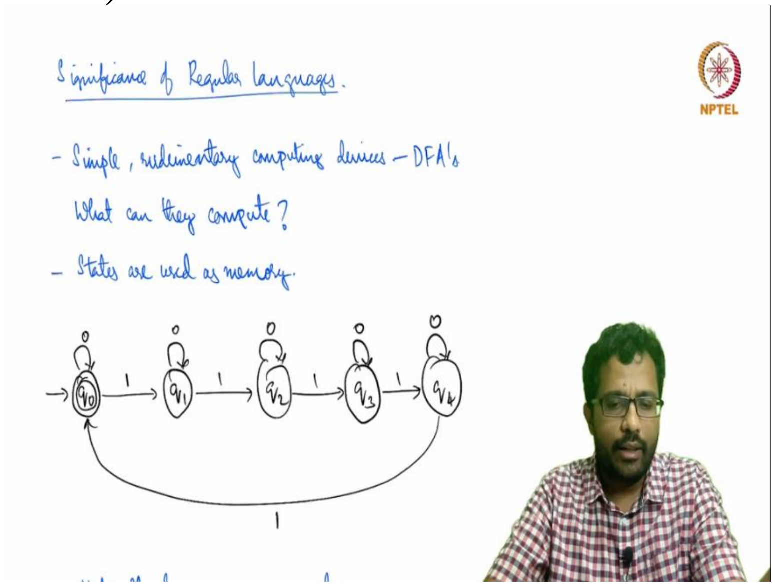
Theory of Computation

Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Lecture 4
Significance of Regular Languages and Regular Operations

(Refer Slide Time: 00:15)



Significance of Regular languages.

- Simple, rudimentary computing devices – DFA's
  What can they compute?
- States are used as memory.

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

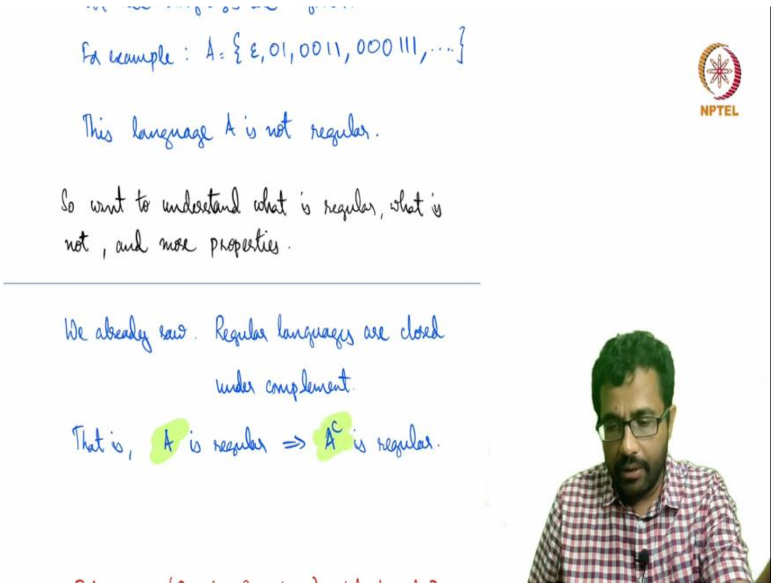
Hello and welcome to the lecture 4 of the course Theory of Computation. So, this is going to be a short lecture. It is mostly going to be about the Significance of Regular Languages. In lecture 3, we saw what deterministic finite automata were, DFA's. And then we defined regular languages as those languages that can be recognized by DFA's.

So, in this lecture, we are just going to explain why they are important. As we saw, DFA's are fairly simple, rudimentary computing devices. So, you have a state control, a machine which reads the input and then it has some states, some of which are designated as accepting. After you read the input, if you are in an accepting state, you accept the input. If you are in a non-accepting state, you do not accept the input. Set of all strings that are accepted by a DFA, constitutes the language of the DFA.

Further, for a specific language if there is a DFA that recognizes that language we say that that language is regular. So, regular languages are a way to characterize the power of DFA's. We want to understand what DFA's can do.

So, one possibility is that when you see DFA's and when you see the different ways you can put them together, one may think that perhaps, for any kind of language we can construct a DFA. So, we want to know what kind of languages can be recognized by DFA's and what kind of languages cannot be recognized by DFA's. So, towards understanding that we will define the regular languages and then we will see some properties.

