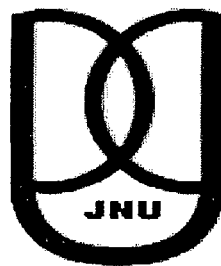


**ELEMENTARY-GB-TREES
BASED PARSING FOR HINDI**

*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
V.R.R Naidu Nagam

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 HINDI**” being submitted by **V.R.R NAIDU NAGAM** 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

Forwarded

01/09/06
Prof. Balasundaram
Dean, SC&SS,
JNU, New Delhi-67



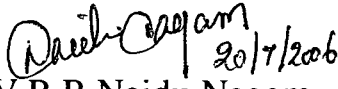
जवाहरलाल नॅहरू विश्वविद्यालय

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 HINDI**” 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.


V.R.R Naidu Nagam,
M.Tech, Final Semester,
SC & SS, JNU,
New Delhi.

**Dedicated to
My beloved
Grand Parents**

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.

(V.R.R Naidu Nagam)

CONTENTS

Chapter 1	1
INTRODUCTION	1
1.1 NATURAL LANGUAGE PROCESSING	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	2
1.1.5 Language Translation	3
1.2 MACHINE TRANSLATION	3
1.3 INTRODUCTION TO PARSING	3
1.3.1 Parsing Control Strategies	4
1.3.1.1 Top-down vs Bottom-up	4
1.3.1.2 Dept first vs Breadth first	5
1.3.1.3 Combinations of Strategies	5
1.3.2 Evaluation of Strategies	6
1.3.2.1 Top-Down vs. Bottom-up Strategies	6
1.3.2.2 Depth-First vs. Breadth-First Strategies	7
1.3.2.3 Node-Selection Strategies	7
1.4 PROBLEM DEFINITION	8
1.5 ORGANISATION OF THE THESIS	8
Chapter 2	10
GOVERNMENT AND BINDING THEORY	10
2.1 INTRODUCTION	10
2.2 OVERVIEW OF GOVERNMENT AND BINDING THEORY	10
2.3 CONSTITUENT STRUCTURE AND SUBCATEGORIZATION	11
2.4 X-bar Theory	13
2.4.1 The Phrasal Projection for a Noun	14
2.4.2 The Phrasal Projection for an Adjective	15
2.4.3 The Phrasal Projection for a Verb	16
2.5 Lexicon Organization	17
2.5.1 Categorical Information	18
2.5.2 Subcategorization Information	18
2.5.3 Thematic Information	19
2.6 Theta Theory	20
2.6.1 The Argument Structure of Other Syntactic Categories	22
2.6.2 The Theta Criterion	23
2.6.3 The Projection Principle	23
2.6.4 The Extended Projection Principle	24
2.7 OTHER CONCEPTS	24
2.7.1 Case Theory	24
2.7.2 Binding Theory	25
2.8 SUMMARY	26

Chapter 3	27
GOVERNMENT AND BINDING BASED GRAMMAR FOR HINDI	27
3.1 Introduction	27
3.2 THE OVERALL STRUCTURE OF THE HINDI VERBS	27
3.2.1 Verb Forms	27
3.2.2 The Overall Structure of the Hindi Nouns	42
3.2.3 Adjective forms for Hindi	46
3.2.4 Postposition Forms for Hindi	47
3.2.5 Adverb Forms for Hindi	48
3.3 Hindi Phrase Structures	48
3.3.1 A Verb Phrase	48
3.3.2 A Noun Phrase	49
3.3.3 An Adjective Phrase	49
3.3.4 A Prepositional Phrase	50
3.3.5 An Inflection Phrase	51
3.3.6 A Complementizer Phrase	52
3.4 SUMMARY	52
Chapter 4	53
Lexicon	53
4.1 INTRODUCTION	53
4.2 STRUCTURE OF HINDI LEXICON	54
4.2.1 Introduction	54
4.3 DESIGN OF LEXICON	58
4.3.1 Word table	58
4.3.2 Subcategory table	59
4.3.3 Interface	59
4.4 SUMMARY	60
Chapter 5	61
PARSER	61
5.1 PARSING STRATEGIES	61
5.1.1 An Overview	61
5.1.2 Bottom-Up Parser (LR Parsing Algorithm)	62
5.2 DESIGN OF PARSER	63
5.2.1 Our Parsing Strategy	63
5.2.2 Over View of Design	65
5.3 explanation of PARSER with an Example	71
5.3.1 With a Correct Sentence's	71
5.3.2 with a Wrong Sentence	85

5.4 SUMMARY	87
Chapter 6	88
CONCLUSIONS AND FUTURE ENHANCEMENTS	88
6.1 CONCLUSIONS	88
6.2 FUTURE ENHANCEMENTS	89
APPENDIX	90
REFERENCES	91

ABSTRACT

An Elementary-GB-Trees Based Parsing system for Hindi language sentences has been developed, which may be used to translate Hindi 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 Hindi, GB Phrase Structures for Hindi 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.

1.1.2 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.

1.1.3 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.

1.1.4 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.

