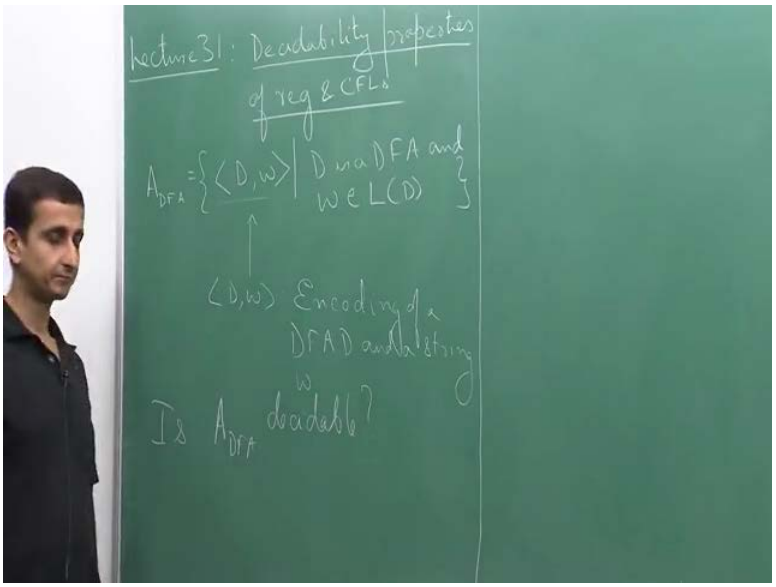
Theory of Computation
Prof. Raghunath Tewari
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture – 31
Decidability Properties of Regular and Context Free Languages

Welcome to lecture 31 of this course. So, today we are going to look at Decidability Properties of Regular and Context Free Languages.

(Refer Slide Time: 00:34)



So, what do mean by decidability properties. So, we want to answer questions like whether the following languages decidable or not. So, for example, consider the languages A DFA, which is encodings of strings I mean encoding of the following nature. So, I have a DFA D, and I have a string w; and I want to answer whether so D is a DFA, and w belongs to the language of D. So, what this means is that I mean this particular notation here, so what this means is that, so this is an encoding of a DFA D and a string w. So, actually we will be looking at encoding of machines encodings of automaton grammar in the next couple of lectures. It is very important that you understand what we mean by this.

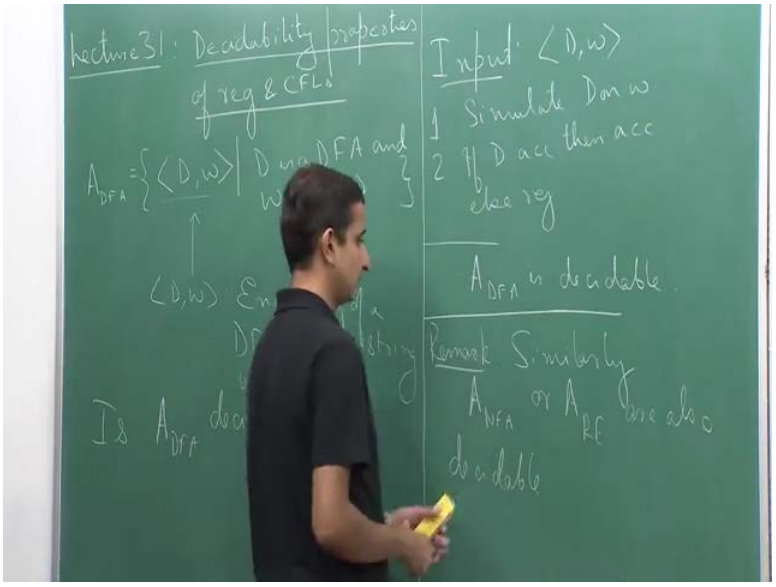
So, remember that what is a DFA. A DFA is nothing but a 5 tuple, which consists of a finite set of states a finite alphabet a transition function again which is a finite function, because it goes from q cross sigma to whatever to q, and it has an accept state and a set

of a start state. Therefore, it has a finite description. So, I can describe a DFA in a finite manner basically by a string which has some finite length. And I can look at an encoding of that string; I mean I can convert that string to a binary string or so on and so. And of course, w by itself is a string. So, I can treat it.

