CSME 206A Natural Language & Speech Processing **Spring Semester**

Lecture 12: Parsing

Lecturer: K.R. Chowdhary *: Professor of CS*

12.1 Grammars and Languages

A language can be generated given its grammar $G = (V, \Sigma, S, P)$, where V is set of variables, Σ is set of terminal symbols, which appear at the end of generation, S is start symbol, and P is set of production rules. The corresponding language of G is $L(G)$.

Consider that various tuples are as given follows:

$$V = \{S, NP, N, VP, V, Art\}$$

$$\Sigma = \{boy, icecream, dog, bite, like, ate, the, a\},$$

$$\begin{aligned} P = \{&S \rightarrow NP\ VP,\\ &NP \rightarrow N,\\ &NP \rightarrow ART\ N,\\ &VP \rightarrow V\ NP,\\ &N \rightarrow boy \mid icecream \mid dog,\\ &V \rightarrow ate \mid like \mid bite,\\ &Art \rightarrow the \mid a\} \end{aligned}$$

Using above we can generate the following sentences:

The dog bites boy.

Boy bites the dog.

Boy ate icecreame.

The dog bite the boy.

To generate a sentence, the rules from P are applied sequentially starting from the beginning. However, we note that a grammar does not guarantee the generation of meaningful sentences, but generate only those are structurally correct as per the rules of the grammar.

In fact, it is not always possible to formally characterize the natural languages with a simple grammar like above. The grammars are defined by Chomsky hierarchy, as type 0, 1, 2, 3. The typical rewrite rules for type 1 are:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow aAB \\ AB &\rightarrow BA \\ aA &\rightarrow ab \\ aA &\rightarrow aa \end{aligned}$$

where uppercase letters are non-terminals and lowercase are terminals.

The type-2 grammars are:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow aSb \\ S &\rightarrow aB \\ S &\rightarrow aAB \\ A &\rightarrow a \\ B &\rightarrow b \end{aligned}$$

The type 3 grammar is simplest having rewrite rules as:

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow a \end{aligned}$$

The types 1, 2, 3 are called context-sensitive, context-free, and regular grammars, respectively, and hence the corresponding names for languages also. The formal languages are mostly based on the type-2 languages, as the type 0 and 1 are not much and understood and difficult to implement.

12.2 Structural Representation

It is convenient to represent the sentences as tree or a graph to help expose the structure of the constituent parts. For example, the sentence, 'the boy ate a icecream' can be represented as a tree shown in figure 12.1.

For the purpose of computation a tree must also be represented as a record, a list or some similar data structure. For example, the tree above is represented as a list:

```
(S (NP ((Art the)
        (N boy))
   (VP (V ate) (NP (Art a) (N Icecream)))))
```

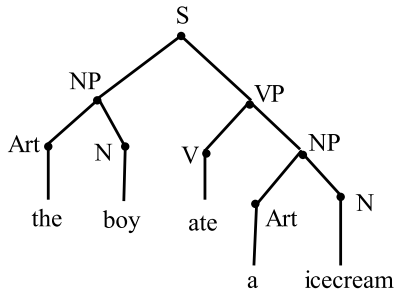

Figure 12.1: A syntactic tree.

A more extensive English grammar can be obtained with the addition of other constituencies such as prepositional phrases PP, adjectives ADJ, determiners DET, adverbs ADV, auxiliary verbs AUX, and many other features. Correspondingly, the other rewrite rules are followings.

$$
\begin{aligned}
&PP \rightarrow Prep\ NP,\\
&VP \rightarrow V\ ADV\\
&VP \rightarrow V\ PP,\\
&VP \rightarrow V\ NP\ PP\\
&VP \rightarrow AUX\ V\ NP\\
&Det \rightarrow Art\ ADJ,\\
&Det \rightarrow Art
\end{aligned}
$$

These extensions allow the increase in complexity of the sentences, along with its expression power. For example, the following sentences.

The cruel man locked the dog in the house.

The laborious man worked to make some extra money.

12.3 Transformational Grammars

The grammar discussed above produce produce different structures for different sentences, even though they have same meaning. For example,

Ram gave Shyam a book.

A book was given by ram to Shyam.

In the above, the subject and object roles are switched. In the first, subject is Ram and object is Book, while in second sentence they are other way round. This, is undesirable feature for machine processing of a language. In fact, sentences having same meaning should map to the same internal structures.

By adding some extra components, we can produce a single representation for sentences having the same meaning, through a series of transformations. This extended grammar is called *Transformational grammar.*

In addition, the semantic and phonological components components, added as new, helps in interpreting the output of the syntactic components, as meaning and sound sequences. The transformations are tree manipulation rules, which are taken from dictionary, where words contain semantic featuring each of the lexicon.

