Theory of Computation

Prof. Kamala Krithivasan

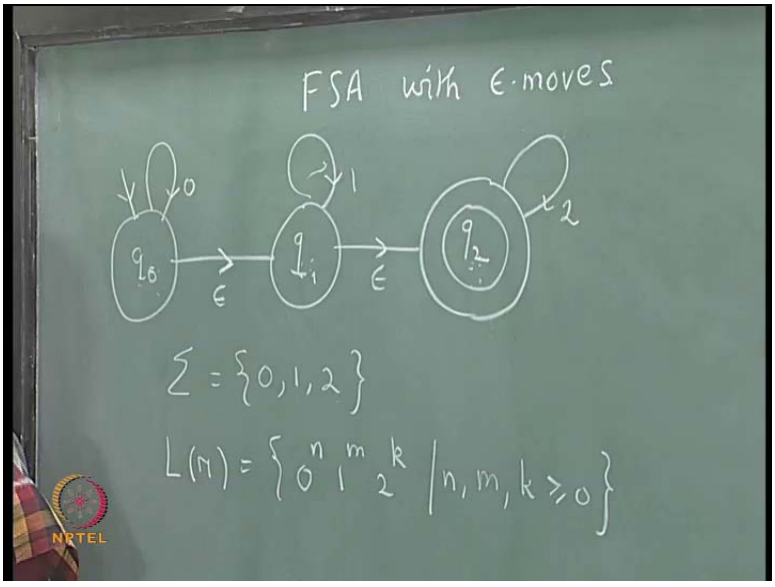
Department of Computer Science and Engineering

Indian Institute of Technology, Madras

Lecture No. # 11

Non Deterministic FSA with Epsilon-Moves

(Refer Slide Time: 00:15)



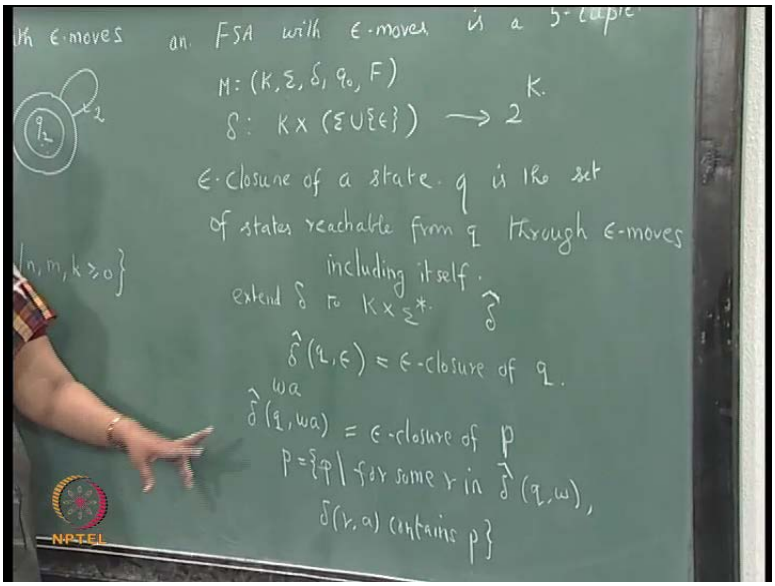
So, today we shall consider FSA with epsilon moves. First we considered deterministic FSA, then we consider non deterministic FSA; non deterministic FSA are equivalent to deterministic FSA, and you can similar it to non deterministic FSA with a deterministic FSA. But the transitions were labelled with symbols from sigma the alphabet, but today we shall also see that there could be epsilon moves.

First let us take an example, look at this diagram. There are three states, q naught, q 1 and q 2. q 2 is the final state and q naught is the initial state. The transitions are as follows, 0, epsilon, 1, epsilon and 2. The alphabet here consist of three symbols 0, 1 and 2. In q naught if you get a 0, you go to q naught and in q naught if you a epsilon or without reading any symbol, you can go to q 1 and in q 1 if you get a 1, you go to q 1 in q 1 without getting any symbol, you can go to q 2, and q 2 if you get a 2, you go to q 2. From the figure, guess, what is the language accepted?

