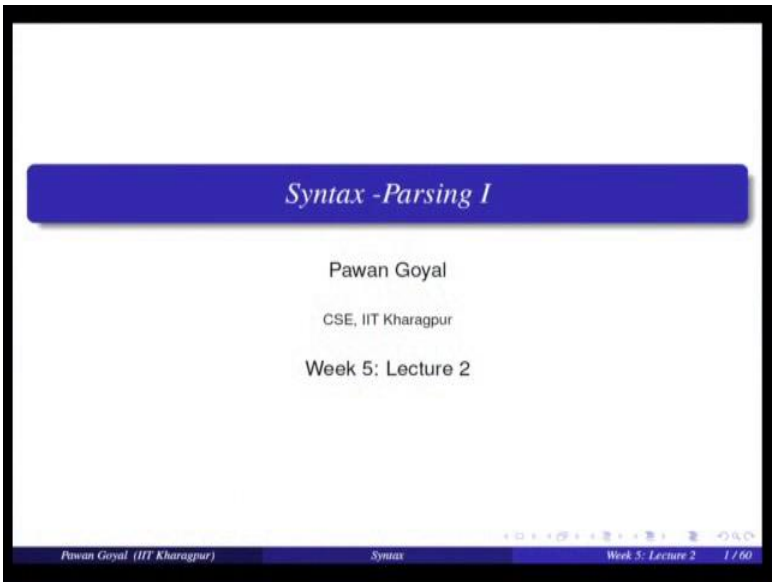
Natural Language Processing
Prof. Pawan Goyal
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 22
Syntax - Parsing I

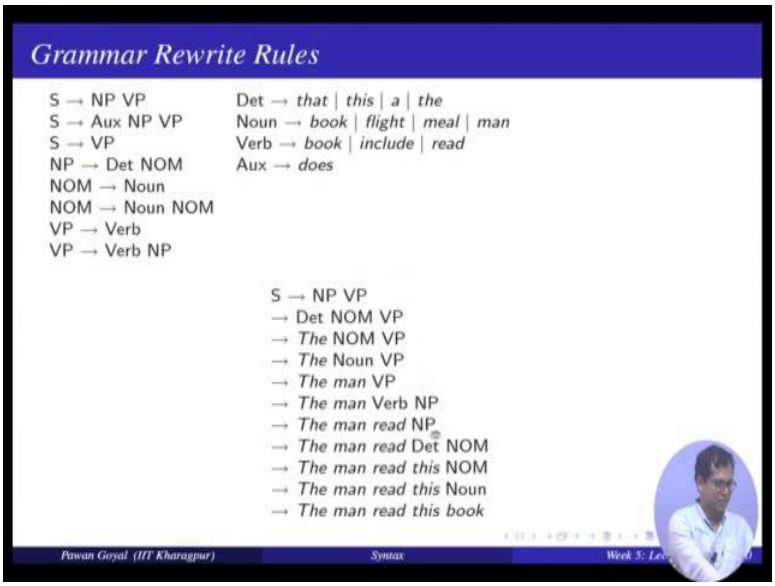
So, welcome back for second lecture of this week. So we had started our discussions on Syntax.

(Refer Slide Time: 00:24)



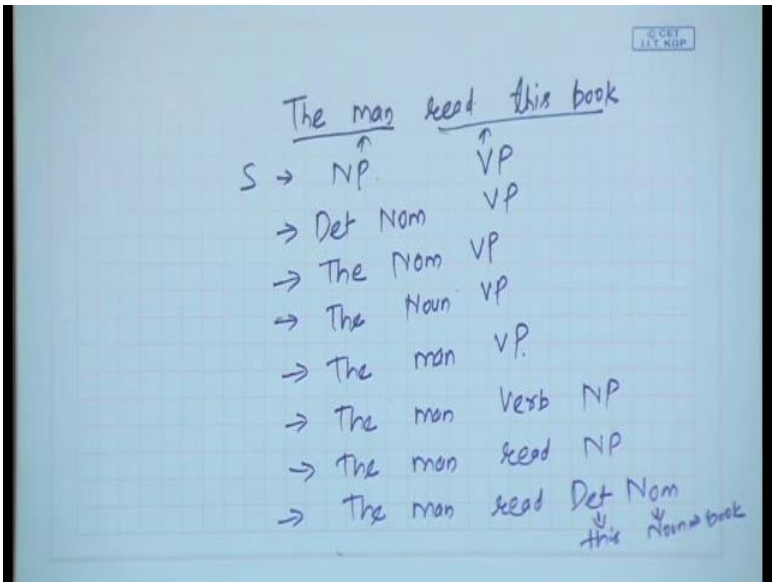
So in the last lecture we had talked about what is the formulation for doing the syntax parsing. So we had talked about context free grammars. Now we will see how do we use context free grammars for the actual parsing.

(Refer Slide Time: 00:46)



So let us take a simple grammar in terms of in a CFG formulation. So you have in the grammar it is said that a sentence can be a noun phrase followed by a verb phrase or an auxiliary followed by a noun phrase followed by verb phrase or single verb phrase. Noun phrase can be a determiner followed by a nominal can be a noun and so on. So all the various possibilities are shown here and similarly these are all the pre terminals that are giving me terminals that are words in my lexical. So this is one such grammar. And now using this grammar I can write various sentences. So for example, I want to write the sentence - the man read this book. So I want to write this sentence the man read this book. Now how do I what kind of rules in the grammar do I use so that I can generate the sentence. So I have to write the man read this book.

(Refer Slide Time: 02:10)



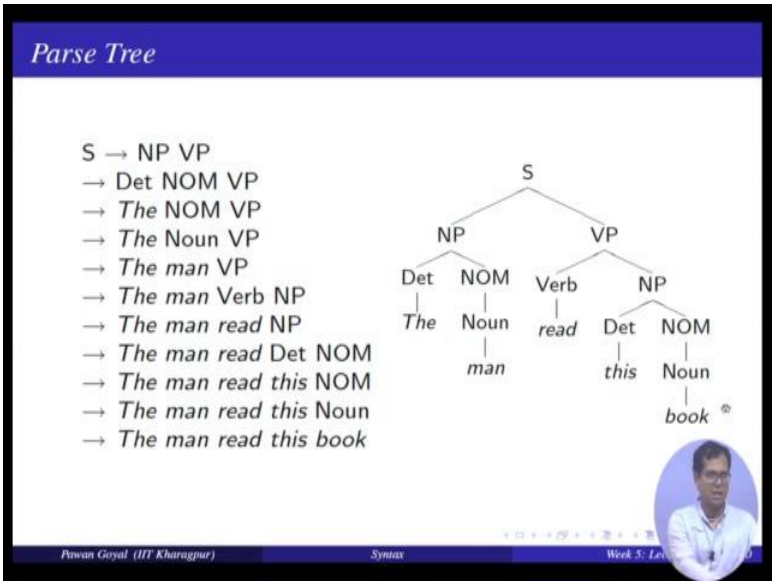
So let me just do it quickly on the paper. Now the sentence can be a noun phrase followed by a verb phrase or an auxiliary followed by a noun phrase verb phrase or a verb phrase. In this case it does not start with an auxiliary nor a verb. So it has to be a noun phrase followed by a verb phrase. So I have to first start by a noun phrase followed by a verb phrase. And I will assume that the man will come in noun phrase and read this book come in verb phrase.

