Theory of Computation

Prof. Kamala Krithivasan

Department of Computer Science and Engineering

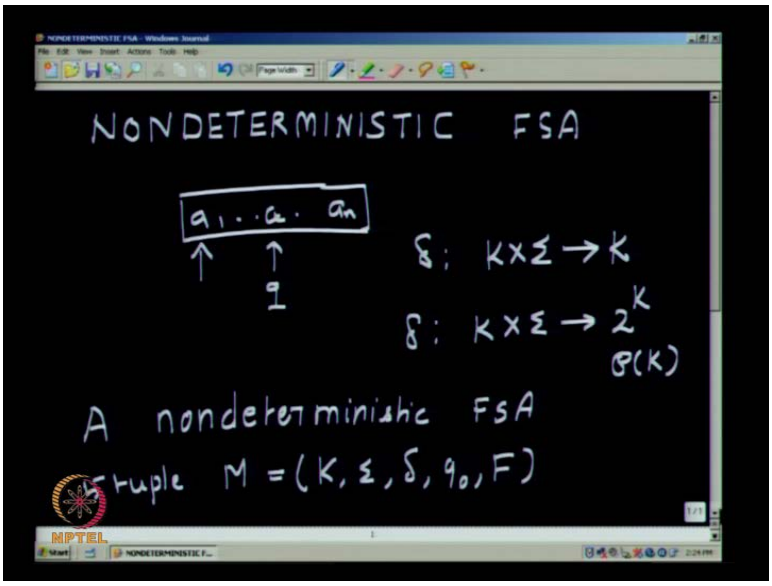
Indian Institute of Technology, Madras

Lecture No. # 09

Nondeterministic FSA

So, in the last lecture we considered deterministic finite state automaton. A deterministic finite state automaton when you have a tape like this.

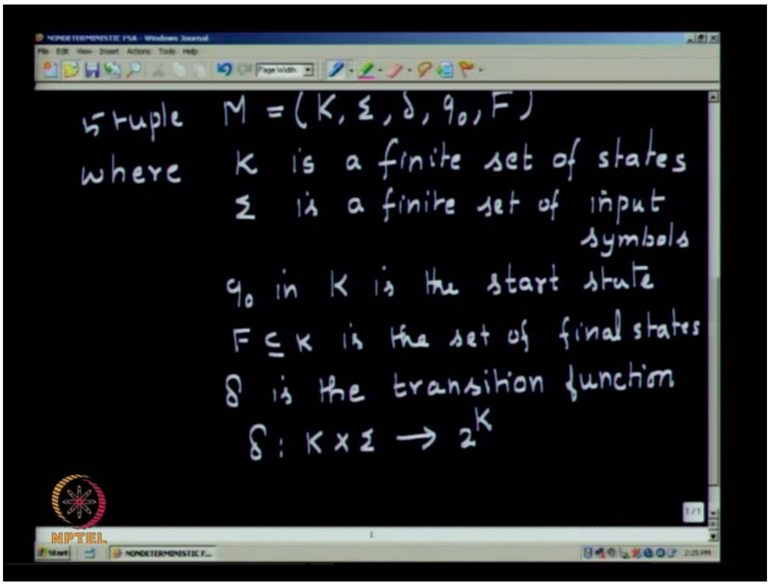
(Refer Slide Time: 00:30)



The input is kept on the input tape and it starts reading the left more symbol in the initial state and it in any instance, if you have a symbol a being read in state say q, depending on q and a the next move is uniquely determined. So, the delta mapping which is the transition function is from K into sigma into k. In a nondeterministic automaton when you have a state and a symbol you have a finite number of choices for the next move, it can move into say q 1 q 2 q 3 a finite number of choices. So, the mapping for a nondeterministic automaton is from K into sigma into 2 power K or power set of K subsets of K. So, this is the definition of nondeterministic of automaton.

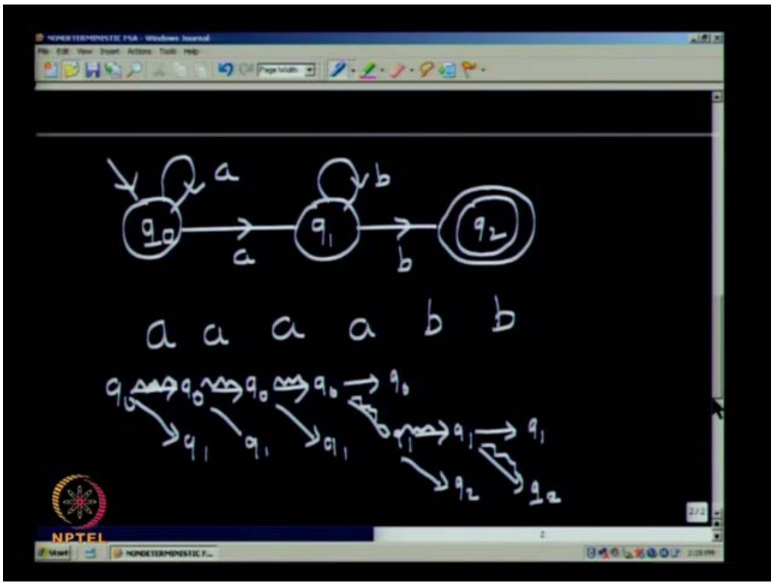
If you want to realize a finite state automaton it can be realized as a synchronous sequential circuit, deterministic automaton can be realized as synchronous sequential circuits. Whereas, nondeterministic automaton is a conceptual thing, because you are going to try out all possibilities in a simultaneous manner so it is an abstract machine. So, coming to the definition of nondeterministic automaton it is defined like this. A nondeterministic (No Audio From: 02:25 to 02:33) F S A it is a 5 tuple M is equal to K sigma delta q naught F (No Audio From: 02:55 to 03:05).

(Refer Slide Time: 03:10)



K as usual is a finite set of states, sigma is a finite set of input symbols, q naught in K is the start state, F contain in K is the set of final states and delta is the transition function (No Audio From: 04:25 to 04:39) delta is a mapping from K into sigma into the subsets of K. Now, we have to define what is the language accepted and how transitions can be represented.

