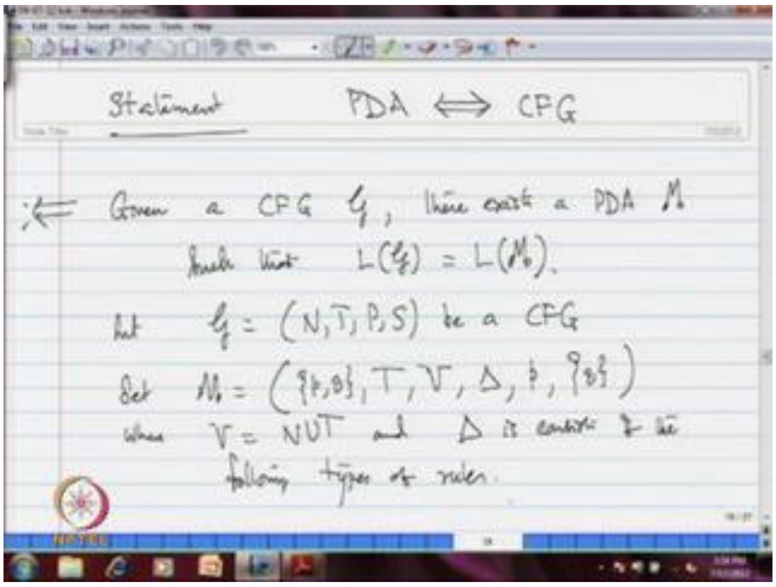
Formal Languages and Automata Theory
Prof. Dr. K. V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module – 10
Pushdown Automata
Lecture – 02
PDA ⇔ CFG

In the last lecture I have started discussing, what are the PDA that we have introduced, that is equivalent to CFG in the sense that, all the context grammars, for all the, for every context free grammar. You know, you will have a PDA accepting, that particular context free language. And any language, which is accepted by PDA, that is a context free language. So, this gives you, you know a better tool to understand context free languages, I mean in the light of automata, this is push down automaton.

(Refer Slide Time: 01:15)



Statement PDA $\Longleftrightarrow$ CFG

$\Leftarrow$ Given a CFG G, there exist a PDA M such that $L(G) = L(M)$.

Let $G = (N, T, P, S)$ be a CFG

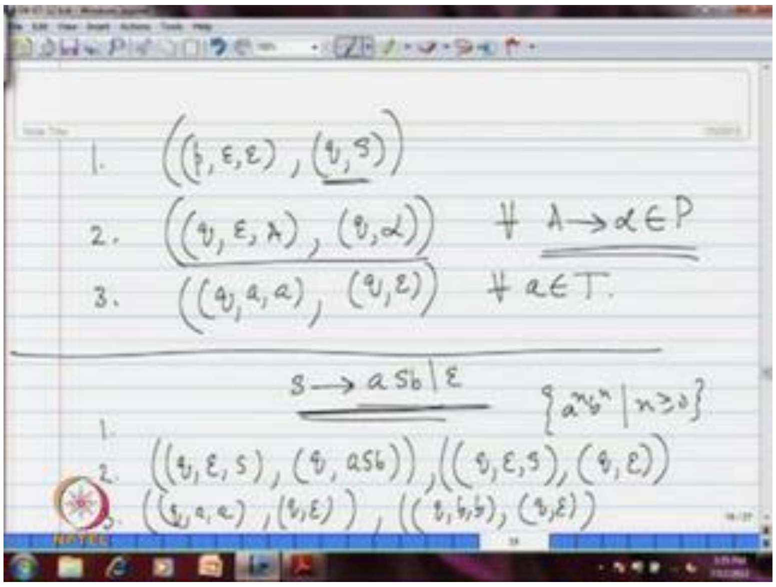
Set $M = (\{p, q\}, T, V, \Delta, p, \{q\})$

where $V = N \cup T$ and Δ is consists of the following types of rules.

So, in the last lecture itself, I have started discussing PDA and CFG (s) are equivalent. In fact, we have done the construction one side, that you give me a context grammar, how to construct a PDA. In fact, I have constructed a PDA with two states. So, as given here, you give will you take a CFG and N, T, P, S. You consider two states p and q terminals, whatever the set of terminals will be the input alphabet and N union T. The terminals and non terminals, we are taking stack symbols, and transitions we have defined ((Refer

Time: 01:35)) p is initial state, and the second state will be the final states only two states.

(Refer Slide Time: 01:41)



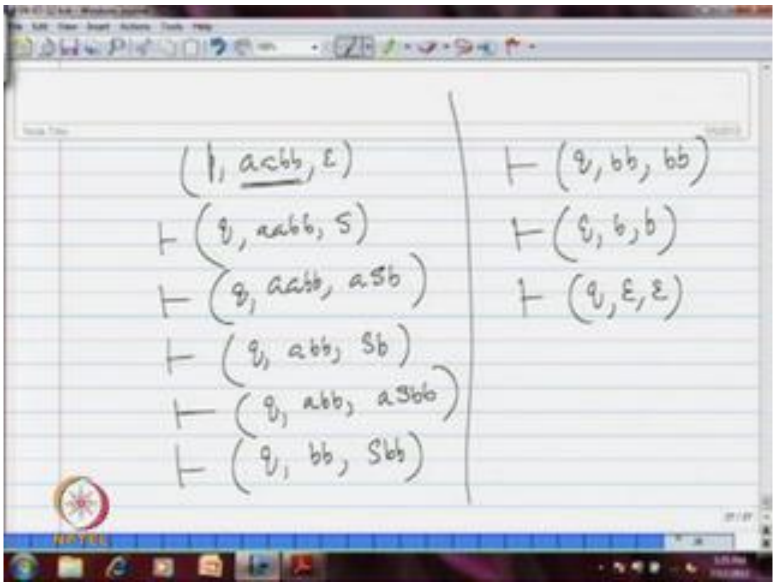
1. $((p, \varepsilon, \varepsilon), (\underline{q, S}))$
2. $((q, \varepsilon, A), (q, \alpha)) \quad \forall\ \underline{A \rightarrow \alpha \in P}$
3. $((q, a, a), (q, \varepsilon)) \quad \forall\ a \in T.$

$\underline{S \rightarrow aSb \mid \varepsilon} \qquad \{a^n b^n \mid n \geq 0\}$

1.
2. $((q, \varepsilon, S), (q, aSb)), ((q, \varepsilon, S), (q, \varepsilon))$
3. $((q, a, a), (q, \varepsilon)), ((q, b, b), (q, \varepsilon))$

And the, then the rules we have declared, the transition we have declared like this without you know, you start and initially by putting in s in to the stack changes to the state q, that is of course, the final state for us. And then what are the string that you are generating by the grammar, through certain rules corresponding to that, you are actually giving transitions like this. So, we have done some sample this thing also, that a power n b power n, n greater than equal to 0, that we generate through this two rules S goes to a s b epsilon, this two rules.

(Refer Slide Time: 02:24)



$(p, \underline{aabb}, \varepsilon)$

$\vdash (q, aabb, S)$

$\vdash (q, aabb, aSb)$

$\vdash (q, abb, Sb)$

$\vdash (q, abb, aSbb)$

$\vdash (q, bb, Sbb)$

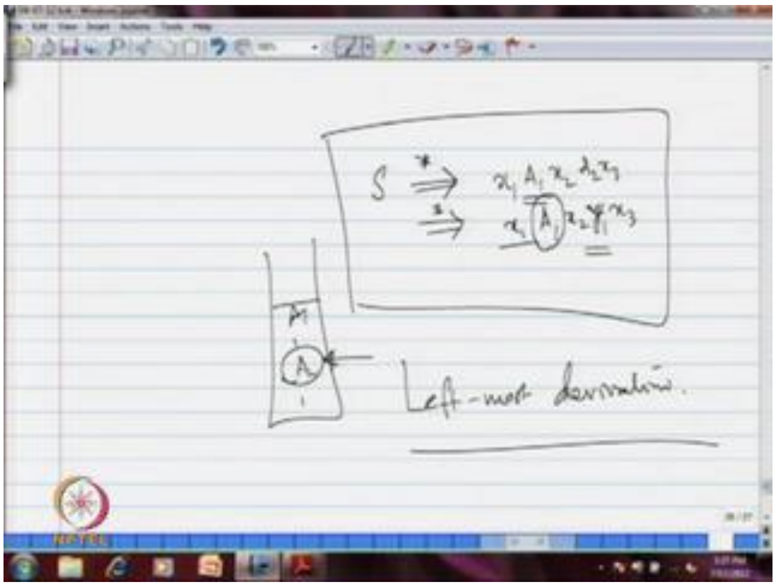
$\vdash (q, bb, bb)$

$\vdash (q, b, b)$

$\vdash (q, \varepsilon, \varepsilon)$

Now, these are the transitions, and I have shown sample computation also in PDA. And here, you understand that we are first putting S in to this stack, and then we are generating what are the string that you generate through that, and then match with the string given in the input, that is what we are doing.

(Refer Slide Time: 02:51)



$S \overset{*}{\Rightarrow} x_1 A_1 x_2 A_2 x_3$

$\overset{*}{\Rightarrow} \underline{x_1 \boxed{A_1}} x_2 \underline{\gamma_1} x_3$

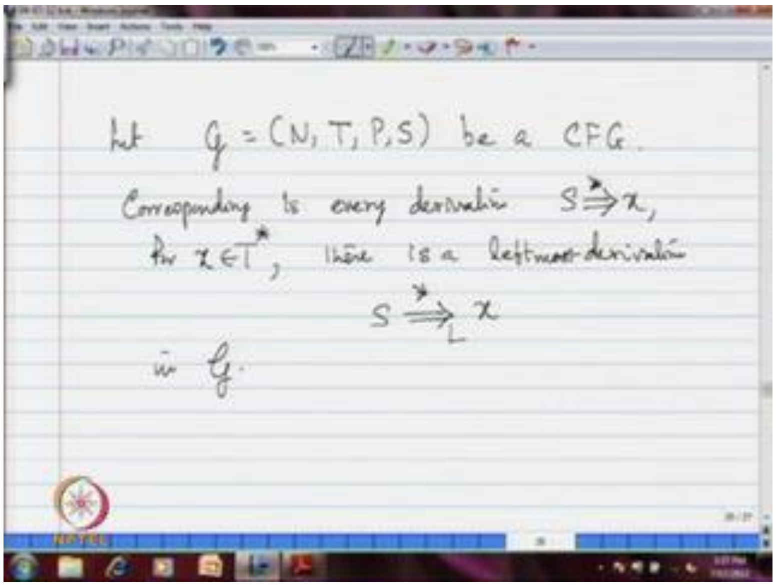
But here, we have a special feature that, you know though derivation that you are applying, if you are having some derivation, in which say for example; you start like this. And you have some derivation, which is getting, you know x 1, one non terminal symbol say for example; A 1 some x 2 another A 2; and say for example x 3. Suppose, after finitely many steps, and then if you are applying in case, if it is the situation like this. In this, say this say for example another ((Refer Time: 03:14)), so this A 2, if it is making some terminal string y 1 say x 3.

Now, you are matching with this, this is the non terminal, that you are getting in the stack on the top, this is the A 1 and the rest of the strings, but you have generated some string y 1 in place of A 2, say if A 2 is there. So, but we are not accessing this, unless you access this A 2, you cannot apply that rule. And then generate the string respective string, and then you match with the input. Suppose, if this is the symbol that, you know for which this is applied.

So, what is the requirement you understand that, if you have a derivation, and in this stack if you are generating the string, which you want to match with that. You have to first replace, you know for an first non terminal symbol in the string; that means, such a derivation we have already talked about, such a derivation is called left most derivation. So, you know that, for every derivation there is a equivalent left most derivations.

So, usually have such a derivation in hand, then what you can do, you can generate this string in the stack. And match with that, because you are having, you are applying left we have we are having left most derivation. So, then the, what are the rule that we are applying on the left most non terminal symbol you apply, and generate the terminal string, and match with the input.

