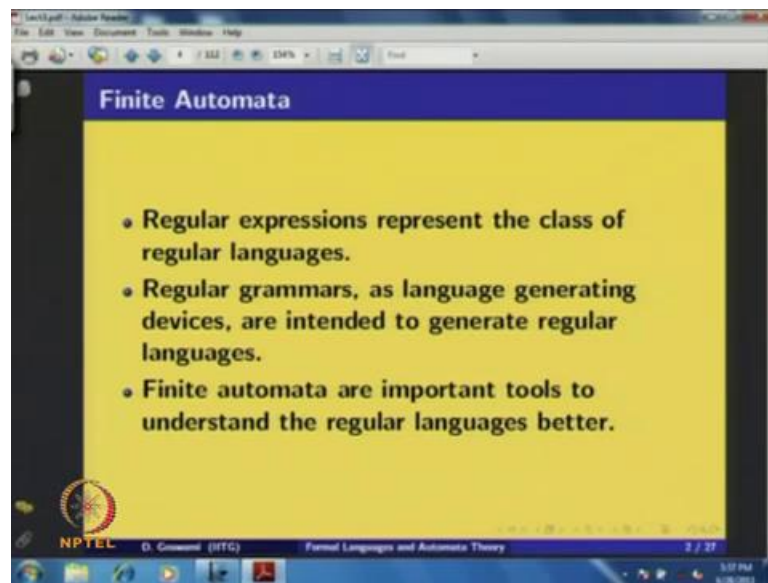


Formal Languages and Automata Theory
Prof. Diganta Goswami
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module - 3
Finite Automata
Lecture - 1
Deterministic Finite Automata

(Refer Slide Time: 00:31)



I have already seen that regular expressions represent the class of regular languages. And regular grammars as language generating devices are intended to generate regular languages. Now will see the important concept or tool finite automata, finite automata are important tools to understand the regular languages better.

(Refer Slide Time: 00:50)

The slide is titled "Finite Automata" in a blue header. It contains two bullet points on a yellow background:

- We will introduce the notion of finite automata and show that they model the class of regular languages.
- In fact, we observe that finite automata, regular grammars and regular expressions are equivalent.

The slide is part of an NPTEL presentation by D. Ganesan (IITG) on "Formal Languages and Automata Theory", slide 3 of 27.

We introduce the notion of finite automata and so that they model the class of regular languages. In fact, we observe that finite automata, regular grammars and regular expressions all are equivalent and the finite automata can model can be used to model many different physical systems. For example digital systems can be modeled or any software engineering problem can be model using finite automata.

(Refer Slide Time: 01:35)

The slide is titled "Finite Automata" in a blue header. It contains the following content:

- Consider the regular language - the set of all strings over $\{a, b\}$ having odd number of a 's.
- The grammar for this language is
$$\begin{aligned} E &\rightarrow bE \mid aO \\ O &\rightarrow bO \mid aE \\ O &\rightarrow \epsilon \end{aligned}$$
- A digraph representation of the grammar is:

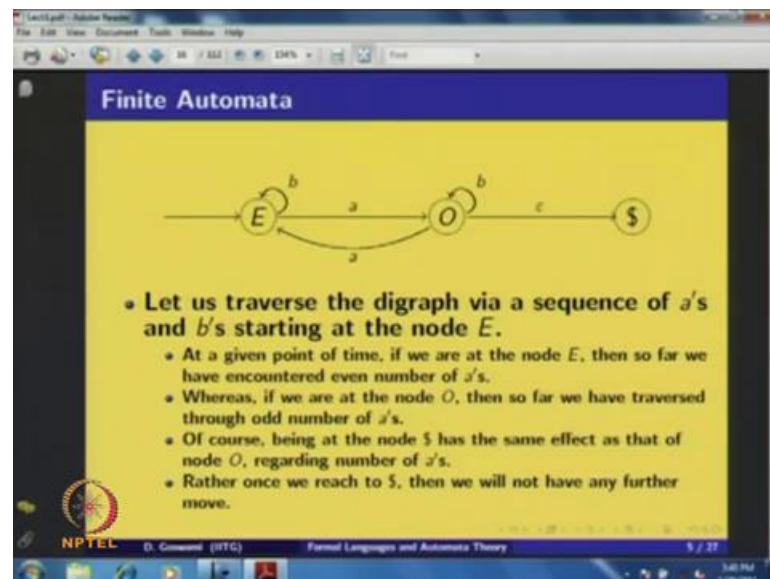
The digraph shows three states: E , O , and S . E is the start state, indicated by an incoming arrow from the left. S is the final state, indicated by a double circle. Transitions are: $E \xrightarrow{b} E$ (self-loop), $E \xrightarrow{a} O$, $O \xrightarrow{b} O$ (self-loop), $O \xrightarrow{a} E$, and $O \xrightarrow{\epsilon} S$.

The slide is part of an NPTEL presentation by D. Ganesan (IITG) on "Formal Languages and Automata Theory", slide 4 of 27.

Just consider, the regular language the set of all strings over a, b having odd numbers of a 's. Already, we have seen that the grammar for this language is E goes to bE , small bE

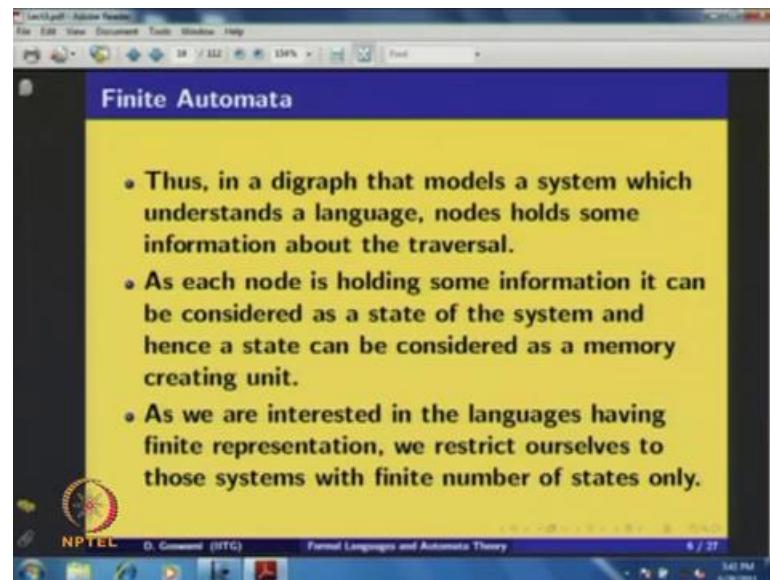
or E goes to a O, O goes to small b O and or o goes to small a E or O goes to absolute, this grammar we have seen that can generate the given regular languages. We have also already given a digraph representation the same grammar, which contains three nouns E labeled as E O and dollar and there are some transitions define according to the rule of the grammar.

(Refer Slide Time: 02:28)



Now, consider these diagram representation of the grammar and let us, traverse the diagram via a sequence of a 's and b 's starting at a node E . If you see that at a given point of time, if we are at the node E , then so far we have encountered even number of a 's. Similarly, if we are at the node O then so far we have traversed through odd numbers of a 's via a tradition from E to O level a . Of course, if we are at the node dollar, it has a same effect is date of node O ; that means, regarding the numbers of a 's encountered so for. Rather once, we reach to dollar then we have we will not have any further move; that means, we have to stay there only.

