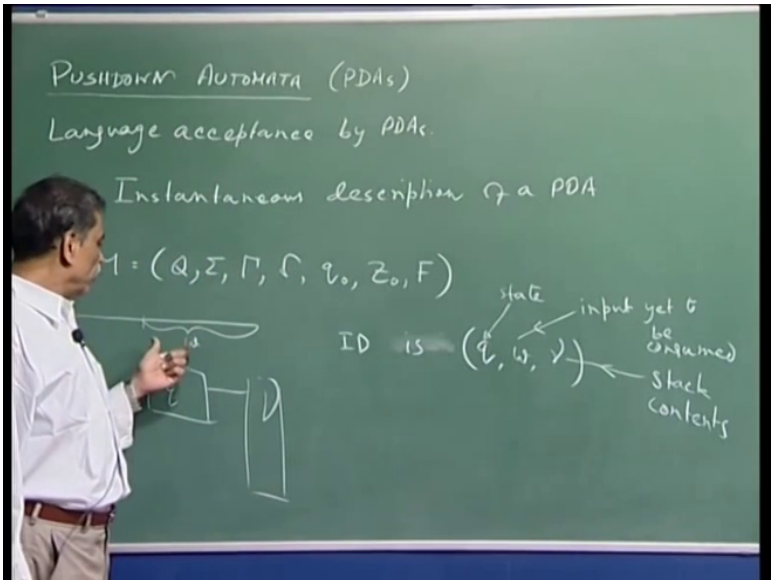
Course on Theory of Computation
By Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur
Lecture 32
Module 1
Pda configurations, acceptance notions for pdas. Transition diagrams for pdas.

Today we will start with the notions of language acceptance by PDA's, there are two ways in which PDA can accept languages but before that we need to define a concept which is called instantaneous description of a PDA.

(Refer Slide Time: 0:40)



The idea behind this is something like this that suppose a PDA is working and as you know we saw an example also last time that it works in discrete steps at every steps the machine is in some state scanning an input symbol top of the stack symbol and depending on that it changes the state, changes the top symbol of the stack by something.

Now imagine that as the PDA is working after a certain step we halt it, the kind of information that we need to restart the PDA once it is halted that is called a instantaneous description, okay. So I will define it properly, so let us say I have a PDA remember PDA is a seven tuple it has a set of states Q then the set of input symbol sigma, gamma the set of stack alphabet or the gamma is

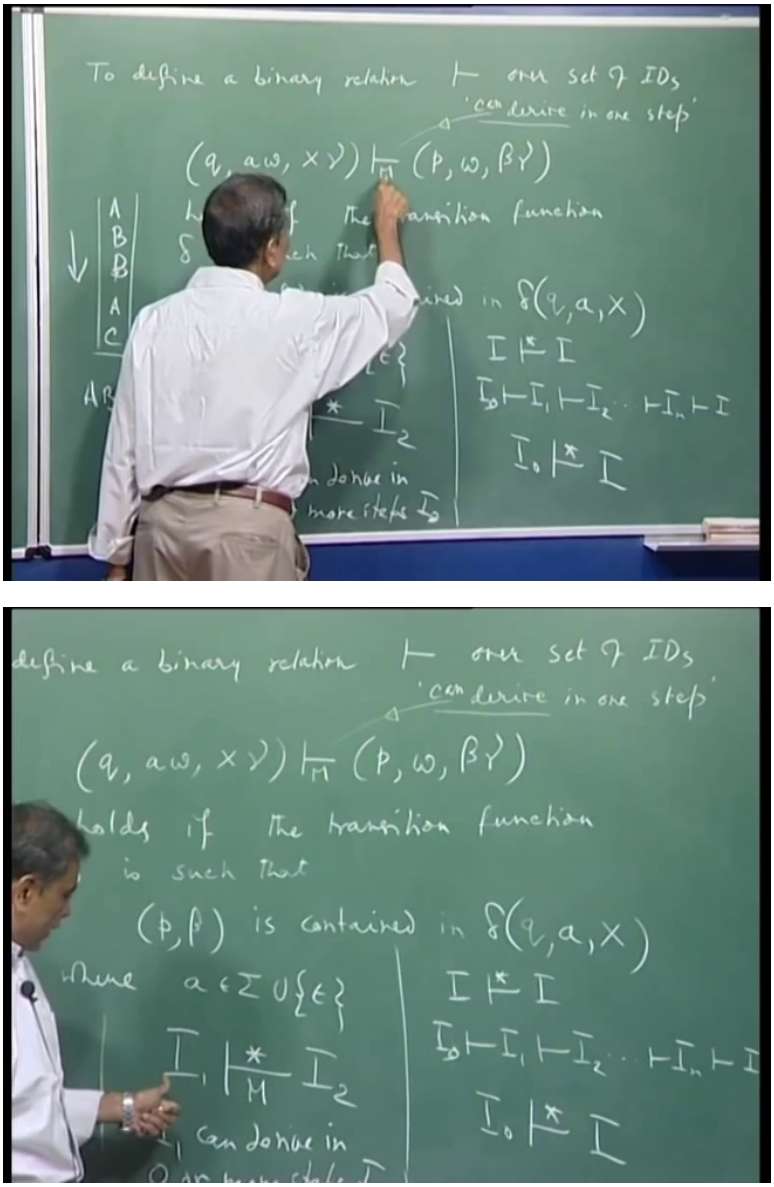
the stack alphabet, transition function, the initial state, right? The initial stack symbol and the set of final states.

