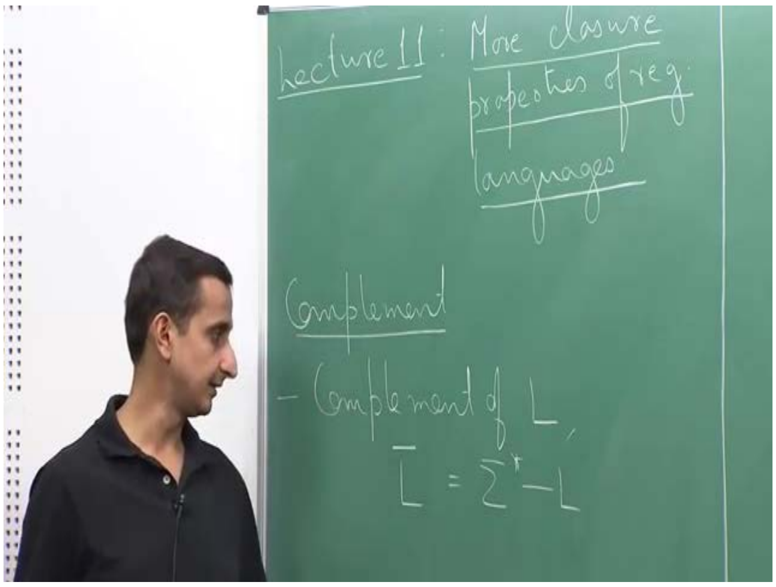
Theory of Computation

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

Lecture – 11
More Closure Properties of Regular Languages

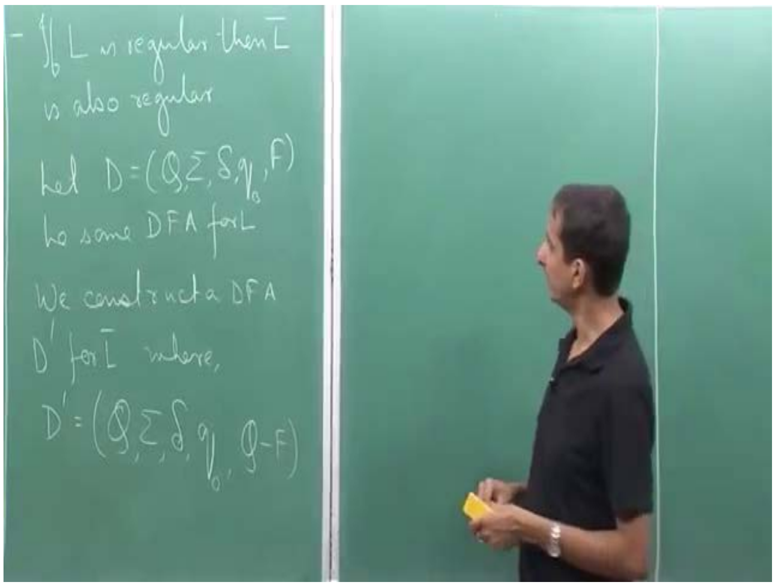
Welcome everybody, this is the 11th lecture. So, today we will discuss some More Closure Properties of Regular Languages.

(Refer Slide Time: 00:14)



The first property that we are going to see is a complement of a language. So, complement is always defined with respect to a universe, let say we are looking at languages over an alphabet sigma. So, sigma star is a set of all strings. So, we define complement of L. This is denoted by L with a bar on top is the set of all strings in sigma star minus L all strings that are there in sigma star, but not in L. So, the question is when we ask whether the complement operation is closed for regular languages, whether regular languages are closed under the complement operation is that if L is regular, is L bar also regular, or is the complement of L also regular. So, the answer is yes.

(Refer Slide Time: 01:47)



