Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 15
Introduction to CFGs

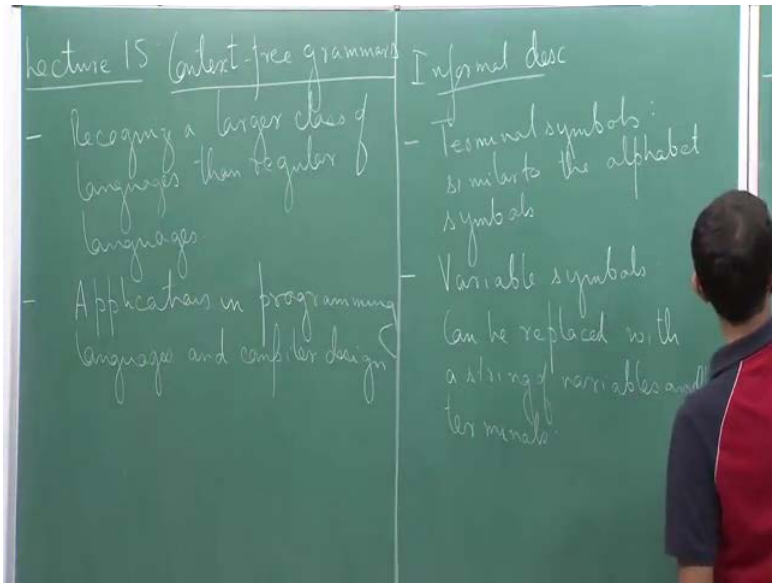
Welcome to the 15th lecture of this course. Today, we are going to talk about a new model of computation namely Context Free Grammars.

So far in this course, we have seen regular languages, and we have seen some models of computation that accept the class of regular languages such as deterministic finite automaton, non-deterministic finite automaton and regular expressions. And thereafter, we saw that there are languages which are not regular; in other words, they cannot be accepted by a finite automaton, and we have seen a proof of that using the pumping lemma.

So, what we will show today is we will talk about this new model context free grammars and we will show that there are non-regular languages, which are accepted by context free grammars. So, there are languages which are not accepted by finite automaton, but context free grammar is able to accept them.

And we will also show that later on in the possibly in the next lecture that the class of all languages that are accepted by context free grammars form a super class of all regular languages. So, every language, so every regular language is also accepted by context free grammar. So, this makes the class of languages accepted by context free grammars, a proper super set of regular languages.

(Refer Slide Time: 02:00)



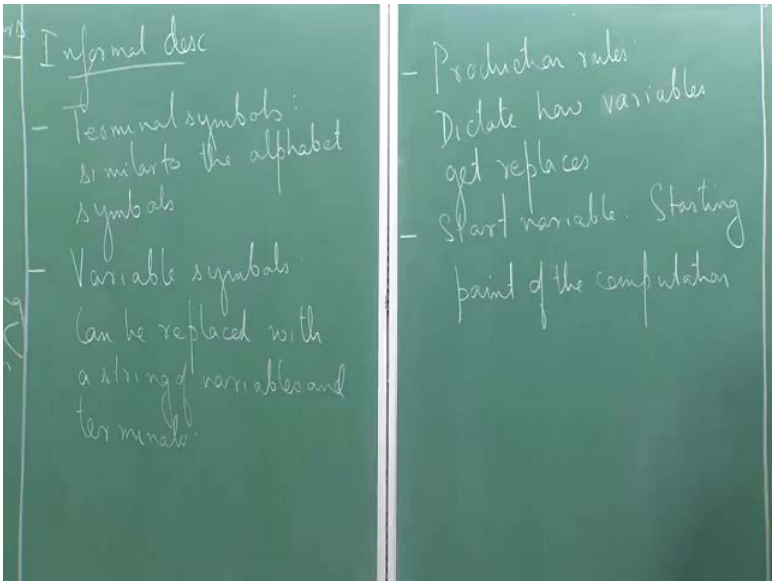
As I said let me briefly summarize the context free grammars recognize a larger class of languages than regular languages. And as we will see, so here the model of computation that we have is not machine model, it is not automaton model, it is what is called a grammar and we will see how a grammar is defined, and how a grammar is able to accept a string. Deviating slightly a context free these objects context free grammars, so they also have applications in programming languages and compiler design. So, these are some practical applications of context free grammars; we will not get into this aspect of context free grammars. Our focus will be to understand the class of languages that are accepted by context free grammars and study their properties.