So that defines some defines a PDA and imagine that at some point of time the of the input the portion which is left is w the machine is in some state q and the stack contains is something gamma the entire contents of the stack is a string of overs gamma and that I am denoting by small gamma. So if you give me this much of information it is clear that I can now from that step onwards simulate that PDA or PDA can work now again, so suppose that something happened then we stop the PDA then some input symbols left the machine is some state q and the contents of the stack, if these three pieces of information I have then we can see how the PDA will evolve in time in future.

So the description which captures this the overall state of the PDA at a certain instance is called the instantaneous description of PDA and as you can see that an ID instantaneous description a three tuple so this state w and gamma where q is the state in which the PDA is of course and w is the portion of the input string yet to be consumed input yet to be consumed and this gamma is the stack contents, clearly what we are saying is it did not matter really for what happens to the PDA in future that does not depend on the symbols it had seen already how did it come to this state or the evolution you know or history through which the current contents of the stack happened.

So once we have all this we have enough information to carry on. And in fact we can define something which is binary relation over the set of all ID's, so what will be the set of all ID's? All tuples of the kind in which the first component is a state, second component is a string over sigma, third component is a string over the stack alphabet, right?

(Refer Slide Time: 5:57)



So what we are trying to do now is to define a binary relation and it is denoted like this symbol is borrowed from logic and the it is called turnstile you know I do not know they have seen such gates where you know you pull this and the gate opens so anyway this is called turnstile this symbol.

So we are defining a binary relation that is using this symbol I mean that binary relation will be denoted by this symbol and over the set of ID's of a particular PDA you know there is some PDA we are talking of the set of all ID's of that PDA and we are defining a binary relation. Now this binary relation I should say what the you know which two IDs are related by this binary relation.

So we say that q, okay aw and then let us say X gamma that is related this particular this as you can see is an ID this ID is related to p, w, beta gamma holds this fact holds if the transition function delta is such that p, beta is contained in delta of q, a and X, where before that I should say that we call that delta of you know delta is a function transition function which maps a state and input symbol or epsilon a can be also epsilon and a stack symbol.

And that contains tuples like this and of course it means that of the machine is in state q and the symbol it wants to consume is or you know current either the input symbol is the where the input head is the symbol a or a is epsilon and X is the top of the stack symbol, then delta gives the possibilities of what will be the next step and what will be the string which will replace this top of the stack symbol. So you know this phrase makes sense that these tuple is contained in this because in general this is a set of such tuples, right?

So now maybe I should write this down where a is an element of sigma union epsilon, right? And other things are clear from the context that p and q are states beta is a string over the stack alphabet gamma and the a as we said can be an input symbol that is an element of sigma or it can be epsilon X is stack alphabet, sorry stack symbol and now you can see that this has to be a string over the stack alphabet and the this is beta is also string over stack alphabet.

In a notation looks unfamiliar in the beginning but what it is saying is very simple that suppose at some point of time the instantaneous description of the PDA is given by this 3 tuple which means as we know that the current state is q and the input yet to be consumed is aw and the contents of the stack is X gamma, I think I told you that the way we suppose we read the stack always this way top to bottom because normally when we write the stack contents it will be a string we write it from left to write.

So the left will left of the string left most symbol of that string corresponds to the top most symbol of the stack, right? So in particular suppose the stack was ABBAC left to right the contents will write like this ABBAC, okay. So therefore when I when we write like this it is quite clear that capital X is the top of the stack symbol.