1.1.5 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 INTRODUCTION TO PARSING

Sentence analysis is fundamental to most NLP applications it is an essential component of, for instance, natural language interfaces, translation systems, text summarization systems, and grammar checkers. This chapter describes sentence analysis, focusing on Systemic sentence analysis. This chapter outlines the basic analysis strategies. The term 'analysis' will be used to describe the construction of a description on any stratum derived from a description on a lower stratum. We thus have 'graphological analysis', where we break up an input text into graphological units (spellings, sentences etc.); lexico-grammatical analysis, whereby we produce a lexico-

grammatical representation based upon a graphological analysis; and micro-semantic analysis, whereby we produce a micro-semantic representation derived from the lexico-grammatical and graphological descriptions. We will use the term parsing to refer specifically to lexico-grammatical analysis.

1.3.1 Parsing Control Strategies

Over the years, many different approaches have been tried to apply a set of grammar rules to the analysis of a sentence. These various approaches can be split up on a number of dimensions, each dimension concerning different priorities in the construction of a parse tree. This section will look at three such dimensions:

- Top-Down vs. Bottom-Up;
- Depth-First vs. Breadth-First;
- Node Selection Strategies.

1.3.1.1 Top-down vs Bottom-up

A **top-down** approach starts at the root of the tree (traditionally the sentence), and builds down to the leaves of the tree (words, or in some cases, morphemes). A bottom up approach starts with the leaves and builds upwards towards the sentence. The top-down algorithm is theoretically based on the idea of using a generative grammar to produce all possible sentences in a language until one is found which fits the input sentence. This would in most cases take too much time, but there are ways to restrict the method to the most fruitful possibilities.

The bottom-up algorithm tries to combine elements in the input sentence in different ways until a tree covering the whole sentence is found. It starts with the single words in the sentence. These are grouped together into larger and larger units until the whole sentence is grouped together. **Left-Corner Parsing:** There are some parsing algorithms which use a mixture of top-down and bottom-up strategies. The Left-Corner parsing algorithm for instance invokes a rule bottom-up and finishes it top-down. Such a parser builds sentence structure in a left-to-right, bottom-to-top fashion, piecing together the left corner of a structural description first.

1.3.1.2 Depth first vs Breadth first

A depth-first approach moves as quickly as possible between root and leaves (in a top down approach) or leaves and root (in a bottom-up approach). A breadth-first approach explores all branches at each level before going up/down a level. There is a difference between whether to explore one path to its limits before trying others (for example, to determine everything necessary about one word of the input, before proceeding to the remaining words), or whether to explore all paths simultaneously (for example, to carry out the first stage of the process on all the words of the input, before proceeding to the next stage). A depth-first approach stresses the vertical dimension of a tree, breadth-first its horizontal dimension, taking into consideration all nodes at the same level in the tree.

1.3.1.3 Combinations of Strategies

The above discussion has defined two strategic alternatives in the design of a parsing algorithm. There are thus four combinations of these strategies:

- Top-Down/Depth-First
- Top-Down/Breadth-First
- Bottom-Up/Depth-First
- Bottom-Up/Breadth-First

Node Selection Strategies

The strategy combinations we have defined above do not limit all the possibilities in the construction of a parse tree. For instance, when parsing bottom-up and depth-first, these strategies do not say which word in the input string we should start with. We could start with the first, but this is only one possibility. When parsing top-down, any of the possible constituents of the present unit could be expanded first. A further restriction on the parsing process is needed to nominate which node of the parse tree should be expanded first. Several possible strategies exist:

- **Left-to-right Expansion:** the left-most node is expanded first. In a bottom-up parser, this means incorporating words of the input string starting from the leftmost and proceeding until the right-most is incorporated. For a top-down parser, this means

expanding the left-most elements of the present unit's constituents before those to the right.

- **Head-driven Expansion:** In a head-driven parser, some key item at each level of structure (the "head") is expanded first, since this unit is seen as the controller of all other units at this level. For instance, a bottom up key-search parser would start by parsing the verb, since this item controls the presence and nature of other structural elements (e.g., a transitive verb expects a Complement, while an intransitive one does not). Nouns may be selected as a second-level key, since they control much of the structure of the nominal group. The head-driven approach first scans the input for a particular item, and then begins the analysis at that item. This option needs to be motivated by a theoretical belief that the chosen item is in some way a key to analyzing the input. For instance, it may be felt that the particular nature of the verb will be the most important information to the parser in, say, segmenting the constituents of the clause. Thus the parsing device will begin its analysis at the verb.
- **Resource-driven Expansion:** A resource-driven approach uses the grammatical resources to direct the analysis, not the representations. The resources thus decide which node is to be expanded first. For instance, a shift reduce parser tries to successively apply all the rules of the grammar to the partially developed parse tree. The parse tree is expanded at the point at which the rule fits the tree (which could be at any point in the parse tree). The input string is thus not necessarily processed left to right (although some shift reduce parsers do work left to right, by applying all the rules to the left-most expansion point in the parse tree, before advancing to the next point).

1.3.2 Evaluation of Strategies

This section evaluates the various strategies discussed above.

