

**Natural Language Processing**  
**Prof. Pushpak Bhattacharya**  
**Department of Computer Science and Engineering**  
**Indian institute of Technology, Bombay**

**Lecture - 37**  
**Parsing**


Today, we will start a new topic, the topic of parsing, just before that we finished words and disambiguation, which was concerned with words their properties, their associations, their meanings, how they relate to the context words, after all the meaning emerges from interaction with the contextual words in the neighborhood, and typically the context is a sentence. Now, what is a sentence, a sentence is a sequence of words but is there anything more than the sequentiality, which is inherent to sentences. Sequential set of words forms a sentence, but this is hardly the necessary condition for forming a sentence, sentences have internal structure.

So, we take this topic of parsing, which is an extremely important field of natural language processing, and probably the most understood, the most investigated area of language processing. The investigations into syntax of sentences, led to other areas of computer science in particular the area of compilers and programming languages. Because it is clear to see that the sentences are at a high level when we work with natural language, we are interacting with other entities at a higher, so what is contained in a sentence, what is the inherent structure?

(Refer Slide Time: 02:17)

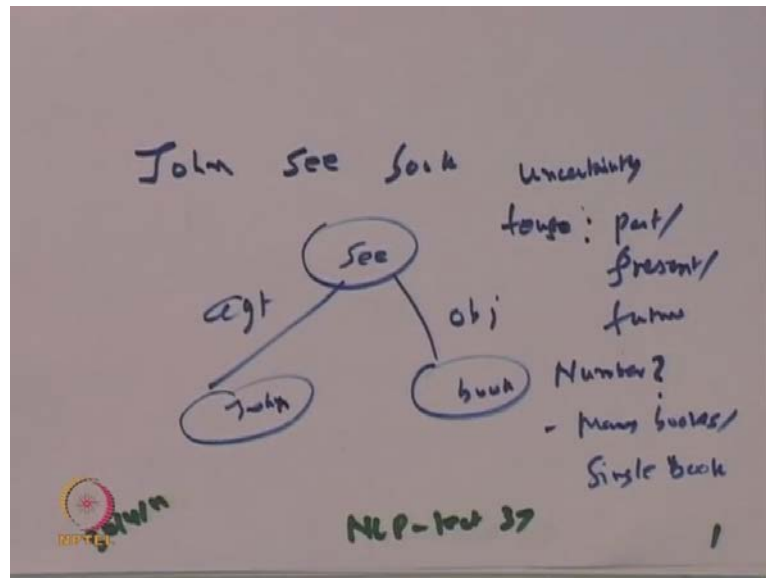
## Need for Parsing

- Sentences are linear structures, on the face of it
  - Is that the right view?
- Is there a *hierarchy*- a **tree**- hidden behind the linear structure?
- Is there a principle in branching
- What are the constituents and when should the constituent give rise to children?
- What is the hierarchy building principle?



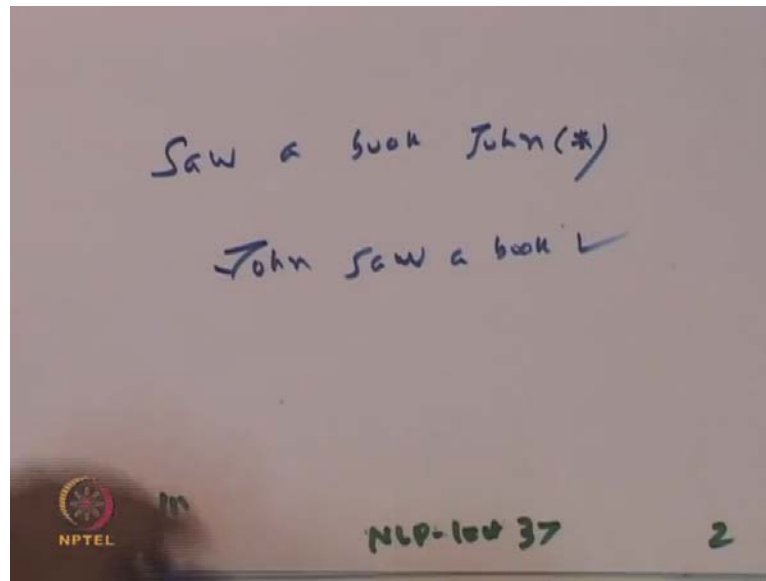
Let us go ahead with these material of the lecture, we ask what is the need for parsing, is parsing really needed, sentences are linear structures on the face of it. But is that the right view? Is there a hierarchy a tree hidden behind the linear structure? Is there a principle in branching? What are the constituents and when should the constituent give rise to children? What is the hierarchy building principle? These are very fundamental questions with respect to the structure of a sentence, now it is also a common observation that, even when the sentence is not grammatical, the meaning is still made, I give a sentence here.

(Refer Slide Time: 03:05)



John see book, so this is not a grammatical sentence, there is a seeing activity here, there is a agent of seeing, which is john and there is the object of seeing, which is the book. But point is there is still uncertainty about the tense information for example, tense is it past or present or future there is uncertainty with respect to that, then there is these number information, many books single book. So, there is incomplete specification of information, when we look at the sentence. But the meaning is still understood. I write a sentence with correct word forms.

