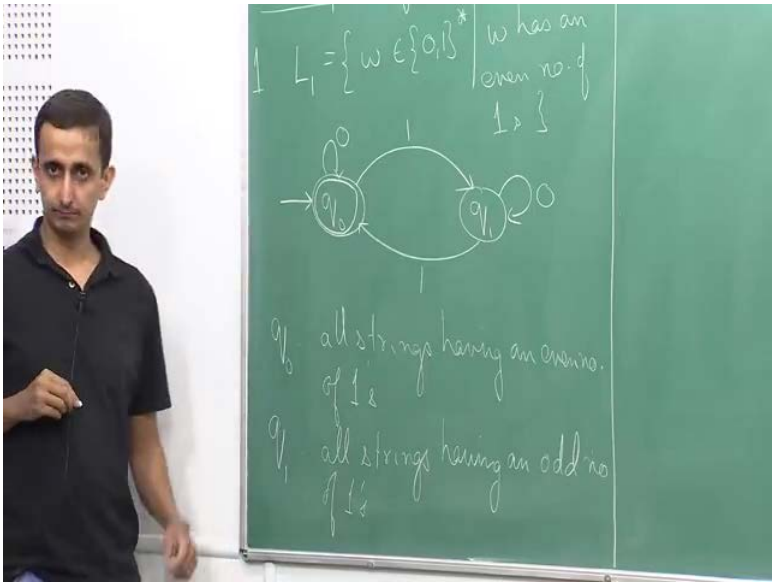
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 03
Example of DFAs

Welcome to the third lecture. Today, we will see some examples of how to construct Deterministic Finite Automata for certain languages.

(Refer Slide Time: 00:35)

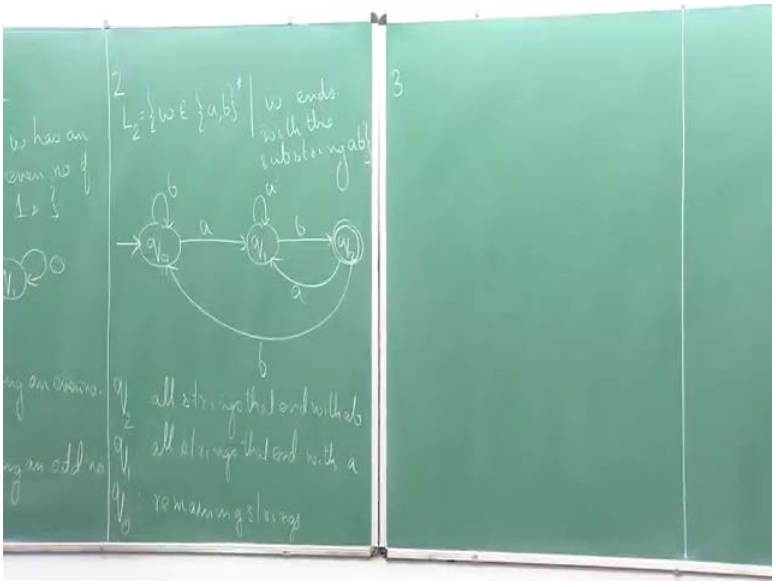


Today we will look at more examples of DFA. In particular, we will try to see what kind of languages that are accepted by these models of computation. The first language that we will see is let us call it L 1 set of all strings over the alphabet 0 1, so this is the set of all strings over 0 1, such that w has an even number of 1s. So all strings that contain an even number of 1s. How do we construct the DFA? Once again, before constructing the DFA, let us try to understand what property the states should capture. If you look at all possible strings over 0 1, either they can have an even number of 1s or they can have an odd number of 1s.

So, we will have two states one capturing all strings that are an even number of 1s and the other state capturing all strings that are an odd number of 1s. We start with a state q 0. So, initially let say q 0 is our start state. So, the empty string by definition does not have any 1s, so which means that it has an even number of 1s. Now if you see single one, we go to a state q 1, which corresponds to see an odd number of 1s. So if you see a single one, we go to q 1. And again from q 1, if you see a 1, it means that we have see an even number of ones we come back to q 0.

And in both these states, if you see a 0, it does not matter, I mean we can see any number of zeros between two successive 1s and it will not affect the parity of the 1s. If you see a 0 from q 0, we stay at q 0; and if you see a 0 from q 1, again we stay at q 1. So, to formalize, so q 0 corresponds to all string having an even number of 1s; and q 1 corresponds to all strings having an odd number of 1s. So, therefore, the accept state in this case is going to be the state q 0. So, instead of even, if we had a language which consisted of all strings having a odd number of 1s, our accept state would have been q 1 instead of q 0.

