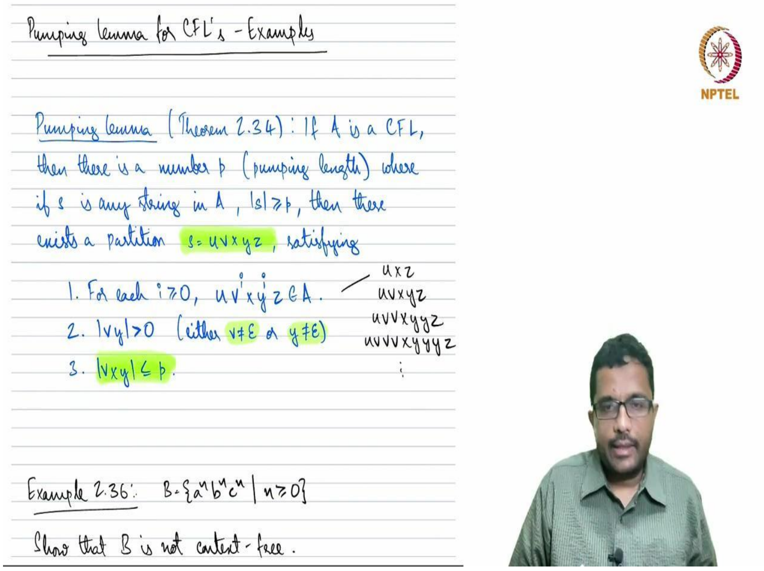
Theory of Computation
Professor. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Examples of Pumping Lemma Usage for Context Free Languages

(Refer Slide Time: 00:17)



Hello and welcome to lecture 26 of the course theory of computation. In lecture 25 we saw pumping lemma for context free languages. In this lecture, we will see some examples for the same one for pumping lemma and another for context free languages. So, just to recap, the statement is that, if A is a context free language then there is a pumping length p such that if you take any string in A which is of length at least the pumping length, we can we can divide the string into 5 parts uvxyz such that the below 3 conditions are satisfied-

First condition is that for each $i \geq 0$, $uv^ixy^iz \in A$. This gives an infinite class of strings including uxz when i is 0 or uvxyz and i is 1 or uvvxyyz and so on.

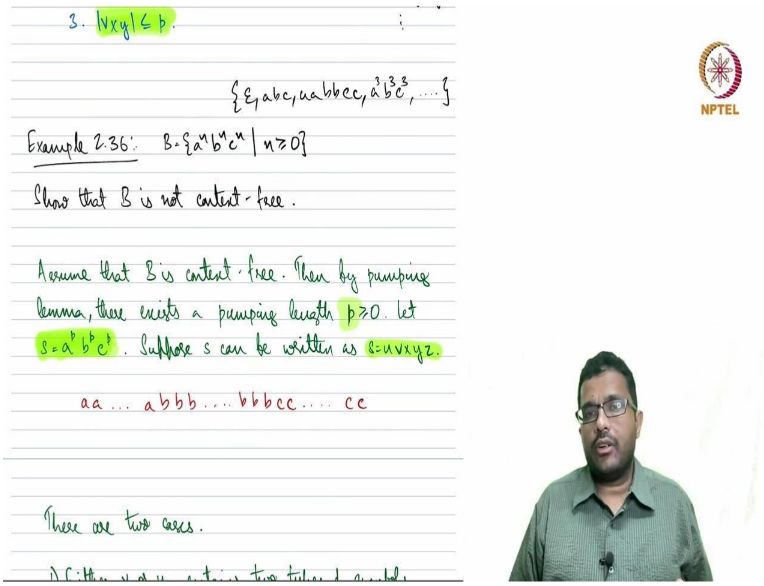
Second condition is that $|vy| > 0$. In other words, this means that either v is non empty or y is non empty.