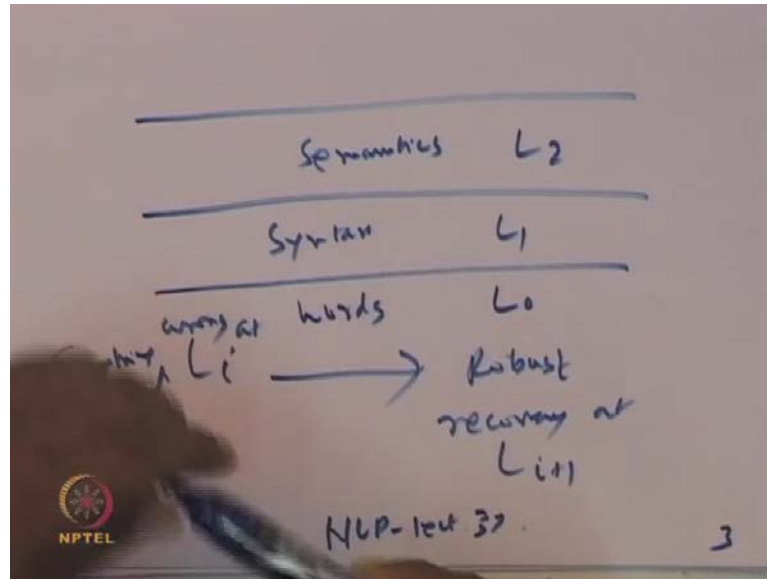
(Refer Slide Time: 04:32)



Saw a book John, this is a wrong sentence as indicated in linguistics or nlp by a star mark, when there is a star mark beside a sentence, it shows that there is something wrong with the sentence, the correct sentence would be John saw a book, this is correct. So, again there is a seeing activity, which takes place in the past, and the object being seen is book, their seer is agent. Now, even when the sentence order is wrong, saw a book john, even being can still make out the meaning, and looks the order is not so sacrosanct after all. But that is not really true, because when we are able to understand the sentence, John or saw a book John, we are invoking actually a higher level knowledge, we are invoking semantics. So, even though the order is disturbed, we can make out that john is the seer, because john is animate, possessing the ability of seeing, the book does not have the ability of seeing.

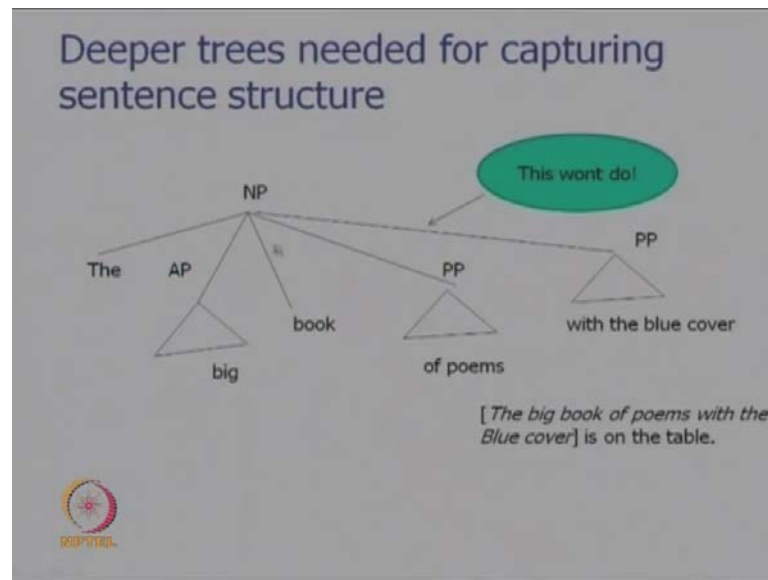
So, when the order is disturbed of a sentence, we have to resort to deeper level of meaning to understand the sentence. So, the cognitive load becomes higher, we sacrifice the order or we sacrifice syntax. But this comes at the cost of more challenging and difficult processing, namely the semantic processing, and notice this fact carefully, that if a language processing entity or in information processing entity does not have, this backup mechanism, does not have this robustness or does not have the layers of intelligence, inbuilt. Then it will fail, when something goes wrong at a lower level, so a picture can depict this fact.

(Refer Slide Time: 07:18)



So, we have the level of words, the level of syntax, the level of semantics, whenever anything goes wrong at a particular level, whenever anything goes wrong at  $L_i$ , something wrong at  $L_i$  would lead to robust recovery at  $L_{i+1}$ . So, something going wrong at the level of syntax, semantics will be invoked, to correct that error. But the point is that why should, we have this cognitive load, why should we have this extra processing, if syntax can be ensured. So, we move now on to this question of need for parsing, sentences look like linear structures, they look like a linear sequence of words but they are definitely is a structure behind them, as our discussion will show.

(Refer Slide Time: 08:42)



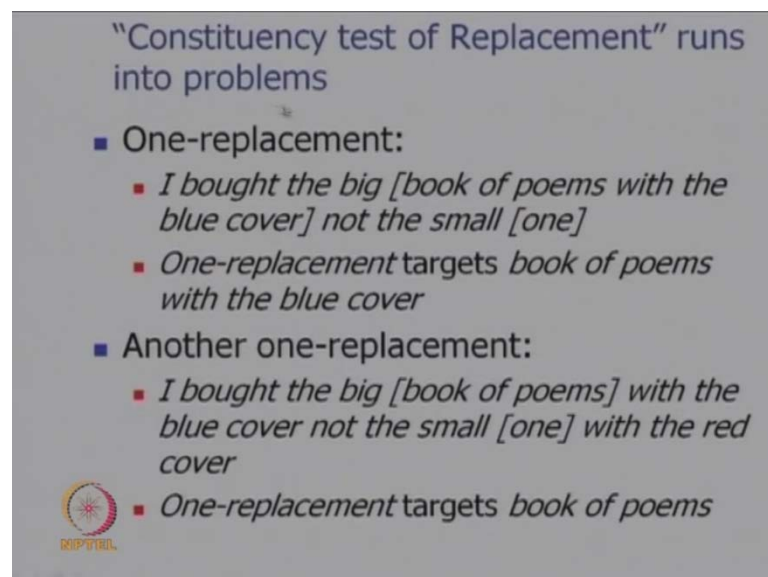
We take up this sentence; the sentence is the big book of poems, with the blue cover. So, it is easy to see here, that the most important entity in this phrase is book, so this is the head of the phrase. It has a special role to play in the phrase, this is the most important entity in the phrase the book, and everything else is a qualifier for this entity. What kind of book? What is the size of the book big book? Which book the book so the indicates, that there was a previous reference in the discourse to book, and the as a definite article comes to signify this, of poems is a qualifier for the book, what kind of book, book of poems, and with the blue cover is another qualifier for the book.

