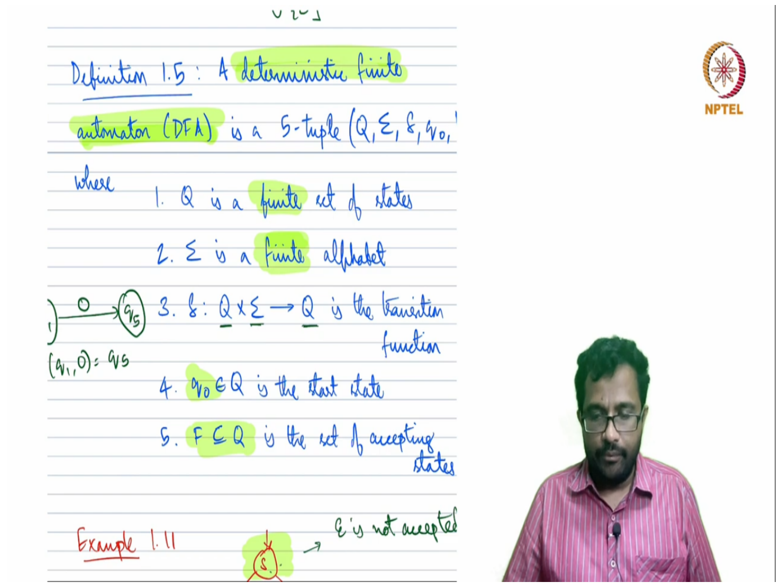
Theory of Computation

Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture 3
An Introduction to Finite Automata and Regular Languages – Part - 02

(Refer Slide Time: 00:16)



Definition 1.5: A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0,$

where

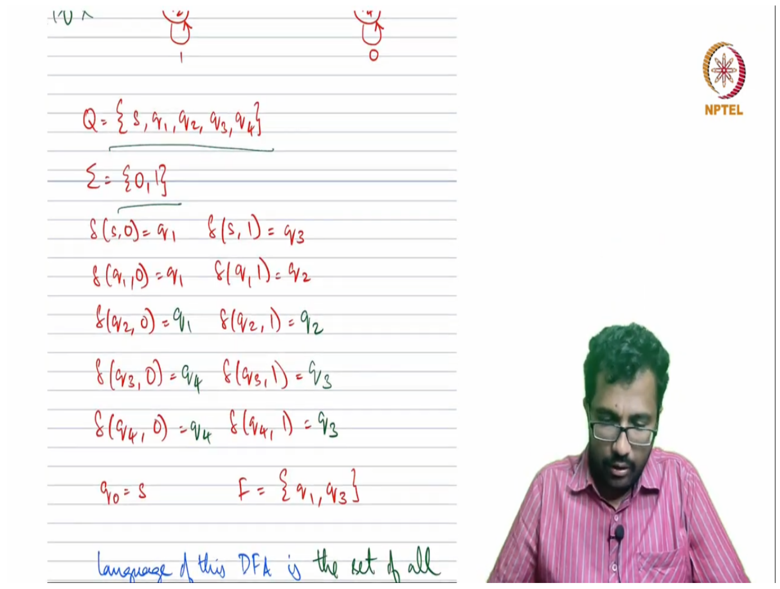
1. Q is a finite set of states
2. Σ is a finite alphabet
3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accepting states

$\xrightarrow{0} q_5$

$(q_1, 0) = q_5$

Example 1.11

s → ε is not accepted



$Q = \{s, q_1, q_2, q_3, q_4\}$

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

$\delta(s,0) = q_1$ $\delta(s,1) = q_3$

$\delta(q_1,0) = q_1$ $\delta(q_1,1) = q_2$

$\delta(q_2,0) = q_1$ $\delta(q_2,1) = q_2$

$\delta(q_3,0) = q_4$ $\delta(q_3,1) = q_3$

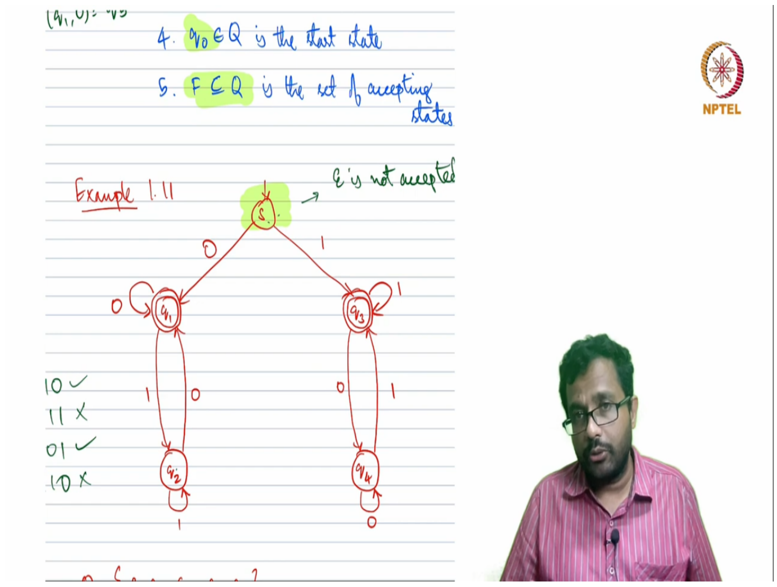
$\delta(q_4,0) = q_4$ $\delta(q_4,1) = q_3$

$q_0 = s$ $F = \{q_1, q_3\}$

Language of this DFA is the set of all

So, now let me formally define what a DFA is. We saw this Q, Σ, δ, q_0 and F. So, this is the Q is the set of states which is finite, Σ is a set of alphabet which is finite, δ is the transition function. In other words, they are the arrows.

(Refer Slide Time: 00:44)



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

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

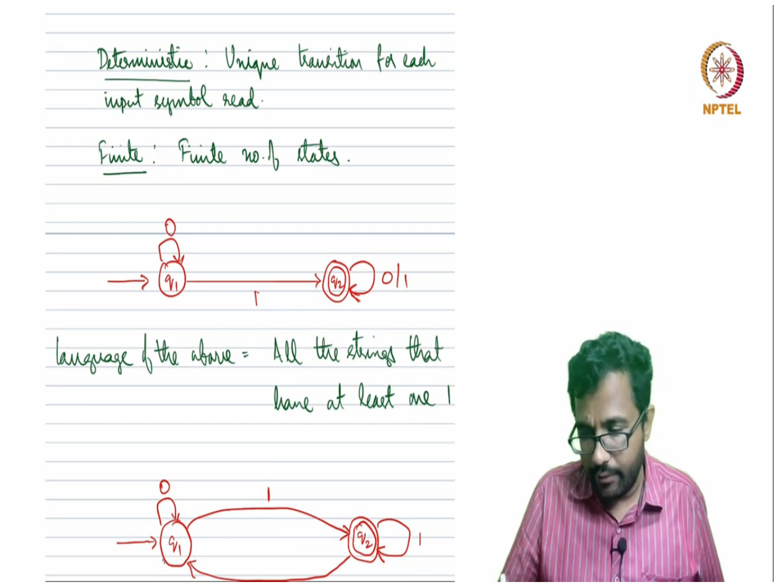
Example 1.11

ε is not accepted

10 ✓
11 ✗
01 ✓
10 ✗

So, if the alphabet is binary, if it is {0, 1}, then every state will have two outgoing arrows. So, here for instance, s has two outgoing arrows, one for 0, one for 1, q1 has two outgoing arrows one for 0, one for 1. So, in this case, one of these outgoing arrows comes back to q1 which is okay.

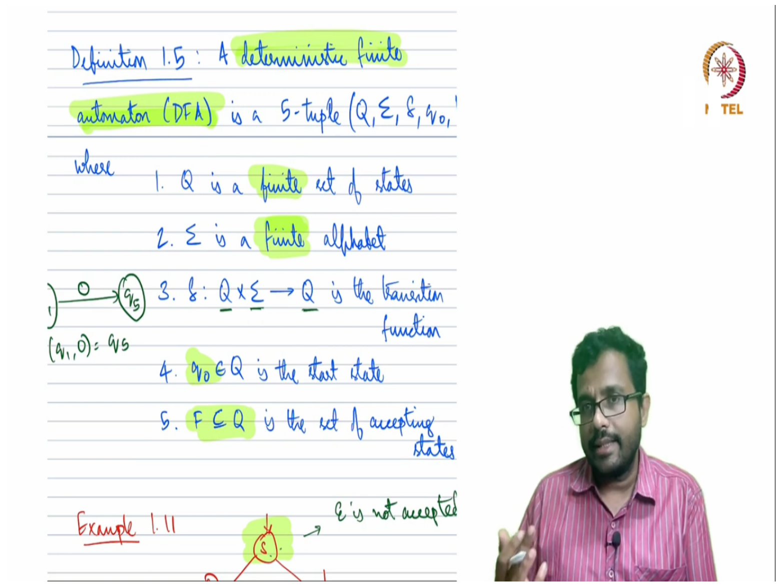
(Refer Slide Time: 01:06)



Deterministic : Unique transition for each input symbol read.