- If L is regular then $\bar{L}$ is also regular

Let $D=(Q,\Sigma,\delta,q_0,F)$ be some DFA for L

We construct a DFA D' for $\bar{L}$ where,

$D'=(Q,\Sigma,\delta,q_0,Q-F)$

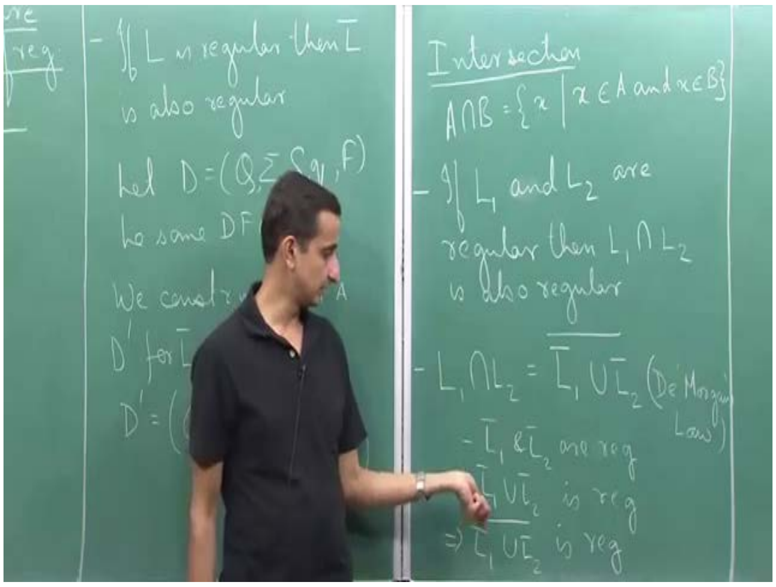
So, if L is regular, then L bar is also regular. So, this is the first property. So, why is this true? So, again when to give a proof of this statement, we will give a constructive proof; in the sense that the since L is regular, we will assume they are existed DFA for L; and using the DFA for L, we will construct an automaton for L bar. Since L is regular, let D equals Q, sigma, delta, q naught F be some DFA for the language L. So, what would be a DFA for the complement of L or L bar? So, observe how a DFA works.

So, if a DFA gets a string, it transits from one state to another on every symbol or on every symbol of that string. And when the string in, the DFA is in either an accept state or not an accept state if the DFA is in accept state the string is accepted otherwise the string is not accepted. So, basically if I want to construct a DFA, which accepts all strings that are not in L, I would just look at the complement of the set of accepting states. So, all the states that are not an accept state to write it formally.

So, we construct a DFA D prime for L bar, where D prime is equal to so the set states remain the same I mean it is essentially the same DFA except for the accept states the alphabet remains the same, the transition function remains the same as that of D, the start state also remains the same. The only thing that changes are the set of accepts state. So, in this case, it will be Q minus F, instead of F. So, this is the DFA for the language L bar.

So, this proves that this statement is true that if L is regular, L complement is regular.

(Refer Slide Time: 04:58)



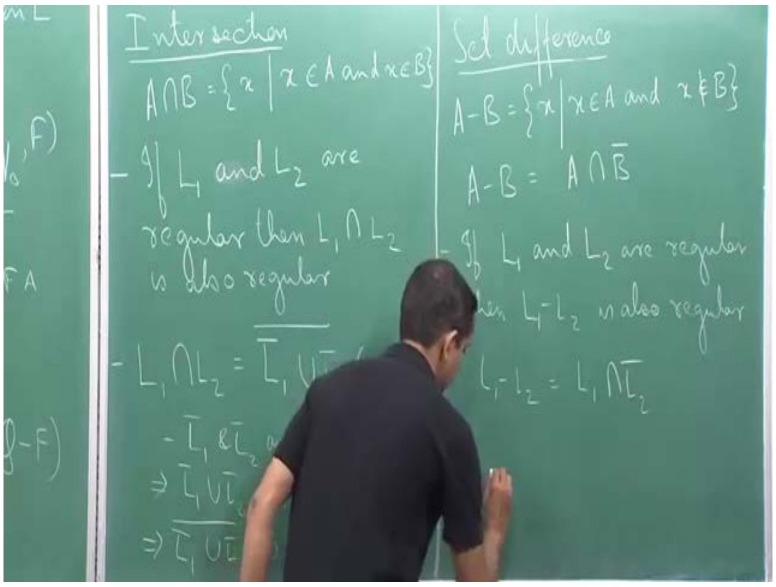
- If L is regular then $\bar{L}$ is also regular
Let $D=(Q,\Sigma,\delta,q_0,F)$
be some DF
We constr A
D' for $\bar{L}$
$D'=($
Intersection
$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
- If L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular
- $L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$ (De Morgan's Law)
- $\bar{L}_1$ & $\bar{L}_2$ are reg
$\bar{L}_1 \cup \bar{L}_2$ is reg
$\Rightarrow \overline{\bar{L}_1 \cup \bar{L}_2}$ is reg

Now let us look at the next property that of intersection So, formally, A intersection B is the set of all strings x such that x is in A and x is in B. So, what we will show is if L 1 and L 2 are regular, then L 1 intersection L 2 is also regular. So, how do we prove this statement? So, to prove this statement, we will actually so instead of constructing an automaton, we will use some of the closure properties that we have seen so far. So, what are the closure properties that we have seen so far? So, earlier, we have union, we have seen the concatenation and we have seen the star operation. And today we saw that under complement also regular languages are closed. So, how can I write L 1 intersection L 2 in terms of these operations?

