

Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Closure Properties of Regular Languages under Union, Concatenation and Kleene Star
Operation - Part 01

(Refer Slide Time: 00:15)

Theorem 1.25: The class of regular languages
is closed under the union operation.



In other words, if A_1 and A_2 are regular,
it means that $A_1 \cup A_2$ is regular.

Suppose A_1 and A_2 are regular, this means
that there exist DFA's M_1, M_2 such that
 $L(M_1) = A_1$ and $L(M_2) = A_2$. Can we build
a DFA M such that $L(M) = A_1 \cup A_2$?

Idea 1: Combine M_1 and M_2 such that first



Hello and welcome to lecture 6 of the course Theory of Computation. In the previous lecture, we saw the 3 regular operations which were union, concatenation and star. In this lecture we will try to first see closure properties of the regular languages, the class of regular languages is closed under the union operation.

And then we will move on to concatenation. And then we will see why that does not work which leads us to the next topic. So the thing that we want to show is that the class of regular languages is closed under the union operation. So, what does this mean, what is closed under the union operation mean?

So, what this means is that if you take the set of all regular languages, you take 2 regular languages and take the union of them, you still get a regular language. So, you can think of all the regular languages as a class. Let us say A is here and B is here. And you take A union B. These are all regular languages and so A union B is also somewhere here.

So, if you take the union of any 2 regular languages, you still get a regular language. In other words, what it means is that if A_1 and A_2 are regular then the union is also regular. The union $A_1 \cup A_2$ is also regular. So, what does it mean?

(Refer Slide Time: 01:58)

Suppose A_1 and A_2 are regular, this means that there exist DFA's M_1, M_2 such that $L(M_1) = A_1$ and $L(M_2) = A_2$. Can we build a DFA M such that $L(M) = A_1 \cup A_2$?

Idea 1: Combine M_1 and M_2 such that first M_1 runs on the input, followed by M_2 .

Does not work. We cannot go back and read the string again.

Moral: We need to keep track of the



So, suppose, A_1 and A_2 are regular. Because we do not know anything about A_1 , we do not know anything about A_2 except for the fact that they are both regular. So, to show the statement, we have to actually take any arbitrary regular language A_1 and arbitrary regular language A_2 and then show that the union is also regular.

The only thing that we know is that it is regular. So, by definition of regular language, there is a DFA M_1 that exists such that the language recognized by M_1 is A_1 . And there is a DFA M_2 , such that the language recognized by M_2 is A_2 . So, this is just pretty much the definition of what is a regular language.

So, now the question is to show that the union is regular, we have to build another DFA M such that the language recognized by that DFA is the union. So, you have one DFA that recognizes A_1 , there is another DFA that recognizes A_2 . So, the first one is called M_1 the second is called M_2 .

Now, we want to somehow use these two DFA's, maybe combine them in some way such that the resulting DFA M recognizes the union language. So, it should recognize exactly those things that are in the union and nothing more. How will we build such a DFA?

So, maybe the simplest thing that one can think of is let us say you get a string w . So, try to run M_1 on it and then M_2 on it. If M_1 accepts w then you accept. If M_1 does not accept then you check M_2 . If M_2 accepts then this does not work.

We cannot go back and run the DFA on the same input again. So, you know how a DFA reads symbol by symbol of the input. So, if the input is 01101, it reads 01001 and at the end we expect it to have made a decision whether to accept it or not. There is no provision to go somewhere and then you go back and read it again, that provision is not there.

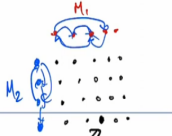
So, we cannot go back and read the string again. This does not work. It's a simple idea and probably is the first natural idea that comes to mind but it doesn't work. So, if we cannot read it again, which means we only have one chance to read it. So, again the natural thing that we may think is that in that one reading itself we want to see what M_1 and M_2 does.

So, basically, we want to see the workings of M_1 as well as M_2 at the same time. This is probably what we need to do. We do not get a second chance to go back and try running M_2 on it. So, how do we ensure that both M_1 and M_2 runs on the string? We want to simulate that together. How do we ensure that? So, that is what we want to accomplish.

(Refer Slide Time: 06:31)

does not work. we cannot go back to previous read the string again.




Moral: We need to keep track of the workings of M_1 and M_2 at the same time. How can we accomplish this?



Idea: Cartesian Product. DFA whose states are of the form (r, s) .

$$\delta((r_1, r_2), a) = (s_1, s_2) \leftarrow \delta_1(r_1, a) = s_1$$

↑ 0



So, the idea is, so, this is something I want to say about this course in general, most of the ideas are fairly straightforward. Only some ideas are there that are a bit more advanced and a bit more intricate but most of the ideas are simple and maybe the difficulty is only in formalizing those ideas.