So what will be the next derivation? So NP have to get the man. So I have to get something that says determiner nominal. So here I will say determiner nominal and verb phrase. And determiner can give me a nominal can give me noun which can give the word man. So in some steps I can determine to the nominal verb phrase the noun verb phrase. And if I want to complete this I will say the man verb phrase. Now the verb phrase I have to derive read this book. So verb phrase goes to verb and noun phrase is the something like that yes verb phrase goes to verb and noun phrase.

So I will use this. And say the man verb noun phrase verb can directly give me read, the man read noun phrase. And from noun phrase I have to derive this book. So I have to again write this determiner for a nominal. This will become a determiner the man read determiner nominal. And this will give me this, and this will give me noun. And this will give me book. And that will give me the whole sentence the man read this book. That is, I derive a sentence using this grammar.

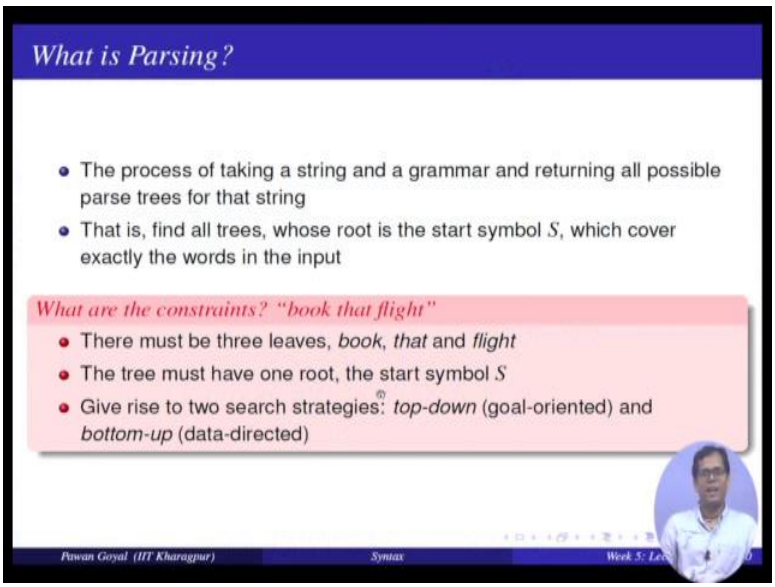
So now once I have this derivation I can use this derivation to denote what is the particular syntax tree that generalizes the sentence as per this grammar.

(Refer Slide Time: 04:40)



So same derivation can be used for to show the parse tree, so this is my derivation and this derivation can be shown using this parse tree. In the question of sentence, you derived NP VP from NP you derived determiner and nominal VP verb and noun phrase. Noun phrase to this particular sub, so this is exactly what was the derivation that you did for the sentence in my grammar.

(Refer Slide Time: 05:09)

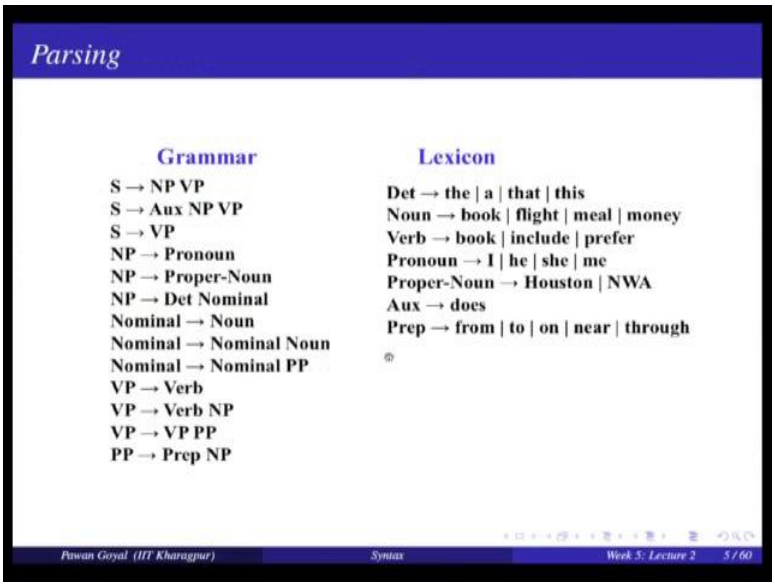


So that can immediately tell me what is the problem of parsing. I am given the grammar and I am given the sentence in my language. So given the sentence in my grammar I want to find out what is it is actual parse tree. And they may not be unique parse they may be multiple parse tree. So the process of parsing is to find out all the possible parse trees for a given sentence as per my grammar. So find out all the trees whose root is the start symbol S because I have to start with the start variable in my context free grammar and which cover exactly the words in the input.

Now, what are the constraints on which I do the parsing? Suppose my sentence is book that flight. So what are the constraint? In my parse they should be 3 leaf nodes. Book that and flight and MS is you start at the start node. And then I have to explore all the possible rules such that I come up with the tree is starting with us as the root node and book that flight as the only 3 nodes 3 leaf nodes. So yes the tree must have one root the start symbol S and that tells me that I can explore it in at least 2 different ways. One is I can with top down you start from S try to find out all the possible trees in finding out I can have as per my grammar and see if there is one tree that can give me the only the leaf nodes book that flight this is top down.

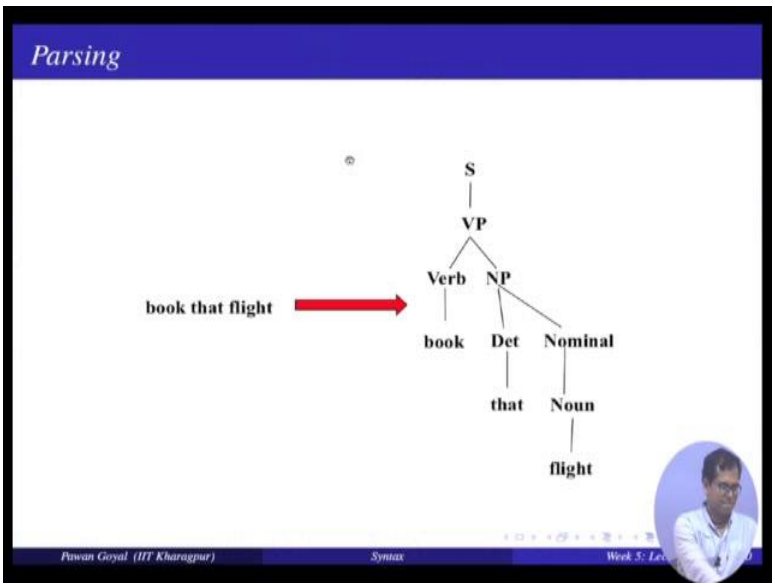
Other approach is bottom up. I start from book that flight go up words what are the non-terminals and then see if any of this combination can lead to a full tree it starting from S. So these are 2 different strategies. And that is what we will discuss how do we use it top down or bottom up approach for parsing given this grammar.

(Refer Slide Time: 07:09)



So let us take this grammar. So you have certain rules. So left hand side is mainly the rules for non-terminals and pre terminals and right hand side are for the lexicon, pre terminals deriving the terminals. Now using this grammar, you want to find out the parse for book that flight. And we will use both the top down and bottom up strategies for finding out this parse tree.

