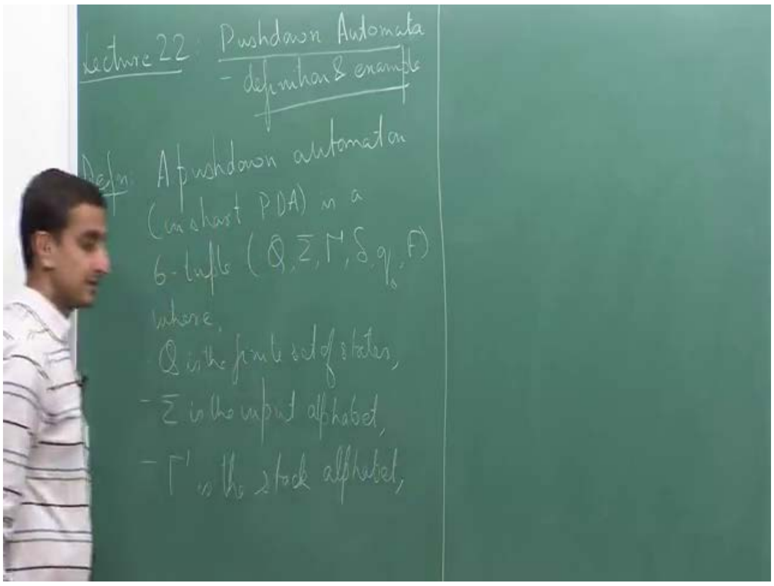
Theory of Computation

Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 22
Pushdown Automata - Definition and Example

Welcome to the 22nd lecture of this course. So, today we are going to carry on our discussion about Pushdown Automata.

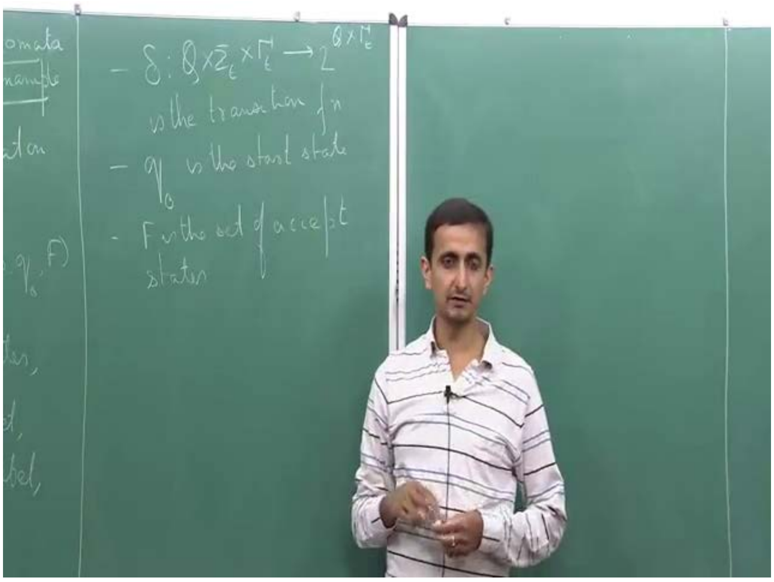
(Refer Slide Time: 00:46)



We will first see the formal definition and what does it mean for pushdown automata to compute a string, and then we will look at the exact computation of pushdown automata using an example. So, first let me define what a pushdown automaton is.