Finite : Finite no. of states.

language of the above = All the strings that have at least one 1



Definition 1.5 : A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0,$

where

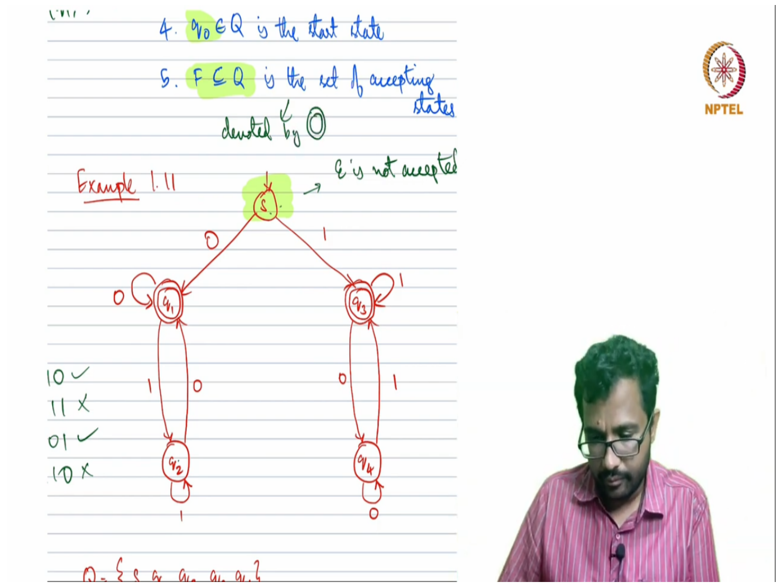
1. Q is a finite set of states
2. Σ is a finite alphabet
3. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accepting states

$\xrightarrow{0} q_5$

$(q_1, 0) = q_5$

Example 1.11

ε is not accepted



4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accepting states denoted by ◎

Example 1.11

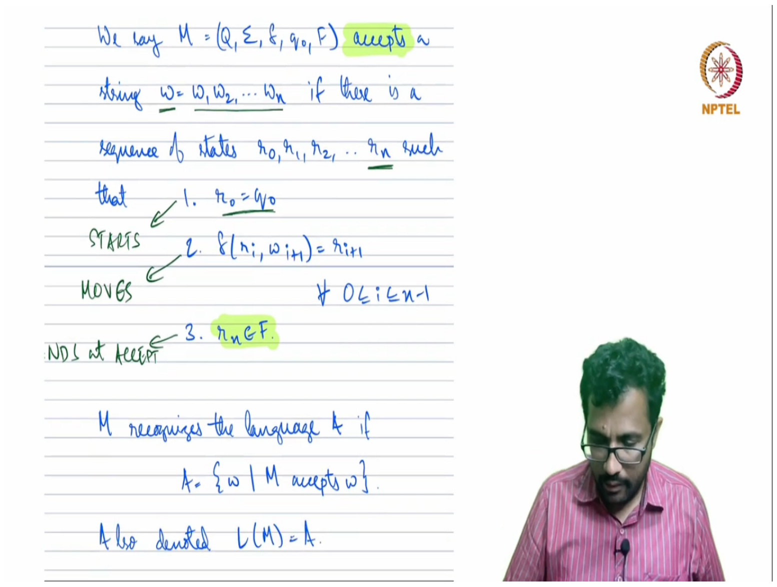
ε is not accepted

10 ✓
11 ✗
01 ✓
10 ✗

$Q = \{s, q_1, q_2, q_3, q_4\}$

And in some cases, some of these things could be combined, like over here we have the outgoing arrow for 0 and 1 both come back to q_2. So, δ is a transition function q_0 is a starting state which is only a single state and F is the set of accepting states which are denoted a double circle in the state diagram.

(Refer Slide Time: 01:47)



We say $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \dots w_n$ if there is a sequence of states $r_0, r_1, r_2, \dots r_n$ such that

STARTS 1. $r_0 = q_0$

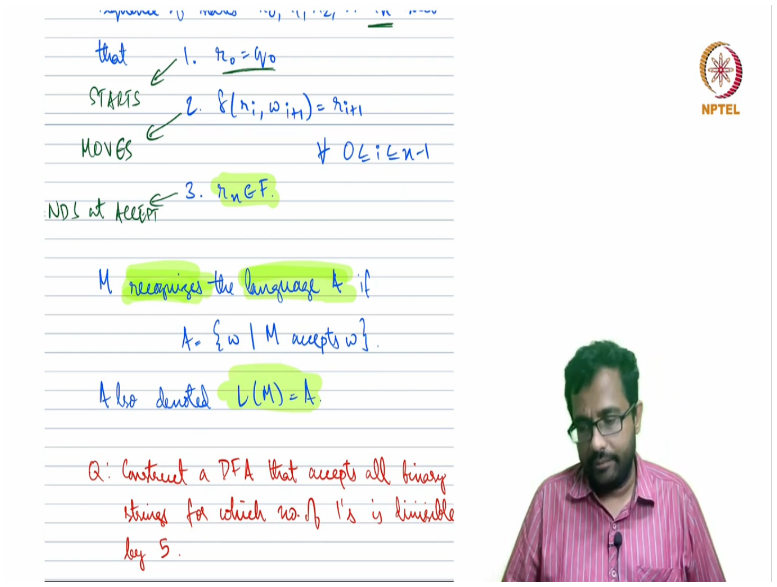
MOVES 2. $\delta(r_i, w_{i+1}) = r_{i+1}$ $\forall\ 0 \le i \le n-1$

ENDS at ACCEPT 3. $r_n \in F$.

M recognizes the language A if

$A = \{w \mid M \text{ accepts } w\}$.

Also denoted $L(M) = A$.



that 1. $r_0 = q_0$

STARTS 2. $\delta(r_i, w_{i+1}) = r_{i+1}$

MOVES $\forall\ 0 \le i \le n-1$

ENDS at ACCEPT 3. $r_n \in F$.

M recognizes the language A if

$A = \{w \mid M \text{ accepts } w\}$.

Also denoted $L(M) = A$.

Q: Construct a DFA that accepts all binary strings for which no. of 1's is divisible by 5.

Now, let us formally define when a string is accepted. We have already understood this and it is just a formalization for the same. So, we say that a machine or a DFA M accepts a string w. So, w is a string, and w_1 and w_2 are the individual symbols in it. So, let us say if w is 1011, then it is like w_1 is 1, w_2 is 0, w_3 is 1 and w_4 is 1. So, these are the four individual symbols. So, this is just w broken up into symbols.

So, we say that a machine M or a DFA M accepts a string if this string upon moving as per the string, you end at an accepting state. So, in other words, basically, there is a sequence of states such that r_0 up to r_n with r_0 as the state before you read anything and r_0 has to be the starting state. And the first symbol w_1 will take Q from r_0 to r_1. And w_2 will take you from r_1 to r_2. So, this is just formalized below.

The i+1 symbol will take you from r_i to r_{i+1}, where the i+1 goes from 1 to n. or in other words, i goes from 0 to n-1. And finally, you ended the state r_n which has to be an accepting state which is what is listed here. So, in other words, what is going on is the machine starts at the starting state and moves as per the rules and ends at an accepting state. So, it is fairly simple. It is just a formalization of whatever we have seen. So, we say a DFA accepts a string if at the end of processing that string, it ends at an accepting state.

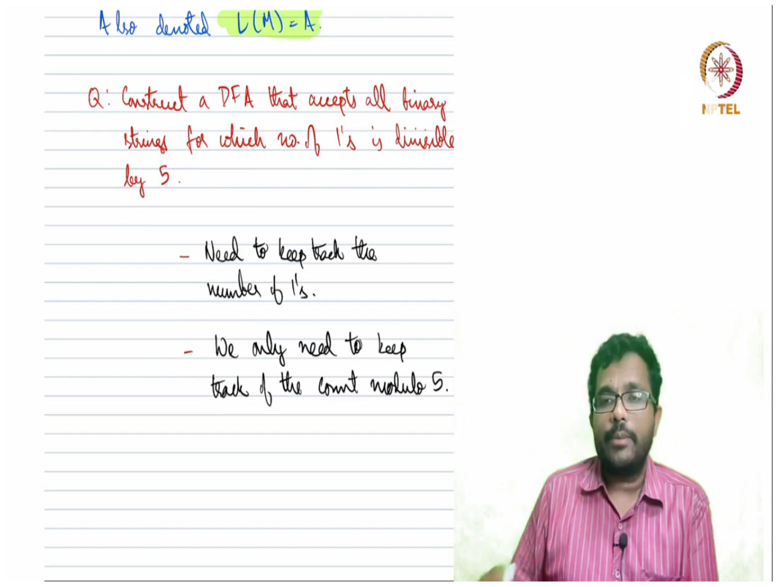
And what do I mean by processing? You start at the starting state and then each symbol you move, you make the correct move. And we say M accepts a string and correspondingly we say the machine or the DFA M recognizes the language A, where A is the set of all strings that are accepted by the DFA M. So, I have been kind of alternatively using machine and DFA, but I will continue to do that. So, think of machine DFA as the same computing device.

