Formal Languages and Automata Theory

Prof. Dr. K. V. Krishna
Department of Mathematics
Indian Institute of Technology, Guwahati

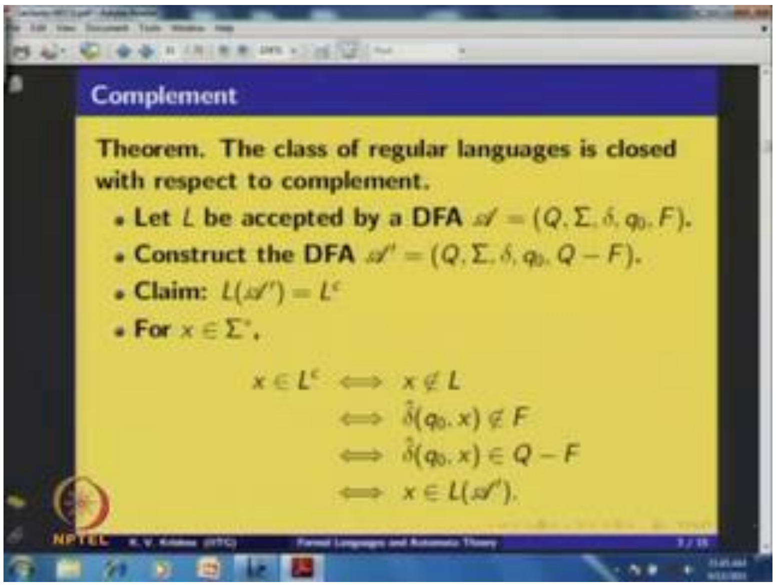
Module - 07
Properties of Regular Languages
Lecture - 01
Closure Properties of RL

In today’s lecture, we discuss about Closure Properties of Regular Languages. Some of the closure properties of regular languages are very straight forward from the definition that you have already observed. For example, if you say the class of regular languages is close with respect to union, that is straight forward from the definition; or you can say that the class of regular languages is close with respect to concatenation that is also straight forward from the definition.

And the class of regular languages is close with respect to cleanly star, that is also the part of the definition. Now, several set theoretic closure properties that we can see, because language is a set, because this is subset of sigma star. So, several set theoretic questions we can have and see like whether the class of regular languages are close with respect to those set theoretic constructs.

Let us go one after another systematically and main point here you can observe that, this closure properties will be helpful to understand further some of the new languages are regular. For example, if you have indentify the language, which is union of some of the known regular languages, you can use the definition or we know the closure property to say that the new language is regular. So, let us first look at this compliment, so that means the question is if you are given a regular language L. Whether L compliment is regular, this is not straight forward from the definition as you understand.

(Refer Slide Time: 02:07)



Complement

Theorem. The class of regular languages is closed with respect to complement.

- Let L be accepted by a DFA $\mathscr{A} = (Q, \Sigma, \delta, q_0, F)$.
- Construct the DFA $\mathscr{A}' = (Q, \Sigma, \delta, q_0, Q - F)$.
- Claim: $L(\mathscr{A}') = L^c$
- For $x \in \Sigma^*$,

$$\begin{aligned} x \in L^c &\iff x \notin L \\ &\iff \hat{\delta}(q_0, x) \notin F \\ &\iff \hat{\delta}(q_0, x) \in Q - F \\ &\iff x \in L(\mathscr{A}'). \end{aligned}$$

Now, the answer is the class of regular languages is close with respect to compliment, that means if you are given a regular language L, L compliment is regular, how do we observe this you consider a DFA for a given regular language. And then you can do this by interchanging the roles of final and non final states, you can quickly understand that a compliment of a regular language is regular.

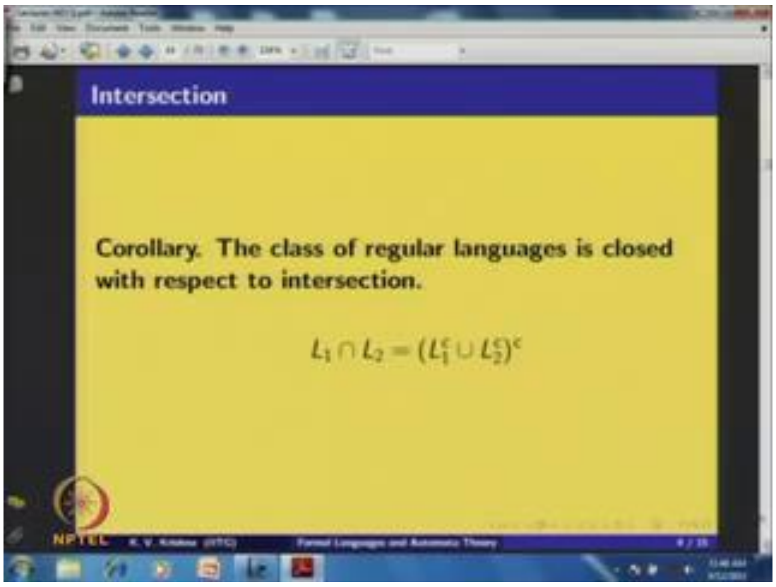
That means, for example, if you consider a language L and a corresponding to which let A be a DFA has given here, Q, sigma, delta, q naught, F. Now, you construct A dash straight set is same as Q as in A, the alphabet of course, is same then the transition map also we are not going to change, the initial state is same. But, the final states now the original non final states you make them as final states, and the original final states you make them as non final states.

That is by interchanging the roles of final and non final states, if you construct, then you understand that the language accepted by the new DFA is compliment of the given language. So, this is the construction of A dash, now you can quickly ascertain that, if you take any x in sigma star, if x is in L compliment that means, x is not is L, that is if you put x in the initial state of the original DFA, then it will not reach to your final state.

If it is not reaching to your final state that means, in the new DFA that is A dash, you see that delta cap of q naught at x by supplying to the initial state, you will be getting a final state of the new DFA. And hence you conclude that this x is accepted by the new DFA and as you see these are equivalent statements, what are what are all written here and hence, we can conclude that the new constructed DFA is precisely accepting the compliment of the given language.

Now, if you have already identified a language is regular, then it is compliment is to understand that the compliment, if you want to see that a language is regular, if it is compliment of a known regular language we can conclude quickly that it is also regular.

(Refer Slide Time: 04:54)



Intersection

Corollary. The class of regular languages is closed with respect to intersection.

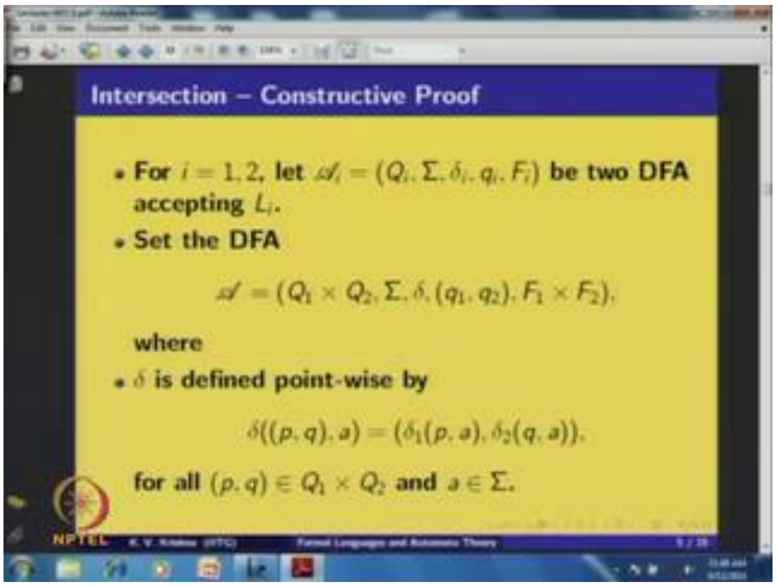
$$L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$$

Now, let us look at certain corollary of this result, for example intersection you consider, the class of regular language is closed with respect to intersection. Why because the intersection has the relation with respect to compliment and unions, as I mean by the De Morgan law I stated here; L 1 intersection, L 2 is L 1 compliment union L 2 compliment whole compliment, this is the De Morgan law.

