

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

Lecture - 38
Parsing Algorithms

In the last lecture, we started parsing, which is also called syntactic processing. Parsing we said was probably one of the most well understood areas of language processing, ((Refer Time: 00:36)) of algorithms have been designed for obtaining the structure of a sentence, and this phrases very important, because from the parse tree one then moves to the stage of semantic processing, where the semantic roles, disambiguated words, disambiguated names, co references, all these difficult problems are solved. However, the first crucial step to all these challenging tasks is syntactic processing or parsing. So, you would like to take a detailed look at, how parsing is done, today's topic will be parsing algorithms. Last time, we described top down parsing in detail, will mention this briefly then moved to bottom-up parsing, and a very famous algorithm called chart parsing, top-down bottom-up chart parsing.

Then, we will discussed, what happens when a sentence is ambiguous, multiple parses as possible, for such a sentence. We described last time, that parsing in it required, even though the meaning is more or less understood based on the word senses, and their arrangements, it is still a critical task to obtain the parse tree of the sentence, that itself resolves a large amount of ambiguity. For example, if we take the sentence, I saw the boy with a ponytail, now with a ponytail should be attached to the boy, because it is a qualifier, for the boy, and when we do this parse tree construction, the tree would reveal, that the whole preposition face, with the ponytail has these, attachment with the boy.


At this stage of syntactic processing, it is possible to find these, attachment now, we also said that, the parsing is again a critical requirement; we took a look at one replacement test. So, I want a white horse, he wants a brown one, so this is known as the one replacement phenomena in language processing, one has a enough for reference to horse. So, these kinds of phenomena need deep parse trees, we have to know the structure of sentence, in pretty good detail. So, parsing is definitely necessary, we now proceed to, the algorithms for parsing, starting with the slides.

(Refer Slide Time: 04:05)

A note on Language Modeling

- Example sentence
 - " ^ The tortoise beat the hare in the race."

Guided by frequency	Guided by Language Knowledge	Guided by world Knowledge		
N-gram (n=3)	CFG	Probabilistic CFG	Dependency Grammar	Prob. DG
^ the tortoise $5 \cdot 10^{-3}$	S-> NP VP	S->NP VP 1.0	Semantic Roles <i>agt, obj, sen,</i> <i>etc.</i>	Semantic Roles with probabilities
the tortoise beat $3 \cdot 10^{-2}$	NP->DT N	NP->DT N 0.5	Semantic Rules are always between "Heads"	
tortoise beat the $7 \cdot 10^{-3}$	VP->V NP PP	VP->V NP PP 0.4		
beat the hare $5 \cdot 10^{-6}$	PP-> P NP	PP-> P NP 1.0		




We start with first, a note on language modeling, now language modeling is a very frequently used term, in natural language processing, when we use the term language modeling, we mean that a system is created, for the language regularities. So, that we can serve 2 purposes, I will write these down.

(Refer Slide Time: 04:44)

Purpose of Language Modeling:

1. Belongingness:
 - Is the sentence a valid sentence of the Language
2. Predict the next word
 - Given $w_1, w_2, w_3, \dots, w_k$ find w_{k+1} ?

5/11
NLP-lect 38



Purposes of Language Modeling; 1 Belongingness is the sentence a valid sentence of the language, question number 2 Predict the next word, so given w_1, w_2, w_3, w_k find w_{k+1} plus y, so these 2 are sake to be the basic tasks, for language modeling. So, the reason for

the word modeling is that, we tried to capture, the underlined processes of language generation, and analysis, and model it, model it by means of some abstractions, going to the slide. We look at this example sentence, beginning of the sentence indicated by the hat symbol, the tortoise beat the hare in the race, then there is a full stop, which is the sentence end marker. So, sentence begin marker is the hat symbol, then we have this words, the tortoise beat the hare in the race, full stop is the sentence ender. Now, we can see that it is possible for us, to have 3 kinds of language models, going from shallow representation, to deep representation.

So, the first language model is the famous are well known N-gram model, here we simply take sequences of N words, in the current example, we show N-grams, with N equal to 3. So, the first trigram, are sequence of 3 words is hat the, and tortoise, and the next trigram is the tortoise beat, next tortoise beat the, next beat the hare. So, we keep on collecting, this 3 word sequences and that becomes our Language Modeling, all beat with some numbers, placed beside them, these numbers are nothing but the probability is of these trigrams. So, it is quite easy to calculate these probabilities, they can be done in multiple ways, one of the obvious ways, is to see how many times, does this pattern act the tortoise, appears in a large enough purpose, and divide that count, by the count, of all possible trigrams in the cuprous.