You can see that a string is accepted if it takes you from the initial state to a final state. Whether it is a deterministic automaton non deterministic automaton, if there is sequence of states which take you from a initial state to a final state that string is accepted. So, you see that you can go along is several times then you can move to q 1. Go along this several times go to q 2, go along this several times. You need not read a 0 also straight away from q naught to q 1, you can go and read only ones and two's or you can just read zeroes and two's alone , or you can just read zeroes and ones and go to q 2. You can just read zeroes alone you can just read 1 alone, you can just read two's alone.

So, the language accepted is L of M or L is 0 power n, 1 power m, 2 power k where n m k are greater than or equal to 0. This is the final state automaton with epsilon moves and because we allow epsilon moves you are able to change the state without reading any symbol. You can go from one state to another state without reading any symbol. Adding this facilities, does it increase the power of the automaton. Here there are only one arcs with 0 you may have another arc with 0 also, that is I mean the basically the diagram is non deterministic in nature apart from being non deterministic in nature.

(Refer Slide Time: 03:54)



You also allow epsilon moves. A formally defining FSA with epsilon moves an FSA with epsilon moves is a 5 tuple. M is equal to k sigma, delta, q naught, F same way you define k is a final set of states as usual, sigma is a set of input symbols, q naught in k is the initial, state F is the set of final states. The only difference will come in delta

mapping, delta is a mapping now from k into sigma union epsilon right into 2 power k subsets of k.

Now, how do we define a string accepted in this case? Basically the idea is it should take you from an initial state to a final state, but formally how do we define for that we define what is known as epsilon closure. Epsilon closure of a state you define epsilon closure of a state. How do you define this? The set of all states reachable from it just by epsilon moves. The epsilon closure of a state q is the set of states reachable from q through epsilon moves including itself.

So, in this case what is the epsilon closure of q naught? Epsilon closure of q naught is q naught that set will also be there then from q naught you can reach q 1 by epsilon move. So, it will be q 1 then from q naught t through q 1 you can reach q 2 also. So, the epsilon closure of q naught is q naught q 1 q 2 and what is epsilon closure of q 1? From q 1 you can go to q, 1, q 1 will be included then q 1 by epsilon moves you can reach q 2.

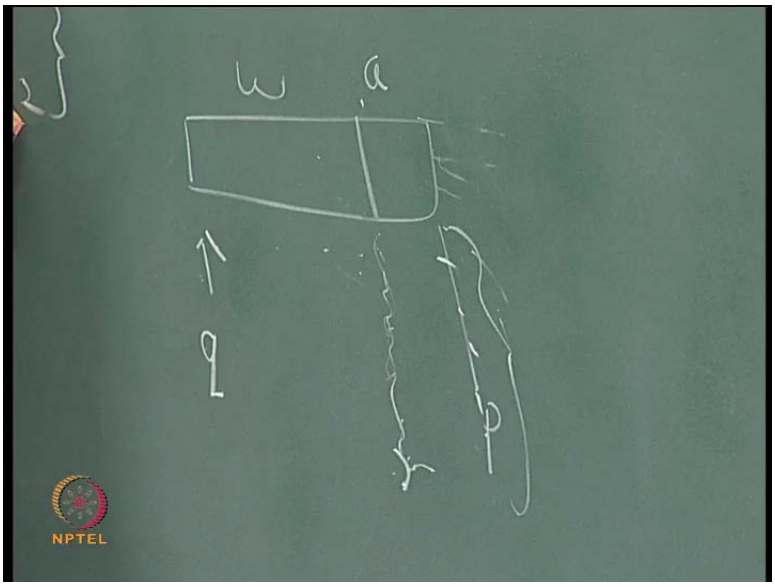
So, epsilon closure of q 1 is q 1 q 2 epsilon closure of q 2 is just itself. Now, we will define, extend delta to k into sigma star. Now you use a symbol delta cap here right, usually when we extend and in the case of deterministic automaton, and non deterministic automaton later on we found that you can use the same symbol delta cap or delta, but here we cannot do that, we will see why. Delta cap q epsilon is epsilon closure of q.

See when you see usually in the non deterministic automaton and deterministic automaton you say delta or cap q epsilon is equal to q isn't it. That is when you are in state q if you are not reading any symbol you will be in the same state, if it is a DFSA or a NFSA, but in the case of final state automaton with epsilon moves. When you are in a state q without reading any symbol you can be in q or you can go to any one of the states in the epsilon closure of q isn't, it just by epsilon moves alone you can go suppose you are in q naught without reading any symbol you can go to q 1 you can go to q 2.