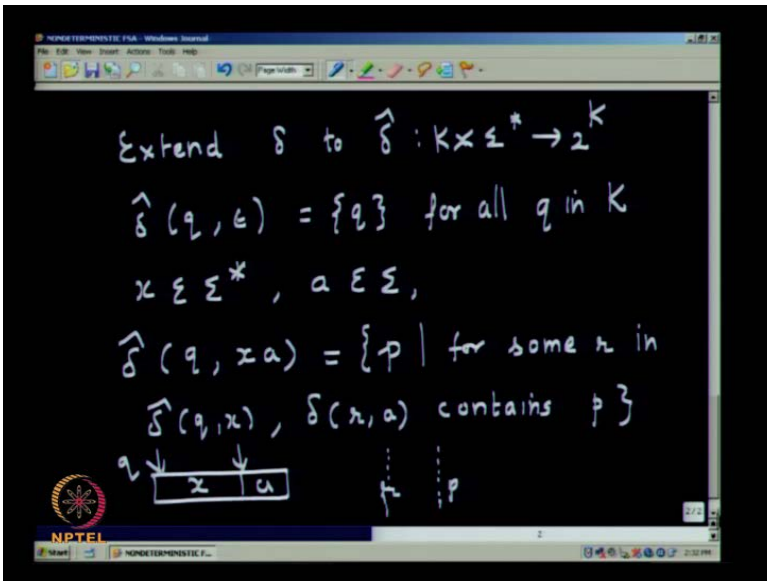
(Refer Slide Time: 05:19)



Now, let before going in to the let us take, take an example (No Audio From: 05:06 to 05:13). Look at this state diagram you have three states q naught q 1 and q 2, q naught is the initial state and q 2 is the finite state the transitions are given by this diagram. The language accepted will be a string of a's followed by a string of b's so, you have a string of a's followed by a string of b's, such a string will be accepted. And this is a nondeterministic diagram, because in state q naught if you get a, you have a two possibilities either you can go to q naught or you can go to q 1, in q 1 if you get a b either you can go to q 1 or you can go to q 2. So, this is basically nondeterministic state diagram of a machine you can draw the state table also.

So, let us see how a string say a a a a b b can be accepted, starting from q naught you have two possibilities q naught and q 1, from q 1 you cannot proceed further so q naught and q 1, q naught and q 1 possibilities are q naught and q 1. Now, in q naught you cannot get a b so, from q 1 if you get a b, you can go to q 1 or q 2, q 1 or q 2. So, after reading the whole string you can be in q 1 or q 2, if one of them is a finite state you accept the string. So, the sequence of states which lead you to acceptance is this, this the sequence which leads you to acceptance (No Audio From: 07:16 to 07: 25).

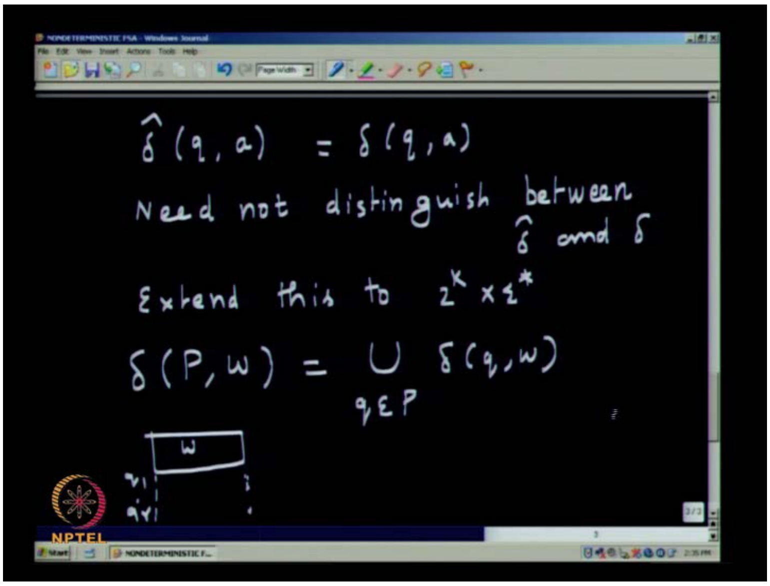
(Refer Slide Time: 07:42)



Now, let us define formally, the accepting how we define acceptance extend delta to delta cap, delta cap is defined from K into sigma star into 2 power K. Instead of K into sigma, we tried to extend it to K into sigma star, how do we define this? Delta cap q epsilon is equal to q for all q in K, what does that mean? If you are in a state q and you are not reading any symbol we are continue continuing in the same state. If you are in q and you are not reading any symbol, you are remain in the same state that is what it means. Then for a string x belonging to sigma star and a symbol a belonging to sigma, what does delta cap? q comma x a mean, this is a set. This set of states starting from q after reading x a what are the possible states in which the machine can be, this is what is denoted by delta cap x a.

How is this defined? This is a set of states P such that for some state r in delta cap q x, delta of r a contains p, what it really means is, starting from q after reading x you may be in a set of states one of them can be r, then from r after reading a there may a some possibilities one of them is p. So, it denotes the set of states p such that after reading x from q you are in a state r and after reading a from r you can go to p, all such possibility possible states are in this set. So, this is the way delta cap is defined (No Audio From: 10:52 to 11:02).

(Refer Slide Time: 11:06)

