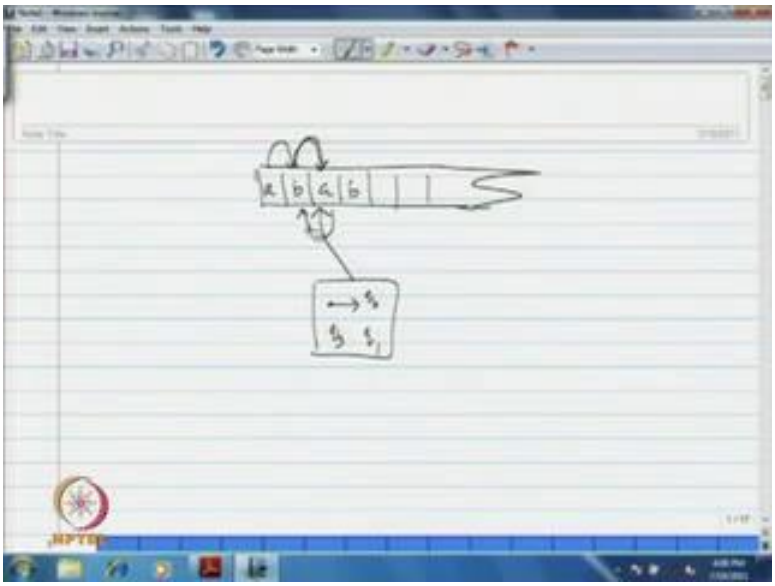
Formal Languages and Automata Theory
Prof. Dr. K.V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

Module - 6
Variants of Finite Automata
Lecture - 1
Variants of FA

(Refer Slide Time: 00:40)

In today's lecture we discuss about some of the Variants of this Finite Automata, you have seen these finite automata as a language acceptor; that means as a language recognizer when given a string of supply that to initial state. And see whether you are reaching to a final state or a non final state according to that you decide that the string is accepted. So, that is how this finite automata we have seen as language acceptors, so among these language acceptors you have known the variant of DFA Deterministic Finite Automaton. You have already learned NFA non deterministic finite automaton in which you know you have Epsilon transitions and for which each transition you can have, if you apply any symbol in a particular state you can have several transitions. So; that means the non deterministic is defined and we have observed that these two are equivalent also, and now there are some other variants in finite automata like two way finite automata, two tape finite automata.

(Refer Slide Time: 01:33)



Like that we have in case of this two way finite automaton if you recollect, this thus schematic diagram of DFA or NFA, there you have input tape divided into two cells. And the input symbol that you are placing say like this and you have a finite control from which a reading and reading head connected to this input tape and there the state components that you have a pointer.

So, the current state is pointed by this pointer, the left is justified, right side infinite tape divided into two cells, the input you are placing on this tape and in the initial state. If you start from the left most cell by the time of completion of this tape I mean that input that you have given, if the current state is final state then you say that the string is accepted that is how the notation we have.

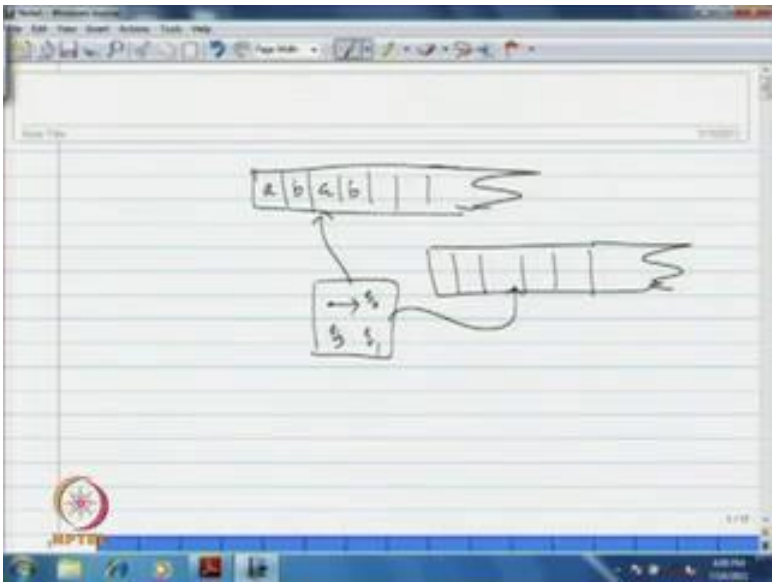
In case of this two way finite automaton, the situation is you look here in case of DFA or NFA, this reading head is after one transition it is taking from this cell to this cell it will move automatically. And then another transition from this cell to this to this cell, this will move and continues till you know the input is exhausted; that means, exactly there will be n transitions for a n string of length n.

But in case of two way finite automaton, we allow this reading head to move left and right in each a transition of course, either left or right in each a transition. So, that is what is the two way mechanism; that means, for example, in a particular transition you might be moving from this cell to this cell, in a next transition you may move back to cell, but

of course, in each transition one left move or one right move of course, this is a reading head only.

So, this reading head will read a symbol and looking at the current state, so you will supply transitions to move right as well as left. So, this kind of mechanism is allowed to I mean in two way finite automaton and in case of the variant two tape finite automaton what you have here we have one only.

(Refer Slide Time: 03:52)



Here we have only one input tape, but in case of other variant you maybe allowed with one more tape and of course, input we will always be given input will given on a one tape only say for example, this is a first tape on this you are given an input. Now, another reading head will be connecting to this and this will be used for such some computation some rougher some sort of things like certain things that you can remember by storing some information on this second tape. And you will be doing the computation on the input of the first tape only and pursue the input and accept by final state.

So, this kind of variant is also there like allowing two tapes of course, left justified right side infinite tape and two reading heads from the finite control will be connected one each to the tape and the computation will be you know processed on the input of the first tape. So, this kind of variant is also there in the literature the two tape finite automaton, but here is a point that like NFA or DFA are equivalent with their accepting regular

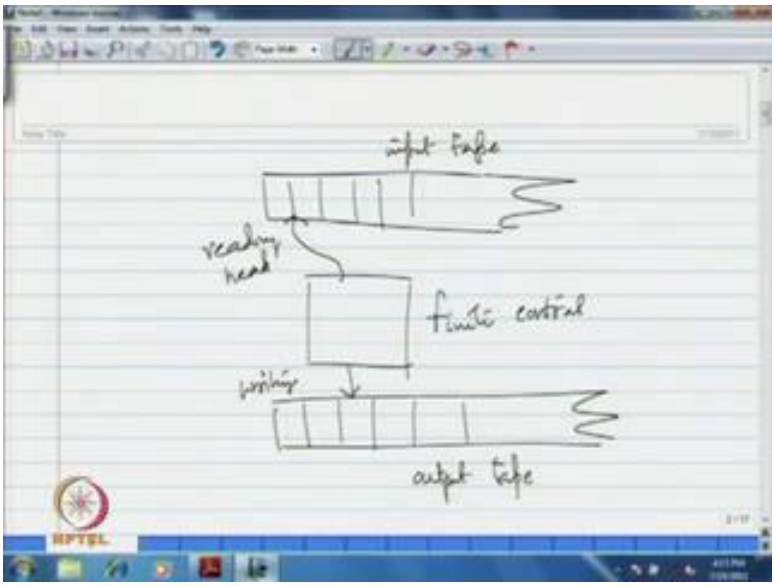
languages, two way finite automata are two tape finite automata these variants also precisely accept regular languages.

So, these results of course, I am not proving it here in this lecture, but you can make a note that these are no more power than DFA; that means, this gives you certain facility like NFA for a given regular language you can construct an NFA very sort of easily compare to the DFA. Sometimes, similarly two way finite automata may be helpful in order to accept certain regular languages very Quickly you can confect two way finite automata relative compare to that of DFA or two tape finite automata.

May be useful in the sense, that maybe give you facility to construct finite automata very Quickly this variant, but of course, the language the class of language accelerated by these any of these things may not be bigger is not bigger than the class of regular languages they are precisely accept the regular languages. Now, these are the part of language acceptors, some variants I have presented, now there is a another notion called finite state transducers.

So, these are actually the sort of like output divisors here in the language acceptors you are giving the input on the tape and you are just seeing that whether the input is accepted or rejected. So, that is what we are seeing in case of language acceptors, but in case of finite state transducers what is a mechanism there you give the input and expect some output. So, this kind of variants are available in the literature and of course, these are also philosophically equivalent to this a finite automata, the class of finite automata like BFA NFA this kind of things. So, in which I just mentioned these two classes like mealy machines and Moore machines.

