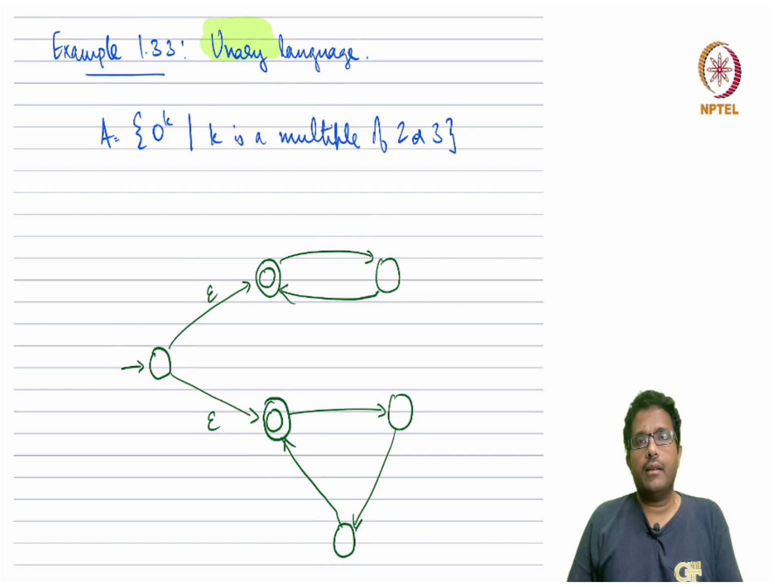
Theory of Computation

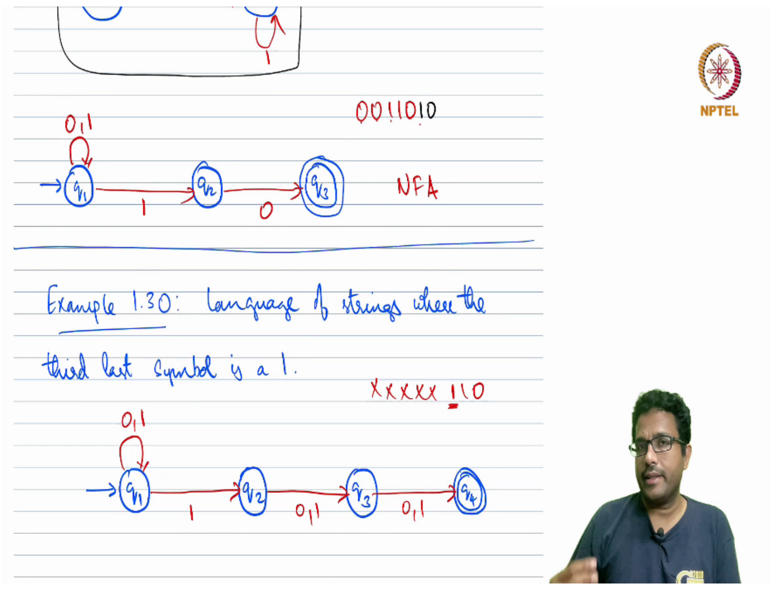
Professor Subrahmanyam Kalyanasundaram
Department of Science and Engineering
Indian Institute of Technology Hyderabad
Formal Definitions and Examples of Non-Deterministic Finite Automata (NFA)

(Refer Slide Time: 00:16)



Example 1.33: Unary language.

