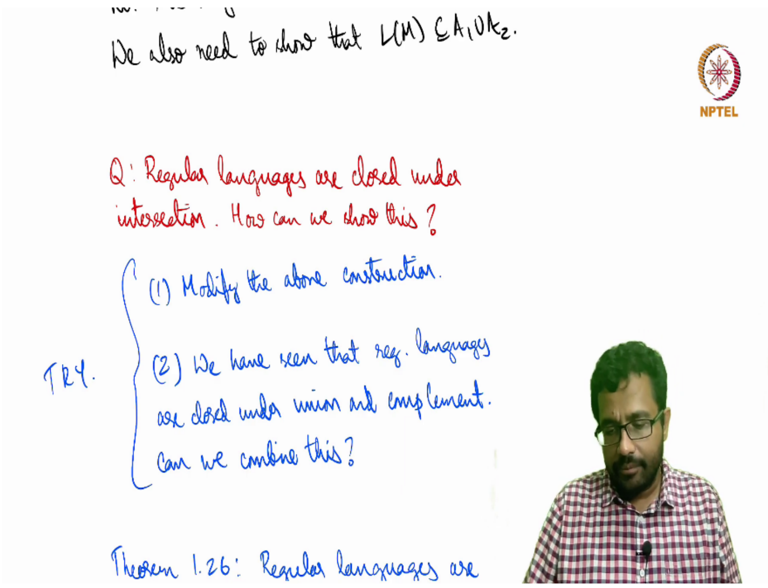
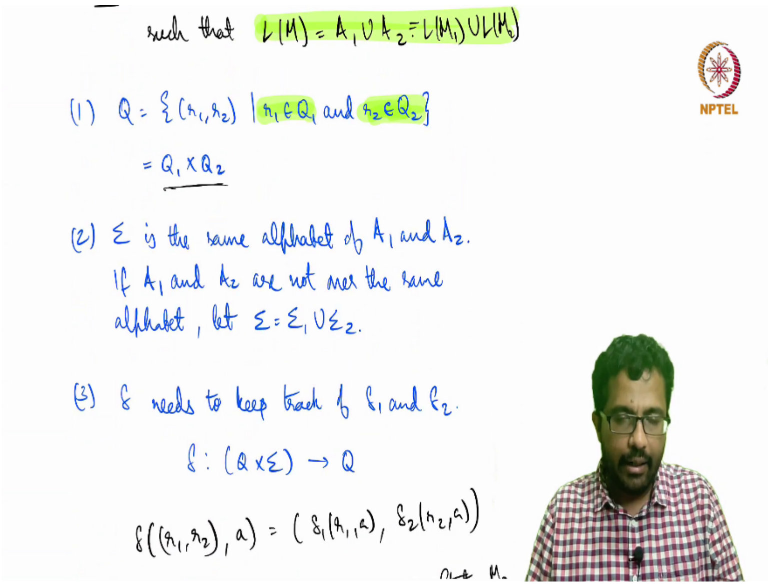
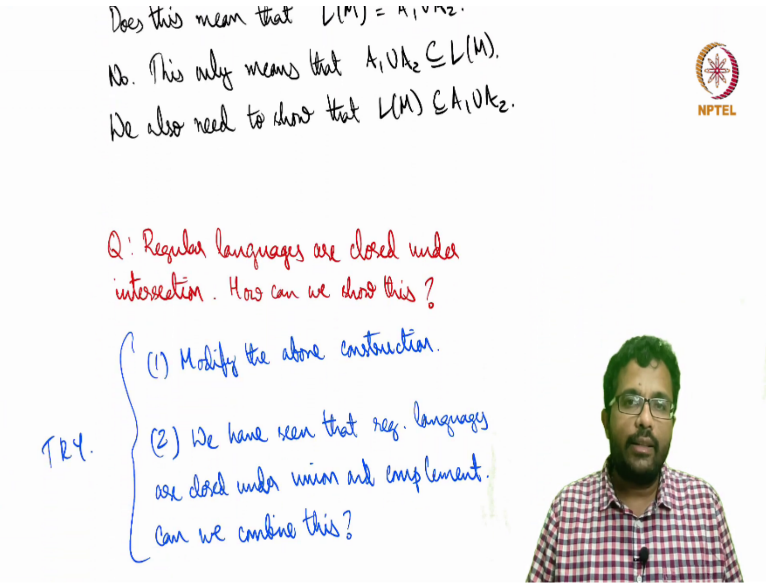
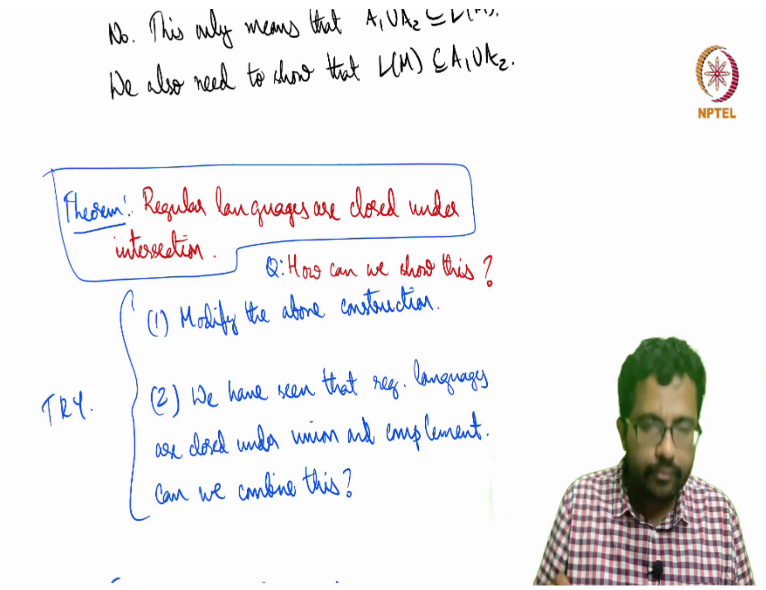
Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Science and Engineering
Indian Institute of Technology Hyderabad
Closure Properties of Regular Languages Under Union, Concatenation and Kleene Star Operation – Part 2