(Refer Slide Time: 03:38)



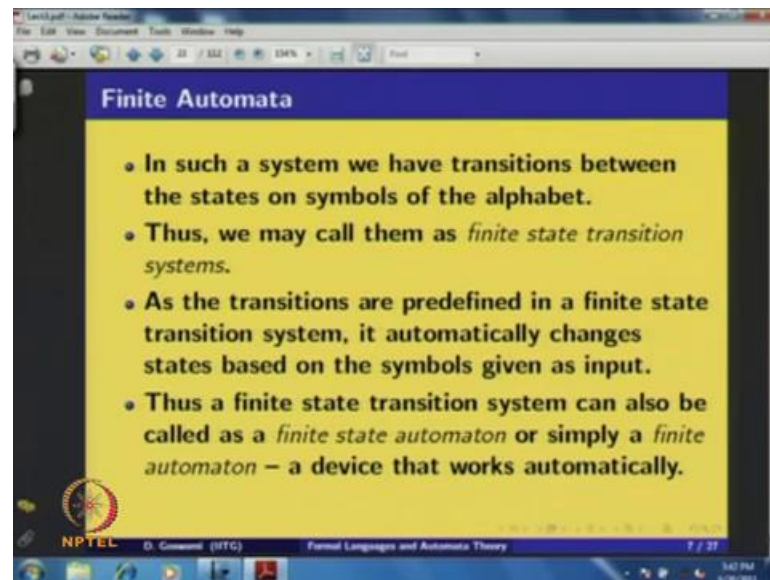
The image is a screenshot of a presentation slide titled "Finite Automata". The slide has a blue header bar with the title in white. The main content area is yellow and contains three bullet points. The slide is displayed in a window with a standard Windows interface, including a taskbar at the bottom with the NPTEL logo and the text "D. Ganesan (IITG) Formal Languages and Automata Theory 6 / 27".

Finite Automata

- Thus, in a digraph that models a system which understands a language, nodes holds some information about the traversal.
- As each node is holding some information it can be considered as a state of the system and hence a state can be considered as a memory creating unit.
- As we are interested in the languages having finite representation, we restrict ourselves to those systems with finite number of states only.

Now, in the diagram that models in this diagram that models a system which understands a language nodes holds some information about the traversal. As each node is holding some information it can be considered as a state of the system and hence a state can be considered as a memory creating unit. As we are interested in the languages having finite representation, we restrict ourselves to those systems with finite number of states only; that means the idea or concept that we have used in the diagram to represent a grammar. We are now going to introduce, the concept of finite automata, the nodes of finite automata we explained with the help of the concept that we had already introduced in the is in the digraph model.

(Refer Slide Time: 04:47)



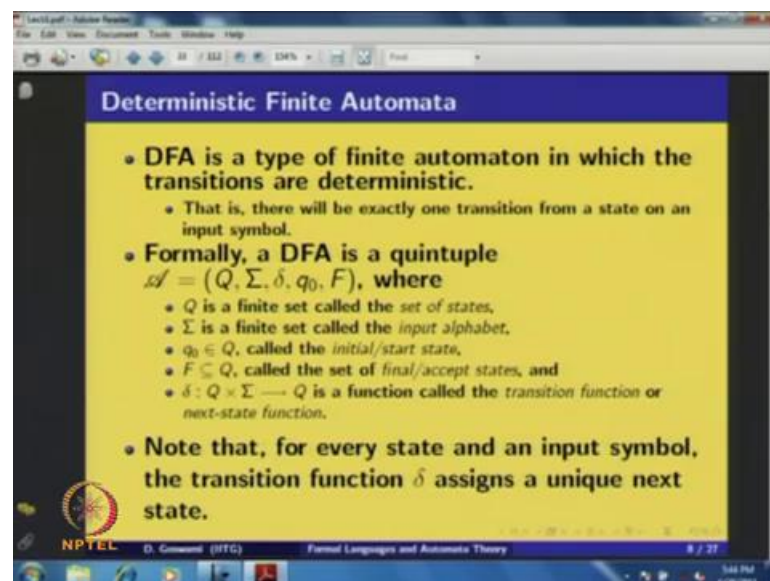
Finite Automata

- In such a system we have transitions between the states on symbols of the alphabet.
- Thus, we may call them as *finite state transition systems*.
- As the transitions are predefined in a finite state transition system, it automatically changes states based on the symbols given as input.
- Thus a finite state transition system can also be called as a *finite state automaton* or simply a *finite automaton* – a device that works automatically.

NPTEL D. Ganesan (IITG) Formal Languages and Automata Theory 7 / 27

So, in such a system we have transitions between the states on symbols of the alphabet. Thus, we may call them as a finite state transition system, because we have finite numbers of states and there, we some transitions and symbols to alphabet from on set to another. Is a transitions are predefined in a finite state transitions system, it automatically changes states bases on the symbols given as input. Thus a finite state transition system can also be called as a finite state automaton or simply a finite automaton that means, the device that works automatically and the plural form of automaton is automata.

(Refer Slide Time: 05:41)



Deterministic Finite Automata

- DFA is a type of finite automaton in which the transitions are deterministic.
 - That is, there will be exactly one transition from a state on an input symbol.
- Formally, a DFA is a quintuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where
 - Q is a finite set called the set of states,
 - Σ is a finite set called the input alphabet,
 - $q_0 \in Q$, called the initial/start state,
 - $F \subseteq Q$, called the set of final/accept states, and
 - $\delta: Q \times \Sigma \rightarrow Q$ is a function called the transition function or next-state function.
- Note that, for every state and an input symbol, the transition function δ assigns a unique next state.

NPTEL D. Ganesan (IITG) Formal Languages and Automata Theory 8 / 27

We have a look at a special kind of finite automata, which is deterministic finite automata are simply a DFA. DFA is a type of finite automation in which the transitions are deterministic in the sense that there will be exactly one transition from a state out of an input symbol. Formally, we define a DFA is a quintuple containing five components Q Σ δ Q_{naught} and F , where Q is a finite set called the set of states, Σ is a finite set called the input alphabet, Q_{naught} which belongs to the set of states is called the initial step or start state of the system. And capital F is subset of the set of states is called the state of final state or the state of accept states and δ is a transition function, is defined from $Q \times \Sigma$ to Q , this is called transition function or the next state. Now note that for every state and an input symbol the transition function δ assigns a unique next state, so in that sense this automation is deterministic.

(Refer Slide Time: 07:16)

DFA: Example

- Let $Q = \{p, q, r\}$, $\Sigma = \{a, b\}$, $F = \{r\}$ and δ is given by the following table:

δ	a	b
p	q	p
q	r	p
r	r	r

- Clearly, $\mathcal{A} = (Q, \Sigma, \delta, p, F)$ is a DFA.
- We normally use symbols p, q, r, \dots with or without subscripts to denote states of a DFA.

NPTEL D. Ganesan (IITC) Formal Languages and Automata Theory 9 / 27

Just consider an example, so the set of Q is contains the elements p, q, r , the input alphabet contains on a two elements symbols so a, b , the set of states is simply contains one state, this set of final states contains only one state from q that is the state r and δ the transition function is defined for express by the following transition table. So δ on p , DFA is in state p and input symbol is a , it will transit to state q , if it is an state p and input symbol is b , then it will transit to state p . Similarly on state q on input symbol a , you will transit to r and so that is how we have defined a transition function. Clearly, this is a DFA because you have define all these elements Q Σ δ p and F , We normally use the small letters p, q, r with or without subscript to denote states of a DFA.

