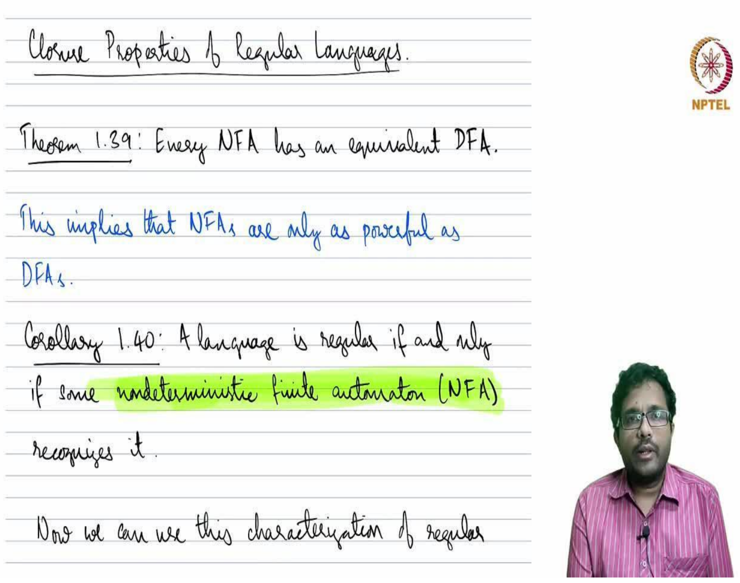
Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Closure of Regular Languages
Under Regular Operations (Using NFA)

(Refer Slide Time: 0:17)



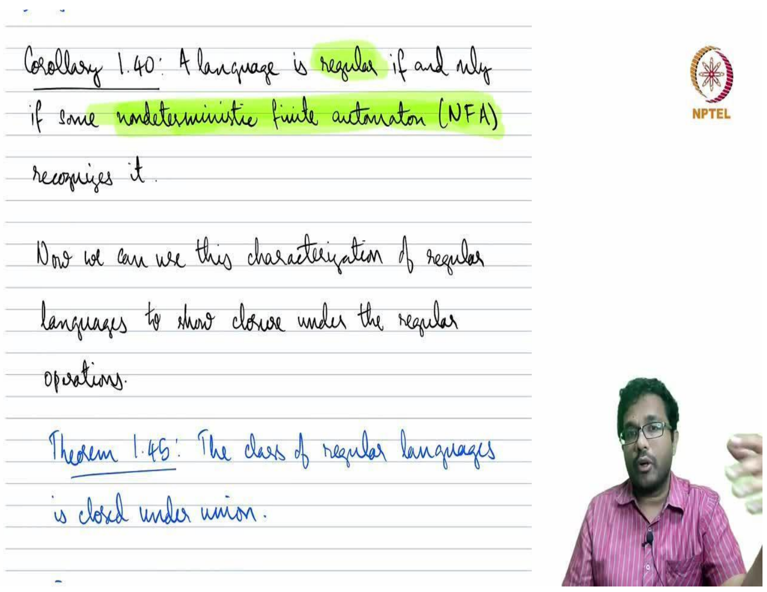
Closure Properties of Regular Languages.

Theorem 1.39: Every NFA has an equivalent DFA.

This implies that NFAs are only as powerful as DFAs.

Corollary 1.40: A language is regular if and only if some nondeterministic finite automaton (NFA) recognizes it.

Now we can use this characterization of regular



Corollary 1.40: A language is regular if and only if some nondeterministic finite automaton (NFA) recognizes it.

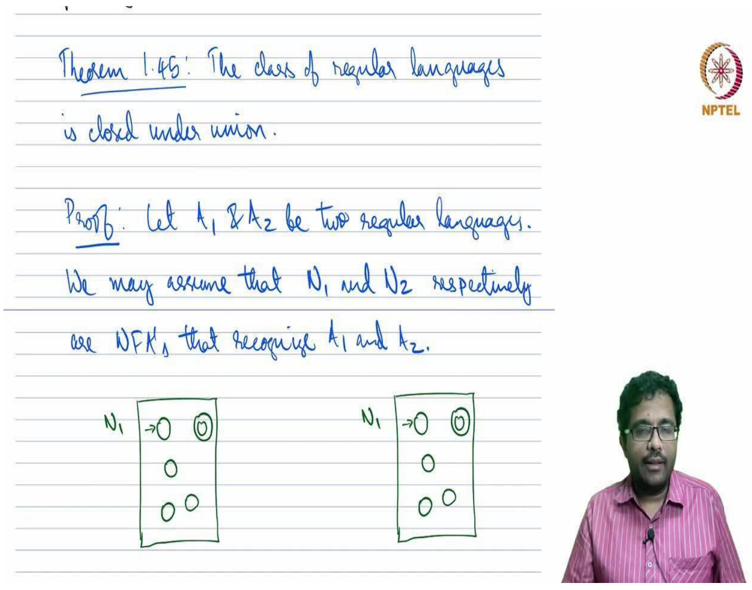
Now we can use this characterization of regular languages to show closure under the regular operations.

Theorem 1.45: The class of regular languages is closed under union.

Hello and welcome to lecture 9 of the course Theory of Computation. In the previous lecture, we saw NFAs and that NFAs are equal to DFAs as far as computability is concerned. Whenever there is a NFA that recognizes a certain language, there is also a DFA that recognizes a certain language. So it is almost immediately clear that NFAs are at least as powerful as DFAs. Now, with the observation that every NFA has an equivalent DFA it implies that the class of languages recognized by NFAs are the same as the class of languages recognized by DFAs and this is what we already know as the class of regular languages. So this gives us another definition or another characterization for the class of regular languages. So if a language is regular or a language is regular if and only if there is some NFA that recognizes it.

