# CHAPTER 4

# SYNTACTIC ANALYSIS

## CHAPTER OVERVIEW

This chapter introduces a number of important notions about syntax. Syntactic parsing deals with the syntactic structure of a sentence. In many languages, words are brought together to form larger groups termed constituents or phrases, which can be modelled using context-free grammar. Context-free grammar is a set of rules or productions that tell which elements can occur in a phrase and in what order. This chapter discusses phrase structural rules for various phrases and introduces feature structures to efficiently capture the properties of grammatical categories. The parsing strategies covered here include top-down and bottom-up parsing, dynamic programming parsing algorithms (namely, Earley and CYK), and probabilistic CYK parsing. Finally, a framework based on Paninian grammar is discussed for Hindi.

## 4.1 INTRODUCTION

In Chapter 3, we talked about word level analysis. We now move on to higher level constituents like phrases and sentences. The word 'syntax' refers to the grammatical arrangement of words in a sentence and their relationship with each other. The objective of syntactic analysis is to find the syntactic structure of the sentence. This structure is usually depicted as a tree, as shown in Figure 4.1. Nodes in the tree represent the phrases and leaves correspond to the words. The root of the tree is the whole sentence. Identifying the syntactic structure is useful in determining the meaning of the sentence. The identification is done using a process known as parsing. Syntactic parsing can be considered as the process of assigning 'phrase markers' to a sentence (Charniak 1997). One must, therefore, have a clear understanding of what phrases are. We will describe various phrases that could appear in a language so that we can specify grammar

Researchers have proposed a number of parsing methods for natural language sentences. It is not possible to discuss all of them in a chapter, so we focus on a few widely-known ones. Most text books use only English for their discussions. We differ by including Indian languages as well. In addition to an introduction to grammar formalism, this chapter also provides an introduction to Hindi grammar.

Two important ideas in natural language are those of constituency and word order. Constituency is about how words are grouped together and how we know that they are really grouping together. Word order is about how, within a constituent, words are ordered with respect to one another, and also how constituents are ordered with respect to one another.

A widely used mathematical system for modelling constituent structure in natural language is context-free grammar (CFG) also known as phrase structure grammar. We begin our discussion with an overview of CFG in Section 4.2. In Section 4.3, we discuss various types of phrases and phrase structure rules. We then introduce feature structures in Section 4.4, to capture certain properties of grammatical categories which cannot be handled efficiently using CFG. We discuss probabilistic grammar and CYK parser based on this in Section 4.5. The CFG is basically a positional grammar and is not suitable for Indian languages, which are free word order languages. We provide a brief overview of Paninian grammar (PG), which is suitable for modelling free word order languages, in Section 4.6. We also discuss a parsing framework proposed by Bharti and Sangal (1990) for Indian languages based on this grammar. Finally, we provide a brief summary of the chapter.

## 4.2 CONTEXT-FREE GRAMMAR

Context-free grammar (CFG) was first defined for natural language by Chomsky (1957) and used for the Algol programming language by Backus (1959) and Naur (1960). A CFG (also called phrase-structure grammar) consists of four components:

1. A set of non-terminal symbols, N
2. A set of terminal symbols, T
3. A designated start symbol, S, that is one of the symbols from N.
4. A set of productions, P, of the form:

$$A \rightarrow \alpha$$

where $A \in N$ and $\alpha$ is a string consisting of terminal and non-terminal symbols. The rule $A \rightarrow \alpha$ says that constituent A can be rewritten as $\alpha$. This is also called the phrase structure rule. It specifies which elements

(or constituents) can occur in a phrase and in what order. For example, the rule S → NP VP states that S consists of NP followed by VP, i.e., a sentence consists of a noun phrase followed by a verb phrase.