(Refer Slide Time: 08:47)

Transition Table

- Instead of explicitly giving all the components of the quintuple of a DFA,
 - we may simply point out the initial state and the final states of the DFA in the table of transition function, called *transition table*.
- For instance, we use an arrow to point the initial state and we encircle all the final states.
- Thus, we can have an alternative representation of a DFA, as all the components of the DFA now can be interpreted from this representation.

NPTEL D. Ganesan (IITG) Formal Languages and Automata Theory 10 / 27

Now, instead of explicitly giving all the components of the quintuple of a DFA for example, the state of states Q improve how at σ transition table δ , so the final states F and so on. We may simply point out the initials state and the final states of the DFA in the table of transition function it is called the transition table. For instance, we may use an arrow to point the initial state and we encircle all the final states. Thus, we can have an alternative representation of a DFA as all the components of the DFA, now can be interpreted from this representation.

(Refer Slide Time: 09:32)

Transition Table

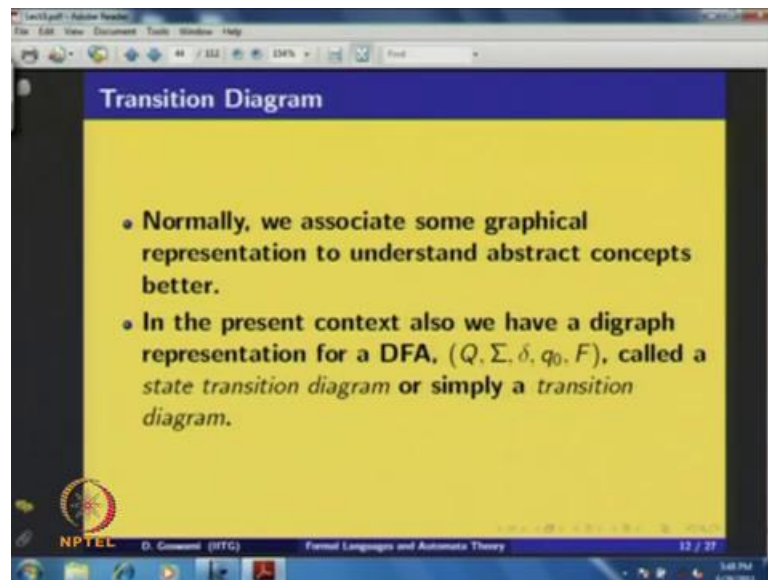
- For example, the DFA in previous example can be denoted by the following transition table.

δ	a	b
$\rightarrow p$	q	p
q	r	p
$\odot r$	r	r

NPTEL D. Ganesan (IITG) Formal Languages and Automata Theory 11 / 27

For example the DFA that you have just given can be denoted by the following transition table. So, in this case p is a start state, so we have written the same transition table, so p is pointed by an arrow, meaning that p is a start state and the state is encircled indicating that r is the final state or only final state. So, in this case all the elements of the quintuple can be found or can be determined from this transition table, because the state of steps contains any three states p , q and r , input alphabet contains two symbols a and b . Then, the transition table is defined like this the start state is p and the set of final states contains only one state that is r , therefore we need not give separately or explicitly all the elements of the quintuple and state. We can describe the DFA by using this kind of transition table, so this is a representation for giving a DFA.

(Refer Slide Time: 11:01)



Normally, we associate some graphical representation to understand abstract concepts better. So, in this context also we have a digraph representation for a DFA, so containing this five tuples which is called a state transition diagram or simply a transition diagram.

(Refer Slide Time: 11:24)

Transition Diagram

- Every state in Q is represented by a node.
- If $\delta(p, a) = q$, then there is an arc from p to q labeled a .
- If there are multiple arcs from labeled a_1, \dots, a_{k-1} , and a_k , one state to another state, then we simply put only one arc labeled a_1, \dots, a_{k-1}, a_k .
- There is an arrow with no source into the initial state q_0 .
- Final states are indicated by double circle.

So, to give the digraph representation, every state in Q is represented by a node. If $\delta(p, a) = q$ as defined as transition table then there is an arc from the node p to the node q labeled on a . If there are multiple arcs from labeled a_1, a_2 up to a_{k-1} and a_k state to another state, then we simply put only arc labeled a_1, a_2 up to a_k from one state to the other state. And there is arrow which no source to the initial state q_0 and all the final states are indicated by double circle.

(Refer Slide Time: 12:27)

Transition Diagram

- The transition diagram for the DFA

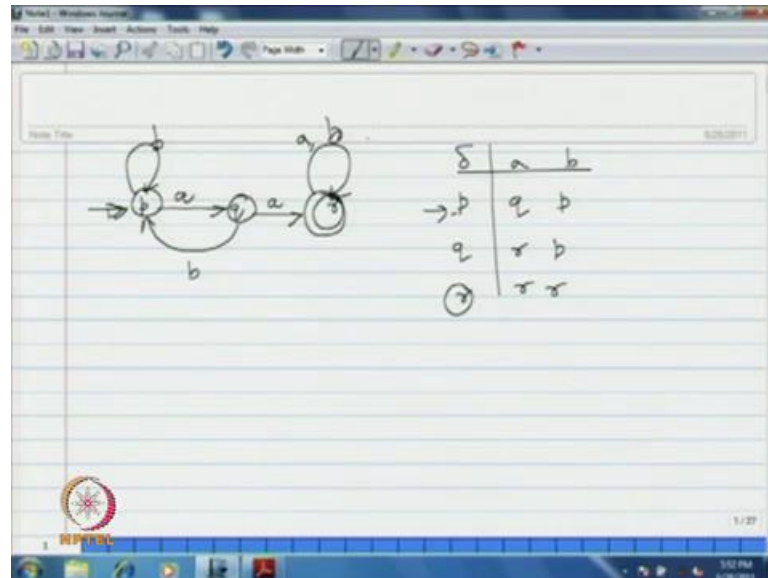
δ	a	b
$\rightarrow p$	q	p
q	r	p
$\odot r$	r	r

is as below:

- The two transitions from the state r to itself on symbols a and b , are indicated by a single arc from r to r labeled a, b .

For example, consider transition diagram for the DFA that we have already given by the transition table which is as per rule that we have discussed. Now, you can draw the transition diagram as below.

(Refer Slide Time: 12:53)



So, the table is given by any delta on a and b, we have three steps p q and r, p is a start state r is the final state, p on a goes to q p is goes to r p and this is r r. Now, according the rule that we defined for contracting transition diagram for a given DFA, we quit nodes for the states. So, this is the start state, we indicated by a circle from an arrow from no way to this state you cater state for node for r r states p q and since r is a final state, we indicate by double circle.

And, as per a transition from p on a goes to q is indicate by given arrow, label a from p to q, similarly, p on b goes to p is indicate by giving a self look to p on symbol b, then q on a goes to r. And arrow labeled, a goes to r, q on b goes to p, q on b goes to p, then r on a and b remains at state r, so on a and b it remains at state r its self, so instead during two looks over here you are using a and we are labeled by a comma b. So, this is the given transition diagram for a given example, so we have got this particular transition diagram for example already distract, the two transitions from the state r on symbols a and b are represent by a single arc from r to r labeled a b as earlier.