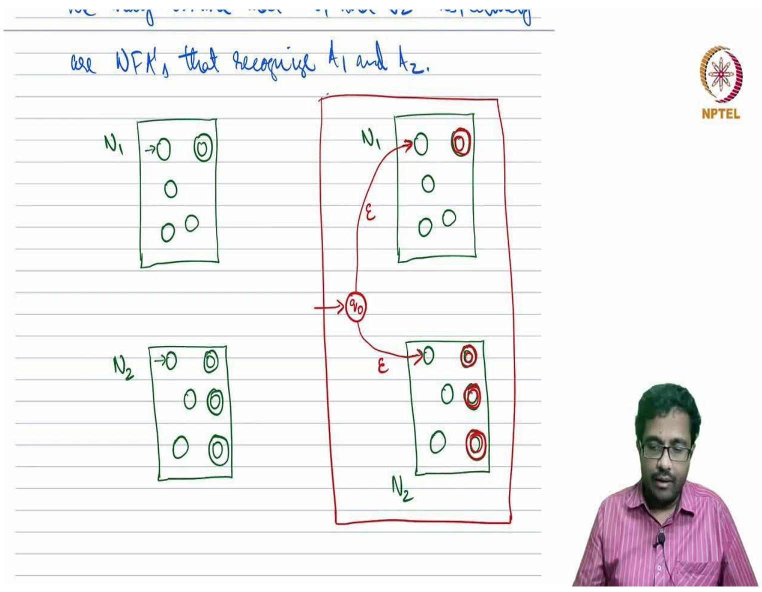
So now, if you recall we showed that regular languages are closed under complement we showed the regular language the code close under the union operation. So now, we will use this to show closure under other operations as well, namely the regular operations. So the regular operations were 3 union, concatenation and star. We already saw closure under union using a cartesian product DFA. Now, we will see a simpler proof which uses NFAs and followed by quite straightforward proofs for the closure under concatenation and the closure under star operation. So let us first see the proof that regular languages are closed under union that use NFAs, so we already saw the through using DFAs. So let us see.

(Refer Slide Time: 02:23)

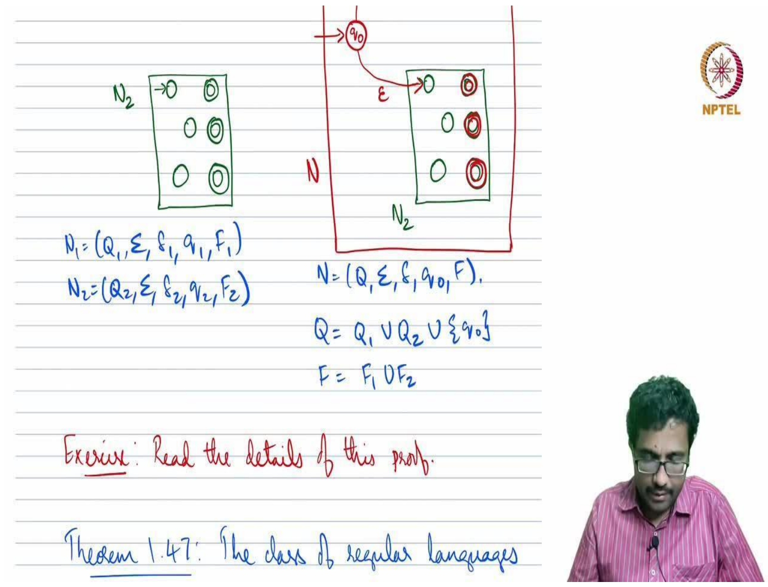


Theorem 1.45: The class of regular languages is closed under union.

Proof: Let A_1 & A_2 be two regular languages. We may assume that N_1 and N_2 respectively are NFA's that recognize A_1 and A_2.



are NFA's that recognize A_1 and A_2.



$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

$N = (Q, \Sigma, \delta, q_0, F)$.

$Q = Q_1 \cup Q_2 \cup \{q_0\}$

$F = F_1 \cup F_2$

Exercise: Read the details of this proof.

Theorem 1.47: The class of regular languages

So to begin with, we say that So we have to show that let us say if A_1 and A_2 are two regular languages, that union is also regular. So that A_1 and A_2, be two regular languages. So we may assume that the NFA is recognizing it. So we may assume that N_1 and N_2 respectively are NFAs that recognize A_1, A_2. And so now the goal is to build an NFA that recognizes the union. So let us say N_1 and N_2, the ones just drawn them N_1 at the top and N_2 at the bottom.

So now on the left, we have the N_1 and N_2. So I am not going to do a full proof in terms of the extent of details, it will be more the proof will be more depicted pictorially, one reason being we have already see one proof, but even the pictorial representation will be fairly clear is what I feel. So suppose N_1 and N_2 are this N_1 and this N_2, these are the NFAs that recognize A_1, A_2. So notice that there is one starting state for N_1, one strategy for N_2 , N_1 has one accepting state N_2 has 3 accepting state and then depicted some extra states, of course, they are not dripped represented there could be many more states, I am not bothered about them, that could be all the transitions there will be on the transitions, which I am not writing down just think of them, the only things that we will require are the start state and accepting states. So that is the extent to which we will operate on this.

