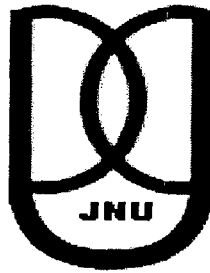


**ELEMENTARY-GB-TREES  
BASED PARSING FOR SANSKRIT**

*Dissertation submitted to Jawaharlal Nehru University, in partial  
fulfillment of the requirements for the award of the degree of*

Master of Technology  
in  
Computer Science and Technology  
by  
**Ch Shiva Prasad**

*Under the Esteemed Supervision*  
of  
**Prof. G. V. Singh**



**SCHOOL OF COMPUTER & SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI -110067**

**July- 2006**



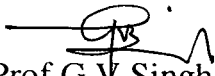
जवाहरलाल नॅहरू विश्वविद्यालय

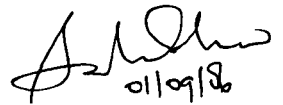
SCHOOL OF COMPUTER & SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI- 110067, INDIA

## CERTIFICATE

This is to certify that the project entitled “**ELEMENTARY-GB-TREES BASED PARSING FOR SANSKRIT**” being submitted by **CH SHIVA PRASAD** to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science & Technology, is a bonafide work carried out by him under the guidance and supervision of Prof.G.V.Singh.

The matter embodied in the dissertation has not been submitted for the award of any other degree or diploma.

  
Prof.G.V.Singh  
Professor, SC&SS,  
JNU, New Delhi-67

  
Prof. Balasundaram  
Dean, SC&SS,  
JNU, New Delhi-67



# जवाहरलाल नॅहरू विश्वविद्यालय

SCHOOL OF COMPUTER & SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI- 110067, INDIA

## DECLARATION

This is to certify that the project entitled “**ELEMENTARY-GB-TREES  
BASED PARSING FOR SANSKRIT**” is being submitted to the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science & Technology, is a bonafide work carried out by me.

The matter embodied in the dissertation has not been submitted for the award of any other degree or diploma.

Ch Shiva Prasad,  
M.Tech, Final Semester,  
SC & SS, JNU  
New Delhi.

**Dedicated to**  
**My beloved**  
**Parents and Sisters**

## **ACKNOWLEDGEMENTS**

I would like to pay obeisance at the feet of my parents for their blessings that are always with me in all my aspirations including my academics.

I would like to sincerely thank my supervisor Prof.G.V.Singh, School of Computer And Systems Sciences, Jawaharlal Nehru University for the help, encouragement and support extended by him in successful completion of this project. His innovative ideas and the valuable discussions we had were very much helpful in keeping the thesis work on the right track.

I would like to gratefully acknowledge Mr. Manjit Singh for his valuable discussions and guidance during development of this project.

I would like to record my sincere thanks to Dean, Prof. Bala Sundaram for providing the necessary facilities. I will fail my duty if I forget my appreciation and thanks to my friends. I also extend my sincere gratitude to my lab mates for their continuous academic as well as morale support through out my dissertation work.

Last, but not the least, I take this opportunity to thank all the faculty members and friends for their help and encouragement during the course of the thesis work.

**(Ch Shiva Prasad)**

# CONTENTS

## Chapter 1

### INTRODUCTION

<b>1.1 NATURAL LANGUAGE PROCESSING</b>	1
1.1.1 Speech Recognition	2
1.1.2 Speech Synthesis	2
1.1.3 Natural Language Understanding	2
1.1.4 Natural Language Generation	3
1.1.5 Language Translation	3
<b>1.2 MACHINE TRANSLATION</b>	3
<b>1.3 NATURAL LANGUAGE PARSING SYSTEM</b>	4
<b>1.4 PROBLEM DEFINITION</b>	5
<b>1.5 ORGANISATION OF THE THESIS</b>	5

## Chapter 2

### GOVERNMENT AND BINDING THEORY

<b>2.1 INTRODUCTION</b>	7
<b>2.2 X-BAR LEVELS AND PHRASE STRUCTURES</b>	8
2.2.1 The Phrasal Projection for a Verb	10
2.2.2 The Phrasal Projection for a Noun	11
2.2.3 The Phrasal Projection for an Adjective	11
2.2.4 The Phrasal Projection for a Preposition	12
2.2.5 The Phrasal Projection for a Sentence (Inflection Phrase)	13
2.2.6 The Phrasal Projection for Complementizer	14
<b>2.3. LEXICON ORGANIZATION</b>	14
2.3.1 Categorical Information	15
2.3.2 Subcategorization Information	15
2.3.3 Thematic Information	16

<b>2.4 THETA THEORY</b>	18
2.4.1 The Argument Structure of Other Syntactic Categories	19
2.4.2 The Theta Criterion	20
2.4.3 The Projection Principle	21
2.4.4 The Extended Projection Principle	21
<b>2.5 OTHER CONCEPTS</b>	22
2.5.1 Case Theory	22
<b>Chapter 3</b>	
<b>GOVERNAMENT AND BINDING BASED GRAMMAR FOR SANSKRIT</b>	
<b>3.1 INTRODUCTION</b>	23
<b>3.2 FORMS OF DIFFERENT GRAMMATICAL CATEGORIES</b>	24
3.2.1 Noun Forms in Sanskrit (Declensions of Nouns)	24
3.2.2 Adjective Form	26
3.2.3 Verb Forms (Verb Conjugation)	27
<b>3.3 SANSKRIT PHRASE STRUCTURES</b>	34
3.3.1 Verb Phrase	34
3.3.2 Noun Phrase	35
3.3.3 Inflection Phrase	36
3.3.4 Complementizer Phrase	37
<b>Chapter 4</b>	
<b>LEXICON</b>	
<b>4.1 INTRODUCTION</b>	38
<b>4.2 SANSKRIT LEXICON</b>	38
4.2.1 Introduction	38
<b>4.3 DESIGN OF LEXICON</b>	41
4.3.1 Noun Table	42
4.3.2 Pronoun Table	42
4.3.3 Verb Table	43
4.3.4 Interface	43

<b>4.4 SUMMARY</b>	<b>44</b>
<b>Chapter 5</b>	
<b>PARSER</b>	
<b>5.1 PARSING STRATEGIES</b>	<b>45</b>
5.1.1 An Overview	45
5.1.2 Bottom-Up Parser (LR Parsing Algorithm)	46
<b>5.2 DESIGN OF PARSER</b>	<b>47</b>
5.2.1 Our Parsing Strategy	47
5.2.2 Over View of Design	49
<b>5.3 EXPLANATION OF PARSER WITH AN EXAMPLE</b>	<b>55</b>
5.3.1 With a Correct Sentence's	55
5.3.2 with a Wrong Sentence	69
<b>5.4 SUMMARY</b>	<b>70</b>
<b>Chapter 6</b>	<b>71</b>
<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>71</b>
6.1 CONCLUSION	71
6.2 FUTURE ENHANCEMENTS	72
<b>APPENDIX</b>	<b>74</b>
<b>REFERENCES</b>	<b>75</b>



# ABSTRACT

An Elementary-GB-Trees Based Parsing system for Sanskrit language sentences has been developed, which may be used to translate Sanskrit into any other language. Various translations systems are being developed across the world using conventional approaches like Ruled- based or Example-based. We have adopted Government and Binding theory (GB) approach for parsing the sentences.

The GB theory with its emphasis on Universal Grammar, its universality in handling Natural Languages, and its computational properties led to its choice over other conventional approaches. The important modules of GB are X-Bar levels and phrase structures, Theta assignment module, Case assignment module, and Binding module

After a thorough analysis of various phrases in Sanskrit, GB Phrase Structures for Sanskrit has been developed. These include Verb Phrase Structure, Noun Phrase Structure, Adjective Phrase Structure, Preposition Phrase Structure, Inflection Phrase Structure and Complementizer Phrase Structure. The analysis includes determining the complements for each lexical type, determining adjuncts and specifiers for each type of Phrase Structure

Whenever a word is read by parser, the type of the word is to be found out. The type may be of verb, noun, preposition, adjective or any other. All this information is stored in the lexicon along with categorization, sub categorization and the number of complements it may take. As per the type of the word, corresponding elementary tree for the word is constructed.

Lexicon is the heart of our whole system, which is typical database containing information about all words. It contain all the words, it's tense, gender, number, person etc. It should contain the required information, using which we can construct a GB phrase tree. It also includes sub categorization information. We built our lexicon strong, so that performance of the system increases and the programming complexity also reduces

Whenever a word type is known from the lexicon, with the help of category and subcategories information, we built elementary tree for that particular word as per the type. Later these elementary trees will be combined to construct the full pledged phrase tree which satisfies the GB rules of language. At this Stage each word projects its own phrase structure tree.

At the end all these phrasal trees are combined to get the sentence phrasal tree. Sentence may result into more than one tree depending on the attributes of words involved.

# Chapter 1

## INTRODUCTION

### 1.1 NATURAL LANGUAGE PROCESSING

A Natural language (NL) is any of the languages naturally used by humans to communicate between them (e.g. Telugu, English, Hindi, Japanese, etc.). Programming Languages or man-made languages such as C, C++, C#, Java, etc. are known as artificial languages. Natural Language Processing (NLP) is a convenient description that attempt to make the computers analyze, understand, and generate Natural Languages, enabling one to address a computer in a manner as one is addressing a human being.

Understanding a human language is not an easy task. The main difficulty lies in knowing the relationship between words, phrases and the concepts they represent. A Natural Language, which is easy for humans to learn and use, is hard for a computer to master. Even long after machines have proven capable of inverting large matrices with speed and grace, they still fail to master the basics of our spoken and written languages.

The difficulties in computer processing of a Natural Language arise from the highly ambiguous nature of Natural Languages. Very simple sentences for humans to speak and understand easily, like “Flying planes can be dangerous”, can be very difficult to a computer that lacks knowledge of the world and a native speaker’s experience with the linguistic structures of the Natural Languages. Plausible interpretations of the sentence “Flying planes can be dangerous” could be that “the pilot is at risk”, or “there is a danger to people on the ground”. Further, should “can” be analyzed as a verb or as a noun. Which of the possible interpretations of “plane” is relevant? Depending on context, “plane” could refer to, among other things, an airplane, a geometric object, or a woodworking tool. How much and what sort of context needs to be brought in to bear on these questions in order to adequately disambiguate the

sentence? These are only the few challenges we face while processing a Natural Language.

The term Natural Language Processing represents any processing that is required or need to be done to understand, generate or interpret the utterances in a given language. But in the subsequent paragraphs we list only some main areas or domains of Natural Language Processing:

## **NATURAL LANGUAGE PROCESSING INCLUDES:**

### **1.1.1 Speech Recognition**

Speech recognition is the process of converting an acoustic signal, captured by a computer, microphone or a telephone, to a set of words. Since different people pronounce the same words differently, the mapping of those sounds to the words in the language turns out to be quite difficult.

### **1.1.1 Speech Synthesis**

Speech synthesis is the artificial production of human speech. A system used for this purpose is termed a speech synthesizer, and can be implemented in software or hardware. Speech synthesis systems are often called text-to-speech (TTS) systems in reference to their ability to convert text into speech. However, there exist systems that instead render symbolic linguistic representations like phonetic transcriptions into speech.

### **1.1.2 Natural Language Understanding**

Natural Language Understanding means moving from words, phrases or sentences (either in written form or derived by a speech recognition system) to 'meaning'. This involves mapping Natural Language units to their meanings as any Natural Language generates infinite number of valid units; mapping of these dynamically generated units to meaning is quite difficult.

### **1.1.3 Natural Language Generation**

Natural language generation is the natural language processing task of generating natural language from a machine representation system such as a knowledge base or a logical form. Some people view NLG as the opposite of natural language understanding. The difference can be put this way: whereas in natural language understanding the system needs to disambiguate the input sentence to produce the machine representation language, in NLG the system needs to take decisions about how to put a concept into words.

### **1.1.4 Language Translation**

Language translation generally referred as Machine translation (MT) is the application of computers to the task of translating texts from one natural language to another. One of the very earliest pursuits in computer science, MT has proved to be an elusive goal, but today a number of systems are available that produce output which, if not perfect, is of sufficient quality to be useful in a number of specific domains. Since the present work relates to Machine translation, we will explain this in some detail in the following sections.

## **1.2 MACHINE TRANSLATION**

Machine translation, sometimes referred to by the acronym MT, is a sub-field of computational linguistics that investigates the use of computer software to translate text or speech in between natural languages.

At its basic level, MT performs simple substitution of atomic words in one natural language for words in another. Using corpus techniques, more complex translations can be performed, allowing for better handling of differences in linguistic typology, phrase recognition, and translation of idioms, as well as the isolation of anomalies. However, current systems are unable to produce output of the same quality as a human translator, particularly where the text to be translated uses casual language.

### 1.3 NATURAL LANGUAGE PARSING SYSTEM

Language parsing plays the significant role in translating the natural language. In machine translation and natural language processing systems, human languages are parsed by computer programs. Human sentences are not easily parsed by programs, as there is substantial ambiguity in the structure of human language. In order to parse natural language data, researchers must first agree on the grammar to be used. The choice of grammar is affected by both linguistics and computational concerns; for instance some parsing systems use lexical functional grammar, but in general, parsing for grammars of this type is known to be NP-complete. Head driven Phrase Structure Grammar is another linguistic formalism which has been popular in the parsing community, but other research efforts have focused on less complex formalisms. Another popular strategy for avoiding linguistic controversy is dependency grammar parsing.

Parsing algorithms for natural language cannot rely on the grammar having 'nice' properties as with manually-designed grammars for programming languages. As mentioned earlier some grammar formalisms are very computationally difficult to parse: in general, even if the desired structure is not context-free, some kind of context-free approximation to the grammar is used to perform a first pass. Algorithms which use context-free grammars often rely on some variant of the CKY algorithm, usually with some heuristic to prune away unlikely analyses to save time. However some systems trade speed for accuracy using, e.g., linear-time versions of the shift-reduce algorithm. A somewhat recent development has been parse re-ranking in which the parser proposes some large number of analyses, and a more complex system selects the best option.

The Universal Grammar (UG) theory claims that the same principles are incorporated in the grammars of all languages; variation between languages amounts to differences in the settings for a limited number of parameters. The principles and parameters involved are couched in terms of the framework familiar in Chomskyan work of the 1980s, usually known as Government/Binding (GB) theory. A proper GB parser has then a strong resemblance to a model of language acquisition.

## 1.4 PROBLEM DEFINITION

For the reasons stated earlier in this Chapter we have decided to engage ourselves in parsing Sanskrit sentences, which in turn may be used to translate into another language. The thesis aims at building An Elementary-GB-Trees Based Parsing system for Sanskrit language.

The GB framework has been adopted for generating the phrasal trees for sentences. The key modules required in developing a parsing system are: building a Lexicon, and design and implementation of a parser. Bottom up approach will be the basis for the design and implementation of the parser. Since the GB framework forms the heart of the overall system, it also requires developing GB based grammar for language. Finally the overall system design and implementation is aimed to make the system available for parsing Sanskrit sentences. In the section 6 of this Chapter we present the organization of this thesis.

## 1.5 ORGANISATION OF THE THESIS

The thesis consists of 6 Chapters along with conclusions and future enhancements.

Chapter 1 overviews the field of Natural Language Processing, gives an overview of Machine Translation and also gives a brief introduction to parsing. Then we discuss the significance of Natural Language Processing. Finally we explain the problem definition.

Chapter 2 analyses the Government and Binding Theory (X-bar Theory). Here we analyze the phrasal projections for Verb, Noun, Adjective, Preposition, Inflection, and Complementizer Phrases

Chapter 3 briefly explains how Government and Binding rules are used for Sanskrit language and various phrasal projections for Nouns, Verbs, Adjectives and Adverbs. This chapter also covers grammar rules of Sanskrit language. We are confined to extent where grammar is sufficient for parsing the sentence. As part of the grammar different noun forms, verb forms, adjective forms, postpositions have been discussed.

Chapter 4 defines lexicon, how the vocabulary in a language is structured, how words were created and importance of Lexicon in parsing. Further it explains the different attributes associated with each grammatical category of Sanskrit. The last section explains the use of interface for storing the words in the database.

Chapter 5 gives the overview of the design of the parsing system. Parsing system includes six different Modules. We illustrate here the parsing Process in detail with examples.

Finally, the Chapter 6 concludes the work with suggestions for future. The Appendix A gives the Phrase Structure Rules for Hindi. References are placed at the end of the thesis.



## Chapter 2

# GOVERNMENT AND BINDING THEORY

### 2.1 INTRODUCTION

Government and binding is a theory of syntax in the tradition of Transformational grammar developed principally by Noam chomsky in the 1980's. This theory is a radical revision of his earlier theories and was later revised in The Minimalist Program (1995) and several subsequent papers —the latest being Three Factors in Language Design (2005). Although there is a large literature on government and binding theory which is not written by Chomsky, Chomsky's papers have been foundational in setting the research agenda.

The name refers to two central subtheories of the theory: Government, which is an abstract syntactic relation, and Binding, which deals with the referents of pronouns, anaphores, and R-expression. GB was the first theory to be based on the principles and parameters model of language, which also underlies the later developments of the Minimalist Program.

The shift from programming language grammars to NLP grammars seriously increases complexity and requires ways to handle the ambiguities inherent in every human language. While most programming languages are expressed by subclasses of well-understood context-free grammars (CFGs), no grammatical formalism has yet been accepted by the linguistic community for the description of human languages. On the contrary, new formalisms (or variants of older ones) appear constantly. These include, for instance, Linear Indexed Grammar (LIG) [18], Range Concatenation Grammar (RCG) [6], and Definite Clause Grammar (DCG) etc. More recent formalisms, like Head-Driven Phrasal Structure Grammar (HPSG) [7], Lexical Functional Grammar (LFG) [22], which relies on more expressive Typed Feature Structures (TFS) [5] or constraints, and Tree Adjoining Grammar (TAG) with trees as elementary structures [2] & [3] have been more

widely used by the Natural Language Processing community. However of all the grammar formalisms mentioned above, the Government and Binding (GB) Theory, with its emphasis on Universal Grammar, has had the most impact in NLP. Because of its very different approach in characterizing a language, its universality in handling Natural Languages and its interesting computational properties, we have adopted GB based approach for our work.

This Chapter briefly presents the Government and Binding (GB also referred as X-bar Theory) Theory. The X-bar Theory is claimed to be Universal making it very suitable for the task of Machine Translation. Here we will analyze X-bar structures of different phrasal structures like VP, NP, AP, PP, etc. This Chapter suggest that the distinction between Specifier, Adjuncts, and Complements are not only based on structural differences as suggested in the existing literature but to the large extent are determined by the relationship of syntax with the lexicon via projection principle, the argument structure of lexical categories, the theta theory, the case theory and the government and binding theory.

The main tenets of GB theory of syntax are developed by Chomsky [8]. We first explain the theory using English language and then later apply it to Hindi language used in the translation system. GB assumes that a large portion of the grammar of any particular language is common to all languages, and is therefore part of Universal Grammar. The next section will focus on introducing X-bar phrasal structures and introduces three level symmetrical rule schemata.

## **2.2 X-BAR LEVELS AND PHRASE STRUCTURES**

Words combine to make phrases, and phrases are one of the basic patterns out of which we build sentences. A phrase is a group of words which acts as a single unit in meaning and in grammar, and is not built round a verb. Phrases can have many different functions in a sentence. They are used as subjects, objects, complements, modifiers, or adverbials.

Understanding phrasal patterns helps us to discuss and explain the effects in our own and others' writing. In the sentence:

The strange green creatures with bobbing heads spoke.

- The phrase the strange green creatures with bobbing heads acts as the subject of the verb *spoke*. The phrase is a single unit both in its meaning and in its grammar.
- The fragment the strange green is not a phrase, because it has no separate meaning and no grammatical function.

For phrase structures we adopt symmetrical X-bar analysis. The same set of rules will be applicable to all phrases thus leading to uniformity and computational efficiency in terms of efforts, time and space. Further, we need exactly three levels of projection, namely  $X^0$ ,  $X'$ , and  $X''$ . For any head  $X^0$ , we require the level  $X'$  to take care of constituents larger than  $X^0$ . Both the compulsory (Complements) and optional (Adjuncts) phrases can be joined at  $X'$ , one at a time, by Adjunct Rule (refer Rule (2) below) are the Complement Rule (refer Rule (3) below), respectively. The highest X-bar which is not dominated by any other X projection is the maximal projection of  $X^0$  denoted by  $X''$  or XP. The Specifier is attached at the  $X''$  level by the Specifier Rule (refer Rule (1) below). Thus we need only three levels namely  $X^0$ ,  $X'$ , and  $X''$ . This three level hierarchical structure 'do so' substitution and 'one' substitution constructs which can be taken care by two level flat structure (Hageman [15], Radford [2000]). Thus the three level symmetrical X-bar analysis is more suitable both linguistically as well as computationally.