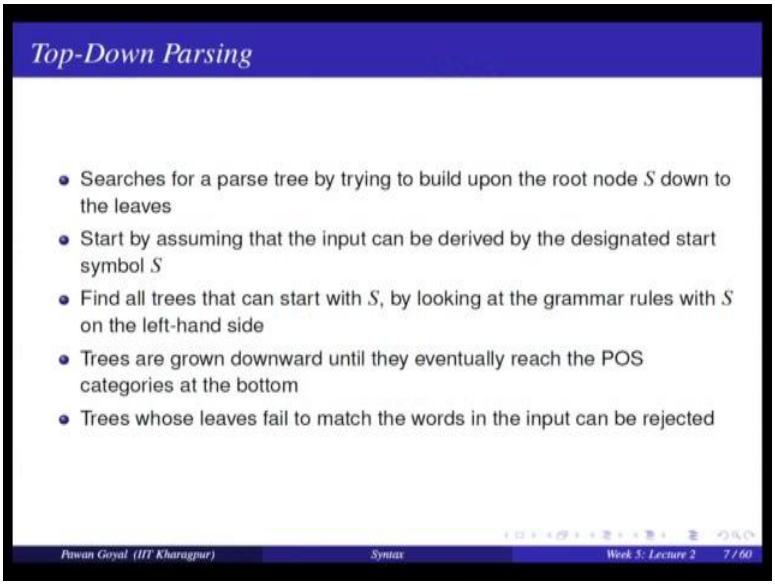
(Refer Slide Time: 07:40)



So yeah this is also what is the expected parse tree. Book that flight this is your whole verb phrase, it is starting from a verb and a noun phrase. So a book is a verb and this that

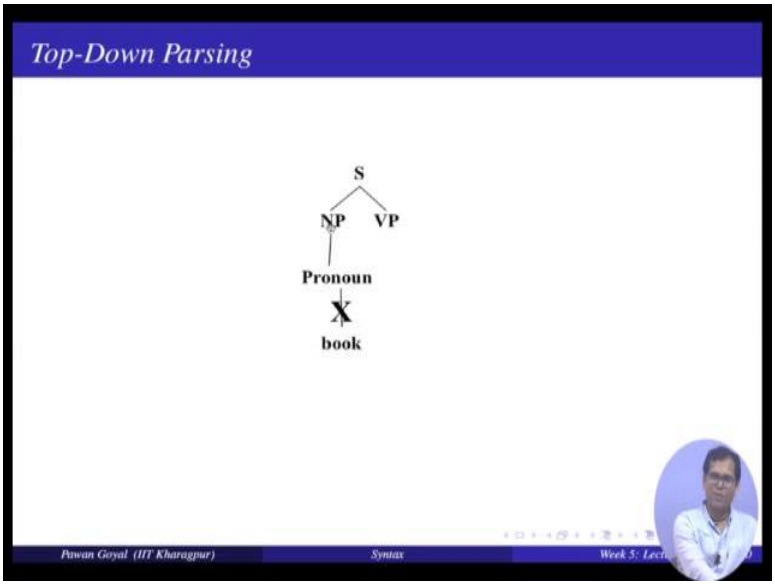
flight is a noun phrase. So I want to come up with this parse tree given the grammar, and how do I do that in a deterministic manner using the using my top down strategy

(Refer Slide Time: 08:19)



So how do I start? So I have to start from my start node root node in top down and using my grammar as my grammar I will see what is the different possible rules I can apply at this point, keep on going downwards.

(Refer Slide Time: 09:20)



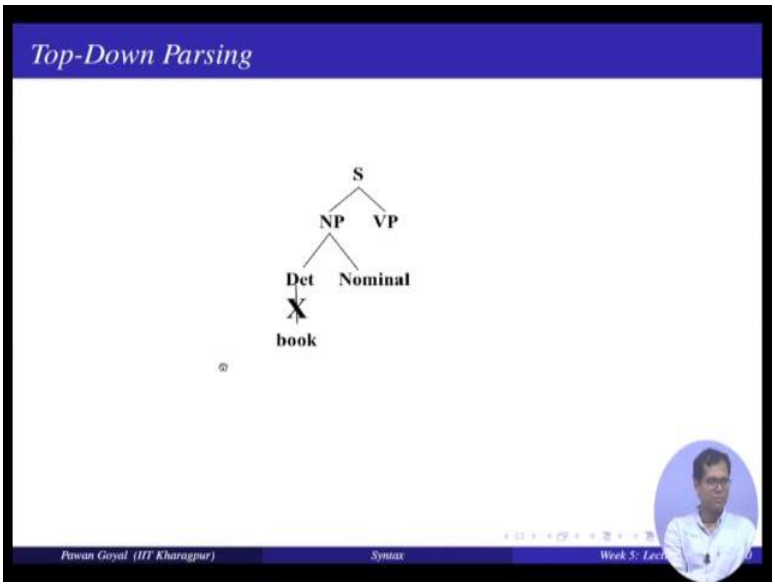
So a start by assuming that the input can be derived by S, then find out the trees that is start with S looking at the rules that are having on the left hand side because what are the

all are different things that S can derive. Now when you are going downward in your tree, once you obtain the part of speech category you will see if that matches the word in the leaf nodes. If it is not matching you will go back and try out some other path. And if there are any trees where the part of speech categories is not matching the words at the leaf nodes you will get (Refer Time: 09:19). So let us see. So I am starting with S.

Now what is my first rule? So what are the different rules from S.? So first rule is S can go to NP VP. So what I will do? I will try to explore that path. Scan got to NP VP. Now from NP what is the next possible rule? So next rule from NP is NP can give me a pronoun, NP give me a pronoun. Again I will try to explore this further. The pronoun gives me the pronoun will be the first word. Now pronoun is the pre-terminal.

Now what is the first pre terminal book? So pronoun can never give me a book. Book is not a pronoun. So this part is not correct. So I will go back from NP I will try to expose some other path. So NP can also give me a proper noun. So you see these are the rules in sequence in my grammar, a NP can give me a pronoun NP can give me proper noun. So again proper noun is a pre terminal that cannot give me a book. So again I cannot accept this path. So I will go back. Next rule is NP gives me a determiner followed by a nominal.

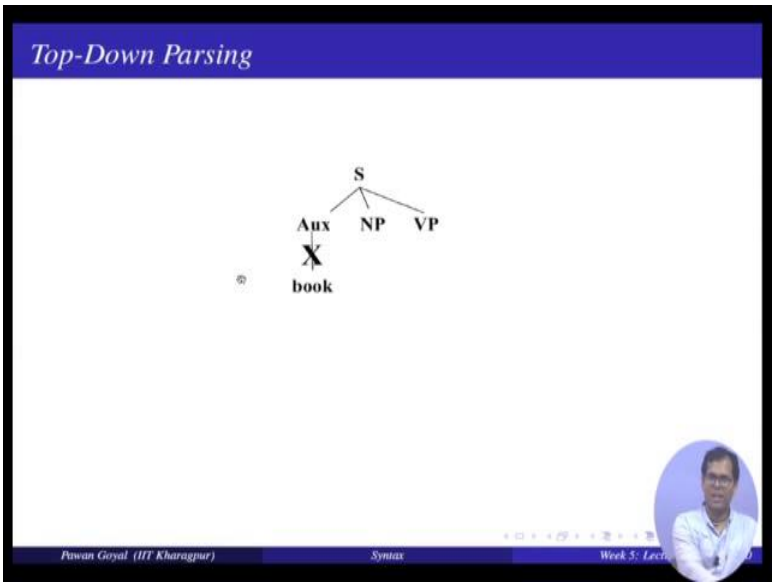
(Refer Slide Time: 10:36)



So again the first word has to be determiner that in the sentence, but the first word is book that is not a determiner. So again this path has to be this path is not correct. Now