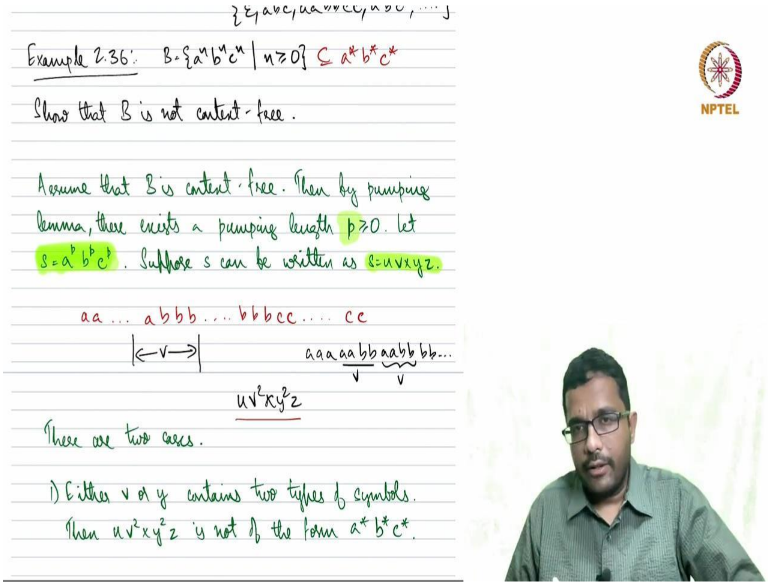
And the final part is that $|vxy| \leq p$

So, as very much like the pumping lemma for regular languages, we cannot use the pumping lemma to show that the language is context free but we can use it to show that a language is not context free. So, the way we will do it is we will assume that the language is context free.

So, then that will give us the fact that there is some pumping length p and we choose a string which cannot be split in a way that satisfies these conditions (usually this is the most creative step). Every time the choice of string is usually interesting. And that will help us show that the language is not context free.

(Refer Slide Time: 03:03)





So, let us just see the 2 examples. The first example is that-

$$B = \{a^n b^n c^n \mid n \geq 0\}$$

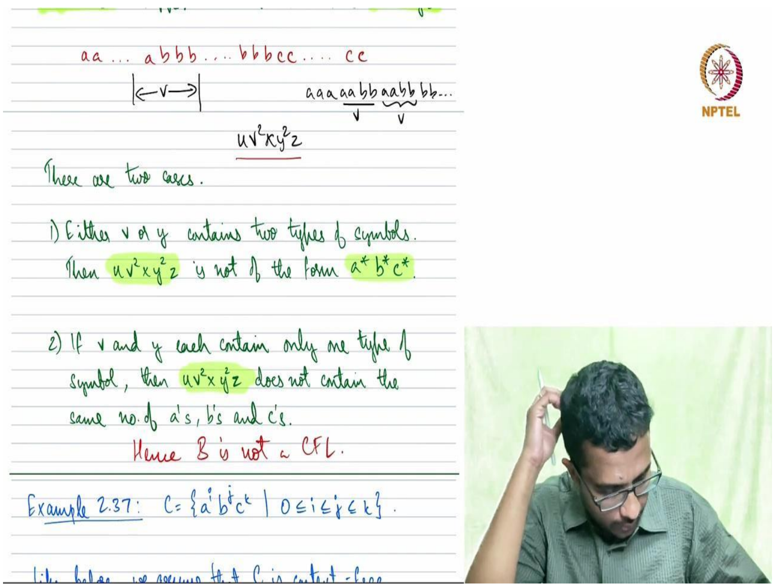
It is an infinite set of languages and we need to show that B is not context free. So, how can we use the pumping lemma to show this? The first step is to assume that B is context free.