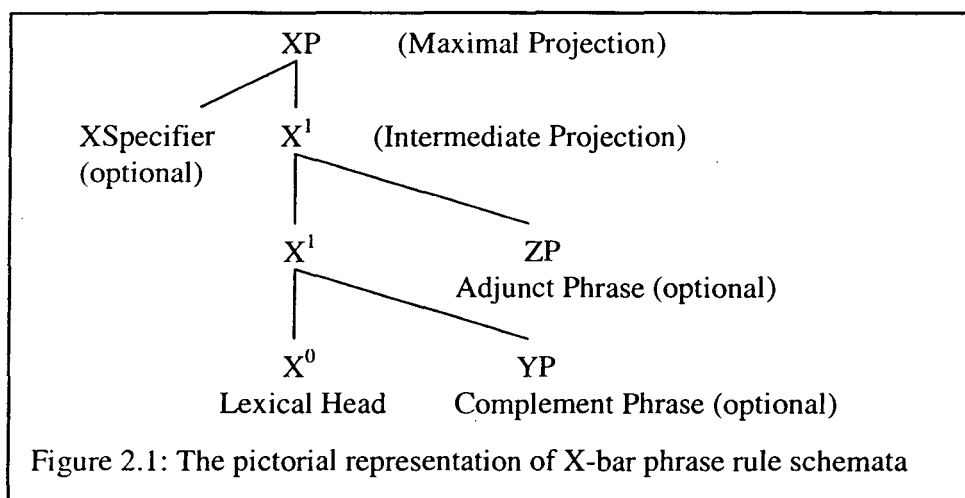
Accordingly, an X-bar phrase in a language is defined by the following rule schemata:

For any lexical category X such as Verb, Noun, Adjective, Preposition etc,  $X^0 = \text{Head}$

- |   |                              |
|---|------------------------------|
| $X'' (XP) \rightarrow (X\text{Specier}) ; X'$ | -- (1) (The Specifier Rule)  |
| $*X' \rightarrow (ZP) ; X'$                   | -- (2) (The Adjunct Rule)    |
| $X' \rightarrow (YP) ; X^0$                   | -- (3) (The Complement Rule) |

The terms included in the parenthesis indicates that these terms are optional, '\*' indicates that the Adjunct rule is optional, and a semicolon in the rules indicates that the terms on

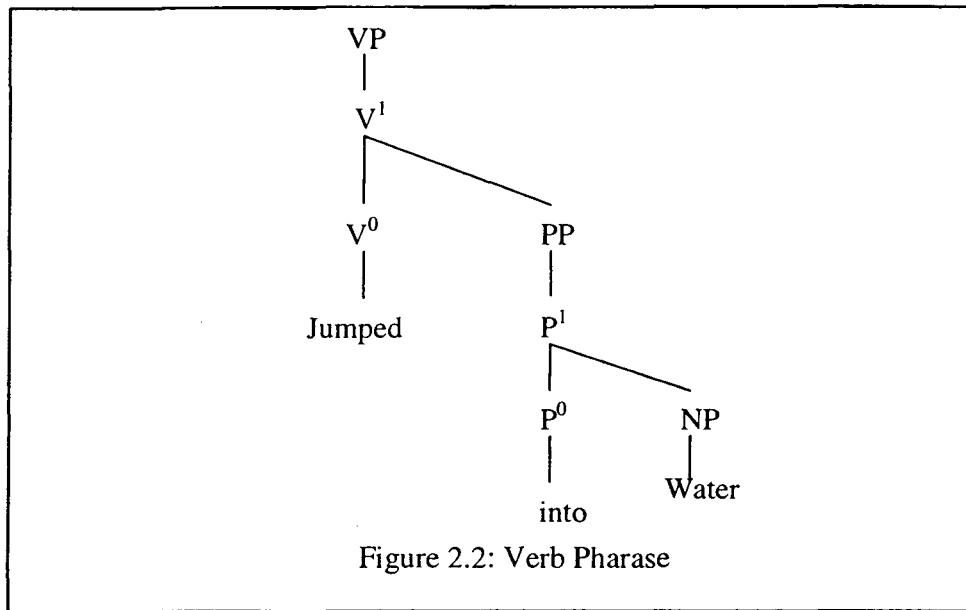
the right hand side are not taken as ordered. Each of ZP and YP are full phrases like XP. Maximal projections for categories Verb(V), Noun(N), Adjective(A), Preposition(P), etc. are denoted as VP, NP, AP, PP, etc. respectively. The tree representation defined by phrasal rules is given in Figure 2.1.



Next, we briefly describe the phrasal projections for each type lexical category.

### 2.2.1 The Phrasal Projection for a Verb

A Verb Phrase (VP) is a phrase having a verb as its head. Depending on a particular head complement(s) may be present or absent. The complement in a VP may be a NP, PP, AP, IP, or a CP. For each verb a Specifier NP, according to the latest theory, is present which then moves to the Specifier position IP as described later in the section. Adjunct(s) in a VP may be NPs, PPs, APs, or ADVP. A Verb Phrase typically functions as a Complement in an Inflection Phrase (IP). For example, the VP 'talked to sue' in IP 'she talked to sue'. The tree representation for VP 'talked to sue' is given in Figure 2.2.

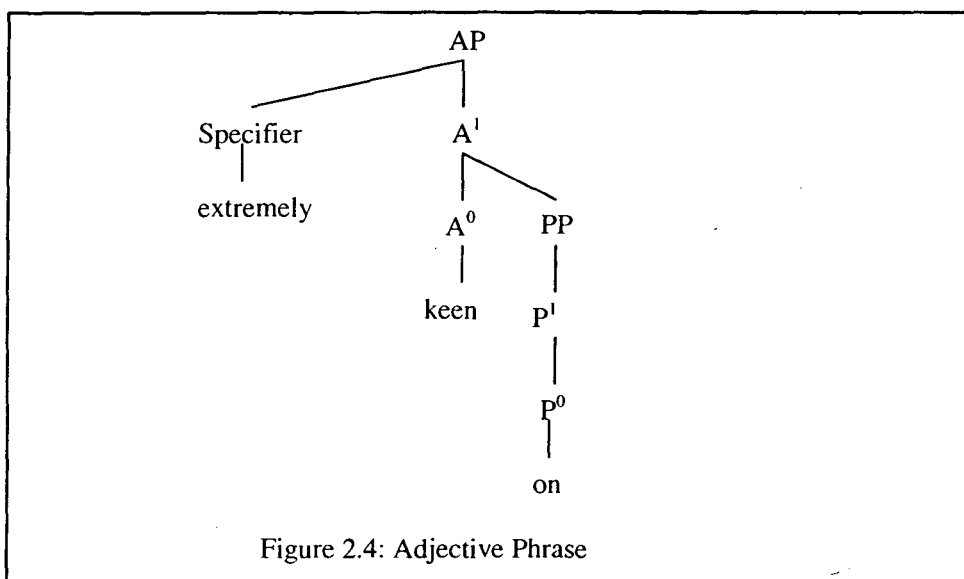
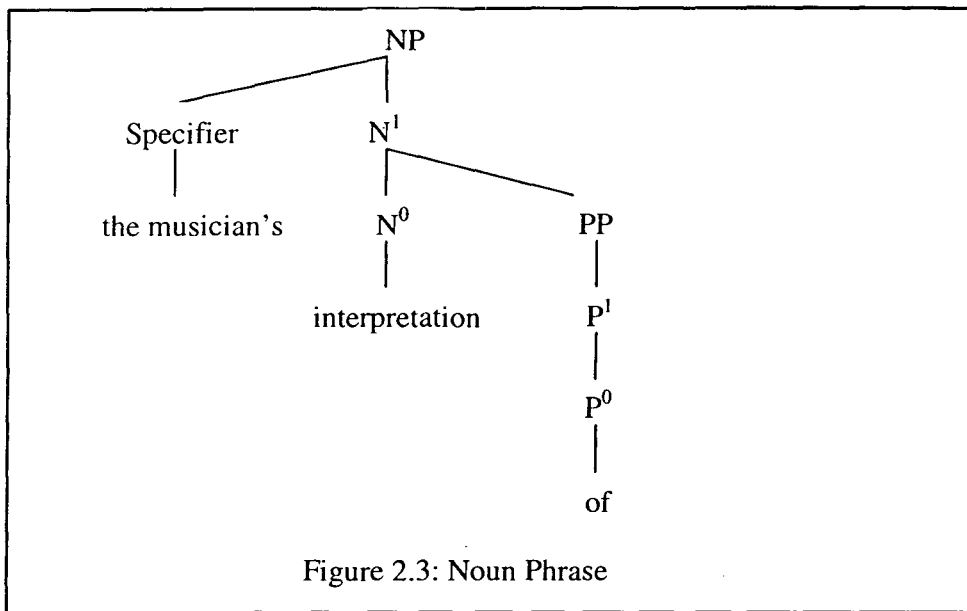


### 2.2.2 The Phrasal Projection for a Noun

A Noun Phrase (NP) is a phrase having Noun as lexical head. The complement of a Noun Phrase is always a PP which is always optional. Like in a VP adjuncts in a NP are optional. An adjunct of a NP may be a PP, an IP, or a CP. The Specifier of a NP may be a determiner, or a genitive NP. A Noun Phrase typically functions as a Specifier in an Inflection Phrase (IP). The tree representation for NP 'the musician's interpretation of that sonata' is given in Figure 2.3.

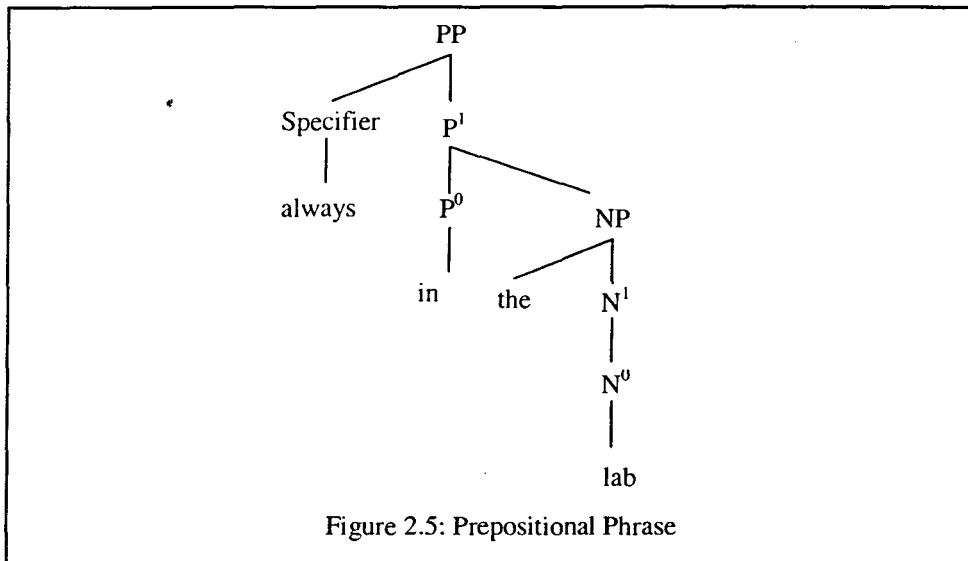
### 2.2.3 The Phrasal Projection for an Adjective

An Adjective Phrase (AP) is a phrase having Adjective as head. The complement of an AP is typically a PP. The Specifier of an AP is optional, but it can take adverb. The tree representation for AP 'extremely afraid of snakes' is given in Figure 2.4.



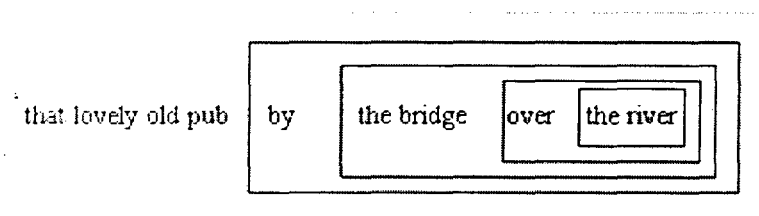
#### 2.2.4 The Phrasal Projection for a Preposition

A Prepositional Phrase (PP) is a phrase having preposition as lexical head. The complement of a Prepositional Phrase is always a NP. The specifier of a PP is always a ADVP which is always optional. The tree representation for PP 'always in the library' is shown in Figure 2.5.



Other than above five categories of phrases which are projected by lexical categories, the following two categories of phrases are projected by functional categories, Inflection (I) and Complementizer (C).

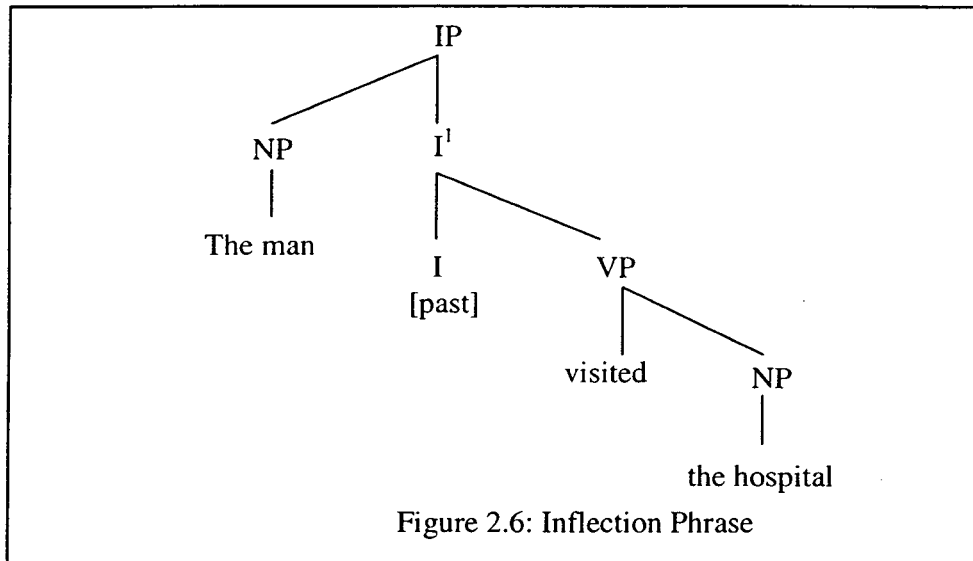
Longer phrases are like Russian dolls – they contain a number of shorter phrases:



The phrases whose heads are **by** and **over** are prepositional phrases

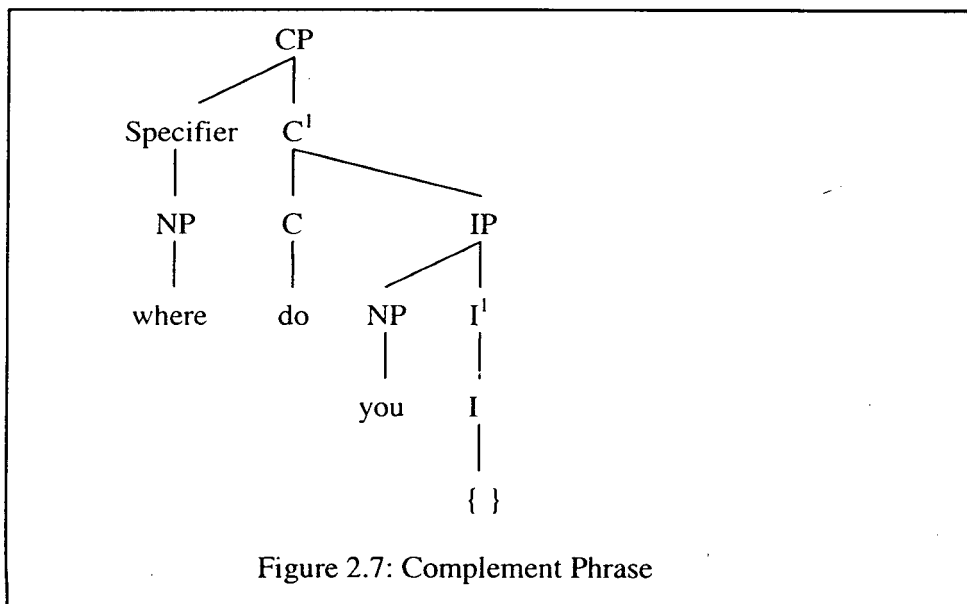
### 2.2.5 The Phrasal Projection for a Sentence (Inflection Phrase)

An Inflection Phrase (IP) is a phrase having the non-lexical element Inflection (I) as its head. The complement in an IP may be a VP or an AP. The specifier of an IP is always a NP. The tree representation for IP 'The bus driver angered the lady' is shown in Figure 2.6.



### 2.2.6 The Phrasal Projection for Complementizer

The Complementizer Phrase (CP) is a phrase having the non-lexical element Complementizer (C) as its head. The Complementizer Phrase corresponds either to a yes-no question (like, for example, 'Is he poor?', 'Will ram come?', etc.), or to a clause headed by a Complementizer ('that') (such as, for example, the CP 'that she will go there' in the sentence 'Indira said that she will go there', etc.). The tree representation for "where do u go" is given in the Figure 2.7.





## 2.3. LEXICON ORGANIZATION

We assume that every speaker is equipped with a **mental lexicon**, an "internal lexicon", which contains all the information they have internalized concerning the words of their language. It contains a lexical entry for each lexical item in the language. Lexical entries contain at least phonological, morphological, semantic and syntactic information. They contain all the information about lexical items that cannot be predicted by the rule system (e.g.: regular past tense forms of verbs are not given in the lexical entry since they can be predicted). We are mainly interested in the syntactic information, which contain the Categorical, Subcategorical Information and Thematic information. In the following sections we briefly describe how this syntactic information is represented in the Lexicon.

### 2.3.1 Categorical Information

For each entry it must be specified what **syntactic category** it belongs to, e.g.:

school: noun (N)

professor: N

student: N

play: verb (V)

read: V

into: preposition (P)

extremely: adverb (Adv)

fast: adjective (A)

The syntactic category determines the distribution of a word, which means that it tells us in what context, in what syntactic environment it can occur.

### 2.3.2 Subcategorization Information

A subcategorization frame specifies in what syntactic environment a category a lexical item can be inserted. It is called "subcategorization" because we can distinguish

subcategories (or subclasses) of categories on the basis of the context in which they appear, e.g.:

- (1) a. play: V, [--]           The boy is playing.  
b. draw: V, [-- NP]           The boy is drawing [a picture].  
c. give: V, [-- NP, PP]       The boy gives [**the ball**] [**to the man**].  
d. give: V, [-- NP, NP]       The boy gives [**the man**] [**the ball**].

These subcategorization frames indicate the position of the verb. Imitate, for example, is followed by one NP (such as the cat) - it selects or subcategorizes for one nominal object. Notice that the subcategorization frame includes only the OBJECTS (or COMPLEMENTS) of the lexical entry, but not the subject.

### 2.3.3 Thematic Information

As described in the section above, each lexical entry is specified for the number and type of arguments it requires. The **arguments** are the participants that are minimally involved in the activity or state expressed by the predicate, for instance: The verb imitate in (1)b requires two participants: one person who does the imitating (the **agent**), and someone who is being imitated (the **patient**). The verb gives in (1)c and d requires three participants: someone who does the giving (the **agent**), something that is given (the **theme**) and someone who receives the given entity (the **recipient**). These participants are called **arguments**, their roles (such as agent, theme, recipient) are called **thematic roles** (or **theta-roles**,  **$\theta$ -roles**). This kind of specification is called **thematic grid**. The  $\theta$ -roles are often represented by Arabic numerals, where the numeral 1 refers to the argument realized as the subject:

- (2) a. play: V; [1]           The boy is playing.  
b. draw: V; [1 2]        The boy is drawing a picture.  
c. give: V; [1 2 3]       The boy gives the ball to the man / the man the ball.

We can now combine the theta grid and the Subcategorization frame, i.e. we add the syntactic categories of the arguments:

- (3) a. play: V; [1]  
 b. draw: V; [1 2]  
 NP  
 c. give: V; [1 2 3]  
 NP NP  
 NP PP

Recall that the subcategorization frame includes only the objects of a lexical item. The thematic role of the subject can therefore not be linked to the subcategorization frame in the same way as the other roles can. The information presented above, is summarized in a tabular form as given in Table 2.1.

Table 2.1: Lexical Entries with their Subcategorization and Argument structures.

