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

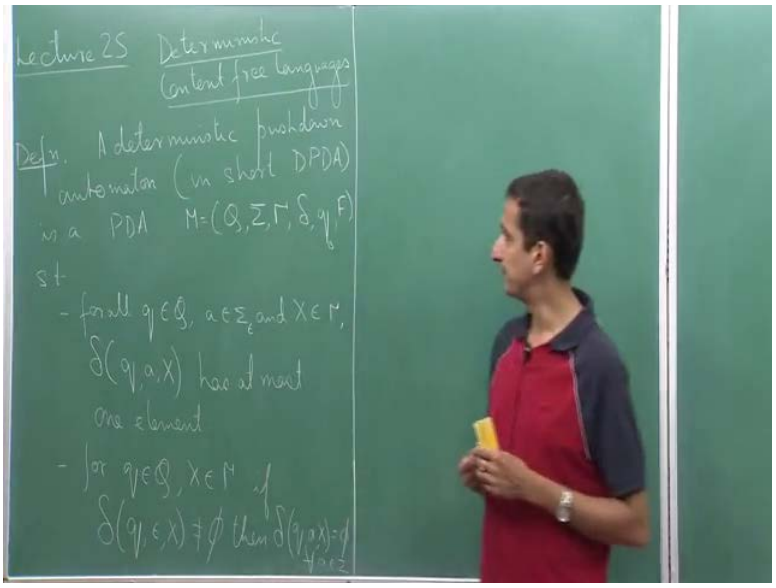
Lecture – 25
Basic Notation and Convention, DFA Edit Lesson

Welcome to the 25th lecture of this course. Today, we are going to discuss another class of languages known as Deterministic Context Free Languages.

Deterministic Context Free Languages are defined with respect to the automaton known as deterministic pushdown automaton. They are class of languages which are a sub class of general context free languages, but a super class of regular languages. So to look at this class, let us first define what deterministic pushdown automaton is. The idea here is that we have seen that a pushdown automaton is defined as an automaton, which has a memory in the form of a stack and it has a finite state which can use non-determinism; in other words, it is basically an NFA together with a stack.

Now, what happens if I do not allow non-determinism in the finite control of this automaton? I allow the stack, but I restrict the automaton to only make deterministic moves. So, when we say deterministic moves in this case, what we essentially mean is that at any step of its computation, the automaton can only make a unique transition. They should not be multiple transitions that the automaton can make. In other words, if I look at the computation path of if I look at the computation of any string on the automaton, it should be a unique path, it should not have tree like structure.

(Refer Slide Time: 02:22)



Let us look at the definition of a deterministic pushdown automaton. A deterministic pushdown automaton in short we will call this in DPDA is a six tuple. Or let me just say this as is a pushdown automaton M equals Q, sigma, gamma, delta, q naught, F; such that there are two conditions, so this transition function delta needs to satisfy two things. So for all q belonging to capital Q, a belonging to sigma, and let say x belonging to gamma; delta q a x has at most one element.

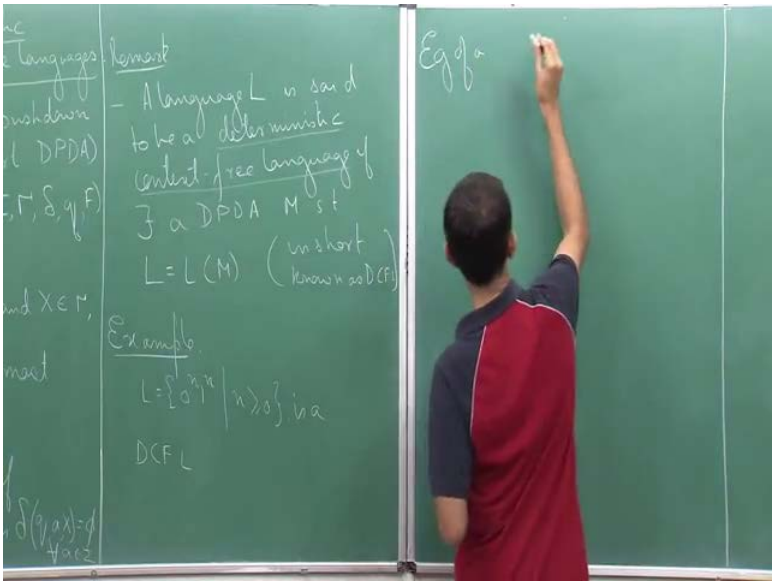
So, here what we can have is that so a can also be the empty string, so we will allow this to be sigma epsilon. So, if I take any state, if I take any input symbol including the empty symbol, and I take some symbol in gamma then q a x has at most one transition. So, from the triplet q a x, I can only have at most one transition.