(Refer Slide Time: 02:20)



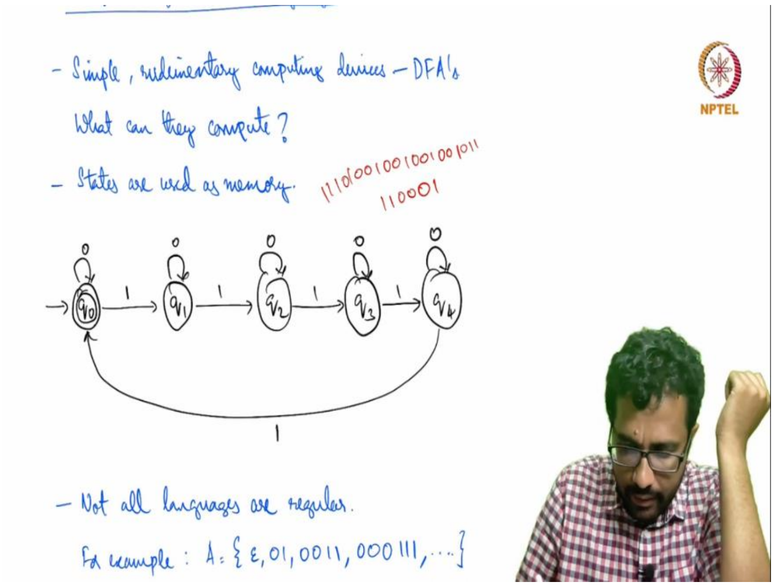
For example : $A = \{ \varepsilon, 01, 0011, 000111, \ldots \}$

This language A is not regular.

So want to understand what is regular, what is not, and more properties.

We already saw. Regular languages are closed under complement.

That is, A is regular $\Rightarrow$ A^c is regular.

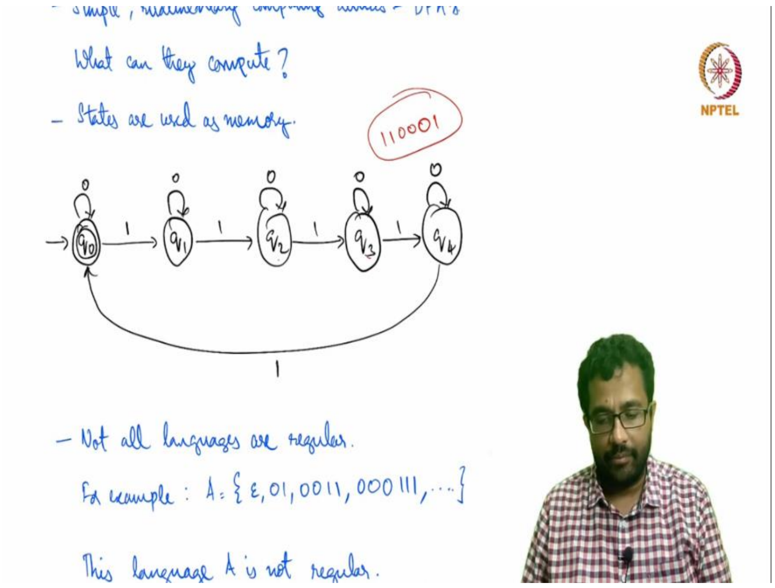


- Simple, rudimentary computing devices – DFA's
  What can they compute?
- States are used as memory.

11101001001001001011
110001

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

- Not all languages are regular.
  For example : $A = \{ \varepsilon, 01, 0011, 000111, \ldots \}$


- Simple, rudimentary computing devices - DFA's

What can they compute?

- States are used as memory.

- Not all languages are regular.

For example : $A = \{\varepsilon, 01, 0011, 000111, \ldots\}$

This language A is not regular.

So, for instance, we saw that regular languages are closed under complement. Meaning, if a language is regular, the complement language. If A is regular, then A^C is also regular. What are the specialties of regular languages? How are the DFA's actually computing, how are the deterministic finite automata computing?

So, for example, let us take this one DFA that we saw in the previous lecture. This accepts all the strings for which the number of 1s is a multiple of 5. So, if your string has let us say 5 1s, it accepts. For instance, this string is accepted because it has 5 1s. If it has 10 1s too, it will accept.

Now, it has 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10 1s. Even this is accepted. As long as the number of 1s contained by the string is a multiple of 5, it is accepted. Let us see what this machine is actually doing. q_0 is an accepting state. Meaning, after reading any string for which the number of 1s is a multiple of 5, you will end at q_0.

Let me erase the previous string and let us consider just this string (110001). Where does this string end up? The first one will take you to q_1, second one will take you to q_2, third one will take you to q_3 and it will remain there. So, q_3 is the state where all the strings, which have three 1s end up.

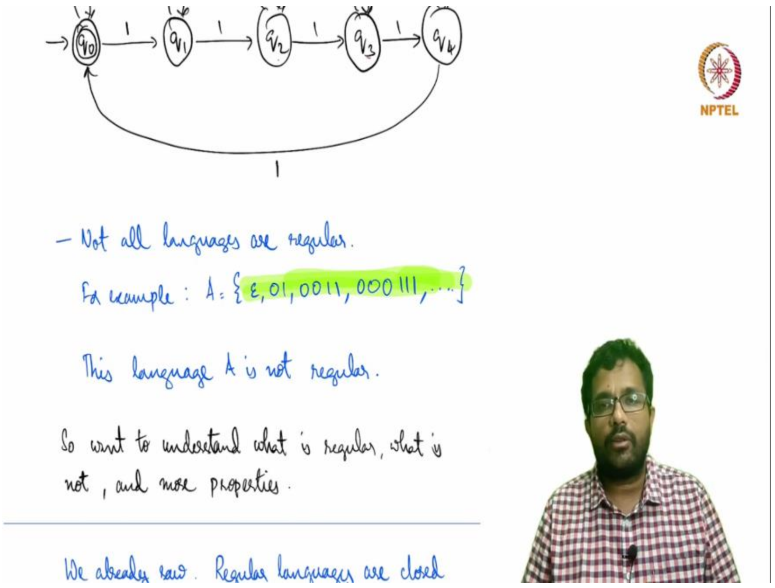
But q_3 is also the state where all the strings that have eight 1s end up. So, q_3 is the state where as long as you have three 1s, eight 1s, thirteen 1s. As long as the number of 1 leaves a remainder of three when you divide by 5, you end up at q_3. In other words, if the number of 1s is 3 modulo 5, you end up at q_3.