Using transformational generative grammar, a sentence is analyzed in two stages:

1. basic structure of the sentence is analyzed to determine the grammatical constitutional parts, which provides the structure of the sentence.
2. This is transformed into another form, where deeper semantic structure is determined.

The application of transformations is to produce a change from passive voice form of the sentence into active voice, change a question to declarative form, handle negations, and provide subject-verb agreement. The figure 12.2 shows the three stages of conversion, from passive voice to active voice of a sentence.

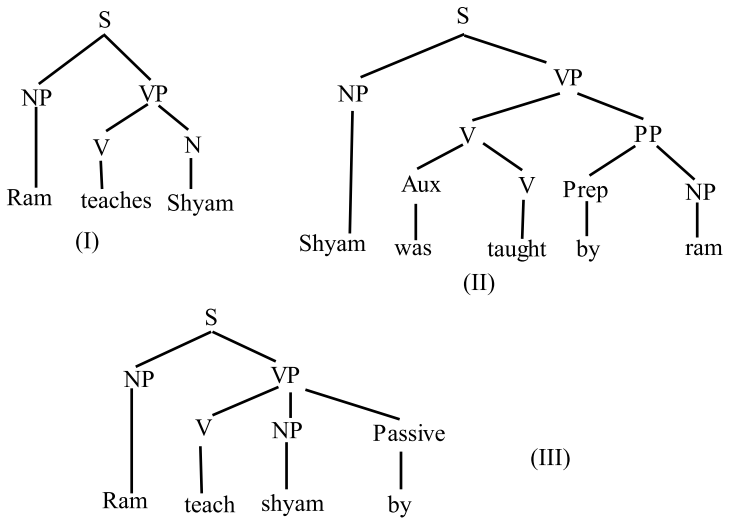

Figure 12.2: Transformational Grammar.

However, the transformational grammars are rarely used as computational models.

12.4 Grammars and NL Parsing

Following are examples, showing the rules and parsed sentences:

```
S -> NP  VP; I prefer a morning flight
VP -> V NP; prefer a morning flight
VP -> V NP PP; leaves Bombay in the morning
VP -> V PP; leaving on Tuesday
PP ->  preposition NP; from New Delhi. (the NP can be location,
        date, time or others)
```

Following are examples of Parts of Speech (POS).

```
N -> flights | breeze | trip | morning |...
V -> is | prefer | like | need | want | fly
Adj -> cheapest | non-stop | first | latest | other | direct ...
Pronoun -> me | I | you | it | ...
Proper-N -> Mumbai | Delhi | India | USA | ...
Det -> a | an | the | this | these | those | ...
Prep -> from | to | on | near
Conj -> and | or | but
```

The following examples show the substitution rules along with values for each POS to be substituted.

```
NP -> Pronoun(I) | proper-N (Mumbai) | det Nomial (a flight) | N (flight).
VP -> V (do) | V NP (want a flight) | V NP PP (leaves Delhi in Morning)
PP -> Pre NP (from Delhi)
```

Making use of above rules, the figure 12.3 demonstrates the parsing of sentence "I prefer morning flight".

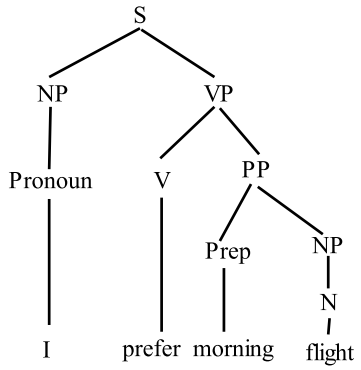

Figure 12.3: Parse-Tree for "I prefer morning flight".

12.5 Sentence Level Constructions

The sentences can be classified as declarative, imperative, and pragmatic, as follows.

- **Declarative Sentences:** They have structure: S $\rightarrow$ NP VP.
- **Imperative Sentences:** These sentences begin with 'VP'. For example, "Show the lowest fare", "List all the scores". The production rules are:

  ```
  S -> VP
  VP -> V NP
  ```

  And, other substitutions for verb are mentioned above.

- **Pragmatic Sentences:** The examples of pragmatic sentences are:

Do all these flights have stops?
Can you give me the same information?
What Airlines fly from Delhi?
What flights do you have from Delhi to Mumbai?

The substitution rule for pragmatic sentences is:

```
S -> Aux NP  VP.
```

Corresponding to the "What", the production rule is "Wh-NP → What". Hence, for the last sentence, "What flights do you have from Delhi to Mumbai?", the first rule to be applied is "S → Wh-NP Aux NP VP". Many times, the longer sentences are conjuncted together using connectives, e.g., "I will fly to Delhi and Mumbai". The corresponding rule is "S → NP and NP". Similarly, there is "S → S and D", and "VP → VP and VP".

