

Theory of Computation
Professor Subramanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Equivalence of NFA and DFA

(Refer Slide Time: 0:16)

Computation Power of NFA's

We saw NFA's, which seem to have more flexibility than DFA's. But are NFA's more powerful? We show that in terms of languages recognized, NFA's and DFA's have the same power.

Defn: Two machines / automata are equivalent if they recognize the same language.

Theorem 1.39: Every NFA has an equivalent DFA.

This implies that NFA's are only as powerful as



Hello and welcome to lecture 8 of the course Theory of Computation. In the past lecture, lecture 7, we saw the formal definition of NFAs. So NFAs seem to have more flexibility and power than DFAs. Because they have the capability of having multiple options from the same state and if they see the same symbol. There is also the ϵ transition available to the NFAs. So, the question is, can NFAs do something more than what DFAs can do? So that is a natural question because they seem to be more powerful. For instance, can they recognize languages the DFAs cannot recognize?

In this lecture we try to address this question. And we show that NFAs and DFAs actually have the same power at least in terms of the class of languages recognized, which means that anything that an NFA can do there is also a DFA that can do the same thing. Meaning if there is a language that is recognized by an NFA then you can also write a DFA for the same language. So, there is no language that an NFA can recognize but a DFA cannot.

(Refer Slide Time: 01:42)

have the same power.

Defn: Two machines / automata are equivalent if they recognize the same language.

Theorem 1.39: Every NFA has an equivalent DFA.

This implies that NFAs are only as powerful as DFAs.

Corollary 1.40: A language is regular if and only if some nondeterministic finite automaton (NFA) recognizes it.

Proof (of Theorem 1.39)



So, towards this end, we need some basic definitions, and then we will proceed to the proof. We first define what we mean by equivalence. So, two machines, so when I say machines, I mean computation devices like NFA, DFA automata. We say that they are equivalent if they recognize the same language. And the main theorem that we will see in this lecture is that every NFA has an equivalent DFA. This means that, even in the last lecture I mentioned this, it is clear that a DFA can be viewed as an NFA. A DFA is already an NFA, where we do not make use of the multiple outgoing, multiple arrows with the same symbol, we do not make use of the ϵ transitions. So, a DFA is automatically an NFA.

But then this statement shows that for anything that an NFA can do, there is an equivalent DFA for it. This means that DFAs and NFAs have the same power at least in terms of what languages they can recognize. It is possible that a language recognized by an NFA, if you write the equivalent DFA, will have way more states than what the NFA has. And maybe the DFA is more complex than the NFA, but that is not the point here. The point is that anything that an NFA can do a DFA can also do. But the resulting DFA may be more involved which is not what we want to focus on, but this is something that I am just stating over here.

And the corollary is that since any NFA has an equivalent DFA, what it means is that the class of languages that is recognized by NFAs is the same as the class of languages recognized by DFAs and we already have a name for the class of languages recognized by the DFAs. The name is regular languages. The thing that I want to say is that a language is regular if and only if there is some NFA that recognizes it. So, this is a new characterization and alternate

characterization for regular languages. So far, we have been saying the language is regular, if there is a DFA for it. Now we are saying that a language is regular if and only if there is some NFA that recognizes it. So, notice both the directions of the statement if and only if.

If an NFA recognizes it, we can make a DFA that recognizes it so it is regular. If a language is regular, there is a DFA that recognizes by the definition that we have already seen. And if there is a DFA recognizing it, the DFA is automatically an NFA. So, if there is a language that is regular there is an NFA for it and if there is a language that has an NFA by virtue of this theorem, theorem 1.39 there is an equivalent DFA and hence it is regular. Notice how both directions are taken care of by this. Now what remains which is not it is a bit long is a proof of the theorem.

(Refer Slide Time: 05:03)

Corollary 1.40: A language is regular if and only if some **undeterministic finite automaton (NFA)** recognizes it.

Proof (of Theorem 1.39)

IDEA: Keep track of the set of all the possible states the NFA could possibly be in.

$\delta: Q \times \Sigma \rightarrow P(Q)$




So, what is the idea? We sort of saw the idea in the previous lecture, so the way we define the transition function of an NFA was something like this. The definition of the transition function was $\delta: Q \times \Sigma_\epsilon \rightarrow P(Q)$. So here itself, the mapping goes to the power set of Q . So this is itself a hint and we also try to keep track of the transitions of let us say of this particular NFA. So we try to see which states it can go to. So maybe let us try to see what happens here when this NFA sees the string, which is 0101.

