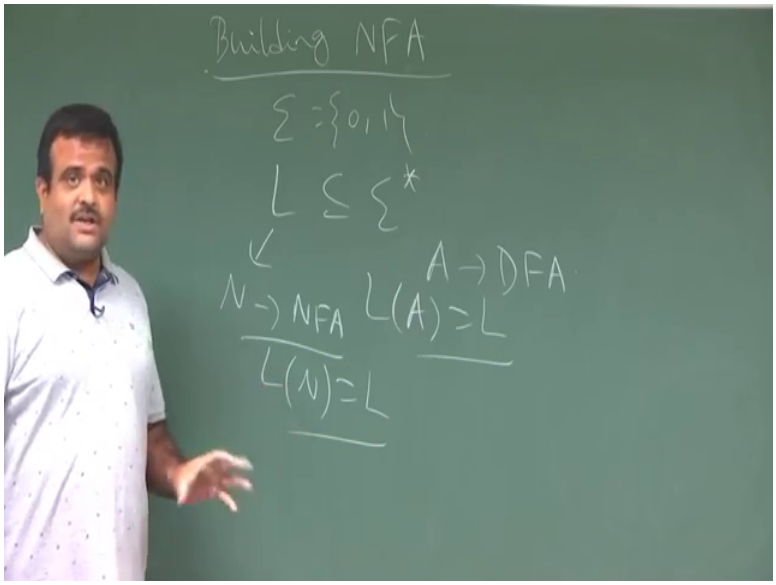
Introduction to Automata, Languages and Computation
Prof. Sourav Mukhopadhyay
Department of Mathematics
Indian Institute of Technology, Kharagpur

Lecture – 09
Equivalence of DFA's and NFA's

So, we are talking about NFA and we are talking about the language accepted by NFA and in this class will see the equivalence between the DFA and NFA. So, now, before that we want look at the how to build a NFA, given the language we want to see whether we can construct a NFA. So, we know that if a language is, given a language if we can have a construct a DFA that is called regular language, this is same I mean if we can have a NFA also and I will see the NFA is easier to build then the DFA. So, to come back equivalency we have to will just discuss how to build a NFA, building NFA is easier than building a DFA.

(Refer Slide Time: 00:57)

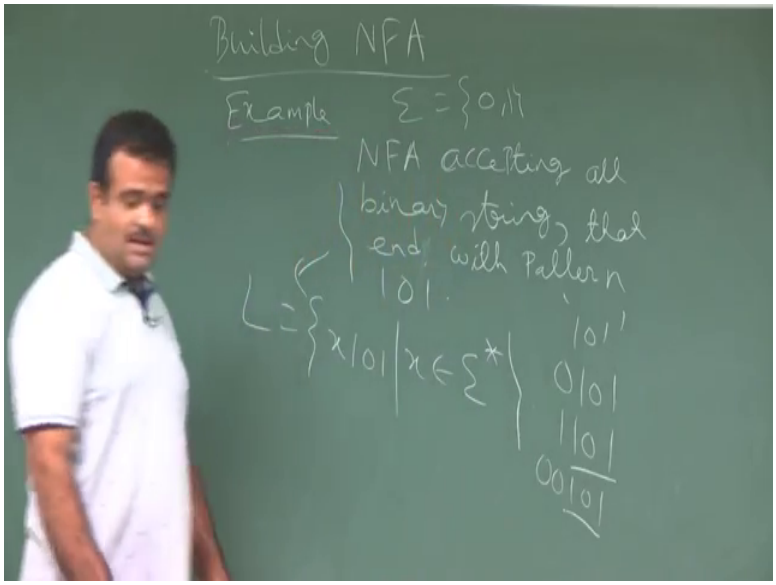


So, suppose we have a sigma, this is I mean for example, maybe 0, 1. Now it could be any anything else also, now suppose we have given a language. Now to check this language is regular or not earlier what we are doing? We are doing that we are trying to construct a DFA A such that the language of the DFA is L; that means, all the string from this L is accepting this. Now again, now we are going to try to build a NFA, given a language we try to build a NFA such that this is L, then also this is called a regular

language because this is the finite automata and will see the DFA or NFA are equivalence. Given a NFA you can construct a equivalence DFA which is accepting same the language of that two are same.