So, basically I can always look at an encoding of a DFA and a string w in whatever suitable format that you want. So, if you want the alphabet to be whatever I mean things like open bracket, close bracket, symbols of the alphabet, symbols and numeric symbols and so on things like that that will helps us encode. In a similar manner, we can also encode things like push down automaton. Again in the case of push down automaton although we can have unbounded stack, but recall that the transition function is a finite function same thing for Turing machine also. So, Turing machine the transition function is always a finite function. Therefore, always when we talk about encodings, encodings are finite objects.

Now suppose we are given an encoding of a DFA, and as string w, can we check if the following is can we check whether the following language is decidable or not. Given D and w whether w is in the language of D or not or whether D accepts w or not, can we write an algorithm to do this. The answer is. The question is A DFA decidable. The answer is yes, it is decidable.

(Refer Slide Time: 04:29)



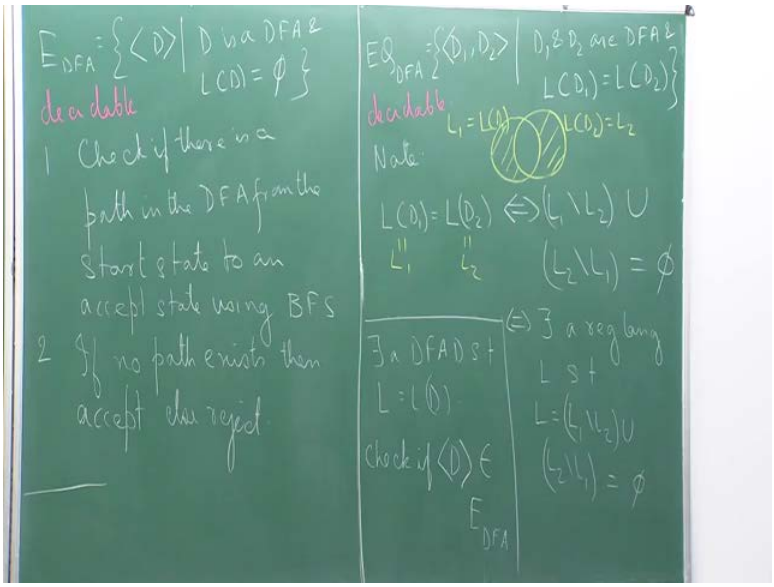
So, consider the following algorithm. So, we have input some D, w. So, an algorithm is just another way of writing a Turing machine because if I can design an algorithm I can always translate an algorithm to A Turing machine. So, I will be using the term constructing a Turing machine and constructing an algorithm or designing an algorithm in a synonyms manner. So, we are given D w. So, what we do basically in this thing is that we simulate D on w, if D accepts then accept, else we reject.

Of course, it must always be the case that either D accepts or D rejects, because once you pass through the strings you just have to pass through the string, so think of D as a graph, so whenever you read one symbol of w you take one step in D. And in some finite number of steps, you will reach the last state; so if the last state is an accept state, you accept otherwise reject. So, this shows that a DFA is decidable. So, we want to study these kinds of properties of regular and context free languages.

Now let me just change the context a little bit. So, instead of a DFA if we were given an NFA, suppose if you are given an NFA and string w, can we still do the same do the same thing we still decide whether w belongs to the language of the NFA. So, once again the answer is yes, and here we will use the equivalence of DFA and NFA. So, what we will do is that first will convert the NFA to a DFA, and then we check do the same thing. So, we have an extra step at the beginning.