(Refer Slide Time: 00:17)







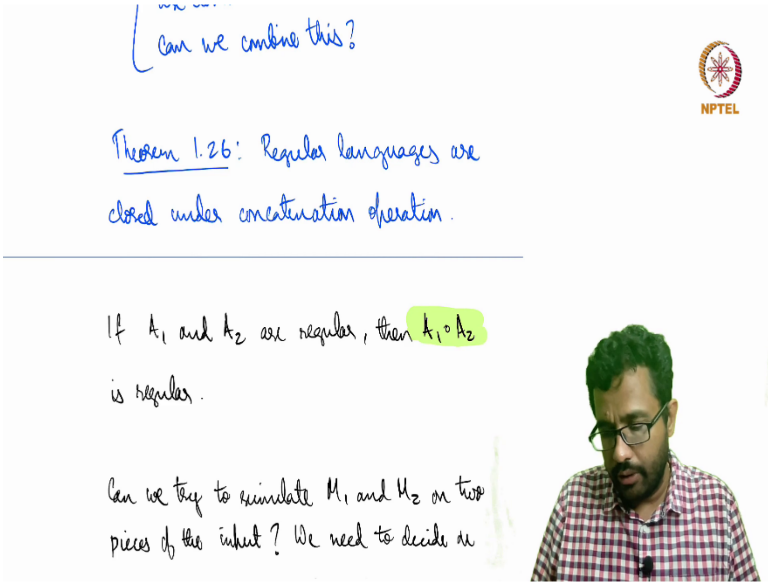


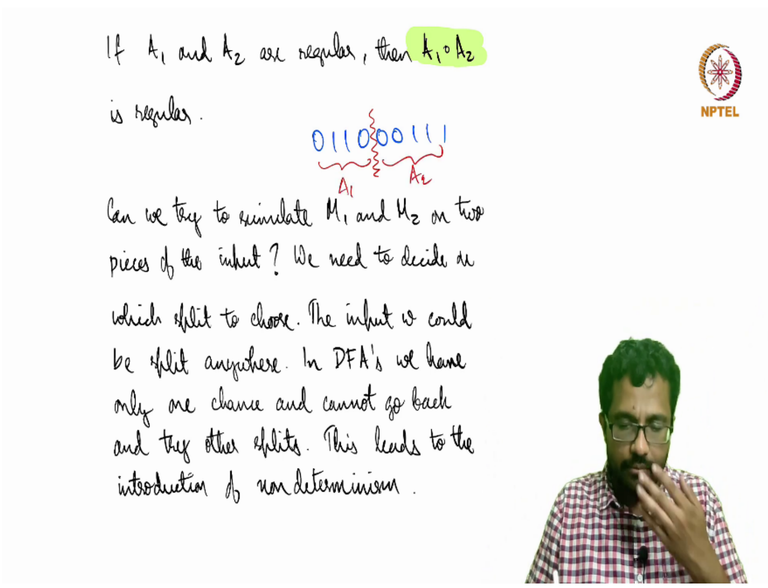
So, the next question that I want to talk about is whether regular languages are closed under intersection. How can we show that? So, there are 2 ways, 1) We can modify the above construction to the same Cartesian product construction, we can modify it to get a DFA that recognizes the intersection language. So, we want to show that if A1 and A2 are regular then A1 intersection A2 is also regular and 2) is something simpler. We have seen that regular languages are closed under union and complement. Can we combine this?

So, I want you to just think about both of these. Neither of these are not that difficult, it is fairly straightforward. First one is we constructed a DFA M that accepts the union language or that recognizes the union language. Can we modify this construction to make a DFA that accepts the intersection that is one and second if we have already seen regular languages are closed under union and the regular languages are closed under complement. Can we somehow combine these 2 inferences to get that regular language that is closed under intersection?