So, all these are modifiers, qualifiers for the head book, the big of poems with the blue cover all of them qualify book. So, that is all probably, it is possible to naively put these qualifiers at the same level, of poems with the blue cover, the big all of them are qualifiers, for the book. And therefore, they are at the same level, but this would not do the structure is not really this, all the qualifiers are not at the same level.

So, let us just reflect for a few minutes on this tree and see, if there is some amount of larger question or larger binding or larger attraction between the head and some of its qualifiers. So, book of poems with the blue cover or you can see that, because of the proximity or adjacency of poems, and with the blue cover, it is possible to say that poems were with blue cover. When this is said, semantics will immediately get activated, and say that no, this is a nonsense interpretation, this attachment will not hold.

So, with blue cover is actually attached to book but in between there is a piece of text of poems, and therefore, we need, a machinery by which book of poems, becomes a single entity, and blue cover is a modifier for that entity. Again big and of poems are there at the same level, there is not much harm in doing that. But still it is good to depress book of poems deeper in the tree, and have big qualify that entity, with blue cover also should qualify that entity. So, there is a theory which governs this phenomena, this is c command and dominance, and prepositions being at the same level is called a flat hierarchy with respect to the head word. But we know that this structure is not correct, there are many evidences for this.

(Refer Slide Time: 12:28)



"Constituency test of Replacement" runs into problems

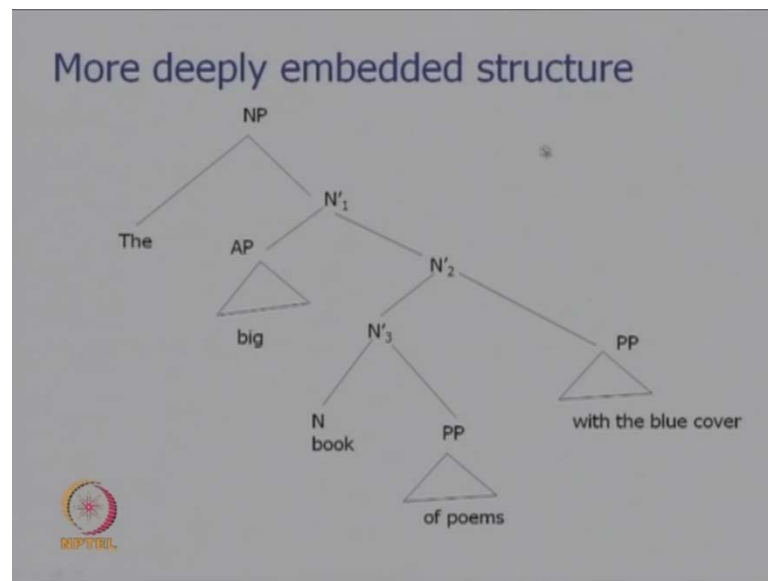
- One-replacement:
  - *I bought the big [book of poems with the blue cover] not the small [one]*
  - *One-replacement targets book of poems with the blue cover*
- Another one-replacement:
  - *I bought the big [book of poems] with the blue cover not the small [one] with the red cover*
  - *One-replacement targets book of poems*

One of the evidences is what is called the Constituency test of Replacement, and when we apply the Constituency test of Replacement. It runs into problems, there is this phenomenon of 1 replacement I bought the big book of poems, with the blue cover not the small one. So, if I introduce an anaphoric one, what does this one bind to, how much of text does it, point I bought what, I bought the big book of poems with the blue cover, not the small one.

So, this one binds to the whole sequence of words book of poems, with the blue cover not the small one. So, we express this by saying that one targets, book of poems with the blue cover, another 1 replacement is shown here, I bought the big book of poems with the blue cover, not the small one, with the red cover .So, the moment we introduce with

the red cover 1 targets book of poems, not book of poems with the blue cover. The amount of text 1 points to changes, now to a smaller region book of poems. So, here 1 replacement targets book of poems, so the question is why does this behavior emerge, why is this behavior shown, to explain this, we have to propose a deeper structure, behind this sentence, and make use of syntactical considerations.

(Refer Slide Time: 14:31)



So, the tree which is actually the right representation for the inherent structure, in the sentence is this, the big book of poems, with the blue cover is actually a noun phrase, and in the noun phrase, there are these bar entities N 1 bar, N 2 bar, N 3 bar or N 1 dash N 2 dash, N 3 dash, traditionally there are called bars, and comes from a theory, due to called x bar theory. So, these nodes are to be introduced, to give the tree the correct structure, so this NP is composed to the and N bar, this N bar or noun is big book of poems, with the blue cover, N bar N 1 bar is again not flat. It is composed of big and N2 bar, N2 bar is not again flat N2 bar is composed of N3 bar, and PP so big N2 bar N3 PP, N 3 bar has N as book, and a preposition phrase of poems, so book of poems is N3 bar.