(Refer Slide Time: 04:29)



Now, let us move on to our next example. Let us look at a language consisting of all strings over so let us in this case change the alphabet, so we will take the alphabet to be

the symbols a coma b by look at all strings over the alphabet a, b such that w ends with the sub string a b. So, how do we construct an automaton for this? Let us try to understand again what kind of information should the states captured, so when we look at the set of strings. When we look at the set of strings that end with an a b, I can look at the set of all strings over a b, and classify them into three sets.

So, one set can be strings that end with an a b; the other set will be set of all strings that do not end with an a b, it just ends with a single a; and the third will be all strings that do not end with an a also. I start at state q 0, if I see single a, I go to a state q 1. If I see a b after that, I go to the state q 2. So, this kind of represents the fact that I have seen sub string a b. Now we must consider strings that end with a b. For example, from here from q 0, if I had seen b, then I will stay at the state q 0 at itself. From q 1, if I see an a, I stay at q 1 itself, because q 1 corresponds to all those string that end with an a. And from q 2, if I see b, it means that I have so the string ends with 2b s; in which case, I come back to q 0. And from q 2, if I see an a, it means that I have a string that does not end with an a b, but it ends only with an a; in which case, I will come back to q 1. And our accept state so firstly, our start state so start state is going to be the state q 0 and our accept state is going to be the state q 2.

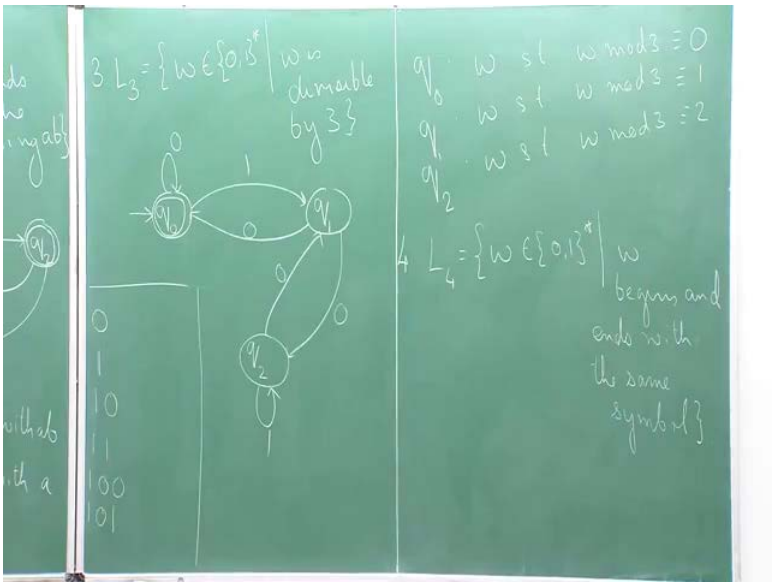
Let me write down the intuition that I mentioned about the states. In this thing, q 2 corresponds to all strings that end with the sub string a b; q 1 corresponds to all strings that end with the sub string a; and q 0 corresponds to the remaining strings basically. All strings that end with b that does not have an a before it; so, it can either be b b or it can be single b or even the empty string epsilon.

Once again in this example what is important is, when we see an a b. The automaton whenever it sees an a b as the last two symbols of its input, no matter which state it is in. If it is in q 0, it will still move to q 2; so if it sees a b it moves to q 2; if it is in q 1, even then it moves to q 2. So from q 1, if it sees an a b, it stays at a and then it moves to b. And the same thing for q 2 as well.

Even from q 2, if it is sees an a b, it moves to q 1, and then back to q 2, so no matter which state the automaton is in. If it sees an a b at the end, it will move to q 2. And if it

does not see an a b at the end for any other pair that can happen, it will not end at q 2 it will either stay at q 0 or it will stay at q 1, so that ensures that the automaton that we have is correctly the captures the language l 2.

(Refer Slide Time: 10:37)



Our third example, so we considered the language L 3 that consists of strings over 0 1, so all binary strings such that w is divisible by 3. If you look at the decimal equivalent of that string, it should be a number that is divisible by 3. Once again here, how do we capture this, I mean how we construct an automata that would accept exactly those strings, exactly those binary strings that are divisible by 3. First of all we have to figure out how many states do we need and what would be the intuition behind those states. So, what would be the strings that those states capture?

