

**Theory of Computation**  
**Professor Subrahmanyam Kalyanasundaram**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Hyderabad**

**Lecture 3**

**An Introduction to Finite Automata and Regular Languages – Part 01**

(Refer Slide Time: 00:15)

A language over  $\Sigma$  is a set of strings over  $\Sigma$ . That is, a language over  $\Sigma$  is a subset  $A \subseteq \Sigma^*$ .

Example: ① Set of all binary strings with an odd number of 1's.  
② Set of all binary strings that are palindromes.

Finite Automata → AUTOMATON: Singular  
AUTOMATA: Plural

- Computers with a limited amount of memory.
- State based devices
- Examples like timers, door open/close controller, thermostat

Hello and welcome to lecture 3 of the course, Theory of Computation. So, the past two lectures we saw the brief overview of the course, and then we saw what our alphabet, symbols, languages, etcetera. So, today we are going to start off by examining the first model of computation that we will see during the course, which is called deterministic finite automata.

Before that, just want to remind ourselves about the definition language, the language over Sigma the sigma is an alphabet, it could be binary alphabet like say 0 1 or English alphabet like a,b,c,d up to z, a language over Sigma is a set of strings over Sigma, or in other words it is a set of, it is a subset of all the possible strings constructed over Sigma.

So, a language is a subset over, a subset of sigma star, so language a is a subset of Sigma Star. So, we saw some examples such as set of all the binary strings with an odd number of ones, instead of all the binary strings is an even number of ones, set of all the binary strings that end with 0 instead of all the binary strings they are palindromes, set of all the English words that contain the letter a, so these are all languages over the respective alphabet. Let us just with that reminder let us move on to finite AUTOMATA.

(Refer Slide Time: 01:55)

Finite Automata → **AUTOMATON** : Singular  
**AUTOMATA** : Plural

- Computers with a limited amount of memory.
- State based devices
- Examples like timers, door open/close controller, thermostat
- These abstract models help us gain understanding. States can be thought of as memory.

Deterministic Finite Automata (DFA)

```
graph LR; q1((q1)) -- 0 --> q2(((q2)))
```

NPTL

So, now just one small thing before we get to the definition, so we will be seeing various types of automata during this course, and just to remind everybody that the word AUTOMATA is plural, right, A-U-T-O-M-A-T-A and the singular is AUTOMATON, but while during while speaking about AUTOMATON or AUTOMATA, I may not be able to make the distinction or use the plural or singular accurately, so this is a distinction AUTOMATON is a single device, and AUTOMATA is a plural.

So, finite automata are basically computational devices or computers which have a limited amount of memory, and basically the memory is obtained using the use of States. So, for instance we may have timers on let us say stop watches or even mechanical devices that are timers that measure a certain time, we may have thermostat like if you recall, let us say a room heater or a water heater in our respective houses will have what is called a thermostat, so the goal of the thermostat is to maintain the temperature of the water in the water heaters.

Let us say it has to maintain at 50 degree Celsius, so it will heat the water up to maybe like 52 or 53 degree Celsius, and then turn off the heating, so the water heater will now it is supposed to be an insulated device but it will slowly start losing temperature because no insulation is perfect. And once it maybe drops below 50, the thermostats comes on again and starts heating again. So, basically there are two states one is to heat and one is to stop the heating, so this is also an example of a state-based device.

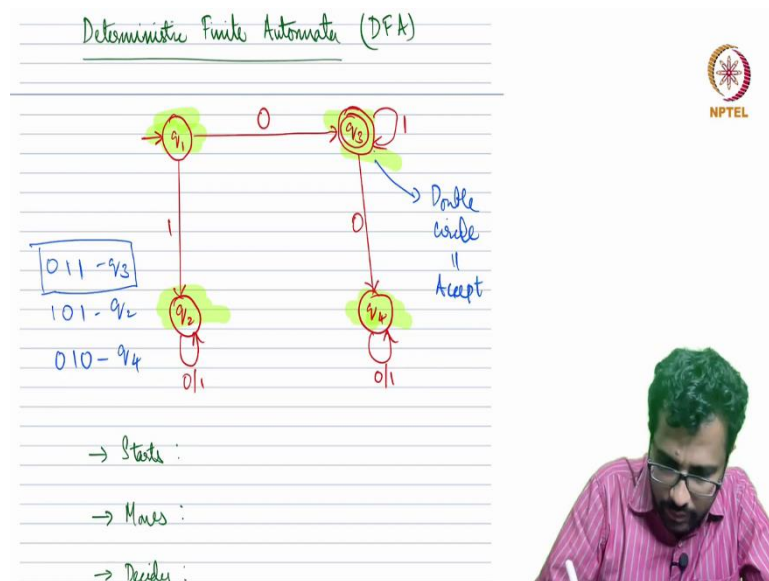
Another one could be an automatic door opening closing controller, so like you will, you would have seen this in a lot of newer buildings where as you walk up to the shop, there will be some glass door or something that automatically opens, and after you have entered and