And similarly, if the number of 1s is 2 modulo 5, you end up at q_2 and so on. Similarly, for q_1 and q_4. So, what is happening here is these states q_0, q_1, q_2, q_3 and q_4 are being used to keep track of the count of the number of 1s with the modulo 5 condition. If you want to actually count the number of 1s, you need a lot of states, you need an infinite number of states.

But as long as you need to count only the remainder when you divide by 5, this machine would do. Here, what is happening is that the states are being used as the counter. So, in other words, the states are being used to the memory here. So, if you look at any DFA that you built, similarly, each state would be trying to have some role.

So, each state would be trying to identify a certain number of or certain subset of the strings that are processed by the DFA. This is how in the case of this particular DFA the states play a role. And similarly, if you take any DFA, you should be able to analyze and see which are the strings that end up at each state. This will give you a lot of insight about how the DFA'S operate.

(Refer Slide Time: 06:24)



$$\begin{tikzcd} \arrow[r] & q_0 \arrow[r, "1"] & q_1 \arrow[r, "1"] & q_2 \arrow[r, "1"] & q_3 \arrow[r, "1"] & q_4 \arrow[llll, bend left, "1"] \end{tikzcd}$$

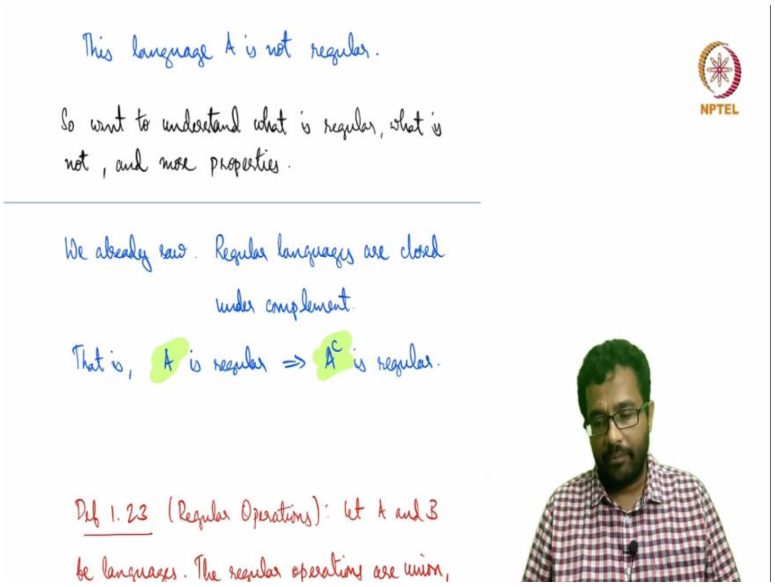
– Not all languages are regular.

For example : $A = \{\varepsilon, 01, 0011, 000111, \dots\}$

This language A is not regular.

So want to understand what is regular, what is not, and more properties.

We already saw. Regular languages are closed



This language A is not regular.

So want to understand what is regular, what is not, and more properties.

We already saw: Regular languages are closed under complement.

That is, A is regular $\Rightarrow$ A^c is regular.

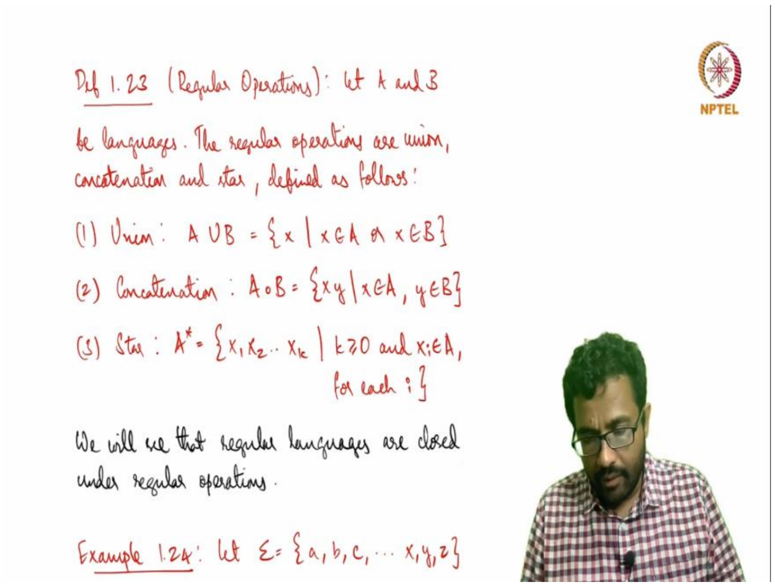
Def 1.23 (Regular Operations): Let A and B be languages. The regular operations are union,