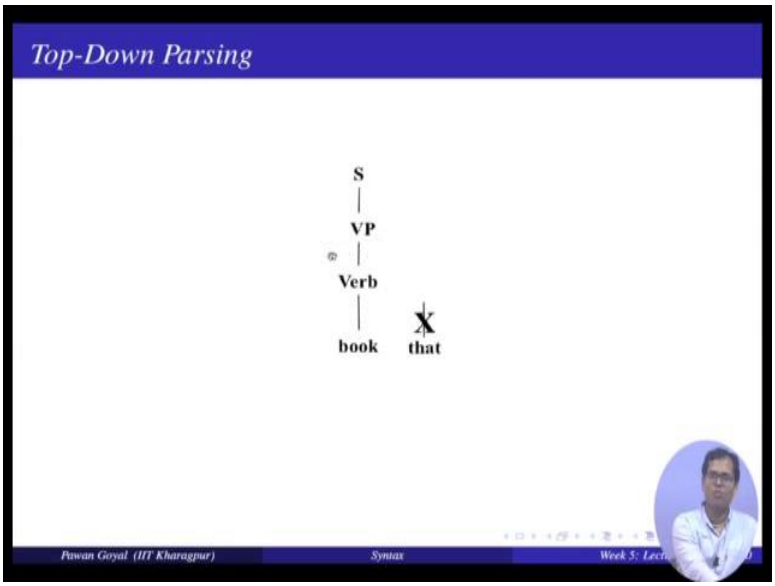
we see in the grammar there is no other rule that has NP on the left hand side so; that means, I have to now go back to the initial assumption that S will derive NP VP. So I have to try out the next possibility, as per my grammar the next possibility is S can give me auxiliary followed by noun phrase followed by a verb phrase. So let me do that.

(Refer Slide Time: 11:18)



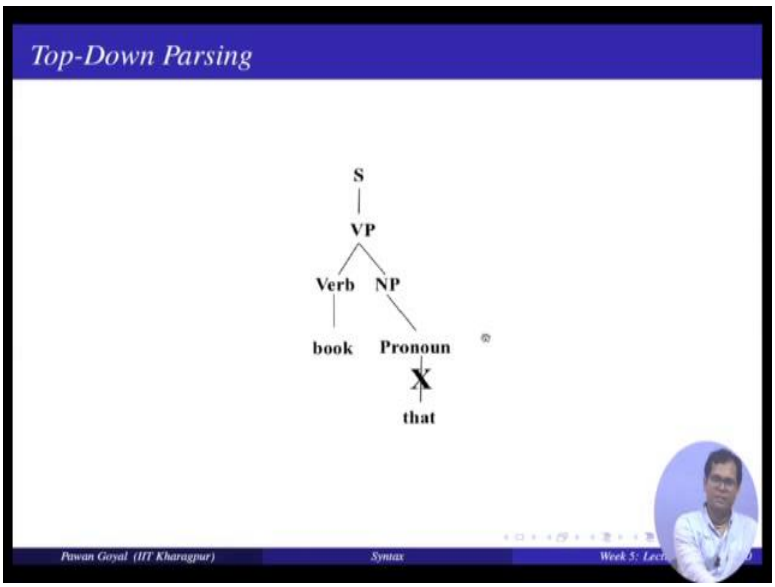
So S give me an auxiliary noun phrase and verb phrase. Again what is auxiliary? Auxiliary is a pre terminal that gives only does it does not give me book. So again this part is not correct and auxiliary gives me anything else. So again I go back and try out something else from S and the only remaining thing is VP.

(Refer Slide Time: 11:39)



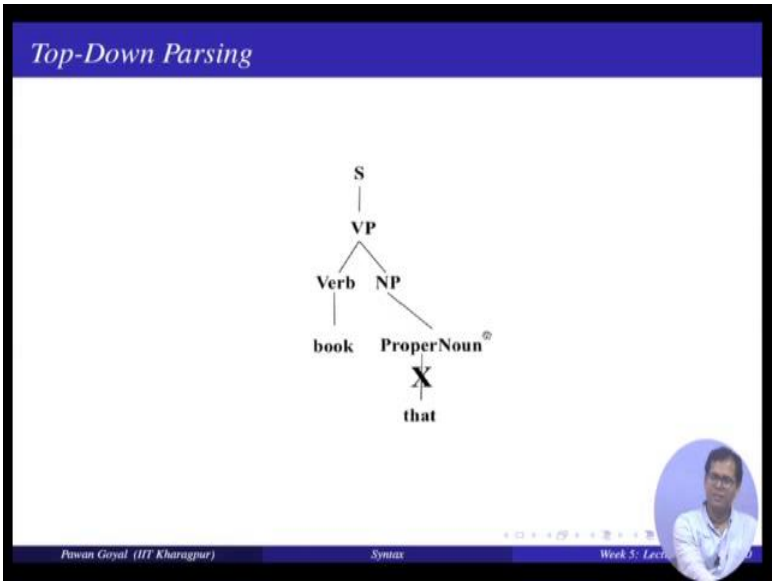
So S gives me a VP. Now I will go to my grammar. What are the rules from VP? VP can give me a verb that is the first thing, VP gives me a verb good. Now verb also gives me book. That that matches there, but what happens to the other 2 words in my sentence that flight. Verb gives me book, but that flight is not covered in this tree. So this is not a valid tree, yes that flight there is no other no node starting from S that captures this this is again not a valid tree. So I will try out something else with VP. So next possibility is verb followed by NP.

(Refer Slide Time: 12:22)



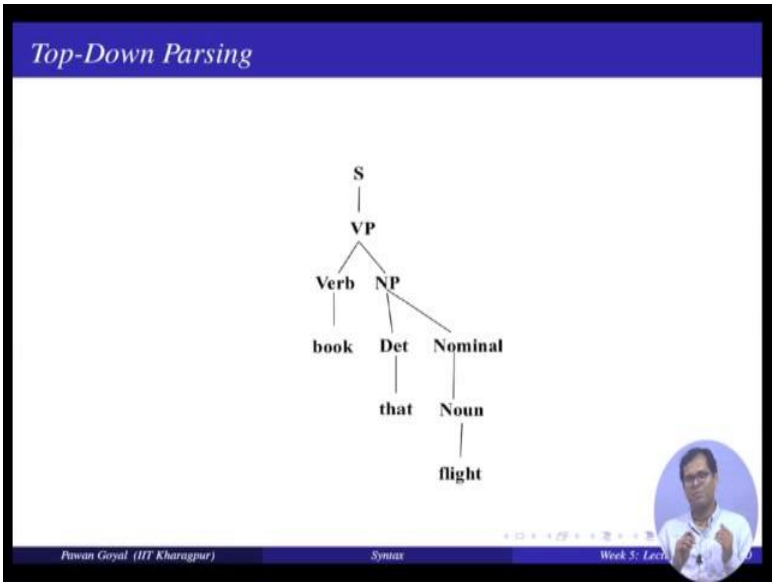
Verb followed by NP. And verb gives me book. Well now from NP I have to get that flight. Now again from NP I can get pronoun.

(Refer Slide Time: 12:41)



Pronoun cannot give me that yes. Then I can get proper noun. It will also not give me that. Then I can from NP I can get determiner followed by a nominal.