So, this shows that book and of poems, have strong affinity for each other, they bind together, and they produce a structure N 3 bar. This N 3 bar is now modified by with the blue cover, and it produces N 2 bar, N 2 bar is now modified by big, which produces N 1 bar, N 1 bar along with the produces the noun phrase, where interested in the big book of



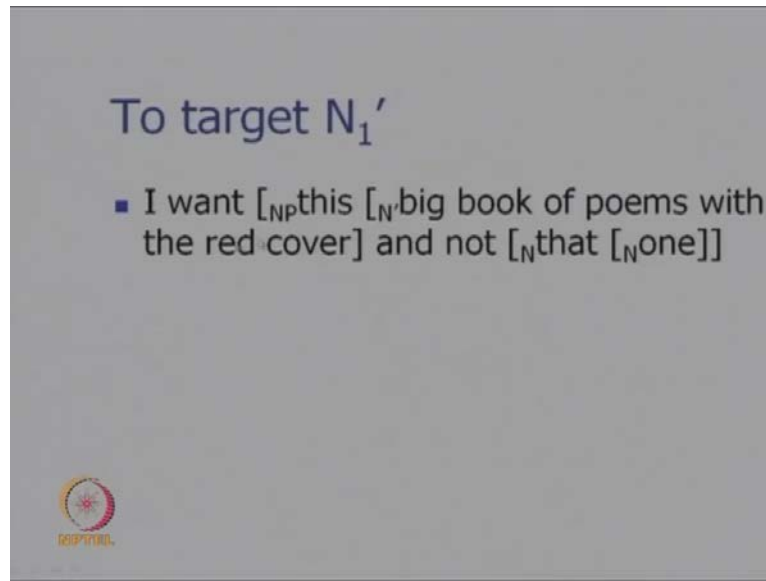
poems, with the blue cover. So, this is the tree structure, which is operative behind the sentence, and it is these, tree which is needed to explain the 1 replacement.

Now, to target N1 bar N1 bar is these, to target N1 bar we have to disturb the internal structure of these, we have to part up this structure, and part of the structure, and we have to disturb one of the constituents of N1 bar, what is available is these as a single entity, and there by the whole thing will be targeted, yes will come back to that. Now, how do you explain this phenomenon, I want the big book of poems with the blue cover, not the small one, so 1 targets this whole N2 bar, what is the reason, the reason is that, big has been replaced by small. And now when we fill out the text for N2 bar for 1, when we have to resolve this, anaphoric reference then we have to bring in this whole piece of text in to one. So, I want the big book of poems, with the blue cover, not the small one.

One will take up this, whole N2 bar where was this other example; I want the big book of poems with the blue cover, not the one, with the red cover. So, what will one refer to now, what does one target, again the, this deep tree comes to our help, with the blue cover is here, and it is disturbed, with the blue cover has become, with the red cover. So, one will target now N3 bar, so principle is that, whichever sub tree is considered, and a child of the sub tree is part up to say or to substitute it, the other child becomes the target for one, I want the big book of poems with the blue cover, not the one with red cover.


So, one with red cover is book of poems, so we have seen how to target N2 dash, how to target N3 dash, now we may ask how to target N1 dash. So, what can we disturb here, what we can disturb is the, so the sentence could be, I want this big book of poems with the blue cover not that one. So, what is happening is that the, is disturbed, and replaced with these, so replaced with that, so N1 dash becomes the target for one.

(Refer Slide Time: 20:14)



To target  $N_1'$

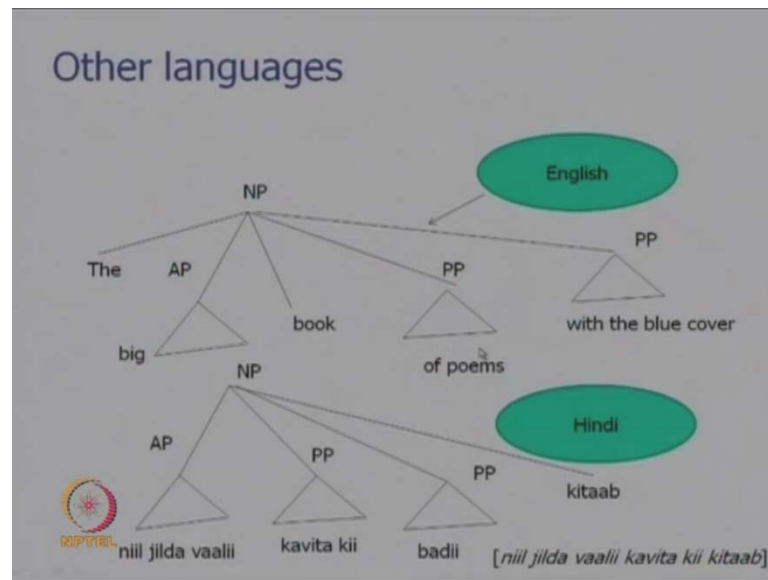
- I want  $[_{NP}$ this  $[_N$ big book of poems with the red cover] and not  $[_N$ that  $[_N$ one]]



This is shown here, I want the big, this big book of poems with the red cover, and not that one. So, the moment you establish correspondence between this and that, one will have to have correspondence, with big book of poems, with the red cover. So, this shows that, there is something to be gained by, constructing these kind of deep parse trees. And if you propose a flat structure for the sentence, we are going to miss, the internal structure of the sentence, and also we will not be able to explain some phenomena.

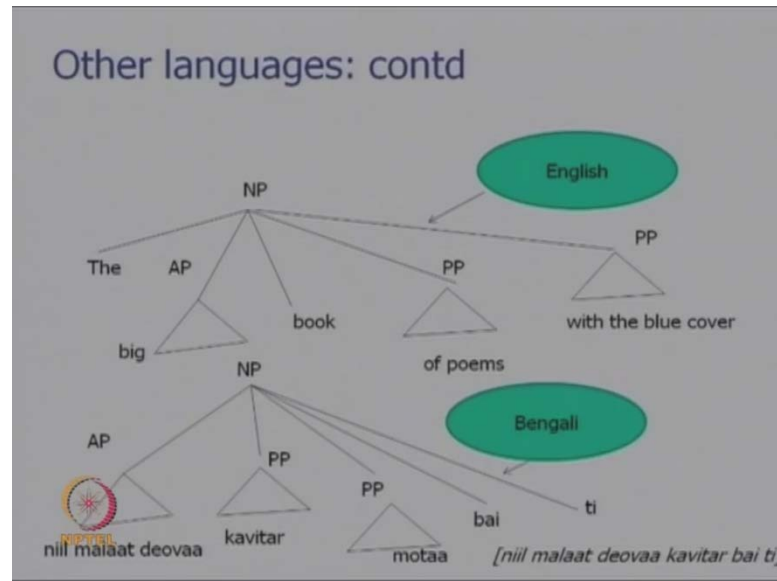
Here the phenomenon is that of do replacement, when we say John laughed heartily, and so did Jill, John laughed heartily, and so did Jill, then what does did target, did is an anaphoric entity, it has to target a piece of text coming before it. So, it has to target laughed, and similar kind of phenomenon, can be found for do replacement too. So, I leave it as an exercise for you to think about, how adverbs and adjectives, also could be replaced anaphorically.

