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

# Lecture 14: Syntax Analysis

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



## 14.1 Introduction

Syntax is the study of the principles and processes by which sentences are constructed in a particular languages. Syntactic investigation of a given language has as its goal for the construction of a grammar that can be viewed as a device of some sort for producing the sentences of the language under analysis. More generally, linguists must be concerned with the problem of determining the fundamental underlying properties of successful grammars.

The central notion in linguistic theory is that of "linguistic level." A linguistic level, such as phonemics, morphology, phrase structure, is essentially a set of descriptive devices that are made available for the construction of grammars; it constitutes a certain method for representing utterances. We can determine the adequacy of a linguistic theory by developing rigorously and precisely the form of grammar corresponding to the set of levels contained within this theory, and then investigating the possibility of constructing simple and revealing grammars of this form for natural languages.

Sentence parsing has two goals: 1. a parser is used to discover, in a spoken or written text, meaningful groups of words and their organization, following the grammatical constraints of the language, 2. a parser must provide the units and identify the domains of other processes (i. e., processes such as those that find the predicate-argument structure or the noun-phrase referent of a text).

Together, these two parser functions help to limit sentence ambiguity so that the particular meaning of an utterance can be recovered from the vast meaning potential of the language.

Parsing is a process; it concerns the incremental production of the syntactic description of texts. There are two opposing ways to view this process. It can be viewed as a series of choices provided by the grammar that result in a minimal set of alternatives (i.e., applications of production rules). Or, it can be seen as refining a description, true at each step, to arrive at a useful syntactic description of a sentence[1]. The distinction between these two views is simply whether, in the course of parsing a sentence, alternatives are represented explicitly.

## 14.2 Generative Methodology

Noam Chomsky published *Syntactic Structures* in 1957 ushering in the generative era of linguistic theory. The essential paradigm shift or methodological innovation was that linguistic analysis was no longer an entirely

[1]For example, breaking a sentence at NP and VP, and breaking NP further as like "Art N", and similarly for VP, and other constituents.

'bottom-up', data-driven purely empirical process, but rather, generative linguists started out with a meta-theory of what grammars of human languages look like and attempted to express specific grammars within this meta-theory. Such grammars are *generative* because they consist of finite sets of rules which should predict all and only the infinite grammatical sentences of a given human language (and what is conveyed about their meaning by their grammatical structure). Thus, *generative grammars* define well-formed sets or mappings between sentences and (part of their) meanings.

Generative grammar came into existence almost at the time the theoretical computer science came, and much of the theory of parsing and compiling programming languages has its antecedents in early generative linguistics (the Chomsky Hierarchy, etc.). For example, context-free grammars and Backus-Naur notation are weakly equivalent formalisms, generating the same class of context-free languages, which seem quite appropriate for capturing the hierarchical structure that emerges from immediate constituent analysis. However, once formulated this way, the analysis becomes predictive because the rules of the grammar generate further sentences paired with hierarchical structure. Generative theory is thus good for capturing the productivity of human language(s). However, even with a meta-theory, we still need methods to choose between analyses and to choose the meta-theory. So, we will focus primarily on *linguistic analysis* (and terminology) in our following discussions.

A language can be generated given its grammar $G = (NT, \Sigma, S, P)$, where $NT$ is set of variables (non-terminal symbols), $\Sigma$ is set of terminal symbols that 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 given as follows, where $NT$ is set of variable symbols (also called non-terminal symbols), as[2],

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

the terminals are words,

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

and the production rules are,

$$\begin{aligned} P = \{&S \to NP \;\; VP, \\ &NP \to N \mid Art \;\; N, \\ &VP \to V \mid V \;\; NP, \\ &N \to boy \mid icecream \mid dog, \\ &V \to ate \mid like \mid bite, \\ &Art \to the \mid a\}. \end{aligned} \tag{14.1}$$

Using above we can generate the following sentences:

The dog bites boy.

Boy bites the dog.

[2]Since the symbol $V$ is already reserved for Verb, we shall use $NT$ that stands for no-terminals, i.e., variables.

Boy ate ice-cream.

The dog bite the boy.

And, many others, out of which many may not be acceptable in a civilized world!

### 14.2.1 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 the ice-cream" can be generated using a parse-tree as shown in Fig. 14.1, using the grammar defined in equation14.1.