(Refer Slide Time: 07:12)



So, in this case like here you have one input tape, of course, now I called it as input tape, a finite control and a reading head to this input tape and now we assume one output tape again divided into two cells and one this writing head will be connected to the output tape. This is input tape, this is output tape, this is a output tape, this is writing head of course, this is finite control as earlier in case of DFA, this is writing head, there are this is reading head here, this is reading head and this is writing head.

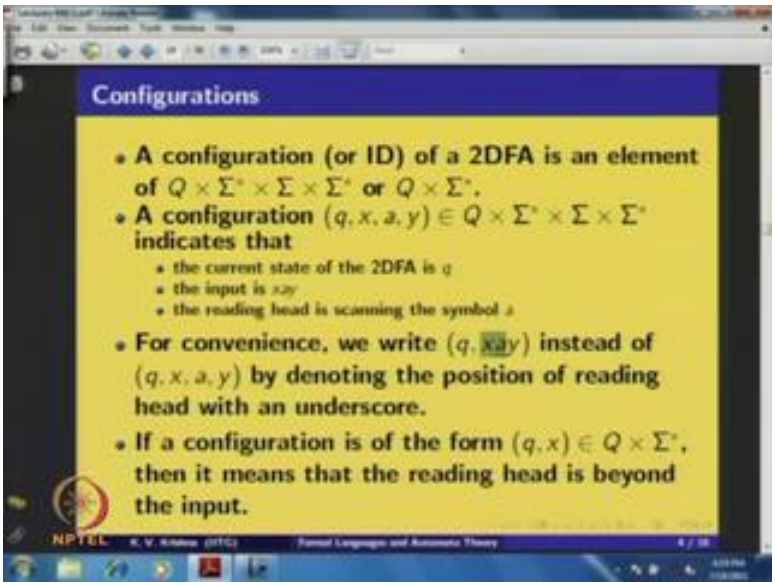
So, in case of Moore and mealy machines this kind of mechanism is allowed and you will be giving inputs on this input tape and according to the transitions some output will be left on the output tape. These kinds of divisors as finite state transducers are available in the literature, among the variants of finite automata. Now, let me just introduce some of the variants and how they are accepting languages are you know how they are giving the output that I will explain through sets of examples.

(Refer Slide Time: 08:54)

Let me start with a two way finite automaton, let me formally define that because, whatever the way that it is working, it is supposed to work that I have already explained that in each transition you may be allowed to go to left or right, other than going the changing the state. So, formally a two way DFA, I am a simply called it two DFA, that is Quintuple, again Q sigma, delta Q naught F like a BFA, where this Q sigma, Q naught F are as in BFA, I mean state set input alphabet Q naught is initial state F.

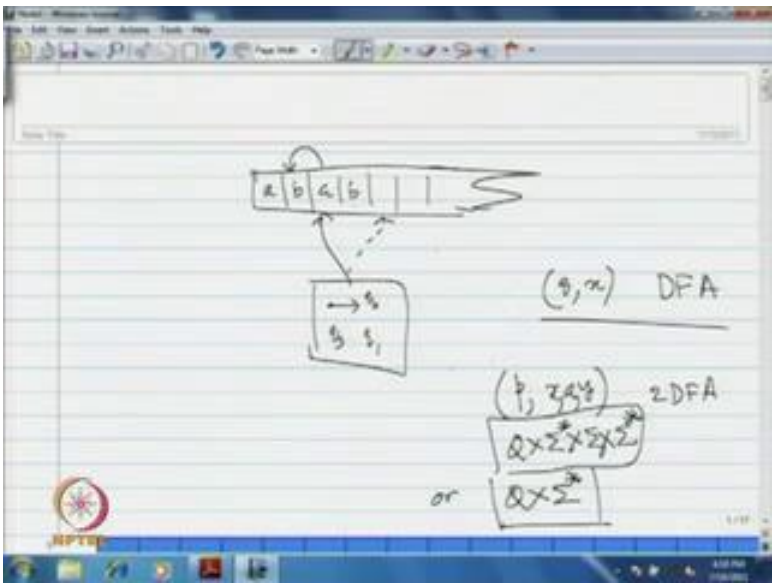
The final state, but the transition function delta, in case of DFA what you have given an state in input symbol you have an next state only, one state is assigned, but here given a state in the input symbol you will be assigning a state and in addition to that you can move either to left or right, you are the transition there in case of DFA you will be moving only to right. Therefore, it is not mentioned, but here you are allowed to move left also; that means, in a transition you may the transition function is thus a function from Q cross sigma to Q cross set of LR where L is indicating a left move and R is indicating a right move.

(Refer Slide Time: 10:17)



Now, to talk about the computation and the language acceptance all those things, now again formally let me discuss about the concept of configurations and computations. All those things here in case of two DFA also a configuration or instantaneous description of two DFA is an element of Q cross sigma star cross sigma cross sigma star are it is an element of Q cross sigma star.

(Refer Slide Time: 10:51)



Because, you look at In case of in case of this two DFA like this is the input given to you, now at this point of time it is pointing to this current state Q naught and you are

reading symbol a, now the situation is in case of DFA you have just mentioned the current state. And whatever the string that you are going to read that much you only will be mentioning here; that means, for example, here the current symbol is a and after that what are these string that is only you are interested in what are the string left to the current position of the reading head you do not require.

Because, you are not going to anyway visit that string, but here you have a possibility to come back that is why you have to mention the entire tape contain and where the current where is the reading head what is the current position of the reading head all the things need to be mention. So, that way in case of DFA, if I would have written just that q x, where the first symbol of x is the currently reading symbol, if this is in case of DFA in case of two DFA I required the total information of the input tape; that means, if the current state is p and say for example, x a y, if I write.

So, the string which is left to the current symbol that we are reading say a they left to this x and right to this y. So, this is how the configuration in two DFA we have to declare, because when the transition is taking to left you have to visit what is the contain of the tape left to it. So, in the because of this reason now you see the element is the state component you have to mention. And left to that whatever is the tape contained that is the string and the element of sigma star a this is an element of sigma the currently reading symbol and then whatever is the string right to that there is a sigma star.

So, there is a configuration is an element of this r we have mentioned that it is an element of Q cross sigma star what is the reason for this for example, you have finished reading this input and you went beyond the beyond the input; that means, say for example, this reading head in a particular situation. Suppose, reach to this position now it is not reading anything; that means, we do not have any transition defined further because, we have defined the transitions from the for the current reading symbol with the current state.

If you know then according to that you can talk about what is the next state and to which it has to move say for example, from here after reading b, if you are asking to go to right; that means, you have gone beyond the input and thus you are not going to have any the current symbol to reach and there is nothing right to that. That means, the tape contained that left to that will only be there that is the element of sigma star and thus the

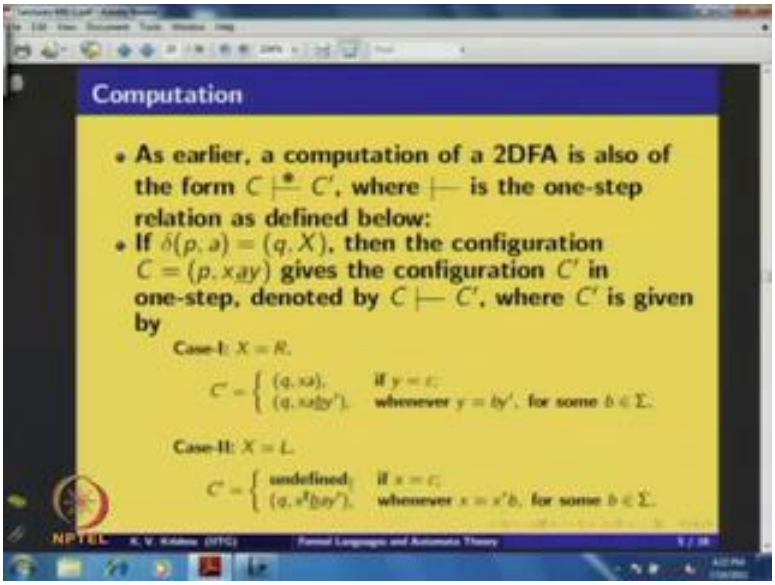
configuration has to be an element of Q cross sigma star in which we are not going to have any current reading symbol thus anything right to that.