(Refer Slide Time: 04:48)

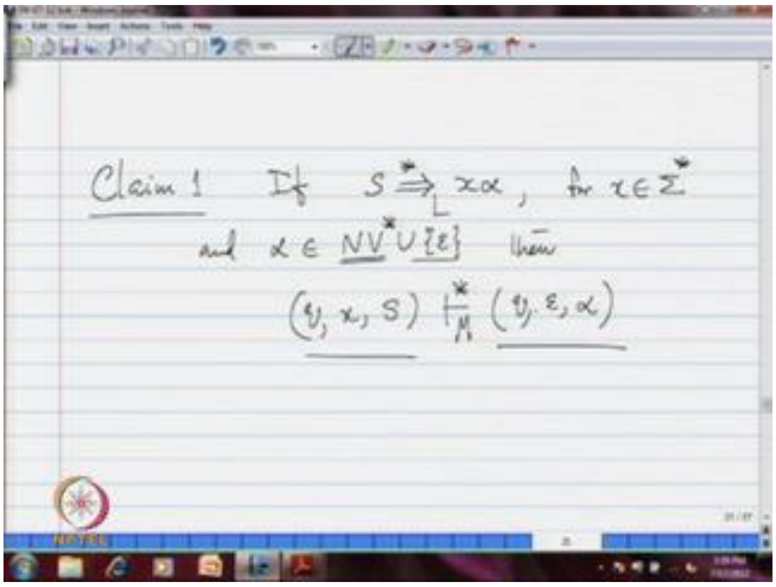


Let $\mathcal{G} = (N, T, P, S)$ be a CFG.

Corresponding to every derivation $S \overset{*}{\Rightarrow} x$, for $x \in T^*$, there is a leftmost derivation $S \overset{*}{\underset{L}{\Rightarrow}} x$ in $\mathcal{G}$.

So, this is the idea, so what do we do, first we recall this statement. If you take a context free grammar g N, T, P, S and corresponding to every derivation, say for example; S generates x infinitely many step, for some x in T star. You know that there is an equivalent left most derivation, I take this in to count to you know; prove that what are the construction we did; in fact that works.

(Refer Slide Time: 05:18)



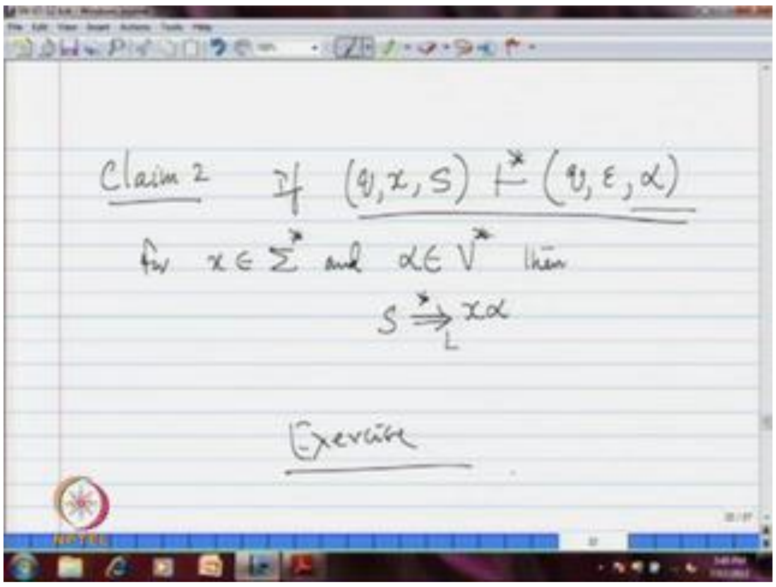
Claim 1 If $S \overset{*}{\underset{L}{\Rightarrow}} x\alpha$, for $x \in \Sigma^*$ and $\alpha \in NV^* \cup \{\varepsilon\}$ then $(q, x, S) \vdash_M^* (q, \varepsilon, \alpha)$

So, for that purpose, what I will do I will just let me state, these things. If S produces the string x alpha, through leftmost derivation infinitely, when is steps for x in sigma star.

And alpha, you know is a string, for which the first symbol is an non terminal symbol N V star when I am writing union epsilon, if alpha is there, you know the first non terminal symbol, so this is a left most derivation x is a terminal string.

So, till this point, suppose if you have generated then, what I am saying it is from this configuration on the top of the stack, if you have a ((Refer Time: 05:50)) in this not, in the stack, if you have s and you can come consume this x in the input from the in the state q to reach to the configuration, in which the stack content is alpha. So, corresponding to each such, you know derivation, you can have this kind of computation in the constructed pushdown automaton.

(Refer Slide Time: 06:14)



Claim 2 If $\underline{(q, x, S) \vdash^{*} (q, \varepsilon, \underline{\alpha})}$

for $x \in \Sigma^{*}$ and $\alpha \in V^{*}$ then

$S \overset{*}{\underset{L}{\Rightarrow}} x\alpha$

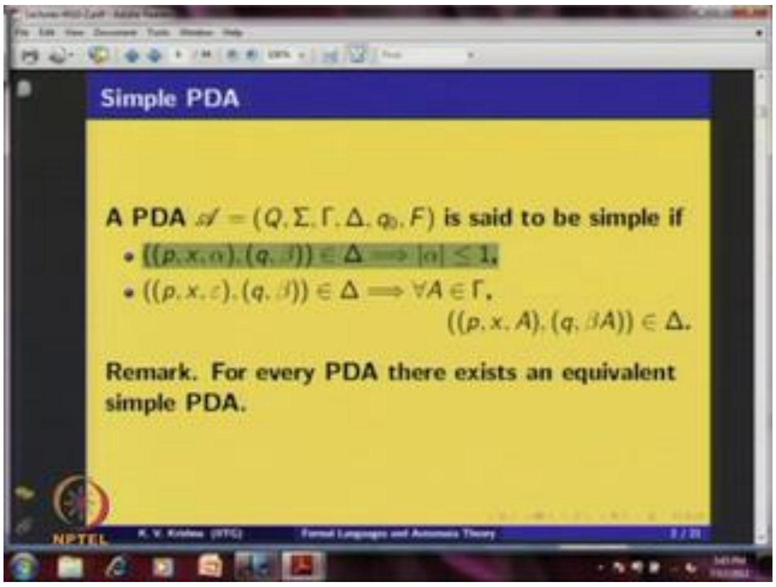
Exercise

And conversely, if you take any configuration of this sort q, x, s and infinitely many steps, you can, you know finish x and left, if you are left with some stack alpha for some x in sigma star, and alpha in V star. Then we can certainly have you know a derivation in the given CFG of the form S gives x alpha infinitely many steps as a left most derivation, you can have. By observing these two claims, you know we can quickly conclude from these two claims, we can conclude that, what are the pushdown automaton, that we have constructed indeed that works.

Now, how to prove this claim one and claim two. In claim one, we are starting with the derivation, and we wanted to observe this computation. Now, what is the idea proof idea here is, we can apply induction on the number of steps in the derivation to get this computation. You can see the basis and inductive hypothesis, and you can exchange induction. And similarly, in the for conversely, if you take the computation in this, in the given in the constructed grammar, which is of this form. You can apply induction on the number of steps in the computation, and you get, and you can prove that, this derivation in the grammar is possible.

So, this as I mention, we will apply induction and do that, I will be working for other converse part of the construction, when equivalence between PDA and CFG. So, these two, I will leave it as exercise, it is not a big deal, because I will be doing this, in this lecture itself for other part. So, you can imitate, and this is a very simple thing, that I hope you can you will be able to do, and do conclude that, the what are the position automaton we have constructed indeed is equivalent to the given CFG.

(Refer Slide Time: 08:17)



Simple PDA

A PDA $\mathscr{A} = (Q, \Sigma, \Gamma, \Delta, q_0, F)$ is said to be simple if

- $((p, x, \alpha), (q, \beta)) \in \Delta \Longrightarrow |\alpha| \leq 1$,
- $((p, x, \varepsilon), (q, \beta)) \in \Delta \Longrightarrow \forall A \in \Gamma,$ $((p, x, A), (q, \beta A)) \in \Delta.$

Remark. For every PDA there exists an equivalent simple PDA.

Now, I will look for the converse part; that means, you give me a push down automaton; you can construct a corresponding context free grammar. So, I start this, with the concept called simple PDA, because this is one corresponding a every PDA an equivalent simple PDA, we can have. First, let me tell you what is let me introduce, what is PDA.

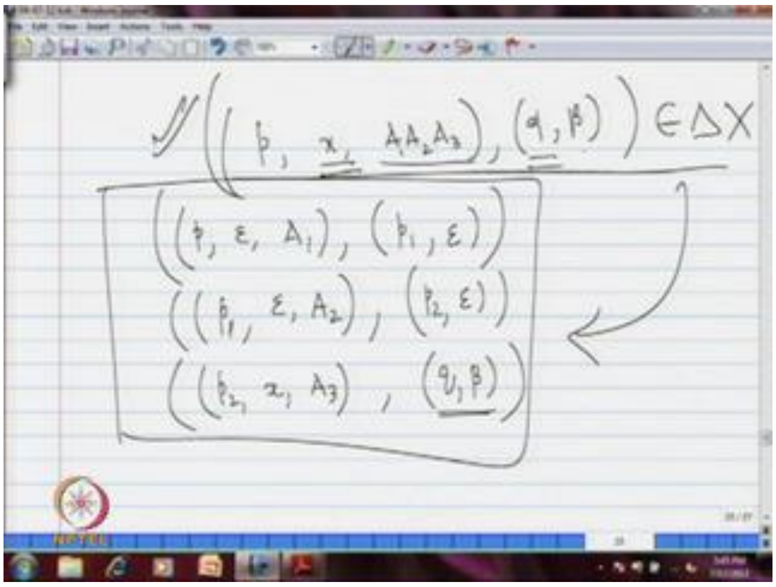
A PDA Q sigma gamma delta q 0 f is said to be simple. If you take any transition p x alpha q beta here, you will have, you know the stack string that you are popping of length should be less than or equal to 1; that means, either empty string or a single symbol, that you will pop at one time. And, the second condition is, if you have epsilon here, you know, then for all A in gamma, for all stack symbol, I should have this kind of transition also.

Now, I give this remark for every PDA, there exists an equivalent simple PDA. Now, the question is, how do you get it, what are the two conditions the transitions they have to satisfy. Any time, if you want to pop, any symbol from the stack, either it should be epsilon; or you know it should be of length one; or if you are having a transition with epsilon here. Then there should be a transition for each symbol A in gamma, with this kind of thing.

Now, the second one can be achieved very easily, whenever you have such a transition, you take each symbol A from gamma, and then write a production rule of this sort. Corresponding to each element of gamma, through which you know, you will achieve this, and we are not going to lose anything, we are not going to you know, change the language. The reason is, because this transition whatever that is ((Refer Time: 10:09)) lying, when I introduce this is such a transition, what it says, whenever you are accessing A on the top of the stack.

You are temporarily removing, and immediately you are putting A in the stack, and what are that beta. So, this particular by introducing this kind of transitions, the number of transitions with respect to the size of the stack, we are not going to gain or lose anything with on the language. So, language remains same, from the given PDA to this.

(Refer Slide Time: 10:45)



$$((p, \underline{x}, \underline{A_1A_2A_3}), (\underline{q}, \beta)) \in \Delta \times$$

$$\boxed{\begin{array}{l} ((p, \varepsilon, A_1), (p_1, \varepsilon)) \\ ((p_1, \varepsilon, A_2), (p_2, \varepsilon)) \\ ((p_2, x, A_3), (\underline{q, \beta})) \end{array}}$$

Now, corresponding to this, I will just give me a simple technique, that say for example, if you have something of this sort, say p x A 1 A 2 A 3; say for example, you have three symbols, and the transitions; say for example, of this form say q beta. Suppose, this is your transition, then what I would suggest you, you just simply remove this transition. And place these transitions of this form, with some you states, you choose some new states, which you have not used, which you do not have earlier; say for example, p 1 and then epsilon.