$A = \{0^k \mid k \text{ is a multiple of } 2 \text{ or } 3\}$

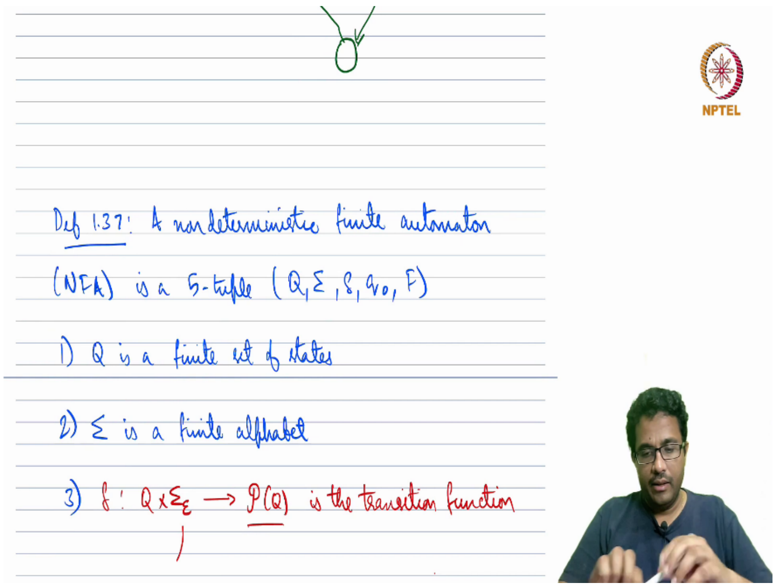


Example 1.30: language of strings where the third last symbol is a 1.

Hello, and welcome to lecture 7 of the course Theory of Computation. In lecture 6, we saw non-deterministic finite automata, which are NFA's. We did not formally define it, but we saw examples and we tried to understand it through these examples. It was the same as DFA's but with some seemingly added flexibilities like you could have multiple outgoing arrows for the same symbol, you could also make epsilon transitions, and then you accept a string.

There could be multiple ways to process it correctly. But if at least one of them is a valid accepting computation, you accept the string. Now, let us come to the formal definition of NFA's.

(Refer Slide Time: 01:15)

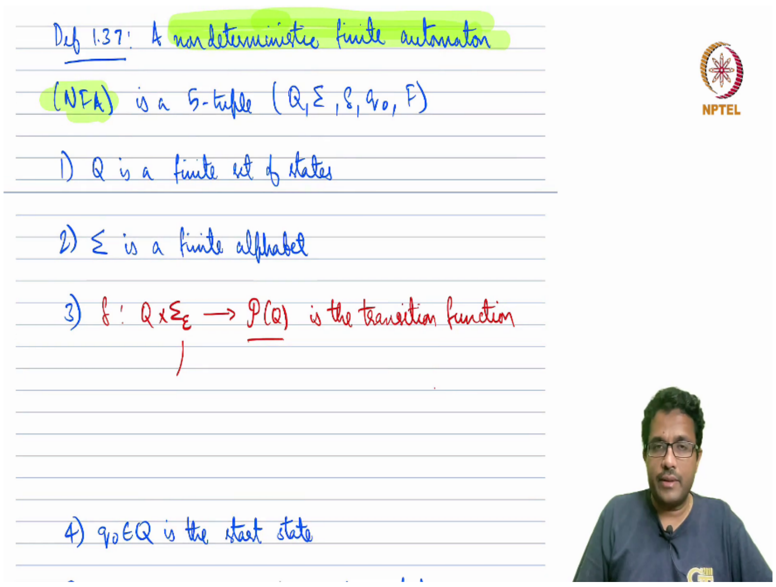


Def 1.37: A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

1) Q is a finite set of states

2) Σ is a finite alphabet

3) $\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ is the transition function