12.6 Ambiguous Grammars

The ambiguous grammars have more than one parse-trees, for the same sentence. Consider the sentence "He drove down the street in the Car". The parse-trees are given in figure 12.5 and 12.7. A process for drawing the parse-trees is grouping the words to realize the structure in the sentence. Figure 12.4 and 12.6, demonstrate grouping of the words for parse-trees shown in figures 12.5 and 12.7, respectively.

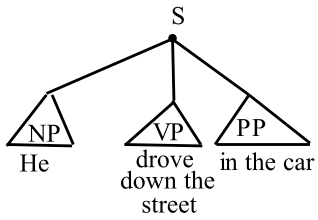

Figure 12.4: Grouping the words for parsing.

12.7 Parsing with CFGs

The parse-trees are useful for:

1. Grammar checking of the sentence,
2. Parsing is an important intermediate stage in semantic analysis.
3. The parsing plays an important role in:
   (a) Mechanical translation,
   (b) Question answering
   (c) Information Extraction

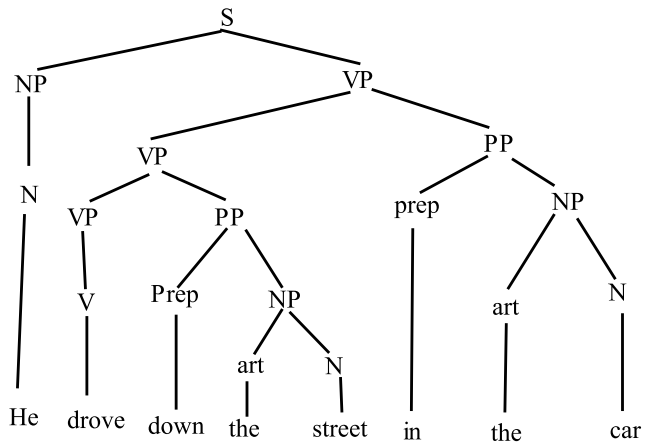

Figure 12.5: Parsing-1: He drove down the street in the car.

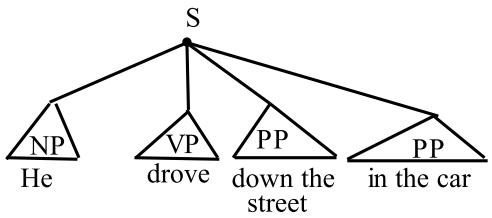

Figure 12.6: Grouping the words for parsing.

12.7.1 Parsing is Search

A syntactic parser can be viewed as searching through the space of all possible parse-trees to find the correct parse-tree. Before we go through the steps of parsing, let us consider the following rules for grammar.

```
S -> NP VP
S -> Aux NP VP
S -> VP
NP -> Det Nom
Nom -> Noun Noam
Nom -> N
NP -> proper-N
VP -> V
VP -> V NP
Det -> a | an | the
N -> book | flight | meal
V -> book | include | proper
Aux -> Does
prep -> from | to | on
Proper-N -> Mumbai
Nomial -> Nomial PP
```

The parse tree is shown in figure 12.8.

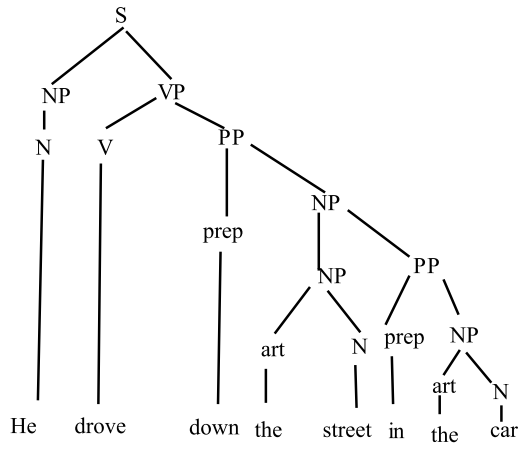

Figure 12.7: Parsing-2: He drove down the street in the car.

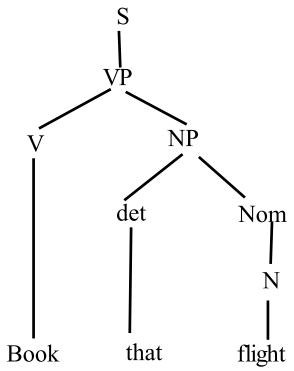

Figure 12.8: Parsing: Book that flight.