And just as I mentioned earlier, not all the languages are regular. So, it is only a special subclass of languages that are regular. For instance, this language that is written over here. It is a fairly simple language. It is easy to describe. It first contains the empty string ε then the string 01. So, I am talking about this language. It first contains the empty string ε then the string 01, then the string 0011, then the string 000111, then the string 00001111 and so on. It is an infinite language but the pattern is very clear. It is easy to describe. I just told you what it is.

And basically, there are some number of zeros followed by the same number of 1s. This happens to be not a regular language. So, you can think, why it is not a regular language. Maybe it is even better, if you just try to construct a DFA for it and then you see why you are not able to do it. So, you will fail because I am telling you so, because it is not regular so you should not be able to construct a DFA.

So, you can try to construct a DFA that may give you some insights as to why this language is not regular. It is a fairly simple language. The point is we want to understand what is a regular, what is not regular and what are the structure of the languages. If a language is regular, what properties does it have? For instance, we already saw that regular languages are closed under complement.

(Refer Slide Time: 08:09)


Def 1.23 (Regular Operations): Let A and B be languages. The regular operations are union, concatenation and star, defined as follows:

(1) Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

(2) Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

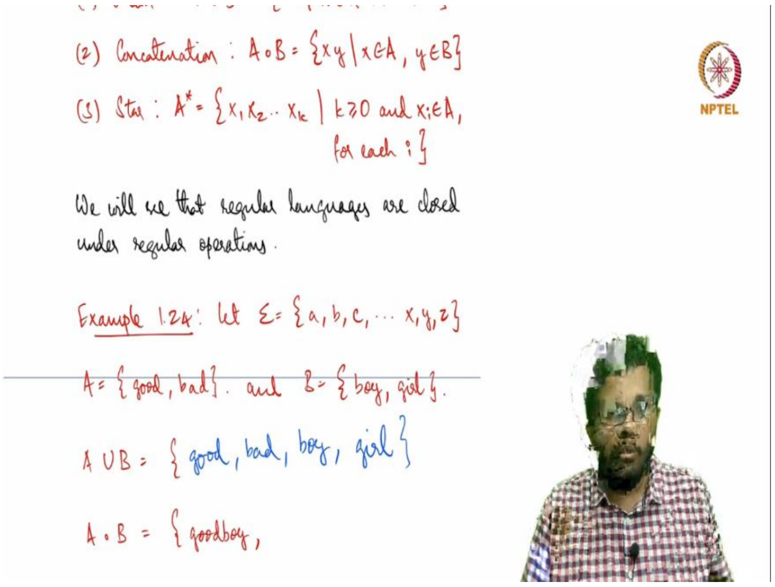
(3) Star: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i\}$

We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{a, b, c, \dots x, y, z\}$

In the next part, the next lecture, we will see regular operations and we will see closure under that. So, what are regular operations? Regular operations are fairly simple straightforward operations. So, three operations are defined. One is union which is just the set wise union. If A and B are two languages, the union of these two languages is a set wise union. So, the set of all strings that are either in A or in B.

(Refer Slide Time: 08:40)


(2) Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

(3) Star: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i\}$

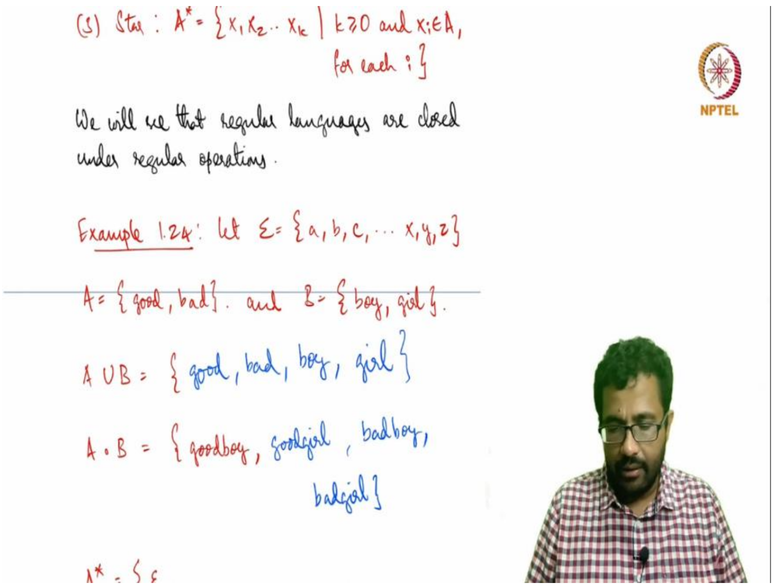
We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{a, b, c, \dots x, y, z\}$

A = {good, bad}. and B = {boy, girl}.