Here we will use very famous result in computer science or not only computer science, but in set theory that is due to De Morgan, it is also known as De Morgan's law which states that L 1 intersection L 2 is can also be written as L 1 complement union with L 2 complement whole complement. So, this is De Morgan's law or De Morgan's theorem. So, this is something that all of you must have seen at some point earlier. So, we will make use of this fact. So, what we know to begin with is that L 1 and L 2 are regular. If L 1 and L 2 are regular by what we just saw L 1 complement is also regular. So, L 1 complement is regular, and L 2 complement is regular.

So, if L 1 complement and L 2 complement are regular then L 1 complement union L 2 complement is regular. And from L 1 complement union L 2 complement, the whole complement is also regular. So, if I write it in steps, what we have is L 1 complement and L 2 complement are regular, which implies that L 1 complement union L 2 complement is regular which implies that L 1 complement union L 2 complement whole complement is regular which is nothing but L 1 intersection L 2. Hence, L 1 intersection L 2 is regular. So, we actually used closure properties that we have seen earlier to prove this intersection.

(Refer Slide Time: 08:58)



Intersection

$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

- If L_1 and L_2 are regular then $L_1 \cap L_2$ is also regular

- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L}}$

- $\overline{L_1}$ & $\overline{L_2}$

$\Rightarrow \overline{L_1} \cup \overline{L}$

$\Rightarrow \overline{\overline{L_1} \cup \overline{L}}$

Set difference

$A - B = \{x \mid x \in A \text{ and } x \notin B\}$

$A - B = A \cap \overline{B}$

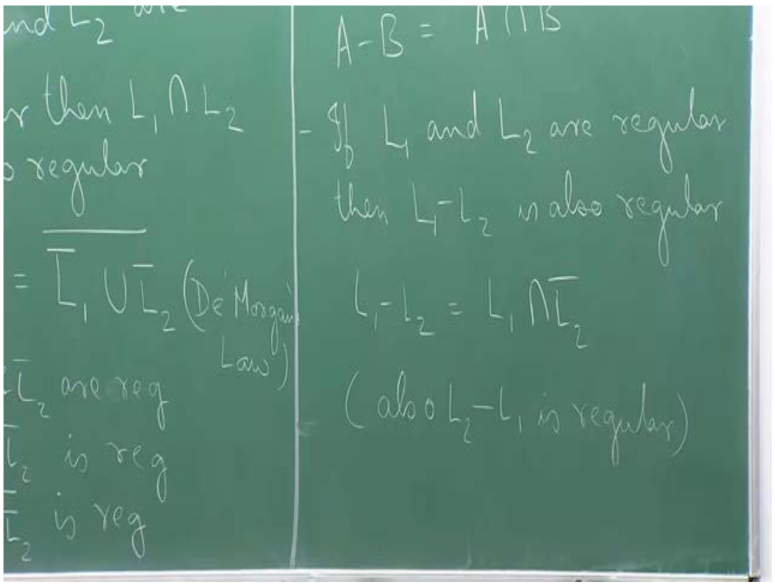
- If L_1 and L_2 are regular then $L_1 - L_2$ is also regular

$L_1 - L_2 = L_1 \cap \overline{L_2}$

The next one that we will see is set difference. So, if A and B are two sets, A minus B is defined as the set of all strings x belonging to A such that let me write it in this manner, so set of all strings x such that x is in A and x is not in B. So, once again if I just expand this out, I can write A set difference B as A intersection with B complement. Now, if L 1 and L 2 are regular then L 1 minus L 2 is also regular.

So, once again the proof follows by a similar logic that if I have L 1 minus L 2 is nothing but L 1 intersection L 2 complement. So, if since L 2 is regular, L 2 complement is regular. And since L 1 and L 2 complement are regular, L 1 intersection L 2 complement is also regular. So, this proves the set difference. So, just a small point here is that, so I have only written for L 1 minus L 2, but the same thing can also be set for L 2 set difference L 1.