Now, if you assume that L 1 and L 2 are regular as we have understood that L 1 compliment is regular, L 2 compliment is regular and hence, of union of regular languages that is from a definition, this is a regular language. And compliment of a regular language, once again you use the result just we have observe and understand that intersection of two regular languages is regular.

So, given a language if you want to see that is regular, if it is intersection of some known regular languages, you can quickly conclude using this results that it is regular. But, one may be interested to find a DFA or an NFA finite automaton for a given language to say that it is regular, for which I will give you an alternative proof for this result that means, a constructive proof.

(Refer Slide Time: 06:15)



Intersection – Constructive Proof

- For $i = 1, 2$, let $\mathscr{A}_i = (Q_i, \Sigma, \delta_i, q_i, F_i)$ be two DFA accepting L_i.
- Set the DFA

$$\mathscr{A} = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2),$$

where

- δ is defined point-wise by

$$\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a)),$$

for all $(p, q) \in Q_1 \times Q_2$ and $a \in \Sigma$.

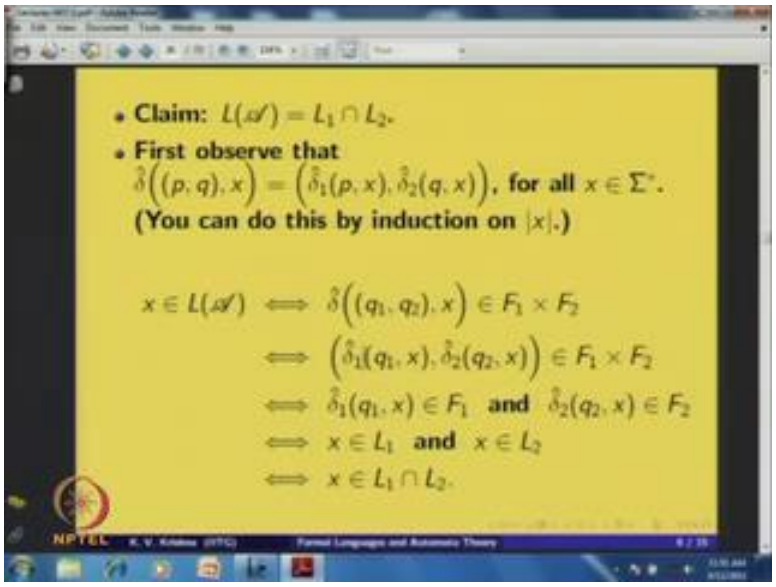
Here, we give a DFA to accept intersection of given two regular languages, so what do you do in general let me give you the construct here, consider two DFA say A 1 and A 2 let me call, I would just state sets Q 1, Q 2 and of course, common alphabet is sigma. And the respective transition maps are delta 1 and delta 2 and the initial states here you may call it as Q 1 and Q 2 and the final states F 1 and F 2 for each here.

So, I am considering two DFA is A i as given here and now if you set the a DFA as given here, this is quintuple, the Cartesian product of the respective states sets of the given DFA. And of course, the common alphabet sigma is input alphabet for this and delta you define point wise, that means if you take any state in the new DFA, it is of the form a ((Refer Time: 07:20)) p, q.

Now, if you take any symbol A in sigma, what do you do, what are the transition that it is taking place in the first two DFA, you give as a first component and similarly in the second DFA what are in the transition it is taking you give it as a second component. And that is how this delta is defined and the initial state component of the first DFA and the initial state of the second DFA, we take it as components and declare that as Q 1, Q 2, here that is the initial state.

And final states is F 1 cross F 2 here, because F 1 is final states of first DFA, F 2 is final states of second DFA, you consider F 1 cross F 2. Now, if you construct a DFA like this, you can understand that this new DFA A is will accept the intersection of L 1 and L 2, where L 1 is the language absolute by A 1 and L 2 is the language absolute by A 2.

(Refer Slide Time: 08:26)



- **Claim:** $L(\mathscr{A}) = L_1 \cap L_2$.
- **First observe that** $\hat{\delta}\big((p,q),x\big) = \big(\hat{\delta}_1(p,x),\hat{\delta}_2(q,x)\big)$, **for all** $x \in \Sigma^*$. **(You can do this by induction on** $|x|$**.)**

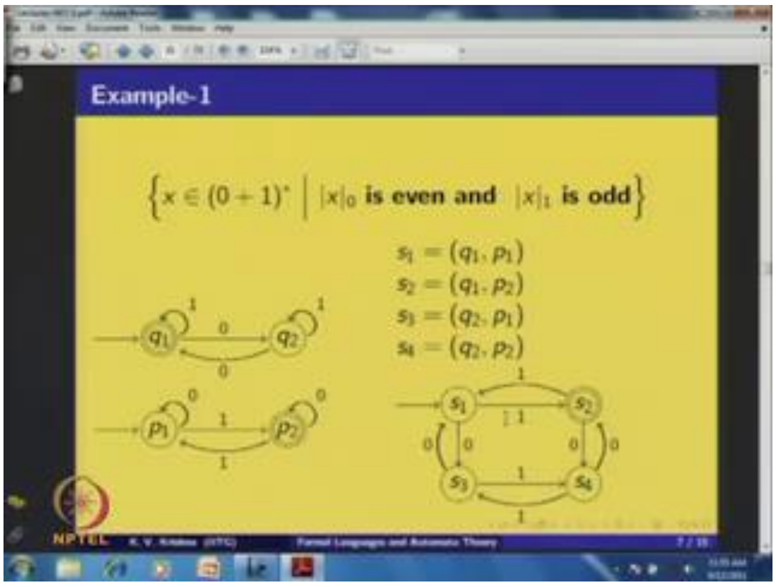
$$\begin{aligned} x \in L(\mathscr{A}) &\iff \hat{\delta}\big((q_1,q_2),x\big) \in F_1 \times F_2 \\ &\iff \big(\hat{\delta}_1(q_1,x),\hat{\delta}_2(q_2,x)\big) \in F_1 \times F_2 \\ &\iff \hat{\delta}_1(q_1,x) \in F_1 \text{ and } \hat{\delta}_2(q_2,x) \in F_2 \\ &\iff x \in L_1 \text{ and } x \in L_2 \\ &\iff x \in L_1 \cap L_2. \end{aligned}$$

For this first observe first observe that delta cap of p, q, for any state p, q of the new DFA if you supply any string x in sigma star, it is precisely delta 1 cap p x comma delta 2 cap q x, this you can observe by applying induction on the length of the string x, that is not a big deal you can take it has an exercise and try that this is happening here. Now, I use this to quickly understand that the new DFA accept precisely alone intersection L 2, now take string that is accepted by the new DFA A.

That means, by definition by this observation you can say that, the component wise that is going in a F 1 cross F 2, because if you say a swing is in a language absolute by A, this is the definition that is going to the final state by supplying the string x in the initial state of the DFA. And now by as this observation, you can say that these are the values of that you are getting in F 1 and F 2, that means if you say a pair is in F 1 cross F 2, the component wise they belong delta 1 cap of q 1 x is in F 1 and delta 2 cap of q 2 x is in F 2.