Similarly, instead of a DFA or a NFA if you are given a regular expression and a string w, again we can do the same thing, because I can convert the regular expression to an NFA first, then I will convert the NFA to a DFA and then I will check whether w belongs to the language of the DFA. So, all these things can actually be decided. Now, let us look at another problem. Let me just make a remark here. Similarly, A NFA or A Re are also decidable, so a NFA and are as I just mentioned.

(Refer Slide Time: 07:59)



Now, let us look at the second problem that is empty DFA EDFA. So, here in the encoding we are only given a DFA D, no string. And what we have to check is whether, so D is a DFA and L of D is let say empty it is the empty set how can we check. So, when is the language of a DFA an empty set, so the language of a DFA is an empty set if there is no path in the DFA from the start state to an accept state. So, what we do is we check if there is a path in the DFA from the start state to accept state using any graph traversal algorithm, using let say BFS; if no path exists then accept else reject. If there is no path, I will just call this second step.

So, if there is no path then there means that you cannot reach from the start state to the accept state on any string which means that the language is empty which means that we should accept. But if there is some path then there is some string on which the DFA accepts in which case we D does not belong to this language; the encoding of D does not belong to this language, hence we reject. So, once again similarly, emptiness of an NFA or emptiness of a regular expression can also be checked. So, I can convert both of them to a DFA first. So, this is also decidable. Let me remark here.

Now, let us look at the next language EQ DFA. So, in EQ DFA we are given actually encodings of 2 DFA, and we need to check whether their languages are equal or not. So, we are given a DFA D 1 and D 2 encoded; and so D 1 and D 2 are DFA; and L of D 1 is equal to L of D 2. So, how can we check this? So, instead of designing an algorithm, we

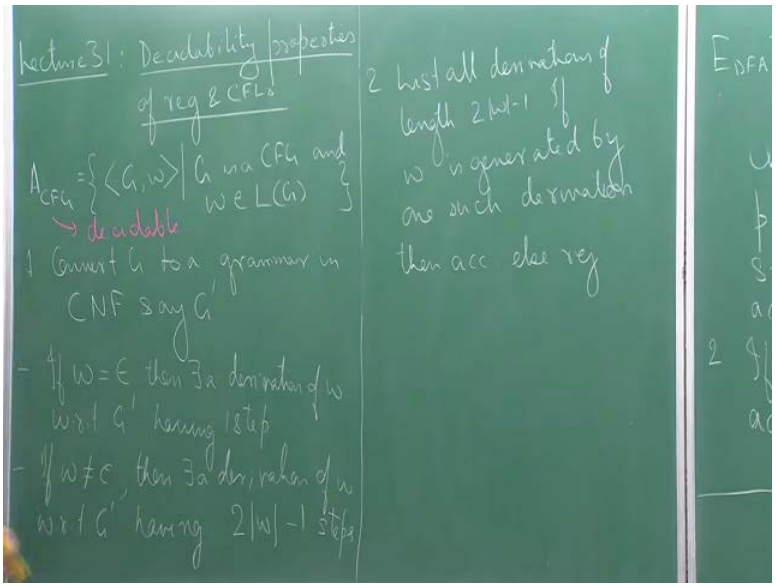
will actually use closure properties to prove, this closure properties and we will kind of reduce this to one of our earlier problems. So, observe the following. So, L of D 1 is equal to L of D 2 if and only if I can write it as follows that L of D 1 minus L of D 2 union with L of D 2 minus L of D 1 is equal to the empty set.

So, if I think of them as sets, so let us actually draw the diagram. So, suppose this is my L of D 1 and this is L of D 2. Let me just for the sake of a convenience; let us just denote this as L 1 and this as L 2. So, if I denote this as L 1 and L 2, I can just write this as little simpler way L 1 minus L 2 union with L 2 minus L 1 is empty. So, L 1 minus L 2 is nothing but this region, and L 2 minus L 1 is nothing but this region. So, observe that L 1 is equal to L 2. So, in this case, this is my L 1 and this is L 2. So, L 1 is equal to L 2 if and only if this shaded region is empty; if the shaded region is not empty then there is some string which belongs to one of the two languages, but not the other only if the shaded region is totally empty then we can say that these two languages are equal.

