Theory of Computation

Prof. Kamala Krithivasan

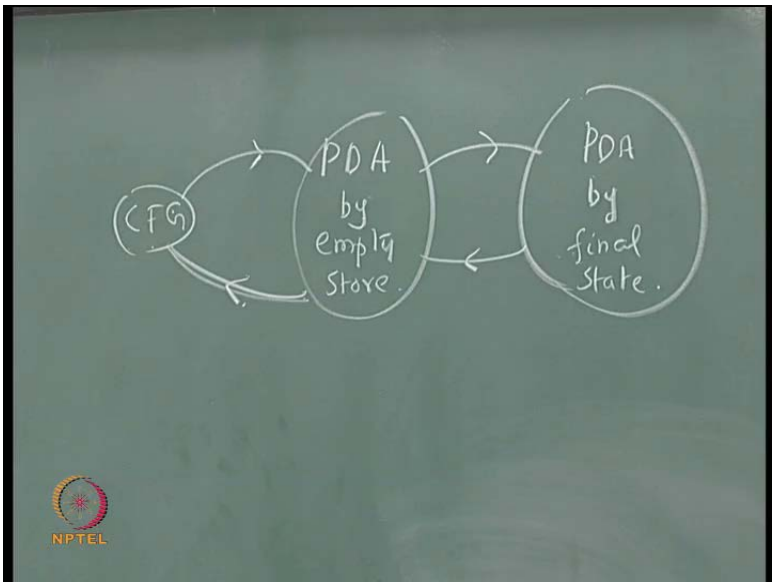
Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Lecture No. # 23

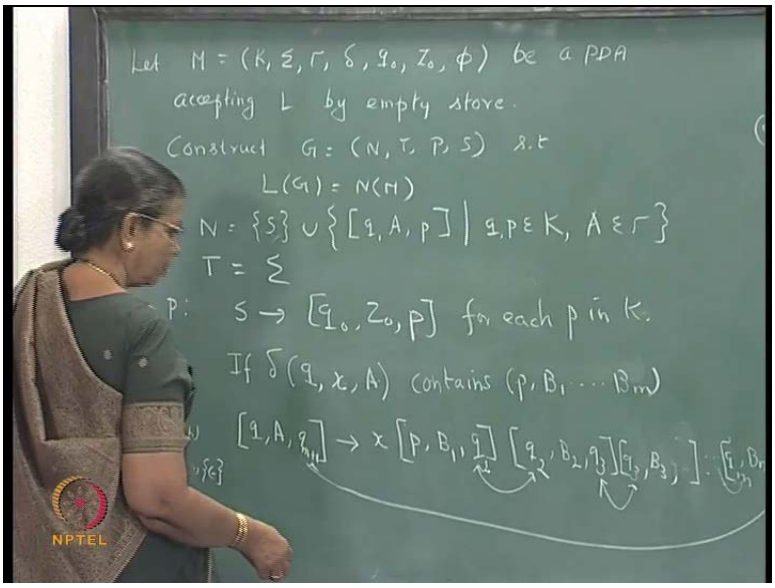
Pushdown Automata PDA to CFG

(Refer Slide Time: 00:16)



We have been discussing pushdown automaton, and we have seen that PDA acceptance of PDA by empty store, and acceptance by PDA by final state; both we considered and we showed the equivalence between the two that is given, this we can construct an equivalent PDA in this, and given a PDA by final state; you can construct equivalent PDA by empty store. And to show the equivalence between CFG, we have seen that given CFG, how to construct the PDA by empty store in the last lecture. Now, in this is what we have to study today, given a PDA by empty store. How to construct an equivalent context free grammar?

(Refer Slide Time: 01:17)



Let M is equal to k, sigma, gamma, delta, q naught, Z naught, phi be PDA, accepting a language. Accepting L a language context free language L by empty store construct G is equal to N, T, P, S, such that L of G the is equal to N of M. The language generated by G is a same as the language accepted by M by empty store. Now, how do we construct G? Now, in this case we have to specify the non-terminals, terminals productions etcetera. Non-terminals will be you have the start symbol plus the non-terminals are triples they are taken as triples q, A, p.

It they are triples of this form where q and p belong to the set of states first and third component are states in the pushdown automaton and the middle component is a pushdown symbol. So, you have to consider all triples. First one can be any state third can be any state then middle one can be any pushdown down symbol. So, if there are K states, what will be the number of non-terminal, if there are K states and m pushdown symbols what will be the number of non-terminals? K into all triples we have to consider.