So, and the constructing building NFA is easier than building a DFA will take, will discuss this through an example.

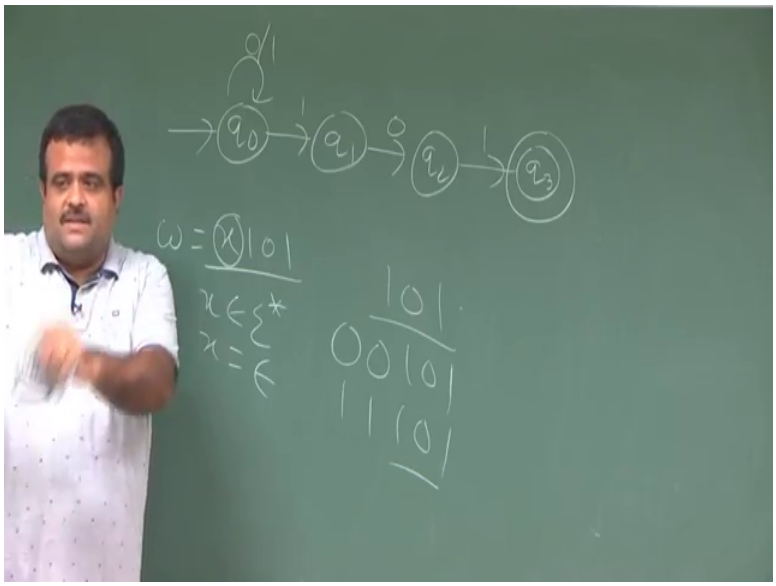
(Refer Slide Time: 02:26)



Suppose we need to build a NFA, which is NFA accepting all binary string strings having the pattern that ends with the pattern string. So, end with the pattern 1 0 1. So, here sigma is 0 1. So, what is this L? This L we should write as this so, this x 0 1.

So, this is our language and we want to see whether this language is a I mean we can have a corresponding NFA which is which is accepting this, I will see the NFA is easier to construct than the DFA ok. So, now, let us try to construct a NFA which is accepting this type of string which is just ending with 1 0 1 the last string is 1 0 1 like 1 0 1, 0 1 0 1, 1 1 0 1. So, last three strings are so 0 0 1 0 1 like this. So, last three strings are 1 0 1. So, now, the question is how to construct a NFA.

(Refer Slide Time: 04:22)

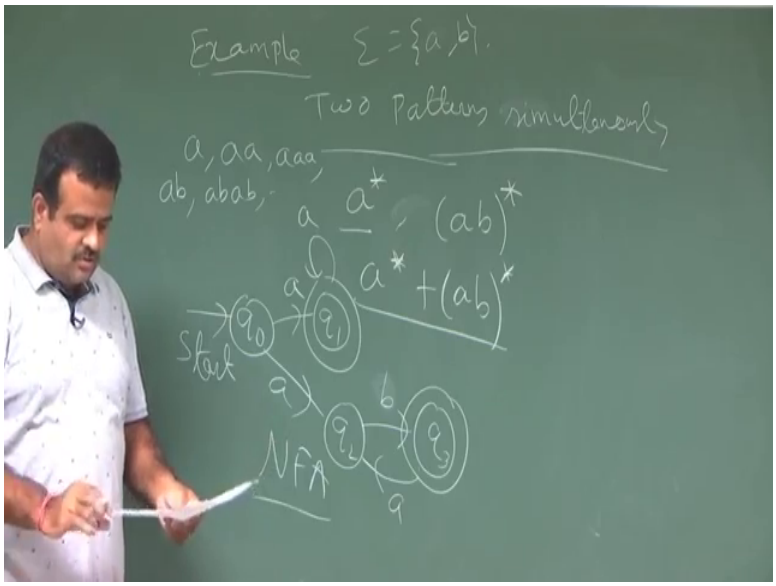


So, we want last three strings are, last three beta 1 0 1; 1 0 1 nah yes ok. So, we have to start with the state q. Now if we see a 1 we will go to here because here we do not need to mention that like deterministic way we do not need to have that delta so, that is the advantage. We can have non deterministic way; even we may not have the move from one state with some input. So, that is the advantage of building a NFA. So, if you see a 1 will reach here then if you see a 0 will reach here. So, slowly we are moving to the something like final state. Now, if you see a 1 will go to the q 3 this is the final state. So, this is one branch I mean if you have just 1 0 1, now we can have.