	Categorical Information	Subcategorization Information	Thematic Information
speak	v	[--PP] (speak to someone)	[1 2]
		[--] (speak)	[1]
receive	v	[--NP] (receive a letter)	[1 2]
		[--NP PP] (receive a letter from some one)	[1 2 3]
		[--NP NP] (receive someone's letter)	[1 2 3]

In the Machine Translation process the word lexicon is used more often than the word dictionary as this information is stored in a machine readable form. So the lexicon can be called as a “*computational dictionary*”. In natural languages one word can have several

forms and it is not wise enough to store all those forms in the lexicon as it simply increase the size of the lexicon. In MT, the lexicon usually contains the root words of the language and the morphological processes use the information present in the lexicon to generate the other forms of the words. Usually this information is stored in the files or as tables in the databases which can be easily retrieved for the computational purpose. However, the way you organize your data is more important as it would directly affect the speed of the computation.

## 2.4 THETA THEORY

The lexicon specifies the number and type of arguments that a verb takes. **Arguments** are the essential participants in the event that the verb refers to. The technical term for this aspect of the verb's meaning is **argument structure**. The lexical item that specifies the argument structure is called the **predicate**. We have already mentioned some of the **theta-roles** associated with arguments, such as **agent**, **patient**, or **theme**. There are other roles such as **experiencer** (someone who is experiencing a physical or mental sensation, refer (4)a below), **benefactive/ recipient** (someone who benefits from the action expressed by the verb or who receives something as in (4)b below), and a few more.

- (4) a. Peter feels cold.  
b. Peter gives Mary the book.

We say that the predicate **assigns** theta-roles to its arguments or that the predicate **selects** its arguments. Theta-roles assigned to complements are referred to as **internal theta-roles**, complements are thus **internal arguments**. Theta-roles assigned to subjects are **external theta-roles**, subjects are thus **external arguments**. Theta roles are assigned by the Governor to its Governees under Government. The definition of Government is given below:

### **Government:**

A node A **governs** a node B iff

- 1) A is a governor.
- 2) A m-commands B and
- 3) No barrier (some YP) intervenes between A and B
  - ❖ Maximal projections are barriers to government.
  - ❖ Governors are heads.

There are two types of *Governors*:

**1) Lexical governors**

- E.g. V (verb), P (preposition), N (noun), A (adjective).

**2) Functional governors**

- E.g. I (agreement), C (complementizer).

### **2.4.1 The Argument Structure of Other Syntactic Categories**

So far, we analyzed verbs as predicates. However, other categories like nouns, prepositions, adjectives etc, have argument structures, as well. In the following section we are going to analyze about those argument structures in brief.

**2.4.1.1 Nouns:** Argument structure is most obvious in nouns that are morphologically related to verbs.

Consider the examples:

The Romans destroyed the city.

The Romans' destruction of the city.

As we can see, the argument structures of the nouns in the above sentences are remarkably similar to that of their corresponding verbs. They can have subjects and objects just like verbs.

We have already said that the syntactic arguments of nouns are usually optional (i.e. can be left implicit). The subject of a noun is always optional, whereas the subject of a verb must always be realized.

For example compare:

“The Romans destroyed the city.” with “Destroyed the city.” and  
“The Romans’ destruction of the city.” with “ the destruction of the city”

Similarly, objects of nouns may also be omitted, even if cannot be left implicit in the case of the corresponding verb.

For example compare:

“Poirot will analyze the data.” with “Poirot will analyze.” and  
“Poirot's analysis of the data.” with “Poirot's analysis”

**2.4.1.2 Adjectives:** Adjectives too can have arguments. Similar to nouns, their arguments can often be left implicit and arguments are syntactically realized as PP's (often with of) or clauses, as shown by the examples below:

Poirot envies Bertie.

Poirot is envious of Bertie.

**2.4.1.3 Prepositions:** Some linguists argue that prepositions can also function as predicates and take arguments. Some prepositions can occur with or without arguments, as shown in the examples below:

Peter is **in** London.

He is **outside** (the house). Here argument “the house” is optional.

## 2.4.2 The Theta Criterion

We have seen that the number of arguments that can show up in a sentence is determined by the number of  $\theta$ -roles that a predicate has. The requirements illustrated in these examples are captured by the **Theta-Criterion**: which states that:

→ *Each argument is assigned one and only one theta-role.*

→ Each theta-role is assigned to one and only one argument.

### 2.4.3 The Projection Principle

We have said so far that the mental lexicon contains a lexical entry for each lexical item in the language. This lexical entry in turn contains phonological, morphological, semantic and syntactic information about that lexical item. The syntactic information contains categorial information, subcategorization information, and thematic information. We have also seen that the information given in the lexical entry must be represented in the sentence that the relevant lexical item is a part of. (For instance, theta-roles must show up in the sentence, they must be assigned to arguments. If arguments may be left implicit, this must be specified in the lexical entry.) We have also seen that the syntactic category of a lexical item determines the syntactic category of the corresponding phrase. This is captured by the **Projection Principle**, which states that: "*Lexical information is syntactically represented.*"

TH-138/0



### 2.4.4 The Extended Projection Principle

The Extended Projection Principle (EPP) requires that sentences must have subjects. For example, in the sentences "It rains." or "It has been snowing.", although the subject 'it' does not contribute to the meaning of the sentence, and it is not an argument of the verb, it cannot be omitted. This follows from the EPP.

So far we have presented in brief the requirements, according to the theory, the analysis and design of Lexicon and how the lexicon is to be organized, so that we can capture all the required syntactic information about lexical items of words in a given sentence. Detailed information about all these concepts can be found in Hageman [15] and Fromkin [29]. In the following section we are going to concentrate on some other important concepts of GB Theory, in brief.

## 2.5 OTHER CONCEPTS

In the following section we are going to give the other important concepts in GB Theory, i.e. Case Theory and Binding Theory in brief.

### 2.5.1 Case Theory

According to Case Theory, every overt NP must be case-marked. Also, even though only pronouns show overt morphological case in English, it is assumed that all NPs have Case (called abstract case) that matches the morphological case that shows up on pronouns.

The degree of its morphological realisation varies parametrically from one language to another. Case Theory accounts for the distribution and form of overt NP's. It defines contexts in which NP's are assigned abstract case.

In English, overt morphological realization of case in full lexical noun phrases is restricted to the GENITIVE case. NOMINATIVE and ACCUSATIVE case can only be seen on pronouns. Heads of some categories (specifically V, P, I, but not A and N) have case-assigning abilities. In English, Verbs assign ACCUSATIVE CASE to their complements, whereas nouns and adjectives don't. Further, the passive participles cannot assign ACCUSATIVE case to their complements and that therefore the internal argument moves to the subject position to receive NOMINATIVE case. This in turn is possible because the passive participle does not assign an external  $\theta$ -role, but this role is absorbed by the passive morpheme. These two properties (failure of the verb to assign ACC case and absorption of the external argument of the verb) have been related by Burzio (1986) by a descriptive generalization which is known as Burzio's Generalization. Similarly **raising verbs** do not assign an external  $\theta$ -role to their subject position.



# Chapter 3

## GOVERNMENT AND BINDING BASED GRAMMAR FOR SANSKRIT

### 3.1 INTRODUCTION

Sanskrit is the language of Devas or gods, and the alphabets in which it is written is called Devanagari, or that employed in the cities of gods. The correct name for the Sanskrit alphabet is Daivanagari sometimes abbreviated into Nagari. Perhaps in the word Devanagari we have a history of the times when the Aryans entered and settled in the Northern India. The Aryans who were much fairer in colour than the aborigines of India are the Devas referred to in the name Devanagari and Nagari means the Aryan settlements within the precincts of which the sacred language was spoken.

In Sanskrit words that are used in a sentence are called Padas. A Pada is a fully inflected word, i.e. the root word combined with a bound morpheme. Padas are mainly of two types, Tingant and Subant. A Tingant is a verbal form obtained by combining a verb root and a Ting (a bound morpheme), which gives tense, aspect, person and number features of the resulting Tingant (the verbal form of the word). A Subant is a nominal form obtained by combining a nominal root (referred to as Pratipadik) and a Sup (a bound morpheme), which gives number, person and case features of the resulting Subant (the nominal form of the word). The gender feature is associated with the Pratipadik. Sanskrit has three genders (masculine, feminine and neuter), three numbers (singular, plural and dual), three persons (first, second and third), eight cases, and ten tenses and moods. The root takes different form based on the suffix added to it. It is for reasons like this, Sanskrit packs a lot of information into a Pada, a word in a sentence.

In our domain of source as well as the Target Language, we consider eight cases of a Noun, these are: Nominative, Accusative, Instrumental, Dative, Ablative, Possessive, Locative and Vocative. In English, the cases are determined by the position of a

Nominal Phrase in a sentence or by the presence of the preposition used with the nouns. However in Sanskrit language a noun, as stated above, is modified with a specific suffix for each case.

### 3.2 FORMS OF DIFFERENT GRAMMATICAL CATEGORIES

#### 3.2.1 Noun Forms in Sanskrit (Declensions of Nouns)

A nominal morpheme, referred to as Sup in Sanskrit, has 24 forms depending on the case and the number of the desired final form of the nominal Pada. Each of these 24 forms of Sup is further modified depending on the gender and the ending of the Pratipadik with which it has to be attached. Though most of the generation is regular in nature, in certain cases irregular processes also operate [Asthadyayi].

Feminine noun endings

Feminine nouns may end in आ, इ, ई, उ, ऊ, or ऋ.

Neuter noun endings

Neuter nouns may end in अ, इ, उ, or ऋ.

Masculine noun endings

Masculine nouns may end in अ, इ, उ, or ऋ.

Therefore only some typical case(s) or example(s) are presented here. Table 3.1 gives the complete table of the Pratipadik राम. Table 3.2 gives the complete table of the Pratipadik रमा. Similarly Table 3.2 gives the Pratipadik ज्ञान. The root राम is अ ending masculine, the root रमा is आ ending feminine, and the root ज्ञान is अ ending neuter.

Case	Singular	Dual	Plural
Nominative	रामः	रामौ	रामाः
Vocative	राम	रामौ	रामाः
Accusative	रामम्	रामौ	रामान्
Instrumental	रामेण	रामाभ्याम्	रामैः
Dative	रामाय	रामाभ्याम्	रामेभ्यः
Ablative	रामात्	रामाभ्याम्	रामेभ्यः
Genitive	रामस्य	रामयोः	रामाणाम्
Locative	रामे	रामयोः	रामेषु

Table 3.1: Pratipadik राम

Case	Singular	Dual	Plural
Nominative	रमा	रमे	रमाः
Vocative	रमे	रमे	रमाः
Accusative	रमाम्	रमे	रमाः
Instrumental	रमया	रमाभ्याम्	रमाभिः
Dative	रमायै	रमाभ्याम्	रमाभ्यः
Ablative	रमायाः	रमाभ्याम्	रमाभ्यः
Genitive	रमायाः	रमयोः	रमाणाम्
Locative	रमायाम्	रमयोः	रमासु

Table 3.2: Pratipadik रमा

Case	Singular	Dual	Plural
Nominative	ज्ञानम्	ज्ञाने	ज्ञानानि
Vocative	ज्ञान	ज्ञाने	ज्ञानानि
Accusative	ज्ञानम्	ज्ञाने	ज्ञानानि
Instrumental	ज्ञानेण	ज्ञानाभ्याम्	ज्ञानैः
Dative	ज्ञानाय	ज्ञानाभ्याम्	ज्ञानेभ्यः
Ablative	ज्ञानात्	ज्ञानाभ्याम्	ज्ञानेभ्यः
Genitive	ज्ञानस्य	ज्ञानयोः	ज्ञानाणाम्
Locative	ज्ञाने	ज्ञानयोः	ज्ञानेषु

Table 3.3: Pratipadik ज्ञान

### 3.2.2 Adjective Form

An adjective is governed by the nominal it modifies. That is it is modified by adding the same Sup ending which is used to modify the governing nominal root. The following examples show how the adjective is governed by the governing Noun it modifies.

एतस्मात् मधुरात् फलात् अहं रसं प्राप्नोमि ।

मधुरां कथां श्रुत्वा, निद्रां करोति बालकः ।

मधुरस्य कृष्णस्य वाणी मधुरा ।

In the above sloka, **कृष्णस्य** is a genitive form of the masculine noun **कृष्ण**. Therefore the adjective modifying the noun is also declined in the same way. Since **फलात्** is an ablative form of the neuter noun **फल**, the adjective is also declined as **मधुरात्**.

### 3.2.3 Verb Forms (Verb Conjugation)

In Sanskrit, there are two kinds of verbs: Primitive and Derivative. Primitive verbs or roots are those which originally exist in the language, while derivative verbs are those which may be derived from the parent stock- a root by adding prefixes. Verbs are associated with ten different forms of usage referred to as Lakaras. Of these six relate to the tenses and four relate to moods. A brief description of Sanskrit Lakaras is presented below. The six Lakaras representing tenses are:

1. लट् Present tense
2. लङ् Past tense - imperfect
3. लुङ् Past tense - aorist
4. लिट् Past tense - perfect
5. लृट् Future tense - likely
6. लृट् Future tense – certain

Four Lakaras representing moods are:

- 1 लृङ् Conditional mood
- 2 विधिलिङ् Potential mood
- 3 आशीर्लिङ् Benedictive mood
- 4 लोट् Imperative mood

A root is conjugated according to a Lakara and also conjugated according to Padas (Atmanepada, Ubhayapada and Parasmaipada). Other than Lakara and Pada, person and number determine the choice of Ting that needs to be affixed to the root Verb.

A Pada of a verb form determines whether the activity specified in the verb applies to the person himself or whether it applies to someone other than the subject of the verb. Verbs referring to the activity for the self are said to be “**Atmanepada**” आत्मनेपद

verbs. Verbs referring to the activity for others are said to be “Parasmaipada” परस्मैपद verbs. Verbs which can take both forms are known as “Ubhayapada” उभयपद verbs.

**3.2.3.1 Simple present tense:** There is only one form for the present tense (लट्). In simple present tense, we add suffixes to the root form of the verb before the terminations are added. The suffix आ is added in first person, while suffix अ is added in second and third persons. The terminations for the verbs in “Parasmaipada” are stated in Table 3.4:

Person/Number	Singular	Dual	Plural
First	मि	वः	मः
Second	सि	थः	थ
Third	ति	तः	अन्ति

Table 3.4

The different forms for the verb पठ् in present tense are given in Table 3.5:

Person/Number	Singular	Dual	Plural
First	पठामि	पठावः	पठामः
Second	पठसि	पठथः	पठथ
Third	पठति	पठतः	पठन्ति

Table 3.5

The terminations for the verbs in “Atmanepada” are stated in Table 3.6:

Person/Number	Singular	Dual	Plural
First	इ	वहे	महे
Second	से	इथे	ध्वे
Third	ते	इते	अन्ते

Table 3.6

### 3.2.3.2 Simple Past Tense: Past tense has three forms associated with it

1. Expressing something that had happened sometime in the recent past, typically last few days.
2. Expressing something that might have just happened, typically in the earlier part of the day.
3. Expressing something that had happened in the distant past about which we may not have much or any knowledge.

For the simple past tense, अ् is the prefix. The terminations for the verbs in “Parasmaipada” are stated below in Table 3.7:

Person/Number	Singular	Dual	Plural
First	अं	व	म
Second	सू	तं	त
Third	त्	तां	अन्

Table 3.7

The forms for the past tense of the verb can be obtained from the above table. The root form of the verb is गच्छ् and the infix corresponding to the root is अ for the second and third person but आ for the first person. The different forms for the verb गच्छ् in past tense are given in Table 3.8:

Person/Number	Singular	Dual	Plural
First	अगच्छं	अगच्छाव	अगच्छाम
Second	अगच्छः	अगच्छतं	अगच्छत
Third	अगच्छत्	अगच्छतां	अगच्छन्

Table 3.8

The terminations for the verbs in “Atmanepada” are stated in Table 3.9:

Person/Number	Singular	Dual	Plural
First	इ	वहि	महि
Second	थाः	इथां	ध्वम्
Third	त	इतां	अन्त

Table 3.9



### 3.2.3.3 Simple Future tense: Future tense has two forms associated with it.

1. Expressing something that is certainly going to happen (लृट्).
2. Expressing something that is likely to happen (लृट्).

The infix for the future tense is स्य. The infix changes its form to इष्य when applied to some rules. In some cases it may also become ष्य. However there is no direct rule which we can be stated in respect to infix. We can see two forms for many verbs. For example,

गम्, गच्छ् are the two root forms for गच्छति

पा, पिब् are the two root forms for पिबति

The form of the verb for future tense will be based on the first root where as the second form of the root will be used in generating the verb in present tense and past tense.

The terminations for the future tense in “Parasmaipada” are given in Table 3.10:

Person/Number	Singular	Dual	Plural
First	ष्यामि	ष्यावः	ष्यामः
Second	ष्यसि	ष्यथः	ष्यथ
Third	ष्यति	ष्यतः	ष्यन्ति

Table 3.10

The forms for the verb गच्छति in the future tense are given in Table 3.11:

Person/Number	Singular	Dual	Plural
First	गमिष्यामि	गमिष्यावः	गमिष्यामः
Second	गमिष्यसि	गमिष्यथः	गमिष्यथ
Third	गमिष्यति	गमिष्यतः	गमिष्यन्ति

Table 3.11

The terminations for the future tense in “Atmanepada” are given in Table 3.12:

Person/Number	Singular	Dual	Plural
First	ष्ये	ष्यावहे	ष्यामहे
Second	ष्यसे	ष्यथे	ष्यध्वे
Third	ष्यते	ष्येते	ष्यन्ते

Table 3.12

In our work we have considered only Present tense, past tense – Aorist, and Future tense – certain. Parasmaipada conjugation of verb “kri” is given in the Table 3.13.

**कृ = करना (परस्मैपदी) kri = to do**

	एकवचन (one person)	द्विवचन (two people)	बहुवचन (more than two)	
लट्	करोति	कुरुतः	कुर्वन्ति	प्र. (first person)
(Present)	(karoti)	(kurutah)	(kurvanti)	
	करोषि	कुरुथः	कुरुथ	म. (second person)
	करामि	कुर्वः	कुर्मः	उ. (third person)
लिट्	चकार	चक्रतः	चक्रुः	प्र.
(Past Perfect)	चकर्थ	चक्रथुः	चक्र	म.
	चकार, चकर	चकृथ	चकृम	उ.

लृट्	कर्ता	कर्तारो	कर्तारः	प्र.
(First future)	कर्तासि	कर्तास्थः	कर्तास्थ	म.
	कर्तास्मि	कर्तास्वः	कर्तास्मः	उ.
लृट्	करिष्यति	करिष्यतः	करिष्यन्ति	प्र.
(Simple future)	करिष्यसि	करिष्यथः	करिष्यथ	म.
	करिष्यामि	करिष्यावः	करिष्यामः	उ.
लोट्	करोतु, कुरुतात्	कुरुताम्	कुर्वन्तु	प्र.
(Imperative mood)	कुरु, कुरुतात्	कुरुताम्	कुरुत	म.
	करवाणि	करवाव	करवाम	उ.
लङ्	अकरोत्	अकुरुताम्	अकुर्वन्	प्र.
(Past imperfect)	अकरोः	अकुरुताम्	अकुरुत	म.
	अकरवम्	अकुर्व	अकुर्म	उ.
वि. लिं.	कुर्यात्	कुर्याताम्	कुर्युः	प्र.
(Potential mood)	कुर्याः	कुर्याताम्	कुर्यात	म.
	कुर्याम्	कुर्याव	कुर्याम	उ.
आ. लिं.	क्रियात्	क्रियास्ताम्	क्रियासुः	प्र.
(Benedictive)	क्रियाः	क्रियास्तम्	क्रियास्त	म.
	क्रियासम्	क्रियास्व	क्रियास्म	उ.
लृङ्	अकार्षात्	अकार्षाताम्	अकार्षुः	प्र.
(Aorist)	अकार्षाः	अकार्षाताम्	अकार्ष	म.
	अकार्षम्	अकार्षव	अलाष्म	उ.
लृङ्	अकरिष्यत्	अकरिष्यताम्	अकरिष्यन्	प्र.
(Conditional)	अकरिष्यः	अकरिष्यताम्	अकरिष्यन्त	म.
	अकरिष्यम्	अकरिष्याव	अकरिष्याम	उ.

Table 3.13: Conjugation of Verb “kri”

In Sanskrit the root word forms which do not under go declensions or conjugations are referred to as Avyayas or unchanging. The types of words falling under the category of Avyayas are adverbs, conjugations and interjections etc.

