Theory of Computation

Prof.Kamala Krithivasan

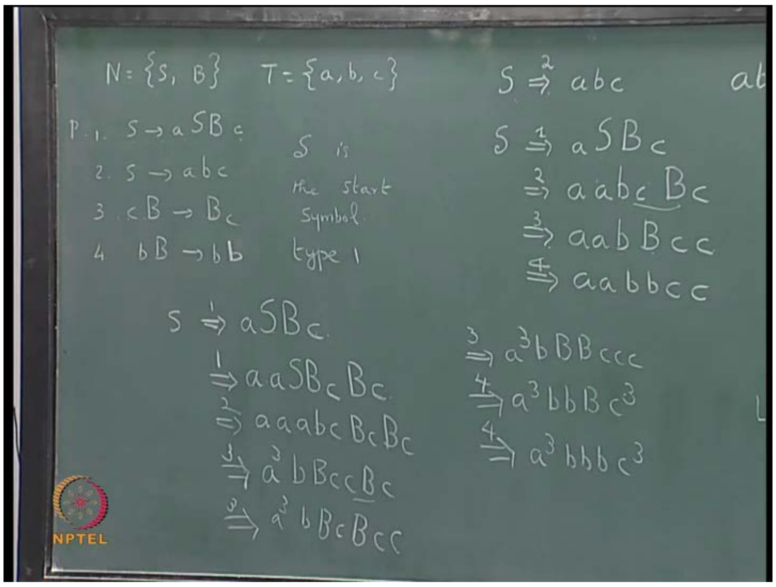
Department of Computer Science and Engineering

Indian Institute of Technology,Madras

Lecture No. # 03

Grammars and Languages Generated (Contd)

(Refer Slide Time: 00:19)



$N = \{S, B\}$ $T = \{a, b, c\}$

P. 1. $S \to aSBc$
2. $S \to abc$
3. $cB \to Bc$
4 $bB \to bb$

S is the start symbol.
type 1

$S \overset{2}{\Rightarrow} abc$

$S \overset{1}{\Rightarrow} aSBc \overset{2}{\Rightarrow} aa\underline{bcB}c \overset{3}{\Rightarrow} aabBcc \overset{4}{\Rightarrow} aabbcc$

$S \overset{1}{\Rightarrow} aSBc \overset{1}{\Rightarrow} aaSBcBc \overset{2}{\Rightarrow} aaabcBcBc \overset{3}{\Rightarrow} a^3bBcc\underline{B}c \overset{3}{\Rightarrow} a^3bBcBcc \overset{3}{\Rightarrow} a^3bBBccc \overset{4}{\Rightarrow} a^3bbBc^3 \overset{4}{\Rightarrow} a^3bbbc^3$

ab

L

So, in the last lecture we considered some examples of context free grammars, the grammars and the language generated by them.Today we shall consider some more examples.The first example we consideris this,there are two non terminals S and B, and three terminal symbols a,b,and c.And there are four rules, the rules are of the form S goes to aSBc, S goes to abc, then cB goes Bc, bBgoes to bb,this is small b.

Now, you can see that this is not a type two grammar or a type three grammar,becauseon the left hand side,if you have a single non terminal it is type two, but here you find that on the left hand side you have a string cB and bBhere. So, this does not belong to type two, but you can also see that the length of the right hand side is always greater than or equal to length of theleft hand side. So, this is a type one grammar.

And let us see what are the strings, terminal strings generated in this grammar,the language generated by the grammar consists ofall terminal strings derivable from the start symbol. So, we will see what the terminal strings derivable? So, using rule two alone,from S we can derive abcusing rule two alone. So, abcbelongs to the language, abcbelongstoL of G.

Now, starting from S if you use rule oneyou will get aSBc,now again for this S you can use rule two,then you will get a this S will be replaced by abc,this abcthen this Bc remains as it is, this is not a terminal string.Now, you can apply the rule three cB goes to Bc to this cB.So, this cB becomes Bc,the idea of this ruleis to bring allthe c(s) to the right and the capital B’s to the left.

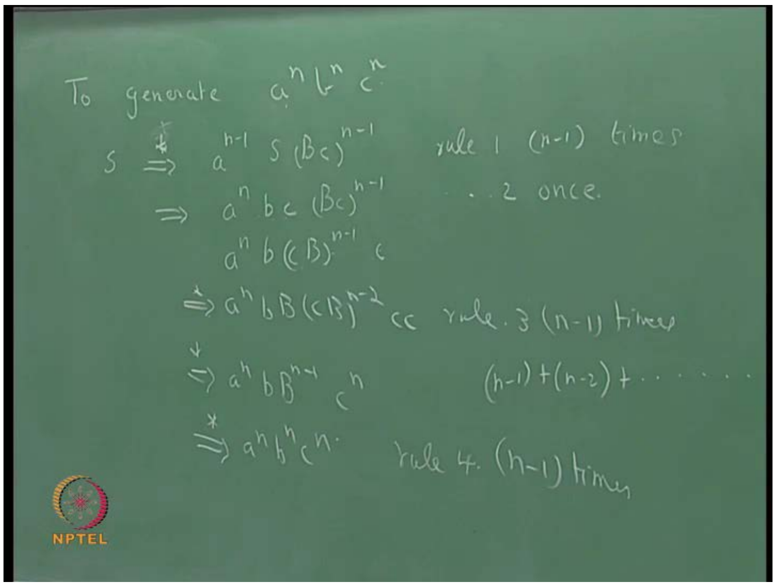
Now, this is applying rule three.Then applying rule four this bb can be, this capital B when there is a small b on the left side will be turned into a terminal b. So, rule four you apply and you get ab,aabbcc.This you can represent as a squared, b squared,c squared and this belongs to L of G. Now, let us consider one more example;I will start from here S use rule oneyou getaSBc,then againuse rule one for this S,then you will get aaSBc,this S is replaced by this aSbc, but this Bc will remain as it is.Now,apply rule two for the S,then you will get a a abc,this Bc will remain as it is,this will Bc will also remain as it is.