So what we will do is, we will build an NFA that which actually combines these two NFAs, such that the combined NFA will accept a string if and only if either one of the NFAs accepts a state. So what we want to do is we want a structure by which either NFA N_1 accepts or NFA N_2 accepts. If either one of them accepts, the combined NFA should accept. The simplest way is to do the following, we want some kind of a redirect, so you want the capability to choose, do you want to try N1 acceptance in N_1 or N_2?

So you want to try acceptance in either an i N_1 of them. So the simplest way is to add a new start state, you add a new start state, just one second, so I will erase this labeling from here, maybe I will write it below it, I will add a new start state. And then what I will do is I will add transitions to the existing start states of N_1 and existing starts states N_2. So this is a new start state, let me call this start state, let us say I will call this q_0. And from q_0 I am adding a transitions to the start state of N_1 and to the start state of N_2, but what symbol should this transition speak? It should be an empty string.

So these are both epsilon transitions. So this NFA is like this. So initially, that is q_0 which is a start state and from which you could make epsilon transitions to either the NFA N_1 or the NFA N_2. The transitions inside N_1, N_2 whatever they were, they will remain intact. And once you make the epsilon transition you are now let us say you move to N_1 using the epsilon transition, you move to N_1. And now you will just get stuck or you will just operate inside N_1 after which there is no going to N_2 and then you get accepted if after reading the string, you are at the accepting state of N_2.

So which is a way the only way to get acceptance inside N_1 is if you entered the accepting state, so that is when the string is a is a member of A_1. And similarly, the only way to get accepted through an accepting state of N_2 is if the string is a member of A_2. So the accepting, so the start state is the new state q_0, the accepting states of the new NFA will be all these accepting states. So the accepting state of N_1, and the accepting states of N_2, all of them. So they have multiple acceptances all of the multiple acceptances. So now, here there is one accept set of N_1 and 3 of N_2. So now the joint machine has four accepts overall, whatever was inside and wanted to remain as it is.

So just to note, if N_1 was let us say Q_1, sigma, delt A_1, small Q_1, F1 and N_2 is Q_2, sigma, delt A_2, small Q_2, and F 2. The new one let us say the new machine let us call it N, let us call this machine N. Let us say if N is (Q,Σ,δ, q_0,F) then we can actually write down all of this what is Q, what is sigma, what is delta, etcetera. Σ is the same for everything but let us say Q for instance is you have Q_1 the states of N_1, together with the states of N_2 Q_2, together with the new state which is q_0.

And F is the is a union of accept states of N_1 and N_2 so it is F is the F1 union F2 so nothing new is added. And similarly you can, so q_0 is near the start series, so almost everything said the only thing that is a bit more involved is the new transition function that requires a bit of detail. You may have a look at the book on how they explain this. So just have a look at the book where the details are provided there. So please go through this and try to understand.

(Refer Slide Time: 10:07)

$Q = Q_1 \cup Q_2 \cup \{q_0\}$

$F = F_1 \cup F_2$

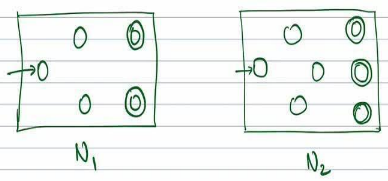
Exercise: Read the details of this proof.

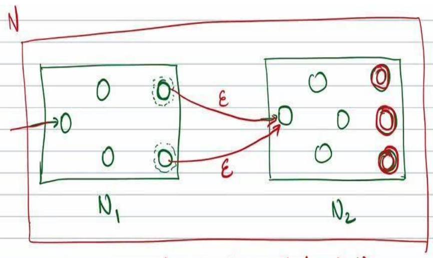
Theorem 1.47: The class of regular languages is closed under the concatenation operation.

Proof Sketch: Let A_1 and A_2 be regular languages recognized by NFA's N_1 and N_2 respectively.

is closed under the concatenation operation.

Proof Sketch: Let A_1 and A_2 be regular languages recognized by NFA's N_1 and N_2 respectively.





Start state of N = Start state of N_1

Accept states of N = Accept states of N_2.

w is such that $w = w_1 w_2$ where $w_1 \in A_1$ and $w_2 \in A_2$

$\Updownarrow$

w is accepted by the above NFA.

So the next result of the next closure property is the closure under the concatenation operation. So we just saw the unit operation. In fact, we had already seen a proof that this is a new proof using NFAs. So let us see the proof for the conclusion of the concatenation. So let similar to what we just saw that A_1 and A_2 be the 2 regular languages and so because they are regular and because we have the new characterization using NFAs we can assume that N_1 and N_2 are the two NFAs are two NFAs recognize A_1 and A_2 respectively.

So now, let us say N_1 is the one on the left side. So this is N_1 and N_2 is the one on the side. So both of them have one start state. So obviously, there is a unique start set for each NFA. N_1 has two accept states and N_2 has 3 except states. And now, we want to combine these N_1 and N_2 in such a way that the new we want to somehow build an NFA using N_1 and N_2 together so that the combine NFA accepts the concatenation language. So what we will make use of so I have ever again drawn them so maybe I will just move it a bit more below. So this will be our working copy. So now we want to somehow add things to this N_1 N_2 so that we want to make it into one machine and such that this one machine accepts the concatenation operation.

So what is concatenation? Concatenation language means w will be in the concatenation language if the string w can be written as w_1, w_2 as a concatenation of two strings. There, the first thing is an A_1 and the second string is an A_2. So the what we are trying to do here is we first pass the string through N_1 and then we want to pass through N_2. If you recall, after we proved a closure under union using DFAs, we discussed this possibility. The issue was that if you have just one string, the string itself a string consisting of symbols, but we do not know where to split into w_1 and w_2 That was the issue with the proof using DFAs.