(Refer Slide Time: 16:01)

Extended Transition Function

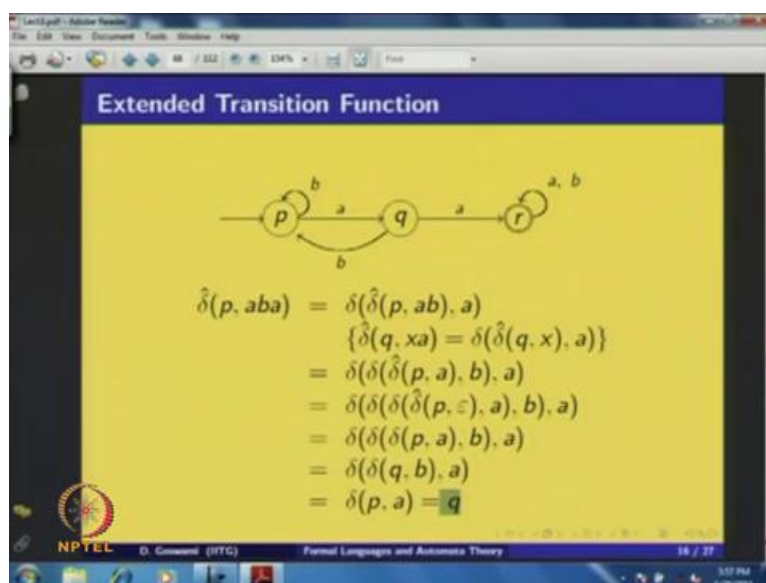
- The transition function δ assigns a state for each state and an input symbol.
- Can be extended to all strings in Σ^* ,
 - assigning a state for each state and an input string.
- The extended transition function $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$ is defined recursively as follows:
For all $q \in Q$, $x \in \Sigma^*$ and $a \in \Sigma$,
$$\hat{\delta}(q, \varepsilon) = q \text{ and}$$
$$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a).$$

NPTEL D. Ganesan (IITG) Formal Languages and Automata Theory 15 / 27

Now, the transition function delta assigns a state for each state and an input symbol eternally this can be extended to all strings in sigma star assigning a state for each state and then input string; that means, we can have an extended transition function from Q to sigma star, which is defined as below.

For all $q \in Q$ and string x within the sigma star and on symbol length to sigma, $\hat{\delta}$ which are extended function $E: Q \times \Sigma^* \rightarrow Q$ on existing will give q ; that means, without taking any input that will remain on the same set and $\hat{\delta}(q, \varepsilon) = q$; that means, without taking any input that will remain on the same set and $\hat{\delta}(q, xa)$ for x is a string and either symbol to the x is a string can be defined as recursively $\hat{\delta}(\hat{\delta}(q, x), a)$ which will give as a state q comma a ; that means, we first apply this extended transition function on x starting at q and then apply this delta original transition function on single input a .

(Refer Slide Time: 17:44)

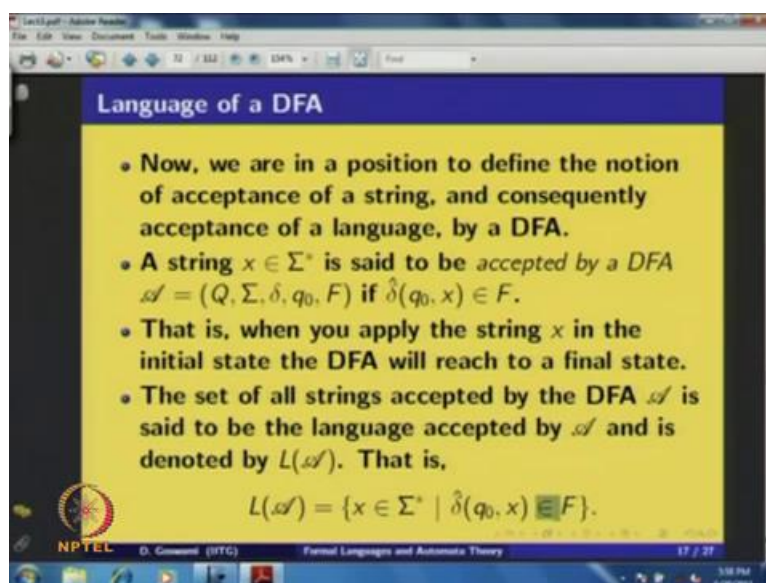


Now, just consider this extended same example that I have already given and let us consider, how the compute? this extended transition function or a string say a b a. So, in this case, this string a b to be consider x according definition and we apply delta hat on the string a b, so p comma a b delta hat p comma a b.

And then we apply delta on whatever state we arrive at here comma a; that means, we use the second rule. Next, we can apply recursively in sets the same we can be apply the word here; that means, delta hat p a comma b and then apply delta on represent the same function delta on a. Similarly, we can convey for the since here after displaying you have only epsilon you apply it delta hat on p epsilon and then we applied the first rule since delta hat p epsilon is p itself, because it will remain the same set therefore, the next step you will get from this will get on the p itself. Hence it will be delta p a, the joint you getting here a delta p a.

Now, since delta p a is already define transition function p a gives you q, so the next hat will be delta p s q, so q b. Now, I apply delta q b, delta q b is again p and it will be delta of p a, so delta of p a, is now q, so it gives now q, therefore apply delta hat extended transition function on a b a starting with this p, we get in the state q. So, this how we compute extended transition function.

(Refer Slide Time: 20:26)



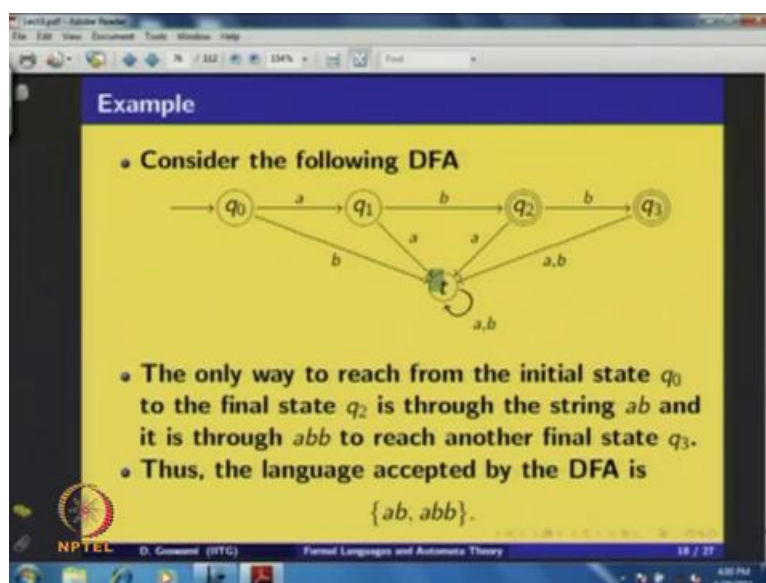
Language of a DFA

- Now, we are in a position to define the notion of acceptance of a string, and consequently acceptance of a language, by a DFA.
- A string $x \in \Sigma^*$ is said to be *accepted* by a DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ if $\delta(q_0, x) \in F$.
- That is, when you apply the string x in the initial state the DFA will reach to a final state.
- The set of all strings accepted by the DFA \mathcal{A} is said to be the language accepted by \mathcal{A} and is denoted by $L(\mathcal{A})$. That is,

$$L(\mathcal{A}) = \{x \in \Sigma^* \mid \delta(q_0, x) \in F\}.$$

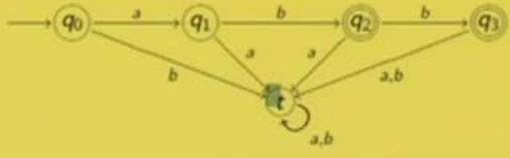
Now, we are in a position to define the notion of acceptance of a string and consequently acceptance of a language by a DFA. A string x belong to Σ^* is said to be accepted by a DFA \mathcal{A} , if $\delta(q_0, x) \in F$; that means, if you process the string x starting in the initial state q_0 , then if you arrive at a final state then we say that the string x is accepted by the DFA. Now, the set of all substrings accepted by the DFA, \mathcal{A} is said to be a language accepted by \mathcal{A} and it is denoted by $L(\mathcal{A})$; that means, $L(\mathcal{A})$ is state of all strings x belong to Σ^* such that $\delta(q_0, x) \in F$.