And now here, now you have got the x is in L 1 and x is in L 2 as well and hence, x is in L 1 intersection L 2 and you understand that, this is you can follow conversely that, if it is in the intersection, this is what is happening. And if it is in L 1 this is a situation with the first DFA and this is situation with the second DFA, and this components if you take this as a pair that is in F 1 cross F 2. And what we have observed by in this line, you can again go back to here, because these two are equal and hence, you can say that x is in language accepted by A. So, this constructive proof gives you a DFA which accepts intersection of two regular languages.

(Refer Slide Time: 10:55)



Example-1

$$\left\{ x \in (0+1)^* \;\middle|\; |x|_0 \text{ is even and } |x|_1 \text{ is odd} \right\}$$

$s_1 = (q_1, p_1)$
$s_2 = (q_1, p_2)$
$s_3 = (q_2, p_1)$
$s_4 = (q_2, p_2)$

Now, let me give you an example, because for example if you consider this all those strings in 0, 1 star with the property that the number of 0's in x is even and the number of 1's in x is odd, suppose if you consider this language, we can observe that it is regular. Of course, you if you know that, with this property if you consider the strings it is regular and with this property if you consider the strings is lower 0, 1 if it is regular.

From the previous result that means, intersection of two regular language is regular you can get it, you see that this DFA this we have constructed earlier, this DFA accepts all those strings over 0, 1 with even number of 0's. Similarly, that this DFA accepts all those strings with odd number of 1's, now from the result you can say the given language is regular, but using the construction you can give the DFA that accepts the given this language.

Now, you consider the Cartesian product of this state's, here you have q 1 q 2, p 1 p 2, now there are 4 pairs that is q 1 p 1, q 1 p 2, q 2 p 1, q 2 p 2, let me call them as S 1, S 2, S 3, S 4. And then the DFA what you have to do you have to put the respective transitions, component wise and construct the DFA, that is a construction given to you, so if you construct using that you will get this. And now you see from the construction i have declare S 2 is final state, because in the first DFA q 1 is final state and second DFA p 2 is final state.

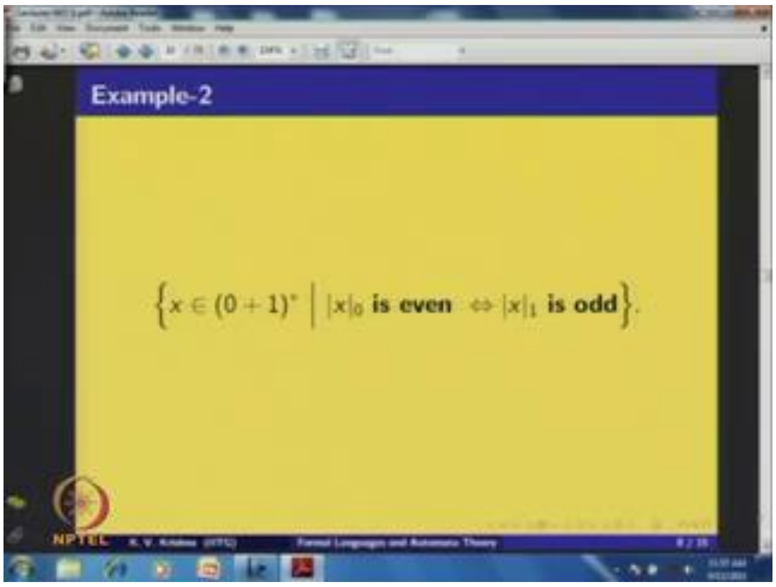
Now q 1 p 2 that is state S 2 if you declare that as final state from the previous result, this particular DFA gives this particular DFA accepts the language given to you, now by choosing appropriate final states here you can vary this. For example if I choose S 3, what is the meaning of S 3 choosing as final state instead of S 2, this q 2 p 1 in q, if you consider the first DFA Q 2 as final state instead of Q 1, you understand that the number of 0 accepted by this, the number of 0's in any string accepted by DFA is art.

And S 3 this is p 1, this is the situation if this is the final state with the second DFA, then the number of 1's in a string accepted by this DFA is even and this you can see if you make S 3 as final state instead of S 2, this accepts all the strings in which the number of 0's of art and number of 1 is even. For example, if you consider S 4 as the final state instead of S 2, that is q 2 p 2, so q 2 if you considering the first DFA this accepts all the strings in which the number of 0's is art.

And in this case number of 1’s is art and this intersection that is number of 0's is art and number of 1’s is also art. Similarly, if you consider S 1 are, now if you consider say S 2 and S 3 or any combination S 2, S 4 both of them that means, then this are this, the property pretending to S 2 are the property pretending to s four. Or if you consider any of the combination you understand that, this particular construct is helpful to understand that various combination of languages with the number of 0's, with the property of number of 0's and number of 1’s whether you an art can be observed here.

So, this construction the Cartesian product construction of course, here we have observed the intersection of two regular languages is regular, but this construction will be helpful to understand some of the complicated language to see they are regular.

(Refer Slide Time: 15:40)

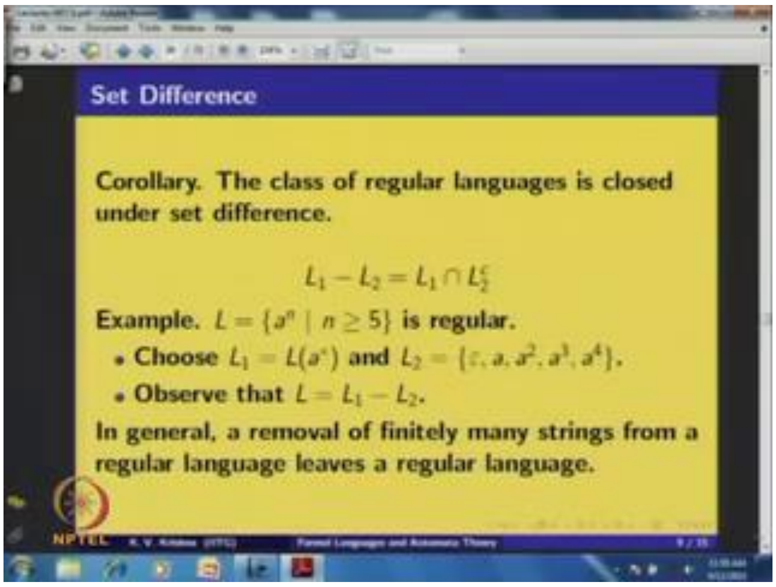


Example-2

$$\left\{x \in (0+1)^* \;\middle|\; |x|_0 \text{ is even} \Leftrightarrow |x|_1 \text{ is odd}\right\}.$$

For example, you consider this set of all those strings power 0, 1 in which and the strings have number of 0's is even if and only if number 1's is odd. If you consider this language you see, this is a conditional statement whenever x has even number of 0's, then the number of 1’s should be odd and conversely, so the condition is looking little bit complicated. So, what do you can little bit analyze this logical statement and formulate the appropriate connectives, in terms of and are not or whatever. And the previous construct, that Cartesian product construct may be helpful to observe that this particular language is regular, you can take it this as an exercise and try to show that this particular language is regular.

(Refer Slide Time: 16:44)



Set Difference

Corollary. The class of regular languages is closed under set difference.

$$L_1 - L_2 = L_1 \cap L_2^c$$

Example. $L = \{a^n \mid n \geq 5\}$ is regular.

- Choose $L_1 = L(a^*)$ and $L_2 = \{\varepsilon, a, a^2, a^3, a^4\}$.
- Observe that $L = L_1 - L_2$.

In general, a removal of finitely many strings from a regular language leaves a regular language.