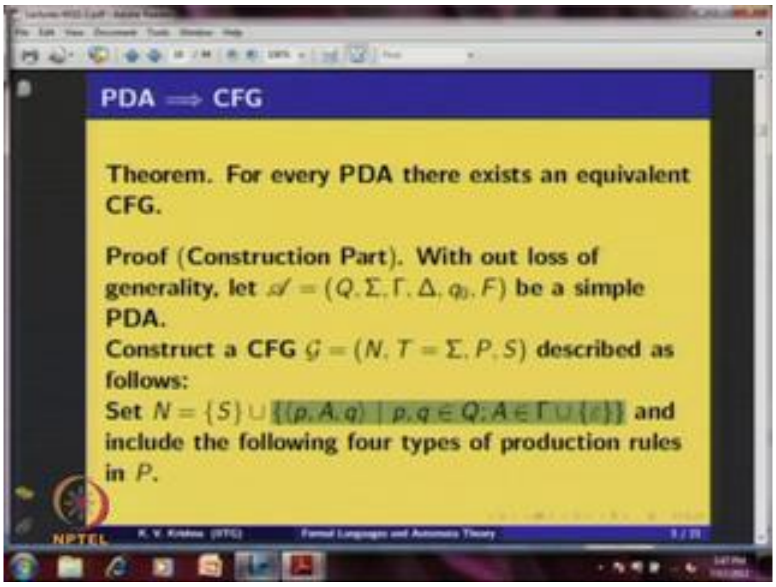
So, this is one transition I put, what I am trying to do here, is instead of, removing three symbols at once, I will remove one after another, that is what I do. So now, I am in p 1 in p 1 without taking the input, I simply remove now the second symbol A 2, and change to another state p 2 and then, since I am in p 2. Now, what I will do, this x need to be consumed in the input. So, now I just consumed in the at the end. So, A 3 I will pop, and then connect to the state whatever you require that is q here, so you change to the state q.

So, what I will do, corresponding to this particular transition, I remove this transition from the set, and I introduce these three transitions in ((Refer Time: 12:04)) of this; that means, what are the purpose of this particular transition. It is exactly doing this, because this p 1 p 2, what I have chosen they are new states here; say for example, if you have n number of symbols here, choose n minus one states. You remove one after the symbols, and the new states corresponding each transition, you know you choose.

And, remove one symbol after another from the stack, and by the time you know remove the last symbol. You consume, what are the input that you have to consume to in this particular transition given. And then, change to the required state here for example; q here, so change to the required state, and put whatever that you have to put it in the stack. So, through this, you can understand that these two rules, you know these two conditions can be, you know achieved.

We can achieve this two conditions, and to make the given PDA to a simple PDA. So, thus every PDA can be converted to an equivalent symbol PDA. So, this concept, I will use this particular concept, I will use simple PDA, and given a PDA, a simple PDA. Because now, you give me a PDA first I will construct an equivalent simple PDA, from that simple PDA, I will construct a given, construct a grammar CFG context free grammar.

(Refer Slide Time: 13:21)



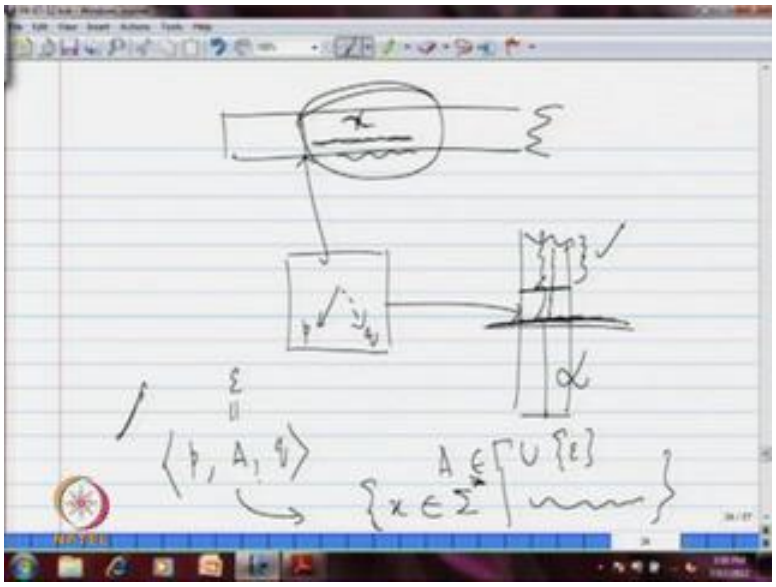
PDA $\Longrightarrow$ CFG

Theorem. For every PDA there exists an equivalent CFG.

Proof (Construction Part). With out loss of generality, let $\mathscr{A} = (Q, \Sigma, \Gamma, \Delta, q_0, F)$ be a simple PDA.
Construct a CFG $\mathcal{G} = (N, T = \Sigma, P, S)$ described as follows:
Set $N = \{S\} \cup \{(p, A, q) \mid p, q \in Q; A \in \Gamma \cup \{\varepsilon\}\}$ and include the following four types of production rules in P.

So, now this side I will consider for every PDA, there exist an equivalent CFG. Let me start formally, a construction part let me first tell you, without loss of generality. Let a Q, sigma, gamma, delta, q 0, F be a simple PDA, you start with the simple PDA. Now, I construct a CFG N T, here P the input alphabet sigma P S as described below. Now, I consider non terminals of this form, one non terminal S for the start symbol I considered here, and then the non terminals, I will consider of this form.

(Refer Slide Time: 13:59)



For that, let me first tell you, some philosophy about this non terminal symbols, in a PDA, what we are doing, we are having this kind of situation. So, we are consulting the stack, what are and you are reading the input for the given state. Now, what do, what is the construction, non terminal symbol construction, we consider things of this form p A q, this kind of non terminals we have considered. For A, it is a stack symbol or it may be epsilon, what is the philosophy with the non terminal symbol, that we considered here is, when you are in the state p, when the top of the stack is A; say for example, when you have A here.

Now, you are for example, reading and writing it is this here, for example, at the particular position. So, here after you will be reading some input here, in the tape say some x you are reading. So, when you are reading this input, the possibility in this push down automaton is, when A is on the top of the stack, you may have you may had some more symbols on the stack, certain symbols. And you will remove, because you know accepting a string input string, what you have to do you have to ultimately make the stack empty, we know that.

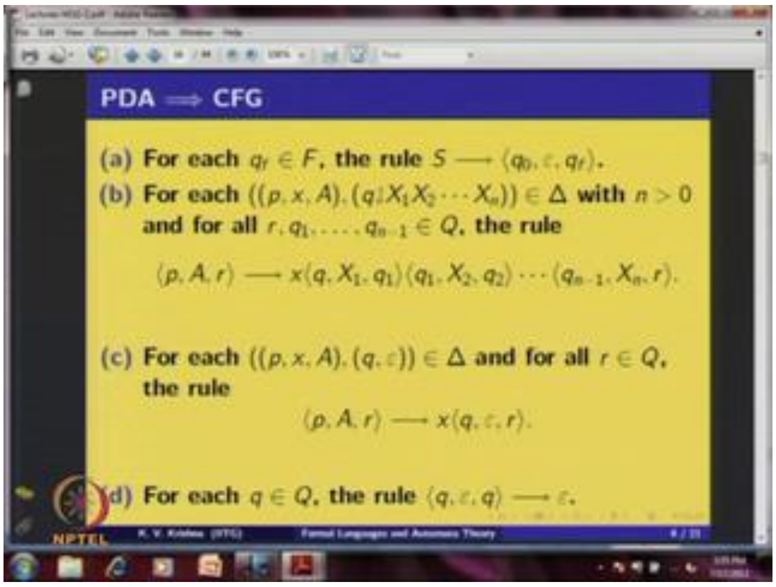
So, between you know the situation say, you are current state is p, when you have A on the top of the stack, from here you might be adding some more symbols on the stack here. And then finally, any way you have to remove ((Refer Time: 15:26)) to make the stack empty. So, by that before by the time, you A remove A from the stack; that means, we are not touching in between, because what you may have some additions on the top of this. And by the time, you remove A from the stack, if you reach to the state q from here to here.

So, it is not two things. So, this you may come to for example, state q whatever the portion of the input that you may be able to read, that set will be denoted by this non terminal symbol. So, this non terminal symbol denote, set of all those strings in the input, such that the current state is p, what I am explaining is this. The current state is p, and when you have A on the A is on the top of the stack. By the time you remove A, if you reach to the state q, what are the amount of input that you read, whatever the input that you read, that I given instance.

So, all those strings will be represented or assumed to be derived through this kind of non terminal symbol that is the philosophy we use. And for example, if A is epsilon and; that means, say I do not have, you may have something in the stack. Let me A is epsilon; that means, whatever that you have say for example, this portion say alpha on the top of this. You might be adding some more things in that stack; that means, without consulting the stack, what are the amount of input, that you read between the states p and q without consulting this stack; that means, the stack information whatever is lying, that will be there, I am not saying here stack is empty.

If you have A is equal to epsilon here, stack you may have something, but without consulting the stack, when without touching what are the stack information already existing. You may add some more things in this between states p and q, and you may remove them, and you come to this same point; that means, whatever is existing, that I am not going to change. So, within these two times, whatever the input that you read, that is represented here, through this non terminal symbol, I hope you understand this philosophy.

(Refer Slide Time: 18:03)



PDA $\Longrightarrow$ CFG

(a) **For each** $q_f \in F$, **the rule** $S \longrightarrow \langle q_0, \varepsilon, q_f \rangle$.

(b) **For each** $((p, x, A), (q, X_1 X_2 \cdots X_n)) \in \Delta$ **with** $n > 0$ **and for all** $r, q_1, \ldots, q_{n-1} \in Q$, **the rule**

$$\langle p, A, r \rangle \longrightarrow x \langle q, X_1, q_1 \rangle \langle q_1, X_2, q_2 \rangle \cdots \langle q_{n-1}, X_n, r \rangle.$$

(c) **For each** $((p, x, A), (q, \varepsilon)) \in \Delta$ **and for all** $r \in Q$, **the rule**

$$\langle p, A, r \rangle \longrightarrow x \langle q, \varepsilon, r \rangle.$$

(d) **For each** $q \in Q$, **the rule** $\langle q, \varepsilon, q \rangle \longrightarrow \varepsilon$.

Now this philosophy is taken to consider this non terminal symbols. So, S start symbol for start symbol for the context free grammar, and then given any two states p and q. I consider non terminal symbols of this form p A q, where A is a stack symbol or epsilon. And then, we include and the following types of rules. First rule, we can quickly understand, because from the start symbol, I will if I connect to this, what is the idea from the initial state. Because what are strings you have to generate in this grammar from the initial state to some final state, we know this stack, you start with it will be epsilon, you do not have anything.

Now, from there you come back to the same status, so; that means, here you require epsilon. So, explaining this rule is much easier that is not a big deal, I hope you understand. So, S should be connected to this kind of non terminal symbols, for each you know this kind of non terminal symbols. You will get one rule, one for each not final state of the given PDA; and then I have you know type b rule, I will explain you this; and then type c rule; and then the type d rule again will be easy to explain, for each state.

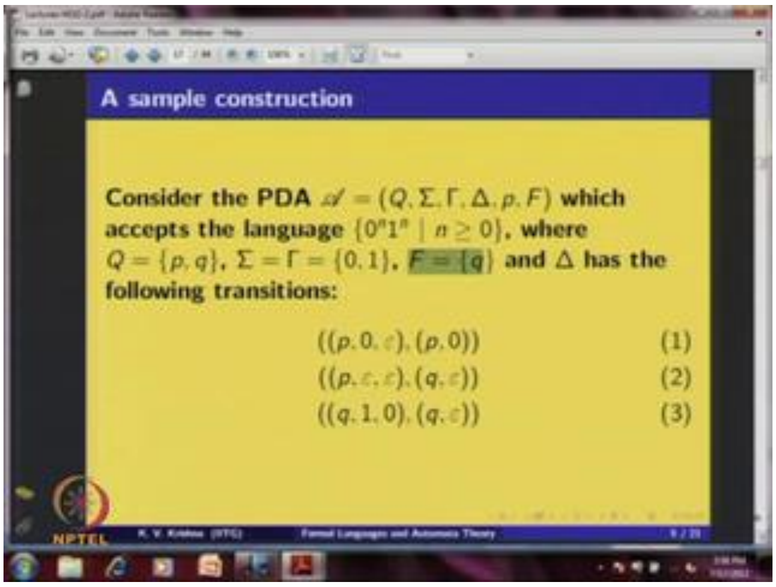
If, I have this p epsilon say q epsilon, q type of non terminal symbol, you know I can nullify that, because from the state to the state q. You know without changing the stack, what kind of symbols, whatever that you are reading that is represented by this, you know any way. And in addition to that, you know I can say that epsilon can be derived through that, so this rule is easy to understand, now let me come to the type b and type c rules.