(( ))

K squared m

(( ))

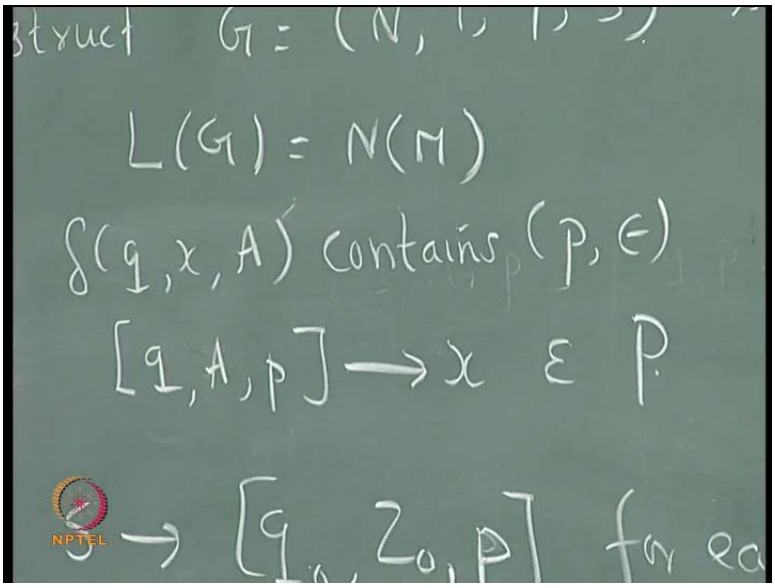
Plus 1, this is K into M into K squared m plus 1. The number of non-terminals will be (( )) what is T, what is a set of terminals same as sigma, same of input symbols. Now, P we have to the productions rules we have to define S, this S is the start symbol a 1 S you have taken that is the start symbol, rules in P are written like this S goes to q naught, Z naught, q naught is initial state Z naught is the initial pushdown symbol and P for each P in K.

Start symbol going in to a non-terminal this a unit production. Please remember that, this we are writing unit production this is to accommodate see for all P. See, when the store is emptied we are not bother about the state in which the machine is is not it when the store is emptied. The string will be accepted, but we are not worried about the state of the pushdown automaton. So, any it can be in any state so, allowing for all possibilities for each P we have role like this right. Then the rules we have to write from the mappings.

So, if delta of q x or may be A is better, if there is a rule of this form I will write x because A usually symbol x. Here x can be a symbol or Epsilon x can be a symbol or Epsilon. Then, corresponding to this you will write a rule of the form q, A, x, P, B 1. The triples I am leaving, but I will fill later B 2, B 3 up to B m you will have a collection of rules of this form P contains rules of this form. Now, I have purposefully left out certain things, the rule the mapping is q, x, A contains P, B 1, B 2, q is there, x is there A is there, P is there, B 1, B 2, B m are what about the other things.

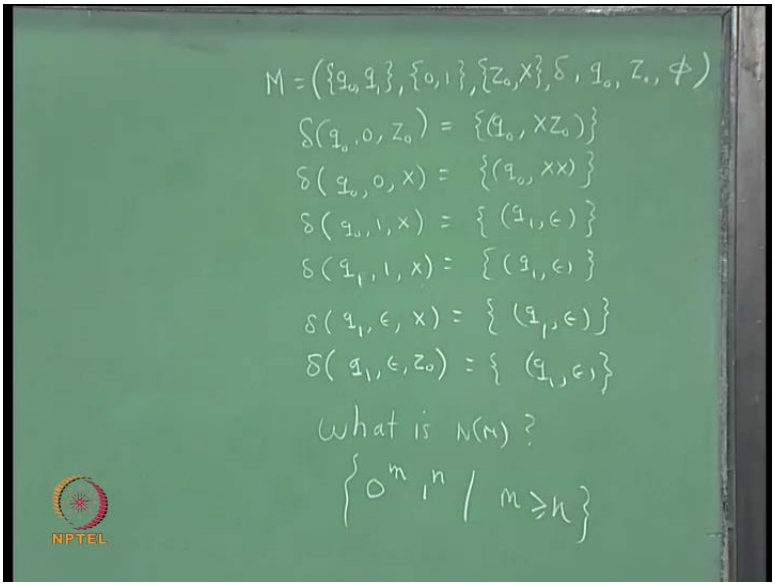
Now, you see that x can be can be a input symbol or Epsilon x can be a input it is an it is an Epsilon moves means x will be Epsilon. If we start true input to move means x will be a symbol from sigma. So, you have q, A goes to x, P, B 1, B 2, B middle components I have written I should have enough space. How do you fill the other components? So, P, B 1, q 2, q 2, B 2, q 3, q 3, B 3 like that cut, last you have q m, B m, q m plus 1. You can fill like this that is you can fill with this portion here with any state, if there are K state you can use any of them. So, for one mapping you will have a collection rules, but whatever you write here you must here the next one, the third component of this will be the first component of this the third component of will be the first component of this and so on. And last one, this will be q m previous one will be q third component of the previous non-terminal will be q m, last you have written q m plus 1 that is here q m, what you write here should be same as this.

(Refer Slide Time: 10:06)



So, for one mapping there will be a collection of rules. Suppose the rule is of this form the rule is of this form delta of q, x, A contains P Epsilon that is m is 0, in this one m is not equal to 0. You have rules of this form. If m is equal to 0, there mapping will be of this form is it not? q, x, A contains p Epsilon then the rule will be q, A, p goes to x is it not? P the rule will be q, A, p goes to x is in p. Now, we have to give the proof that this construction is such that L G is equal to N of M.