(Refer Slide Time: 21:28)



Example

- Consider the following DFA



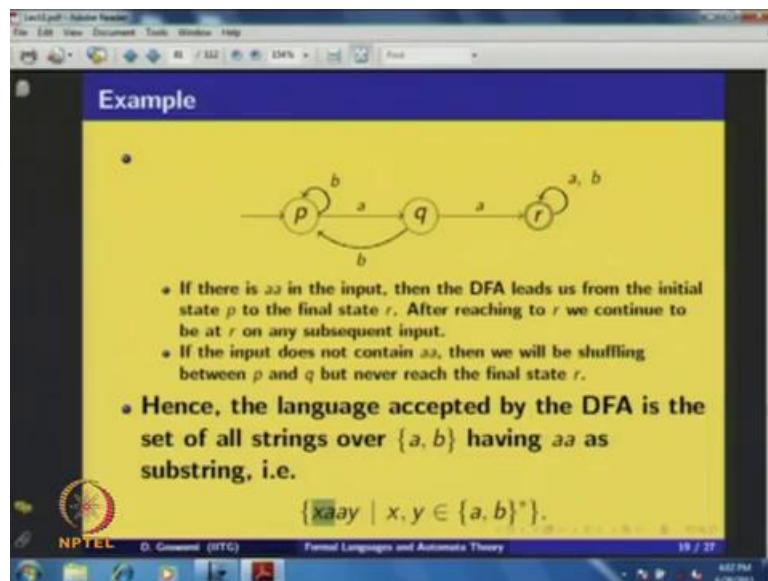
- The only way to reach from the initial state q_0 to the final state q_2 is through the string ab and it is through abb to reach another final state q_3 .
- Thus, the language accepted by the DFA is

$$\{ab, abb\}.$$

Now, consider the DFA which contains this five steps to the following the transition has given the figure. Now, if you look carefully the only way to reach from the initial state q_0 to the final state q_2 is through the string $a b$, because they follow the path from q_0 to q_1 and q_1 to q_2 that is the only path to which we can arrive at into starting at q_0 .

And, again from q_0 , you can arrive at the other final state q_3 following the path q_0 to q_1 , q_1 to q_2 , q_2 to q_3 . So where, the level the part is $a, a b, b$; that means, we can reach the final states either q_2 or q_3 , but following the strings following the paths with labels as a strings the $a b$ and $a b b$; that means, the language accepted by the DFA is only two strings $a b$ and $a b b$, no other strings you will be accepted by the DFA. So, in this case is a DFA because you have defined the transitions on every step on every input symbol. You see that here, t is a transcript because there is no path out of t from t once we have arrived at t there is no way to go out of t to any other state. So, you have to remain the same set.

(Refer Slide Time: 23:31)

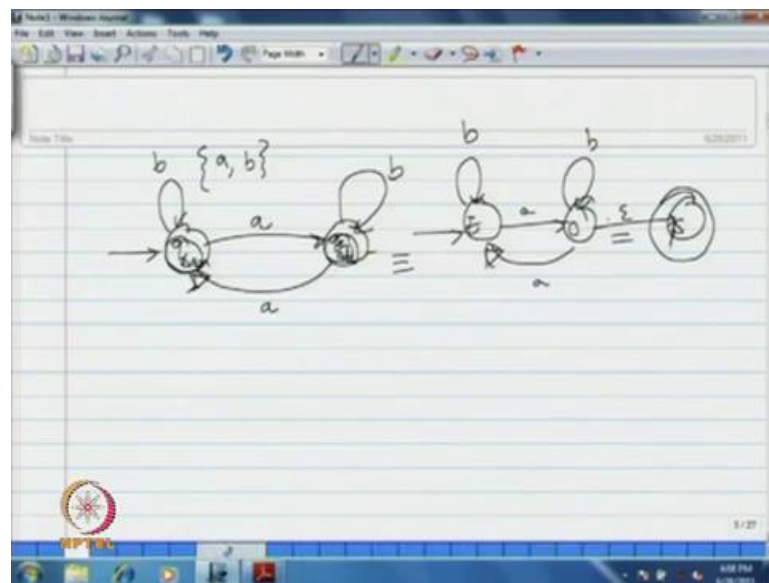


Consider the example that we have already described. So, let us see for the language of these DFA, if we observe carefully find that if you have a a in the input is a and a a in the input, then DFA leads us from the initial state, p to the final state; that means, the string aa is accepted by the DFA. So, once we are in the state or let us say continue two states on r whatever they include we consider on any such kind of input we have to be in the same state r . So, if the input does not contain aa , then we will be shuffling between the states

p, on the state q because whatever we take it is a string of b here in the same state p. If you take a single a will come to a q and from q if you do not take n a any view read as again to p show we have to shuffle between p and q.

Therefore, the language accepted by the DFA is the set of all strings a b having a a as substring, which can written as set all strings out of from x a a y solves that x y belongs to sigma star. So; that means, any string which has a a as substring will be accepted by the DFA and no another string will be accepted by the DFA.

(Refer Slide Time: 25:18)



Let us consider some more examples, just consider the example for which we have given a regular grammar; that means, the set of all strings over a b set of all strings over a b having odd numbers of sets, we are already given a regular grammar for each one and we have given a digraph representation. Now since a complaint is a set of all strings over a b containing odd numbers of else we need to remember the numbers of a's encounters so for and, the sets as to remember that information.

Note that, if it is a star set say q, whatever so if we first take b as an input, you see that we have to been same set, because it need not remember numbers of bits you need to ignore a numbers of bits, so therefore on b it will remain on the same state. So, s one as it erects the first a it has to go to some others next set to remember that it has ordered odd numbers of sets, so on a it will entered. So, call it as q odd and call it that as q even.