But in the case of NFAs this can be handled because NFAs have the capability of non-determinism. So the NFAs itself we can have the capability inbuilt or inbuilt in the construction such that there is an accepting computation if there is some way to split this string w into w_1 and w_2, such that w is in A_1 and w_2 is in A_2. So that is our goal.

So we want to accept w if there is any way of splitting it could be the first symbol of w and last N minus 1 symbols of w or it could be the first N minus 1 symbols of w in A_1 and the last lone symbol accepted by A_2. So it could be any of these four possibilities, but whatever be the possibilities w will be accepted if there is a possibility. So this non-determinism comes in handy here. That is why it did not work in the case of DFAs and that is why it will work in the case of NFAs.

So let us try to see what we need to do. So we first have N_1 in the left side which is the NFA for A_1 and N_2 in the right side. So like before, we are not going to bother about states internal states of N_1 and internal states of N_2 and all the internal transmissions there is no change for any of that. The only operation that we will do will involve messing with the start state and the accept states of both N_1 and N_2.

So what we will do is we at any point, if we want to see if w if this string can be split into w_1, w_2 where is w_1 is in a string in A_1 which means w_1 ends in an accepting state of N_1 and then w_2 starts. Meaning there should be a way to go from the accept states of N_1 to the start state of N_2. So what we will do is we will have epsilon transitions from the accept states of N_1 to the start state of N_2.

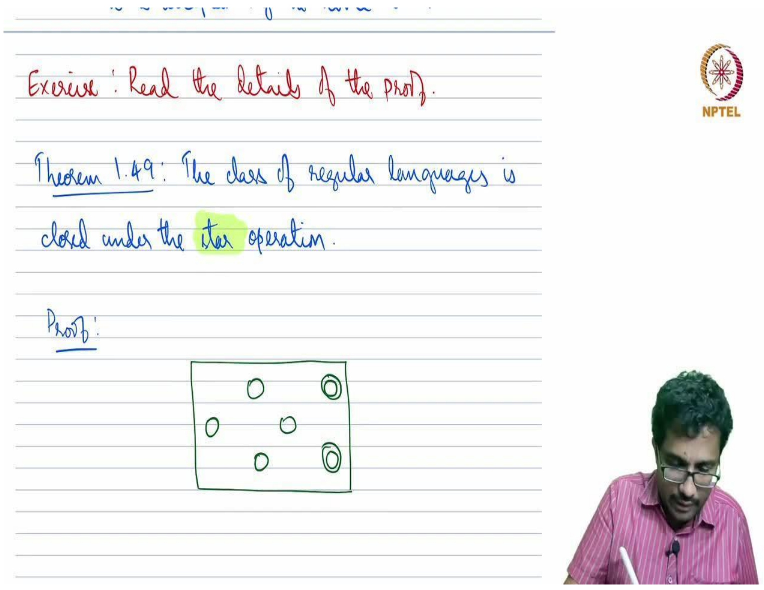
And now, the rest the like, so now the only strings if you see the only strings that will get accepted in this combined machine are now notice that in my figure, the accept states of N_1 are still listed and are still drawn as an accept state, so that should not be the case. So now let me see if, so I will just draw them as these are not accept states so this, maybe I will just put some this well accepting states of N_1, but now they are not. So I am just indicating that by this dotted line, so do not think of them as accept states. So these two are not accepting states of the combined machine.

So this is our combined machine and let us call it N. And the accept states of N would be just the accept states of N_2, these 3 will be accepts states of N. So the only way to if you see what is happening, if there is any way to split the string into w_1, w_2, such that w one accepted by N_1 and w_2 accepted by N_2 that way we can accept after when you are done processing w_1, you will be at an accepting state of N_1, and then you can take the epsilon transition to the start state of N_2, which is no longer the start state of N. And then you can complete processing w_2 which will take you to an accepting state of N_2, so that is.

So just couple of details, start state of N is the start state of N_1 and accept states of N will be the accept states of N_2. So and the rest of the details you can see in the book. But pretty much this is the proof, it is quite simple. We do not tamper with the internal states or the internal transitions, we just make the start state of the N_1 the start state of N, accepts states of N_2 the accept states of N. And we just add epsilon transitions from the accept states of N_1 to the start state of N_2 and that is it. This is our NFA that accepts the concatenation language.

So now, you can see it is very-very simple, much more simpler than the Cartesian product idea for the union that we had using DFA. So this must give you some idea of why NFAs are useful, why we are harnessing the power of non-determinism to be able to prove quite easily what we want. So that is the next closure under the concatenation operation. The final regular operation is a star operation.

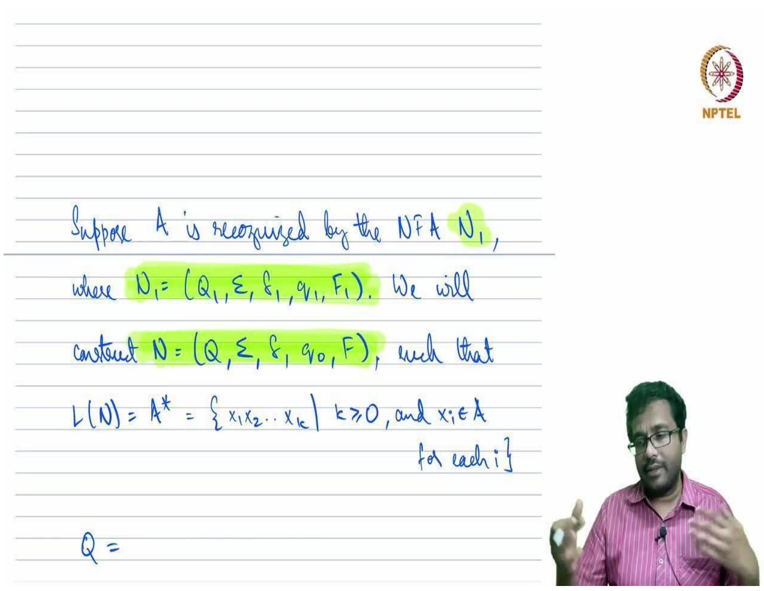
(Refer Slide Time: 18:30)



Exercise: Read the details of the proof.

