Theory of Computation

Professor Somenath Biswas
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

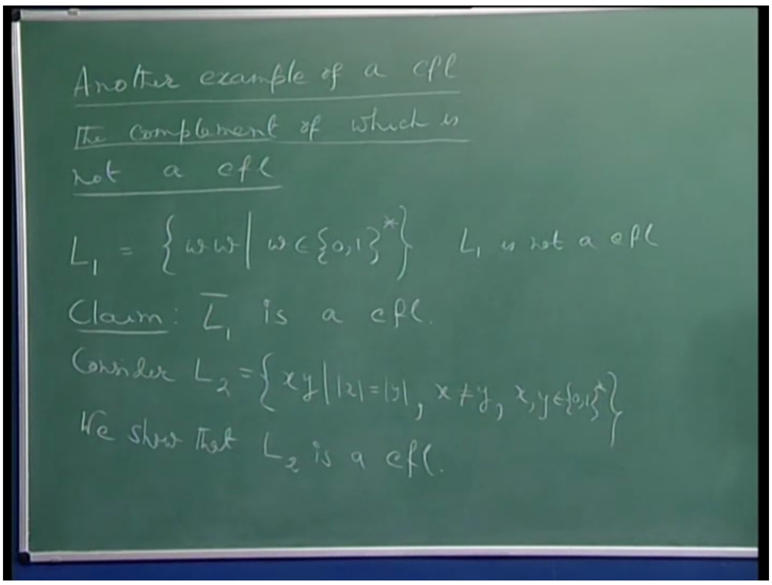
Lecture 29
Another Example of a CFL whose Complement is not a CFL
Decision Problems for CFLS

We had proved that the class of context free languages, this class is not closed under complementation, right? And last time we did provideexample of a language which was not context free. In fact that is the very first language we proved not to be context free. If you recall a n, b n, c n. But we proved that the complement of that language is indeed a CFL.

So you know when we say that this class is not closed under complementation what it means is that in general it is not the case that if you take a context free language its complement is going to be a context free language, it maybe, it may not be. In some cases of course like for example you take a regular language which is of course also a context free language, its complement is also regular language therefore a context free language. So here is an example of a language whose you know the language itself and its complement both are context free.

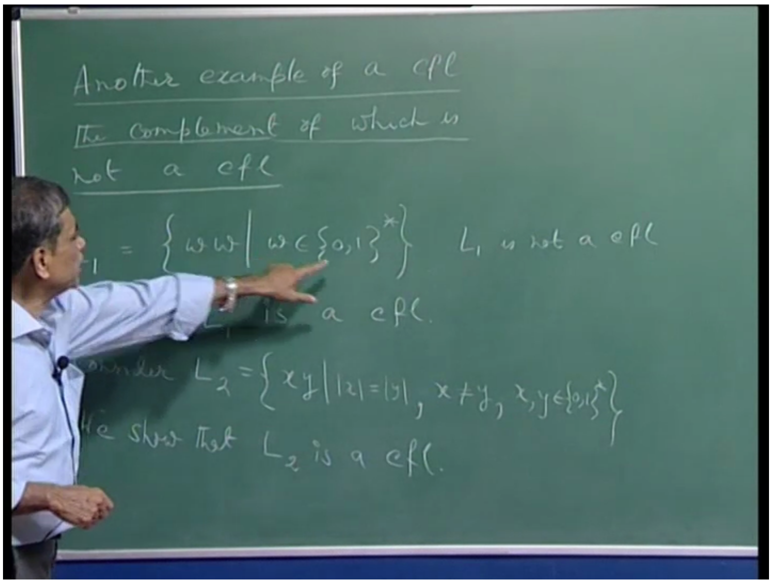
Here are couple of examples which is kind of interesting that on one sidethe maybe the language is context free, its complement is not context free, okay. And of course there will be languageseach of which is neither context free nor you know its complement also context free. So that is also possible. So all these cases are possible.

(Refer Slide Time: 02:06)



And the particular reason why we are giving this another example is because this is an interesting example of showing a kind of you know initially a certain language might appear not to be a context free but on some analysis it turns out to be context free. Sothe example today we are taking up as one such. Alright, this language L 1 which isw w where w is a binary string. This language L 1 is not a CFL.

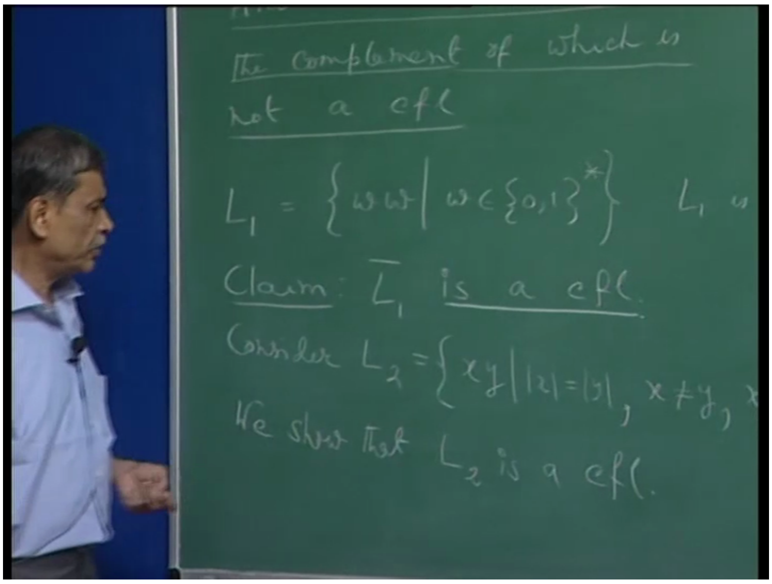
(Refer Slide Time: 02:39)



Nowjust notice, what is this language? It is any string w repeated once more. So takeany binary string w, concatenate that same string with itself, so you get w w. So a repetitionany binary string, this particular language is not a CFL. And we had argued sometime back that why this is not a CFL, right? In fact if you just take w to be a n b n, a n b n, right? So it’s a n b n repeated twice where n is greater than equal to 1.

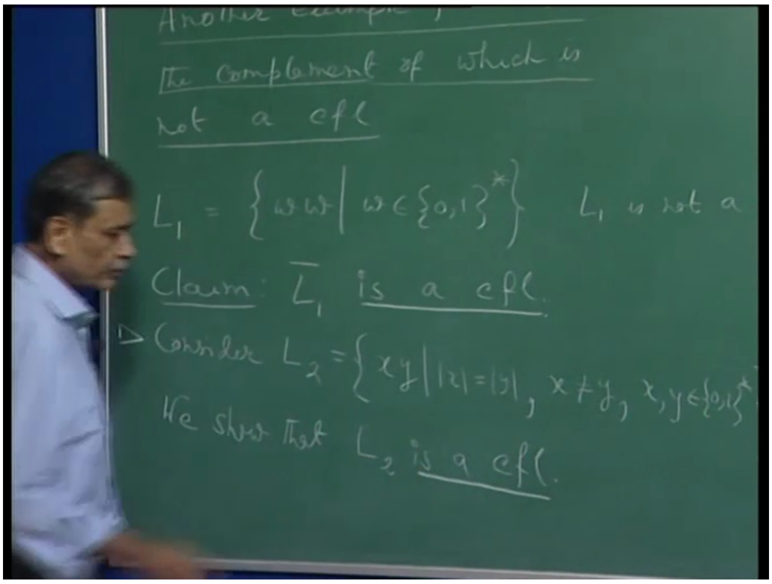
That string you know on pumping will be able to get a string which is not of the form w w. Although that string a n b n a n b n was of this form. Andin that we choose for a particular string forour use of pumping lemma that n is the pumping lemma constantfor that language if it was a CFL, right? So this issomething we have seen. And what we want to claim is that its complement, it is L 1 complement is a CFL. This is what we would like to show.

(Refer Slide Time: 04:32)



And in particular what I would like to do today to begin with is that this particular language. What is this language?L 2 which is concatenation of two strings x and y, both are binary strings and the length of x is same as the length of y. So concatenation of two equal length strings which are not equal, x is not equal to y, this particular language is a CFL.