Informally, a context free grammar let me talk about let me given an informal description. It consists of four components. So, we have what are known as terminal symbols. These are similar to the alphabet symbols that we considered when we talked about finite automaton. In fact, the set of terminals symbols, you can of it as the alphabet of the grammar then there are these symbols known as variable symbols. So, a context free grammar has these two types of symbols; one are the terminals symbols which are similar to the alphabet that we have seen so far, and the second type of symbols are known as the variable symbol.

The variable symbols so they the reason they are known as variables is because they can be replaced with a string of variables and terminals. So, during the computation, you can

take a string and you can replace that string, you can take a string you pick a variable from that string and you can replace that variable with a string of some other string of variables and terminals. So, this is how computation proceeds.

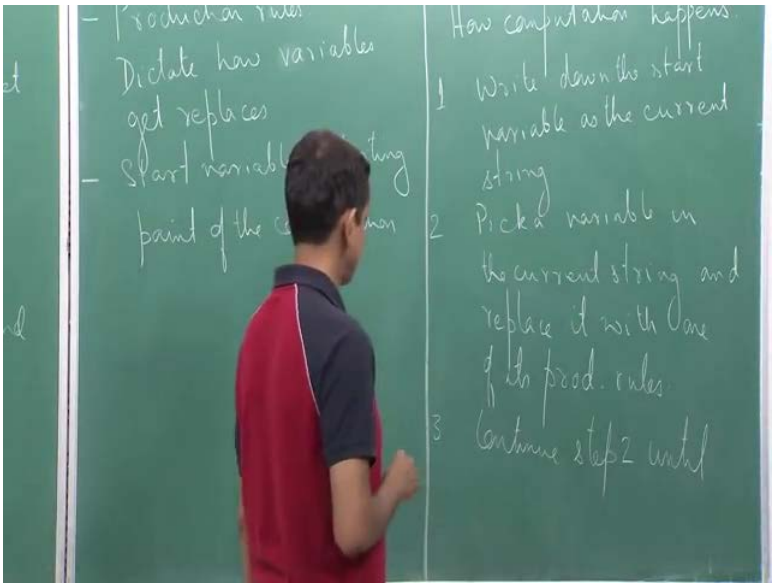
(Refer Slide Time: 06:48)



So, the way this replacement occurs is by the help of production rules. So, these rules, so they dictate how variables get replaces, so this is essentially the rules, which tell us how a variable get replaced with the string of variables and terminals. And finally, we have the start variable.

Let us write the starting point of the computation. So, in each step of the computation, variable get replaced with a string of variables and terminals, but where do we start, we need to start at some variable, so that is what is the start variable, so that function is performed by the start variable. So, it is similar to the start state of an automaton.