(Refer Slide Time: 21:59)



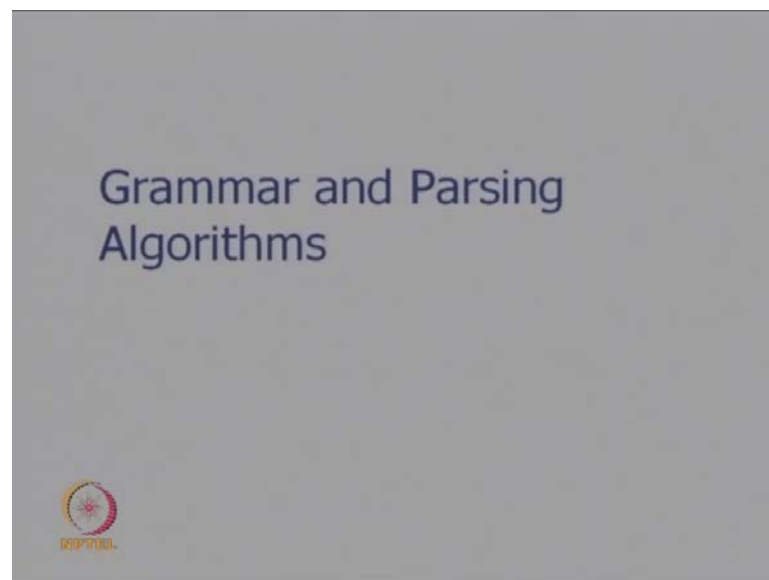
So, it is not the case that only English displace this kind of deep structure behavior, all languages have these phenomena. So, the big book of poems with blue cover, we have written it in Hindi is with blue cover of poems big book. So, Hindi has the modifiers all coming before the head, unlike English where some modifiers come before, some after. Here of course, the proximity of the modifier helps to create constituents.

(Refer Slide Time: 22:54)



so if we look at the structure for Bengali, it has similar structure as Hindi, [fl] t is called as classifier, and the modifiers all come before, the head except for t, which is a classifier. And again one can discard, this flat hierarchy and create a deeper tree.

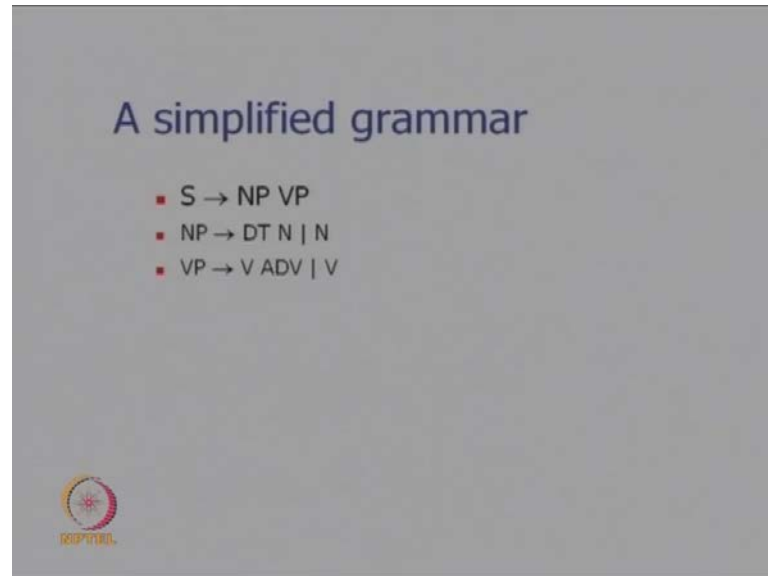
(Refer Slide Time: 23:23)



So, this discussion was to show, that sentences indeed have structure concealed, within them what looks like linear sequence of words, is hardly linear. It is a deep parse tree and many language phenomena cannot be explained unless, we propose a deep parse tree,

from this now, we move on to algorithms for parsing and grammar for parsing which controls algorithms, Grammar and Parsing Algorithms.

(Refer Slide Time: 23:23)




We will have our discussion with these Simplified grammar, sentence goes to noun phrase and verb phrase, noun phrase goes to determiner noun or noun verb phrase goes to verb and adverb or so for all these rules, we can find example sentences. But the meaning of this is sentence is composed of noun phrase, and verb phrase, noun phrase is composed of a determiner noun or it could be a single noun, verb phrase is composed of a verb and adverb or it could be a single verb, the meaning of bar is or so noun phrase is a determiner noun combination or a noun.

(Refer Slide Time: 24:52)

### A segment of English Grammar

- $S' \rightarrow (C) S$
- $S \rightarrow \{NP/S'\} VP$
- $VP \rightarrow (AP+) (VAUX) V (AP+) (\{NP/S'\}) (AP+) (PP+) (AP+)$
- $NP \rightarrow (D) (AP+) N (PP+)$
- $PP \rightarrow P NP$
- $AP \rightarrow (AP) A$



Here is a much more complicated grammar, the grammar which was shown before is a very simple fragment of English this, is a more complex description of English. So, there is a complimentary and sentence, then a sentence can be embedded sentence and a word phrase or it is noun phrase followed by verb phrase, which is more common word phrase can have, any number of adjective phrase, and auxiliary verb, verb and AP plus actually is adverb verb phrase.