Meaning suppose A1 and A2 are there, we want to get a DFA that recognizes A1∩A2. Suppose A1 and A2 are there, we know that there is a DFA that recognizes a union, we know that there is a DFA that recognizes the complement of A1 complement of A2 and complement of something else. Anything else that is also regular. So, can we combine this to infer that regular languages are closed under intersection? The question is how can we show this? This is the theorem.

(Refer Slide Time: 03:45)

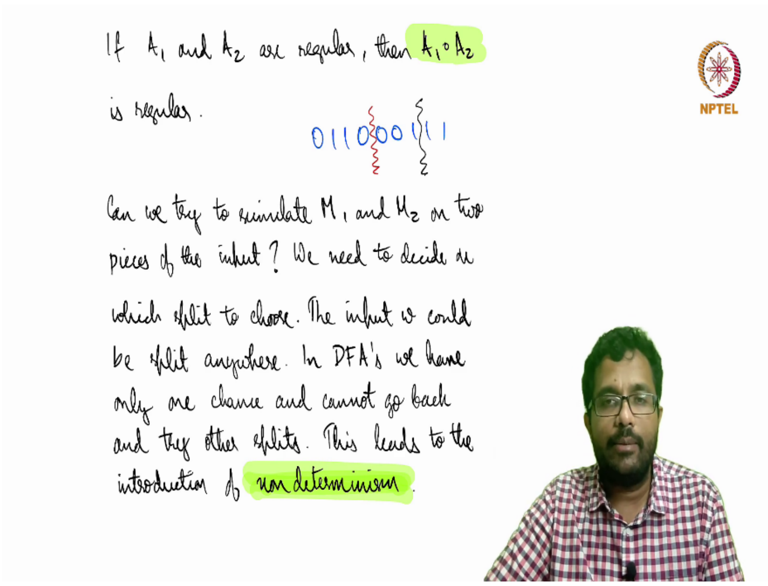




And the next thing, so we saw, we said that there are 3 regular operations. One is union, one is concatenation, and one is star. So, the next thing to try is to show that regular languages are closed under concatenation operation. Meaning if A1 and A2 are regular, then the concatenation A1⊙A2 is also regular. So, the natural thing or the immediate thing to try now that we are on the they are successfully shown that union is regular language that closed under union is to now we have an M1, which recognizes A1, M2, which recognizes A2.

How can we combine this somehow? So, suppose we get a string, let us say 0110011. Suppose we get a string like this. Now, we can try for instance, maybe 0110 is part of A1 and 00111 is part of A2. And we verify that 0110 is part of A1 by M1. So, we just check with 0110 is accepted by M1 and 00111 is accepted by M2 and both of them accept then we accept. So, then this is a concatenation. This part is from this part belongs to the first part belongs to A1 and the second part belongs to A2 and then we accept it.