From this ID now we can give a kind of name that we can read it as that this ID from this ID the machine can go to this ID. So you can say you can write it as derives or can derive in one step (())(12:49) this is how we may read it if you wish that this ID I can derive in one step this ID,

why are we saying that can derive instead of saying derives? Because we remember that in general our machine is a non-deterministic machine.

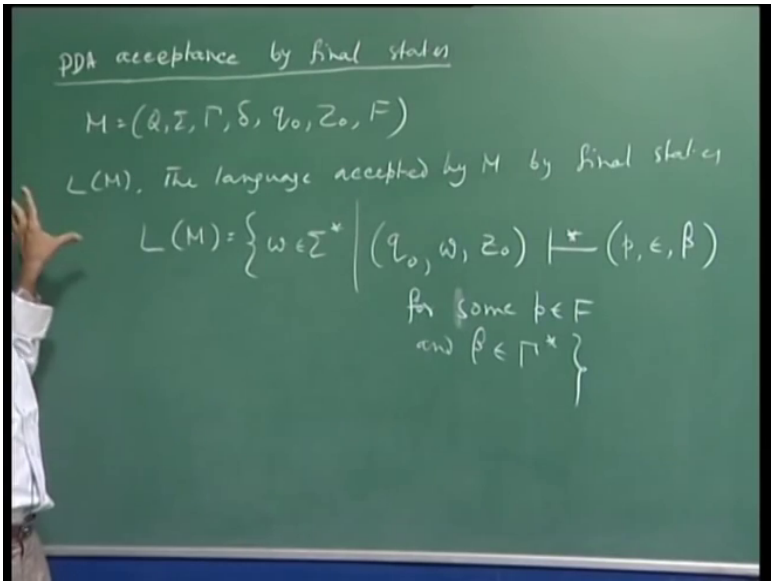
So from some point of time so that means when this particular ID holds in general there will be several IDs next ID which is possible. So in particular what we are saying that next ID can be this next instantaneous description can be this and that is the reason is very simple because we go from this ID to this ID following a legal move because delta q, a, X contains p beta, right? Now we will define the notions of acceptance before that again we will what we will do is let me denote by this symbol you know read this symbol as can derive in one step if we write I 1, I 2 and put this symbol in between what I am trying to say is that this is also a binary relation over IDs but this binary relation is derived from this particular binary relation which is can derive in one step and this really means that I 1 can derive in 0 or more steps the ID I 2, so let me write it down this actually is saying by definition I 1 can derive in 0 or more steps the ID I 2.

So in other words you see that those are few whole would like to put it that way that this particular relation is nothing but the reflexive transitive closure of that relation. So in other words for every ID it is the case, is it? Because in 0 step if you can go from this ID to itself that is all it means and suppose I had I going to I 1 to I 2 dot dot dot I n to. So this is I 0 and this is I so what we are saying is that the our machine is such that from the ID I 0 it can go to the ID I 1 from I 1 it can go to the ID I 2 and so on.

Then this situation we can write it as I 0 in 0 or more steps can go to or can derive the ID I, right? So this is the concept we will use to the define the notion of acceptance and I should also say that when there is possibility of confusion when there are more number of more than 1 PDA you are talking about simultaneously or considering simultaneously. So then under to say that which you know which set of which IDs you are talking of you mention the machine name for which these IDs are.

So if our recall that our machine we said this machine M and if we are talking about the basically that fact that M can derive in one step something in more technically one writes M below this and similarly this means the machine M can derive in 0 or more steps the ID I 2 starting from the ID I 1. Now we will define we said the two notions of acceptance by PDAs the first notion is the kind of acceptance notion we are used to.

(Refer Slide Time: 17:58)



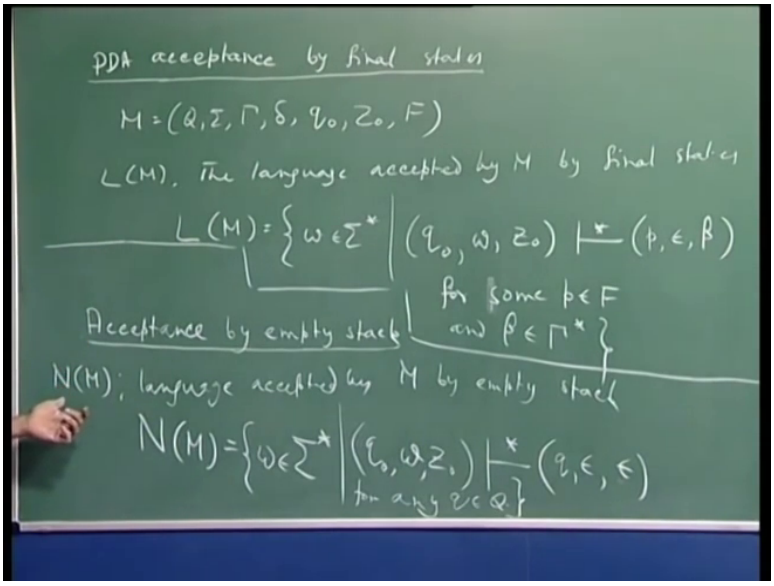
So PDA acceptance by final states so again let me write down a PDA or PDA M as Q, sigma, gamma, delta, q0, z0 and some set of final states F and now the language accepted by M by final states.

