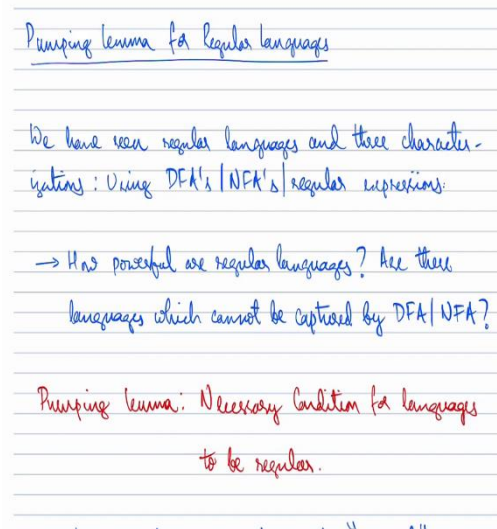


**Theory of Computation**  
**Professor Subrahmanyam Kalyanasundaram**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology Hyderabad**  
**Pumping Lemma for Regular Languages – Part 01**

(Refer Slide Time: 0:18)



Hello and welcome to lecture 12 of the course theory of computation. This is also the first lecture of week 3. So, in the previous weeks, we saw regular languages and we saw three characterizations for regular languages using DFAs, NFAs and regular expressions.

So, there are three models and interestingly all these models, even though they are different in their own ways, turn out to have the same capability, meaning all of them recognize the same class of languages, which are regular languages. So, now, the question is how powerful are regular languages, how powerful are these models?

Are there languages that cannot be captured using these models? So, we know that all of them have the same power, but are there languages that cannot be captured by these models? Or is it the case that almost every language can be captured by these. So, we want to understand what is the computational power of these?

So, pumping lemma gives us one way to understand, one way to answer this question. Pumping lemma is a necessary condition for languages to be regular. Meaning, if a language is regular, certain conditions have to be satisfied. It is necessary that certain conditions have to be satisfied.

(Refer Slide Time: 01:43)

*to be regular.*

$A \text{ is regular} \Rightarrow A \text{ can be "pumped"}$

$\nRightarrow$  *the converse is not true!*

Contrapositive  
 ↑  
 The above implies that if  $A$  cannot be pumped, then  $A$  is not regular. This gives a way to show that certain languages are not regular.

*Caution: This is not a test to show that a language is regular.*



In other words, if  $A$  is regular, then it says that  $A$  can be “pumped”. So, I put pumped in quotes, because when I say it can be pumped it means it can satisfy the conditions, the necessary conditions that I meant. However, this is only in the forward direction, it is not a necessary and sufficient condition. What I mean by that is the converse is not true. Just because a language  $A$  satisfies these conditions, we cannot infer that the language is regular.

So, it is only necessary condition. If  $A$  is regular, it means  $A$  has to satisfy these conditions or  $A$  has to be pumped. Just because  $A$  satisfies this condition does not necessarily imply that  $A$  is regular. However, what does the contrapositive give us? The converse is not true. But what is the contrapositive of the above.

So, what is the contrapositive? The contrapositive is that, if  $A$  cannot be pumped, then  $A$  must necessarily be not regular. So, for  $P \rightarrow Q$  the contrapositive is  $\neg Q \rightarrow \neg P$ . So, the contrapositive here is if  $A$  cannot be pumped, then  $A$  is not regular. So, this means that if  $A$  cannot be pumped  $A$  is not regular. So, we will highlight that.

This is a way to show that certain languages are not regular. If we show that a language cannot be pumped, then it implies that the language is not regular. Once again it is only the contrapositive that is true. For any statement the contrapositive is and the converse is not true. So, just because it can be pumped, we cannot infer that a language is regular. It is only a necessary condition which means, if it does not meet this condition, then you can infer the language is not regular. The converse is not true. So, just because it satisfies this condition, we cannot infer the language is regular.

(Refer Slide Time: 04:10)

$\{ \epsilon, 01, 0011, 000111, \dots \}$



Example:  $B = \{ 0^n 1^n \mid n \geq 0 \}$ .

A DFA that recognizes  $B$ , in some sense, has to count the number of 0's and then cross check it with the number of 1's. But  $n$  can be arbitrarily large. So this cannot be done with a limited no. of states.

*This is still an intuition!*

Pumping lemma gives us a way to formalize this.