Now, use rule threerepeatedly to bring the capital B’s to the left and shifts the small c(s) to the right. So, you will have a cubedb, if you apply the rule for this cB you will get b,this b is herethen you will get Bc cBc.Again, you apply rule threea cubed bb c,if you apply to this cB,b will be shiftedto the left and c will be shifted to right, you will get a cubed bb cc.

Now, again to this cB you can apply rule three. So, you will get a cubed bBBccc.Now,you see that all the terminal c(s) are shifted towards the right, all the a(s) are on the left,now you have 1 b and capital B’s in the middle,the capital B’s have to be converted into small b(s) using rule four. So, apply rule four repeatedlyyou get a cubed bbB c cubed,again you apply rule four a cubed bbb c cube.

So, the string generatedis a cubedb cubedc cubedbelongs to L G.Now, you can see the pattern what will be the language generated by the grammar, the language generated will beL of G will bea power n,b power n,c power n,n greater than r equal to 1. So, this is a type one language or a context sensitive language,this cannot be generated by a type two grammar.In type two, what happens is when you replace by another non terminal, you can generate equal number of a(s) and b(s),we considered an example,where you can generate a power n, b power n.But,you have to keep the same count for a,band c, for thissomething else is required, you have to shift the as seen this example, you have to shift the c(s) generate equal number of b(s) c(s) etcetera, but then shift the b(s) to the side and shift the c(s) andso on, and that require some context. So, this cannot be... Lateron we give a formal proof that it cannot be generated by a type two grammar after learning some theorem.But,at present we will just make a note that this is the typeone language which cannot be generated by any type two grammar.

(Refer Slide Time: 07:38)



To generate $a^n b^n c^n$

$S \overset{*}{\Rightarrow} a^{n-1} S (Bc)^{n-1}$ rule 1 (n-1) times

$\Rightarrow a^n bc (Bc)^{n-1}$ . . 2 once.

$a^n b (cB)^{n-1} c$

$\overset{*}{\Rightarrow} a^n bB(cB)^{n-2} cc$ rule. 3 (n-1) times

$\overset{*}{\Rightarrow} a^n bB^{n-1} c^n$ $(n-1)+(n-2)+\cdots$

$\overset{*}{\Rightarrow} a^n b^n c^n.$ rule 4. (n-1) times

Now to generate a power n b power n, how do you go about...To generatea power n b power nc power n,how do you go about doing this?First,from S rule one you have to apply repeatedly, I will put star.You have to use rule one repeatedly, when you use rule one S will be replaced by a S B c. So, one a will be generated on this side, one B c will be generated on this side.

So, n minus 1 times if you apply rule one,rule onen minus 1 times you will get a power n minus 1 S B c n minus 1, BcBc Bc like that.Then you applyrule twoonce, you will get a power n Bc Bc powern minus 1. So, this you can also write as a power nb cB to the power of n minus 1 cright?Now, all this cB you have to shift the capital B to the left and small c to the right,for which you have to use rule number three there cB goes to Bc.How many times you have to apply this?

(( ))

There are n minus 1 c b(s),

(( )).

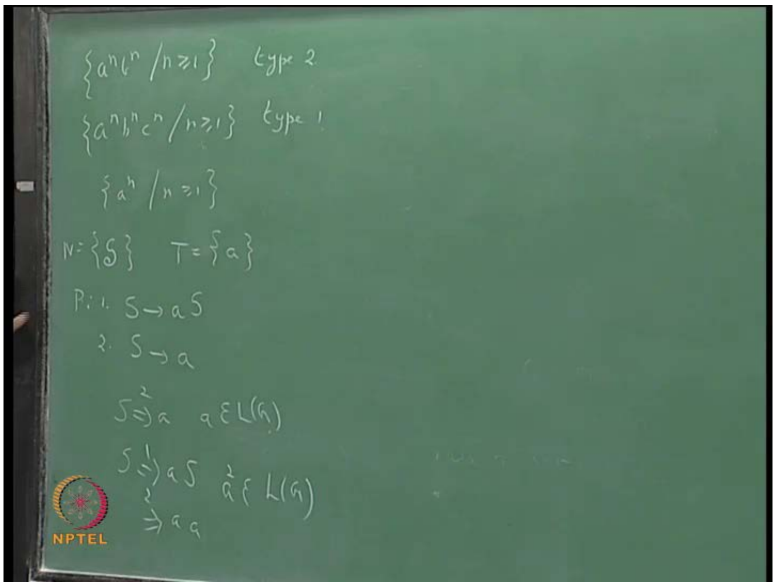
So, how many times you have to apply this?Ifyou can apply the rule n minus 1 times to this, but then at the end one capital B will come here, but then you will again have n minus 2 c b‘s. So, after applying sayrule three n minus 1 times,what you will get is a power n bb cB power n minus2 c.Now, again to this n minus 2 c b(s) youhave to apply the rule again,that will end up with n minus 3 cB in the middle you have to apply again andso on.