(Refer Slide Time: 11:18)



Now, we will prove that by induction in both directions but before that, let us consider an example take pushdown automaton with has got two states and input symbols are 0 and 1 pushdown symbols are Z naught and X, q naught is the initial state i Z naught t is the initial pushdown symbol and the mappings are given by this. It is a acceptance by empty store the mappings are given in this manner. Now, look at the mappings and tell me what is N of m, we will write the grammar for this, but before that what is N of m you have to start with q naught and Z naught or I will give you few minutes to tell me what is N of m.(no audio from 12:12 to 13:10)

(( ))

0 power m

(( ))

0 power m 1 power

M r (( ))

What does the first mapping say, if you have 0 and does it not you add x then this says whenever you see a 0 you keep on adding x's. So, as long as there are zeros you will be adding x's then once you see a 1 you go to q 1 and start removing. Now, when you read a 1 what does this mean, when you read a 1 you remove x, but without reading one also you can remove x, this is an Epsilon move which tells without reading m also one also you remove x so, what does that mean.

(( ))

(( ))

0 power m 1 power n which is greater

(( ))

m greater than r equal

(( ))

Right

(( ))

More zeros than once is this a deterministic machine or a nondeterministic machine is this PDA a deterministic one or a nondeterministic.

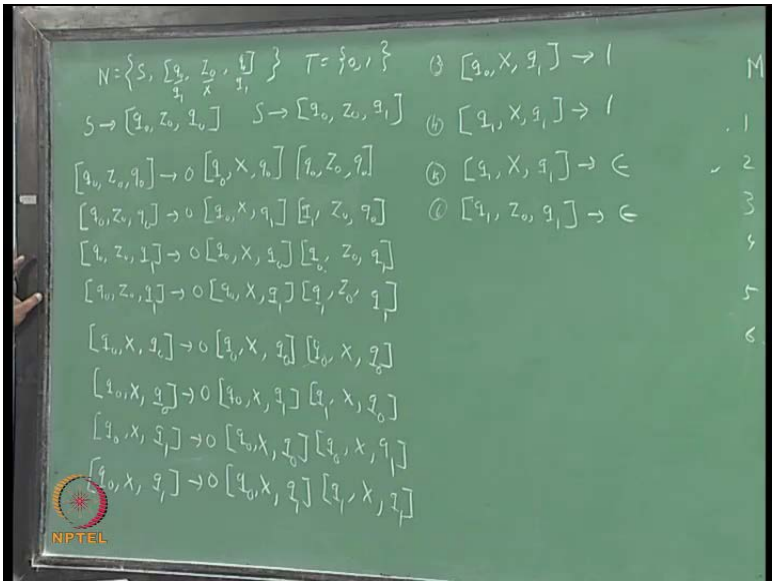
Non[deterministic]-

Non-deterministic, why?

(( ))

With q 1 and x you can use a true input move or an Epsilon move so, the machine is nondeterministic even though there is a singleton here, it is nondeterministic.

(Refer Slide Time: 15:42)



So, let us write the grammar for this by this method, what are the non-terminals? non-terminals are S then you have triples of the form first one can be a q naught or a q 1 the middle one can be a q is Z naught or X third can be a q naught or q 1. So, 8 triples we can have isn't it you have triples the first one can be a q naught or a q 1 second one can be X or Z naught, third one can be q naught or q 1.

So, 8 plus 1, 9 non-terminals. T is equal to 0,1. S is the start symbol and we have to write the rules now with S you have two rules q naught, Z naught, q naught see the first two are fixed q naught and Z naught third one can be a q naught or q 1.

So, q naught, Z naught, q naught, S goes to q naught, z naught, q 1 then I have to write the rules for the this first, let me write the rules for these four which is simple q naught, 1 x is equal to q 1 Epsilon, what will be the corresponding rule for this I will write 4, sorry no it is 3 1, 2, 3, 4, 5, 6. 6 mappings are there, right? Corresponding to three mapping is q naught, delta of q naught. 1 X contains q 1 Epsilon so, you will have q naught, X, q 1 goes to one it will be a terminal rule. Non-terminal going into a terminal rule then corresponding to 4 what will be the rule q 1, 1 X goes to q 1 Epsilon. So, you will have q 1, x, q 1 goes to what q 1, 1 X goes to 1 in a corresponding to 5 q 1 Epsilon X goes to q 1 Epsilon.

So, the rule for that will be q 1, X, q 1 goes to Epsilon and corresponding the 6th mapping delta of q 1 Epsilon Z naught contains q 1, Epsilon the map rule for that will be q 1, Z naught, q 1 goes to Epsilon, this terminal rules are easy to write. Now, the other rules for 1st and 2 we have to write the 1st one is this delta of q naught, naught, Z naught contains q naught, t X, Z naught.