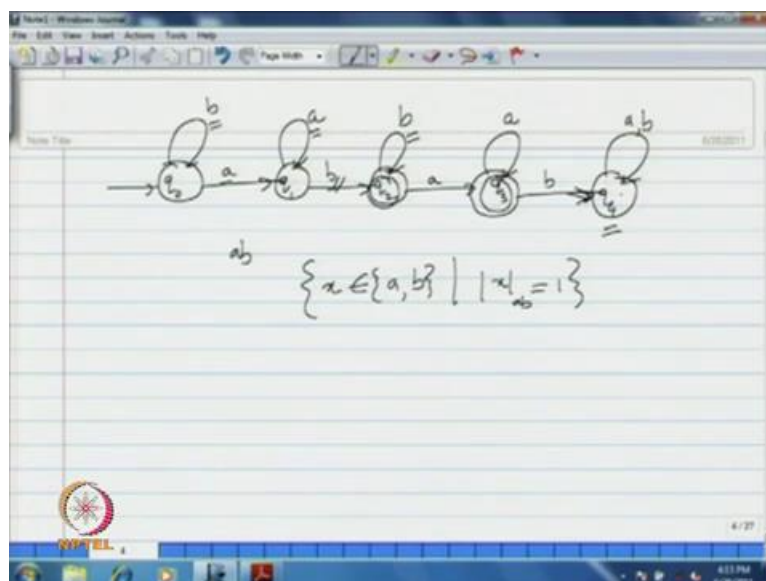
Because, the first at ϵ in fact, remember the odd numbers s encounter so far and the second state will remember that it as encounter, so far odd numbers of else and a first set will remember even numbers of else. So in the second state, that generate q odd if you gets some b again, so if we not do anything it remain the same state because it need not remember numbers of b 's, but once it is gets an input as a .

So, it as to transit from this state to the under set which nothing but, the first set because in sub cycles the numbers of a 's will be even. Therefore, it is very clear that if the DFA remains in the state q even, the numbers of a 's in the strings so far a process so far will be even and the numbers of a 's process so far and we are if it is in the state q .

So, therefore, if you make q r as the final state then this DFA will accept all strings having odd numbers of one's, so therefore, these are equal DFA for a given language over a b , set of all strings over a b having odd numbers of a 's. Similarly, instead of making this state is final state if we make this state is the final state first one, then clearly this DFA we accept all those strings having even numbers of a 's.

Just remember that the digraph representation that even for the language, the set of all strings having odd numbers of a 's was like this is a E o . So, this was the digraph representation for a given language which we construct for from the given regular language and see these two are identical if you assume that this is a final state. We discuss of course, the differ by this state and this condition have seen you and we did not allow normally in DFA this kind of transition on input ϵ , but later on will see that if you allow this kind of transition, we get some order task of finite automata which mean discussed later.

(Refer Slide Time: 30:30)



Let us consider another example of a DFA say this DFA has q_0 is a first step on input b . Suppose, it may it sends the same set on a it tends to q_1 , then on a it remains the same set on b it goes to state q_2 and say q_2 is a final state, on b it remains the same set and on a q_2 goes to state q_3 , which is again suppose a final state. So, on q_3 on a it remains q_3 itself and on b it goes to say q_4 and q_4 on a b it remains in a same set.

Now let us see, the language accepted by the DFA will see the behavior of this DFA by observing what its steps represents or what information its set retains. It is clear that in the input contains only b 's then the DFA remains in the initial state q_0 on b q_0 on b will remains in the same set.

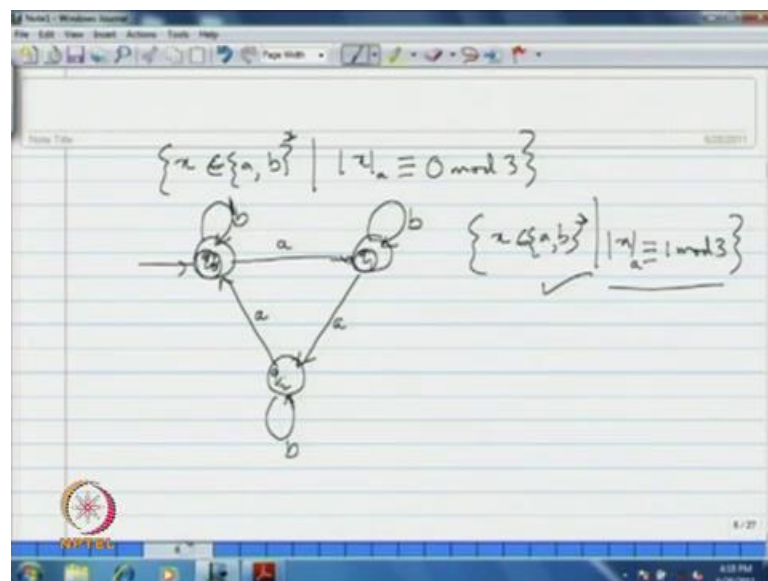
On the other hand if the input has an a in the DFA goes to state q_1 from q_0 it goes to state q_1 on input a . An subsequence its will remain in the same sets q_1 , thus the role of state q_1 is to remember that are understand that the input hence at least one a . So, that is both information in remember by the state q_1 for the DFA goes from state q_1 to q_2 via b , so on input b in the DFA will go from q_1 to q_2 . And, remains in the state q_2 on any for a input which is b does q_2 recognizes that the input has an occurrence some a b because once at arrives at q_2 the input must have an occurrence of a b , an since on any input any for a b remain the same state q_2 .

Now here, q_2 you alter that q_2 is a final state therefore, the DFA expects all those things which have one occurrence of a b , if it is a mono concern a b the string, then this

string accepted by this DFA. Subsequently we see that if we have a number a's q 2 on a and go to q 3 and q 3 on any for a is remain and same set q 3 itself and q 3 is also a final state.

So, all substrings will be accepted if we which has one a b, subsequently terminate by any numbers of b's or any numbers of a's. But from q 3, if you takes input b, then we will arrive at state q 4, here q 4 is a step that because from q 4 on any input if you remain in the state q 4 itself and q 4 is not a final state or exhausted. So, therefore, we see that the language of this DFA is nothing but the set of all strings x belong to a b such that the number of a's source that the substring a b occurs exactly once. So, this for the language expected by the b DFA. So here, the roles of that sets are very clear in a congaing language.

(Refer Slide Time: 36:08)



Consider in the example, suppose the language the set of all strings over a b such that say numbers of a's in a string a is divisible by set. Let us construct, a DFA is x service language that consider this to a star set say q 0, now this auto metal need not remember how many views it has raise so far. So, sets as remember the numbers of age only, if the numbers of age in the string is deviating then it should accept otherwise you should reject should not accept.

So, q 3 on b it will remain in the same set, it simply ignores the numbers of b's in the input. So once you gets on input either input a 1 then input, then it will go to understatement

say it is q_1 ; that means, in q_1 , its given in set whenever it arrives set q_1 , the numbers of a's it is so far is at least one a it has arrived that will remember at this point.

So again, it will ignore any b's that consider again on input a it will arrive at sets saying q_2 , in step q_2 its clear at either the red two a's so far. So, again it we ignore any b's it has rates; that means, and it will remains the same state q_2 , on q_2 if gets a's an input which and go to q_0 , because at a point it as red three is sure any numbers of these, but we as got so for three a's. So therefore, if you make q_0 as the final state you see that.

So, numbers of a's that is so far will be divisible by three, because you can take the same part again from q_0 q_1 ; that means, if you from q_0 , again after taking it once if goes from q_0 to q_1 , there will be four ends from q_2 to q_1 to q_2 one goes on input a it has got five ways from q_2 to q_0 again as got is six ways. So, s one as it is state q_0 , the numbers of age erase so far, we multiple of 3.

Similarly, the some automaton can be modified to accept the state of all strings over a b, such that numbers of a's will divided by 3 it gives one as reminder. So, to accept this language simply what we have to do instead of this as final state we need to make q_1 is a final state. Similarly, if you want to have is if you divide the numbers of a's by chain and the remainder is two in such a case q_2 has to be a final state instead of q_0 are q_2 . So, therefore, here the roles of states are very clear remembering what information or what input it as let so far.