So, that is why formally we defined configuration or instantaneous description, if you take a snapshot or instantaneous description of this you have you can use this information; that is sufficient and thus formally it is a element of Q cross sigma star cross sigma cross sigma star this is one option, another option is Q cross sigma star, when the reading head is going beyond the input.

So, I have been mentioning that the configuration q x a y is the element of this indicates that it has just I have explained the current state of two DFA is q, the input is x a y that is a total strings on the tape the reading head is in the currently scanning symbol a. So, that what it is a for convenience instead of writing this may when four components, we may simply denoted as two component I mean that is as pair, but we have to mention what is a currently reading symbol.

So, for convenience we make a mention the configuration as q x a y and with an underscore at a to understand that the current reading symbol is you know the reading head reading head is at this particular position and this have second type of configuration that q x; that means, that the reading head is beyond the input as I have explained.

(Refer Slide Time: 15:19)



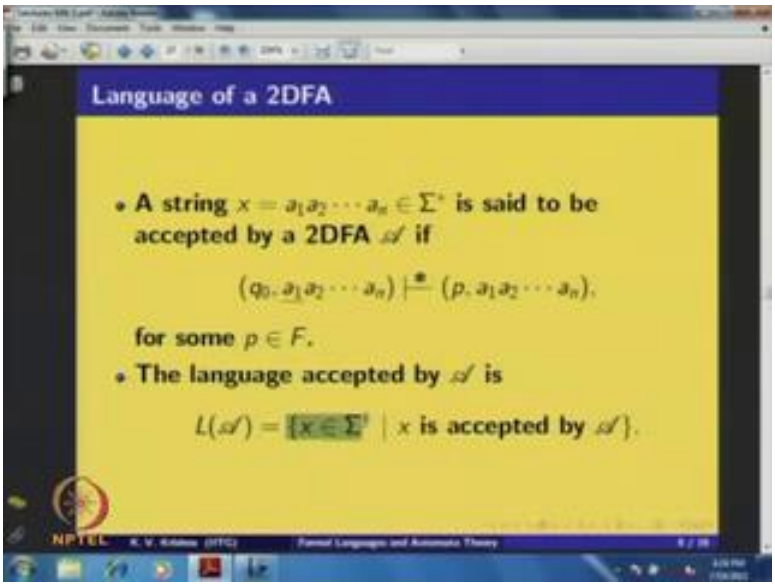
Now, let us come to the part of computation into 2 DFA as earlier in case of DFA computation is of this form where this is the relation that is a one step relation between configurations. So, this is a reflects to the transcript colder of this relationship as earlier, now let me define one step relation in con in the context of 2 DFA, let us look at this because this one step transition is because of the transition function delta is a current state and a current reading symbol based on the whatever is the transition given to you that transition has to be implemented under on the instantaneous description on the particular configuration, that is how you are getting next configuration in one step.

So, how we are defining this one step relation that is if delta of p a is the current state is p and where your reading symbol is a, if the transition is giving you q x, where x can be you know either L or R. Then the configuration C, where the current state p with the input x a y with a as a reading thing gives the configuration C dash on and we are denoting it with this relation C related to C dash in one step in this following cases.

In case x is right what do you understand there are again two things, one is if y is a Epsilon; that means, there is a no string right side to the current currently reading symbol then you are changing to state Q and then the reading head is going beyond the tape; that means, this is a element of Q cross sigma star that kind of configuration is coming. There a C dash or in case you have some string which is non empty, then I can always mention it as b y dash, because a right side string some non empty string is that is of the form b y dash for some b in sigma, then the currently reading symbol will be b next to that a.

So, this kind of configuration will be now we see as C dash it is in case of if x is L, then now you look at the left side if there is nothing left to that; that means, you are going beyond the tape and in this situation we say the machine hangs. So, we do not have any definition, if left to the current symbol if there is no string or there is no cell to move otherwise of course, there if there is some symbol that particular symbol will be the currently reading symbol in C dash, so thus the configuration is of this form. So, this is how we defined one step relation this is how we defined one relation and then that reflects to transcript colder of that relation, one step relation we denote by star of that relation and that is how we defined the computation.

(Refer Slide Time: 18:15)

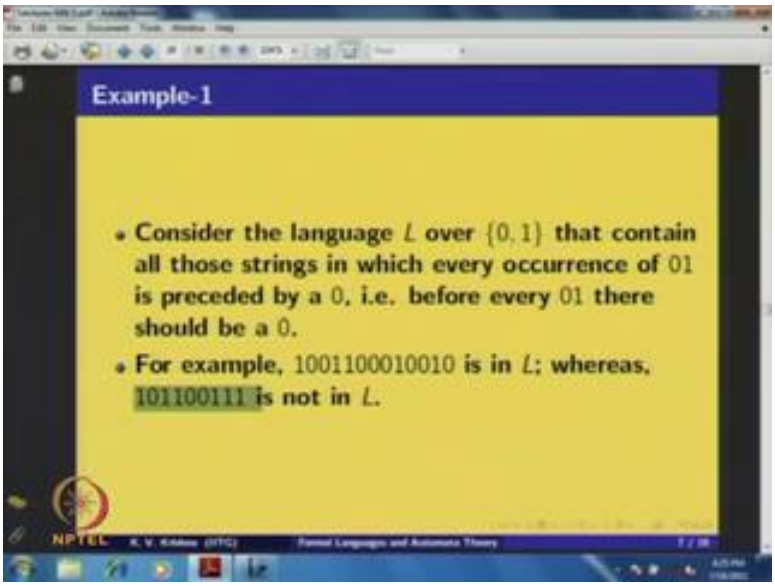


Now, if you give a string a 1, a 2, a n in sigma from sigma star, we say that is accepted given 2 DFA, if you give this in initial state start with the initial state, where the currently reading symbol is a left move symbol of the input infinitely many steps. If you because, the input we are not going to transform many thing, but the reading head if it is going beyond the input.

Of course, with the situation that it is to the right move stand, because the reading head is going beyond the input does not mean, that you will you will go towards left hand and hang there here the configuration. We are defining only when the reading head is going beyond the input right side, because where the cells are existing because, we have right side infinitely many cells, the reading head reading head has to go out beyond the input, towards right of the input.

So, if you get this kind of configuration p a 1, a 2, a n without having the indication of that the reading head for some p in F, then we say the string x is accepted by the given 2 DFA. Now, as earlier we can now say that the language accepted by 2 DFA, a is of course, denoted by L of a is set of those all strings in sigma star that is accepted by a.

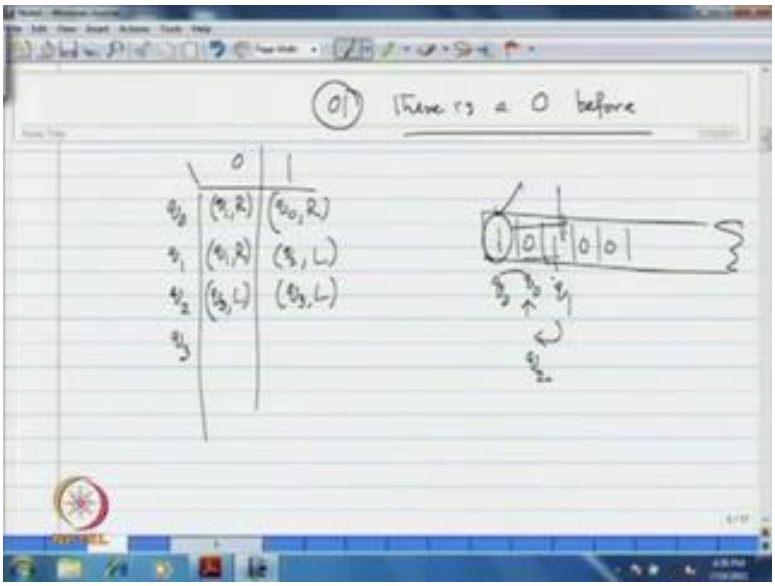
(Refer Slide Time: 19:39)



Now, let us look at some examples because how this variant of 2 DFA will be helpful in order to understand some regular languages, now I mentioned this language here, consider the language over 01, that contain all those strings in which every occurrence of 01 is preceded by a 0, whenever you find this 01, then you should have 0 before that. This is a regular language that of course, you can find the DFA for this.