So, for that what you will have is q naught, Z naught, 0, X, Z naught, q naught here then rules have to be of this form now, what I write here I should write here and what I write here I should write here right and again either write q naught or q 1 both possibilities should be taken into account. So, how many rules will be there, there will be two possibilities here there will be two possibilities here.

So, four possibilities will be there so, let us write the four possibilities q naught, Z naught, 0, q naught, X, Z naught, q naught, Z naught, 0, q naught, X, Z naught, q naught, Z naught, 0. So, I have just written the same thing four times now, the possibilities are I can write q naught here and q naught here, q naught here and q naught here. I can write q naught here, q naught here, q 1 here, q 1 here I can write q 1 here, q 1 here q naught here, q naught here, q 1 here, q1here, q1 here is the other what I write here, I am writing here, what I write here, I write here.

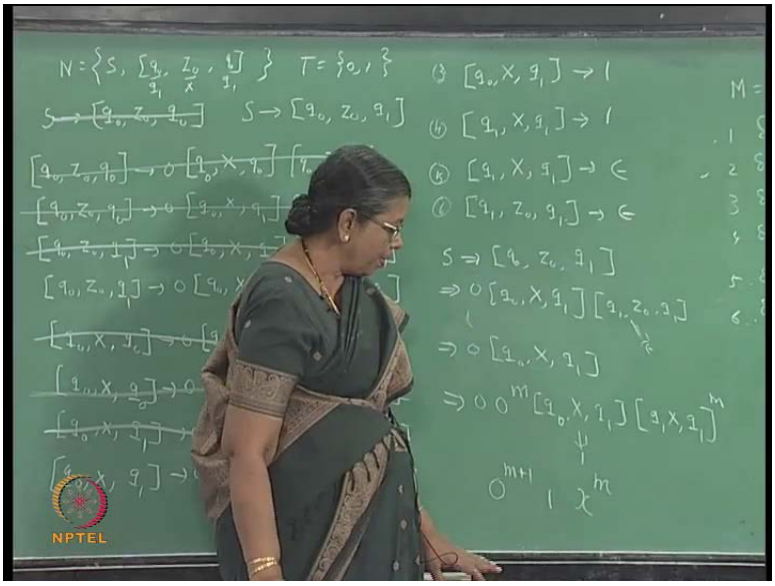
Now, let us consider this one the mapping is delta of q naught, 0, x contains q naught, xx. So, the rules will be q naught, X goes to 0. X this is q naught the rules will be of this form, here again what I write here I must write here what I write here I must write here I can write a q naught or q 1 both possibilities will be there. So, 4 rules will be there that is q naught, X, q naught X, q naught, X again all the possibilities we have to consider, if I

write q naught here I must write q naught here and if I write q naught here I must write q naught here. That is one possibilities then q naught here, q naught here if I write q 1 here I must write the same q 1 here again q 1,q 1, q naught, q naught, q 1, q 1, q 1 that is all so, how many rules it has

(( ))

4 plus 4 8 plus 2 10 plus 4 14 rule. Now, you can remove the useless non-terminals, you can remove, something has to terminate a non-terminal should you lead you to a terminal string.

(Refer Slide Time: 25:18)



Then you see that there are no terminal or there are no rules with q 1 and q naught like this, if there are no rules like that in fact q 1 with the first component as q 1 and the last component as q naught there are no no rules like that with this on the left hand side.

So, such non-terminals will be useless first component q 1 third component q naught will be useless first component q 1. q 1 this you cannot rewrite it. For the if you apply this rule you cannot rewrite this further right so, this is useless similarly, if you have apply this rule q 1, X, q naught this cannot be rewritten as further because with this symbol on the left there are no rules so, this is also you useless you can remove them.

Now, what about the others if you have q naught, Z naught, q naught. Then you will when you apply this rule you will get q naught, X q naught, then when you apply q

naught, X, q naught again you will get a q naught, X, q naught you can do it again and again, but that q naught, X, q naught there is a rule with this on the left hand side, but when you write again you will get a q naught, X, q naught you will not be able to terminate.