So, this is accepting 0 1 0, but if we have something like 0 0 1 0 1 or 1 1 or just 1 0 1 like. So, for that what we do? Will simply do is hoping over a head, that is all very easy because that will take care of this x we need to accept all the string like this w. So, this x is, will be take x is coming from. So, it could be, if it is empty then it is directly going there, if it is not empty let us hop there. So, this is the way we can. So, it is very easy construct DFA on this it may not be that much easy to construct.

So, that is why if the building NFA is easier. Now suppose if you can now in now for next will see the equivalence between DFA and NFA, given a NFA we can have a DFA which is accepting the same language. So, then we have we can say this language is a regular language ok. So, we showed that let us take one more example of building NFA.

(Refer Slide Time: 06:41)



Suppose we have the say two patterns are simultaneously coming 0 1, we have two or it can be a b I mean instead of 0 1 we can have a b.

So, suppose you have a language we have two patterns are coming simultaneously. So, like this is denoted by a star is basically if it is it could be so or a b star, a star or a b star. So, this is basically this is what is for regular expression will talk about this in more details. So, it is expecting like this.

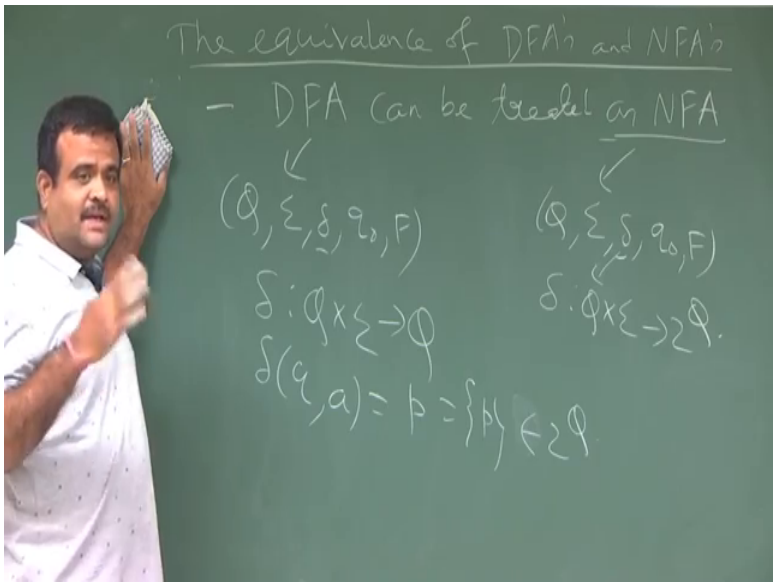
So, if it is a or if it is even null string also it is accepting. So, what is the corresponding NFA then? We can have starting state then with a we can go to q 1 and with a we can go to q 2 and this is if want if you want to accept this type of string.

So, we can hop you had if you have a. So, this is one of the final state and then we can have a another state q 3 this is also final state if you say a b ok. Now if you see a form here after this so, a b then we need to have b also. So, we go a here like this. So, these are the, this is the NFA which is accepting this type of regular expression this is called regular expression we will discuss in more details on this. So, simultaneously it is coming. So, it is expecting the string like a, a a, a a a this is one branch which is accepting or it is accepting a b, a b a b like this.

So, a b a b like this so, this two types of string this accepting. So, by the regular expression way we can write it as a plus the union, this is one language this is another

language you can take the union we will we will discuss this type of thing in more details. So, we will talk about the regular expression. So, now, we will move to the equivalence between DFA and NFA. So, given a NFA how we can construct a corresponding DFA which is having accepting the same language.