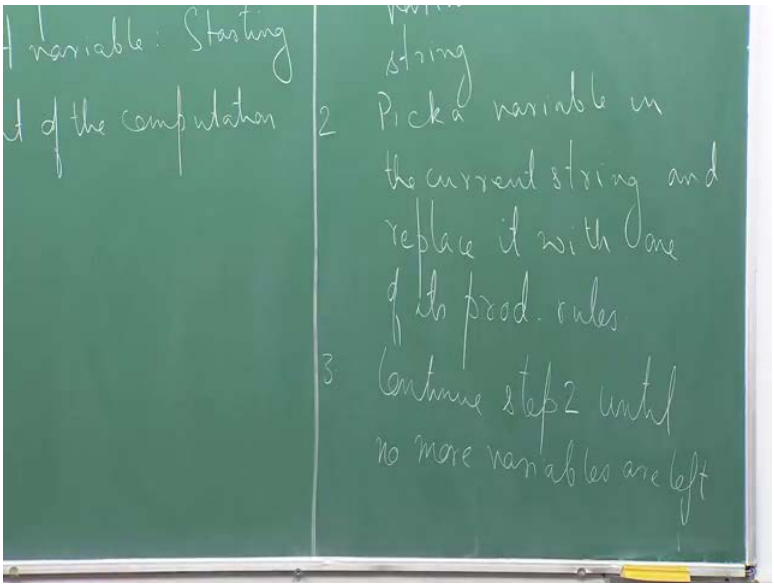
(Refer Slide Time: 08:36)



Now how does computation proceeds. This is a description of a context free grammar, but how does computation proceeds in context free grammar? So, first how computation happens, so first you write down the start variable as the current string then pick a variable in the current string and replace it with one of its production rules. So, note that corresponding to one variable, we can have multiple production rules it is not necessary that every variable as a single production rule.

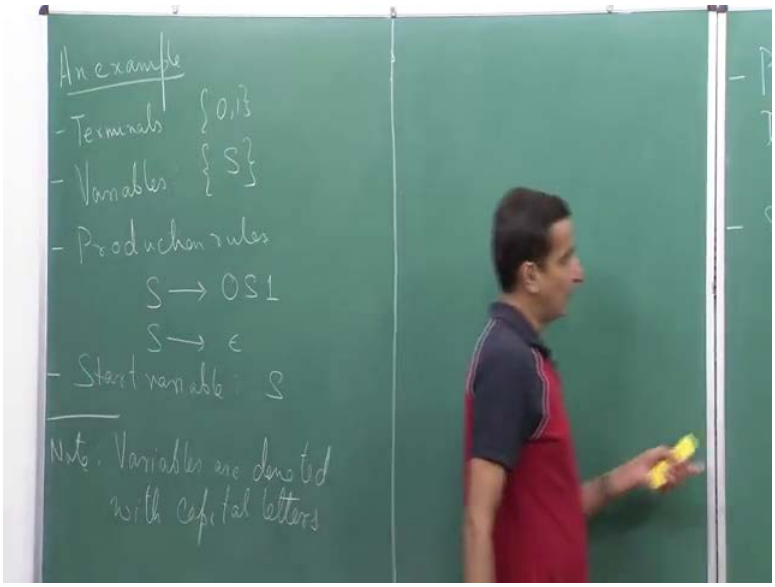
So, what we do is, we first write down the start variable that is the starting string, then we look at the current string; in this case, the current string consists only of the start variable. We replace it with one of its production rules and then we get the new string. Again in the new string, we do the same thing. So, we pick a variable in the current string, and we replace this with one of its production rules.

(Refer Slide Time: 11:18)



So, continue step 2 until no more variables are left.

(Refer Slide Time: 11:39)