So it starts with q_1 when it sees 0 it can only remain in q_1 , there is no other option. When it sees 1, there are three possibilities, it can take the self-loop to q_1 or it can go to q_2 or it can go to q_2 and then take the ϵ . And then let us say the next symbol is 0. So, if it was at q_1 , 0 gives you the option to remain at q_1 and that is all. If it was at q_2 there is this option of going to q_3 . If it is at q_3 , there is no next step available, so this path kind of ends here.

Now, let us say the next symbol was 0, let us say the next symbol was 1 again. So q_1 1 there are three options, we saw that in the second step q_1 , q_2 , and q_3 . And q_3 1, there is only one option which is to go to q_4 . This is where all the strings can be or where all the NFA can be after reading the string 0, then the string 01, then the string 010 and then the string 0101.

So, this we tried in the previous lecture as well. So now, the point here is what are we doing here we are trying to see if the string can be accepted. And towards that end, we are trying to see where all it can possibly go. And then we see that the string is indeed accepted because the accepting state is q_4 . After reading 0101, there is one way. There is one way to reach this, the accepting state, after reading this, which is like this. The highlighted path is an accepting path. The first 0 you remain at q_1 , the next 1, you do go to q_2 , the next 0, you go to q_3 , and then the last one you go to q_4 . So, this is an accepting computation path. 0101, is accepted by the NFA.

So, this is the idea that we try to do. Let us see how we will try to do this. What we will try to do we will try to keep track of sets or the set of states where it could be. What we, I will try to draw the same thing in a different way. So, upon starting, it can only be in q_1 and so now I am writing it as collections. So, when you see a 0 from q_1 , you can only be at q_1 , 0 will actually take you to keep you at q_1 . If you see a 1 for instance, however you could go to q_1 , q_2 or q_3 , because 1 can take you to keep at q_1 or you can take you to q_2 or you can take you to you can go to q_2 and then take the ϵ transition.

Now from q_1 , q_2 , q_3 if you have 0. There are three states here so for each of them we have to see. If you are at q_1 , if you see a 0 you can go to q_1 . From q_2 if you see a 0 you can go to q_3 ,

from q_3 if you see a 0 there is nowhere to go. So, there are only two next destinations possible. If you see a 0, which means this arrow leads you to $\{q_1, q_3\}$, the set containing q_1 and q_3 . Suppose you see a 1 from q_1, q_2, q_3 , so q_1 itself, if you see a 1, you can go to q_1, q_2 or q_3 . q_2 , if you see a 1, there is no next step, q_3 if you see a 1, you can go to q_4 . So, if you see a 1 from here, you could be in any of these four states.

So, the point I want to make here is that if you were in any of these three states, and if you see a 0 the next time, you can only be in these two states, q_1, q_2 , sorry, q_1, q_3 . And if you are in q_1, q_2, q_3 , and then you see a 1, the next time you could be in any of these four states, q_1, q_2, q_3, q_4 . Like that, we can keep track of where all we can possibly be after that step. This is how we will convert the non-deterministic computation into a deterministic computation. So, you try to keep track, the set of all states that we can be at after reading some part of the input, that is what we try to do here.




And so, the starting state will be the set containing the starting state of the NFA, q_1 . Now, if you see a 0 here, there is no other state you could go to. If you see a 1, the next state, you could be at any of the three states q_1, q_2 , or q_3 , which is what we have here. And like that, I could write other states as well, so this is not a complete diagram. And then you accept if there is some one of these, at least one of these states is an accepting state. So now you would accept the string 11, because 11 takes you from the starting state. Maybe I will just denote it separately, so this is the starting state q_1 . And 11 takes you to q_1, q_2, q_3, q_4 , out of which q_4 is an accepting state of the NFA. This means there is a way to reach q_4 from q_1 upon seeing 11 in the NFA. So, you would make this as an accepting state.

So maybe I will make it a double. The point is, you want to reach the accepting state of the NFA at some point. This is the rough outline of the proof. Now what remains is to make this more formal.

(Refer Slide Time: 13:39)

Proof: Suppose there are no ϵ transitions. Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA, recognizing some language k . We want to construct a DFA recognizing the same language. Let our target DFA be $M = (Q', \Sigma, \delta', q_0', F)$.