Let me just spell out some instances of this language, this is an element, this is a string in L, you just look at for example, if I take this 01, first 01, I have paired to that 01, if we consider this 01 I have 10 before that for this 01, I have 10 before. This in this example for this 01 I have a 0 here, but for this 01 I do not have a 0 before this, so this is not a string in L. So, I hope now you understand what kind of language is under consideration if you take any sequence of zeros and ones a string over 01, whenever you have an occurrence of 01 there should be a 0 before that 01, that is what is the criteria for this language.

(Refer Slide Time: 21:10)



Now, let us write constructing a 2 DFA for this for every occurrence of 01, there should be a 0 before that is what is the there is a 0 before this for every occurrence, there is a 0 before this. Now, let us let me consider this as the tape and the input is given on this and the input symbols are from 01 start with the state say Q naught, now you keep looking at the symbols going to the right on this tape for example, from here I am going towards right I scan for 01.

So, whenever you are getting 01 you remember that and then go back to cells left to that go back to cells left to that and see that particular cell is having 0 or not, that is how I will be designing here. So, that means, in the initial state q naught, if I receive 1, I do not have to or read to remember or anything, I simply go to right in the same state. Because, state component is remembering certain things, if I have 0, I have to now wait whether I am getting 1 to come back that is why I am changing the state, let me call it as q 1 and go to right to scan the next symbol.

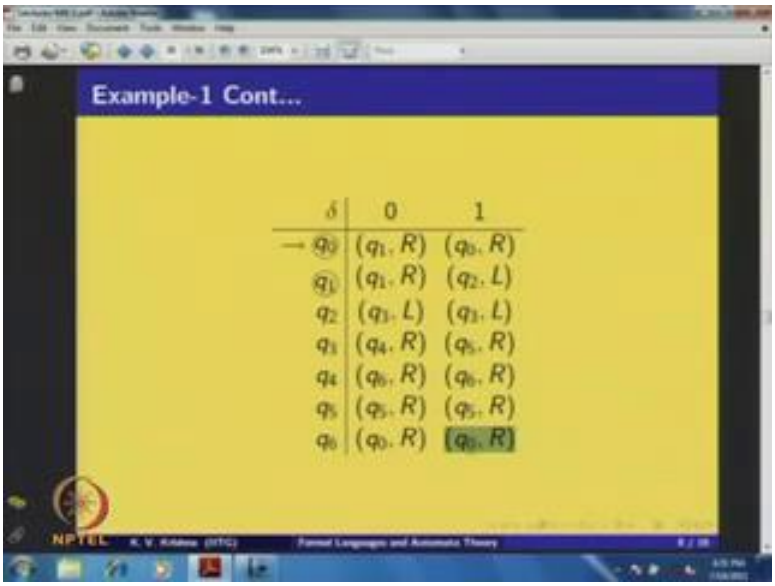
Now, in q naught I have defined the transition for 0 and 1 now let me consider q 1, in q 1 the situation is if I have 1 in q 1, now I have to move to two cell left, so I will come to left now let me call it as q 2, if in q 1 I have received 0; that means, I will continue to wait for 1. So, in the same state I continue to go to right that is how we defined here, now in q2 for example here, I have 1, 0, 1 say 0 like that.

Of course, this string cannot be accepted by because before this 01 I have 1, but not 0, so when I am reading this 1 I continue in q naught and move to right I read this 0. So, at this particular place I am in q naught. When, I move to right here also I am in q naught, after reading this 0, I move to q 1 and go to right; that means, while reading this 1, I am in q 1. So, in q 1 when I am reading this I move to q 2, but come back to this left cell; that means, I am coming to this q 2, but this cell I am reading now, the second cell I have to check this symbol is 0 or not, that why I have to come back in q 2 once Moore; that means, I have to remember you know the number of cells that I have moved.

So, in q 2 we are in; that means, one cell is moved we know that is always 0 here, but I am not worried about it I am not worried about it. So, you just move once more; that means, moving is now denoted by q 3 move to left and move to left of course, you know that when you are in q 2 you will be receiving 0 only, but reading one is not possibility we know we are not worried we just defined it like this.

Now, in q 3 you have to cross check now in q 3 you have to cross check whether you are having in this position 0 or 1. If you have one you continue the transitions and keep going to right beyond the input and you do not accept it, you can put a state which is not a final state. For example, if you have a 0 here, what do you do then you keep processing this way that you continue the transitions which continues beyond this 01 and then from this point. Again, you have to repeat the same mechanism, you keep searching for 01 and come back whether you have 0 or not you cross check and so on.

(Refer Slide Time: 25:41)



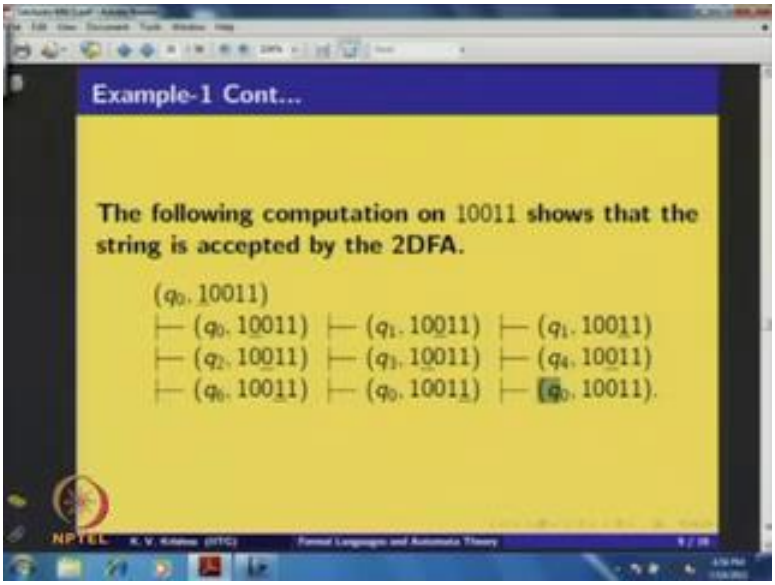
So, with this logic if I pursue of course, see here I have defined it as this in q naught I am changing to when I am writing 0, q 1 and q naught in q 1, we are continuing to write and changing to q 2, and coming back to left and in q 2, I am coming further back in q 3, if I am receiving 0 that is the situation that I will continue to cross check again. Whether, 01 is existing or not on the tape and whenever 01 is existing and I will come back to cell and see whether I have 0.

So, whenever it is failing; that means, in q 3, if I am getting 1; that means, before 01 I do not have 0, but 1 is there, then in that situation I am putting it in q 5, in q 5 you just cross check that what I am doing is whatever the symbols I receiving whether it is 0 or 1, I am just keep I am just going a right; that means, I will go beyond the input and q 5 is not declared as a final state. Thus, the input will not be accepted on the other hand of I have received you know 0, there then I am putting it in q 4, you see I have to pass 2 cells.

So, I will be passing these 2 cells 0 like the input 01, so to indicate that I have these two states q 4, q 6, after passing 2 cells, I will be putting the setting back to q naught and q naught will pursue the input to cross check whether again 01 is existing before 01, whether 0 is there or not like that it is pursue that. So, thus you see whatever the condition is exactly given to you for every occurrence of 01, before that 0 is there or not that is to be cross checked.

Now, we are not characterizing this language, so I mean characteristic properties of this language you can analyze and probably you can find the DFA corresponding to this, but you see whatever way the language is presented, the property is given that if 01 exists before that you have to have 10. So, that means, you have to come back on cross check, unless we have the mechanism, because just as it is if you want to implement a finite automaton, unless we have the mechanism to come back means; that means, the reading head if it if it cannot come back, then you are unable to see what is the particular symbol before that. So, this particular variant will be helpful to construct a variant of finite automaton mean finite automaton to accept this kind of language and the property whatever is given as it is you cross check and constructed.