1.3.2.1 Top-Down vs. Bottom-up Strategies

The most powerful of the bottom-up parsing methods, the so-called LR(k) parsers, accept a wider class of languages than do the most powerful top-down parsers. Also,

top-down parsing tends to be less efficient than bottom-up parsing, because a large degree of hypothesis making may be required before the input string is reached. The more complex the grammar, the higher the degree of hypothesis-making. Bottom-up algorithms start with the input sentence (data-driven), and thus tend to be more efficient they do not need to hypothesize about what could be in the sentence. On the other hand top-down methods are simpler to implement, and easier to understand. Systemic grammar, with its realization metaphor, is well-suited to top-down parsing. I implemented various top-down parsers before switching to bottom-up approaches for efficiency reasons. There thus tends to be a trade-off between these strategies: the implementational simplicity of top-down parsers vs. the processing efficiency of bottom-up parsers. For large-size grammars, the lower efficiency of top-down approaches may become prohibitive.

1.3.2.2 Depth-First vs. Breadth-First Strategies

For top-down parsing, depth-first building is the preferred option, since "reaching the input as fast as possible will give the earliest indication of error in hypothesis." For bottom-up parsing, breadth-first is usually preferred, since these approaches wait to see which other words are present, and how they are structurally related, before climbing up a level. Depth-first approaches are more rash, building up towards the sentence level based on just the first word of the sentence. A single word generally provides insufficient evidence to make decisions at higher levels. For either the top-down or bottom-up case, the best option is to see the input text as quickly as possible.

1.3.2.3 Node-Selection Strategies

The head-driven (keyword search) strategy can be more efficient, since it expands those nodes of the parse-tree which are most likely to constrain the structure of neighbouring units. However, this strategy requires the resources to contain a statement of what the key, or head, elements are. Some grammars already contain such information (e.g., dependency grammars). For grammars which do not automatically provide this information, a separate resource would need to be provided.

1.4 PROBLEM DEFINITION

For the reasons stated earlier in this Chapter we have decided to engage ourselves in parsing Hindi sentences. The thesis aims at building An Elementary-GB-Trees Based Parsing system for Hindi 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 Hindi 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 developments to present date in the field 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 an SOV ordered language like Hindi and various phrasal projections for Nouns, Verbs, Adjectives and Adverbs. This chapter also covers grammar rules of Hindi 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 people use and store words, how they learn words, the history and evolution of words, types

of relationships between words as well as how words were created and importance of Lexicon in parsing. Further it explains the different attributes associated with each grammatical category of Hindi. 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 along with its Modules is explained. 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

This chapter is designed to provide a data-motivated, stepwise introduction to the main tenets of the Government and Binding (GB) theory of Syntax, which was developed mainly by Chomsky (1981, 1982, and 1986). The majority of the data used throughout will be in English in order to understand the theory without the hindrance of working with an unfamiliar language at the same time.

2.2 OVERVIEW OF GOVERNMENT AND BINDING THEORY

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 GB view is that Universal Grammar can be broken down into two main components: levels of representation and a system of constraints.

Government and binding is a theory of syntax in the tradition of Transformational Grammar developed principally by Noam Chomsky in the 1980's.[23]. This theory is a radical revision of his earlier theories and was later revised in The Minimalist Program (1995)[24] 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 sub-theories 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.

2.3 CONSTITUENT STRUCTURE AND SUBCATEGORIZATION

A word, such as a noun, verb, adjective or preposition is a lexical category. In structural terms, they are called heads. Phrases are meaningful groupings of words built up from the lexical category of the same type that they contain. Examples of phrases are: NP, VP, and AP (=AdjectiveP), and PP. But the particular head is choosy about what can combine with it to form a phrase.

VP examples:

- 1 died / *died the corpse / *died to Sue about politics
- 2 relied on Max / *relied / *relied from Max / *relied to Max
- 3 dismembered the corpse / *dismembered
- 4 talked (to Sue) (about politics) / *talked that the economy is poor
- 5 read (the book) (to John) / read that the economy is poor
- 6 supplied the Iraqis (with arms) / *supplied
- 7 told Sylvia (that it is raining)
- 8 revealed (to John) that problems exist / revealed the answer (to John)

A **complement** is a phrase that a lexical category takes or selects. Which complements are taken by a particular verb is an arbitrary property of that verb: in (1) *died* cannot take any complements; in (2) *relied* must have a PP complement with *on* as the preposition; in (3) *dismembered* must take an NP complement; in (4) *talked* can take an optional PP complement with *to* as the preposition and/or an optional PP complement where the preposition is *about*, etc.

We can represent these complement selection requirements in subcategorization frames, as shown in (9), where the square brackets delimit the phrase and the environment bar indicates the position of the lexical head. Required complements are simply listed, whereas optional complements are enclosed in parentheses. Finally, in cases where a complement with a particular head is subcategorized for, the head is listed as a feature on the complement (as for *rely* and *talk*).

- (9)
- | | |
|------------|-------------------------------|
| die, | V, [_] |
| rely, | V, [_ PP[on]] |
| dismember, | V, [_ NP] |
| talk, | V, [_ (PP[to]) (PP[about])] |

Adjectives, nouns, and prepositions also subcategorize for their complements.