(Refer Slide Time: 05:31)

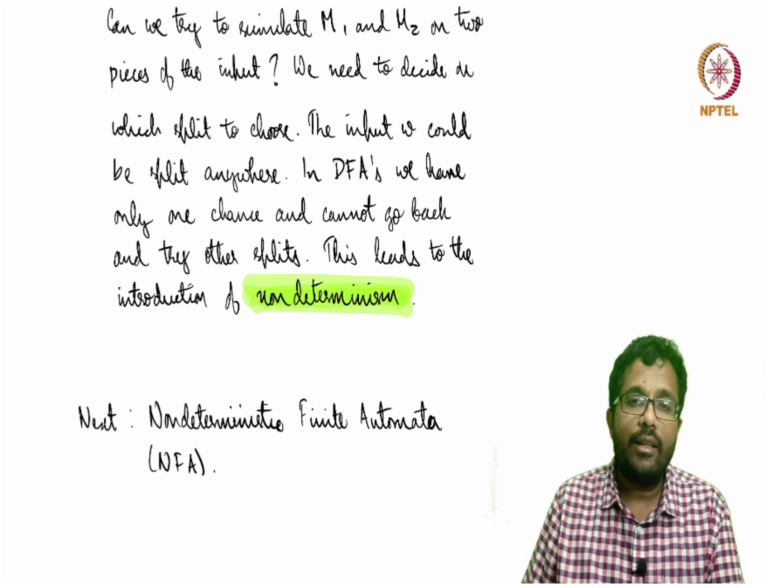


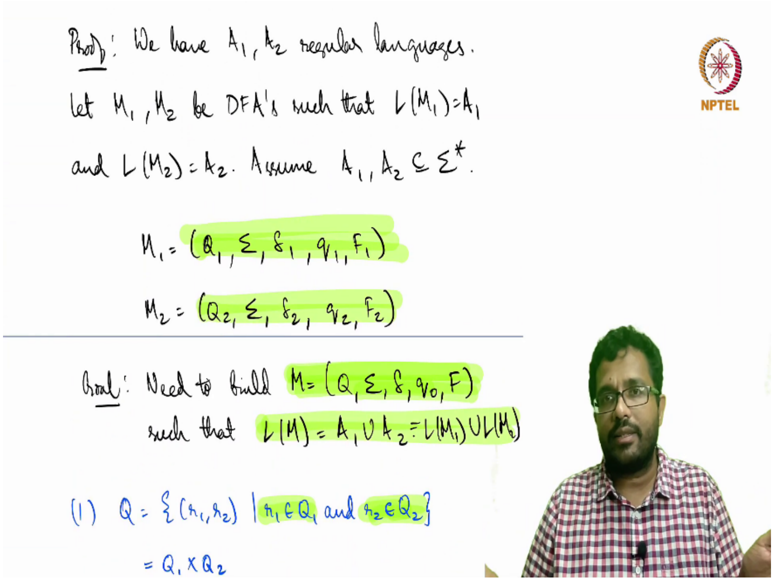
But, what if but, what if we tried something else? What if we tried something like we tried breaking the string like this, instead of the first 4 and the last 5 we try to feed the first 7, 0110001 this we check whether it is accepted by M1 and then we check whether 1, 1 is accepted by M2. Perhaps these are not accepted by the respective or at least one of the respective DFA's. Then, how can we infer that, how can we conclude that. So, we want a DFA that tells us whether that accepts all the concatenations.

So, I am seeing 01100100111 is a concatenation of A1 and A2 because the first part is in A1 the first 011001 is in A1 and 00111 is in A2, but when you just give a string like this, then the combined machine M does not know where to split it, it does not know where which prefix was from M A1 and which suffix was from M A2. So, and there is no way to kind of encode that information also because it could, it is possible that it is part of multiple such combinations also.

But again, like we saw in the first attempt of trying to show that trying to show that regular languages are closed under union we cannot try one split and again try go back and try another splits in DFA you just have one chance to check whether the string is accepted by the DFA or not, we do not have we cannot go back and try again. So, that kind of forces us to think of other approaches. So, it turns out that we cannot do this easily in a deterministic machine or in a DFA. So, this requires us to think of other concepts. And that leads us to the next concept called non-determinism.