### 3.3 SANSKRIT PHRASE STRUCTURES

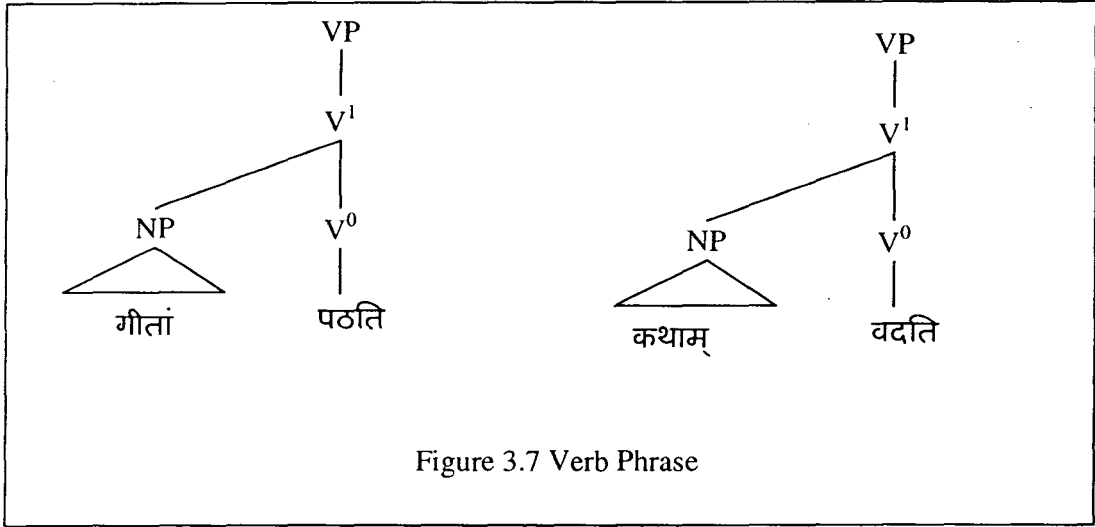
Sanskrit is a highly inflectional language and due to this property it is a completely word- order free language. To fit Sanskrit in the GB frame work we have superimposed a Hindi like word order on Sanskrit. So we assume that Sanskrit is a SOV language. As described in the above section, SOV languages are **head-final** and **specifier-initial languages**. So the phrase structures for **our Sanskrit** will be similar to that of Hindi, given in the above section and are repeated below for easy reference. The basic phrase structure rules are:

$$XP \rightarrow \text{Specifier } X'$$
$$X' \rightarrow \text{Complements } X'$$
$$X' \rightarrow \text{Adjuncts } X^0$$

To convert a given Sanskrit Phrase/Sentence into a SOV structure, a preprocessor which converts word-order free Sanskrit phrases into their SOV form needs to be developed. In our work, however we are manually converting Sanskrit sentences into SOV structures before feeding them to the parser. Since we are considering only simple sentences, and in that only Verb Phrases (VP), Noun Phrases (NP), and Inflection Phrases (IP), we are going to present in the following, these Phrase Structures only. The rules forms of Sanskrit Phrase Structures are given in Appendix A.

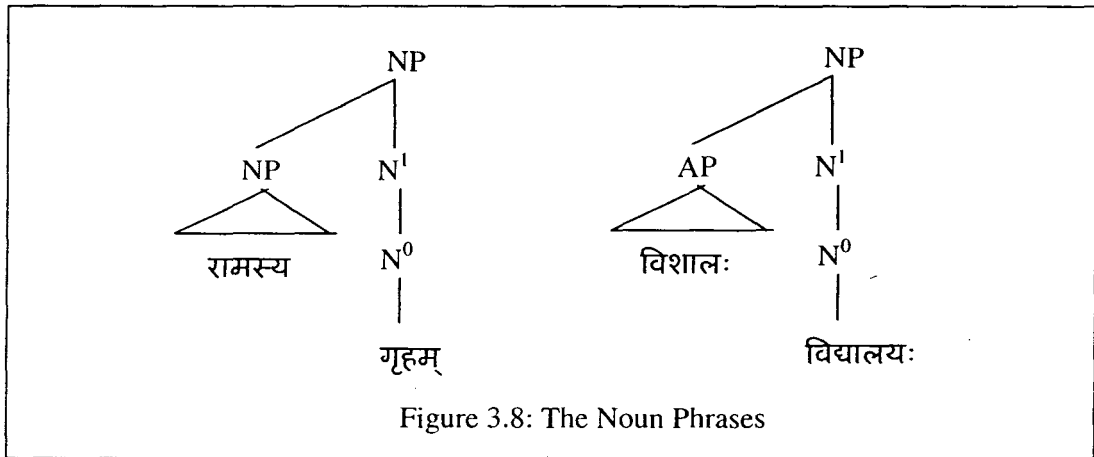
#### 3.3.1 Verb Phrase

In the Verb Phrase given in the Figure 3.7, the complement is an NP. However, in general a Verb Phrase complement could also be an AP, an IP or a CP.

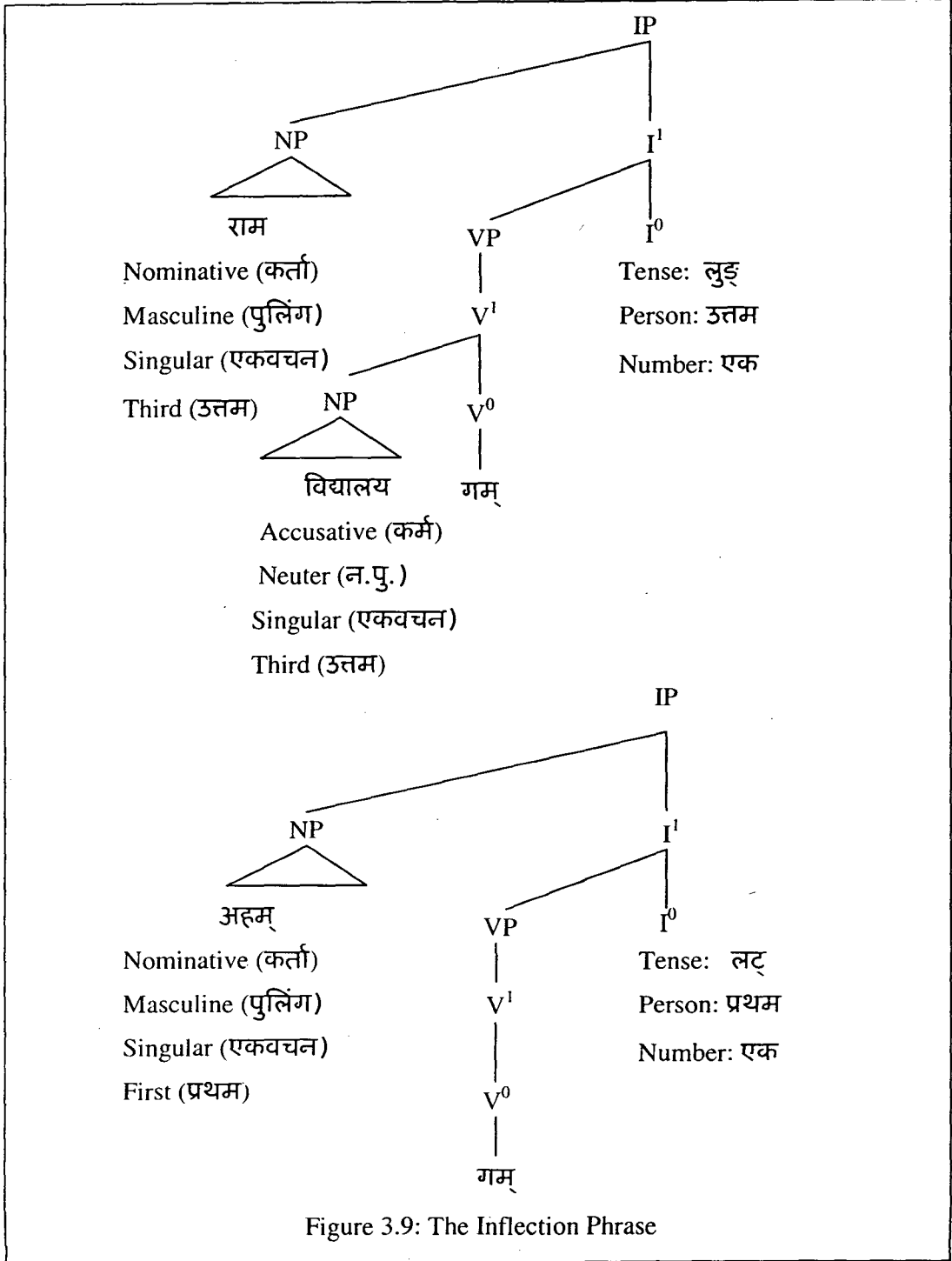


### 3.3.2 Noun Phrase

Two examples of Noun Phrase are given in Figure 3.8. In the first example the specifier is a Genitive NP and in the second example it is an AP.



**3.3.3 Inflection Phrase :** The tree representation for the Sentence “रामः विद्यालयम् आगमम्” and “अहम् गच्छामि” are shown in Figure 3.9a and Figure 3.9b respectively. In the example the specifier is an NP. However it could be an IP or a CP.



### 3.3.4 Complementizer Phrase

The tree representation for a Complementizer Phrase is shown in Figure 3.10.

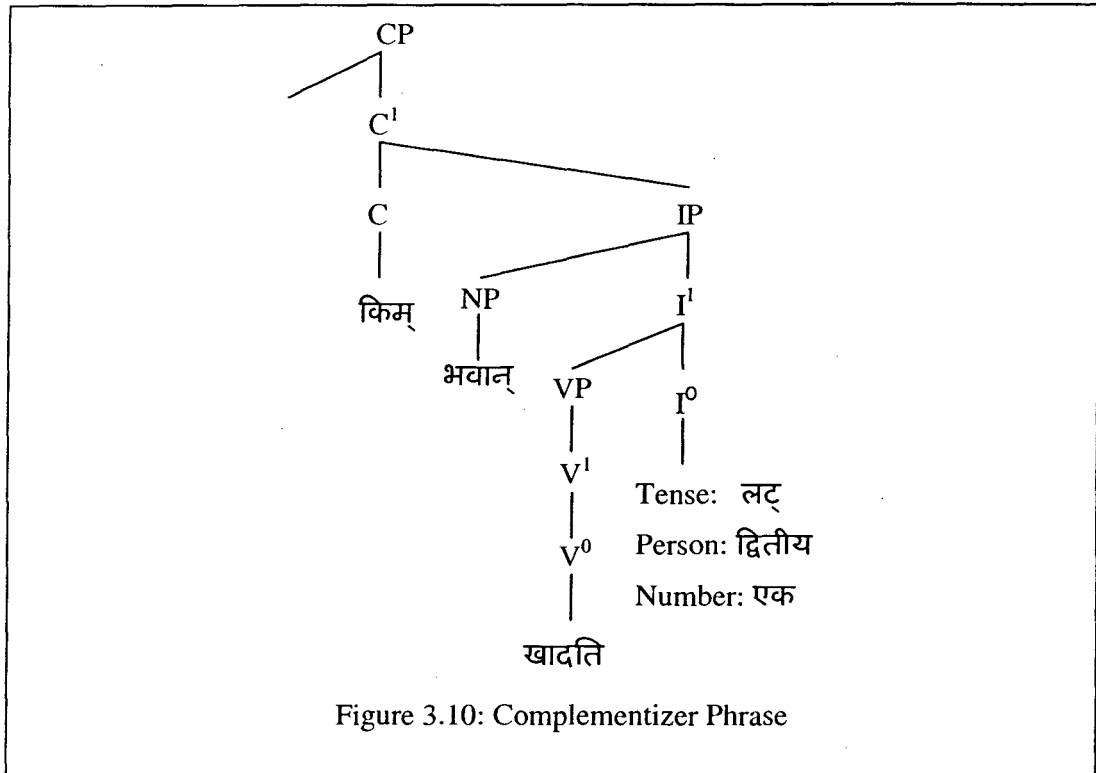


Figure 3.10: Complementizer Phrase

# Chapter 4

## Lexicon

### 4.1 INTRODUCTION

A lexicon is a repository of words and knowledge about those words. This knowledge may include details of the grammatical structure of each word (morphology), the sound structure (phonology), its part of speech, and the meaning of the word in different textual contexts, e.g. depending on the word or punctuation mark before or after it. Lexicons may be ordered either alphabetically or semantically. A useful lexicon may have hundreds of thousands of entries. Lexicons are needed for every language of application. There are a number of special cases which are usually researched and produced separately from general purpose lexicons: dictionaries of proper names, terminology databases, and wordnets.

Lexicon is the important module in our work. Lexicon will have all the information of every word of language. This information is like tense, aspect, number, gender and person etc. These attributes will change from one category to another category of words as per the language grammar. Lexicon also contains the sub-category information and thematic information of every word. As this information plays major role in parsing the sentence, this information has to be stored in proper way in database such that we can eliminate redundancy and retrieve data in optimum way.

### 4.2 SANSKRIT LEXICON

#### 4.2.1 Introduction

Sanskrit Lexicon contains information of every word as per the grammatical category of that word. In Sanskrit, words are mainly classified as Noun, Pronoun, Verb, and Adjective as per the grammar. So, we developed different table for each category. Along with these tables subcategory information table also exist, which provide the



sub-category information of words. Sanskrit have mainly two categories i.e. Verbal and Nominal.

#### 4.2.1.1 Noun Information

Noun comes under Nominal category.

Mainly nouns have following attributes

- Noun Case
- Number
- Gender

Word	Noun Case	Number	Gender
छात्रः	Nominative	Singular	Masculine
लेखं	Accusative	Singular	Masculine
नदी	Nominative	Singular	Feminine

Table 4.1 Noun Information

#### 4.2.1.2 Pronoun Information

Pronouns also come under Nominal category. Pronoun has following attributes

- Noun Case
- Number
- Gender
- Person

Word	Noun Case	Number	Gender	Person
अहम्	Nominative	Singular	Both	First
त्वम्	Nominative	Singular	Both	Second
सः	Nominative	Singular	Male	Third

Table 4.2 Pronoun Information

### 4.2.1.3 Adjective Information

Adjectives are also come under Nominal category. In Sanskrit adjectives and nouns will have the same attributes. These are

- Noun Case
- Number
- Gender

Word	Adjective Case	Number	Gender
सुन्दरः	Nominative	Singular	Masculine
सुन्दरी	Nominative	Singular	Feminine
सुन्दरम्	Nominative	Singular	Neutral

Table 4.3 Adjective Information

### 4.2.1.4 Verb Information

Verbs come under Verbal category.

The important attributes of the verbs are

- Number
- Person
- Tense
- Aspect

Word	Number	Person	Tense	Aspect
पठति	Singular	Third	Present	Simple
पठयते	Singular	Third	Present	Simple
अगच्छन्	Plural	Third	Past	Simple

Table 4.4 Verb Information

### 4.3 DESIGN OF LEXICON

The previous section explains, what the different important categories of words are, and their important attributes which plays major role in parsing the sentence. Current section discusses with the organization of lexicon for Sanskrit.

After the through study of the Sanskrit vocabulary we came to know that most of the words of similar category have the same attributes values. So, keeping it in mind in order to eliminate the redundancy, table for every grammar category contains only attributes, but not the word.

The words are stored in a separate table, where it can have index to its attributes in a particular category table. As sub-categorical information also similar for many words, it's also store in other table and words will have index to its sub-categorical information.

Word table looks like this

Word	Category	Category Index	Sub-category Index
छात्रः	Noun	1	1
पठति	Verb	1	2

Table 4.5 Model of word Information Table

Sub-category table looks like

Sub-category Index	No of Complements	1 <sup>st</sup> Complement	2 <sup>nd</sup> Complement
0	1	NounP(Obj)	
1	2	NounP(Obj)	AdjectiveP

Table 4.6 Model of Sub-category Information Table

### 4.3.1 Noun Table

As Noun and Adjective have same attribute only one table is maintained for that, in which one column introduced to differentiate whether it's a noun or adjective. The header of the table is shown in table 4.7.

Noun Index	Noun Case	Number	Gender	Type
------------	-----------	--------	--------	------

Table 4.7 Noun Table

In this Noun case tells case of noun, Number and Gender indicates number and gender of word respectively. Type indicates word category, i.e. whether Noun or Adjective. Here Noun Index is referred by Category Index in Word table and Noun Index of Pronoun table.

### 4.3.2 Pronoun Table

As Pronoun also have same attributes as noun category, with an extra attribute person. Only noun index is stored in pronoun table which will eliminate redundancy of data and person information is stored in this. The header of the Pronoun table is shown table 4.8

Pronoun Index	Noun Index	Person
---------------	------------	--------

Table 4.8 Pronoun Table

Here Pronoun Index referred by the Category Index of word table. Noun Index refers to the Noun Index of Noun table.

### 4.3.3 Verb Table

Verb table contains all information to represent a verb. Head of Verb table shown in Table 4.9.

Verb Index	Number	Person	Tense	Aspect
------------	--------	--------	-------	--------

Table 4.9 Verb Table

### 4.3.4 Interface

In order to store the data into lexicon, an interface has been designed which allows to enter data very easily into database. A screen shot of that interface is shown in figure 4.1.

The screenshot shows a window titled "Sanskrit Data Entry" with the following fields and controls:

- Word:
- RootWord:
- Category:
- Number:
- Gender:
- Person:
- Tense:
- Aspect:
- NounCase:
- Mood:
- Voice:
- SubCategory:
- NoOfComplements:
- Comp1:
- Comp2:
- Comp3:
- Save:

Fig 4.1 Screenshot of Sanskrit Lexicon Data Entry

Lexicon data entry interface enables user to enter the data into lexicon very easily. The interface is prepared very user friendly, so that a novice can enter data. When a particular category selected from category combo box, the combo boxes which corresponds to that category are only enabled and other are disabled. So, that user can't enter the data wrongly. And all expected values are already stored in combo boxes, it can eliminate problem of spelling mistakes. As number of complements taken by word varies from sentence to sentence, we maintain sub categorization information based on number complements. Depending on the number of complements concerned combo boxes are enabled and other is disabled. It avoids user to enter wrong data.

#### **4.4 SUMMARY**

This chapter explains what lexicon is, and its importance in parsing. Explains what are the different attributes associated with each grammatical category of Sanskrit. A next section explains about design of tables of lexicon database and the interface designed to build the Lexicon database.

# Chapter 5

## PARSER

### 5.1 PARSING STRATEGIES

#### 5.1.1 An Overview

The present Chapter describes the Design and Implementation of Parser. There have been various approaches to the parsing problem. Main approaches include two left-corner parsing algorithms, a variant of the Cocke-Kasami-Younger algorithm, Early parsing algorithm, and Tomita's generalized LR parsing algorithm, in an LR(0) version. These are all Context-Free parsers. The context-free grammar (CFG) formalism, introduced by Chomsky, has enjoyed wide use in a variety of fields. CFGs have been used to model the structure of Programming languages and Natural languages [25]. Canonical methods for general CFG parsing are the CKY algorithm and Earley's algorithm. Both have a worst-case running time of  $O(gn^3)$  for a CFG of size  $g$  and string of length  $n$ , although CKY requires the input grammar to be in Chomsky normal form in order to achieve this time bound. Asymptotically faster parsing algorithms do exist. Graham, Harrison, and Ruzzo give a variant of Earley's algorithm that is based on the so-called 'four Russians' algorithm for Boolean matrix multiplication (BMM); it runs in time  $O(gn^3/\log n)$ . Rytter further modifies this parser by a compression technique, improving the dependence on the string length to  $O(n^3/\log^2 n)$ . But Valiant's parsing method, which reorganizes the computations of CKY, is the asymptotically fastest known. It also uses Boolean Matrix Multiplication; its worst-case running time for a grammar in Chomsky normal form is proportional to  $M(n)$ , where  $M(m)$  is the time it takes to multiply two  $m \times m$  Boolean matrices together. In the next section we are going to explain the bottom-up parser.

### 5.1.2 Bottom-Up Parser (LR Parsing Algorithm)

In bottom-up parsing we have various parsing algorithms like Shift-Reduce parsing, SLR, CLR and LALR.