So L M the language accepted by M by final states is defined in this manner that L M is the set of all strings w of course they have to be input strings so the set of all strings w such that from the ID q0 remember q0 is the initial state q0 w and z0 in 0 or more steps we go to p epsilon and any beta, right? For some p in F and beta in gamma star, okay alright? So now what it is saying? It is saying that look the language accepted by this machine M by so called final states is the set of all strings w, right? Such that initially when you have on the input the string w and of course the initially what will you have the machine will have initial state is q0 and the only stack symbol only thing in the stack is the initial stack symbol namely z0.

So this is the initial ID initial instantaneous description if the machine can go from this initial description to an ID where this state is one of the final states, right? F is remember the set of final states of the machine M having consumed the input, so remember that second component is what is left in the input so now it says there is nothing left in the input to be consumed no symbols so that string it is right now the input entire input has been consumed so what is left is an empty string and any stack contents we do not care some beta, so long in other words so long

the machine can go from its initial ID to an ID where it is in the final state is in some final state having consumed the entire input such a string w will be the language accepted by M.

(Refer Slide Time: 22:19)



As I said there is another notion of acceptance and that is called acceptance by let me by empty stack, okay. So we say what the same machine M, N M is the language accepted by M by empty stack, okay where this N M is defined like this that this N M is again the set of all inputs strings, right? Such that again from the initial ID which is q0 w z0, right? This is of course the initial ID the machine can go to some q and having consumed the input entirely so here it is epsilon and the stack is empty, right?

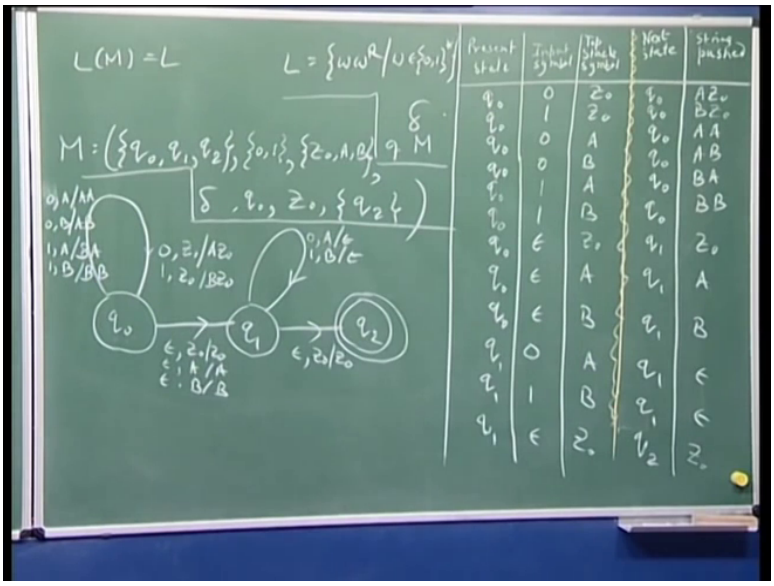
So that means see third component gives the contents of the stack, so stack is also empty, correct? So I should write that for any q in Q, in other words we do not care where the state finally is so long the input is consumed as and the stack is empty, right? So all those strings which can take the machine from the initial ID to an ID where the entire input is consumed and simultaneously the stack is empty such strings are in the language accepted by the machine M by empty stack.