And the second condition is that for q belonging to Q, and let say x belonging to gamma, if delta q epsilon x is non-empty, then delta q a x is empty, for all a in sigma. So let us try to understand the second condition, but the second condition is saying that if I have a triplet q epsilon x if I look at the triple q epsilon x. In other words basically I making an epsilon move that is without reading any symbol, I am making a move. So from any state q, if I make an epsilon move, then from that same state q and on the stack symbol x there should not be any other moves.

So for all a in sigma delta q a x should be empty, so at most one of them can be non-empty; either this is non-empty or this side is non-empty. So, it cannot be that both are

non-empty. So, this is basically at the definition of a DPDA. So, if you look at the definition it is a restricted form of PDA, so it is a PDA where the transition function satisfies these two additional constraints.

(Refer Slide Time: 07:28)

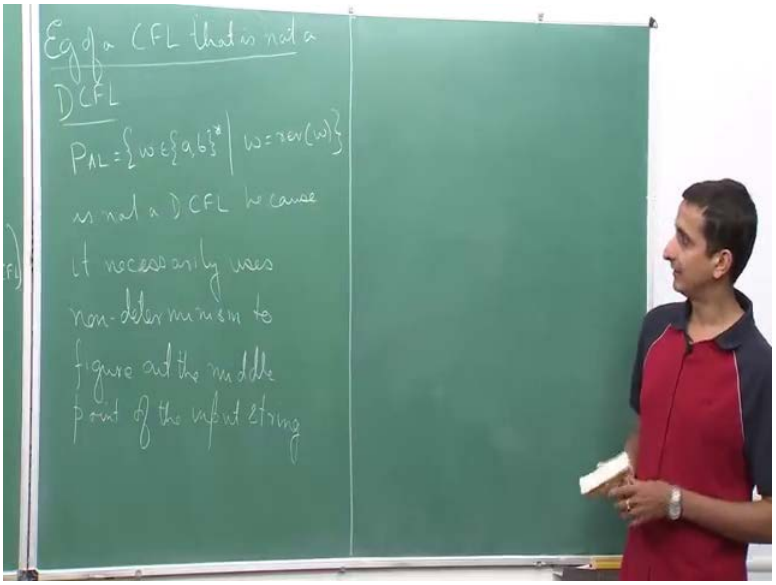


Now what can we say about the power of a DPDA. There are few things for example, so a language L is said to be a deterministic context free language; if there exist a DPDA M such that L equals L of M. So, deterministic context free languages are the class of languages that are accepted by deterministic pushdown automaton. So, an example is the language that we have seen 0 to the power n 1 to the power n. So, you can easily see, in fact, you can try this as an exercise to construct a pushdown automaton for this language, which is a deterministic pushdown automaton. The intuition behind this example is the following.

If you look at any string which has a sequence of 0s followed by a sequence of 1s, so whenever you are reading a 0, you deterministically keep on pushing a symbol onto the stack; then when you see a 1 you again deterministically keep out popping that same symbol. So, you do not need to use non-determinism anywhere in order to accept a string of this kind; you can modify the pushdown automata that we had seen earlier to make it into a pushdown automaton that is deterministic.

But there are examples of context free languages that are not deterministic context free languages. So, here let me just write this. So, this is a deterministic context free language. So, I will just write it here in short known as DCFL.

(Refer Slide Time: 10:57)



Now let us look at an example of a non-DCFL. So, here is an example. So, consider the language of all palindromes. So, I will just call it pal, so we had seen this example last time. So, consider all strings w over let say a b star such that w equals the reverse of w, so no matter in which direction I read w, it appears the same. So, last time, we saw how to design a pushdown automaton for this language.

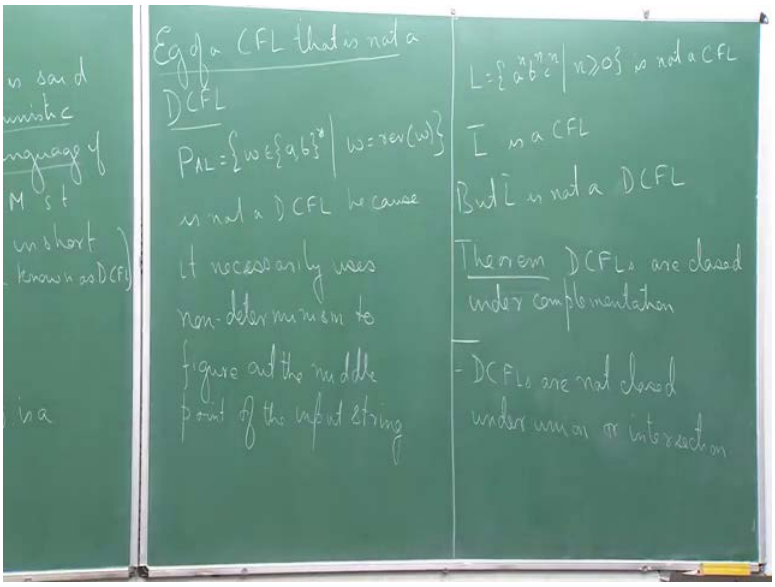
And if you recall in the pushdown automata, I mean we use non-determinism in order to decide what was the middle point of the string. So, basically the idea of the pushdown automaton was that you keep on pushing in symbols until you reach the middle point of the string, or in other words you push half of the, the first half of the string onto the stack and then you compare the stack with the remaining half of the string, whether it is the same string or not. So, if the first half matches the second half in reverse order which is the way a stack works then you say that the string is a palindrome or in other words it belongs to this language.