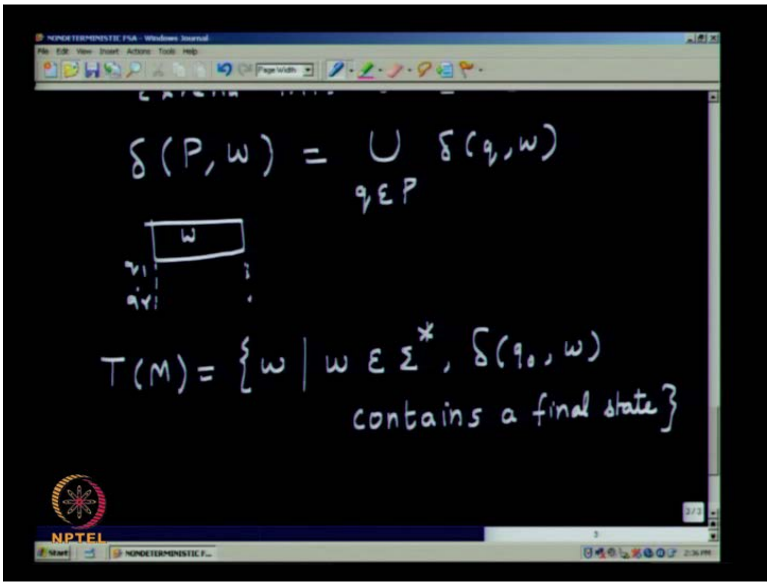


Now, what can you say about delta cap q of a for a symbol, this is the same as delta of q of a, that is starting from q after reading a, what are the symbols; what are the states we can go to, by the same whether you use delta cap or delta. So, in this case also you need not distinguish between while writing between delta cap and delta, does not matter. Whereas, next one more thing we will see in that you have to distinguish between delta and delta cap. Now, you can extend this to 2 power k into sigma star map this, that is, we can say that delta of p comma a, a is a single symbol, p is a set of a's. We can since, we need not distinguish delta between delta and delta cap, otherwise put it as.

(No Audio From: 12:45 to 13:04)

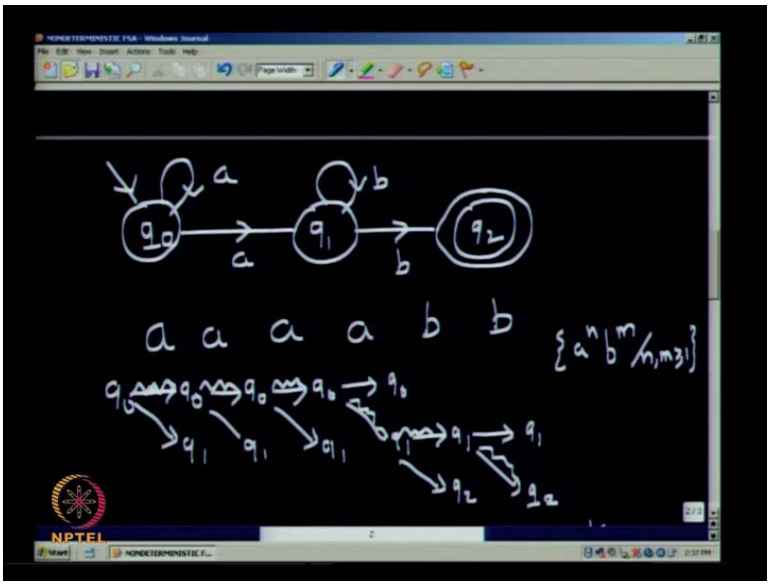
P W, where p is a set of states is union of q belongs to p delta of q w, when you want to extend it to a set of states, that is, from a set of states p after reading w, what can be the set of states in which you can be in, take each one of them and find out and find the unit that is what it means. In a sense, if you have some w like this, initially we can start with say some q 1 q 2 q r, after reading w what can be the set of states, from q 1 find out the possibilities, from q 2 find out the possibilities, from q 3 find out the possibilities and so on, the union is denoted like this.

(Refer Slide Time: 14:26)



Now, what is a language accepted? The language accepted is denoted by t of m, where m is the machine it consists of the set of strings w belongs to sigma star and starting from q naught delta dash q naught w denotes the possible states in which you can be in after reading w starting from q; q naught. This should contain a final state if delta q naught w contains a final state, this denotes a set of rate at least one of them should be a final state in that case w will be accepted. If delta q naught w has one final state at least, then w will be accepted the language accepted contains such strings. So, going back to this one.

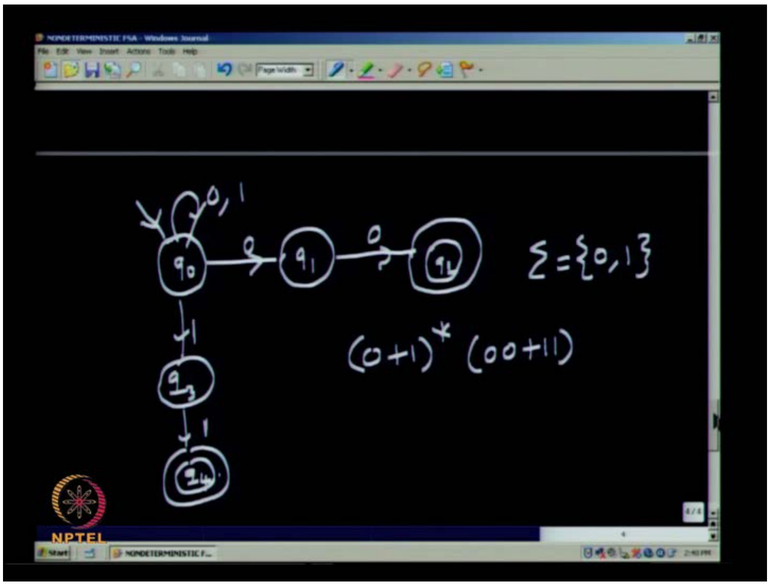
(Refer Slide Time: 15:34)



You see that starting from q naught after reading a you can be in q naught or q 1 after reading two a's we can be in q naught or q 1 and so on. So, these strings cannot be accepted after reading a a a, b you can be q 1 or q 2, q 2 is a final state so a a a, b can be accepted, a a a b, b also will be accepted, because one of them is final state. So, the language accepted here will contain strings of the form a power n b power m n m greater than or equal to 1 (No Audio From: 16:17 to 16:29).

Let us consider one or two more examples and then see whether the power is really increased. Can a nondeterministic F S A accept something which is not acceptable by a deterministic F S A that is not true. Whatever you can accept by a nondeterministic F S A you can also accept by a deterministic F S A the set accepted is called the regular set. So, the power of nondeterministic F S A is the same as the power of a deterministic F S A and how do you get this? You want to simulate a nondeterministic F S A with the deterministic F S A, the proof consists of that. Before that we will consider one or two more examples.