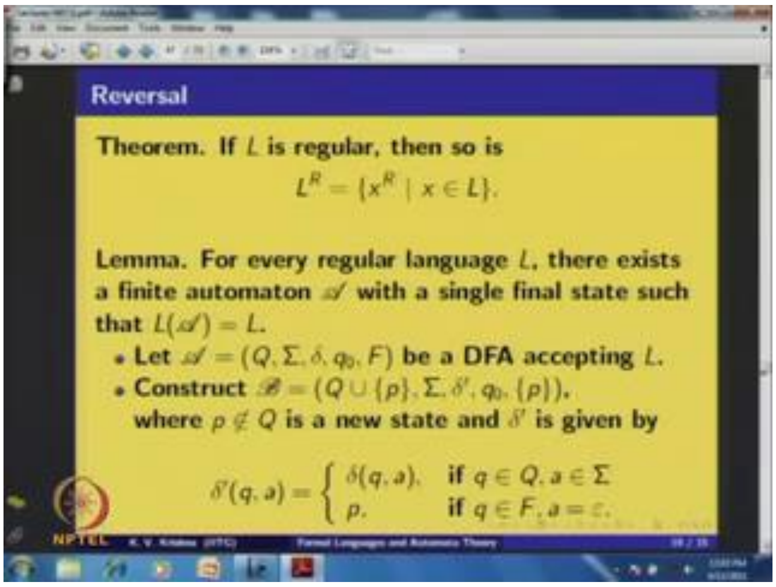
Now, let me consider another set theoretic construct this set difference and see that from the previous results, that means the compliment and intersection if you consider the class of regular languages is close on a set difference. That means, if L 1 and L 2 are regular, then L 1 minus L 2 that is also regular, that L 1 minus L 2 is essentially L 1 intersection L 2 compliment, this is set theoretic property related to set difference.

You see if L 2 is regular L 2 compliment is regular, L 1 is regular is given and L 2 compliment is regular intersection of two regular languages is regular, so you can understand that set difference that preserves regularity or regular languages. Now, let me give you an example consider the language a power n such that, n greater than equal to 5, I means all the strings power single letter alphabet whose length is bigger than equal to 5 this is regular.

Of course, constructing a DFA is not a big deal for you, but let me use this you take L 1 to be L of a star that means, the language represented by the regular expression a star and hence, it is regular. And L 2 you consider this strings, empty string a, a square, a cube, a power 4 consider this strings and a finite set is a regular language, this is regular and L 1 is regular, L 2 is regular, L 1 minus L 2 is also regular and L 1 minus L 2 is nothing else, but the give the language under question.

Here, from this I will give one important observation that is in general, if you remove finitely mini strings from a regular language, that leaves a regular language, this property will be useful in several context. Because, you see that you have come up with the language in which certain things are missing, if the missing things from a known regular language, if what are the missing strings if they are finitely many, then you can conclude that the given language is also regular.

(Refer Slide Time: 19:17)



Reversal

Theorem. If L is regular, then so is

$$L^R = \{x^R \mid x \in L\}.$$

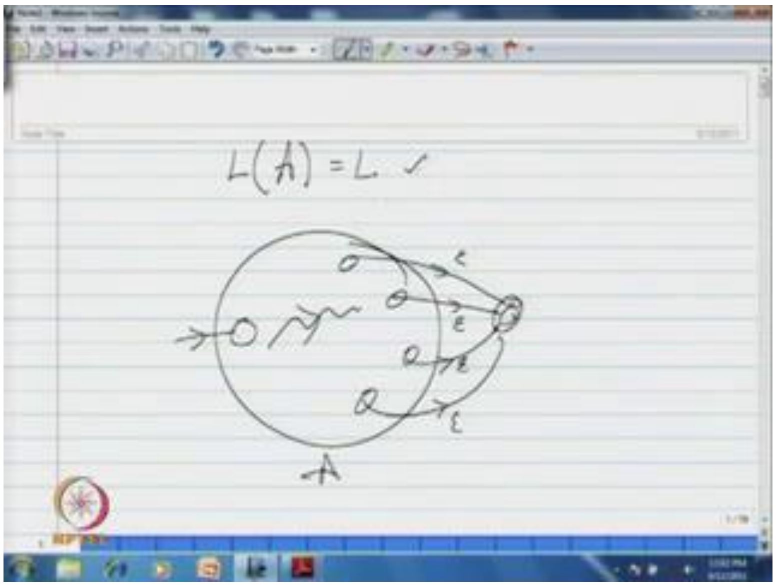
Lemma. For every regular language L, there exists a finite automaton $\mathscr{A}$ with a single final state such that $L(\mathscr{A}) = L$.

- Let $\mathscr{A} = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting L.
- Construct $\mathscr{B} = (Q \cup \{p\}, \Sigma, \delta', q_0, \{p\})$, where $p \notin Q$ is a new state and δ' is given by

$$\delta'(q, a) = \begin{cases} \delta(q, a), & \text{if } q \in Q, a \in \Sigma \\ p, & \text{if } q \in F, a = \varepsilon. \end{cases}$$

Now, I consider this construct that is reversal, reverse of a string, now the property here is if L is regular, then L power R that is defined by, by considering all the strings by reversing the strings of L L power R is equal set of all x power R such that, x is in L. So, take a string from L reverse and put it in L power r, so this is what is l power r, and now I observe that if L is regular l power r is also regular. For this I require a small result, that is for every regular language L, there exist a final automaton A with a single final state such that, L of A is equal to L.

(Refer Slide Time: 20:18)



So, here if you are considering a DFA, what I will do if you are considering a DFA assume L of A is equal to L, for L we construct suppose the DFA is like this, certain states are there, some initial state is there certain final states are there assume some n number of final states are there. Now, what would certain transitions here, what I would suggest, you consider a new state outside this state set of given DFA, you make that as a final state, and remove that there final states you just make them as non final states, these are non final states, whatever the final states that you have earlier in this.

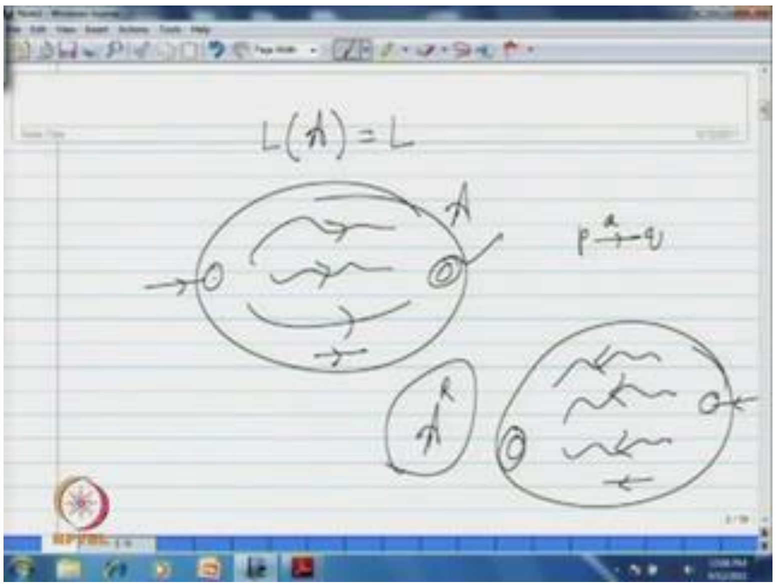
But, you put epsilon transition from here to the new state, and you see any string that is accepted by A is a original DFA, what is the situation any string, if you put it in the initial state here it takes the transition and finally, end up with the final state here. But, since there is an epsilon transition from each final state to this new final state, the original final state to the new final state.

