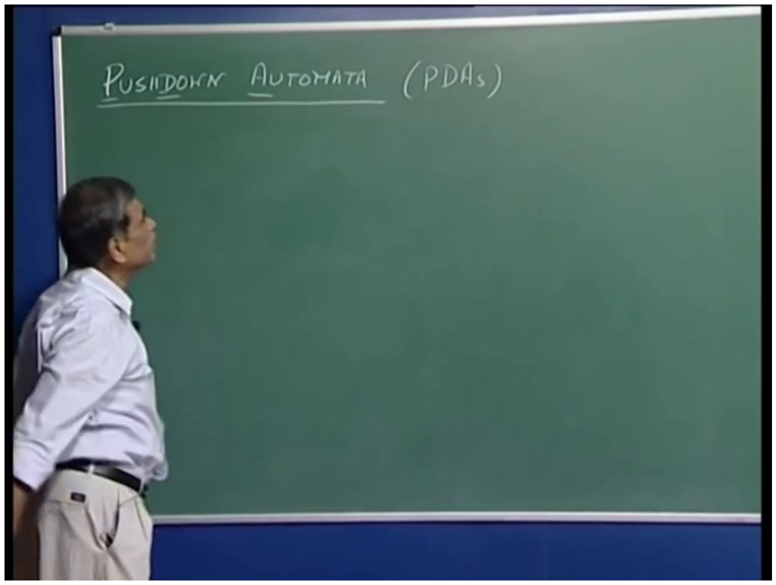
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Lecture 31
Module 1
Introduction to pushdown automata (pda).

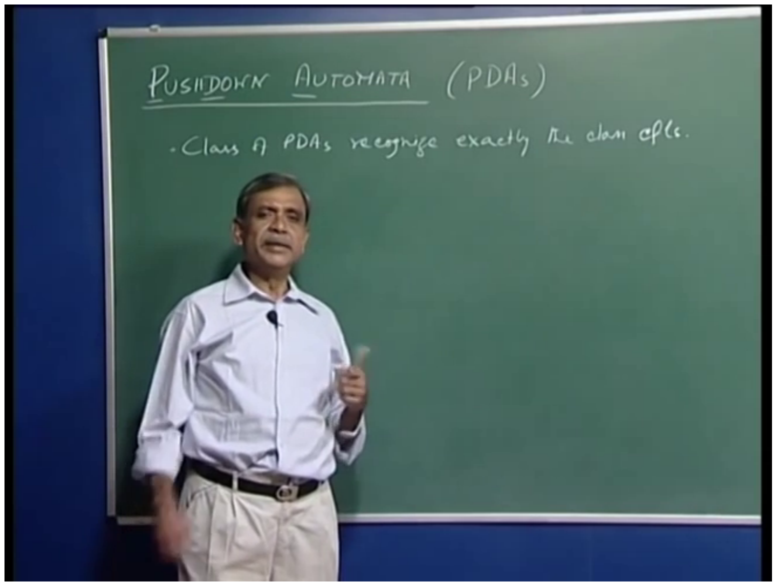
We shall now study a new class of automata called push down automata PDA these three letters constitute the acronym for this class of automata.

(Refer Slide Time: 0:26)



So for short we called them PDA's push down automata or PDA's these PDA's we will see they recognize the class of push down automata this class recognizes precisely the class of context free languages.

(Refer Slide Time: 0:54)



PUSHDOWN AUTOMATA (PDAs)

- Class of PDAs recognize exactly the class of CFLs.

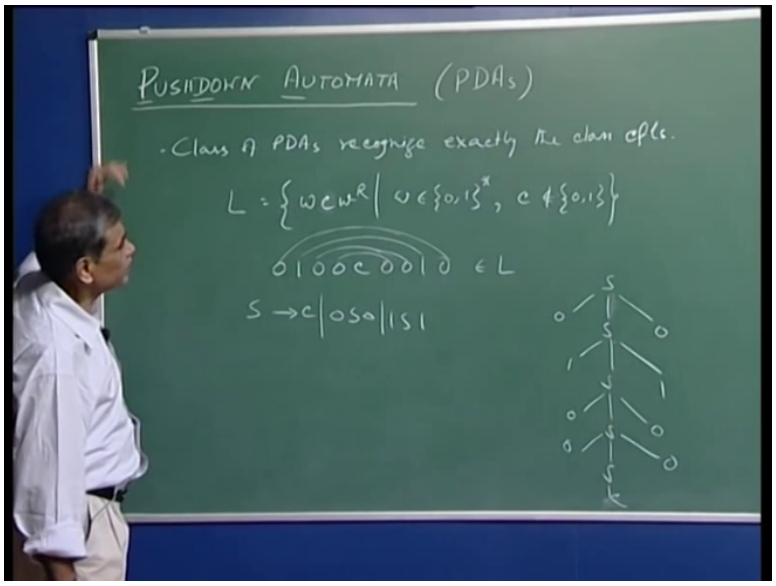
So let me write this important fact class of PDA's recognize exactly the class of context free languages this is something we will show in a while.

The point about this PDA's is that so far we have studies context free languages in terms of context free grammars remember that grammars are generative devices we say a string is generated a grammar in particular given a context free grammar G that will generate a set of string which we will constitute a context free language this is something we have seen but now what we are kind of moving to is a machine characterization of context free languages now remember that machines or automata they are not generative devices, right?

There will be some inputs string and the machine will decide whether or not it will accept that string, if it accepts that string it is in the language accepted by that machine otherwise that string is not in the language accepted by that machine. So all these things we had studied these notions we had seen in term when we discussed finite state machines and so all those notions will be there that is for every PDA there will be exactly one language which is the language accepted by that PDA.

So all these notions we will formulize but first of all let me give a kind of informal description of what these PDA's are. In fact let me motivate ourselves to these machines to these automata by means of an example.