So, delta cap q epsilon is equal to epsilon closure of q now if you have a string w a string w and a symbol a. How do you define delta cap q? w a this you define as epsilon closure of some p, p is the set of states p, here what is p? P is a set of states p small p for some r in delta cap q w delta of r a contains p. You defined extend define delta cap q epsilon is epsilon closure of q, then w a take a string w a how do you define delta cap q w a? What

is essentially means is starting from q after reading w a what are the possible states in which you can be that is what it means.

(Refer Slide Time: 11:22)

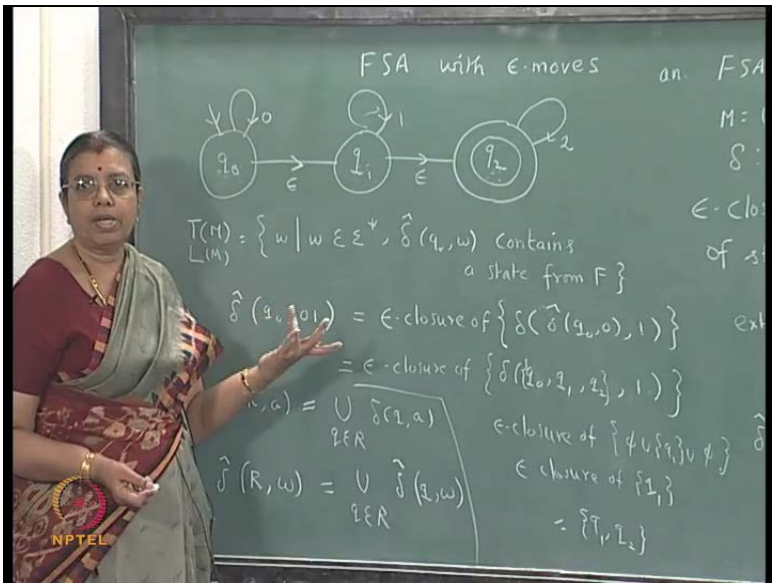


How do you define that it is a epsilon closure of p. Where p this is capital P to make a difference this is the set of states p. What are the states p for some r in delta cap q w delta for a contains p. What is essentially means is supposing you (( )) there is a string w a starting from q after reading w you can be in a set of states that is denoted by delta cap q w. So, r is 1 of them from r after reading a, you can go to some sets of say as p some set certain sets.

Now, from such sets you can also make some more epsilon moves. So, you consider the epsilon closure of these states you can just read epsilon and go to some more states also. All these states will be in the epsilon closure p, p itself will be in the epsilon closure of p. So, starting from q after reading w you can be in a set of states in between you may be making beginning or in the end you may be making epsilon moves r is one of them . So, you consider all the union of all sets of (( )) moves from r after reading a, you can go to the state p, but afterwards also you can make some more epsilon moves.

So, what it means is for some r in delta cap q w delta r a contains p, then afterwards it can make some epsilon moves that is why you write it as epsilon closure of p. So, the language accepted.

(Refer Slide Time: 13:19)



So, this it is easy to remember, I will rub off. So, the language accepted is T of M or L of M sometimes, it's also denoted as L of M. This is the set of strings w, w belongs to sigma star delta cap q naught w contains a state from F, this is the way the language accepted.

Now, you see that for example, delta of q naught 0 in this example is q naught, but what is delta cap q naught 0, starting from q naught. What are the possible states you can reach after reading a 0 you can reach q naught you can reach q 1 you can reach q 2. So, this will be q naught q 1 q 2. They are not the same see in the case of n F s c and d F s c when you restrict it to symbol they became equal that is why you can use the same symbol, but here they are different. So, you cannot use the same symbol delta cap has to be differentiated I mean it is different from delta you have to use two different symbols here they cannot be the same.

Now, let me take delta what is delta cap q naught 01. What does it mean starting from q naught after reading 0 and 1 what can be the states in which you can begin. So, starting from here after reading 0, you can go here after reading 1 you can go here or q 2 that isn't it. So, the answer is q 1 q 2 using the formal notation how do you get this? This is equal to epsilon closure of the set of states delta of delta cap 0, 1 sorry delta of delta cap q naught 0 comma 1 in this way if you write using this symbol formal notation.

