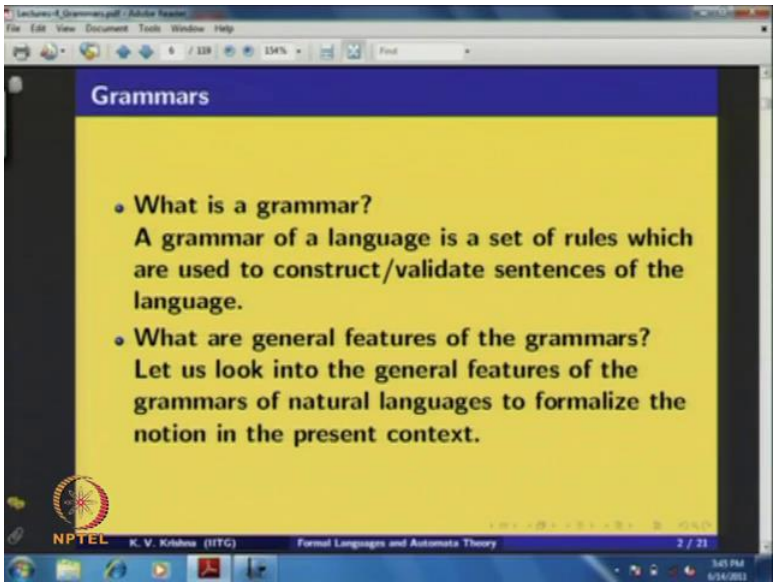
Formal Languages and Automata Theory
Prof. Dr. K. V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module - 2
Grammars
Lecture - 1
Context- Free Grammars

So, for we have learnt, what is formal definition of a language and how to represent them finitely finite representation? In that context we have called the notion called regular expression to the language. Now, there we have understood that some of the language we can represent to through regular expression and sum of them you could not. So, in this context we will introduce a better tool to represent a language finitely; the finite representation.

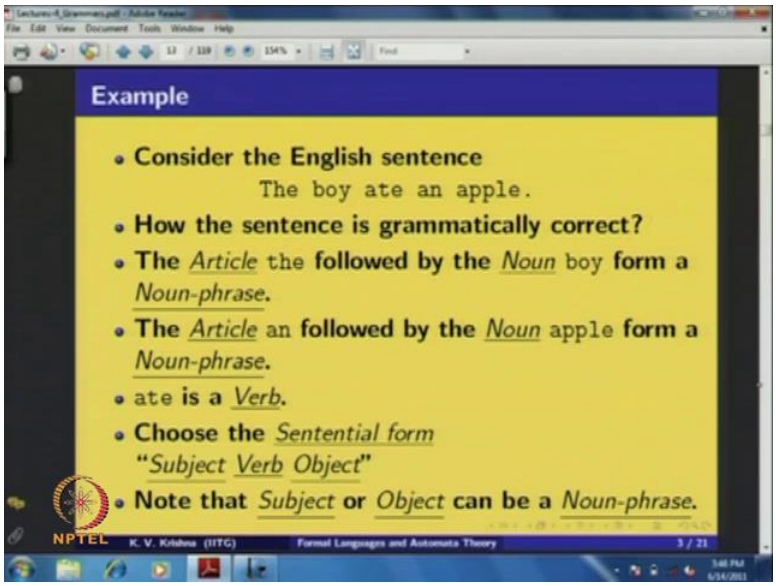
(Refer Slide Time: 01:10)



So, in the context we are introducing the tool called grammars as naturally one may expect to understand the language. So, again here the notion of grammar word grammar is familiar to you and you know grammars in the context in the natural languages. Now, first here we formalize the notion and then make you familiar with this formal notion so certain examples. First what is the understanding we have about grammar; that is a grammar of a language is a set of rules which are used to construct or validate sentence of the language; that is how we know grammar that is the expectation we have. Now, to

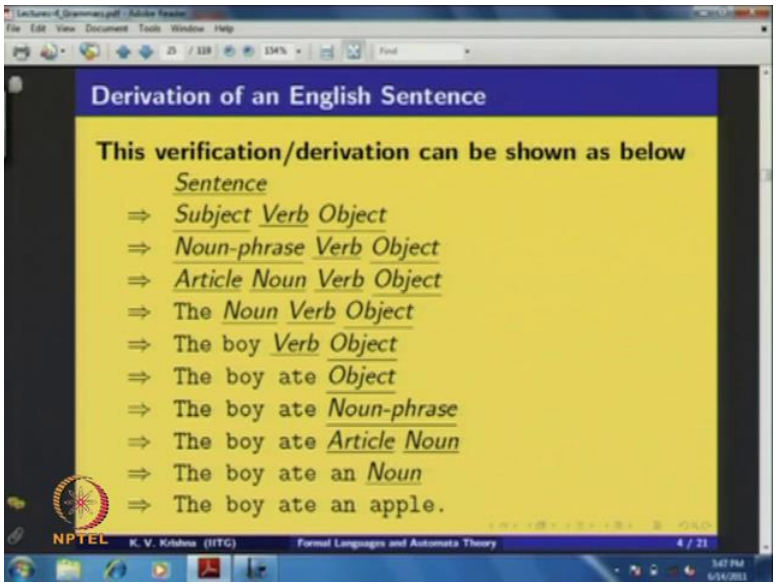
formalise we have motion; we have to look at certain general features of the grammars; that we know already from which the important features that may be captured and may be reflected the formulation of the notion of a grammar. So, when we are looking for the general grammar that means in the context of natural languages you may first look at.

(Refer Slide Time: 02:18)



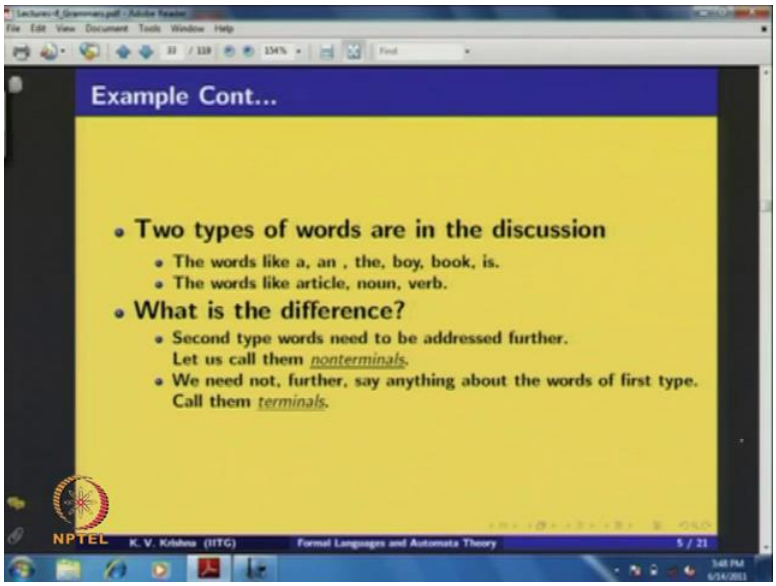
So, from English let us the consider the sentence the boy ate an apple; if you question that this is grammatically correct or not? Let us look at this way the article the followed by the noun boy form a noun phrase. The article an followed by the noun apple form a noun phrase; and ate is a verb and now if you choose a sentential form svo, so called subject verb of the object. Now, you can understand note that subject or object can be a noun phrase and you can validate the sentence.