$A \cup B$ = {good, bad, boy, girl}

$A \circ B$ = {goodboy,



(3) Star : $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0$ and $x_i \in A$, for each $i\}$

We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{a, b, c, \ldots x, y, z\}$

$A = \{\text{good}, \text{bad}\}$. and $B = \{\text{boy}, \text{girl}\}$.

$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$

$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$

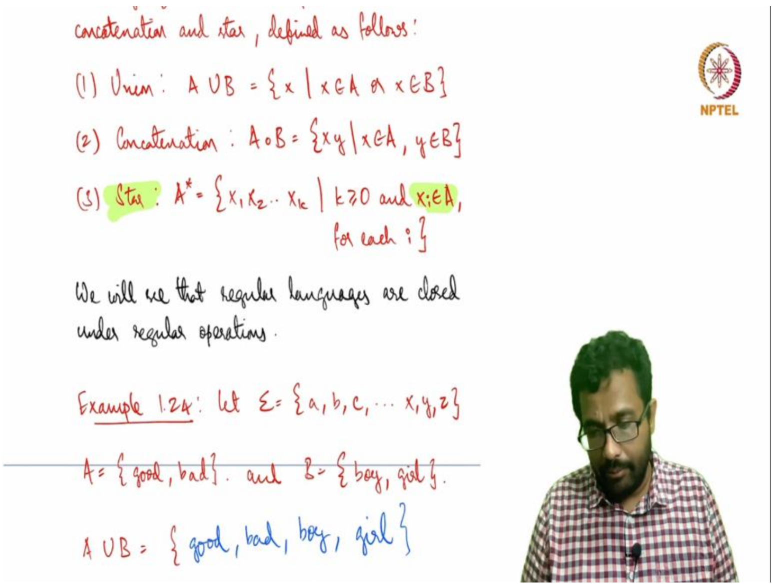
$A^* = \{\varepsilon$

Two, concatenation. Maybe it is easy to go with the example. Suppose A is the language $\{\text{good}, \text{bad}\}$. So, A is the language $\{\text{good}, \text{bad}\}$ and B is the language $\{\text{boy}, \text{girl}\}$. The alphabet Σ, is all the English alphabet. Then $A \cup B$ is simply $\{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$. These are the four strings that constitute the union language.

Basically, it is the set wise union. So, what is concatenation? Concatenation is the set of all strings that are formed by concatenating a string of A with a string of B in that order. Concatenation respects the order. A·B means you take a string from A and append a string of B to that. So, string of A followed by a string of B. So, A·B need not be the same as B·A because the orders are changed.

So, let us see what A·B is in this case. A is $\{\text{good}, \text{bad}\}$, B is $\{\text{boy}, \text{girl}\}$. So, A·B is $\{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$. These are the four strings in the concatenation language. So, good appended by boy, where good is from A, boy is from B. Similarly, good followed by girl where good is from A, girl is from B. Similarly, badboy and badgirl.

(Refer Slide Time: 10:45)


concatenation and star, defined as follows:

(1) Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

(2) Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

(3) Star: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i\}$

We will see that regular languages are closed under regular operations.

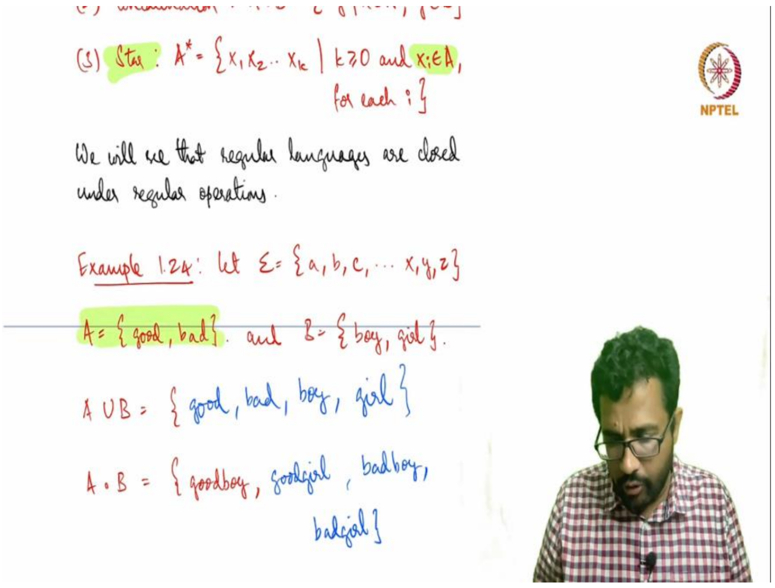
Example 1.24: Let $\Sigma = \{a, b, c, \ldots x, y, z\}$

$A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$.