So, important point in this argument was to determine or to figure out what was the middle point of the string. And non-determinism actually helped us achieve that, but because deterministic pushdown automata cannot use non-determinism by it is

definition, so from the state q 1, if you remember the states from last lecture, if you want to go from q 1 to q 2, so you actually have to deterministically do that. I mean you cannot have an epsilon transition which goes from q 1 to q 2 without reading some bit as well as a transition on some symbol a or b, so that was the reason so for going from q 1 to q 2 the non determinism that is used. It is actually essential to check whether a string belongs to this language or not. You actually cannot do without it.

So we would not be able to prove this that this language is not a deterministic context free language in this course, I mean the tools the techniques that we required are beyond this course, but at least it is good to know that there are languages, there are context free languages which are not deterministic context free languages. So, this is not a deterministic context free language, because it necessarily uses non-determinism to figure out the middle point of the input string.

(Refer Slide Time: 16:04)



Let us look at one more example. So, recall that last time we had shown that the following language, will not last time, but a couple of lectures back. We had shown that the language L which is of the form a to the power n b to the power n c to the power n or n greater than or equal to 0 is not a context free language. But as it turns out, if you look at L compliment, L compliment is a context free language, so this is again an exercise that you can try out.

So, what is the language L compliment? So, first of all L compliment can consist of all strings that are not of the form a star b star c star, so that have some other form that you can actually easily check, because to check whether a string is of the form a star b star c star is essentially a regular operation, because it is characterized by a regular expression. Hence it is compliment is also a regular language.

So if it is of the form a star b star c star then the only thing that we need to check is to check whether a string belongs to L compliment. We need to ensure that either the number of a(s) is not equal to the number of b(s) or the number of b(s) is not equal to the number of c(s). And you can actually represent both these using a context free language, and context free languages are closed under union, hence L compliment is a context free language, but so that I will leave as an exercise. But what I want to say at this point is that this language L compliment although it is a context free language, it is not a deterministic context free language.

In fact, here we have a theorem that is a crucial difference between CFL is and DCFL is that deterministic context free languages are closed under complementation, so this is a fundamental difference between DCFL(s) and CFL(s). There are other things so for example, so again I would not prove this theorem, I will just state it. There are some more non closure properties, so DCFL(s) are not closed under union or intersection, so they are not closed under union; they are not closed under intersection as well.

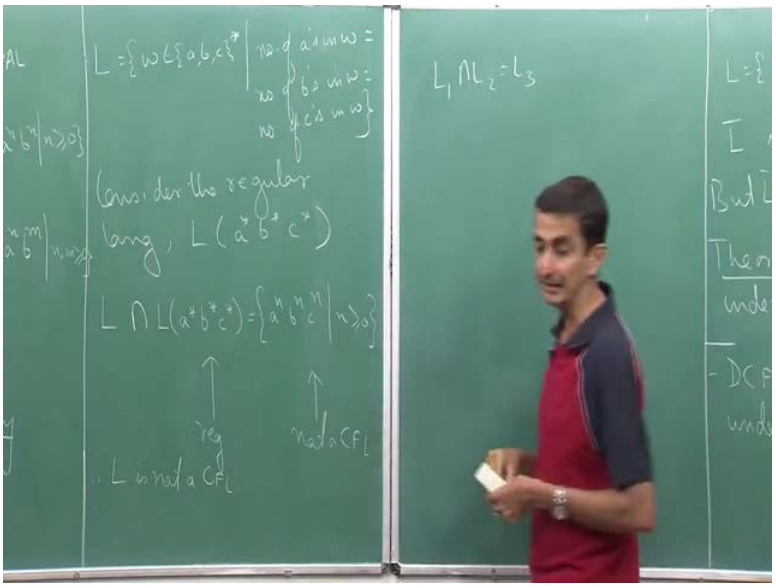
(Refer Slide Time: 20:01)

Let us look at the current picture of the class of languages that we have. So, we have the class of regular languages. So, regular languages are closed under most of the standard operations that we have such as union, intersection, concatenation, star, compliment etcetera. So an example of a language here is for example, the language a to the power n b to the power m ,where n coma m are greater than or equal to 0, because here I just need to check whether the string has some number of a(s) followed by some number of b(s) they need not be equal. Then we have deterministic context free languages which are a super set of regular languages.