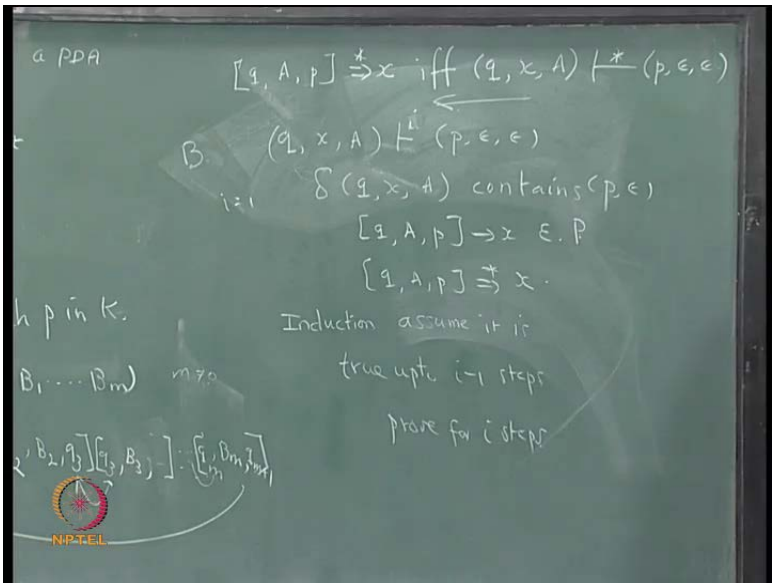
So, such q naught, X, q naught. q naught, Z naught, q naught are also useless they will not lead you to terminal strings q naught, X, q naught. q naught, Z naught, q naught they are also useless because ultimately it will not lead you to a terminal string right. So, such non-terminals and the rules involving them can be removed what are they you can remove this, we can remove this, we can remove this, we can remove this and we can remove this so, ultimately you are left with 4 plus 1 5 6 7 rules.

Now, look at this and see what will be the derivation starting with this you have to apply this rule then with this when you generate 1, 0 and q 1, Z naught, q 1 then this q naught, X, q 1 when you apply you will generate 1, 0 and q 1, X, q 1 right and the derivation will of this form S drives q naught, Z naught, q 1 which derives 0, q naught, X, q 1, q 1, Z naught, q 1 and q 1, Z naught, q 1 goes to Epsilon.

So, this will go to Epsilon so you have 0, q naught, X, q 1 now, for q naught, X, q 1 this is the rule q naught, X, q 1 it will again it is a linear rule q naught, X, q 1 will I should not say linear because this is also non-terminal q naught, X, q 1 generates the recursive rule it generates q naught, X, q 1 on the left it generates q naught and on the right it generates q 1, X, q 1.

So, applying it several times you will get 0, some 0 power n. q 1, X, q 1. 0 power m. q naught, X, q 1 then q 1, X, q 1 power m, it will like that now, this will q naught, X, q 1 goes to 1. So, this will go to 1. q 1, X. q 1 it can go to one or Epsilon, some of them will go to one some of them will go Epsilon so, from this you will generate 0 power m plus 1, 1. And then you will have x power m where x can be 0, x can be 1 or Epsilon this will be (( )) now, x can be 1 or Epsilon so, string is of the form 0 power m plus 1, 1 x for m. So, it will generates the language 0 power m, 1 power n, m greater than or equal n, m n greater than or equal to 1. So, you can see how the construction works we have illustrated the construction with an example.

(Refer Slide Time: 32:07)

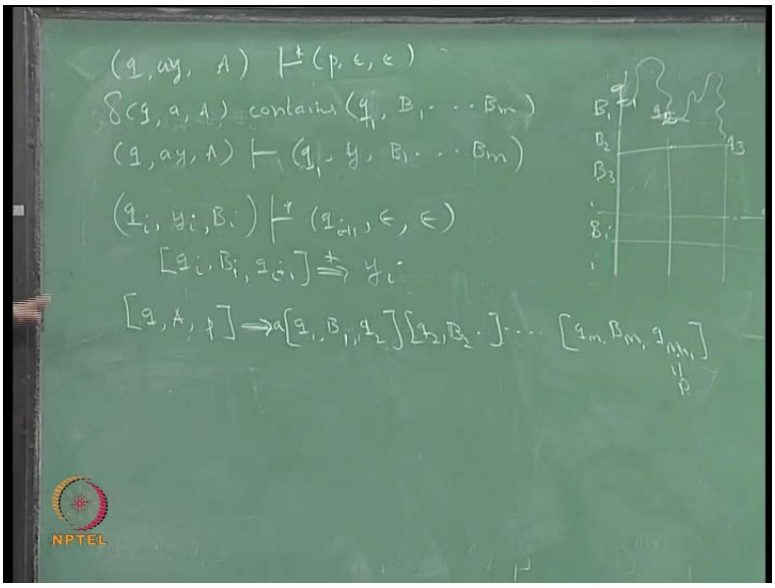


Now, we have to formally give the proof how do we formally proof this, you show that q, A from a non-terminal q, A, p you can derive a string x, if and only if from the configuration q, x, A you can go to the configuration p, Epsilon, Epsilon. So, from this if this is true that is from the non-terminal q, A, p you can derive a string x, if and only if from the ID q, x, A you can go to the ID p, Epsilon, Epsilon this is what we want to prove we have to this if and only if we have prove in two directions.

So, first in this direction again we have to use induction q, x. A derives P, Epsilon, Epsilon in induction on the number of steps so, one step this is in n steps you have to prove basis class will be in one step you derive basis class, if you have this what will be the corresponding rule by our construction. What is our construction? If you can go from this ID to ID that a mapping should be delta of q, x, A contains p, Epsilon then only you would have got this going from one this from this ID to this ID. Now, what is the mapping rule for the if delta of q, x. A contains p, Epsilon then q, A, p goes to x is a rule.