So, once you understand what the notation is, what these complex looking symbols represent then it is fairly just translating your idea into those symbols. It is not really, again I said it in the previous lecture, I will say it again, do not be scared of these symbols, it is just another way to represent what we know. So, the idea is this.

So, suppose let us say, these dots are the states of M_1 , these red dots. Let us say, these blue dots are the state of M_2 . And the idea is we will make kind of a 4 by 5 grid in such a way that if you end up in the third red dot and the fourth blue dot you will end up in the dot that is the black dot that is in the third column and the fourth row.

And if we can manage to construct a DFA that simulates this kind of a grid situation and ends like this. This intuitively it seems to capture both the row information and the column information. So, this is the basic idea. The rest is just notation in order to realize this. So, let us see what we do.



(Refer Slide Time: 09:26)

Idea: Cartesian Product. DFA whose states are of the form (r, s) .

$$\delta((r_1, r_2), a) = (s_1, s_2) \leftarrow \delta_1(r_1, a) = s_1$$

$\delta_2(r_2, a) = s_2$

The diagram illustrates the Cartesian product of two DFAs. On the left, a DFA with two states and transitions labeled 0 and 1. In the middle, a 2x2 grid of states with transitions labeled 0 and 1. On the right, a small DFA with two states and transitions labeled 0 and 1.





So, the idea is this. So, now we need to make a DFA M . So, we have a DFA M_1 corresponding to the language A_1 and DFA M_2 that corresponds to the language A_2 . So, now, we will build a DFA whose states are of this form, the Cartesian product. So, they are of the form of (r, s) where r is a state of the first DFA and s is a state of the second DFA. So, maybe you will come back to this figure in a bit.

(Refer Slide Time: 09:59)

Proof: We have A_1, A_2 regular languages.
let M_1, M_2 be DFA's such that $L(M_1) = A_1$
and $L(M_2) = A_2$. Assume $A_1, A_2 \subseteq \Sigma^*$.

$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$
$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$



Goal: Need to build $M = (Q, \Sigma, \delta, q_0, F)$
such that $L(M) = M_1 \cup M_2$.



let M_1, M_2 be DFA's such that $L(M_1) = A_1$
and $L(M_2) = A_2$. Assume $A_1, A_2 \subseteq \Sigma^*$.

$$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$$
$$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$$

Goal: Need to build $M = (Q, \Sigma, \delta, q_0, F)$
such that $L(M) = A_1 \cup A_2 = L(M_1) \cup L(M_2)$

$$(1) Q = \{ (q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2 \}$$
$$= Q_1 \times Q_2$$


I have already said all this, so we have A_1 and A_2 regular languages and M_1 and M_2 are DFA's that recognize A_1 and A_2 . And for the sake of simplicity, we may assume that A_1 and A_2 belong to the same or are over the same alphabet Σ . So, this is not a very unreasonable assumption because if you want to take the union etcetera, if they are in different alphabet, it does not really do much.

So, we want it to make more sense if they are both the same like A_1 is over binary A_2 is over English, it does not really make much sense to do this. So, let M_1 be the following DFA $(Q_1, \Sigma, \delta_1, q_1, F_1)$. So, q_1 is the starting state, as I already said, we assume that both A_1 and A_2 are over the same alphabet Σ .

δ_1 is a transition function of M_1 , small q_1 is a starting state of M_1 and F_1 is the set of accepting states of M_1 . Similarly, we have $(Q_2, \Sigma, \delta_2, q_2, F_2)$ where the one thing that is the same here is the alphabet, everything is different, everything else is potentially different for M_1 and M_2 .

Our goal is to build a DFA M using which has say over again over the same alphabet but starting with a set of states Q , δ and starting state q_0 and accepting states F such that the set of strings accepted is the union of A_1 and A_2 . Or in other words, it is just

$$L(M_1) \cup L(M_2)$$

Because M_1 and M_2 are machines or DFA's, you cannot take the union of DFA's.



(Refer Slide Time: 12:32)

such that $L(M) = A_1 \cup A_2 = L(M_1) \cup L(M_2)$

(1) $Q = \{ (r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2 \}$
 $= Q_1 \times Q_2$

(2) Σ is the same alphabet of A_1 and A_2 .
 If A_1 and A_2 are not over the same alphabet, let $\Sigma = \Sigma_1 \cup \Sigma_2$.