So, ultimately when you geta power n bb,this is again star b powern minus 1 c power n when you get this,how many times you would have applied rule three?You would have applied itn minus1 plus n minus 2 plus plus plusplus one times.What is this value? n minus 1 inton by,so you would have applied rule threeso many times for example, when n is equal to1 you do not apply that rule at all, when n is equal to 2 you are applyingn minus inton by 2 is 1,you are applying rule threeonly once.

When n is equal to 3,what is n minus 1 into n by 2? 2 into 3 by 2 that is 3, 3 times we are applying check,we are applying 3 times. So, when you want to generatea power n, b power n, c power n, first you have to apply rule one n minus 1 times to get a string of this form,then this is equivalent to this and you have toyou get this, and then you apply rule two once to get the abc,to get rid of that is S you have to use rule two.Then when you get a string of this form, you have to bring all the capital b(s) to the left and small c(s) to the right. So, you have to apply the rule cB goes to B c repeatedly. So, first time you can use n minus 1 times, then again n minus 2 times andso on,so you will be using that rule n minus 1 into n by 2 times.When you do that you get a string of this formthen finally, rule four is used to convert all the capital b(s),which is a non terminal capital B is a non terminal,those b(s) have to be converted into small b(s). So, how many times you will be using rule four?Rule four will beusednminus 1time,so ultimately you get. So, you must realize that, if you have a c you cannotc to the left of b you cannot convert the capital B into a small b.

Only when the b get shifted in the middle towards the b, the capital B gets converted into a small b. So, you may sure that all the b(s) are shifted, if you write b goes to b in the middle, somehow the b(s) you can replace in the middle also that you do not want. So, you want to convert the capital b(s) into small b(s) only when they are shifted to the middle,that is why this rule is there.This rule makes sure that all the capital b(s) are shifted to the left, then you are converting them into small b’s.

(Refer Slide Time: 14:59)



$\{a^n b^n / n \geq 1\}$ type 2

$\{a^n b^n c^n / n \geq 1\}$ type 1

$\{a^n / n \geq 1\}$

$N = \{S\}$ $T = \{a\}$

$P:$ 1. $S \to aS$

2. $S \to a$

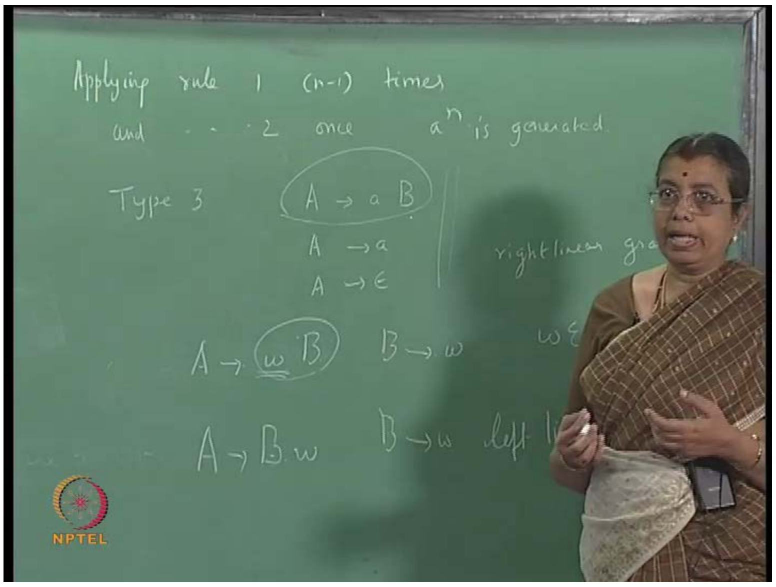
$S \overset{2}{\Rightarrow} a$ $a \in L(G)$

$S \overset{1}{\Rightarrow} aS \overset{2}{\Rightarrow} aa$ $a^2 \in L(G)$

So, this is an very simple example of a context sensitive language or a type one language.Now, we have seenthe grammar for a power n, b power n, we gave a grammar in the last lecturefor thiscontext; this is type two we gave.And a power n, b power n, c power n,today we have considera type one grammar. I will consider one more a power n,n greater than or equal to 1.What grammar can you give forthis?

Consider this grammar with S as a non terminal; only non terminal,and the only terminal is aand the rules are of the form, S goes to a S,S goes to a.There are only two rules; S goes to a S, S goes to a. So, with this we can see that applying the second rule aloneyou generate a,applying rule oneand then applying rule two you get a a andso on, aa belongs to l g.