And there could be adverb verb phrases more than one then there could be a complete sentence followed by again adverb verb phrase preposition phrase, adverb verb phrase. So, this is a more complex description of the language, and we need this kind of powerful grammar, if we have to handle all the complex phenomena of a natural language. Now, however we will be working with this simple grammar, which we showed, because our goal is to understand the algorithms.

(Refer Slide Time: 26:10)

Example Sentence

People laugh

1 2 3

These are positions

Lexicon:

People - N, V

Laugh - N, V

This indicate that both Noun and Verb is possible for the word "People"

We take the simple sentence, People laugh now since we will be discussing algorithms, now it is very important to mark these numbers 1 2 and 3, the whole parsing algorithm is controlled by these between word numbers. They are extremely important for controlling the flow of the algorithm, 1 is for the position before the first word, 2 is for the position between the first and the second word, 3 is for the position after the second word and so on. These kind of position convention is very, very common, in parsing algorithms, and let us note them carefully, parsing is surely controlled by a lexicon or a dictionary consisting of a set of words, and their parts of speech or categories grammatical categories, people can be noun or verb laugh, can also can be noun or verb.

So, this comma indicates that 2 different grammatical categories are possible, for both the words, so what would be the example sentences, in People laugh, People is noun, laugh is verb but if you would say the house was peopled with, many old persons, the house was peopled with many old persons, in this case people is a verb, that is why V is shown here, he had a good laugh. Here laugh is a noun, so People laugh though the sentence is small, in terms of parts of speech, many different combinations are possible, however because of the mutual disambiguation, people will ultimately be noun, laugh will be verb.

So, suppose for the moment that we do not worry about grammatical categories, the parts of speech disambiguation has not been done. If you run a parts of speech tagger, over

People laugh then what will happen is that, People will be marked as noun, laughter will be marked as verb, and these will not cause any problem in the parsing stage, where all the words have got their grammatical categories.

(Refer Slide Time: 28:58)

### Top-Down Parsing

State	Backup State	Action
1. ((S)1)	-	-
2. ((NP VP)1)	-	-
3a. ((DT N VP)1)	((N VP) 1)	-
3b. ((N VP)1)	-	-
4. ((VP)2)	-	Consume "People"
5a. ((V ADV)2)	((V)2)	-
6. ((ADV)3)	((V)2)	Consume "laugh"
5b. ((V)2)	-	-
6. ((.)3)	-	Consume "laugh"

Termination Condition : All inputs over. No symbols remaining.  
 Note: Input symbols can be pushed back.

First, we illustrate top down parsing, in top down parsing what happens is, that the state and the position of the pointer for the input string is maintained. So, this shows that S is the start symbol and 1 is the position of the input pointer, so the input pointer is here initially, what is the meaning of this, the meaning is that the star symbol S is waiting to be expanded, to get the other non terminals, and 1 is the position of the pointer, which means it is looking at the first word, and waiting to make decisions based on the first word. So, a stack is maintained, stack is the right data structure S is expanded, we get NPVP noun phrase, verb phrase and the position of the input pointer is still 1. Because no word is consumed after this NP is the first non terminal, which is waiting on the stack.

It is expanded to DT and N determiner noun, and verb phrase nothing happens to it, input pointer is still at position 1, however we take another action here, an additional action which is the other production, noun phrase is going to N is also brought in by expansion, and the verb phrase is of course, there. So, these an alternate state, which the machine could have gone to provided it, had expanded NP as N, because NP is extended expanded as DT N, this is the state which has resulted.



Now, we cannot do anything, because the stack says that, we need a non terminal DT data terminal is needed but what the input pointer is looking at is people, people is not a terminal it is either noun or verb. So, this expansion or this resolution by the word cannot happen, so the machine the so the machine it takes the alternative state, which would be used was NVP, so that is brought on NVP and the input pointer is at 1. Now, the non terminal which is waiting to be resolved is N, people is both noun and verb, so it can play the noun role here.

So, people will be consumed, the input pointer will move to 2, and the non terminal N will disappear, saying that it has found a word of type noun. So, now VP gets exposed the input pointer is at 2 it is looking at the next word, which is laugh position number 2, this is position number 2, now the input pointer is looking at laugh, VP cannot as such be resolved by laugh, so VP will be expanded VP is expanded to V ADV verb and adverb, no input is consumed, so the input pointer stays at 2, and the backup state is maintained, where VP goes to V and 2.

Now, there is laugh in front of input pointer, and V is a non terminal which needs to be expanded, so laugh is consumed input pointer moves to 3, exposing the non terminal ADV but however people was consumed, laugh is also consumed. There is no more word but there is a non terminal ADV something is wrong, what is wrong the stacks has not reached the S symbol but the input stream of words is over. So, in this case what will happen is that, it will bring the backup state V 2, so V 2 is brought here, now laugh is consumed by V, the input pointer moves to 3.

There is no more symbol on the stack, there is no more symbol on the input tape is the condition for termination. So, all inputs should be over, no symbols remaining on the stack, and here the sentence has been parsed correctly, even though on the way there were 4 steps, which needed undoing and bringing the backup step, backup stack, backup state, rather so the alternatives are maintained, and their explore so this is top down processing. So, what is the essential idea, we start with a symbol, we bring the non terminals by expansion, for any non terminal that is expanded, if it had another way of expanding it.