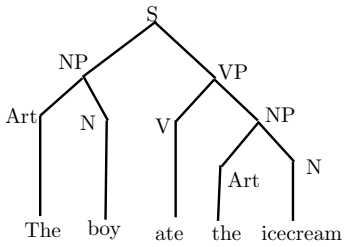


Figure 14.1: A syntax tree for "The boy ate ice-cream"

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 can be represented as a *list* below:

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

which says that a sentence is $NP$ followed with $VP$; and $NP$ is "Art $N$", $VP$ is "$V$ $NP$"; $NP$ is "Art $N$" is "the boy". Next, "$V$ $NP$" is "ate Art $N$"; "Art" is "the", "$N$" is "Ice-cream".

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 classes. Following are some more rewrite rules:

$$\begin{aligned}
&PP \rightarrow Prep \;\; NP, \\
&VP \rightarrow V \;\; NP \mid V \;\; ADV \mid V \;\; PP \mid V \;\; NP \;\; PP \mid AUX \;\; V \;\; NP \\
&Det \rightarrow Art \;\; ADJ \mid Art \\
&Art \rightarrow a \mid an \mid the.
\end{aligned} \tag{14.2}$$

These extensions allow the increase in complexity of the sentences, but at the same time increase the 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.*

## 14.3 Grammars

Almost every NLP system has a gramamr and associated parser. A grammar is a finite specification of potentially infinite number of sentences, and a parser for the grammar is an algorithm that analyzes a sentence and assigns one or more structural descriptions to the sentence according to the grammar, if the sentence can be characterized by the grammar. The structural descriptions are necessary for further processing, and for example, for semantic interpretation. The mathematical interpretation to grammar was given by Chomsky in the form of formal grammar in late 1050s, which was the beginning of the investigations of mathematical and computational modeling of grammar. The hierarchy of grammar, namely regular grammar, context-free grammar, context-sensitive grammar, and unrestricted grammar are due to Chomsky.

It is useful to think grammar in a language as model or parser as the stages of constituent combinations, and transitions between configurations describe how constituents are combined in deriving larger constituents. For instance, the configurations may be the states of a finite-state machine (say, PDA – pushdown automata) and transitions represent how words may be appended to the end of a sentence prefix.

We understand larger textual units by combining our understanding of smaller units. The aim of linguistic theory is to show how these larger units' meaning arise out of the combination of smaller ones. This is modeled by grammar. It has been the tradition to subdivide the task into *syntax* and *semantics*, where syntax describes how different formal elements of a textual unit, most often the sentence, can be combined, and the semantics describes how the interpretation is calculated.

In most language theory applications, the encoded knowledge, i.e., the grammar, is separated from processing components. The grammar consists of lexicon, and rules that syntactically and semantically combine words and phrases into larger phrases and sentences. A variety of languages have been developed for the encoding of linguistic knowledge, some are geared towards conformity with formal linguistic theories, while others are designed to facilitate certain processing models or specialized applications.

To generate a sentence, the rules from production set $P$ 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 that are structurally correct as per the production rules of the grammar. In fact, it is not always possible to formally characterize the natural languages with a simple grammar like descriptions mentioned above.

In the following, a brief description of each class of the grammar is mentioned.

### 14.3.1 Grammar Classes

The grammars are defined by Chomsky hierarchy, as type $3, 2, 1, 0$, called as Regular Grammar, Context-free Grammar, Context-sensitive Grammar, and un-restricted Grammar, respectively. A grammar is a 5-tuple, $G = (NT, \Sigma, S, P)$, we can define the grammar classes as follows.

**Type-3 Grammar** The type-3 Grammar (*Regular Grammar*) is simplest one, having rewrite rules like:

$$A \to aB \mid a \tag{14.3}$$

Here $A, B$ are non-terminals, and $a$ is terminal.□

**Type-2 Grammar** The type-2 Grammars, called CFGs (*Context-Free Grammars*), are defined as:

$$A \to \alpha, \tag{14.4}$$

where $A$ is non-terminal, and $\alpha \in (NT \cup \Sigma)^*$, i.e., any combination of terminals and non-terminals.□

**Type-1 Grammar** The typical rewrite rules for type-1 grammar (*Context-Sensitive Grammar*) are:

$$A \cup (A \cup \Sigma)^* \to (A \cup \Sigma)^*. \tag{14.5}$$

That is, in the LHS of a rewrite, there are terminals and non-terminals, with at least on terminal symbol, while in the RHS there can be arbitrary combination of terminals and non-terminal symbols. In addition, there is one restriction in the rule $\alpha \to \beta$, that is, $|\alpha| \leq |\beta|$. Hence, the languages are called tape limited grammars. That is, the size of the string during the processing shall never be more than the final string.