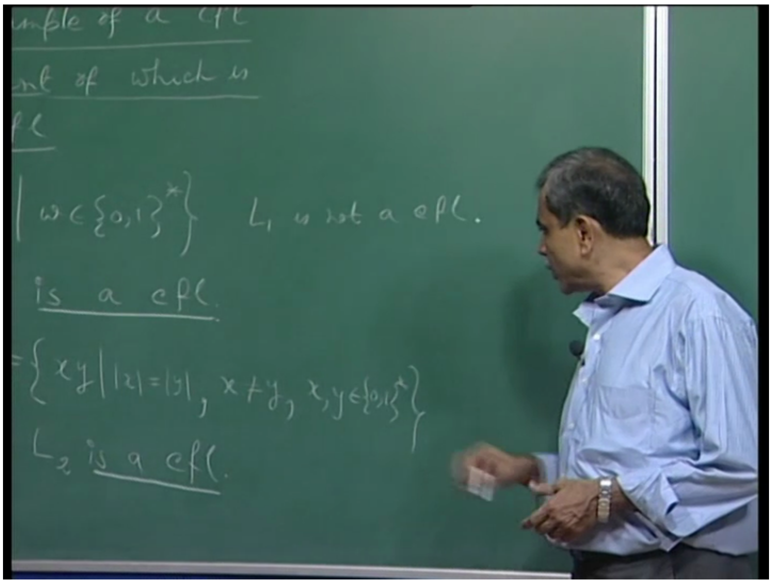
(Refer Slide Time: 05:12)



And this is the hard part or relatively harder part that if you prove this then you should be able to prove that L 1 complement is also a CFL. I will leave that partas something you can do yourself. So once more what I am trying to say is that this particular language L 2 which is the concatenation of two equal length unequal strings, right? The language consisting of such strings that is a CFL.

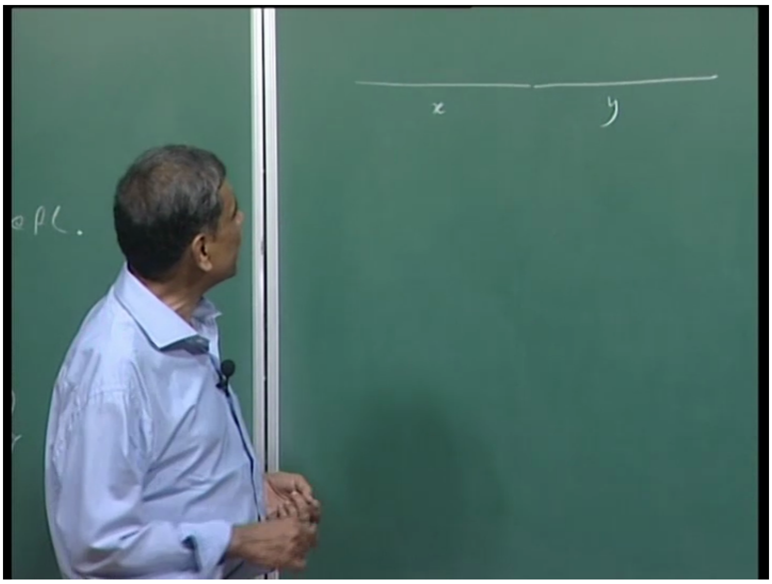
How do we go ahead and prove this to be a CFL? And this is where I think little bit ofvery simple analysis of this condition will help us. Sowhat is it saying?

(Refer Slide Time: 06:10)



So let us think of a string let us say this is x and this is y. Firstly it is saying that the lengths of these two strings are equal. As you can see I mean I am trying to analyzeall these three conditions and so this is y. And the way we have drawn these two strings are of equal length for our analysis that we can see.

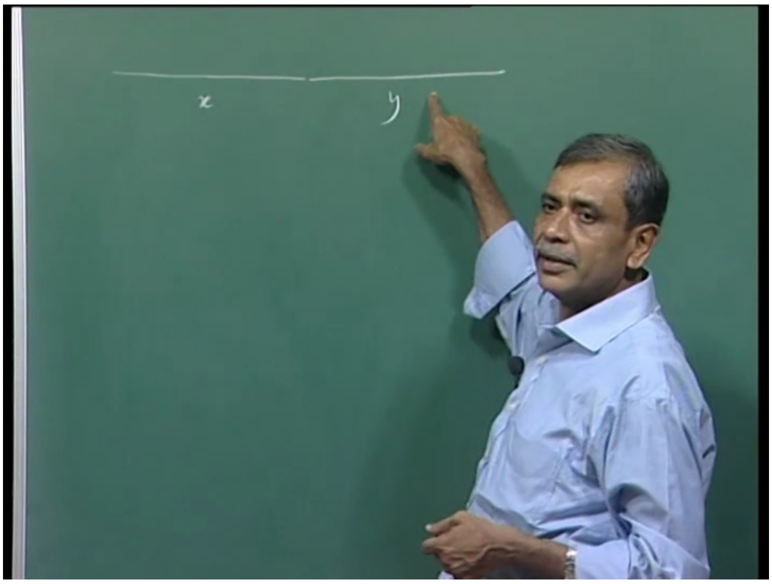
(Refer Slide Time: 06:43)



Now it says they are binary strings which is okay. So x and y both are over this alphabet 0 1. And now what does it mean to say that the string x and y they are not equal. Remember what is our task? Our task is to provide a context free grammar G which will generate all strings of this kind. That is all strings x y where x and y have equal lengths.

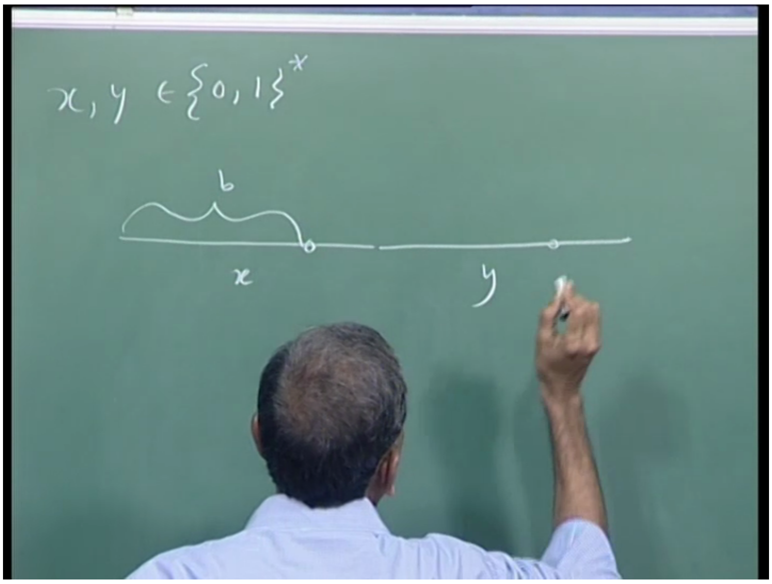
They are binary strings but x is not equal to y. How do we go ahead and fulfill this condition? What does it mean to say the string x is not equal to y?that is the crucial thing. What it means is if you think about it I meanhow do I convince you that this string and this string they are not equal?

(Refer Slide Time: 07:51)



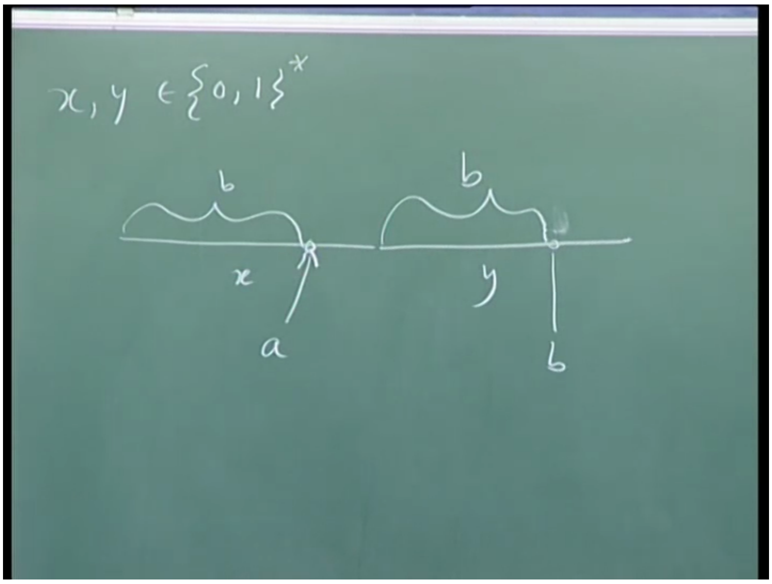
Supposing I have what is the simplest way of convincing you that two equal length strings are not identical. When I say equal that means identical. That means as a string they are not same. Ifwe pause for a minute or even less than a minute what it would seem one way of convincing you, first of all we are assuming that these are two equal length strings, right? Now take some bit here which is in after b bits in this, this bit and the corresponding bit here, okay.