Now, you understand that this particular string, using this first particular string from initial state to the final state you can always reach. And when you can reach to this final state, any string that is ending if you put in a initial state ending in this state, only you can reach to this new final state. Thus you can understand that this worsen of NFA can be constructed with unique final state and which precisely accept to the given language L that is...

Now, if you consider Q, sigma, delta, q naught, F a, DFA accepting L, what do you do you set B to be Q union single term p, a new state p and of course, the original alphabet. All the original transitions of the given DFA will be lying there in addition to that, you take the transition from the final states to the new final state, because this p is declared as new final state, the original initial state is the initial.

So, this delta dash is defined as, if this q is in the original state in a in sigma, then as it is delta of q, a and if this q is in the final state, so in addition to this transitions. We are declaring that, through epsilon transitions you will be reaching to the final state, that is what is a construction I have shown. So, if you construct a finite automaton of course, this is an NFA, then you can understand that this accepts the same language, but you have only one final state.

(Refer Slide Time: 23:36)



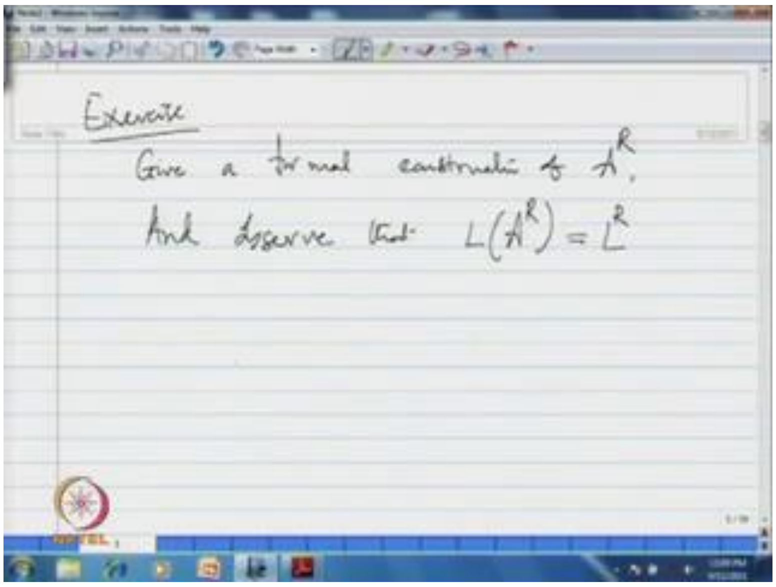
So now, once you have this, what I will do given a regular language I consider finite automaton with unique final state, so given L I consider a finite automaton with unique of course, initial state is always unique, now I have only one final state. Now, you have certain transition here, you have certain transition in this finite automaton, what I would suggest you now whenever you have an arrow in A for example, from p to q, if you have an arrow under A, you simply reverse this arrow in the new DFA.

So, let me call this DFA to be say A R finite automaton, let me not say is a DFA, because whatever that we have constructed with unique final state, that is non deterministic automaton NFA. So, what do you do the final state of the finite automaton A you make this as initial state and the initial state of this you make that as a final state and whenever you have an arrow here from p to q you reverse that arrow, that is all.

So, the transitions will now flow back from state to the initial state here, thus in A power R, you can understand that whatever is the string accepted in A, it is reversal will be accepted in the new finite automaton. Because, you take a string a 1, a 2, a n that you apply it in the initial state here, you are reaching to the final state, now since arrows are reverse.

So, whatever the edges that you have here they are reverse you can understand that, by applying that particular string the reversal of the string a 1, a 2, a n in this state, you will come back to this state; so now what do you do, you give a formal construction.

(Refer Slide Time: 25:54)



Exercise

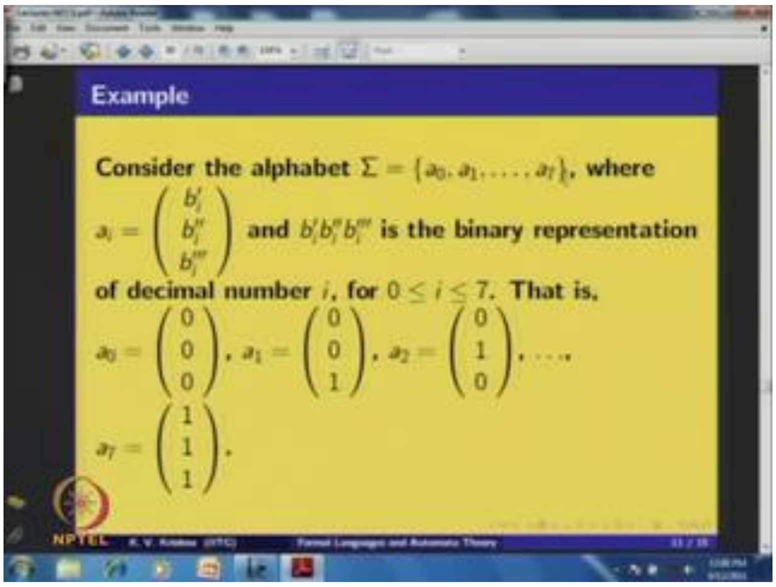
Give a formal construction of A^R.

And observe that $L(A^R) = L^R$

So, an exercise is here is give a formal construction of A power R, you see when I have demonstrated the idea that if a language is accepted by a DFA are finite automaton. For every regular language there is a final automaton with unique initial state, I have demonstrated like you have to construct this way and of course here you have to observe that language accepted by B is nothing else, but L, so this is a part of exercises.

And now you have one more exercise, first you do the formal construction whatever the idea I have explained here, you give a formal construction here of A power R. And observe that this new automaton accepts precisely L power R, so that this as an exercise and give a formal proof that if L is regular, L power R is regular.

(Refer Slide Time: 27:27)



Example

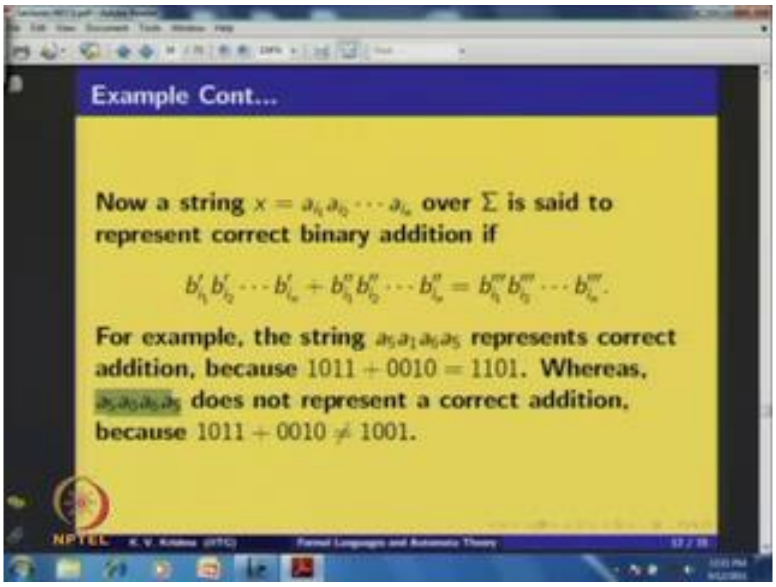
Consider the alphabet $\Sigma = \{a_0, a_1, \ldots, a_7\}$, where $a_i = \begin{pmatrix} b_i' \\ b_i'' \\ b_i''' \end{pmatrix}$ and $b_i'b_i''b_i'''$ is the binary representation of decimal number i, for $0 \le i \le 7$. That is,

$$a_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, a_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, a_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \ldots,$$

$$a_7 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Now, what are the properties, so far we have discussed using this, let me give you an example using all these properties, we conclude that a particular language is regular, is a very important language that you feel, once I present this language. Consider the alphabet which 8 symbol say a naught to a 7, what are this symbol, there essentially the binary representation in this form of the numbers, decimal numbers 0 to 7, that is a naught is triple 0 0 0 like this a 1 is 0 0 1 and so on, a 7 is 1 1 1, so what is a naught, what is a 1 and so on.

