Theory of Computation

Prof. Kamala Krithivasan

Department of Computer Science and Engineering

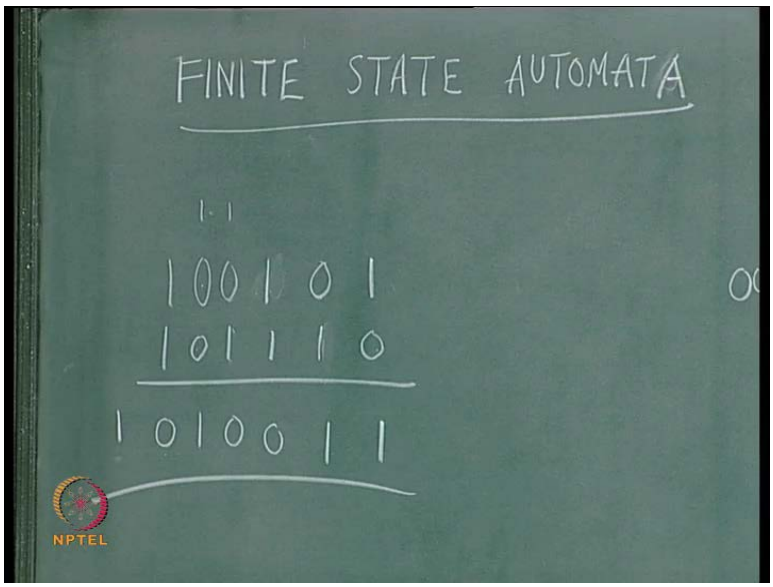
Indian Institute of Technology, Madras

Lecture No. # 08

Final State Automata

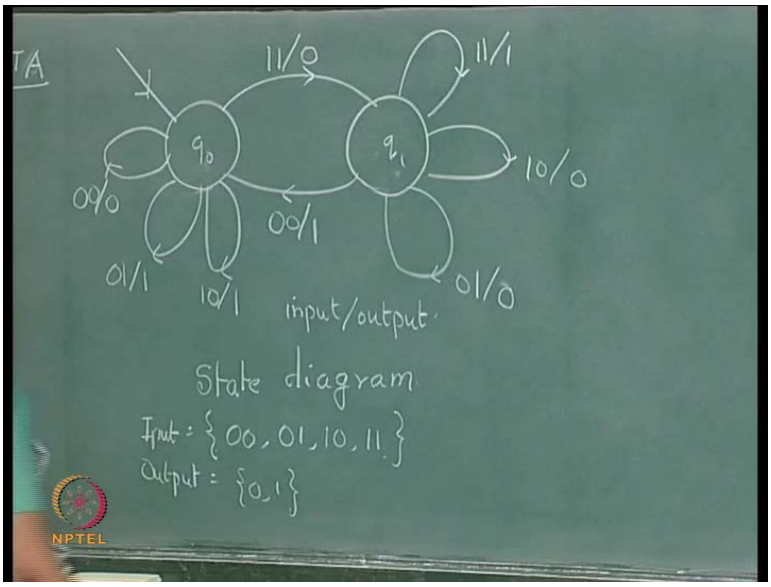
Today, we shall consider the topic finite state automata, this is very useful topic. It has got lot of applications; it finds use in the lexical analysis part of a compiler. You know that a compiler is a very big program which translates a high level language into a machine language and it has got several parts. And in the lexical analysis part of a compiler the idea of a finite state automaton has found a lot of use. And it is also used in text editing and in computer networks and in image compression and so on. There are lot of application for that though we may not study all of them, I will indicate to you the application of finite state automaton in the lexical analysis part. Actually, if you know about sequential circuits the finite state automaton is an abstract model of a synchronous sequential circuit. So, let us start with the serial adder and go into the concept of a finite state automata.

(Refer Slide Time: 01:19)



Consider 2 binary numbers and consider the addition of these 2 numbers 1 0 gives you 1 0 1 also gives you 1 1 1 gives you 0 with a carry 1 and that carry along with this 1 gives you a 0 and there is a carry and carry and the 0 0 gives you a 1. Now, there is no carry this 1 1 also gives you a 0 and there is a carry and that is given here.

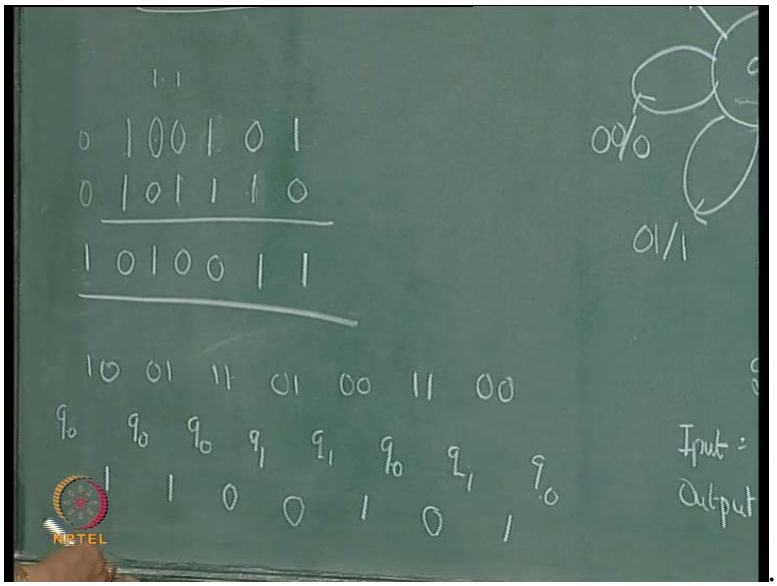
(Refer Slide Time: 01:58)



Now, if you want to represent it as a state diagram like this, this is called a state diagram. (No audio from 01:58 to 02:08) First you get zero 1 0 the input is 1 0, then the input is 0 1, then the input is 1 1, then 0 1, then 0 0 1 1, the output is 1 1 0 0 1 0 1. Let us see how you get this from this diagram, first you get a 1; you are starting in the initial state q naught which is marked like this. Then in this you get a 1 0, a 1 0 gives you a output 1 and you go here, that is what you get. Then you get a input 0 1 then also the output is 1 and you go here then you get a input 1 1, so the output is 0. And you go here and then you get a input 0 1 so you get 0 as the output and you go here then you get the input 0 0 so you go here with the output 1. And from here you get the input 1 1 so you output is 0 and go here and from here you get a 0 0 that is there is no output the last output for this there is no input, but, without loss you can assume it is 0 0 so you from here you get a 0 0 you output a 1.