(Refer Slide Time: 10:00)



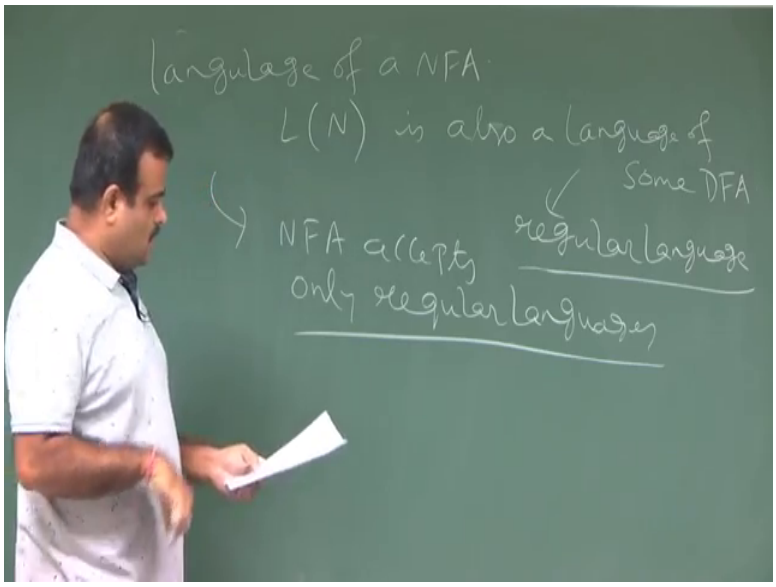
So, that is the next the equivalence of DFA's and NFA's. So, now, first of all first observation is a DFA can be treated as NFA.

So, DFA can be treated as NFA, NFA is more general. So, what is DFA? DFA is Q sigma delta q 0 F, NFA is also five tuple Q sigma delta q 0 F only there is a difference in delta. So, here delta is a function from Q cross input alphabet to Q, it is a fix single term, single term set and this delta is a function form Q cross sigma to power set. But this is a single term set, but this also can be I mean if you have a a it is going to some p. So, p can be written as set form. So, this is also subset of q. So, this is also belongs to two to the power q although it is going to a single form set we have only one option, but that can be treated as a NFA.

So, the DFA is can be treated as a NFA this one observation, another observation if we have a given NFA, if the language accepted by NFA then there will be a corresponding DFA which is accepted the same language. So, then so, that we have to show that equivalency. So, if you can show that when the regular language means the language for which we have a DFA and if you can show these two are equivalence. So, this if we can

have a NFA because NFA is easy to construct, easy to build then we can have a this regular language for regular language we can just construct, try to construct a NFA. Because we know the DFA and NFA are equivalence, I mean that we have to prove.

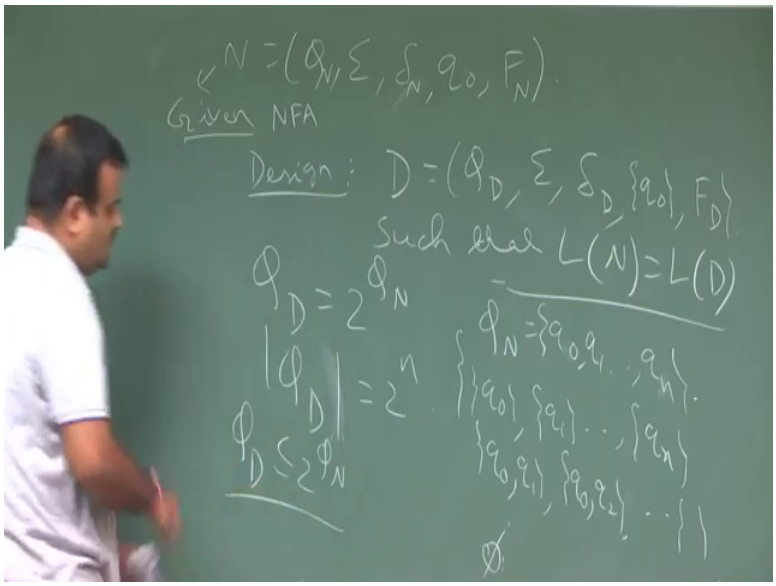
(Refer Slide Time: 12:38)



Language of a NFA that is L of N is also a language of some DFA, also a of some DFA. And we know that language of some DFA is called regular language, this is the definition you have use if we have a finite automata and that finite automata could be a NFA also. So, that means, we can say NFA accept regular language. So, if your language is called regular if it is accepted by a NFA.