Pumping lemma was discovered by Michael Rabin



So, before we get on to the formal statement of the lemma, pumping lemma, let us just see an example of a language that is not regular. The language is the language  $B = \{ 0^n 1^n \mid n \geq 0 \}$ . So, when  $n$  is 0, it is just the empty string. When  $n$  is 1 it is the string 01 when  $n$  is 2 it is 0011 then 000111, where  $n$  is 3 and so on.

So, there are infinite strings here. For each value of  $n$ , there is a string here. We have  $n$  number of 0s followed by  $n$  number of 1s. It is a very clearly defined language. So, it is very easy to understand what this language is. Interestingly, this language is not regular. It is a simple enough language. But interestingly, it is not regular. The reason is that, if it is regular, then we know there is a DFA that recognizes this language.



So now the DFA has to, in some very vague sense, check how many 0s are there, followed by how many 1s are there. And they have to be equal. So first of all, the string has to be of the form some 0s followed by some 1s. And then it also has to be checked that the number of 0s and the number of 1s are the same. We know a DFA can only scan the string once. So, it means that we have to count the number of 0s. And then we count the number of 1s and then we have to check that they are the same.

But what can  $n$  be? So, this is an infinite language,  $n$  could be an arbitrary large.  $n$  could be 1000,  $n$  could be a million. So, if you give me a DFA with, let us say, 100 states, it cannot keep track of a huge count, of a count of million with 100 states. If you give me a DFA with a fixed number of states, I will give you a string where  $n$  is really, really large. It will be kind of clear that this relatively small sized DFA cannot keep track of such a big count.

So, that is the reason. That is kind of an intuition, why this language is not regular. So, this is still not a formal proof. Perhaps it does not need to count, it can do some other thing. Perhaps, somehow without counting itself, it is able to do something using some strange modular arithmetic or something like that. So, how do we know that we indeed have to count, so that is another valid question.

But then this is just an intuition. Let us see pumping lemma which will give us a clear proof that this is not regular. So, we will not see the proof in this lecture, we will see the proof in the next lecture. But in this lecture, we will see the pumping lemma itself. So, this is still an intuition. One has to remember this is still an intuition. So, we cannot say that it has to count.

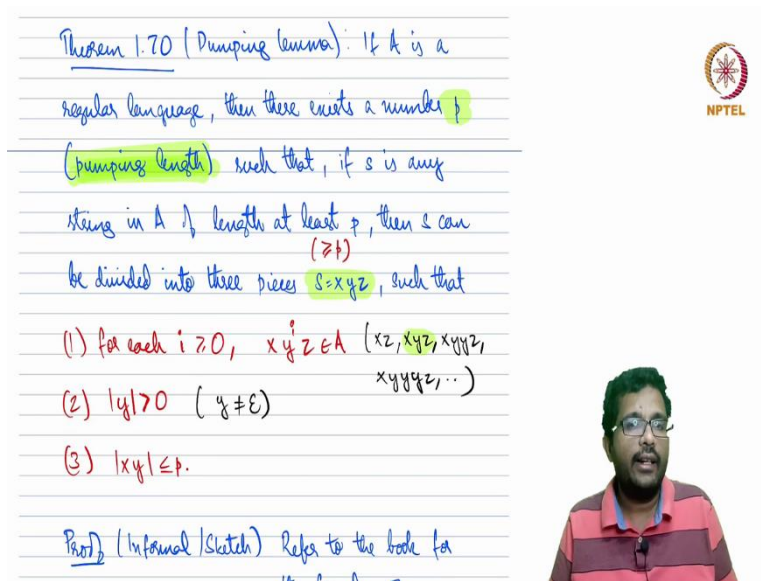
(Refer Slide Time: 07:35)



Pumping lemma gives us a way to formalize this. And this is pumping lemma for regular languages. We will also see another version of pumping lemma for context free languages a few weeks from now. It is not the same, but it is something similar. What we have seen now is pumping lemma for regular languages. It gives us a necessary condition for regular languages, it was discovered by Michael Rabin and then Dana Scott in '59, followed by Yehoshua Bar-Hillel, Perles and Shamir, in '61.

In those days, the science used to travel slow, like we did not have electronic communication. It was very common that somebody discovers this and somebody else also discovers the same thing. Independently, because things have to be studied published and printed. And then journals, books have to travel across the globe and so on.