We will so since we are look at divisibility by 3 note that every string will have the property that either the string when divided by 3 leaves reminder zero or it leaves a reminder 1 or it leaves a reminder 2, so every string either has one of these three properties. We will have states corresponding to these three type of strings, so we have q 0 which corresponds to all those strings that leave a reminder 0 and divided by 3. We will have q 1 that corresponds to all those strings that leave reminder 1 when divided by 3. And we will have q 2 that leave a reminder 2.

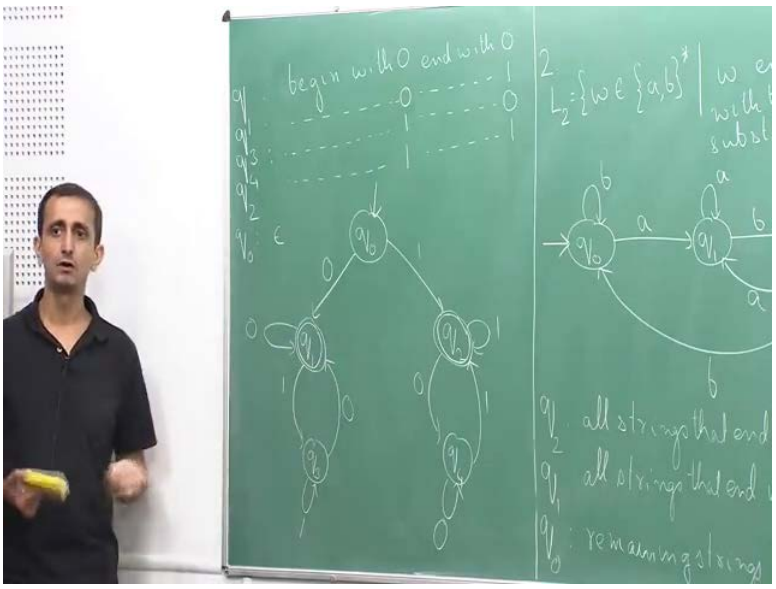
So, we start at the state q 0. And from q 0, if we see single 0, so if we see the string, so let us keep this as our workspace. So what happens when we see single 0, so 0 is the string which when divided by 3 leaves reminder 0. So if I see 0, or if I see any sequence of 0s, it always means that I have I get a reminder 0. Now, what happens if I see a 1. So from q 0, if I see a 1, it means that the string ends with a one in which case it means that when divided by 3, it leaves a remainder 1, on 1, I go to q 1. Now what about q 1, so from q 1, if I see a 0, it means that I have seen the string 1 0. So, 1 0 corresponds to the decimal number 2, so if I divide 2 by 3, it leaves a reminder 2. So from q 1, if I see a 0, I go to q 2. And from q 1, if I see a 1, so after I have a single one, if I see another 1, so this corresponds to the decimal number 3 which means that it is divisible by 3, so if I see a 0, I go back to q 0.

Now, let us look at q 2. So, if I am at q 2, it means that I have seen string of the form 1 0. So after I see 1 0, if I see 0 after this, so this corresponds to the decimal number 4, so which means that from q 2, if I see a 0, I should go to q 1, because 4 when divided by 4 leaves reminder zero. And from 1 0, if I see a 1, it means so this corresponds to the decimal number 5, hence I stay at q 2, because 5 when divided by 3 leaves reminder 2. So, what is our accept states, so accept state in this case will be the state q 0.

Once again q 0 is all strings w such that w mod 3 gives reminder 0; q 1 corresponds to all strings w such that w mod 3 gives reminder 1; and q 2 corresponds to all those strings that leave a reminder 2 when divided by 3. Similarly, you can actually construct automaton that accepts strings that are divisible by 4, 5 or any other number that is in fact, a good exercise to look at into try and construct automata for divisibility by other higher numbers.

Now, let us move on to the last example that I want to discuss today. So, L 4 is the let say the set of all strings that w over 0 1 star such that w begins and ends with the same symbol. I want to look at all those strings whose first and last symbol are the same. So, how do we go about constructing an automaton for the language L 4. Again before I begin, let us try to understand the language L 4. So, let us try to partition the set of strings that the set of all strings over 0 1 star into some sets and try to understand them.