The basic idea of a bottom-up parser is that we use grammar productions in the opposite way (from right to left). Like for predictive parsing with tables, here too we use a stack to push symbols. If the first few symbols at the top of the stack match the right hand side of some rule, then we pop out these symbols from the stack and we push the lhs (left-hand-side) of the rule. This is called a *reduction*. For example, if the stack is  $x * E + E$  (where  $x$  is the bottom of stack) and there is a rule  $E ::= E + E$ , then we pop out  $E + E$  from the stack and we push  $E$ ; i.e., the stack becomes  $x * E$ . The sequence  $E + E$  in the stack is called a *handle*. But suppose that there is another rule  $S ::= E$ , then  $E$  is also a handle in the stack. Which one to choose? Also what happens if there is no handle? The latter question is easy to answer: we push one more terminal in the stack from the input stream and check again for a handle. This is called *shifting*. So another name for bottom-up parsers is shift-reduce parsers. There two actions only:

1. Shift the current input token in the stack and read the next token. and
2. Reduce by some production rule.

Consequently the problem is to recognize when to shift and when to reduce each time, and, if we reduce, by which rule. Thus we need a recognizer for handles so that by scanning the stack we can decide the proper action. The recognizer is actually a finite state machine exactly the same we used for regular expressions (REs). But here the language symbols include both terminals and non-terminal (so state transitions can be for any symbol) and the final states indicate either reduction by some rule or a final acceptance (success).



## 5.2 DESIGN OF PARSER

### 5.2.1 Our Parsing Strategy

The efficiency of the parser plays a crucial role in machine translation systems. Therefore after studying various parsing approaches, we have decided to implement the LR Parsing algorithm. The LR parser uses the bottom-up approach.

#### 5.2.1.1 Generating Data Structure

In the data structure generator phase, for every word its Lexical information is picked up from the Lexicon. This lexical information of word is stored in a data structure which has link to previous and next structure for other words, and which can hold all lexical information of a word. This data structure for Verb category is shown in figure 5.1. Only, attributes which are specific to particular category varies from one category data structure another category data structure.

```
struct Node {  
  
    string word;  
    string category;  
    string number;  
    string person;  
    string tense;  
    string aspect;  
    int numberOfComplements;  
    string Complement1;  
    string Complement2  
    Tree treePointer;  
    Node next;  
    Node previous;  
    Node up;  
    Node down;  
}
```

Figure 5.1 Data Structure for verb category

Here verb category attributes are number, person, tense and aspect. So this attributes changes for another category. If we consider for noun category, attributes are noun case, number and gender, so, these attributes will be presented in noun category data structure instead of number, person, tense and aspect of verb category data structure.

In the Lexicon, one or more Lexical entries may be found corresponding to a word, for all entries, nodes will be created, and these nodes are connected vertically. This way, a multilevel structure (that is linked list of structures) will be created for a word. Same routine will be executed for every word and these nodes are connected horizontally, as shown in figure 5.2.

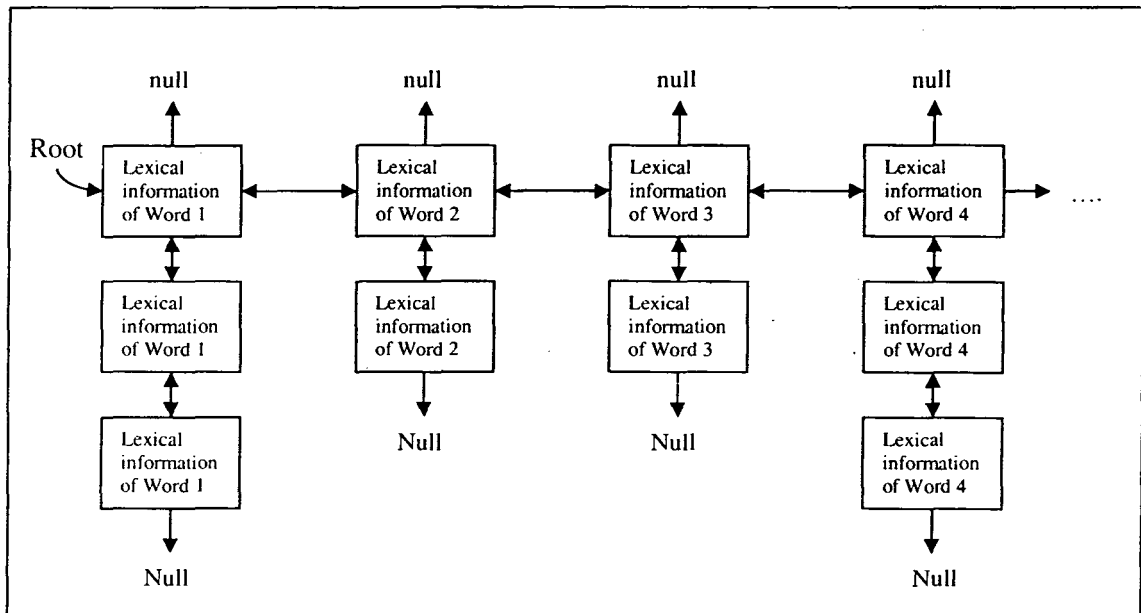


Fig. 5.2 Linked List Structure for sentence

### 5.2.1.2 Generating and Combining Trees

After the creation of linked list of words with its Lexical information, as per the category and sub-category information, Lexical Elementary-GB-Tree will be generated. After that, every tree will try to full fill all its requirements. I.e. it will try to get all required complements and specifiers and optional adjuncts. When ever a tree's all conditions are satisfied it is treated as realized and eligible to attach any other tree, provided it is having some relation with it. If it is not realized then tree will

deleted. This process will go on until the end of words, at the last one or more trees may be generated.

## **5.2.2 Over View of Design**

The analysis of the problem and its possible solutions led to the following design of the system. The overall design of the parsing system, in terms of the main modules and the inter-connection between them is shown in Fig.5.3. The Parsing System consists of 6 modules namely, Input Module, Preprocessor, Tagger, Data Structure Generator which is connected with Lexicon, X-Bar tree generator and Parser.

### **5.2.2.1 Input Module**

The Parser System contains a Text Box which can allow the Unicode characters also, where user enters Sanskrit sentence.

### **5.2.2.2 Preprocessor**

The module removes redundant spaces, if existing between words in the given sentence. In the Fig.5.3, this module is shown with dotted lines. The output of this module is referred to as normalized input and this is given as input to the Tagger.

### **5.2.2.3 Tagger**

The normalized input (the output of the Preprocessor) is subdivided into lexical items. The process of dividing the sentence into lexical items is more often known as Lexical Analysis. Given “john reads newspaper in the morning daily” as input this module would give the array (john, reads, newspaper, in, the, morning, daily) of subdivided lexical items.

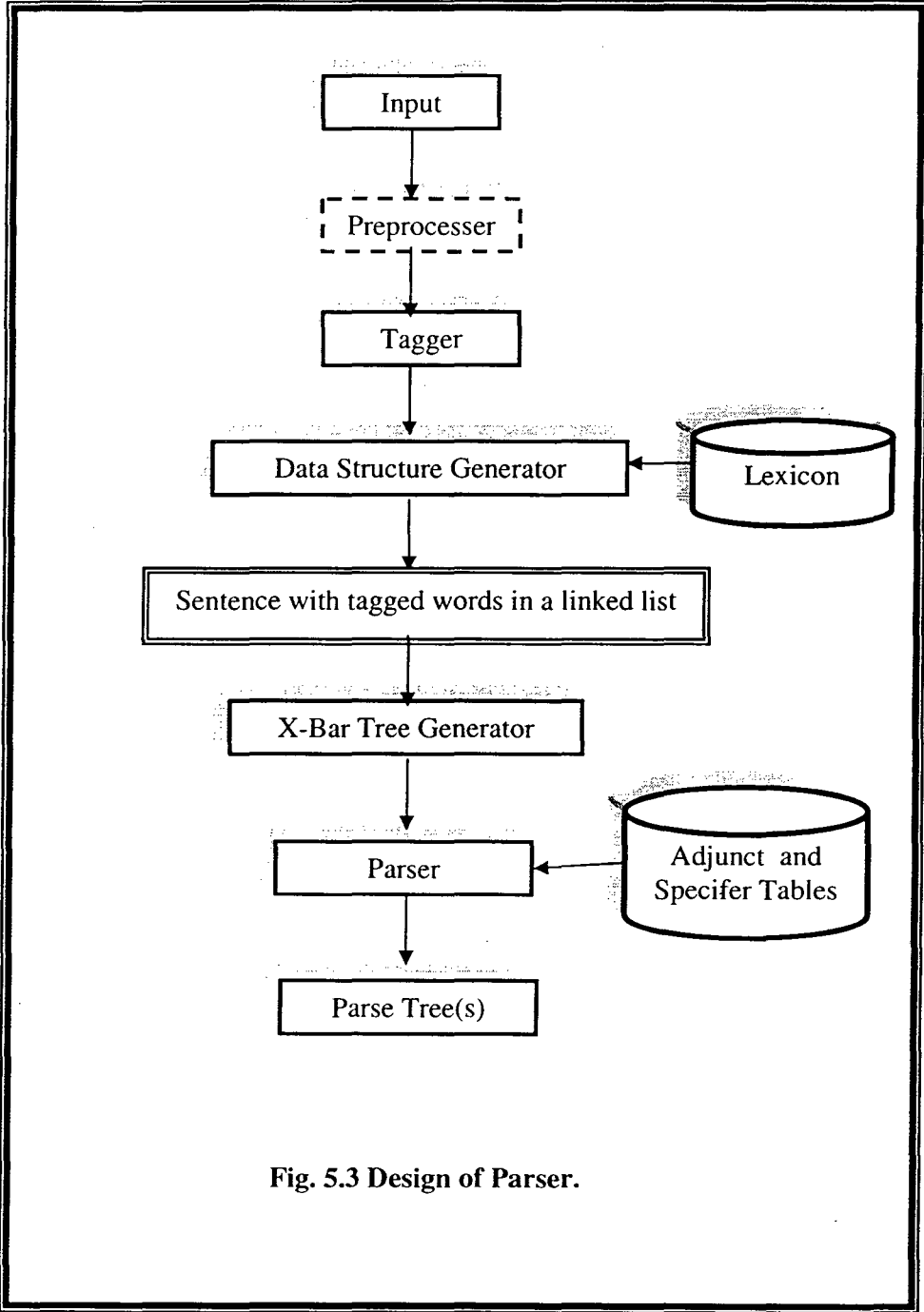
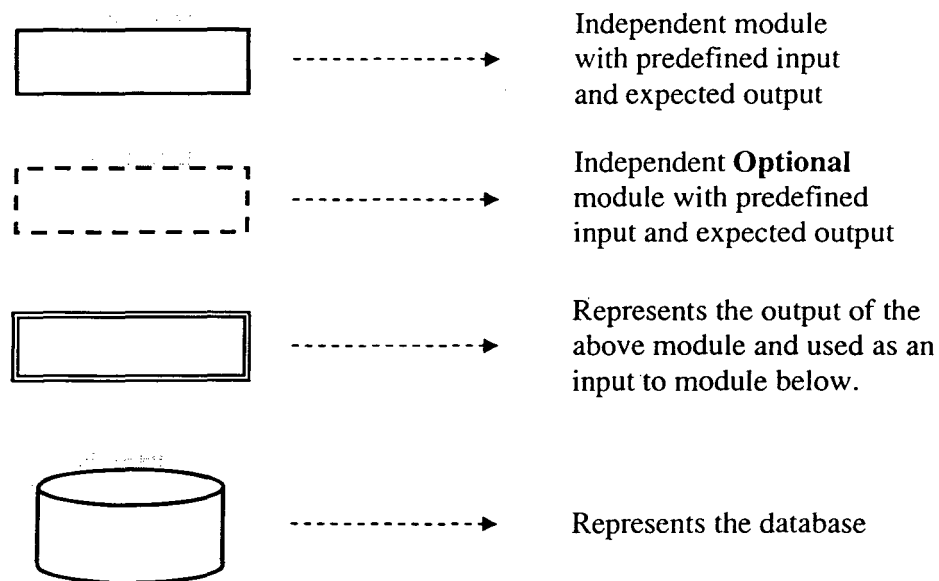


Fig. 5.3 Design of Parser.

### REPRESENTATIONS:



**Figure 5.3**

#### **5.2.2.4 The Lexicon**

The structure of the Lexicon has been described in Chapter 4. We may note that a given word may carry different category and subcategory information and other attributes also. For every word one or more lexical entries will be stored in the lexicon, which depends solely on the word.

#### **5.2.2.5 Data Structure Generator**

In run time, all the words are stored in a specially designed data structure along with all their attributes. This data structure is basically a doubly linked list, where every node has a pointer to the previous and next nodes. Here the nodes contain lexical information about words. Every word may have one or more entries in the Lexicon, so for a particular word, if it contains more than one entry in the Lexicon, the nodes will be attached vertically. So, it maintains a doubly linked list in vertical direction also. A typical data structure of a

node of word for verb category looks like as shown in Figure 5.1. As per the category some fields may vary. Here Number of complements may be less than are equal to 3, if only one complement available remaining fields kept empty.

In data structure, tree pointer contains a pointer to a tree which will be constructed as per the information stored in node. The complete data structure after reading all words shown in Figure 5.2.

### 5.2.2.6 X-Bar Tree Generator

The Lexicon stores the category information and sub-category information of each word, by retrieving this information X-Bar Tree generator generates the corresponding elementary tree for given word.

The Building X-Bar tree start from the bottom and goes to up, i.e. a lexical word is generated, followed by a “level 0” tree. And word is added as child to “level 0”(X0) tree. Then “level 1” (X’) tree will be generated. X’ may have none, one or two complements, which can be known from the lexicon information of word, as per this information number of child pointer will be present in X’ node. And “level 0” tree will be attached as child to “level 1” tree. Then phrase level (XP) tree will be generated and level 1 tree added as child for that. This tree is pointed by the “Tree pointer” attribute in node.

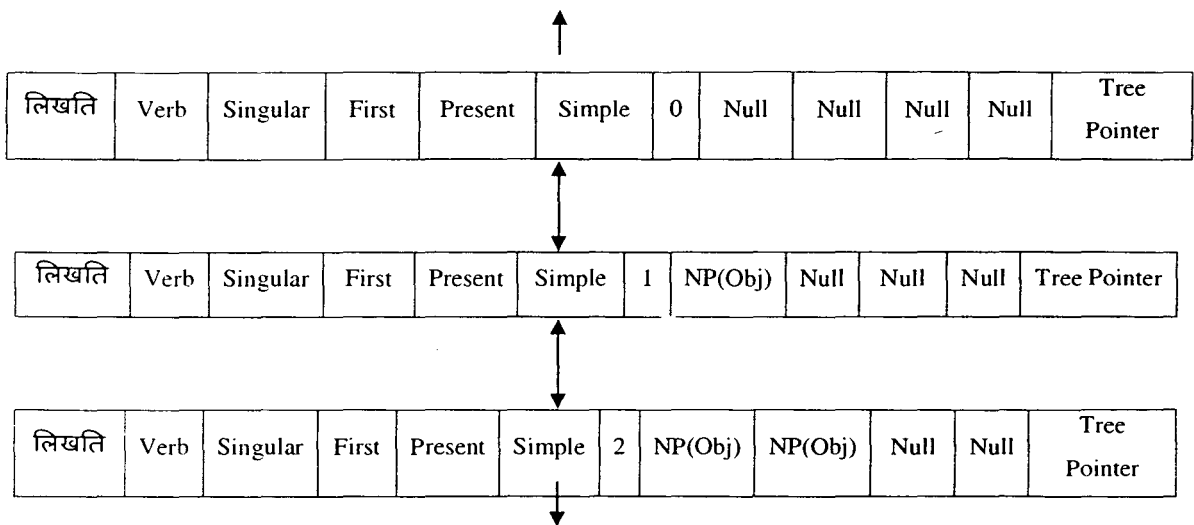


Figure 5.5 Data Structure for word लिखति

Example: if word लिखति encountered by X-Bar Tree generator, it will find three entries for this word as shown in Figure 5.5, first entry with no complements, second entry with one complement i.e a Noun Phrase and third entry with the two complements as Noun Phrases. X-Bar elementary trees for these entries are shown in Figure 5.6.

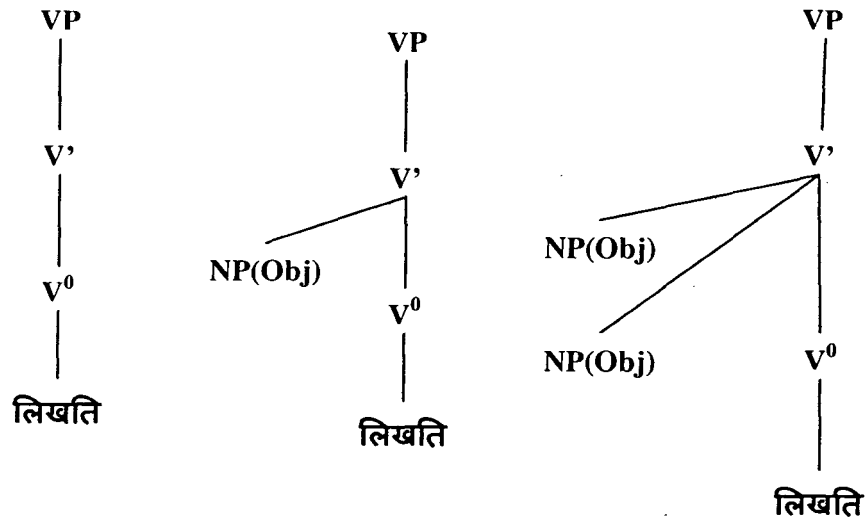


Figure 5.6 X-Bar Elementary Trees

### 5.2.2.7 Adjuncts and specifiers table

Every word type has its specific adjuncts and specifiers. For each kind of word type separate tables of adjuncts and specifiers will be maintained. A typical table of adjuncts for Verb category is shown in figure 5.7.

#### Verb Adjunct Table

NP(Ins)
NP(Dat)

Figure 5.7 adjuncts table for Verb category

### **5.2.2.8 Parser**

We have developed a Bottom-Up approach parser, using GB Phrase rules. The parser may generate zero or more parse trees for the given source language sentence. No parse tree is generated if the sentence does not conform to the GB rules of the source language. More than one tree may be obtained as the Natural Languages are ambiguous at each level, i.e. at word level, phrase level as well as at sentence level.

#### **5.2.2.8.1 Attaching Complements**

The parser traverse nodes from left to right in linked list as shown in the figure 5.2, for every node the parser tries to get its complements if it requires. In Sanskrit all complements are available in the left side of that node, so the parser checks in the left side of node to find the required complement, if it succeeds to find a complement, then it will attach complement tree at corresponding complement position. This complement position in present tree will be known using number of complements it has and index of the complement.

If the tree fails to get the required complements, that particular node will be removed from the data structure; as a result the corresponding tree will be removed. So that it cannot be attached as complement to any other tree, hence the parser can avoid generating unnecessary trees. As every node is being visited by parser, and the elementary tree in that node acquires required complements, at end complete trees will be generated when all nodes have been visited.

#### **5.2.2.8.2 Attaching adjuncts and specifiers**

After searching and connecting the complements for a particular word, parser checks for adjuncts to it with the information available in adjunct and specifier table. If parser gets any adjuncts, prepares a list of adjuncts and connects these to tree by modifying existing tree. After that parser searches for specifier in the left side of the word and connects it, if found.



### 5.2.2.8.3 Checking the completeness of Tree's

As for every sentence, subject will be treated as special case; IP should have subject of the sentence as the specifier. If IP does not find specifier, the tree will be discarded. Some trees may not have all words of given sentence, these sentences also discarded by parser. Remaining trees which represents the whole words in given sentence are treated as final trees, these are syntactically correct, in these some trees may be eliminated at the semantic level checking.