(Refer Slide Time: 08:06)

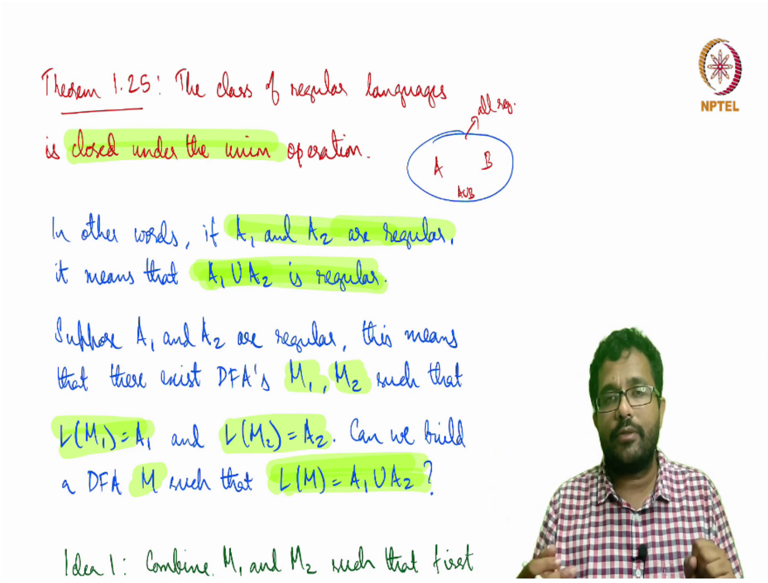


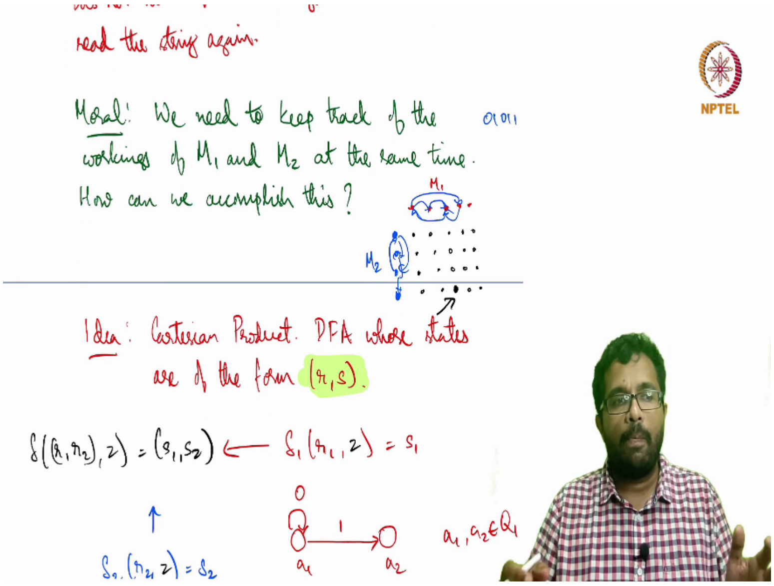


So, the next thing we will see is another type of automated called non-deterministic finite automata or abbreviated as NFA. So, it is just like the same as deterministic finite automata. But, instead of deterministic, it is the opposite of non-deterministic. So, you may recall that I said that in a deterministic machine, given a certain state and let us say and a certain symbol 0, you have only one destination to go to, but now with the non-deterministic, non-deterministic finite automata there could be multiple such outgoing arrows with the same symbol from the same state.

This is one of the, this is just one of the things that is different about non-deterministic finite automata, more we will see in the next lecture. So, even given a state and a symbol below there could be multiple options and there are choices and this leads to more confusion and also adds more power also. So, let me just summarize what we just saw in this lecture.

(Refer Slide Time: 09:22)



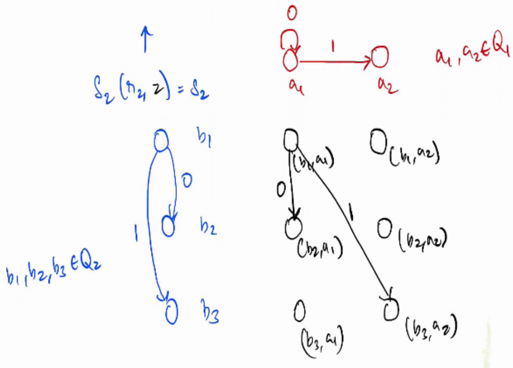


are of the form (r,s).

$$\delta((r_1,r_2),z) = (s_1,s_2) \leftarrow \delta_1(r_1,z) = s_1$$

$$\uparrow$$

$$\delta_2(r_2,z) = s_2$$



Proof: We have A_1, A_2 regular languages.