1) $Q' = P(Q) = \{R \mid R \subseteq Q\}$
 \hookrightarrow Power set of Q .








recognizing the same language. Let our target DFA be $M = (Q', \Sigma, \delta', q_0', F)$.

1) $Q' = P(Q) = \{R \mid R \subseteq Q\}$
 \hookrightarrow Power set of Q .

2) $q_0' = \{q_0\}$ q_1, q_2, q_3, q_4

3) $F' = \{R \subseteq Q \mid \exists r \in R \text{ s.t. } r \in F\}$ q_1, q_2, q_3
 $= \{R \subseteq Q \mid R \cap F \neq \emptyset\}$ $R \cap F$

So, what we want to do, let us break this down in two parts. First, we assume that the NFA has no ϵ transitions. So, the NFA only has one of the flexibilities which is that there could be multiple outgoing arrows, for the same state. So, some state is there, which has, let us say, one, two outgoing arrows or three outgoing arrows with the same symbol. Now, this is only flexibility, there is no ϵ transitions. Now this is the NFA, $N = (Q, \Sigma, \delta, q_0, F)$. For that, we want to show that there is an equivalent DFA, so we will construct a DFA, we will call it M . So $M = (Q', \Sigma, \delta', q_0', F')$. The alphabet is the same Σ , for everything else I am just adding prime or dash to the to Q, δ, q_0 and F .

So now let us try to see what each of them is. We have already kind of seen the idea. Let us try to formally write it down. 1) Q' is the set of states of the DFA. At each time we want to keep

track of where all you can possibly be. We have already seen that in the set of states in the DFA, each state is a subset, a set of possible states of the NFA. So, the set of states of the DFA is actually the $P(Q)$, which is the set of all possible sets of states. In other words, it is called the power set of Q . So, each state of the DFA is a subset of the states of the NFA. So, if the NFA has, let us say 3 states, the resulting DFA will have 2^3 states because that is the number of elements the power set contains.

Σ is the same, so there is nothing to say there. So, I will start with the simpler things. So q_0 , the starting state of the DFA, is basically the set containing only the starting state of the NFA. You may be tempted to say that q'_0 is actually equal to q_0 , but this is not correct, because q_0 is not a state of the DFA. The states of the DFA are subsets of the set of all the NFA states. So, it is the singleton set containing only the starting state of the NFA, so that is the starting state of DFA. So q'_0 , the starting set of the DFA is the set containing only q_0 .

Now, the next thing is the set of accepting states of the DFA. So, you want to accept a certain string. You do some computation, let us say out of q_1, q_2, q_3, q_4 , like the example about q_4 is only accepting state. So now, if after reading a string, you can end at q_1 or q_2 , will you accept that string? The answer is no, because neither of them is an accepting state. Suppose instead of q_1, q_2 , you also had q_4 , which means after reading that string you can end at three states, either q_1 or q_2 or q_4 , meaning there are valid computation paths that let you end at q_1 let you end at q_2 and let you end at q_4 .

Now, will you accept this string? Of course, yes, because this means that there is a way to compute/process the string ending at q_4 , which is an accepting state of the NFA. Since there is one state which is an accepting state of the NFA, which is q_4 , you would consider this set to be an accepting state. What I want to say here is that F' is a collection of states of the DFA, but each state of the DFA is a subset of the states of the NFA. So let us consider one state, let us say $R \subseteq Q$ is a state of the DFA or is a subset of states of the NFA. Now, is this an accepting state? So, it will be an accepting state of the DFA.

So let us say R is $\{q_1, q_2, q_4\}$. Is it an accepting state? Yes, because it contains an accepting state of the NFA. So, the way to say it is, $\exists r \in R$ such that $r \in F$, where F is a set of accepting states of the NFA. Another way to say the same thing is R is an accepting state of the DFA, so R is a subset of the states of the NFA. We are saying that R contains an accepting state of the NFA. Another way of saying this is with the intersection of R with the set of accepting states.

So, if you look at R and you look at F, there has to be at least one common state, or there has to be some intersection between them. So $R \cap F \neq \emptyset$. So, this is another way to say it. $R \subseteq Q$ is the same as saying $R \in Q'$, because the members of Q' are the subsets of Q . So this is another way of writing the set of accepting states, F' .