Now, here, the input is a pair of binary digits 0 0, input is either a 0 0 or a 0 1 or a 1 0 or a 1 1, 2 digits; output is 0 or 1. In the diagram the input is written first with the slash and the output.

(Refer Slide Time: 04:24)

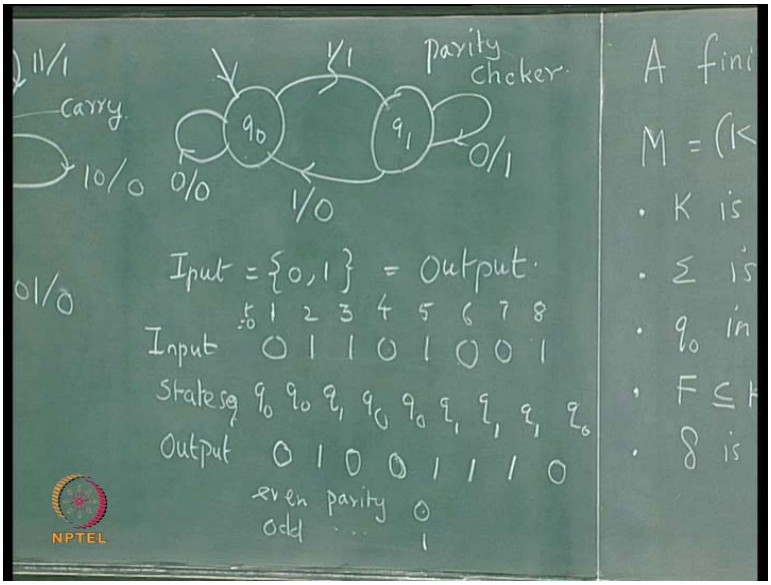


So let us write the inputs like this this is the 1 which you are going to get first, then this, then this and so on. So, first you get 1 0, then 0 1, then 1 1, then 0 1, 0 0, 1 1 and 0 0. So, initially you are starting in state q naught, after getting a 1 0 you are again in state q naught and after and at the time the output will be q naught 1 0 gives you in a output 1. Then next here in state q naught and you get 0 1 so you go to q naught again and output is a 1 these 2 are called states. So, from q naught if you get a 1 1 you go to q 1 and the output is 0 now. From q 1 if you get a 0 1 you go to q 1 again and the output is 0, from q 1 if you get a 0 0 you go to q naught and output is a 1, from q naught if you get a 1 1 you go to q 1 and output is 0, from q 1 if you get a 0 0 you go to q naught and output is a 1. So, when the inputs are 1 0 0 1 1 1 0 1 0 0 1 1 0 0 that is this sequence; the sequence of states will be q naught, q naught, q naught, q 1, q 1, q naught, q 1, q naught and output will be 1 1 0 0 1 0 1 1 1 0 0 1 0 1 this is what you get and this is you get from this diagram.

Now, when you look at this diagram carefully, you see that this state q naught corresponds to a no carry state. When there is no carry, the machine is in q naught and when there is a carry this corresponds to a carry state. When you read a 2 binary numbers, digit by digit there is parse of digits you read. Then after reading a particular portion either there will be a carry from the previous digits or there will be no carry. So, when you you see that initially when you start there is no carry. So, when there is no carry, if you add a 0 and 1 you get 1 again there is no carry, without a carry then when

you add a 0 and 1 you get a 1 and now also there is no carry but, when you add 1 and 1 you get the output 0 but, there is a carry. So, you have to distinguish between 2 types of states, one is when there is a carry and the other is when there is no carry. The whole thing can be represented by a diagram like this this is called a state diagram and this is an abstract model of a synchronous sequential circuit. You know that a serial error is a synchronous sequential circuit and the abstract model is represented like this.

(Refer Slide Time: 08:13)



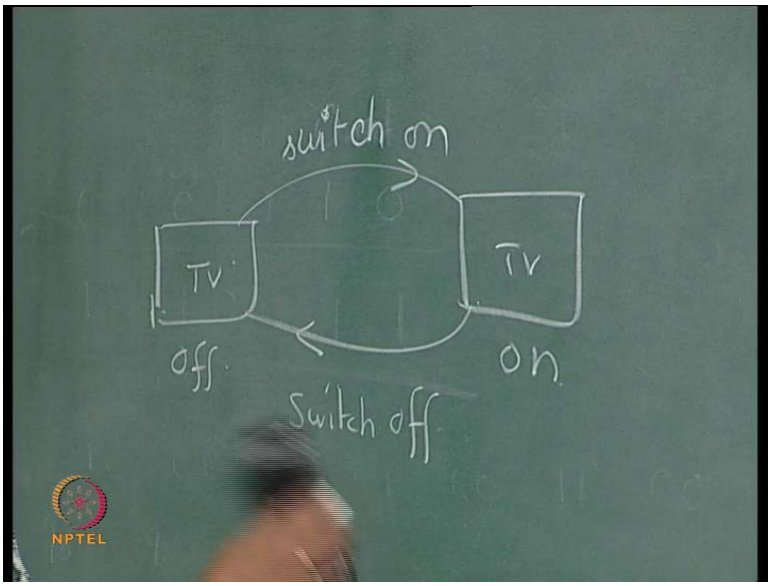
Now, let us consider one more example, consider this example, q naught and q 1. When you get a 0 output is a a 0 when you get a 1 the output is a 1 when you get a 0 here the output is a 1 when you get a 1 the output is a 0. Here the input is the symbols 0 and 1 that is also the output q naught is a initial state, the initial state is marked by an arrow like this. So, suppose I get the input 0 1 1 0 1 0 0 1, suppose I get this input what will be the state sequence state sequence? Initially, the machine is in state q naught when you get a 0 it goes to q naught again and the output, what will be the output? Output will be 0 here, in q naught if you get a 1 you go to q naught, if you get a 1 you go to q 1 state q 1, here again the input and output are written with a slash if first symbol is input then slash then the output this is the convention with which we write. So, in q naught when you get a 1 you go to q 1 and then the output is a 1, in q 1 when you get a 1 you go to q naught and output is a 0, in q naught when you get a 0 you are in q naught and output is a 0, in q naught when you get a 1 you go to q 1 and output is a 1, in q 1 when you get a 0 you

remain in q 1 and output is a 1, in q 1 again when you get a 0 you remain in q 1 and output is a 1, in q 1 when you get a 1 you go to q naught and output a 0.