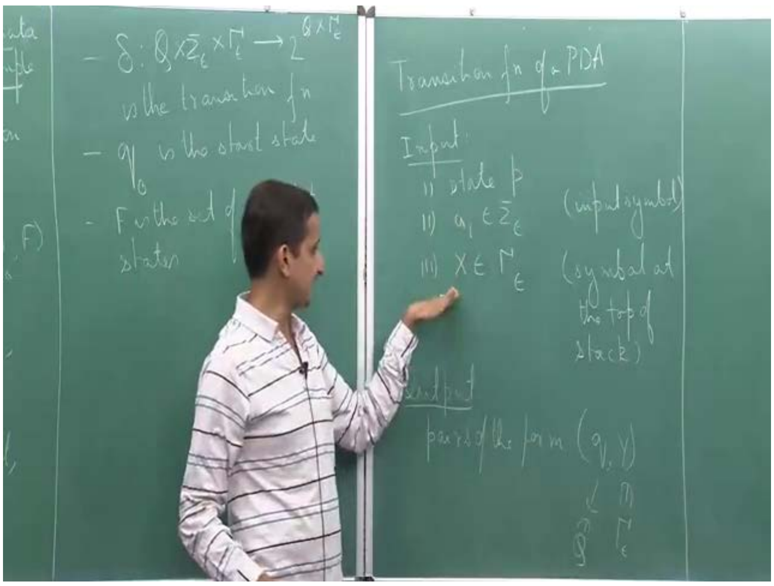
So, a pushdown automaton in short PDA is a 6 tuple Q, sigma, gamma, delta, q naught, F, where essentially the symbols represent the usual things except for gamma, which is the stack alphabet. So, Q is the finite set of states; sigma is the input alphabet; gamma is the stack alphabet.

(Refer Slide Time: 02:29)



Delta, which is a map from Q cross sigma epsilon cross gamma epsilon to 2 to the power Q cross gamma epsilon is the transition function; q naught is the start state, and F is the set of accept states. So, let me talk a little bit about the transition function because that is the thing which is different from that of epsilon NFA.

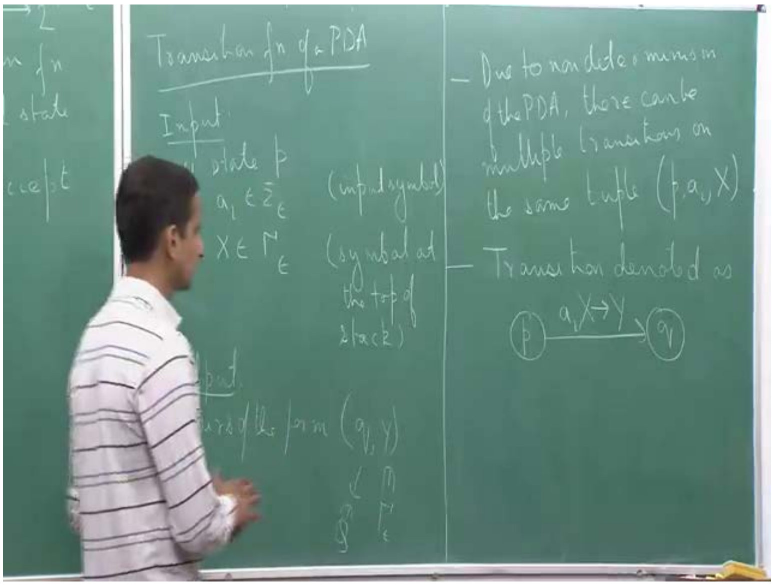
(Refer Slide Time: 04:00)



So, the transition function takes the following as input. So, if you look at it, firstly, it takes a state as input, it takes a symbol a i in sigma epsilon as an input. So, basically this is the input symbol, and it is reading a symbol x in gamma epsilon. So, this is input symbol, and this is symbol at the top of the stack. So, it is taking these things as input.

And it what it is producing is, it is producing pairs of the form some q comma y where q is the new state that you go to, and y is the symbol with which you replace x. So, this guy is in belongs to capital Q, and this guy belongs to gamma epsilon. So, what do I mean by pairs, what do I mean when I say pairs. So, it is not that it is just it can output just one state comma stack symbol pair given a triplet of the form. So, let say here the state is p. So, given let us say p, a i and X, it can actually output multiple q comma y.