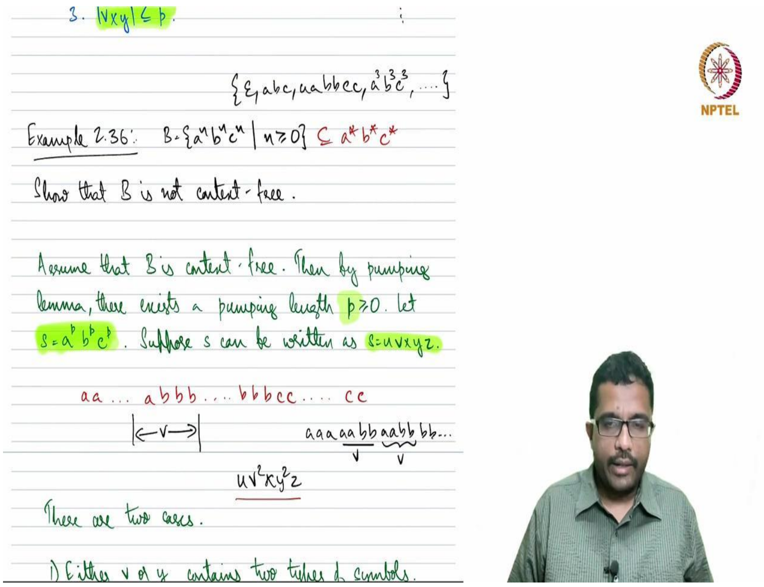
That means that there is a pumping length p as per the pumping lemma with the string $a^p b^p c^p$. We will show that this string cannot be split in the way that we want. We can split the string s as uvxyz and now we will show that whichever ways we choose u, v, x, y and z, it will violate some condition or the other from the pumping lemma.

Suppose v ranges such that it kind of spans both a and b. Now if you consider what will happen when I take uv^2xy^2z, it will have entirely a's and then we have v, which has a's and b's and again it has a's and b's because of v^2 and then xyyz.

Now notice that you have some a's followed by some b's followed by some a's again, then followed by b's. So, now, you see that it is not of the form, some a's followed by some b's followed by some c's. So, notice that the language B is actually a subset of $a^*b^*c^*$ and uv^2xy^2z is not of that form because v is spanning both a's and b's. So, this means that we cannot span both a's and b's. Similarly, we cannot span both b's and c's also because then again, it will not be able to form $a^*b^*c^*$. Similarly, y also cannot span the boundary of a's and b's or the boundary of b's and c's.

(Refer Slide Time: 06:26)





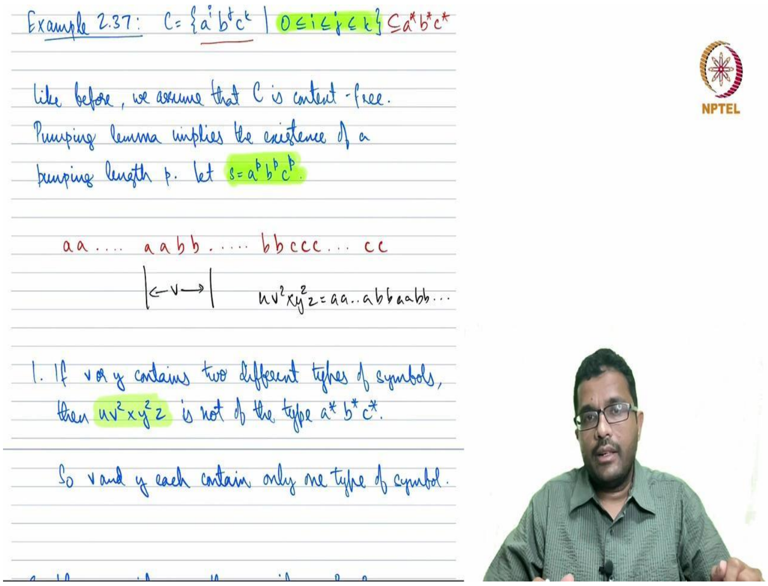
So if v or y contains 2 types of symbols which can happen if they span across the boundary then uv^2xy^2z is not of the form $a^*b^*c^*$. This means all of v and y have only one type of symbol.