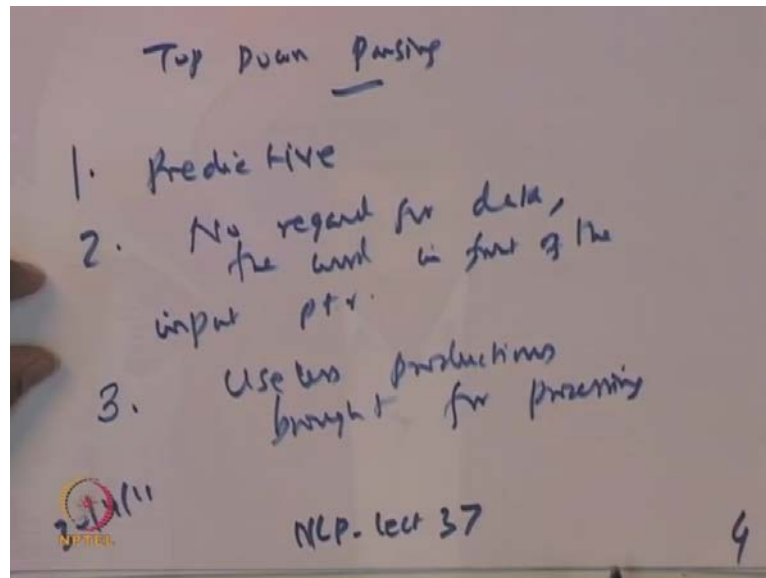
Then that backup state is maintained in weight, and if we run into a dead end, while processing, we resort to the backup stack, and bring the backup states. Now, how can we

take a false step, let us examine that question, we take a false step when there is a particular grammatical category, waiting either to be expanded or to be resolved, with what on the input. So, when the symbol, the non terminal with a particular grammatical type, cannot be matched with a word in front of the input pointer, this is a case of temporary failure.

So, in this case we may have to go deeper into the parse tree, developed so far or we may have to backup a long distance, to obtain the backup states, to explore and alternate path. Now, if you look at the algorithm carefully, you might have noticed that, the processing the syntactic processing, proceeds without any regard for the data, what is the data, that is in front of the input pointer, what data should I resolve, what is the type of the data, no such consideration is given to top down processing. This algorithm is called top down processing, because we start with the non terminal symbol S and go on expanding it, until we have been able to assign a word or a group of words to a grammatical category.

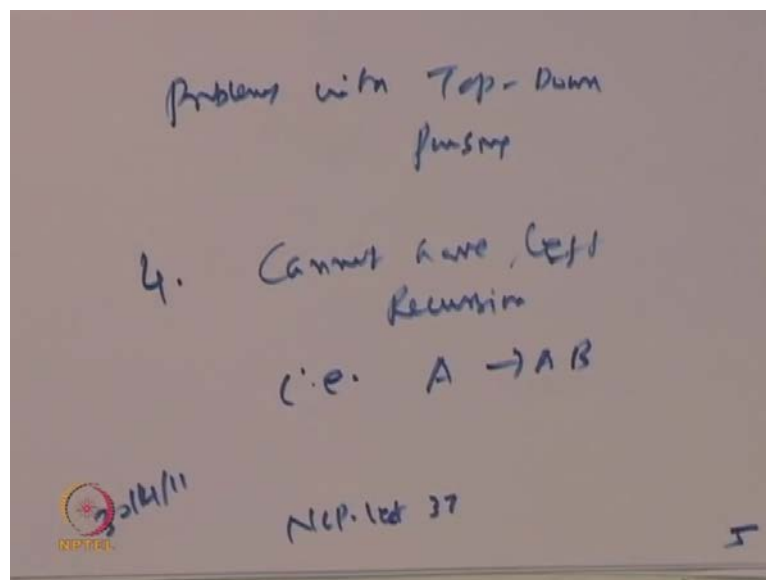
And finally, the whole sentence is over, all the input is consumed, this is known as top down parsing, this is also called predictive parsing, because whenever we expand a non terminal. We expect that, the next word would be of the type, of the non terminal, which comes under stack. So, NP goes to DT N this means, that I expect a determiner to come on the input string, and I predict or I expect it, without any regard to the data, without any regard to whatever data, I am now seeing or likely to see or have seen before. So, this is a very simple minded strategy but it can give rise to a very elegant parser, in particular prolog has an eyes way of implementing, top down parser. This kind of parsing algorithm is also called recursive descent parsing or top down predictive parsing, many names for this algorithm. So, we have discussed, 1 problem with this algorithm.

(Refer Slide Time: 38:37)



So, Top-Down Parsing is predictive, no regard for data, the word in front of the input pointer, no regard for this, and useless productions brought for processing

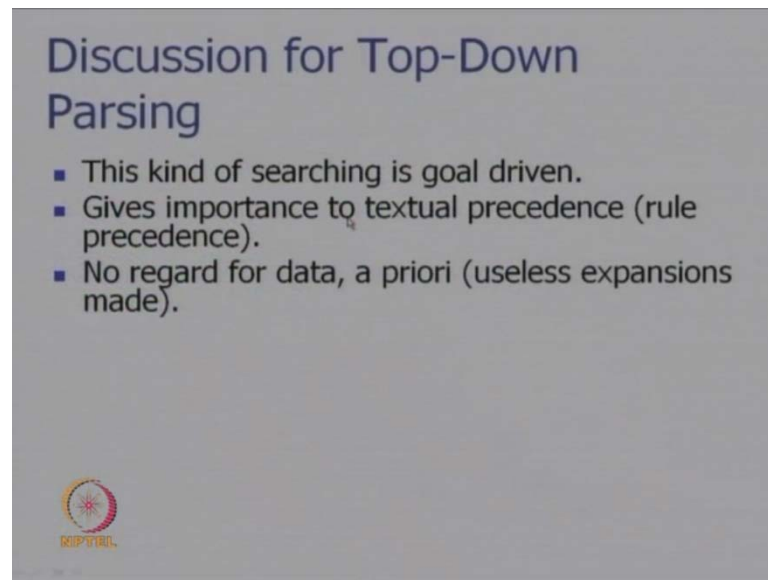
(Refer Slide Time: 39:40)



However, more serious problem is that problems with Top-Down Parsing a more serious problem is that, cannot have left recursion, that is rules of the form  $A$  goes to  $AB$ , that should be easy for you to see, if you have the production of the form  $A$  goes to  $AB$ . Then when  $A$  is on the stack, it is expanded to  $AB$ . Now, again  $A$  needs to be expanded, when  $A$  is expanded another  $A$ , becomes in these  $AB$  has  $A$  in front which needs to be

expanded. So, this can go add infinity term, so if there is left recursion in the production rule, you cannot make use of Top-Down Parsing in particular without removing the left recursion. So, this is an important short coming of drop down parsing.