So, you will have q, A, p goes to x belongs to p that is q, A, p derives x so q, A, p derives x this is for i is equal to 1 from this you are going to this 9 steps are something like that then induction portion, assume it is true up to i minus 1 steps, prove of i steps this is strong induction up to i minus 1 steps we assume, the result is true then we want to proof for i steps. How do you prove this?

(Refer Slide Time: 36:18)



Now, we are starting with the ID q, x, A right now, suppose I write x in the form a, y. A can be a true symbol or Epsilon so, instead of x. I will write it as a, y. A can be are Epsilon by the mapping which I will apply will be of this form delta of q, a, A contains p B 1, B 2, B m there will be a mapping like this and we have to apply that mapping first step of the move of pushdown automaton will be something like that isn't it so, from q, a y, A you will go to the ID p, y, B 1, B 2.

This is first step now, if you look at it as this one we have considered even earlier this set of a diagram A was there on the stack right now, A and you are in state q. Now, A has been replaced by B 1, B 2, B m on the stack state has gone to p now, then some moves may take place after some 2 moves take place the first time B 2 becomes the top of the stack you would the state will be say some q 1 or maybe I use q 2, q 2. In between the stack may grow and come back does not matter and B 1 need not be the always here B 1 can be change to something else that is possible, but so for we have not touch the B 2, when B 2 becomes the top of the stack the state is q 2.

Then again now, the stack may grow and come back then when B 3 becomes the top, the stack the state is q 3, but while going from this p to q 2, some portion would have been read, some portion of the input would have been read and again some portion would have been read from the time you start with B 2 on the stack and end with B 3 on the stack as it of symbol.

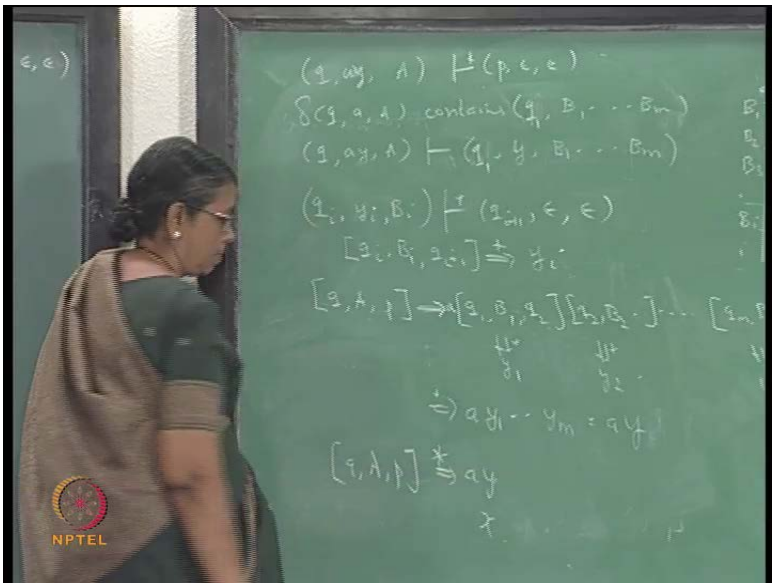
So, ultimately when B m is on the top you will be in state q m and ultimately this also be raised because finally, from this ID you are going to some ID E Epsilon, I need not use the same p instead of p I can say q 1 that is better q 1 if it confuses both p confuses instead of that q 1 I can say q 1 here start so, this is also q 1 in both places I need not use p that is why. So, when you start with say state q i and B i on the stack, B i on the stack and when you read the portion y i this y you can write as y 1, y 2, y m separate it.

So, when you start with q i and read y i with B i on the top you will ultimately you will go to q i plus 1 whole of y i would have been read and now, the when q i plus 1, B i plus 1 becomes the top of the stack. So, this is the situation up to this B i you have read up to some y i minus 1 and you are in state q i now, when you read that y i portion you go to q i plus 1 and B i plus 1 becomes the top of the stack they has been erased B i has been erased, but this is a situation in that case what do you get.

You get q I, B I, q i plus 1 derives y i by the inductive hypothesis this should have happen in a lesser number the whole thing it has taken (( )) from this to this it takes i steps right the first step is this so, the rest of the steps will be i minus 1 so, each one will be must less than i minus 1 and by a inductive hypothesis this will be the situation.

Now, for this mapping what will be the rule q, A I will write p here goes to q 1, B 1, here a, B 1, q 2, q 2 you can have a rule like this, for this mapping by the cell mapping for this mapping you have a rule of the form q, A, p goes to a, q 1, B 1, a, q 2, B 2, q m, B m, q m plus 1, but this q m plus 1 and this should be the same this is actually equal to p.