So, what will this be? This is epsilon closure of what is delta cap q naught 0 q naught q 1 q 2 all from q naught after reading 0, it can be q naught q 1 q 2. So, this will be delta of q naught q 1 q 2. Now, for a single state we have defined this, when you extend it to multiple states what can you say delta of r a, where r is a set that you by definition you can define this is defined delta of r a is union of q belonging to r delta of q a.

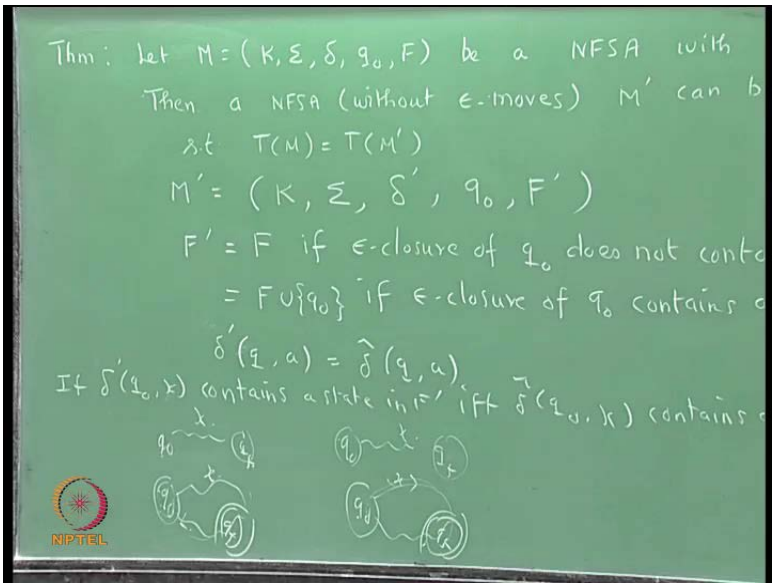
If you have a set from each one what are see you have a set initially you can be in a certain set of states after reading a. What can be the set of states in which you can be in staring from this, what are the states starting from this, what are the states starting from this, what are the states. So, find the union and similarly for strings delta cap rw, r is the set of states from these set of states if you read w what the possible states in which summation can be are. This will be defined as union of q belonging to r delta cap q w.

So, you have extended delta to delta k into sigma star and you called it as delta cap. When you want to extend delta and delta cap to a set of states you extend like this using this definition. What is the epsilon closure of delta of (( )) I did not write that 1 here forgot to write the 1 here. This 1 delta cap q naught w is q naught q 1 q 2 and this 1 is here. So, what is delta of q naught, q 1, q 2 1 it will be delta of q naught 1 union delta of q 1 1 union delta of q 2 1, but you find the delta of q naught 1 is empty delta of q 1 is q 1 delta of q 2 1 is also empty.

So, from these 2, 3 states only 1 of them will lead you to a final one. So, that will be epsilon closure of here it will be 5 union q 1 union 5 it will be like this. So, that will be epsilon closure of q 1 really and what is the epsilon closure of q 1? That is equal to q 1 q 2. So, starting from q naught after reading a 0 and 1 what are the states in which you can be look at the diagram. It is very clear you can be in q 1 or q two, but formally using the definition how would you arrive at that we arrive at it like this right.

Now, it looks as though by adding the epsilon transition the power I mean, you are giving more power right epsilon transitions, but the power is not really increased the same language you will get you will not get anything more than the non deterministic FSA or deterministic FSA. So, the idea is given NFSA with epsilon transitions you remove the epsilon transitions right and get an NFSA without epsilon transitions.

(Refer Slide Time: 21:24)