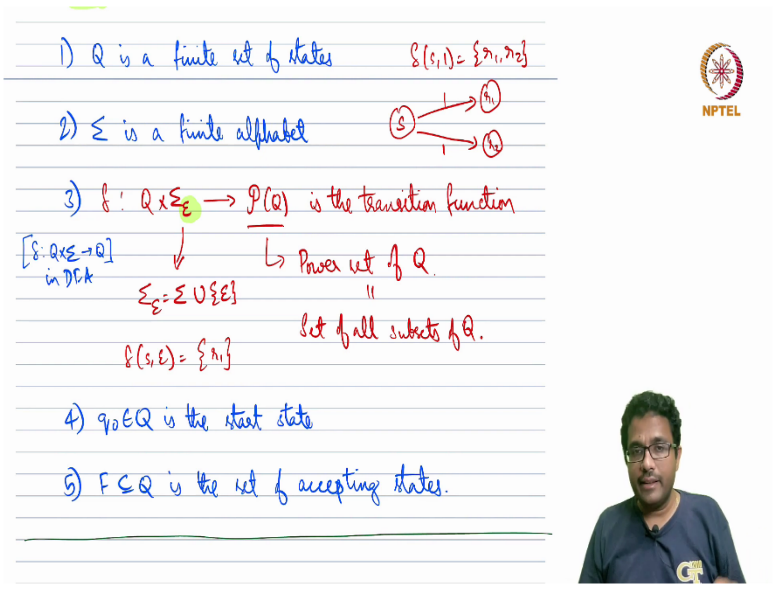
Def 1.37: A nondeterministic finite automaton (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

1) Q is a finite set of states

2) Σ is a finite alphabet

3) $\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ is the transition function

4) $q_0 \in Q$ is the start state



1) Q is a finite set of states $\delta(s,1) = \{r_1, r_2\}$

$$\begin{tikzcd} & r_1 \\ s \arrow[ur, "1"] \arrow[dr, "1"'] & \\ & r_2 \end{tikzcd}$$

2) Σ is a finite alphabet

3) $\delta : Q \times \Sigma_\varepsilon \longrightarrow \underline{\mathcal{P}(Q)}$ is the transition function

$[\delta : Q \times \Sigma \to Q]$ in DFA

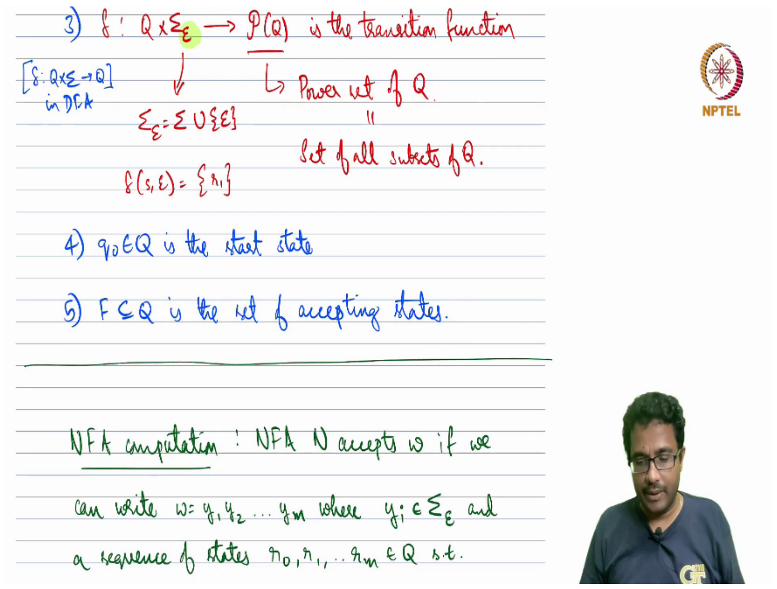
$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

↳ Power set of Q
$\|$
Set of all subsets of Q.

$\delta(s,\varepsilon) = \{r_1\}$

4) $q_0 \in Q$ is the start state

5) $F \subseteq Q$ is the set of accepting states.



3) $\delta : Q \times \Sigma_\varepsilon \longrightarrow \underline{\mathcal{P}(Q)}$ is the transition function

$[\delta : Q \times \Sigma \to Q]$ in DFA

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

↳ Power set of Q
$\|$
Set of all subsets of Q.

$\delta(s,\varepsilon) = \{r_1\}$

4) $q_0 \in Q$ is the start state

5) $F \subseteq Q$ is the set of accepting states.

<u>NFA computation</u> : NFA N accepts w if we can write $w = y_1 y_2 \ldots y_m$ where $y_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots r_m \in Q$ s.t.