Now if you look at this very carefully this is a input 0 1 1 0 1 0 0 1 and this is the output 0 1 0 0 1 1 1 0. Now, this is the time with which you start, time t is equal to 0, t is equal to 0 output then the time at after reading time 1 you read this, time instance 2 you read this, time instance 4 and so on. Now, what does this do? Say at time 5, you have read this portion of the input and what is the output at time 5? 1. At time 6, you have read this portion and what is the output 1, at time 8 you have read this portion and what is the output that it is 0. At any time instance, the output denotes the number of 1 ’s you have read so far whether it is an odd number of 1 ’s or even number of 1 ’s. So, this machine is actually a parity checker this is a parity checker.

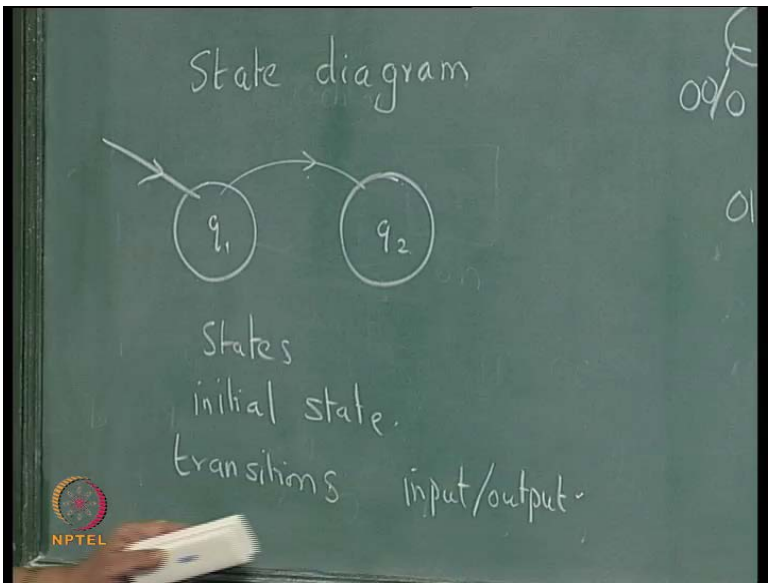
So, at time instance 0, you are in q naught then at time instance 1 you have read a 0 and how to (( )) 0 that is so far you have not read any 1 so even parity. Now, next you read a 1, so you have read 1 1 up to this point so the odd parity so you get 1 at a next instance you have read 2 1 ’s so in now you have read even number of 1 ’s so even parity you get a 0. So, fourth instance also you get a 0 so this tells you that you have read an even number of 1 0 so the output is a 0. So, whenever there is even parity the output is 0 so you get this, even parity you get output 0 or parity is denoted by the output 1 this is known as a parity checker. This is again a straight diagram and there are 2 states q naught and 1 q 1 and input and output are written like this input slash output and this is known as a finite state automata. So, this is another example, you know that parity checker also can be implemented by synchronous sequential circuit and this is a abstract model.

(Refer Slide Time: 14:18)



Now, apart from this let us consider a simple 1 suppose you are having a Tv and when you switch switch on the switch on switch switch on the Tv is initially off, then when you switch on it goes to the on stage; when you switch off it goes to the off stage is a small instance of a day to day life which you can a represent it by means of a state diagram like this.

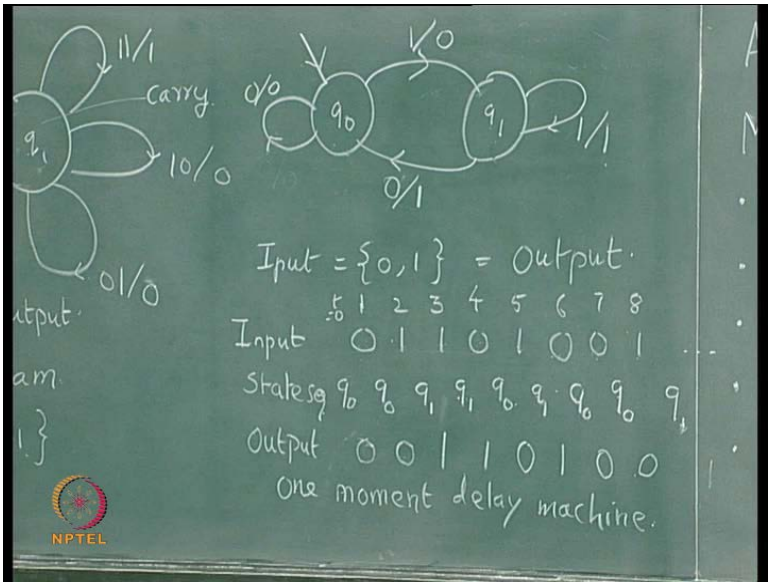
(Refer Slide Time: 15:10)



In general a state diagram consist of the following things, it has states written in the form of a circle states will be written like this they are called states. And the initial state will

be denoted like this you have initial state marked then transitions transitions they are marked by arrows and another transitions you have input slash output. And later on we shall consider that you can look at it also as a recognition device in which case you would not talk about output but, only input but, somehow the states will be designated as final states.

(Refer Slide Time: 16:49)



So, let us consider 1 more example which is like this this, this also has got 2 states and the input can be 0 1 which is also the output. In that case, when you get a 0 the output is 0 when you get a 1 you go to q 1, in q 1 when you get a 1 you remain there again when you get a in q 1 you get a 0 you go here but, the output will be a 1.

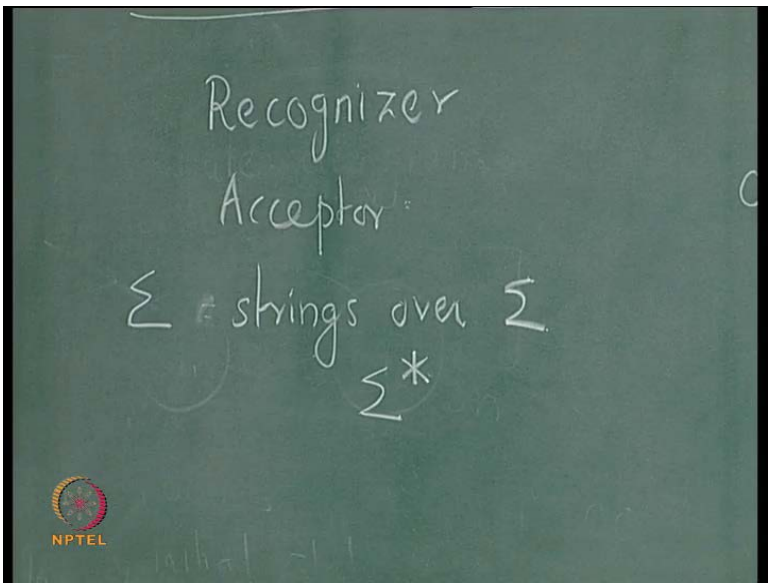
So, this is another state diagram of a machine and let us see, what this machine does? Let us consider the state sequences, initially, you are in q naught and when you get a 0 you are in q naught when q naught when you get a 1 you go to q 1, in q 1 when you get a 1 you go to q 1, in q 1 when you get a 0 you go to q naught and in q naught when you get a 1 you go to q 1 and in q 1 when you go get a 0 you go to q naught and in q naught when you get a 0 you go to q naught and in q naught when you get a 1 you go to q 1. Now, let us see, what is output? The output is q naught 0 is 0 q naught 1 is 0 q 1 1 output will be a 1 q 1 0 you go to q naught but, output a 1 and q naught when you get a 1 you output a 0 and go to q 1, in q 1 when you get a 0 you output a 1 and go to q naught, in q naught

when you get a 0 you go to q naught and output a 0, in q naught when you get a 1 you go to q 1 and output a 0.