So, and that is also true because we have this equivalency. So, that part we have to show. So, NFA accepts only the regular languages. So, now, let us go to the main part of it, the equivalency because NFA is easy to construct. So, to check whether a language is regular or not sometimes DFA is difficult to construct, but if it is a regular there has to be a DFA, but NFA is easy to construct as we have seen in the last example. So, if you can construct a NFA then it is regular, because the NFA is having the corresponding DFA which is accepting the same language ok. So, let us try to construct the given NFA to the DFA.

(Refer Slide Time: 14:43)



So, suppose you have given a NFA, given a NFA. So, N which is Q sigma delta q 0 F. So, this is the state this is the delta n will denote this by N to indicate this. So, this is the N. So, from here we are going to given NFA, from here we are going to have a DFA which is accepting the same language. So, out of construct a DFA let us design like this is the construction. So, D or sometimes you refer as A anyway Q D sigma will be same, delta D will be different the tangential rules will be different and this is.

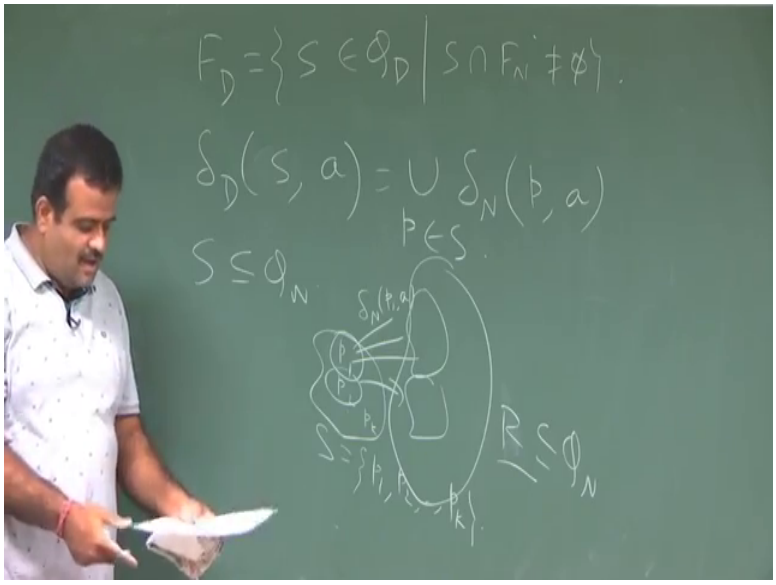
So, starting state is same, but this is and say this and F D would be same. So, yeah so now, such that we want to design such DFA such that language accepted by this N same as language accepted by this is our goal. So, then we say these two are equivalence ok. So, now, what is Q D? Q D will be 2 to the power Q N. So, if Q N is, if we have a some states of this NFA q 0, q 1, q q k say this is say suppose there are k states. So, this is basically the power set of this q. So, setup all possible subsets. So, the what is the cardinality of this? Cardinality if there are n states. So, if there are n states in q.

So, cardinality of this is 2 to the power n. So, set up set up all possible subset of this, these are our single state of the DFA. So, like these are all state q 0 this is single term set q 1, Q N this is all (Refer Time: 17:20) then q 0 q 1, q 0 q 2 two together like this, three together like this. So, these collection and also the empty said through these collection is called the power set and this power set is nothing, but our view states of this DFA ok, this

is the way how we are define this. So, if this is having n state then this DFA is having 2 to the power n state.