(Refer Slide Time: 28:07)

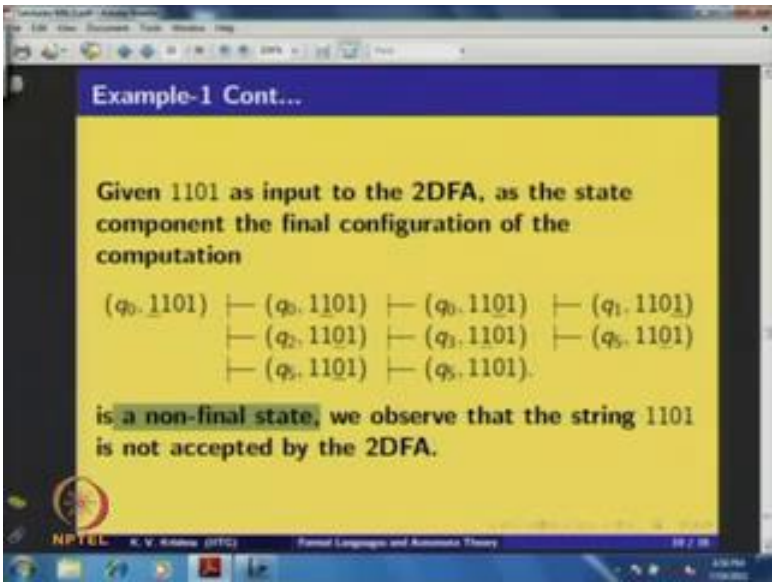


Let me look at these some computations in this particular a finite automaton, if you take this string 10011, you see before this 01 occurrence of this 01 you have 0, now if you just follow that DFA I mean the 2 DFA. Whatever is constructed and implement this is the computation that you can realize and see at the end of that the finally, the reading head goes beyond the input in the state q naught and we have declared that both q naught and q 1 as final state and see that the string is accepted, because of this configuration.

And you can verify this computation because as I have mentioned in q naught you are reading 1 and you will be continuing to write and you will be continuing to write you see this underscore is mentioning that what is the current reading head position. So, after

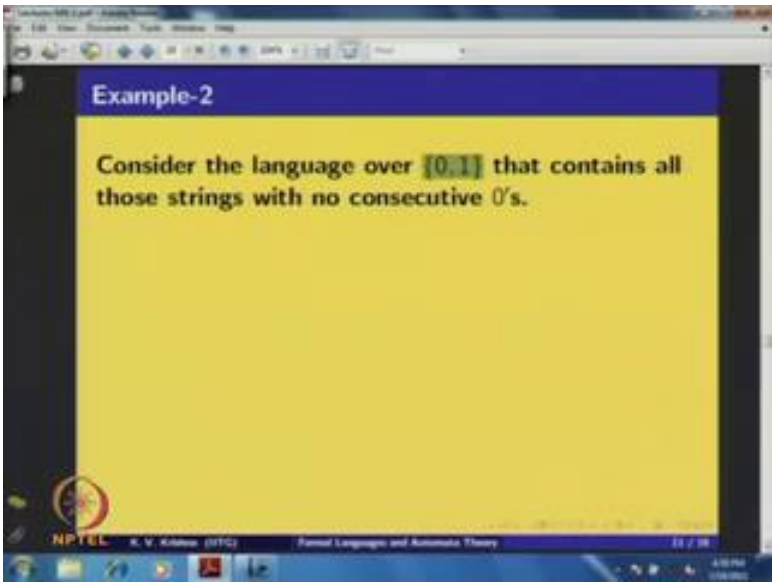
completion of this 01 you will be coming back you will be coming back to this 0 and since it is 0, you are going again now these 2 cells right, that is what y F, q 4 and q 6 that we have indicated and in q 6. At we will again it will set back to q naught and it will continue to cross check whether 01 is existing and after this portion there is no 01 and once it goes beyond the input the current state is q naught. And hence thus the string is accepted, that is how the computation is presented for this on the string.

(Refer Slide Time: 29:34)



Now, if you take this string 1101 you can clearly see that this is not a string in the language and if you observe the computation that is as follows and finally, you are having this as a final configuration the reading head is going beyond the input. Of course, to right and the current state of this configuration is q 5, and thus you see that this configuration is not having a final state as a state component, this is a non final stator and hence you know we say that the string is not accepted by this 2 DFA.

(Refer Slide Time: 30:15)



Now, let me give you one more example if you consider the language or 01 that contains all those strings with no consecutive zeros of course, for this I hope that you have constructed the DFA by now, but no consecutive zeros. Suppose, if I say; that means, they should not be any 2 zeros in the input, so 00 is not a string; that means, here you can cross check whenever you have 0, you just come back and you can cross check, whether you have a 0 there.

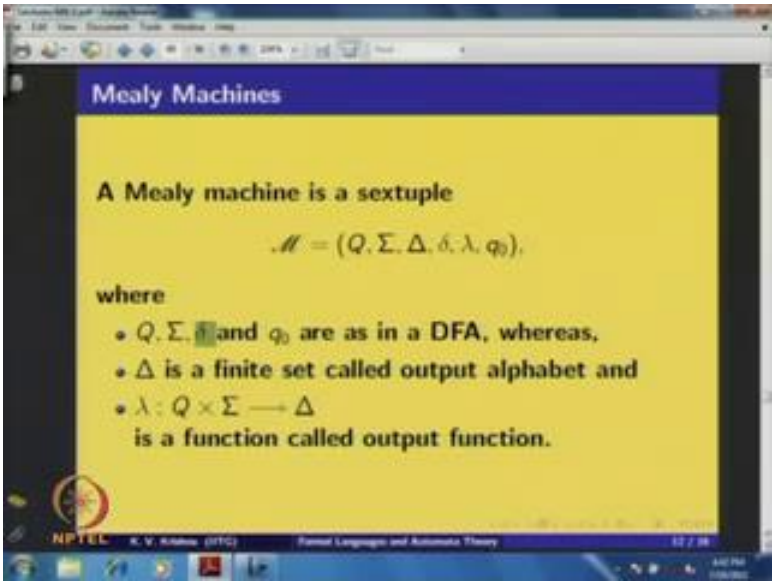
If you have one you are happy to continue further, if you have 0 there because, whenever you are getting 0, you just come back and see what is the current what is the symbol before that. So, here you have an advantage here you have a mechanism that you come back and cross check, so I can state this as a property for this no consecutive zeros, so you have the DFA.

Now, you can construct the 2 DFA using this you just whenever you are getting 0, you just come back and cross check what is the current symbol, now take this property to account and now you try constructing the 2 DFA for this and you will understand that you can get a you can get a better advantage in understanding this language through 2 DFA. Now, we will intro introduce another variant of finite automata as I have concentrated on one particular language acceptor.

Now, I will go to final state transducers and define and formally introduce those things and discuss certain examples, because here I have mentioned that there are two types of

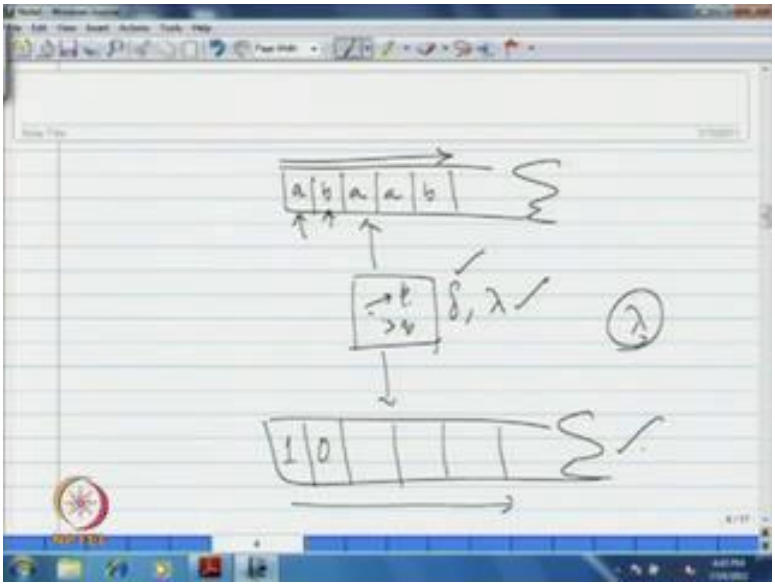
variants one is as a language acceptor another is a finite state transducers. So, we have touched this is a 2 DFA, now let me introduce one finite state transducers

(Refer Slide Time: 32:12)



In that class I am considering Mealy machines as mentioned earlier here the situation will be like this.

