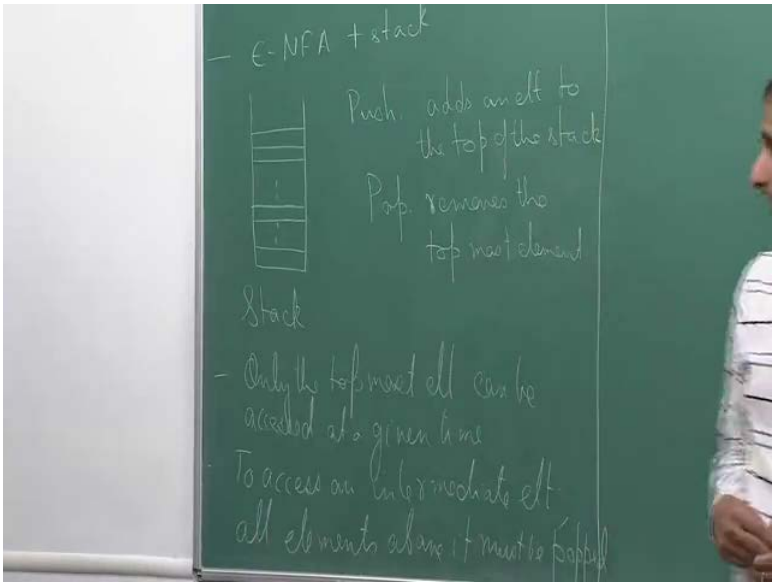
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 21
Pushdown Automata

Welcome everybody. This is the 21st lecture of this course. So, today we are going to talk about this new type of automaton known as Pushdown Automata.

(Refer Slide Time: 00:41)



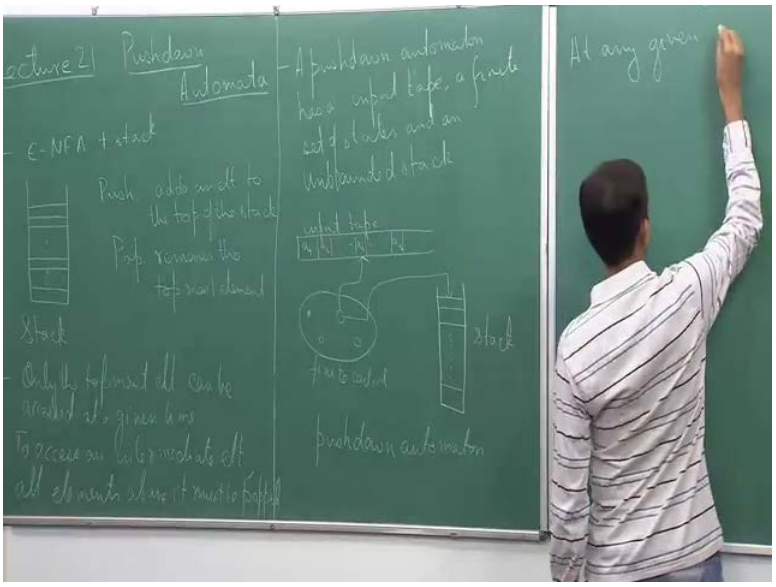
So, what is the pushdown automaton? So basically a pushdown automaton is an epsilon NFA appended with the stack. So to put it in the simplest words, it is epsilon NFA plus a stack. So what exactly does it mean? How do we define a pushdown automaton in the first place? Does it actually add any power over epsilon NFAs?

We have seen earlier in the context of regular languages that epsilon NFAs are exactly those classes of machines or those machines which accept regular languages, you can convert any epsilon NFA to a DFA, but if I add a stack to it, does it actually add more power. So we will answer these questions in the next few lectures.

But today, what we want to understand is what exactly is this model and give some examples of languages that it can accept. So, once again before I go into the model, I want to give a brief recap of what a stack is. So, stack is a data structure that most of you must have seen in your data structure course, nevertheless let me give a brief recap. So a stack is a data structure which has it is a linear data structure, it has a linear form. And you can only access elements of a stack from one end, so usually we think of that end as the top end. So, if I want to add an element, I can only add it to the top of the stack. And if I want to access an element I can only access the top most element; if I want to remove it, I can only remove the top most element.