(Refer Slide Time: 08:33)



Theorem 1.70 (Pumping lemma): If  $A$  is a regular language, then there exists a number  $p$  (pumping length) such that, if  $s$  is any string in  $A$  of length at least  $p$ , then  $s$  can be divided into three pieces  $s = xyz$ , such that

- (1) for each  $i \geq 0$ ,  $x y^i z \in A$  ( $xz, xyz, xyxz, xyxyz, \dots$ )
- (2)  $|y| > 0$  ( $y \neq \epsilon$ )
- (3)  $|xy| \leq p$ .

Proof (Informal/Sketch) Refer to the book for



So now let us get to what the pumping lemma is. For pumping lemma, there is a statement in the book, it is theorem 1.70. If  $A$  is a regular language, then there is a number  $p$ , which is called the pumping length, such that for any string  $s$  in  $A$  that has length at least  $p$ , by at least I mean, greater than or equal to  $p$ , then we can split  $s$  into three pieces  $x y z$ , or  $s$  can be written in this manner  $xyz$  such that the following three conditions are satisfied. What are the three conditions?

Consider the string  $xy^i z$ . For each  $i$  consider this string, then all of these strings are in  $A$ . With  $i$  as 0, this gives us a string  $xz$ . So,  $xz$  is what we get when  $i$  is 0. When  $i$  is 1, it gives us  $xyz$ . When  $i$  is 2 it gives us  $xyyz$  when  $i$  is 3 it gives us  $xyyyz$  and so on.

So, this is an infinite family of strings. What it is saying is that all the members of this family must be in  $A$ . If  $A$  is a regular language, there is a pumping length such that if you give me a string of length at least as much as the pumping length, then there should be a way to divide it into  $x y z$ , such that, now you repeat  $y$  twice, so in  $xyz$ ,  $y$  comes once, you repeat it twice, thrice, four times, even 0 times, all of these strings  $xz, xyz, xyyz$ , are in  $A$ .

So, we know that  $xyz$  is in  $A$  because  $xyz$  is equal to  $s$ , and by the statement itself, we chose  $s$  to be a string in  $A$ . But what we are saying is that all the other strings, in fact, an infinite number of strings, all of them have to be in  $A$ . And further, two more conditions. The length of  $y$  is strictly greater than 0.

There is only one string of length 0, which is the empty string. So, in other words, it is saying that  $y$  is not the empty string. So, it cannot be the empty string, it could be any other string,

that has length at least 1. The third condition says that length of  $x$  and  $y$  put together has to be at most  $p$ , less than or equal to  $p$ .

So, once again, if  $A$  is a regular language, there is a pumping length  $p$  such that for any string that you take in the language, of length at least  $p$ , there is some way to divide this string into three pieces  $x y z$ , such that these three conditions are satisfied. So, condition 2 and 3 are length conditions on the length of  $y$  and  $x$ . Condition 3 says that  $xy$  has to be of length at most  $p$ , condition 2 says that  $y$  has to be non-empty. And condition 1 says that  $xy^i z$  is in  $A$ , meaning  $xz$  is in  $A$ ,  $xyz$  is in  $A$ ,  $xyyz$  is in  $A$ ,  $xyyyz$  is in  $A$  and so on.

So, any string of length that is at least the pumping length can be split in a manner that satisfies these three conditions. That is the pumping lemma. Now, the way to show that a certain language is not regular is that, for that language we will have to come up with a string, which we cannot split in a way that meets these conditions.

Informally, we say that string cannot be pumped. So, let us see why this is true. Why is it the case that there is some length beyond which any string can be split in this manner? So that is the proof. Let us see the proof.

(Refer Slide Time: 12:58)

SIPSEK

Proof (Informal/Sketch) Refer to the book for the formal proof.

Let  $M = (Q, \Sigma, \delta, q_1, F)$  be a DFA that recognizes  $A$ .

We will show that pumping length  $p = |Q|$ .  
(number of states)

→ What if all strings  $s \in A$  are of length  $< p$ ?

Then pumping lemma holds vacuously.  
(VACUOUSLY)

→ For strings  $s \in A$  with  $|s| \geq p$ , we will use an argument based on the pigeonhole principle.

We will show max pumping length  $p = |Q|$ .  
(number of states)

→ What if all strings  $s \in A$  are of length  $< p$ ?

