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

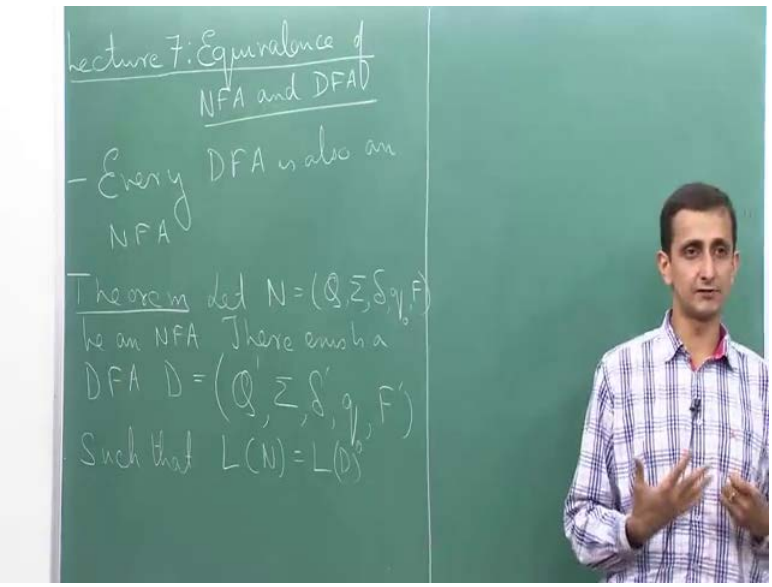
Lecture – 07
Equivalence of NFA and DFA, Closure properties

Welcome to the 7th lecture of this course. So far, we have defined a Deterministic Finite Automaton. We have also seen a generalization of finite automaton, in the sense that we allow computation to happen to go from one state to various states so this gives us a non-deterministic finite automaton. But, the question is that how much powerful or how much extra power does that add? In the sense that, whatever languages that can be computed by an NFA can we also construct a DFA for these languages?

Of course, whatever a DFA can compute an NFA can also compute, because NFA or a DFA is just a special type of NFA, where we do not use its non-determinism; where from every state comma symbol pair, computation goes to a unique state. But, on the other hand given a language that is expected by a NFA can we also construct a some DFA for that language? This is the question that we will try to answer and we will answer this affirmatively, by actually giving a construction. We will give a language accepted by an NFA, for that, we will give the construction of a DFA that accepts the same language.

Let us begin.

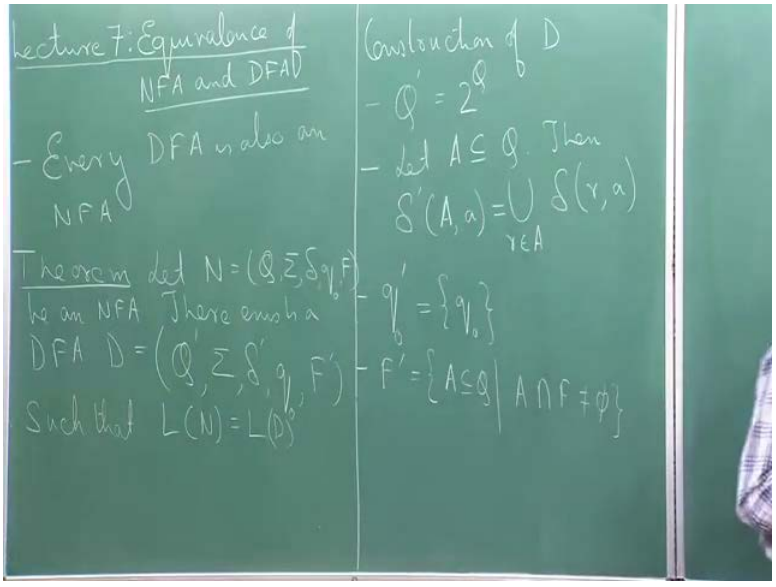
(Refer Slide Time: 01:53)



As I said that every DFA is also an NFA. It is just a NFA which does not use its non-determinism. So, what we want to show today is that, I will state it as a theorem. Let N equals Q sigma delta q naught F be an NFA; then there exists a DFA D; let us give names to its tuples. D equal to Q prime; the alphabet stays the same. So, we have the same alphabet sigma for the NFA and the DFA; Q prime sigma delta prime q naught prime and F prime, such that, L of N equals L of D. Basically, for every NFA, there is an equivalent DFA that accepts the same language. This is what we are going to show for, show next.