(Refer Slide Time: 3:25)



PUSHDOWN AUTOMATA (PDAs)

- Class of PDAs recognize exactly the class of CFLs.

$L = \{wcw^R \mid w \in \{0,1\}^*, c \notin \{0,1\}\}$

$0100c0010 \in L$

$S \rightarrow c \mid 0S0 \mid 1S1$

So consider this language L w c w R where w is a string in over this binary alphabet and c is some symbol some terminal symbol which is different from 0 and 1. So an example of a string in this language would be 0100, then C and then 0010 in other words this string is the language because the part before the portion of the string before C namely 0100 is the reverse of the string which occurs after C.

So you can see that if I reverse this string I will get 0010 and that is the string which is there, so therefore this is in the language here. We can very easily show that this language is not a regular language because we can use the regular language pumping lemma to show that this language is not regular and we can also similarly provide a context free grammar which will accept this language, and what is that grammar? In fact let us do it right way so we can say that S this is the start symbol goes to you know C as well as 0S0 as well as 1S1, right?

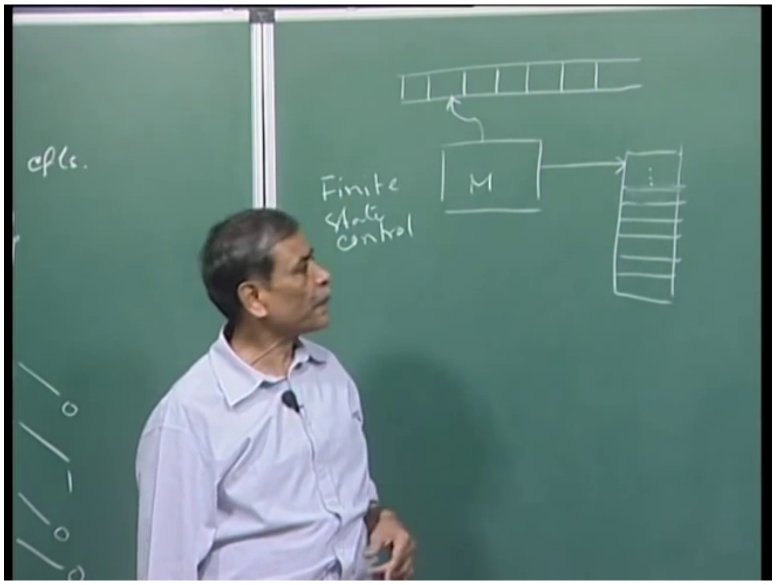
So there is only one non terminal S and the remember we said the 0, 1 and C are the terminal symbols so this grammar I am not writing the set of non-terminals which is of course just 1 this S and you can see that this particular string will be generated from is by this grammar we will generate the string because from S we go to S you know the outer words first, so 0S0 then again 1S1 then 0S0 0S0 and this particular S is rewritten as C, right?

So the string generated is 0100C0010, right? Now notice the reason this grammar could generate this particular string is because the correspondence between this 0 and this 0 that is being taken care of. Similarly between this one and this one that this correspondence was taken care of in this rule in this using this particular rule in its application. And we had remarked some time ago that context free languages can very nicely capture you know this intuitive notion of nesting that this particular string is nested between these two 0's and that correspondence is what you know I pointed out here.

And you can also see again intuitively why this string or strings like this cannot be recognized by finite state machines the problem is that the correspondence that is if in the beginning 0 has to happen then at the end also that 0 has to happen, similarly if the kth letter is 0 from the left end that kth letter from the right end also should be 0 till we see the C. So this kind of nesting you know this is there is this correspondence and the other correspondence is nested within it and so on, right? We had remarked about this kind of intuitive property of strings generated by a context free grammar.

And now we would like to capture this idea in a push down machine or in a machine. The what push down machine or push down automata will have? It will have as usual as in the case of finite state machine it will have a finite state control. So again we will elaborate all this but the point is that such a machine also has a finite number of states, right? But what it has which is in addition to something different from what finite state machines they have is this machine is also equipped with a stack or you know another name of stack is push down store.

(Refer Slide Time: 9:40)

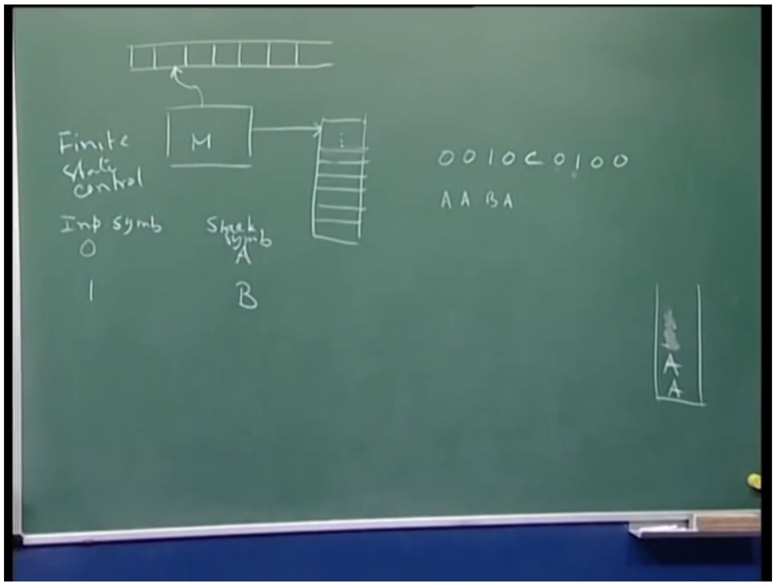


So let me draw a picture, so such a machine would look like that this is the part which is the finite state control of the machine. Now it has two heads two if you like two heads and one head is on a for reading the input, so this side is somewhere here and at this moment if the head is here whatever symbol is here that is the symbol the machine is scanning. In addition to this head there is another head and this head is for reading the top symbol of a stack, so this is the stack. And mind you that the stack is something which is can grow arbitrarily large similarly can shrink also so you know a stack can grow in size and again it can come down again grow and so on.