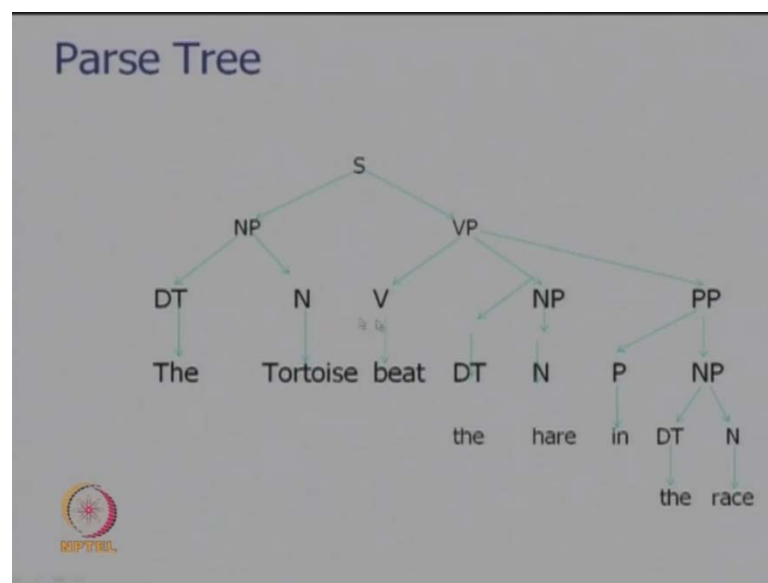
So, these requires, calculating the frequencies of individual trigrams, and dividing the frequency of a particular trigram, by the total number of trigrams in the cuprous. So, this is a fairly useful thing to do, because it shows the surface structure of the sentence. Now, this surface structure can be captured very easily, by fast processing of the text, the next representation is guided by the knowledge of language, the previous language model, was simply obtained by sliding a window, on the text and calculating frequencies.

The next model requires knowledge of the language, so we have 2 schemes here, the first is context is free grammar, where the production rules are always of the form, non-terminal going to non-terminal, non-terminal or non-terminal going to 1 non-terminal or a non-terminal, going to a terminal. This is context free grammar, it is called context free, because these, kind of rewriting depends, only on the non-terminal, and not on anything, in the neighborhood, the context invest the non-terminal appears is immaterial.

So, this that context free grammar, the next representation is called the probabilistic context free grammar, where we again have this kind of rules, but they are augmented with probability values. We will see, the meaning of this probability values, in the context of probabilistic context free grammar, the final representation is, guided by many times, the word knowledge, because we have to produced, what are call semantic roles like agent, object, seen, instrument and so on. And semantic roles or always, between heads first of all and between the head on the modifier, probabilistic a dependency grammar or semantic role leveler comes with semantic roles, which have probabilities associated, with them.

So, we find here that the Language Modeling goes from, a very simple representation of sequence of words, and the probabilities to grammatical knowledge of language, and then to knowledge of word, knowledge of language all combined. So, this is the most sophisticated language model, which can find many applications, but such language model or very challenging to create, needing as they do, various kinds of disambiguation.

(Refer Slide Time: 12:14)



A typical Parse Tree shown here, the tortoise beat the hare in the race, now it is very clear, from the structure, as to how the phrases are composed, and how do, these phrases give as to the sentence. So, here the tortoise is a noun phrase, the as the part of speech, DT the terminal, tortoise as the part of speech, as N noun. These two together form the noun phrase, and the noun phrase is the first important constituent of a sentence, then we

moved to the verb phrases component, of the verb phrases component of the sentence. The verb phrase in this current example is composed of 3 constituents, the verb which is beat a noun phrase, which is that hare, and the preposition phrase, which is a preposition and a noun phrase, the noun phrase is the race. Now, let us understand the attachment quite carefully in this case, we see that in the race is a preposition phrase, and it as a modifying role, what does it modify, in this particular case, there are a 2 possibilities it can be a modifier for the head in the verb phrase, named in the verb or it can be the modifier, for the hare, in the race just like a boy, with a ponytail.

However, the semantics on the next level of processing, which uncovers the meaning of the sentence tells us, that noun in the race is actually is seemed, for the beating activity, defeating activity. So, this should be attached to the verb phrase, at the top-level, they are by becoming a modifier, for the verb beat. So, the tortoise is the agent of beating, whom does it beat, it beats the hare, and in what situation, in the race. So, this is the, essential piece of knowledge, which uncovers the meaning of the sentence but the crucial step in the direction is, the creation of the parse tree.

Now, let us understand a few more things, about the Parse Tree itself, the sentence always starts, with the start symbol S, S is the start symbol sentences, always have noun phrase, and the verb phrase, the noun phrase itself, and come in many different forms one of the forms, which is shown here has determiner and noun sequence, the noun phrase also could have been a simple noun, boys for example, this is a plural noun, without anything coming before it, and conform the noun phrase.

It is also possible, that this noun phrase is composed of a article, and a string of adjectives and finally, the noun. So, the little quick or the little cute tortoise, so the little and cute are a adjectives coming between the and tortoise, and they will form the noun phrase. Similarly, at the verb phrase again, come in many different shapes, it can simply be a single verb, the tortoise sleeps for example, in this case verb phrase as only a verb, it could also be a verb, with a there are single noun phrase as an, as an adjective the tortoise beat the hare, and there is no preposition phrase, now a thought form is possible, where the verb phrase as verb, a noun phrase.

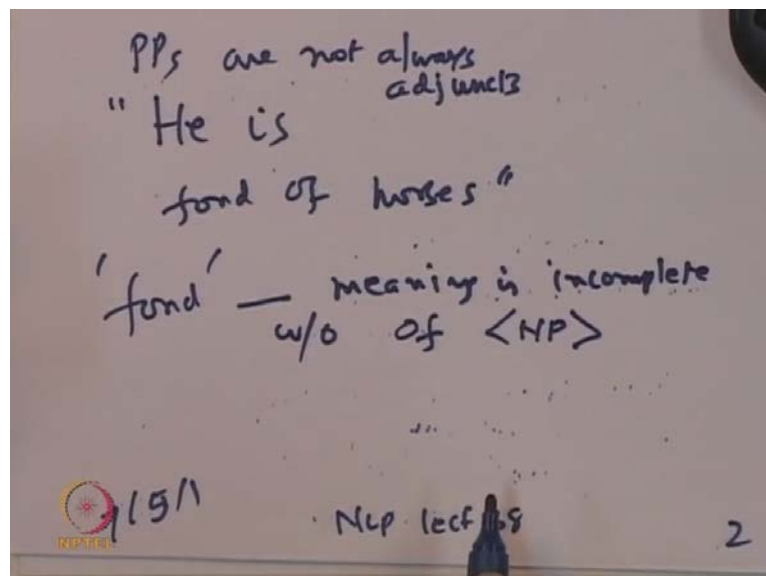
However, there is a qualifier for the head of the noun phrase itself, which is the preposition phrase, and this preposition phrase, get us attached to the noun phrase, all