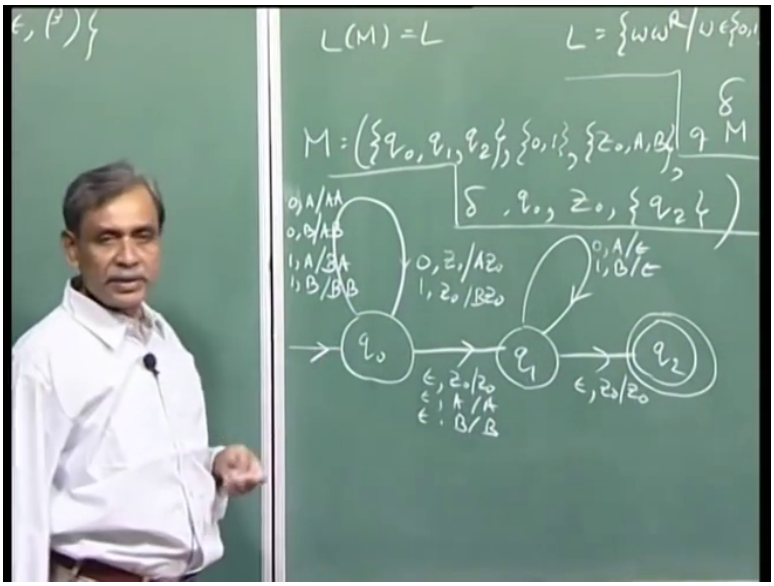
So these two notions you know this notions of course looks pretty usual you know finite state machine also we said strings which can take the machine from initial state to final state and this that this is a kind of you know the same way this particular definition has been you know we defined this particular notions of acceptance. So this notions seems new the reason is actually we

will find that this notion is more easy to use and we will also show that if for a particular language we have a PDA which accepts that language by final state then there is some other PDA which will accepted by empty stack and vice versa.

So in that sense both these notions are equivalent, so in other words if a PDA or let me put it this way PDA can accept a language by final state if an only if some PDA can accept that same language by empty stack. So these two are kind of they are equivalent notions and depending on the situation whichever is convenient we use one of the two notions.

(Refer Slide Time: 26:47)





Last time in the last lecture we had given an example of a PDA to accept the language if you recall the language was w w reverse and w was a string in 01 star this is the language is kind of even like (())(27:07) the language of all even length (())(27:11) and we described the PDA and the same PDA if you recall it had three states q0, q1 and q2 and we give the gave the transition diagram not diagram but the transition function and that machine I am re-describing here and in a tabular form I am this is the delta, right? Delta of M and what I would like to kind of argue or convince you that L M the language accepted by final state is this particular language.

So this is what I would like to convince you but we just recall this is essentially what it was doing is that initially during the w part it is in more or less in the state q0 entire time it is scanning the w part and depending on a 0 it would push an a depending on a 1 it would push a b on the stack without taking off the top symbol and then once the this part comes then it will check off the input symbol 0 with the top of the stack symbol a so that is that time it is in state q1 the it is in state the q1 and the input symbol is 0 top of the stack is A remains in q1 so it is basically removing the top stack symbol meaning that this 0 has been checked off.

And this is how we do and by the way how did we go from this part to that part how does the machine go from state q0 to q1 it essentially guess that now that w part is over and we should go over to the w R part we are now we are going to scan the w R part once that guess is reflected by you know it is q0 on epsilon whatever be the top of the stack symbol z0, A or B it moves to the state q1 and without disturbing the top of the stack symbol, right?

And you know of course one could describe the transition function in this manner but tabular format is somewhat not a very very easy to view there is a graphical view of representing PDA which I describe the way to represent such PDA is you know somewhat familiar way that we represented our finite states machines so it is also going to be a graph with nodes and each node is a state, right? And each arrows will represent a transition. So this is 3 states but maybe first of all I should clearly write what the machine is that it has 3 states as we can see there q0, q1, q2 what is the input alphabet is of course 0, 1, what is the stack alphabet? Stack alphabet is z0, A, B, okay.

And what is the initial state is q0, what is the initial stack symbol? z0 and of course we had the transition function delta which I had put down as a table and the set of final states here the set of

final states that consists only of 1 state and that is q2. So this is our machine and delta is elaborated in this table, so all that information we can capture nicely in a diagram and the it is called that this kinds of diagram will be called the transition diagram of PDA's and the so as we said that for each state we will have node in the graph so let me say this is q0, this is q1, this is q2.

So let see for example take this from q0 take the first one very first one q0 on input 0 top of the stack symbol is z0 it can go to q0 and writing A and then of course retaining the top of the stack symbol this situation you know from q0 it is going back to q0 so it is as you can imagine it is a self-loop and here what way what this arrow has you know I must mention the what is the input symbol so on what input is this transition possible we said 0 and now the other two information is the what is the top of the stack symbol z0 and that is going to be replaced by the string Az0, right?