(Refer Slide Time: 11:18)



nd L_2 are

r then $L_1 \cap L_2$

o regular

$= \overline{\overline{L_1} \cup \overline{L_2}}$ (De Morgan's Law)

$\overline{L_2}$ are reg

$\overline{L_2}$ is reg

$\overline{L_2}$ is reg

$A - B = A \cap \overline{B}$

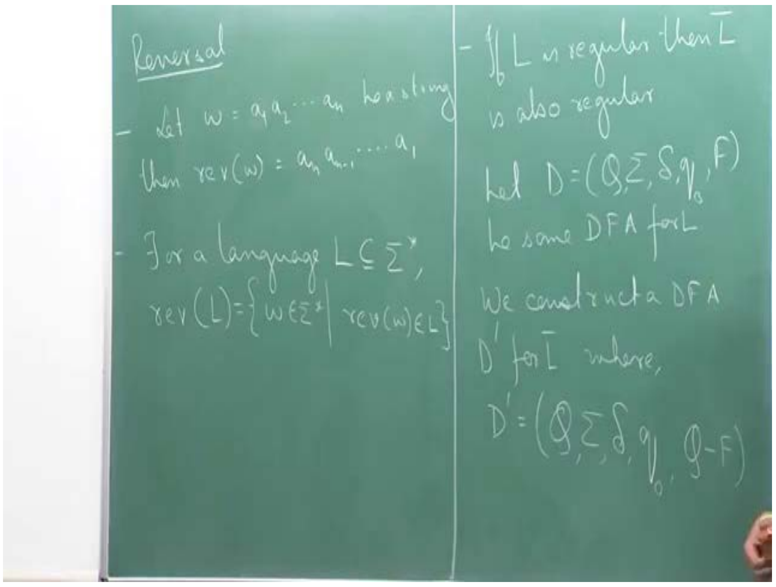
- If L_1 and L_2 are regular then $L_1 - L_2$ is also regular

$L_1 - L_2 = L_1 \cap \overline{L_2}$

(also $L_2 - L_1$ is regular)

So, by similar idea let me write it here; also L 2 minus L 1 is regular, I mean you just replace interchange L 1 and L 2 in this argument.

(Refer Slide Time: 11:39)



Reversal

- Let $w = a_1 a_2 \cdots a_n$ be a string
  then $rev(w) = a_n a_{n-1} \cdots a_1$
- For a language $L \subseteq \Sigma^*$,
  $rev(L) = \{ w \in \Sigma^* \mid rev(w) \in L \}$

- If L is regular then $\bar{L}$ is also regular

Let $D = (Q, \Sigma, \delta, q_0, F)$ be some DFA for L

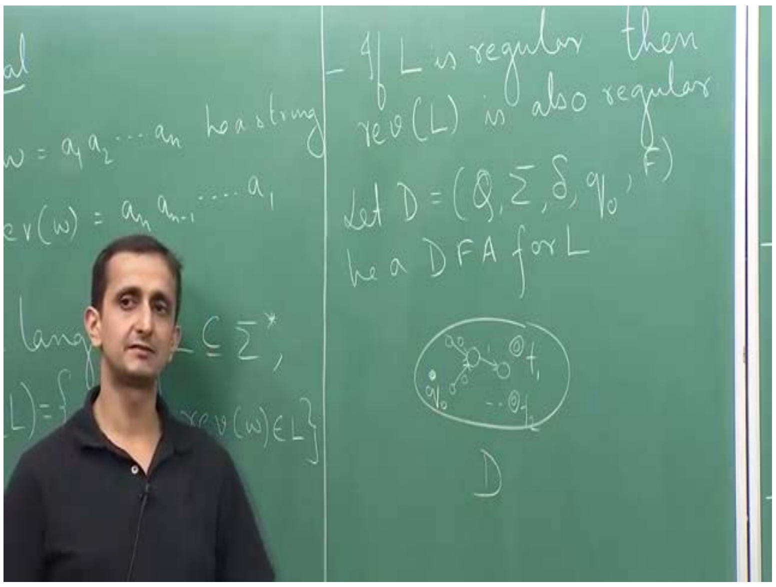
We construct a DFA D' for $\bar{L}$ where,

$D' = (Q, \Sigma, \delta, q_0, Q-F)$

The next property that we will see is reversal of languages. So, how do we define the reverse of a language? So, the reverse of a language is defined in terms of its strings. So, we define what is reverse of a string, and the reverse of a language is basically the reverse of all strings in the formal language. So, the reverse of a string is nothing but looking at the string from the reverse direction. Let me define what we mean by reversal of strings and reversal of languages. Let w equal to a 1, a 2 so on till a n be a string then reverse of w is the string a n, a n minus 1 so on till a 1 this the reversed string.