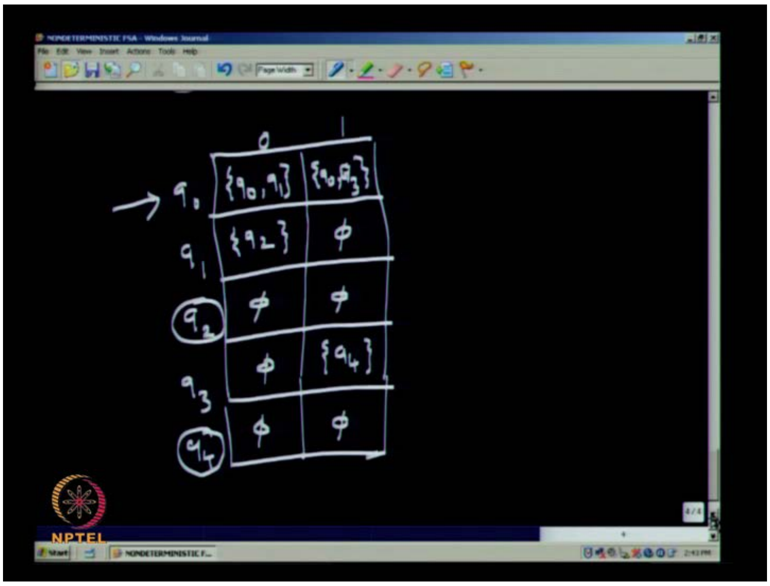
(Refer Slide Time: 17:24)



So, look at this nondeterministic diagram (No Audio From: 17:24 to 17:38) q 2 (No Audio From: 17:40 to 17:51) q 3 q 4 and the transitions are like this 0 1 0 0 1 1. So, what sort of strings will be accepted? The alphabet here, the alphabet here is 0 comma 1, what sort of language will be accepted? You can have any zeros or one's you can read any zero or one, but it should end up with two zeros or it should end up with two one's the language accepted will consist of such strings. In the sigma star notation this will be 0 plus 1 star 0 0 plus 1 1 this is called a regular expression, we will come to that in a movement. So, this really means any string of zeros and one’s and it has to end up with 0 0 or 1 1 so that is what it means.

Now, this diagram or a finite state automaton is basically nondeterministic in nature, because starting from q naught, if you get a 0 you have two possibilities, you go 0 again here, if you get a 1 you can go to q naught or q 3 two possibilities are there. So, basically it is a nondeterministic state diagram.

(Refer Slide Time: 19:54)

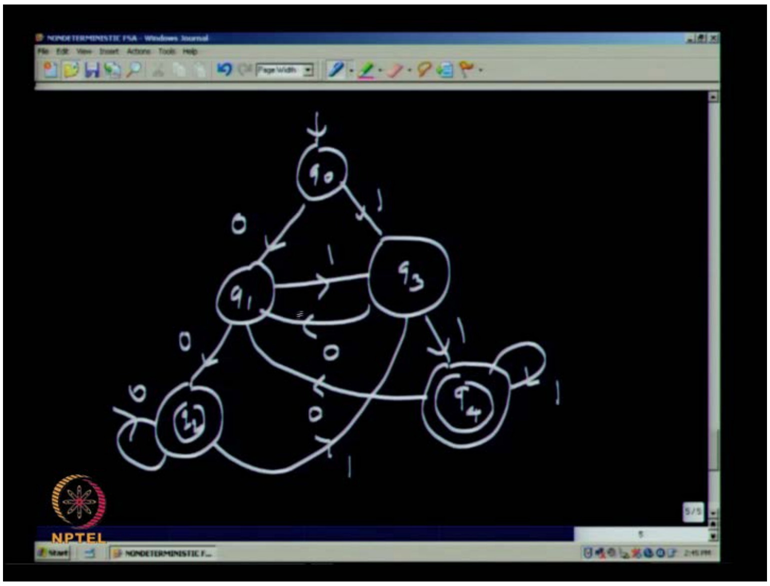


Let me draw the state table for this, this state table they will be one column for each symbol and one row for each state.

(No Audio From: 20:10 to 20:27)

Look at this state diagram (No Audio From: 20:30 to 20:38) from q naught if you get a 0 you can go to q naught or q 1, if you get a 1 you go to q naught or q 3, from q 1 if you get a 0 you go to q 2 you cannot get a 1, from q 3 if you get a 1 you go to q 4 you cannot get a 0 here. So, the table will be like this, from q naught if you get a 0 you go to q naught or q 1, from naught if you get a q 1 you go to q naught or q 3, from q 2 you cannot get anything you cannot go anywhere, from q 4 also you cannot go anywhere from q 1 if you get a 0 you go to q 2 you cannot get a 1, from q 3 if you get a 1 you go to q 4, but you cannot read a 0 initial state is this q 2 and q 4 both are final states. So, the state table will be written like this, we can see that in each compartment or each cell you have a subset of the set of states. Whereas, in a deterministic diagram you will have only one state that this is the state table.

(Refer Slide Time: 22:18)



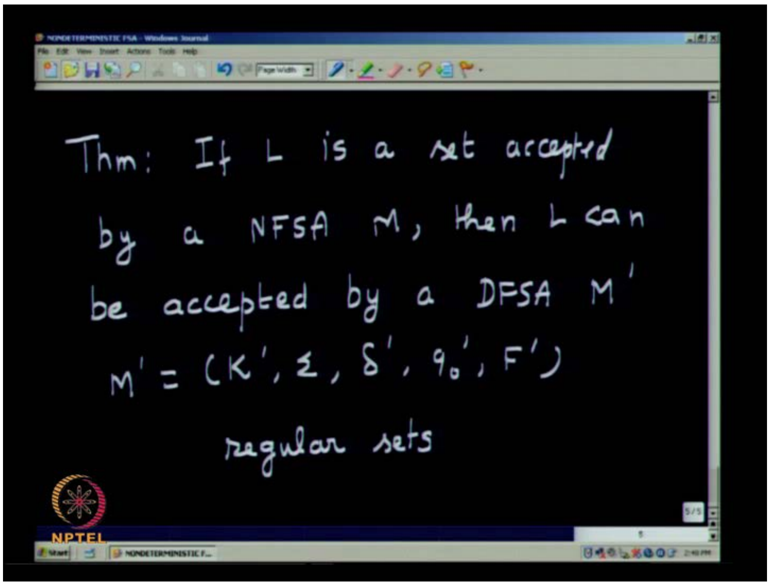
Now, let me draw another diagram.