In general, you know if I explain type b with n is equal to 0, you are essentially having type c rule. If you take a transition of this form p x A, which is in the state p by consuming the input A, another stack symbol A. So, A can be not a stack symbol or epsilon, so because it is simple PDA. Now, if you have done this; that means, you are changing to the state q, and you are putting X 1, X 2, X n inside the stack. Then, what we are suggesting here, is from p we are changing to the state q; that means, eventually I will leave you in the state q.

So, the corresponding non terminal may be, you know the production rule may be, this is p A r; that means, you are consulting this stack with A. And now, A is removed X 1, X 2, X n is inserted, so that means, corresponding to X 1, X 2, X n this non terminal symbols. This stack symbols you know are representing some information of the input, when we are putting X 1, X 2, X n; these are representing some information of the input, when you are removing this systematically, what are the string that you wanted to accept; that means, to be generated in the grammar, that we are going to construct.

So; that means, whatever the input that we have consumed, that is generated here, that is clearly mentioned, what are the input that you have consumed in PDA, that need to be generated. So, that is, what is this, and then corresponding to this X 1, X 2, X n. Now, for all the possible states q 1, q 2 and so on, or q n minus 1 and r are all possible states, because I am choosing this all possible states. I give such a rule this q, x 1, q 1, q 1, x 2, q 2 and so on, q n minus 1, x n are and this r q i (s) are arbitrary. So now the connection between this from, the state p to q.

(Refer Slide Time: 21:20)



A sample construction

Consider the PDA $\mathscr{A} = (Q, \Sigma, \Gamma, \Delta, p, F)$ which accepts the language $\{0^n1^n \mid n \geq 0\}$, where $Q = \{p, q\}$, $\Sigma = \Gamma = \{0, 1\}$, $F = \{q\}$ and Δ has the following transitions:

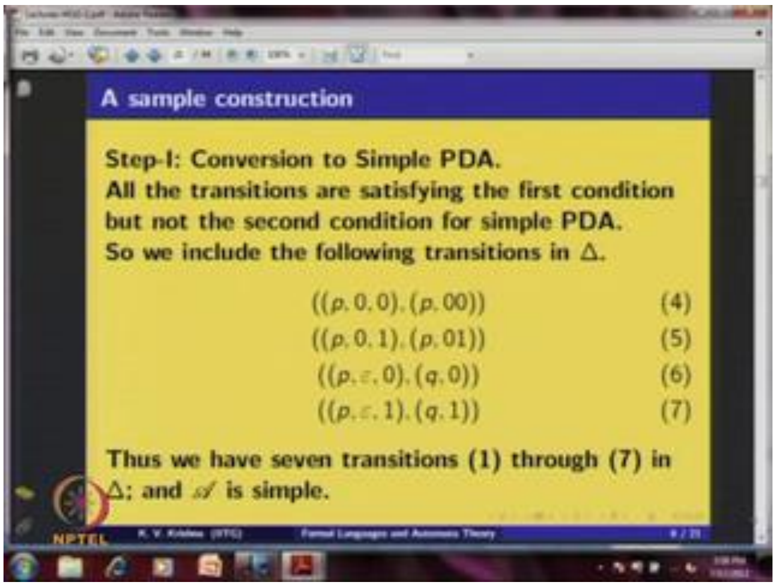
$$((p, 0, \varepsilon), (p, 0)) \quad (1)$$
$$((p, \varepsilon, \varepsilon), (q, \varepsilon)) \quad (2)$$
$$((q, 1, 0), (q, \varepsilon)) \quad (3)$$

Now, I will give you a sample construction, then you will understand this better, you know this PDA. So, for 0 power n 1 power n, n greater than or equal to 0, you can consider PDA as follows. You take this state p read 0, you know and insert 0 in the stack, and non deterministically I am changing, because after some point of time I will change to q. And then, when I have a match, I will remove. You know, when the input is 0 power n and 1 power n, whenever there is a match, you will be in final state by consuming all the ((Refer Time: 21:47)) the entire input and making the stack empty through this rules, that you can quickly understand.

And see that this particular PDA accept 0 power n 1 power n. Now, here notice that, I have considered two states and sigma 0 one of course, in gamma I am using only 0, I am putting only 0 in the stack; however, I have considered 1 also, just for the sake of you know showing this particular construction. Anyway, final state is one is like this. Now, corresponding to this PDA, I will just give a sample construction of context free grammar.

So, first converting simple PDA, you know that, what are the things, that I have to introduce. Because, here is epsilon in the first rule, now for all the possible non terminal a stack symbols, I have to have this thing, and here type full sort of that first condition of simple PDA. Because it is anyway satisfied, because in this third position all are the lengths are less than or equal to 1.

(Refer Slide Time: 22:49)



A sample construction

Step-I: Conversion to Simple PDA.
All the transitions are satisfying the first condition but not the second condition for simple PDA.
So we include the following transitions in Δ.

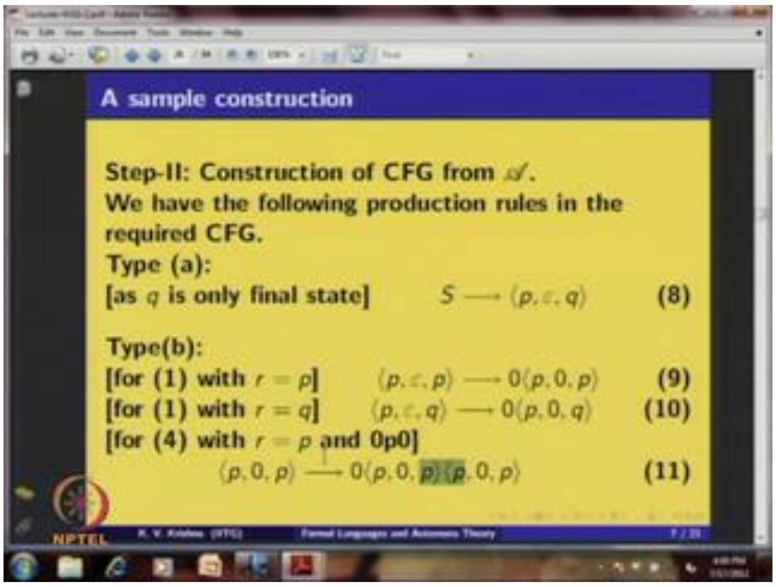
$$((p, 0, 0), (p, 00)) \quad (4)$$
$$((p, 0, 1), (p, 01)) \quad (5)$$
$$((p, \varepsilon, 0), (q, 0)) \quad (6)$$
$$((p, \varepsilon, 1), (q, 1)) \quad (7)$$

Thus we have seven transitions (1) through (7) in Δ; and $\mathscr{A}$ is simple.

So, condition one is of simple PDA is anyway satisfied; condition two that I have to consider. So, then what do I do, I have to introduce the rules like this, because example considered here is, p 0 epsilon, here the here is, 1 and p epsilon epsilon goes to q epsilon, these two rules that I have, so p 0 epsilon. Now, p, 0, 0 that is p, 0 0 that is what, because now by taking the possibility of 0 of the stack symbol and now, similarly here, by taking the possibility of 1 has a stack symbol. I hope you understand, here all though this is not very much used in the stack, since I have considered stack symbol 0 1, I am putting in this.

So, that you see that, when I am constructing, and since in the PDA this has no role, this has no role in the you know, the constructed CFG also, but just for the sake of example, I have considered that, and I am showing this rule also. Now, corresponding to p epsilon epsilon, we are changing in to q epsilon, we are changing the state, changing to final state. Now, by choosing 0 non terminal stack symbol and similarly, the possibility of choosing 1 as a stack symbol. So, these productions that I have to add in addition to the given three, there are four rules.

(Refer Slide Time: 24:20)



A sample construction

Step-II: Construction of CFG from $\mathscr{A}$.
We have the following production rules in the required CFG.
Type (a):
[as q is only final state] $S \longrightarrow \langle p, \varepsilon, q \rangle$ **(8)**

Type(b):
[for (1) with $r = p$] $\langle p, \varepsilon, p \rangle \longrightarrow 0\langle p, 0, p \rangle$ **(9)**
[for (1) with $r = q$] $\langle p, \varepsilon, q \rangle \longrightarrow 0\langle p, 0, q \rangle$ **(10)**
[for (4) with $r = p$ and 0p0]
$\langle p, 0, p \rangle \longrightarrow 0\langle p, 0, p \rangle\langle p, 0, p \rangle$ **(11)**

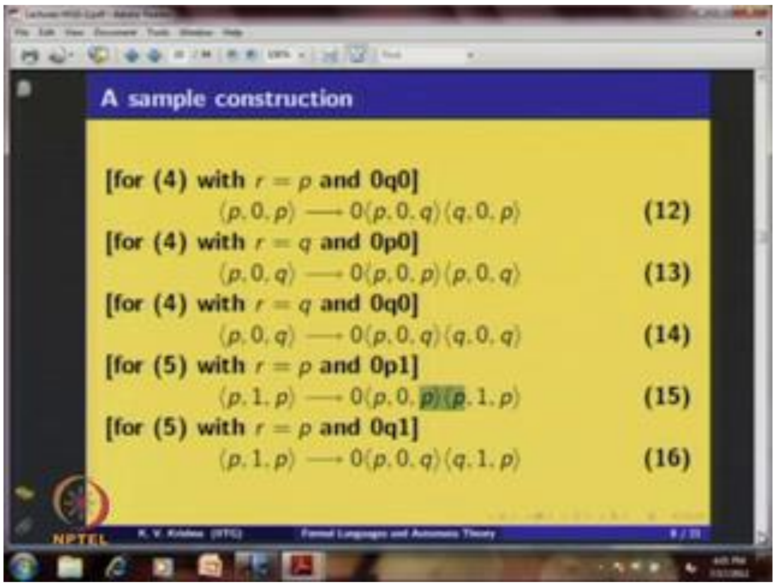
K. V. Krishna (IITG) Formal Languages and Automata Theory 7 / 21

Now, you can quickly see that, this is a simple PDA, because now the conditions both the conditions of simple PDA are satisfied. Now, using this simple PDA, we construct CFG from A. So, how do I construct as the first rule is saying that, I have to give a production rule of this form S goes to p epsilon q, this I have to give such a rule for one for each final state, here of course, only one final state therefore, we have only one rule here. And type b remember corresponding to each transition of this form, I have to give one, because here the transition means for n greater than equal to, for n greater than 0 I am giving, whatever that you are putting inside X 1, X 2, X n.

If something non zero non empty that we are putting I am using this, if you are putting empty, then these things are not there simply X is produced q epsilon r. So, this is the distinction, as there it is type b rule or type c types of rule Fine. Now, let us look at that. So, in type b type of rule by taking first rule, and this kind of transitions you we need to introduce r equal to p, when you choose the possibility p here, because you look at in the rules, I have mentioned for all the possibilities of the states, I have to write this kind of rule.

Now, by taking the possibility p, I write this rule. And then for by taking the possibility q for r, I take the I write this rule. Now, then you know by taking r is equal to p, and the possibility is 0 p 0, by considering this, because this is for rule four, you look at rule four how it is, because there are two zeroes. So, in between I consider p; that means, in between these, I mean, because I have mentioned it like this for the possibility of rule four. We have 0 0 there, so what I what I am doing by considering p here, that is one possibility r is p.