In other words if I want to let say access an intermediate element let say some element that is contained in a cell much below, in order to access it, what I have to do is I have to pop out all the elements from the top and to be able to access it. So, a stack has two operations defined on it. So it has the push operation, so push basically adds an element to the top of the stack; and pop removes the top most elements. So these are the only two action these are the only two action that one can define on a stack and use to access the data structure. So, this is a stack. So few points as I said that so only the can be accessed at a given time. And to access an intermediate element, all elements above it must be popped; in other words, removed. So this is what a stack is.

(Refer Slide Time: 05:36)



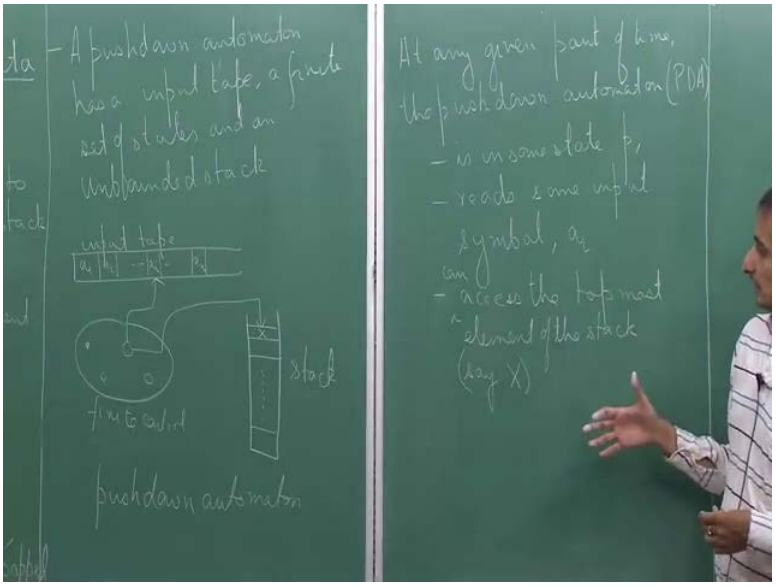
Now let us see what do we mean when we say an epsilon NFA appended with a stack. So, informally so I am just going to give the informal description today. So, a pushdown automaton has an input tape, a finite set of states and an unbounded stack. So, just a brief mention of this what unbounded. So, when we talk of a stack, we do not put any limit to the number of elements that it can have.

In other words, the stack can store arbitrary amount of information, so that gives that supposedly it should give more power to it than an epsilon NFA. So, pictorially it would look like this, so we have what is called the finite control. So, this is our finite control. So, this has the states So it has may be some state over here, some state here and so on - infinite number of state.

There is an input tape which contains the input bits. Let us call them a 1, a 2 up to some a n. And at any given point of the computation, the automaton is in some state and let say it is reading some input bit a i. So, this is essentially what an NFA or a DFA does. So, it has it is input; it is it reads the input one bit at a time from left to write and it is at some state. So, whenever it reads the next bit, it accordingly moves to a new state or it stays at the same state and so on, so that is how a finite automaton behaves.

Now, in addition to this structure, we have so this is, let me just write it; this is the input tape. So in addition to this structure, we have unbounded stack, and the automaton can also access only the top most element of that stack. But it can do whatever it wants to do with it; I mean, it can add more elements to it or it can remove elements or do whatever, but there is no bound on this stack. So this is what a pushdown automaton basically looks like.