these different cases are possible, and they refer to different kinds of verbs. So, a verb which can be present singly in the verb phrase, is an example of an intransitive verb, it does not take an object, a verb phrase which takes an object, but has also a preposition phrase embedded in it, is an example of a verb phrase, with an adjunct.

So, this is an adjunct and it is an additional piece of information, which is not essential for the meaning of the sentence but is providing additional information, about the activity going on. So, this part is the adjunct, this is called an argument for the word beat, beat is a transitive word, it always requires an object. So, this noun phrase is that essential element, to complete the meaning structure of the beat. Similarly, the tortoise which is the agent of the beat activity is also essential.

Now, for a verb all those textual elements inside the sentence, which are essential to explicate the meaning of the verb, are called the arguments of the verb, and the additional piece of information, which comes as an adjunct, is called the adjunct. So, this preposition phrase is the adjunct, one must understand, that this additional piece of information, in the form of preposition phrase, is not only an adjunct. So, preposition phrases are not only adjuncts, will write an example, here is an example.

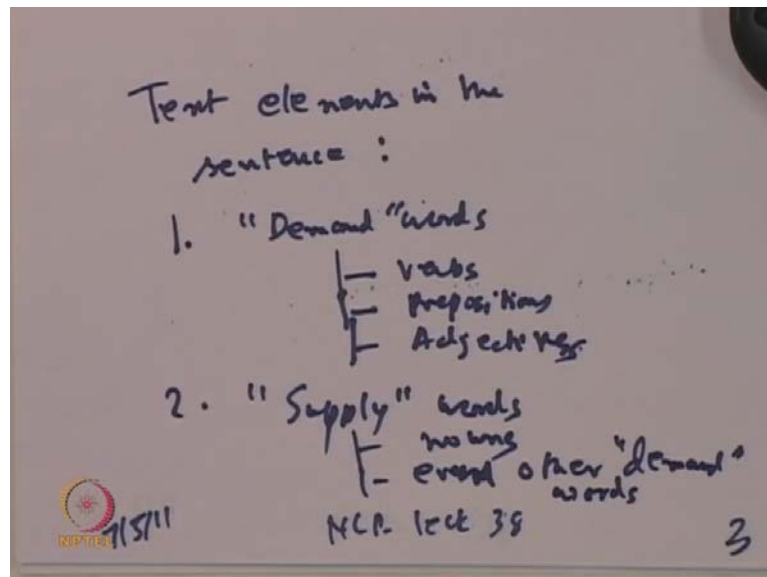
(Refer Slide Time: 19:50)



He is, so PPs are not always adjuncts, so if you have a sentence, He is fond of horses, then fond meaning is incomplete, without "of" and a noun phrase. So, it is important to realize that, preposition phrases are not always adjuncts, they can be arguments to fond is

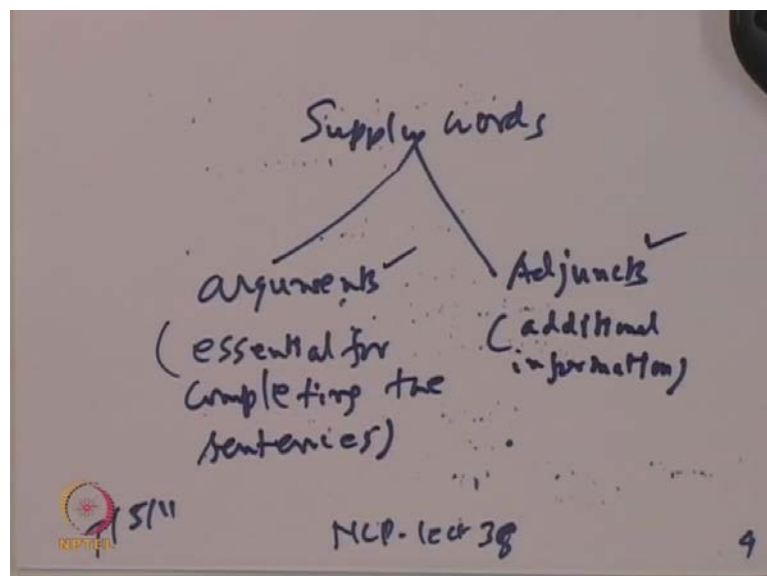
such an example fond has to take of and then an noun phrase to complete the sentential structure. Similarly, we have this example of desires, is desires of something, in this case 2 this of something is not an adjunct but is essential piece of information, and an argument to desires, so we note down these points, on the paper.