(No Audio From: 22:07 to 23:10)

There are five states here and q naught is the initial state q 2 and q 4 are final states, is this diagram deterministic or not. From q naught there are two (( )) leaving with label 0 and 1, from q 1 there are two (( )) with label 1 with 0 and 1 with label 1, q 3 0 1 from here 0 1 from here 0 1. If in any state you get a symbol 0 the next state is uniquely determined, if you get a 1 also then x state is uniquely determined. So, what can you say about this diagram it is a deterministic F S A. Now, what sort of language will be accepted by this deterministic F S A, we can see that starting from here if we get two 0's it will be accepted, if we get more 0's also it will be accepted right.

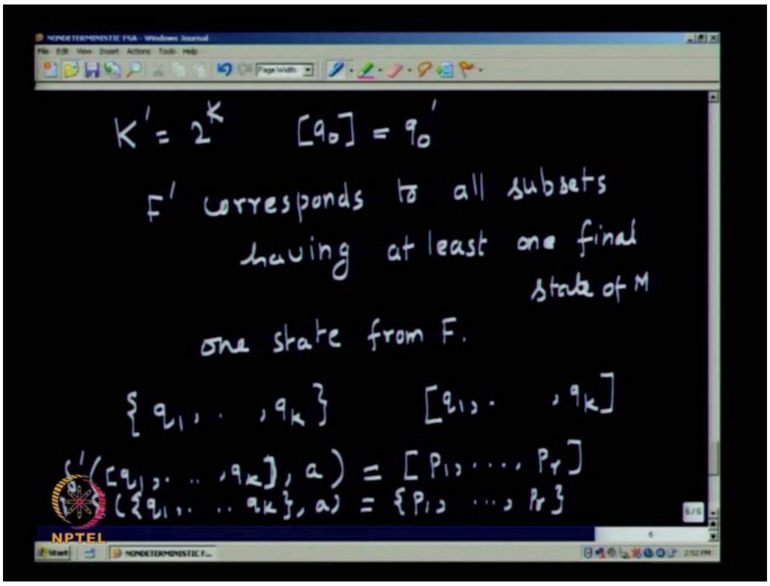
So, anything ending with two zeros will be accepted, if we get 1 here again you go back here and if you get one more 1 it can be get accepted. So, starting from here if you get two 1's it will be accepted any number of 1's also can be accepted, if you get a 0 you go back here, but 1 more 0 you can accept. So, anything ending with two 0's or two 1's will be accepted. So, this is a deterministic automaton and which will accept the same language. And in this case in this example so happens that both are having the same number of sets, but when you want to simulate a nondeterministic automaton with the deterministic automaton the number of states will be increased.

(Refer Slide Time: 25: 23)



Let us see how we can do that. So, we have this result theorem if L is a set accepted is a language or a set is accepted by a NFSA M, then L can be accepted by a DFSA M dash. M dash is equal to K dash sigma delta dash q naught dash F dash, if L is accepted by a NFSA it can be accepted by a DFSA, DFSA by definition form a subset of NFSA is it not. So, if a language accepted by a NFSA can be accepted by a DFSA, what does that means? They are equivalent, the languages accepted by DFSA and the family of languages accepted by deterministic FSA and the family of languages accepted by nondeterministic FSA or equivalent, they are called the regular sets.

(Refer Slide Time: 27:26)



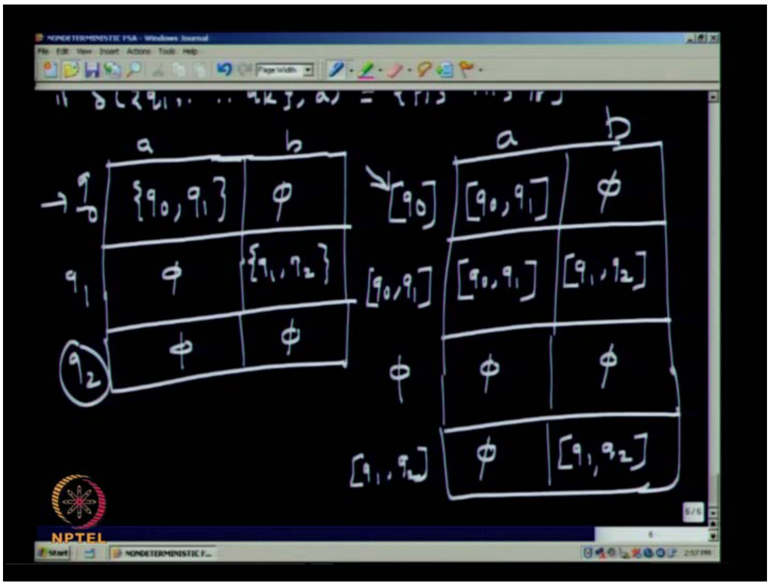
Now, how do we find out what is K dash and how do we define F dash etcetera. Now, K dash is really the subset all subsets of K so, if we have three states in the nondeterministic automaton you can have 2 power 3, eight states in the deterministic automaton. And the set q naught, the subset having q naught alone corresponds to q naught dash, initial state is a one which corresponds to the subset containing q naught alone. And F dash corresponds to all subsets having at least one final state of F; of M, that is, F at least one state from F, one state from F. Now, we have to so a subset of states is usually denoted by q 1 q 2 q k, where each one of them belong to f k, this subset corresponds to a single state in the deterministic machine.

In the nondeterministic machine this corresponds to a subsets of states, in the deterministic machine this subset will corresponds to a single state and that is denoted by q 1, q 2, q k. Note that to denote a subset we are using flower brackets, when it becomes a single state in the deterministic automaton we denote a square bracket. And delta dash is defined like this, delta dash of q 1 q 2 q k comma a this is defined as p 1 p 2 p r note that this square bracket. From this set after reading a you go this state, from this state after reading a you go to this state, if delta of q 1 comma q 2 q k a equal to p 1 comma p r note the difference this is flower brackets, this is square brackets. From this subsets of states in the original machine after reading a you go to this subset, in the deterministic automaton you define it as from this state after reading a you go to this state.