nobody else is entering it will close, so basically there also there are some states like usually it will be closed and when somebody is near it will sense that somebody is near and then it will open.

And when it senses that nobody else is there it will close, so these are some examples of state-based devices, so these are some simple examples. I am not going into the detail of why they are automated or why they are like automata. And the goal of studying finite automata is like as a precursor to seeing more complex models of computation that we will see later.

So, we will first try to understand simpler models before going into more involved setups. So, this have, so as I said in the finite automata, the states are the main way that the machine keeps track of where it is and what it is supposed to be doing, so in in a sense the states function as the memory, so the memory there is no extra memory apart from what is there in the states.

(Refer Slide Time: 05:23)



So, let us just see an example which will clarify things. So, in finite automata itself, we will first see the deterministic finite automata, so we will see what is deterministic, what is finite about this automation zone. So, this is the deterministic finite automation, so this is an example, so what do we see here? So, there are four circles  $q_1$   $q_2$   $q_3$  and  $q_4$ , these four circles are states, and the starting circle is  $q_1$ , because the starting circle is  $q_1$ , this is the starting Circle because this is a starting State because there is an arrow next to it, so you see there is an arrow next to it so it is a starting state so in in an arrow like this indicates a starting state.

And so, there are if you see for each state there are two outgoing arrows, one for 0 and, one for 1, so q1 outgoing arrow to q2 for 0 and outgoing arrow to q3 for 1 and q2 outgoing arrow to q3 for 0 and outgoing arrow to q4 for 1 and q3 outgoing arrow to q3 for 0, I have missed to actually denote another outgoing arrow, so this should have been this, so there is this, it the outgoing arrow actually comes back into q3. So, the outgoing arrow for 1, so this is the two outgoing arrows for q3 for q2. And q4 basically if you see 0 or 1 the arrows just go back to itself.

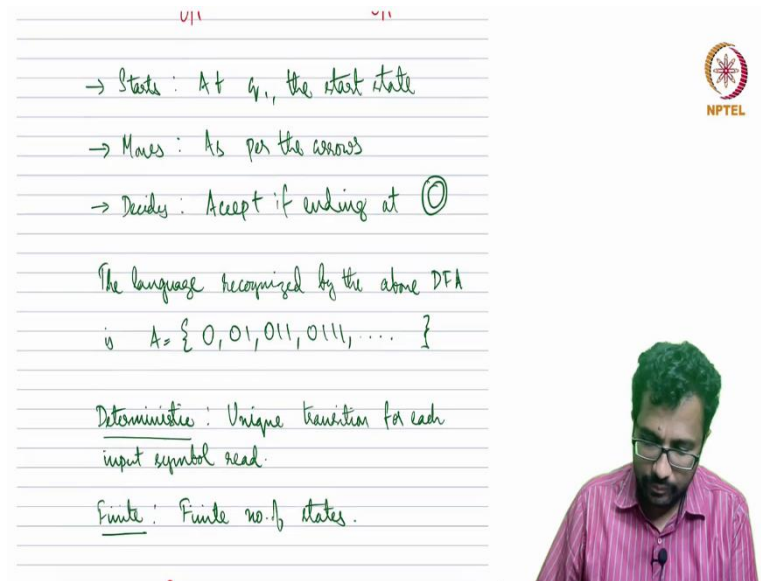
So, let us try to see what it does, maybe it is simplest by let us say what happens when a string such as let us say let us say 0 1 1 is read by this machine, so what happens is the starting position is q1, when it reads 0 it moves to q3, when it reads 0 it moves to q3, and then when it sees 1 it remains at q3, and when it sees the second one it again remains at q3, so after reading 0 1 1 it ends at q3.