Now, you look at it carefully, what does the output represent? The output represents like this leave alone the first output the first output will be 1 0 here always, afterwards look at the input sequence and the output sequence 0 1 1 0 1 0 0. So, the same input you are getting here except for the first 1, first 1 will always be a 0. And you are in state q 1 now, next whether you get a 0 or a 1 in q you are getting a q 1 next instance you may get a 1 or a 0 but, the output will be a 1, next instance whatever maybe the input the output will be only 1.

So, if you forget about the first output, the same input sequence you are getting here and this machine is called a one moment delay machine one moment delay machine. The first output is 0 afterwards this input is output 1 instance later, second instance you get 1 but, you output what you got at the first instance. Third instance input is 1 but, you are outputting what you got at the second instance and so on. That is why this machine is called a one moment delay machine; this is again another example of a finite state machine.

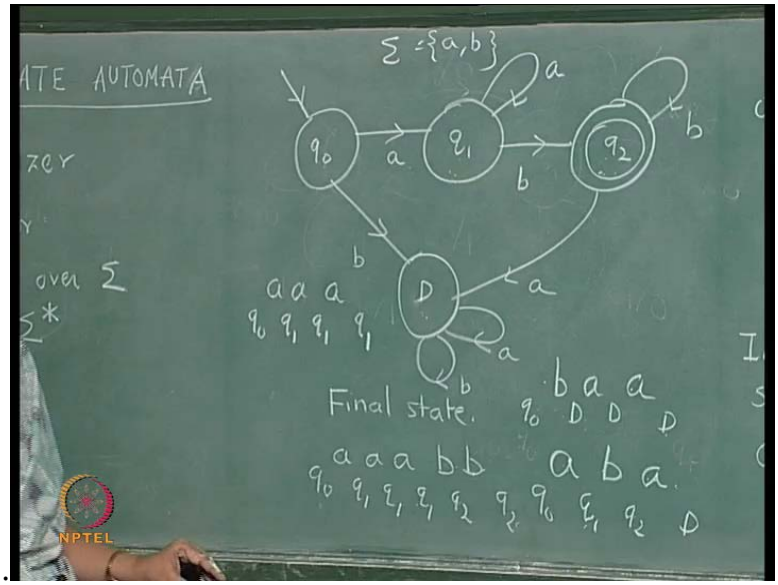
(Refer Slide Time: 21:26)



So, like that we can consider several examples. So, in all this examples we have input, we have output but, we did not say anything about final states and so on. You can also look at the finite state automaton as a recognizer recognizer or an acceptor, it is

accepting strings. You consider an input alphabet sigma and consider strings over sigma this is denoted as sigma star this we know, strings over sigma are denoted by sigma star. A finite state machine can be designed such that it accepts a subset of sigma star.

(Refer Slide Time: 22:18)



Let me consider one example (No audio from 22:07 to 22:17) look at this machine (No audio from 22:19 to 22:32) the alphabet is a b.

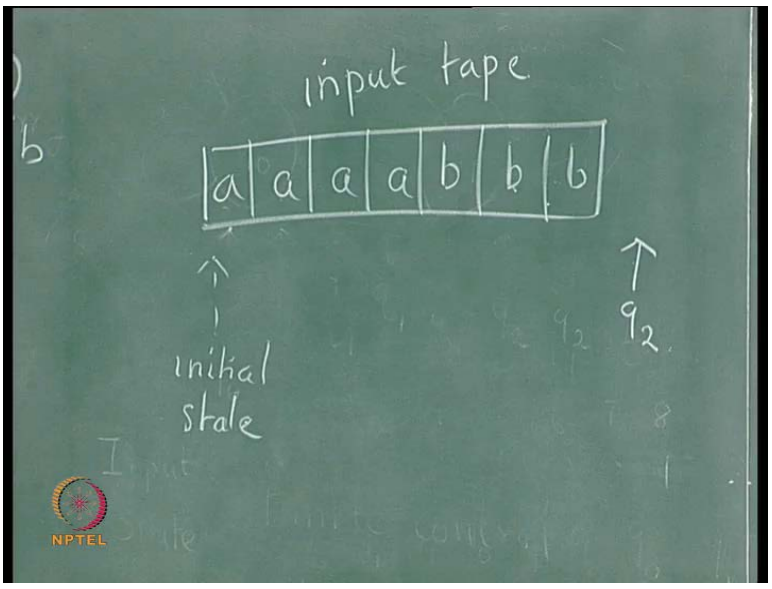
(No audio from 22:38 to 23:23).

Let us consider this diagram, there are 4 states q naught, q 1, q 2, d. Now, you see that, you are not marking any of them like input slash output, the transitions it has only one symbol which denotes input now, we are bother only about the input. And you see that this state is marked with the double circle that is called the final state final state and this is the initial state. Now, if you traverse a path along for a string for example, consider a string a a a b b this is the input, what will be the state sequence for this? Q naught a will give you q 1, q 1 a will give you q 1, q 1 a will give you q 1, q 1 will b will give you q 2, q 2 b will give you q 2. So, for the sequence a a b b a a a b b so after reading the string a a b b a a a b b you reach the state q 2 which is a final state. So, given a string you start with q naught and after reading the whole string, if you reach a state which is a final state that string is set to be accepted by this final state automata.

Consider this string b a a a or consider the string a b a, what will be the state sequence for this? This is q naught q naught b will be d, why I am calling it as d and not q 3 we will be clear in a moment d a is d d a is d. Similarly, start from here q naught a is q 1, q 1 b is q 2, q 2 a is d after reading these strings you are in the state d which is not a final states. So, these 2 strings b a a a b a they are not accepted by the machine. Look at this string also a a a start from q naught after reading a you will be in q 1, after reading a again you will be in q 1, after reading a you will be in q 1, q 1 is not a final state so this string also cannot be accepted by the finite state automata. So, starting from the initial state after reading the string if you reach a final state the string will be accepted. So, you look at this this string will be accepted whereas, this string or this string or this string will not be accepted by this machine.