AP examples:

- 1 red / *red that Sylvia would win
- 2 afraid (of snakes) / *afraid to this issue
- 3 orthogonal (to this issue)
- 4 ambivalent ((to Joe) about her feelings)
- 5 certain (that Sylvia would win)
- 6 insistent (to Joe) (that we leave)

NP examples:

- 1 group (of scientists)
- 2 individual
- 3 book (about photography / *to Fred)
- 4 generosity (to Fred)
- 5 dislike of Fred
- 6 ambivalence ((to John) about my feelings)
- 7 rumor (that all is well)
- 8 message (to the Contras) (about the guns)

PP examples:

- 1 about [the talk]
- 2 before [we leave]
- 3 from [over the hill]
- 4 [looked] up

We can generalize that the lexical categories (V, N, A, P):

- a. Subcategorize for their complements.
- b. Precede their complements in the phrase.²
- c. Co-occur with other constituents.

Heads and complements are not the only parts of phrases. For example, NPs can be preceded by words (or sometimes phrases) like: *the, no, some, every, John's, my mother's*. APs can be preceded by degree words such as: *very, extremely, rather, quite*. These items differ from complements because they precede the lexical category and they are not subcategorized for. They are called **Specifiers**.

2.4 X-bar Theory

GB seeks to capture the similarities between different categories of lexical phrases by assigning the same structure to them. Rather than having different phrase structure rules for VPs, NPs, etc., just the two basic rules as shown below cover all the lexical categories. 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) and 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. 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) |

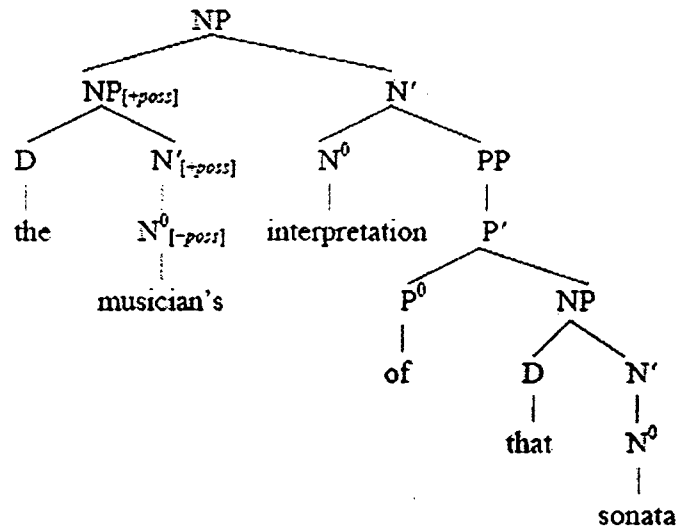


Figure 2.2.

2.4.2 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 2.3.

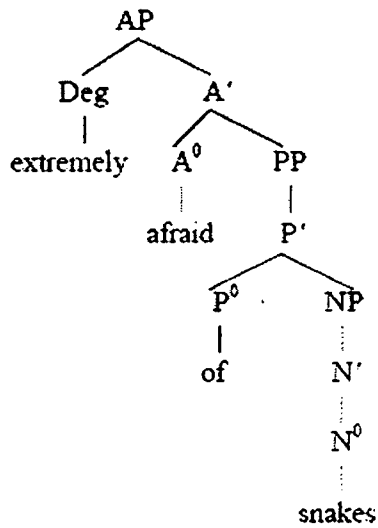


Figure 2.3.

2.4.3 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 about politics' is given in Figure 2.4.

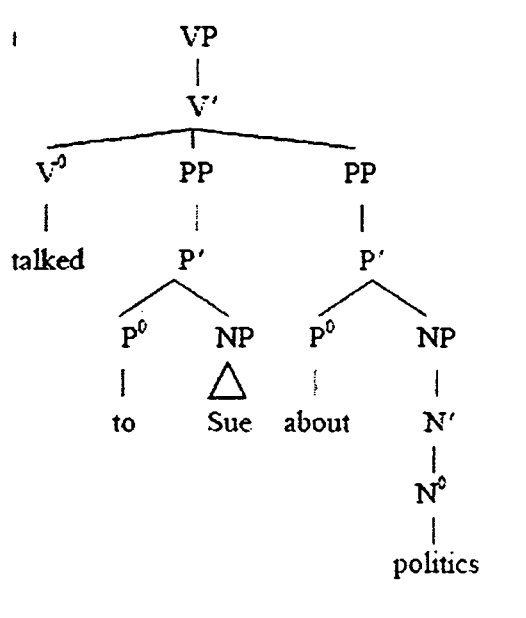


Figure 2.4.