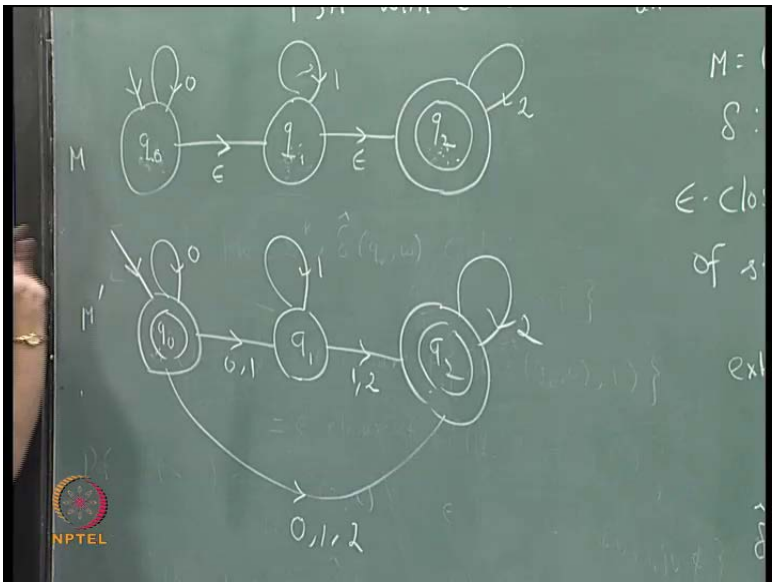
So, how do you go about it? Let M is equal to k sigma delta q naught F be a NFSA with epsilon moves then, a NFSA without epsilon moves M dash can be constructed. Such that the language accepted is the same t of M is equal to t of M dash that is you are trying to remove the epsilon transitions without changing the language.

Now, how is M dash defined? M dash has the same set of states k same input symbols then, mapping you have to define delta dash. The same initial state you have q naught final states will be changed F dash. So, if you do that how do you define F dash how do you define delta dash F dash is F. If epsilon closure of q naught does not contain a state from F, if the epsilon closure of q naught does not contain a state from F, it is the same as F its equal to F union q naught, if epsilon closure of q naught contains a state from F. When will a string epsilon be accepted by the machine a non deterministic FSA without epsilon transition. If the initial state is a final state, but a NFSA with epsilon transition epsilon can be accepted if the epsilon closure of q naught contains states from F. So, in this example see epsilon will be accepted starting from here you can reach a final state without reading any symbol. So, epsilon is accepted by the machine.

Now, if you want to construct if you want to construct an equivalent machine and if it acts to accept epsilon then, you have to make the initial state a final state right. So, just for epsilon alone the difference is made otherwise, it is the same set of final states you have. If want to accept epsilon empty string then, make q naught as the initial state

otherwise q naught as a final state otherwise leave it. How do you define delta dash? delta dash q a, a is a symbol q is the state is defined as delta cap q a that is starting from q. What are the states in which you can be in a after reading after reading a that will be the same here. So, in this example let us define.

(Refer Slide Time: 26:55)



So, let us look at this example, (no audio from 26:36 to 26:52) what do the set of states are the same this is M how do you construct M dash q naught q 1 q 2. Now, this is the initial state and because epsilon is accepted this will be a final state and this will also be a final state. Set of states are this, because epsilon is accepted this is made a initial state and this is already a final this is a final state this is also the initial state this state is already a final state.

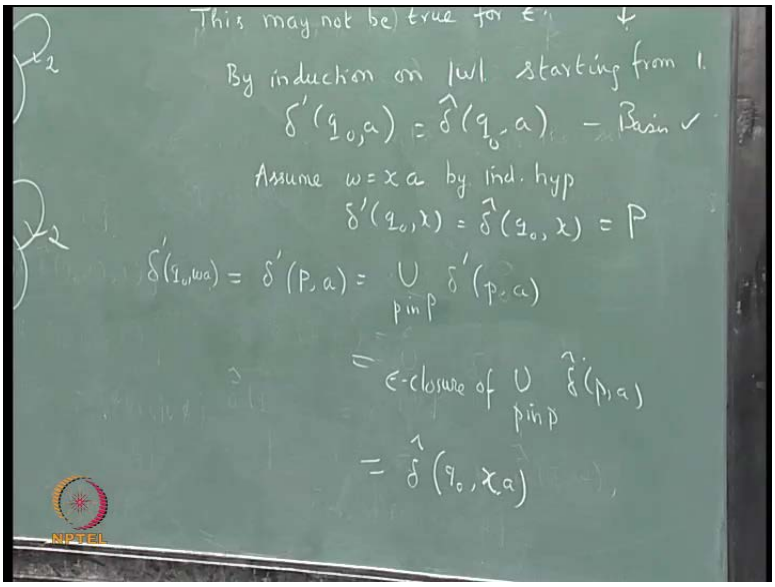
Now, from each one you have to find the arcs corresponding to 0 and 1 and 2. What is delta of q naught delta dash q naught 0 from q naught after reading 0 what are the states in which you can be in q naught q 1 q 2. So, there will be one arc q naught like this, there will be one arc like this, there will be one arc like this. Starting from q naught after reading a 1 what are the states in which you can be after reading 1 q 1 or q 2.