(Refer Slide Time: 16:29)


Applying rule 1 (n-1) times
and . . . 2 once a^n is generated
Type 3 $A \to aB$
$A \to a$
$A \to \epsilon$
right linear gra
$A \to wB$ $B \to w$ $w \in$
$A \to Bw$ $B \to w$ left l

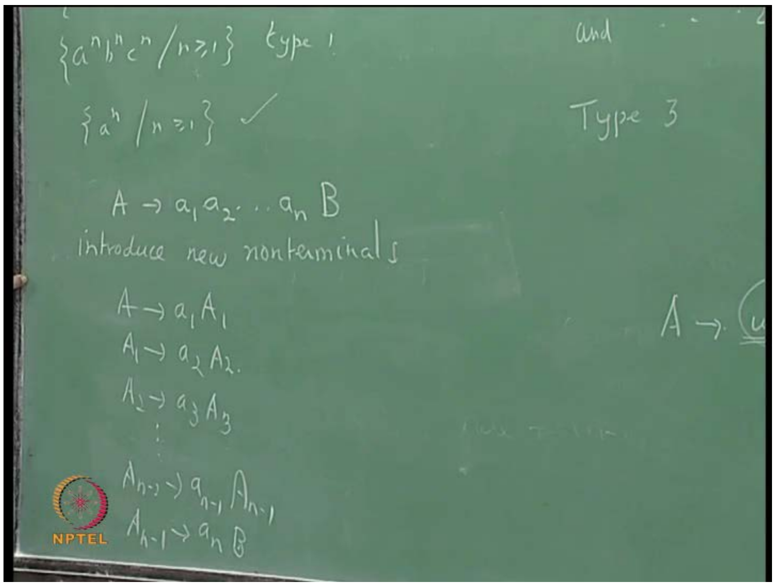
Similarly, applying rule one n minus 1 times,applyingrule 1 n minus 1 timesandrule 2 once you will get a power n. So, the language is this,the language is thiswhat type of a grammar is this?This is regular grammar or a type three grammar.Actually, you will find that there are slight difference definitions given for type three. So, the way we defined this;type three wedefined is the rules are of the forma goes to a B something like this, left hand side non terminal right hand side as terminal symbol followed by a non terminal,or like this a non terminal going into a terminal, or you can also have something like a equals to epsilon.Sometimes for example, in this one instead ofthisif I have epsilon,what will be the language generated?

The empty string

The empty string also will be generated. So, you will get the language as a power n,n greater than or equal to 0.Sometimes, the definition of type three is given like this, on the left hand side you have a non terminal,on the right hand sidea terminal string(( ))by a non terminal or a non terminal going into aterminal string. So, right hand side you can have a terminal string it is a string; w is a string,w belongs tot starw belongs to t star and b goes to w, that is on the right hand side you can have a terminal string followed by a non terminal or just a terminal string,it can be empty string also.

Now, the point to be noted is, this capital B which is a non terminal occurs there is only one non terminal on the right hand side,and that is the last symbol.This definition is also considered as right linear grammar.rightlinear grammarEven though you write like this, these two definitions are equivalent,what is the reason?Suppose, I have a rule of this form,

(Refer Slide Time: 19:59)



$\{a^n b^n c^n / n \geq 1\}$ type 1

and

$\{a^n / n \geq 1\}$ ✓

Type 3

$A \rightarrow a_1 a_2 \cdots a_n B$

introduce new nonterminals

$A \rightarrow a_1 A_1$

$A_1 \rightarrow a_2 A_2.$

$A_2 \rightarrow a_3 A_3$

$\vdots$

$A_{n-2} \rightarrow a_{n-1} A_{n-1}$

$A_{n-1} \rightarrow a_n B$

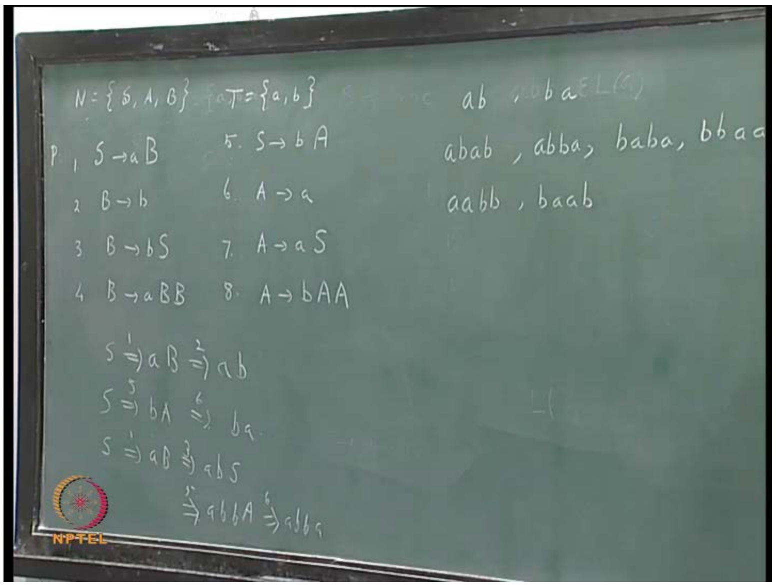
$A \rightarrow$

I have a rule of the form say a 1 a2a n B, this is a string instead of a single symbol, all are terminal symbols, a 1 a 2 a n are all terminal symbols.If I have a rule like this,I can split this into several rules of this type.How is that possible? I can introducenew non terminals,introducenew non terminals;A going intosay a 1 A 1,A1 goes toa 2 A2,A2 goes to a 3 A3,like thatup toA n minus 2 goes to a n minus 1 A n minus 1, then A n minus 1 goes toa n B.Now, each oneof this rule is of that form, left hand side you have a single non terminal,on the right hand side you have a single terminal followed by a single non terminal.