So, the definition is very similar to what we saw in DFA's. NFA or non-deterministic finite automata 5-tuple (Q, Σ, δ, q_0, F). Here if you see the definition Q is the set of states, Σ has a finite alphabet, q_0 is a start state and F is a set of accepting states, this is exactly the same as what we saw in DFA's. The only place where this definition differs is the transition function written in red.

So, in the case of DFA's, we saw δ: Q x Σ → Q. So, basically, if you are at a certain state, and you have a certain symbol, then you go to the next state. But over here, it is not like that, because here we have flexibilities. If you see here at a certain state S and then you see a 1 there could be two possible next states that you can go to, or there will be three possible next states, or maybe there is no outgoing arrow labelled 1.

So now we define the transition function as δ: $Q \times \Sigma_\epsilon \rightarrow P(Q)$, here $P(Q)$ is the power set of Q which is the set of all subsets of Q. So, the idea is that if you are at a certain state, let us say s, and the 1 can take you to let us say r_1 and r_2, then you will say that

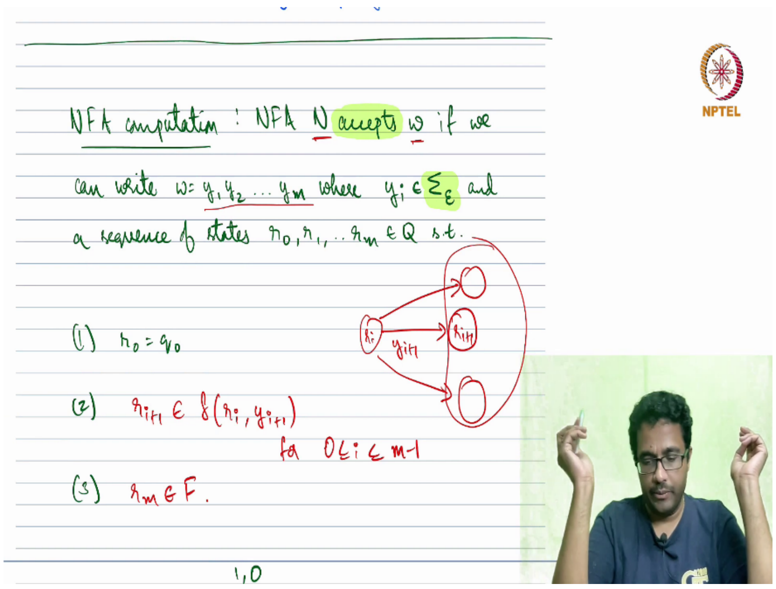
$$\delta(s, 1) = \{r_1, r_2\}$$

If δ can take you to 2 possible states, then you say that the transition function outputs a set of the possible states that 1 takes you to. This could also be a singleton set, if there is only one outgoing arrow marked 1 or it could also be an empty set ϕ if there is no outgoing arrow marked as 1.

We have $Q \times \Sigma_\epsilon$ (notice the subscript epsilon), why do we have a subscript epsilon? Basically, we want to accommodate the empty transitions that we said earlier. So, Σ_ϵ is nothing but $\Sigma \cup \{\varepsilon\}$. So, it is the set of all symbols plus the empty symbol. This enables the empty transitions.

We may also write things like $\delta(s, \epsilon) = \phi$ which means that there are no empty transitions, outgoing from s. So, if instead of an empty set you have $\{r_1\}$. This means that there is one outgoing arrow mark empty string which takes you to r_1. And q_0 is the start state F is a set of accepting states it is exactly the same as before.