(Refer Slide Time: 03:05)



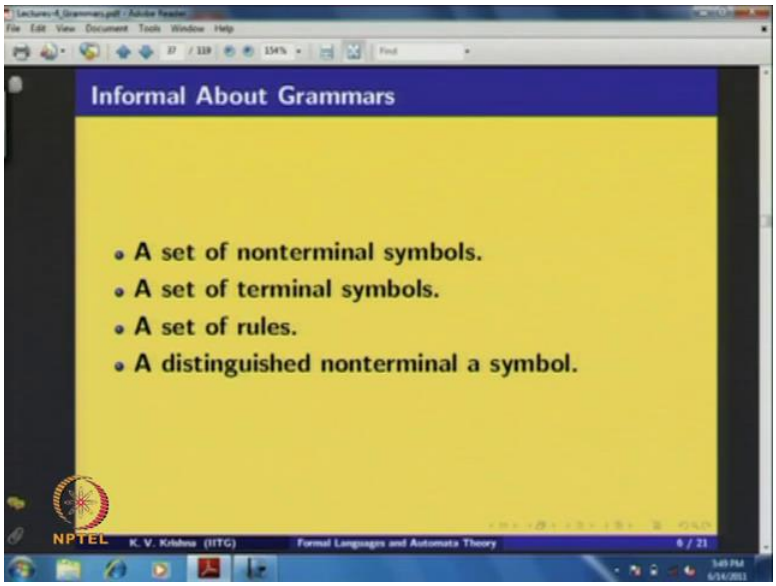
For example, you can validate the sentence or you can verification sentence the sentence will be geometrically correct in English. So, in the as I am showing here you may possibly validate that take the sentential form, take the sentence that rule subject verb object then subject you may place by noun phrase. And then the noun phrase can be an article followed my noun that rule you may use. And the article you placed by the article b and then noun by boy and then verb by ate. In case of object you are choosing noun phrase the rule and the noun phrase can be article followed by noun and the article here choosing noun and the noun apple. So, the boy an apple can be past or can be verified through English grammar in this way.

(Refer Slide Time: 04:16)



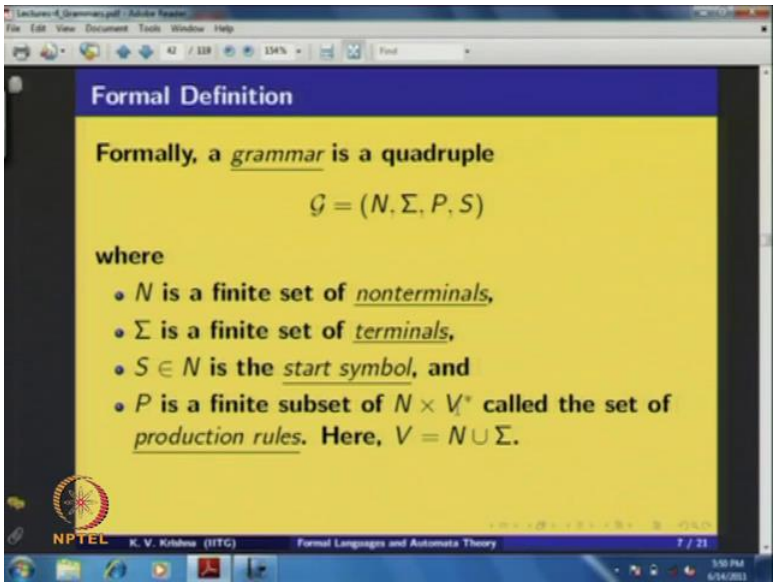
Now, 2 formulize a notion of a grammar; first let us see in this context what type of words that we have considered; the words like a, an, the, boy, book, is, this kind of word that you have encountered. The words like article, noun, verb, noun phrase; this kind of word that we have encounter. So, what is the difference between 2? If you look at the second type of words that means article noun, verb this kind of things you have to address them further. That means, we can say noun what is that noun you are going to talk about. So, you have to tell something about that further and that means this is not get terminated. So, you have to further tell something about this. Now, if we look at the first type that one; if you say an, the, boy that kind of words there is nothing further you are going to say in English. So, that way we may call them as terminals and the first one we call them as non-terminal.

(Refer Slide Time: 05:23)



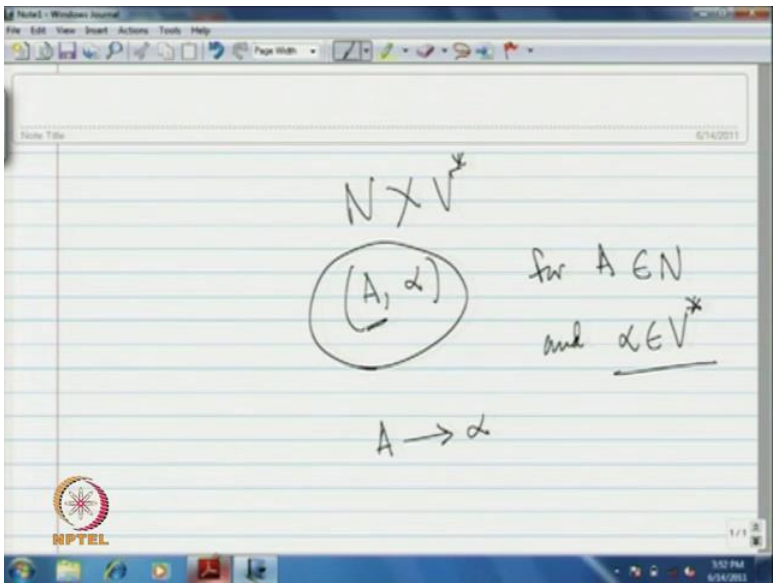
So, informally when I am talking about grammars; what are the things that you have understood here is a set of non-terminal symbols have to be there, as a set of terminal symbols and set of rules. For example, noun phrase can be a article followed by noun or a subject can be noun phrase; this kind of rules we have use in the derivation. So, a set of rules that you are requiring. Now a distinguished non-terminal symbol; that means for any sentence what is the main target in a grammar? So, you are going to validate sentences or you are going to derive a sentence through the rules the existing in the grammar. So, essentially target in to derive a sentence; so a distinguishable non terminal symbol that you are indicating as a sentence. So, that is to derive; any sentence from there.

(Refer Slide Time: 06:20)



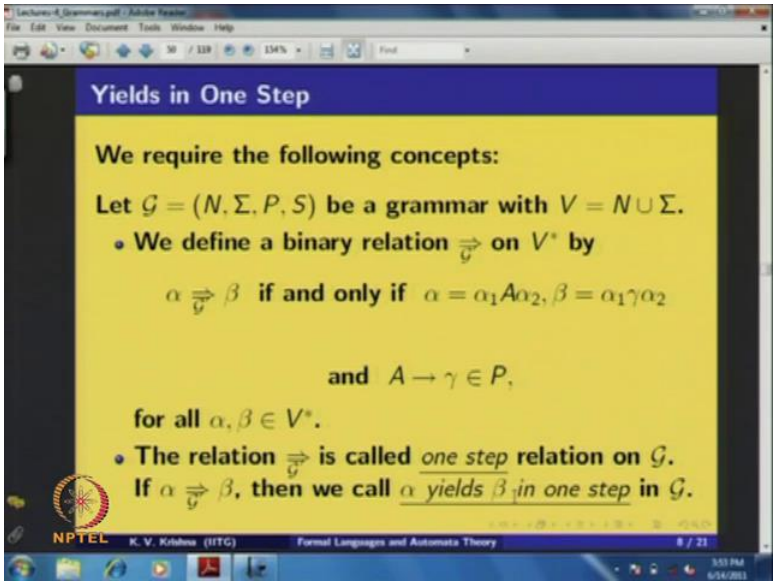
Now, formally we will introduce the combinatorial system; as a quadruple I am writing here G N, sigma, P, S; where N is a finite set we may call them as non-terminal symbols; sigma is a finite set again, a terminals again and S belongs to N; that means this is a non-terminal symbol; that we call them start symbol; this is a representing phrase sentence. And, P is finite subset of N cross V star called the set of production rules. What are the rules we are here having there are called production rules and here V equals to N union sigma.

(Refer Slide Time: 07:11)



Now, let us look at N cross V star here the elements are of the form N cross V star; a non-terminal symbol and a string of V square. That means, here alpha is a mix of terminals and non-terminal and string over mix of terminals and; this A is a non-terminal symbol. So, the rules are essential phrase which are of the form A alpha. Now, for convince since we are calling it a rule we write A alpha in this way; and called A can be alpha. Now, before going to talk about the sentence that are to be validated through a grammar and our sentence which can be derived or generated are using a grammar; we require certain concept.