(Refer Slide Time: 20:45)

3) $F' = \{R \subseteq Q \mid \exists r \in R \text{ s.t. } r \in F\}$ (q_1, q_2, q_4)
 $= \{R \subseteq Q \mid R \cap F \neq \emptyset\}$
 $R \subseteq Q'$ $R \cap F$

4) $\delta'(R, a)$ where $R \subseteq Q$ and $a \in \Sigma$.
 $\delta'(R, a) = \{q \in Q \mid \exists r \in R, a \in \Sigma$
 for some $r \in R\}$
 $= \bigcup_{r \in R} \delta(r, a)$

$r \in R \rightarrow \delta(r, a)$

$\delta(\{q_1, q_2\}, 0) = \{q_1, q_2, q_3, q_4\}$
 $\delta(\{q_1, q_2\}, 1) = \{q_1\}$

This takes care of all the transitions except the ϵ transitions. To handle ϵ transitions, we define



The next thing is the it is probably the most involved of them all is a transition function δ' . So δ' , we have to define for a state of the DFA. We are considering R as a state of the DFA, so R is a subset of the set of all states of the NFA and the symbol a in the alphabet. This is what we have to define, $\delta'(R, a)$. Where $R \subseteq Q$, where Q is the set of all states of the NFA and $a \in \Sigma$. So maybe I will just give a brief example. So let us say I have $q_1, q_2, q_3, q_4, 0, 1, 0, 0$, let us say for good measure there is another 0 here, now consider R to be this set containing q_1, q_2 .

Now, in this case, what is $\delta'(\{q_1, q_2\}, 0)$? Upon seeing the input 0, if you are at q_1 , one option is that you remain at q_1 , one option is you go to q_2 , another option is you go to q_3 . So you could be at q_1, q_2 or q_3 . But if you are at q_2 , another option is to go to q_4 . So, $\delta'(\{q_1, q_2\}, 0) = \{q_1, q_2, q_3, q_4\}$. There are four options available, one is to remain at q_1 , another is to go to q_2 , another is to go to q_3 , another is to go to q_4 . What about $\delta'(\{q_1, q_2\}, 1)$? If you are at q_1 , 1 does not take you anywhere, if you are q_2 , 1 only takes you to q_1 . So, it is just a singleton set containing q_1 . $\delta'(\{q_1, q_2\}, 1) = \{q_1\}$. This is how you define transitions.

So let us see how to more formally define it. $\delta'(R, a)$. For each element of R, let us say $r \in R$. Now, you want to see where all can you go from r if you see the symbol a. In the NFA that is

given by $\delta(r, a)$. So $\delta(r, a)$ is the transition function of the NFA tells you where all you can reach from r upon seeing a .

And then you have to do this for each $r \in R$. In other words, I want to do the following, $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$. It is a set of all states of the NFA, which can be reached from r , upon seeing a for some $r \in R$. So, for some member of R , if a takes you somewhere you include that. Another way to say the same thing is $\bigcup_{r \in R} \delta(r, a)$. Consider all $\delta(r, a)$ for some r , then you take the union over all the $r \in R$.

So, this is the definition of δ' . For each element of capital R , you see where all you can go upon seeing a , and you take the union. This completes the definition of the DFA that is equivalent to the NFA that we started with. So, we are not explaining or we are not formally writing down the proof the construction itself is fairly explicit and it is fairly straightforward to see that the construction takes you to an equivalent DFA.

(Refer Slide Time: 26:21)

Proof: Suppose there are no ϵ transitions. Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA, recognizing some language L . We want to construct a DFA recognizing the same language. Let our target DFA be $M = (Q', \Sigma, \delta', q_0', F)$.

1) $Q' = P(Q) = \{R \mid R \subseteq Q\}$



$$= \{R \subseteq Q \mid R \cap F \neq \emptyset\}$$

$$R \subseteq Q'$$

4) $\delta'(R, a)$ where $R \subseteq Q$ and $a \in \Sigma$.

$$\delta'(R, a) = \{q \in Q \mid \exists r \in R, a \in \delta(r, a)\}$$

$$\text{for some } r \in R\}$$

$$= \bigcup_{r \in R} \delta(r, a)$$

$r \in R \rightarrow \delta(r, a)$

This takes care of all the transitions except the ϵ transitions. To handle ϵ transitions, we define $E(R)$ for every $R \subseteq Q$ (or equivalently $R \subseteq Q'$)

$\delta(\{q_1, q_2\}, 0) = \{q_1, q_2, q_3, q_4\}$
 $\delta(\{q_1, q_2\}, 1) = \{q_1\}$



Now, notice that we made this assumption that suppose there were no ϵ transitions. So now, we have to somehow incorporate this into the proof. So let us see how to incorporate this. So, one of the places where this has to be incorporated is the transition function. So here we said that we go to all the states q which can be reached from any $r \in R$, upon seeing a symbol a .