$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$

And finally, the star operation is similar to the Kleene star that we defined earlier where we defined it for alphabets. Here, we are defining it for languages. A* where A is a language is basically x_1 x_2 … x_k concatenated together. It is k strings concatenated where each x_1, x_2, x_3, x_4, x_k, all of them belong to A. Each of them belongs to A. x_1 belongs to A, x_2 belongs to A, everything belongs to A. And k could be anything. k could be 0, k could be 1, k could be 2, k could be 3, it is okay. k could be any finite number. So, let us see.

(Refer Slide Time: 11:48)


(3) Star: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i\}$

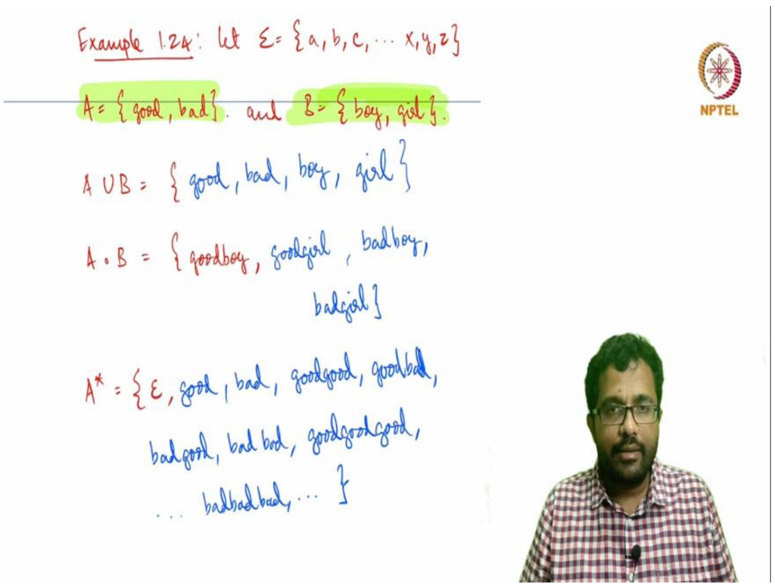
We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{a, b, c, \ldots x, y, z\}$

$A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$.

$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$

$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$



Example 1.24: let $\Sigma = \{a, b, c, \dots x, y, z\}$

$A = \{\text{good}, \text{bad}\}$. and $B = \{\text{boy}, \text{girl}\}$.

$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$

$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$

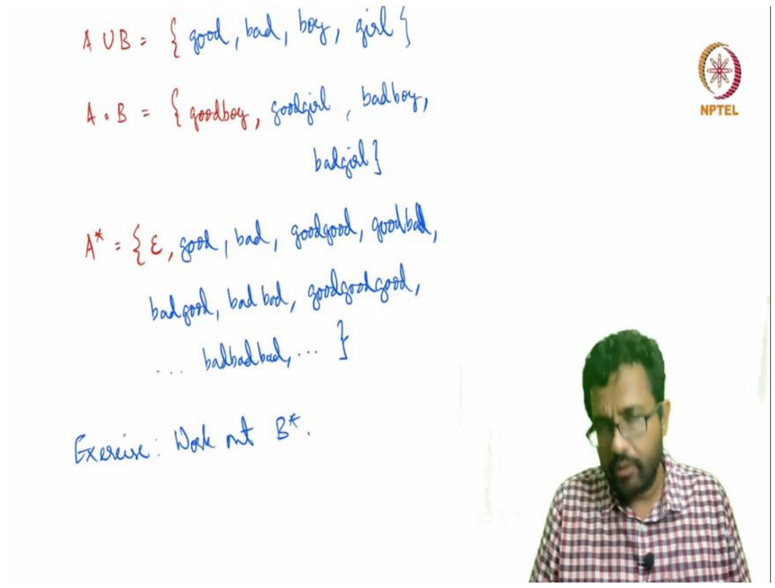
$A^* = \{\varepsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badgood}, \text{badbad}, \text{goodgoodgood}, \dots \text{badbadbad}, \dots\}$

For instance, when A is the language {good, bad}. A* contains the following. First of all k can be 0 also. So, k can be 0 meaning no string is there. Which means, it results in the empty string ε. Then k could be 1 where you just have one string, x_1 which belongs to A. So, the strings that belong to A are good and bad. And set of x_1 x_2 where k is equal to 2.

So, two strings where it belongs to A. It could be goodgood, it could be goodbad, it could be badgood, it could be badbad. These are the four possibilities of writing x_1 x_2 where x_1 comes from A followed by x_2 also coming from A. x_1 and x_2 need not be the same, they need not be different. They could be the same, they could be different. That is why there are four possibilities. Then let us go to the case when k is 3. So, when we have x_1, x_2, x_3 where each one of them comes from A. Let us start with goodgoodgood then followed by goodgoodbad and goodbadbad and so on. So, then at the end you will have badbadbad and so on.

So, again this is an infinite set because k could be 1, 2, 3, 4, 5, 6, it could be any number 100, 1000 and so on. And A* is an infinite set just that a finite number of strings from A are connected together to form a string in A*. This is just an example. So, we took two languages A and B. We saw the union, what the union is, what the concatenation is and what the star of A is. As an exercise, you can work out.