(Refer Slide Time: 08:29)

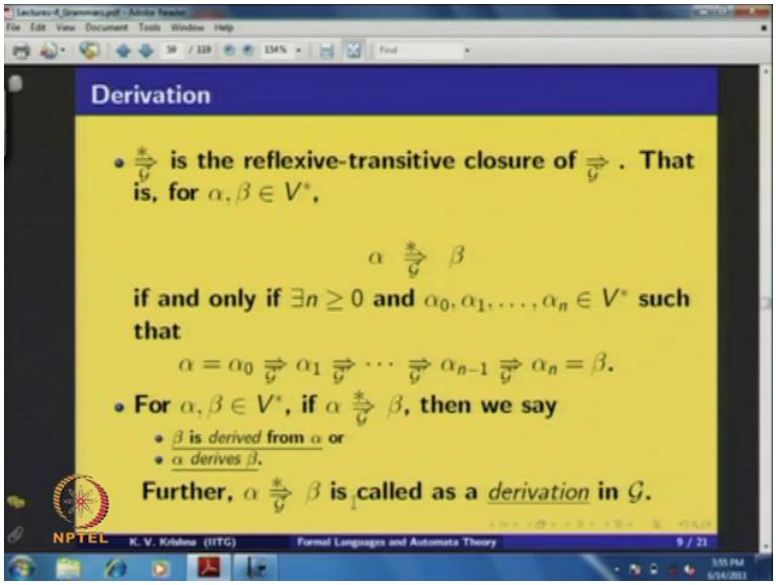


So, let us first look at those concepts. Let us consider a grammar g that is a with V is N union sigma; first we define a binary relation that is represented here with a implies symbol; that is with subscript g on V star. That means, we are depending on a binary on V star; V star again I repeat this the strings with mix terminals and non-terminal.

How we are defining this relation alpha related to beta using this relation if and only if alpha is of the form alpha 1 A alpha 2 beta is the form alpha 1 gamma alpha 2. Here, the relation between alpha and beta that for a non-terminal symbol A that we are replacing by the gamma. That means, it is replaced by a production rule; then we say that they are related. So, that means here and A gives gamma is a production rule in this grammar; for all alpha in beta in v star. Now, the relation this is called one step relation on g. Because

applying the 1 rule from alpha we get beta in 1 step. So, this is called one step relation on g. And, if alpha gives beta in one step then we call alpha yields beta in one step in g.

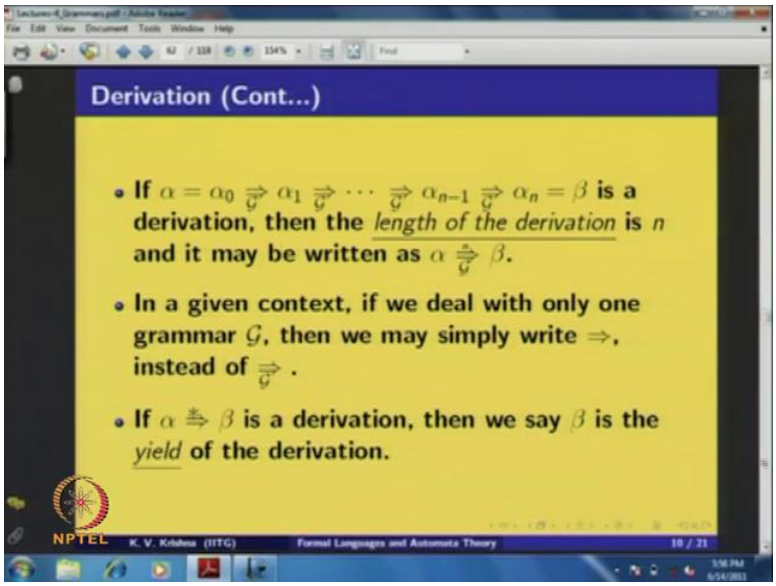
(Refer Slide Time: 09:59)



Now, another notation here; this relation with super script star if we write; that is reflective transitive closure over the one step relation that we have defined. That means, if you take any 2 strings alpha, beta in V star; we say alpha related beta with respect to this reflexive transitive closure of one step relation if and only through finitely ((Refer Time: 10:33)) these 2 are related. That means, they exist number n greater then equals to 0 and strings alpha naught alpha 1 and so on alpha n star such that this alpha naught is alpha and you can get in 1 step alpha 1 from alpha naught and so on. And, alpha n you will get it from alpha n minus 1 that alpha n is beta.

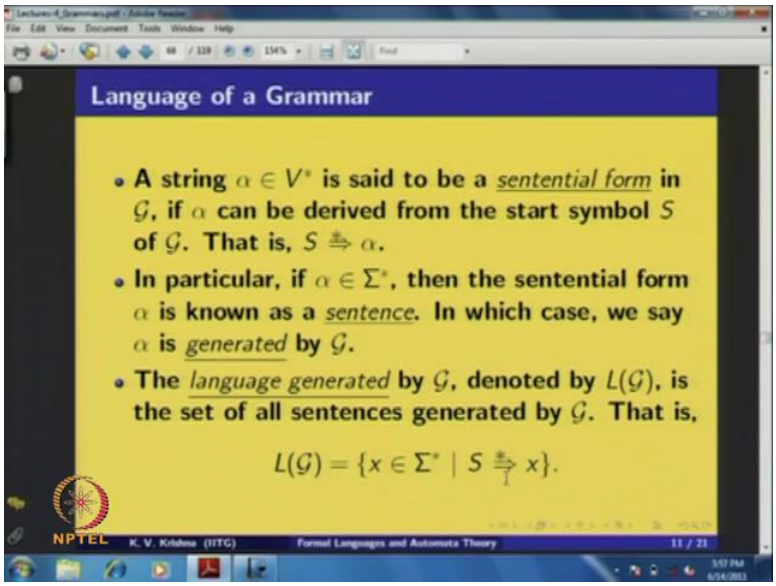
So, that means essentially this reflexive transitive closure the relation reflexive transitive closure is by through finitely steps of 1 step relation; you are relating 2 strings. Now, for alpha beta; if this two are related to be reflexive transitive closure relation that means through finitely meaning steps you can get beta form alpha. Then, we say beta is derived from alpha or we can say alpha derives beta; further this expression alpha gives infinitely beta is called derivation in g.

(Refer Slide Time: 11:42)



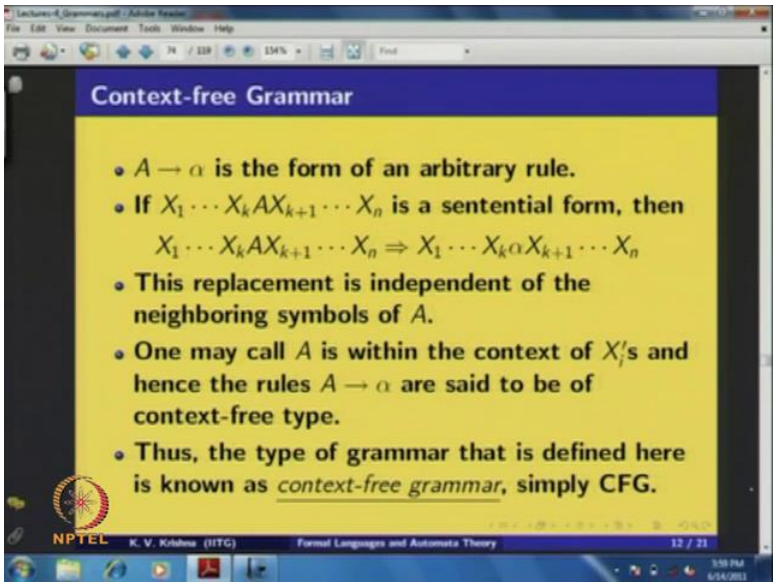
The notion of derivation is reduced here formally. Now, if you consider a derivation that is alpha gives alpha 1 1 first step, and alpha 2 is second step and so on alpha n that is beta. If is derivation then the number of steps in this will be counted and say that it is length of derivation; we can see that here n steps and thus we called length of the derivation is n. And, in which case one by denote by taking the n in the subscript super script of the relation. Now, let us on the convention very time we are using the subscription g in give a context if you are handling with only one grammar g. Then, you may simply use the symbol implies using the implies with the subscript g. Now, if you consider derivation alpha if gives beta infinitely in many steps; we also called beta is an yield of the derivation.