(Refer Slide Time: 21:19)



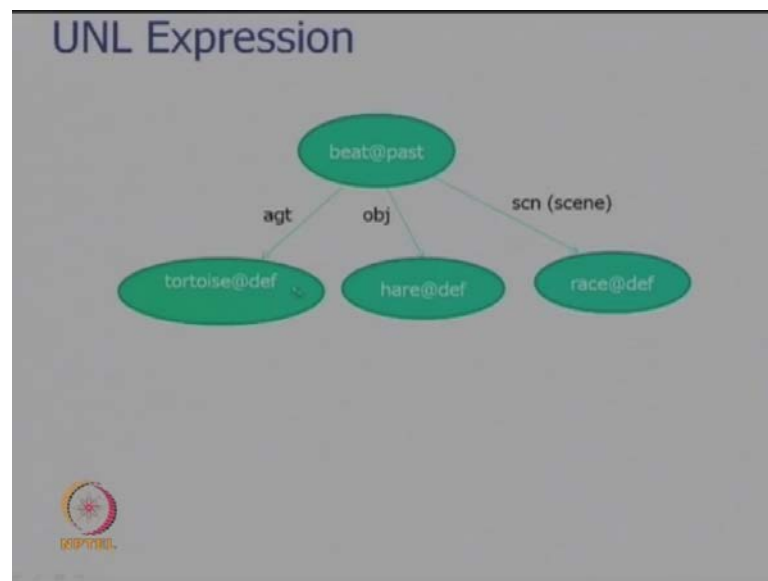
So, text elements in the sentence can have, Demand words, these are verbs, prepositions, adjectives and so on. And then, we have Supply words, which can be nouns, and even other demand words.

(Refer Slide Time: 22:41)



So, these Supply words, the Supply words can be a, 2 kinds, arguments and adjunct, arguments are essential for completing the sentences, adjuncts on the other hand are additional information. Now, the notions of arguments and adjuncts are fundamental to any kind of parsing, beat dependency, beat probabilistic, beat constituency does not matter. The notion of arguments and adjuncts, and how they meet they desire or the [FL] of the demand words is a fundamental notion in processing of sentences, to obtain their ultimate structure, we proceed with the slides.

(Refer Slide Time: 24:11)



We saw the sentential structure, in terms of the parse tree, now we show a semantic graph of the sentence, UNL here stands for universal networking language, this was a representation proposed back in 1996 by an international team of nlp experts for capturing the meaning content of sentences. So, even else provide additional or an additional platform, for capturing the semantic roles, as they exist in sentences. So, many others such semantic role leveling scheme exists, notable amongst them, are frame net link parser, the representation in mini parser, for example, and so on lexical conceptual dependency is also an example, in this list of semantic role levelers, one also remembers conceptual structures by so on.

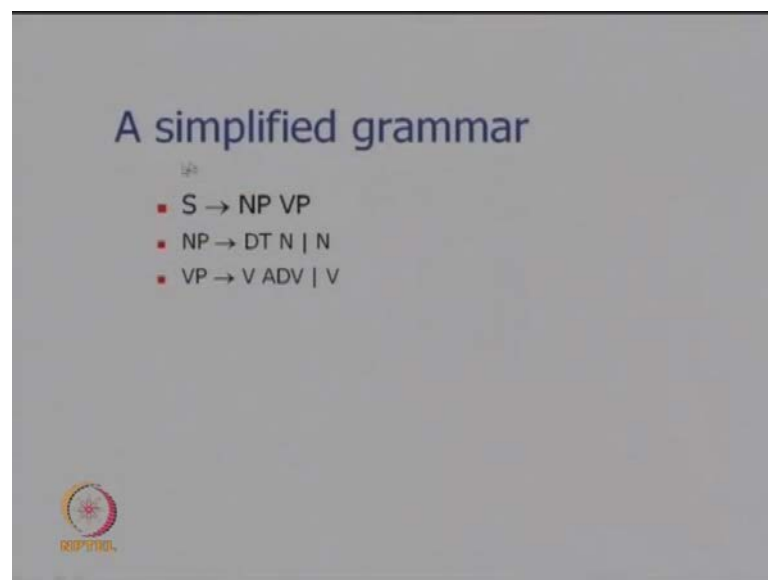
Then of course, the Indian system of [FL] or k s marker, a is an important development in the theory of capturing semantics, so UNL comes in that line of work, and possibly the most modern entry into ways of representing semantics. So, we look at this UNL

expression here, the tortoises beat the hare in the race, so notice how, the key nouns or put together, to obtained the semantic, a beat is the most important activity in this sentence, so this forms what is called the root of the graph or the entry note for the graph.

And the agent of this beat activities tortoise, the first argument for beat, the second argument is hare, which is the object are the patient or the team, this is designated by many different names for beat, and then the adjunct appears as the seen, where this beating of hare by tortoise takes place. This is the race see how, the content words like, the word beat, main verb beat, the noun tortoise, the noun hare, the noun race, they relate to the main verb of the sentence, by the difference semantic relations like agent, object and seen.

And this a, an important point to note in the graph, the preposition and other function words like conjunct for example, do not normally, make their presence in the semantic row, nouns differently do as does the main verb, which is possibly that, most important entity in the representation, the whole graph is rooted at the main verb, and then the nouns are linked, with the main verb, with semantic roles, proceeding further.