An example of a language here is a to the power n b to the power n, n is greater than 0, so this is deterministic. And as we saw the deterministic context free languages are closed under compliment, but they are not closed under union, intersection. And again we have another class of languages which are a super set of deterministic context free languages which are context free languages. There are several examples here, but some problem an example of a language that is a context free language, but not a deterministic context free language is for example, palindromes. So, we just saw this example today.

So, this hierarchy in fact, in the next part of this course, when we look at even larger class of problems so that is even super sets of context free languages. So, this hierarchy is known as the Chomsky hierarchy. So, this hierarchy of class of languages, so the smallest is regular then deterministic context free, then context free, and we will see some more later on. So this is something that I wanted to mention. So, before ending, I just want to give an application of a closure property of context free languages. So, last time we saw that context free languages are not closed under intersection with themselves, but they are closed under intersection with a regular languages.

(Refer Slide Time: 23:18)



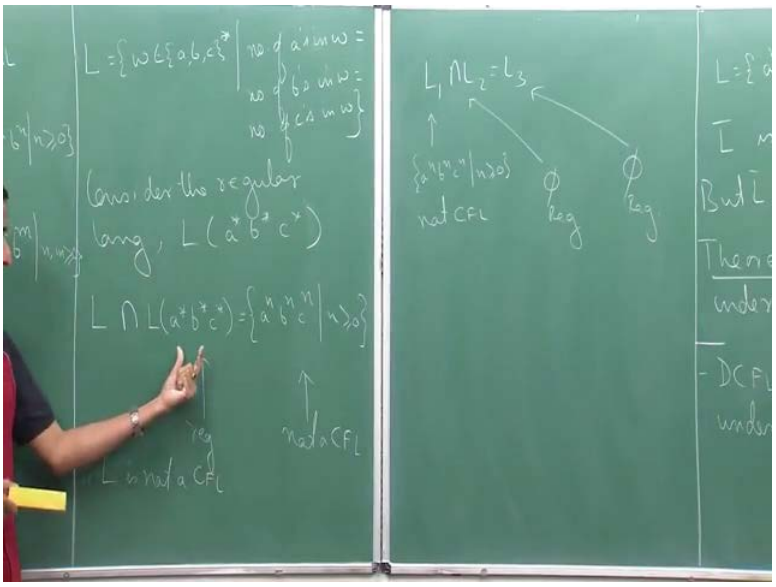
Let us look at an example of an application of this fact. So, consider the language L which is equal to set of w(s) belonging to a, b, c star such that the number of a(s) in w equals the number of b(s) in w, which is equal to the number of c(s) in w. So, how do we show that and we want to argue that this language is not a context free language. So how do we show this? So, consider the regular language the language of a star b star c star; so consider the language generated by the regular expression a star b star c star that is you have a sequence of a(s) followed by a sequence of b(s) followed by a sequence of c(s).

Now what is the language L intersection with L of a star b star c star? So, observe that this is nothing but the language a to the power n b to the power n c to the power n, for n greater than or equal to 0. Because if I look at all strings in L that have an equal number of a, b and c and it is of the form a(s) followed by b(s) followed by c(s), then it must be of this form know that this language is not a context free language, we have proved this. We know that this language is regular. Therefore, if L is a context free language then an intersection of a context free language with the regular language would have given as a context free language, hence for L is not a context free language, so this is what we have.

So, again before I end, I just want to point out one thing. So, on the left hand side, we have a language that is a regular language and we do not know what the status of a(s) L is. So, suppose if L is a context free language, and or maybe L is not a context free language, L is not a context free language and this the second language is a regular

language it can still happen that this language on the right hand side is a context free language or maybe even a regular language.

(Refer Slide Time: 27:01)



Let me just mention that. For example, if you have L 1 intersection L 2 giving a language L 3, it can very well happen that L 1 is not context free, L 2 is regular and L 3 is also regular, so that can very well happen. But again it is not a generic statement it need not happen for every such L 1 and L 2. But for example, if I take L 1 as a non context free language let say a to the power n b to the power n c to the power n, and I take L 2 to be the empty language, so the empty language is of course, regular then what is L 3.

So intersection of the empty language with any language is of course, the empty language therefore, this guy is not context free, this guy is regular, and this guy is regular, so that can very well happen. But what we are using here is the fact that if the right hand side is not a context free, and one language in the left hand side is regular, then the other language must necessarily be a non context free language. So, I will stop here today.

Thank you.