(Refer Slide Time: 12:46)



Now, if we take a string alpha is V star we call this is a sentential form in G; if can be derived from the start symbol S; in the grammar and consideration we have start symbol S from which we can derive this string alpha. Then, we say this is sentential form. In particular if this alpha only with terminal symbols. That means, if it is an element of sigma star a then sentential form is called sentence; in this case alpha is generated by G. And, now we formally depending the notion of generated by the grammar G; the language generated by g denoted by L (G) is the set of all sentences generated by G; that is L (G) set of all x and the sigma star; those terminal strings which can derived from in the start symbol.

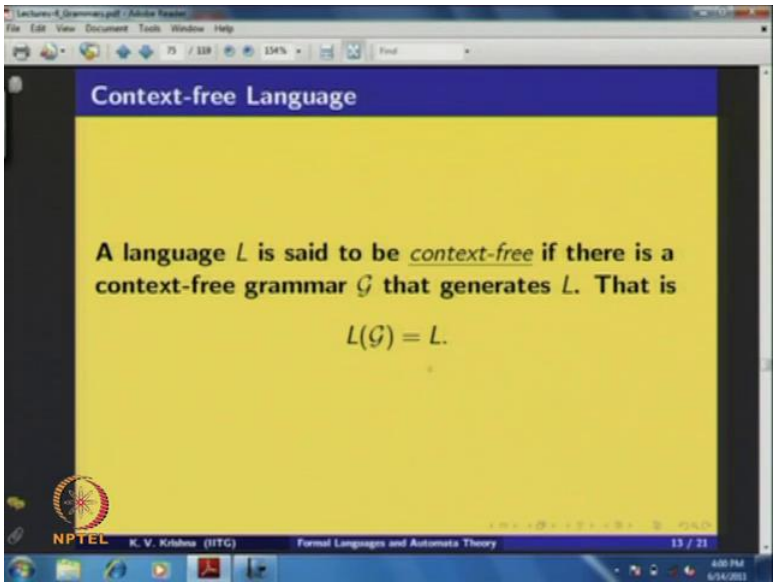
(Refer Slide Time: 13:50)



Now, if you look at an arbitrary rule in the grammar that we have defined; it is the form A gives alpha; for A goes to alpha. If the sentential form that means this string X1 and so on X k A X k plus 1 and so on X n; if this sentential form that means this is derived from the start symbol. And, as this A goes alpha is a rule; by applying this rule in the sentential form you can get a next step which is are the form X 1, X 2 and so on X k alpha X k plus 1 and so on X n. Now, in the sentential form the replacement is independent of the neighboring symbol of A, because of the production rule is of the form A goes to alpha wherever A is occurring you can substitute A by alpha to get a sentential form resulting like this.

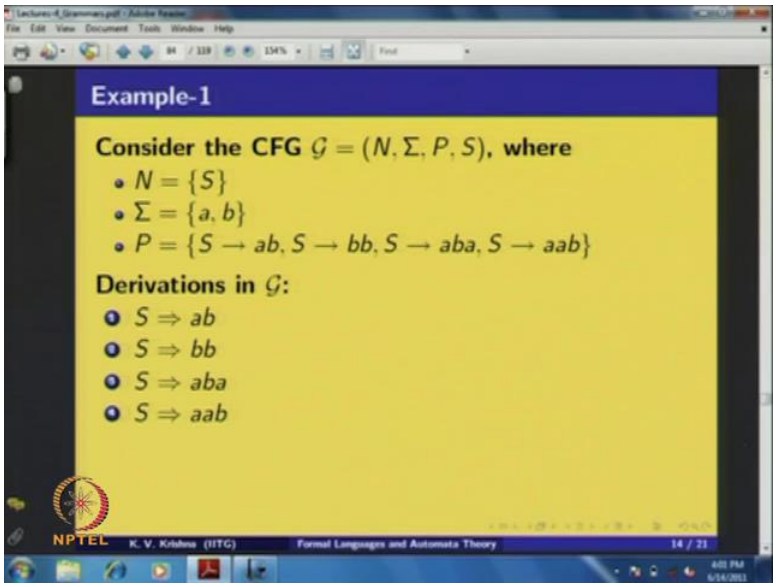
Now, as the replacement independent of A; one may call A is within the here you observe first that the non-terminal symbol A is within the context of X i's the neighboring the symbols X 1, X k the neighboring symbols of A are say for example X k X k minus 1 X k plus 1 or if you go little more X k minus 1 X k, X k plus 1; these are the neighboring symbols of A. Now, one may call A is within the context X i's depending on the neighboring symbols that you are considering. Now, hence the rule A goes to alpha is said to be of context free type; the reason why when you substituting A by alpha we are not worried about the neighboring symbols of A. And, this rule is a context free type and the type of grammar the defines of now here we called the context grammar for simply C F G; context free grammar.

(Refer Slide Time: 16:04)



Now, the motion of context free language is as follows; a language L is said to be context free if there is a context free grammar G that generates L; that is L (G) equal to L; the language generated by G should be equal to L.

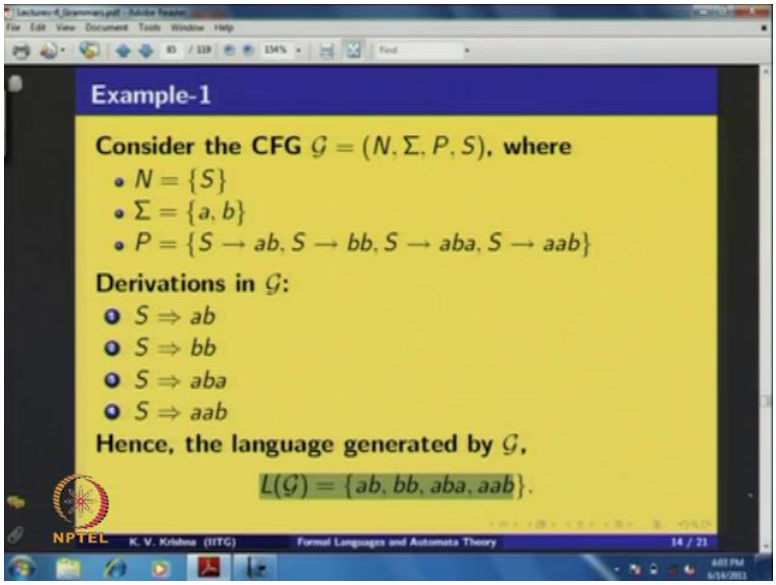
(Refer Slide Time: 16:25)



Now, look at the certain examples to understand the notion defend here. Consider the C F G N, sigma, P, S; where N is single time S, sigma is set of a, b. And, the production rules we are considering S gives a b, S gives b b, S gives a b a, S gives a a b finitely many rules. And, left side only one non-terminal symbol is only one non-terminal

symbol is there and right side mix of terminals and non-terminal are allowed. Here, we are considering only terminal symbols. For example, here a b and b b here and a b a here and a a b here; finitely many rules are under consideration. Now, what let us look at the certain example derivation in G; S derives string a b in one step by using the rule S gives a b. Let us look at another derivation S derives a b by using the second rule S gives you can derive the terminal string bb in one step again. Similarly, S derives a b a you can apply the third rule, we can one step we can that the terminal string; also S derives a a b and can we have some other derivation in this grammar; there is only non-terminal symbol you will start there is only one non-terminal symbol.