(3) δ needs to keep track of δ_1 and δ_2 .
 $\delta: (Q \times \Sigma) \rightarrow Q$

So, as I already said, the set of states of this DFA M will be the Cartesian product. So, every state of the new DFA constructed will be of this form (r_1, r_2) where r_1 comes from the state of the first DFA M_1 and r_2 comes from the state of the second DFA M_2 . In other words, it is just

$$Q = Q_1 \times Q_2$$

And the alphabet is the same as the alphabet of M . So, in just some cases A_1 and A_2 are over different alphabets, then we can let the alphabet of M be the union of these 2 alphabets. So, again this is and you can view A_1 and A_2 as the languages over the union. So, this is possible.



(Refer Slide Time: 13:35)

$= Q_1 \times Q_2$

(2) Σ is the same alphabet of M_1 and M_2 .
If M_1 and M_2 are not over the same alphabet, let $\Sigma = \Sigma_1 \cup \Sigma_2$.

(3) δ needs to keep track of q_1 and q_2 .
 $\delta: (Q \times \Sigma) \rightarrow Q$

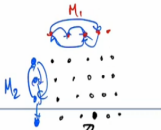
(4) Starting state: $q_0 = (q_1, q_2)$

And the main key thing is how are we keeping track of the transitions of M_1 and M_2 .

(Refer Slide Time: 13:52)

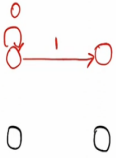


Moral: We need to keep track of the workings of M_1 and M_2 at the same time.
How can we accomplish this?



Idea: Cartesian Product. DFA whose states are of the form (q_1, s) .

$\delta((q_1, q_2), a) = (q_1, s_2) \leftarrow \delta_1(q_1, a) = s_1$

\uparrow
 $\delta_2(q_2, a) = s_2$
 Q

use of the form (q, s) .

$\delta((q_1, q_2), a) = (q_1, q_2) \leftarrow \delta_1(q_1, a) = q_1$

$\delta_2(q_2, a) = q_2$

$b_1, b_2, b_3 \in Q_2$

$a_1, a_2 \in Q_1$

Proof: We have A_1, A_2 regular languages.

So, in our example over here, we try to explain that we want to keep track of what M_1 does through the columns and what M_2 does through the rows. So, if a certain string ends up at the middle in the third state for M_1 and the fourth dot from the top for M_2 , we want it to end in the third column and the fourth row of the Cartesian product.

So, basically, whatever happens in the row, it must be captured by the transitions that happen across the rows, and whatever happens to the columns must be captured by the transitions that happen across the columns.

Suppose, this is just a snapshot of the Cartesian product. Suppose, there are 3 states, call them b_1, b_2 and b_3 . These are the 3 rows of the machine. So, b_1, b_2 and b_3 belong to Q_2 . And let us say, these red ones, a_1 and a_2 belong to Q_1 . What do we want these to represent?

For instance, the top left black dot actually represents (b_1, a_1) , this one represents (b_1, a_2) and so on, $(b_2, a_1), (b_2, a_2), (b_3, a_1)$, and (b_3, a_2) . So, suppose we start from the top left, (b_1, a_1) state and we see a 0. If you see in the rows, if you see a 0, you go to the one row below.

And in the columns if you see a 0, you remain in the same point a_1 . So, b_1 goes to b_2 and a_1 does not change. So, if you see a 0 you should just go down to b_2 but remain at a_1 . Suppose, you see a 1 at (b_1, a_1) you skip down to 2 levels, you go to b_3 .

And if you see a 1 at a_1 you move to a_2 . So, you should go to (b_3, a_2) . This is if you see a 1. So, this is what happens when you try to see a 1 from (b_1, a_1) and see a 0 from (b_1, a_1) . If you

look at it both these transitions are faithful to their respective rows as well as their respective columns.

So, what we are trying to do is make the transition such a way that we are capturing both the row wise transition of M_2 and the column wise transitions of M_1 . In other words, if the M_1 transition is given by $\delta_1(r_1, z) = s_1$ and M_2 transition is given by $\delta_2(r_2, z) = s_2$.

(Refer Slide Time: 18:01)

use of the form (r, s) .

$\delta((r_1, r_2), z) = (s_1, s_2) \leftarrow \delta_1(r_1, z) = s_1$

$\delta_2(r_2, z) = s_2$

$b_1, b_2, b_3 \in Q_2$

$a_1, a_2 \in Q_1$

Proof: We have A_1, A_2 regular languages.

$= Q_1 \times Q_2$




(2) Σ is the same alphabet of A_1 and A_2 .
If A_1 and A_2 are not over the same alphabet, let $\Sigma = \Sigma_1 \cup \Sigma_2$.