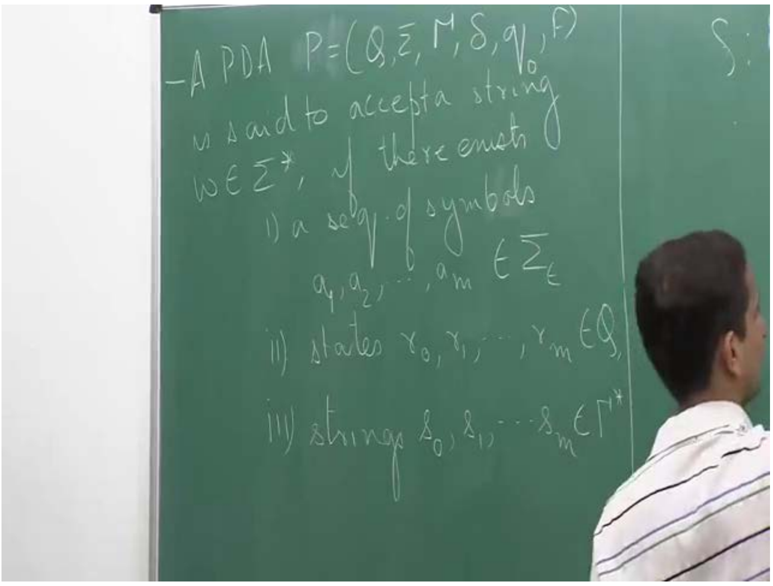
(Refer Slide Time: 06:42)



So, this is because of the non-determinism. So, due to non-determinism of the pushdown automata, there can be multiple transitions on the same tuple let say p, a i and X. So, this is why we say that it is an epsilon NFA - nondeterministic finite automata, because as I said that there can be multiple such pairs.

So, this transition is denoted as from state p you are going to some state q on reading the bit a i, and the stack symbol X gets replaced with Y. So, this is how a transition is denoted. So, in the case of finite automata, we had just from p to q on a symbol a i, but now we have this additional thing to represent how the stack changes - the behavior of the stack.

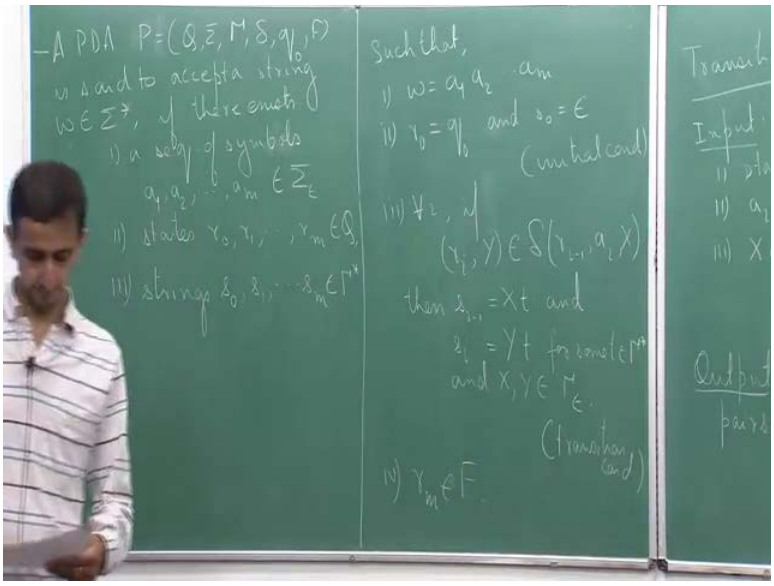
(Refer Slide Time: 08:53)



So, now let me define the computation of pushdown automata. So a PDA, so let us call it P equals Q, sigma, gamma, delta, q naught and F is said to accept a string w in sigma star. If there exists, so there exists three things. Firstly, there exists a sequence of symbols a 1, a 2 up to a m, each belonging to sigma epsilon. So, there exist m symbols belonging to sigma epsilon. There exists states r 0, r 1 through r m belonging to Q, and strings s 0, s 1 up to s m belonging to gamma star.

So, essentially there are there exist, so we assume three things a 1 through a m which are symbols in sigma epsilon; there exists states are 0 to r m; and there exists strings s 0 through s m over gamma star. So, these are strings over the tape alphabet. So, essentially intuitively what these m states mean is that the PDA on when it receives the string w, it basically accepts w in m steps, where it at each step it is reading a i, so a i can be empty also. At each step, it is going from a state ri minus 0 to r i, and at each step the contents of the tape the contents of the stack is basically s i. So, whatever string that is contained in the stack from top to bottom is s i.