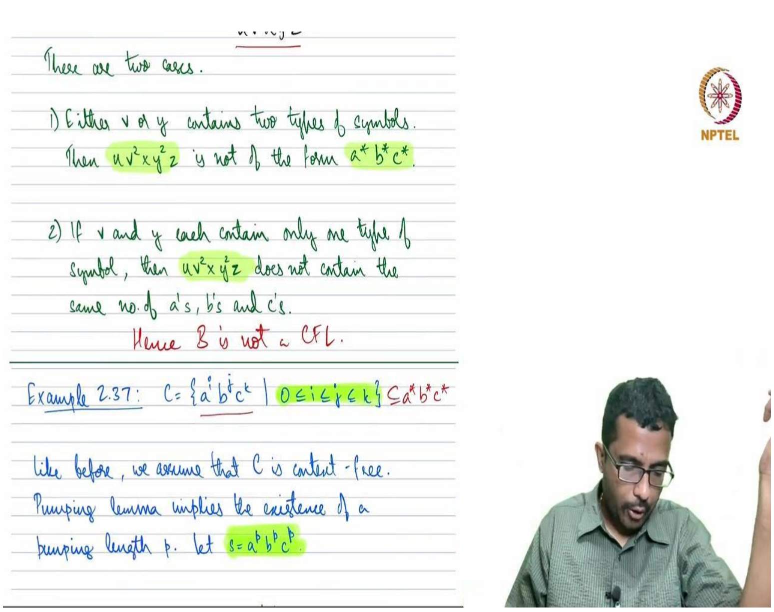
We have 3 symbols a, b and c, v can only contain one type and y can only contain one type. Assume that v contains only b's and y contains only c's, which means a's are not in v or y. Consider the string uv^2xy^2z, this will have more b's than c's but the same number of a's as uvxyz. This is again not part of the language.

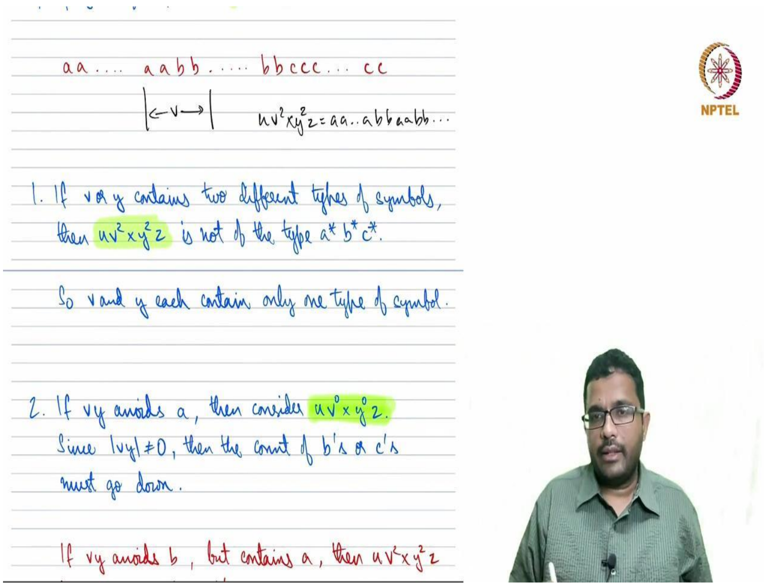
Therefore whichever way you take v and y, it cannot spend all 3 symbols and hence either it loses in structure or the count. So, whichever be the case the resulting string is not in the language B hence the language is not context free.

Small interesting side note is that we saw that a^nb^n was not regular, but we saw that it is context free, but $a^nb^nc^n$ is not context free also. So, that is kind of somewhat amusing. When you want to match to like 2 symbols for equal count. Then you cannot do with regular languages but then you can do with context free languages. But when you have 3 symbols and you want to match them up, even that you cannot do that with a context free language.

(Refer Slide Time: 10:10)







The next language that we want to show is not context free is-

$$C = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\} \subseteq a^* b^* c^*$$

Clearly this is a subset of $a^*b^*c^*$. Earlier we wanted the counts to be equal, here we want the count of a's to be less than or equal number of b's less than or equal number of c's.

So again we assume the same thing that it is context free, which means there is a pumping length p and the string that we consider is $a^p b^p c^p$ which is of the form $a^*b^*c^*$ and the number of a's, the number of b's and the number of c's are equal.

Here equal is allowed, we just want $i \leq j \leq k$. It cannot be that the number of a's cannot be more than the number of b's or number of b's cannot be more than the number of c's.

Once again like before we can reason that v and y cannot contain two different characters as if it would have then it is not of the form $a^*b^*c^*$.