Now the machine can actually you know the move of the machine is basically will be that it reads a symbol it reads something on the top of the stack and depending on these two information and of course depending on the current state the machine is in it decides, it decides to do what of course it will take a decision about what will be the next state it will also decide about what to do with the top symbol. So it what essentially in general what happens is the top symbol is rewritten by a string, alright?

So in fact I will give the formal definition a little later but let us see how this particular language will be recognized or accepted by this by a machine like this.

(Refer Slide Time: 12:12)



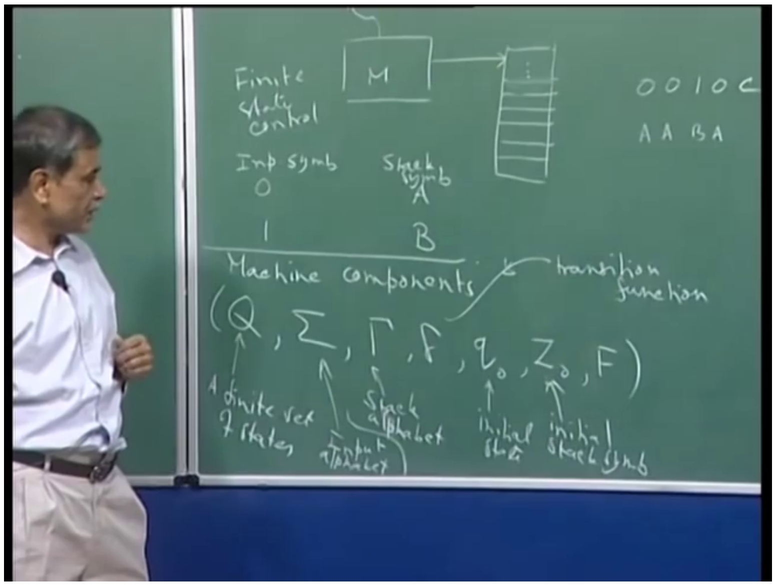
See what happens is before the C comes whatever symbol the machine sees in the input a symbol corresponding to you know 0 or 1 will be written in the stack. So let us say we make this correspondence that we will have for input symbol 0 we will have a corresponding stack symbol let us say A and for input symbol 1 we will have a stack symbol which corresponds to it B and let see how what is this idea of correspondence, what happens is for example it sees this string 0010C0100.

So in the beginning it sees this 0 and because in the beginning it has seen this 0 it writes in the push down stack the symbol which corresponds to 0 so in that it is A. Then it sees you know it has consumed this particular symbol so it comes here it will see another 0 and corresponding to that 0 it will write the symbol A, now it sees a 1 and therefor it will push on the stack this symbol corresponding to 1 and again it will see a 0 here so it will do a 0 and now you see when it sees the C it knows it has seen the first part and what it should now see is the reverse of the part it has seen so far but you see if you see what is there on the stack and if you go from top of the stack to bottom that spell out this particular string the corresponding string which is this.

So basically remember 0010 the we according to our correspondence the string is AABA and now from the reverse it is ABAA and that is exactly what is the string which we have on the stack and now what it can do it will see a 0 and the top of the stack is A, so therefore this 0 does indeed has a it knows it has a matching symbol which is this here which is this. So now therefore it checks it out and then again it sees 1 and that matching symbol is here so it is B then 0 then 0, right?

So at the end of it when the stack is empty the machine would know that the string that it has seen so far is indeed the kind of string which is there in the language so therefore it will accept that string. So now you can see from this very informal description how we can formulize this kinds of machines.

(Refer Slide Time: 15:50)



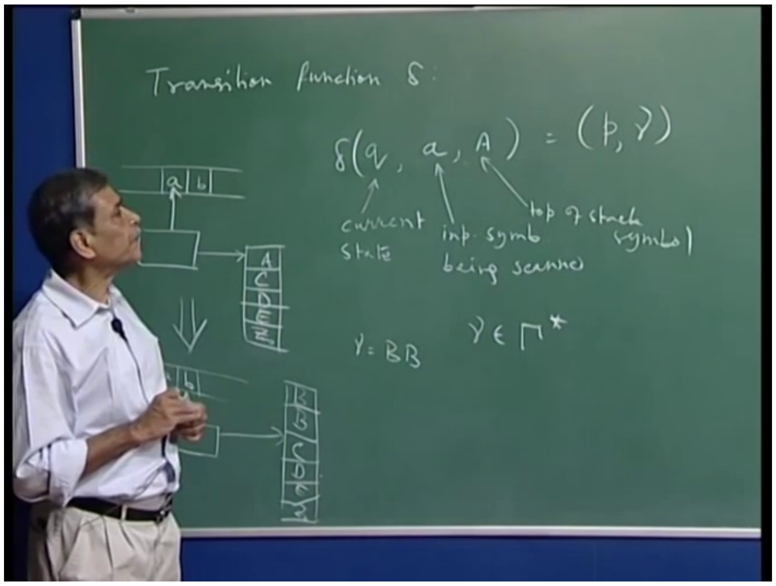
So you see you can clearly see what is such a machine will have of course first of all so let me write here machine components it of course has a set of states and this is a finite set of states, then to define this machine I must say what is the input alphabet here in this example the input alphabet was 0, 1 and C.