So, if you look at the diagram very carefully, you will realize that any string where you have a sequence of a's followed by a sequence of b's will be accepted. But, there should be at least 1 a and 1 b there should be 1 a and 1 b so a sequence of a's followed by a sequence of b's will be accepted by this machine with the condition that there should be at least 1 a and 1 b any other string will not be accepted.

(Refer Slide Time: 27:38)

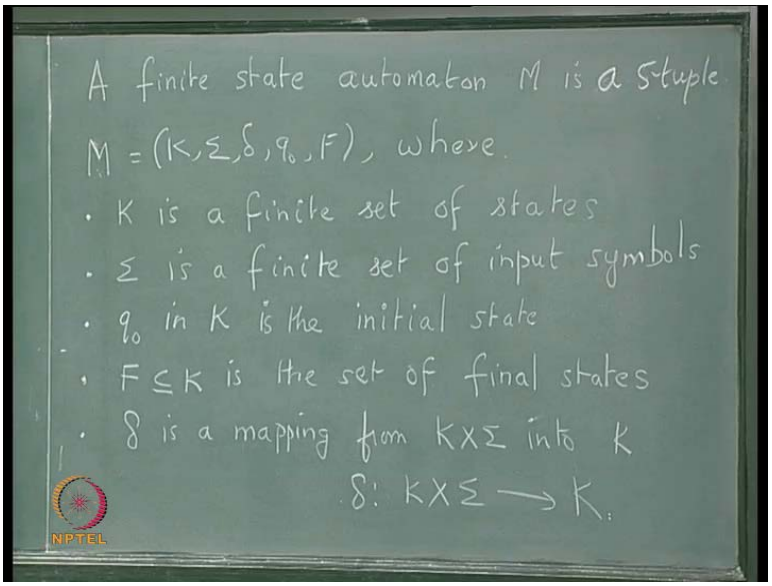


So, this is the idea of a finite state automaton as a recognizer, let us see the formal definition. Now, before going into that the idea is represented like this you have an input tape in which you have the symbols, say for example, here you have the symbols there is

a finite control this is called finite control which represents the state of the machine and there will be an input head. Initially, the input head will be pointing to the leftmost symbol of the input tape this is the input tape. Now, the input is a a a a b b b and initially, the machine will be in state q naught which is the initial state. Then at any particular instance depending upon the state and the symbol you go to the next state and the input pointer will be moved 1 1 point to the right.

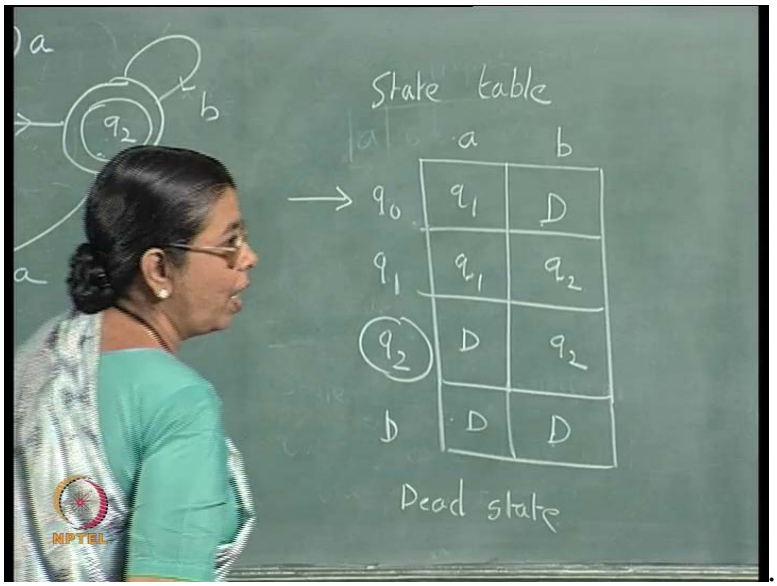
So, let us consider this example, so initially, the machine is in q naught reading the symbol a, what is q naught a? Q naught a is q 1. So, the next instance it will go here input pointer will be moved and you will be in state q 1. In q 1 it is reading a a, so depending upon the state and the symbol it will move the pointer to the next cell and the state will be changed. What is q 1 a? From this diagram, you can see that q 1 a is q 1 so here you will get this. So, in state q 1 again it is reading a a q 1 a is again q 1 so it will move its pointer here and it will be reading this in q 1. Again q 1 a is q 1 so the input pointer will be moved and you will go to the next cell. What is q 1 b in the diagram? Q 1 b will be q 2 so from q 1 b you will get the machine will go to q 2 and the input pointer will be moved next. In q 2 if you read a b you will get only a q 2 the next state will be q 2 now the input pointer will be moved and you will go to this cell in state q 2. In q 2 again you are reading a b so the next state is q 2 so after reading the whole input you are in state q 2 which is a final state. So, initially you start reading the leftmost symbol in the initial state and after reading the whole input if you reach a final state the string will be accepted by the automata. If you reach a non final state then the string will not be accepted by the automata.

(Refer Slide Time: 30:55)



So, let us go to the formal definition. A final state automaton M is a five tuple, M it is denoted formally like this, M is equal to K sigma delta q naught F. Where K is a finite set of states, sigma is a finite set of input symbols, q naught and K is the initial state, one of the states is designated as the initial state. A subset of the set of states is designated as final states, F contained in K is a set of final states. Delta is a mapping from K into sigma into K or you denote delta as K into sigma into k. And the whole thing is denoted by a state diagram like this, the states are marked with circles the initial state is marked with this the transitions delta mappings are denoted like this.