A context-sensitive grammar is also like a CFG, except that the rewriting of a nonterminal is dependent on the context surrounding the nonterminal, unlike the rewrite rules of CFG where the rewriting is context independent.□

**Type-0 Grammar** These are called unrestricted grammars and the corresponding languages are unrestricted languages or Turing recognizable languages. These are same as that type 1, except that there is no restriction of $|\alpha| \leq |\beta|$ of context-sensitive grammars.□

If we consider the above productions as $\alpha \to \beta$, then for type-1 the condition is that $|\alpha| \leq |\beta|$. But, in type-0, called *unrestricted* grammar, there is no restricted like in type-1.

The formal languages are mostly based on the type-2 languages. The type 0 and 1 are not much understood yet, and difficult to implement.

## 14.4 Sentence Level Constructions

People use language to accomplish certain kinds of acts, broadly known as *speech acts*, which are distinct from physical acts, e.g., drinking a glass of water, or mental acts, like thinking about drinking a glass of water, etc. Speech acts also include asking for a glass of water, promising to drink a glass of water, threatening to drink a glass of water, ordering someone to drink a glass of water, and so on.

The most common classes are *declarative*, *imperative*, and *pragmatic*.

**1. Declarative Sentences**

$$\texttt{S} \to \texttt{NP VP} \tag{14.6}$$

The act in these sentences is *assertive*. These sentences conveys information, either true or false. The sentence parsed using the parse-tree shown in Fig. 14.1 (page no. 14-3), is a declarative sentence. Other examples of declarative sentences are:

"Jenny got an A on the test"

"The girls took photos"

"Biak took the food"

**2. Imperative Sentences** The speech acts corresponding to these sentences are *Orders* and *Requests*. These sentences begin with 'verb'. For example, the sentences: "Show the lowest fare" and "List all the scores", are imperative sentences. Naturally, the sentences should start with verb-phrasesS. The top level productions rules for generating *Imperative sentences* sentences are:

$$\begin{aligned} \texttt{S} &\rightarrow \texttt{VP} \\ \texttt{VP} &\rightarrow \texttt{V NP} \end{aligned} \tag{14.7}$$

The parse tree for the sentence "Book that flight." is shown in Fig. 14.2.

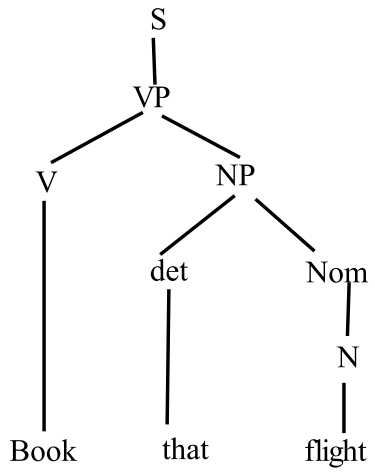


Figure 14.2: Parsing of: "Book that flight"

The other substitutions rule for verb, are given in rule (14.2) on page no. 14-3.

**3. Pragmatic Sentences** *Pragmatic* means practical or logical, i.e., you tend to think in terms of the practical or logical rather than the ideal situation. Words can mean different things, and often the same word can mean something different depending on the context in which it is used. A pragmatic view means that one does not think in ideal or abstract terms. In other words, they look at how we apply these words in practical, everyday language.

The examples of pragmatic sentences.

"Do all these flights have stops?"

"Can you give me the some information?"

"Do any of these flights stop at Mombai?"

"Does Air India flight 96UP serve dinner?"

The substitution rule for the pragmatic sentences is:

$$\texttt{S} \rightarrow \texttt{Aux NP VP} \tag{14.8}$$

In addition, we have the following more examples, which start with "Which", "What", "Who", "Whose", "How", "Where". Following are some examples.

"Which is flight to Delhi?" (POS tag is: Wh-determ.)

"What Airlines fly from Delhi?" (Wh-pronoun)

"Where this flight goes"? (Wh-adverb))

"Who is captain of this Flight"? (Wh-pronoun)

"Whose ticket is this"? (Wh-Possess ProN)

The Penn Treebank part-of-speech tagset of these "Wh-" things are given in table in other lecture notes (POS).