So, the distinction here is that M accepts a string and M recognizes a language. Sometimes, people say M accepts the language but that can lead to a bit of confusion. So, that is why we are trying to use M recognizes the language A. So, M recognizes the language means A is a set of all strings that are accepted by M. It is exactly the set of all strings.

So, A is not a super set of the strings or it is not that A is a subset of the strings. A contains certain strings that are accepted by M. If such a string is not accepted, it should not be in A. It is exactly the set of strings that are accepted, not a subset or a not a super set. Sometimes, we also denoted by this notation

$$L(M) = A$$

(Refer Slide Time: 05:41)



Also denoted $L(M) = A$.

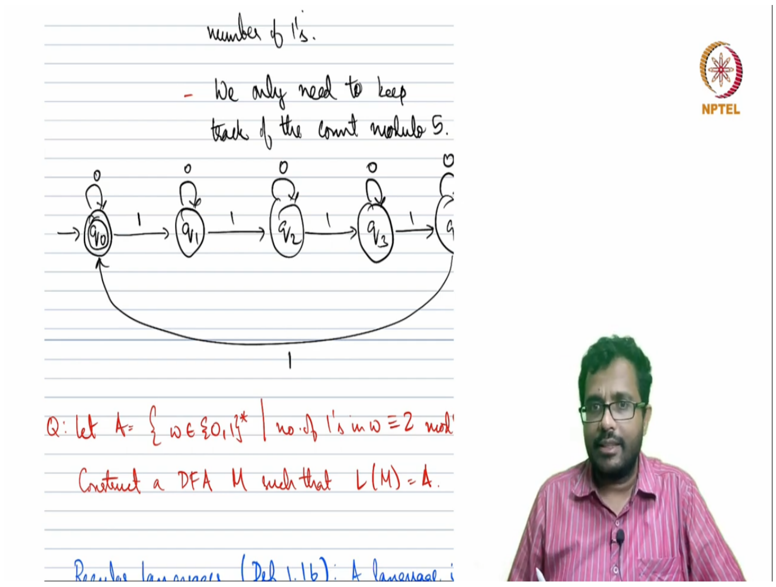
Q: Construct a DFA that accepts all binary strings for which no. of 1's is divisible by 5.

- Need to keep track the number of 1's.
- We only need to keep track of the count modulo 5.

So, now, let us see another example. Let us try to construct a DFA that accepts all the binary strings which have the number of 1s in those binary strings are divisible by 5. So, if you count the number of 1s it should be divisible by 5. So, let us try to see how to make use of the states. So, recall we have only a finite number of states. We only have a finite number of states and we want to use that. So, here the DFA needs to keep track of the number of 1s, but we do not need to keep track of the number of zeros, that is okay.

But then there is a problem like, let us say, if the string is huge and it has 100 1s. So, you need to keep track of the number 100. So, number 100 has to be divisible by 5, i.e. is a multiple of 5. So, do you keep counting? If so, is there a limit? So, how do you keep track of that much? The answer to that is we do not need to keep track of the exact count, we only need to keep track of the Count % 5. So, we do not need to keep track of the exact count and we only need to keep track of the Count % 5. In other words, the remainder of the count when divided by 5.

(Refer Slide Time: 07:44)



number of 1's.

- We only need to keep track of the count modulo 5.

Q: Let $A = \{ w \in \{0,1\}^* \mid$ no. of 1's in $w \equiv 2$ mod

Construct a DFA M such that $L(M) = A$.

Regular languages (Def 1.16): A language i

So, the key idea here is that we need to have a set of states and the thing we need to keep track should be the number of 1s. So, basically, we can have something like this, q_0, q_1, q_2, q_3, and q_4. So, try to keep track of the number of 1s that you have seen. So, at q_0, if you see a 0, you remain there. If you see a 1, you move to q_1. So, which means you have seen 11. If you see a 0, you remain here. If you see a 1 here and the same thing you keep repeating. And what happens when you see a 1 at q_4?

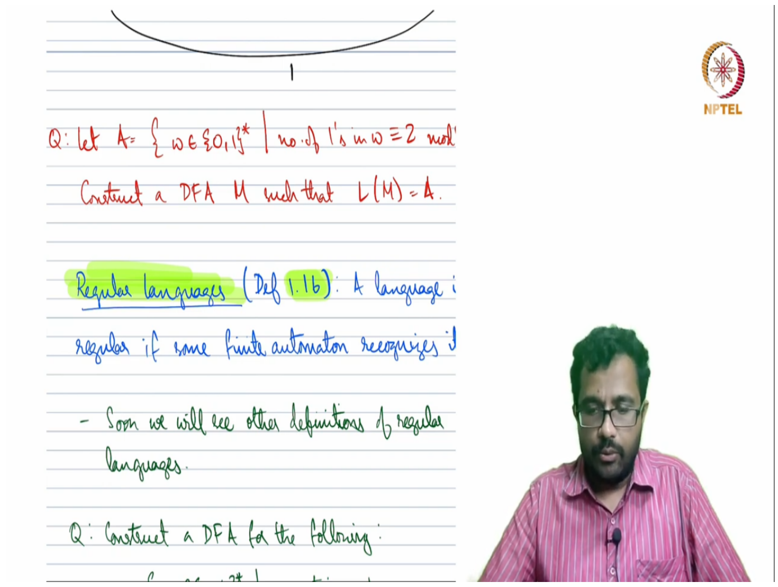
So, which means again it is like we have seen five 1s. So, now it is like we just need to keep track of the count of 1s % 5. So, if you see 5 1s, it is the same as 0 1s and so you can come back to q_0. Now, I have five states and for each state I have two outgoing arrows. The thing that is missing is which are the strings that are accepted. So, as the picture is currently no string is accepted. So, the strings that have to be accepted are those strings in which the number of 1s is divisible by 5 or is a multiple of 5.

So, let us just try some examples. So, let us just think of the string 11111. So, you start at q_0 and end at q_0. So, this has to be accepted. So, let us mark q_0 as an accepting state. And what if the string is 111110. So, it is the same movement and then you remain at q_0. So, this is also getting accepted. And now you can kind of play around with this and see that any string that is desired indeed gets accepted. So, this is the way to construct a DFA.

So, basically, we need to figure out what is the information that we are trying to preserve. Because we only have a finite set of states and we need to keep track of what is the information that we need to keep track of. So, I have a question here in red color, the set of all binary strings where the number of 1s is 2 % 5. So, here the number of 1s was a multiple of 5. So, now, we are saying 2 % 5. What is the DFA that gives us this language?

So, if you think about it, the same kind of counting mechanism still works, but the accepting state may have to change. So, maybe I won't give away the answer now, but you can think. Can we use the same structure but change something small, just think about it.

(Refer Slide Time: 11:50)



Q: Let $A = \{ w \in \{0,1\}^* \mid$ no. of 1's in $w \equiv 2$ mod

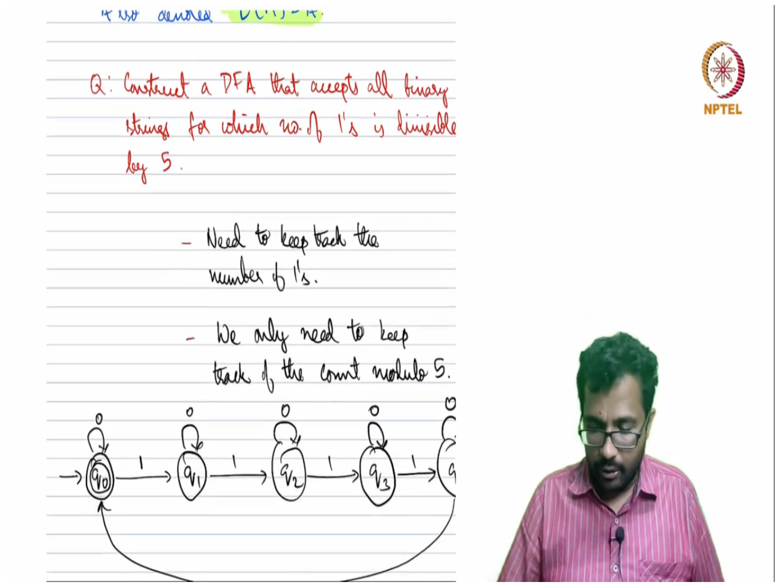
Construct a DFA M such that $L(M) = A$.