Let us look at the construction. So, how do we give this construction? Before I give the formal construction let me give an intuitive idea as to how the construction works. So, recall how an NFA actually performs its computation. In an NFA, from a state comma symbol pair we can go to multiple states. I can think of the set of states that I can go to in each step as some element of the power set of the set of states of the NFA. Basically, at each step the NFA goes to a subset of its states and a subset of Q is basically an element of the power set of Q. I will define the set of states of DFA to be the power set of Q and accordingly, I will define the transition function in a deterministic manner that given an element of that power set. It takes the symbol of the alphabet and in the next step, it goes on to some other element of the power set, that is defined accordingly. So, this is how we are going to define the DFA D.

(Refer Slide Time: 05:15)

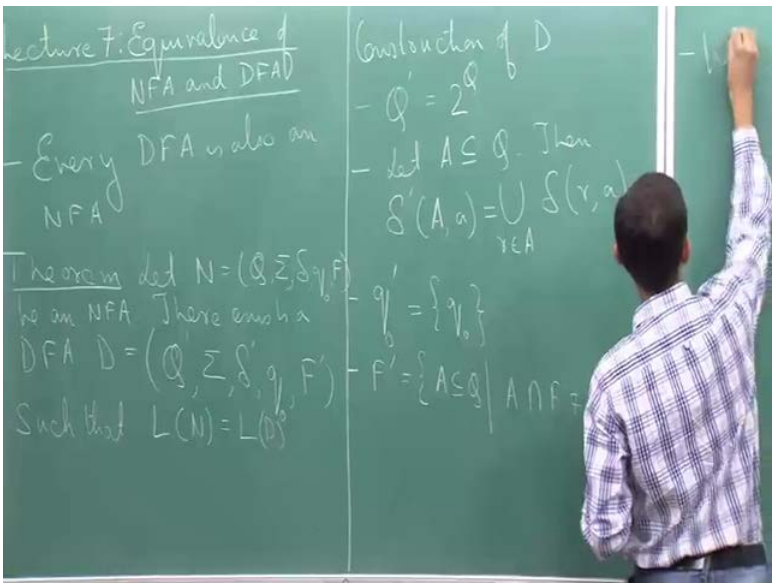


Let us look at the construction. As I said firstly, the set of states of the DFA are all possible

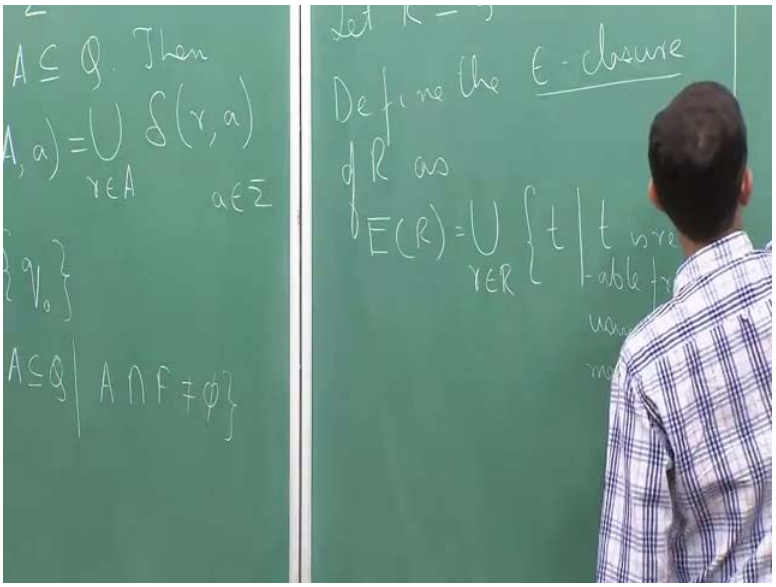
subsets of Q. So, that is my set of states. I will come to delta prime in a moment. I will define delta prime. So, delta prime of; let A be a subset of Q then, delta prime of A comma small a, I will just come to this definition in a moment, but let us look at the other things. Q naught prime, so which subset corresponds to the start state of the DFA. So the start state corresponds to the subset that contains the unique start state of the NFA. In other words, it corresponds to the subset which contains q naught. Again, observe that this is basically an element in the power set of Q. F prime is the set of all subsets. Let me write it as A such that, all A subset of Q such that, A intersection with F is not the empty set. Basically, I look at all subsets which contain some final state. So, that is my F prime. And now, let us come to the delta prime. So, how do we define delta prime? So, delta prime of A comma a. Basically, from a set A, I want to go to all possible states that I can go to, from each and every element of my set capital A, ok.