## 5.3 EXPLANATION OF PARSER WITH AN EXAMPLE

### 5.3.1 With a Correct Sentence's

This section explains the working of the parser with the help of an example. Let the Sanskrit sentences given to parser at different instances will be:

रामः लिखति

रामः मोहनम् लेखं लिखति

रामः लेखन्या सुन्दरम् लेखं लिखति

The Input Module reads the sentence as it is, presents this as an input to the Preprocessor. The Preprocessor removes the redundant blanks and converts the input sentences into a normalized form as shown below:

रामः लिखति

रामः मोहनम् लेखं लिखति

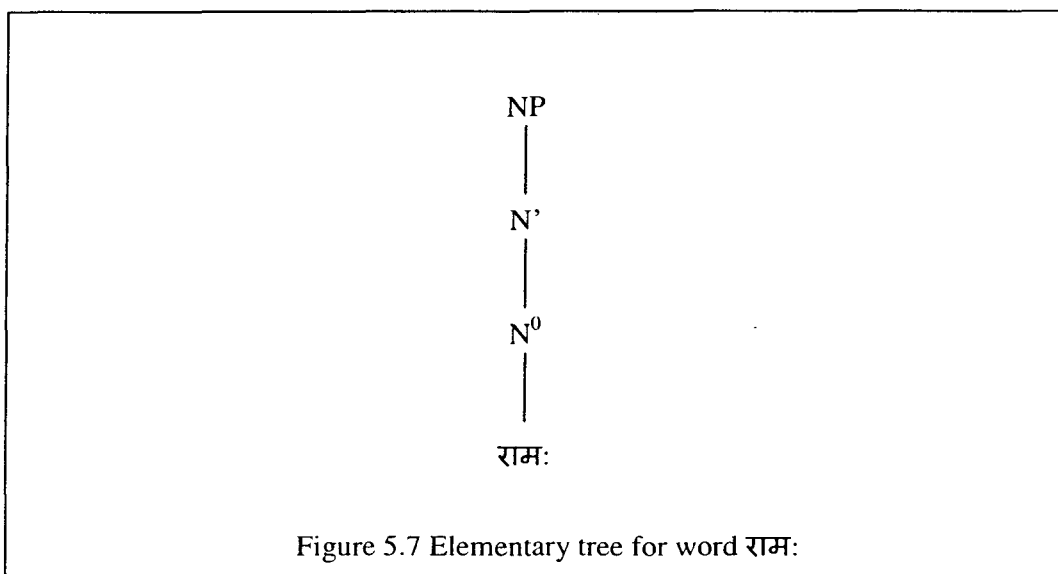
रामः लेखन्या सुन्दरम् लेखं लिखति

Next, the normalized sentence is sent to Tagger. The Tagger first divides the sentence into lexical items रामः and लिखति for first sentence, रामः, मोहनम्, लेखं and लिखति for second sentence and रामः, लेखन्या, लेखं and लिखति for last sentence.

These words will get all its attributes from the lexicon and as per the Lexicon information, elementary trees will be created. For creating elementary tree, category of word, number of complements and type of complements information only used.

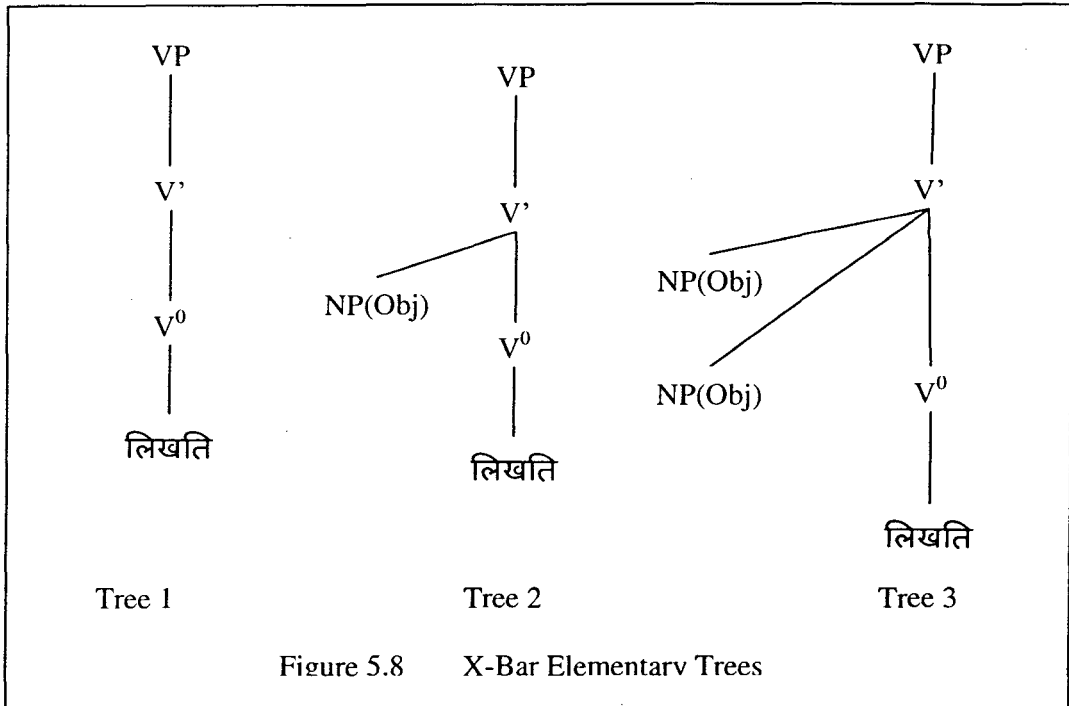
Now for every sentence parser reacts differently to build the complete parse tree. First we will see, how parser will work for the sentence “रामः लिखति”.

The word रामः have one entry in lexicon, so it will generate an elementary tree, this is shown in figure 5.7.

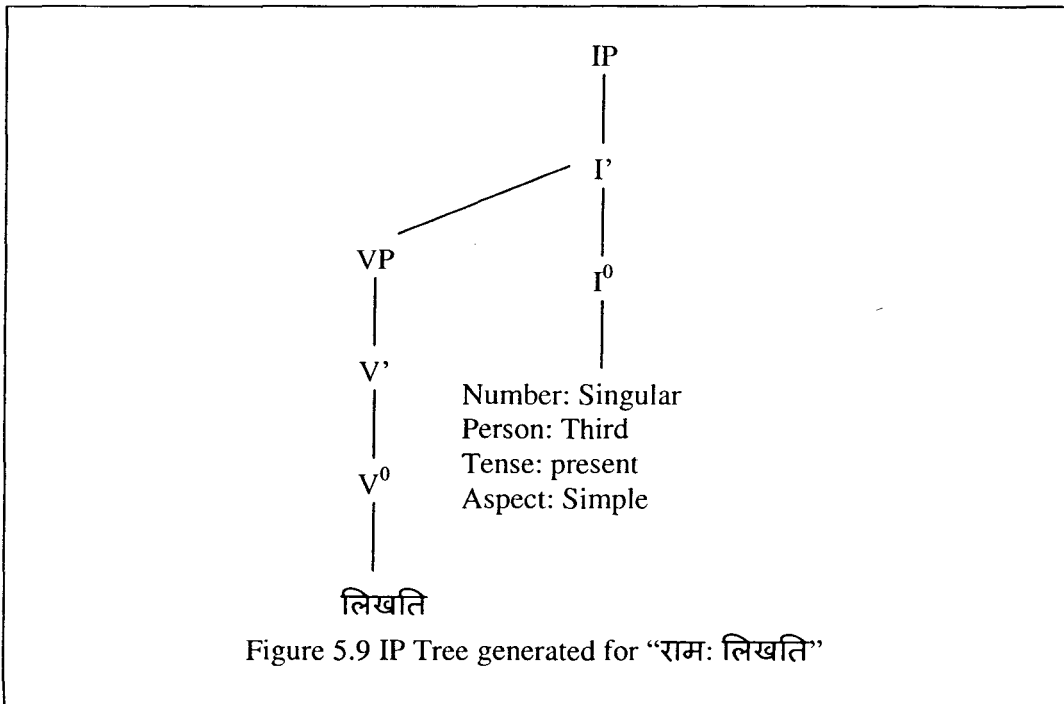


As it does not require any complement, parser not searches for complements. Now parser will check for the adjuncts with using the information stored in adjunct table in sentence for word रामः, as no adjunct is available, tree will remain same.

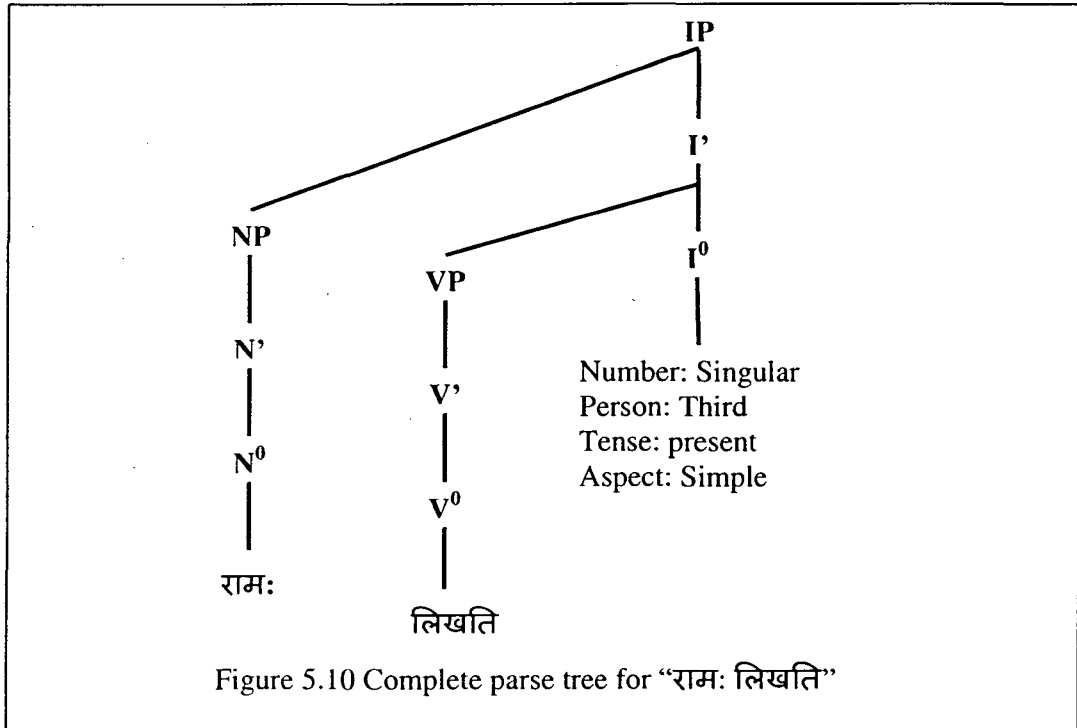
Now the word लिखति will be encountered by parser. This word has three entries in the lexicon, so three elementary trees will be generated as shown in Figure 5.8.



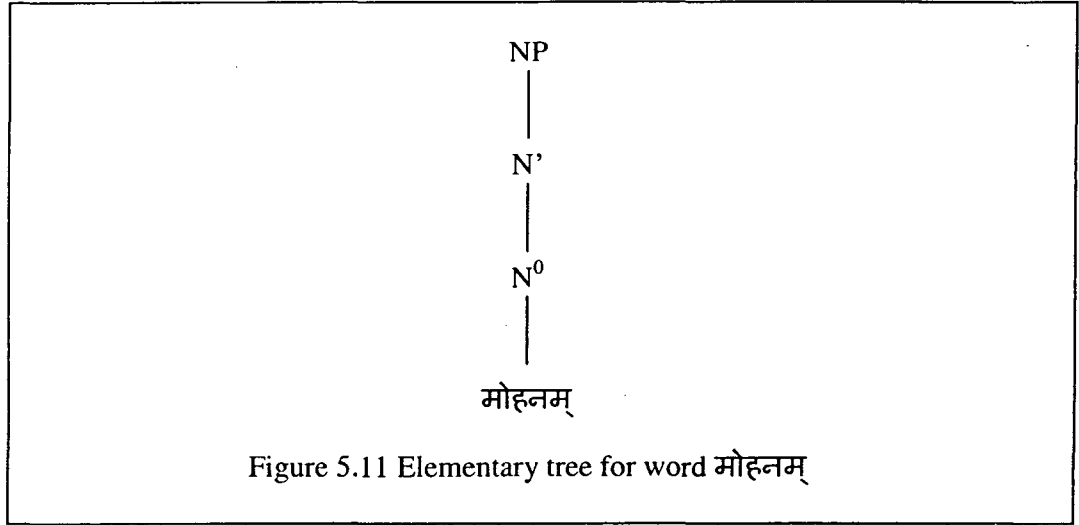
The tree 2 and 3 will be deleted, as these trees can not find required complements. So, only tree 1 will remain. As parser encountered VP, it generates an IP and connects VP as complements for it. So, resulting tree after generating IP is shown in Figure 5.9.



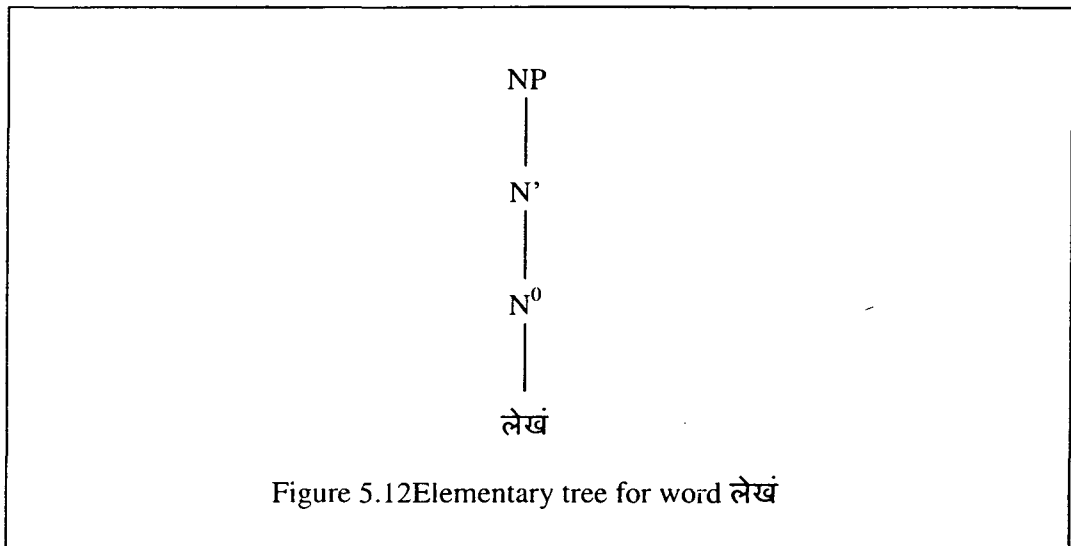
Now parser will check for the specifier for the IP, specifier is to be known from specifier table. Specifier of IP always a NP, with a Nominative case marker. So the final tree will be generated as shown in figure 5.10.



The next sentence “रामः मोहनम् लेखं लिखति” the parser acts differently. For word “रामः” parser acts similarly as acted for “रामः लिखति” and generates tree shown in figure 5.7.

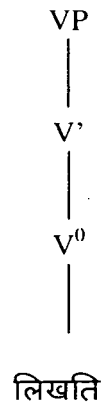


The next word is मोहनम् of sentence can have no complements. As word “मोहनम्” also did not get any adjunct, elementary tree generated will not be modified. The elementary tree is shown in Figure 5.11.

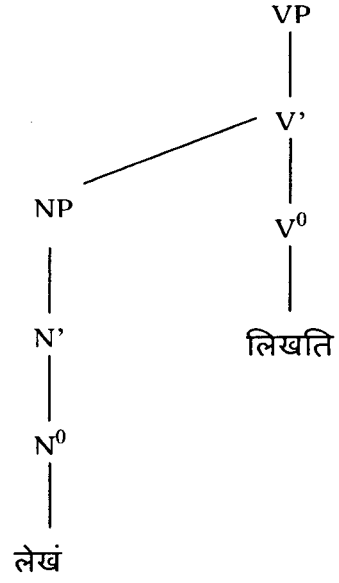


The next word is लेखं of third sentence can have no complements and can not get any adjuncts. The elementary tree is shown in Figure 5.12.

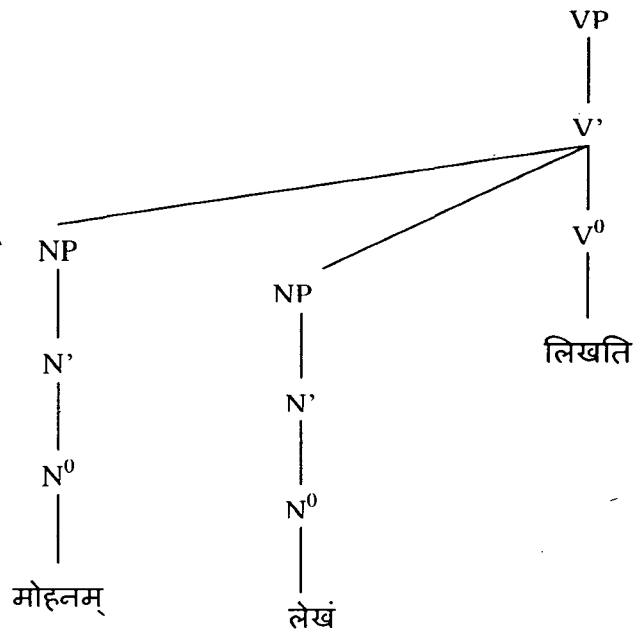
Now parser encounters word “लिखति” and generates three trees as shown in figure 5.8. All three trees will remain, as every tree can get required complements. That is first tree does not require any complement, second tree require one NP(Obj) complement which is available in form of “लेखं” and third tree requires two NP(Obj) as complements and can get these in form of मोहनम्, लेखं. So resulted trees looks like in Figure 5.13.