A language is usually defined through the concept of derivation. The basic operation is that of rewriting a symbol appearing on the left hand side of production by its right hand side. A CFG can be used to generate a sentence or to assign a structure to a given sentence. When used as a generator, the arrows in the production rule may be read as 'rewrite the symbol on the left with symbols on the right'. Consider the toy grammar shown in Figure 4.1. The symbol S can be rewritten as NP VP using Rule 1, then using rules R2 and R4, NP and VP are rewritten as N and V NP respectively. NP is then rewritten as Det N (R3). Finally, using rules R6 and R7, we get the sentence:

*Hena reads a book.* (4.1)

We say that the sentence (4.1) can be derived from S. The representation of this derivation is shown in Figure 4.1. Sometimes, a more compact bracketed notation is used to represent a parse tree. The parse tree in Figure 4.1 can be represented using this notation as follows:

$[_S\ [_{NP}\ [_N$ Hena$]]\ [_{VP}\ [_V$ reads$]\ [_{NP}\ [_{Det}$ a$]\ [_N$ book$]]]]$

The set of all the strings containing terminal symbols which can be derived from the start symbol of the grammar, defines the language generated by that grammar. The parse tree shown in Figure 4.1 essentially represents a mapping of a string to its parse tree. This mapping process is called parsing. We will come back to this issue in Section 4.4.

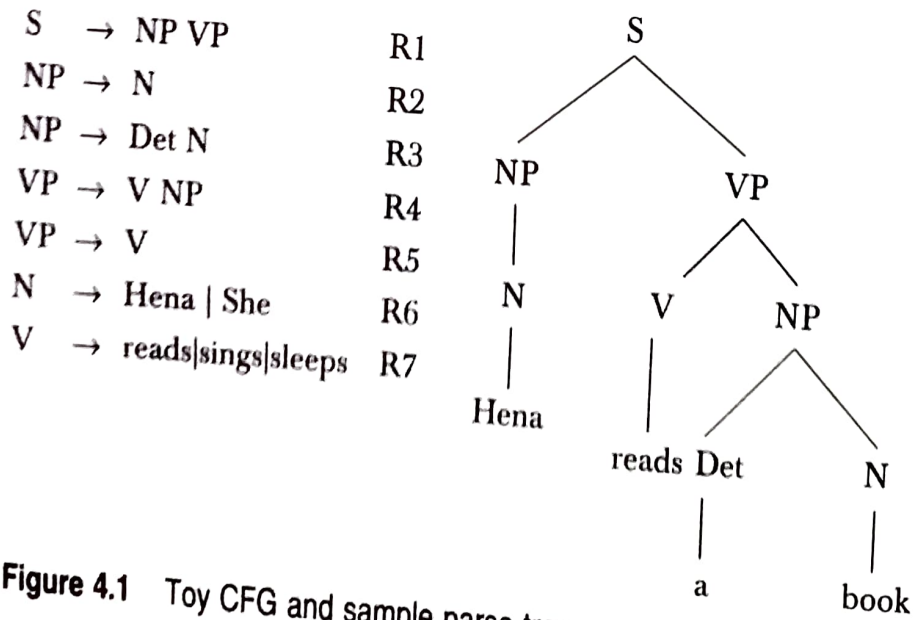


**Figure 4.1** Toy CFG and sample parse tree

## 4.3 CONSTITUENCY

Words in a sentence are not tied together as a sequence of part-of-speech. Language puts constraints on word order. For example, certain words go together with each other more than with others, and seem to behave as a unit. The fundamental idea of syntax is that words group together to form constituents (often termed phrases), each of which acts as a single unit. They combine with other constituents to form larger constituents, and eventually, a sentence. *The bird, The rain, The Wimbledon court, The beautiful garden* are all noun phrases that can occur in the same syntactic context. For example, they can all function as the subject or the object of a verb. These constituents combine with others to form a sentence constituent. For example, the noun phrase, *The bird*, can combine with the verb phrase, *flies,* to form the sentence, *The bird flies.* Different types of phrases have different internal structures. In this section, we discuss some of the major phrase types and try to build phrase structure rules to identify them.

### 4.3.1 Phrase Level Constructions

As discussed earlier, a fundamental notion in natural language is that certain groups of words behave as constituents. These constituents are identified by their ability to occur in similar contexts. One of the simplest ways to decide whether a group of words is a phrase, is to see if it can be substituted with some other group of words without changing the meaning. If such a substitution is possible then the set of words forms a phrase. This is called the substitution test. Consider sentence (4.1). We can substitute a number of other phrases:

*Hena reads a book.*
*Hena reads a storybook.*
*Those girls read a book.*
*She reads a comic book.*

We can easily identify the constituents that can be replaced for each other in these sentences. These are *Hena, she,* and *Those girls* and *a book, a storybook,* and *a comic book.* These are the words that form a phrase. In linguistics, such constituents represent a paradigmatic relationship. Elements that can substitute each other in certain syntactic positions are said to be members of one paradigm.