(Refer Slide Time: 32:22)



Because, you have an input tape finite control the reading head to this a writing head to output tape that is how it is, so corresponding to that we require the number of components here, that we thus we declare as x triple a mealy machine m state

component. So, here state component that input alphabet transition map the q naught as initial state thus as in BFA

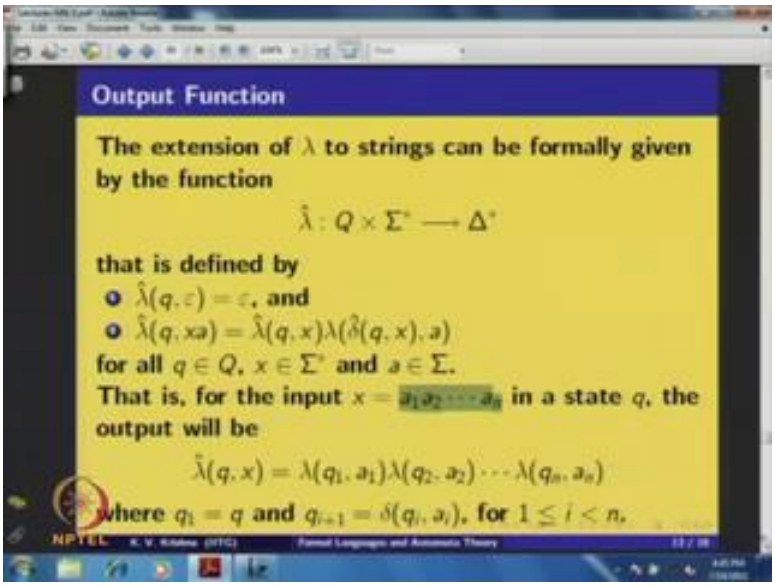
Now, whatever the new thing that you are seeing the delta is here of course, that is again a finite state called output alphabet, but the transition function is not when the transition function you know there that is as in BFA. That means, given a state an in input symbol you are getting a next state assigned, but another new component that you are seeing is little lambda, that is this lambda is a function from Q cross sigma to delta is a function called output function.

So, you that means, what it is doing given a state under the symbol you are assigning an output an output symbol to this, so Mealy machine is formally defined like this. So, if you are reading a particular symbol on the input tape, immediately you will be changing the state. Through, the state transition delta, that our transition function and simultaneously you are printing a particular symbol on the output tape. For example if this is a situation and what is the current state say for example, you have the input a b a a b for example, and while reading a if you are answering to move from p to q.

So, from the current state p it will move to q and on the tape for example, corresponding to a, when you are reading in b, if it is assigned that we have to print 1, it will print 1 here, then automatically this reading head goes here second place. And what is the corresponding state on the symbol whatever the assignment is given to you will change through the state and according to that whatever you are asking to print by delta the output function lambda.

So, you will be the printing the output here and so on, so by the time you finish this finish reading this you will be printing this also simultaneously according to the transition and a delta and the output function lambda given in the finite control. So, that is the behavior.

(Refer Slide Time: 34:57)



Now, situation is we have mentioned this finitely here, finite states are here, involve the transition function is Q cross sigma to q, of course,, you know in DFA how to explain these two naturally to all the strings similarly. This output function this is define finitely here; that means, state and the symbol is given an output symbol is assigned here, now you can extend this to input strings; that means, the elements of sigma star.

Naturally, that is how that is given here the extension of lambda to the strings can be formally defined can be given by this function I am calling it as lambda cap, that is from Q cross sigma star to delta star, that is defined by as I have mentioned in this mean by mechanism you read a symbol you change the state. And simultaneously you print corresponding to that whatever lambda is saying and then you are going to the next place.

Now, I am reading b depending on the current state when you are reading b order, the state it is asking to change by delta, you change that state move to right one more cell here and according to the lambda. Whatever, you are asking to print it here; that means, whatever the sequence of input that you are reading here, corresponding to that while finishing this phi symbol, you will be printing for this phi symbol here and that is what is naturally need to be considered here.

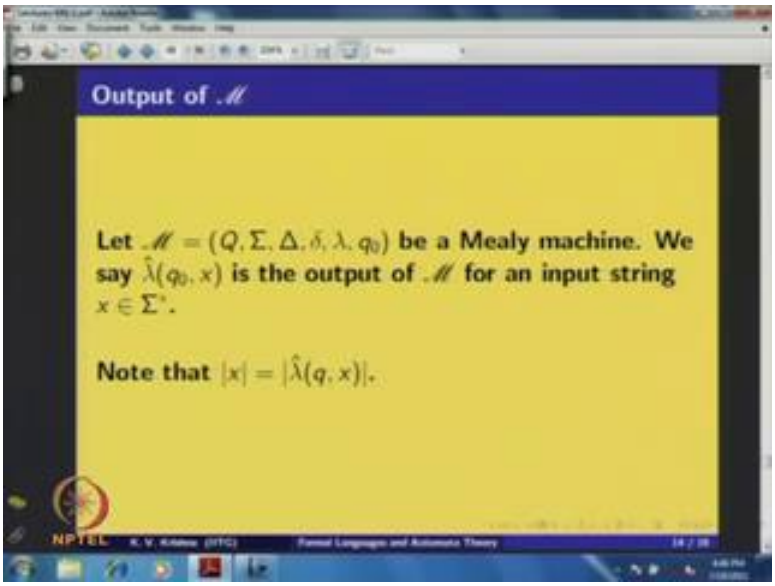
Formally, if you are reading m t string on the tape, you will have m t string as output, if you are reading x a, the string of the form x a, where a is a symbol of sigma and x is a

string. Then, whatever the output for x you give it and in the current state after pursuing x that is what is delta cap of Q comma x that you know in this state, if you apply a whatever is a symbol that is to be printed that is what is given by function lambda that symbol lead to be just suppose to the string lambda cap.

This is how inductively we have defined that is how exactly what I have explained, so that is for the input a 1, a 2, a n in the state q, the output will be the output will be that is lambda cap of Q comma x will be in the state q 1, you apply a 1, where q 1 is a state q because that is a current state. Whatever, the symbol that you printed here now you go to q 2 what is the q what is the state q 2 in q 1, if you apply a 1, you are going to q 2 that is what I assuming.

So, in q 2 if you apply a two whatever the output symbol you are asking to print that you will be printing in the second cell and so on in q n; that means, in q n minus 1 if you apply a n minus 1, whatever the state that you are getting I am calling it as q n and in which you are applying a n. And, whatever the symbol that corresponding to that is assigned by lambda need to be show here, you understand that a 1, a 2, a n, if it is input the output sequence will be this and the length of the input is same as the length of the output here.

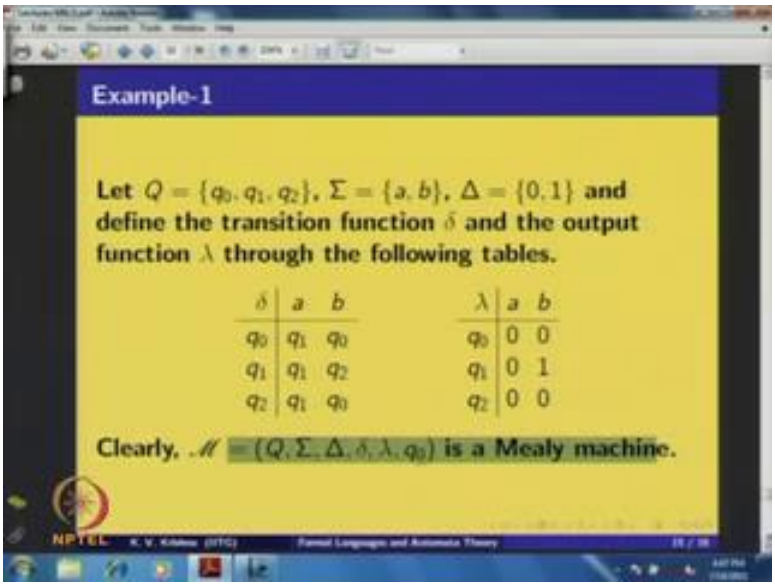
(Refer Slide Time: 38:09)



Now, the output of a Mealy machine is defined to be the string lambda cap Q naught x where q naught is a initial state of the machine, because you have to set the machine in

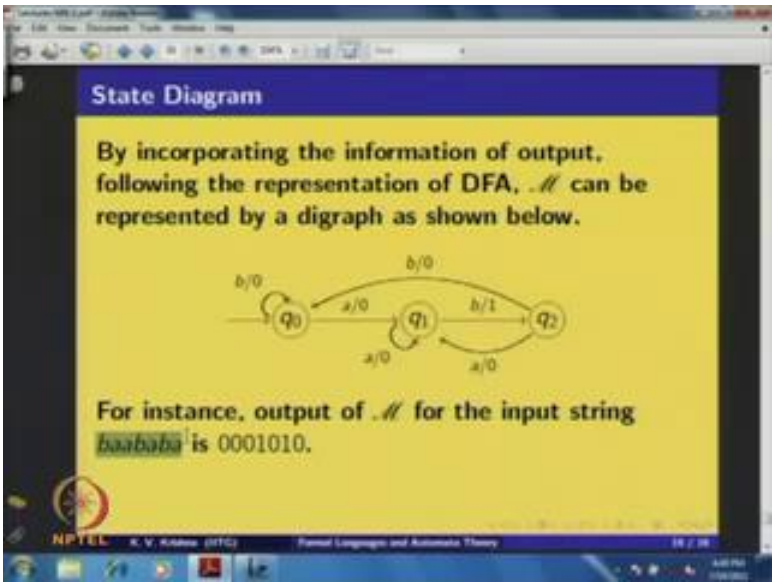
initial state and give the input and whatever is the output for in which that is given because, whatever the string you give in any state and you see what is the output that you are getting that is how we have define lambda cap. But what is the output of the Mealy machine is whatever the string you give it in initial state and whatever is the output I have from initial state, that you get that is what we called as output of machine of course, as I have remarked earlier that the length of the input is same as length of the output.