And how can I check this we have seen that regular languages are closed under set difference and union operation which therefore, implies that there exist a so regular language L such that L is equal to L 1 minus L 2 union with L 2 minus L 1 which is the empty set. Now because there exist a language L I can construct the DFA for L. So, starting with the DFA for L 1 and L 2, so first what I do is that I construct the DFA for L 1 minus L 2, then I construct a DFA for L 2 minus L 1, then I construct the DFA for the union of these 2 things and that will give me a DFA for L.

And now using the emptiness of the DFA algorithm I just check whether the language of whether the language L is empty or not which means that there exist a DFA D such that L equals L of D now check if this D belongs to E DFA, if it belongs then we accept, otherwise reject. So, this proves that EQ DFA is also decidable. So, these were three decidability properties of regular languages.

(Refer Slide Time: 17:36)



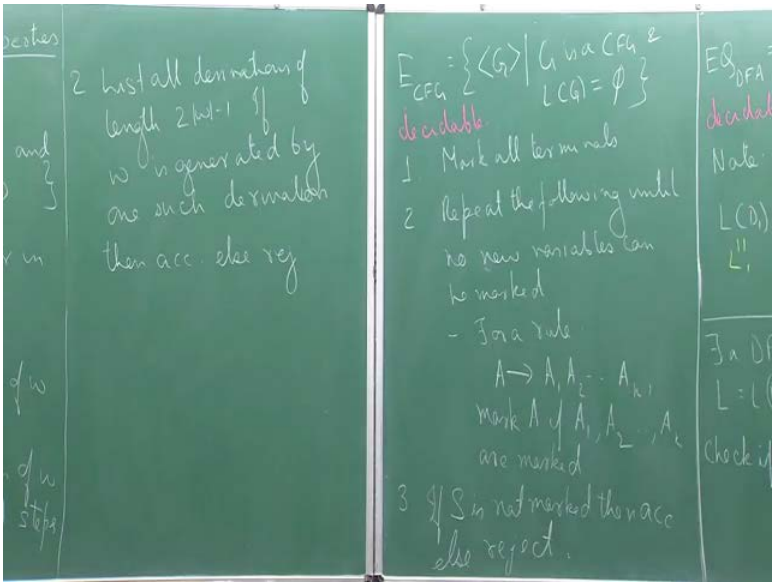
Now let us look at the corresponding properties of context free languages. Now, instead of a DFA, let us say we have A CFG and instead of DFA D, we have a grammar G. So, G is a context free grammar, and w belongs to L of G. So, how can we check whether some string belongs to the language of a grammar or not? So, to check this, what we first do is that we convert G to a grammar in Chomsky normal form. Now once we have a grammar in a Chomsky normal form, Chomsky normal form say G prime. So, once we have a grammar in a Chomsky normal form what we can say is that if we are given a string w, and then if w is not the empty string, then every derivation of w will have twice the length of w minus 1 length.

So, what I mean to say is that, so if w for example, is the empty string, if w is epsilon then there exist a derivation of w in with respect to G prime having only one step. Else, if w is not the empty string, then there exists derivation of w with respect to G prime having twice mode w minus 1 step. So, this is what we will use. So, if G prime is a grammar in Chomsky normal form then there is a derivation of the string w that has this many number of steps if w is not empty; and if it empty then there is only one step. So, from the start variable, you just go to epsilon that is all; otherwise, there are this many.

Now what we do is that so our algorithm, so let me continue the algorithm. So, first we convert G to a grammar in Chomsky normal form then list all derivation of length twice w minus 1, if w is generated by one such derivation then accept, else reject. So, you just

use this property of Chomsky normal form grammar, and we list out all possible derivations that can be generated. And if w is generated then we accept; otherwise, we reject. Now let us look at this problem. So, this proves that A CFG is decidable. So, once again A PDA is also decidable, because I can convert PDA to a context free grammar and then I will convert the context free grammar to a grammar in Chomsky normal form.