Theorem 1.49: The class of regular languages is closed under the star operation.

Proof:



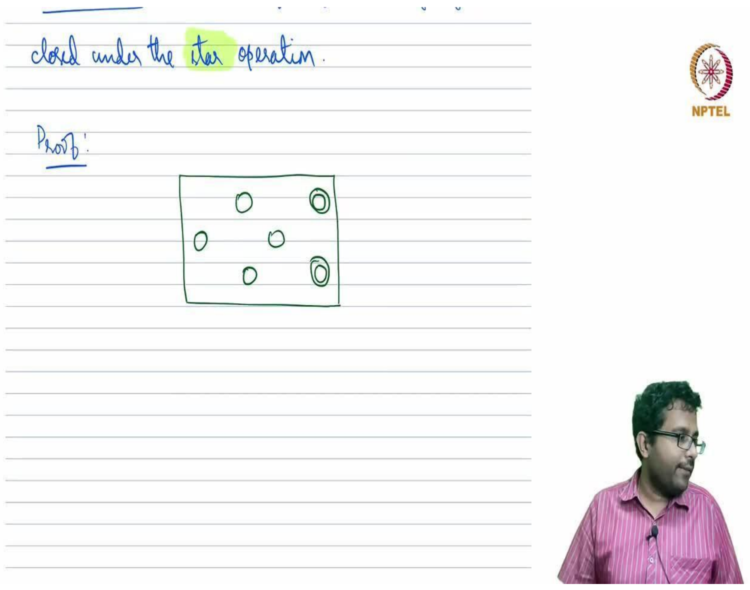
Suppose A is recognized by the NFA N_1, where $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$. We will construct $N = (Q, \Sigma, \delta, q_0, F)$, such that $L(N) = A^* = \{x_1 x_2 \dots x_k \mid k \geq 0, \text{ and } x_i \in A \text{ for each } i\}$

$Q =$

So now we want to show closure under the star operation. So the claim is that a class of regular languages is close to the star operation. So like before, we may assume that let you can see. Maybe I will just move this a bit down. So we want to so let A be regular and N_1 be the I think a may be seeing it below. So it is already written here so I do not want to repeat it. I will just erase it, move this back up. So let A be regular.

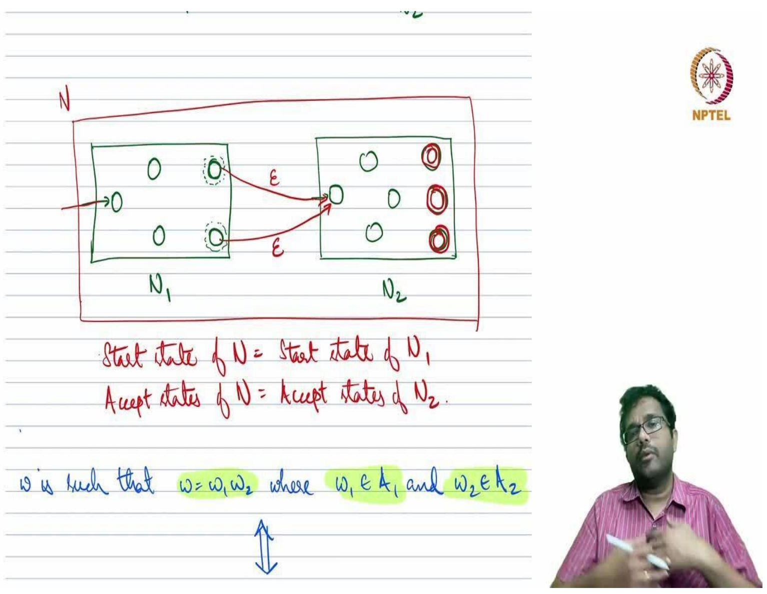
So A will be recognized by some NFA let us call N_1 which is given by N_1 is (Q1,Σ,δ, q_0 ,F1). And our goal is to construct N where N is has a states Q the alphabet is the same sigma and delta as a transition q_0 is a starting and F is an accepting function. Now we want to construct N such that the language recognized by N is the concatenation start of A, which means it should be a concatenation of some number of strings, each of those strings should be in A. The sum number is not specified, it could be 1, like it could vary also. So it could be 0, 1, 2, 3, any numbers. So when you have zero strings from A, it gives you the empty string, when you take one string from it is just the language A itself. If you take 2 strings from A it is like A concatenation with itself so it is the union of all this gives us a star.

(Refer Slide Time: 20:36)



closed under the star operation.