(Refer Slide Time: 28:16)



Example Cont...

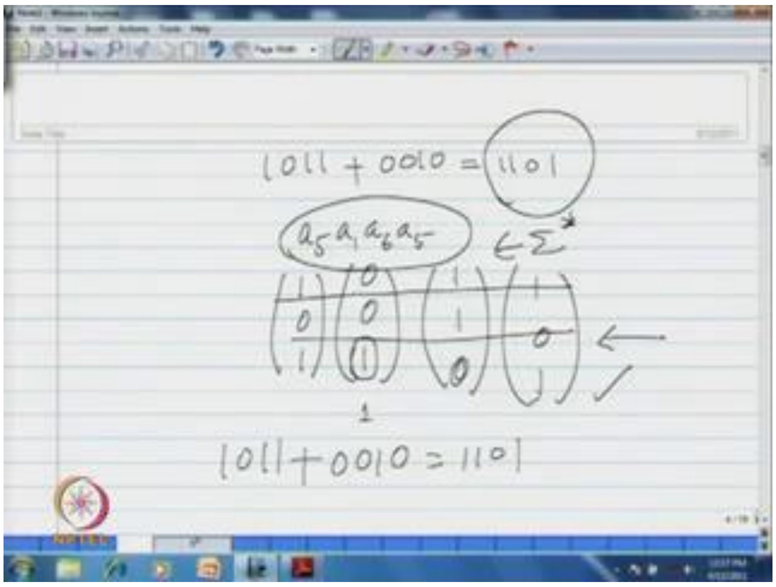
Now a string $x = a_{i_1} a_{i_2} \cdots a_{i_n}$ over Σ is said to represent correct binary addition if

$$b'_{i_1} b'_{i_2} \cdots b'_{i_n} + b''_{i_1} b''_{i_2} \cdots b''_{i_n} = b'''_{i_1} b'''_{i_2} \cdots b'''_{i_n}.$$

For example, the string $a_5 a_1 a_6 a_5$ represents correct addition, because $1011 + 0010 = 1101$. Whereas, $a_5 a_5 a_6 a_5$ does not represent a correct addition, because $1011 + 0010 \neq 1001$.

Now, a string over the sigma this alphabet is set to represent correct binary addition, if correct binary addition, if the first row I will say that is b i 1 dash, b i 2 dash and so on, b i n dash that is a first row of each symbol. And plus second row of each symbol is equal to third row of each symbol, because whatever the string that you consider a i 1, a i 2 and a i n. If you put them side by side the first row of each symbol plus second row of each symbol if you consider those strings, these are the binary numbers, it should be equal to the third row of each symbol whatever the string that you are getting. For example a 5 a 1, a 6 a 5 represents correct addition, because this a 5 a 1, a 6 a 5 if you consider what you have essentially, let me write here.

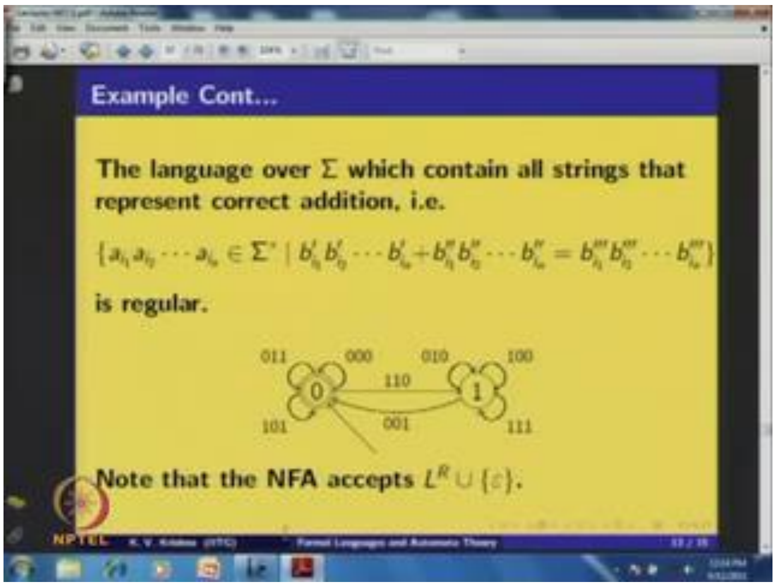
(Refer Slide Time: 29:27)



1 0 1 1, a 5, a 1, a 6, a 5 if you consider a 5 is essentially this 1 0 1 a 1 is 0 0 0 0, this is 0 0 1, a 6 is 1 1 0, a 5 again is 1 0 1, we say this string represents correct addition, because if you consider this first row that is 1 0 1 1, the second row that is 0 0 1 0. If you sum up these two you get this 1 1 0 1 and thus I say that, this string a 5, a 1, a 6, a 5 power sigma represents correct addition.

For example, if you take this a 5, a naught, a 6, a 5 if you put the symbol side by side and see the first row and second row, if you take up the sum, that is not equal to the third row. And thus you see a 5, a naught, a 6, a 5 power this alphabet that does not represent correct addition.

(Refer Slide Time: 31:19)



Example Cont...

The language over Σ which contain all strings that represent correct addition, i.e.

$$\{a_{i_1}a_{i_2}\cdots a_{i_n} \in \Sigma^* \mid b'_{i_1}b'_{i_2}\cdots b'_{i_n} + b''_{i_1}b''_{i_2}\cdots b''_{i_n} = b'''_{i_1}b'''_{i_2}\cdots b'''_{i_n}\}$$

is regular.

Note that the NFA accepts $L^R \cup \{\varepsilon\}$.

So, now you consider the language power sigma, which contains all those strings that represent correct addition that is all those strings a i 1, a i 2, a i n power sigma star, if you sum up the first row with the second row, that should be equal to third row, consider those strings, we observe that it is regular. That means, essentially we are understanding that this binary addition you can represent a language of this form and we can see that particular language is regular.

Now, if you remember we have constructed a mille mission to perform the addition, in which whatever the format that we have followed we have precisely adopted the same format here, whether there you have consider a pair as input and the particular bit as output. Here, the this bit is also included in the input and we have consider each symbol as a triplet and thus you have 8 symbols, so among over this 8 symbols we have formulated the strings and we consider only those stings, which represent correct addition has defined here.

Now, we observe that this particular language is regular, you follow the same pattern as mile mission, for 0 state carry 1 state and you can observe very quickly that, you can identify an NFA accepting this particular language. Let me present that here, this is how the mille mission we have constructed, there only things we have mention, for example if you have 0 1 1, there we have 0, 1 as input 1 as output, because this is a carry 0 state and this carry 1 state.

And for example, here if you give 0 0 as input in the 0 carry state, you will have 0 as output in case of mille mission, here we made it as a triplet, because 0 0 0. So, for convenience I am writing this is horizontally 0 0 0, but the what is the symbol this is vertically we are writing in a bracket that 0 0 0. So, likewise here for each state I have only four symbols you used for 0 carry state and 1 carry state, so among this symbols depending on the carry that if you add for the first two symbols, the third symbols will be reflected as a third component of this triplet.

And thus we have constructed here an NFA and now you can quickly understand by recollecting the construction mille mission are right here also. If you take the string like this a 5, a 1, a 6, a 5 what we are doing, we are summing up this two and whatever is the corresponding symbol that is given as the third symbol here in a triplet. And if you have some carry here, that carry will be considered while counting while summing of the components of the second symbol to look at the answer in the third component.