You are introducing new non terminals A1 A2 A 3 A n minus1. So, the application of a single rule of this form is equivalent to applying a sequence of rules like this.If you apply this first, then this, then this, then this andso on,the effect will be the same as applying this rule. So, this rule in essence you can split in this form,so even though you considered this definitionor this definition there are equivalent, whichever way you consider it is immaterial.Now, as forthe present we will keep the definition like this, does not make much of difference if you consider like this also, or equivalently you can also have in this form,this is also will generate the same type of languageBw,B goes to w.

The only non terminal on the right hand side, it can occur at the last symbol or it can occur as a first symbol,such a grammar is known as a left linear grammar.left linear grammarAgain, this also generates the same type three language, but you must not mix up this and this.In the same grammar, either you should haveall the rules left linear or all the rules linear, if you have one left linear and one right linear that no longer it can generate higher class. So, thesegrammars generate what are known as regular sets,language generated is called aregular set.The regular set is also any regular set is accepted by a final state automaton, the equivalent machine characterization isFSA.

(Refer Slide Time: 24:16)



$N = \{S, A, B\}$ $T = \{a, b\}$

P. 1. $S \to aB$ 5. $S \to bA$

2. $B \to b$ 6. $A \to a$

3. $B \to bS$ 7. $A \to aS$

4. $B \to aBB$ 8. $A \to bAA$

$S \overset{1}{\Rightarrow} aB \overset{2}{\Rightarrow} ab$

$S \overset{5}{\Rightarrow} bA \overset{6}{\Rightarrow} ba$

$S \overset{1}{\Rightarrow} aB \overset{3}{\Rightarrow} abS$

$\overset{5}{\Rightarrow} abbA \overset{6}{\Rightarrow} abba$

ab , ba $\in L(G)$

abab , abba, baba, bbaa

aabb , baab

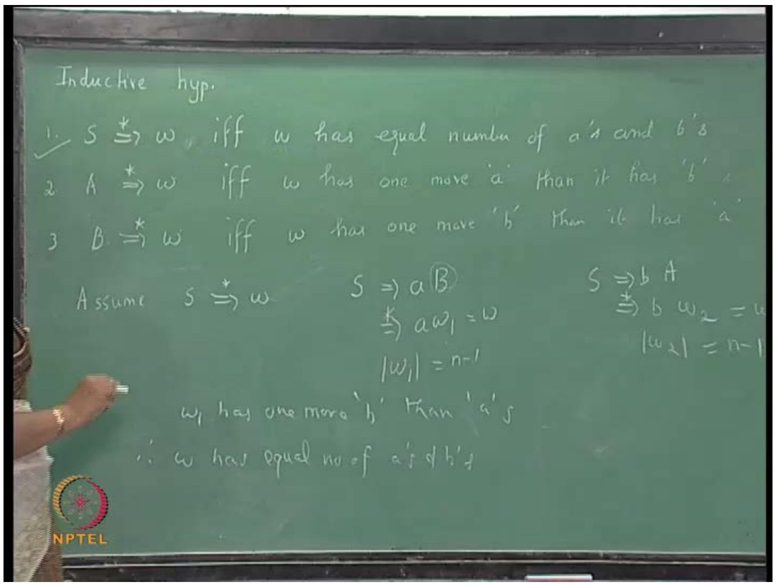
So, let us consider one more example.The non terminals there are three non terminals now,another grammar we will consider; this is of type two.The terminal symbols area and b.There are eight productions;the production consists of S is the start symbol, and the production symbols,production rules are like this; S goes to a B, Bgoes tob, Bgoes tob S,B goes to aB B.S goes tob A,A goes to a,A goes to a S,A goes tob A A.S is the start symbol, there are eight rules; two terminal symbols a and b, and three non terminals S capital a capital B.Try to generate a few strings in the language,what are the strings generated?Let us try to generate some strings S goes to a B rule one, thenrule two you will get a b,S goes tob A rule five,rule six you can generate a B you can generateb A.Then you can also generate S goes to a B rule one, rule three if you use a bS,then rule five a b b A,rule six a bb a.You can verify that length two abcan be generated, bacan be generated,length four ab abcan be generated, ab bacan be generatedba ba can be generatedbb aacan be generated, aa bb can be generated,aa bb bb aa ab baba ab, ba abalso can be generated, baba ab ab.

In general, you will find that you cannot derive strings of odd length, you will be able to derive only strings of even length from S,and also that will have equal number of,any string generated in this grammar will have equal number of a(s) and b(s). It is not very obvious,see looking at the rules that is not very obvious, is not it?So, we will have to prove thatright we have to prove.

So in general, what your purpose will be given a new language, you want to design a new language; programming language and construct the compiler means, first step will be to generate the grammar.You areframing the grammar, but then you must show thatthis grammar generates all syntactically correct programmes in theprogramming language you are designing,and all syntactically correct programmes can be generated by the grammar, that is the grammar generates all syntactically correct programmes inthe language,and only those not anything else.

So, that way you have to prove, then only you can proceed for constructing the compiler. So, this is for example, here we know that the language generated hasany string which has got equal number of a(s) and b(s),that is not very obvious from the grammar, isn't it?

(Refer Slide Time: 29:11)



Inductive hyp.