(Refer Slide Time: 27:20)

$E(R)$ for every $R \subseteq Q$ (or equivalently $R \subseteq Q'$)

$E(R) = \{q \mid q \text{ can be reached from } R \text{ by travelling along } 0 \text{ or more } \epsilon \text{ transitions}\}$

Notice that $R \subseteq E(R)$.

We need to redefine δ' and q_0' to account for ϵ transitions.

$\rightarrow \delta'(R, a) = \{q \in Q \mid \exists r \in R, a \in \delta(r, a) \text{ for some } r \in R\}$



But suppose, you reach a certain q . So r and then you see an a and you see you reach a q . Now, suppose this is an ϵ transition from q that is that s , now you can reach s also because from r you see an a then you go to q , then you take the ϵ transition and then you reach s . And suppose there is another ϵ transition from s and then you reach t , then we have to include that as well. So basically, they are all states you can reach from q by just using ϵ transitions. Suppose, there

is another ϵ transition that is some other state even this has to be included. So, consider all the states that you can reach just by using ϵ transitions, all of this has to be included.

(Refer Slide Time: 28:17)

ϵ transitions. To handle ϵ transitions, we define $E(R)$ for every $R \subseteq Q$ (or equivalently $R \in Q$)

$E(R) = \{q \mid q \text{ can be reached from } R \text{ by travelling along } 0 \text{ or more } \epsilon \text{ transitions}\}$

Notice that $R \subseteq E(R)$.

We need to redefine δ' and q_0' to account for ϵ transitions.



So, towards this end, we define what is called $E(R)$. That is a notational definition. If you asked me to name it something, I will call it the ϵ closure of R , because this is the set of all the states that you can reach from R by just using ϵ transitions. You may not use ϵ transition by using 0 or more ϵ transitions. So, I am allowed to use 0 ϵ transitions which means the states that are in R are included by default.

So just to give a pictorial. Suppose, these are the states in R , these three other states in R . Now, this is an ϵ transition, this is a non ϵ transition and suppose this is an ϵ transition, so this is something else, suppose this has no ϵ transitions. Now, suppose this is R , maybe I would use the same color again.

Suppose, this is R , then the set of states that you can reach from R by using ϵ transitions R this included, this included, this included, this included, so these are the states that you can reach. Notice that this state over here, the one that I marked x cannot be reached from the top state in R which I have marked as let us say y using ϵ transitions, but however it can be reached from, let us say, this one, the one that I am working z using ϵ transitions.

Hence, hence x will be in the set $E(R)$. So $E(R)$ is the one that I have drawn with the red outline. So, anything that you can reach from R using ϵ transitions, when I say ϵ transitions, it could be no ϵ transitions also. Because of which the set $E(R)$ contains R by default, because I

have defined it as the set of all states that can be reached from capital R , using 0 or more ϵ transitions. So if I use 0, if I am only allowed 0 ϵ transitions, I can only remain at R , if I want to use one step of ϵ or more steps, then I will include other things that you can reach. In this figure, whatever I can reach ϵ I have marked it. So, it depends on the case.

So now, this is the way to incorporate ϵ transitions into the proof. Suppose there are ϵ transitions. So we have to include ϵ transitions. One way to do it is to include them after each transition. So, you try to see where all you can possibly go upon seeing a symbol, let us say a , now if you see a symbol a where all can you possibly go? Let us say you got a collection of states, which is given by $\delta'(R, a)$. This is a set of states. Now from this set of states, perhaps there are ϵ transitions that take you to some other places now, we want to include that as well.