Phrase types are named after their head, which is the lexical category that determines the properties of the phrase. Thus, if the head is a noun, the phrase is called a noun phrase, if the head is a verb, the phrase is

called a verb phrase; and so on for other lexical categories such as adjective and preposition. Figure 4.2 shows a sentence with a noun phrase, verb phrase, and preposition phrase.

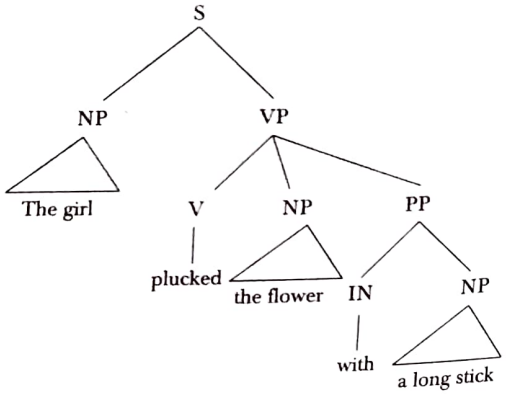


**Figure 4.2** A sentence with NP, VP, and PP

### *Noun Phrase*

A noun phrase is a phrase whose head is a noun or a pronoun, optionally accompanied by a set of modifiers. It can function as subject, object, or complement. The modifiers of a noun phrase can be determiners or adjective phrases. The obligatory constituent of a noun phrase is the noun head—all other constituents are optional. These structures can be represented using the phrase structure rule. As discussed earlier, phrase structure rules are of the form $A \rightarrow BC$, which states that constituent A can be rewritten as two constituents B and C. These rules specify which elements can occur in a phrase and in what order. Using this notation, we can represent the phrase structure rules for a noun phrase as follows.

$NP \rightarrow Pronoun$
$NP \rightarrow Det\ Noun$
$NP \rightarrow Noun$
$NP \rightarrow Adj\ Noun$
$NP \rightarrow Det\ Adj\ Noun$

We can combine all these rules in a single phrase structure rule as follows:

$NP \rightarrow (Det)\ (Adj)\ Noun|Pronoun$

The constituents in parentheses are optional. This rule states that a noun phrase consists of a noun, possibly preceded by a determiner and an adjective (in that order). This rule does not cover all possible NPs. A noun phrase may include post-modifiers and more than one adjective. For example, it may include a prepositional phrase (PP). More than one adjective is handled by allowing an adjective phrase (AP) for the adjective in the rule. After incorporating PP and AP in the phrase structure rule, we get the following.

$NP \rightarrow (Det)\ (AP)\ Noun\ (PP)$

The following are a few examples of noun phrases:

They
*The foggy morning* (4.2a)
*Chilled water* (4.2b)
*A beautiful lake in Kashmir* (4.2c)
*Cold banana shake* (4.2d)
(4.2e)

Let us see how the phrases (4.2a–e) can be generated using phrase structure rules. The phrase (4.2a) consists only of a pronoun; (4.2b) consists of a determiner, an adjective (foggy) that stands for an entire adjective phrase, and a noun; (4.2c) comprises an adjective phrase and a noun; (4.2d) consists of a determiner (the), an adjective phrase (beautiful), a noun (lake), and a prepositional phrase (in Kashmir); and (4.2e) consists of an adjective followed by a sequence of nouns. A noun sequence is termed as nominal. None of the phrase structure rules discussed so far are able to handle nominals. So, we modify our rules to cover this situation.

$NP \rightarrow (Det)\ (AP)\ Nom\ (PP)$
$Nom \rightarrow Noun \mid Noun\ Nom$

A noun phrase can act as a subject, an object, or a predicate. The following sentences demonstrate each of these uses.