(Refer Slide Time: 26:11)



A sample construction

[for (4) with $r = p$ and 0q0]

$\langle p, 0, p\rangle \longrightarrow 0\langle p, 0, q\rangle\langle q, 0, p\rangle$ (12)

[for (4) with $r = q$ and 0p0]

$\langle p, 0, q\rangle \longrightarrow 0\langle p, 0, p\rangle\langle p, 0, q\rangle$ (13)

[for (4) with $r = q$ and 0q0]

$\langle p, 0, q\rangle \longrightarrow 0\langle p, 0, q\rangle\langle q, 0, q\rangle$ (14)

[for (5) with $r = p$ and 0p1]

$\langle p, 1, p\rangle \longrightarrow 0\langle p, 0, p\rangle\langle p, 1, p\rangle$ (15)

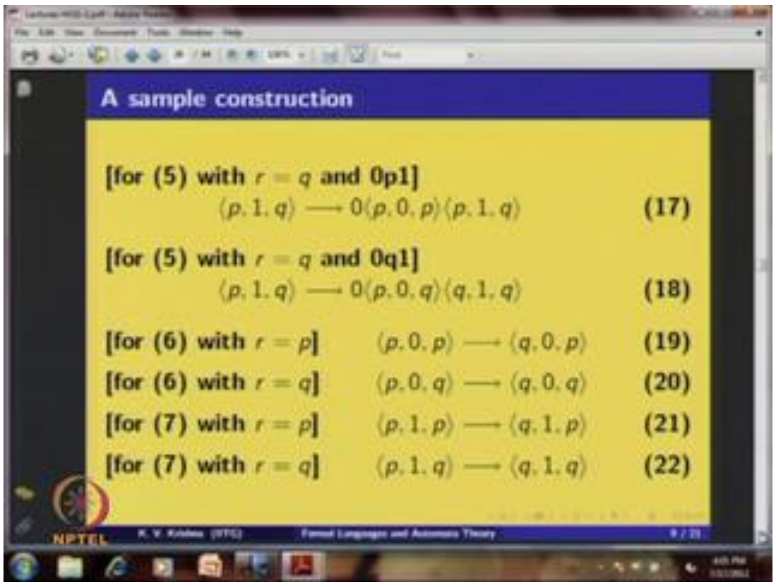
[for (5) with $r = p$ and 0q1]

$\langle p, 1, p\rangle \longrightarrow 0\langle p, 0, q\rangle\langle q, 1, p\rangle$ (16)

And then again for four, you have in between you can consider q, this is one possibility and r is p. Now, you can consider r may be q, and in between 0 0, you can consider p, I mean this, the possibility of the state. And then, r can be q in between you can consider q, so this is for rule four, the corresponding. And for rule five similarly, this and for rule five by considering q in between, you look at.

In general for r you have the possibility of p; for r you have the possibility of q, there are the two states possibility, when I have consider a string, whose length is greater than one, I can actually here. There are actually two symbols out of, so in between what do you want to consider. Because this, you will introduce for every state for every possibility of the state.

(Refer Slide Time: 27:07)



A sample construction

[for (5) with $r = q$ and 0p1]
$\langle p, 1, q\rangle \longrightarrow 0\langle p, 0, p\rangle\langle p, 1, q\rangle$ (17)

[for (5) with $r = q$ and 0q1]
$\langle p, 1, q\rangle \longrightarrow 0\langle p, 0, q\rangle\langle q, 1, q\rangle$ (18)

[for (6) with $r = p$] $\langle p, 0, p\rangle \longrightarrow \langle q, 0, p\rangle$ (19)

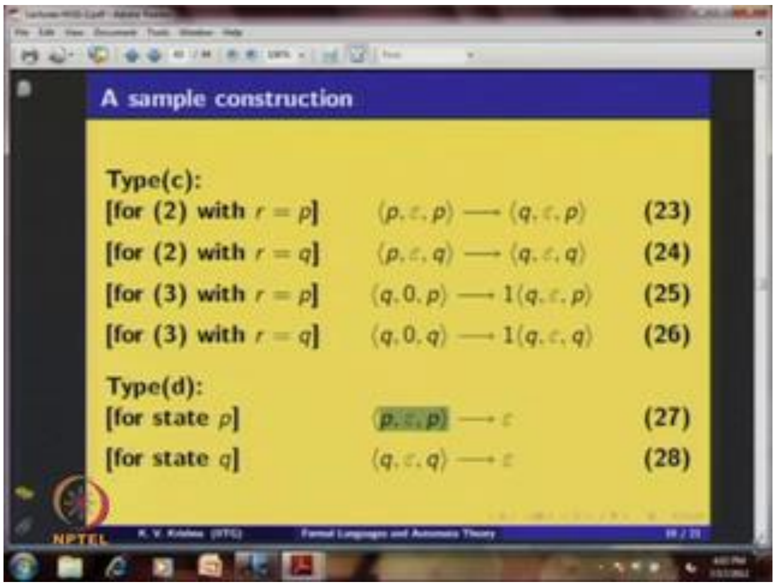
[for (6) with $r = q$] $\langle p, 0, q\rangle \longrightarrow \langle q, 0, q\rangle$ (20)

[for (7) with $r = p$] $\langle p, 1, p\rangle \longrightarrow \langle q, 1, p\rangle$ (21)

[for (7) with $r = q$] $\langle p, 1, q\rangle \longrightarrow \langle q, 1, q\rangle$ (22)

So, for rule five like this, and one more you get now. Now, for rule six, this rule they all coming under type b, this rule by choosing r the possibility p and q. For rule seven, again by choosing the possibility of p and q, these two rules 22 rules.

(Refer Slide Time: 27:26)



A sample construction

Type(c):

[for (2) with $r = p$] $\langle p, \varepsilon, p\rangle \longrightarrow \langle q, \varepsilon, p\rangle$ (23)

[for (2) with $r = q$] $\langle p, \varepsilon, q\rangle \longrightarrow \langle q, \varepsilon, q\rangle$ (24)

[for (3) with $r = p$] $\langle q, 0, p\rangle \longrightarrow 1\langle q, \varepsilon, p\rangle$ (25)

[for (3) with $r = q$] $\langle q, 0, q\rangle \longrightarrow 1\langle q, \varepsilon, q\rangle$ (26)

Type(d):

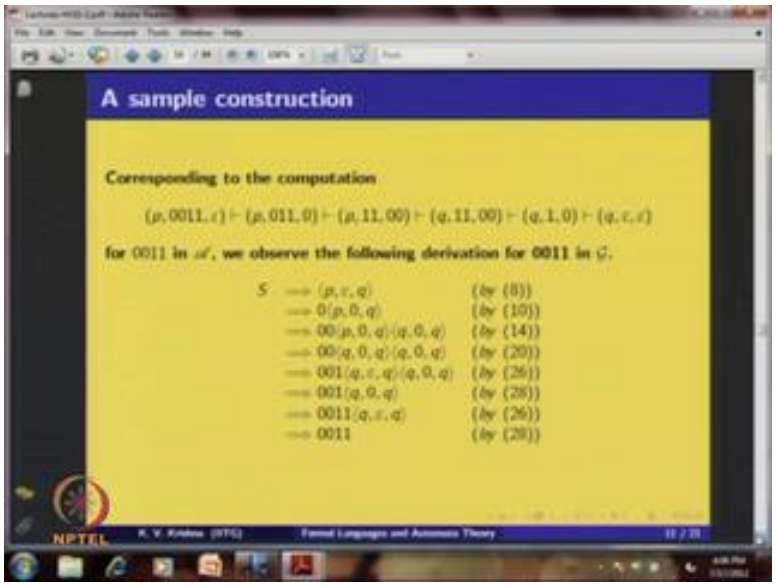
[for state p] $\langle p, \varepsilon, p\rangle \longrightarrow \varepsilon$ (27)

[for state q] $\langle q, \varepsilon, q\rangle \longrightarrow \varepsilon$ (28)

And then type c, you see; now by choosing r to be p, you have got this and by choosing r to be q corresponding to rule two. We have this 23 and 24 rules, and then for rule three, corresponding to three, that is of type c, that is coming under type c. You know, type c means this, because when you are inserting epsilon in the, you know, you are just popping something. And, that is just a pop rule, if it is of this form.

So, essentially rule two and rule three, the transitions two and three are of this form. So, we are introducing these two production rules, this four production rules now type c. And in type d, you see there are two states, so how to nullify this particular non terminal symbol, that p epsilon p goes to epsilon, q epsilon q goes to epsilon. So, these are the two rules type d, that we introduced.

(Refer Slide Time: 28:25)



A sample construction

Corresponding to the computation

$$(p, 0011, \varepsilon) \vdash (p, 011, 0) \vdash (p, 11, 00) \vdash (q, 11, 00) \vdash (q, 1, 0) \vdash (q, \varepsilon, \varepsilon)$$

for 0011 in $\mathscr{A}$, we observe the following derivation for 0011 in $\mathcal{G}$.

$$\begin{array}{lll}
S & \Longrightarrow \langle p, \varepsilon, q \rangle & (by\ (8)) \\
 & \Longrightarrow 0\langle p, 0, q \rangle & (by\ (10)) \\
 & \Longrightarrow 00\langle p, 0, q \rangle\langle q, 0, q \rangle & (by\ (14)) \\
 & \Longrightarrow 00\langle q, 0, q \rangle\langle q, 0, q \rangle & (by\ (20)) \\
 & \Longrightarrow 001\langle q, \varepsilon, q \rangle\langle q, 0, q \rangle & (by\ (26)) \\
 & \Longrightarrow 001\langle q, 0, q \rangle & (by\ (28)) \\
 & \Longrightarrow 0011\langle q, \varepsilon, q \rangle & (by\ (26)) \\
 & \Longrightarrow 0011 & (by\ (28))
\end{array}$$

K. V. Krishna (IITG) Formal Languages and Automata Theory 11 / 23

I hope you understand this construction. So, what did we do, we have considered PDA for zero power n and 1 power n, there are only three rules. But first by introducing another four, there only four three transitions by introducing another four transitions, we made that simple. And using that simple PDA, what are the four types of rules that we have mentioned, we have constructed you know a context free grammar.

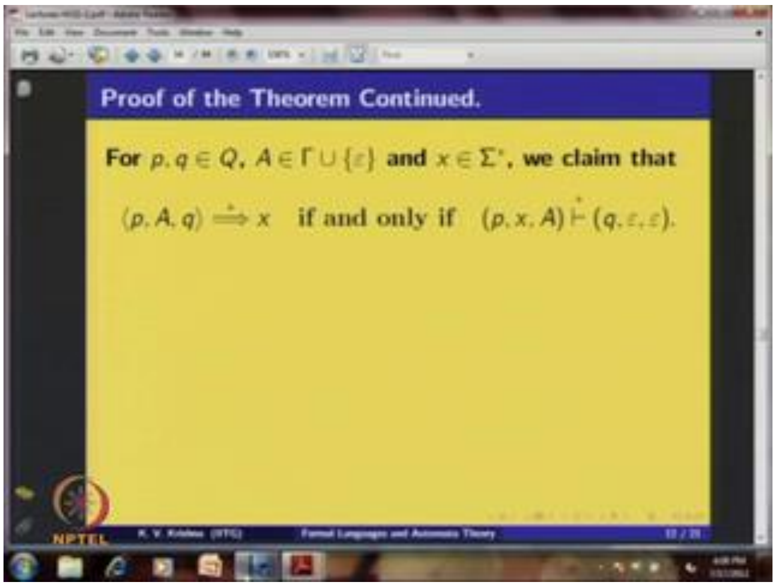
Now, let me just give you a small derivation in this context free grammar, corresponding to a computation for 0 0 1 1 in the given PDA. You can see that, this is how we are actually accepting 0 0 1 1 in the given PDA, you start from the initial state 0 0 1 1 with the stack empty, you know, you are reading those two zeros in the state p. And the non deterministically you are changing to the state q, then when you have one, and you have zero on the top of the stack, you just match them like that this. We know and you are in a final states, so this is how the computation is considered.

Now, corresponding to this, let us observe the derivation in the g, I am just marking this rules in the production rules the and the constructed CFG. So, from s, we will consider this you know, because this is the only possibility that we have. And then, you see I am reading zero in the input, and we I am putting zero in the stack, that is what we did in the first step of the computation. The same thing here is shown.

So, the targeted state, the final state is q that is any way there, but what is the current state now, ((Refer Time: 30:14)) I am continuing in the state p, so that is the state p, here it is shown that the zero is in the stack. And I have consumed zero in the input; that means, I have derived now in this derivation zero, that is what is this marked. Now, when I have consumed two zeros in the input, here I derived two zeros and then in the stack we have two zeros, those two zeros are shown up.