(Refer Slide Time: 18:13)

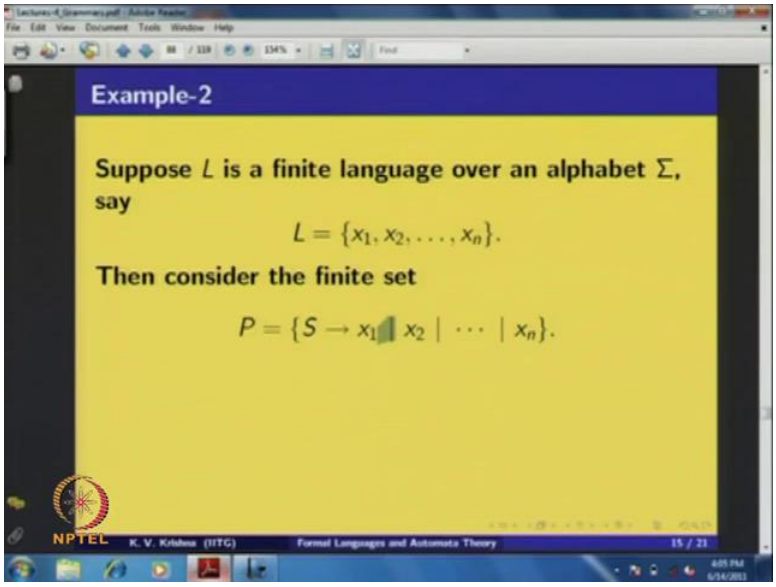


You will have a non terminal symbol A and you have to apply one rule to the next sentential form in a derivation under consideration. The possibility understand here you may apply the first rule and second rule, third rule or fourth rule whatever you are applying for example, apply first rule S gives a b. Then, after that to continue this derivation you require non-terminal symbol in this and there is no non-terminal symbol here. And, thus you cannot continue this derivation by starting with fist rule; if you want to the derivation more in one step. For example, if we consider the second rule S gives bb we have similar situation because right side immediately we got at terminal string. Similarly, third rule or fourth rule because every rule here essentially is of the form x goes to say some x; where x is sigma star terminal string; I do not have A option to continue to derivation further. And, thus here you can understand that any derivation in

this grammar is essentially have only 1 step here; in 1 step here you can generate a b or b b, a b a or a a b; there only 4 strings that we can derived here.

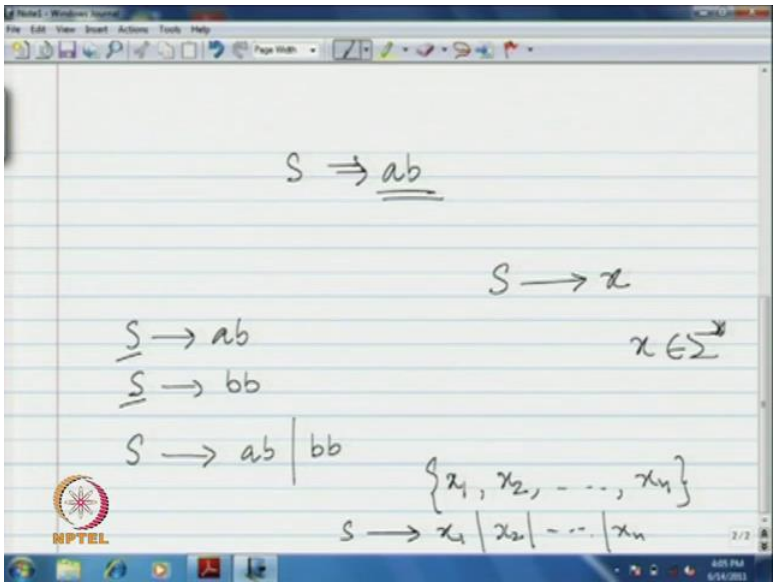
Thus, you can understand that the language generated by this grammar is continue precisely 4 strings a a, b b, a b a, a a b. And, this 4 strings can be generated to understand that this set is equal to L (G) and every string in this set can be generated by as G. So, derivations and these are only strings that can be generated in this grammar; thus the language generated by G is this.

(Refer Slide Time: 20:11)



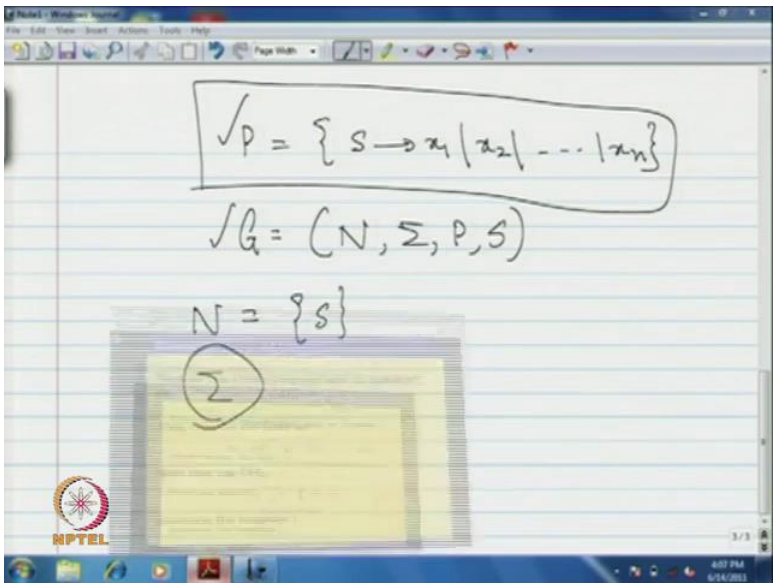
Let me consider another example by extending this motion whatever the examples here we have define here. In the previous example once again here we consider the 4 strings and this is the grammar we have define. Now, if you take any finite languages you have finitely strings in that so L equal to X 1, X 2 and so on X n; finitely strings are there. Now, if we consider finite sets P contain S gives X 1 here we are introducing the notation are symbol r X 2, r X 3 and so on r x n.

(Refer Slide Time: 21:03)



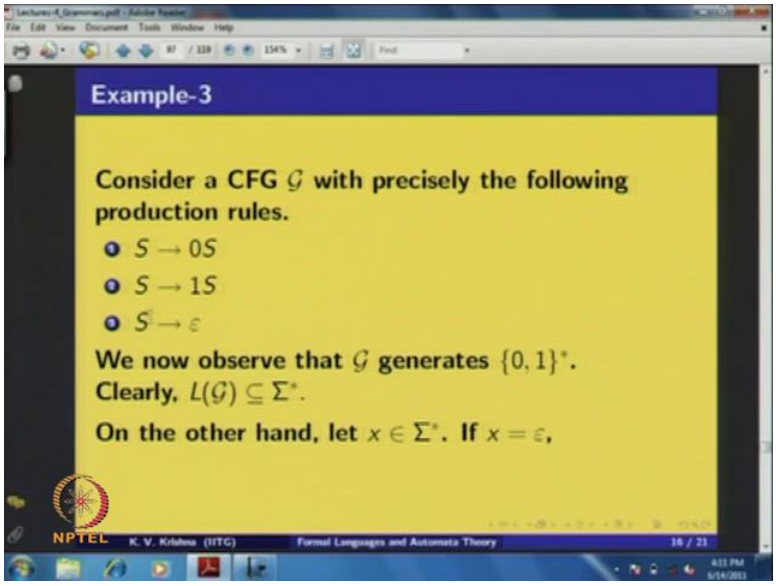
So, here the notation what we are introducing here is in the earlier case S gives to a b is 1 production rule, S goes to b b 1 production rule; that the notation we adopt instead writing 2 rules separately; when the non-terminal symbol left side when both having S we may write it as S goes to a b or b b; this is how we read. And, in the present context when we have finitely strings X 1, X 2 and so on X n the finite language; by considering the production rules s gives X 1, r X 2 and so on r X n where n number of rules. By considering we can generate the language X 1, X 2 and X n.