I look at it as the union of every r belonging to capital A, of delta of r comma a. So, for every state r belonging to the set A, I look at the subset of states that I can go to, on seeing the symbol a, and now I take the union over all possible r s. So, the subset of Q that I get, that is the subset that I will go to, from the set of states capital A, on seeing the symbol small a. Now, there is one small thing about this construction, that is, what about epsilon transitions.

(Refer Slide Time: 09:05)



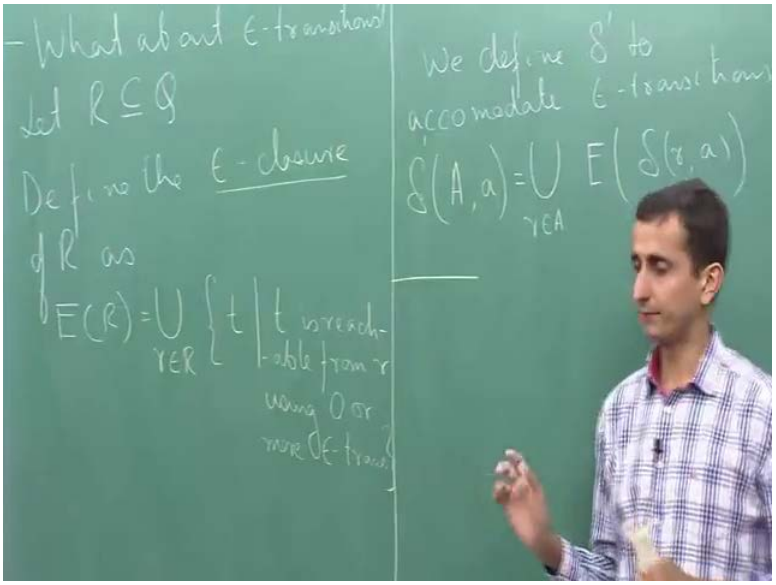
(Refer Slide Time: 09:28)



In this construction, we have not defined what happens if, let us say, this small a is, is the, is the epsilon symbol. Here, we are only looking at small a, where small a belongs to an element of sigma. So, to take care of epsilon transitions, we define what is called the epsilon closure of a set. Let R be some subset of Q; define the epsilon closure of R as, so this is denoted, we will denote this as E of R. So, this is basically a set of states. So, the epsilon closure of a set of states is all the states that I can go to, by taking one or more or zero or more epsilon transitions.

So, for every element r belonging to R, for every r belonging to capital R I look at the set of states; call it t such that t is reachable from r using zero or more epsilon transitions. I look at basically, from every state r, I look at, I try to see if there is a path from r to t which consists only of epsilon transitions and these paths can have zero epsilon arrows in them. In other words, it does contain the state r itself and I take the union over all small r belonging to capital R. Now, with this definition I redefine this.

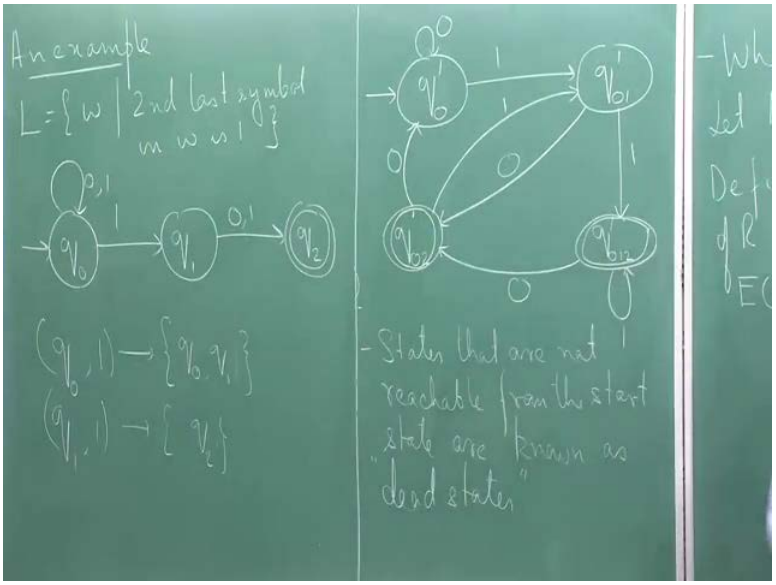
(Refer Slide Time: 12:28)



So, we define delta prime to accommodate epsilon transitions. So, delta prime of capital A comma small a is union over r, r belonging to A of the epsilon closures of delta r a. So, pick a state r belonging to capital A; look at the set of states that I can reach from r on the symbol small a; and now, look at its epsilon closure; look at the set of all states that I can go to using epsilon transitions after there; and now, take the union over all such r s. This is how we define the u delta prime.