Regular languages (Def 1.16): A language i

regular if some finite automaton recognizes i

- Soon we will see other definitions of regular languages.

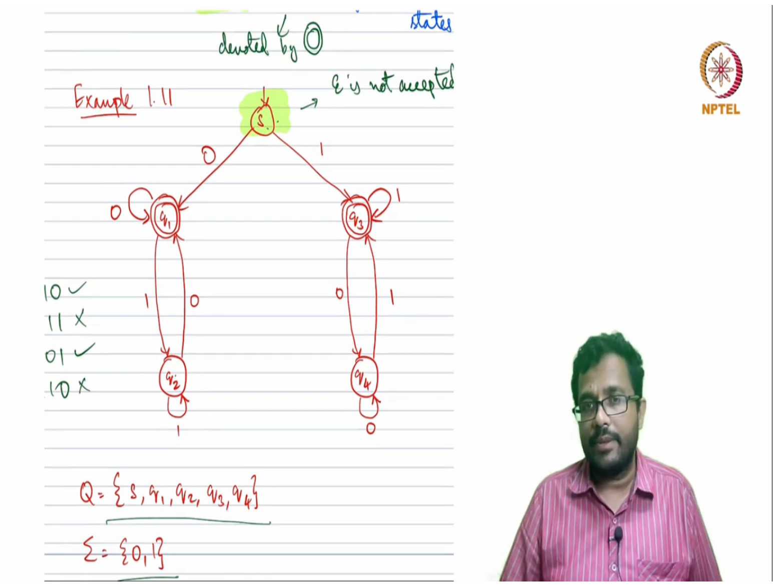
Q: Construct a DFA for the following:

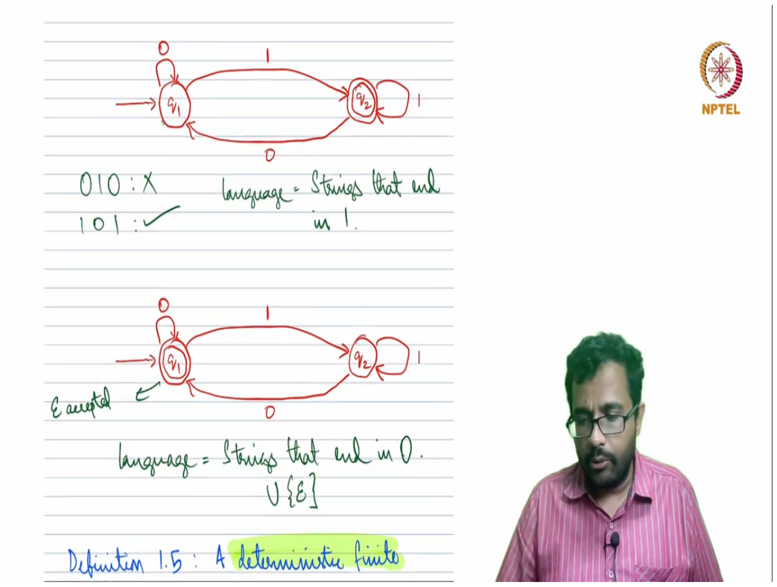


Q: Construct a DFA that accepts all binary strings for which no. of 1's is divisible by 5.

- Need to keep track the number of 1's.
- We only need to keep track of the count modulo 5.

$$\begin{tikzcd} \arrow[r] & q_0 \arrow[loop above, "0"] \arrow[r, "1"] & q_1 \arrow[loop above, "0"] \arrow[r, "1"] & q_2 \arrow[loop above, "0"] \arrow[r, "1"] & q_3 \arrow[loop above, "0"] \arrow[r, "1"] & q \arrow[loop above, "0"] \arrow[llll, bend left] \end{tikzcd}$$



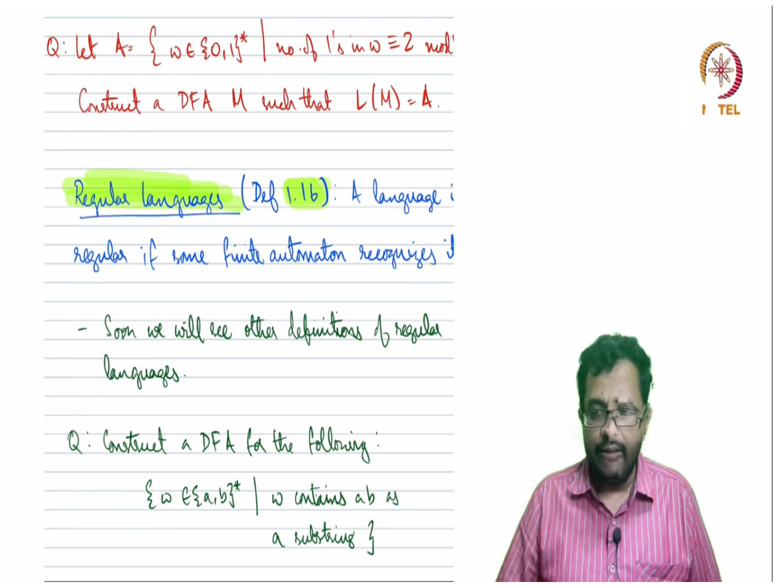

0
1
q1
q2
1
0
010 : ✗
101 : ✓
language = Strings that end in 1.
0
1
q1
q2
1
0
ε accepted
language = Strings that end in 0. ∪ {ε}
Definition 1.5 : A deterministic finite
NPTEL

And now we come to another definition. So, notice that I am using these numbers here like. So, this is the definition of regular languages, I am using these numbers. These are just the numbers that are there in my edition of Sipser which is the second edition, I think, maybe international edition.

But I am just using that to keep track. If you have a copy of the textbook by Michael Sipser that may not have the same numbering, I will try to keep it consistent within my this course. So, we say that language is regular if there is some DFA that recognizes this. So, this is a definition.

All the languages that we have seen so far: set of all strings that have where the number of 1s divisible 5 is a regular language because we drew a DFA for it. Also these languages: set of strings that start and end with the same symbol that is a regular language. Set of strings that end in zero is a regular language. Strings that end in 1, it is a regular language and so on.

(Refer Slide Time: 13:03)



Q: Let $A = \{ w \in \{0,1\}^* \mid \text{no. of 1's in } w \equiv 2 \text{ mod}$

Construct a DFA M such that $L(M) = A$.

Regular languages (Def 1.16): A language i

regular if some finite automaton recognizes i

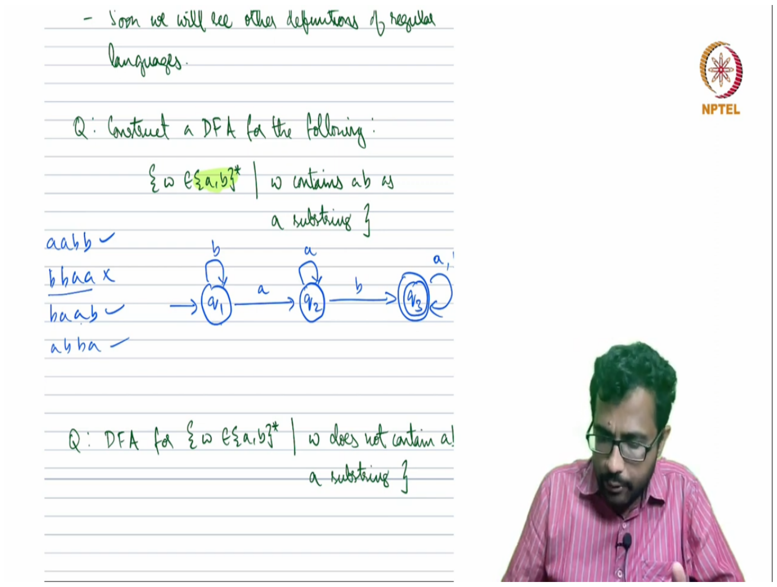
- Soon we will see other definitions of regular languages.

Q: Construct a DFA for the following:

$\{ w \in \{a,b\}^* \mid w \text{ contains ab as a substring} \}$

So, a regular language is a language if some finite automaton recognizes this.

(Refer Slide Time: 13:10)



- Soon we will see other definitions of regular languages.