Proof:

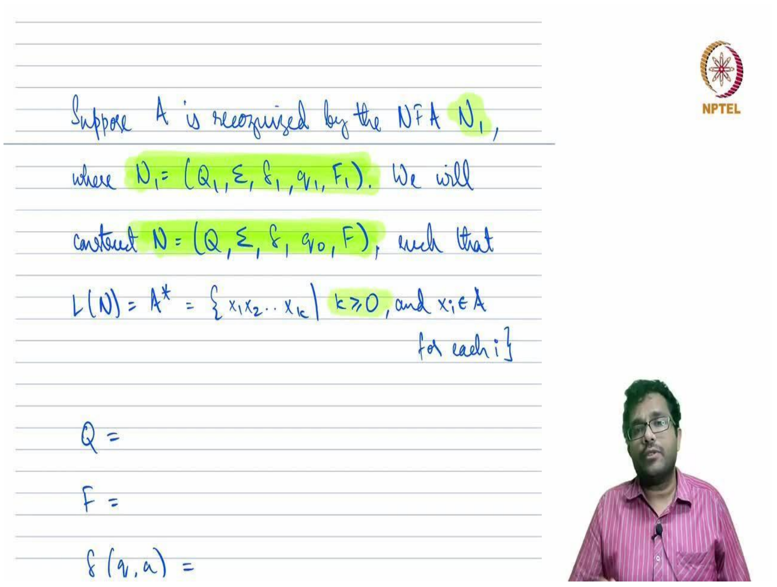


Start state of N = Start state of N_1

Accept states of N = Accept states of N_2.

w is such that $w = w_1 w_2$ where $w_1 \in A_1$ and $w_2 \in A_2$

$\Updownarrow$

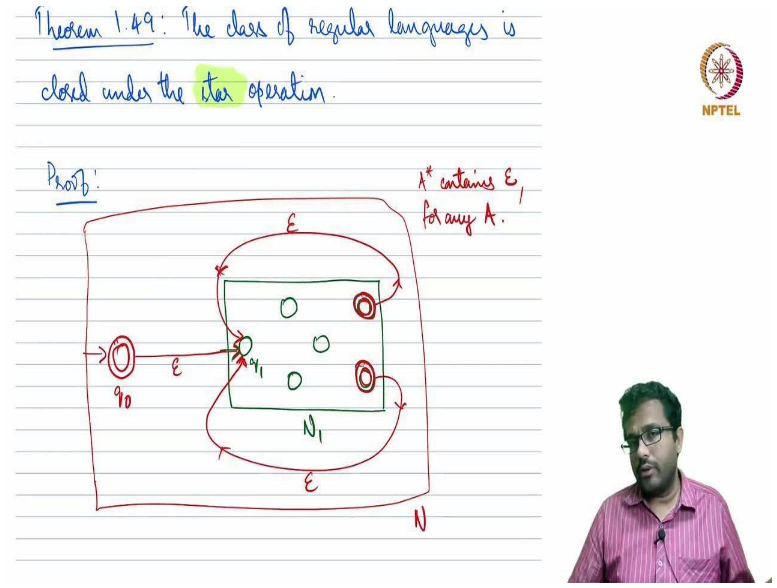


Suppose A is recognized by the NFA N_1, where $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$. We will construct $N = (Q, \Sigma, \delta, q_0, F)$, such that $L(N) = A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0$, and $x_i \in A$ for each $i\}$

$Q =$

$F =$

$\delta(q, a) =$



Theorem 1.49: The class of regular languages is closed under the star operation.

Proof:

A* contains ε, for any A.

So now what we have is the let us say this is N_1 So this is the NFA for N_1, let us say this is the start state of N_1. Now how do we modify this in such a way that we accept all the star operation language? So now, so the idea is kind of similar to what we saw here. So here in concatenation, after reading, going through N_1, we want to go to N_2. But here there are no two languages there is only A, so now we want to accept A star, which means we want to be able to come back to the we want to be able to like say if one string breaks into two pieces, let us say w breaks into w_1 and w_2, where both of them are in A w_1 and w_2, so basically, we want to pass through the NFA twice. Or it will be 3 pieces w_1, w_2, w3. So we want to an each of them are in A, so we want to go to the NFA thrice.

So after getting to the accept state, we again want to come back to the start state and again should be able to go. So the simple idea that we have to execute here is to add epsilon transitions from the accept state of the machine to the start state of the same machine to start state of the same machine so that is what I am doing here. So I am adding epsilon transitions from the accept states to the start state.

So now, this will accept now and the start state remains the same and the accept states also remain the same, but his will accept all strings of the form x_1, x_2, etcetera. where each x i is in the language A. But there is one small issue here, the one small issue is that the star operation so any language A star, this usually contains the empty string because in this definition with a star, we can in this definition over here we can have k equal to 0, which means there is no strings, like there is nothing and that gives us that empty string.

So for any language A, A star contains empty string, for any language A. But here, if the empty string has to has to be accepted there is only one way you have to have the start state as an accepting state. Or maybe you should have an epsilon transition from the start state to either state which should be an accepting state. But we do not know whether we just took a general language A and the NFA corresponding to it N_1, we do not know if these properties are satisfied in this NFA. So for instance, A may not have epsilon in this, A may not contain epsilon. So the start state of A may not be an accepting state, if the start state is an accept state then A contains epsilon.