(1) $Q = \{(r_1,r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

$= Q_1 \times Q_2$

(2) Σ is the same alphabet of A_1 and A_2.
If A_1 and A_2 are not over the same alphabet, let $\Sigma = \Sigma_1 \cup \Sigma_2$.

(3) δ needs to keep track of δ_1 and δ_2.

$$\delta : (Q \times \Sigma) \rightarrow Q$$

$$\delta((r_1,r_2),a) = (\delta_1(r_1,a), \delta_2(r_2,a))$$

(4) Starting State: $q_0 = (q_1, q_2)$ — Starting M_2.

$$\delta : (Q \times \Sigma) \rightarrow Q$$

$$\delta((r_1,r_2),a) = (\delta_1(r_1,a), \delta_2(r_2,a))$$

(4) Starting State: $q_0 = (q_1, q_2)$ — Starting M_2.

Starting of M_1

(5) Accepting States: We accept w if it ends at (r_1,r_2) if either $r_1 \in F_1$ or $r_2 \in F_2$.

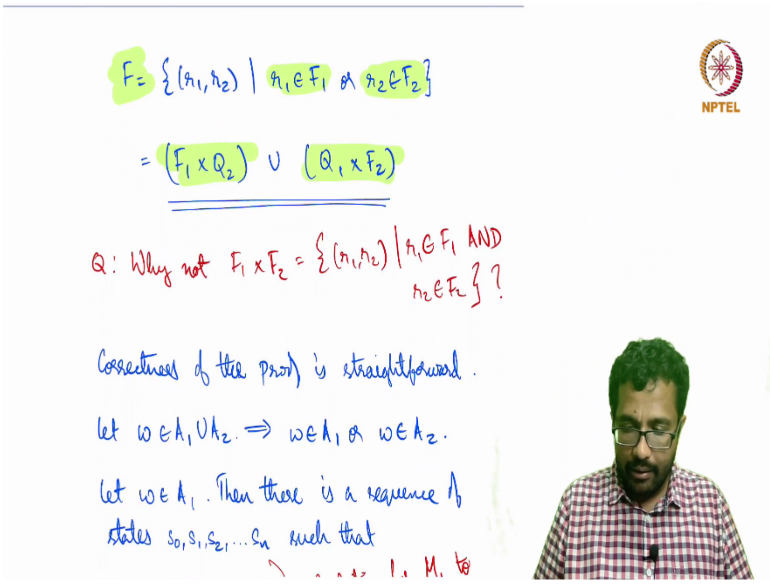
$$F = \{(r_1,r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$$

$$= (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

So, we want to show that a class of regular languages is closed under ∪ operation, we made this DFA which is a Cartesian product. So, suppose M1 is the DFA of A1 and M2 is a DFA of A2, we made a DFA M which replicates the M1 and M2. So, if M1 has, let us say 10 states and M2 has 4 states, that machine the DFA M has 10 times for 40 states. So, it keeps track of where, what M does in a certain string and what M1 does it is a certain string and what M2 does with a certain string as well.

So, by having this kind of grid kind of structure, it faithfully reproduces the transitions as well. So, if you just look at which column it is in, you will see where M1 takes the string 2, if you just see which row it is in, you will see where M2 takes that string 2, and the rest is kind of straightforward. You have (Q1 x Q2) number of the states and the transitions are as I described, the starting state is the state which is combined by the starting state of M1 and starting state of M2 and accepting states are the set of states where either r1 is in the accepting state of M1 or r2 is in the accepting state of M2. So, notice that there is an ‘or’ not an ‘and’.

(Refer Slide Time: 10:58)



$\underbrace{\quad}_{\in F}$ $\underbrace{\quad}_{w \in L(M_2)}$

Q: Is it enough to show that for all $w \in A_1 \cup A_2$, w is accepted by M?

Does this mean that $L(M) = A_1 \cup A_2$?

No. This only means that $A_1 \cup A_2 \subseteq L(M)$.

We also need to show that $L(M) \subseteq A_1 \cup A_2$.

$\boxed{\text{Theorem: Regular languages are closed under intersection.}}$

Q: How can we show this?

No. This only means that $A_1 \cup A_2 \subseteq L(M)$.