So, the respective production rules, that I am marking here, rule production ten here, production fourteen as I have constructed. And now, I can use twenty to make this as p 0 q, I can make it has q 0 q. And now, I can rule twenty six use you know you produce one then q epsilon q, then q epsilon q can be nullified and then this 0 0 1. So, the left over thing is q 0 q, and then 0 0 1 1 q epsilon q rule twenty six, I use. And then finally, you know, this q epsilon q again I use twenty eight ((Refer Time: 31:25)) the production rule twenty eight to show that 0 0 1 1 is generated in this.

(Refer Slide Time: 31:35)



Proof of the Theorem Continued.

For $p, q \in Q$, $A \in \Gamma \cup \{\varepsilon\}$ and $x \in \Sigma^*$, we claim that

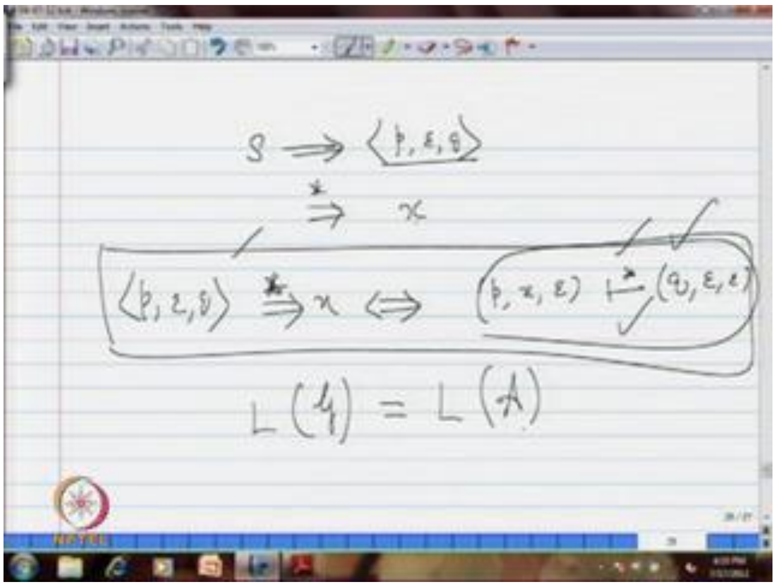
$$(p, A, q) \overset{*}{\Longrightarrow} x \quad \text{if and only if} \quad (p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon).$$

Now, so this is a just a sample construction, and this philosophy, I hope you know through this particular example; what are the corresponding computation; what are the derivation I have shown. I hope you understand, what is the philosophy behind in you know, introducing such a non terminal symbol. And then once you understand that, then the proof part of proof part will be easy. So, just I display this derivation, there is a non terminals the importance of the non ((Refer Time: 32:09)) the way that it is, you know, introduced you understand.

Now, what I prove the construction is valid; to show the construction is valid; that means, the whatever the grammar constructed indeed generates the language ((Refer Time: 32:28)) given PDA, what I do, I prove this statement. You take any two states p q, and a symbol A the stack symbol or epsilon, and any string terminal string; that means, in put string I can call here sigma star. We claim that, this non terminal symbol p A q in finitely many steps if it can derive x, this non terminal x of course, gamma.

Then this kind of configuration; that means, this is the stack information of course, there is nothing else, this is the configuration p x A from this configuration, I will be able to get the configuration q epsilon epsilon and conversely, this is what I am going to prove. By proving this, you know that the constructed grammar indeed generates the language acceptable given PDA, how if you know the first one you will connect to this.

(Refer Slide Time: 33:34)


$$S \Rightarrow \langle p, \varepsilon, q \rangle \overset{*}{\Rightarrow} x$$

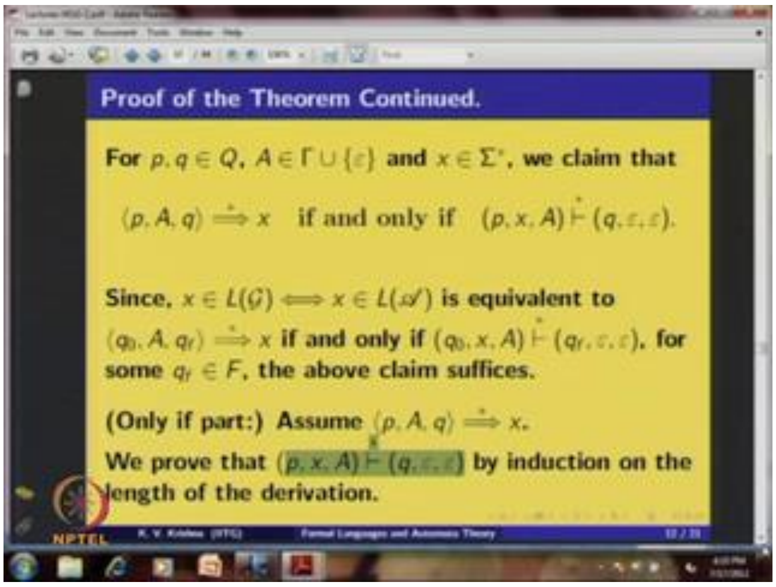
$$\langle p, \varepsilon, q \rangle \overset{*}{\Rightarrow} x \iff (p, x, \varepsilon) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$$

$$L(G) = L(A)$$

The first one you will connect to this, because in the in one rule that you produce p epsilon q to this you will connect, and there after infinitely many steps. Suppose if you have produced x, you see in this, then the corresponding to the computation, you know you have this computation and conversely, so what happens here. So, this production you have considered, now in the computation, suppose if you have the computation, which produce which says that, I will reach to this final state q. So, from the configuration acts epsilon, if I can get infinitely many steps q epsilon epsilon, then what is the meaning of that, through this you know, we are getting this x as this and conversely.

Because, this if and only if, what I am going to prove, if p A q derives x in finitely many steps; if and only if, p x A in finitely many steps, you are getting this kind of configuration. So, by getting this kind of configuration, so corresponding to this computation, this you have p epsilon q produces x infinitely many steps. So, if you if I have correspondence one two one correspondence between these two type of things. Corresponding to this derivation if I have this computation and vice versa, you understand this x can be generated in the grammar, and whenever you generate from here, x some terminal string you know there is a computational in this.

(Refer Slide Time: 35:33)



Proof of the Theorem Continued.

For $p, q \in Q$, $A \in \Gamma \cup \{\varepsilon\}$ and $x \in \Sigma^*$, we claim that

$$\langle p, A, q \rangle \overset{*}{\Longrightarrow} x \quad \text{if and only if} \quad (p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon).$$

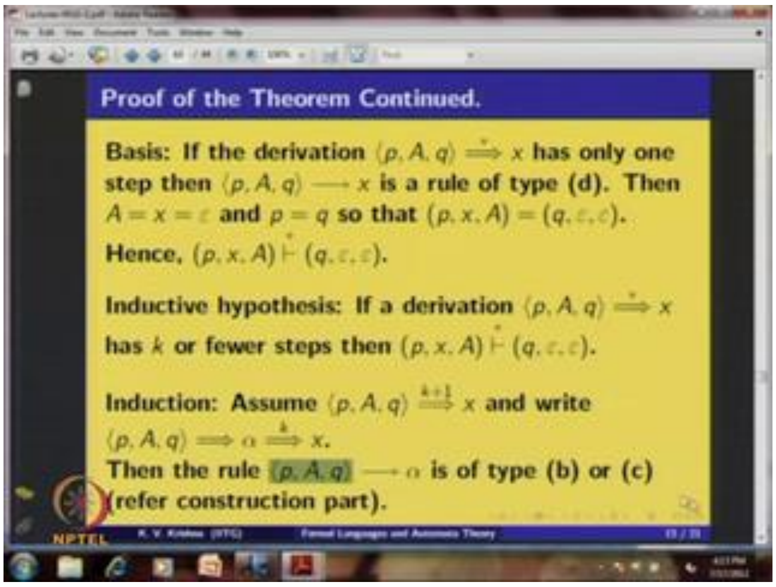
Since, $x \in L(\mathcal{G}) \iff x \in L(\mathscr{A})$ is equivalent to $\langle q_0, A, q_f \rangle \overset{*}{\Longrightarrow} x$ if and only if $(q_0, x, A) \overset{*}{\vdash} (q_f, \varepsilon, \varepsilon)$, for some $q_f \in F$, the above claim suffices.

(Only if part:) Assume $\langle p, A, q \rangle \overset{*}{\Longrightarrow} x$.

We prove that $(p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$ by induction on the length of the derivation.

So, that you can understand that, the language generated by the constructed grammar is same as language ((Refer Time: 35:26)) given PDA, so we get this. How do we do this, this I will, so let me just say that x belongs to L of g if and only if, x is in L of A, which is equivalent to this statement. Now, the only if part; that means, p A q produces x infinitely many steps, then we have to prove this thing. We prove this computation by induction on the length of the derivation.

(Refer Slide Time: 36:02)



Proof of the Theorem Continued.

Basis: If the derivation $\langle p, A, q\rangle \overset{*}{\Longrightarrow} x$ has only one step then $\langle p, A, q\rangle \longrightarrow x$ is a rule of type (d). Then $A = x = \varepsilon$ and $p = q$ so that $(p, x, A) = (q, \varepsilon, \varepsilon)$.
Hence, $(p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$.

Inductive hypothesis: If a derivation $\langle p, A, q\rangle \overset{*}{\Longrightarrow} x$ has k or fewer steps then $(p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$.

Induction: Assume $\langle p, A, q\rangle \overset{k+1}{\Longrightarrow} x$ and write
$\langle p, A, q\rangle \Longrightarrow \alpha \overset{k}{\Longrightarrow} x$.
Then the rule $\langle p, A, q\rangle \longrightarrow \alpha$ is of type (b) or (c) (refer construction part).

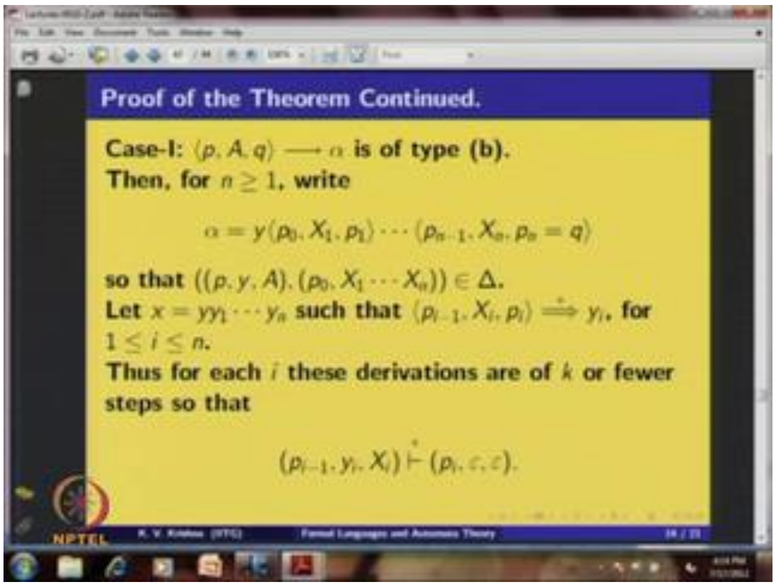
If the derivation p A q has only one step, because see here, I can start with because this is the reflexive transitive closure, one can ask that, what about you know for the basis considering zero steps. You know this is a non terminal symbol, this is a terminal string, you know, here non terminal symbol cannot be you know a terminal symbol here, this type of a special type of symbols that we have constructed. And therefore, here I cannot assume zero number of steps, so let me assume, you are getting this in one step, that is how the basis I have started with.

Then automatically, you understand this arrow x, this non terminal symbol arrow x has to be a production rule, if it is in one step. So, then it is a production rule of type d, if it is a production rule of type d, you can understand A equal to x equal to epsilon and p equal to q, so that you have p x A is equal to q epsilon epslion. So, this configuration and this configuration there equal, and hence you are producing this configuration in zero number of steps, and finitely many steps. And if you this is the basis, so basis is very clear from this.