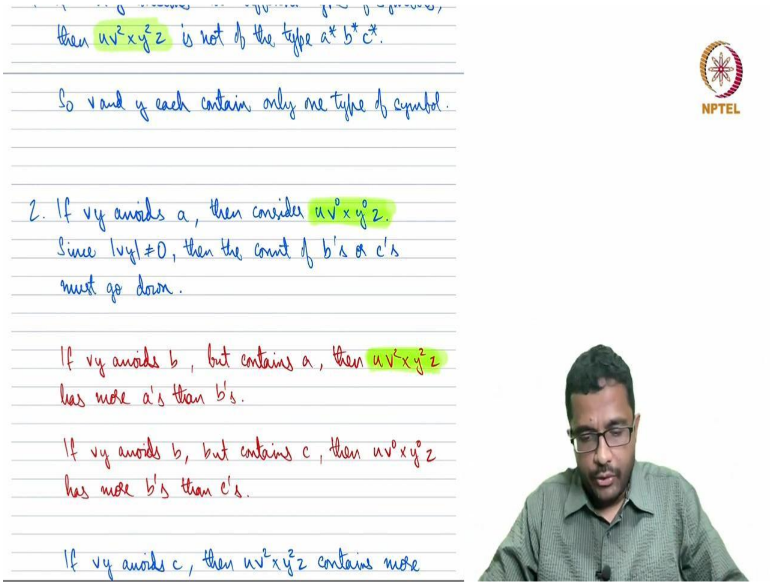
Also note that earlier we wanted all of them to be the same count and then we could find contradictions easily. Here it requires a bit more work, not too much more work but slightly more work.

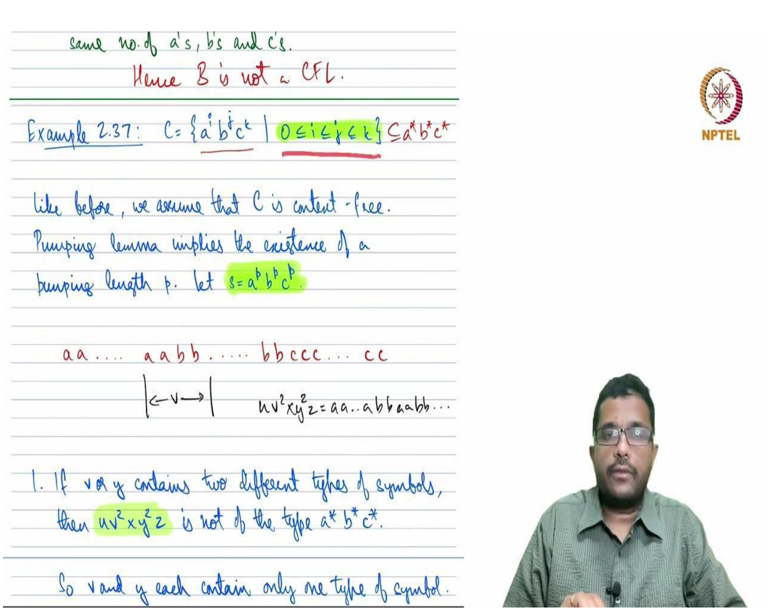
So, let us first assume that, so there are different cases v and y contain only 1 type of symbol v could contain 1 type and y could contain may not be the same type maybe a different type, but then there we have {a, b, c} which means one of these 3 symbols has to be avoided. So, let us say v and y avoids a. So, these are the 3 cases let us say v and y does not have any a in it, which means either v could be in b's and y could be in c's or both of them could be in c's, anything.

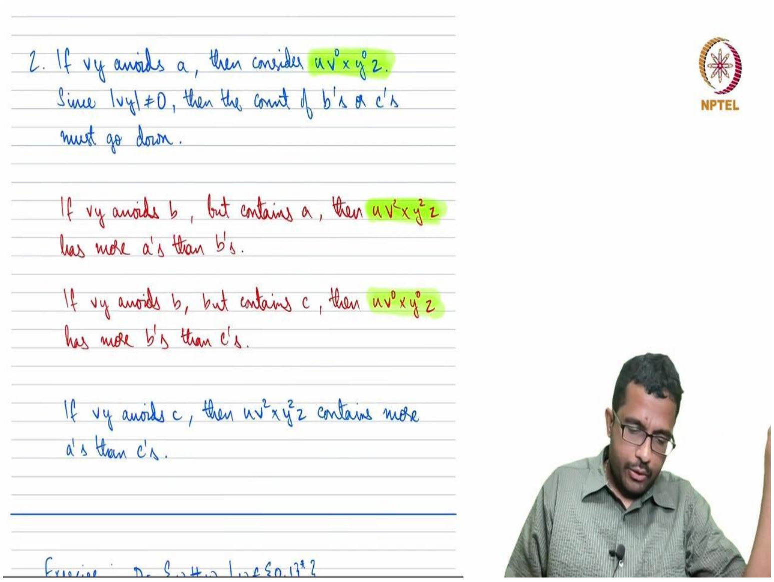
Now, consider the string uv^0xy^0z meaning from the original string s. So s was $a^pb^pc^p$. I am removing a copy of v and removing a copy of y. I know that b y is non empty which means either I am bringing down the number of b’s or the number of c’s in the string because if B or y if they avoid a they must contain b’s or c’s and we know it is not empty.