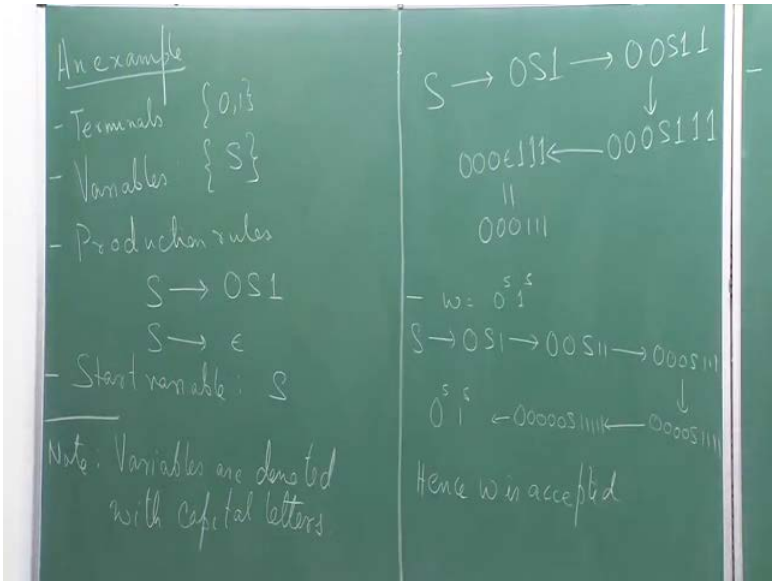


Let us look at an example so to see how computation happens with the context free grammar. So, we look at a context free grammar, where over terminals are the symbols 0 and 1. So, we have only two symbols two terminal symbols 0 and 1. The variables, so this context free grammar has only one variable, it is the symbol S.

Production rules - there are two production rules; S going to 0 S 1, and S going to epsilon. So, recall the definition of a production rule, I mean we have been defined it, but

production rule essentially dictates how a variable gets replaced. So, we have these two production rules. The first production rules say that S can get replaced either with the string 0 S 1 or S can get replaced with the string epsilon. So, these are the two rules. And since the grammar has only one variable that is necessarily the start variable also, the start variable is S here. So, just a small note of conventions, so variables are denoted with capital letters. So, this is the convention that is usually followed.

(Refer Slide Time: 14:25)



Let see how a string is generated. If you look at how a computation happens, we first write down the start variable. So, we first write down S; now we replace S with one of it is production rules, let say I replace S with the first production rule. So, the way it is denoted is S single arrow. So, S gets replaced with 0 S 1; now this is our current string. So, in this current string, we have one variable. So, again we replace this variable with one of its production rule, either I can choose this or I can choose this.

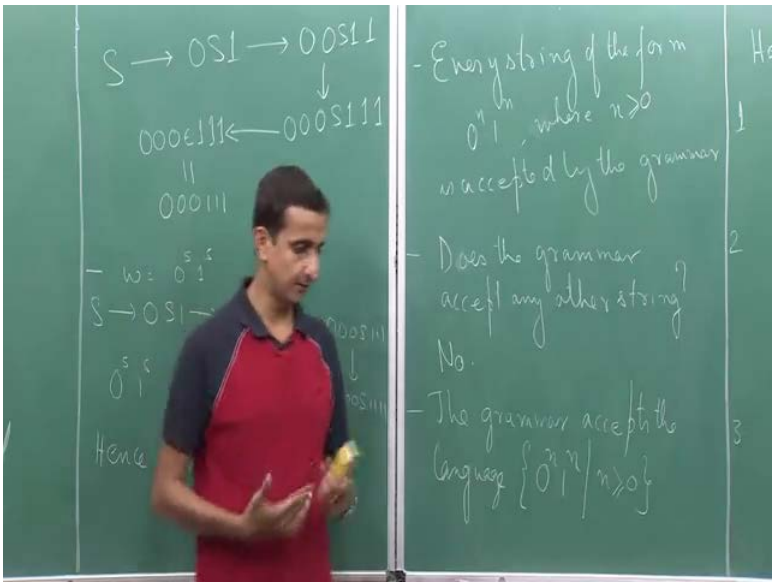
Again let us choose the first one. So, if I replace S with 0 S 1, what do we get. So, we have the first 0, now we replace S with 0 S 1, so we get 0 S 1 and then we have the last 1. Once again this is our current string this as one variable S, so we again let say replace S with the first production rule to get. Now will get 0 0, instead of S, we have 0 S 1 and then we have 1 1.

Once again our current string has a single variable. Let say we replaced S with again one of its production rules this time let us choose epsilon. So, this time let us replace S with

the second production rule that is epsilon. So, what we get is 0 0 0, and we have epsilon. Let us write down epsilon and then we have 1 1 1, but this string is the same as the string 0 0 0 1 1 1. Now we have a string which does not have any more variables left hence we stop. So, this sequence of derivation yields the string 0 to the power 3, 1 to the power 3.