(Refer Slide Time: 22:07)



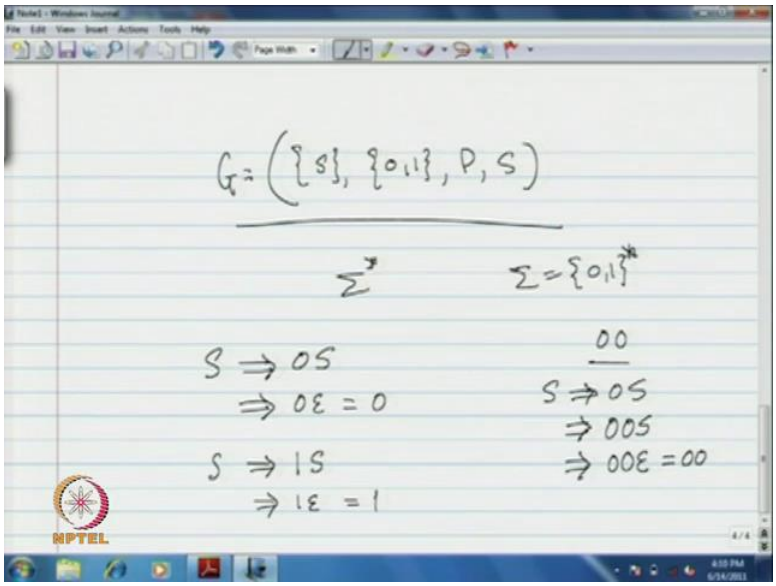
Now, here one more notation that we are adopting namely; if we are giving just production rules in this context S goes to X 1, r X 2, X n; we are not giving what is that quadruple for the grammar; we may not specially specify whatever the non-terminal symbol that is occurring in this set that you can considering in an. For example, if we here only non-terminal symbol and sigma whatever the terminals under consideration and production rules are defined and the start symbol S if non-terminal from non terminal. So, using this notation just by stating the set of production rules; one can always write quadruple G. And, thus using that notation I am simply stating note that the C F G; single terminal sigma, P, S generates the language L.

(Refer Slide Time: 23:21)



Now, let me do one more example. Consider the C F G; G with precisely the following production rule; 1 S gives 0 S; 2 S gives 1 S; 3 S can be epsilon. Now, every that the grammar G generate 0, 1 star; here I am not stating the quadruple; only the production rule are stated. If one is intersect to right it is easy.

(Refer Slide Time: 24:16)



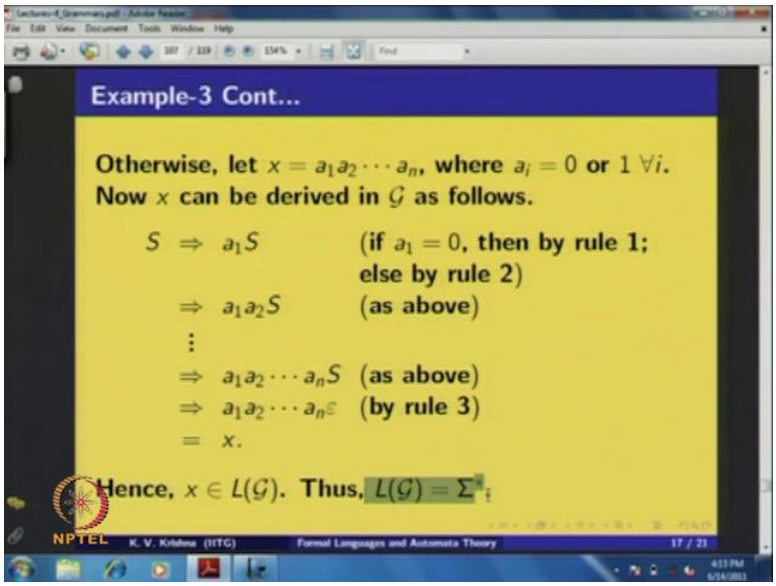
In this context you have again only one non-terminal symbol; terminal symbols are 2; 0 and 1; the production rules are stated and the start symbol S you consider. And, thus the quadruple G is this in the present context. And, if I call the grammar G we observe that grammar G generates sigma star; here sigma is 0, 1. So, all the strings or sigma can be generated in this, using this. To prove this first what are the possible string that we can generates using this rules; that we have to evaluate. First a fall can we generate some screen through this rules that we have understand.

For example, using the third rule S gives to epsilon one can clearly generate in one step ((Refer Time: 25:27)) epsilon or if you consider the first rule we get S gives the 0 S in 1 step. And, if you use the third rule then 0 epsilon that equal to 0. So, the string is 0 can be generated using this grammar; that means we are generating certain strings through this rules. Similarly, by using second rule one can derive 1 S and applying the third rule we get simply 1. So, 0 can be generated, 1 can be generated; if you want 0, 0 can be generated consider the first rule once we can derive like this. And, then we apply once more; the first rule we generate string 0, 0 S.

Now, if you apply the third rule 0, 0 epsilon the anti string epsilon that is essentially 0, 0. So, 0, 0 can be derived; and understand here the 3 rules; S goes 1 S and S goes epsilon you are deriving certain strings. And, whatever the terminals that we are deriving they are whole set 0, 1. And, thus first we understand that ever the strings generated this

grammar the L (G) is a subset of sigma star. Because sigma star is a set up all strings are 0, 1. And, therefore L (G) is a subset of sigma star; conversely we have to understand that every string of sigma star can be derived in the grammar; taking arbitrary string x and sigma star.

(Refer Slide Time: 27:40)

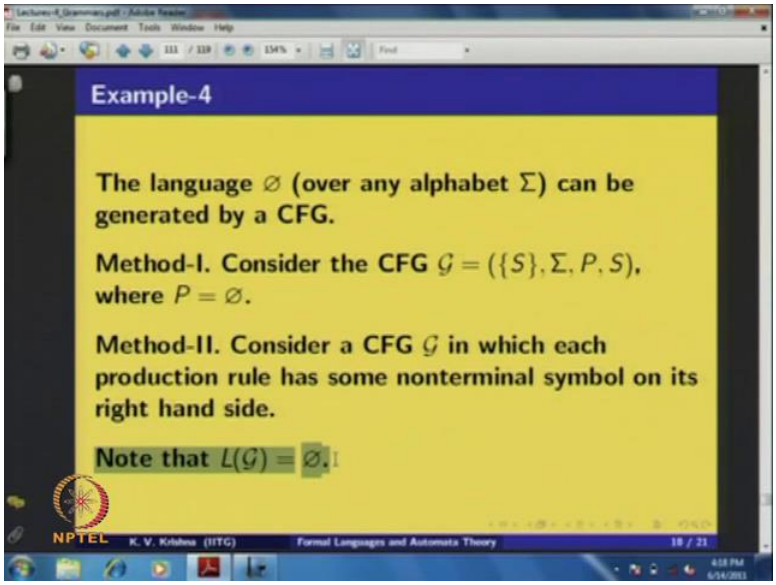


If is empty string using third rule in one step you can derive it if you write it as here a 1, a 2 and so on a n; where a i is 0 or 1; some n number of string if you consider. Now, this a 1, a 2, a n we can derived as shown below; this a 1 can be 0 or 1; if it is 0 then apply rule 1 to get that S gives a 1 S; if it is 1 then you can apply rule 2 to get 1 S here. Similarly, to derive the a to n either it is 0 or 1 I will discussed here; if a 2 is 0 apply rule 1 to get a 2 S; if a 2is 1 apply rule 2 to get a 1 a 2 S and continue this to get a 1 a 2 a n S. And, at the end applied rule 3 and nullify the non-terminal symbol here to get a 1 a 2 a n that it what is x.

So, any string x a 1 a 2 can be derived in n number of steps you are getting a 1 a 2; in the first step a n, in the second step get a 1 a 2 and following S and n number of steps, you are getting a 1 a 2 an S and n plus 1 step you can nullify S by substituting n 2 string. And, thus you can getting a 1 a 2 an this n string that we are deriving here n plus 1 step; the length of the derivation here n plus 1 and n; n means string can be derived here exactly n plus 1 step. And, thus you understand that any string or sigma star can be

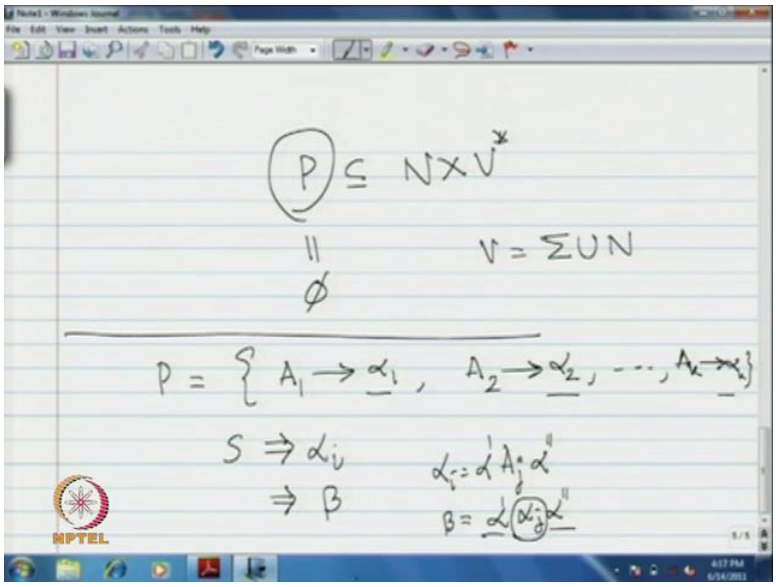
derive to the grammar; hence x is an L (G). So, we have got the include L (G) is sigma star.