This definition actually takes care of epsilon transitions in our NFA as well, ok. I am not going to give a proof of why this construction is correct I mean, I will just leave it at this. If you are interested, you can try to give a proof yourself or you can also read a proof of this construction in any text book on theory of computation. So, what I will do next is I will to understand this construction little better; I will give an example of how this construction works.

(Refer Slide Time: 14:47)



Let us look at an NFA. So, consider the language L, which consists of strings w, such that, the second last symbol in w is 1. So, what would be an NFA for this? We saw a similar construction last time. We have our start state. Let us call it q zero, on a zero or 1 I stay at q zero, if I see a 1, I move to q 1 and then, if I see a zero or a 1, I move to q 2 which is also my accept state. This is an NFA which has 3 states. Now, let us try to construct the equivalent DFA for this NFA. So, the way I will construct the corresponding DFA, I will not list out all the 8 states.

This NFA has 3 states; the power set will contain 2 to the power 3 that is 8 states. Instead of listing out all the 8 states, I will do them in a, in a phased manner I will only write down a state when I require it, ok. So, what would be the start state of the DFA? The start state of the DFA, by definition, is the state which contains q zero. I will just denote it with, let us call it, let us call it zero. So, this is my start state. Now, what happens at q zero? At q zero, if I see a zero, I stay at q zero. I stay at only the subset q zero. I will do a similar thing here. I will just change this a little bit. So, just to avoid confusion, let me call this state q prime zero. If I see a zero, I stay at q zero; what happens when we see a 1? So, when we see a 1, we go to 2 states, q zero and q 1.

Simultaneously in other words, we go to the subset q zero comma q 1. I will denote that using the notation q prime zero 1. So, this is the subset which contains zero and 1. Now, let us look at this subset. What happens from q zero 1, when we see a zero? So, once again we have, recall that, we have to take the union. So, from q zero, when we see a zero, we go to q zero; from q 1, when we see a zero, we go to q 2. If I take the union, it is q zero q 2. So,

from q zero 1, if I see a zero, I will go to the state q prime zero 2. Once again, from zero on a zero, I go to zero; from 1 on a zero, I go to 2 hence it is zero 2.

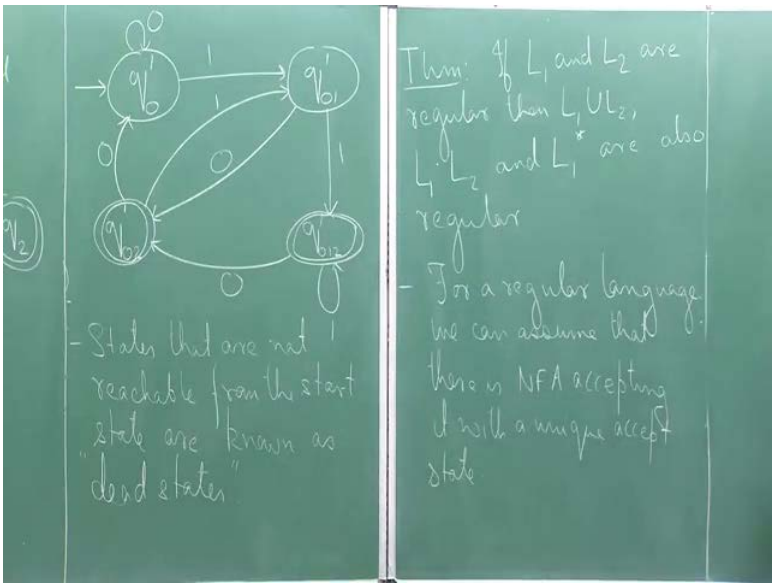
What about from q zero 1 on a 1. So, from q zero on a 1, let me write it here. So, from q zero, if I see a 1, we go to q zero and q 1; and from q 1, if I see a 1, I go to q 2. Now, if I take the union of these 2 sets, I get q zero q 1 and q 2. So, that state is not present here. I will create a state corresponding to that. So, q primes zero 1 and 2; and from q zero 1, I go to this state on the symbol 1. Now, I need to draw transitions for these 2 states as well. From q zero to, if I see a zero. So, observe that from q zero, if I see a zero I go to q zero, from q 2 if I see a zero, I do not go anywhere. It is the empty set. Therefore, the union of the empty set and q zero is only q zero.