(Refer Slide Time: 12:38)



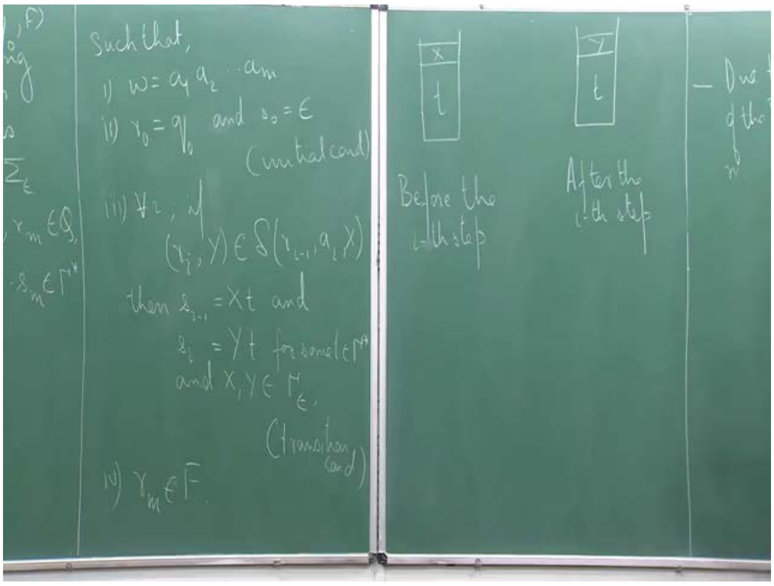
Let me just rephrase what I just said. Such that firstly, w is the string the concatenation of the symbols a 1, a 2 up to a m. So, whatever input w that is given to us, it is nothing but the concatenation of the symbols that I rewrite each step. So, again note that some of these can be empty. So, it is not that the length of w is m; the length of w is something that is less than or equal to m. Second is the initial condition that is so what happens at the beginning.

At the beginning, we have r 0 equals q 0 and s 0 is empty. So, at the beginning, the stack contains nothing. So, this is the initial condition. Third is the transition condition. So, we write the transition condition as for all i what it does is if delta… If r i comma y is contained in delta r i minus 1 comma a i comma x then s i minus 1 is x t and s i is y t for some t in gamma star, and x comma y in gamma epsilon. So, this is the transition condition.

Essentially what this is saying is that look at some i between 1 and m. So, this is i in the set 1 through m. Now suppose if I look at delta r i minus 1 a i and x, for some x, x can be any symbol in gamma epsilon. And if that set contains the pair r i comma y, then the stack in i minus 1 eth step should have had x at the top followed by some string y.

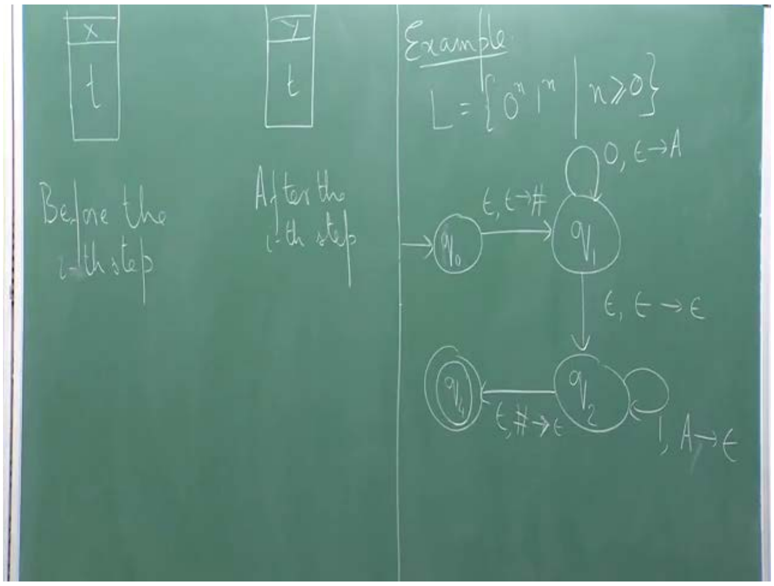
And in ith step it got replaced that x got replaced with y, and the remainder of the stack remains the same. So, it was t earlier, it remains t. And this essentially means that the competition moves from the state r I minus 1 to the state r i. So, this is what transition means. And when do we accept w, so we accept w if finally, the last state is the accept state, so if r m is, it belongs to F.