But all state may not be we do not need to consider all state. So, you just consider those states which are reachable from which are accessible from q 0 by the NFA. So, there will be some states, some subset which is not accessible from q 0. So, we will just discard that; so, will take an example on that. So, that is why will write this as subset of 2 to the power Q N because we will just discarding all those states which are not accepted accessible from the from q 0. So, now, this is the, now what is F D?

(Refer Slide Time: 18:34)

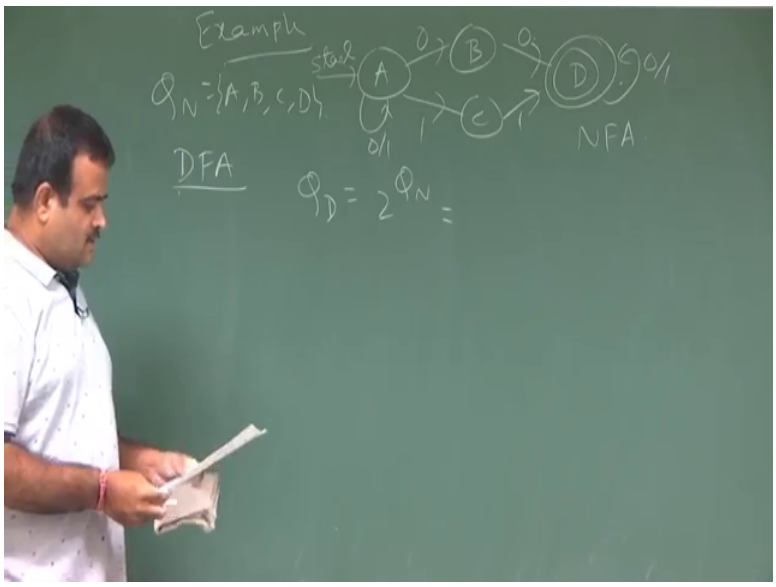


So, F D is the final state. So, this will be the set of subset Q D because Q D I mean the states is the subsets such that it is intersect some with the F D is null, not null.

So, a is intersects at F N is not null, this is the way we defined because it should have a at least 1. So, we call so this is the subset all state of DFA is a subset. So, in this subset consist of some states at least one state should be a final state. So, those subsets have to be a the final state of, final state of the DFA ok. So, this is the where we construct is and the rule the delta, now we have to define the delta. So, delta D of S comma S is the set sorry S is the set S comma a. So, now, this S is the subset of this Q N delta D of S comma this is nothing, but it is union of delta N of p comma a p belongs to S. So, how you define this?

So, we have a state. So, this is a collection of states we have some p 1, p 2 say p k now this is one state in d this is the subset. So, this is basically p 1, p 2, p k. So, this is the subset of S. So, now how to apply this? Delta on this, delta on this is nothing, but so, this will go to. So, if you apply individual of these is going to solve under these delta N on a it is going to some set it is also a p 2 is also going to some set. So, this collection is basically our again some S R which is the subset of this Q N. So, this collection is our move delta D of S comma a ok. So, this is how we define this thing. So, now will take one example it will be more clear ok.

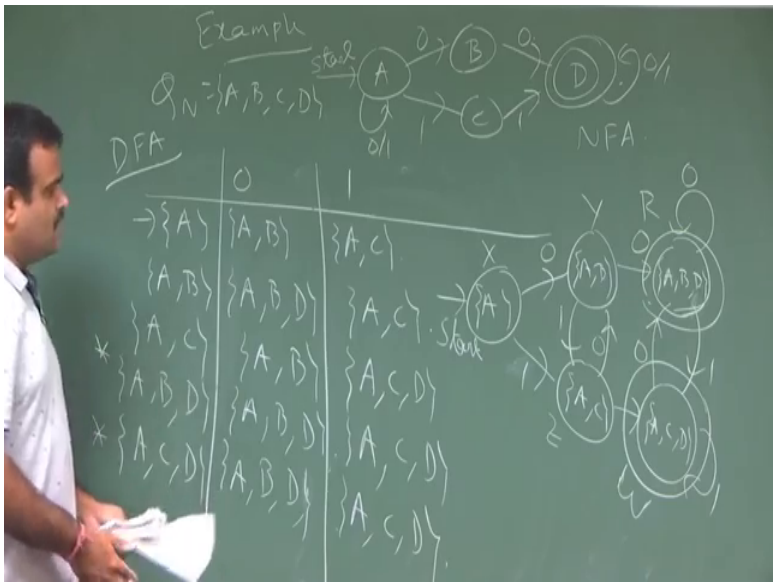
(Refer Slide Time: 21:33)