(Refer Slide Time: 12:47)

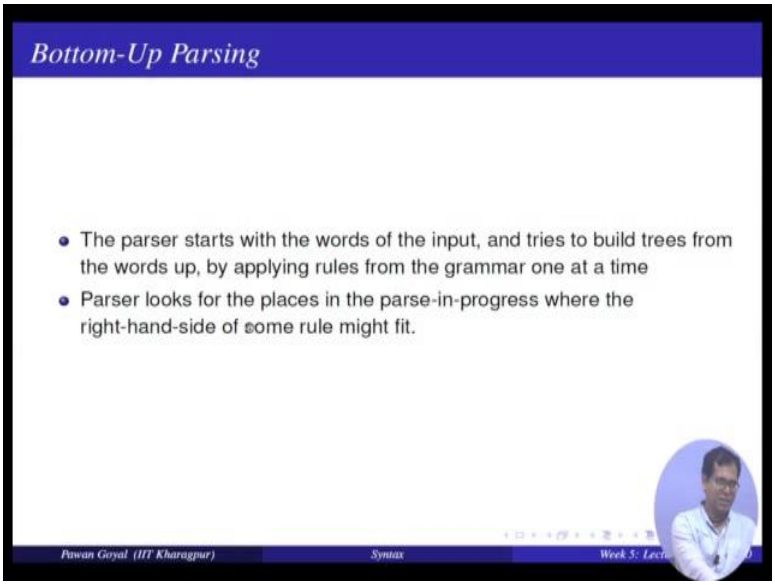


And that is a determiner and from nominal I can go to noun and this can give me flight. So that means, by doing all these explorations systematically I can come up with a parse

tree that starts from S and exactly covers these leaf nodes book that flight in my input. So that is my top down parsing strategies. So hope that is clear.

So we are given all the rules it starts from S and try to explore in some order. You can take it you can try to explore in the same order that is in which they are given to you. You might try to put them in the order in which they are actually used in in language. So which one is more probable than other that is also. But again you take can you see that this requires a far too many steps right. So you are exploring paths that will probably never lead to the whole parse trees. So this this this may need very certain visual space. So we will try to take that problem that how we can avoid that. So this is my a, this was my top down parsing.

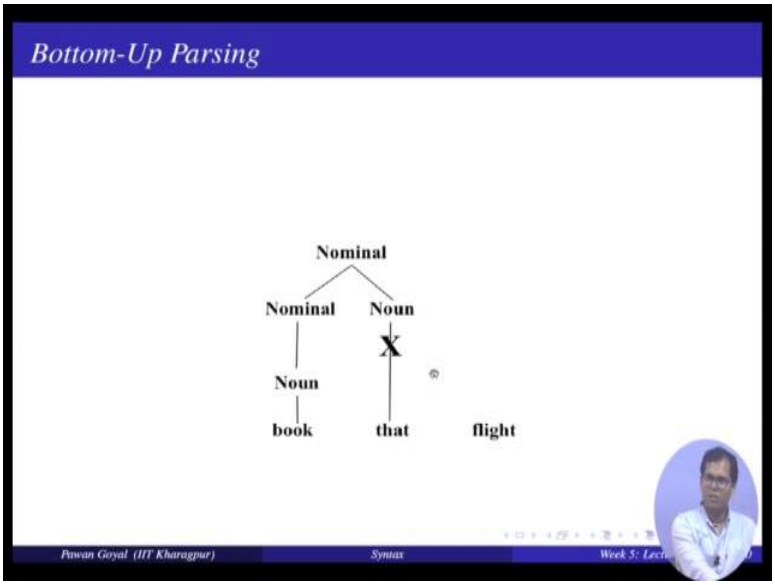
(Refer Slide Time: 13:59)



Now what do I do in bottom up parsing. In top up I start with S. In bottom up I will start with my leaf nodes. I will start with book that flight. And I will try to go my tree upwards and see which one can give me if complete tree it is starting from S. So the parser starts with the words of the input and tries to build trees from the words up by applying rules from the grammar one at a time. And parser looks for the places in the parse in progress where the right hand side of the rule might fit. In top down we were looking at always the left hand side, if the current non-terminal what is the rule in the left hand side. So that accordingly I will I am trying to generate the right hand side. Here I

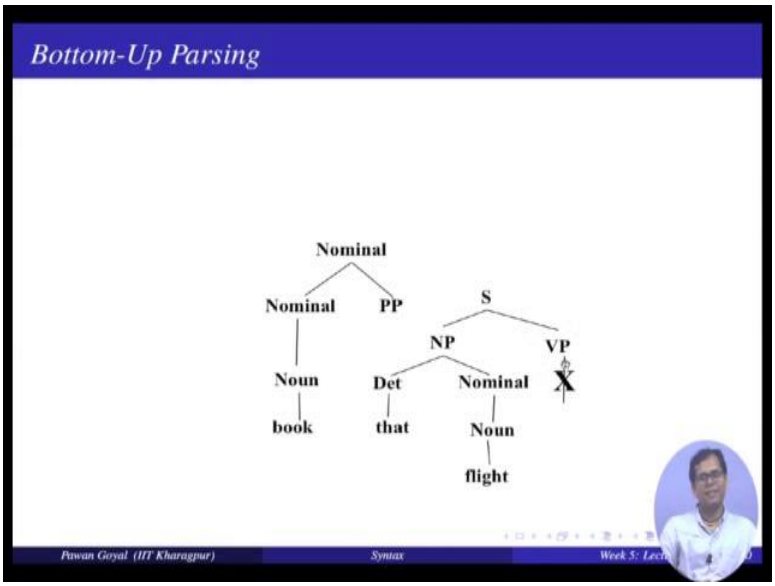
am seeing what in the right hand side I already have accordingly I select a rule in my grammar.

(Refer Slide Time: 14:52)



So let us do this bottom up parsing. So I have the sentence book that flight. I will start by seeking what are the nodes in my grammar that can that can generate this. So I start by say book. And the first rule that their word book is noun. Then I go up to noun and say nominal gives me noun, fine. And nominal can gives me nominal followed by a noun. So I am going my tree upward, but now I arrive at noun that is a pre terminal and noun will not give me that. So that is not a noun. So this is the inconsistency. So I have to go back and see.

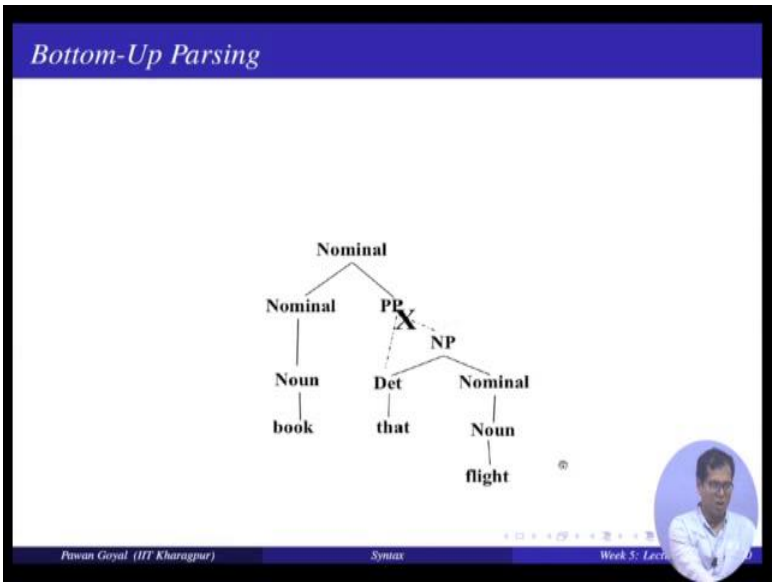
(Refer Slide Time: 15:38)



What is the next possible rule from nominal? So I have to derive from nominal followed by PP. So remember, we are seeing in which rule this occurs in the right hand side and accordingly I will take the production. Nominal gives me nominal followed by PP, now can PP give me that flight? So from PP, so from that I try to go it upwards I see determiner. Determiner comes in right hand side of this rule NP gives me determiner nominal. And flight again I grow it upwards it gives me noun a nominal can give me noun. So this looks a nice tree.