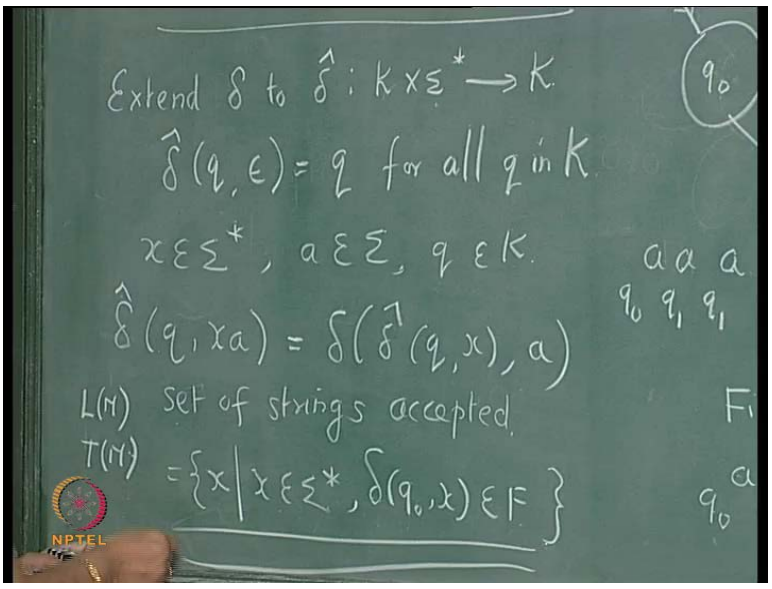
(Refer Slide Time: 32:08)



The whole thing can be represented by what is known as a state table also state table, there will be one column corresponding to each input symbol and there will be one row corresponding to each state q naught q 1 q 2 d. In this particular example, let us consider this example, the initial state is marked like this q naught is the initial state so it is marked like this and q 2 is the final state which is marked like this, the transitions are marked in the state table like this. What is q naught a? Q naught a is q 1 q naught b is d so q naught a is q 1 q naught b is d. What is q 1 a? Q 1 a is q 1, q 1 b is q 2, q 1 a is q 1, q 1 b is q 2, q 2 a is d, q 2 b is q 2 so this is d, this is q 2, D a is D and D b is also D. So, the same diagram you can represent as a table like this. You see that each cell contains a single state and so on. Why this state D is I am denoting as D is once you reach this state afterwards you cannot go to any final state so the string has to be rejected only so such a state is known as a dead state dead state. D corresponds to a dead state here that is why we are calling it as D.

So, we have seen that a final state automaton is a five tuple and it has got five components K, sigma, q naught, f. We have seen what is that, delta is a mapping from K into sigma into K and this transition this is called the transition mapping transition transition and that can be represented by a state diagram like this or by a state table like this. Now, let us define the language accepted in a formal manner.

(Refer Slide Time: 34:59)

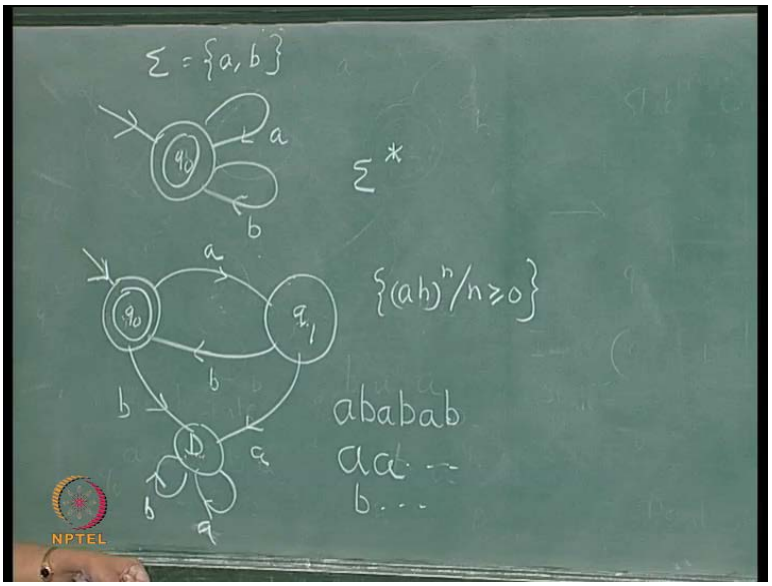


Now, extend delta to delta bar where this is from K into sigma star to K delta is from K into sigma now we are extending it to K into sigma power into K you define like this delta cap q epsilon the empty word equal to q for all q in K and if x is a string and a is a symbol then you define delta cap cos q belongs to K, what do can you say about delta of q x delta cap q x a that is equal to delta of delta cap q of x comma a. In a sense it means like this, you have a string x the last symbol of which a you start in the beginning with a state q, what is the state to which you go after reading this x a that is denoted by delta cap q of x a that is after reading q x you will be in a particular state and from that s a p then from that state you read a symbol a and go to a state. So, after reading x you are in a state and from that state you read a, and go to this state there is the r this is what is meant by this.

Now, if you look at this carefully for a single symbol delta cap q a is a same as delta of q a so we need not have to write delta cap and delta. Actually, delta cap is when you have a string and delta is when you have a single symbol because they are the same when you take a single symbol you need not have to distinguish and write 2 different symbols you can just chose the symbol delta itself.

So, what is the language accepted? The language accepted is denoted by this sometimes it is denoted as L(M), sometimes it is denoted as T(M) where M denotes the finite state automaton that is a set accepter set of strings accepted strings accepted and that is denoted by set of strings of the form x belongs to sigma star delta of q naught x belongs to f. So, the set of strings which take you from the initial state to a final state or when you stay start reading a string starting with the initial state and if you reach a final state such a string will be accepted by the machine that is what we have seen in this example also.

(Refer Slide Time: 38:41)



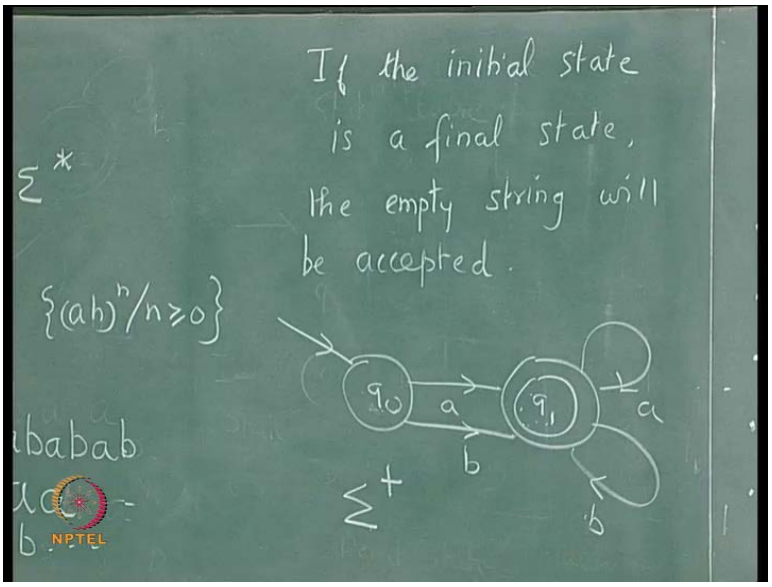
So, the language accepted is denoted like this. Let us consider a few more examples. (No audio from 38:26 to 38:34) Let us take the alphabet sigma is equal to a b input alphabet is this look at this diagram there is only 1 state and you have this diagram. What is the set of strings accepted? Any string if you take consisting of a's and b after reading that starting from q naught you will be in q naught only there is only one state so the set accepted or the language accepted is just sigma star in this case, any string of a's and b's will be accepted by this machine.