So, it goes like this, it first goes to q1 and then into q3 and then to q3 and then to q3, again, so starting at q1 then goes to q3 and then remains there, so this is what happens when it reads 0 1 1, it starts at q1 it is reading the 0 it goes to q3 then reading the 1 it remains in q3 and reading the 1 it remains at q3 again, so at the end of reading 0 1 1 it remains at q3. Now, let us see 101, it starts at q1 when it reads at 1 it moves to q2 and when it reads say 0, it remains at q2 and then it reads at 1, it remains at q2 again.

So, in fact once you reach q2 you are stuck at q2 you are not going to move out of q2, maybe another example, maybe 0 1 0, so you first 0 takes you to q3, 1 make keeps you at q3 and then 0 again takes you to q4. So, I have taken three strings and told and shown where it ends, and the only thing that I have not described is the notion of accepting a certain string.

We say that this machine accepts a certain string if it ends at a certain state which is like this double circle. So, in this case the only state with the double circle is q3, so double circle means it is an accepting state. So, which means that this string gets accepted 0 1 1 because it ends in q3, while these two do not get accepted q2, 1 0 1 and 0 1 0 do not get accepted.

(Refer Slide Time: 10:02)



→ States : At  $q_1$ , the start state

→ Moves : As per the arrows

→ Decides : Accept if ending at  $\odot$

The language recognized by the above DFA  
is  $A = \{ 0, 01, 011, 0111, \dots \}$

Deterministic : Unique transition for each  
input symbol read.

Finite : Finite no. of states.

So, it starts, so just to summarize this, it starts at  $q_1$ , the start state I am going to give the definition formally but moves as per the arrows, so for instance 101, it start with  $q_1$ , then it moves to  $q_2$ , then 0 it again remains at  $q_2$  and 1 it again remains at  $q_2$ , so it moves as per the arrows and decides, so accept if ending at a state which has a double circle.

So, let us try to understand what are the strings that this machine accepts? what are all the strings? So, we know two strings that are not accepted, 101 is not accepted because it ends in  $q_2$  and 1 0 1 0 is not accepted because it ends in  $q_4$  so the only strings are accepted are the ones that are ending in  $q_3$ , so which are the strings that end in  $q_3$ ?

Well, it is not that hard to see because it has to start at  $q_1$  and then move to  $q_3$  there is only one way to move to  $q_3$ , the first symbol should be 0, and then after that for it to remain in  $q_3$  it only has to see once, so the string must be of the form 0 followed by 1 1 1 1, some number of ones, it could be no one's or it could be 1 1 or it could be 2 1s, it could be 3 1s or it could be 100 ones, because if it sees a 0 again it will move to  $q_4$ .

So, the starting symbol must be 0, and then on it should only be once, so the set of strings that are accepted are 0, 0 followed by 1, 0 followed by 2 ones, 0 followed by 3 ones, and so on so there are infinite number of strings possible, so you could have 0 followed by 10 ones, 20 ones, 100 ones, 1000 ones, everything is accepted by this deterministic finite automaton.

So, I have given a brief, I have given an example and kind of explained which are the strings accepted, and we say that this set of strings that are accepted by this DFA is the language

recognized by this DFA. So, I will define everything formally and state all this formally once again, but this is just to give an informal picture.

(Refer Slide Time: 13:02)

input symbol read.

finite: Finite no. of states.

language of the above = All the strings that have at least one 1.

So, why is it called deterministic? It is called deterministic because every state if you are at  $q_1$  and see a, if you are at  $q_1$  and you see a 0, you know exactly where you have to go, you have to go to  $q_3$ , if you see a one at  $q_1$  you do not have to go to  $q_2$ , so you know exactly where to go. So, the movements are fixed if the string is fixed, so that is why it is called deterministic. So, you may think of this as a natural thing to do but later we will see models that do not have this property.

Even if you see a 0 at a certain State there could be multiple options, so that is why it is called deterministic, every time when you see a certain symbol at a certain State you know exactly what to do, that is why it is called deterministic. And finite because this picture here has four states, so in any deterministic finite automation and the number of states should be finite, so that is why it is called finite.