Q: Construct a DFA for the following:

$\{ w \in \{a,b\}^* \mid w \text{ contains ab as a substring} \}$

aabb ✓

bbaa ✗

baab ✓

abba ✓

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

Q: DFA for $\{ w \in \{a,b\}^* \mid w \text{ does not contain a}$

a substring }

So, let us just see some more simple examples for construction. So, a DFA for the following: where the alphabet is {a, b} not {0, 1}, where the string contains ab as a substring. So, the input must contain ab as a substring. So, if at any point of the string ab appears so aabb must be accepted, bbaa must not be accepted, baab must be accepted because there is an ab here. and abba must also be accepted because there is an ab there.

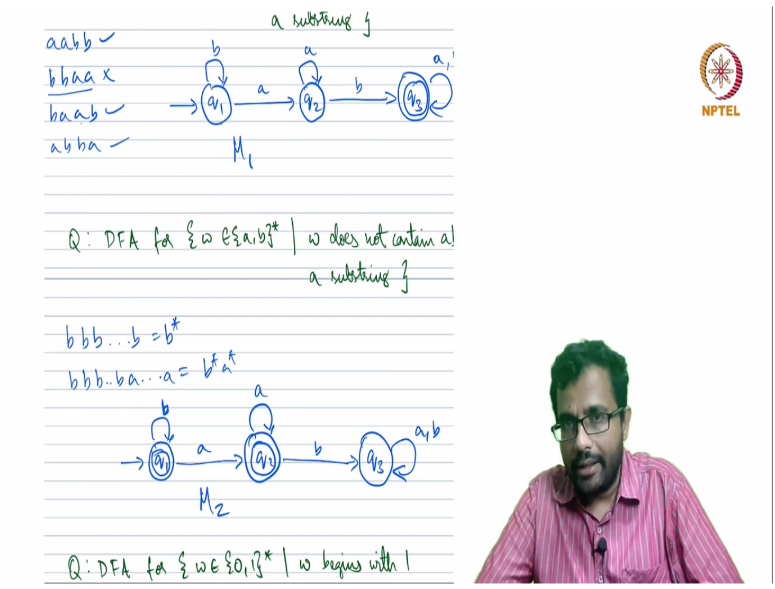
So, let us try to draw a DFA. We have a starting state. Now, to see if ab is a substring, we first have to get an a. So, if there is a b from the static position, we can remain in the same state. And if you see an a, you move to the next state. And after seeing an a, you could still keep seeing aaaaaaa. So, for instance like bbaa so that you may be here. So, but once you keep, once you see an a, now, you are halfway there. You have seen an a.

Now, you just need to, if either you keep seeing aaaaaaaa, or at some point ab will appear. So, whenever the first b appears that will result in an ab substring. So, when the first b appears you can accept, you can lead to an acceptance because now whatever comes after this, so, either you may see an a or you may see a b.

Now, whatever comes it does not matter. Now, you can accept it because you already seen ab as a substring. This DFA indeed captures all the strings that have ab as a substring. You can work out and see what happens. So, one exercise is what are the strings that you can try out at home is what are the strings that end in the first state.

What are the strings that end in the first or second or the third state? We know what are the strings that end in the third state. These are the strings that contain ab as a substring. Because it is a single accepting state. But try to see what are the strings that end in the first or the second.

(Refer Slide Time: 16:25)



a substring }

aabb ✓

bbaa ×

baab ✓

abba ✓

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

M_1

Q: DFA for $\{w \in \{a,b\}^* \mid$ w does not contain a
a substring }

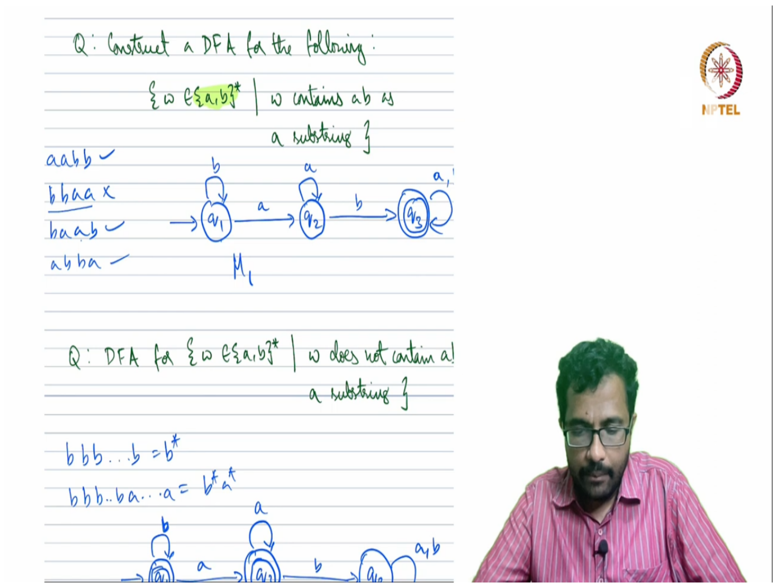
$bbb\ldots b = b^*$

$bbb..ba\ldots a = b^*a^*$

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

M_2

Q: DFA for $\{w \in \{0,1\}^* \mid$ w begins with 1



Q: Construct a DFA for the following:

$\{w \in \{a,b\}^* \mid$ w contains ab as
a substring }

aabb ✓

bbaa ×

baab ✓

abba ✓

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

M_1

Q: DFA for $\{w \in \{a,b\}^* \mid$ w does not contain a
a substring }

$bbb\ldots b = b^*$

$bbb..ba\ldots a = b^*a^*$

Now, let us try to look at another question which seems very related. Find DFA for the set of all strings where w does not contain ab as a substring. So, it is exactly the opposite of what we are seeing now. So, how can we not contain ab as a substring? So, as I said earlier, once you see an a you cannot have a b later. Because, once you see an a, either you keep seeing aaaaa or if you see a b you are then you have a substring ab.

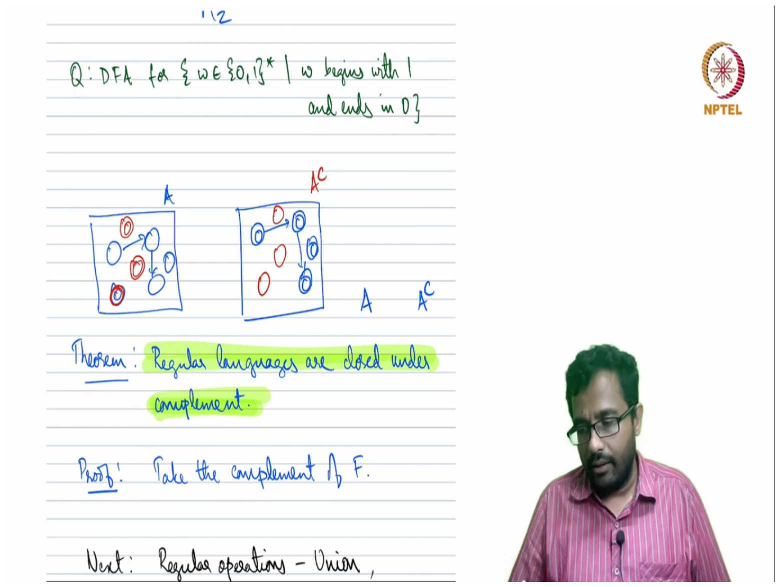
So, this is not possible. So, once you see an a you have to only be seeing a’s. But you could see b’s prior to the a’s, but you cannot have a’s is prior to the b’s because then again it is an ab. So, the only two things possible are strings of the form bbbbbbb like a full peace or in other words it is like b* or it could be of the form some number of b’s followed by some number of a’s. It is like b*a*.

So, these are the only set of strings that should be accepted and if you notice which is kind of what we did over here. So, you have a state q_1 where till you keep seeing b's you are stuck here. If you see an a, you move to q_2 and then you remain there if you see an a. If you see b again, you move to q_3 and then you are stuck at q_3. So, q3 is a set of strings where ab features as a substring. So, q_2 is where you have seen at least one a and q_1 is where you have only seen b's. So, all b* end in q_1.

Once you see an a, you go to q_2. Now you can check that this DFA indeed accepts all the strings that do not contain ab as a substring. And now you can see one interesting thing. Though I reasoned it slightly differently, these two DFA's, the one over here and the one over here, they are kind of the same DFA. The transitions are exactly the same. It is just that their accepting states are different.

So, in fact, the set of accepting states of the second DFA say M_2 and accept states of previous DFA M_1 are the opposite! So, M_1 and M_2 are kind of complementary in that sense. And the language also interestingly what we were looking for is the complement. So, here in the case of M_1, we were looking for all the strings that contain ab as a substring. M_2 recognizes all the strings that do not contain ab as a substring. So, M_1 and M_2 are complementary.