(Refer Slide Time: 08:41)



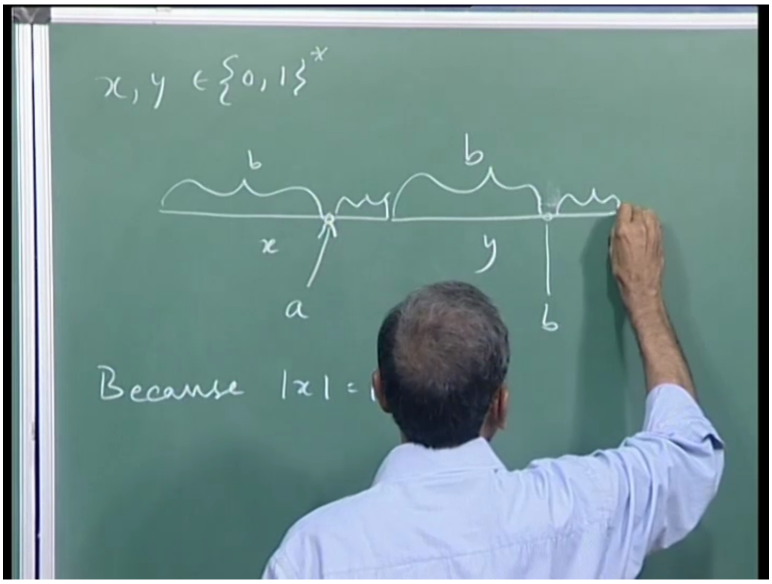
Sowhat it means that the bit which we obtained in this string x after skipping b bits, right? B symbols from this you know starting from the left I move b positions and then the symbol that I get. And here again I do the same thing. This length is b, right? And let me say this symbol isa and this symbol is b. Of course a and b they are either 0 or 1.

(Refer Slide Time: 09:28)



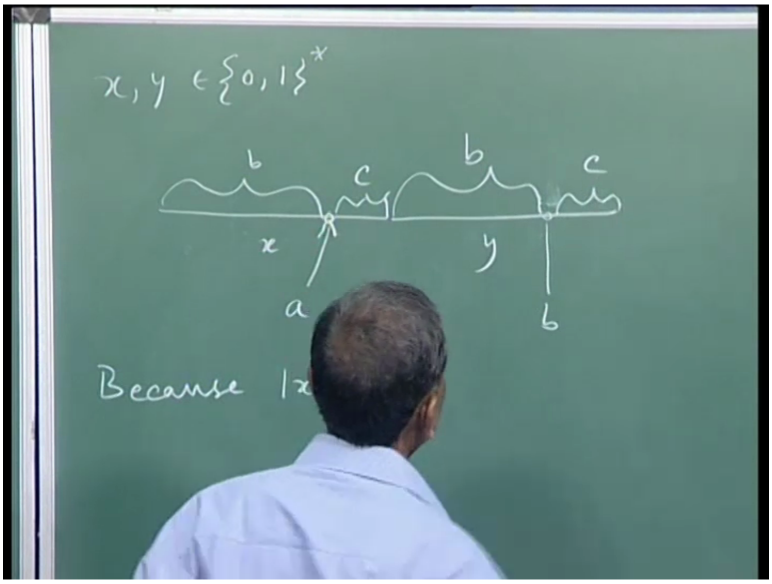
And because the two strings, so let me write it, this is the kind of analysis I am talking of. Because length of x is equal to length of y, right? Then what it means immediately is that therest of the string here, its length is same as the length here, is not it?

(Refer Slide Time: 09:54)



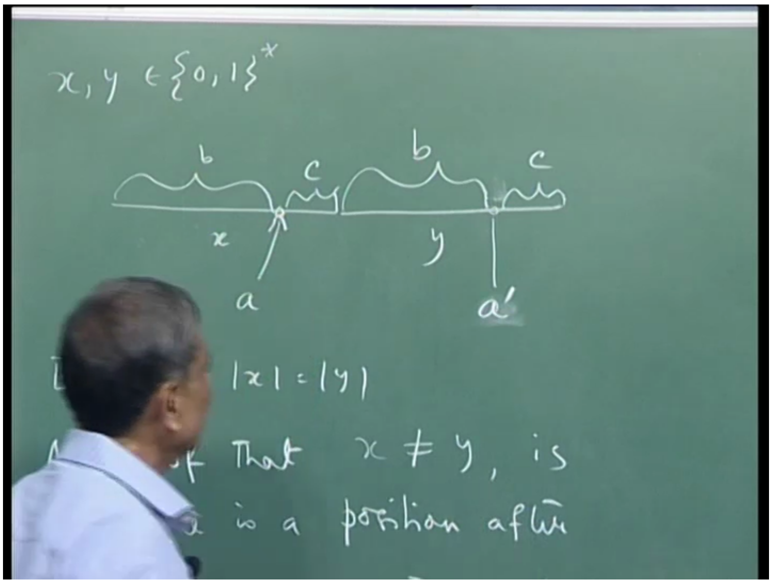
So to begin with the two strings are of equal length and considering the symbol which comes afterb symbols from the left here and theb symbolsfromthe left in the second string and whatever will be left in the first string that length is obviously equal to the length here. So this is c, okay.

(Refer Slide Time: 10:28)



So now I claim a proof that the string x is not equal to or not identical to string y is that there is a position after b bits in x such that the symbol there is differentfrom the corresponding symbol in string y. This is again not difficult to see but all I am saying that your task of providing a proof that x is different from y you know reduces to finding out the position in the string x which comes after you know b bits. Actually this b and this b are different. So let me call it this is a and this is a prime, right?

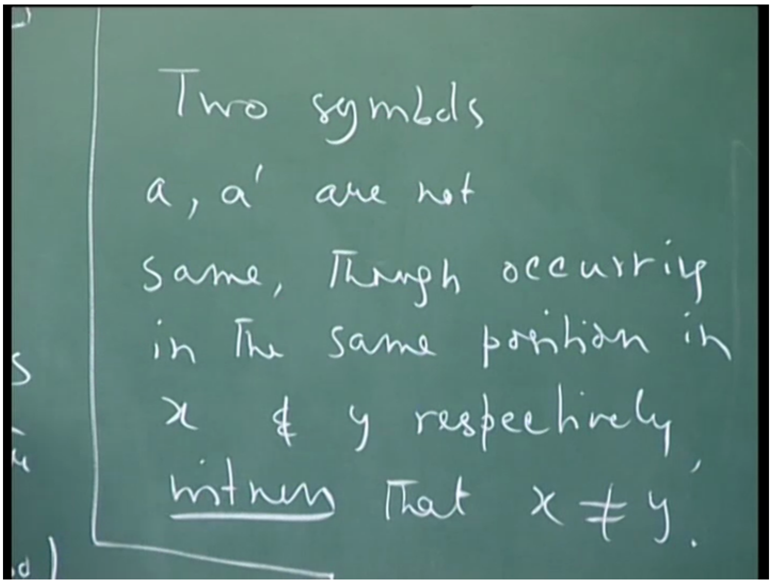
(Refer Slide Time: 12:25)



So a and a prime are two symbols and there the two symbols at a corresponding position. That means you know if it is the kth symbol from the left here, this is of n x, a prime is the kth symbol in yand that is you are counting k from the left of y, alright.

Is this clear? I think it should be you know kind of obvious that if I managed to show or finda, a prime such that the two symbols a, a prime are not same though occurring, I am just saying the same thing in a different way, in the same position in x and y respectively. These two symbols witness that the string x is not identical to the string y, right?