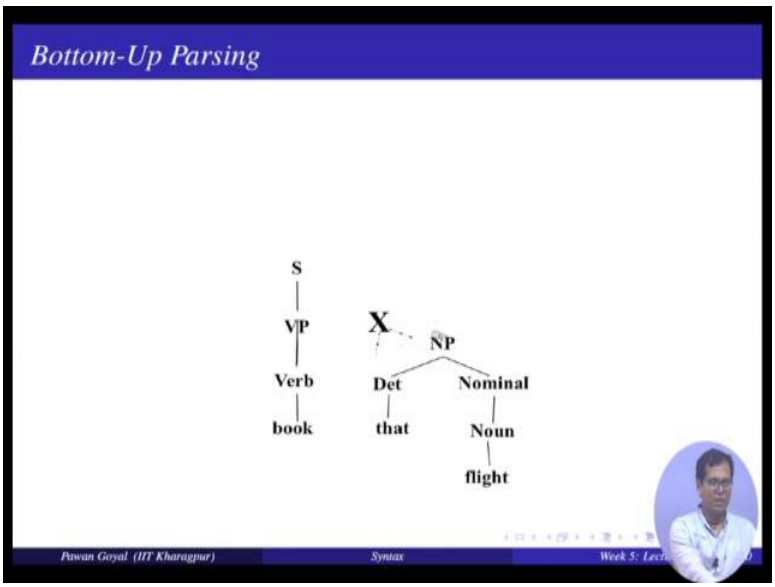
But now can I attach PP to NP. So S if I go NP upwards I will say S can give me NP followed by VP. This creates a problem in that this verb phrase does not have anything in. So it cannot take me to any leaf node.

(Refer Slide Time: 16:46)



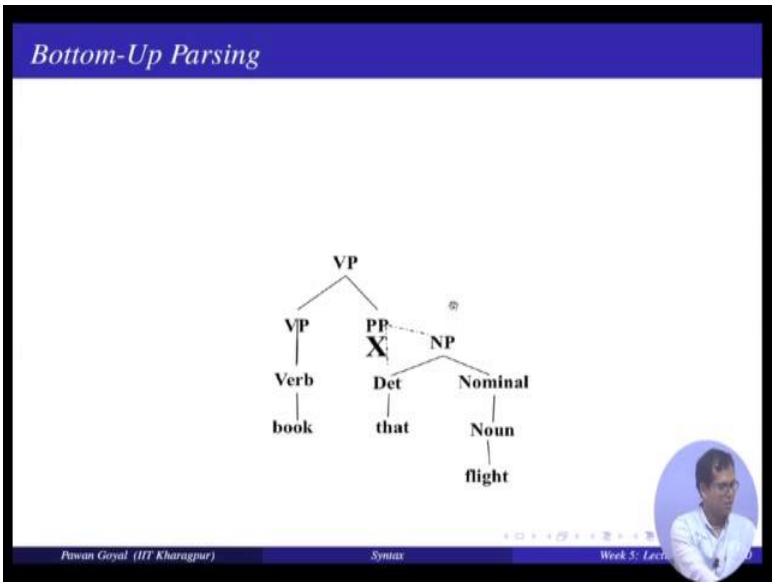
So this is not valid. If I go to this part can PP attached to n determiner and NP again there is no rule in my grammar that PP can be determiner followed by NP.

(Refer Slide Time: 16:58)



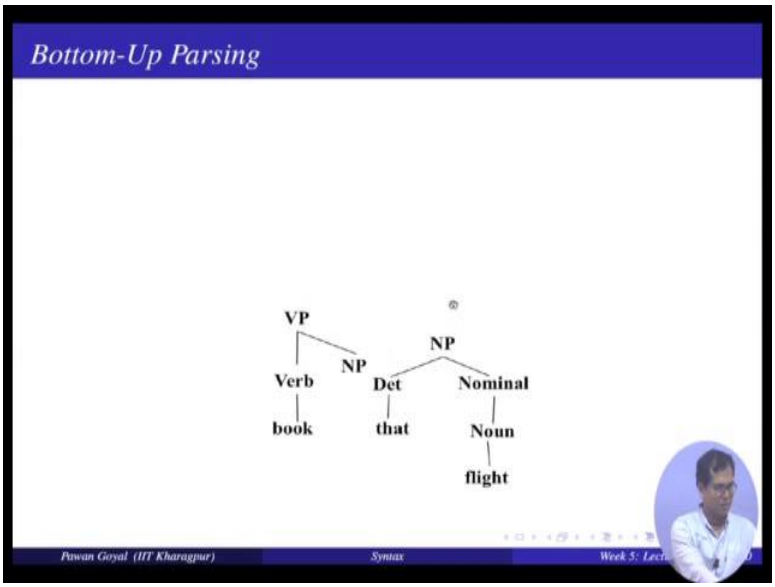
So I have to go back again. Try some other paths we have got on this is right hand side. VP S, S and determiner nominal and noun phrase, what can they give to me S again this does not work out.

(Refer Slide Time: 17:13)



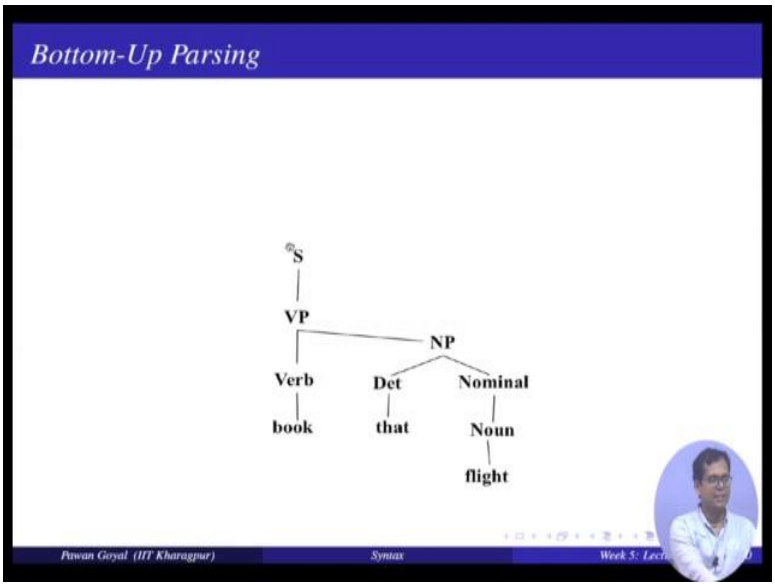
So I try out something else with VP. So I say VP gives me VP followed by a prepositional phrase. Again prepositional phrase does not give me determiner followed by a noun phrase, this we will not work out. Again go one step down.

(Refer Slide Time: 17:28)



So earlier I was saying VP is deriving a verb, now I am saying VP derives verb followed by an NP does that work.