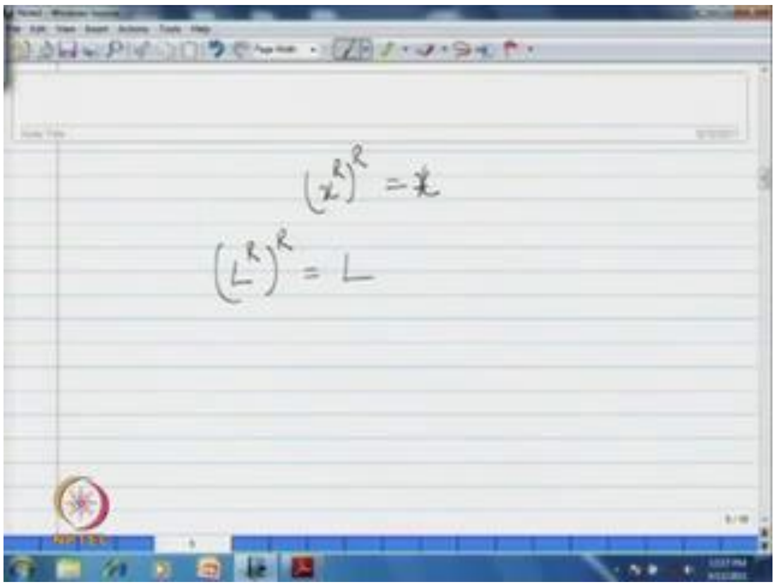
For example, here 1 1 that is you see 1 1 when you are adding whatever is coming up, but what we have to look at here is, when you are adding these things. Because, 1 0 1 1 and this is 0 0 1 0 when I am adding this 1, 0 that is 1 here, when I am adding 1, 1 here 1, 0 here you have 0 as the result and but you have 1 here as a carry. So, when you have 1 carry here, so that is going that is changing the state there the input is 0 0, but you have 1 carry, so the result it is reflecting, 1 1 is reflected.

But, you look at when I am giving the string a 5, a 1, a 6, a 5 I am not getting these transition, because this transitions are essentially carried from a 5, a 6, a 1, a 5 in this direction if you fit to the constructed NFA here, then it represents the correct addition. Because, you are adding from right to left not from left to right, because the carry is important when you are coming from this side to this side, in a usual way of adding to binary numbers.

And thus you understand here whatever is the string addition to by this, those symbols that you are adding component wise from right to left and hence, this strings which are accepted here is the elements of L power R. And of course, since we met this initial state, this initial state as final we have one more string that is empty string accepted by this, but what do we require, we require reversals of the strings in L power R, but we do not want this epsilon as well.

Now, what do you do you apply this properties, this is the language accepted by this NFA and hence, this language is regular, you can safely remove finitely mini strings from a regular language to observe, there is resultant language is regular the properties we have observed. And this what I would suggest first you remove this, what is the resultant 1 that is L power R and from that property you say that L power R is regular.

(Refer Slide Time: 36:51)



$$(x^R)^R = x$$

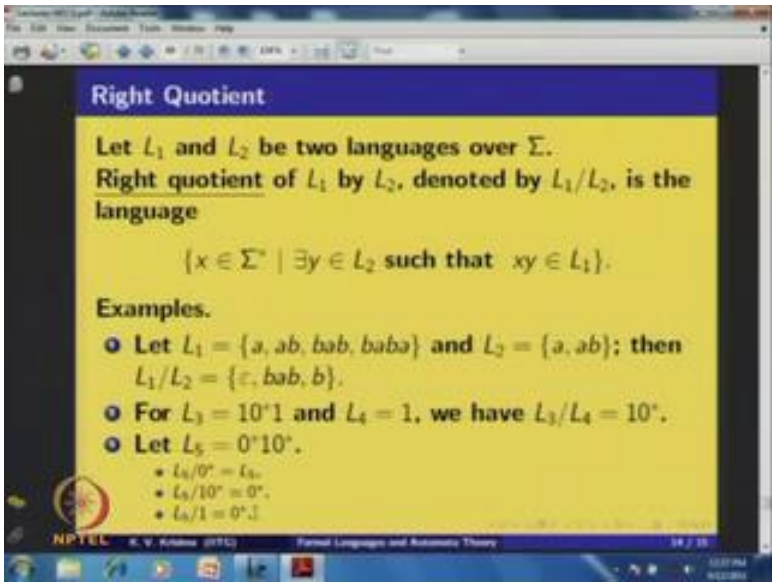
$$(L^R)^R = L$$

Now, since if x power R power R is x this property, this is the property of a reversal of the string, you can understand that if you consider L power R power R, because for each string again you take and obtain the reversal that is what is essentially L. And hence, since L power R is regular it is power that is L only, where L is given precisely here, so we can conclude that for L for this language, we have a finite automaton and hence, this language is regular see what are properties.

So, far we have discussed like reversal of a languages, regular languages regular, intersection and differences, so this particular example has demonstrated some of the things to understand that. But, the language is looking little bit complicated, but you see applying this properties we can conclude, applying this closure properties we can conclude that the language is regular.

Now, let me introduce one more, that is right quotient this is new, because so far whatever that we have discussed, all these things we have already defined earlier, this operations. Now, this is the new operation that we introduce here, that is right quotient between languages.

(Refer Slide Time: 38:35)



Right Quotient

Let L_1 and L_2 be two languages over Σ.
Right quotient of L_1 by L_2, denoted by L_1/L_2, is the language

$$\{x \in \Sigma^* \mid \exists y \in L_2 \text{ such that } xy \in L_1\}.$$

Examples.

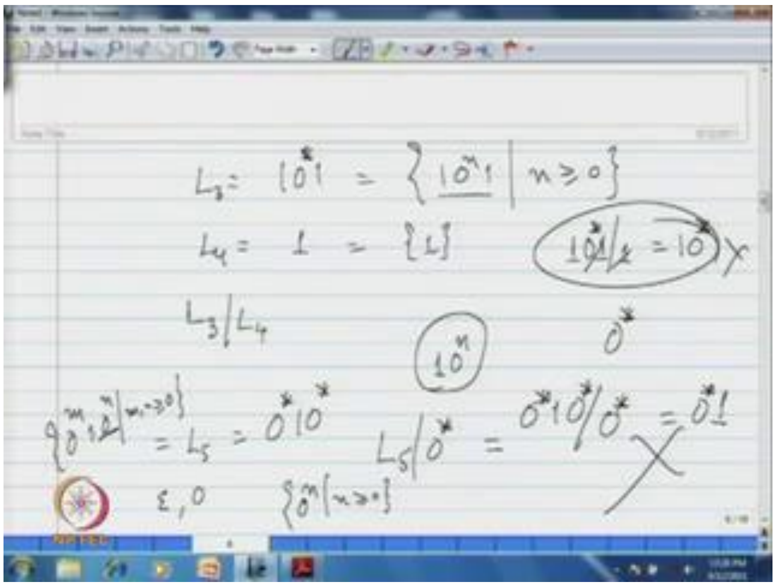
- Let $L_1 = \{a, ab, bab, baba\}$ and $L_2 = \{a, ab\}$; then $L_1/L_2 = \{\varepsilon, bab, b\}$.
- For $L_3 = 10^*1$ and $L_4 = 1$, we have $L_3/L_4 = 10^*$.
- Let $L_5 = 0^*10^*$.
  - $L_5/0^* = L_5$.
  - $L_5/10^* = 0^*$.
  - $L_5/1 = 0^*$.