Figure 2.5 illustrates how conjunction and adjunction fit into the X-bar schemata. The conjunction rule is shown for *black* and *white*; *huge*, *black and white*; and *extremely angry* are all adjuncts which are adjoined to N', showing how the adjunction rule is recursive; *the* is in the specifier position and *dog* is the head of the whole NP

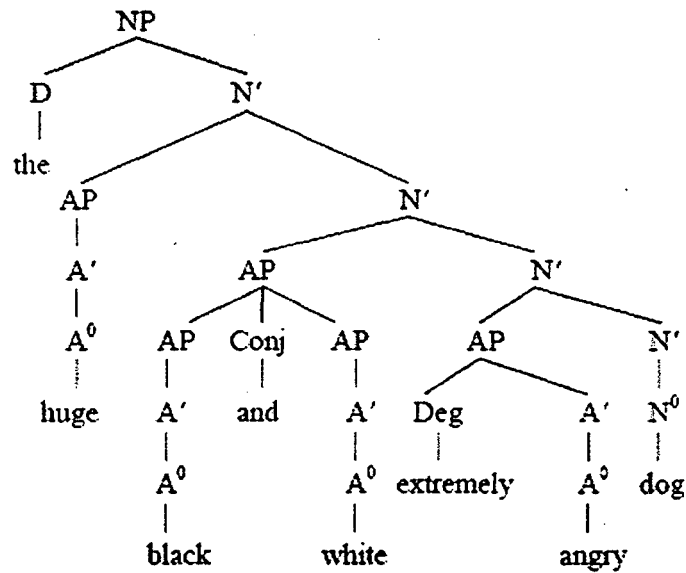


Figure 2.5

At this point, even though we can draw trees for some complex phrases, we still cannot do even a simple complete sentence such as *John hit the ball*. The rule $S \rightarrow NP VP$ does not fit the X-bar schemata. We also cannot draw a tree diagram for a clausal complement to a verb, such as the *that*-clause in *Bill read that the economy is poor*. In order to make sentences and clauses fit X-bar theory, we need to determine the head, specifier, and complement for each. This will be the next topic addressed.

2.5 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.

2.5.1 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.

2.5.2 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.

2.5.3 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**, **θ -roles**). This kind of specification is called **thematic grid**. The θ -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.

	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]

Table 2.1: Lexical Entries with their Subcategorization and Argument structures

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.

2.6 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).



TH-13809

2.6.1 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.

2.6.1.1 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”

2.6.1.2 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.

2.6.1.3 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.

2.6.2 The Theta Criterion

We have seen that the number of arguments that can show up in a sentence is determined by the number of θ -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.*

2.6.3 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 categorical 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.”*

2.6.4 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.

2.7 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.

2.7.1 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 θ -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 θ -role to their subject position.

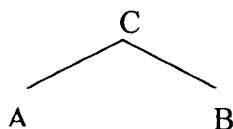
2.7.2 Binding Theory

The Binding Theory captures the distributional properties of overt NPs of these types in terms of their ability or obligation to be co-indexed with other NPs. Binding theory essentially examines the relation between NPs in A-positions (argument positions, i.e. theta- and case-related positions), not with NPs in A'-positions (non-argument positions such as Spec-CP or adjoined positions). Binding is done under c-command and its definition is given below:

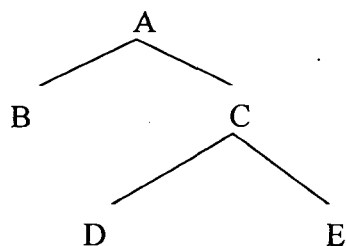
C-Command:

Node A **c-commands** node B if and only if

- 1) A does not dominate B and B does not dominate A; and
- 2) The first branching node dominating A also dominates B.



In the above diagram both A and B c-commands each other.



In the above diagram, B c-commands C, D, and E. C c-commands B, but D and E does not c-command A. Since the first branching node dominating D and E (i.e. C) does not dominate B.

2.8 SUMMARY

This chapter gives an overview of Government and binding theory that forms the base for our work. In Government and Binding Theory, we discuss the x-bar theory, phrasal projection of different grammatical categories of words, which is used to develop elementary trees. It also explains organization of Lexicon which is heart of our whole system which emphasizes on organizing the information of words in database along with categorization and sub categorization information. Theta theory, Case theory and Binding theory are the other aspects that are explained in detail here, which helps to combining elementary trees to develop parse tree for a given sentence.

Chapter 3

GOVERNMENT AND BINDING BASED GRAMMAR FOR HINDI

3.1 INTRODUCTION

Hindi is one of the most widely spoken languages of the world, possessing speakers of the same order of magnitude as those of English and Russian. In India it has been accorded the status of 'official Language' and, along with English, is recognized by the central government for use for most administrative purposes. It is spoken natively by at least 150 million persons in the Indian states of Uttar Pradesh, Madhya Pradesh, and Bihar and as a second language by a like number in other states of North India. It is also an Official language of Uttar Pradesh, Madhya Pradesh, Bihar, Haryana, Rajasthan and Himachal Pradesh, as well as of the Delhi union territory. Urdu, a language so closely related to Hindi to allow some to consider the two to be variants of a single tongue, is spoken by tens of millions, either as a first or second language, both in Pakistan and India. Members of emigrant Indian communities the world over use Hindi as a lingua franca. Hindi enjoys some order of official status in countries as diverse as Fiji, Mauritius, and Guyana.