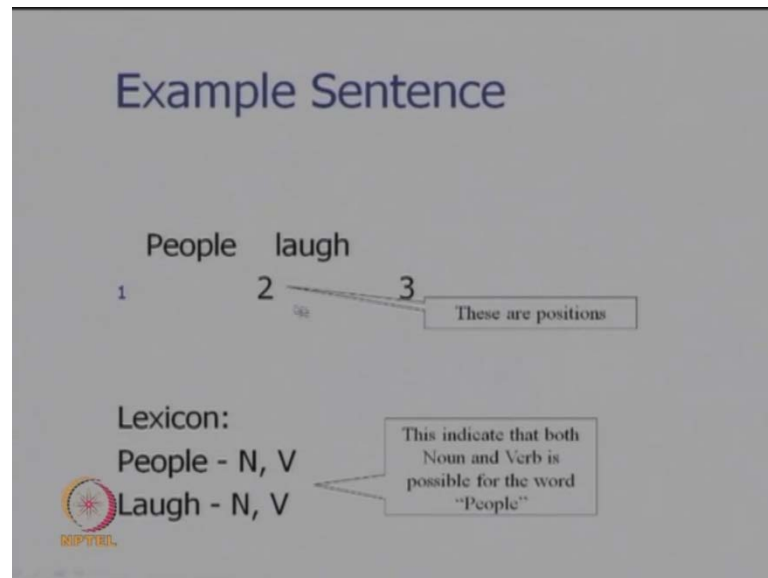
(Refer Slide Time: 28:16)



We go to words discussion of the parsing algorithm, has in the last lecture, we take this simplified grammar of English, were sentence goes to noun phrase and verb phrase, noun phrase are only 2 kinds determiner and noun, are a noun itself, verb phrase also can be a

single verb, are a verb, and adverb combination. So, with the simple grammar, we will proceed to discuss, the parsing algorithms as was done.

(Refer Slide Time: 28:54)



Last time, we completed top down but let us very quickly, remember what we did, we recapitulate the top-down parsing, we have this example people laugh. So, people can be both noun and verb, as indicated in the dictionary or the lexicon laugh also can be a noun or a verb. The most of the parsing algorithms are controlled, by what are called between what positions, between the sentence beginner, and the first word is the number 1, between the first word and second word is the number 2 between the, last word on the sentence, under a full stop is the position number 3. So, let us see very careful, with this kind of numbering, because the parsing algorithm is controlled by these positions, this between word positions, obtaining non-terminals, and these non-terminals are finally, combined to produce the sentence ultimately, now this example sentence.

(Refer Slide Time: 30:20)

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"

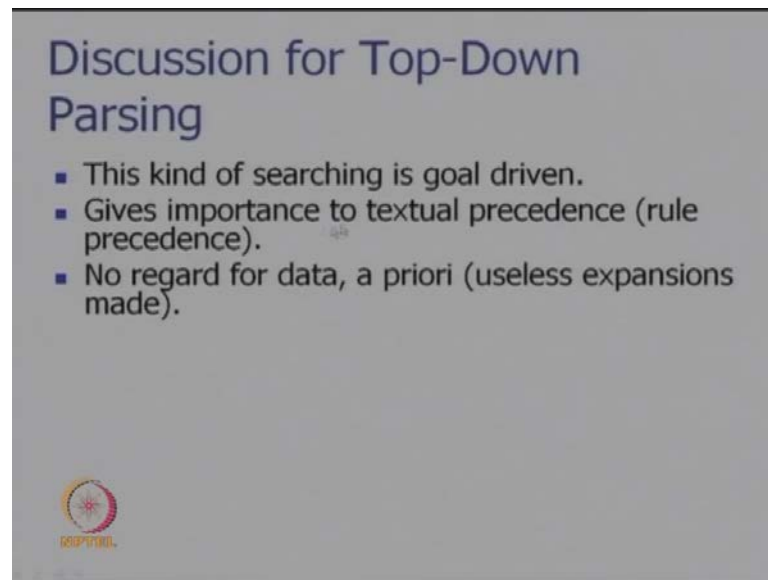
Termination Condition : All inputs over. No symbols remaining.
Note: Input symbols can be pushed back.

If you processed by top-down parsing, we find that the first non-terminal symbol is S, the sentence starting symbol, and this 1 indicates the position of the input pointer. So, the input pointer as it advances, the number gets incremented, first S is expanded into noun phrase, and verb phrase, input point does not move. So, the position remains the 1 noun phrase, we now expanded into two determiner and noun, input point remains at 1. However, there is a backup state, noun only and verb phrase, because NP can be only a noun 2 and the input pointed is at 1, then comes the failure to match phrase, there is no determiner or in the input sentence, it begins with the noun people, so DT cannot be matched. So, we threw away this particular entity, the non-terminals symbol and the input pointer combination, we discard this tack entity, and bring the back of state, which is noun, verb phrase and the position of the input pointer.

So, when it comes on the stack, now it is possible to match, the first word in the sentence in the people, that first word is consumed, the input pointer moved to position number 2. Now, we are expecting a verb phrase, the verb phrase is expanded to V and ADV for an adverb combination, the backup state is verb, input pointer 2, now you match the verb law, and that exposes at verb. So, you are expected adverb but what happened is that the sentence got over, the input pointer moved to position number 3, the sentence got over. And we know that the parsing cannot be completed, now we bring the backup state, which is V2 and discard the adverb 3 state, and now the parsing will succeed, because verb can match law, the sentence also finishes, there is nothing on the stack, there is

nothing on the input t, this indicates success of the parse process. Now, it is an easy matter to traverse, the stack are make use of the entries in the stack go back from the last state, and produce the parse tree.