Hence, from q zero 2, if I see a zero, I go to q prime zero. What about 1? From q zero if I see a 1, I go to q zero and q 1 and from q 2, if I see a 1 I go to the empty set. So, the union will give me the following that from q zero to on a 1, I go to q zero 1. So, that completes the states q zero prime, q zero 1 prime and q zero 2 prime. Now, what about q zero 1 2 prime? Again, let us analyze. So, from q zero on a zero I stay at q zero, from q 1, on a zero, I go to q 2; and from q 2 on a zero I go to the empty set. Basically, from q zero 1 2 on a zero I will go to q zero 2 prime.

Similarly, on a 1 from q zero, I go to q zero and q 1; and from q 1, I go to q 2. Therefore, from q zero 1 2, on a 1 I stay at q zero 1 2. What are the accept states? So, the accept states will be, all states that contain some accept states. So, the only accept state of my NFA is q 2. So, this will be an accept state, because, this contains a 2 and this will be the other accept state. Now, you can see that, although, if I look at all possible 8 subsets of zero, 1 and 2, I also get other subsets, I mean, I also get other states like, for example, q. So, we have zero, we have zero 1, we have zero 2 and zero 1 2. So, there are other subsets like, only 2 or only 1 or the empty subsets; but, we do not bother about those subsets, because, those subsets are not reachable from the start state.

These are the only 4 states that are reachable from the start state and this gives the complete DFA. Let me mention it here, states that are not reachable from the start state are known as dead states and we do not need to show them in our construction of the DFA. So, this is the example.

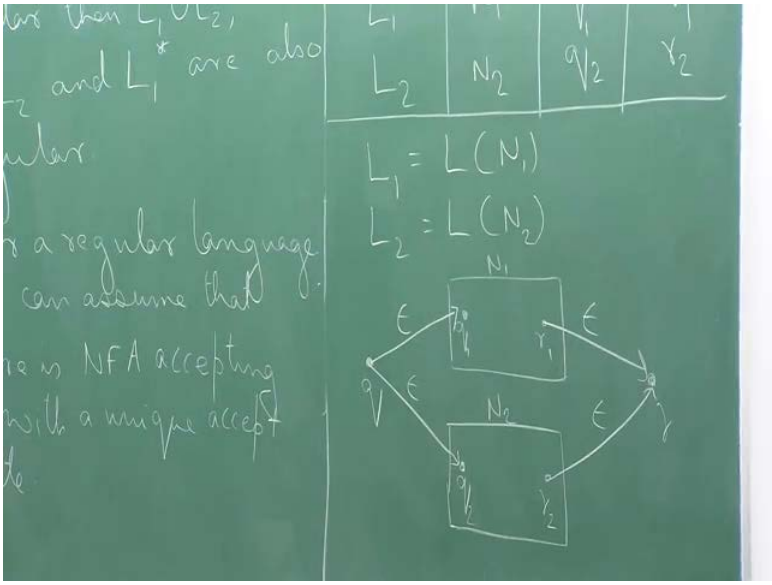
(Refer Slide Time: 23:25)



Now, let us come to, let us come back to the topic of regular operations. So, we have defined the 3 regular operations union concatenation and star. We will see why these operations are called regular operations. In other words, what we will show next is that. Let me state this. If L 1 and L 2 are regular, then, L 1 union L 2, L 1 dot L 2 and L 1 star are also regular. So, this is not very difficult to show, once we make use of the fact that, a language is regular, if and only if, there is some NFA for it also. Our actual definition of regularity was by a DFA; but, because we have just shown that DFA and NFA are actually equivalent. Therefore, for every regular language, we can assume that, there is a corresponding NFA for it.