Let us try to understand by seeing a couple of more examples, so this one, let us see, there are only two states of course the symbol is binary, so you can see what happens. The only strings that are accepted are those ending in  $q_2$  and which are the ones that end in  $q_2$ ? So, let us try to see, so if the first symbol is 1 you move to  $q_2$ , and then you remain there, so the first symbol, so all the strings that have the first symbol 1 are accepted, but is that are those only ones, let us say the string 0 1 0, this is get accepted. 0 keeps you at  $q_1$ , 1 takes you to  $q_2$ , and then once you move to  $q_2$  you are accepted for sure.

So, this is accepted, let us see as I said earlier 1 0 0 is accepted because the from q1, you immediately go to q2, so q1 is the starting State because it has a 0 by the left side. So, any string, so that has a one string seems to be accepted, so maybe another way to think about this which are the strings that are not accepted. The strings that are not accepted are those that remain in q1, because once you go to q2 you are stuck at q2 and you are accepted.

So, which are the strings that are not accepted, those that that remain in q1 and which are the how do you remain at q1 if the string is only containing zeros, so if you contain only zeros, then 0 or 0 0 0 0 0 like some number of zeros, because once you see a one at any point you immediately move to q2, and then you are accepted.

So, just to summarize the set of strings accepted or language of the above is equal to all the strings that have at least one 1, because if you see the only way to not get accepted is if you are a string of zeros, so anything that are not a string of zeros means if it has at least one 1 you move to q2 and then you are accepted the first 1 that you see will take you to q2.

(Refer Slide Time: 16:51)

Have at least one 1

010 : X  
101 : ✓

language = Strings that end in 1.

Accepted

language = Strings that end in 0.  
∪ {ε}

NPTEL

Now, let us see another DFA, so the starting set is q1 here and let us see what happens. So, let us try to see what this this thing does. So, whenever you see a one you move to q2 unlike the earlier DFA, once you move to q2 you can come back to q1 here. So, whenever you see a 1 you come to q2, whenever you see a 0 you come to q1, and if you see what is happening is it regardless of whether you are at q1 or q2, regardless of whether where you were if you see a 1 you move to q2.

If you see a 1 from  $q_1$  you move to  $q_2$ , if you see a 1 from  $q_2$  also you move to  $q_2$ , regardless of where you are if you see a 0 you move to  $q_1$ . So, in other words let us say, let us consider two strings 0 1 0, so you are  $q_1$ , 0 keeps you at  $q_1$ , takes you to  $q_2$ , and 0 takes you back to  $q_1$ , it is not accepted.

101 for instance, first symbol is 1 the 1 takes you to  $q_2$ , the 0 takes you to  $q_1$  and the 1 takes you to  $q_2$ , again this is accepted so 0 1 0 is not accepted while as 101 is accepted. So, now you may be seeing what is going on here, so for it to be accepted you have to end in  $q_2$ , and how do you end in  $q_2$ ? If the last symbol is 1. So, the language, all the strings, sorry, that end in 1, all the strings that end in 1, because if you have to be accepted you have to end in  $q_2$  and to end in  $q_2$  you have to end the last symbol scene should be 1.

This is quite similar, in fact the movements the transitions are exactly the same, the only difference is in the accepting state.  $q_1$  and  $q_2$  and the arrows are exactly the same, the labels are exactly the same, the only difference is  $q_1$  is accepting here, and  $q_2$  is not accepting here. So now, if you see anything that ends in  $q_1$  is accepted, and it is not that difficult to see this in, you can work out the reasoning yourself.

The language is the set of all strings that end in 0, because in this picture if you see 1 0 1 0 1, so 1 takes you to  $q_2$ , 0 takes you to  $q_1$ , and 1 takes you back to  $q_2$ , and it is not accepted. Whereas 0 1 0, 0 keeps you at  $q_1$ , 1 takes you to  $q_2$ , and 0 takes you back to  $q_1$ , this is accepted, so anything that ends in 0 is accepted. So, we have seen some examples but not the formal definition, so let us come to the formal definition.

(Refer Slide Time: 20:44)

Definition 1.5: A deterministic finite automaton (DFA) is a 5-tuple  $(Q, \Sigma, \delta, q_0, F)$

where

1.  $Q$  is a finite set of states
2.  $\Sigma$  is a finite alphabet
3.  $\delta: Q \times \Sigma \rightarrow Q$  is the transition function
4.  $q_0 \in Q$  is the start state
5.  $F \subseteq Q$  is the set of accepting states.