And for a language L subset of sigma star, reverse of L is the set of all strings w in sigma star such that reverse of w was in L. So, I look at all strings that are there in L, I reverse them and the language that I get as a result is reverse of L. So, once again we want to study closure properties of reverse of L; in other words, we want to pass the question that if L is regular is reverse of L also regular. And as it turns out that like all other properties that we have seen today, this is also true.

. (Refer Slide Time: 14:06)



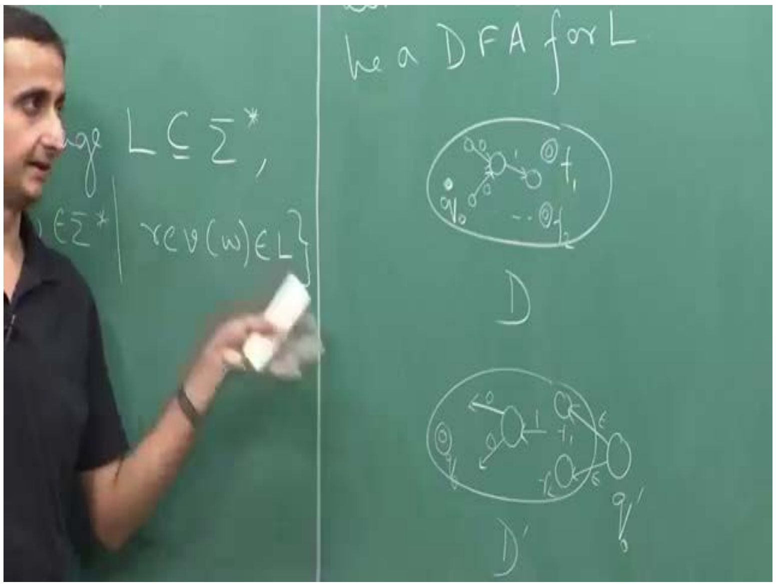
If L is regular then reverse of L is also regular. So, what is the proof for this statement, how do we prove this? So, we will prove this by giving a construction. So, like the where we showed that complement is closed under regular languages are closed under complement we will assume that since L is regular, there is a DFA for L; and using the DFA for L, we will construct an automaton for reverse of L.

So, how do we construct an automaton for reverse of L? Let D equal to Q, sigma, delta, q naught, F be a DFA for L; in other words L equals L of D. So, what is the intuition I mean what is the idea behind constructing an automaton for reverse of L. So, the idea is look at the set of strings that are there in language reverse of L. So, reverse of L basically contains all those strings that we get by reversing the strings in the language L.

So, we will try to construct an automaton that goes back from the accept state to the start state. Or whenever I have a path in my automaton or a walk in my automaton from the start state to the accept state in the automaton D, I should have another corresponding path in the new automaton that I am trying to construct which has a walk from the accept state to the start state. So that is essentially the idea. So, and what kind of a automaton will it be will it be deterministic or nondeterministic.

Let us look at an example or let us look at pictorial representation. Let see this is the automaton D that we have. So, we have the start state q 0, maybe this is accept state F 1, this is another accept state F 2. And maybe we have other states; so we have a state here which has may be two arrows coming to it from 0, and there is something going to another state on a 1 and so on. So, what we will do is we will reverse all the transitions that we have or in other words we will think of this as a graph, and we will construct another directed graph where all the transitions are reversed.

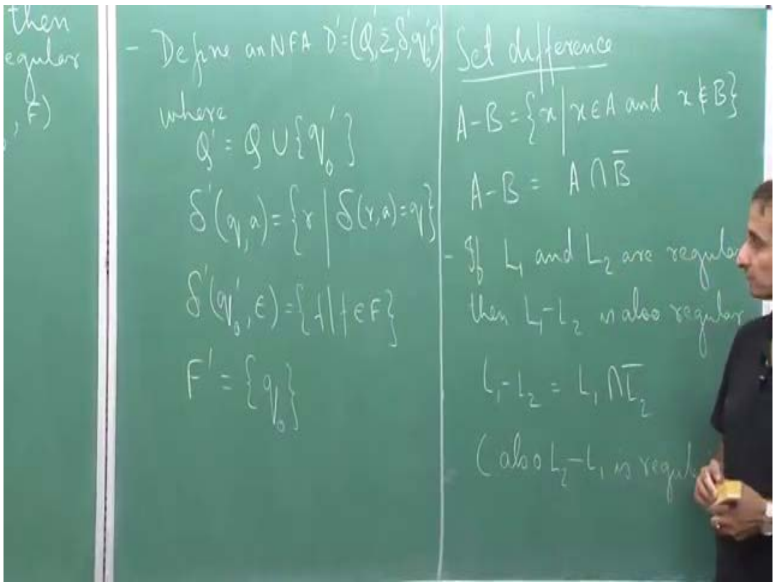
(Refer Slide Time: 17:45)