So, either the number of b’s or number of c’s should come down and in the string s number of a's b’s and c’s are all equal. And now when you bring down the number of b’s or the number of c’s, it makes the number of b’s less than the number of a's or the number of c’s less than the number of a's which means the resulting string is not in the language capital C. Because the number of c’s has to be greater than or equal to number of b’s, which has to be greater than or equal to number of a's. So, if v y avoids a then you pump down meaning you remove b and y.

(Refer Slide Time: 15:10)



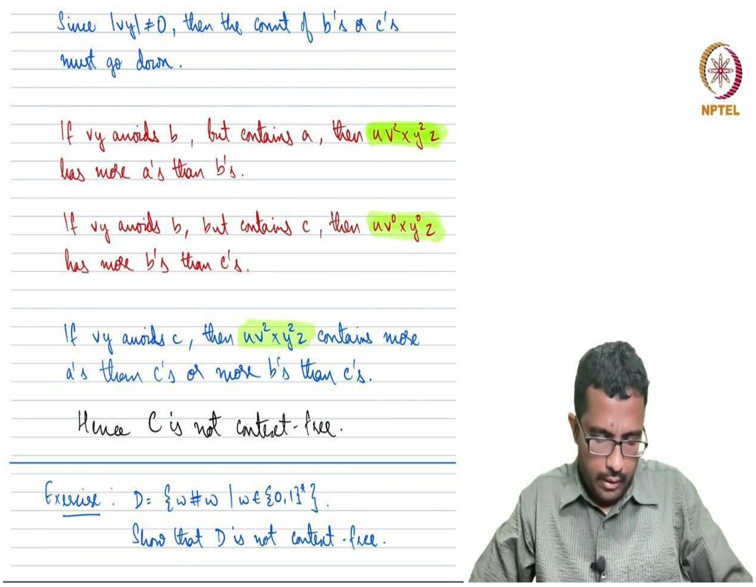




If v y avoids B meaning, v y could be either c’s alone or a’s alone or a and c both, so we will consider both the cases. If v y avoids b, but contains a now consider uv^2xy^2z, by assumption it contains a which means when you pump up the number of a's goes up, number of b’s is unchanged because v and y does not have B. So, number of a's and b's and c's were equal in s number of a's b’s and c’s were equal now number of a's goes up, but b’s remained the same, which is again not as per the rules of the language c.

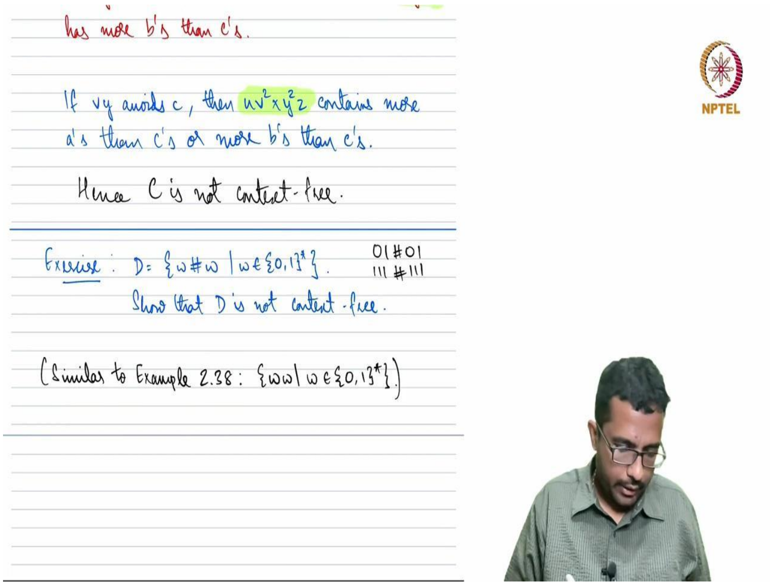
So, if v y avoids b, but contains a then uv^2xy^2z is not in the language. If v y avoids b but contains c, so we do not know about a but it contains c. Then you pump down so, it avoids b but contains c. So, when you remove v and y then there is a reduction in the number of c’s, but the number of b’s remain the same. So, it has more b’s than c’s, which is not allowed as per the language. So, this is an issue when v y avoids b.