1. $S \overset{*}{\Rightarrow} w$, iff w has equal number of a's and b's

2 $A \overset{*}{\Rightarrow} w$ iff w has one more 'a' than it has 'b'

3 $B \overset{*}{\Rightarrow} w$ iff w has one more 'b' than it has 'a'

Assume $S \overset{*}{\Rightarrow} w$

$S \Rightarrow a\boxed{B}$

$\overset{*}{\Rightarrow} aw_1 = w$

$|w_1| = n-1$

w_1 has one more 'b' than 'a's

$\therefore$ w has equal no of a's & b's

$S \Rightarrow bA$

$\overset{*}{\Rightarrow} bw_2 = w$

$|w_2| = n-1$

So, you have to prove that,how do you prove?We will prove by induction,so the inductive hypothesis is this.If a string is derivable from S in this grammar,thenor if and only ifw hasequalnumber of a(s) and b(s),but this is the one we want to prove, but in order to prove that we need to prove something more also.

String is derivable from thenon terminal a, if and only ifw hasone moreathanit has b(s). Then another step would bea string is derivablefromB;aterminal string,w is a terminal string.If and only if w hasone more b,than it has a(s),so this is the inductive hypothesis. So, taking the basis classinduction is on the length of the string,you must note that there are two things,if every statement has if and only if,so you have to prove in both directions.

So, for induction baseswe can note that no string of length zero epsilon, empty string cannot be derived from any non terminal here, a bor S.When the length is one from a you can derive a,it has one a and zero b(s),so the induction hypothesis is satisfied.From b you can derive a B which has got one more b than it has a(s),so the induction hypothesis is satisfied.From S you have seen that we can derive a borb a,that is a smallest string you can derive from S,and that has one a and one b,so it has got equal number of a(s) and b’s.

So, the bases class for n is equal to1,n is equal to 2 you have checked,so the basis class holds.Then the induction class, we have to prove the induction portion,we will just prove this first statement alone,the second and the third can be proved in a similar manner, just a matter of details only. So, let us take the first one only, we will assume that the inductive hypothesis istrue up to n minus 1. So, assume upton minus 1,upton minus 1 this n is the length of the string,length of the stringinduction is on the length of the string.

Assume up to n minus 1, that assumption is all the three you are assuming upto n minus 1,prove for n. So, if you have a string wbelongs to t star and the length of w isn,you have to prove that S derives w if and only ifw has equal number of a(s) and b(s). So, there are two parts, if S derives wthen w hasequalnumber of a(s) and b(s). And the other way round, w hasequalnumber of a(s) and b(s) thenit is possible to derivew from S,you have to prove two ways on two directions.

If it is possible to derive w from S then w has equal number of a(s) and b(s),and if you have a string w which is having equal number of a(s) and b(s),it should be possible to derive that from w,both ways you have to prove.Let us see how to go about this?First,assumefrom S w is derivable.So,what you do is, from S if you start from S you have to use rule number 1 or 5.

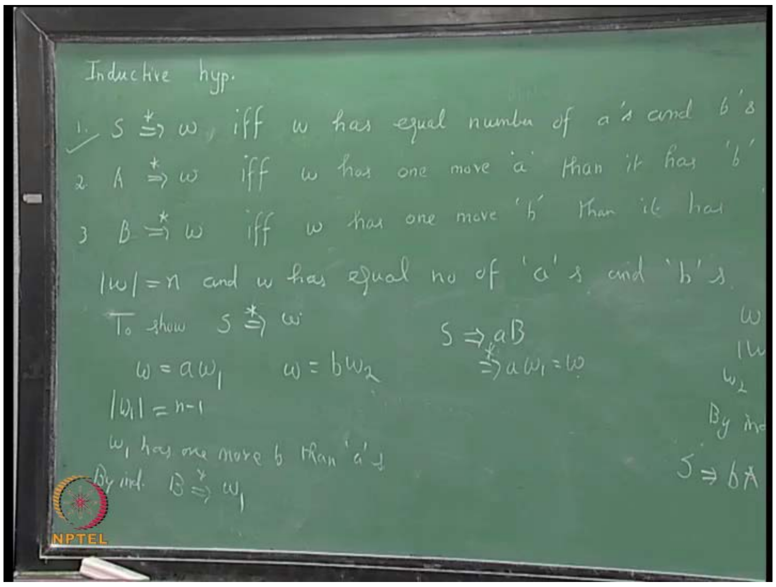
If we use rule number one the first step will be like this,if we use rule number five the first step will belike this.Thenfrom byou are deriving a string w 1 in many steps,and you are getting a string a w 1 which is equal to w. So, w 1 is derived from b,and what you can say about the length of w 1?Length of w is n,so what can a is taken away,so what can you say about the length of w 1?n minus 1.

And the induction hypothesis holds for all strings upto n minus 1,that is strong induction.All strings upto n minus 1it holds. So, if w 1 S n length of w 1isn minus 1and it is derivable from b, what can you say about this?A string w is derivable from b, if and only ifit has more one more b then it has a(s).

So, w 1 hasone morebthan a(s).So, along with this a what can you say about w? w will have equal number of a(s) therefore, w hasequalnumber of a(s) andb(s).This one if you take S goes to b A the firstrole is like this, then from a you are deriving a string w 2,what can you say about the length of w 2?This is equal to w;length of w is n,so length of w 2 is n minus 1 and w 2 is derived from A.