(3) δ needs to keep track of δ_1 and δ_2 .

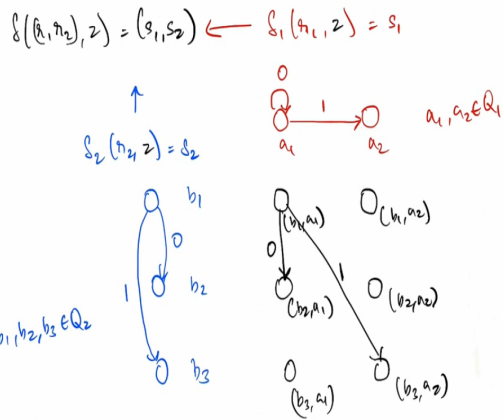
$\delta: (Q \times \Sigma) \rightarrow Q$

$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

(4) Starting state: $q_0 = (q_1, q_2)$

Idea: Cartesian Product. DFA whose states are of the form (q, s) .



such that $L(M) = A_1 \cup A_2 = L(M_1) \cup L(M_2)$



(1) $Q = \{(q_1, q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$
 $= Q_1 \times Q_2$

(2) Σ is the same alphabet of A_1 and A_2 .
 If A_1 and A_2 are not over the same alphabet, let $\Sigma = \Sigma_1 \cup \Sigma_2$.

(3) δ needs to keep track of q_1 and q_2 .

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

$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$



if A_1 and A_2 are not over the same alphabet, let $\Sigma = \Sigma_1 \cup \Sigma_2$.



(3) δ needs to keep track of q_1 and q_2 .

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

$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$

(4) Starting state: $q_0 = (q_1, q_2)$

(5) Accepting states: We accept w if it ends at (q_1, q_2) if either $q_1 \in F_1$ or $q_2 \in F_2$.



Then from the Cartesian product we get $\delta((r_1, r_2), z) = (s_1, s_2)$. That is pretty much the key thing here.

In other words, the transition function for this DFA is given by-

$$\delta((r_1, r_2), z) = (\delta_1(r_1, z), \delta_2(r_2, z))$$

So, basically, what is happening is that where does z take r_1 in the machine M_1 and where does z take r_2 in the machine M_2 and the combination is where you end up. Then a takes or after you see a from the combined state $r_1 r_2$ in the machine M .

And now we need to define what is the static state and what is the accepting state. This is fairly straightforward.

(Refer Slide Time: 20:37)

The slide contains handwritten notes in blue ink. At the top right is the NPTEL logo. The notes are as follows:

- (4) Starting State: $q_0 = (q_1, q_2)$. An arrow points from q_1 to the text "Starting M_1 " and another arrow points from q_2 to "Starting M_2 ".
- (5) Accepting States: We accept w if it ends at (r_1, r_2) if either $r_1 \in F_1$ or $r_2 \in F_2$.

The set of accepting states is defined as:

$$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$$
$$= \underline{(F_1 \times Q_2) \cup (Q_1 \times F_2)}$$

Below the equations, there is a red question: "Q: Will... not $F_1 \times F_2$?"

A video inset on the right shows a man with glasses and a beard, wearing a checkered shirt, speaking.

The starting state is simply the Cartesian pair consisting of the starting states of M_1 and M_2 which is simply $q_0 = (q_1, q_2)$ is the starting state of M . Now which states should we earmark as the accepting states?

So, the thing is that we are trying to generate a machine for the union of two languages. So, if r_1 was an accepting state of M_1 , we should accept. Or if r_2 was an accepting state of M_2 , we should accept. In other words, either if the string gets accepted in M_1 or M_2 we should accept. So, if $r_1 \in F_1$ or $r_2 \in F_2$ we should accept it.

So, in other words, the set of accepting states of the machine M which we call F is simply the set of all pairs (r_1, r_2) where either r_1 comes from the accepting state of F_1 or r_2 comes from F_2 .

The exact expression will be as follows-

$$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

This is not a disjoint union, of course, there would be pairs that r_1 comes from F_1 and r_2 comes from F_2 which will feature in both.

(Refer Slide Time: 23:34)

$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$

$= (F_1 \times Q_2) \cup (Q_1 \times F_2)$

Q: Why not $F_1 \times F_2 = \{(r_1, r_2) \mid r_1 \in F_1 \text{ AND } r_2 \in F_2\}$?