Let us consider this example, q naught is the initial state as well as the final state there is a state q 1, if you read a a, you go here if you read a b you go here then there is one more state. (No audio from 39:53 to 40:11) What is the set of strings accepted? If you get a string of the form say a b a b a b start from here a b a b a b you will reach a final state q naught. So, any string of the form a b a b a b a b like that will be accepted by the machine but, if you have a string of the form a a or something like that or a string beginning with a b, what will happen? A a you will go to this afterwards you will remain in state d only similarly, if the string begins with a b you will go here and afterwards you will be in the dead state only, once you reach a dead state you cannot go back to other states so string cannot be accepted.

So, the language accepted is the set of strings of the form a b power n, n greater than or equal to 0. What do I mean by n greater than or equal to 0 is epsilon the empty string will

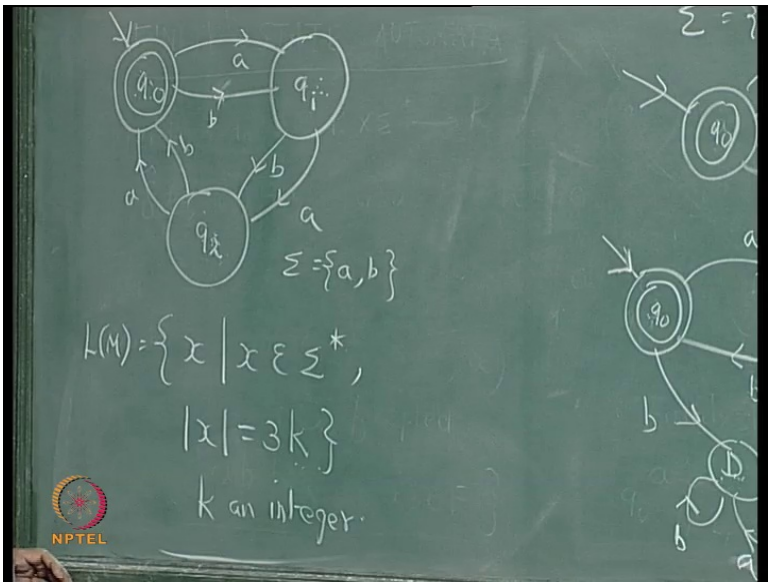
also be accepted by this machine, this machine accepts a empty string also. When does epsilon get accepted? If the initial state is a final state, the empty string will be accepted.

(Refer Slide Time: 41:36)



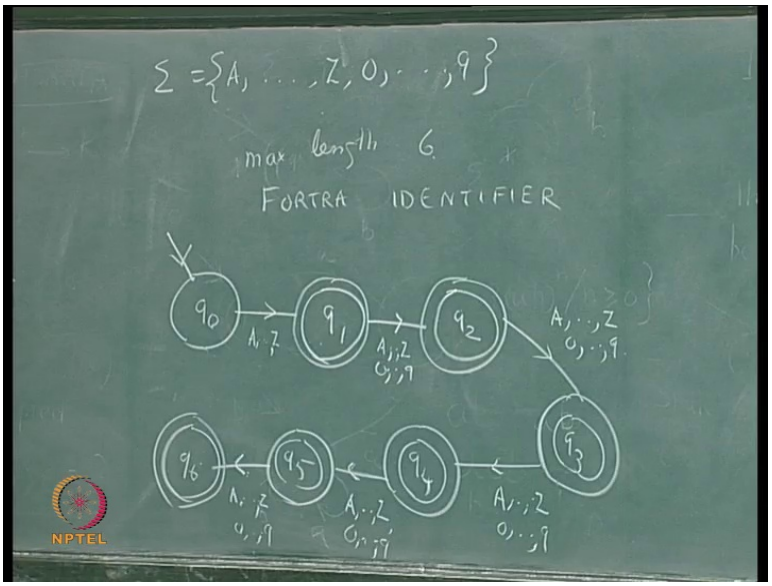
If the initial state initial state is a final state the empty string will be accepted. Look at this diagram, (No audio from 42:12 to 42:28) what sort of strings will be accepted by this machine? Here we are having 2 states q naught and q 1 and q naught is the initial state q 1 is the final state. Here also any string of a's and b's will be accepted, first you have a a then you have and have any string first you have a b and then you can have any string. So, any string of a's and b's will be accepted but, q naught is not the final state so at least one symbol should be there in the string to get it accepted so epsilon will not be accepted by this machine. This machine does not accept the empty string but, any other string of a's and b's will be accepted. So, the language accepted is sigma plus any string of a's and will b's will be accepted but, empty string will not be accepted by this machine so the language accepted by this is sigma plus.

(Refer Slide Time: 43:45)



Let us consider a few more examples, so this idea is clear. Look at this machine, q naught q 1, q 2, a b a b make the initial state as the final state so epsilon will also be accepted. What sort of strings will be accepted by this machine? Here again you see that if you take any string a b a b b b something like that it will be accepted a b a b b b will be accepted. But the condition is the length of the string should be divisible by 3. So, if you take just a a alone starting from here you will go here, it will not be accepted or if you take a five letters 1 a a b a b you will reach the state it will not be accepted, if you take a 4 letter 1 a b b a say starting from here a b b a you would go to q 1 it will not be accepted this any string of a's and b's will be accepted provided the length is divisible by 3. So, the language accepted are L(M) or T(M) is the set of strings x x belongs to sigma star of course, sigma is a b here a comma b and the length of the string is of the form 3 k, k is an integer; k an integer, this is another example like that we can consider several examples.