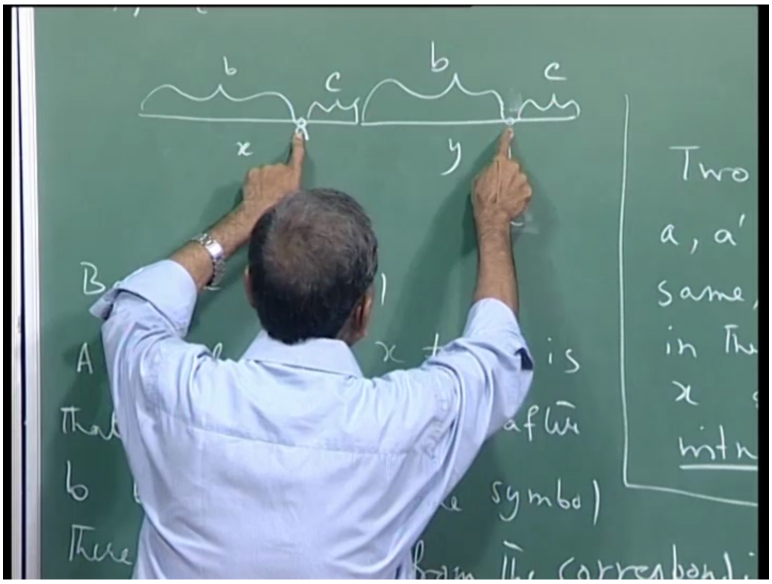
(Refer Slide Time: 13:51)



So this is aboutour analysis. And what then I need to come up with a grammar which will do two things simultaneously. One is that it will generate two equal length strings. That is easy,is not it? Thatif Iensure the grammar generates only strings of even length then clearly whatever it generates it can be seen as two strings concatenated and these two strings are of equal length. So that part is okay.

The harder part is that grammar should also ensure that you know there is some position b in the string x, you know thiscondition that I have written, that the symbol a and a prime are different. And you knowif you stare at the problem what you see is not immediately clear that I can indeed define such acontext free grammar which will ensure thatyou knowat least for one particularcorresponding position here the two symbols are different.

(Refer Slide Time: 15:19)

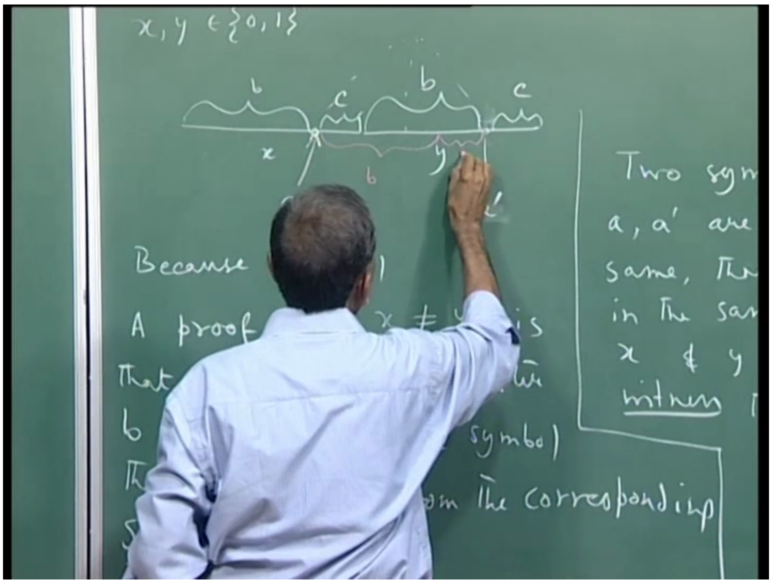


You see the point is this becauseI am trying to ensure something about these two symbols. So you can imagine thatone way of doing that would have been you know generate somethingfrom here and you see that like as in the case of x reverse that the correspondences between these two can be taken care of provided I am generating pairs of symbolsstarting from the middle.

And somewhere I make sure thatthe pair that I generate at some point is different, in the sensethe pair is not 0 0 or 1 1. The pair can be 0 1, 1 0. But then that is easy. However then how do I ensure thatthe other part here you knowtheselength considerations? How do I ensure that after that we need to generate only c bits here, but on the left side we need to generate b bits here?So if you think about it that approach is not going to work.

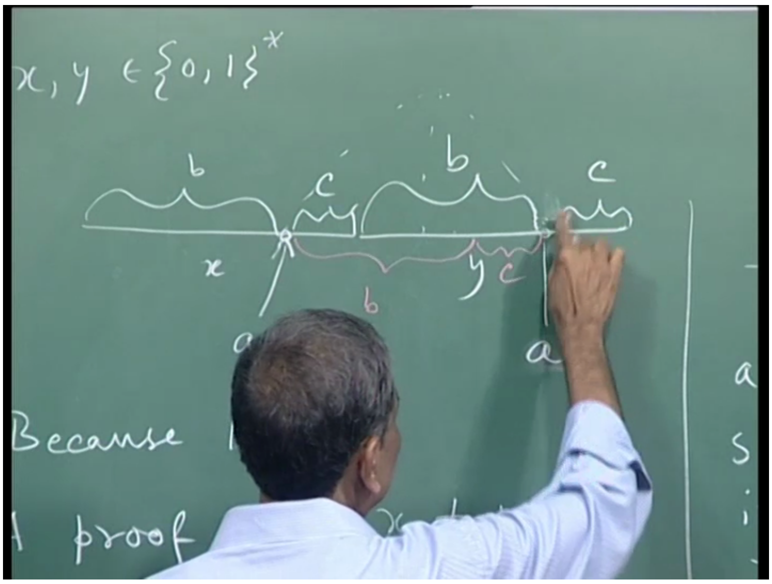
But you know thispicture if I just do something slightly different immediately, the solution will stare in our face. Seethese are two strings. This is of length c, this is of length b, okay. Why do not I do onething? Instead ofthe way I have written let meview this string a little differently. What we are going to do is instead of viewing this part of the string as first a string of length c and then followed by a string of length b, why not let me do this? First a string of length b and then a string of length c, right?

(Refer Slide Time: 17:55)



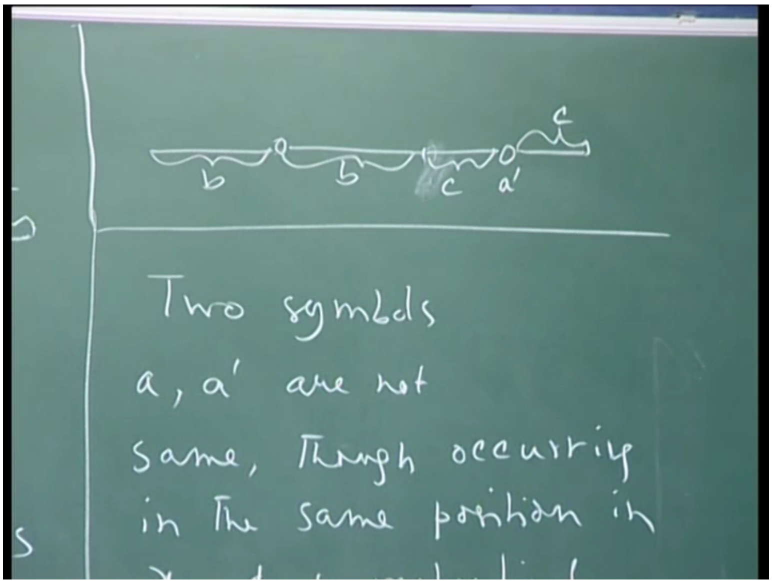
We are just doing some accounting if you like differently. So all I am saying now consider this entire string to be what? That first some string of length b,then a string of length c plus b, then a string of length c. Now I am doing it equivalently as if I am saying that let me count the b part right from here rather than at the end that a length b, then this symbol a followed by a string again of length b followed by a string of length c then the symbol a prime, then this another string of length c, okay.