We also need to show that $L(M) \subseteq A_1 \cup A_2$.

$\boxed{\text{Theorem: Regular languages are closed under intersection.}}$

Q: How can we show this?

TRY.

(1) Modify the above construction.

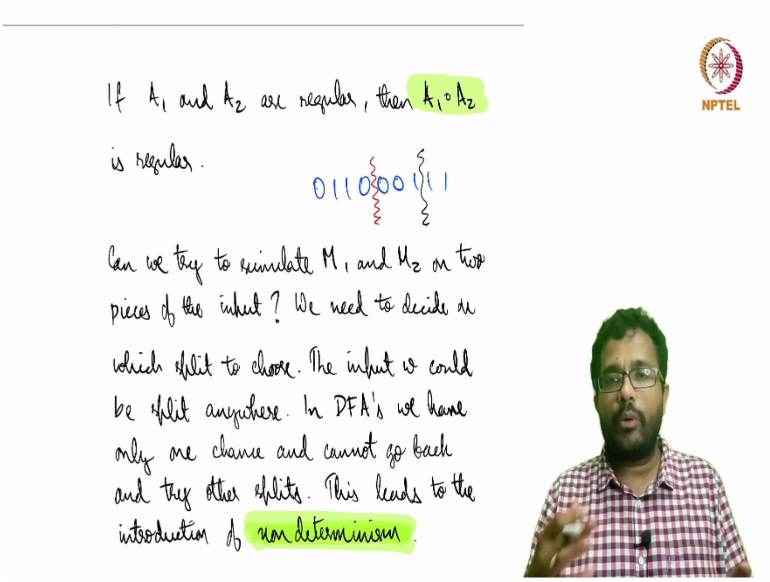
(2) We have seen that reg. languages are closed under union and complement. Can we combine this?

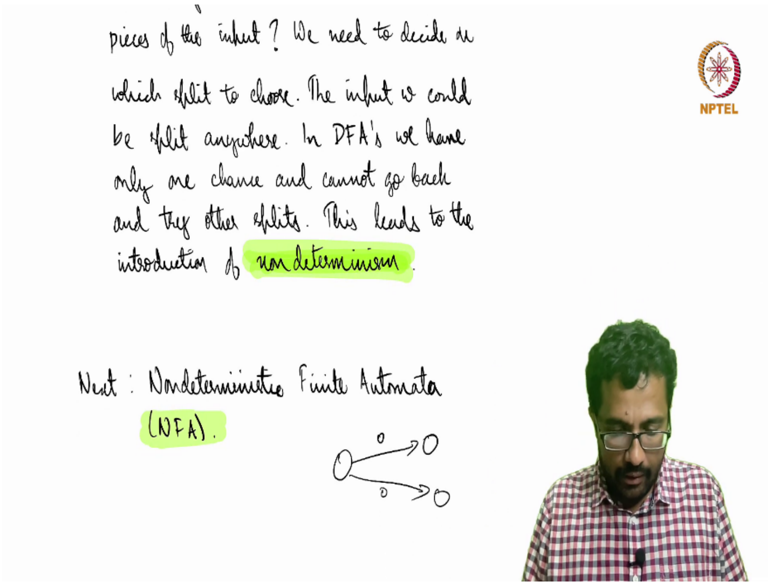
Theorem 1.26: Regular languages are closed under concatenation operation.

If A_1 and A_2 are regular, then $A_1 \circ A_2$ is regular.

01100011

Can we try to simulate M_1 and M_2 on two pieces of the input? We need to decide on which split to choose. The input w could be split anywhere. In DFA's we have





And I asked what will happen if it is an and that is for you to think through, and the proof is fairly straightforward. I also kind of remembered that it is not enough to show that. So, what we showed is the set for all w that is in the union w is accepted by M, this just establishes that A1 ∪ A2 is in the language recognized by M to be for the proof to complete we also need to argue that the language recognized by M contains nothing more, meaning if A1 if there is a string that is not in the union it is not accepted.

The other thing that I mentioned is regular languages are closed under intersection. And then I stated the fact that regular languages are close under the concatenation operation. And then we kind of saw that the strategy that we followed so far in trying to build a DFA does not quite work. And that motivated us to use that to motivate the introduction of non-determinism and non-deterministic finite automata. And we will see non-deterministic finite automata in the next lecture. So that is it for lecture number 5. Thank you.