(Refer Slide Time: 38:54)



Let me give you some example of this variant let us consider q 2 be this q naught q 1, q 2 3 states input alphabet I am taking a b, output alphabet 01 and transition function delta and the output function lambda are given by this tables. Now, I have been by definition you can clearly see I have the states at input alphabet, output alphabet, transition, output function, transition function, output function and I am declaring q naught is as initial state from by definition this is clearly a mealy machine, but what is the behavior of this.

(Refer Slide Time: 39:37)



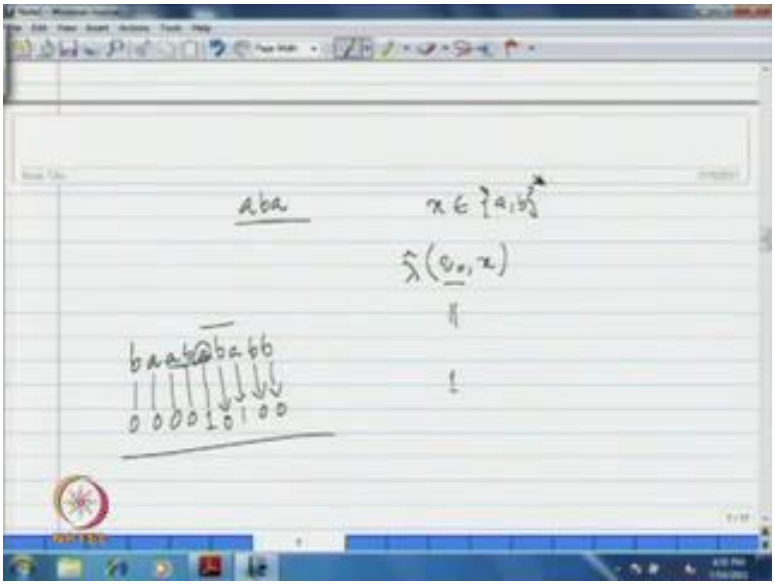
Be before looking at that you can first draw the diagram, I mean through digraph representation like a BFA at the initial state q naught, if I am applying a, whatever is the output I am assigning in this lambda table, you just look at in q naught, if I apply a, I am printing 0 as output in q naught. If I apply a, I am going to q 1 and I am printing 0 as output. So, incorporating the output also on this transitions in this form where the input slash output I can use this digraph, now in this digraph you see how the input is actually pursued to give the appropriate output, you look at here if you are getting some b, it is simply continuing in q naught on the printing 0.

So, as long as you are getting b in the initial state you are printing only the zeros whenever you get a you are changing the state; that means, you are remembering that you have grade a of course, you are printing 0, but whenever from this state of course, in this state. If you are still getting a you are continuing in the same state; that means, you are not worried to remember further zeros and printing 0 only, but in this state q naught if you get b, but then in which case you are printing 1 on that case, we are printing and changing state to q 2.

Now, in q 2 if you are getting b, then we are resetting to q naught, we are coming back to original position, if you are getting a that I am coming back to q 1; that means, essentially you see here whenever I am getting a b whenever I am getting a b end of that part I am printing 1. Otherwise I am printing 0 that is what the mechanism is going on

for instance the output m for this input string 0 say for example, a b b b a b a, you look at that we are printing 0001, because this is corresponding this a b and after that a b again I have printed this 1 and if you have a that is 0. So, that means, if you look at what is called the number of one's of the output, that is exactly counting the number of a b occurring in this a input.

(Refer Slide Time: 42:18)

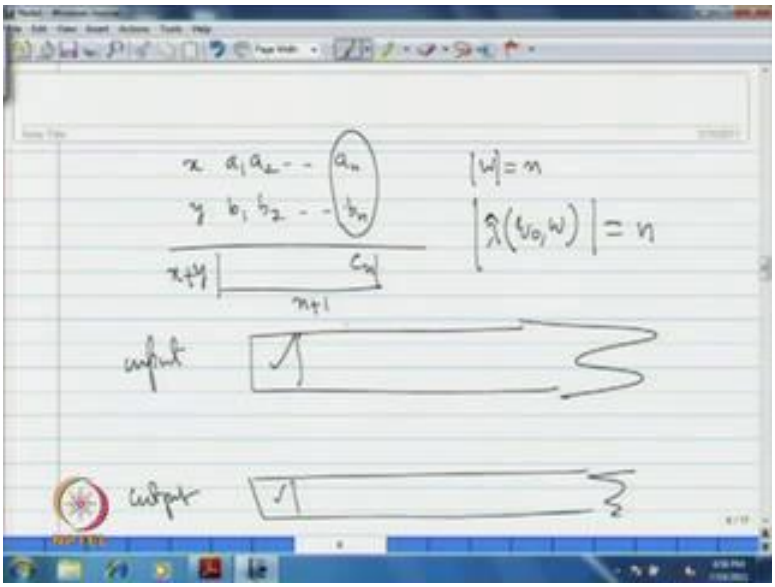


Now, likewise you can think of some other examples, when's to count the number of you know certain substrings for example, let me give you this is an exercise when a similar 1, count the number of a b a. That means, what is my instruction here is if you take a string from a b star, you are asking to print I mean I am assuming q naught to be initial state. This is for x is a 1, a 2, a n, whenever you are having a b a you will be printing 1, otherwise 0, so; that means, end of this. For example, b a a b a b a b b for example, this is the input corresponding to this I am printing 0, this is also 0 and a b a has not come.

So, this is also 0, this for this b you will be printing 0, since I have got this here a b a I will print 1 here, you look again this is an a b a, thus you have clearly that a b this is only a b a with a common you have another a b a. So, that way you see you have to print zero here,, but here 1 as its not resetting here after this rather it has to remember this particular a then corresponding to this b 0 you print and for this. So, this is the output expected.

Now; that means, the exercise is construct the mealy machine that prints 1 for every occurrence a b a in the input otherwise 0, so a similarly example, but you know we have to take care of this common a and appropriately design that.

(Refer Slide Time: 44:24)



Now, let us discuss another example that you find the concept of Mealy machine is sort of appropriate to have as a usual computer if you consider two binary numbers let me call say a 1, a 2 and so on a n, b 1, b 2, b n. Now, what I am expecting is these 2 binary numbers given to you x y, somehow as input how we have to give the other things that we have to discuss now, you will be asking to give the output x plus y the binary addition need to be performed that means.