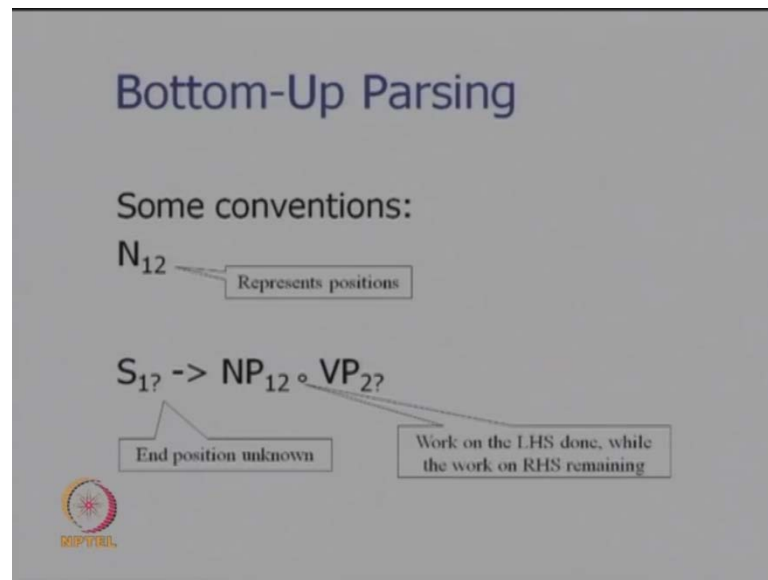
(Refer Slide Time: 33:27)



The slide is titled "Discussion for Top-Down Parsing" in a large, dark blue font. Below the title, there are three bullet points in a smaller, dark blue font. The first bullet point says "This kind of searching is goal driven." The second bullet point says "Gives importance to textual precedence (rule precedence)." The third bullet point says "No regard for data, a priori (useless expansions made)." In the bottom left corner of the slide, there is a small circular logo with a red and yellow design, and the text "IITM" below it.

This is top down parsing, for top-down parsing is a kind of searching, which is goal driven, goal here is to finally, reach a state where there is nothing on the input it, nothing on the stack, what process is guided by the start symbol S, during rise to non-terminal and the non-terminals giving rise to more non-terminals finally, consuming the data on the input sentence. And however, there is this point of importance given to textual precedence NP goes to N, it came after NP goes to DT, and which was the useless role expansion, there is no regard for the data be plays DT on the stack, even though they was no determiner, and the we input t, why should such determiner is be brought here after all, why should we do that.

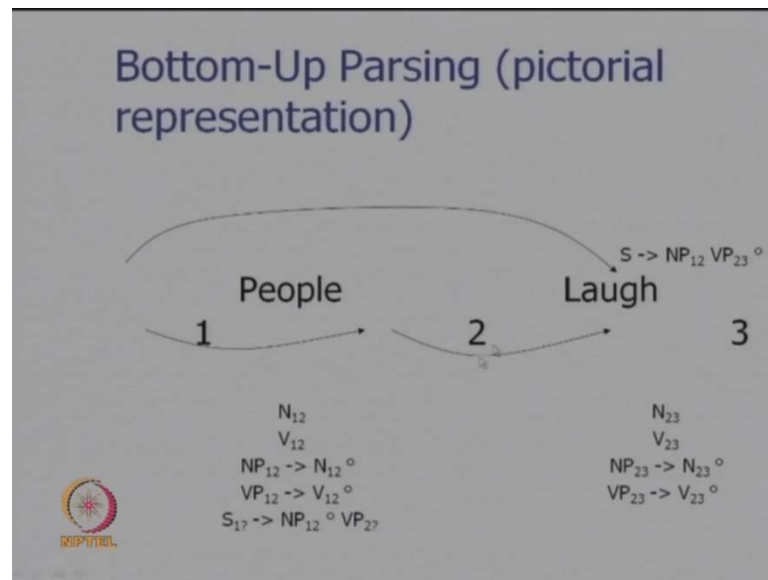
(Refer Slide Time: 34:38)



So, this is completely data ignorant algorithm, there is the problem arising out of texture precedence of rules, and then more problems like left accessions ion, so if you goes to a b then, because of the very nature of top down parsing, the expression process will go into on in finite look. So, we move on to bottom-up parsing, and some conventions here are very useful. So, if you take the symbol N_{12} , this means that between position 1 and 2, there is a noun, and if we take such suffixes for entity is in a rule, and look at the complete rule. Then such a situation obtains S_1 question goes to $NP_{12} \cdot VP_2$ question means that the following that in noun phrase has been found, between the position 1 and 2, the dot is to the right of this non-terminal that indicates completed work, and the dot in front of another non-terminal VP that indicates expected verb.

So, we expect to find your verb phrase noun, and if you find that, then the dot will move over VP, and the whole rule will be covered, the dot coming to the rightmost position in a, S goes to NP VP role signals, the completion of the parsing process. So, this dot symbol which is also known as handle in parsing a programming languages, and compilers is an all-important entity, because that dot keep track of completed work to it us left, and expected work to it us right, we hope to find non-terminals which is to the left of the dot, no we will already found non-terminals, which are to the left of the dot, and we hope to find the non-terminal, which are to the right of the dot these has to be done.