Example 1.11



The formal definition of a DFA or a Deterministic Finite Automaton is that it is a five Tuple, so just a small warning before this definition, so when you see 5 tuple with  $Q, \Sigma, \delta$  with all these Greek symbols, so one may get a bit like over hour by this or scared by this, so do not be so scared it is the concept is simple, which is why I decided to do the example before coming to the actual definition. The concept is fairly straightforward, this is just a way to like formalize this notion and make a formal definition. So, it is more important that you understand what is going on than understanding this this bunch of notations.

If you understand the mechanics or what happens when a certain string is input to a certain finite automation that is more important than the definition itself. So, it contains  $(Q, \Sigma, \delta, q_0, F)$ , so  $Q$  is the set of States, so set of states is finite, so this set of states is finite, this is the definition of a deterministic finite automation. So, even in this picture we saw the set of states of all of these are finite, we did not draw any infinite States thing.

$\Sigma$  is the alphabet, in both the examples or in all the examples that we solved, the alphabet was 0 1. So, that so we had arrows label 0 and 1 and the strings that we are dealing with are also 0 and 1, so the alphabet was always binary in in the examples that we saw so that is the alphabet, so that is in this case it is finite. So, Delta is a transition function, so Delta it takes two things one is the  $q$  the current state and the next symbol that you are reading and then what is the next state.



So, if you are in a certain State let us say  $q_1$ , and if you see a 0, this may say you go to  $q_5$ . So, in other words, this Delta is saying  $\delta(q_1, 0) = q_5$ , if you are in  $q_1$  and  $C$  is 0 you move to  $q_5$ , that so that is pictorially depicted above and kind of notationally depicted as the using the delta. And in the set of States one state is designated at the start state and that we call it  $q_0$ , so I think in our examples the start state was  $q_1$ , here also  $q_1$ , here also  $q_1$ , here also  $q_1$ , so we had  $q_1$ , so we basically, we need a designated start saving maybe called  $q_0, q_1$  whatever but it there has to be one designated start state.

(Refer Slide Time: 23:58)

$S, F \subseteq Q$  is the set of accepting states.

Example 1.11

$Q = \{S, q_1, q_2, q_3, q_4\}$



So, there is an example below where the start state is depicted with the letter denoted with the letter S. And F is the set of accept and F is a subset of states, so start set is a single state and while F is set of accepting States. So, but in all the examples that we have seen so far, the set of accepting States also has been a single, so here it is only  $q_3$ , here it is just  $q_2$ , here also it is  $q_2$ , and here it is  $q_1$ , so in all the examples that we have seen, the access set of accepting states have been a single State, but you could have multiple States being accepted. So, maybe let us try to see fit a picture to a definition, so this is another yet another DFA.

(Refer Slide Time: 24:53)



Example 1.11

$Q = \{S, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\delta(S, 0) = q_1$     $\delta(S, 1) = q_3$

10 ✓  
11 ✗  
01 ✓  
10 ✗



So, how many states does it have? It has 5 states, there is a starting State S, then  $q_1$   $q_2$   $q_3$  and  $q_4$  there are five states S  $q_1$   $q_2$   $q_3$  and  $q_4$ , so that is depicted here, and the alphabet if the

arrows are labelled with 0 and 1, so the alphabet is binary, so  $\Sigma$  is 0 and 1. And let us see the  $\delta$  here,  $\delta(S,0) = q_1$ ,  $\delta(S,1) = q_3$ , so that is I noted it here,  $\delta(q_1,0)$ , so if you see 0 at  $q_1$ , you come back to  $q_1$ , so  $\delta(q_1,0) = q_1$ ,  $\delta(q_1,1) = q_2$ ,  $\delta(q_2,0) = q_1$ ,  $\delta(q_2,1) = q_2$ ,  $\delta(q_3,0) = q_4$  and  $\delta(q_3,1) = q_3$ .