Tree 1



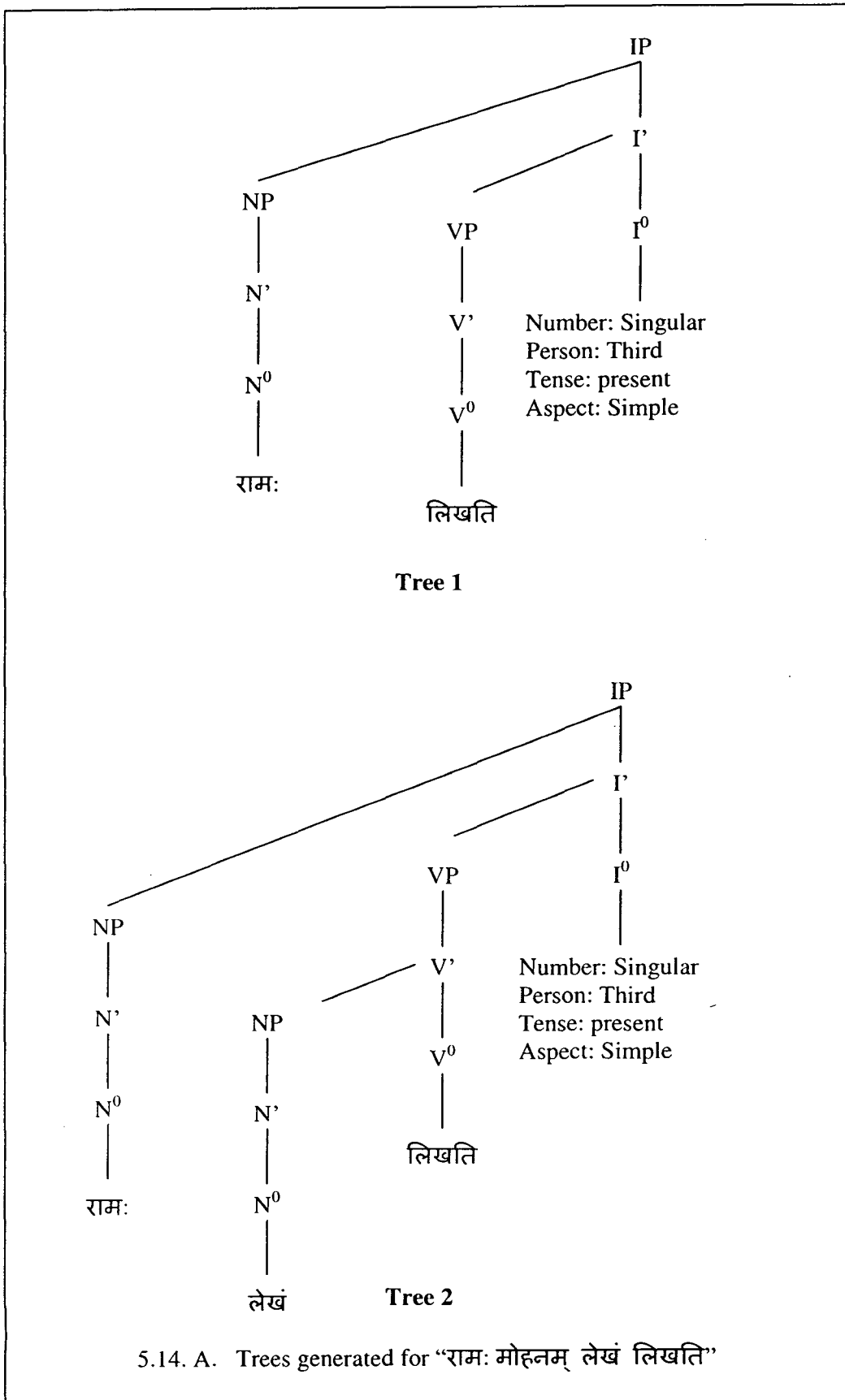
Tree 2



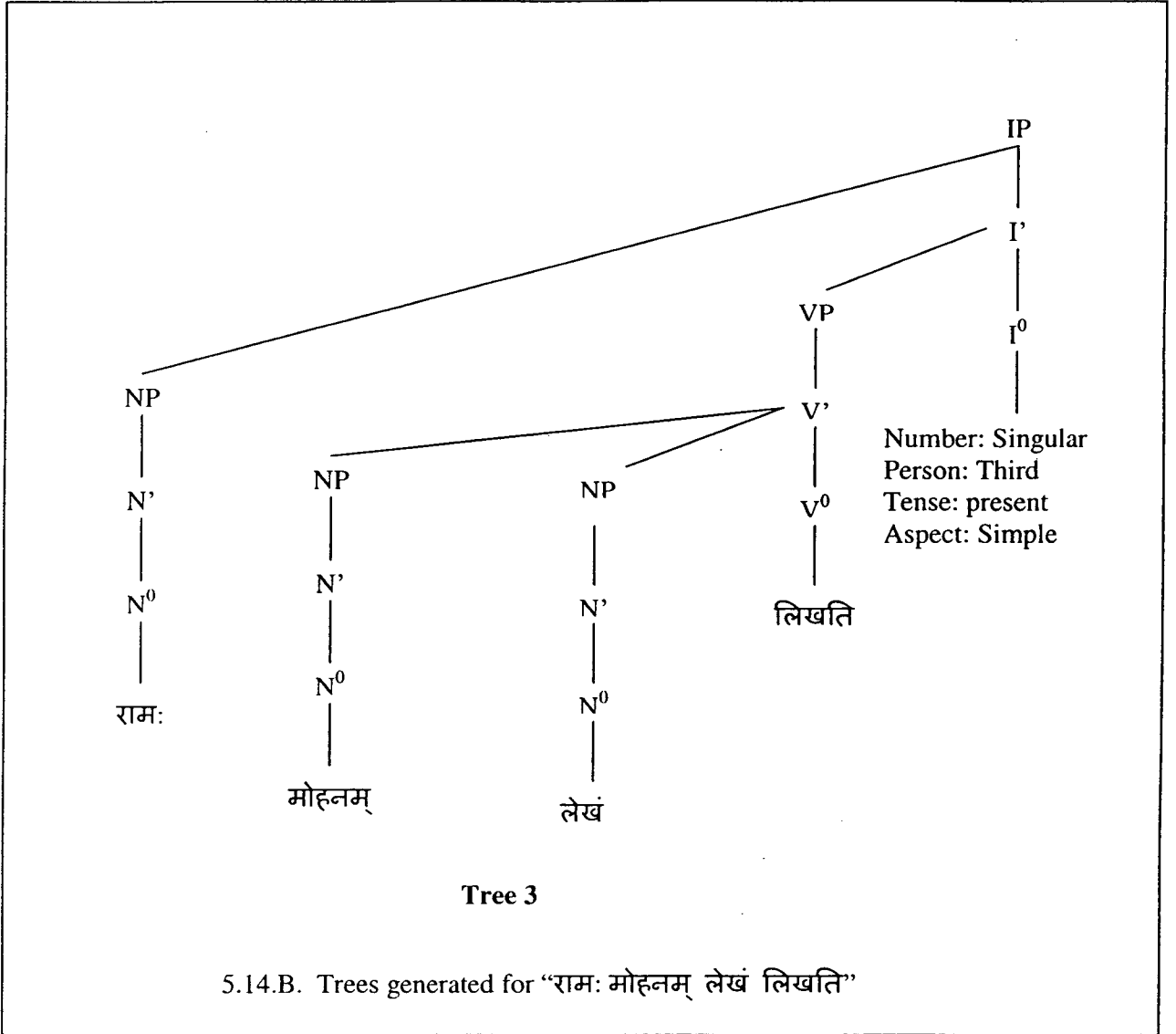
Tree 3

Figure 5.13 Intermediate trees for रामः मोहनम् लेखं लिखति

Next an IP will be generated for each VP and specifier will be attached to that tree.

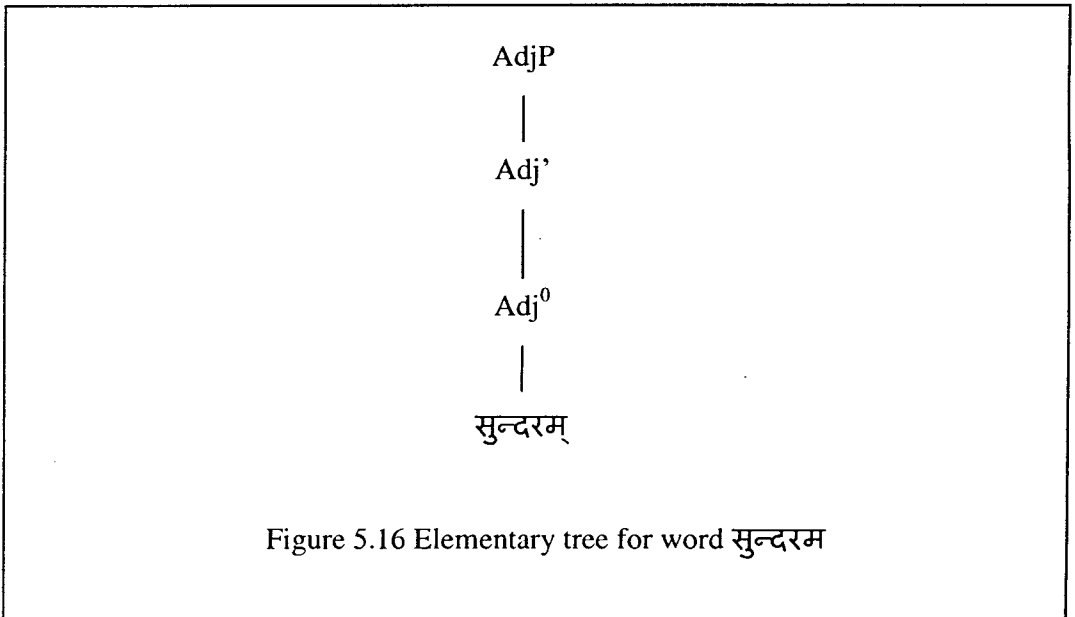
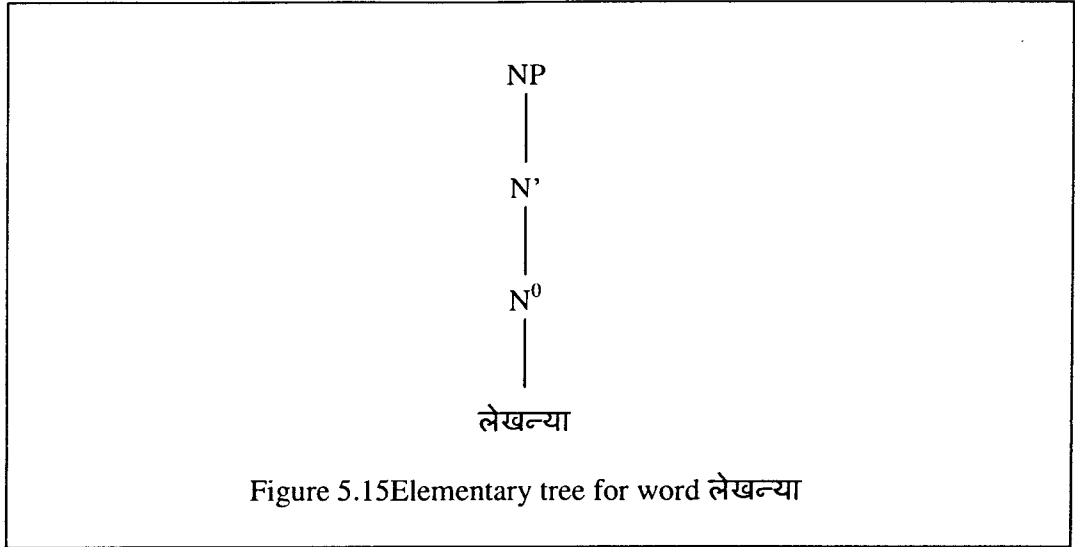




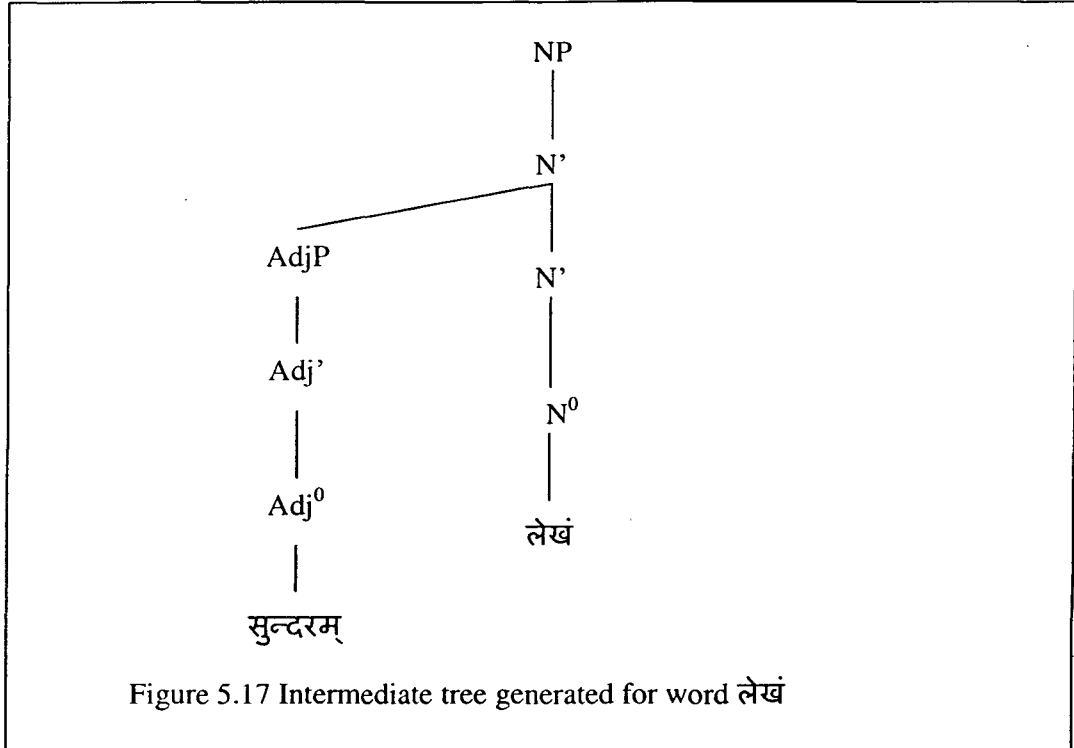


Here three final trees are generated, which are shown in figure 5.14. But only tree 3 represents all words in the sentence. So, remaining trees will be deleted. Tree 3 will become final tree.

The next is sentence “रामः लेखन्या सुन्दरम् लेखं लिखति”. In case “रामः”, its similar to earlier cases. So, it generates a tree shown in 5.7. The next word “लेखन्या” also not have any complements and its adjunct details does not match with any word in sentence, so a tree shown in figure 5.15 will be created. The next word is सुन्दरम् of the third sentence, which is adjective, will not find any matching adjuncts, so elementary tree generated as shown in figure 5.16.



The next word is लेखं can have no complements and can find a matched adjunct to it as, "सुन्दरम्". The elementary tree will be altered and tree shown in figure 5.17 will be created.



Now parser encounters “लिखति”, only first two trees can get required complements and resulted trees can look like as shown in Figure 5.18. Third tree will be deleted.

Now parser checks for the adjuncts for लिखति, parser can get “लेखन्या” as the adjunct for tree 2 of figure 5.18. “लेखन्या”, “सुन्दरम्” will be adjuncts for tree 1 of figure 5.18. So trees will be generated as shown in figure 5.19.

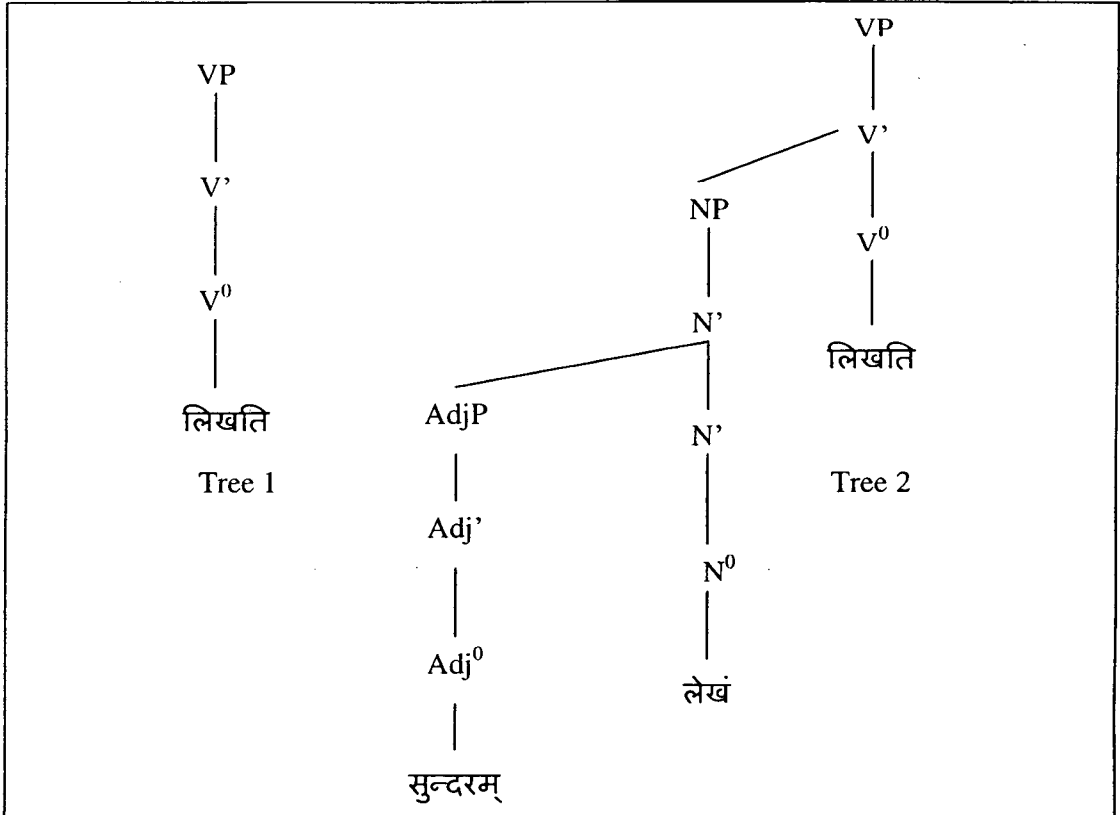


Figure 5.18 Intermediate trees for "रामः लेखन्या सुन्दरम् लेखं लिखति"

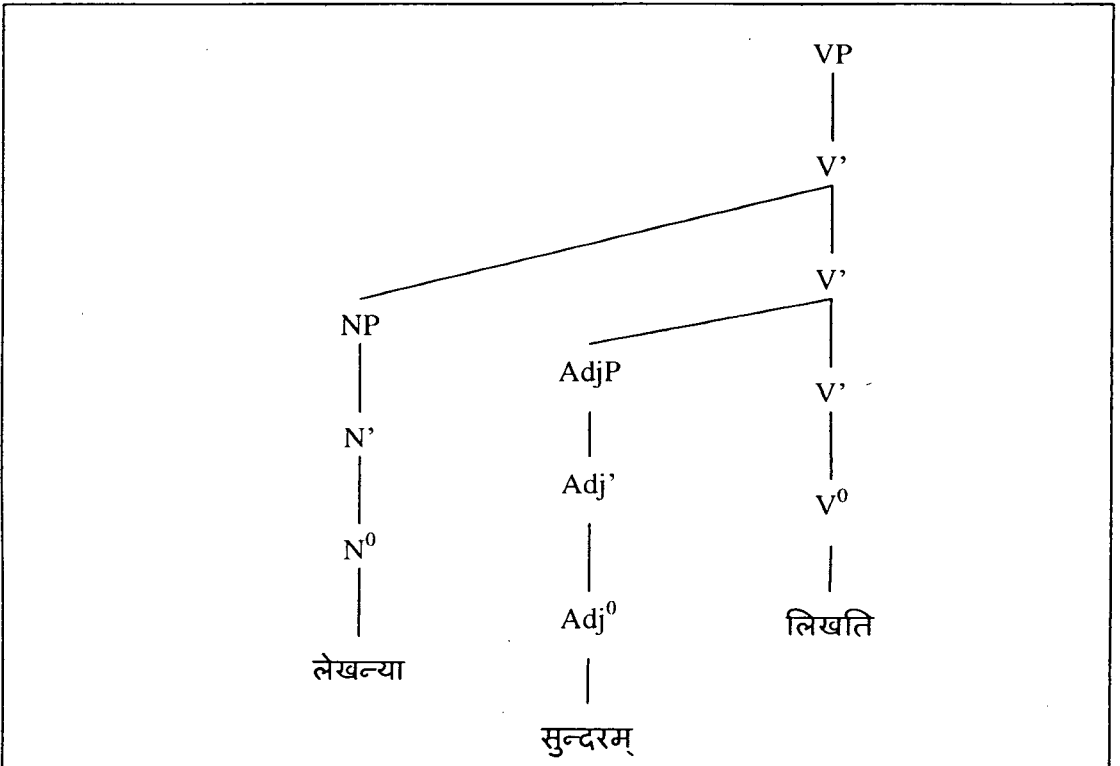
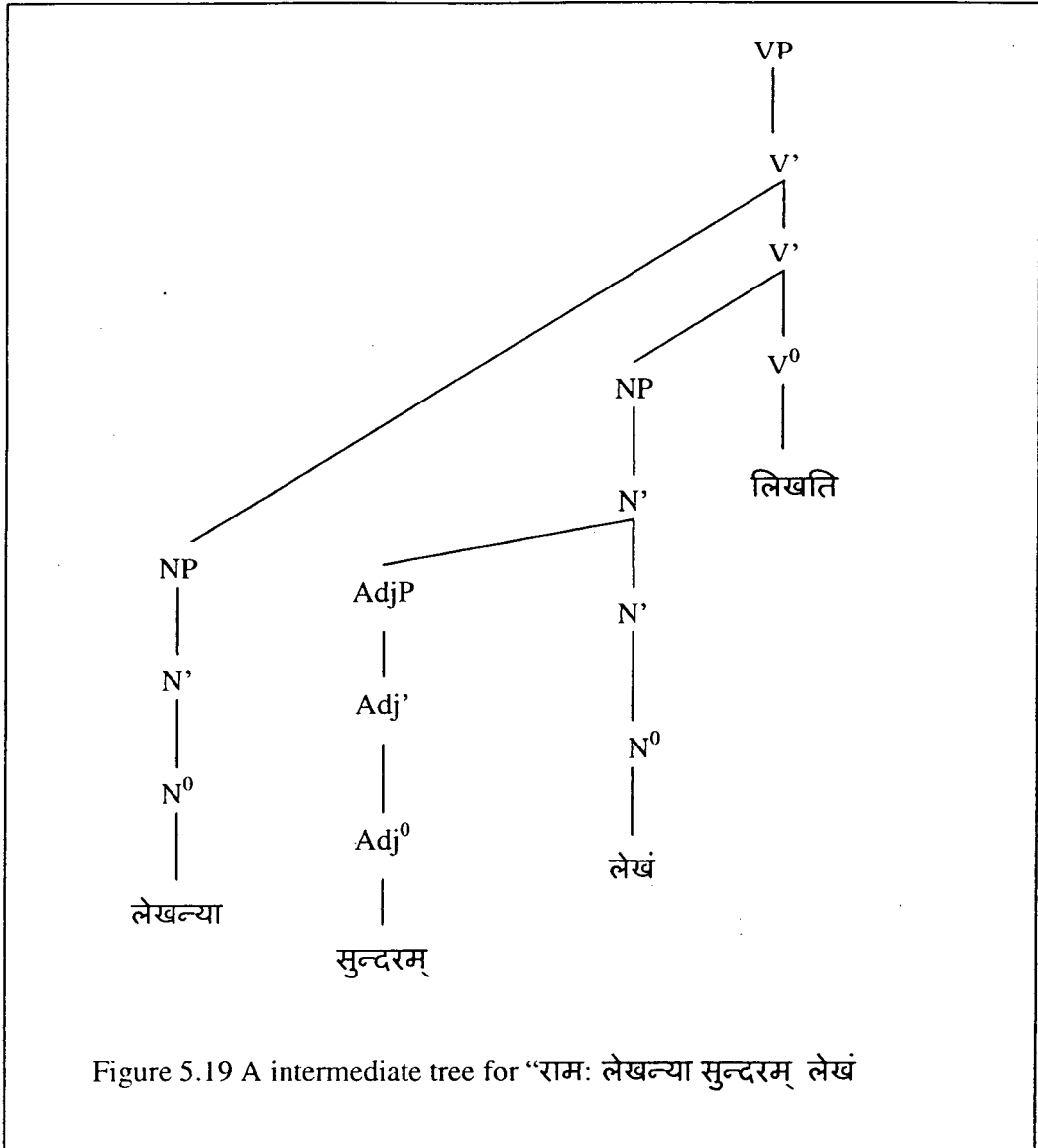
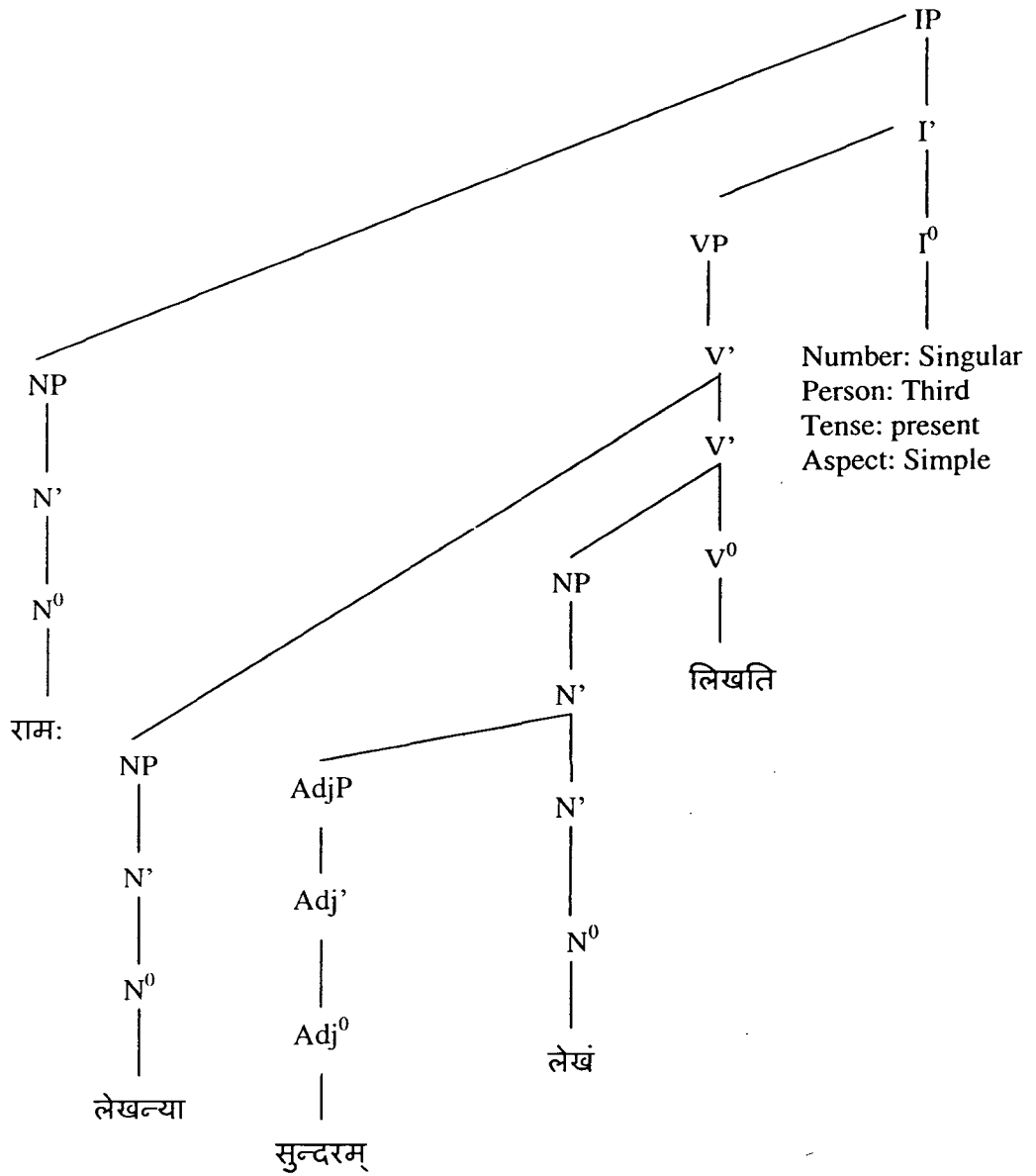