(Refer Slide Time: 40:51)

Slide Time

$$\{x \in \{a,b,c\}^* \mid |x|_a + |x|_b \equiv 0 \pmod{3}\}$$

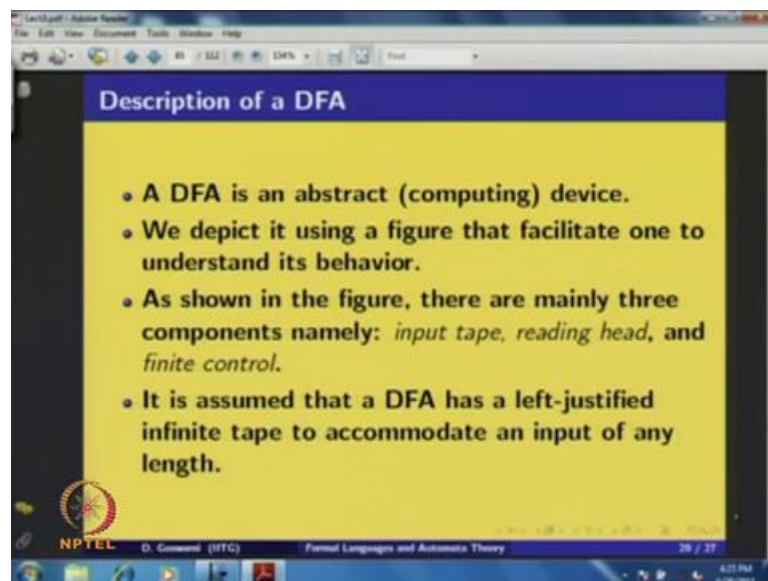
$$|x|_a + |x|_b \equiv 2 \pmod{3}$$

6/10/2011

4:10 PM 6/10/2011

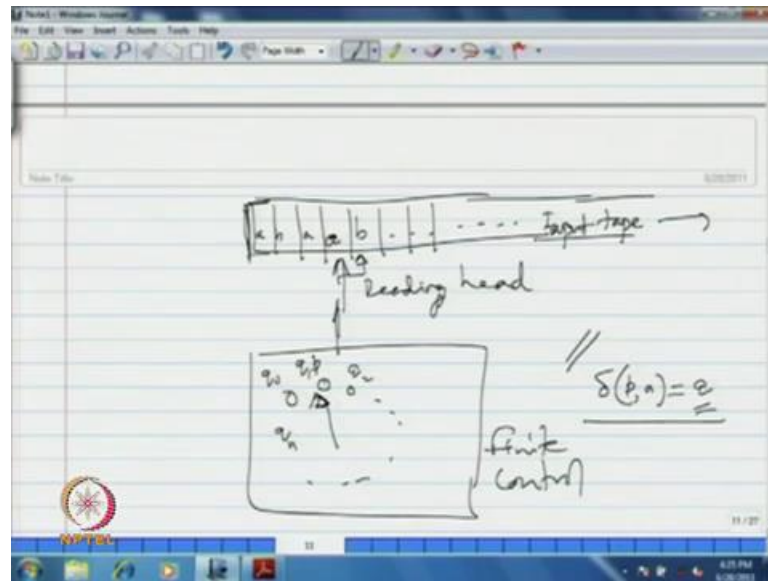
Now, just consider an example of a language, so x belongs to $a^*b^*c^*$ that is the set of all strings are $a^*b^*c^*$ such that the number of a 's and the number of b 's is divisible by 3. Just considering the concept that you have described in the previous example we can construct the DFA for this language, similarly you can have many other variations. So, the over $a^*b^*c^*$ such that the number of a 's and the number of b 's in the string only defined by three. So, it is two is the remainder and you can have any combinations from this.

(Refer Slide Time: 42:11)



Now, you want to give the behavior of DFA in formal way, but part of that we will first see since we have describe we have say that the DFA is a computing device. We introduce the no some of computation; we first depict the behavior of DFA by using a figure that facilitate to understand how it behaves.

(Refer Slide Time: 42:54)

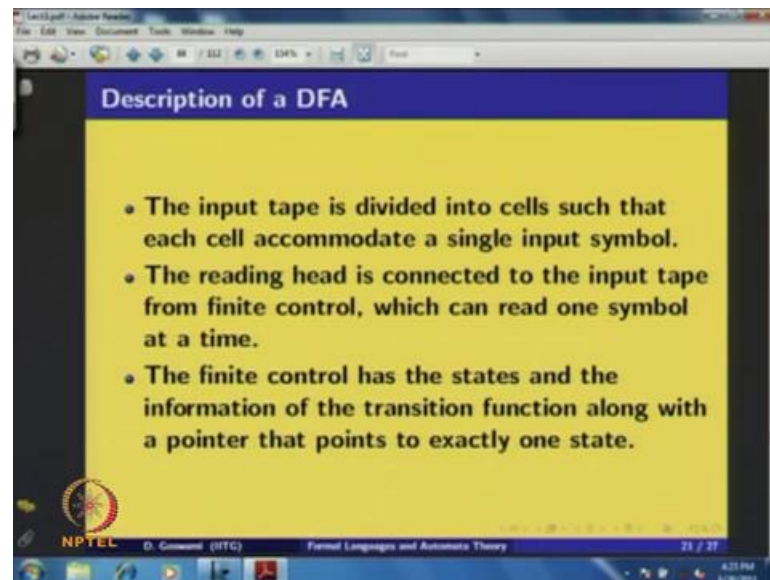


Just consider, you explain the behavior DFA by using this kind of figure. So, DFA contains node only an infinite, so the infinite and this step is lets justify it; that means, left side justify and to the right side you can go to infinity, does a reading hat which reads the input notion input up.

They reading head and the finite control, the input tape is normally divided into some source. So, the right side it is infinite, so the we infinite numbers of upsets unit, so every can contain a symbol fontal input of habit. So, it is a b a a b and so on, and the reading head can a read one symbol at time from the input the reading as connected to the finite control, finite control contains various steps are the definite automata. So, it is q_0, q_1, q_2 and so on, how to say? Some q_n and the pointer to point to a particular step the steps which the finite automata will be in at a any given pointed to by this pointer.

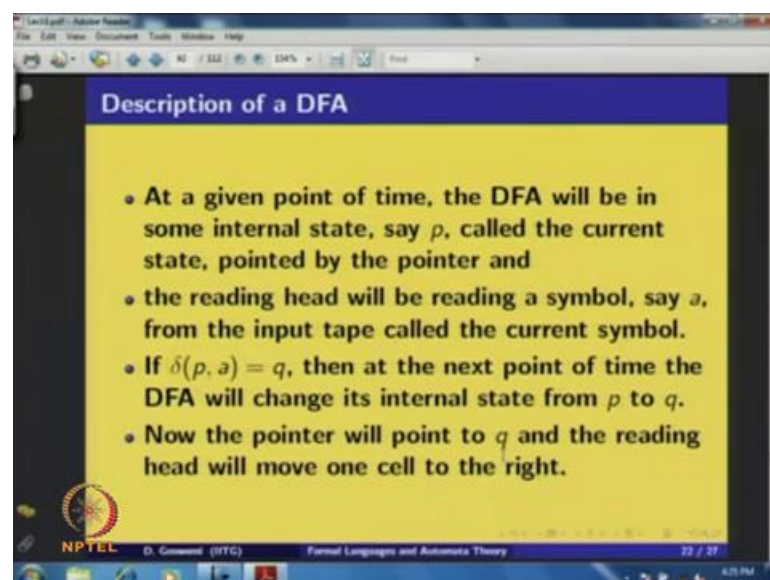
Now would this diagram, we can describe the behavior of the DFA are a finite automata then from the Mainer. So, it is on a figure they are mainly three components namely input tape reading head and finite control, it is assumed that a DFA has a left justified infinite tape to accommodate an input of any length, because the right side is if you may go to infinity.