(Refer Slide Time: 06:04)



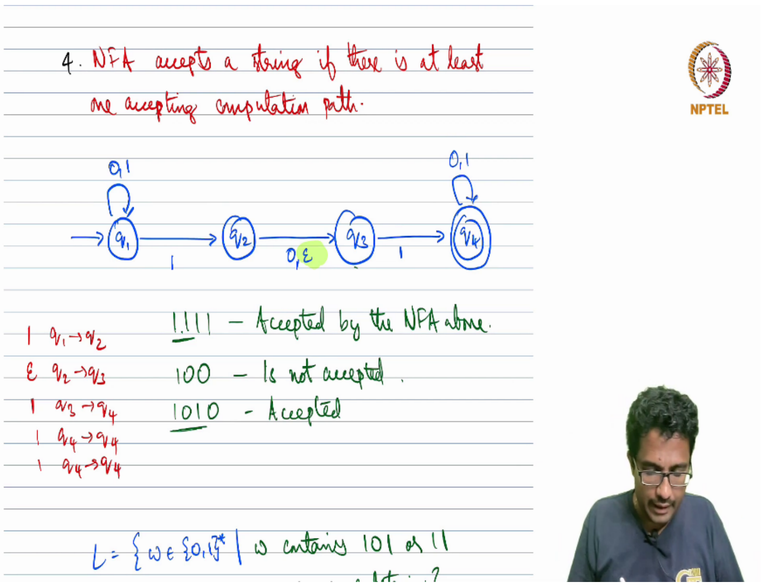
NFA computation: NFA N accepts w if we can write $w = y_1 y_2 \ldots y_m$ where $y_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots r_m \in Q$ s.t.

(1) $r_0 = q_0$

(2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for $0 \le i \le m-1$

(3) $r_m \in F$.

r_i
r_{i+1}
y_{i+1}
1,0



4. NFA accepts a string if there is at least one accepting computation path.

0,1
q_1
1
q_2
0,ε
q_3
1
q_4
0,1

1 $q_1 \to q_2$	1111 – Accepted by the NFA above.
ε $q_2 \to q_3$	100 – Is not accepted.
1 $q_3 \to q_4$	1010 – Accepted
1 $q_4 \to q_4$	
1 $q_4 \to q_4$	

$L = \{w \in \{0,1\}^* \mid w$ contains 101 or 11

Now, we have to formally define what constitutes acceptance states. So, in the case of DFA, it was very easy because there was only one way to process strings. And at the end, are you in an accepting state or not? Now, there are multiple ways, but as I have been repeatedly saying, you accept at least if there is one valid way to process that string. This notational thing is just for formal representation and completeness, even if this is confusing, even if the notation is confusing, as long as you have a thorough understanding of what constitutes acceptance, that will be good.

So, we say that N accepts w, N is the NFA, w is the input string, if we can write w as

$$w = y_1\ y_2\ y_3 \ldots\ldots y_m$$

So, notice that I am not using n as length, instead, I am using m since there may be empty symbols in between. If you look at the DFA that we saw above which accepts the string 1111.

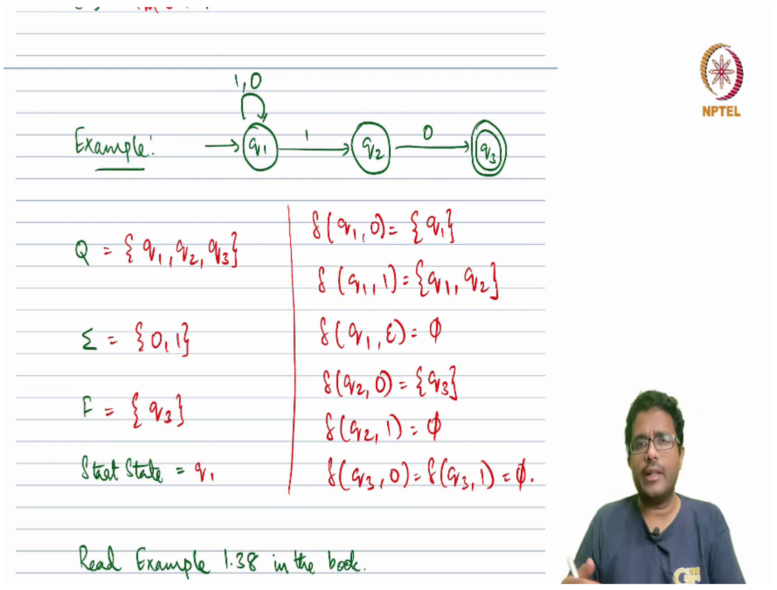
So, how does it accept the string 1111 So, the first one takes it from q_1 to q_2, then an empty transition takes it from q_2 to q_3, then the next one takes it from q_3 to q_4 and the last two 1's keep it at q_4.