(Refer Slide Time: 32:16)

$R \in Q$

4) $\delta'(R, a)$ where $R \subseteq Q$ and $a \in \Sigma$.

$$\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R\}$$

$$= \bigcup_{r \in R} \delta(r, a)$$

$r \in R \rightarrow \delta(r, a)$

This takes care of all the transitions except the ϵ transitions. To handle ϵ transitions, we define $E(R)$ for every $R \subseteq Q$ (or equivalently $R \in Q'$)

$$E(R) = \{q \mid q \text{ can be reached from } R \text{ by } \epsilon\}$$

$\delta(\{q_1, q_2\}, 0) = \{q_1, q_2, q_3, q_4\}$
 $\delta(\{q_1, q_2\}, 1) = \{q_1\}$



We need to redefine δ' and q_0 to account for ϵ transitions.

$$\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$$

$$= \bigcup_{r \in R} E(\delta(r, a))$$

$$\rightarrow q_0 = E(\{q_0\})$$

After reading any string, if N can possibly reach the states $R \subseteq Q$, then M will reach $R \in P(Q)$.



Notice that $R \subseteq E(R)$.

We need to redefine δ' and q_0 to account for ϵ transitions.

$\rightarrow \delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R\}$

$= \bigcup_{r \in R} E(\delta(r, a))$

$\rightarrow q_0 = E(\{q_0\})$.

After making these changes, if necessary.



So how do we include that? One way to include is by, so $\delta'(R, a)$. So for each $r \in R$, earlier we took all the states that are in $\delta(r, a)$. which is the set of states reachable from $r \in R$ using the transition a . Now instead of $\delta(r, a)$, now we are saying maybe I will use a different color here. I am saying $E(\delta(r, a))$, similarly in the union definition, earlier, I just had $\bigcup_{r \in R} \delta(r, a)$. Now, from whichever states that you can reach from r using the transition a , we want to add the other states that you can reach after the transition by just using ϵ .

So if this is a set of states, let us say $\delta(r, a)$, now let us see where all you can reach by just using ϵ 's. So we want to include this as well. So now, we draw a bigger circle, which is basically the circle is $E(\delta(r, a))$ the set of states that you can reach from $\delta(r, a)$ by using ϵ transitions. $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$. This takes care of all the transitions. After each transition, we are seeing where all you can possibly be by using ϵ transitions. So, this is the transition function we have to modify, so instead of the four defined as the transition function being defined as this, this is the actual definition.

And the other thing that we have to change is the starting state, because in the transition function, we are accounting for the ϵ transitions after we make the transition. So what if the starting state had ϵ transitions? If q_0 had let us say, one ϵ transition, say q_1 , how to include that? One way to include that is to instead of defining the starting state as just the set containing q_0 , we can define the starting state as the ϵ closure of the set containing q_0 . So again, this is a new thing, it is $E(\{q_0\})$. So these are the two changes that we need to make for taking the ϵ transitions into account.

So Q' is a set of all subsets of Q or $P(Q)$. The starting state q'_0 is $E(\{q_0\})$. The set of accepting states of the DFA, F' is any subset of Q , which is the set of all states of the NFA, which has an intersection with the set of accepting states of the NFA. The set of all R which have a non-empty intersection with F , so $R \cap F \neq \emptyset$. And finally, the transition function is given by this, $\delta'(R, a) = \{ q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R \}$. For each transition function from a set R upon seeing a symbol a . So, you look at each $r \in R$ where all can you go upon seeing a , so that is given by $\delta(r, a)$, it is a set of states.

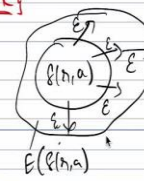
Now, you take the ϵ closure of it, meaning where all can you go after this by just using ϵ transitions. Now, you take the union over all the possible $r \in R$, that is the transition function. So, this completes the construction of the DFA that is equivalent to the NFA. The correctness is fairly straightforward.

(Refer Slide Time: 37:18)

some $r \in R$?

$$= \bigcup_{r \in R} E(\delta(r, a))$$

$\rightarrow q_0 = E(\{q_0\})$



After reading any string, if N can possibly reach the states $R \subseteq Q$, then M will reach $R \in P(Q)$.



Read example 1.41 in the book, for an illustration of the construction in the above proof.



Computation Power of NFA's



We saw NFA's, which seem to have more flexibility than DFA's. But are NFA's more powerful? We show that in terms of languages recognized, NFA's and DFA's have the same power.

Defn: Two machines / automata are equivalent if they recognize the same language.

Theorem 1.39: Every NFA has an equivalent DFA.

This implies that NFA's are only as powerful as



Theorem 1.39: Every NFA has an equivalent DFA.