So, from here there should be an arc here. I will mark it like this; that means, from q naught if you read a 1 you go to q 1, from q naught if you read a 1 it should also go to q 2, that is starting from q naught after reading a 1 you can be in q 1 or you can be in q two. So, from here 1 from here you can mark. Starting from q naught after reading a 2

where can you go after reading at 2. So, from q naught there is an arc with label 2 going to q 2. From q 1 after reading 0 where can you go there is no map its empty. So, you need not have to draw an arc from q 1 after reading a 1 where can you go either q 1 or q 2. So, there will be an arc with label q 1 here, there will be an arc with label q 1 here from q 1 if you read a 2, where can you go from q 1 you can go to q 2 and read a 2 isn't it. So, from q 1 if you read a 2 you can go to q 2. So , there's also an arc from q 1 to q 2 with label 2.

From q 2 you cannot read a 0 you cannot read a one, but if you read a 2 from q 2 you can go to two. So, this is the NFSA without epsilon transitions which is equivalent to this the language accepted is the same. Just see 0 power n 1 power n 2 power n can be accepted, because first 1 you can first 0 I mean the any number of 0 you can read 1 0 you can read any number of one's you can read 1, 1you can read and any number of two's you can read. If you do not want to read one's only zeroes and two's starting from here read any number of zeroes and then two you can omit 1 also. So, anything it is a 0 power n 1 power M 2 power k where n M k are greater than or equal to 0.

(Refer Slide Time: 31:46)



Now let us go to the formal proof (no audio from 31:18 to 31:48) we show that delta dash q naught w is equal to delta cap q naught w. In the machine M dash after reading w starting from q naught the set of states is the same as starting from q naught in M and reading a string w this will not hold for epsilon.

This may not be true for epsilon this you have to prove using induction, but for epsilon alone this may not hold. So, induction should start from length 1, you have to prove this by induction on length of w starting from 1 it will not hold for 0 isn't it. Here delta dash q naught epsilon is q naught only is it not? delta dash q naught epsilon is q naught only whereas, here delta cap q naught epsilon is q naught q 1 q 2. So, epsilon that is why if epsilon is accepted you have make the initial state also a final state, start from induction on length 1 you say delta dash q naught a.

It is the same as delta of q naught a, this is by definition how did we define for any q we defined like this isn't it the definition was like this. So, bases class this is the bases class by definition it has been defined like this. So, it is ok now, assume w is equal to some x a and by induction hypothesis by induction hypothesis delta dash q naught x is equal to delta cap q naught x. The result holds up to string of length n this is string of length n plus 1 and so on.

So, assume that the result holds up to string of length n w is the string of length n plus 1 x is length n say a is one. So, now let this is the set of states say some p now what is delta dash p a, this is union of p n p delta dash p a delta dash q naught w a is equal to delta dash p a equal to union of delta dash p a. So, epsilon closure of union of p n p delta cap p a equal to delta cap q naught x a.

Now, you have to show that, if delta of q naught x contains a state in F if and only if delta cap q naught x contains a state in F in F delta dash F dash. Now, starting from q naught after reading x you are in a state q F say here, the same thing will happen starting from q naught after reading x you will be in a state the same state q F this is for any other string, but if it is epsilon if x is epsilon purposefully you are making the initial state a final state if epsilon has to be accepted. So, epsilon if the original machine accepts it will be accepted by this machine, but if x is different from epsilon starting from q naught, because of this result you will see the starting from q naught if you are able reach a final state here also starting from q naught you will be able to reach a final state and starting from here, if you are able to reach a final state here starting from here you will be able to reach the same final state here.