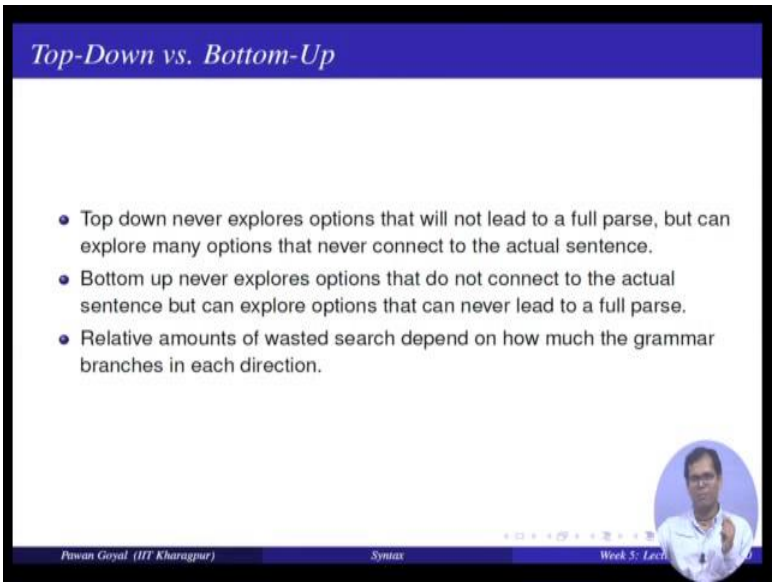
(Refer Slide Time: 17:40)



Yes, if I can attach this VP to this verb and this noun phrase. And then I can say that S can derive this VP and this gives me the whole parse tree. So this is my bottom up strategy.

I start with the words in the leaf in the leaves. Try to grow them upwards by seeing what is the rule in my grammar where this occurs in the right hand side. Similarly, I will see here and so on. And finally, can I build them as a single tree starting from S. And that strategy finally, gives me this particular tree. So if we just try to compose this top down versus bottom up approaches. So what do we see?

(Refer Slide Time: 18:25)

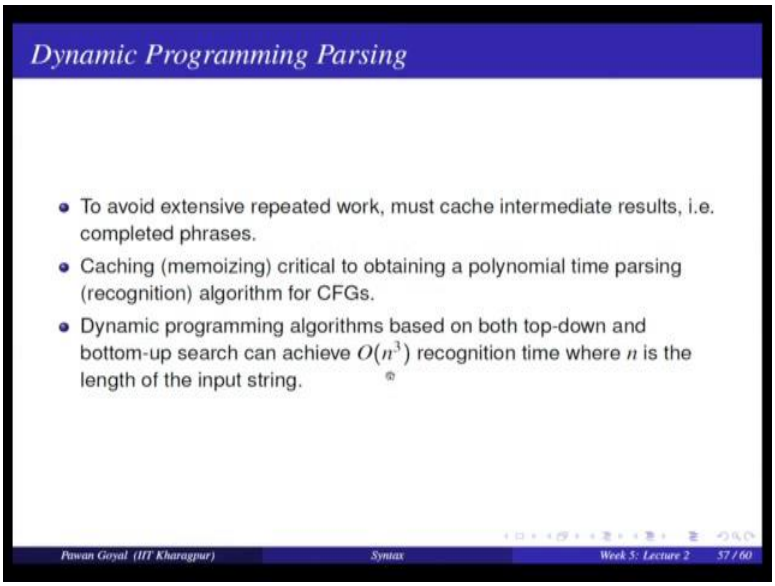


So what we were see in top down, we were always seeing wherever this non-terminal appears in the left hand side and I immediately do the production.

So what happens that I can x, I can always explore options that will never lead to the full parse. I can always find options that will only give me one word. So all the options were given me a parse is starting from S in in the top down strategy, but it might happened that some options are giving me a particular parse tree where all the words are not covered or some extra words are covered. So this actual correction the actual sentence is not taken care. So in the bottom up it is the other way around. So we have always seeing the words first you are exploring that those paths that are covering the whole sentence, but it might happen that the parse that you come up with is not a complete parse because it is not having a root at S. So they may not be a full parse. And there are ways to this specimen both top down and bottom up.

So you are seeing you are experiencing lot many parse that are probably not valuable and this depends on branching of my grammar in either direction.

(Refer Slide Time: 19:51)

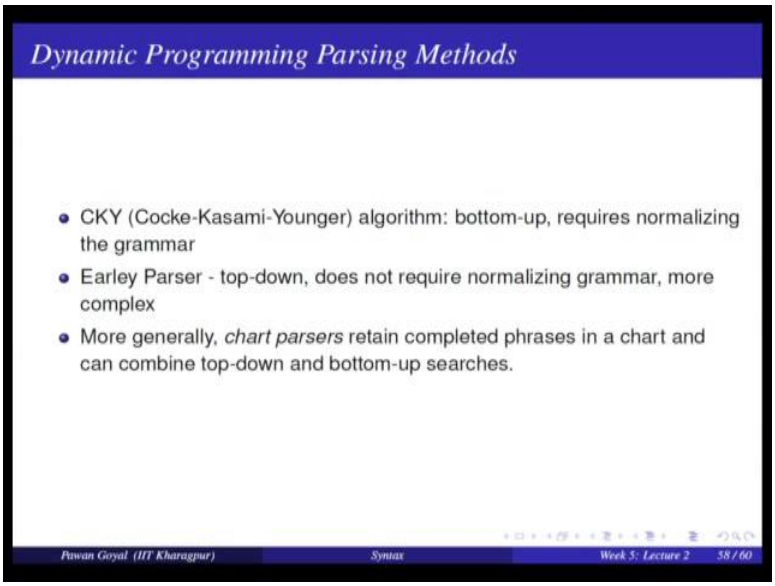


So to avoid this problem and obtain an algorithm that works in polynomial time. So you will use some dynamic programming approach. So we have been using dynamic programming a lot in this course. So we use that for edit distance then we use that for (Refer Time: 20:07) coding kind of algorithm.

Now, we will see how to use that for obtaining an efficient parsing algorithm. So idea is that can we cache some intermediate results instead of exploring all the different possibilities that are not relevant. So by doing this caching I can obtain a polynomial time parsing algorithm for context free grammars. And there are different dynamic programming algorithms that are both top down as well as bottom up and they can work in roughly order of n cube time where n is the length of the sentence number of words in my input string.

So what are the different approaches for dynamic programming parsing?

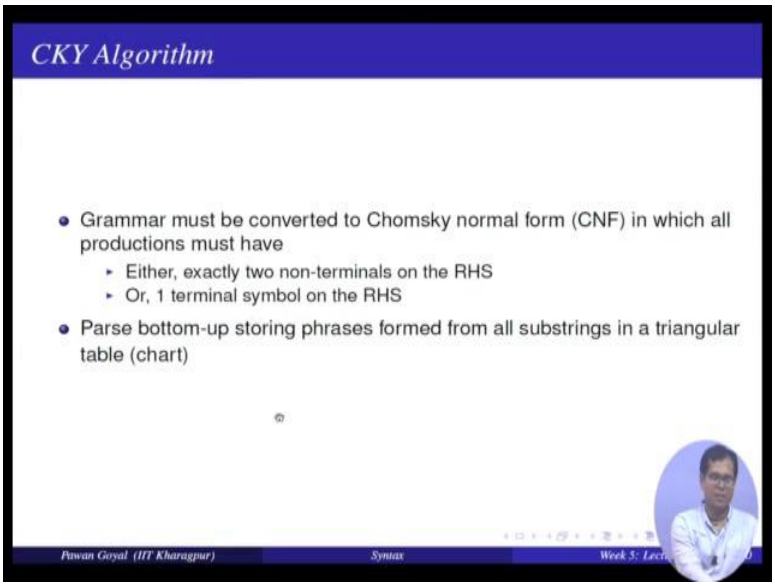
(Refer Slide Time: 20:53)



So, one very popular algorithm is CKY algorithm that works in bottom up manner. So you will do it for individual words then you are going to sequence of 2 words and so on. Up to you go to the whole 6 sentence. And this is the only thing is that it requires some normalizing of the grammar that you will also see what is the normalization. Then there is early parser that is again very popular that works in top down manner. It does not require any normalization of grammar and slightly more complex than the CKY algorithm. And a generic frame work is something called a chart parser where for individual phrases in the sentence they will see what are the possible trees they were retain in the chart and use that for the higher level trees. So and they combined both of these approaches bottom up and top down.