3.2 THE OVERALL STRUCTURE OF THE HINDI VERBS

As Hindi is SOV ordered language, verb comes last in the sentences. We have categorized the Hindi verbs into three types, intransitive verbs, transitive verbs and the ditransitive verbs. The Hindi verbs are inflected with respect to:

- Gender of the subject (masculine, feminine)
- Number of the subject (singular, plural)
- Tense (present, past, future)
- Aspect (simple, continuous, perfect, and perfect continuous)
- Degree of respect (intimate, familiar, respect)

However, in simple past tense the verb agrees with the object of the sentence. Hindi verbs are often referred to in their infinitive noun form which ends in ना. Examples: बोलना. (to speak). लिखना. (to write), लेना (to take), आना (to come), etc. The root forms are the infinitive form minus the ना ending: बोल, लिख, ले, आ, etc

Hindi verbs and auxiliaries change according to the tense, aspect, number, person and gender. For example, let's take the verb "जा (go)" and see how it changes according to gender:

Male राम घर जाता है ।

Female सीता घर जाती है ।

3.2.1.1 Verb Forms in Simple Present Tense:

The simple present is used for habitual actions. It is formed by adding ता, ते, or ती to the stem of the verb followed by the present tense of होना.

The auxiliary here changes with the number, and the person as shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	singular	में जाता हूँ	I go
female	First	singular	में जाती हूँ	I go
male	First	plural	हम जाते हैं	We go
female	First	plural	हम जाती हैं	We go
male	Second	singular	तू जाता है	You go

female	Second	singular	तू जाती है	You go
male	Second	singular	तुम जाते हो	You go
female	Second	singular	तुम जाती हो	You go
male	Second	plural	तुम जाते हो	You go
female	Second	plural	तुम जाती हो	You go
male	Second	singular	आप जाते हैं	You go
female	Second	singular	आप जाती हैं	You go
male	Second	plural	आप जाते हैं	You go
female	Second	plural	आप जाती हैं	You go
male	Third	singular	वह जाता है	He goes
female	Third	singular	वह जाती है	She goes
male	Third	plural	वे जाते हैं	They go
female	Third	plural	वे जाती हैं	They go

3.2.1.2 Verb Forms in Present Continuous:

The present continuous is used express the idea that something is happening now, at this very moment or simple for on going actions. It is formed according to the rule given:

Root + रहा/ रहे/ रही + present tense of होना.

The auxiliary here changes with the number, and the person as shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	singular	मैं जा रहा हूँ	I am going

female	First	singular	में जा रही हूँ	I am going
male	First	plural	हम जा रहे हैं	We are going
female	First	plural	हम जा रही हैं	We are going
male	second	singular	तू जा रहा है	You are going
female	second	singular	तू जा रही है	You are going
male	second	singular	तुम जा रहे हो	You are going
female	second	singular	तुम जा रही हो	You are going
male	second	plural	तुम जा रहे हो	You are going
female	second	plural	तुम जा रही हो	You are going
male	second	singular	आप जा रहे हैं	You are going
female	second	singular	आप जा रही हैं	You are going
male	second	plural	आप जा रहे हैं	You are going
female	second	plural	आप जा रही हैं	You are going
male	Third	singular	वह जा रहा है	He is going
female	Third	singular	वह जा रही है	She is going
male	Third	plural	वे जा रहे हैं	They are going
female	Third	plural	वे जा रही हैं	They are going

3.2.1.3 Verb Forms in Present Perfect

The present perfect is the actions which have been just completed. In Hindi those are formed by the rule: Root + चुका /चुके/चुकी + the present tense of होना

The changes in the auxiliary with the number, and the person are shown below.

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	singular	मैं जा चुका हूँ	I have gone
female	First	singular	मैं जा चुकी हूँ	I have gone
male	First	plural	हम जा चुके हैं	We have gone
female	First	plural	हम जा चुकी हैं	We have gone
male	Second	singular	तू जा चुका है	You have gone
female	Second	singular	तू जा चुकी है	You have gone
male	Second	singular	तुम जा चुके हो	You have gone
female	Second	singular	तुम जा चुकी हो	You have gone
male	Second	plural	तुम जा चुके हो	You have gone
female	Second	plural	तुम जा चुकी हो	You have gone
male	Second	singular	आप जा चुके हैं	You have gone
male	Second	singular	आप जा चुकी हैं	You have gone
male	Second	plural	आप जा चुके हैं	You have gone
female	Second	plural	आप जा चुकी हैं	You have gone
male	Third	singular	वह जा चुका है	He has gone
female	Third	singular	वह जा चुकी है	She has gone
male	Third	plural	वे जा चुके हैं	They have gone
female	Third	plural	वे जा चुकी हैं	They have gone

3.2.1.4 Verb Forms in Present Perfect Continuous

We use the Present Perfect Continuous to show that something started in the past and has continued up until now. In Hindi this is formed by the below rule:

Simple Present form of root + रहा/ रहे/ रही + present tense of होना

The changes in the auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
Male	First	Singular	मैं जाता रहा हूँ	I have been going
female	First	singular	मैं जाती रही हूँ	I have been going
male	First	plural	हम जाते रहे हैं	We have been going
female	First	plural	हम जाती रही हैं	We have been going
male	Second	singular	तू जाता रहा है	You have been going
female	Second	singular	तू जाती रही है	You have been going
male	Second	singular	तुम जाते रहे हो	You have been going
female	Second	singular	तुम जाती रही हो	You have been going
male	second	plural	तुम जाते रहे हो	You have been going
female	second	plural	तुम जाती रही हो	You have been going
male	Second	singular	आप जाते रहे हैं	You have been going
female	Second	singular	आप जाती रही हैं	You have been going
male	Second	plural	आप जाते रहे हैं	You have been going
female	Second	plural	आप जाती रही हैं	You have been going
male	Third	singular	वह जाता रहा है	He has been going

female	Third	singular	वह जाती रही है	She has been going
male	Third	plural	वे जाते रहे हैं	They have been going
female	Third	plural	वे जाती रही हैं	They have been going

3.2.1.5 Verb Forms in Simple Past

The simple past is used for actions in the past. In this form the verb takes its past tense form and is mostly irregular. Here auxiliary is absent.

The changes in the verb with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
Male	First	singular	मैं गया	I went
Female	First	singular	मैं गयी	I went
Male	First	plural	हम गये	We went
female	First	plural	हम गयीं	We went
Male	Second	singular	तू गया	You went
female	Second	singular	तू गयी	You went
male	Second	singular	तुम गये	You went
female	Second	singular	तुम गयीं	You went
male	Second	plural	तुम गये	You went
female	Second	plural	तुम गयीं	You went
male	Second	singular	आप गये	You went
female	Second	singular	आप गयीं	You went
male	Second	plural	आप गये	You went

female	Second	plural	आप गयीं	You went
male	Third	singular	वह गया	He went
female	Third	singular	वह गयी	She went
male	Third	plural	वे गये	They went
female	Third	plural	वे गयीं	They went

3.2.1.6 Verb Forms in Past Continuous

The past continuous is used for ongoing actions in the past -- like the "-ing" form in English. It is formed by the rule: Root + रहा/ रहे/ रही + past tense of होना. The changes in the verb and auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
Male	First	Singular	मैं जा रहा था	I was going
female	First	Singular	मैं जा रही थी	I was going
male	First	Plural	हम जा रहे थे	We were going
female	First	Plural	हम जा रही थीं	We were going
male	Second	Singular	तू जा रहा था	You were going
female	Second	Singular	तू जा रही थी	You were going
male	Second	Singular	तुम जा रहे थे	You were going
female	Second	Singular	तुम जा रही थीं	You were going
male	Second	Plural	तुम जा रहे थे	You were going

female	Second	Plural	तुम जा रही थीं	You were going
male	Second	Singular	आप जा रहे थे	You were going
female	Second	Singular	आप जा रही थीं	You were going
male	Second	Plural	आप जा रहे थे	You were going
male	Second	Plural	आप जा रही थीं	You were going
male	Third	Singular	वह जा रहा था	He was going
female	Third	Singular	वह जा रही थी	She was going
male	Third	Plural	वे जा रहे थे	They were going
female	Third	Plural	वे जा रही थीं	They were going

3.2.1.7 Verb form in Past Perfect

The past perfect describes an action which is completed before a certain moment in the past. In Hindi the past perfect is formed like this: root + चुका/चुके + past form of होना.

The changes in the verb and auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	Singular	मैं जा चुका था	I had gone
female	First	Singular	मैं जा चुकी थी	I had gone
male	First	Plural	हम जा चुके थे	We had gone
female	First	Plural	हम जा चुकी थीं	We had gone
male	Second	Singular	तू जा चुका था	You had gone
female	Second	Singular	तू जा चुकी थी	You had gone

male	Second	Singular	तुम जा चुके थे	You had gone
female	Second	Singular	तुम जा चुकी थीं	You had gone
male	Second	Plural	तुम जा चुके थे	You had gone
female	Second	Plural	तुम जा चुकी थीं	You had gone
male	Second	Singular	आप जा चुके थे	You had gone
female	Second	Singular	आप जा चुकी थीं	You had gone
male	Second	Plural	आप जा चुकी थीं	You had gone
female	Second	Plural	आप जा चुकी थीं	You had gone
male	Third	Singular	वह जा चुका था	He had gone
female	Third	Singular	वह जा चुकी थी	She had gone
male	Third	Plural	वे जा चुके थे	They had gone
female	Third	Plural	वे जा चुकी थीं	They had gone

3.2.1.8 Verb forms in Past perfect Continuous

Past perfect continuous is used for an action that began before a certain point in past and continued up to that time. It is formed like this: simple past form of root + रहा/रहे + past form of होना.