Now, inductive hypothesis, if you take a derivation of this form p A q produces x in you know, if you are getting that in k or fewer steps, then this configuration gives this configuration in finitely many steps. So, if the derivation is of length k or less than that, then you are getting this in some finitely many steps, that is what is inductive hypothesis. Now, you take a derivation of k plus 1 steps p A q gives x in k plus 1 steps, then let me rewrite this derivation of this form p A q produces alpha in the first step, and after that this alpha produces x infinitely in many steps let me assume like that.

Now, what is the possibility for this alpha. Now, then the rule p A q alpha is either of type b or type c, you know it cannot be type a; type a is of the form s goes to something of this kind of non terminal symbol. So, it is not of type a or it is not of type you know d, because we have here k plus 1 steps. So, it is not of that we have considered, so this is either type b or type c form.

(Refer Slide Time: 38:41)



Proof of the Theorem Continued.

Case-I: $\langle p, A, q\rangle \longrightarrow \alpha$ **is of type (b).**
Then, for $n \geq 1$**, write**

$$\alpha = y\langle p_0, X_1, p_1\rangle \cdots \langle p_{n-1}, X_n, p_n = q\rangle$$

so that $((p, y, A), (p_0, X_1 \cdots X_n)) \in \Delta$.
Let $x = yy_1 \cdots y_n$ **such that** $\langle p_{i-1}, X_i, p_i\rangle \overset{*}{\Longrightarrow} y_i$, **for** $1 \leq i \leq n$.
Thus for each i **these derivations are of** k **or fewer steps so that**

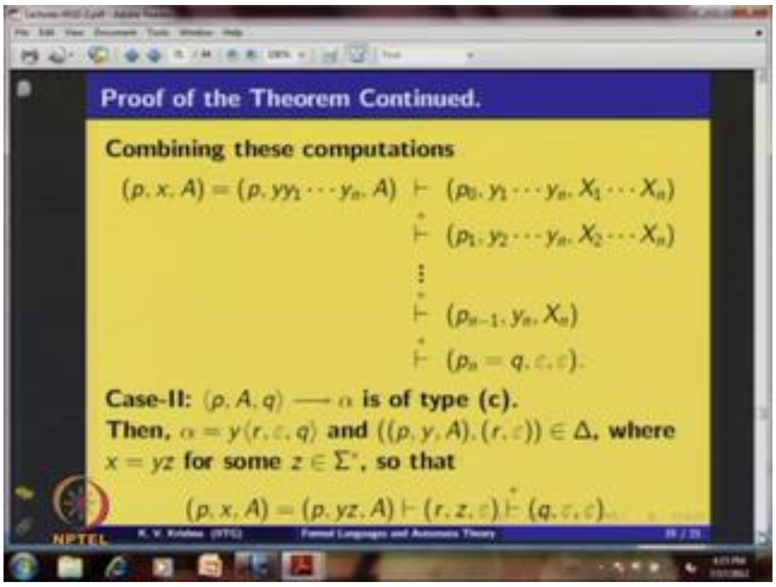
$$(p_{i-1}, y_i, X_i) \overset{*}{\vdash} (p_i, \varepsilon, \varepsilon).$$

Now, if it is of type b, let me consider this case, then alpha then for n greater than equal to 1, we write alpha is equal to y p 0 x 1 p 1 and so on. Something of this form, so that you know, you have this transition rule in the given PDA, I have this kind of production rule means that should be corresponding to this transition. Now, what I will do, I will write x to be you know of this form y, y 1 and so on y n.

Because this is the terminal string now I will break them into this y 1 y 2 y n of course, y is already shown. Corresponding to each non terminal symbol, that it is deriving in the given derivation, such that, this particular non terminal symbol here is producing y i, the component y i in the terminal string, that y i can be epsilon also that is the different, I mean that can be situation.

So, thus for each i this derivations, because the total derivation is of k or fewer steps, whatever we have considered here. And thus whatever that we are producing the string, since it has k or fewer step, I mean it is in k step. So, this each such derivation is of k or fewer steps very clearly. And therefore by inductive hypothesis you know, you get from this configuration, to this configuration in the PDA, this is the inductive hypothesis.

(Refer Slide Time: 40:14)



Proof of the Theorem Continued.

Combining these computations

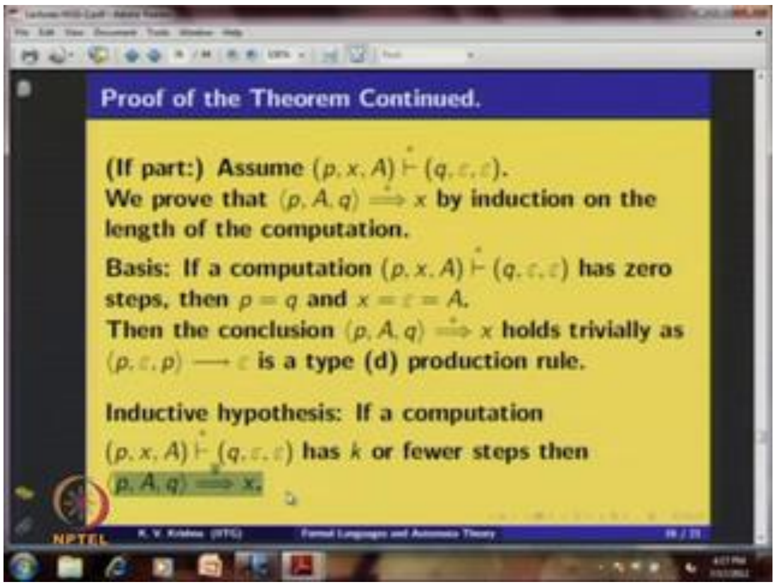
$$\begin{aligned}(p, x, A) = (p, yy_1 \cdots y_n, A) &\vdash (p_0, y_1 \cdots y_n, X_1 \cdots X_n)\\ &\overset{*}{\vdash} (p_1, y_2 \cdots y_n, X_2 \cdots X_n)\\ &\ \vdots\\ &\overset{*}{\vdash} (p_{n-1}, y_n, X_n)\\ &\overset{*}{\vdash} (p_n = q, \varepsilon, \varepsilon).\end{aligned}$$

Case-II: $(p, A, q) \longrightarrow \alpha$ **is of type (c).**
Then, $\alpha = y\langle r, \varepsilon, q\rangle$ **and** $((p, y, A), (r, \varepsilon)) \in \Delta$**, where** $x = yz$ **for some** $z \in \Sigma^*$**, so that**

$$(p, x, A) = (p, yz, A) \vdash (r, z, \varepsilon) \overset{*}{\vdash} (q, \varepsilon, \varepsilon).$$

Now, you see, now combining this computations, I can observe that p x A, that is p y y y 1 and so on, y n A can give that p 0. So that, y you have consumed; that means, you have write, and you have x 1 x 2 x n in the stack. And then, when x 1 is removed, you have consumed y 1 in the input and so on, in p n you have this. So, if it is, so what you, what did you, what did I get then p x A infinitely many steps. I have q epsilon epsilon, if this is rule of type c, then it is very quick, then it is of the form y r epsilon q. And this can be the transition corresponding to that, and where now x can be written as this y z for some z in sigma star, so that you have this computation, you will get this computation in the PDA.

(Refer Slide Time: 41:14)



Proof of the Theorem Continued.

(If part:) Assume $(p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$.
We prove that $(p, A, q) \overset{*}{\Longrightarrow} x$ by induction on the length of the computation.

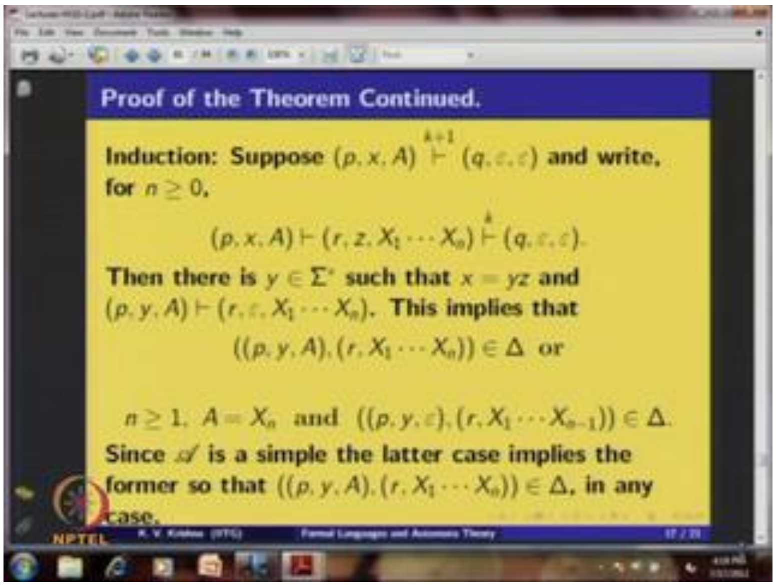
Basis: If a computation $(p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$ has zero steps, then $p = q$ and $x = \varepsilon = A$.
Then the conclusion $(p, A, q) \overset{*}{\Longrightarrow} x$ holds trivially as $(p, \varepsilon, p) \longrightarrow \varepsilon$ is a type (d) production rule.

Inductive hypothesis: If a computation $(p, x, A) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)$ has k or fewer steps then $(p, A, q) \overset{*}{\Longrightarrow} x$.

So, if you have the derivation, what did I get, so by assuming what is called, by inductive by inductive hypothesis on the length of the derivation. We have observed that, you get that particular computation in the PDA. Now, conversely that if part, if I assume this computation, I have to show the derivation. So, we prove this derivation by induction on the length of the computation, so what are the computation under consideration.

If this computation has zero number of steps, here I am saying say for the basis; I am saying zero number of steps, because the possibility is there. Then, it is zero number of steps means these two configurations are same; that means, p equal to q, x equal to epsilon A equal to A is also epsilon. So, in which case, you see this derivation holds trivially as p epsilon p goes to epsilon this is a type d production rule, you see the production rule, we have this kind of rule. And therefore, it is possible in the constructed grammar, and hence we have the basis.

(Refer Slide Time: 42:42)



Proof of the Theorem Continued.

Induction: Suppose $(p, x, A) \overset{k+1}{\vdash} (q, \varepsilon, \varepsilon)$ **and write, for** $n \geq 0$,

$$(p, x, A) \vdash (r, z, X_1 \cdots X_n) \overset{k}{\vdash} (q, \varepsilon, \varepsilon).$$

Then there is $y \in \Sigma^*$ **such that** $x = yz$ **and** $(p, y, A) \vdash (r, \varepsilon, X_1 \cdots X_n)$. **This implies that**

$$((p, y, A), (r, X_1 \cdots X_n)) \in \Delta \text{ or}$$

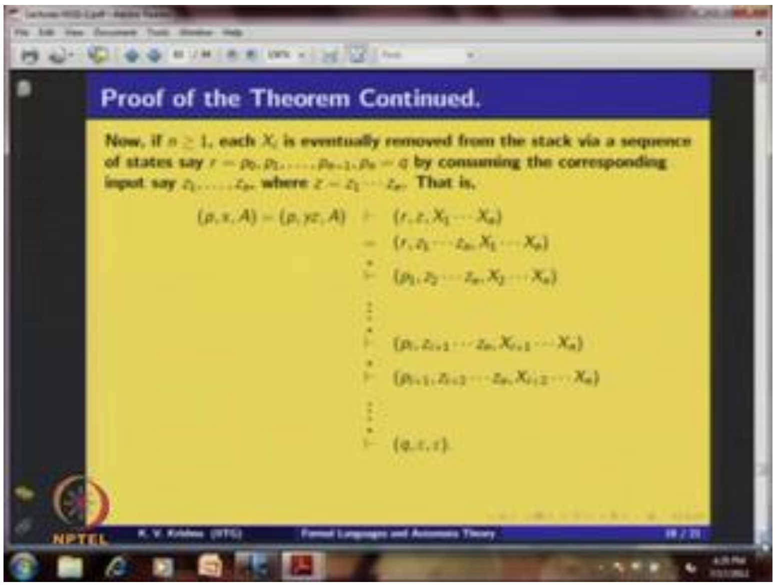
$$n \geq 1, \; A = X_n \text{ and } ((p, y, \varepsilon), (r, X_1 \cdots X_{n-1})) \in \Delta.$$

Since $\mathcal{A}$ **is a simple the latter case implies the former so that** $((p, y, A), (r, X_1 \cdots X_n)) \in \Delta$, **in any case.**

Now, for inductive hypothesis, if a computation is having k or fewer steps, then assume this derivation is possible; that means, this p A q produces x in infinitely many steps. So, for inductive step, you know you just assume k plus 1 steps computation in the PDA. And now, for n greater than or equal to 0, I can write like this, because this say for example, in the first step of the computation you are having this kind of thing; that means, from the stack A, you have removed A and x 1 x 2 x n you would have introduced or the possibility is, this x n can be A.