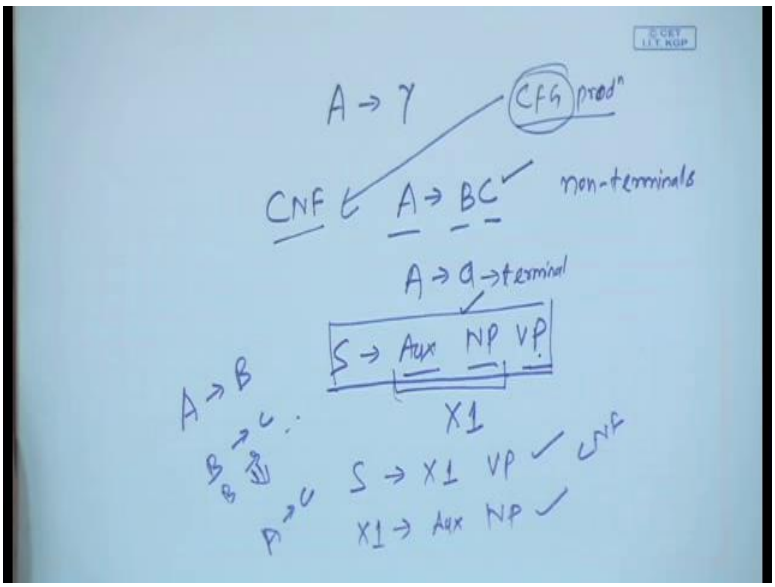
So we will only focus on CKY algorithm that how do we use this for finding out parser in efficient polynomial time. So how does this CKY algorithm works? Now, before that we have seen that this requires normalization of grammar. So what is that?

(Refer Slide Time: 22:14)



So to apply CKY algorithm, so my grammar must be converted to a normal form called Chomsky normal form. And what is the constraint in Chomsky normal form? So the constraint is that all the production of my grammar should be having one of these 2 forms. That is either exactly 2 non terminals on the right hand side or one terminal symbol on the right hand side.

(Refer Slide Time: 22:47)

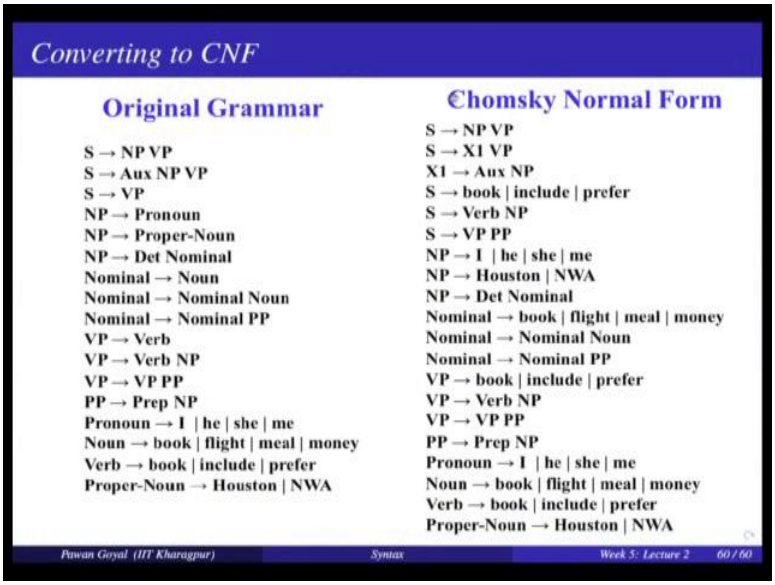


So what do I mean by that? In the context free grammars a rule is of the form A goes to gamma, where A is a non-terminal and gamma can be a sequence of terminals and non-

terminals. This is general CFC production rule. So what happens in the case of Chomsky normal form? The rules are constraint to be of these 2 forms A goes to B C. All these are non-terminals or A goes to small a small a is a terminal. So left hand side is always the same it is only one non-terminal. In Chomsky normal form what happens is that you put some constraints on the right hand side. So it can have either only 2 non terminals or only one terminal.

So to apply CKY algorithm, I must convert my grammar in any generic context free grammar form to Chomsky normal form. So what are the steps involved they are (Refer Time: 23:50) what are necessary for our case. So we will also see that how do we store all the possible phrases what are their parses in a triangular table in the CKY algorithm.

(Refer Slide Time: 24:10)



So let me quickly see let me quickly show how we convert a grammar to Chomsky normal form. So in the left hand side we have a grammar, and I want to convert that the Chomsky normal form. So can you quickly see go through the grammar and find out the rules that are not in Chomsky normal for, so all these are one non-terminal giving me a one terminal. Yes, in all these cases. So they are pre terminal to terminal, they are always in Chomsky normal form, yes, a non-terminal giving me a single terminal. Now let me go upwards. So PP gives me preposition followed by a noun phrase, again one non-terminal giving me 2 non terminals in CNF. This is also in CNF, this is also in CNF, but this rule is not in CNF. VP goes to verb one non-terminal giving another non-terminal.

So how do I actually convert this to Chomsky normal form? So idea would be I will find out what are the terminals that this derives. So verb derives book include and prefer. So instead of this rule I will add a new rule, verb phrase gives me book include and prefer and this is the strategy. Similarly, here this is in CNF, this is in CNF. What about this rule nominal goes to noun? Again I will find out what are different things that noun derives book flight meal and money and I will add a rule nominal derives book flight meal and money this is in CNF.

So here again NP goes to pronoun I will find out pronoun goes to I he she me. So NP gives me I he she me. S goes to VP is again not in CNF. So VP gives me verb and also verb NP. So I have to add these rules S goes to verb NP S goes to VP NP S goes to preposition NP, plus for VP going to be verb at now takes it next rule VP verb going to book include prefer and the rule I will add is S goes to book include and prefer, S going to NP VP is fine.

But what do I do with this rule S goes to auxiliary NP and VP. So for this kind of rule S goes to auxiliary NP VP where a single non-terminal derives pre non-terminals. So what do I do? I coin some new non-terminals, so I will say that auxiliary noun phrase together makes new terminals say X1. So what will be my grammar it will be S goes to X1 followed by VP and X1 will be auxiliary followed by NP, and this is equivalent to this rule. And now this is in my Chomsky normal form.

So idea is that whenever you have a rule where you have 2 more than 2 non terminals you try to break them down such that it is one non-terminal another non-terminal this you can again further break down if needed. And this is a simple approach and if we have a rule like this A goes to B and B goes C you take this 2 a goes to small c this is a terminal. And also if B has something else you need take care of everything in this yes that is what we took we saw in this particular example there.

So now if I convert this (Refer Time: 27:59) grammar to Chomsky normal form, this is how it will look like. So you see that here S goes to VP now has multiple rules. S goes to verb NP S goes to VP NP and S goes to book include prefer and so on for the other case also.

So in this lecture we had seen that how do we do a simple parsing using top down approach and bottom up approach, but they are not very efficient, so can we do

something better by using dynamic programming parsing approach. And for that we are trying we will be seeing CKY algorithm. So how do you use CKY algorithm for an efficient parsing, but CKY algorithm requires normalization to some Chomsky normal form and we saw how do we convert a grammar to Chomsky normal form.

So in the next lecture we will start with this Chomsky normal form and see given a new, given a string how do we parse that using CKY algorithm.

Thank you.