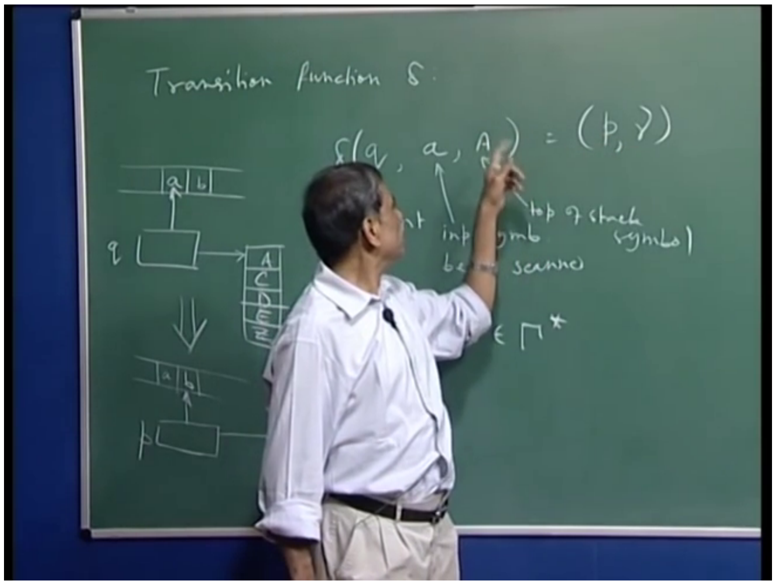
So there is an alphabet which I can say the input alphabet, right? Input alphabet, then you are also writing you know strings on the you are reading the top of the stack and so on you are writing something on the stack. So the stack has its own set of symbols or an alphabet, so let me call that this as the stack alphabet which is capital gamma so this is stack alphabet. Then the machine is making moves then that moves for that move we must follow a certain transition function and that will called delta and we will elaborate we will see exactly what this what is the form of this delta etcetera.

And what are the things to remember like in finite state machine we have to say what is the initial state to define a finite state machine. So here also we need to say what is the initial state so let me say q0 will be the initial state, right? Also for reasons or we know we will have a convention that usually the stack is not empty means stack is empty on a very very special occasion. So to begin with there will be some something in this stack and what is in the stack there will be some initial stack symbol the stack would initially will have only that initial stack symbol, right?

Let me call that as z0 initial stack symbol and of course it like any machine like finite states machines also you know it will have some set of final states or accepting states so let me call that F. So it is not difficult to see in order to formularize or describe or define such a machine I need to provide all these 7 components of the machine namely the set of states the input alphabet, the stack alphabet, the transition function this was transition function then the initial state the initial stack symbol and the set of finial states.

(Refer Slide Time: 20:08)





So now let us try to say what this transition function delta is, transition function delta, what is a transition? Transition is in this case the machine is in some state so at a given step you can see the machine has to be in a particular state so it can be in some state q and then the input head is scanning some symbol in the tape which is the input tape, right? An input this particular head and you should be it is kind of clear we are not writing anything here only writing that we do is in the on the stack.

So this is we can think of this step as a read only tape and what more we will see that this head never moves back this head will move from left to right. So coming back to what a description of a transition function at a particular point of time the machine the PDA is in a certain state q it is scanning an input symbol some symbol in the input tape call it A and what is this read I mean the stack head doing, it is scanning the top of the stack that is the top stack symbol it is you know it is a kind of convention or it is by definition when we use stack we can access if you have a stack we can at a certain time we can only access the top of the stack, right?

So we can push that we can push something on the top of the stack or we can pop the top symbol of the stack. So it is therefore seeing some symbol A on the top of the stack, right? So let me say this is the current state symbol input symbol being scanned and top of the stack symbol, right? And depending on these three things what the machine will do? The machine will seeing this it will go to there will be some next state p and what is the other thing that will happen that you know it will manipulate the stack.

By manipulating the stack what we will always mean is what is written in place of the top of the stack symbol. So if I write this so gamma is therefore some string over the stack alphabet and suppose we say that what it means is that whenever the PDA is in state q scanning the input symbol a with top of the stack being capital A then what is going to be the situation so let me write it so that there is no confusion or draw a picture rather that this is your input here it was a and this is the machine it is in state q and on top of the stack symbol is A there can be some other things.

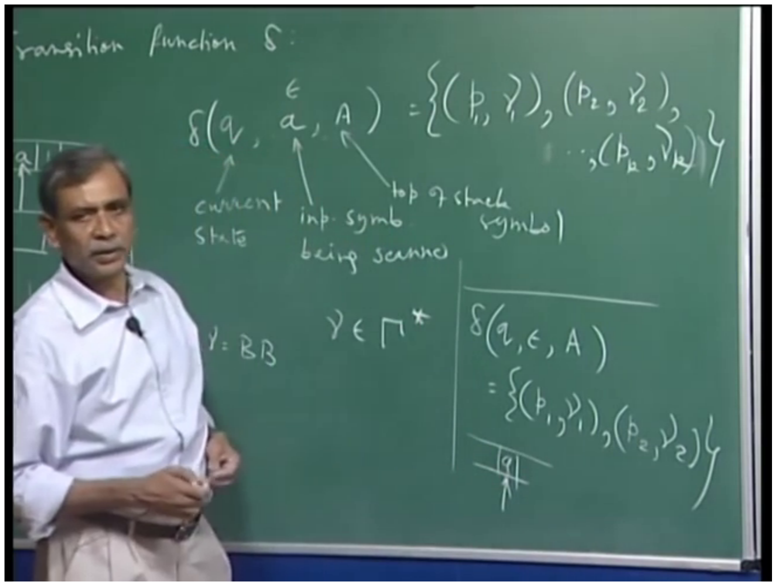
If we mean if we say that the transition function says that from you know from this the next state is p and the string that is written is gamma. So in particular let me just me see that in this case gamma is let me say BBA BB just BB, right? So then from that so you can see this is supposing this is the situation in a particular point of time then in the next instant and let me also say here the symbol was b let us say. If the machine has made use of that transition that means now you know it has consumed this particular symbol has been use of.