(Refer Slide Time: 46:08)

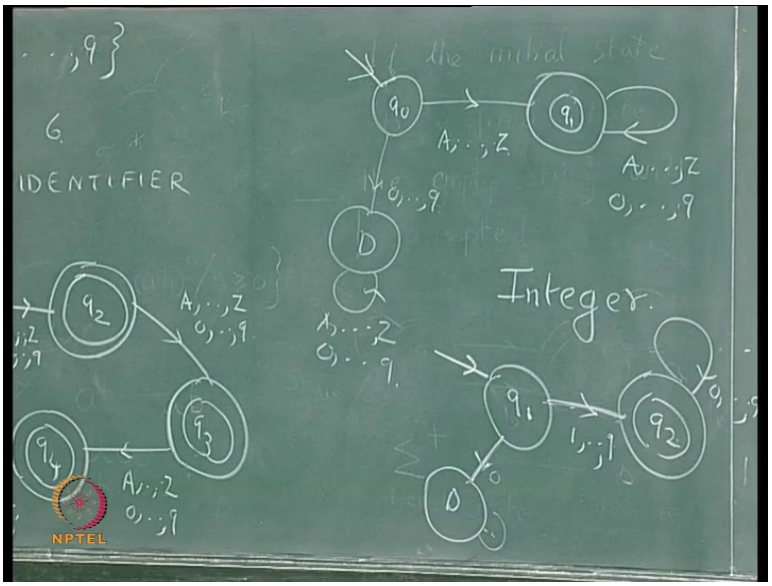


Now, let us see how an interior can be represented by this and a Fortra identifier or a pascal identifier will be accepted by this. Let us take the alphabet sigma to be, I shall used only capital letters A to Z and 0 to 9. I am just taking only capital letters small letters also can be taken without I mean in some other example but, here the alphabet consist of the symbols A to Z and the digits 0 to 9.

Now, a Fortra identifier has a maximum length of 6 maximum length 6, Fortra identifier and it begins with a letter A to Z and afterwards you can have letter or a digit. So, that can be represented by a state diagram like this we start with q naught the initial state then q 1, q 2, q 3, q 4, q 5 q 6 so it begins with the letter A to Z. Starting from q naught if you get a a or a z a b c d any z you go to q 1 and there it can be just that, you may end there also. So, this also happens to be a final state but, you can have 2 letter identifiers so the second symbol can be A to Z or 0 to 9 A to Z or 0 to 9 then you go to q 2 again you can end up here or continue further so this also happens to be a final state.

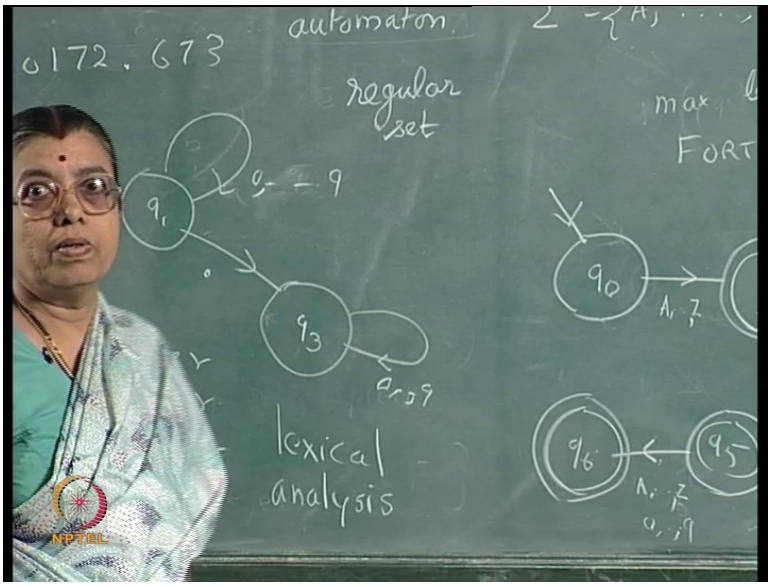
Then 3 letters an identifier consisting of 3 symbols the first one of course, is the letter second one is a digit, third one can be a digit or a letter so you go here. The fourth one is again it could be a A to Z or 0 to 9 then this can again be a final state you can have a fifth letter which is as A to Z or 0 to 9, again this can be a final state this also can be a final state 6 letter can be letter from A to Z or 0 to 9. In pascal you do not have the limit on the length of the thing.

(Refer Slide Time: 49:18)



So, suppose I use only capital letters then you can have like this, any number of symbols you can have, the first letter has to be A to Z first letter then you may get q 1 A to Z or 0 to 9. Of course, I should also have something like the identifier cannot begin with the digit so if you get 0 to 9 I should go to a dead state and like this A to Z, 0 to 9. So a Pascal identifier can be given by this a Fortran identifier can be given by this an integer beginning with a one can be denoted an integer we should not begin with a 0, how do you represent and integer using a finite state diagram? Initially, you have q 1 the first symbol you do not want to be a any integer, decimal integer so 1 to 9, if you get you go to q 2 then it can be followed by any digits 0 to 9. So, this is an this diagram represents an integer of course, when you get a 0 you do not expect it to begin with a 0 you go to a digit and so on.

(Refer Slide Time: 51:03)



So, you see that you can represent identifiers and integers using a state diagram. You can also represent something like know 10172.673 like this a decimal number like this using a tested diagram. Initially, you will have an integer q naught, q 1 then it can be followed by any digit, then a decimal point so here I am just writing those symbols will lead you to find states here the alphabet will also consist of a dot. So, when you do not get a proper thing you have to go to a dead state that portion of it I am not marking on the diagram q 3 and in q 3 you may get any digit 0 to 9, see here 0 to 9.

So, this represents, this is not now, this represent a decimal number in lexical analysis part of the compiler you have 1 final state automaton to represent the identifier 1 to represent integer integer and 1 to represent keywords and so on. So, all these final state automaton will work together on the input. The input is the program is taken as a sequence of symbols, it will work on this. And whenever you recognize an identifier or a input or a special symbol logical operators like that you will take you will take it as a token and that will be entered in a proper table, a symbol table or a hide in the constant table and you will put it there and corresponding action will take place then the parsing will start taking place. So, that recognizing each one as a token is called the lexical analysis and that is called a lexical analysis. And in this part, the finite state automaton plays a very useful part.

Not only that in text editing suppose, I want to replace a particular portion of a text by some other thing that particular text alone can be represented by a finite state automaton and in that you will remove that and replace it and so on. So, the same idea is useful in text editing also. So, this is one of the very important use of finite state automaton and one of the things you must notice here is that when you are in a particular state and when you get a next symbol, the next state is uniquely determined. So, whatever we have considered today is called a deterministic finite state automaton. Whatever we have considered that is called a deterministic deterministic finite state automaton. In contrast to this you may also have non deterministic finite state automaton about which we shall study in the next lecture.

In the deterministic finite state automaton, the mapping is from K into sigma into K whereas, in the nondeterministic finite state automaton it will be from K into sigma into finite subsets of K. We shall consider some examples and see whether the power is really increased or not. The power will not be increased nondeterministic automata also have the same power. One more thing is the language accepted by a finite state automaton is called a regular set; it is called a regular set. So, in the next class, we shall see more about finite state automata in fact nondeterministic finite state automata and also me minimization procedures.