(Refer Slide Time: 36:52)



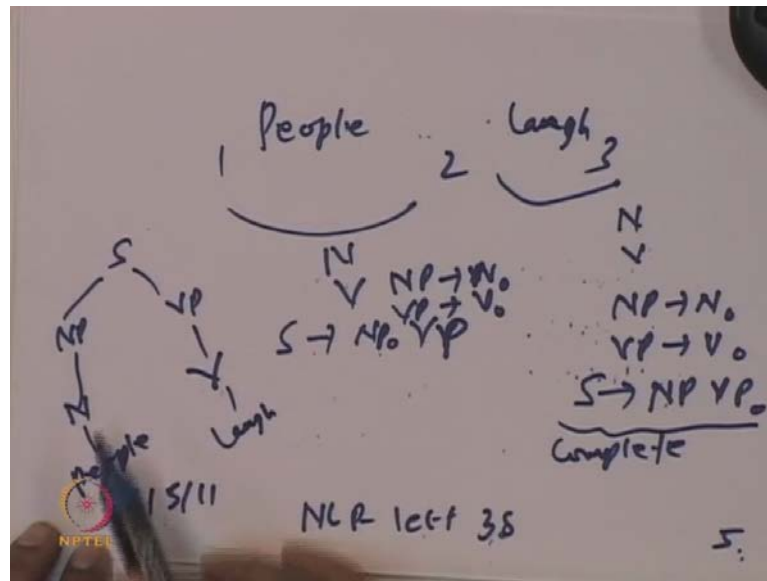
So, we take this example, our running example people laugh let us see, what is goes on here people is either a noun are a verb, and we plays this non-terminals N and V, with the suffixes 124 them, indicating that we have found a noun, between position 1 and position 2, in the sentence, we have also found a noun in the position 1 in 2 in the sentence. Whether, this should be useful or not is another matter, now having found an noun between 1 and 2, we move that dot to the right, for such a rule and P goes to N, this dot shows that, we have found in N, and seems the dot is to the rightmost position, in a rule even the non-terminal, involve in the rule is also reserved.

So, we have found a noun phrase between position 1 and 2, has these dot indicates we have also found a verb phrase between 1 and 2 as an indicated by VP going to V1 to dot. So, all these rules along with dot remain, to show how much of the work has been completed and, these forms a useful knowledge, for the next stage of processing. Now, if we go to the example again, we have finished this items set here, this is also called as items set, we have now decided to move forward, and here is the item set, for Laugh can be noun or a verb, there is a noun between position 2 and 3 similarly, a verb between position 2 and 3, and noun has been found, noun phrase has been found indicated here, we have also found about phrase as it indicated in this rule.

Now, if we combine what has been done, here at this step with the work done in the previous step, now then we can see that the task of finding a verb phrase, which was

waiting has been finally, done and the dot will move over VP. So, S is the rule where dot is move toward VP, we have seen all the words, and that indicates completion of the parse process, it is again an easy matter, to consult the charts and shown in the picture, and from the chart to recover in the parse tree, how shall we do it, the verb phrase goes to v, we can write it here.

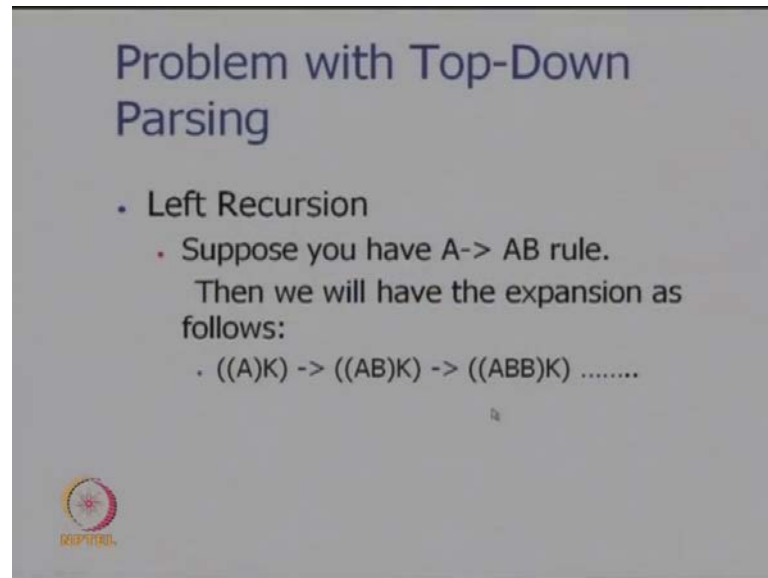
(Refer Slide Time: 39:43)



So, People Laugh, so we have in N here, there were verb here also, and we have S where noun has been found, NP has been found, VP next to be found, and why is NP found, because NP goes to N, the dot is after this rule similarly, VP goes to V dot comes here. At laugh, similar thing happens, N has been found, V has been found, again NP goes to N has been found, VP goes to as V has been found, and at this stage I can finished this rule, with dot moving over the VP and this is complete.


Now, if I want to look at the charts and draw the tree, how will I do it, I will first the SN P VP, so I will write S NP VP that because the task of this rule, resolving this rule is done as indicated by the dot here, and then I find that and NP resolution was done in the previous chart. So, that how was that result, that was result by NP going to N, and VP going to V is in the current chart VP going to V, here and then N goes to people, this is done in this chart and V goes to laugh, this is in this chart. So, looking at the 2 charts to items sets here, I can create the parsing tree very easily, so we proceed further.