(Refer Slide Time: 29:53)



Let us consider one more example; if you consider the languages empty; the languages are empty over any alphabet consider; we can say can be generated by a CFG. How do give the grammar? I give you 1 method here; if you consider the grammar single sigma P S; where P is empty.

(Refer Slide Time: 30:36)



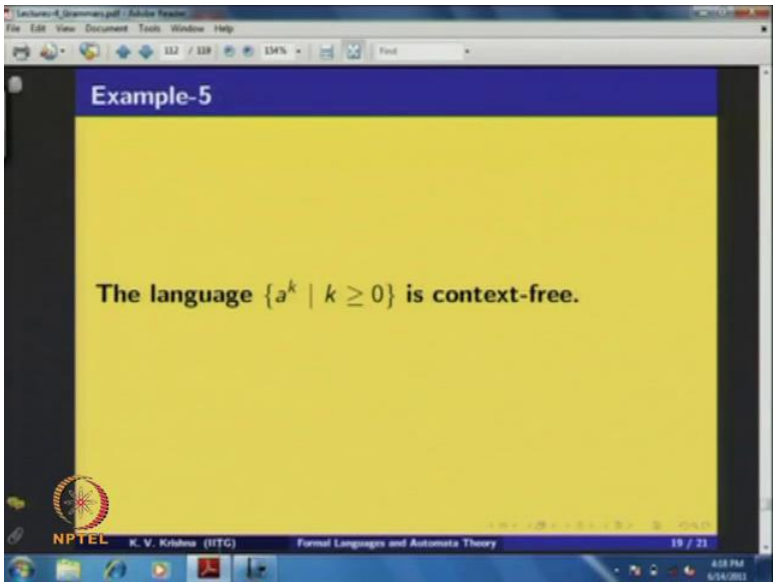
The set of production rules we did not put any restriction we simply said that this production rules is a subset of N cross V star; where V is sigma union N there is no restriction on this; it can be empty set also V subset of this. So, if you choose the production rules to be empty then we are not going to derive we will not able to derive any string in this grammar. And, thus the language generated by grammar G define here is empty or if you we want some production here we can follow this method 2; if you consider the grammar in which each production rule has some non terminal symbols on its right hand side; some non-terminal symbols are right hand side on each production rule. So, what is the situation that you get? That means, say finitely production rules that you would have; let me take for example, A 1 goes to alpha 1, A 2 goes to alpha 2 and so on; some finitely rules A k goes to alpha k finitely.

And, the assumption here is each alpha i has some non-terminal symbol on its right hand side. Now, you start the derivation start you may start with derivation; what is the symbol S? If start symbol equal to i here you may get that alpha i. And, then in alpha i what is the non-terminal symbol is appearing and we will continue to sum a j you substitute that. And, you are getting some string beta where beta is obtained from alpha i ;by substituting the non-terminal symbol that you have. Say for example, you have a j in alpha i that means in alpha i is of the form some alpha 1 alpha 2 and by substituting the what are the beta we have got beta is alpha 1 a j substituted by alpha j and alpha 2.

And, clearly as for our assumption let me call it as instead of writing alpha 1 alpha 2 which as used here already; let me call it as alpha dash say alpha double dash. So, what we got beta is alpha dash alpha j alpha double dash and in beta you can clearly that there is non-terminal symbol, because alpha j has non-terminal symbol. Now, they may non-terminal symbol alpha dash and alpha double dash also. Now, you see the non terminal alpha j there is no non terminal symbol.

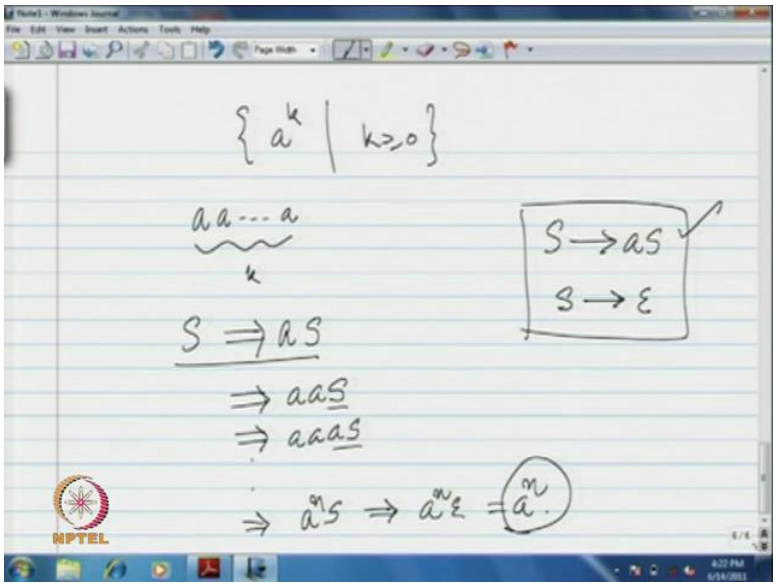
For example, if you choose and substitute and if you continue like this; every time will observing the there is one non-terminal symbol in each step that we are continuing to. So, you are continuing to non-terminal symbol always in each step and thus any derivations that you start with; it will never terminate. And, thus no terminal string can be generated using the grammar. And, thus this C F G with this property you will derived only one set. So, thus we have understand in either case, if you follow method 1 and if you follow method 2 the grammar and consideration generates the language empty.

(Refer Slide Time: 34:43)



Let us look at some more.

(Refer Slide Time: 34:56)

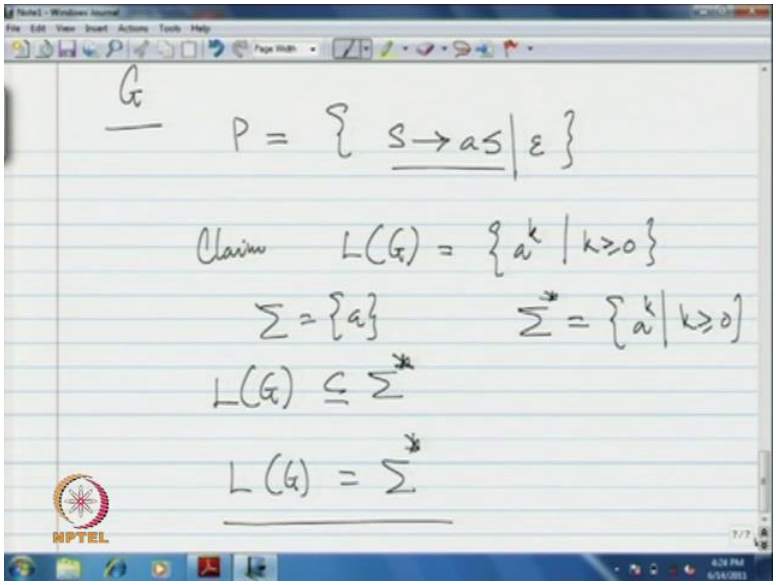


So, for what I have considered here; I have gave some grammar and understood that language is generated by those grammars. Now, in the previous case I have considered empty language and given an appropriate grammar to generate. Now, let us consider this; because what are the examples that I am starting with they are all known to you; that they can be represented by regular expressions. Because the finite set with 4 strings you know the regular expression of that. And, finitely ((Refer Time: 35:36)) that is the

regular language, sigma star that is the regular language and empty this is the regular languages. Now, one more here in the consideration you know this is also a regular language. And, here we are defining a contextual grammar for the languages; we are also observing that we languages are context free. So, a typical string in this having k a s; it can be empty string 2 a s, 3 s and so on a power k.

