, where both A and B are non-terminals, is said to an unit production and this is epsilon production. Now, we would liked to simplify a grammar by removing these productions, because sometimes it may be difficult to determine, whether applying a production of these types in a derivation makes any progress toward deriving in a terminal string.

For example, if we use say this production A goes to B first and so B goes to C and then, C goes to A, they are all unit productions. And we suppose apply this in a sequence and such a case, we will even enter a loop A goes to B, B goes to C, C goes to A and again A goes to B, B goes to C, C goes to A, so it may so happened we are entering a loop. Similarly, we can generate a long string deriving starting with some non-terminal and we can keep on deriving a long string and then make it empty by applying some say B goes to epsilon, so all these B empty.

On the other hand, a derivation of CFG G without epsilon production, so there is no epsilon production and there is no suppose unit production, then we can be very sure that there will be a demonstrable progress at every step. In the sense that either a terminal symbol will appear in the right hand side or in sentential form or the sentential form will get strictly long. So, in that sense it will always better to eliminate unit production and epsilon production of course, if we get read of epsilon production, the grammar now cannot generate any epsilon the string epsilon in the language.

(Refer Slide Time: 48:22)



Now, I will construct this theorem say for any CFG G, which is say N sigma, P, S, there is a CFG G dash with no epsilon production such that, L of G dash is equivalent to L of G, but the string epsilon is not there. So, L of G minus the set containing epsilon, so L G can G dash can generate all those strings, that is there in G except for epsilon and all those strings, which will be there in L of G will also be L of G, so let us prove this.

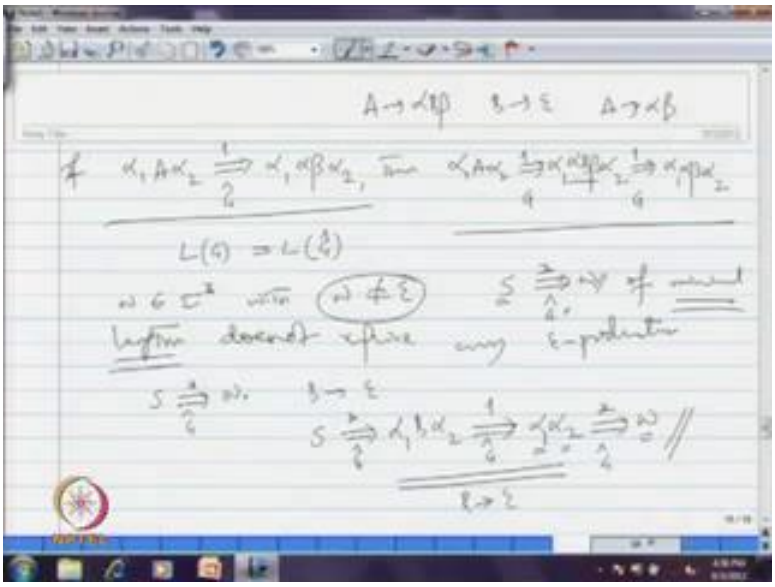
So, we will construct given the grammar G, from grammar G we will construct a grammar say G hat, which is say N, sigma, P hat that is that means, we modify the set of productions with a original grammar G. So, P dash is basically constructed by using a few rules, so first is that if A goes to alpha B beta and B goes to epsilon are in P, then we include A goes to alpha B alpha beta in P hat.

That means since B goes to epsilon, we can remove B, because B can be replaced by epsilon, and we in right hand we have only alpha beta, so therefore A goes to alpha beta will be included in P hat. So, there is a new production that we have included in P hat and then, it easy to see that P is finite, then so is P hat, so P hat juggles finite provided P is finite.

Now, clearly you can show that L of G is a subset of L of G hat, because L of G contains some more productions, beside the productions that are there in L of G, so it is quite clear that L of G is subset of L of G, since P is a subset of P hat. Therefore, every production or every derivation in G must be derivation of G dash as well, again L of G

hat can be shown to be a subset of L of G. That means, whatever you can derive in grammar G, G hat must be there in G as well, because every new production added to P hat, because the above rule that we have already constructed can be simulated in 2 steps, but two productions that causes to be added in P hat.

(Refer Slide Time: 52:15)



That is say alpha 1 A alpha 2 from this sentential form of course, we have derived in G hat in 1 step, say alpha 1 alpha beta alpha 2, suppose this is derivation in G hat in 1 step, then in grammar G, what we can do we start with alpha 1 A alpha 2. Then in 1 step in grammar G this A can be replaced by alpha B beta, then we have alpha 2 as well, so A goes to, because A goes to alpha B beta and B goes to epsilon both are deriving P, that is why you have included A goes to alpha B in P hat.