(Refer Slide Time: 18:49)



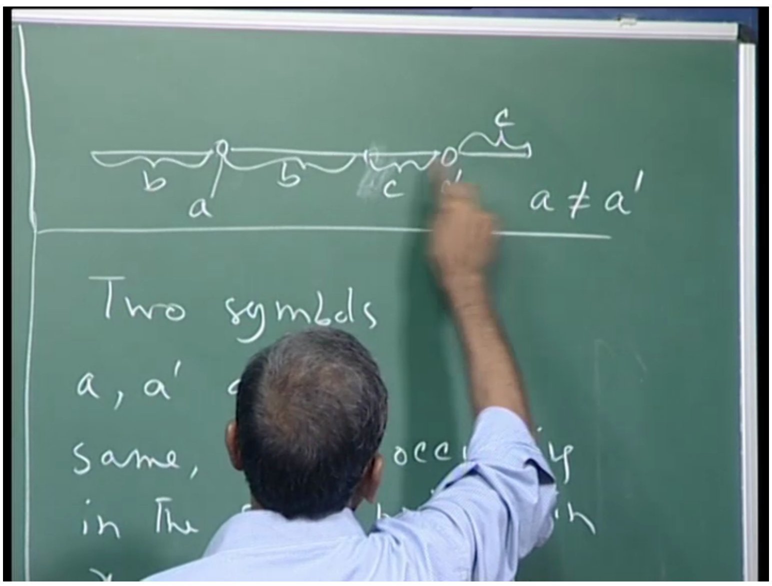
Soyou will first of all agree that I have not changed anything in that string however I am just viewing that string a little bit differently. So let me rub this out, the old way of accounting, okay. Nowthe solution is coming out. See what is it saying is thatwhat I have thisentire string isa string like this. It is some symbol a flanked by two equal length strings b. I am (re) redrawing it.Then a string c of length c followed by the symbol a prime. Again a string of length c.

(Refer Slide Time: 20:12)



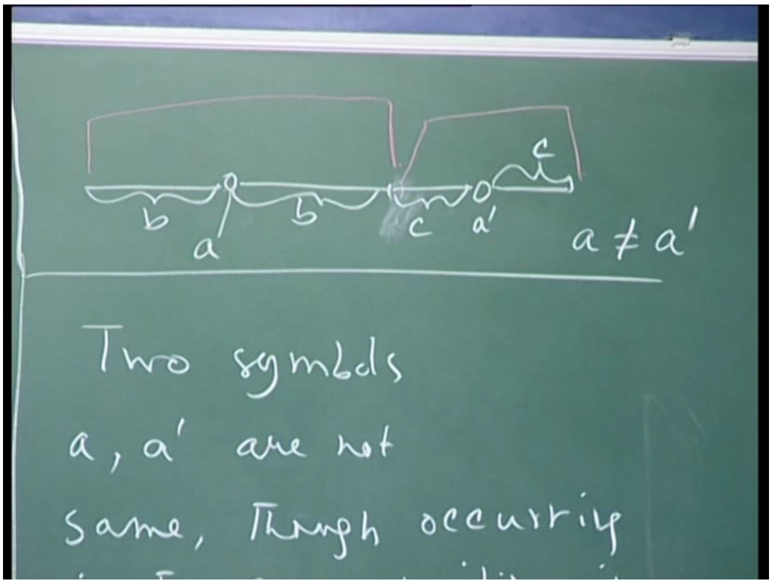
Now is not it this is the same thing I have just named some parts of the strings a little differently. It is the same string, I am viewing it differently. But what is the condition? That this symbol a is different from symbol a prime. So let me write that. Can you see now how this can be donequite easily? So essentially generate some string and odd string. So this is the string of length 2 b plus 1. You know this is one symbol so 2 b plus 1. This is a string of length 2 c plus 1.

(Refer Slide Time: 20:55)



So there are two odd lengthbinary strings and every odd string has a unique center. So the center of this 2 b plus 1 string, center of this string is this symbol and center of the rest of the string is this particular symbol a prime, right?

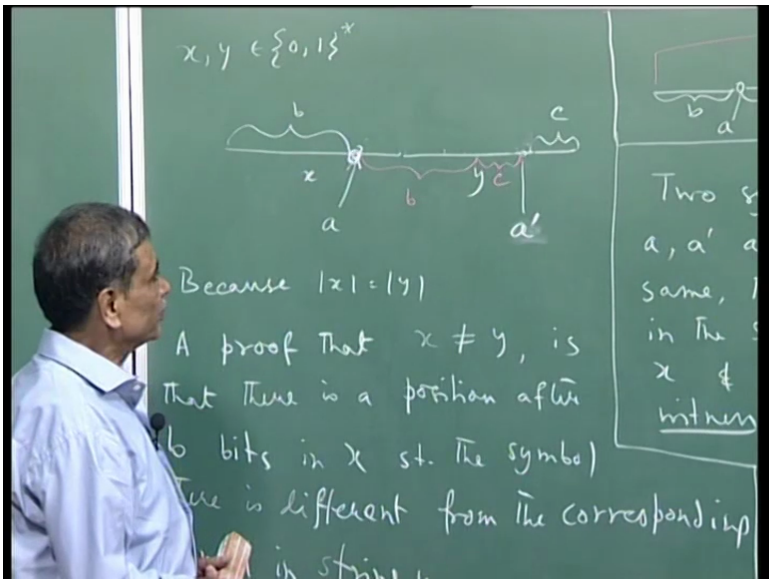
(Refer Slide Time: 21:27)



What we are saying is that the two centers are different. This string solet me now summarize. What I am saying issuch a string, what is such a string? A binary string which is composed of two strings of equal lengths but the two strings are not identical, can be seen equivalently as two strings of odd length. The centers of the two strings being different, alright. Now oncewe understand that then grammar for thatcomes out verytrivially.

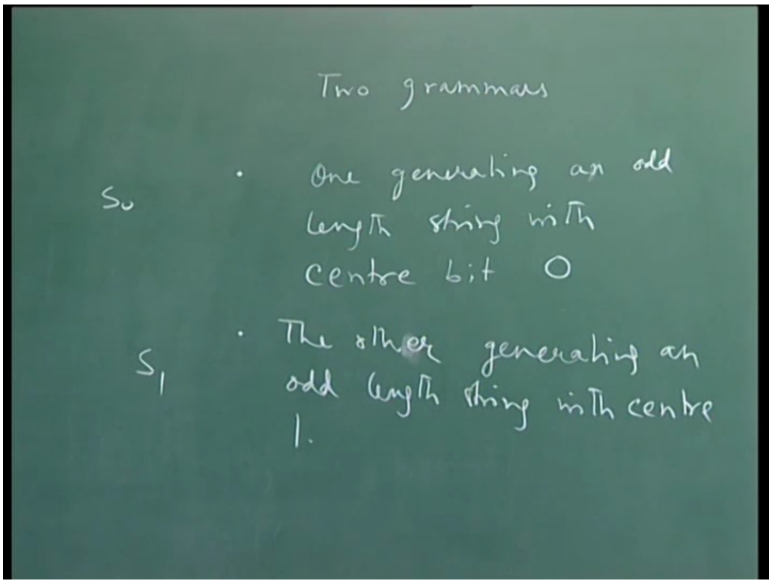
So first of all by the wayis there any restriction about this lengthsb's and c's? Do you see that lengths b's and c's are kind of independent? They have nothing to do with each other, is not it? Because if you take a large string then some bit position is at b here, the same bit position at b in the next string and whatever is left is c.

(Refer Slide Time: 22:51)



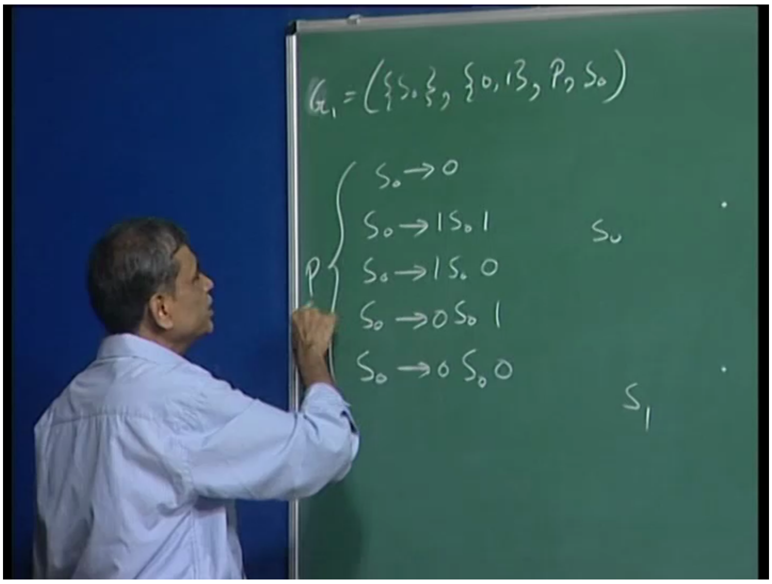
So you know b and c so far as in general the set of all these strings are concerned they are quite independent. And now let us try to giveacontext free grammar which will generate this strings. Let me define two context free grammars. One which generates, let me write this two grammars, okay. One generating an odd length string with center bit 0. And the othergenerating an odd length string with center 1, okay. So let me say at the start symbol for this is 0 and the start symbol for this is S 1.