Then pumping lemma holds vacuously.  
(VACUOUSLY)

→ For strings  $s \in A$  with  $|s| \geq p$ , we will use an argument based on the pigeonhole principle.

Consider the string  $s = s_1 s_2 \dots s_n$

$s = s_1 s_2 s_3 s_4 s_5 s_6 \dots s_n$

So, what I will be describing here is kind of an informal proof or somewhat more developed sketch of the proof, because I think that is much easier to communicate. There is a formal proof in the Sipser book. So, in the Sipser book, the formal proof has more notations and formalisms. What I am going to explain is somewhat more informal, but it is pretty much the entire proof. I think for instructive purposes, this is better.

So, we have a regular language  $A$  and then we have to show that there is a pumping length such that the following conditions are satisfied. Because the language is regular, we can assume that there is a DFA  $M = (Q, \Sigma, \delta, q_1, F)$ . Let  $Q$  be the set of states and  $q_1$  is the starting state.

Now, the pumping length will be the number of states. So, I am upfront telling you the pumping length. We know that DFA means it is a finite set of states. So,  $Q$  is finite. So, pumping length

$p$  is the size of  $Q$ , cardinality of  $Q$ , in other words, the number of states. So, let me ask you a simple question, which is like a basic question on logic, what if all the strings are of length at most  $p$ . So, pumping lemma talks about strings greater than or equal to  $p$ .

So, what if all the strings are of length strictly less than  $p$ . So, if all the strings are of length less than or strictly less than  $p$  then let us read this statement. If  $A$  is a regular language then there is a number  $p$  such that if any string is of length greater than or equal to  $p$ , then this has to be satisfied. So, let us say we take an  $s$  that does not meet this condition, that is of length less than  $p$ , then nothing has to be satisfied. So, this is a situation where we say, the statement is vacuously true.

So, if  $P$  then  $Q$ . Let us say this there is a statement of the form if  $A$  then  $B$ . Now, suppose the condition  $A$  itself is not satisfied then trivially the statement is true, because  $A$  itself is not satisfied. It is like saying, if it rains today, I will give you 1000 rupees. Suppose it does not rain today, then the statement is true even if I do not give you 1000 rupees, because it is a conditional statement and the condition itself was not met.

Similarly, if all the strings that in the language are of length strictly less than  $p$ , then this statement is true in a vacuous sense or in an empty sense, because the condition itself is not met. So, then pumping lemma holds vacuously. So, suppose there are strings of length at least  $p$ , suppose  $A$  contains strings of length at least  $p$ , greater than or equal to  $p$ . Then let us see how to deal with it.



(Refer Slide Time: 17:03)

Consider the string  $s = s_1 s_2 \dots s_n$

$s = s_1 s_2 s_3 s_4 s_5 s_6 \dots s_n$

$q_1 q_3 q_9 q_{16} q_{12} q_9 q_{11} q_6 q_{13}$

Consider the sequence of the states that  $M$  goes through while processing  $s$ . This starts with  $q_1$ , then say  $q_3 \dots$  etc. till say  $q_{13}$  → Accept State

If  $|s| = n$ , then this sequence has  $n+1$  states, with some possible repetitions. When  $|s| = n \geq k$ , by pigeonhole principle, there exists at least one

So, let us consider some string  $s$ . So, let us say  $s$  has length  $n$ . And here I have written  $s$  as  $s_1, s_2$  up to  $s_n$ , I am writing each symbol, symbol by symbol, so length of  $s$  is equal to  $n$ . I am writing  $s_1$  as a first symbol, second symbol, and so on. Now, let us see how the machine processes this. So before reading any symbol, the machine, let us say is in state  $q_1$ . It has to be in state  $q_1$  because  $q_1$  is the starting state of the machine.

Before reading any part of the string  $s$ , let us say the machine is in  $q_1$ . After reading  $s$ , let us say it transitions to  $q_3$ , and then it goes to  $q_9$ , after reading  $s_1, s_2$ . Then let us say  $q_{16}, q_{12}$ , and so on. And finally, after reading all the  $n$  symbols, it ends at  $q_{13}$ . So, this is just some numbers that I am taking. So the first state has to be  $q_1$ , because it is a starting state.

And let us say it ends at some state, let us say  $q_{13}$ . What do we know about  $q_{13}$ ? One point is that  $s$  is a string that we assumed to be in the language. So, which means the DFA will accept the string because it is in the language and the DFA is for that language. So, this has to be an accepting state. So this has to be an accept state of  $M$ . Because the string is in the language. So, we know  $q_1$  is the starting state, we know  $q_{13}$  is one of the accept states.