12.7.2 Top Down parsing

The searching is carried out from the root node. The substitutions are carried out, and progressing sentence is compared with the input text sentence to determine whether the sentence generated progressively matches with the original. The figure 12.9 demonstrates the steps for the top-down parsing for the sentence "Book that flight".

To carry out the top down parsing, we expand the tree at each level as shown in the figure. At each level, the trees whose leaves fails to match the input sentence, are rejected, leaving behind the trees that represent the successful parses. Going this way, ultimately get the sentence: Book that flight.

12.8 Summary

1. Natural language processing is a complex task, due to variety of structures of sentences, and ambiguity in the language. The ambiguities occur at phonetic levels, semantic levels, and pragmatic levels.

2. The languages are defined as per the Chomsky hierarchy, as type 3, 2, 1, 0, from mots simple to most

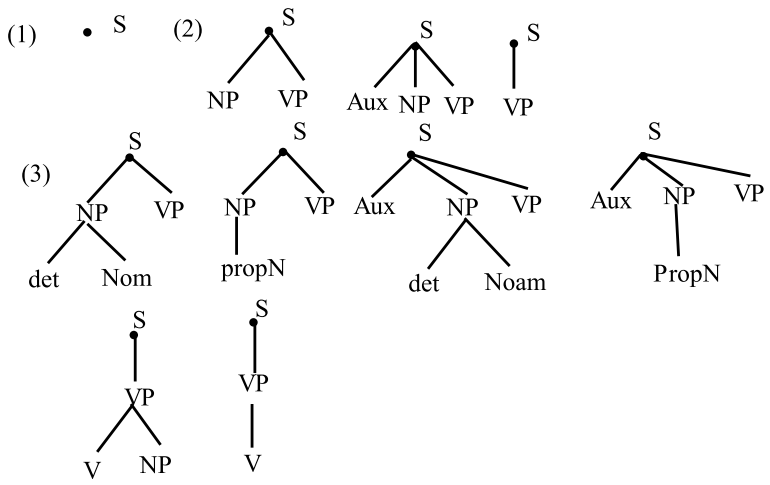

Figure 12.9: Top-down parsing of: Book that flight.

complex, called generative grammars. Though, the NL is not context-free, but due to non-availability of proper theory of type 0, and 1, the theory of type 2 (context-free) grammar is applied to NLP also.

3. The subject of NLP is particularly important because, NLP has enumerable applications, which have further expanded due to Internet and WWW.

4. The sentences of NL can be generated by constructing the parse-tees, one for each sentence.

Exercises and Review Questions

1. Develop the parse tree to generate the sentence "Rajan slept on the bench".

2. Draw the tree for the following phrases:

   (a) after 5 pm.

   (b) on Tuesday.

   (c) From Delhi.

   (d) Any delay at Mumbai.

3. Draw the tree structures for the following sentences:

   (a) I would like to fly on air India.

   (b) I need to fly between Delhi and Mumbai.

   (c) Please repeat again.

4. Convert the following passive voice to active voice. Construct the necessary trees. Also write the steps.

   The village was looted by dacoits.

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow N \\
NP &\rightarrow Det\ N \\
VP &\rightarrow V\ PP \\
PP &\rightarrow Prep\ NP \\
N &\rightarrow Rajan \mid bench \\
Det &\rightarrow the \\
prep &\rightarrow on
\end{aligned}
$$

5. Given the parse-tree in figure 12.10, construct the grammar for this.

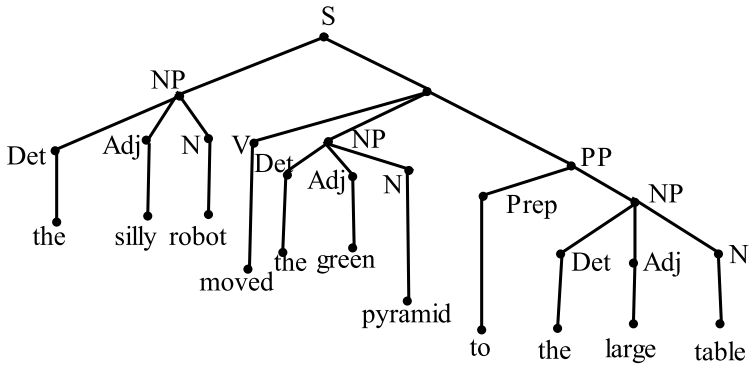

Figure 12.10: Parse-tree.

6. Construct the grammars and parse tree for the following sentences.
   (a) The boy who was sleeping was awakened.
   (b) The boy who was sleeping on the table was awakened.
   (c) Jack slept on the table.

References

[1] D. Jurafsky and J. Martin, "Speech and Language Processing," *Pearson India*, 2002.