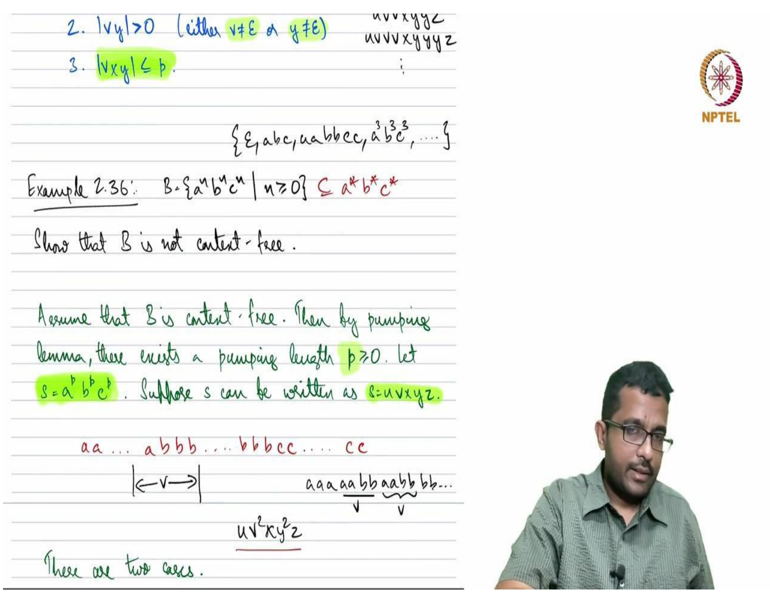
(Refer Slide Time: 16:45)



Now if v y avoids c now you can pump up again which means v and y contains a’s or b’s or both. Now, when you pump up either the number of a's or the number of b’s goes up. So, it did not necessarily contain more a’s, contain more a’s than c’s or more b’s than c’s. So, if v y avoid c’s, avoid c’s, then when you pump up either a’s goes above the number of c’s or b’s goes above the number of c’s. So, whichever one it avoids a b or c then you can accordingly pump up or down and get us something that is not in the language c. So, again, this language C is also not context free.

(Refer Slide Time: 18:17)





We will just state 1 more example which I will not work out here-

$$D = \{w\#w \mid w \in \{0, 1\}^*\}$$

Now show that this language is not context free.

In fact, this is similar to some example in the book, where it is ww without the hash symbol. So, the same string repeats twice and they have worked this out. So, you can go through the example and see how we can maybe modify this proof for the language D. And that is all they have for as far as examples of pumping lemma for context free languages are concerned.

So, we restated the pumping lemma and saw 2 examples in both cases, the template is the same, same as what we saw in regular languages as well. We assume that the language is context free. Then as a result of that assumption, we get that there is a pumping length and using that pumping length b choose a string.

And this string has to be chosen in such a way that whichever way you try to split the string as uvxyz, it will violate some condition or the other and then we saw the 2 examples and that is it as far as examples is concerned and this also completes the part on context free languages.

So, we saw the regular languages in chapter one, this is the end of chapter 2 which is context free languages. We saw context free grammars and PDAs, Chomsky normal form etc. and now we have seen pumping lemma and this completes the pattern context free languages. What comes next is Turing machines and the beginning of computability theory. So, see you in lecture 27 with the beginning of computability theory and Turing machines.