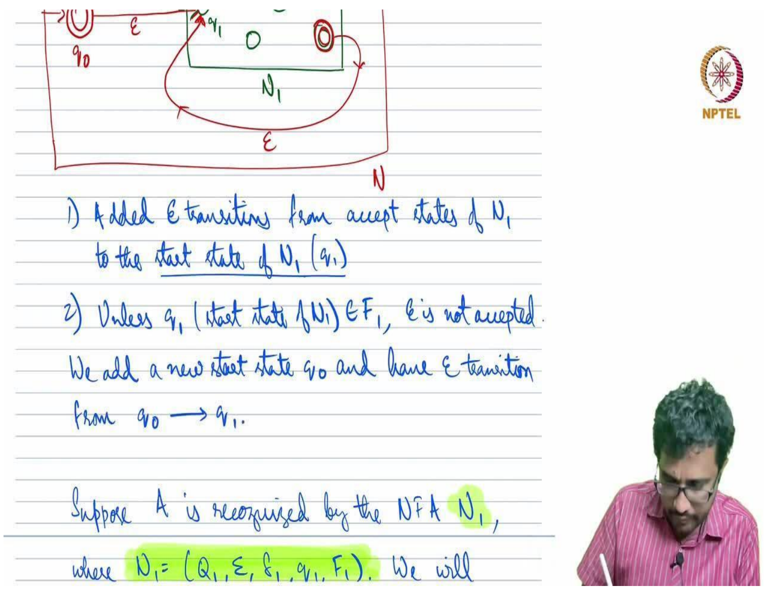
So we cannot insist that the start state will be an accepting state. And if we modify the start state to be an accepting state, that can cause other problems, because if you modify the start state, if you make the start state to be an accepting state, whenever in between computation you reach the start state, you may end up there is a possibility of accepting and this will result in extraneous strings getting accepted. So we cannot just convert the start state in our accepting state.

So we want another way to accept the empty strings not by modifying the start state because that can cause like suppose there is some computation where you go this, this, this, something and this ends up being accepted if you make the start state being accepted, at being an accepting state, so we do not want to do this. So we have to have some other way of accepting the empty string.

So what is the other simplest way? The other simplest way is to just add a new start state which is an accepting state. So you add a new start state which is an accepting state, let us called this q_0. So this this old one was say Q_1 and we have an epsilon transition from q_0 to Q_1. So now, the start state will be the empty string and will be accepted because the start state is an accepting state. So the empty string is automatically accepted.

And now, so now what is N? So the N is this maybe I will just again I will just pull it down a bit so that I can draw a box around this it, N is this, this is our N. And the accept states of N are q_0 and origin accepting states of N_1 So which is this? And the start state of N is small q_0 and that is it. So now, we are accepting both the empty string as well as x_1, x_2, x_3 or x_1, x_2, x_3, x4, where each of these excise are in A_1.

(Refer Slide Time: 27:08)



1) Added ε transitions from accept states of N_1 to the start state of N_1 (q_1)

2) Unless q_1 (start state of N_1) $\in F_1$, ε is not accepted. We add a new start state q_0 and have ε transition from $q_0 \longrightarrow q_1$.

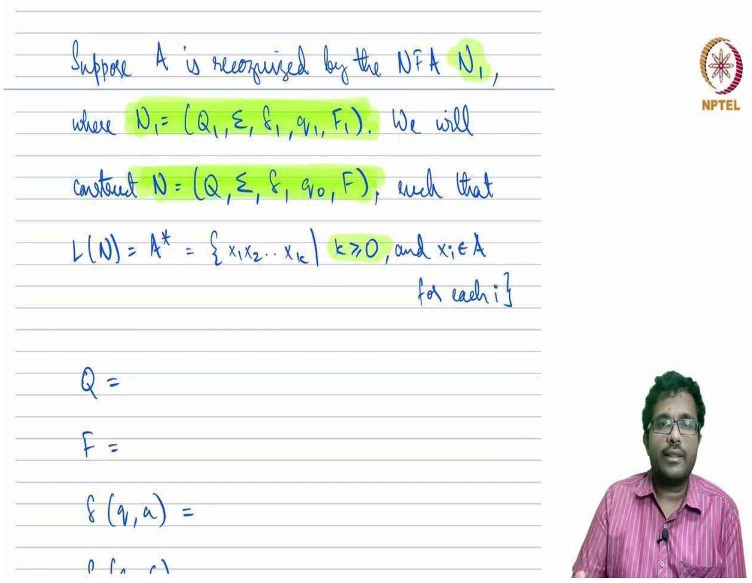
Suppose A is recognized by the NFA N_1, where $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$. We will

So let me just list down what all we did it. So one is that we added epsilon transitions from accept states of N_1 to the start state of N_1 and second thing to ensure that empty string is in the language. So unless Q_1 which is the start state of N_1, so this is Q_1 let the start state of N_1 is Q_1. So unless Q_1 which is the start state of N_1 unless this is an accepting state of N_1 let F1 is the accepting state of N_1 we need to maybe I am kind of running out of space maybe let me see what I can do, maybe what I will do is I will just push this back up a bit. So unless q in F1. Epsilon string is not accepted. So then we add a new start state q_0, we add a new start set q_0 and have epsilon transition from q_0 to Q_1. So that is the extent of the construction.

So, you added these loops, epsilon loops from the accept state to the start state, and then you add a new start state and have an epsilon transition from the new start state to the old existing start state. And so, since we did not write down the details of the earlier one, we will write down the details, but before that, I hope the idea is clear that this indeed accepts all the strings that are in A star and does not accept any other string.

So if you can see the only string that it accepts through q_0 is empty string, because you can never come back to q_0. And only strings that accept through the original accepting states, whichever ones they are you do some transitions you come to the accept state and then you again go back to start state and again come back to the accepting state. So it must have gone from start to accept some k number of times, which is exactly the start operation and nothing else is accepted.