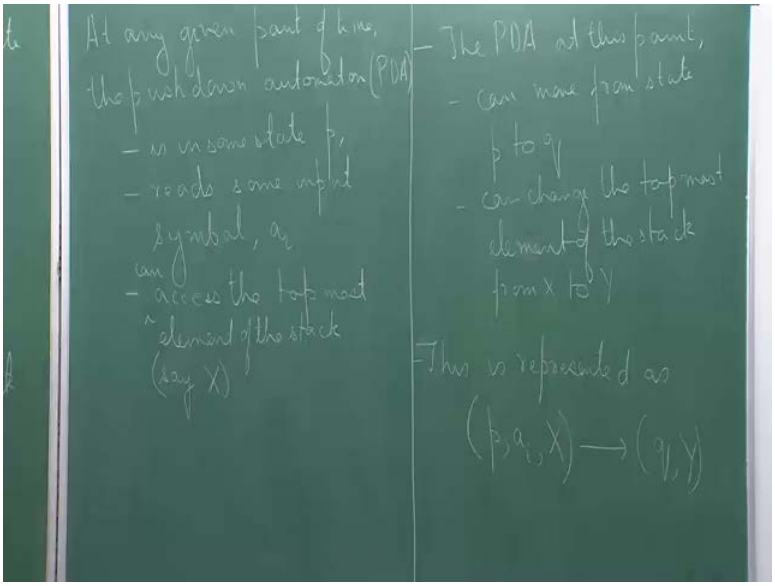
(Refer Slide Time: 09:34)



Now let us look at how the pushdown automaton. So, how can I define an instantaneous description of the pushdown automaton? So, at any given point of time, the pushdown automaton, so in short this is also refers to as a PDA, for pushdown automaton. So, it has is in some state p, reads some input symbol let say a i and can access the topmost element of the stack say capital X. So, let say this is capital X. So this is so these are the things that the pushdown automaton has accessed to at any given point of time, let say it is at some state p. So, this is the state p. It can read input bit a i and it can access the topmost element x.

Now, from this position, in the next step, it can go to, it can transmit the state, so it can go from p to some state q it goes from p to q, and it can change the topmost element from x to some y, so that is how we define a pushdown automaton. So, it can change the top element x to y.

(Refer Slide Time: 12:47)

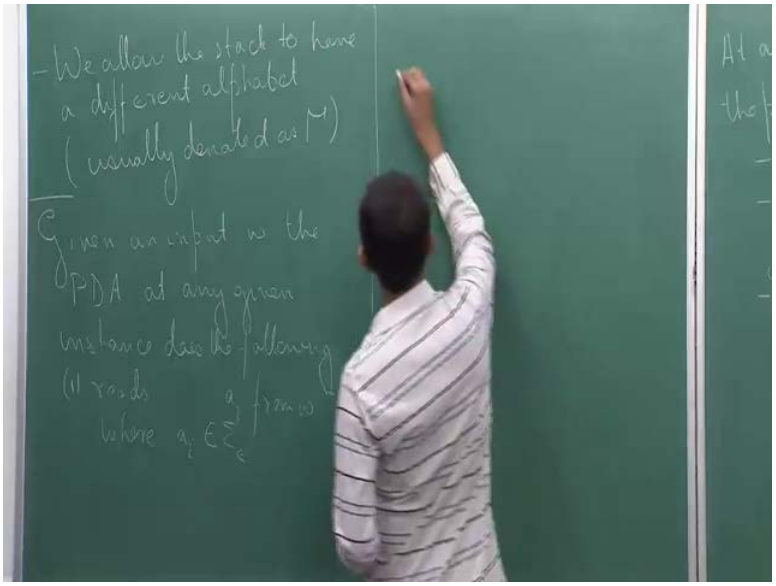


The PDA makes and this point can move from state p to q, and can change the topmost element of the stack from x to y. So, this is represented as so using the notation from p on reading the input symbol let say a i, and reading the stack symbol x, the automaton goes to q - state q and replaces x with y. So, this is essentially what it means. From state p it goes to state q, on reading a i, and it replaces x with y. So, this is what forms the basis of one step of the computation of a pushdown automaton.

One more important point that I should mention at this stage is that when we talk about finite automata, there was only one alphabet. So, the alphabet from which we used the symbols of our input, it was called the input alphabet. But now since we are dealing with the stack, we will allow the stack to actually have its own set of symbols; it is not necessary that the stack will have the same symbols as the input.