Now, observe that if I take if I look at some other string. Now, if I take the string w equals 0 to the power 5, 1 to the power 5. Can I generate this string using this context free grammar? The answer is - yes. How do we generate this, so we start with S, we replace it with 0 S 1, then we replace again S with 0 S 1, so we get 0 0 S 1 1 then we get 0 0 0 S 1 1 1 then we get 0 0 0 0 S 1 1 1 1. Again we replace S with the first production rule, so we get 0 0 0 0 0 S 1 1 1 1 1. Now that we have five zeros and five ones, I will replace S with epsilon and I get 0 to the power 5, 1 to the power 5, hence w is accepted by this grammar.

(Refer Slide Time: 18:13)



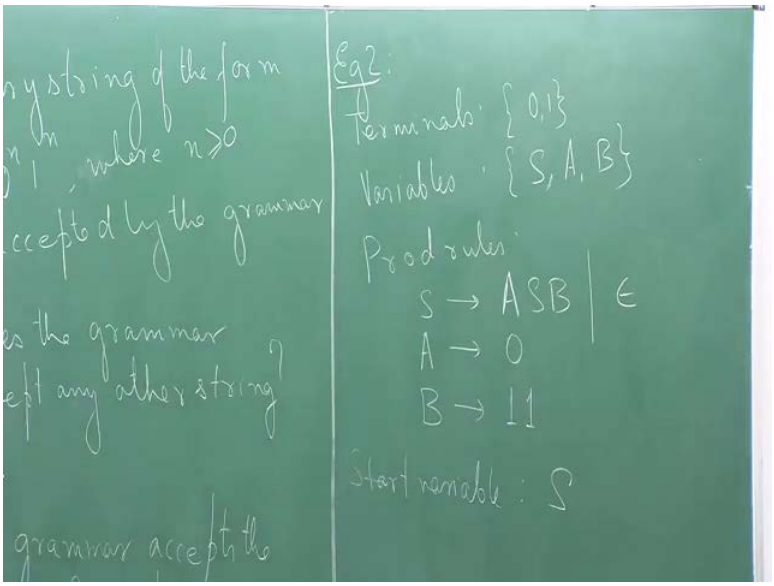
Now, you can easily see that what kind of strings get accepted by this grammar. So, observe that every string of the form 0 to the power n, 1 to the power n, where n is greater than or equal to 0 is accepted by the grammar. And this grammar accepts only these strings. So, does the grammar accept any other string, so the answer to this question is, no. And it requires, all though it requires a formal proof to answer this, but we would not go in to the proof of this fact, we will just allude to how the idea of the proof goes.

So, if you look at this grammar, so if you look at the grammar, this grammar as only two production rules. So, every time the first production rule is used, the current string that I get as only one variable in it; and the way that variable occurs is it as an equal number of 0s before and in equal number of 1s after it. So, the number of 0s before it equals the number of 1 after that variable and then if I invoke the second production rules that variable goes. So, I just get a string over the terminal symbols.

So, if you take any other string any other string any other string consisting of 0 and 1 which is not of the form 0 to the power n, 1 to the power n, you can show with some effort that it cannot be generated by this grammar. So, the answer to this is no. So, what this proves this that the grammar accepts the language 0 to the power n, 1 to the power n, where n is greater than or equal to 0. And if you recall from what we have shown in the past few lectures that this is a non-regular language. So, we have a computational model that is actually able to accept a non-regular language. And as we shall see that there are even other non-regular languages that can be accepted by context free grammar.

So, in our next lecture, we will show that even so every regular language is accepted by a context free grammar, which shows that the class of languages accepted by context free grammar is a proper super set of the class of regular languages.