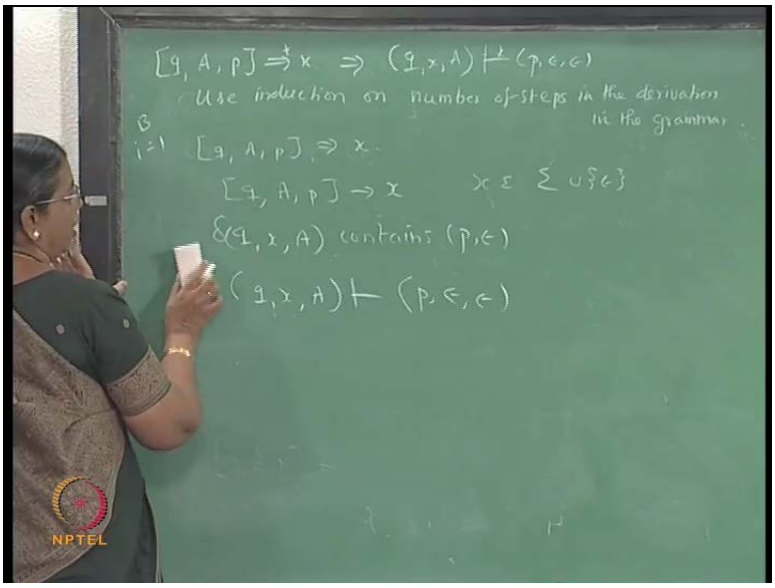
(Refer Slide Time: 43:58)



This is the way we have written the rules. Now, by induction hypothesis this will go to y 1, this will go to y 2, this will go to y m. So, that is a y 1, y m which is a y so q, A, p derives a y or what is a y, a y is nothing, but the x with which we started x. I told you we can write as a y isn't it derives right here also. (( )) So, in one direction we have prove if we can go from this ID, if we can go from this ID to this ID then it is possible to derive x from this non-terminal and we use induction on the number of moves of the PDA.

Now, the other way around we have to prove that is, if this is true then this happens that is if it is possible to derive x from this non-terminal, then it should be possible for the PDA to go from this ID to this ID here we have to use induction on the number of steps in the derivation, we have to use induction on the number of steps in the derivation.

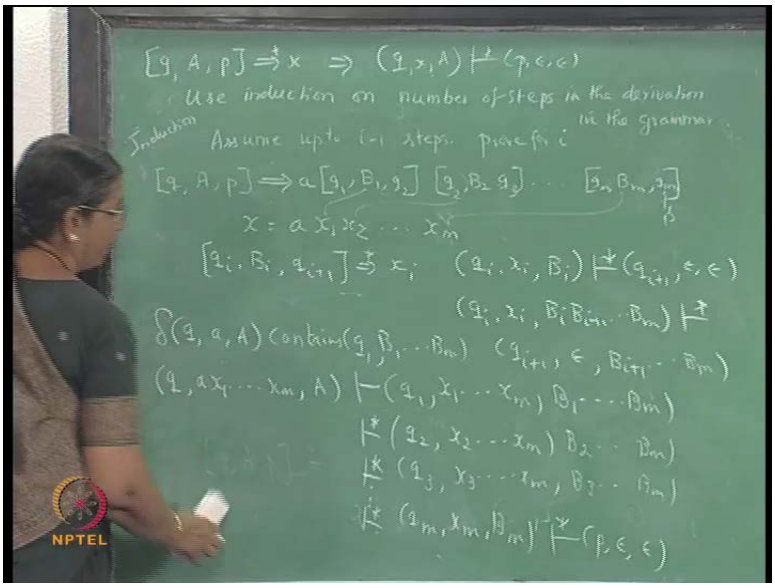
(Refer Slide Time: 46:04)



So, what we want to prove is if from q, A, p we can derive x this implies from q, x, A you can go to p, Epsilon, Epsilon here use induction on number of steps in the derivation in the grammar. Now, suppose q, A, p derives x number of steps is one basis class. In one step you get this then you should have been come from it should have come from the rule this rule right x can be a symbol or a Epsilon x can, x can be x can be a symbol or Epsilon and when is this rule possible it should have come from a mapping delta of q, x, A contains p, Epsilon only when we have mapping like this you would have written this rule.

Now, we are going from the machine to the rule does it only if there had been a mapping like this you would have written this rule. So, if there is a mapping like this what does that mean q, x, A from this ID you go to p, epsilon, epsilon.

(Refer Slide Time: 48:31)



So, for basis class we can prove this, I will rub this off. Assume induction portion assume up to i minus 1 steps, prove for i prove for i steps strong induction.

So, the rules you know are of the form, you are starting with q, A, p then in the first step you will use some rule and get B 1, B 2 the rules are of this form right, first step of the derivation you are using a rule like this where what is this? this q 1, q 2, q 2, q 3, q m, q m plus 1 which is p this and this should be the same.

So, the x is and afterwards this is in one step and afterwards in i minus 1 steps you would have derived a x I am sorry sorry x right so, x really can be written in the form a x 1, x 2 x m. x can be really written in this form where a is here from this you derive x 1 from this you derive x 2 and so on from this you derive x m.