(Refer Slide Time: 20:27)



'12

Q: DFA for $\{ w \in \{0,1\}^* \mid w$ begins with 1 and ends in 0$\}$

Theorem: Regular languages are closed under complement.

Proof: Take the complement of F.

Next: Regular operations - Union,

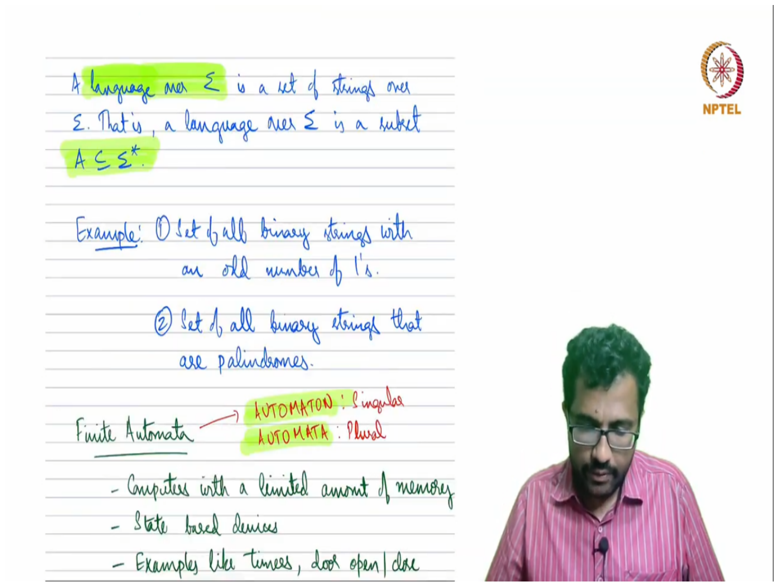
And this gives us the following statement that regular languages are closed under complement. Because, you can have the same construction of the DFA just that whatever is accepted by the, let us say you. So, what does the statement mean? Suppose there is a language A which is regular then the language A^C which is all the strings that are not in A that is also regular. This is what it means.

So, A is regular if and only if A^C is regular. Why is this true? Suppose, A is regular then there is a finite automaton for A. So, maybe some states, some transitions,. Now, this is the machine that accepts A.

Now, to get A^C, you could have exactly the same thing but with complementary accepting states. So, something like this is, it is not a, it is only a pictorial thing. So, all the blues are accepting and all the reds are not accepting. So, this will be A^C. So, take any regular language, there is a DFA for it.

Now, you take the same DFA, you make accepting as not accepting and you take the not accepting as accept states. So, you take the set of accepting states and you take the complement of that. That will give you the DFA for A^C. This is what it means to say, regular languages are closed under complement if A is regular, then A^C is also regular. I have one more question here that you can try: the set of all binary strings where this string begins with one and ends in zero. So, try to construct this as well.

(Refer Slide Time: 23:26)



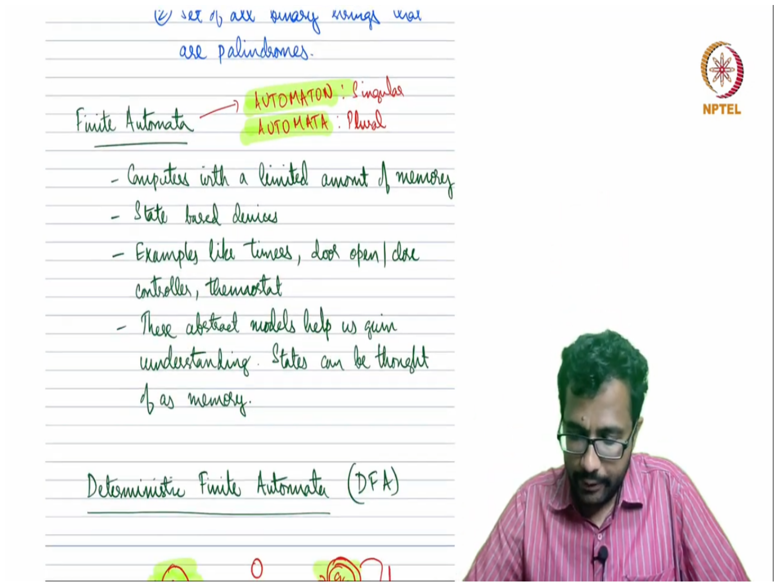
A language over Σ is a set of strings over Σ. That is, a language over Σ is a subset $A \subseteq \Sigma^*$.

Example: ① Set of all binary strings with an odd number of 1's.

② Set of all binary strings that are palindromes.

Finite Automata → AUTOMATON : Singular; AUTOMATA : Plural

- Computers with a limited amount of memory
- State based devices
- Examples like timers, door open/close

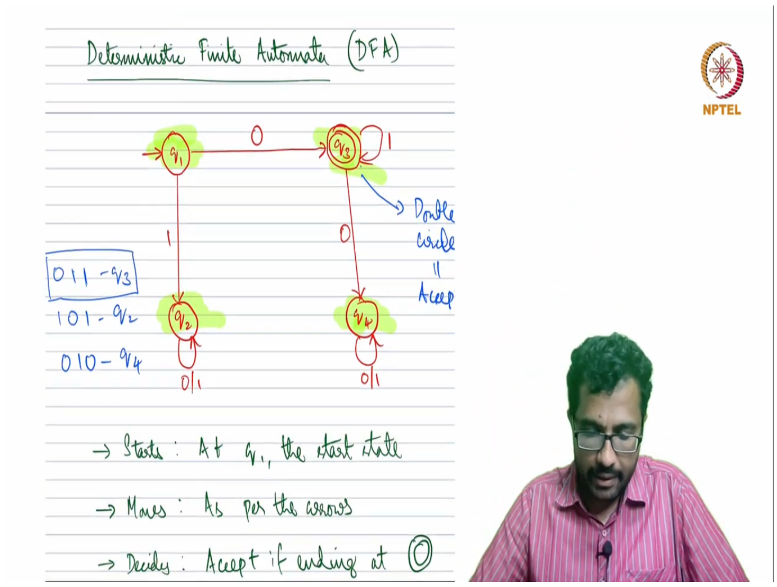


② Set of all binary strings that are palindromes.

Finite Automata → AUTOMATON : Singular; AUTOMATA : Plural

- Computers with a limited amount of memory
- State based devices
- Examples like timers, door open/close
controller, thermostat
- These abstract models help us gain understanding. States can be thought of as memory.

Deterministic Finite Automata (DFA)



Deterministic Finite Automata (DFA)

$$\begin{tikzcd}
\arrow[r] & q_1 \arrow[r, "0"] \arrow[d, "1"] & q_3 \arrow[loop right, "1"] \arrow[d, "0"] \\
& q_2 \arrow[loop below, "0|1"] & q_4 \arrow[loop below, "0|1"]
\end{tikzcd}$$

Double circle = Accep

$\boxed{011 - q_3}$

$101 - q_2$

$010 - q_4$

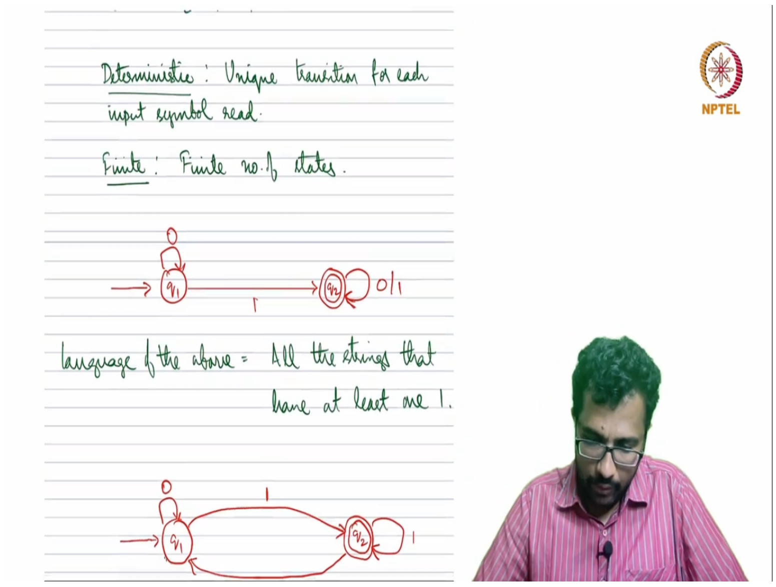
→ Starts : At q_1, the start state

→ Moves : As per the arrows

→ Decides : Accept if ending at ◎

So, I will just end this lecture here. So, I will just summarize what we have seen so far. So, we saw what is a finite automata, we saw some examples of a deterministic finite automata or DFA's as we called it.