You can see this, so let us take the next one that is again such a loop and you know I will just write one more possibility here that if it input is one and top of the stack symbol is z0 then it writes basically corresponds to this maybe it is getting crowded so I will write it here. So is the notation clear now that we go from state q0 back to q0 on input 0 when the top of the stack symbol is z0 and we replaced the top of the stack z0 by this string, in other words if I have in more generally so let me write this supposing this is p and q and I had a and comma, right? X stroke beta what does it mean? That q beta is in delta of p, a, X.

So suppose this is the situation that on state from state p on input a input symbol a with this top of the stack symbol X the machine can go to state q replacing the top of the stack X by the stack string beta such this thing we are writing like this you know it is clear that we are going from state p to q the symbol on input symbol a clearly it can be epsilon also you know if this was epsilon this particular thing will be epsilon and you know we were saying what is the sting that replaces the top of the stack symbol. So in these cases if you see we are not removing anything we are just pushing another symbol, so another way of saying that is that z0 is replaced Az0, right? And so on.

So maybe let me write one more strings so q0 0 A so q0 from q0 on 0 and if the top of the stack is A then we replace it by AA, right? Then third one is 0, B we replace it by AB and maybe I can

complete it q0 1 A we go back to q0 writing B and then of course we are just pushing B on this 1, B will be BB, right? And so on. And finally q0 from q0 you see on epsilon whatever be the stack symbol we can go to q1 and we do not change the stack that same symbol remains, so is in this we can write like this that it is saying that on epsilon we can always go from q0 to q1 if the top of the stack symbol is z0 we do not change it, similarly epsilon A, A epsilon B, B, okay.

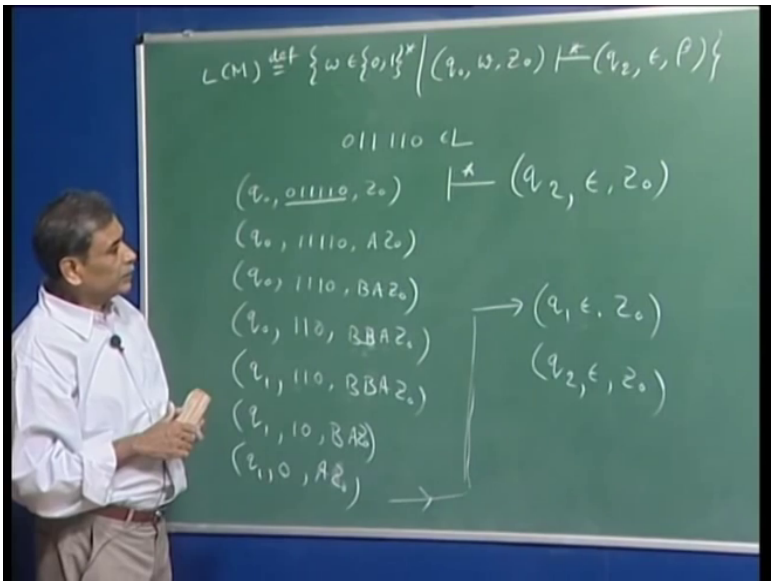
And when we go you can see that there is only one so this part I am not writing you see that from q1 this is actually easy, right? So maybe I can write it there only two such things going from q1 to q1 so on 0 if the top is basically epsilon here that means this A is being popped 1, B is epsilon and here we will go taking this transition and that is on epsilon if it is z0 you can write it z0, right?

And the fact that q2 is the only accepting state we you now denoted the way we marked the final states or accepting states of a finite states machine by putting this concentric circles two concentric circles same way any state in which you know this kind of two circles are there then such a state is the final state or an accepting state, this is not part of the diagram, right? This was I was explaining what in general a transition right? If this is the transition this is the corresponding diagram, right?

And I would now like to explain that this machine accepts by final state the language w, w, R. So we remember by definition for that machine M PDA M the language accepted by final states is the set of all binary string such that you know each of those strings you know that is this ID from this ID the machine can go to that ID, what is the initial ID? Of course the initial state is q0 this is the initial state by the way we marked the initial state the same way as we did in case of finite state machines the initial state of a PDA will be marked by an arrow coming from nowhere into the state.

So that is q0, w, z0 initially only the stack contains z0 and all those strings which can take the machine from the initial state to the final state having consumed the input entirely. And it is not too difficult to see that this machine is like that.