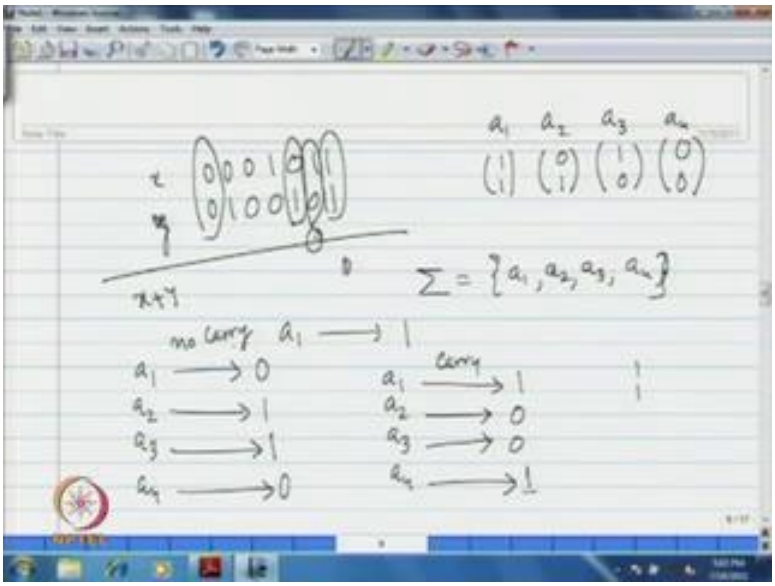
Now, what I will do as you understand in case of Mealy machines, if you given input string say for example, double u of length n, you know the output that lambda cap of Q naught double u, that is also of length n and when I am constructing the Mealy machine on the tape in a cell I can give only one symbol. Now, on corresponding to that symbol on the output tape this is input assume this is output. So, corresponding to this symbol I can print only one symbol here.

Now, what kind of input to be considered and the output how I appropriately we have to get here that we should understand that now when we are adding naturally in case of binary, so you will add these two things let me call it as c n and so on, you will do that and you may have a carry. So, thus the string here possibly of length n plus 1, another

thing is just to balance all those strings, whenever you want to add 2 strings or I mean 2 binary numbers.

First, you ensure that first position this a 1, b 1 to zeros, so that whatever the carry for that you will get in the addition, that you have 1 bit ready to place that and more point is if the length of x; that means, the number of you know the thus as the strings when you are considering the particular binary number. If it is less than y, then you will fill them with leading zeros and you consider the input this always of you can always manage that input both the x and y of same length, but now the question is there are 2 strings here how do you place as input to the Mealy machine.

(Refer Slide Time: 47:17)



Whatever, such as you consider the symbols when because we give them as symbols input here whatever the probabilities because, when you are taking sequence of zeros and ones for example, 1011 and I mean I want to add this with for example, 100101 what my suggestion is as mentioned you first to fill this 2 with zeros and take 1 more leading 0 here this I am calling it as x this is as y say. So, these two are to be added now, whatever I get I know that these many places will be sufficient.

Now, whatever the length here I made up that will be sufficient here now at a given point of time you will be adding you know these two these two like that. So, thus what I will do these two things together you will be accepting it as a input; that means, the input

now I will consider the sequences either it is 11 or maybe you have the probabilities 01 or the probabilities 10 or the probabilities 00, these are the four possible input symbols.

So, now you can take sigma to be let me call it as say for example, a 1, a 2, a 3, a 4 or a b c d whatever you like to call, so these are the input symbols you considered now the situation is naturally what you are doing you are adding from this side. So, you consider the same because, we are because we are constructing a mealy machine by considering this way now whatever the output that you are getting you can of course, sever sit and present to understand that this is what is essentially the sum of that.

So, now what would I suggest you that you are calling them as a 1, a 2, a 3, a 4, or a b c d whatever that and now you see the situation is when you are adding 11 here, you clearly see that I get 1 here, 1 is carry forward. And when I am adding this thing here, these 2 I have to cross check whether I have some carry if there is a carry then I have to include that also to identify the output for the corresponding symbol.

Otherwise, you know if there is no carry; that means, if there is no carry then in which case I did not have I will continue just whatever the current symbols; that means, first of all from this you should understand you should have 2 states. One is indicating that when you are given an input either 10 or 11 or 01 or 00, depending on whether you have carry or not the output is associated to the particular symbol; that means, for example, for a 1, the output will be you know just 1, if there is no carry.

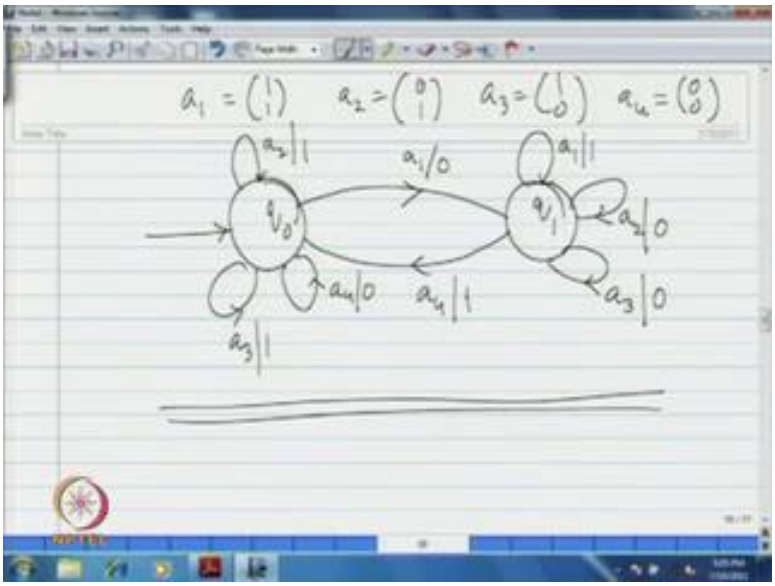
Suppose, already some carry is there for a 1 what will happen, because you have 11, you want to add and you have 1 more carry, so that way what is the total sum, so here this is otherwise it is 0, if there is no carry, you have to because this is 1 plus 1 that means, you will have 1 output. So, this is how we have to distinguish for example, let me consider these symbols which are coming a 1, a 2 a 3, a 4, assume I do not have any carry; that means, as it is when I am adding 1 with 1, I have to print 0.

Of course, there will be one carry when I am adding this 0 with 1; that means, the output has to 1 here there is no carry of course, in this case when I am having a 3 as input, I will be adding 10; that means, here 1 as output, when I am adding 0 with 0. This is 0 here, the assumption is these are the outputs with assumption that no carry while adding that particular, if there is carry will means the carry probabilities is 1. So, that is say when

you have a 1, a 2, a 3, a 4 when you are getting them as input, then I have to consider that into the picture because a 1, is 1,. 1 carry is there here carry whenever carry is there.

So, here now I will have 1 as output, 1 carry will be there for a 2 that is 01 I have 1 carry 01, when you are adding that is 1, but 1 carry is there thus you will get 0 and here you have a carry similarly, when I am adding 1,0, you have 1 carry; that means, this is 0 and you have a carry here when I am adding 00 I have carry 1. So, that way when you add this 00, the output will be one this is how we give and the finally, since as I have we have assumed that the input will always maintain one leading this pair 00, so the carry if carry is existing that will give you the appropriate output.

(Refer Slide Time: 52:23)



Now, following this if we consider q naught as a state which is having no carry and q 1 for the state which is having carry 1, so now of course, in the beginning at the initial state q naught, no carry state now as I have mentioned here a 1, is 11, this is 011000. So, let me once again write that say a 1 is 11, this is the symbol a 2, is a symbol that is 01, a 2 is the symbol 01, a 3 is 10 and a 4 is 00, this is how we have taken now you see the transition for a 1 and when you are adding this with this you are getting carry here, but the output is 0.

So, in a 1 you have a carry; that means, you are going to the state q 1, so a 1 as input and output is 0 and a 2, when you are adding then you do not get any carry because then you do not have any carry; that means, in q naught. So, you will have a loop here a 2 as input

and output is 1 similarly a 3, you will have output 1 carry 0 for a 4,0 as output and no carry similarly as I had listed here when the carry is there; that means, in the state I am in q 1.

So, now you can declare that when I am getting 11 here I have already 1, so; that means, it will become carry 1, because 1 plus 1 plus 1 there; that means, carry 1 and the output is also 1. So, a 1 you will have output 1 and continuing to this and when you are getting a 2 here, you have already carry 1. So, 1 plus 0 plus 1 which is a carry already because of the state q 1.

So, a 2 we will continue here, but here the output is 0 for a 3 also similarly situation you get the carry, but the output is this when you are having a 4 as input, that is 00 and in q 1 you have 1 carry. So, you will get the transition this 0 carry, when there is no carry, so you are transforming to q naught state; that means, a 4 with a output here 00 plus 1, so here you get 1.

So, thus you can construct a Mealy machine to pursue this addition of course, thus we are pursuing from the way usually that you consider from right to left, that is how this is pursuing and you know in case of regular languages all these things that the reversing mechanism, that you can handle using the BFA. So, thus the binary addition including that concept binary addition, you can see that you can represent why a Mealy machine as constructed here.