(Refer Slide Time: 22:05)

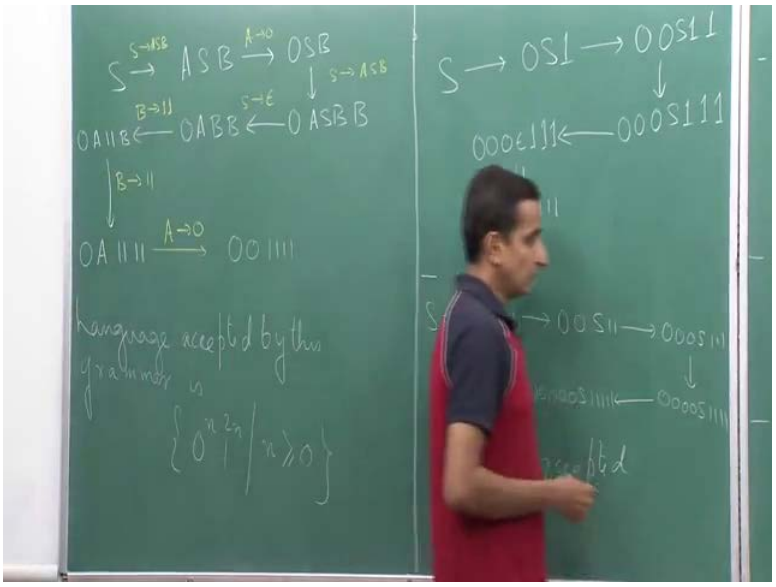


Let us look one more example. And let us try to figure out what is the language accepted by this context free grammar. So, our terminals in this case are the symbols let say 0 and

1; variables are S, A and B. So, it has now we have a grammar which as three variables. The production rules are as follows; so we have S going to A S B; A going to small a; or A going to let say sorry 0; and B going to 1 1. And our start variable is S. So, S usually denotes the start variable. So, what is the language accepted by this context free grammar.

Let us look at a couple of examples. Let us see how this. So, I we need to add one more production rules. So, we have S going to A S B, and we also have the production rule S going to epsilon. But usually when we have such a things, so when we have multiple production rules for the same variable, the usual convention of for writing this is using this line as a separators. So, we have A S B, and we have epsilon. So, this essentially denotes that there are two production rules S going to A S B, and S going to epsilon. So, we will follow this convention.

(Refer Slide Time: 24:23)



First, I write the start variable. Now what can I replace the start variable with. So, I have I have the option of using epsilon or I can replace it with A S B. Now, I have a string, which has three variables in it; and I can replace any of the 3 variables with one of their production rules, let say I decide to replace the first variables. So, I replace with so A has only one production rule. So, I replace this with 0 S B.

Now let say I replace the second production rule with again A S B, so I get 0 A S B, and there is already one B left. So, here I am using the production rule for S, first production

rule for S. Again let say this time I replace S with the second production rule. So, just to make this clear, let me write this down. So, here I am replacing S with A S B; here I am replacing A with 0, here I am replacing again S with A S B. Now, let us again replace S with epsilon. So, what we get, if I replace S with epsilon, I get 0 a instead of S I have epsilon. So, I get B B epsilon concatenated with anything gives back the same string.

Again I can choose to replace any variable here. Let us say I replace the first B with it is production rule that is 1 1. So, I replace B with 1 1, what I get is 0 A 1 1 B. Now again I let say a replace B with 1 1, what I get is 0 A 1 1, again 1 1. Now there is only one variable left. So, I do not have any choice, I will replace A with 0. So, the string that I get is 0 0 1 1 1 1.

Now as probably all of you can guess that what is the language accepted by this grammar. So, the language accepted by this grammar is, so we have a language consisting of if we have n number of 0s then after that we have 2 n number of 1s. So, the number of 1s is the double the number of 0s. So, we have is 0 to the power n, 1 to the power 2 n; and what is the starting value of n now because we can generate epsilon also. So, the starting value of n is 0, so for n greater than or equal to 0.

In our next lecture, we will formally define a context free grammar, and we look at some more examples. And also we look at some the formal definition of what does it mean for a context free grammar to generate a string and so on.

Thank you.