So it has consumed the a this symbol and so right now the head has moved one square to the right, and what is happening to the stack? So let me just say what was written below was maybe C, D, E and here your bottom of the stack symbol. Remember we said that such a thing means the top of the stack symbol is replaced by gamma and we have taken this as an example that gamma is the string BB so it will mean that what new stack will be or currently what will be there in the stack is BB in place of this symbol A, so that is the point that this symbol is rewritten by this string gamma.

And then the other things of course would not be disturbed so C, D, E and z0, right? So and the machine is right now in state p. So this is one move of the machine such a move will be there such moves will be there and these are what the transition function will specify. Now right away I should tell you that we will always consider not see this is a kind of deterministic thing that if the this situation if I write like this that whenever the machine is in state q the input symbol is a and the top of the stack is A then the machine goes to state p and replacing A with gamma that is determinism, right? If this happen and it has to happen next state.

But will right away make our definition of PDA's to be non-deterministic, where can non-determinism enter.

(Refer Slide Time: 28:12)



ransition function δ:
$\delta(q, \overset{\epsilon}{a}, A) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_k, \gamma_k)\}$
current state
inp. symb being scanned
top of stack symbol
$\gamma = BB$
$\gamma \in \Gamma^*$
$\delta(q, \epsilon, A) = \{(p_1, \gamma_1), (p_2, \gamma_2)\}$

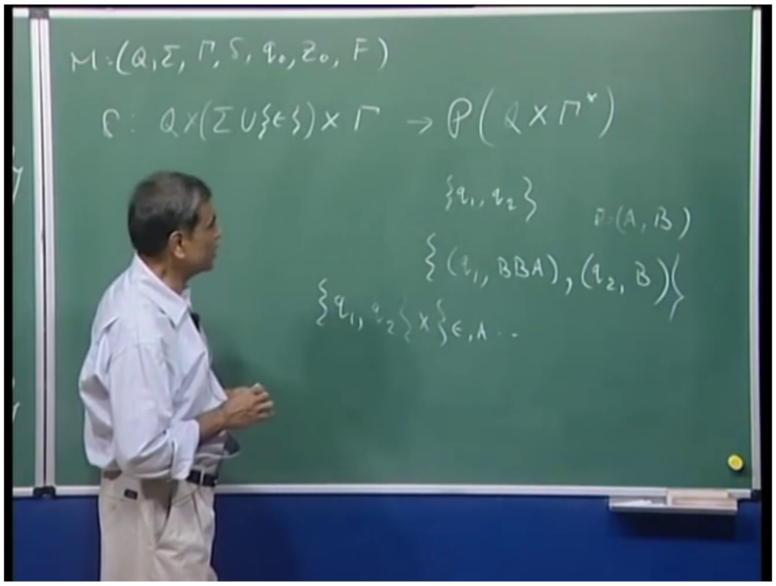
See it can enter in two ways, the first way is obvious that suppose I had a number of such tuples it is pk gamma k, okay. So we appreciate this would mean that suppose the machine is in state q scanning the symbol input symbol A top of the stack symbol A then either the machine moves to state p1 writing gamma 1 in place of A or it moves to state p2 writing gamma 2 in its in place of A and so on.

Or it goes to state pk writing gamma k in place of A. So this is one obvious way non-determinism comes in, we will also allow something which corresponded to the notion of epsilon move in for finite state machines, what was epsilon move? Remember that a finite state machine on an epsilon move could do what it could do otherwise basically change state without consuming the current symbol in the input.

So this is something we will allow also, so what it means is that this a also could be not just a concrete symbol of the alphabet input alphabet it could also be epsilon. So in that case what would it mean? Independent of supposing it to us let me write it this way that, suppose delta cube epsilon A was p 1 gamma 1 and p 2 gamma 2 this set what it would mean? It would mean that if the machine in state q independent of whatever is in the input the symbol that is in the input it could change state to p1 or p2 writing gamma 1 or gamma 2 but the point is suppose at that time the machine was scanning some symbol the input head does not move forward, right?

So given all this how should we write this delta let me clarify this because you see there now as I said there are two kinds of non-determinism we can or we should take care of one is in terms of state what is to be written on the stack that pair as well as whether the input symbol is to be consumed or we are resorting to a epsilon move.

(Refer Slide Time: 31:50)



$M:(Q,\Sigma,\Gamma,\delta,q_0,z_0,F)$
$\delta: Q\times(\Sigma\cup\{\epsilon\})\times\Gamma \rightarrow \mathcal{P}(Q\times\Gamma^*)$
$\{q_1,q_2\}$
$\Gamma:(A,B)$
$\{(q_1,BBA),(q_2,B)\}$
$\{q_1,q_2\}\times\{\epsilon,A\ldots$

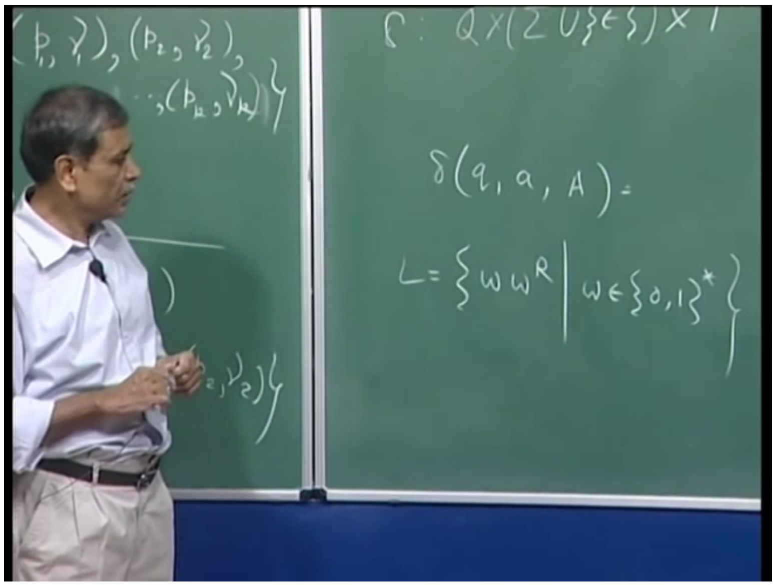
We will say therefore that the transition function of a PDA so let me write the PDA as q sigma remember this is the set of states this is the set of the input alphabet gamma which is the tape alphabet then delta which is a transition function q0 which is the initial state z0 the initial stack symbol and the set of final states and then if the machine is this then the transition function is a mapping, right?