(Refer Slide Time: 41:06)



So for example let us take an first of all what happens when a strings is in the language so let us say 011 and the correspondingly we will have. So this particular string is in the language L, right? We defined this language L to be the set of all strings which are of the form w, w, R and clearly that this particular string is in the language what happens so if I see you know so let me write the initial ID q0, 011 110, then I will have z0. Now which of so there are several things which are possible actually if you see on state q0 when the our first input symbol is 0, so therefore this particular transition is applicable in the diagram 0 I have z0 here and this therefore I should can do this but also see this another transition is possible.

So non-deterministically we need to choose the machine need to choose for that string to accept which one of this it should use you can see that in the first phase in the when it is in the q0 the idea is that it should push on the stack a string which is equivalent to or you know really a an encoding or another rendering of the w part, right? So instead of taking this particular transition let us say that machine non-deterministically chooses to remain in q0 consuming this 111 I am consuming this 0 so what is left of the string is this particular string and it pushes the symbol A on the stack so I will have A z0, right?

Then again instead of going from q0 to I mean q1 because it kind of guesses that as yet the w part is not over so from here it will go to q0 it will consume this symbol so 1110 then for this corresponding to this one it will write BA z0, right? Do you see which transition it is taking? It is

taking that I have 1 and the top of the things stack symbol is A and therefore this A is replaced by the string BA that is what we take this A is replaced by this BA, right?

And again the third again in the same manner it will remain in q0 this is the particular computation path and the so it is it consumes this 1 so the what is left in the input is 110 and corresponding to this 1 symbol B is pushed so BAA z0, right? And now non-deterministically the machine guesses that it should move to state q1 I should I mean once more remind you this non-deterministic machine working is not really not anything mysterious all it means simultaneously all these paths are possible and we are just talking of the accepting part.

So really speaking the remember that we said that non-determinism means that whenever there is a choice machine makes two copies of itself and pursues both the possibilities simultaneously. We are just looking at that possibility which will take the machine to the accepting situation. So now in that path the machine will go from q0 to q1 without consuming any symbol from the input and without disturbing anything which is on top of the stack, right?

So this is the one that we are this transition is what we are using, So here the top of the stack symbol is B that is fine, so it will remain as B in fact nothing else will get change so q1, 110 then BAA z0 and now in q1u will keep using the (())(46:39) which are there on 0 if the top of the stack symbol is A you will remove that and consume this 0 on 1 if the top of the stack symbol is B you will consume that consume the input and pop off this symbol and that is exactly what will happen.

So one top of the stack is, we had made I think a mistake because so let us see there are 011 so when the so this A and I you know for this another B should be put and I wrote here an A this A is corresponds to this A, okay so let us see when that was a mistake but you can see what is happening q0, 1110 so this symbol is 1 top of the stack symbol is B under this situation if it wishes to remain in q0 what it should it do. So 1 top of the stack symbol is B, right? 1, B so it would replace this B by BB that is what it is doing here, right? So this will be also B I am sorry but now you can see things will work out.

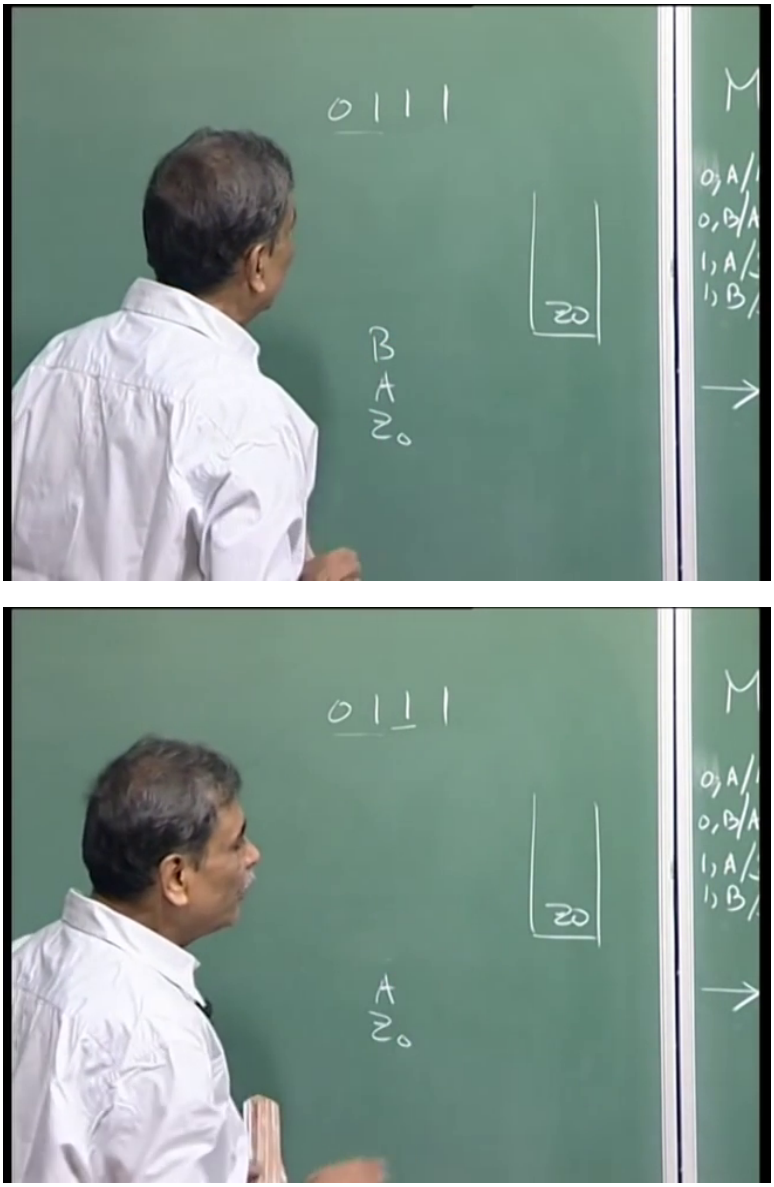
So q1 it sees 1 top of the stack symbol is B so if you go to that you will see that it is a machine will can do this that consume this so in the input all will have is 10 this is popped off BA, right? And q1 this 1 is consumed so only thing left on the input will be just a symbol this and this will