(Refer Slide Time: 23:37)



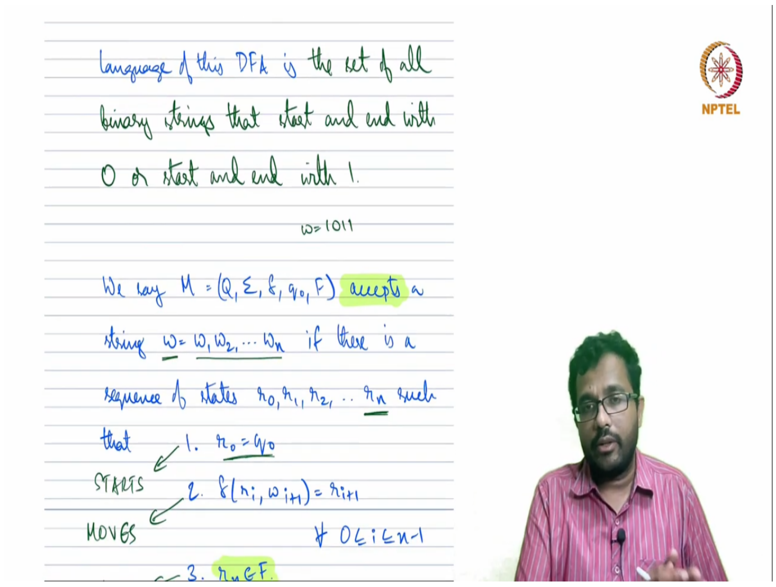
Deterministic : Unique transition for each input symbol read.

Finite : Finite no. of states.

$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "0"] \arrow[rr, "1"'] & & \circledcirc q_2 \arrow[loop right, "0|1"] \end{tikzcd}$$

language of the above = All the strings that have at least one 1.

$$\begin{tikzcd} \arrow[r] & q_1 \arrow[loop above, "0"] \arrow[rr, bend left, "1"] & & \circledcirc q_2 \arrow[loop right, "1"] \arrow[ll, bend left] \end{tikzcd}$$



language of this DFA is the set of all binary strings that start and end with 0 or start and end with 1.

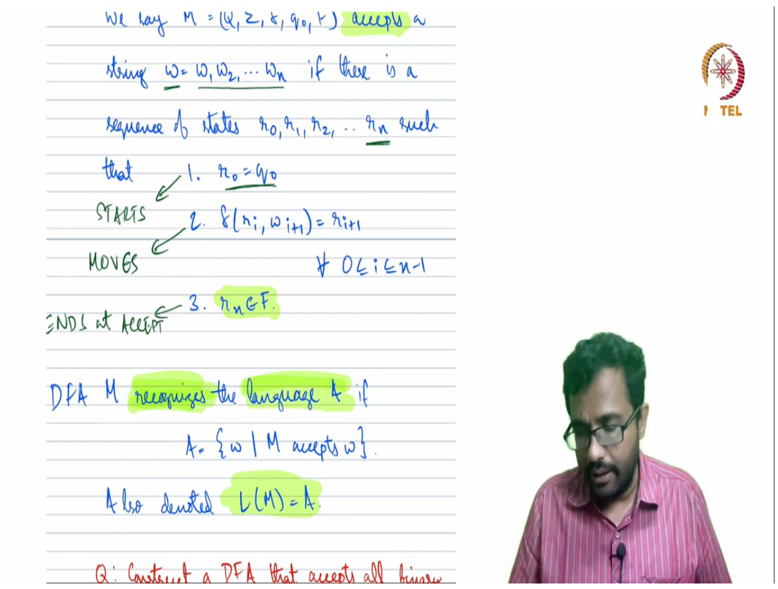
$w = 1011$

We say $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $\underline{w} = \underline{w_1 w_2, \dots w_n}$ if there is a sequence of states $r_0, r_1, r_2, .. \underline{r_n}$ such that

1. $\underline{r_0 = q_0}$ ← STARTS
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ ← MOVES $\forall\, 0 \leq i \leq n-1$
3. $r_n \in F.$

It is deterministic because each state has exactly one outgoing arrow for each symbol that we see over there and it is finite because there are only a finite number of states. So, the language of a DFA is the set of strings that are accepted by the DFA. And we formally saw the definition. The definition is a five tuple with the number of states. The states alphabet transition function starting state and accepting states. We saw some examples about some DFA's and which language they accept. We formally define what it means for a machine to or a DFA to accept a string, even though we had informally seen that.

(Refer Slide Time: 24:26)



We say $M = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w = w_1 w_2 \ldots w_n$ if there is a sequence of states $r_0, r_1, r_2, \ldots r_n$ such that

STARTS ← 1. $r_0 = q_0$

MOVES ← 2. $\delta(r_i, w_{i+1}) = r_{i+1}$ $\forall\ 0 \le i \le n-1$

ENDS at ACCEPT ← 3. $r_n \in F$.

DFA M recognizes the language A if

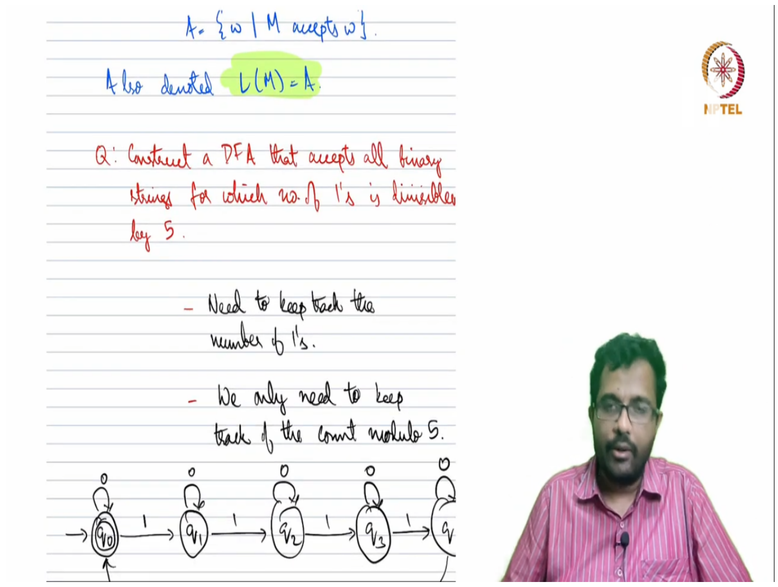
$A = \{w \mid M \text{ accepts } w\}$.

Also denoted $L(M) = A$.

Q: Construct a DFA that accepts all binary

We say that DFA M recognizes the language A if A is a set of all strings that are accepted by M.

(Refer Slide Time: 24:44)



$A = \{w \mid M \text{ accepts } w\}$.

Also denoted $L(M) = A$.

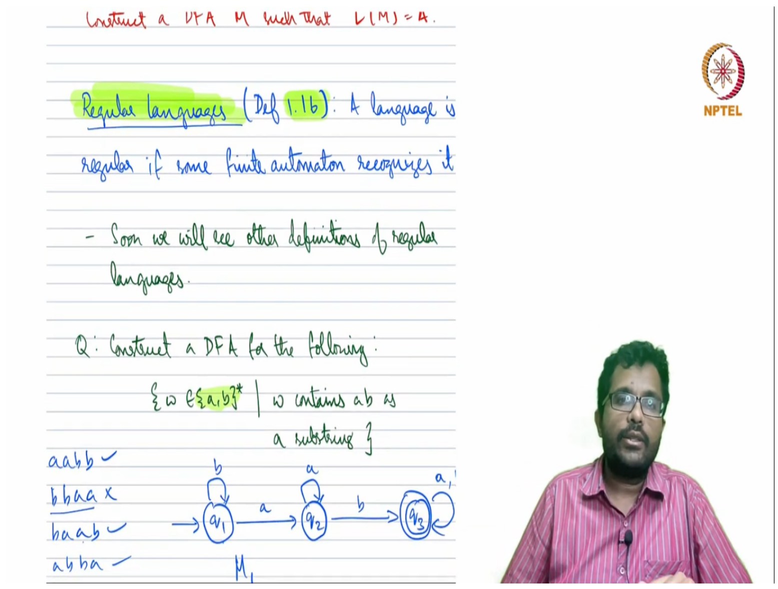
Q: Construct a DFA that accepts all binary strings for which no. of 1's is divisible by 5.

- Need to keep track the number of 1's.
- We only need to keep track of the count modulo 5.

$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[loop above, "0"] \arrow[r, "1"] & q_1 \arrow[loop above, "0"] \arrow[r, "1"] & q_2 \arrow[loop above, "0"] \arrow[r, "1"] & q_3 \arrow[loop above, "0"] \arrow[r, "1"] & q_4 \arrow[loop above, "0"]
\end{tikzcd}$$

And we saw some languages for questions of this form where we are given a certain language and we wanted to create a DFA for it.