Now, let us see how many such states are there. So, how many states are there? There are  $n$  symbols here. You start with something and after each symbol is processed, there is one transition. You would get one more state, so there are  $n + 1$  such states listed here. And out of these  $n + 1$ , I am not claiming that all the states are distinct. So, there could be I am just saying that the  $n + 1$  states appear and some of them could be repetition.

(Refer Slide Time: 19:26)



If  $|s| = n$ , then this sequence has  $n+1$  states, with some possible repetitions. When  $|s| = n \geq p$ , by pigeonhole principle, there exists at least one repeated state, since  $n+1 \geq p+1 > p = |Q|$ .



In the above,  $q_9$  is the repeated state. let

$x =$  string till the first occurrence of  $q_9$ .



So, I will highlight that “with some possible repetitions”. Now, the point is, if the length of  $s$  which is equal to  $n$  is at least the pumping length, at least  $p$ . Suppose  $n$  is at least  $p$ , which is that  $n$  is greater than or equal to  $p$ . Then there at least  $p + 1$  states listed here. But you remember, we chose pumping length as the number of states.

So, if we write  $p + 1$  or more states, we know that the machine itself has  $p$  states, so something has to repeat. Let us say you write an eleven digit number. At least some digit has to repeat because we only know ten digits 0 1 2 3 up to 9. So, like that, there has to be at least 1 repeated state because  $s$  is long enough.

And in the manner I wrote this particular string, the repetition happens with  $q_9$ . Repetition happens with  $q_9$ . It is seen after  $s_2$  and it is seen again after  $s_5$ . So, now, let us do the following. Now, I will call the string till the first occurrence of  $q_9$  as  $x$ . From the first occurrence till the second occurrence, I will call it  $y$ .

And from the second occurrence till the end of the string, I will call it  $z$ . So, here in this figure,  $x$  is  $s_1 s_2$ ,  $y$  is  $s_3 s_4 s_5$  and  $z$  is  $s_6 s_7$  up to  $s_n$ . So, the rest of the string. Again,  $x$  is the string till the first time  $q_9$  occurs,  $y$  is the string from the first till the second occurrence of  $q_9$ . And  $z$  is the rest of the string. So, now let us see what  $x y$  and  $z$  are doing.

(Refer Slide Time: 22:04)

$x =$  String till the first occurrence of  $q_9$ .

$y =$  String from the first to the second occurrence of  $q_9$ .

$z =$  String from the second occurrence of  $q_9$  till the end of  $s$ .

In the DFA  $M$ ,  $x$  takes  $M$  from  $q_1$  to  $q_9$ .

$y$  takes  $M$  from  $q_9$  to itself.

$z$  takes  $M$  from  $q_9$  to  $q_{13}$ .

Consider  $x y z$ . This will also be accepted by  $M$ .

The difference is that there are two rounds of  $y$  instead of one. Similarly for  $x y z$  and  $x z$ .

So, upon reading  $x$ , the machine transition from  $q_1$  to  $q_9$ . So, it went from  $q_1$  to something, then to  $q_9$ . Upon reading  $y$ , it started from  $q_9$ . So, before reading  $y$ , it was at  $q_9$ , then it went to  $q_{16}$ ,  $q_{12}$ , finally, it came back to  $q_9$ .

So,  $x$  moved the machine from the start state to  $q_9$ ,  $y$  started from  $q_9$  and took it through some states and brought it back to  $q_9$ . And  $z$  took the machine from  $q_9$  to  $q_{13}$  which is an accept state.

So, let us try to maybe draw it, depict it pictorially. So let us say  $x$  does this,  $q_3$ ,  $q_9$ . And then let us say some states,  $q_9$  does something. And so,  $x$  takes it from  $q_1$  to  $q_9$ ,  $y$  takes it through the circle. So, through some states and then comes back to  $q_9$ . And  $z$  takes it from  $q_9$  to  $q_{13}$ . Notice that  $q_{13}$  is an accept state so I have drawn the double circle. This is what happens, this

is what each of these sub strings does. And together  $x$  followed by  $y$  followed by  $z$  is equal to the string.