(Refer Slide Time: 24:36)



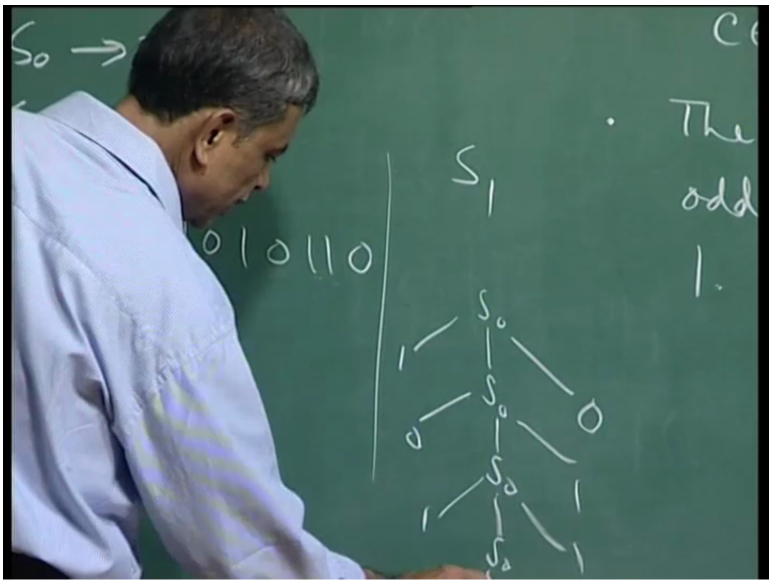
So what are the productions of this first grammar? It is quite easy. S 0 can be 0 or S 0 can be 1 S 0 1, 1 S 0 0, 0 S 0 1, 0 S 0 0. So if you think about this grammar I mean I have given the productions and it has only one nonterminal S 0, okay. So the grammar is G 1 I said. G 1 I am defining it as S 0, 0 1, P S 0. And these are the productionsand this I am calling it P.

(Refer Slide Time: 25:55)



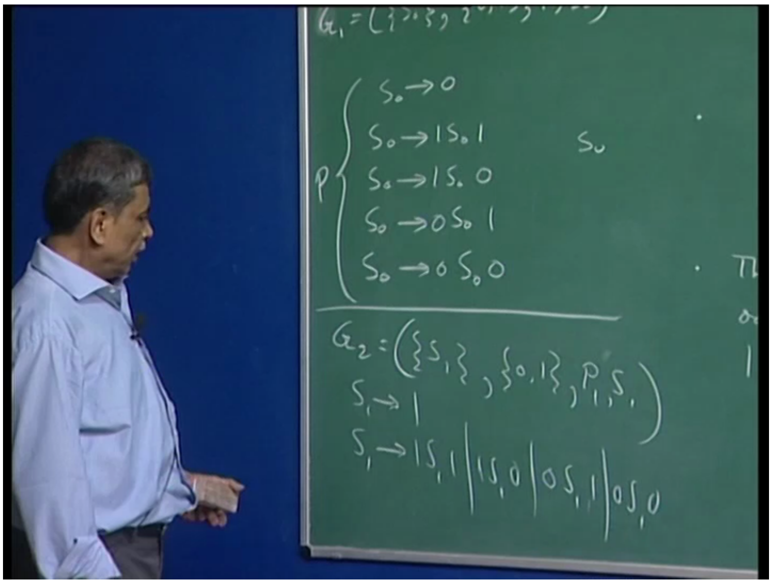
Now it is verysimple to see first of all that it will generate only odd strings, right? Odd length strings and its center will be 0 because why? Seeyou know think of let me just give an example. So let me say 1 0 1 0 1 1 0, this is an odd length string with center 0.Center symbol is 0. So thathow are we generating in this? Now start with S 0 and 1, S 0, 0. That takes care of the two outermost thing. Then 0, 1, S 0. So these two are gone. Then 1, 1, S 0 and finally 0.

(Refer Slide Time: 27:00)



So the final S 0which is replaced that gives the center, right? And that is always 0. In the same way we can define another grammar. The grammar for this is going to bevery similar, obviously identical almost. S 1, 0 1, P 1 let me say and S 1. And S 1 is 1 and S 1 is, you know other thing it is similar,S 1 1, 1 S 1 0, 0 S 1 1, 0 S 1 0, okay.

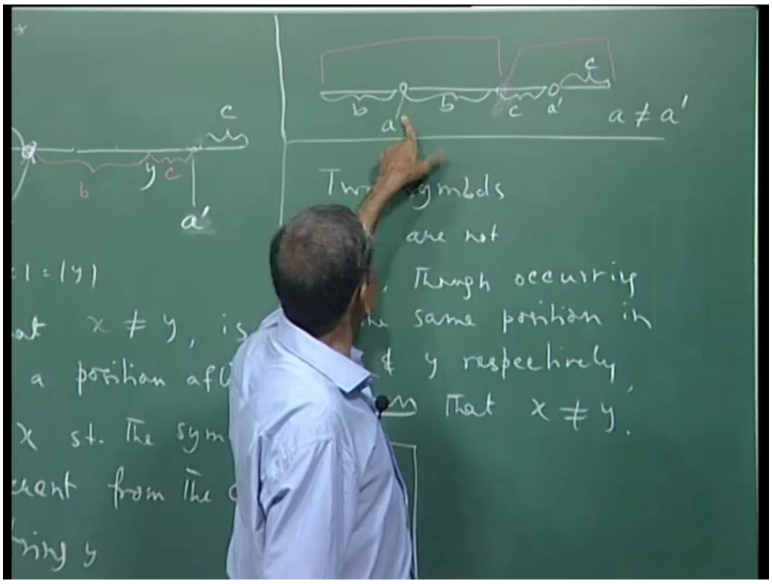
(Refer Slide Time: 28:11)



So G 2 will generate odd length strings with center 1 for the same reason as we have given. And now imagine thatI combine these two grammars to have one single grammar,right? So what I am going to do I haveall these productions at the same place, okay right. Andsee, what isour goal? Our goal is to generate an odd length string with some center bit and then generate another odd length string whose center bit is different.

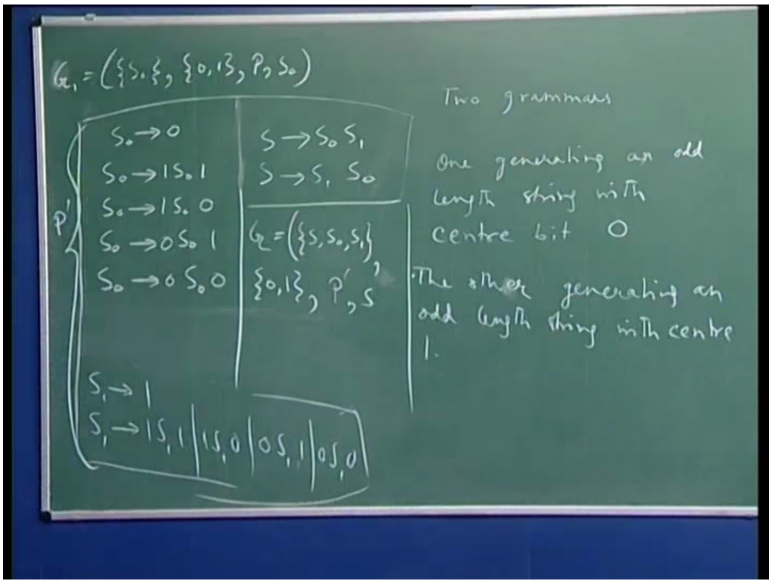
So do you see this is going to work that my new start symbol is S. S I will sayeither you first generate a string which can be derived from S 0 followed by aodd length string with center 1. So that will mean that somewhere this a is zero, this a prime is 1.