So we can accept a string if there is a way to split the string into y_1 to y_m, including epsilon, such that the NFA for this particular splitting up there is a valid computation that takes it from the starting state to the accepting state. So, r_0 must be the start state and for each i we have-

$$r_{i+1} \in \delta(r_i, y_{i+1}) \text{ for } 0 \leq i \leq m-1$$

So, what we want to say is that the set contains r_{i+1}. Also r_m must be an accepting state. So, which means if there is a valid if there is a way to split up the string in a valid way and processes strings such that it ends in an accepting state, we see that the NFA N accepts w.

(Refer Slide Time: 12:15)



Example:

$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "1,0"] \arrow[r, "1"] & q_2 \arrow[r, "0"] & \boxed{q_3} \end{tikzcd}$$

$Q = \{q_1, q_2, q_3\}$

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

$F = \{q_3\}$

Start State $= q_1$

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

$\delta(q_1, 1) = \{q_1, q_2\}$

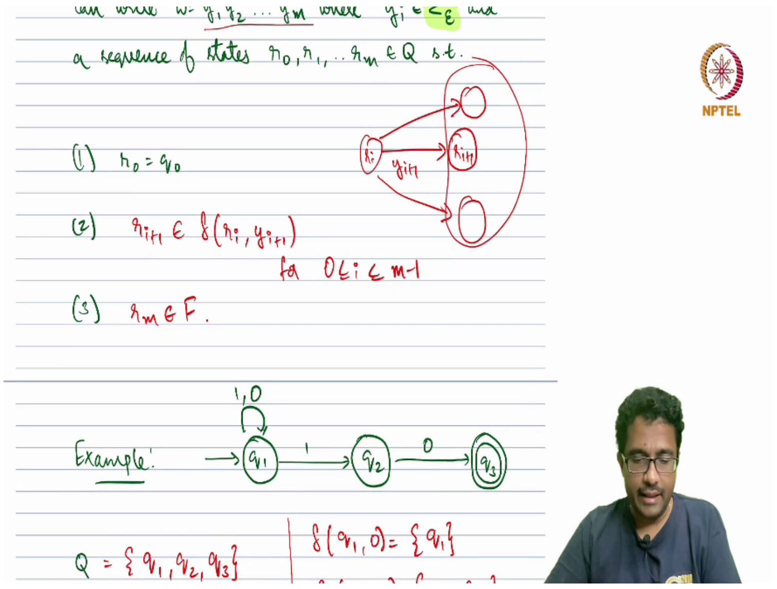
$\delta(q_1, \epsilon) = \phi$

$\delta(q_2, 0) = \{q_3\}$

$\delta(q_2, 1) = \phi$

$\delta(q_3, 0) = \delta(q_3, 1) = \phi.$

Read Example 1.38 in the book.



can write $w = y_1 y_2 \dots y_m$ where $y_i \in \Sigma_\varepsilon$ and

a sequence of states $r_0, r_1, \dots r_m \in Q$ s.t.