Figure 5.19 A Intermediate trees for "रामः लेखन्या सुन्दरम् लेखं लिखति"



Now for these 2 trees IP will be generated and, specifier will be connected. But tree 1 can not represents all words, so it will be deleted. So final tree is looks like as shown in figure 5.20.



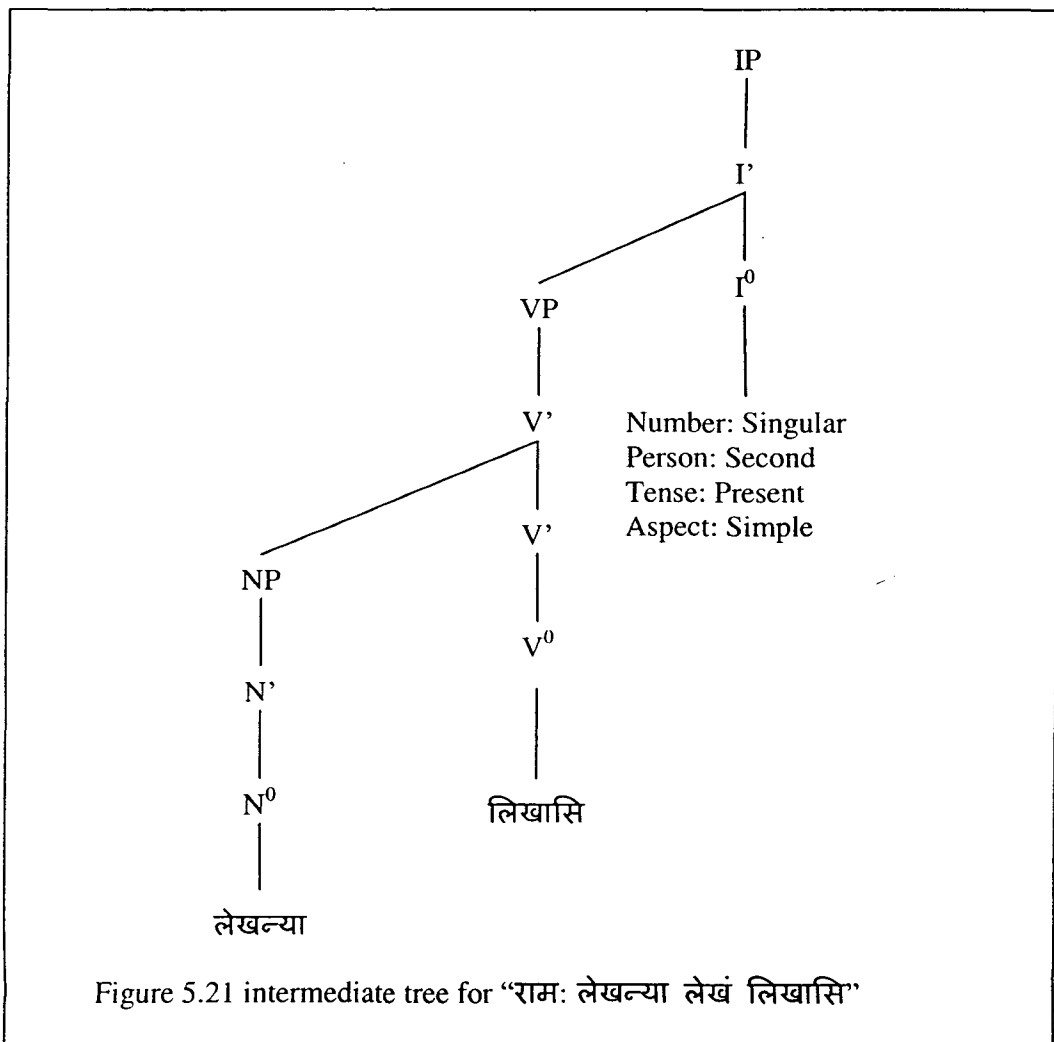
5.20 Complete parse tree generated for रामः लेखन्या सुन्दरम् लेखं लिखति

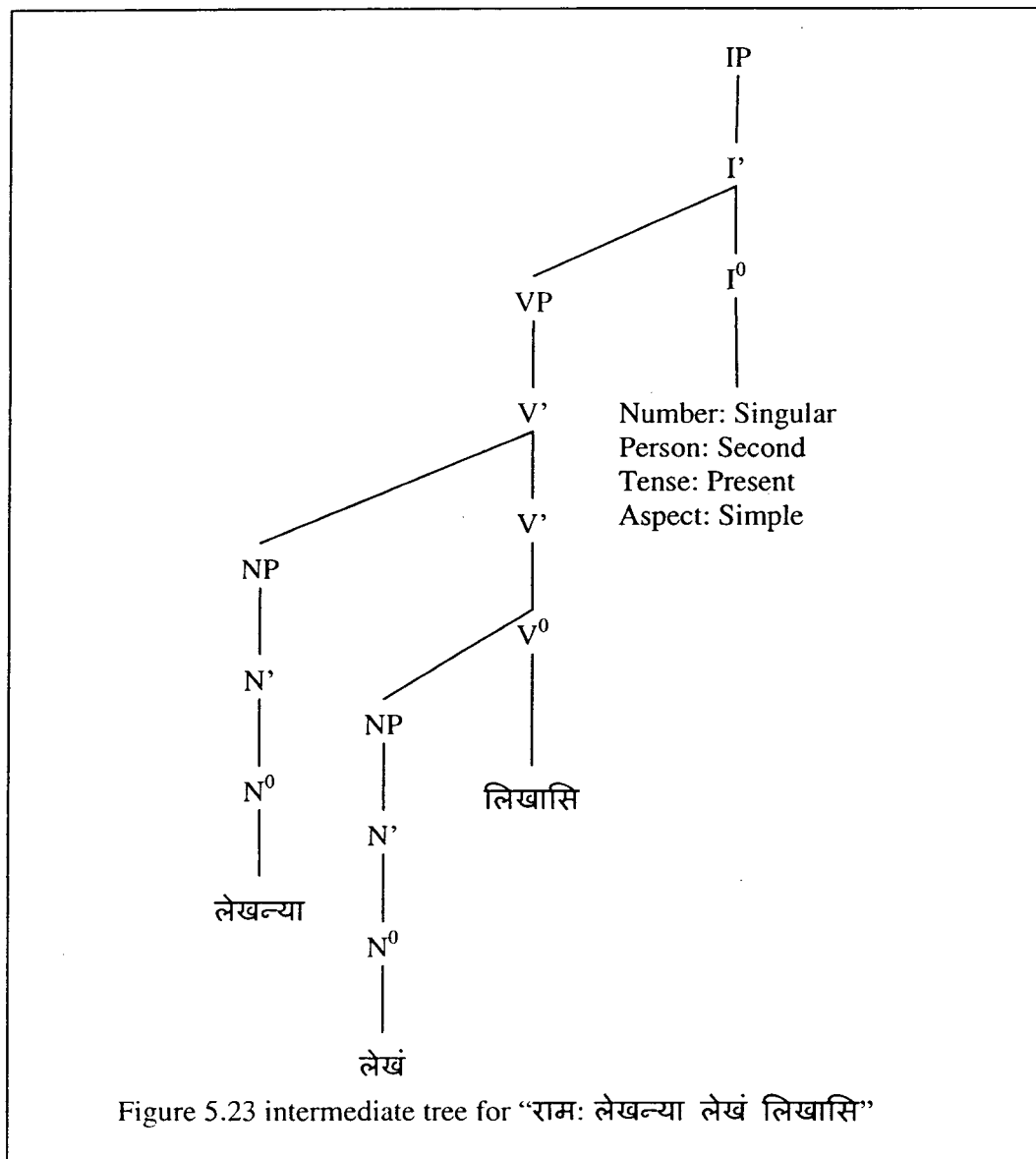
### 5.3.2 with a Wrong Sentence

This section explains how parser will reject a wrong sentence, when it given as input to the parser. If the following sentence given as input to parser,

रामः लेखन्या लेखं लिखासि

रामः, लेखन्या and लेखं will generate elementary trees as explained in previous section. The word “लिखासि” is the verb. Which have the same entries in lexicon as of “लिखति”, but only one attribute is different. That is, person is “second”. The IP will be generated as shown in figure 5.21 for this sentence.





Both trees need a specifier which is having a person attribute as second only, but word “रामः” is second person. So, parser will reject the sentence. So no tree will be generated finally.

#### 5.4 SUMMARY

Parsing is the heart in translation system, so various kinds of parsing strategies are studied. As we are handling with the elementary tree's we chosen the bottom-up approach parsing for parse the sentence. As per the lexicon information elementary X-Bar trees will be generated and every tree will go on get the complements to fill its requirements. at the last complete parse tree for a sentence is created.



## Chapter 6

# CONCLUSION AND FUTURE ENHANCEMENTS

### 6.1 CONCLUSION

We have developed Elementary trees based parsing for Natural Language with Sanskrit as language. While various parsing systems are being developed across the world using conventional approaches like Ruled- based or Example-based, we have adopted Government and Binding (GB) elementary tree approach in our Parsing system. The GB theory with its emphasis on Universal Grammar, its universality in handling Natural Languages, and its computational properties led to its choice over other conventional approaches. The GB frame work provides symmetric structures for the translation between any two pair of languages. The important modules of GB are X-Bar levels and phrase structures.

The phrase structure rules are developed by us both Sanskrit include Verb Phrase Structure, Noun Phrase Structure, Adjective Phrase Structure, and Inflection Phrase Structure. These Phrase Structures have been obtained after a thorough analysis of various phrases in Sanskrit language. The analysis includes determining the complements for each lexical type, determining adjuncts and specifiers for each type of Phrase Structure. In a sentence there are only two main types of phrases, Verb Phrase and the Noun Phrases.

A robust lexicon has been developed for the parsing system, which contain category and subcategory information of the every word. In general, the Lexicon contains the category and subcategory information for the words, the phonetic information (relating to speech sounds), and thematic information. However, in our case we are not using phonetic and thematic information. The lexicon developed by us for the parsing system can be further improved by defining and adding finer thematic information and phonetic information.

A Bottom-Up approach parsing technique using is developed of the parser. Since all we need to do is, recognizing the input (i.e. syntactic structure of the input), then Bottom-Up approach parser is the best method of choice. Our parser is able to parse all kinds of sentences, which may be ambiguous. Even for parsing, the complex sentences we need not change the basic parsing module, but simply enter more data in lexicon. Parser may produce more than one parse tree for sentence which are syntactically correct, but not semantically. Parser is developed such that it can work for any language provided lexicon data entered properly for that particular language.

The system is implemented using VISUAL C#. The rich GUI, Unicode support, .NET technology, easy connectivity with the databases, and its user-friendly nature led to the choice of Visual C# over other languages. The Lexicon is stored in databases. We thus have SQL Server 2000 functioning at the back-end of our translation system. The UNICODE has been used for storing the information.

## **6.2 FUTURE ENHANCEMENTS**

As pointed out above enhancements are needed in the area of lexicon for storing more information. Due to paucity of time, we have not covered movement, traces, empty categories, binding, and case assignment. In our opinion, one way of handling all these issues is to suitably modify the Phrase Structure (Rules). This will require few changes in the Parser. This therefore will be the next task we would like to take up. The current parser's space complexity also can be reduced by a little as its storing many intermediate trees.

It can be extended for more complex sentences which contain connectives like conjunctions and Question forms. At this time it is also handling simple sentences not having Complementizer Phrases. It can be extended for interrogative sentences.

We have already said that Sanskrit is a completely word order free language. There is a need therefore to convert a word order free sentence into a structured sentence needed for GB frame work. Here I assumed every sentence is in Subject-Object-Verb order, if sentence is in other order that has to be modified. This is quite a heavy task. Further, in Sanskrit Sandhi plays a very important as it is a common practise in Sanskrit to present combinations of words as a single word. What this means is that in

a sentence two or more words may be written together as a single combination replacing the original individual words. This problem in itself is quite complicated and needs lot of thinking along with developing Sandhi and Sandhi-Vichedi modules.

Reverse morphology (finding root word from derived word) process has to included which can reduce the size of lexicon and improve efficiency of parser.

As the work will progress further we may see the necessity of other modifications and processes in the framework.

# APPENDIX

CP stands for Complementizer Phrase

NP stands for Noun Phrase

VP stands for Verb Phrase

PP stands for Prepositional Phrase

AdjP stands for Adjective Phrase

ADV stands for Adverb

## REFERENCES

- [1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: principles techniques and tools*. Addison-Wesley, 1986.
- [2] Aravind K. Joshi. 1987. An Introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*, John Benjamins, Amsterdam.
- [3] Aravind K. Joshi and Yves Schabes. 1992. Tree-adjoined grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*. Elsevier Science.
- [4] Arnold, D., Balkan, L., Humphreys, R. L., Meijer, S. & Sadler, L. (1994), *Machine translation: an introductory guide*, Blackwells/NCC, London. An HTML and a Postscript version of this book is available at <http://clwww.essex.ac.uk/MTbook/> and <http://clwww.essex.ac.uk/MTbook/HTML/> respectively.
- [5] B. Carpenter.  
The Logic of Typed Feature Structures with Applications to Unification Grammars, Logic Programs and Constraint Resolution, Cambridge University Press, 1992, no ISBN 0-521-41932.
- [6] Boullier, P.: Range concatenation grammars. In: Proceedings of IWPT '00, Trento, Italy (2000) 53–64
- [7] C. Pollard, I. A. Sag. *Head-Driven Phrase Structure Grammar*, University of Chicago Press, Chicago, 1994.
- [8] Chomsky, Noam. 1981. *Lectures on Government and Binding*. Dordrecht: Foris Publications.

- [9] Fromkin (2000), Chapter 3.2 (Constituent Order, Case Marking and Thematic Roles)
- [10] Fromkin, V.A. and Rodman, R. 1997. Introduction to Language. Harcourt Brace. 6th edition. Translated into Portuguese, Japanese, Chinese, Korean, Hindi, Dutch.
- [11] Gazdar, G. (1985), Applicability of indexed grammars to natural language. Technical Report CSLI{85-34, Center for the Study of Language and Information, Stanford.
- [12] Haegeman, Liliane. 1994. Introduction to Government and Binding Theory. 2 ed. Oxford: Blackwell.
- [13] Hutchins, W. J. & Somers, H. L. ( 1992), An introduction to machine translation, Academic Press, London.
- [14] M. R. Kale. (1988), A Higher Sanskrit Grammar
- [15] McCawley, James D. 1998. The Syntactic Phenomena of English. 2 ed. Chicago: University of Chicago Press.
- [16] R. M. Kaplan, J. Bresnan.  
Lexical-Functional Grammar: A formal system for grammatical representation,  
in: "The Mental Representation of Grammatical Relations, Cambridge, MA", J.  
Bresnan (editor)., Reprinted in Mary Dalrymple, Ronald M. Kaplan, John  
Maxwell, and Annie Zaenen, eds., Formal Issues in Lexical-Functional  
Grammar, 29-130. Stanford: Center for the Study of Language and Information.  
1995., The MIT Press, 1982, p. 173-281.
- [17] [http://pt.wikipedia.org/wiki/Tradutor\\_autom%C3%A1tico](http://pt.wikipedia.org/wiki/Tradutor_autom%C3%A1tico)
- [18] [www.systransoft.com](http://www.systransoft.com)

- [19] <http://www.languageinindia.com/jan2005/aparnasanskritdissertation1.html>
- [20] Panini, *The Ashtadhyayi*
- [21] Ullman J. D. and Hopcroft J. E. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [22] Wren & Martin, 2001, *High School English Grammar & Composition*.
- [23] [http://en.wikipedia.org/wiki/Government\\_and\\_binding](http://en.wikipedia.org/wiki/Government_and_binding)
- [24] Noam Chomsky, *The Minimalist Program (Current Studies in Linguistics)*. Massachusetts Institute of Technology, 1995.
- [25] Aho, Sethi, and Ullman, 1986; Jurafsky and Martin, 2000; Durbin et al., 1998