(Refer Slide Time: 22:47)



Now consider ECFG, how can we check whether the language of a context free grammar is empty or not. So, here the idea is once again to kind of work backward. So, what we do is that we will mark variables in an iterative manner. So, first of all, we pick a rule, and we look at the right hand side of the rule. If the right hand side consists only of terminals, we mark the variable on the left hand side; we do this for all rules.

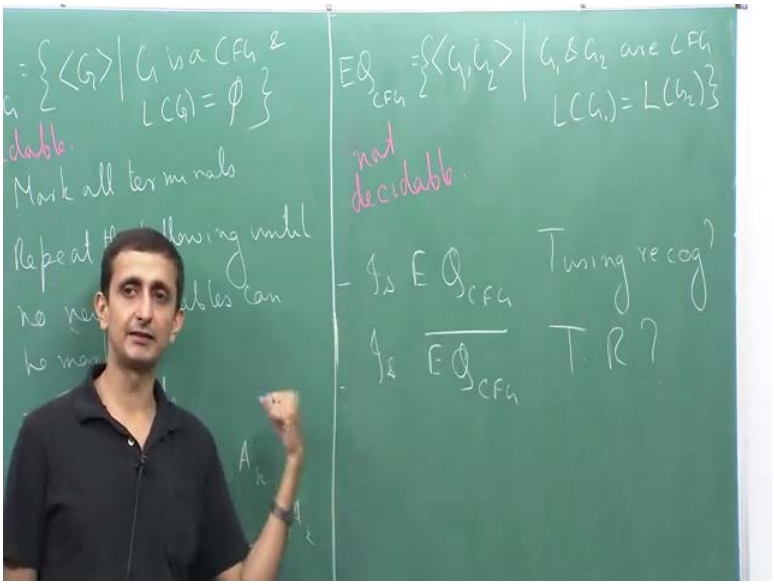
Now, again in the second step, I again pick the rules one by one, and I look at the right hand side. So, if there is a terminal or variable that is already mark then I will if all the terminals and variables are marked, then I will mark the variable on the left hand side and I keep on doing this until no new variables can be marked. Now, if the start variable gets marked then I say that the grammar generates a string which means that it is not empty the language of the grammar, otherwise it is empty.

The algorithm is as follows. So, first mark all terminals. Next, what you do is repeat the following until no new variables can be marked. So, what do you do? So, for a rule, let say that A going to A 1, A 2 through A k; mark A, if A 1, A 2 up to A k are marked. So,

I will mark A, if all of them are marked; otherwise, I do not mark; and I keep on doing this process.

So, finally, if S is not marked then accept, else reject. So, S is the start variable. So, if the start variable is not marked which means that you cannot generate string of terminals from the start variable which means that the languages is empty then you accept otherwise you reject. So, this proves that this language is decidable.

(Refer Slide Time: 26:33)



Now we come to the last thing. So, we have EQ CFG where we are given two grammar G 1 and G 2; and G 1 and G 2 are CFG such that L of G 1 is equal to L of G 2. So, this language is actually not decidable. So, observe that we cannot use closure properties in the case of context free grammars, because as we had seen earlier context free languages are not known to be closed under the set difference operation.

So, like regular languages, we cannot use that property. And it turns out that we cannot use any other approach also. So, this is something which is not decidable. Although, we will not be able to prove this in this course I mean we will not be giving prove of this, but this is something that you should remember.

So, an interesting question that one can ask is well it is not decidable this is fine, but this EQ CFG Turing recognisable, or if this is not true is the compliment of EQCFG Turing recognisable. So, of course, both of these cannot be true, because if both of them are true

then it would imply that EQ CFG is decidable which is not true. So, is one of the two statements true - so this is again something that you can think about. So, I will stop here.

Thank you.