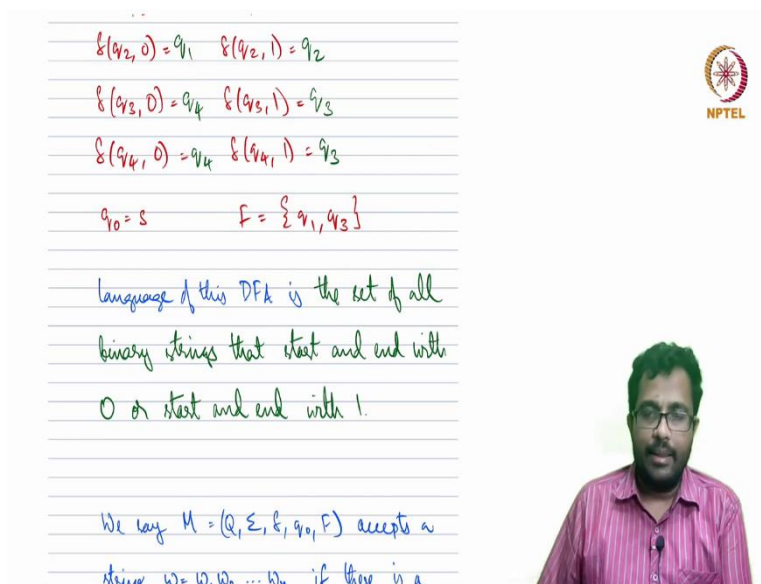
And finally,  $\delta(q_4,0) = q_4$ , and  $\delta(q_4,1) = q_3$ . So, the Delta is also defined so you can verify this from the notes. The starting state is  $S$  and here we have two accepting States  $q_1$  as well as  $q_3$ . So, now let us try to understand what are the set of strings that are accepted by this deterministic finite automaton.

So, one thing that we can see is that if the first symbol is 0 from  $S$ , it comes to  $q_1$ . So, let us try some examples, so it is 0 1 0 is it accepted, first symbol is 0, you move from  $s$  to  $q_1$ , and then you see a one you move to  $q_2$  and then you see a 0 you move to  $q_1$  back and which is an accepting state, so 0 1 0 is accepted.

How about 0 1 1? from  $S$  you move to  $q_1$ , then you move to  $q_2$  upon seeing the second one, so and the third one keeps you at  $q_2$  which is not accepted, so this is not. How about 1 0 1? from  $S$  you move to  $q_3$  and then 0 takes you to  $q_4$  and back to  $q_3$ , so this is accepted. How about 1 1 0? 1 takes you to  $q_3$ , 1 keeps you at  $q_3$ , and 0 takes you to  $q_4$  which is not accepted. So, now you may have understood what is going on. So, what is happening is that the first symbol decides whether you move to the left part or the right part.

If you move to the left part, by the left part what I mean is this part over here, then you remain it, you go to  $q_1$  if the last symbol is in a 0 or  $q_2$  is the last symbol saying is one all, so what is happening is that to get into the left side you need to start with 0, and then to be accepted you need to end with 0, and to go to the side you need to start with 1, and to you need to start with 1 and to be accepted you also need to end with 1.

(Refer Slide Time: 29:18)



The slide contains handwritten notes in red and blue ink on a lined background. The notes define a DFA with states  $q_0, q_1, q_2, q_3, q_4$  and transitions:  $\delta(q_2, 0) = q_1$ ,  $\delta(q_2, 1) = q_2$ ;  $\delta(q_3, 0) = q_4$ ,  $\delta(q_3, 1) = q_3$ ;  $\delta(q_4, 0) = q_4$ ,  $\delta(q_4, 1) = q_3$ . The start state is  $q_0 = S$  and the final state is  $F = \{q_1, q_3\}$ . The language is described as the set of all binary strings that start and end with 0 or start and end with 1. The DFA is defined as  $M = (Q, \Sigma, \delta, q_0, F)$  and it accepts a string  $w = w_1 w_2 \dots w_n$  if there is a

The NPTEL logo is in the top right corner, and a video feed of a man in a pink shirt is in the bottom right corner.

So, the language of this DFA is the set of all binary strings that start and end with 0 or start and end with 1. In other words, starting and ending with the same symbol, so that is the language of this DFA. The set of all strings that start and end in the same symbol. Let me ask you another question this is something, where will the empty string, so we also talked about an empty string which does not have which has 0 length, will it be accepted by this machine, by this automata? So, the empty string is like it does not have any symbols, so the empty string will end at the starting state which is in this case S, which is not an accepting state, so the empty string will not be accepted. And hence, it is not part of the language.

I am just saying this, we can go back and check couple of others DFAs also. Here, empty string is in the language, so because the starting state is where the empty string ends. So, the language here is the strings that end in 0 Union the empty string. Over here, empty string is not accepted because starting symbol is not, the starting state is not an accepting state, same here also. So, these two languages remain the same, whereas here the strings that end in 0 plus the empty string should be the language.