(Refer Slide Time: 18:18)



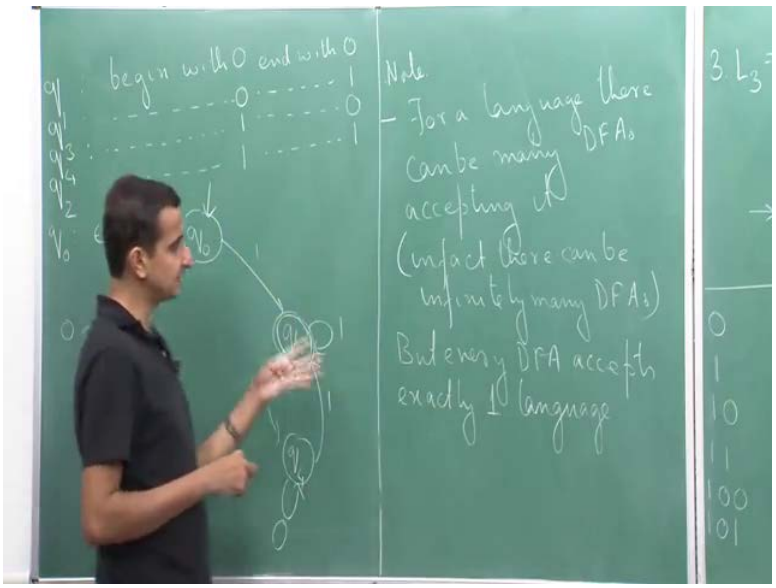
I can write the set of all strings as follows. So, strings that begins with 0 and end with 0. Then I can have set of strings that begin with 0, and it ends with 1. The third is set of all strings that begins with a 1, and ends with a 0. And lastly, I have set of a string that begins and ends with a 1. These are the four possibilities that can happen if I look at the first and the last bit of any string. Now, using this idea, we will construct our automaton. So, we start at some state q 0. And the moment we see the first bit of the string, we go into different direction corresponding to either beginning with a 0 or the string begins with a 1.

So, if the string begins with a 0, I go to state q 1 let say; and if the string begins with a 1, I go to state q 2. And now from q 1, if I see a 1, I go to q 3; and if I see 0, then I stay at q 1. So, q 1 corresponds to all those strings that begin with a 0, and it ends with a 0. From q 3, if I see a 1, then I will stay at q 3 itself; and if I see a 0, then I go back to q 1, because this means that the last symbol that I have seen is a 0. And I do a similar thing at q 2 as well. From q 2, if I see a 1, I stay at q 2; if I see a 0, I go to a state let us call it q 4. At q 4, if I see a 0, I stay at q 4; and if I see a 1, I go to the state q 2.

Now, I can just replace this I mean I can say the following that these four types of strings. These four classes of strings that we have can be corresponds to the following

states. Strings that begin with 0, and end with 0 correspond to the state q 1. Strings that begins with 0, but ends with a 1, so it could correspond to this state q 3. Now strings that begin with 1, and ends with 0 corresponds to the state q 4 here. And a string that begins with 1 and end with 1 corresponds to the state q 2. And lastly for the empty string, the string that we get without reading any symbol, so that is the state q 0. So q 0 in this case corresponds to only the empty string, so this start string. And they accept states are going to be the states q 1 and the states the state q 2. This is an automata where we have two accepts states, since it is the first automata where we have the first example that we have seen, where we have more than one accept state.

(Refer Slide Time: 22:51)

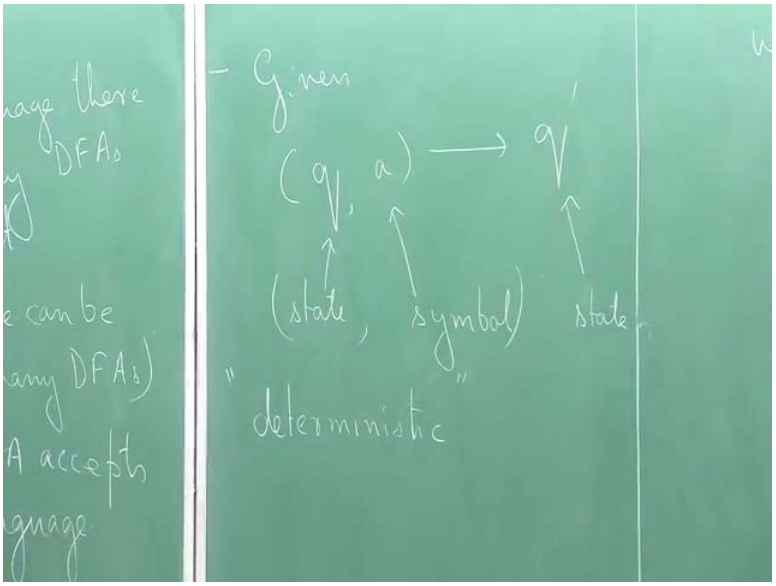


Now let us look at some important observations about deterministic finite automata. The first point that I want to make is that for a language, there can be many DFAs accepting it, in fact, there can be infinitely many language. So for the same language, you can have infinitely many DFAs that accept the same language. But on the other direction, once if you fix a DFA, then each DFA accepts exactly one language, so that is why when we talk of DFA m, it makes sense to talk of the language corresponding to the DFA m. We have seen it earlier that it is denoted by the notation l of m. This is an important point. But if you talk about a language, we cannot talk about just the DFA that accepts that language,

because as I said there can be infinitely many. So, you have to specify which DFA are we talking about.