(Refer Slide Time: 14:25)

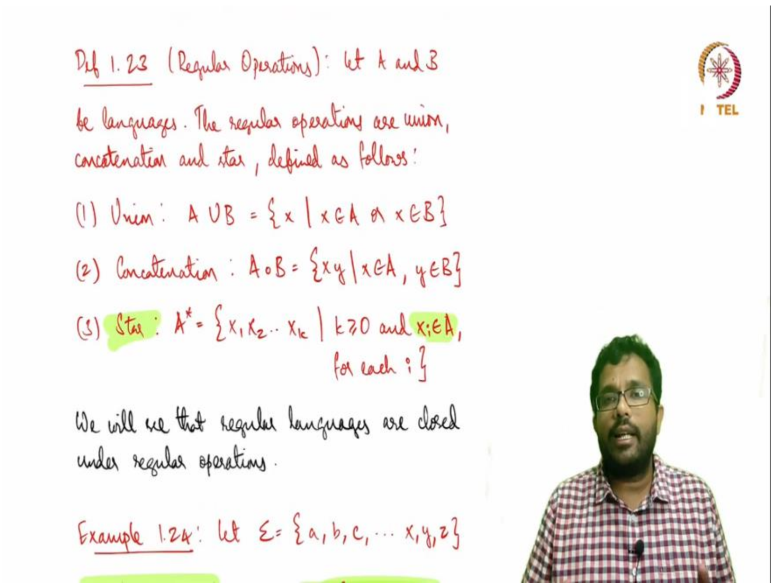


$A \cup B = \{ good, bad, boy, girl \}$

$A \circ B = \{ goodboy, goodgirl, badboy, badgirl \}$

$A^* = \{ \varepsilon, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, \ldots, badbadbad, \ldots \}$

Exercise: Work out B^*.



Def 1.23 (Regular Operations): Let A and B be languages. The regular operations are union, concatenation and star, defined as follows:

(1) Union: $A \cup B = \{ x \mid x \in A \text{ or } x \in B \}$

(2) Concatenation: $A \circ B = \{ xy \mid x \in A, y \in B \}$

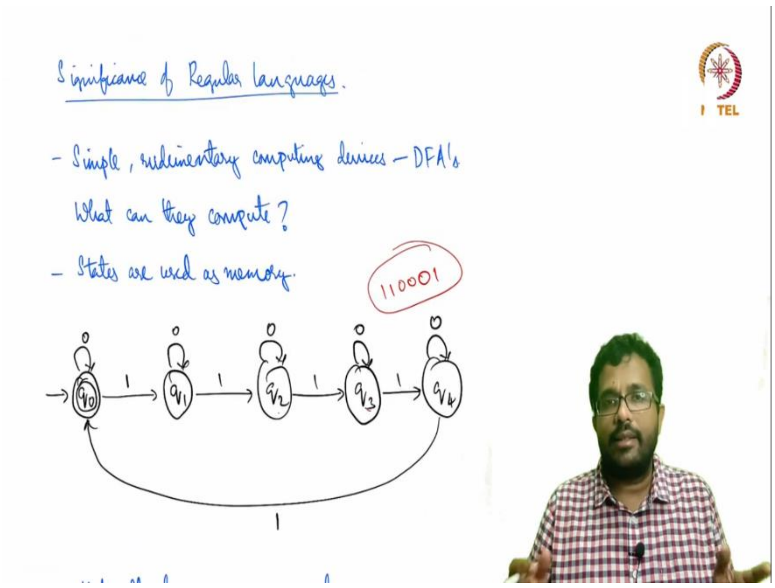
(3) Star: $A^* = \{ x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i \}$

We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{ a, b, c, \ldots x, y, z \}$

So, as an exercise, work out B*. You can write try to write down what B* is. Notice that ε, the empty string is also part of A*. Like I said, we already saw that regular languages are closed under complement. And then in the next coming lectures, we will see that regular languages are closed under regular operations as well. Meaning, if A is regular and B is regular, $A \cup B$ is regular. If A is regular and B is regular, A·B is regular. And if A is regular, A* is regular. e will see all of that in the next lecture. And not all of that, some of that. So, that is it.

(Refer Slide Time: 15:23)

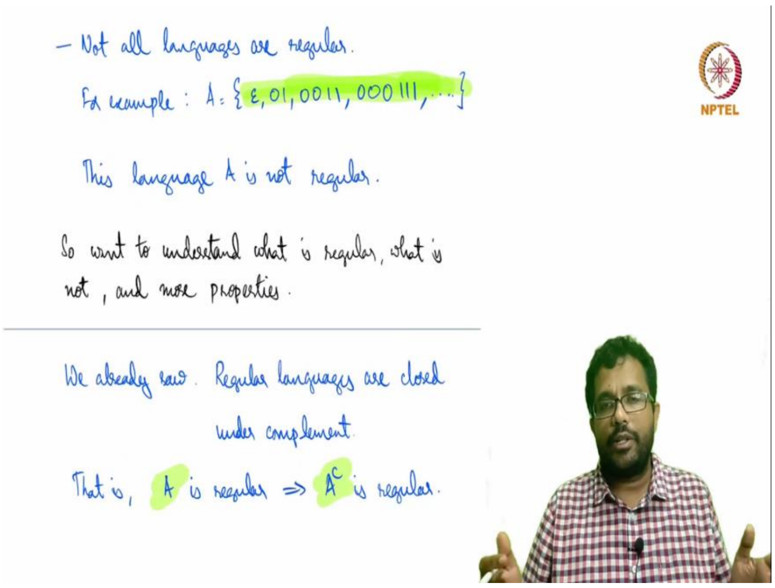


Significance of Regular languages.