(Refer Slide Time: 17:01)



So, again if I want to give a pictorial representation, so then if I look at the stack before the ith step, it contains t with x on the top. And after the ith step, it moves to t followed by y at the top. Now, enough of definitions let us look at example. So, what we will show is something that will prove that the power of pushdown automata is more than the power of finite automata. In other words, pushdown automata can accept non-regular languages.

(Refer Slide Time: 18:13)



So, example, so we will construct pushdown automata for the popular language 0 to the power n 1 to power n. So, this is a language which has helped us in many places. So, first it helped us to show that to construct the first example of a non-regular language to show that a language is non regular. And we showed that context free grammars are able to accept this language. And today what we will show is that even pushdown automata can accept this language. So, what is the idea? I mean the how does the pushdown automata work I mean how does it go about accepting this.

So, what we will do is whenever we will have pushdown automata, where whenever it reads 0s the pushdown automata will be at a certain state let say at some state p, and it will push some symbol on to the stack, let say it will push the symbol A - capital A onto the stack. So, for each 0 that is being read, it will keep on pushing 0s.

And then the moment it sees the first one, it will move from the state p to some other state let say q, and it will start popping out a(s) from the stack. So, it will keep on doing this and finally, if the when the input is completely read, if the stack becomes empty whatever it was at the very beginning, then we say that the pushdown automata has correctly I mean the input that was given is of the form 0 to the power n 1 to the power n, because the number of A(s) pushed matches the number of A(s) popped when the string is completely read; otherwise we will say that it is not.

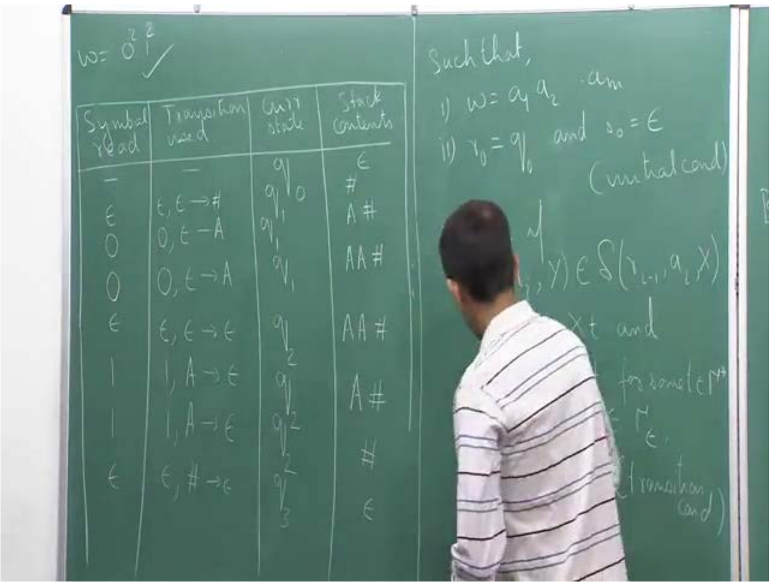
So, we will look at some examples. So, here we will use, we will construct the pushdown automata. So, we have a state let us call it this is our start state let us call it q 0. From q 0, I go to a state q 1 on reading epsilon and I push a symbol hash onto the stack.

Let us try to understand this. So, this transition means that I do not read any input bit, I just see epsilon, but what I do is that from by reading nothing from the stack, I push hash onto it, some symbol basically this symbol will be used to check whether the end of stack has reached or not. So, finally, we need to check whether we have reached the end of stack. We will use q 1 to push A(s) to read 0 and push a(s). So, on q 1 what we will do is that if I receive a 0 I will push an A on to the stack. So, this means pushing an A.