(Refer Slide Time: 40:54)



Proceeding further, we can re iterate some of these points made, these kind of Top-Down Parsing is a kind of search, and it is goal driven, it gives importance to textual precedence, rule precedence, no regard for data f r useless expansions, can come. And the fact, that gives preference to textual precedence, should be clear the rule NP goes to DTN was textually before NP goes to N. So, that rule was invoked first, and NP goes to N was held in abeyance in the backup state.

(Refer Slide Time: 41:38)

**Bottom-Up Parsing**

Some conventions:

$N_{12}$  — Represents positions

$S_{1?} \rightarrow NP_{12} \cdot VP_{2?}$

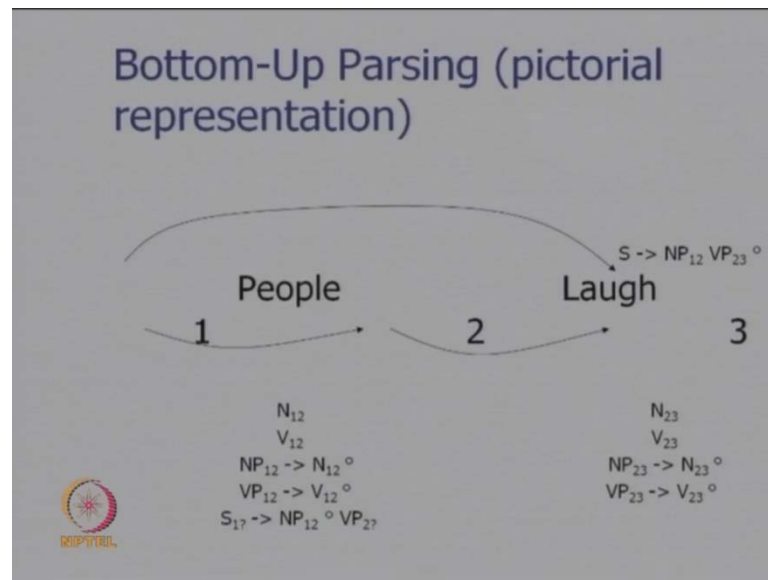
— End position unknown

— Work on the LHS done, while the work on RHS remaining

Bottom-Up Parsing is the complementary approach to Top-Down Parsing, here some conventions are useful, so  $N_{12}$  the suffix 12 it represents the positions, the meaning of  $N_{12}$  is that, in the sentence between position 1 and position 2. I have a noun  $N$ , so using this convention I can write the production rule for  $S$  goes to  $NP VP$  as follows, I know that the sentence starts at the position number 1 but I do not know where it ends that is why this question mark  $S$  goes to  $NP$ . So,  $NP$  begins at position 1 to be fully correct, we may say I also do not know, where  $NP$ , the noun phrase ends, so it could be 1 question mark, and then  $VP$  to be completely correct, there could be a question mark, we do not know, where the  $VP$  begins, and if you do not know, the length of the sentence, we do not know where it ends also.

Now, this dot plays a very important role in the whole Bottom-Up Parsing algorithm, the movement of the dot controls the parsing strategy, all the entities which are before the dot, show the completed work, and entities coming after show work to be done. So, in this example, if we see the position of the dot and the fact, that there is a noun phrase before it, and verb phrase after it. This means that we have already done the work of finding a noun phrase, a noun phrase has been found in the position 1 to 2, the task of finding the verb phrase from the position 2 remains to be done.

(Refer Slide Time: 44:00)



Now, here is a pictorial representation of the algorithm of Bottom-Up Parsing, we have these all important positions 1 2 and 3, before the sentence, after the sentence and between what is, so the way Bottom-Up Parsing goes is that first people is resolved, people can be noun or a verb, so write  $n_{12}$  or  $V_{12}$ , which means we have found a noun or a verb between position 1 and 2.

So, we can now resolve, this rule  $NP$  goes to  $N$ , because you have found noun, between position 1 and 2. So, this says that you have found a noun phrase also between 1 and 2, and we have also found a verb phrase between 1 and 2, because people can be verb 2,  $V_{12}$ . Now, we can move up to the level of  $S$ , and say that  $S_{17}$  question, there is a noun phrase between 1 2, dot means the work, done is that of finding the noun phrase, we have to find a verb phrase, now it can begin from 2.

Then you move on to laugh, laugh can be both noun and verb, the noun possibility is discarded, the noun possibility gives rise to  $NP$  getting resolved, between 2 and 3, and  $VP$  getting resolved, between 2 and 3. So, a verb has been found, since the work of finding  $VP$  is over, we can go back to this rule, and move the dot over to this place. So, the sentence is parsed by finding a noun phrase here, verb phrase here, this is Bottom-Up Parsing and it is complementary to drop down parsing, it is data driven, the words decide what kind of non terminals are invoked, and that in turn decides, which non terminals can be resolved. So, this we will discuss in more detail, and then move on to the

combination of drop down and Bottom-Up Parsing very famously known as chart parsing in the next class.