So, therefore, in 1 step we use this one and then, in the next step in grammar G, we can use, so alpha 1 B goes to epsilon, so alpha 1 alpha beta alpha 2, so whatever we derive in G hat can also be derived in G. But, here the length of derivation will be 1 step more, that is what we have done, so therefore L of G is equal to L of G hat, so it is quite clear. Now, what we can show is that for any string w belongs to sigma star with of course, it is not equal to epsilon, this w is not equal to epsilon any derivation S starting with S under grammar G hat of minimum length does not require any epsilon production.

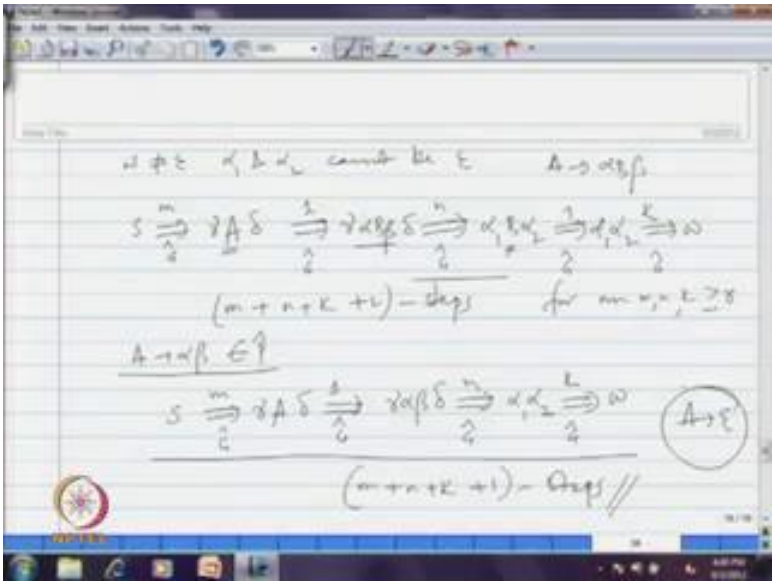
So, in G hat you can, if we can derive w starting with S and if that derivation is a minimum length, then you can show that this minimum length derivation of w in G hat

does not require any epsilon production. So, therefore, we can always throw out all those epsilon productions without changing the language of the grammar, so therefore we do not require any epsilon production.

Suppose, for contradiction we assumed that there is a minimum length derivation as goes to w in G hat, that quenches an epsilon production to derive w, suppose there is a minimum length derivation, which is as epsilon production, say that epsilon production would form say B goes to at any point of derivation. That means, we have a derivation like this S goes to under G hat alpha 1 B alpha 2, then in 1 step under G hat it uses B goes to epsilon, so it is alpha 1 alpha 2.

And eventually suppose in zero or more steps under G hat it gives us w, suppose these are scenario that we have considered for contravention, it uses in this step the production B goes to epsilon. And I have said this is the minimum length, now we will arrive at a contravention.

(Refer Slide Time: 56:09)



Now, since w is not equal to epsilon, so both alpha 1 and alpha 2 cannot be epsilon, because alpha 1 and alpha 2 both cannot be epsilon, since w is not epsilon, that we have already said, w is not epsilon. Now, since B is since alpha 1 and alpha 2 cannot be epsilon, B must appear earlier in a derivation, when a production of the form say A goes to alpha B beta was applied, that is the above production we have written as S derives in some m steps suppose, say gamma A delta.

Then in 1 step under G hat gamma A goes to alpha B beta was applied and then, in suppose n numbers of steps under G hat eventually, it goes to it has taken the forms alpha 1 B alpha 2. And then, in 1 step under G hat, we have got alpha 1 alpha 2 applying rule B goes to epsilon and then, in suppose k steps under G hat it has derived w, so clearly this derivation has a length of m plus n plus k plus 2.

So, many steps are created this is a length of derivation, because m steps over here, n steps, k steps and 1 plus 1, 2 steps for some m, n, k greater than or equal to 0, this is a case. Now, according to the construction of G hat, since we have constructed G hat, A goes to alpha beta belongs to P hat, because we have applied we know that A goes to alpha B beta is in the grammar G hat. And B goes to epsilon is also there in the grammar, therefore A goes to alpha beta must also be there in production or hat.

So, therefore, what we can do is that starting with S, suppose in m steps under G hat you have got alpha A delta, so therefore at this point since A equals to alpha beta is a production here, we can apply in 1 step that production, and we will get gamma alpha beta delta under G hat. And then, using this n steps we will arrive at alpha 1 alpha 2 and then, using those k steps eventually we will get the string w, but this derivation has clearly m plus n plus k plus 1 step.

So, we have seen that, this is a minimum length derivation not the previous one, so therefore, this is the contradiction, that means we do not actually need the production and of the from A goes to epsilon which are there in the grammar G hat. So, therefore, we can threw out all those productions and eventually we will get a new grammar, which do not contain any epsilon production. And which can generate all the strings that was there in the earlier language, except for the epsilon production.