Correctness of this proof is straightforward.

let $w \in A_1 \cup A_2 \Rightarrow w \in A_1 \text{ or } w \in A_2$.

let $w \in A_1$. Then there is a sequence of states $s_0, s_1, s_2, \dots, s_n$ such that

(1) $s_0 = q_1$

Also think about why the accepting states were not $F_1 \times F_2$ (I am not going to explain now but think about this here) which is defined as follows-

$$F_1 \times F_2 = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$$

The correctness of the proof is fairly straightforward. Let us say a certain string that is in the union, some w is there that is in the union. So, now the way this combined Cartesian product machine M operates on w , faithfully replicates what happens when M_1 reads w through the columns and M_2 reads w through the rows.

Now suppose M_1 accepts w and w belongs to the union. Therefore w belongs to A_1 . This means that M_1 accepts w because M_1 is the DFA corresponding to w . Which means there is a sequence of states s_0, s_1, \dots and each transition is proper, and you end up at an accepting state.

(Refer Slide Time: 25:46)

Correctness of the proof is straightforward.



let $w \in A_1 \cup A_2 \Rightarrow w \in A_1$ or $w \in A_2$.

let $w \in A_1$. Then there is a sequence of states $s_0, s_1, s_2, \dots, s_n$ such that

- (1) $s_0 = q_1$
- (2) $\delta(s_{i-1}, w_i) = s_i$
- (3) $s_n \in F_1$

Condition for M_1 to accept w .
When M reads w , it will end in (s_n, n)

When M reads w , then M ends in $F_1 \times Q_2$. Similarly for $w \in A_2$ as well.





- (1) $s_0 = q_1$
- (2) $\delta(s_{i-1}, w_i) = s_i$
- (3) $s_n \in F_1$

Condition for M_1 to accept w .
When M reads w , it will end in (s_n, n)

When M reads w , then M ends in $F_1 \times Q_2$. Similarly for $w \in A_2$ as well.

$\underbrace{F_1}_{\in F} \times \underbrace{Q_2}_{w \in L(M_2)}$

Q: Is it enough to show that for all $w \in A_1 \cup A_2$, w is accepted by M ?



So, s_n is the accepting state of M_1 . Therefore conditions for M_1 to accept w are-

- 1) $s_0 = q_1$
- 2) $\delta(s_{i-1}, w_i) = s_i$
- 3) $s_n \in F_1$

Now, when M reads w this part will be faithfully reproduced in the first coordinate of M .

So, what I want to say is when M reads w , it will end in some state (s_n, r) where r is some state in Q_2 which is in M_2 .

So, which means M ends in $F_1 \times Q_2$. And by our definition it belongs to the accepting state of M . Similar reasoning works for if w was accepted by M_2 as well.

So, what we have reasoned here is that if w is in the union then w is accepted by M_1 or w is accepted by M_2 , it will be accepted by M .

(Refer Slide Time: 28:40)

ϵF $w \in L(M_2)$

Q: Is it enough to show that for all $w \in A_1 \cup A_2$, w is accepted by M ?



Does this mean that $L(M) = A_1 \cup A_2$?

No. This only means that $A_1 \cup A_2 \subseteq L(M)$.

We also need to show that $L(M) \subseteq A_1 \cup A_2$.

$L(M) = B$

Q: Regular languages are closed under intersection. How can we show this?



What we have shown is for all w in A_1 union A_2 , w is accepted by M . So, the question is, is this enough to reason that $L(M) = A_1 \cup A_2$.

The answer is no. Because all this means is that any string that is in A_1 or A_2 is accepted by M . So this only means that $A_1 \cup A_2$ is a subset of $L(M)$. Meaning any string that is an A_1 or A_2 is accepted by M . But we actually have to show equality by saying that no string which is not in $A_1 \cup A_2$, it is not accepted by M .

You also need to show that $L(M)$ is a subset of $A_1 \cup A_2$. Meaning, anything that is accepted is in A_1 or A_2 . This is not that difficult to see as you can reason pretty much the same way. Suppose, a string is not in A_1 or A_2 then it will end in some state (r_1, r_2) where r_1 is not in the accepting state of M_1 and r_2 is not in the accepting state of M_2 .

So, that also needs to be reasoned. What I am saying here or what I am emphasizing here is that when you want to show that a certain DFA corresponds to a certain language, it is not enough to show that all those language members are accepted by the DFA.

We also have to show that the strings that are not in the language are also not accepted. So, in order to show that a language of M is equal to a certain language, it is not enough to show that so, in order to show that what I am saying is, in order to show that language of M equal to a certain language B .

It is not enough to show that all members of B are accepted by M . We also have to show that all members of, all strings that are not in B are not accepted also. So, this just means, if you just show that all members of B are accepted by M , it just means that B is a subset of the language accepted by it. So, we also have to rule out that anything that is not in B is not accepted.