(1) $r_0 = q_0$

(2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for $0 \le i \le m-1$

(3) $r_m \in F$.

$$\begin{tikzcd} & \bigcirc \\ r_i \arrow[ur] \arrow[r, "y_{i+1}"'] \arrow[dr] & r_{i+1} \\ & \bigcirc \end{tikzcd}$$

Example:

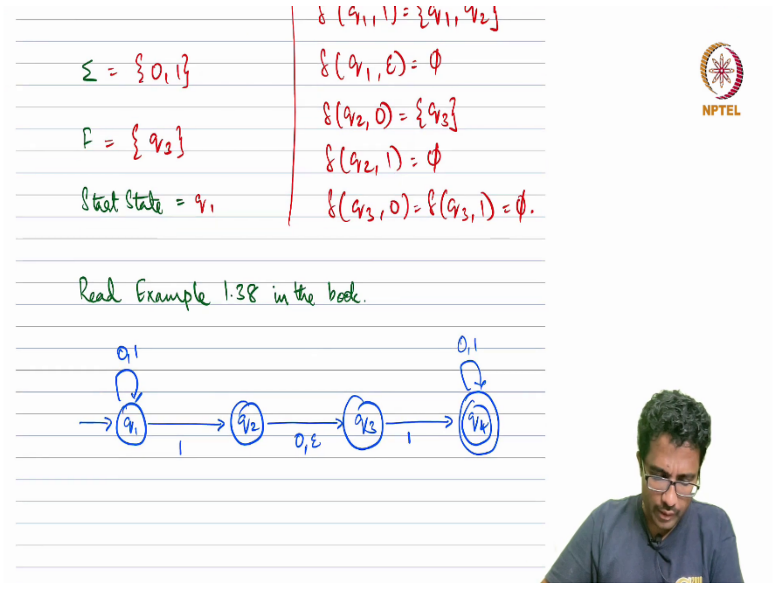
$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "1,0"] \arrow[r, "1"] & q_2 \arrow[r, "0"] & \boxed{q_3} \end{tikzcd}$$

$Q = \{q_1, q_2, q_3\}$ | $\delta(q_1, 0) = \{q_1\}$

The above example shows a NFA which accepts all the strings that end in 10. So, let us see what are the states here? So, states are 3 states $\{q_1, q_2, q_3\}$, the alphabet is binary $\{0, 1\}$. The set of accepting states is just $\{q_3\}$, the start state is q_1. Now the transition function δ also has to be defined for each Q and Σ_ϵ (please refer to the image attached below).

I have not marked a couple of empty transitions, but you can fill this in. So, this is an example and the formal notation stating when a string gets accepted, hopefully this is clear.

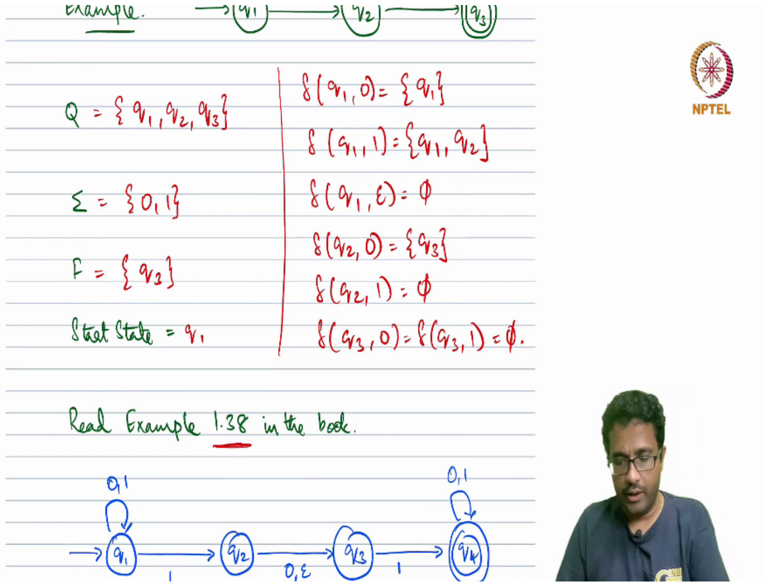
(Refer Slide Time: 15:03)



	$\delta(q_1, 1) = \{q_1, q_2\}$
$\Sigma = \{0, 1\}$	$\delta(q_1, \varepsilon) = \phi$
	$\delta(q_2, 0) = \{q_3\}$
$F = \{q_3\}$	$\delta(q_2, 1) = \phi$
Start State $= q_1$	$\delta(q_3, 0) = \delta(q_3, 1) = \phi$.