In addition, what we can also assume that, there is an NFA, which has a unique accept state. The reason why we can assume this is, because, if there is an NFA with multiple accept state, I can create a new accept state and put epsilon transitions from all the old accept states to this new accept state. So, for a regular language, we can assume that, there is a NFA accepting it with a unique accept state. Let us look at the proof of this theorem. So, the proof is again not very difficult and it follows through construction.

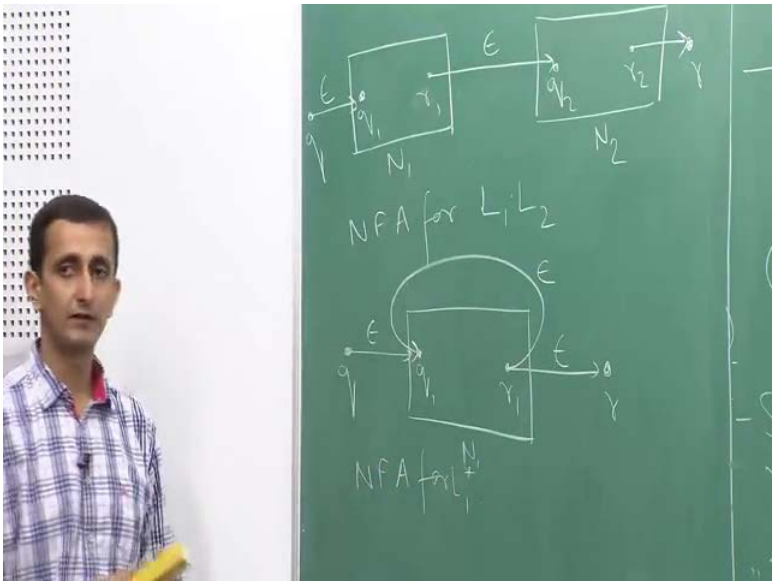
(Refer Slide Time: 26:40)



So, L 1 is regular. So, for the language L 1, let N 1 be a NFA that. In fact, let me just write it as a table. So, we have our language L 1. So for language L 1, let N 1 be the corresponding NFA for it, which has start state, let us call it q 1 and accept state r 1. Similarly, for the language L 2, let N 2 be an NFA that accepts it with start state q 2 and accept state r 2 . So, to be more precise, L 1 is equal to L of N 1 and L 2 is L of N 2. So, how can we construct an NFA for L 1 union L 2. So, observe that, if I construct an NFA. Let us say that, this is my NFA N 1 with q 1 and r 1 and this is my NFA N 2, with q 2 and r 2.

If I create another NFA which has, let us say, start state q and accept state r and I add epsilon transitions from q to q 1, q to q 2 and from r 1 to r and r 2 to r, every string that is accepted by N 1, I can basically take the computation from q to q 1, to r 1 to r and every string that is accepted by N 2, I can take the computation via q to q 2, from q 2 to r 2 and then from r 2 to r. So, every string that is there in one of the 2 languages is there in this NFA. This is the NFA for L 1 union L 2, which proves that L 1 union L 2 is also regular.

(Refer Slide Time: 29:47)



Similarly, for concatenation let me just state it here. So, this is the NFA for L 1 union L 2. Similarly, for concatenation, we have N 1 with q 1 and r 1 and we have N 2 with q 2 and r 2. I add a state q with epsilon transition from q to q 1. I add an epsilon transition from r 1 to q 2 and I add a state r with epsilon transition from r 2 to r. So, every string that can be written as a concatenation of 2 strings, such that one is accepted by N 1 and the other is accepted by N 2, is accepted by this NFA. So, this is basically, an NFA for L 1 concatenated with L 2.

Finally, for L 1 star, if I, let us say this is my NFA for L 1, with q 1 and r 1. I create a new start state q and I create a new accept state r; and I put an epsilon transition, first from r 1 to q 1; I put an epsilon transition from q to q 1 and from r 1 to r. So, look at any string that is there in L 1 star. If a string is there in L 1 star, it can be written as a concatenation of strings, each belonging to L 1. So, each of those strings, I can accept, by going from q 1 to r 1, for the first string; taking epsilon transition back to q 1; again, going from q 1 to r 1, for the second string and so on.

Similarly, every string that is accepted by this NFA can be written as a concatenation of strings each belonging to L 1. Hence, that string belongs to L 1 star. Therefore, this is a NFA for L 1 star. This proves that the languages L 1 union L 2, L 1 concatenated with L 2 and L 1 star are also regular.

Thank you.