(Refer Slide Time: 24:53)



Construct a DFA M such that L(M) = A.

Regular languages (Def 1.16): A language is regular if some finite automaton recognizes it

- Soon we will see other definitions of regular languages.

Q: Construct a DFA for the following:

$\{w \in \{a,b\}^* \mid w$ contains ab as a substring$\}$

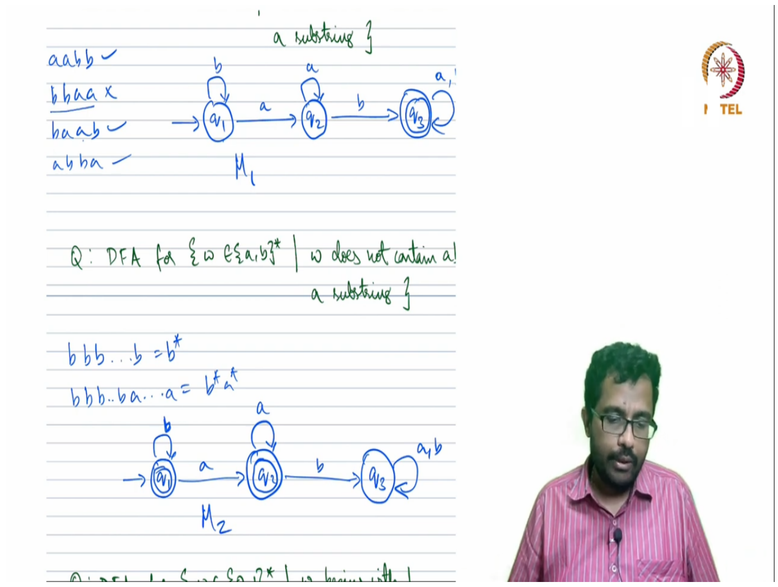
aabb ✓
bbaa ✗
baab ✓
abba ✓

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

M_1

And we defined regular languages which are the set of all languages for which we can construct a DFA.

(Refer Slide Time: 25:04)



a substring}

aabb ✓
bbaa ✗
baab ✓
abba ✓

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

M_1

Q: DFA for $\{w \in \{a,b\}^* \mid w$ does not contain ab as a substring$\}$

$bbb\ldots b = b^*$

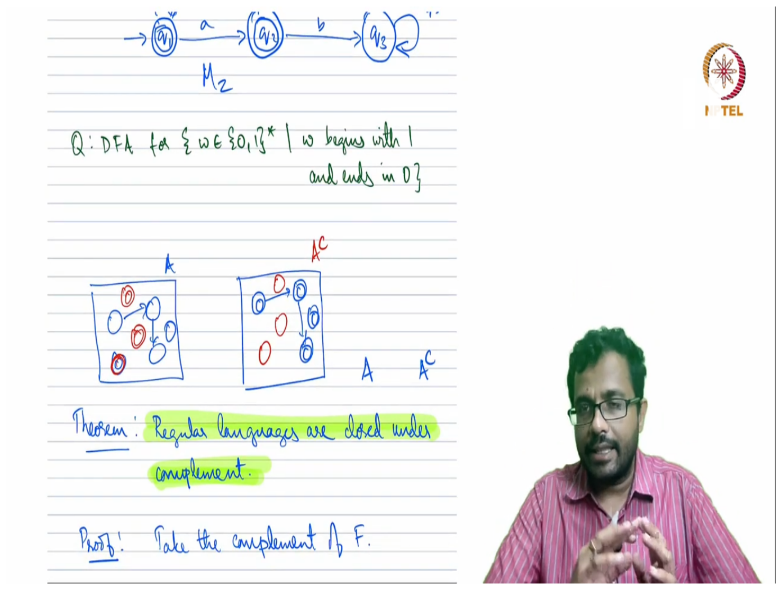
$bbb..ba\ldots a = b^* a^*$

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

M_2

So, one may ask at this point, why do I make a definition like this? It may feel to you, it may seem to you that for anything you can make a DFA. All languages you may be able to make a DFA but that is not the case. At some point, we will see why that is the case. Maybe not at some point but very soon.

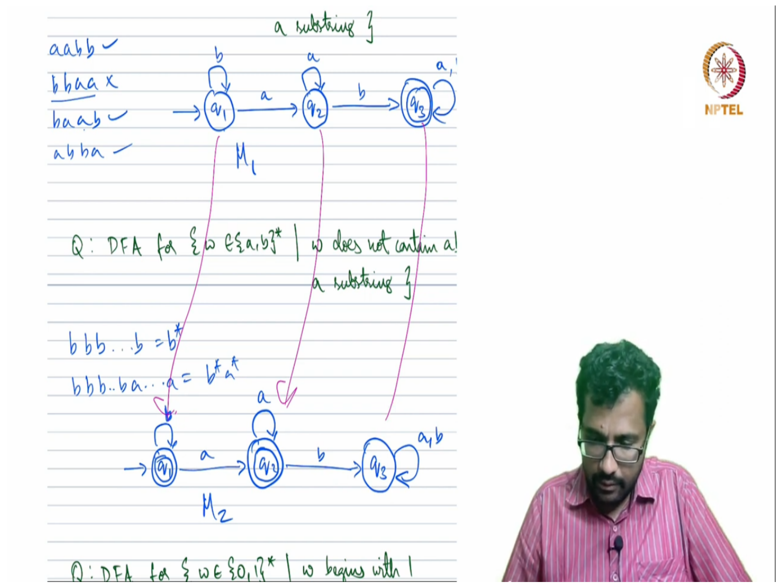
(Refer Slide Time: 25:28)



Q: DFA for $\{w \in \{0,1\}^* \mid w$ begins with 1 and ends in 0$\}$

Theorem: Regular languages are closed under complement.

Proof: Take the complement of F.



Q: DFA for $\{w \in \{a,b\}^* \mid w$ does not contain ab as a substring$\}$

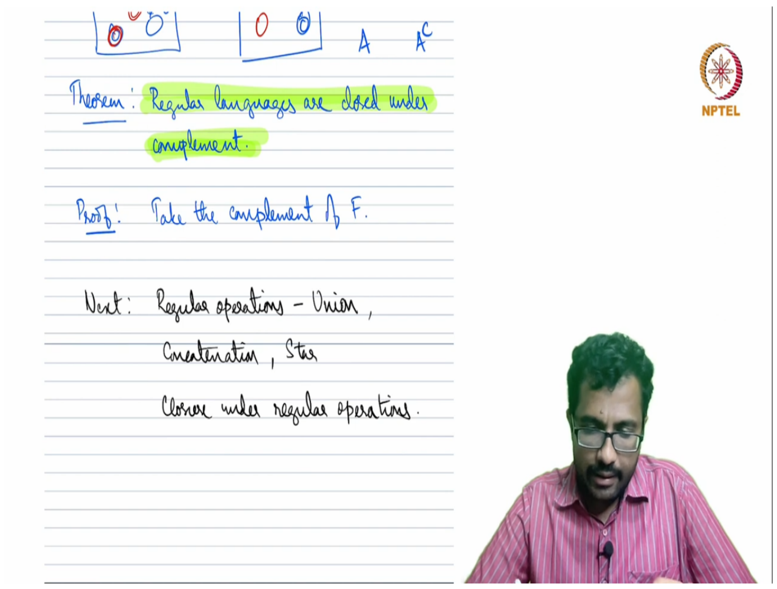
$bbb\ldots b = b^*$

$bbb..ba\ldots a = b^* a^*$

Q: DFA for $\{w \in \{0,1\}^* \mid w$ begins with 1

So, these are a special class of languages for which we can make DFA’s. Not all languages for, we cannot make DFA’s for all languages. And then we said that if A is the regular, meaning if we can get it, if we can draw a DFA for A, we can also draw a DFA for A^C. In fact, it is the same DFA but just that we change the accepting state to non-accepting and non-accepting state to accepting, just like what we did over here. So, accepting to non-accepting and non-accepting to accepting. So, that is a summary of today's lecture.

(Refer Slide Time: 26:23)



A A^c

Theorem: Regular languages are closed under complement.

Proof: Take the complement of F.

Next: Regular operations - Union, Concatenation, Star

Closure under regular operations.

And in the next lecture, it will be lecture 4, we are going to see regular operations. So, these are union, concatenation and star operations. And we are going to also, after that we are going to explore closure under regular operations.

So, we want to say things like if A is regular and B is regular and then A union B is regular. Meaning that, if there is a DFA for A and a DFA for B, we can build a DFA for the union of A and B and similar statements for concatenation and star. So, this is all for the lecture on regular languages and see you in the next lecture.