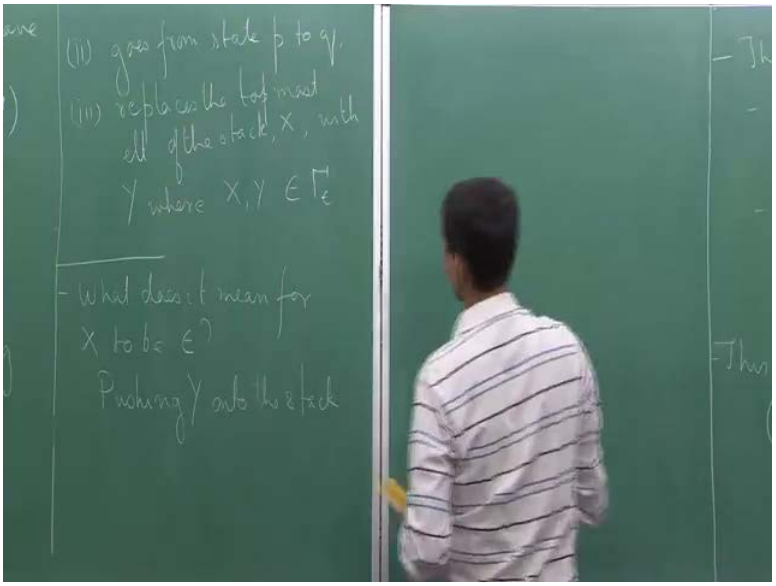
Of course, it can have, it can have the same set of symbols as the input symbols, but we will allow it to have its own alphabet so which it is called the stack alphabet, and it is usually represented with capital gamma. And the symbols in the stack alphabet, we will usually denote them with capital X or capital Y or capital Z or something like that, so that is the usual convention which we will follow for our lectures.

(Refer Slide Time: 16:34)



So, we allow the stack to have a different alphabet say usually denoted as gamma. So, formally so given an input w, the PDA at any given instance does the following. It reads a bit a i from w. And now because it is an epsilon NFA, a i can be the empty string, where a i belongs to sigma epsilon. So, recall this notation that we had used earlier. So, sigma basically denotes the input alphabet and sigma epsilon denotes the input alphabet together with the string epsilon. So, each bit a i, I should not say a i, so reads well I will just say reads a i from w and this a i can either be a sigma or it can be epsilon.

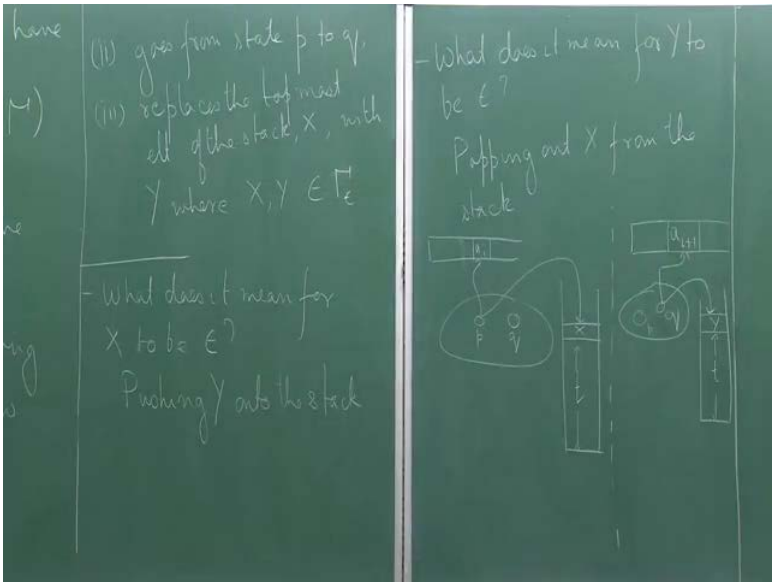
(Refer Slide Time: 20:17)



Goes from state let say p to q, and replaces the topmost element of the stack comma x with y, where x comma y, once again are symbols in gamma epsilon. So, actually we allow x and y to also take the value epsilon. So, what does it mean, so what does it mean for x to be epsilon. So, when we have x epsilon, and y let say some other some other string it may be some symbol in gamma, what does that mean.