So, by the induction hypothesis what can you say?A string is derivable from A,if and onlyif it has one more A then it has b(s),so w 2 hasone moreathan b(s). So, counting along with this b w hasequalnumber of a(s) andb(s). So, this is one portion, what we have proved is this, if a string is derivable from S then it has equal number of a(s) and b(s). The other way around we have to prove, if you have a string which has got equal number of a(s) and b(s),it should be possible to derive that from S,that way also you have to prove.

(Refer Slide Time: 39:33)



Inductive hyp.

1. $S \overset{*}{\Rightarrow} w$, iff w has equal number of a's and b's

2. $A \overset{*}{\Rightarrow} w$ iff w has one more 'a' than it has 'b'

3 $B \overset{*}{\Rightarrow} w$ iff w has one more 'b' than it has

$|w| = n$ and w has equal no of 'a's and 'b's

To show $S \overset{*}{\Rightarrow} w$

$w = aw_1$ $\quad w = bw_2$

$|w_1| = n-1$

w_1 has one more b than 'a's

By ind. $B \overset{*}{\Rightarrow} w_1$

$S \Rightarrow aB$

$\overset{*}{\Rightarrow} aw_1 = w$

w

$|w$

w_2

By ind

$S \Rightarrow bA$

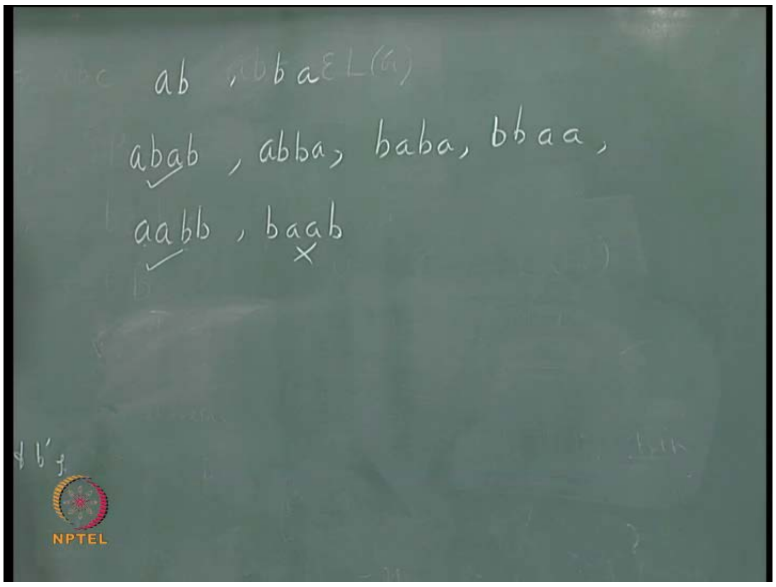
So, how do you go about proving that?Assume, that length of w is n,andw hasequalnumber of a(s) andb(s).Now, you have to showit is possible to derivew from S, but to show that it is possible to derive w from S.Now, there are two possibilities;w is a string of a(s) and b(s) there are two possibilities; the first symbol can be a or the first symbol can be a b. So, it is possible that it is a w 1 oryou can write w as b w 2, and what can you say about w 1?Length of w 1 will be n minus 1, and w 1has w has equal number of a(s) and b(s),1 a is taken out. So, w 1 hasone morebthan a(s).So, by the induction hypothesis it should be possible to derive w 1 from b. So, by inductive hypothesisit should be possible to derivew 1 from b, but we have a rulewe have a rule S goes to a B,we have a first rule is this. So, apply this rulethen from byou derive w 1,that is from S you can derive w. So, if w has equal number of a(s) and b(s) and assuming that it begins with the a,it is possible to derive w from S,that is what we have proved now.

Now, second case take w is equal to bw 2,so what can you say about the length of w 2?The length of w 2 is n minus 1,and w 2 will have one more a than it has b(s). w 2 hasone moreathan b(s). So, by the induction hypothesis what do you get?It is possible to derive w from a if it has got one more a than it has b(s). So, by induction hypothesisby inductionhypothesisit should be possible to derive w 2 from a.

So, but we know that there is a rule of the form S goes to b A rule number five, S goes to b A is there.Apply this first, then from this ayou can derive w 2, but b w 2 is nothing but

w. So, it is possible to derive w from S. So, the language generated consists of equal number of a(s),any string having equal number of a(s) and b(s) will be generated by this grammar.What is the difference between this and the Dyck set which we considerin the last class.Dyck set is the well formed strings of parenthesis,

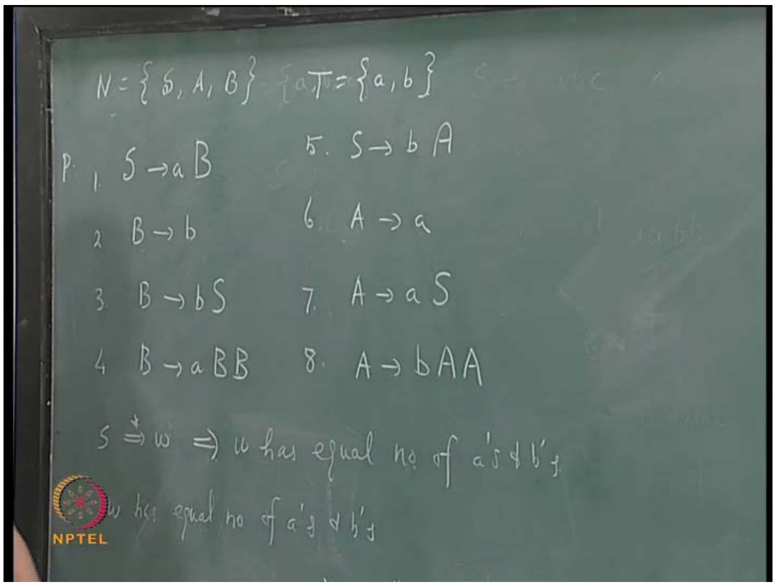
(Refer Slide Time: 44:06)