*The foggy damped weather* disturbed the match. (4.3a)
I would like a *nice cold banana shake*. (4.3b)
Kula botanical garden is a *beautiful location*. (4.3c)

In (4.3a), the noun phrase acts as a subject. In (4.3b), it acts as an object, and in (4.3c), it is a predicate.

### *Verb Phrase*

Analogous to the noun phrase is the verb phrase, which is headed by a verb. There is a fairly wide range of phrases that can modify a verb. This makes verb phrases a bit more complex. The verb phrase organizes various elements of the sentence that depend syntactically on the verb.

The following are some examples of verb phrases:

Khushbu *slept*. (4.4a)
The boy *kicked the ball*. (4.4b)

Khushbu *slept in the garden.*
The boy *gave the girl a book.*
The boy *gave the girl a book with blue cover.* (4.4

As you can see from these examples a verb phrase can have a verb [VP → Verb in (4.4a)]; a verb followed by an NP [VP → Verb NP in (4.4b)]; a verb followed by a PP [VP → Verb PP in (4.4c)]; a verb followed by two NPs [VP → Verb NP NP in (4.4d)]; or a verb followed by two NPs and a PP [VP → Verb NP NP PP in (4.4e)]. In general, the number of NPs in a VP is limited to two, whereas it is possible to add more than two PPs.

VP → Verb (NP) (NP) (PP)*

Things are further complicated by the fact that objects may also be entire clauses as in the sentence, *I know that Taj is one of the seven wonders.* Hence, we must also allow for an alternative phrase statement rule, in which NP is replaced by S.

VP → Verb S

***Prepositional Phrase***

Prepositional phrases are headed by a preposition. They consist of a preposition, possibly followed by some other constituent, usually a noun phrase.

We played volleyball *on the beach.*

We can have a preposition phrase that consists of just a preposition.

John went *outside.*

The phrase structure rule that captures the above eventualities is as follows.

PP → Prep (NP)

***Adjective Phrase***

The head of an adjective phrase (AP) is an adjective. APs consist of an adjective, which may be preceded by an adverb and followed by a PP. Here are few examples.

Ashish is *clever.*
The train is *very late.*
My sister is *fond of animals.*

The phrase structure rule for adjective phrase is

AP → (Adv) Adj (PP)

***Adverb Phrase***

An adverb phrase consists of an adverb, possibly preceded by a degree adverb. Here is an example.



Time passes *very quickly.*

AdvP → (Intens) Adv

### 4.3.2 Sentence Level Constructions

Having discussed phrase structures, we now focus our attention on sentences. A sentence can have varying structure. The four commonly known structures are declarative structure, imperative structure, yes-no question structure, and wh-question structure.

Sentences with a declarative structure have a subject followed by a predicate. The subject of a declarative sentence is a noun phrase and the predicate is a verb phrase, e.g., *I like horse riding.* The phrase structure rule for declarative sentences is

S → NP VP

Sentences with an imperative structure usually begin with a verb phrase and lack subject. The subject of these types of sentence is implicit and is understood to be 'you'. These types of sentences are used for commands and suggestions, and hence are called imperative. The grammar rule for this kind of sentence structure is

S → VP

Examples of this kind of sentences are as follows:

*Look at the door.*
*Give me the book.*
*Stop talking.*
*Show me the latest design.*

Sentences with the yes-no question structure ask questions which can be answered using yes or no. These sentences begin with an auxiliary verb, followed by a subject NP, followed by a VP. Here are some examples:

Do you have a red pen?
Is there a vacant quarter?
Is the game over?
Can you show me your album?

We expand our grammar by adding another rule for the expansion of S, as follows:

S → Aux NP VP

Sentences with wh-question structure are more complex. These sentences begin with a wh-words—who, which, where, what, why, and how. A wh-question may have a wh-phrase as a subject or may include another subject. Consider the following wh-question:

Which team won the match?

This sentence is similar to a declarative sentence except that it contains a wh-word. A simple rule to handle this type of sentence structure is

S → Wh-NP VP

Another type of wh-question structure is one that involves more than one NP. In this type of questions, the auxiliary verb comes before the subject NP, just as in yes-no question structures.