How do I define this, let L 1, L 2 be two languages for sigma and alphabet, right quotient of L 1 by L 2, we denoted by L 1 by L 2, these are language comparison all those strings x and sigma star. Such that, there is why in L 2 such that, x y is in L 1 that means, you see if you have any string x y in L 1, and there is a suffix in L 2, you can remove that suffix and the remaining you can put it in L 1 by L 2, that is how it is define.

And this you can do it for any string and the resultant language you have construct, let us look at examples, consider because this is a finite language, so that we can understand the things very quickly. Consider L 1 to be this having four strings 1, 2, a, a b, b a b, b a b a and in L 2 I simply consider a, a b, if you want to have L 1 by L 2, what I have to do this two strings wherever that you get as suffix those suffixes you remove. And whatever the remaining prefix that you will be considering in L 1 by L 2, that is the definition.

Now, you see a is suffix of this and this and the resultant strings now here it is, this a for the first string a this is a suffix and the remaining prefix is epsilon and hence, epsilon will be in L 1 by L 2 and a is suffix of this string also. And if you remove this a the resultant string is b a b that will be in L 1 by L 2, similarly if you look at a b, a b is suffix for this for this these two strings and as this is suffix of this and the remaining prefix is epsilon. So, epsilon will go L 1 by L 2 already it is there, and now for this string the remaining prefix is b, so b will go in L 1 by L 2. So, L 1 by L 2 is now epsilon b a b and b, so these three strings are there in L 1 by L 2 consider, let me consider some infinite language, so L 3 is 1 0 star 1 given by this regular expression.

(Refer Slide Time: 41:20)



$L_3 = 10^*1 = \{\underline{10^n1} \mid n \geq 0\}$

$L_4 = 1 = \{1\}$

$\boxed{10^*\cancel{1}/\cancel{1} = 10^*}$ X

L_3/L_4

$\boxed{10^n}$

0^*

$\{0^m10^n \mid m,n \geq 0\} = L_5 = 0^*10^*$

$L_5/0^* = 0^*10^*/0^* = 0^*1$ X

$\varepsilon, 0$ $\{0^n \mid n \geq 0\}$

That means 1 0 star 1 means, for this regular expression the language is 1 0 power n 1 such that, n greater than or equal to 0 and L 4 is consider to be 1, this is given as L 3 represented by this, L 4 is given as 1 the regular expression; that means, this is single term 1. Now, what is L 3 by L 4, suppose if one questions you consider all the strings, which is having 1 as suffix in L 3 and remove that suffix and all the prefixes you collect.

So, a general string is of this form 1 0 power n 1, so in general for each string 1 is there as suffix and thus by removing that you get this is 1 0 star, now the question is, is there any other string will coming to the picture. Here since the regular expression this is 1 0 star 1 and the strings are typically of this form, every string has 1 as suffix and only that suffix you can remove. And the remaining strings are this 1 0 power n of course, for each n greater that equal to 0, and hence you can conclude very quickly that L 3 by L 4 is simply 1 0 star.

Now, let me consider one more example, consider L 5 to be 0 star 1 0 star, now if you consider L 5 by 0 star what is that, that is L 5 given to be 0 star 1 0 star. Now, if you take the quotient with 0 star it is not that, just it is not something like this, do not get confuse with this example, that is in the previous example what has happened. If you consider the regular expression 1 0 star 1 by 1, what we have got this is 1 0 star does not mean that you have cancel this 1 1 and to obtain this, this is not the situation, you cannot adopt the same to conclude here something like 0 star 1 0 star by 0 star by canceling 0 star 0 star, if you write 0 star 1, this is wrong.

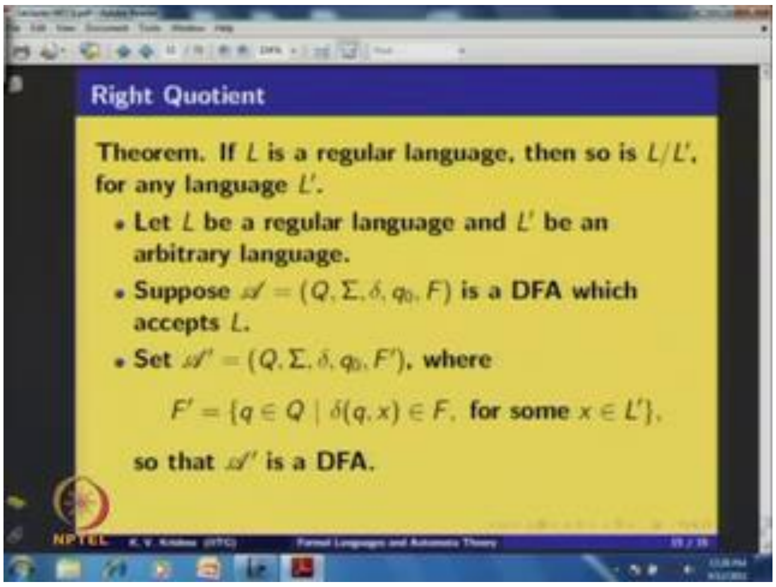
What you have to do you consider from the definition, you consider the strings in which 0 power n, because 0 star is having 0 power n such that, n greater than or equal to 0. So, any string from this that is the typical of the form 0 power n, which is having this as a suffix and that particular suffix you remove and the resultant string that you take it in L 5 is 0 star.

Now, you see for example, if you consider epsilon of course, for any string epsilon is suffix and all the strings are coming into the picture, because of epsilon, because L 5 by signal term epsilon if you look at all the strings are coming. And even if you take say for example, single 0 you see here 0 star 1 0 star, this is having all possible number of 0's as suffixes, so that is 0 power m 1 say 0 power n such, that m n independently greater that equal to 0, so this is the language here.

Since all possible number of suffixes of 0's are available you take anything the remaining string is of the form 0 power m 1 say 0 power n 1, where n 1 can assume any natural number. And here you can observer that it is in fact, L 5 that is 0 star 1 0 star all stings will come in L 5 a 0 star, and now if you consider 1 0 star. Of course, as you see in the arithmetic here you may feel it, but of course you have to do that calculation that looking at a possible subscripts.

Here this one will be helpful to understand that this is 0 star and now if you consider L 5 by 1, that is the language containing single term 1, you understand that it is 0 star, so this is the quotient the operation, quotient between languages.

(Refer Slide Time: 46:43)



Right Quotient

Theorem. If L is a regular language, then so is L/L', for any language L'.

- Let L be a regular language and L' be an arbitrary language.
- Suppose $\mathscr{A} = (Q, \Sigma, \delta, q_0, F)$ is a DFA which accepts L.
- Set $\mathscr{A}' = (Q, \Sigma, \delta, q_0, F')$, where

$$F' = \{q \in Q \mid \delta(q, x) \in F, \text{ for some } x \in L'\},$$

so that $\mathscr{A}'$ is a DFA.

Now, the result is if L is a regular language, then so is L by L dash for any language L dash, so that means, the quotient with any arbitrary language, the quotient of a regular language with respect to any arbitrary language is regular, that is what we are proving here, that is the result. Consider the regular language and an arbitrary language L dash, what do you do since L is regular, you have a DFA let me assume a to be the DFA here, now set A dash to be the same state set same in of course, alphabet is same.

And now consider here L and L dash over the same alphabet sigma assume this way and now transition are same initials state is same, but the final states we change it this way. Consider all those states in q by applying any string x in that particular state, if you reach to your final state, then you can say that state to be the new final state, that is in F dash you can quickly understand that this is a DFA and you can observe that this DFA accepts L by L dash. So, you can take this is an exercise, take any string x that is in L of A dash and observe that this is in L by L dash, so this is a formal proof to understand that L of A dash is L by L dash, you can this is an exercise and conclude that.