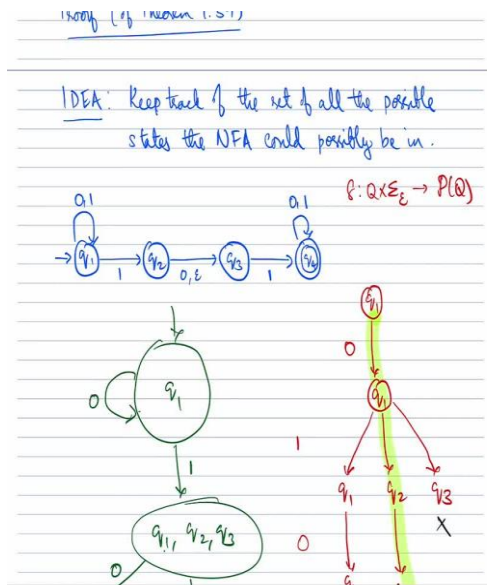
This implies that NFA's are only as powerful as DFA's.

Corollary 1.40: A language is regular if and only if some non-deterministic finite automaton (NFA) recognizes it.

Proof (of Theorem 1.39)

IDEA: Keep track of the set of all the possible states the NFA could possibly be in.





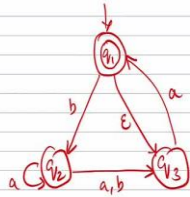
If you read any string and the DFA reaches the states r_1, r_3, r_{10} , so the DFA reaches a state r_{10} . This means that the NFA after reading the same string can reach exactly these three possible states it could reach r_1 in the NFA or r_3 in NFA or r_{10} in the NFA, these three states are the exact states possible and this is an accepting state if either three of them or if any one of them or some subset of them is an accepting state of the NFA. It is fairly clear by the construction that if the non-deterministic finite automata, the NFA can reach three states r_1, r_3, r_{10} , then the DFA by construction will reach the state containing exactly these three states.

So that is the main theorem that I want to state in this lecture. The point is that the computation power of NFAs is equivalent to DFAs, meaning in terms of the languages that can be recognized by the NFA it is the same as what can be recognized by the DFA. This is proved by showing that every NFA has an equivalent DFA and that is the construction that we saw. And the main idea is to build a DFA where each state of the DFA is a subset of the set of all states of the NFA. One point is that this NFA has four states, the NFA in blue over here has four states.

So, the equivalent DFA as per this construction will contain how many states? It will contain as many states as the power set of this contains. The power set of a set of four elements contains 2^4 which is 16 states, 16 elements. The equivalent DFA will contain around 16 states, one for each subset of this. This set of states could be the empty set, it could be the singleton sets, the set containing two elements, it can be three elements and the set containing all the four states. So that could be 2^4 , which is 16 states and it can get a bit involved.

(Refer Slide Time: 39:46)

Read example 1.41 in the book, for an illustration of the construction in the above proof.



Eqvt DFA contains $2^3 = 8$ states \rightarrow 6 states

they recognize the same language.

Theorem 1.39: Every NFA has an equivalent DFA.

This implies that NFAs are only as powerful as DFAs.

Corollary 1.40: A language is regular if and only if some non-deterministic finite automaton (NFA) recognizes it.

Proof (of Theorem 1.39)

IDEA: Keep track of the set of all the possible states the NFA could possibly be in.

Maybe one thing that you can do is an example in the book, which is example 1.41. So, the DFA that I have written below contains three states q_1, q_2, q_3 . q_1 is the starting state as well as the lone accepting state and it is over the binary alphabet. If you see the alphabet is just a and b and there is an empty transition ϵ as well. For this state in the Sipser book they construct an equivalent DFA, and the equivalent DFA as you would expect contains 2^3 , which is equal to 8 states, but then the example of the book notices that some states are not really useful, so they finally reduce.

So, the equivalent DFA contains 2^3 equal to 8 states this is what we can say without any actual construction, but when you construct actually there are some unreachable states, some states which are useless. So, as you can see in the example, they actually cut down two states because

it is not serving any purpose in the DFA, so it ends up being 6 states. So please go through the example in the book.

And so that is all I wanted to say in this lecture, where we saw the main theorem which is that every NFA has an equivalent DFA. Also, the implication is that this gives a new characterization for regular languages which is that the regular languages are the class of languages that have some NFA that recognizes it. So, the earlier definition was that a language is regular if there is some DFA that recognizes it. Now I have a new definition. A language is regular if there is some NFA that recognizes it. That is all I have for lecture 8. See you in lecture 9. Thank you.