So,  $s$  is equal to  $xyz$ . So, what did we do? We this first of all claimed that if the string is of length that is at least the pumping length, some symbol has to appear repeated, meaning two times at least. Now, you look at the first occurrence of that symbol and the second occurrence of that symbol. And then you consider start to the string to the first occurrence is  $x$ .

First occurrence to the second occurrence is  $y$  and second occurrence to the end of the string is  $z$ . The, the claim is that this  $x$  this split  $x y z$  satisfies the conditions of the pumping lemma. So, all we used was that there is a DFA for the language. And then we set the pumping length to be the number of states. And that in turn gave us everything. So, again same thing that I said earlier,  $x$  takes  $M$  from  $q_1$  to  $q_9$ ,  $y$  takes  $M$  from  $q_9$  to itself and  $z$  takes  $M$  from  $q_9$  to  $q_{13}$ , which is an accept state. So, we will put double circle here as well.

Now, let us see what if instead of  $xyz$  what would have happened if I had  $xyyz$ . So,  $x$  would have taken from  $q_1$  to  $q_9$ ,  $y$  would have taken from  $q_9$  to itself. Again, we would have taken one more round to the same path. So, two  $y$ 's and then that would take you to accepting state.

Again, if I had  $xyyyz$ , it is the same story. Instead of two rounds of  $y$ , we will have three rounds of  $y$ . Even if  $xz$ ,  $x$  would take from  $q_1$  to  $q_9$  and  $z$  will take from  $q_9$  to  $q_{13}$ . So, all these strings  $xz$ ,  $xyyz$ ,  $xyyyz$  all of these will be accepted by the DFA. Which means that all of them are in the language. Which means all of these strings that I wrote  $xz$ ,  $xyyz$ ,  $xyyyz$ , they are in the language. Because the DFA is a DFA for the language. So, anything that is accepted by the DFA is in  $A$ .

(Refer Slide Time: 26:10)

Consider  $x y z$ . This will also be accepted by  $M$ .  
The difference is that there are two rounds of  $y$  instead of one. Similarly for  $x y^2 z$  and  $x z$ .

Hence  $x y^i z$  is accepted by  $M$  and hence is in  $A$  for all  $i \geq 0$ .

Since there are two occurrences (or more!) of at least one state ( $q_9$  here in the above example), there exists a non empty string that is processed in between. Thus  $|y| \geq 1$  (or  $y \neq \epsilon$ ).

at least one state ( $q_9$  here in the above example), there exists a non empty string that is processed in between. Thus  $|y| \geq 1$  (or  $y \neq \epsilon$ ).

Pigeonhole principle guarantees that the first repetition occurs on or before  $s_p$  (the  $p$ th symbol of  $s$ ) is processed. Hence  $|x y| \leq p$ .

Exercise: Go through the formal proof in the book.

Hence, we get that all the strings  $xy^iz$  are in the language. So, whatever I wrote are the strings in  $xy^iz$ , they are all in the language. So, this is the condition one of pumping lemma. Now, we have to show that  $y$  is not empty and length of  $xy$  is at most  $p$  which is not that hard. And we know that there are two occurrences of some state.

So, between the two occurrences to say  $q_9$  the closest that these two occurrences can come is just before a symbol and after a symbol. And this length, the string  $y$  is from the first occurrence to the second occurrence. So, the first and second occurrence are at least one symbol apart. Because, at least one symbol is needed.

Hence, there exists at least one symbol for  $y$ . So, the length of  $y$  cannot be 0, it has to be strictly greater than 0. In other words,  $y$  cannot be the empty string. And finally, we want to show that

the length of  $xy$  is at most  $p$ . So, to ensure that what we do is we look at the first  $p + 1$ , we look at the first  $p + 1$  states here. We look at the first  $p + 1$  states. In other words, we look at the first  $p$  symbols of the string.

The first  $p$  symbols of the string itself take it to  $p + 1$  state starting from the starting state, second state, third state. After reading the  $p^{\text{th}}$  symbol it will be in the  $p + 1$  state. So already there are  $p + 1$  states. But in the  $p + 1$  states that the DFA traversed, we know that the DFA has only  $p$  states. So, we know for sure that even after reading the first  $p$  symbols, some state has to repeat.

This means that by pigeonhole principle, there must be some repetition occurring before the  $p^{\text{th}}$  symbol of the string. Pigeonhole principle is something like saying, if there are ten objects and nine holes, you put the objects into holes. Then at least one hole has two objects. So, here we have  $p + 1$  states, but we know that the actual number of states are only  $p$ , so some state has to come twice.