So from Q cross remember we said that the this transition function will be defined it can make use of epsilon moves in the for the input symbol. So the for the this part second component here we can either have a symbol of the input alphabet or epsilon, right? Cross gamma, what does it mean? That this the argument is a triplet first is a state of the machine, second is an input symbol or epsilon, third is a top of the stack alphabet and this is a mapping to power set of what Q cross gamma star, right?

So let us understand this, this is not you know we are writing like this it might be slightly confusing what is let say I had Q to be q1, q2 and I had some gamma to be A, B and so let us say in a particular move the machine could either go to let us say q1 and the top stack symbol would have been replaced by BBA or let us say q2 and let us say B, okay. So in that case you see what is Q cross sigma star what is this set? This set is q1, q2 cross all of these strings like you know whatever gamma you know, right? There infinitely many string.

So this will be the set of all tuples the first or set of all pairs first of which is a state the other is a string and a power set of that will be a subset, so this is a subset of Q cross gamma star, right? So this is that is about it we are just writing like and it should not confuse us.

(Refer Slide Time: 35:35)

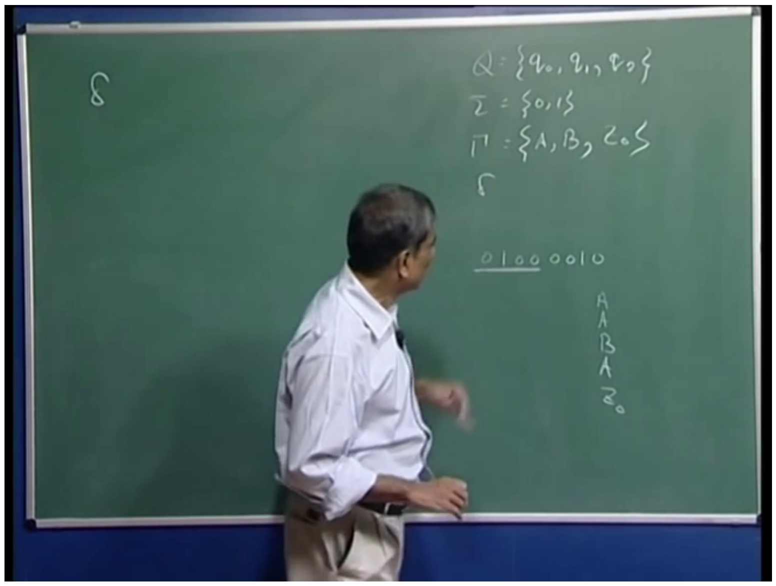

$(p_1, \gamma_1), (p_2, \gamma_2), \dots, (p_k, \gamma_k)\}$
$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$
$\delta(q, a, A) =$
$L = \{ w w^R \mid w \in \{0,1\}^* \}$

And now the point is that you can see we could have used q, a, A our old example so this now is so we need to define what is for suppose q is Q this is a, this is gamma then what should be what are the moves so that is what your delta would give, right?

So you know like whatever I had you can see that is a subset of Q cross gamma star. And now let us take an example to concretize what we are saying we will write down the transition functions properly and let me take the example of a slightly in fact language which is similar to what we had seen but which will necessarily use non-determinism and that language is w w R, w is a string over 0, 1 star. So you can see that in a previously I had in between this w and w R I had a symbol C which you would you know necessarily indicate the first half of the string is over and the second half the reverse part is now to be expected but now I will have just a binary string, right?

So the machine will need to you know standard way we can imagine what it will do it will guess the middle as come, so therefore it will make use of non-determinism to capture or to recognize strings like this. So again informally first let me describe what such a machine will be which will accept this language and in fact why am I saying that this is somewhat in formal because I have not yet defined the notion of what does it mean to say that a language is accepted but informally we understand something about acceptance and that is what is enough to describe the machine which will take of or accept this language.

(Refer Slide Time: 38:09)

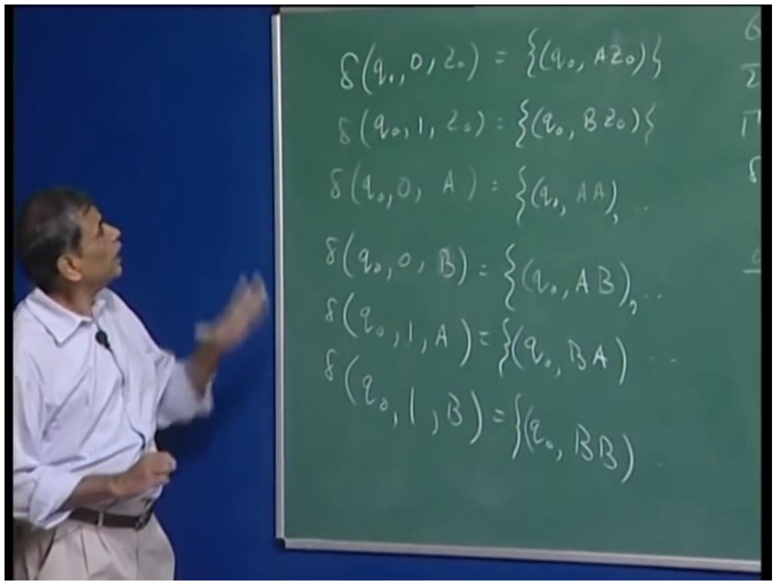


So let me describe a delta for a machine which will accept this language and we will see that we can do with 3 states so therefore our Q will be q0, q1, q2 and of course right away I now the sigma is 0 and 1. Now what I indicated in the previous example that we would use a stack symbol corresponding to 0 and a stack symbol corresponding to 1 and so we would have we said that for 0 we can use A, for 1 we can use B and we will also need you know initial stack symbol which we call this is z0.