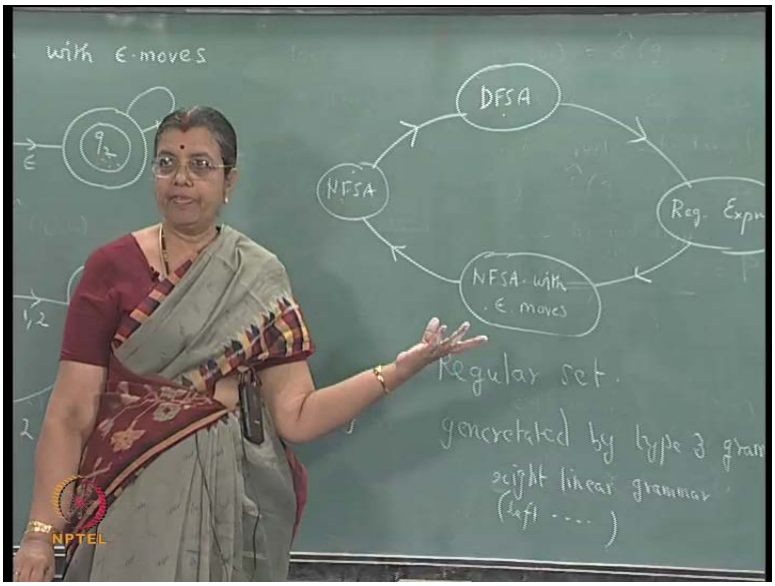
But there should be one difference is in this if you accept a string x starting from q naught by going to q naught and q naught has been made a final state, because epsilon is

accepted q naught is the final state here q naught is not a final state here, but q naught is a final state here. And a string x is accepted here by going from q naught to q naught here also by going here, x you will able to go from q naught to q naught, but here q naught is not a final state.

This particular case you have to distinguish and in that case what happens is how did you make q naught a final state? Because the epsilon closure of q naught contains a final state. So, there will be another state q F and the epsilon closure of q naught will contain that q F, and that q F will also be a final state here. So, starting from q naught after reading x you may go to q naught, but you will be able to q F also here that is the only difference if you are able to go from q naught after reading a x to q naught it will be possible for you to go to q F also starting from q naught.

So, there if a string is accepted by M, it will be accepted by M dash and if the string is accepted by M dash, it will be accepted by m. So, the language of a accepted is not affected. So, we have this result t of M is equal to t of M dash. So, what so, you have the same construction we have followed here. So, in essence we have proved 2 steps in the diagram which we have considered earlier, this is you have DFSA and you have NFSA and you have NFSA with epsilon moves and you have regular expressions.

(Refer Slide Time: 40:23)



We have not formally defined this, but we have already seen what is a regular expression in examples for example, in this one the language accepted is 0 power n 1, power M 2

power k n M k greater than or equal to 0 and this the regular expression will be 0 star 1, star 2.That is the string of zeroes followed by a string of one's followed by a string of now what we have earlier seen is given an NFSA. How to construct an DFSA.

This is by what is known as a subset construction and given a NFSA with epsilon moves. How to construct a NFSA without epsilon move that is what we have seen today. How to remove the epsilon transitions? So, given a NFSA with epsilon moves, we can construct NFSA without epsilon moves and from that you can construct a DFSA. That if your given a problem to construct a DFSA its better to start straight away constructing the DFSA, because if you can do this and then convert this and (( )) it will take lot of time and the number of states may be very much, because from here to go to here from n you may go up to 2 power n and, these 2 we have to prove.

Given a expression, how regular expression. How to construct the NFSA without epsilon moves and given a DFSA? How to construct the regular expression these 2 portions? We have to see it the language accepted by all these 3 is the same it is the regular set it's called a regular set. The language represented by this see you say that the language accepted by these is this the language represented by a regular expression is this a regular set and a language generated by type 3 grammar or a right linear grammar or a left linear grammar generally, you do not consider this so, much is a regular set.

So, there are 3 ways of defining a regular set either by means of acceptance, by a final state automaton either DFSA or NFSA or NFSA with epsilon moves or it can be represented by what is known as a regular expression or it can be generated by type 3 grammar or a right linear grammar or a left linear grammar. What is a right linear grammar? If the rules are of this form this is called a right linear or a regular grammar of course, when you want to accept epsilon you must make a slight distinguish goes to epsilon you must allow or sometimes a goes to epsilon also you may allow it does not matter much. In the case of regular sets and context free grammars, it does not make much of a difference, if you include epsilon on the right hand side, but only type 1 grammar epsilon should not be on the right hand side and if you want to include epsilon you must be very careful. So, next we will consider regular expression.