This is the same as so in the terminology of writing computer programs, this kind of corresponds to the following fact that if you are trying to solve a certain computational problem. For example given an array of numbers let say we want to sort the numbers so then there can be many different programs that you can write to sort those numbers. It is not necessary that there is only one particular algorithm or one computer program that is able to perform the sorting operation. In fact, if there is a class of students, each student can have his or her own computer program to perform the sorting function.

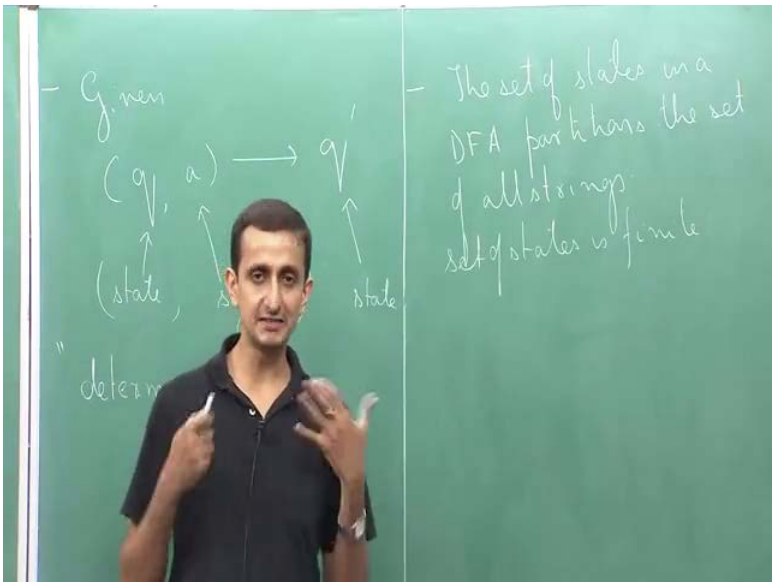
(Refer Slide Time: 26:20)



The second point that I want to make is that so given pair of the form let say q coma a where q is a state and is a symbol. So, from a given q coma a pair by the definition of the transition functions, the way we defined the transition function for our DFA, there is unique state q prime that the automaton goes to. So, from a state coma symbol pair, the automaton goes to always unique state. For example, in the last example that we had earlier, if you look at the state q 1, and we look at the symbol 0, then from q 1 coma 0 the from the pair q 1 coma 0, we go to the state q 1.

Similarly, from the pair q 1 coma 1, we go to the state q 3, and that is actually true for all the states and symbol pair in this example, so that is the point. If you fix state and if you fix symbol then we go to a unique state from it. And this is the reason why we have the word deterministic in the description of this computational model. So, deterministic refers to the fact that given state coma symbol pair; the model deterministically goes to a unique state.

(Refer Slide Time: 28:27)



And the last point that I want to make is that the set of states in a DFA partitions the set of all strings. In other words, what I mean here is that if you pick any string from the set of all possible strings. Suppose, if you have an alphabet sigma, and you take any string from sigma star, and you feed the string to the DFA, then because of this deterministic property of the DFA, there will be unique state that you will end up at. And this happens basically for all strings. So, for every string you always end up at a unique state when starting from the star state. We can think of the set of all strings has been partitioned corresponding to the set of states of the DFA.

So, each state basically corresponds to all those strings that end up at that particular string. And we know that the set of states is finite and this basically means that whenever we are reading any input string we can only remember a finite amount of information.

Because, whenever we are reading a string at any point of the computation, we are at one of the given states, and from that point onward whatever happens for the remaining part of the string is independent of whatever we had seen earlier. Because there is no way in which we can store that information; so the only information that we have with us is what is the current state at which the DFA is right now.

Whatever action that the DFA takes hence forth only depends on that current state, and because there are only a finite number of states. Therefore, the amount of information that can be captured by a DFA is a finite amount of information; it cannot remember information which is a function of let say the input length. This is the reason why we again have the term finite in the terminology of a DFA. Stop here.