But now if I reverse all the transitions what do we get, so we will get another automaton d prime let say, where if I look at this state has - one incoming transition on 1, but it can have two outgoing transitions on a 0. So, it not necessary that from every state on an input symbol, there is a unit transition; you can have two transitions, you can have more than two transitions, and you can have zero transition.

In essence what we get is not a DFA, but we get a NFA. And the start state of NFA, we will non-deterministically decide which final state I want to start from. So, instead of so we have F and F 2 over here, so I will construct a new state let us call it q naught prime, and I will add epsilon transition to all my accept state in the DFA D. And my accept state of D prime is going to be the start state of D. So, this is essentially the idea.

(Refer Slide Time: 19:09)



then
egular
, F)

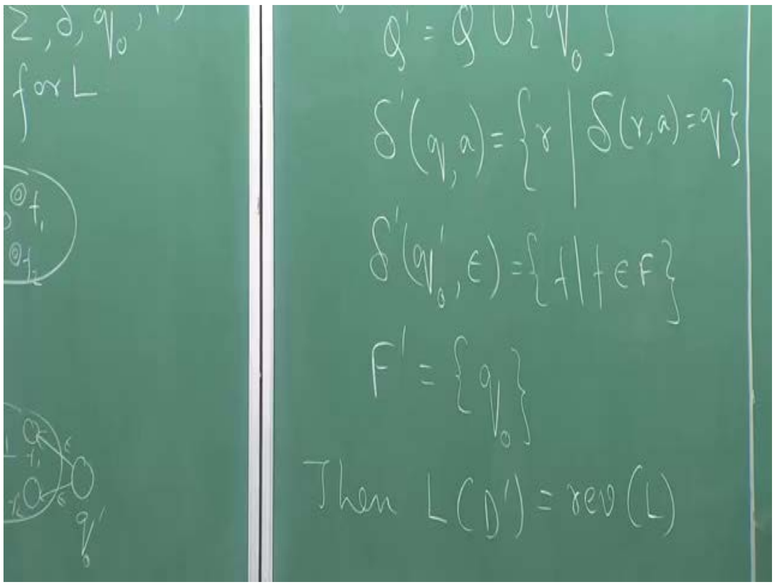
- Define an NFA $D' = (Q', \Sigma, \delta', q_0', F')$
where
$Q' = Q \cup \{q_0'\}$
$\delta'(q, a) = \{r \mid \delta(r, a) = q\}$
$\delta'(q_0', \epsilon) = \{f \mid f \in F\}$
$F' = \{q_0\}$

Set difference
$A - B = \{x \mid x \in A \text{ and } x \notin B\}$
$A - B = A \cap \overline{B}$
- If L_1 and L_2 are regular then $L_1 - L_2$ is also regular
$L_1 - L_2 = L_1 \cap \overline{L_2}$
(also $L_2 - L_1$ is regul

So, formally let us represent the DFA D prime. So, how do we formally define D prime. So, define an NFA D prime equals Q prime, sigma is going to stay the same, delta prime, q naught prime and F prime. Where Q prime is basically Q union with the new state I will call it this q naught prime, this new start state that I am adding; delta prime from a state q on a symbol a, what are the states that we go to. So, from delta prime on q on a I will go to all states r such that delta.

Let us try to picture this. So, from delta prime on q from a state q on a symbol a, I go to all those states such that from those states I had a transition on the same symbol to q. So, all those states are such that delta r comma a, had taken me to q. And I also need to add transition corresponding to this q naught prime. So, delta prime of q naught prime on epsilon should take me to all states F such that F was a accept state in the DFA D. So, we have delta prime done q naught prime is of course, I have defined that has as an extra state and what is F prime. So, F prime is the set which contains only the start state of the DFA D. So, this is the unique accept state of the NFA D prime.