And you have consumed some input, and you have just left over in the input, and after k number of steps, you have the final configurations out of this q epsilon epsilon, so this is a possibility. So, then there is y in sigma star such that x is equal to y z; that means, that y, so p y A is the transition you know r epsilon x 1 x 2 x n, this is the thing, this implies that you have either this kind of transition are you know for n greater than equal to 1, this A is equal to x n p y epsilon is r x 1 x 2 x n minus 1, this kind of transition. Because this A maybe there, you are not removing that, but you would have added x 1 x 2 x n minus 1, but since what are the PDA that we have considered, it is simple you know, it is sufficient like the latter case implies the former one, and in many case we have this kind of production rule.

(Refer Slide Time: 44:13)



Proof of the Theorem Continued.

Now, if $n \geq 1$, each X_i is eventually removed from the stack via a sequence of states say $r = p_0, p_1, \ldots, p_{n-1}, p_n = q$ by consuming the corresponding input say $z_1, \ldots, z_n$, where $z = z_1 \cdots z_n$. That is,

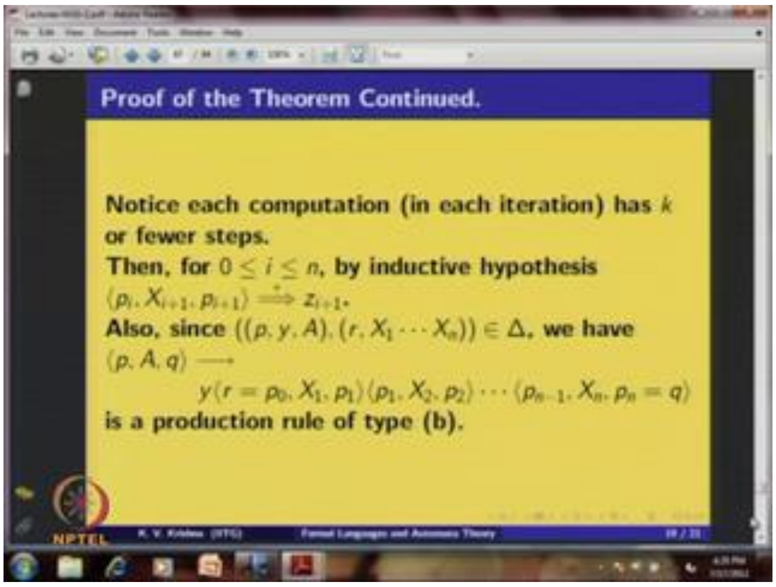
$$\begin{aligned}(p, x, A) = (p, yz, A) &\vdash (r, z, X_1 \cdots X_n)\\ &= (r, z_1 \cdots z_n, X_1 \cdots X_n)\\ &\overset{*}{\vdash} (p_1, z_2 \cdots z_n, X_2 \cdots X_n)\\ &\vdots\\ &\overset{*}{\vdash} (p_i, z_{i+1} \cdots z_n, X_{i+1} \cdots X_n)\\ &\overset{*}{\vdash} (p_{i+1}, z_{i+2} \cdots z_n, X_{i+2} \cdots X_n)\\ &\vdots\\ &\overset{*}{\vdash} (q, \varepsilon, \varepsilon).\end{aligned}$$

K. V. Krishna (IITG) Formal Languages and Automata Theory 18 / 21

Now, you see, the situation is corresponding to this computation. So, p x A that is p y z A, that is how we writing. Now, y is consumed that, what I have this r z x 1 x 2 x n, this is what is there. In k number of steps, you are getting this q epsilon epsilon, but in between you have to remove x 1 x 2 and so on, x n, this stack symbol need to be removed. So, whenever we are removing x 1, what are this what are that string that you are consuming from the input, I call it has you know, z 1 and similarly for x 2 z 2 and so on, by the time you remove x x n, I assume that z n is consumed in the input.

So, I appropriately mention that ((Refer Time: 44:59)) two states intermediate states p 1 p 2 and so on. So, the computation is presented like this, so r z x 1 x 2 x n by the time I finish remove x 1 from the stack, then I have consumed z 1, the remaining is z 2 and so on, z n. So, similarly all these and so on, by the time you finish x n, you have that q epsilon epsilon ((Refer Time: 45:21)) configuration, this is the computation that we have considered.

(Refer Slide Time: 45:35)



Proof of the Theorem Continued.

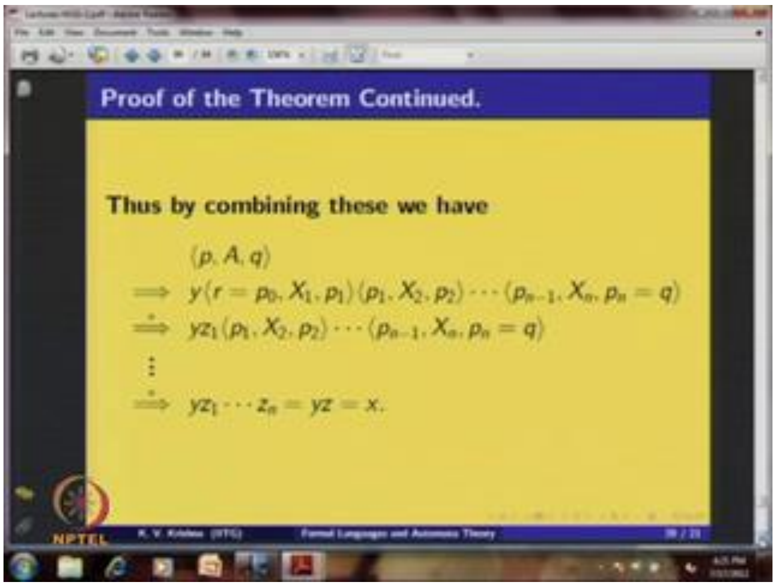
Notice each computation (in each iteration) has k or fewer steps.
Then, for $0 \leq i \leq n$, by inductive hypothesis $\langle p_i, X_{i+1}, p_{i+1} \rangle \overset{*}{\Longrightarrow} z_{i+1}$.
Also, since $((p, y, A), (r, X_1 \cdots X_n)) \in \Delta$, we have $\langle p, A, q \rangle \longrightarrow$

$$y \langle r = p_0, X_1, p_1 \rangle \langle p_1, X_2, p_2 \rangle \cdots \langle p_{n-1}, X_n, p_n = q \rangle$$

is a production rule of type (b).

Now, you see this total computation here, we shown up is k or fewer steps and therefore, by inductive hypothesis notice that each computation has k or fewer steps. Now, by inductive hypothesis, you have this p i X i plus 1 p i plus 1 gives z i plus 1, you know infinitely many steps, that is a inductive hypothesis. And then, since p y A is you know this is the transition you have, ((Refer Time: 45:56)) combine all this, corresponding to that, then you have this kind of production rule this is a type b.

(Refer Slide Time: 46:04)



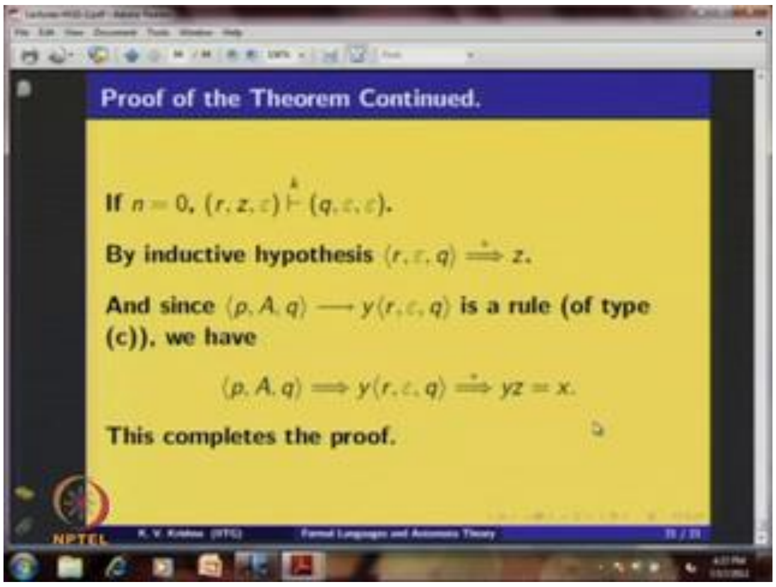
Proof of the Theorem Continued.

Thus by combining these we have

$$\begin{aligned} & \langle p, A, q\rangle \\ \Longrightarrow\ & y\langle r = p_0, X_1, p_1\rangle\langle p_1, X_2, p_2\rangle \cdots \langle p_{n-1}, X_n, p_n = q\rangle \\ \overset{*}{\Longrightarrow}\ & yz_1\langle p_1, X_2, p_2\rangle \cdots \langle p_{n-1}, X_n, p_n = q\rangle \\ & \vdots \\ \overset{*}{\Longrightarrow}\ & yz_1 \cdots z_n = yz = x. \end{aligned}$$

Now, combining this, you get you start from p A q, and then you apply that production rule type b production rule, then there after you know, corresponding to each of these symbols. First you produce z 1 that from the inductive hypothesis I have got this, then z 1 then this is non terminal symbol is lying and so on. You will be producing y z 1 z 2 and so on, z n, that is how you will be producing, because this z 1 you have produced, and then you have a symbol which can be nullified. So, then this p 1 X 2 p 2 from which you can produce z 2 and so on. This p n minus 1 X n p n you can produce z n. So, you have produced x, so thus you have a derivation for x.

(Refer Slide Time: 46:52)



Proof of the Theorem Continued.

If $n = 0$, $(r, z, \varepsilon) \overset{k}{\vdash} (q, \varepsilon, \varepsilon)$.

By inductive hypothesis $\langle r, \varepsilon, q \rangle \overset{*}{\Longrightarrow} z$.

And since $\langle p, A, q \rangle \longrightarrow y\langle r, \varepsilon, q \rangle$ is a rule (of type (c)), we have

$$\langle p, A, q \rangle \Longrightarrow y\langle r, \varepsilon, q \rangle \overset{*}{\Longrightarrow} yz = x.$$

This completes the proof.

K. V. Krishna (IITG) Formal Languages and Automata Theory

So, this is one case, and if n is 0, because you look at here, if ((Refer Time: 46:57)) than equal to 1, then I have shown this. So, corresponding this, I have this particular type d type b production rule, and then I have this derivation in the grammar. If n is 0, then simply you have this situation, guess steps you are getting this. Now, by inductive hypothesis this r epsilon q produces z in finitely many steps, and since this p A q produces y r epsilon q is a rule, this of type c, then you have this kind of derivation. And thus, you know we have the proof.

So, now you understand like you know, whatever the context free grammar I have constructed, I have shown this particular statement that, whenever I have a some computation there, corresponding to I have a derivation. So, for which what I did, I have applied induction on the number of steps in the computation and derived it. And conversely, if you consider a derivation in the grammar, we have shown that the computation the corresponding to that derivation there is a computation also in this.

So that, you know, we have observed that, the language generated by the grammar constructed is essentially same as the language accessible by the given PDA. So, we have shown the equivalence like this. So, this I hope, you know gives an idea, how to prove based on the number of steps, by induction on the number of steps of the derivation or the computation for the previous one also, whatever I left is an exercise, I hope you can complete that exercise. So that, you observe that PDA, that push down automata and context free grammars they are equivalent. So, with this I conclude this lecture.