So these are the three symbols which are there in for which will be which can which will constitute the stack alphabet and now the delta is what I should describe. So let us see how we will go about doing this, so the idea is simple that supposing I had a string like 0100 and 0010 the idea is something what we had said before that for whenever so long we are in the first half we will keep pushing the corresponding symbol, so 0 for 0 we should push.

So initially remember this z0 is there in the stack so ABAA this is what will happen and now after the first half it is a matter of checking that the string should correspond to this in the manner so 0 will correspond to A so this will get wacked out and so on, right. And you can see that how do I know when the middle of this you know we have seen the first half that will that the machine will guess and that is when non-determinism comes in, right.

(Refer Slide Time: 40:45)



$$\delta(q_0, 0, z_0) = \{(q_0, Az_0)\}$$
$$\delta(q_0, 1, z_0) = \{(q_0, Bz_0)\}$$
$$\delta(q_0, 0, A) = \{(q_0, AA), \ldots$$
$$\delta(q_0, 0, B) = \{(q_0, AB), \ldots$$
$$\delta(q_0, 1, A) = \{(q_0, BA) \ldots$$
$$\delta(q_0, 1, B) = \{(q_0, BB) \ldots$$

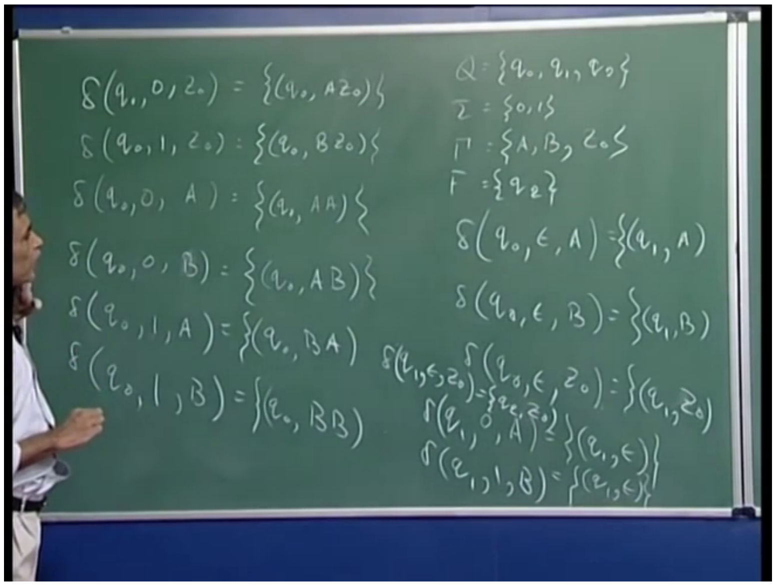
So it is not difficult to see for this strategy so to begin with if I was in the machine is in state q0 and it sees a 0 and of course the state the top of the stack symbol is let us say z0, then what it should do? It should push the corresponding symbol, so it would been without changing state so q then for 0 I had A, right? And remember the top of right now what is the top of the stack symbol which is z0 if I just write q0 A it would mean z0 is replaced by A but that is not what we want to do we want to add this symbol on top of the stack so z0 is replaced by A z0, right?

Then it would mean that this z0 is there this is replaced by A z0 so therefore in effect what I am doing is I am just pushing the symbol A, right? Similarly delta q0, 1, z0 it should be q0 B z0, right? And if we could also the thing is q0, 0 and let us (())(42:30) let me just talk of the pushing phase so we are in q0 we see 0 and top of the stack is A then if I consider that it is this 0 is part of the first part then what I should do is of course you know go I should be in the same state and for this 0 I should put an add an A, right?

We will need something more I will come to that a little later and similarly delta q0, 0, B, right? This will be q0 for this 0 I should write an A, right? This is in the we will worry about little later whether anything else is needed there. And in a correspondingly we will also have these other two 1, A q0, BA, right? We need to push for 1 B again I will worry whether I need to put something here and here what I need to is supposing this is the other thing, right? 0, 0 and here we are doing 1 and suppose I had B q0 then corresponding to this one I will push a B okay and we will what it means.

Now I should remember we said that we also need to guess when the middle has come that can happen whatever symbol you are scanning it is possible in q0 the middle has come, right? It is possible.