*Which cameras can you show me in your shop?*

The rule for this type of wh-questions is

S → Wh-NP Aux NP VP

A simplified view of the grammar rules discussed so far is summarized in Table 4.1.

**Table 4.1** Summary of grammar rules

| | | |
|---|---|---|
| S | → | NP VP |
| S | → | VP |
| S | → | Aux NP VP |
| S | → | Wh-NP VP |
| S | → | Wh-NP Aux NP VP |
| NP | → | (Det) (AP) Nom (PP) |
| VP | → | Verb (NP) (NP) (PP)* |
| VP | → | Verb S |
| AP | → | (Adv) Adj (PP) |
| PP | → | Prep (NP) |
| Nom | → | |

### Coordination

The grammar rules in Table 4.1 are not exhaustive. There are other sentence-level structures that cannot be modelled by the rules discussed here. Coordination is one such structure. It refers to conjoining phrases with conjunctions like 'and', 'or', and 'but'. For example, a coordinate noun phrase can consist of two other noun phrases separated by a conjunction 'and', as in

*I ate [NP [NP an apple] and [NP a banana]].*

Similarly, verb phrases and prepositional phrases can be conjoined as follows:

*It is [VP [VP dazzling] and [VP raining]].*

Not only that, even a sentence can be conjoined.

*[S [S I am reading the book] and [S I am also watching the movie]]*



We need to devise rules to handle these constructions. Conjunction rules for NP, VP, and S can be built as follows:

NP → NP and NP
VP → VP and VP
S → S and S

### Agreement

Most verbs use two different forms in present tense—one for third person, singular subjects, and the other for all other kinds of subjects. The third person singular (3sg) form ends with a -s whereas the non-3sg does not. Whenever there is a verb that has some noun acting as a subject, this agreement has to be confirmed. Here are a few examples that demonstrate how the subject NP affects the form of the verb.

Does [$_{NP}$ Priya] sing? (4.5a)
Do [$_{NP}$ they] eat? (4.5b)

In the first sentence, the subject NP is singular. Hence, the -es form of 'do', i.e. 'does' is used. The second sentence has a plural NP subject. Hence, the form 'do' is being used. Sentences in which subject and verb do not agree are ungrammatical. The following sentences are ungrammatical:

[Does] they eat? (4.5 c)
[Do] she sings? (4.5 d)

Sentences (4.5c) and (4.5d) point out that a grammatical phenomenon can lead to ungrammatical sentences—a problem known as over-generation. Rules that handle the yes-no questions of Example 4.5 are as follows:

S → Aux NP VP

To take care of the subject-verb agreement, we replace this rule with a pair of rules as follows:

S → 3sgAux 3sgNP VP
S → Non3sgAux Non3sgNP VP

These rules ensure appropriate subject-verb agreement. We could add rules for the lexicon like these:

3sg Aux → does| has| can
Non3sg Aux → do | have | can

Similarly, rules for 3sgNP and Non3sgNP need to be added. So we replace each of the phrase structure rules for noun phrase by a pair of rules as follows:

$$\begin{bmatrix} FEATURE_1 & VALUE_1 \\ FEATURE_2 & VALUE_2 \\ \ldots & \ldots \\ FEATURE_n & VALUE_n \end{bmatrix}$$

**Figure 4.3** An attribute value matrix (AVM)

An AVM consisting of a single NUMBER feature with the value SG is represented as follows:

$$[NUMBER \quad SG]$$

The value of a feature can be left unspecified and represented by an empty pair of square brackets, as in the following example:

$$[NUMBER \quad []]$$

The feature structure can be used to encode the grammatical category of a constituent and the features associated with it. For example, the following structure represents the third person singular noun phrase.