So, we have to prove that the language accepted is the same, before proving that let us take an example and convert it into the deterministic automaton. So, the one which we are consider we will take again, maybe I should go back to the diagram, we have considered this earlier (No Audio From: 31:20 to 31:32). This is the nondeterministic machine, let us construct the deterministic machine equal, we have already seen what is the deterministic machine equivalent to that by this construction how do we get that, let us see that.

(No Audio From: 31:45 to 32:06)

(Refer Slide Time: 32:38)



So, first let us draw the state table.

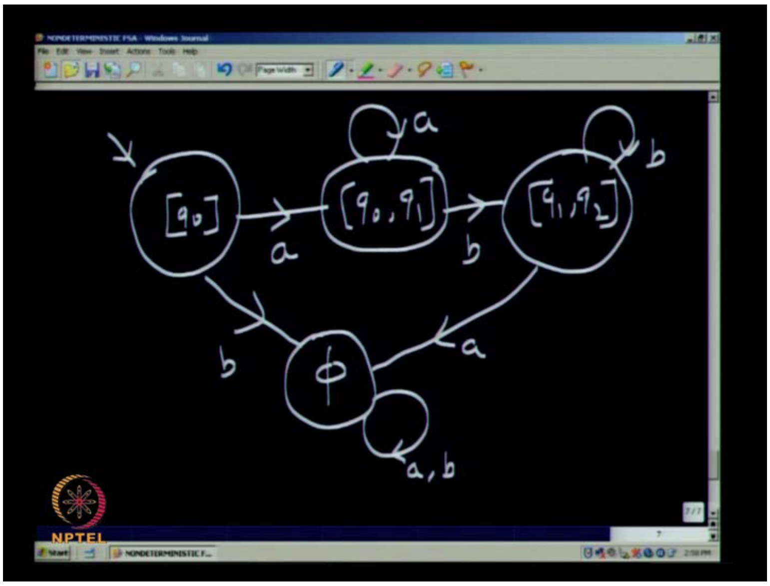
(No Audio From: 32:10 to 32:37)

The state table is like this from q naught if you get a you go to q naught or q 1, if you get a b you cannot go to anything, from q 1 if you get a b you go to q 1 or q 2 you cannot read a, from q 2 you cannot read anything this is the state table of the nondeterministic diagram, this is the initial state and this is the final state. Now, converting into the deterministic diagram you have the same number of columns one corresponding to a, one corresponding to b, the initial state is q naught, this is the initial state. So, from q naught after reading a what can you be, what can be the set of states in which you can be q naught or q 1. So, this will be q naught q 1, please note that to denote it a single state we are using square brackets whereas, as a set it is denoted by flower brackets. And after reading b you cannot go to any particular state so this is empty, empty this is the subset of the set of states, empty set of the; empty set is also a subset right. Now, this empty set you will see that corresponds to something like to dead states.

Now, afterwards you take this q naught q 1, from q naught q 1 after reading a what can be the set of states in which you can be in, you have to find the union of this, that is, again q naught q 1. And after reading b what can be the set of states in which you can be find the union of these two that will be q 1 q 2. So, with this you have seen what are the next transitions of course, empty set we have to see so, empty set means of next transition will be only empty. Now, we have seen for this we have we have to see for this q 1 q 2, if you are in q 1 q 2 and if you get a, the next state is the union of these two that will be empty, if we get b the next state will be the union of this that will be q 1 q 2. So, we have considered this we have considered this and so on so we need not prolong the table.

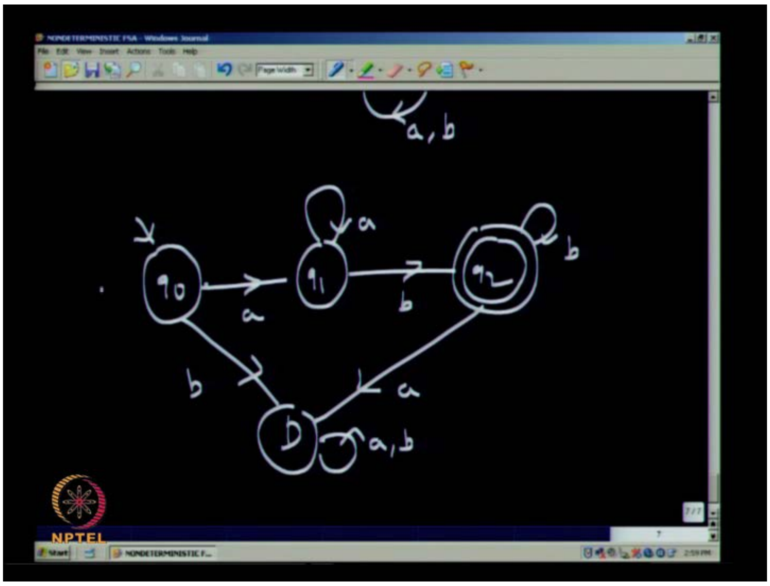
Even though 2 power 3 eight states are possible, you can have an empty state, one corresponding to q naught, one corresponding to q 1, one corresponding to q 2, then q naught q 1 q 1 q 2 q 2 q naught q 2 also you can have q naught q 1 q 2 you can have, but they will not be useful at and at this stage the table is complete so, you need not have to proceed further. Even though 2 power K states, that is, in this example 2 power q 2 power 3 eight states are possible only four will be useful.

(Refer Slide Time: 36:37)



So, if you draw the diagram for this (No Audio From: 36:26 to 36:32) it will be like this q naught, q naught q 1 q 1 q 2 and the empty state, the transitions will be like this a a b b b a a comma b, you can see that this is exactly the same which we considered earlier.