Read Example 1.38 in the book.

$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "0,1"] \arrow[r, "1"'] & q_2 \arrow[r, "0,\varepsilon"'] & q_3 \arrow[r, "1"'] & \boxed{q_4} \arrow[loop above, "0,1"] \end{tikzcd}$$



Example.

$$\begin{tikzcd} \arrow[r] & q_1 \arrow[r] & q_2 \arrow[r] & q_3 \end{tikzcd}$$

$Q = \{q_1, q_2, q_3\}$

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

$F = \{q_3\}$

Start State $= q_1$

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

$\delta(q_1, 1) = \{q_1, q_2\}$

$\delta(q_1, \varepsilon) = \phi$

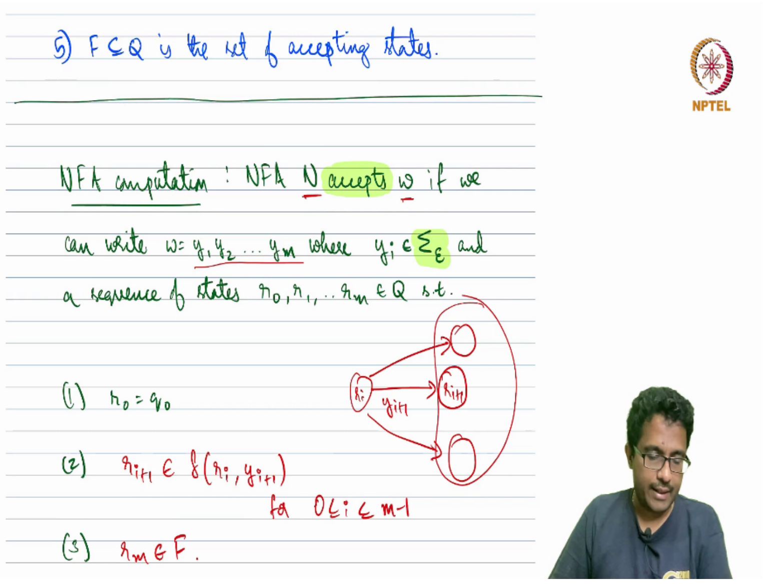
$\delta(q_2, 0) = \{q_3\}$

$\delta(q_2, 1) = \phi$

$\delta(q_3, 0) = \delta(q_3, 1) = \phi.$

Read Example 1.38 in the book.

$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "0,1"] \arrow[r, "1"'] & q_2 \arrow[r, "0,\varepsilon"'] & q_3 \arrow[r, "1"'] & q_4 \arrow[loop above, "0,1"] \end{tikzcd}$$



5) $F \subseteq Q$ is the set of accepting states.

NFA computation : NFA N accepts w if we can write $w = y_1 y_2 \ldots y_m$ where $y_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots r_m \in Q$ s.t.

(1) $r_0 = q_0$

(2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for $0 \leq i \leq m-1$

(3) $r_m \in F$.

$$\begin{tikzcd} & \circ \\ r_i \arrow[ur] \arrow[r, "y_{i+1}"'] \arrow[dr] & r_{i+1} \\ & \circ \end{tikzcd}$$

Please also read example 1.38 from the book. So, basically the same exercise that we just did for this NFA, I want you to read example 1.38. In that example there are 4 states and then binary alphabet etcetera, but please, please try to work out these: what is Q, what is Σ_ϵ, what is δ and so on.

So, we have seen in the NFA's formal definition what constitutes acceptance. And we already said in lecture 6 that NFA's are at least as powerful as DFA's. In the next lecture, we will see that they are no more powerful than DFA's. So, we will see why they are equivalent. And that is all. That is all from me in lecture 7 and see you at lecture 8. Thank you.