(Refer Slide Time: 41:56)



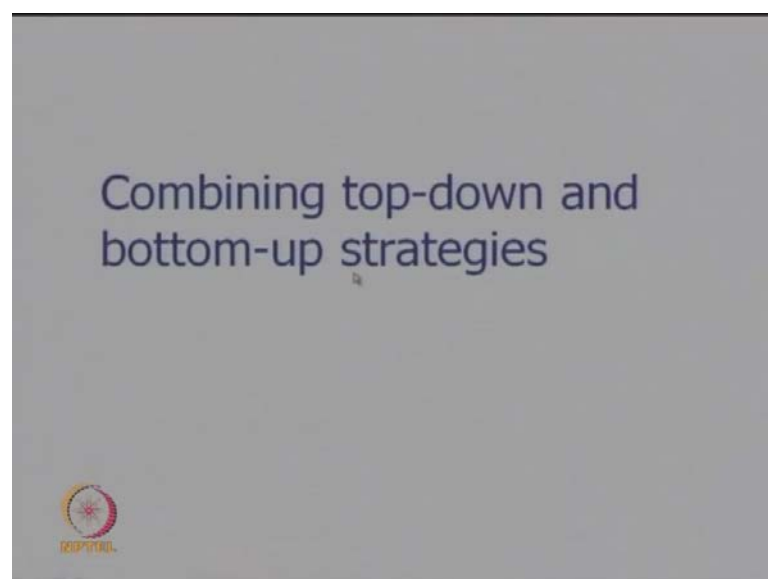
Problem with Top-Down Parsing

- Left Recursion
 - Suppose you have $A \rightarrow AB$ rule. Then we will have the expansion as follows:
 - $((A)K) \rightarrow ((AB)K) \rightarrow ((ABB)K) \dots\dots$




And problems with these individual algorithms, in top down parsing left recursion is a problem, it can lead to infinite number of expansions, and in the bottom up parsing, we have useless rules coming up, as have written in the paper, there is no reason why, this people, this verb for people is invoked, there is reason no reason by N for Laugh should be invoked. So, let us combined data driven with gold driven approach, this whole procedure is data driven, we look at the words, and the properties and get the non-terminals. Now, we will see the top down approach to parsing, and Combined with Bottom-Up Strategies.

(Refer Slide Time: 42:46)



Combining top-down and bottom-up strategies




This is a famous top-down and bottom-up strategy for parsing.

(Refer Slide Time: 42:50)

Top-Down Bottom-Up Chart Parsing

- Combines advantages of top-down & bottom-up parsing.
- Does not work in case of left recursion.
 - *e.g.* – “People laugh”
 - People – noun, verb
 - Laugh – noun, verb
 - Grammar –
 $S \rightarrow NP VP$
 $NP \rightarrow DT N \mid N$
 $VP \rightarrow V ADV \mid V$




Also called TD b u chart parsing top-down bottom-up chart parsing, it combines advantages of top-down and bottom-up parsing, does not in again work in case of lift recursion, left recursion removal is a standard procedure, quite easily done. We take this example, people laugh people can be noun and verb, loss also can be noun and verb, the grammar given is S goes to NP VP, NP goes to DT N or N VP goes to VADV and V.

(Refer Slide Time: 43:22)

Transitive Closure

People laugh


1 2 3



$S \rightarrow \bullet NP VP$
 $NP \rightarrow \bullet DT N$
 $NP \rightarrow \bullet N$

$NP \rightarrow N \bullet$
 $S \rightarrow NP \bullet VP$
 $VP \rightarrow \bullet V ADV$
 $VP \rightarrow \bullet V$

$VP \rightarrow V \bullet$
 $S \rightarrow NP VP \bullet$
success



So, we apply what is called the Transitive Closure, idea of Transitive Closure, first before you begin processing as you there is a input pointer at 1, and we know that S goes to NP VP as to be finally, resolved. So, there is a dot sitting in front of NP, and we know noun that, we have to do the work of finding the NP noun phrase, which again and tiles finding a determiner are noun, is simply noun any of the 2 possibilities, can be tried are both may have to be tried. Now, this is the top-down part of the top-down bottom-up chart parsing strategy, what is happening here is that, S is expanded to NP VP, and in anticipation of future need, we also expand NP, because we had to do the work anyway, so NP goes to DT and NP goes to N, here also dot in, and we plays dot before them, before DT and before N thereby indicating, that we have 2 find such entities in the text.

Now, we shift to the data driven approach the bottom-up approach, so now we know from the previous items set, of the previous chart, that we are looking for a determiner a or you are looking for a noun. Now, the data driven approach same, he says that yes people is a noun, we are found a noun. So, N p goes to noun is now resolve, the dot moves over noun N p is resolve, since N p is resolve this N p in the S goes to NP VP rule is all so resolve.

So, the dot will move towards NP, now that exposes VP at this stage, the top-down part becomes active, the top-down parsing becomes active, VP goes to VADV, and VP goes to V so VADV, we have the dot coming before verb, and then a adverb here also the dot comes, before the verb and thereby we are telling the system, that we now expect a verb, it may be followed by a verb, in an adverb or it may be followed by a nothing but we are expecting about, that what has to be done.

So, you see here the combination of top-down and bottom-up parsing, this is the item set which illustrates this very clearly NP goes to N, and how this is happen is that N has been match that people, which is a bottom-up for strategy, and once we have resolved in the NP, we have resolved NP part of his, now dot comes report VP and immediately the transitive closure operation quick in, so VP goes to dot VADV, VP goes to dot V, this is the data driven bottom up approach, when this data driven bottom-up approach is over and NP goes to N, under V go and VP is exposed with V and ADV is coming, VP coming with V, that is the top-down approach, so bottom-up and top-down.

Now, again it goes into the bottom-up mode data driven mode, V is resolve with dot moving will be, now dot between NP and VP can go and VP, and thereby obtained success by resolving S and finishing the input data items, for this shows the bottom-up data driven parsing. In the next lecture, we will take up, the top down bottom up parsing, and then moved on to probabilistic parsing, the start probabilistic parsing.