be A and of course there is z0 here always, right? From here it can take up to this ID q1 0 A so this 0 you can check off with the top of the stack symbol A so let me write it here q1, epsilon this is over so z0, and in q1 you can always go to q2 without consuming any symbol so it is q2 was nothing left in the input so it remains so and this is z0.

So what has happened? So you see we can say that from this initial ID the machine on in so many steps can go to q2 epsilon, right? And z0, so therefore clearly this particular string 011110 is in the language accepted by the machine by final state L because this state is the final state and so on. And therefore it is not difficult to see that any string in this language will be accepted by that machine but that is not enough you know we can convince ourselves of that that any string of L will indeed be accepted by the machine M by final state but we need also to show that a string which is not in the language cannot be accepted by the machine, right?

And that part is a little more involved possibly to argue formally but you see what it means is that if I take a string which is not of the form w, w, R, then what can this machine do? You know let us just informally argue what can this machine do on a string which is not of the form w, w, R. So let us take a simple string which is not of the form w, w, R and let us see what we can do, okay. Let me argue what happens on a string which is not in the language, right? So on such a string the or we should be able to argue the machine can never go from q0 to q2 totally consuming the input epsilon, right?

(Refer Slide Time: 51:50)



See for example take this string 0111, right? So this is not of the form w, w R because when we chop it in the middle this is 01 and that is not same as that. Now you see what can this machine do on this machine on this string, it can go to the state q2, can it? You see look at this initially gives first of all can on this string the machine go from q0 to q2? Yes actually it can, you see what it can do is initially the stack is z0 remember initially the only thing in the stack is this think and the machine is in q0.

So on q0 on epsilon, right? It is without consuming anything it you know goes to q1 and this is there and in q1 again on z0 it goes q2. So it can indeed so write in the without doing anything

without consuming any symbol the machine can move from q0 to q2 but is the string accepted, no because if you recall it has to consume totally the input string and also simultaneously go from the initial state q0 to q2, so just because it can go from q0 to q2 is not sufficient reason for the string to be accepted the string must be consumed and now you can see that so that does not work so therefore what it can do of course is the many other things it can do remember it is an non-deterministic machine it can keep pushing the symbols corresponding to 0 and 1 so you know z0 then A then B, right?

And suppose it at this point it was in q0 so choose to move to q1 then again this symbol can be you know it can consume this symbol having removed the top symbol B and now the input is 1 and the machine is in A, so there is in q1 if you have the input symbol 1 and the top of the stack is A it is not defined we do not have a you know machine cannot do anything in fact no transition is defined. So you can see that this will not lead to q2 and this way you can argue wherever it may go from it has to start from q0 you know it has to sometime move to q1 and then finally to q2 and the only way it can do so move from q0 to q2 having totally consumed the input and the you know take this transition and go to q2 is only for those strings which are in the language.

And the next lecture we are going to talk about the acceptance by empty stack with some example and the equivalence of these two notions.