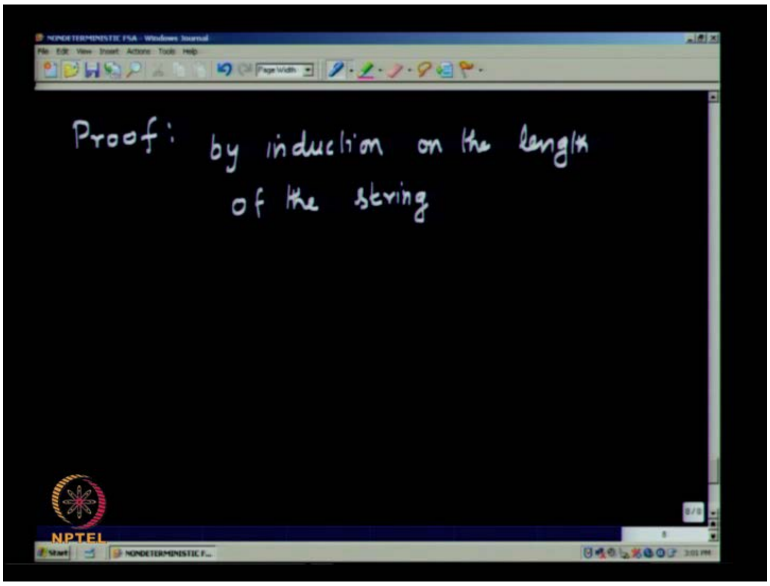
(Refer Slide Time: 37:35)



The earlier case without naming the states the diagram was exactly similar, but I think we use d symbol for this, q 1 for this, q 2 for this, q naught for this and the transitions were exactly the same right. So, you can see that this is a deterministic diagram, because from here if you get a you go here, if you get b you go here, from this if you get a go here you get b go here, from this if you get a go here, from this if you get b go here, this corresponds to the dead state, the empty set corresponds to the dead states. So, using this construction we have converted the nondeterministic automaton into a deterministic automaton. The diagram if the earlier one which you consider was like this (No Audio From: 38:30 to 38:44).

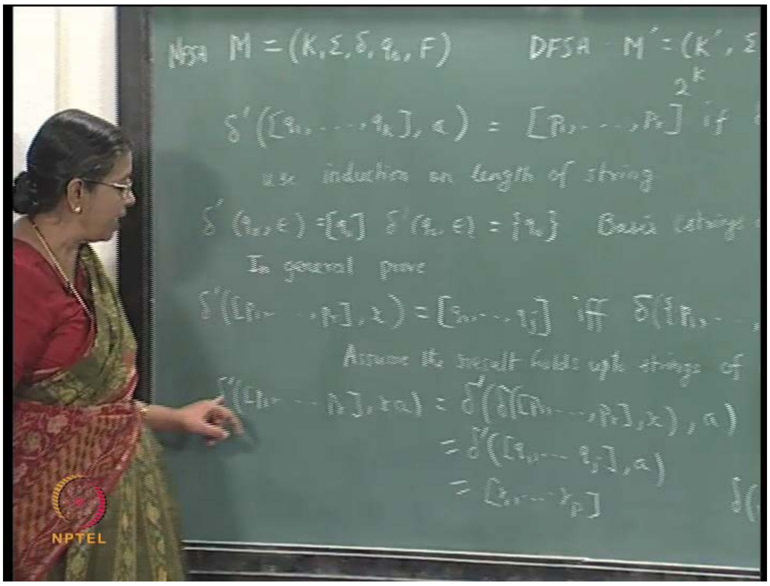
Now, what are the initial states, the initial state is just the one containing q naught alone and in the original diagram q 2 was the final state. So, any subset among these this is the only subset containing a final state so this will be made as the final state. So, if you write you can form this is the final state. So, this is to illustrate how we can convert a nondeterministic automaton into a deterministic automaton you seen this construction and this is known as the subset construction. Usually we call it the informally, we call it a subset construction, it is not you are constructing the subsets this method is called subset, because you are using the idea of subsets.

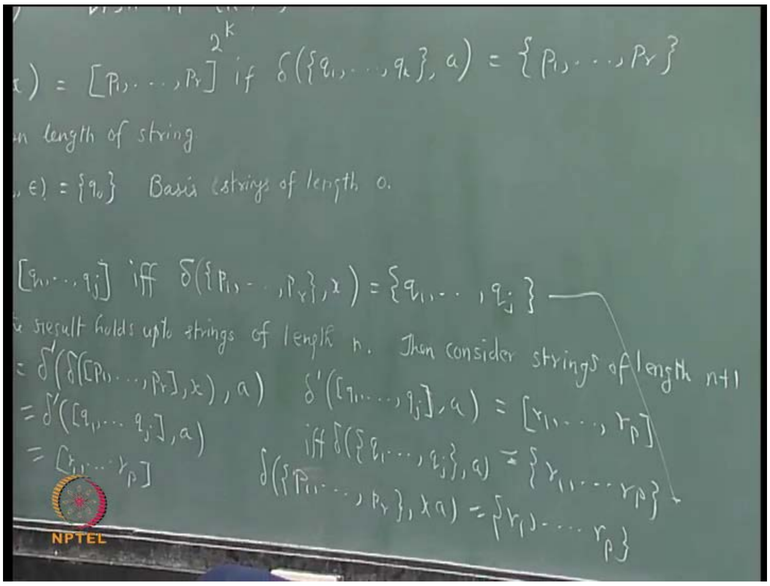
(Refer Slide Time: 39:50)



Now, the proof you have to prove that the automaton which you have constructed, the deterministic automaton which you have constructed accept this (( )) language as the original nondeterministic automaton. The proof is by induction on the length of the string NFSA.

(Refer Slide Time: 40:28)





You have given NFSA K sigma delta q naught F and you are constructing a DFSA M dash equal to K dash sigma delta dash q naught dash F dash. Now, this is the set of subsets and we have defined delta dash and F dash, how did we define delta dash? We define delta dash q 1 q 2 q k a is equal to p 1 p 2 p r if delta of q 1 q 2 q k a is equal to p 1 p 2 yeah and F dash is the set of states which corresponds to subsets which contains a states from F. Now, use induction on length of string accepted (No Audio From: 41:58 to 42:13).