So, let us take as example suppose we have a NFA like this A. So, this is the starting state and we have B, we have C and we have a final state D and the moves are like this 0. So, this is 0 or 1, this is a NFA. So, we can have non deterministic move. So, this is 0 this is 1 and this one is 0 or 1, suppose this is a given NFA and we want to convert this NFA to a equivalence DFA. So, for these what we need to do? So, this is our NFA now we are going to construct a DFA.

So, for this so what is the Q N? Q N is A, B, C, D now we have to construct also the Q D will be Q D will be 2 to the power Q N; that means, set of all possible subset of this. So, among these will take only those subset which is visible from A ok.

(Refer Slide Time: 23:04)



So, if you do that we can just construct the DFA. So, this is A 0 1, 0 1. So, from A so what we do from A we can go to B and C. So, this B and C with the 0 move know with the 0 move we can go to B and A. So, this is A B will write and now with 1 we can only go to. So, 1 we can go to A and C.

So, this is the way and these are all states in the DFA. So, this is the DFA we are constructing DFA done these are all the function ok. Now, we take A, B from A, B where we can go from A, B if you take a 0 we can go form a we can go to B and from B we can go to D with the 0 move and from A we can go to A itself. So, A, B, D and from A, B with 1 move from a with one move we can go to A or C and from B with one move there is no move. So, this is same as A C now we take A C over here this is another state in the DFA. So, from A C if you take a 0 we can go to B and if you take A. So, this same as A B and if you take from A C, if you take a one move from A we can go to C or A and from C we can go to D.

So, this is nothing, but A C D. So, this way if you continue so, A B D if you take this is the final, one of the final state because it contain D, D as a final state. So, if you work on these which is be A, B, D again and this one A, C, D. So, and then again if you take another final state A, C, D then it is going to A, B, D and A, C, D. So, there are other subset also, but those we do not need to consider because those are not accessible from

A, only these are the subsets which are accessible from A. So, this is our delta now let us write the graph for these.

So, this is our A, this is the starting state this is the DFA. So, now, if we take a 0 move from A we can go to either A, B. So, we can rename this states if we like and with one move we can go to B A C because this is the deterministic. So, you should have a move and from A B if we, from A B if we give A 1 move we come here and if we get A 0 move will go to another state which is from A B; A, B, D A, B, D this is with 0 move this is with the one move and form A C again, if we A C we can go to A B with the 0 move and with 1 move will go to A C D. A C D and from A C, A C D or is A B, A B D if you take a 1 0 will be here and one move will go here. And from A C D if you take a 0 will go to here and if you take a 1 here and these are the 2 accepted state of this DFA.

So, this is the equivalence DFA. So, whatever language is accepted by, whatever string is accepted by this NFA, the same string will be accepted by this DFA. So, this is the equivalency between the DFA and the NFA ok. So, yeah so we do not need to consider the all the state subset because many subsets are there because, these are 4 so 2 to the power 4 because empty state is the subset. But we do not need to consider those because those are not accessible from the starting state, we are only bothering about the language accepting that.

So, we can rename these as X state Y state Z state W state and something like R state. So, we can rename that this thing. So, this is one example to construct a given NFA to DFA. So, every NFA will have corresponding DFA, which is accepting the same language. So, the language of DFA NFA is same as the language of that corresponding DFA so; that means, to check whether a language is regular or not what we do will just try to construct a NFA which is easier to construct then the DFA. So, if you can construct a NFA we can straight away say that language is regular.

Thank you.