(Refer Slide Time: 21:48)



for L

$Q' = Q \cup \{q_0'\}$

$\delta'(q,a) = \{r \mid \delta(r,a) = q\}$

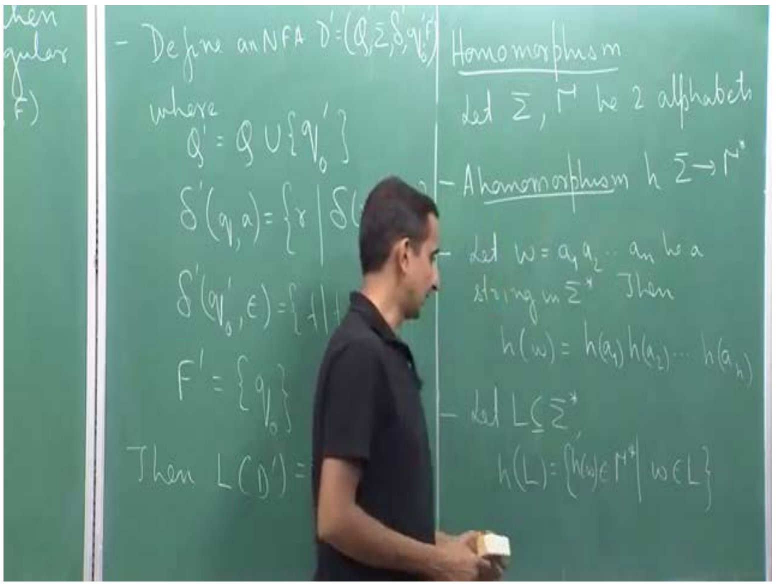
$\delta'(q_0', \epsilon) = \{f \mid f \in F\}$

$F' = \{q_0\}$

Then $L(D') = rev(L)$

So, then L of d prime is nothing but reverse of the language L. So, we have constructed an NFA which accepts the reverse of the language L, hence reverse of L is regular. Now, we will look at two more operations, and we will study closure properties under those operations these are slightly non conventional. So, these are related to what are known as homomorphism of strings.

(Refer Slide Time: 22:39)



- Define an NFA $D' = (Q', \Sigma, \delta', q_0', F')$ where

$Q' = Q \cup \{q_0'\}$

$\delta'(q,a) = \{r \mid \delta($

$\delta'(q_0', \epsilon) = \{f \mid f$

$F' = \{q_0\}$

Then $L(D') =$

<u>Homomorphism</u>

Let Σ, Γ be 2 alphabets

- A <u>homomorphism</u> $h \; \Sigma \to \Gamma^*$

- Let $w = a_1 a_2 \ldots a_n$ be a string in Σ^*. Then

$h(w) = h(a_1) h(a_2) \ldots h(a_n)$

- Let $L \subseteq \Sigma^*$,

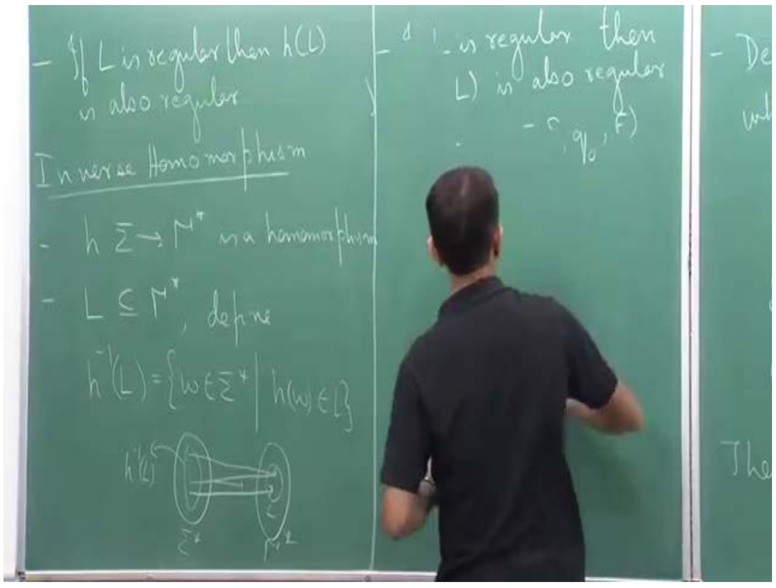
$h(L) = \{h(w) \in \Gamma^* \mid w \in L\}$

First let us understand I mean let us look at the definition of homomorphism. Let sigma be an alphabet so sigma is some alphabet. So, a homomorphism h is a map from sigma to some, one second, so let me just we can have different alphabets. Let sigma comma gamma be 2 alphabets, I mean they can be different alphabets one can be over 0 1, the other can be a b c or even they can be the same it is not necessary that they are different may be both are 0 1. So, this is also the alphabet 0 1, this is also the alphabet consisting of 0 1. Then a homomorphism h is a map from sigma to gamma star. So, basically what it does is that for every symbol in sigma, it gives a string in gamma, so some string over the alphabet gamma.