Now from q 1, I go to a state q 2, or q 2 I will use to pop out A(s). So, on q 2, if I see a 1, I will pop out A. And then finally, what I do is that again without reading anything, I will check if the last symbol is hash, if it is a hash, I just let say pop it out, it does not matter; and I reach a state q 4 which is also my accept state. So, now the only thing needed is to label this transition. So, when will I go from q 1 to q 2? So, I just do that non-deterministically.

So, whenever I am at a state q 1, I will non-deterministically do one of the two things, either I will check whether the next bit is a 0 or not, or without reading any symbol and changing anything on the stack the stack remains as it is epsilon to epsilon I just go to this. So, this essentially means that non-deterministically I am trying to see if the next symbol is a 0, or if the next symbol is going to be a 1 that is all. So, this is a PDA for the language L.

(Refer Slide Time: 23:36)



Now, let us try to look at some example to see how the PDA behaves. So, let us take w equal to a string 0 square, 1 square. So, on this string, what we will do is we will construct the following table, where I have symbol read. Just write it clearly. Then transition used current state, and stack contents, so these are the things that we will try to see. So, initially, at the beginning, we do not have anything. So, the automaton is at the state q 0, and the stack has nothing on it, so it is empty. So, this is the initial condition.

Now which symbol will it read? So, first if we look at the automaton, first it has to go from q 0 to q 1 it reads nothing. So, it reads the symbol epsilon. It uses the transition epsilon comma epsilon to hash, it pushes in hash on to the stack and it goes to state q 1. So, now the stack has hash on top.

In the next step, it is going to read the first 0. So, it will read 0, it will use the transition 0 epsilon to A, it stays at the state q 1. And now it has pushed A on to it. So, now, the stack has if I just want to write it as a string, it has A hash A on top and hash below it. It reads the next 0 sees uses the transition again epsilon going to A, uses stays at the state q 1, and now the stack has A A hash. Now the 0s are exhausted.

Now, what we will do is that we will use the transition epsilon comma epsilon going to epsilon. So, we will not read any input bit. So, the symbol read will be epsilon. It moves to state q 2, the stack remains unchanged.

In the next step, it sees 1; it uses the transition 1, A to epsilon which means that it is popping of A and it stays at state q 2. So, the current condition of the stack is A hash. Then it again reads a 1, it uses the transition A going from A to epsilon, and it stays at the state q 2. Now the stack has hash on it. Now the ones are read. Now it can again without reading an input symbol, it can use the transition epsilon hash going to epsilon, move to the state q three and basically accept. So, now if I concatenate all these, I get w which is 0 star 1 star which means that this guy gets accepted.

(Refer Slide Time: 28:16)

So, now, let us look at a similar table for a string which is not accepted a string w, which is 0 square 1. Suppose, if I try to accept 0 square 1 what happens. So, initially again I have q naught and epsilon. On reading epsilon, I used the transition epsilon going from epsilon to hash and push move to state q 1, and push a hash onto the stack. Then on reading 0, I used the transition 0 epsilon 2 A q 1 a hash. Again I read 0, again I use this transition and go to A A hash. Now I use the transition epsilon-to-epsilon move to state q 2 stack remains unchanged.

Next I read 1, so it is 1 comma A gets popped out I remain at state q 2 and the stack contains A hash. And at this step, my input is exhausted; I do not have any more bits to read. So, I cannot use this transition to go to q 4 from, so this should be q 3, I cannot use that because now my top of the stack contains A. So, I cannot use that transition. So, I am struck at the state q 2 and I cannot accept. So, because of this, this string does not get accepted and it is nondeterministic. So, this is just one computation part that I showed; you can also check that there may be other computation parts you can try out other computation parts, and you will be able to see that on none of the computation paths will you actually reach the accept state.

Another example that you can try out is for the string 0 1 square for 0 1 square what you can see is that although you are able to reach the state q 3, but when you reach the state q 3, you would not be able to exhaust the entire input. For example, if I have the string. So, if I have the string 0 1 square then if I read the first 0 and 1, I will be able to go to state q 3, because 0 1, but then the input is not completely exhausted which means that w will not get accepted. So, I will stop here today. And in the next lecture, we look at some more examples of languages, which are accepted by pushdown automata.

Thank you.