Corresponding production rules are as follows:

$$\begin{aligned}
\texttt{Wh-determ} &\rightarrow \texttt{Which} \\
\texttt{Wh-pronoun} &\rightarrow \texttt{What} \\
\texttt{Wh-adverb} &\rightarrow \texttt{Where} \\
\texttt{Wh-pronoun} &\rightarrow \texttt{Who} \\
\texttt{Wh-Possess ProN} &\rightarrow \texttt{Whose}
\end{aligned}$$

Instead `Wh-pronoun`, we can also use `Wh-NP`, e.g., "`Wh-NP` → `Wh-pronoun`". Consider the sentence,

"What flights do you have from Delhi to Mumbai?",

we shall apply the first rule as: `S` → `Wh-NP Aux NP VP` to generate the sentence.

$$
\begin{aligned}
\texttt{S} &\Rightarrow \texttt{Wh-NP Aux NP VP} \\
&\Rightarrow What\ \ flights\ \ \texttt{Aux NP VP} \\
&\Rightarrow What\ \ flights\ \ do\ \texttt{NP VP} \\
&\Rightarrow What\ \ flights\ \ do\ you\ \texttt{VP} \\
&\Rightarrow What\ \ flights\ \ do\ you\ V\ \ PP \\
&\Rightarrow What\ \ flights\ \ do\ you\ have\ \ Prep\ \ NP \\
&\Rightarrow What\ \ flights\ \ do\ you\ have\ \ from\ \ N\ \ PP \\
&\Rightarrow What\ \ flights\ \ do\ you\ have\ \ from\ \ Delhi\ \ PP \\
&\Rightarrow What\ \ flights\ \ do\ you\ have\ \ from\ \ Delhi\ \ prep\ \ NP \\
&\Rightarrow What\ \ flights\ \ do\ you\ have\ \ from\ \ Delhi\ \ to\ \ Mumbai
\end{aligned}
$$

Many times, the longer sentences are conjuncted together using connectives, e.g., "I will fly to Delhi and Mumbai". The corresponding rule is "`S` $\rightarrow$ `NP` and `NP`". Similarly, there is "`S` $\rightarrow$ `S` and `D`" and "`VP` $\rightarrow$ `VP` and `VP`".

Following are examples showing the more production rules, using which we can generate the sentences of different classes discussed above:

$$
\begin{aligned}
S &\rightarrow NP\ \ VP \mid VP \mid Aux\ NP\ \ VP \\
NP &\rightarrow Det\ \ Nom \mid prop\text{-}N \\
N &\rightarrow flight \mid meal \\
Nom &\rightarrow Noun\ \ Nom \mid Nom\ \ PP \mid N \\
Proper\text{-}N &\rightarrow Mumbai \mid Delhi \mid India \mid USA \mid ... \\
PP &\rightarrow Prep\ \ NP \\
VP &\rightarrow V \mid V\ NP \\
V &\rightarrow book \mid include \mid proper \\
Aux &\rightarrow Does \\
prep &\rightarrow from \mid to \mid on \mid near \\
Det &\rightarrow a \mid an \mid the \mid that \mid this
\end{aligned}
$$

In the above, *Aux* is Auxiliary Verb, *prop-N* is proper-noun, *Nom* is Nomial, *PP* is prepositional phrase, and *Prep* is preposition, e.g., *in, at, on, of, to.* Following are some rules and sentences constructed using them.

$S \rightarrow NP\ VP$: I prefer a morning flight (See Fig. 14.3)
$VP \rightarrow V\ NP$: Prefer a morning flight
$VP \rightarrow V\ NP\ PP$: Leaves Mumbai in the morning
$VP \rightarrow V\ PP$: Leaving on Tuesday
$PP \rightarrow Prep\ NP$: from New Delhi.

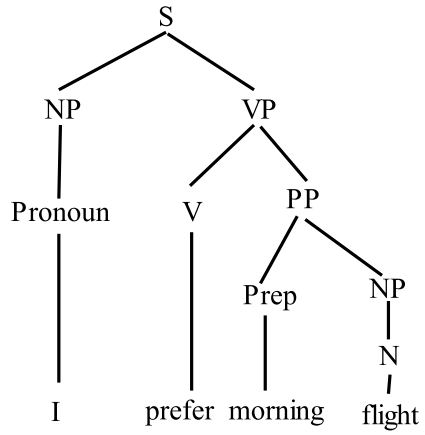


Figure 14.3: Parse-Tree for "I prefer morning flight"

The $NP$ in the $PP$ can be also be location, date, time or others.

The components of a sentence, i.e., noun, pronoun, adjective, verb, adverb, determiner, preposition, and conjunction, are called parts-of-speech. Following are the rewrite rules where LHS is a part of speech.

$$
\begin{aligned}
N &\rightarrow flights \mid breeze \mid trip \mid morning \mid ... \\
V &\rightarrow is \mid prefer \mid like \mid need \mid want \mid fly \\
Adj &\rightarrow cheapest \mid non\text{-}stop \mid first \mid latest \mid other \mid direct... \\
Pronoun &\rightarrow me \mid I \mid you \mid it \mid ... \\
Conj &\rightarrow and \mid or \mid but
\end{aligned}
$$