(Refer Slide Time: 31:30)



Suppose A is recognized by the NFA N_1, where $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$. We will construct $N = (Q, \Sigma, \delta, q_0, F)$, such that $L(N) = A^* = \{x_1 x_2 \dots x_k \mid k \geq 0$, and $x_i \in A$ for each $i\}$

$Q =$

$F =$

$\delta(q, a) =$

$Q = Q_1 \cup \{q_0\}$ where q_0 is the new start state.

$F = F_1 \cup \{q_0\}$

$\delta(q,a) = \delta_1(q,a)$ when $q \in Q_1$ and $a \neq \varepsilon$

$\delta(q,\varepsilon) = \delta_1(q,a)$ when $q \notin F_1$ and $a \neq \varepsilon$

$\delta(q,\varepsilon) = \delta_1(q,\varepsilon) \cup \{q_1\}$ when $q \in F_1$

$\delta(q_0,a) =$

$\delta(q_0,\varepsilon) =$

state.

$F = F_1 \cup \{q_0\}$

$\delta(q,a) = \delta_1(q,a)$ when $q \in Q_1$ and $a \neq \varepsilon$

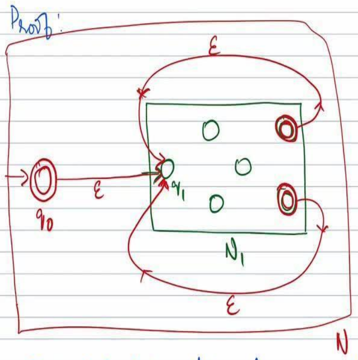
$\delta(q,\varepsilon) = \delta_1(q,a)$ when $q \notin F_1$

$\delta(q,\varepsilon) = \delta_1(q,\varepsilon) \cup \{q_1\}$ when $q \in F_1$

$\delta(q_0,a) = \phi$ when $a \neq \varepsilon$

$\delta(q_0,\varepsilon) = \{q_1\}$

Proof:



A^* contains ε, for any A.

1) Added ε transitions from accept states of N_1 to the start state of N_1 (q_1)

2) Unless q_1 (start state of N_1) $\in F_1$, ε is not accepted. We add a new start state q_0 and have ε transition

So let me just in this particular closure proof, I want to just write down the what is q, what is F, what is transition function, etcetera. So, as already mentioned, N_1 is this NFA that recognizes A or N_1 is one of the NFAs that recognizes A and our goal is to construct N which recognizes the A star and $(Q_1, \Sigma, \delta(A_1, q_1), q_0,$ F1)are the five tuple corresponding to N_1. And for N the things corresponding things are Q, sigma, delta, q_0 and F. So, what is Q here? Let us try to write down what is Q, Q is nothing but the only new state that is added is q_0, q is nothing but q union q_0, where q_0 is the new start state. And what is F? If you see all the old start states or all the old accepting states are still accepting states. In addition q_0 is also an accepting state. So, F is also, so q is Q_1 union q_0, F is F1 union q_0 too. So in the case of F is also the newly added start state is an accepting state in addition to the existing accept states.

Now, let us consider the transition function. So, the sigma is the same, start state is already done, the only thing that is remaining is the transition function. So, now, let us consider all the transitions. So what is delta in terms of the existing transition function? So, let us first consider q, a, the pair where q is some state and a is some symbol which is not the empty state. So in the existing transitions will be the same as the existing transition delta_1{q, a} when q is in Q_1, where it is an existing state, not the newly added state and A is not epsilon. Because whatever is not epsilon the only new transitions that we added are the epsilon transitions. And what if, what would (q, ϵ) for the existing states q?

So, if you were in an accepting state you are adding this new epsilon transitions if you are not accepting states, you do not have to do anything. So this is also delt A_1, q, a, when q is not in F 1 and a is not epsilon, this is also the same. But when q is in F1, q is in an accepting state. When q is in F1, what is it? So, when q is in F1, what is q, epsilon? So whatever. So, it is possible that from this accept state, there were already some epsilon transitions to some other states, so some other epsilon transitions may have existed. So now we are adding a new epsilon transition. So, what we have to say is, whatever was already there delta_1{q, ϵ} ∪ Q_1, where Q_1 is the start state, old start state. And so this completes all the transitions of the existing states.

So we first wrote all the transitions for all the states except epsilon transitions, then epsilon transitions of all the states except accept states, and then the epsilon transition of the accepts states that is the three ways we split the thing. The first one is all states, but not epsilon transitions. Second is non accepting state of origin of machine, but not epsilon transitions, then accept states and epsilon sorry I did not need this is an error. First is all states, but not epsilon transitions, then epsilon transitions of non-accepting states and then epsilon transitions of accept states.

Now, what remains is the only the transitions of the new state q_0, so that is also very simple. So, when what is delta{q, a} ? This is actually the empty set when a is not epsilon, because the only transition that is there is the epsilon transition and when a is epsilon, it is nothing but the set Q_1, there is only one outgoing arrow and that is it, this completes the construction and hopefully, the correctness is evident in this case and as it was in the other three cases, other two cases and that completes the proof that regular languages are closed into the star operation.

So, now you see, we saw that regular languages are closed under union, concatenation and star and both are all the three groups are fairly simple and fairly intuitive. And you could understand the proof and even the correctness by just looking at the figures of the pictures of the proofs. So of course, we can formally define it and I urge you to go through the formal proof which is there in the book, which we gave for the third case, but not we did not give for the first two cases.

So using the power of NFAs we are able to show that regular language the closure union concatenation and start, we also saw closure under complement earlier. So, that completes lecture 9. In next lecture, we will see yet another characterization of regular languages using expressions. Thank you.