$$\begin{bmatrix} CAT & NP \\ NUMBER & SG \\ PERSON & 3 \end{bmatrix}$$

Similarly, a third person plural noun phrase can be represented as follows:

$$\begin{bmatrix} CAT & NP \\ NUMBER & PL \\ PERSON & 3 \end{bmatrix}$$

The values of CAT and PERSON features remain the same in both structures. This explains how feature structures aid in generalization while making the necessary distinction possible. As mentioned earlier, feature values are not limited to atomic symbols. A feature can have another feature structure as its value. Consider the case of combining the NUMBER and PERSON features into a single AGREEMENT feature. This makes sense because grammatical subjects must agree with their predicates in NUMBER as well as PERSON properties. Using this new feature, we represent the grammatical category third person plural noun phrase by the following structure:

$$\begin{bmatrix} CAT & NP \\ & \end{bmatrix}$$

In order for feature structures to be useful, we must be able to perform operations on them. The two most important operations we need to perform are merging the information content of the two structures that are similar and rejecting structures that are incompatible. The computational technique that is used to perform these operations is called unification. Unification is implemented as a binary operator (⊔) that takes two feature structures as arguments and returns a merged feature structure if they are compatible, otherwise reports a failure. Here is a simple application of the unification operator for performing an equality check.

$$[NUMBER \quad PL] \sqcup [NUMBER \quad PL] = [NUMBER \quad PL]$$

The unification succeeds as the two structures have the same value for the NUMBER feature. A feature with an unspecified value in one structure, can be successfully matched with any value in a corresponding feature in another structure. In such cases, the unification operation produces a structure with the value provided by the structure having non-null value. For example,

$$[NUMBER \quad PL] \sqcup [NUMBER \quad []] = [NUMBER \quad PL]$$

In this example, the two structures are considered compatible and merged into a structure with PL as its value for the NUMBER feature. The value PL of the first structure matches the value [] of the second structure and becomes the value of the NUMBER feature of the output structure.

However, the following application of unification results in failure as the NUMBER features of the first and second structures have incompatible values.

$$[NUMBER \quad PL] \sqcup [NUMBER \quad SG] \text{ Fails}$$

The CFG rules can have feature structures attached to them to realize constraints on the constituents of the sentence.

## PARSING

A CFG defines the syntax of a language but does not specify how structures are assigned. The task that uses the rewrite rules of a grammar to either generate a particular sequence of words or reconstruct its derivation (or phrase structure tree) is termed parsing. A phrase structure tree constructed from a sentence is called a parse. The syntactic parser is thus responsible for recognizing a sentence and assigning a syntactic structure to it. It is possible for many different phrase structure trees to derive the same sequence of words. This means a sentence can have multiple parses. This phenomenon is called syntactic ambiguity.



Garden pathing is another phenomenon related to syntactic parsing. It refers to the process of constructing a parse by exploring the parse tree along different paths, one after the other till, eventually, the right one is found. The popular example given for garden pathing is the sentence

*The horse ran past the barn fell.* (4.6)

In the first attempt, most of us come up with a parse corresponding to the sentence *The horse ran past the barn*, leaving no possibility for the word *fell* to be incrementally added in the sentence. In order to complete the parse of the sentence, we have to backtrack.

Finding the right parse can be viewed as a search process. The search finds all trees whose root is the start symbol S and whose leaves cover exactly the word in the input. The search space in this conception corresponds to all possible parse trees defined by the grammar. The following constraints guide the search process.

1. *Input*: The first constraint comes from the words in the input sentence. A valid parse is one that covers all the words in a sentence. Hence, these words must constitute the leaves of the final parse tree.
2. *Grammar*: The second kind of constraint comes from the grammar. The root of the final parse tree must be the start symbol of the grammar.

These two constraints give rise to the two most widely used search strategies by parsers, namely, top-down or goal-directed search and bottom-up or data-directed search.

### 4.4.1 Top-down Parsing

As the name suggests, top-down parsing starts its search from the root node S and works downwards towards the leaves. The underlying assumption here is that the input can be derived from the designated start symbol, S, of the grammar. The next step is to find all sub-trees which can start with S. To generate the sub-trees of the second-level search, we expand the root node using all the grammar rules with S on their left hand side. Likewise, each non-terminal symbol in the resulting sub-trees is expanded next using the grammar rules having a matching non-terminal symbol on their left hand side. The right hand side of the grammar rules provide the nodes to be generated, which are then expanded recursively. As the expansion continues, the tree grows downward and eventually reaches a state where the bottom of the tree consist only of part-of-speech categories. At this point, all trees whose leaves do not match words in the input sentence are rejected, leaving only trees that represent successful