If we look at the description, so the description says that it replaces x with y, x is the empty string. So, it is replacing the empty string with some symbol y essentially what that means is that it is pushing y onto the stack, so that is what it literally means.

(Refer Slide Time: 22:28)



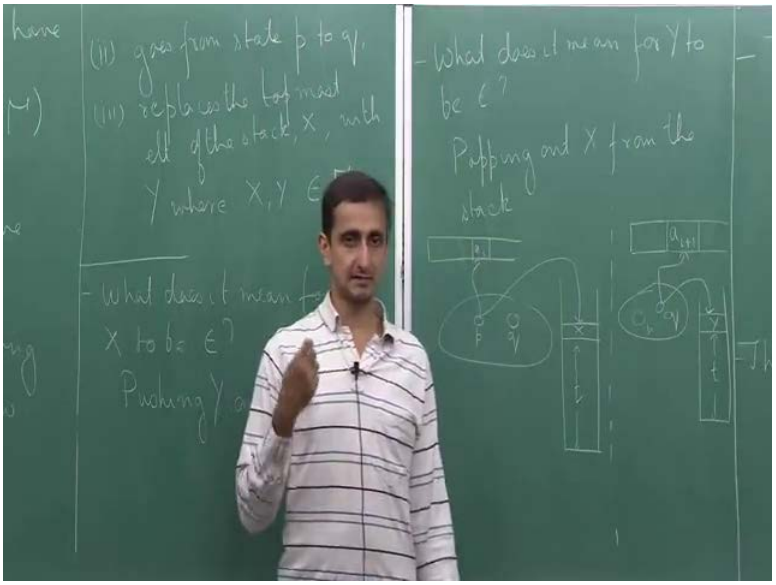
And similarly, what does it mean for y to be epsilon. So, once again observe that if we just look at the definition that it is replacing x with y, if y is epsilon, it just means that it is replacing x with epsilon, which is nothing. So essentially, what it is doing is it is popping out x from the stack; so x was already there on the stack, and now it is popping out x. So, this means that it is popping out x from the stack.

Basically if we have a computation like this, I mean pictorially what it would look like is, so let say that I have my finite control; and in my finite control, I have state p somewhere, I have a state q somewhere, the machine is the automaton is reading the i eth bit let say a i at any point of time. And I have my stack; and on my stack, I have x at the top of the stack, and let say some string t below it. So, t is the entire string that lies below. So, what happens in the next step is so if this is the current step.

What happens in the next step, is from a i the input head moves to the next symbol that is a i plus 1. The finite state, it moves from state p to state q. And the top of the stack is replaced so that the top of the stack which earlier had x got replaced with y; and the remainder of the stack remains the same. So, if there was some string t below it, the same string t actually remains. And as I said that it is not necessary that it actually reads a bit at every step, it can read the empty string also, the epsilon also. If that is the case then

actually the input head does not move, so the input heads stays where it is, the input head if it is reading a i, well it will read in the next step.

(Refer Slide Time: 26:35)



So, let me give a brief view of what we are going to see in the next lecture, in the next few lectures. So, next we will see a definition of or a formal definition of pushdown automata. We will look at some examples, and then we will discuss the computational power of these objects. So, if you recall at the beginning, I said that we want to study these objects; from the point of view that adding this extra data structure, which is a stack, how much extra power does it add that is our final goal.

So, we want to understand this question, what is the computational power of these objects, are they more powerful then finite automata, are they equally powerful or if they are more what kind of extra languages do they accept. But before that we will see the formal definition of pushdown automata, what does it formally mean for pushdown automata to accept an input, and then we will see some examples. So, I will stop here today.

Thank you.