(Refer Slide Time: 29:33)



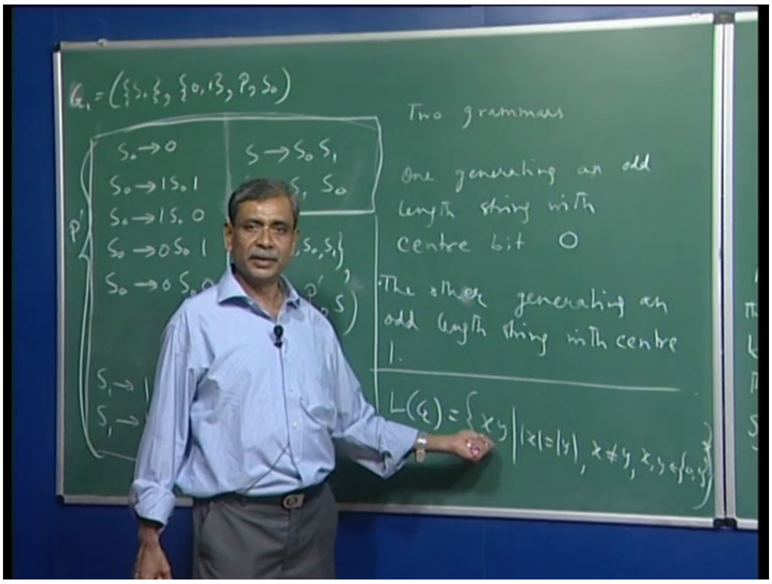
Or it can be, right? And then my grammartherefore will now have threenonterminals. So the actual G that we derived is S, S 0, S 1. This is the set of nonterminals and followed by set of terminals is of course 0, 1 and then I have this new set of productionsP prime which is all this followed by the start symbol S. And remember P prime has this as well as all this. This is your P prime, alright? And this is the grammar.

(Refer Slide Time: 30:39)



Now I claim L Gis, what the grammar L 2 that we wrote, that set of all strings x y such that length of x is equal to length of y, right?X is not equal to y and x y are both binary strings. Language generated by this grammar is indeed that language that we had earlier called L 2 and by analysis or analysis and our way of viewing it we got fairly simple grammar actually, is not it, to generate this language which is concatenation of two equal length strings which are different.

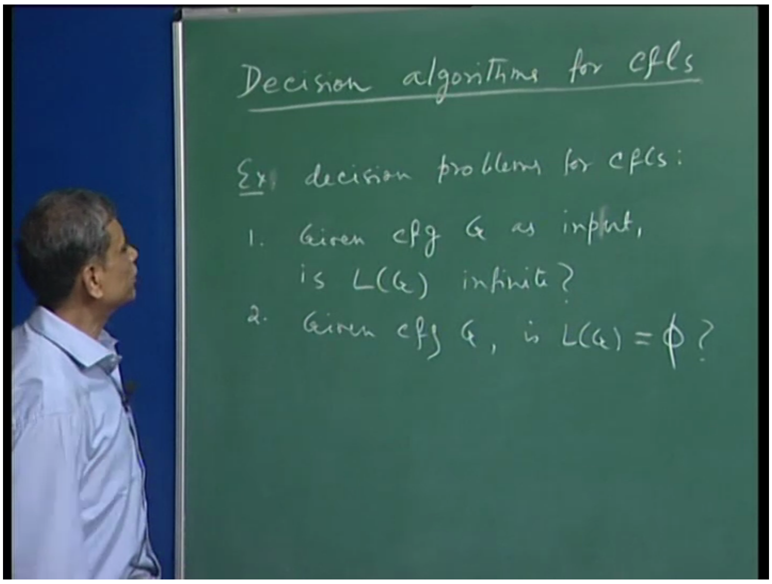
(Refer Slide Time: 32:02)



And since I could generate this language using a context free grammar this language is a context free language. So now we will consideranother sub topic of context free languages is about their decision algorithms. Now what isa decision algorithm? A decision algorithm is an algorithm which solves a decision problem and in turn what is a decision problem? A decision problem is a problem which will have yes no answer.

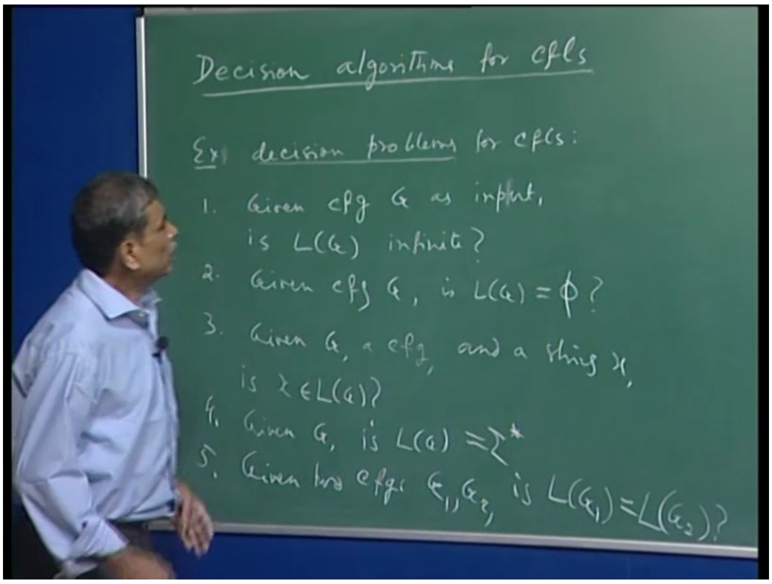
You need to decide given the input you need to decide yes or no. For example in case of context free grammars that your input is usually you will be grammar or a pair of grammars or a grammar and a string. Soexample decision problems for CFLS. So let us say one is given CFG G as input is the language generated by G infinite. So if you say yes that means L G is infinite. If you say no, L G is finite. Similarly given CFG G is L G empty, right?

(Refer Slide Time: 34:11)



So this is another decision problem. There can be more. Given G, a CFGand a string x is x in the language generated by the grammar G. And of course let me take another or two others. Given G is L G is equal to sigma star where the grammar G, the input alphabet is sigma. Another given two CFGS G 1, G 2 is L G 1 equal to L G 2.

(Refer Slide Time: 35:34)

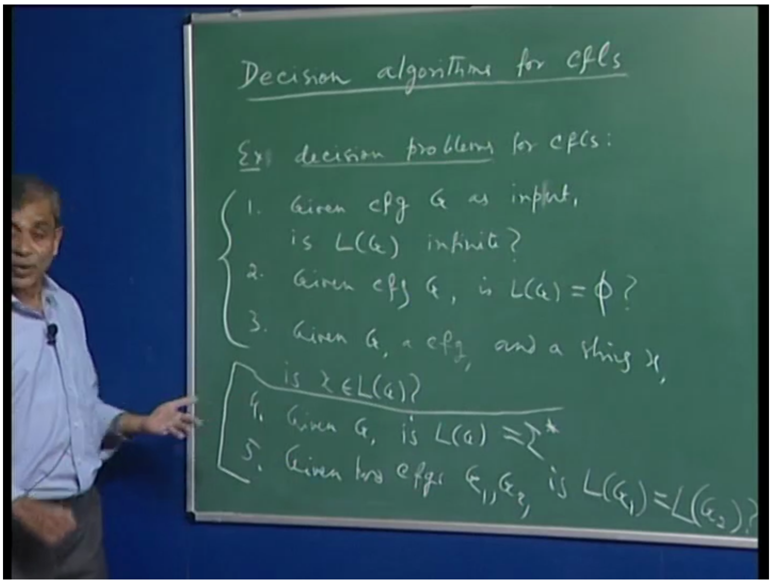


Now for a momentinstead of CFG, think of a regular languages. Inother words instead of context free languagesof course we could have posed the same problems for the class of regular languages. So there for example something a problem like this will be given DFA M as input is the language accepted by M infinite. Again given a DFA M is the language accepted by M empty and so on.

And sofor example last two would be that given twoDFASM 1, M2 is the language accepted by M 1 same as the language accepted by M 2. Now if you go back and remember what we did for regular languages,for all this decision problemsfor each of them we could give an algorithm.

So in fact almost everything thatwe could think of as decision problem for regular languages we could give an algorithm to solve that decision problem. In case of thisclass, context free languages the situation is verydifferent. In fact we will indeed be able to show or give algorithms for these decision problems and for these last two one would be able to prove later on.