- Simple, rudimentary computing devices – DFA's
  What can they compute?
- States are used as memory.

110001

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



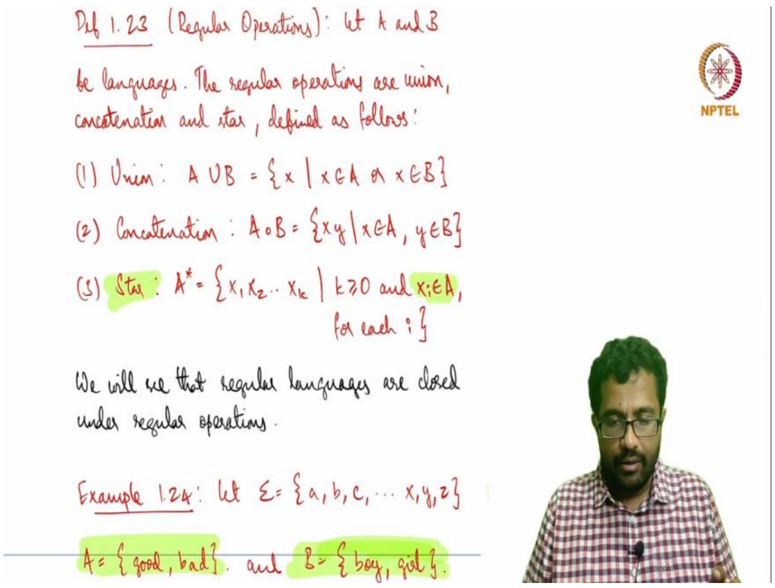
- Not all languages are regular.
  For example: $A = \{\varepsilon, 01, 0011, 000111, \dots\}$

  This language A is not regular.

So want to understand what is regular, what is not, and more properties.

We already saw. Regular languages are closed under complement.

That is, A is regular $\Rightarrow$ A^c is regular.



Def 1.23 (Regular Operations): Let A and B be languages. The regular operations are union, concatenation and star, defined as follows:

(1) Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

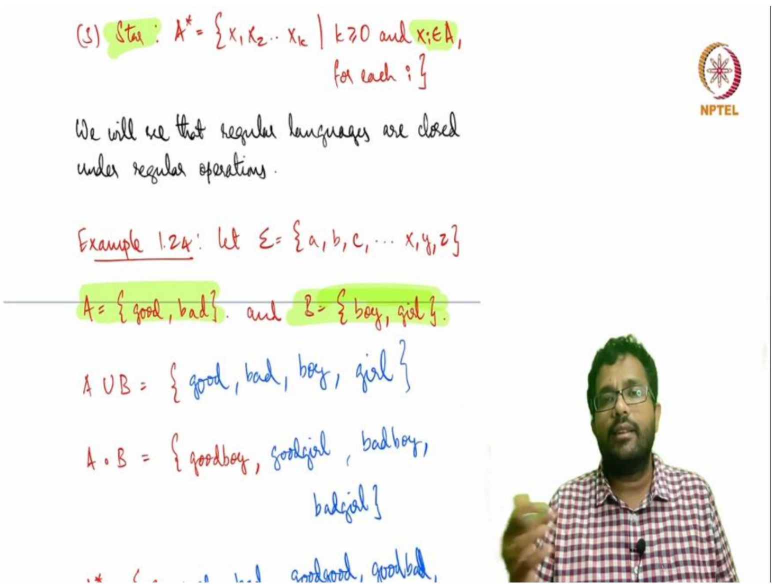
(2) Concatenation: $A \circ B = \{xy \mid x \in A, y \in B\}$

(3) Star: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i\}$

We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{a, b, c, \dots x, y, z\}$

$A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$.


(3) Star: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and } x_i \in A, \text{ for each } i\}$

We will see that regular languages are closed under regular operations.

Example 1.24: Let $\Sigma = \{a, b, c, \dots x, y, z\}$

$A = \{\text{good}, \text{bad}\}$. and $B = \{\text{boy}, \text{girl}\}$.

$A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$

$A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$

goodgood, goodbad,

So, we briefly explained the significance of regular languages, how computation happens through states and we said why all languages are not regular and hence it is interesting to see which ones are regular and which ones are not and what properties do regular languages have.

And so, these closure properties become interesting for the same reason and hence we explained the regular operations. So, regular operations are three operations on languages and we saw them. We said that regular languages are closed under regular operations without proof. And we will see the proof in the coming lectures.