Now, for this purpose if you consider anyway you required non-terminal symbol as a start symbol; let me called as I said S; I have to derived a so let me take a and I should be able to further continue. So, because I am going to produce only a. So, if I consider the rule S goes to a s; I can have this kind of situation S gives a s and using the same rule I can derive 2 a a s. And, again apply the same rule we can get a a a s; because this S substitute by a s; using this rule. And, if continue this we can produce n number of a s after n step; if you terminate this by substituting S goes to epsilon the empty string. Then, you can terminate this that is what is a power n. But you observe this by considering by this 2 rules S goes to a s; S can be epsilon; we can clearly thus in n plus 1 step to generate a power n.

(Refer Slide Time: 38: 14)

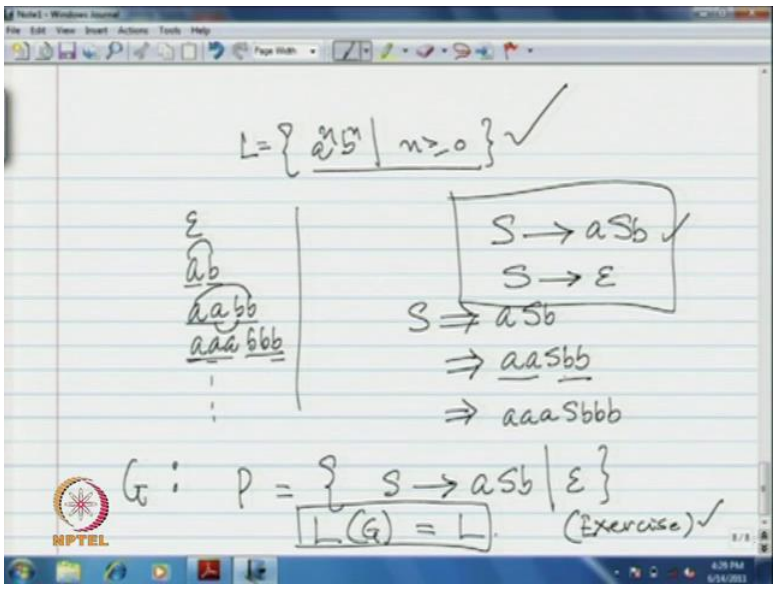


Thus, if I consider the grammar with this production rule; that is I am going to write the quadruple as a I said you can consider this non-terminal symbol and terminal set in singleton set. And, production rules are stated here; so just stating this we are stating the grammar. So, let G be the grammar with this following production rules; the claim now

is the language generated by this grammar is and as you understand here that; what are the strings are derive L (G)? That first we have look at and you have to look at sigma a single term a under consideration; what is sigma star? This is precisely a power k; k greater than equal to 0 and one can observe that whatever the strings that you are generating in this grammar is a subset of sigma star as discussed earlier. Now, to show that any string of sigma star can be generated; we can look at the previous example and quickly understand that we have consider by taking sigma to be consideration. So, thus you are understanding that L (G) equal to sigma star.

So, what is the point here is we are treating considering sigma to be single term a; as a special case of previously example. And, thus you understand that a power k greater than equal to 0; this is called to context free. Because if you considering sigma to be single term a this is what is to be sigma star.

(Refer Slide Time: 40:52)

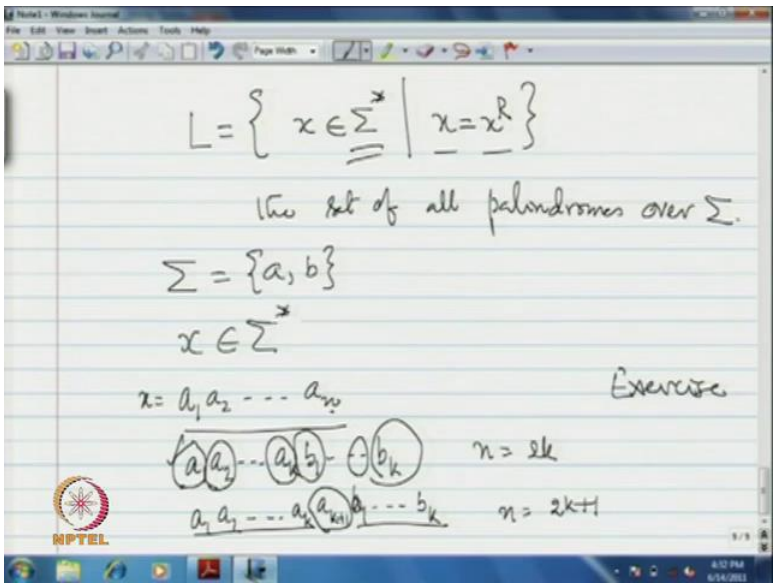


Let me discuss one more example; this you have not seen earlier; the language a power n b power n such that n greater than equals to 0. So, how does atypical string look like in this language; epsilon by taking the n equal to 0; you can have a, b here by taking n equal to 1; you can have a a b b and a a a b b b and so on; these are this kind of string are there in this language strings. Now, look at the pattern of strings here; corresponding to each a here you have a b; in this you see there are 2 a here 2 b, 3 a and 3 b. Let me look at the corresponds like this; this a is corresponding to this b. So, here let me correspondence

like this. And, similarly here this a is corresponding to this b and second a is corresponding to this second from the right side. Suppose, if you give this correspondence then you can think of the production rule; that can generate this language. Because in each step we have to remember that a and b are having the this correspondence. So, if I write the production rules like this and this production rule if you keep applying in each step you can generate left side n number of S. And, similarly right side n number of the strings. Let us look at the derivation of apply this rule to get a s b I have written only one rule. So, you have the choice of applying same look that 2 a s sorry here it will be b; 2 b are generated. Now, if you apply this rule once again what do you get a a S b b. Now, you understand how to terminate this 2 derivations by choosing the rule S goes to epsilon.

Now, this 2 strings sorry this 2 rules can generate any string of this language and conversely; ay string of this language let me call L can be derived through this 2 rule. Now, if you consider the grammar with production rules as define here S goes to a S b or epsilon; the grammar with this production rule. You can observe that the language generated by the grammar G is L; a power n, b power n such that greater than equals to 0; you can take this as exercise a string. That means, what you have to show? A string generated in G is on the form a power m b power m that is in L. And, if you take any string in L; that means a string of the form a power m, b power m can be generated through G. So, prove this as an exercise? Let me disused some more examples.

(Refer Slide Time: 45:06)



Let me disused some more examples power any alphabet if you consider the language; x power R; what is the language? All the strings in sigma star which is equals to its reversal. That means, the set of all palindrome; if consider this the previous example; the concept of example can be used to define a grammar for this language; for simplicity let me consider sigma to be a b; from that you can understand to give a grammar for the arbitrary alphabet sigma. What is the situation here; if you take any string x from the language L here; let me call L for this language; a string x would of the form a 1, a 2, a 3 and so on a n. There are two cases; one there may be even number of symbols here this is a even line, there may be of hard line. If it is that even line what do you have a 1, it is of the form a 1, a 2 and so on a k, let me write.

And then there after a k plus 1 say there are b 1 b 2 b k ((Refer Time: 47:19)) it is of length 2 k; n is equal to length 2 k. And, if it is of odd length I may write it as a 1, a 2, a k, a k plus 1. And, then let me write the b 1, b 2, b k let us write like this; this even length string and this is odd length string. Here, n is of the form 2 k plus 1; this is the line. Now, look since it is a palindrome the symbol a 1 should be equal to this b k; the a 2 should be equal to b k minus 1 and so on; this a k should be equal to b 1. In case of odd length the same situation on either side and a k plus 1 can be anything between a and b. Then, you can think off the production rules for the sigma set of a, b and give a grammar we generates palindromes; try this as an exercise. And, hence you can extend this for any arbitrary alphabet sigma.