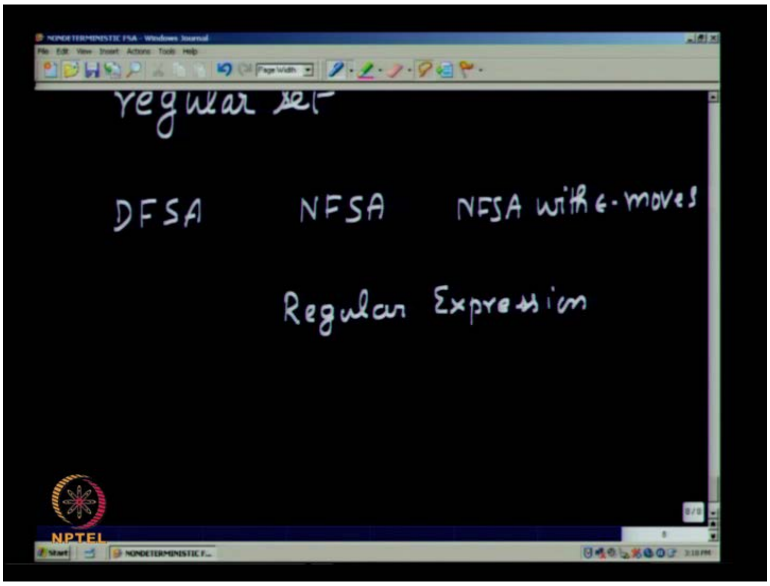
So, if you start from q naught delta dash q naught epsilon what can you say about this? This will be delta of q naught epsilon it is; it will be equal to q naught only here it will be equal to q naught and it will be equal to q naught here. And if you have, if delta of q naught x is equal to; so for strings of length basis this is a basis class, you consider strings of length 0 so, if epsilon is in the language it will be in this language and so on. So, if it is accepted by this it will be accepted that it will be accepted by that and so on. If q naught is a final state only it can be accepted by the nondeterministic FSA and also by the deterministic FSA.

Now, we show that if delta of q naught x is equal to or in general you can prove like this, in general you can show that (No Audio From: 44:05 to 44:15) prove. Delta dash of p 1 p 2 p r x is equal to some q 1 q 2 q j if and only if, I am sorry this is square brackets, delta of q 1 q 2; delta of p 1 p 2 p r x is equal to q 1 q 2. Now, if you take just x is equal to epsilon string of length 0 obviously delta of p 1 p 2 p r epsilon will be p 1 p 2 p r and so it will hold for this. Now, assume induction hypothesis assume the result holds up to strings of length n, then consider strings of a length n plus 1. So, what can you say about this? Delta dash say p 1 p 2 p r x a, what it will be this? You know that delta of p 1 p 2 p r x a is by definition delta of delta of p 1 p 2 p r x a, this is the definition of a deterministic automaton.

But by this one this is equal to q 1 q 2 q j, delta of p 1 p 2 p r x is q 1 q 2 q j, by the way we have defined delta dash I am sorry this is delta dash. By the way we had defined delta dash what can you say about delta dash q 1 q 2 q j, this will be equal to say some r 1 r 2 r p if and only if delta of q 1 q 2 q j comma a is equal to r 1 r 2 r p, this is by definition. So, what can you say about this? This will be equal to say r 1 r 2 some r p. But we have delta of p 1 p 2 p r x is equal to q 1 q 2 q j and delta of q 1 q 2 q j please note that they are subsets a is equal to r 1 r 2. So, from this and this you get delta of q 1 q 2, I am sorry delta of p 1 p 2 p r x a is equal to r 1 r 2 r p. So, when you find that delta you get delta of p 1 p 2 p r x a is equal to this and delta; delta dash p 1 p 2 p r x a is equal to this and this is this is yes this is delta dash and delta of p 1 p 2 p r x a is equal to this.

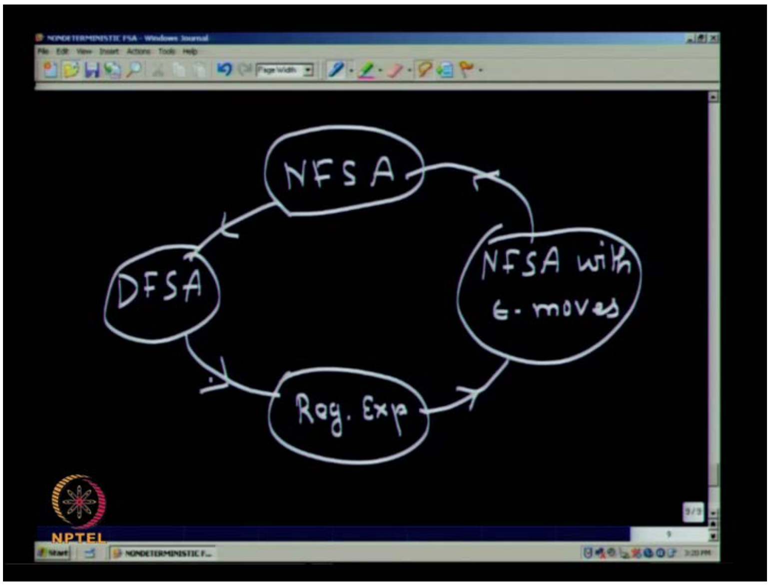
So, this happens, this is equal to this if and only if this happens and this proves the induction step. So, when you start with the initial state, the initial state here is q naught, here it is the single subset containing q naught and any subset containing a final state will be a final state here there the deterministic automaton. So, you can show that if a string is accepted by the nondeterministic automaton, then it will be accepted by the deterministic automaton and vice versa. So, the language accepted by both are the same T of M is equal to T of M dash so thus we show that the power by using nondeterministic is not increased yet. Any set accepted by a nondeterministic automaton is accepted by a deterministic automaton such set is called a regular set.

(Refer Slide Time: 51:03)



So, we this is the step we have considered (No Audio From: 51:14 to 51:24). First we have considered DFSA now, we have considered NFSA, we will also consider one more thing called NFSA with epsilon moves even in this case the power will not be increased it will accept only regular sets. So, and we will also define what is called a regular expression, all these are all equivalent none of them have more power or less power. So, the way the proof will go is like this.

(Refer Slide Time: 52:17)



You have NFSA, you have DFSA, you have NFSA with epsilon moves and you have regular expressions, all are equivalent. So, what we proved today is only this portion given in NFSA how to construct the DFSA. In the next lecture we will see what is an NFSA with epsilon moves and how to remove the epsilon moves? We will see that given an NFSA with epsilon moves, you can construct an NFSA without epsilon moves this direction will prove. And we will also define what is meant by a regular expression and given a regular expression how to construct an NFSA with epsilon moves. Then we will also see that given a DFSA how to find the regular expression corresponding to this which will prove the equivalence of all these four. So, we shall see it in the next class.