So, before we reach this  $p^{\text{th}}$  symbol or at the time we reach  $p^{\text{th}}$  symbol there has to be a repetition. So, we know that. What is  $x$  and what is  $y$ ?  $x$  is the part from the beginning to the first time the symbol repeats, the state repeats and  $y$  is it from the first time to the second time. So,  $xy$  is at most  $p$  symbols.

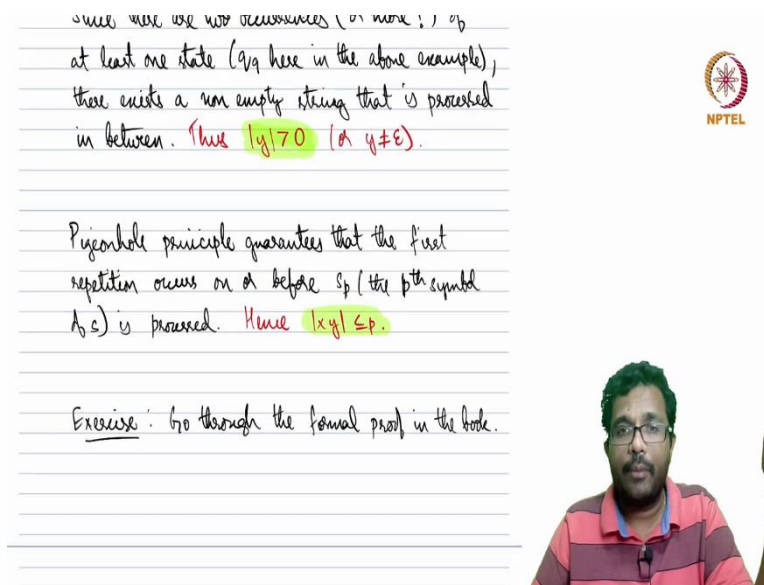
So, which is what the third condition is, length of  $xy$  is at most  $p$ . And that is all. So, because that is the third condition. And that first part which is  $xy^i z$  is in  $A$ . Second is  $y$  is non empty and third is the length of  $xy$  is at most  $p$ . These three conditions together form the pumping lemma. So, this is fairly rigorous, though I present it in a somewhat informal way. You may read and have a look at the formal proof in the book, which is quite notation heavy but very terse like just two short paragraphs.

But it conveys the same message. So, just to understand the formalism involved, I suggest that you go through this, go to the proof. And once you have the informal idea it should be much easier to understand what is going on. If you read the terse proof in the book after listening to this presentation.

So, just to summarize, pumping lemma for regular languages is a necessary condition for the languages to be regular. It is not a sufficient condition. If a certain language does not meet the conditions of pumping lemma or it cannot be pumped, then we can infer that the language is not regular. It is not a sufficient condition. Because it meets the conditions, we cannot infer

that a language is regular. It is only a necessary condition. We can only use this to show that languages are not regular. We cannot use this to show that languages are regular.



(Refer Slide Time: 30:53)



Since there are  $n$  occurrences (or more) of  
at least one state ( $q_i$  here in the above example),  
there exists a non empty string that is processed  
in between. Hence  $|y| > 0$  ( $\& y \neq \epsilon$ ).

Pigeonhole principle guarantees that the first  
repetition occurs on  $\alpha$  before  $s_p$  (the  $p$ th symbol  
of  $s$ ) is processed. Hence  $|xy| \leq p$ .

Exercise: Go through the formal proof in the book.



So, it says that, if  $A$  is a regular language, then there is a pumping length and any string that is of length greater than pumping length can be split into three pieces  $x y z$ , such that these three conditions are satisfied. And the main idea of the proof was that first of all, if the language is regular then there has to be DFA. So, suppose a DFA has ten states, now, you consider a string of length at least ten.

We know that when processing the first ten symbols, the DFA traverses through at least eleven states. So, some state has to repeat. This repetition has to occur when the first ten symbols are processed. Using this repetition, we have figured out the splits  $x y$  and  $z$ . And it is quite easy to see that it meets these conditions. That is the summary of the proof and of the pumping lemma. In the next lecture, we will see some illustrations, some examples of the how pumping lemma can be used. Thank you.