Now, such that so now we can define homomorphism on strings also, so let w be a string in sigma star. Then h of w so let us write so let w equals a 1, a 2 up to a n be a string in sigma star; then h of w is basically the string h a 1 concatenated with h a 2 concatenated with h a 3 so until h a n. So, basically every string in sigma star is map to some string in belonging to gamma star.

So, the closure property that we have with homomorphism is that if. So before define the closure property, let me define what is homomorphism of a language, so this gives us homomorphism of a string. Now let L be a language in sigma star h of L is defined as set of all strings w belonging to gamma star. Such that so I will write it as set of all strings h of w belonging to gamma star such that w belongs to l. So, I look at all strings w that are in the language L and I look at what is the map of that string with respect to the homomorphism h.

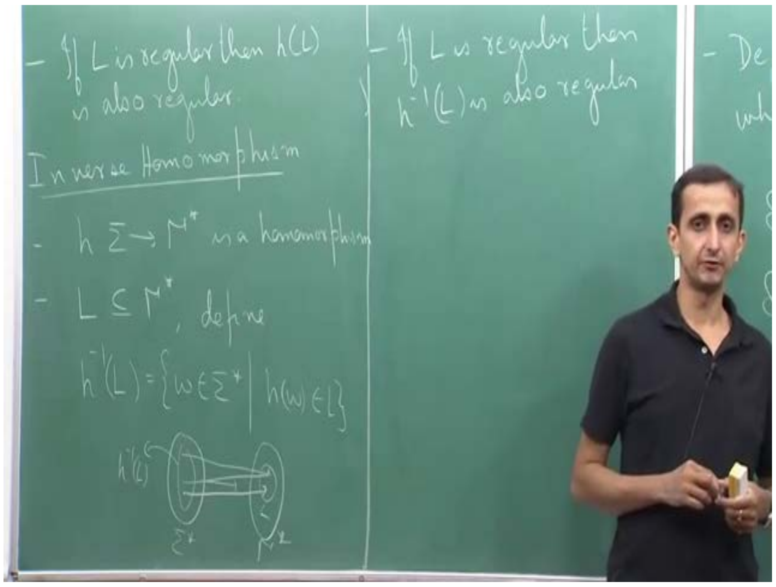
(Refer Slide Time: 27:24)


- If L is regular then h(L) is also regular
Inverse Homomorphism
- h Σ → Γ* is a homomorphism
- L ⊆ Γ*, define
h⁻¹(L) = {w ∈ Σ* | h(w) ∈ L}
h⁻¹(L)
L
Σ*
Γ*
is regular then
L) is also regular
q0, F)
- De
The

Now we are ready to state our the closure property that if L is regular then h of L is also regular, and the reason why this is true I mean I am not going to give a proof of this I am just going to briefly give a idea. So, you look at so since L is regular there will be some regular expression for L. So, look at the regular expression for L, so in the regular expression well will consist of symbols from your set your alphabet sigma. So, replace every symbol in sigma with the corresponding homomorphism. So, replace a belonging to sigma with h of a that will give you a new regular expression and that will be the regular expression for the language h for L, so that is essentially the idea.

Similar to homomorphism or we can study the inverse of it, which is known as inverse homomorphism. Let say that h from sigma to gamma star is a homomorphism then for a language L belonging to gamma star. So, take some language in gamma star, I can define the inverse of this language with respect to h. Define h inverse of L as the set of all strings w in sigma star such that h of w belongs to the language L. So, if you want to take it as a map, so let say this is sigma star, we have gamma star. So, if I take a language L over here, look at all the strings that map to some string in L and so on. So, the collection of all the strings, which map to some string belonging to L is the language h inverse of L.

(Refer Slide Time: 30:39)


- If L is regular then $h(L)$ is also regular.

Inverse Homomorphism

- $h: \Sigma \to \Gamma^*$ is a homomorphism
- $L \subseteq \Gamma^*$, define

$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}$

- If L is regular then $h^{-1}(L)$ is also regular

$h^{-1}(L)$
Σ^*
Γ^*

And what we again can prove is that if L is regular, then h inverse of L is also regular. So, I am not going to give a proof of this fact, but I will just leave it as an exercise for you to try out.

Thank you.