So, you have q I, B I, q i plus 1 derives x i and how many steps in the derivation it would have taken less than i minus 1. Totally everything has taken i minus 1 steps so it will be less so, induction hypothesis holds so what do you get q i, x i, B i from this ID you can go to the ID q i plus 1 Epsilon, Epsilon. That is if I start with q i and x i on the input tape and B i on the top after reading x i it will erase B i suppose, I have something B i plus 1 etc below that is not going to affect anyway. So, if it at erase the before finishing reading x i it cannot complete further right see you need at least one symbol on the top of the stack to read it to make a move so, this also would mean q i, x i, B i, B i plus 1 B m from

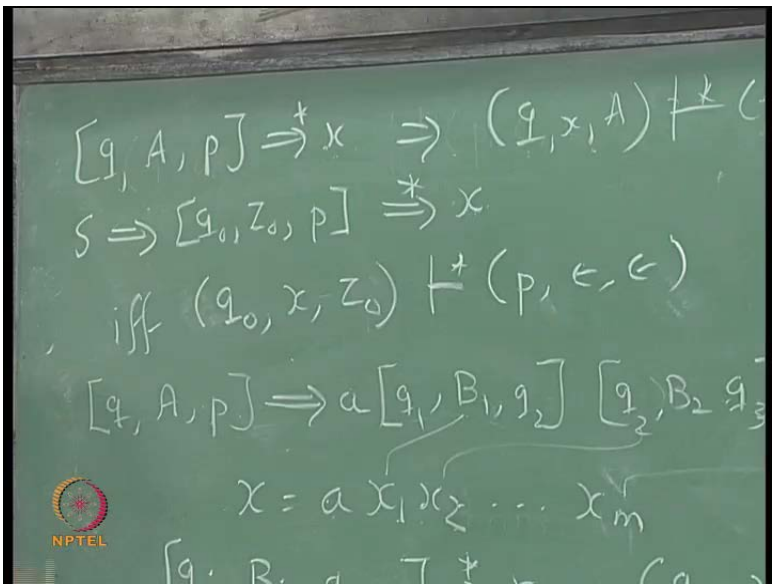
this ID you can go to q i plus 1 B i plus 1 B m am sorry write like this q i plus 1 it would have read Epsilon, Epsilon stack B has been erased B m.

Now, this rule what about the first rule, it should have come from mapping delta of q, a, A contains q 1, B 1, B 2, B m it should have come from a mapping like this. How can you write a rule like this you are applying a rule like this that is a rule is q, A, p goes to a, q 1, B 1, q 2 etcetera and that rule you would have been able to write only if you had a mapping like this.

So, starting with q and a string of the form a x 1, x 2, x m and a on the stack this whole thing is x. x we have written the form a x 1, x 2, x m. Now, in one move it goes to q 1 this a has been read and you are left with x 1, x 2, x m a has been replaced by B 1, B 2, B m we can go in from this to this in one move. Now, because of the induction hypothesis after reading x 1 it will erase B 1 on the stack and B 2 will be x supposed in between stack may grow and come back that does not matter.

So, when it goes to q 2. x 1 has been read and x 2, x 2, x 3, x m is the reaming 1 m now, stack contains B 2 to B m now after reading x 2 it goes to q 3 now, the portion to be read is x 3 to x m and stack will contain B 3 to B m proceeding like that ultimately you will have q m, x m, B m and that will be also erase and you will go to q m plus 1, but q m plus 1 what is q m plus 1 is p. So, from that you will go p, Epsilon, Epsilon stack whole stack will be emptied and you will go to.

(Refer Slide Time: 55:40)



So, we come to the conclusion that if this non-terminal derives x that means from this ID you can go to this ID. From the first rule is of this form q naught, Z naught, any derivation will be q naught, Z naught some p. Then that derives x. A derivation will be of this form now, from this if we can derive x that means from ID q naught, x, Z naught you can go to p, Epsilon, Epsilon.

So, if x is derivable from S if x the string access derivable from S, then the first step will be like this then you derive x then this is possible means this is possible if and only if from this ID, we can go to this ID by what we have proved just now, and what does this means this is the initial ID after reading x you go to final ID, which is empting the store that means x has been accepted by empty store so with the grammar derives a string x then that string is accepted by empty store by the pushdown automaton and vice versa.

So, what we have done is given the pushdown automaton, which accepts a language by empty store we have constructed the grammar further and shown that both of them pushdown accept, automaton accepts (( )) means grammar generates only L by and so on vice versa there are the same the language generated by the grammar and the language accepted by the pushdown automata by empty store are the same. So, this is proving in the other way around. with this and we have proved the equivalence with final states so, you see context free grammars pushdown automata accepting by empty store, pushdown automata accepting by final state they are equivalent and when you say pushdown automata we mean nondeterministic pushdown automata.