(Refer Slide Time: 45:46)



$\delta(q_0, 0, Z_0) = \{(q_0, AZ_0)\}$

$\delta(q_0, 1, Z_0) = \{(q_0, BZ_0)\}$

$\delta(q_0, 0, A) = \{(q_0, AA)\}$

$\delta(q_0, 0, B) = \{(q_0, AB)\}$

$\delta(q_0, 1, A) = \{(q_0, BA)$

$\delta(q_0, 1, B) = \{(q_0, BB)$

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{A, B, Z_0\}$

$F = \{q_2\}$

$\delta(q_0, \epsilon, A) = \{(q_1, A)$

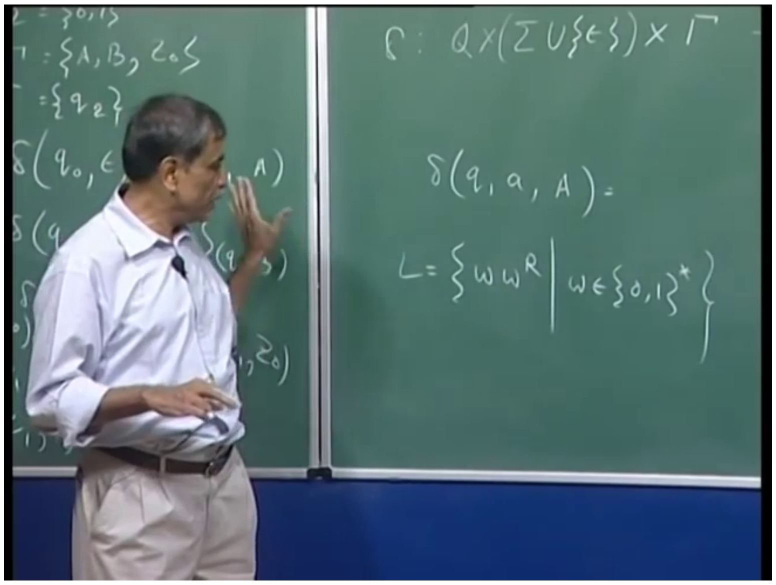
$\delta(q_0, \epsilon, B) = \{(q_1, B)$

$\delta(q_0, \epsilon, Z_0) = \{(q_1, Z_0)$

$\delta(q_1, \epsilon, Z_0) = \{q_2, Z_0)$

$\delta(q_1, 0, A) = \{(q_1, \epsilon)\}$

$\delta(q_1, 1, B) = \{(q_1, \epsilon)\}$



$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$
$\delta(q, a, A) =$
$L = \{ww^R \mid w \in \{0,1\}^*\}$

So we why don't we have one particular epsilon move to go to a state which we should be once the middle has come, so let me say that state is 1 so 1 indicates if we are the machine is in the state q1 it would indicate that the first half is over and we are in the second half, right? So delta why do not we simply do this that q0 on epsilon, see at this time we are just making the decision that we have seen the first half, right? And suppose this is 0 then we go to q1 and all it means I am sorry this third component cannot be 0 it can be either A or B so if it is A you do not change the thing and similarly delta q0 epsilon B it is just we are making the decision that we have seen the first half so we just go to on epsilon without doing anything on the stacks, so this will be like that only and in this and you see it is also possible that in q0 on epsilon the top of the stack is z0 you will go to you can go to q1 and without changing the top of the stack.

When we need to do this? Because you see in this language can be the string epsilon is in the language, right? So initially the input is empty there is nothing there only an the stack only has z0 we should then of course decide we should be able to decide in such case that the first half is over, so this is the thing that we are doing. And do you see now the q0 moves are, right? So this is what is happening, now we are in q0 so let me now in fact say that this is it, right? Let us see whether, and then I need to elaborate what q1 moves are, right?

In q1 the idea is in q1 I should check out check off, right? In q1 if I see a 1, right? And then here top of the stack of symbol is B then that means it is checking out remember 1 corresponds to B so I am in the second half so let me write it, the delta q1 that means we are in the second half in the input we are seeing a 0 in the top of the stack symbol is A this is good, right? This is what we expect that this 0 therefore can be checked off with the top of the stack symbol by checking off what do you mean is that we pop this A so now that this particular A has been checked off.

So remain in q1 and write epsilon, so is it clear what is happening? In such a case it means the top symbol A is popped when A the A here I have the top stack symbol and here I have epsilon that means for A I am writing epsilon in effect it means that top symbol is being rewritten. So this is again is a good case we are in the second half as we are in state q1 we are seeing the symbol 1 here, right? In the input and on the top of the stack we have B which is fine so that means we can check it out so that means it will be q1 epsilon.

We have defined a transition for you know from state q1 on both 0 and 1 in the input but we did not say anything about what happens when this what I am trying to say is that for q0 I mean this is okay this is that we are just checking out checking off the stack, right? 0 checked off with A, 1 checked off with B, here this is what is important from q0 on epsilon we are going to z0 which is okay. Now in q1, right? If you see z0 on top of the stack then what does it mean? It could mean that everything is alright, right? That some string was there the first half which was put on the stack and now z0 has been exposed because we are in the second half and we are seeing z0, right?

So that means what so let me write this and delta in q1, right? On epsilon and z0 I should go to a new state q2 I can say z0 what and this q2 will be my final state, okay. So what I am saying is that we are in the second half we have exposed the bottom of the stack symbol, so one good possibility is that if the input is also over that means we have taken care of the checking part of the first half because so far we have been successful, right?

You see the checking part would not be you know you will not succeed in q1 we are seeing a 0 but the top of the stack is B. So in that case we have no transition defined so the machine would not proceed any further but this means it is in the right kind of situation we have checked off all the first half with the second half and now we all that is left in the stack is z0, so we should go to a state indicating that we have reached the accepting or final state so this is q2.

So what I am going to do in the next lecture we will try to informally at least prove the correctness of what we did you know right now it is looking all very clumsy because so many things are there on the board but we will be able to see once we tell you what is the notion of acceptance that whatever we have done so far indeed we will be the machine that would accept this language and we will continue with this example after we prove after we define what the notion of acceptance is and then we will again look at this example and convince ourselves that this language is indeed accepted this machine, right?

So we have actually defined the entire machine we have given the Q which is the set of states sigma, gamma and delta and we have been we have told you clearly that initial state is q0 and the z0 is the initial stack symbol. So you see all the seven components we have given four here that is the description of delta all this and initial state is q0 the initial stack symbol is z0 and the set of final states is just this state q2. So all the seven components of the machine have been defined so we have completely actually defined the machine but we need to convince ourselves this machine indeed does what we wanted to do.