(Refer Slide Time: 45:40)



The input tape is divided into cells such that each cell accommodate one input symbol. Now, the input for that the reading head is connected to the input tape from the finite control which can read one symbol at a time from input tape. The finite control has the states and the information of the transition function along with a pointer that points to exactly one state.

(Refer Slide Time: 46:06)

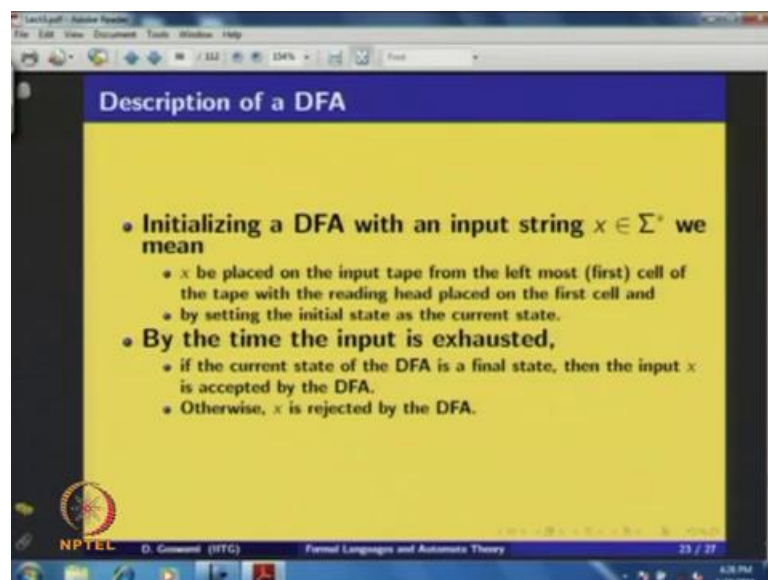


At a given point of time the DFA will be in some internal state say p called the current state pointed by the pointer, and the reading head will be reading a symbol say a from the

input tape called the current symbol; that means, at any instant time it is reading a symbol a from the input tape and it is at a particular step suppose it is p this instead p pointer two by the p is a current set and a is the input it is reading from the input tape.

Now, the condition defined by the DFA which is defined as $\delta(p, a)$. So, based on transition this automata or this DFA we transition automatically, suppose $\delta(p, a) = q$ you can look they automatically go to the state q and the pointer will now point to the particular state q which it has transition to and the reading head we move one cell to read the next cell on the input tape. So this, how the automata DFA we behave? So, its $\delta(p, a) = q$, then at the next point of time the DFA will change its internal state from p to q and the pointer will point to q and the reading head will move one cell to the right.

(Refer Slide Time: 47:51)

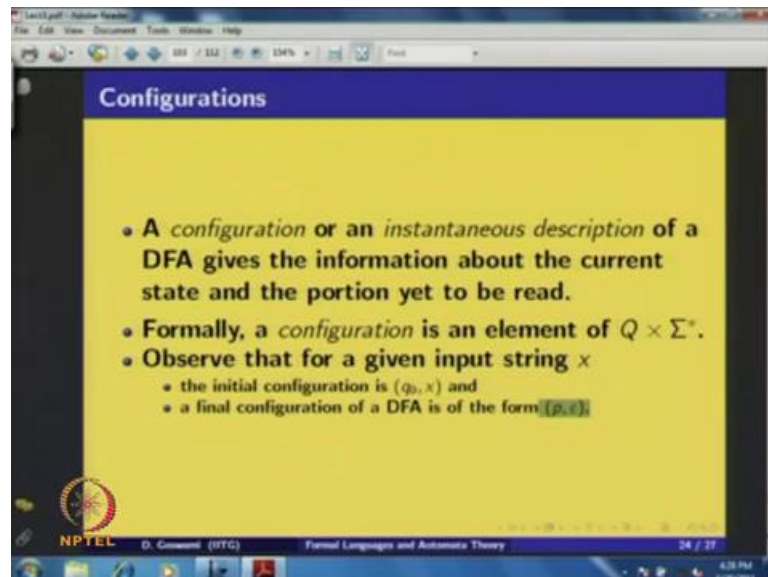


Now, you can initialize a DFA with an input string x by initialization we mean that x be placed on the input tape from the left most first cell of the tape with the reading head placed on the first cell. So, x replaced on the input tape starting from the left and the reading head points to the very first cell that will read the first input and the pointer in the finite control we point to the initial state and that is the current state of the DFA.

Now, since every transition defined on any state on any input symbol some transitions define therefore the DFA we transit to a next state as the one more transition table automatically, and it will continue on though the exacted. So, by the time the input is

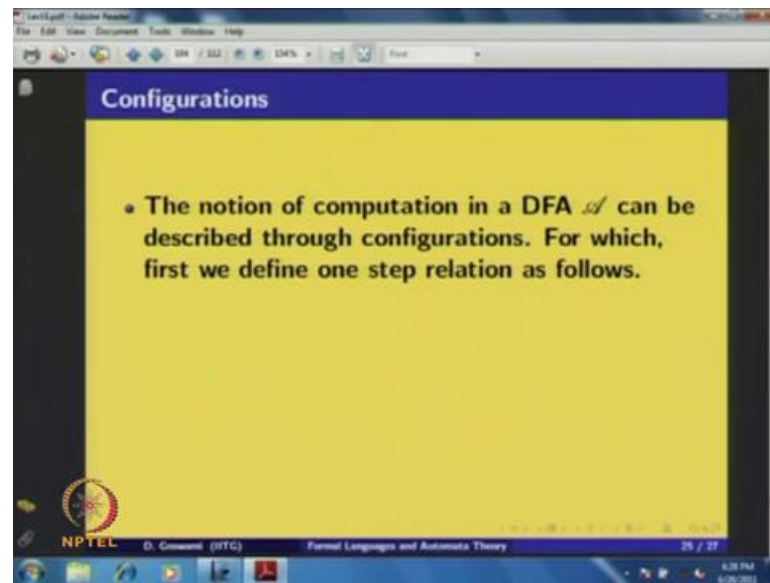
exactly the current state of DFA is a final state then the input x is accepted by the DFA. Otherwise, if the current state after exactly the input is not a final state or exactly then we say that x is rejected by the DFA. So, that is how the DFA behaves?

(Refer Slide Time: 49:26)



Now, we see how we can compute by using DFA, to see the computation we first describe configurations. The knowledge of configuration of the DFA, the configuration or an instantaneous description of a DFA gives the information about the current state and the portion yet to be read. Formally, a configuration is an element of $Q \times \Sigma^*$. Now, for a given input string x , the initial configuration over $q_0 x$, because the initial state is q_0 and the portion yet to be read is x , the whole input string. And the final configuration of a DFA will be all the form $p \epsilon$; that means, the input string is exactly this x is exactly that as I will get ϵ , and we are arrived at sums that p . So, this is the initial configuration and this is the final configuration.

(Refer Slide Time: 50:46)



The notion of computation in a DFA can be described through configurations.