so for example,if you look at a as a left parenthesis,b as the right parenthesis,this will be belong to the language, this will belong to the language, but this is this will not, because this will be right left left rightlike that,which is not a well formed string. So, there is a difference, here any string will have equal number of a(s) and b(s).

Dyck set also, any string will have equal number of a(s) and b(s),but there is some more restriction, what is that restriction?Left should come first I mean; the well formed string parenthesis means,left parenthesis should come before the matching right parenthesis.Now, one more point to note is,can this grammargenerate a empty string?This grammar cannot generate the empty string.

(Refer Slide Time: 45:10)



$N = \{S, A, B\}$ $\{a$ $T = \{a, b\}$

P 1. $S \to aB$ 5. $S \to bA$

2 $B \to b$ 6. $A \to a$

3 $B \to bS$ 7. $A \to aS$

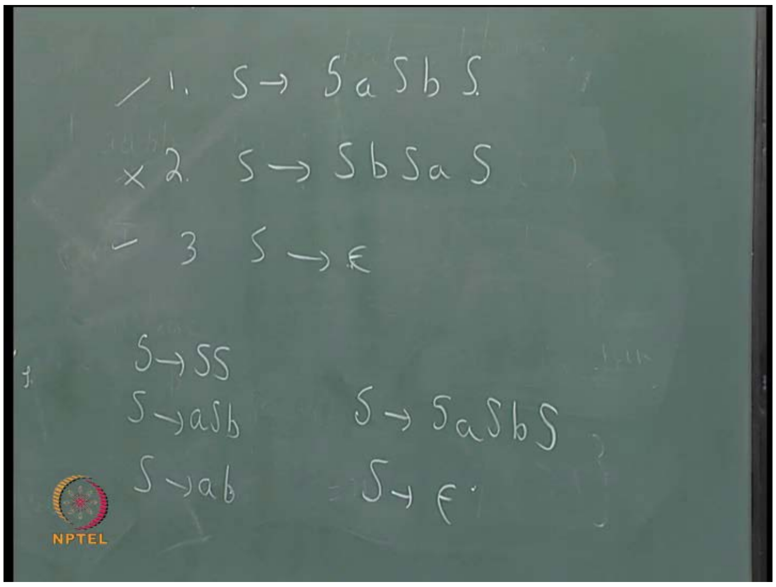
4 $B \to aBB$ 8. $A \to bAA$

$S \overset{*}{\Rightarrow} w \Rightarrow w$ has equal no of a's & b's

w has equal no of a's & b's

The smallest string generated by the grammar is of length two a borb a.aalone can derive a, bcan derive b, but from S the smallest string derivable is of length two a B (( ))

(Refer Slide Time: 45:32)



✓ 1. $S \to SaSbS$

× 2. $S \to SbSaS$

✓ 3 $S \to \epsilon$

$S \to SS$

$S \to aSb$ $S \to SaSbS$

$S \to ab$ $S \to \epsilon$

Now, I will write another grammar, look at the rules.There is only one non terminal S,terminals area and b,the rules there are three rules S goes to S a S b S,S goes to S b S a S,S goes to epsilon,a set of reductions consists of these three.Here the empty string can be generated,the empty string can be generated here, but you can see that whenever i apply rule,1 a and 1 b will be generated.

Whenever, I apply rule two1band 1a will be generated, but ultimately any string generated will have equal number of a(s) and b(s).In between some of the s’s we can make it go to epsilon. So, it isvery obvious that any string generated by the grammar will have equal number of a(s) and b(s), but you have to prove the converse also.Any string having equal number of a(s) and b(s) will be generated by this grammar,that is not as easy, that is not very trivial,one direction it is very easy to see.Any string derivable in thisgrammar will have equal number of a(s) and b(s), but if the language consists ofall strings having equal number of a(s) and b(s) you must also prove that, any string having equal number of a(s) and b(s) will be generated by this grammar, you can try this that is not verytrivial step, you have to prove that.

Now, the dyck set the grammar which we gave was this, S goes to S S, S goes toa S b, S goes to a b.Here again note that epsilon cannot be generated, the empty string cannot be generated.An equivalent grammar which will also generate the empty string is this,that is in this grammar leave out this have these two rules alone.This again will generate only well formed strings of parenthesis, but in this case epsilon is generated, in this case epsilon is not generated.Similarly, these twogenerate the same language apart from epsilon, here epsilon will be generated, epsilon will not be generated in the other one. So, these are some of theexamples which we consider for type two and type three, and type one also we have considered one more examples.In the next lecture we will see some more example and alsoabout ambiguity.By the way, you can try to prove that the other converse of it, that is any stringhaving equal number of a(s) and b, a(s) and b(s) will be generated by this grammar.