(Refer Slide Time: 37:40)

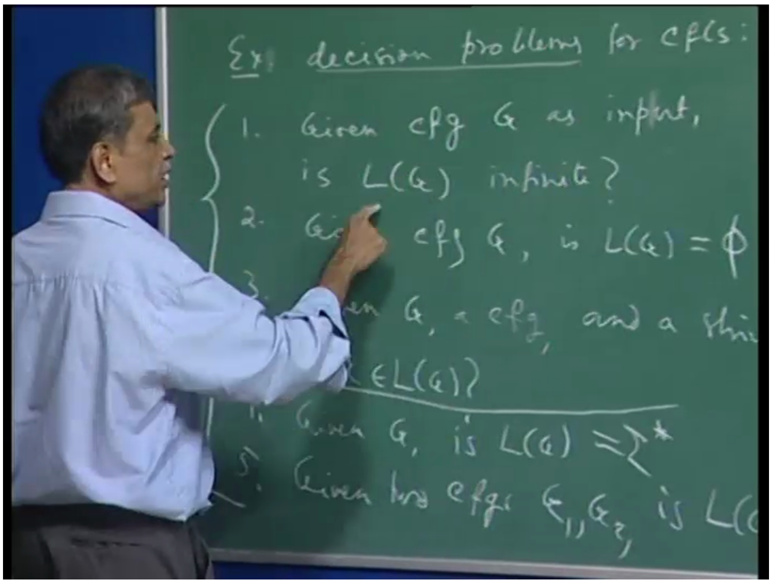


If youdo a little more of this course thenthere are no algorithms to solve these decisions problems. So you see what I am trying to say is whereas I will be able to provide an algorithm to solve let us say any one of these first three decision problems when it concerns context free languages. For the other two, for example here what you are (diff) asking a question is it seems very innocuous that here is this grammarwhich generate strings over an alphabet sigma, does it generate every string of the alphabet sigma?

And interestingly there is no algorithm. No algorithm at all for solving this decision problem. Similarly giving two context free grammars, do both of them generate the same language? Again there is no algorithm to decide this question. And such decision problems are called undecidable problems which when we do during machine will be able to go deeper into the class of undecidable problems.

So right now let us provide the positive results. That is all we can do. Let me provide algorithms for 1 2 and 3. So the first problem isthat given a grammar CFG G is L G infinite.

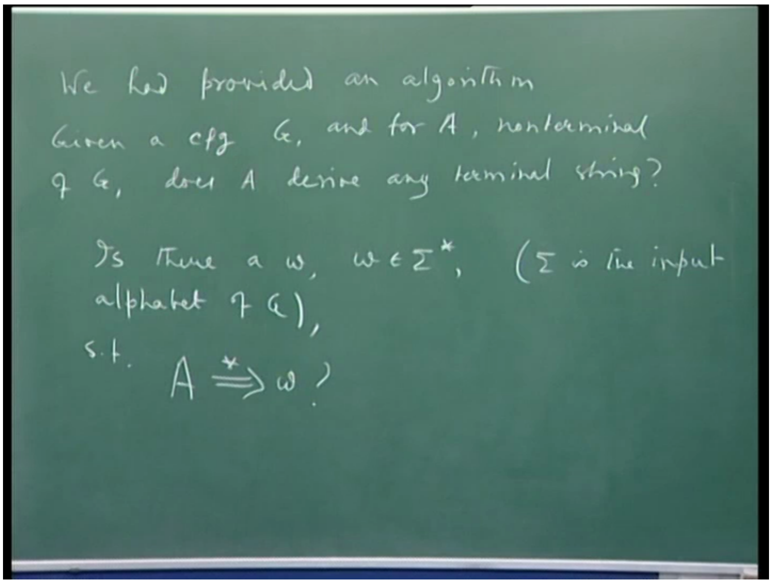
(Refer Slide Time: 39:45)



Actually thesimplerthing let me start with which is the second problem given a CFG G is L G empty. Nowone algorithm follows the proof of that proof that we had seen of pumping lemma for both 1 and 2. But let me give you a more direct algorithm for G which actually we have done. So you recall we had provided an algorithm for the following problem that given a CFG Gand for A which is a nonterminal of G, does A derive any terminal string?

And remember that this is one way of supposing in other words actually I should have saidwe can write this or we can symbolize thisway that is there a w, w is an element of sigma star, sigma is the input alphabet of G. So is there a w, w is in sigma star such that this nonterminal A derives the string w, right?

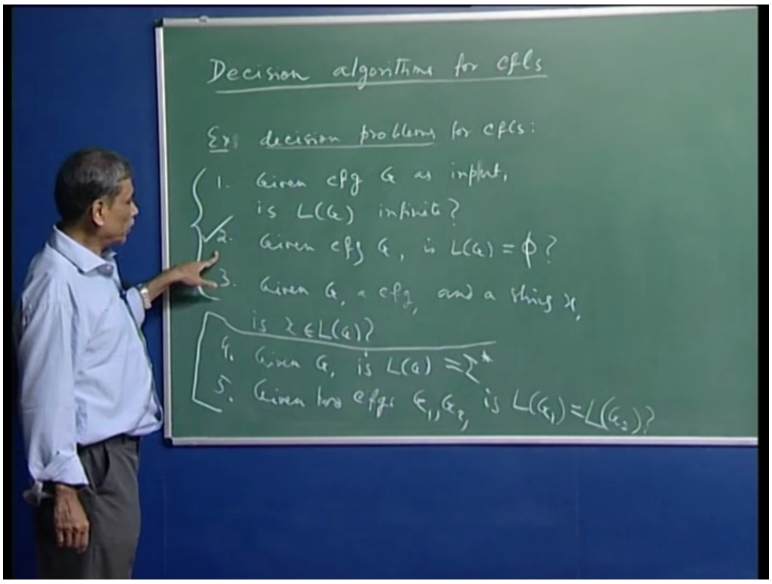
(Refer Slide Time: 42:49)



If you go back we will be able to remember this algorithm if you do not remember right away. The idea was that we kind of iteratively figure out all those nonterminals which generate terminal strings and at some point of time we will be able to discover all those nonterminals which generate terminal strings and then if this nonterminalA happens to be in that set of nonterminals which generate terminal strings then I can say that this A does derive a terminal string. That was the idea.

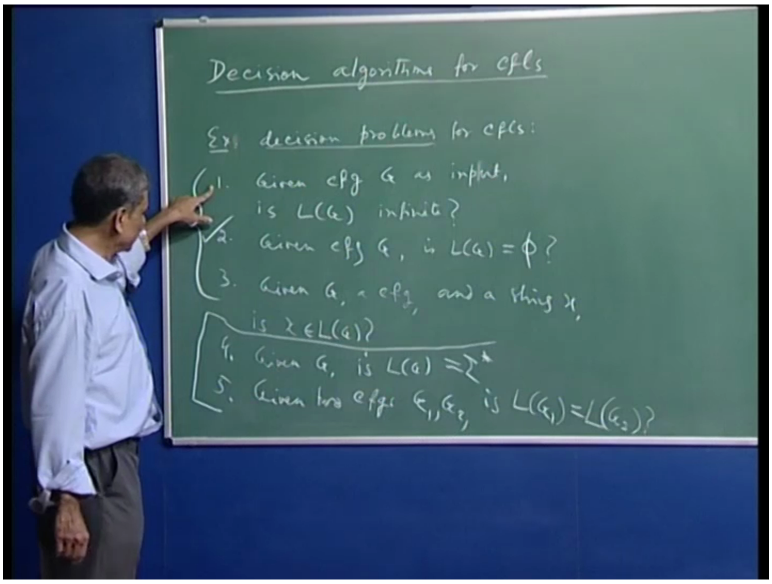
Now you see for a grammar G to be empty that means what? S does not derive anyterminal string at all, right? So you know I can just run through the same algorithm for the input grammar G and if I find that the start symbol S does not derive any terminal string then clearly the grammar generates the empty language. So thatgives us the algorithm to solve this decision problem.

(Refer Slide Time: 44:26)



The other two we will take up in the next lecture and this one also will be quite simple. In fact we have done most of the work for this.

(Refer Slide Time: 44:40)



This is a very interesting algorithm which we will provide. So in the next class we will continue with this topic decision algorithms for CFLS. In particular we will provide decision algorithms for problems 1 and 3.