The changes in the auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
Male	First	Singular	मैं जाता रहा था	I had been going
Female	First	singular	मैं जाती रही थी	I had been going

Male	First	plural	हम जाते रहे थे	We had been going
Female	First	plural	हम जाती रही थीं	We had been going
Male	Second	singular	तू जाता रहा था	You had been going
Female	Second	singular	तू जाती रही थी	You had been going
male	Second	singular	तुम जाते रहे थे	You had been going
female	Second	singular	तुम जाती रही थीं	You had been going
male	Second	plural	तुम जाते रहे थे	You had been going
female	Second	plural	तुम जाती रही थीं	You had been going
male	Second	singular	आप जाते रहे थे	You had been going
female	Second	singular	आप जाती रही थीं	You had been going
male	Second	plural	आप जाते रहे थे	You had been going
female	Second	plural	आप जाती रही थीं	You had been going
male	Third	singular	वह जाता रहा था	He had been going
female	Third	singular	वह जाती रही थी	She had been going
male	Third	plural	वे जाते रहे थे	They had been going
Female	Third	plural	वे जाती रही थीं	They had been going

3.2.1.9 Verb forms in Simple Future

The simple future is used to refer to the future as well as to make assumptions about the presents (just like in English). It is formed by adding ऊगा / ऊगी, ऐगा/ऐगी, ऐगा/ऐगी, or ओगा/ओगी to the root for or the verb.

The changes in the auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	Singular	में जाऊँगा	I will go
female	First	Singular	में जाऊँगी	I will go
male	First	Plural	हम जाएँगे	We will go
female	First	Plural	हम जाएँगी	We will go
male	Second	Singular	तू जाएगा	You will go
female	Second	Singular	तू जाएगी	You will go
male	Second	Singular	तुम जाओगे	You will go
female	Second	Singular	तुम जाओगी	You will go
male	Second	Plural	तुम जाओगे	You will go
female	Second	Plural	तुम जाओगी	You will go
male	Second	Singular	आप जाओगे	You will go
female	Second	Singular	आप जाएँगी	You will go
male	Second	Plural	आप जाओगे	You will go
female	Second	Plural	आप जाएँगी	You will go
male	Third	Singular	वह जाएगा	He will go
female	Third	Singular	वह जाएगी	She will go
male	Third	Plural	वे जाएँगे	They will go
female	Third	Plural	वे जाएँगी	They will go

3.2.1.10 Verb forms in Future Continuous

The future continuous is used to refer to ongoing actions in the future. It is formed as the present simple and the future form of होना.

The changes in the auxiliary with the number, and the person here are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	Singular	में जा रहा हूँगा	I will be going
female	First	Singular	में जा रही हूँगी	I will be going
male	First	Plural	हम जा रहे होंगे	We will be going
female	First	Plural	हम जा रही होंगी	We will be going
male	Second	Singular	तू जा रहा होगा	You will be going
female	Second	Singular	तू जा रही होगी	You will be going
male	Second	Singular	तुम जा रहे होंगे	You will be going
female	Second	Singular	तुम जा रही होगी	You will be going
male	Second	Plural	तुम जा रहे होंगे	You will be going
female	Second	Plural	तुम जा रही होगी	You will be going
male	Second	Singular	आप जा रहे होंगे	You will be going
female	Second	Singular	आप जा रही होगी	You will be going
male	Second	Plural	आप जा रहे होंगे	You will be going
female	Second	Plural	आप जा रही होगी	You will be going
male	Third	Singular	वह जा रहा होगा	He will be going
female	Third	Singular	वह जा रही होगी	She will be going

male	Third	Plural	वे जा रहे होंगे	They will be going
female	Third	Plural	वे जा रही होंगी	They will be going

3.2.1.11 Verb forms in Future Perfect

The Future Perfect tense is used to talk about actions that will be completed by a certain future time. The future perfect is formed as: Root + चुका /चुके/चुकी + the future tense of होना to the root.

The changes in the auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	First	Singular	मैं जा चुका हूँगा	I would have gone
female	First	Singular	मैं जा चुकी हूँगी	I would have gone
male	First	Plural	हम जा चुके होंगे	We would have gone
female	First	Plural	हम जा चुकी होंगी	We would have gone
male	Second	Singular	तू जा चुका होगा	You would have gone
female	Second	Singular	तू जा चुकी होगी	You would have gone
male	Second	Singular	तुम जा चुके होंगे	You would have gone
female	Second	Singular	तुम जा चुकी होगी	You would have gone
male	Second	Plural	तुम जा चुके होंगे	You would have gone
female	Second	Plural	तुम जा चुकी होंगी	You would have gone
male	Second	Singular	आप जा चुके होंगे	You would have gone
female	Second	Singular	आप जा चुकी होंगी	You would have gone

male	Second	Plural	आप जा चुके होंगे	You would have gone
female	Second	Plural	आप जा चुकी होंगी	You would have gone
male	Third	Singular	वह जा चुका होगा	He would have gone
female	Third	Singular	वह जा चुकी होगी	She would have gone
male	Third	Plural	वे जा चुके होंगे	They would have gone
female	Third	Plural	वे जा चुकी होंगी	They would have gone

3.2.1.12 Future Perfect Continuous:

The Future Perfect Continuous tense is used for actions which will be in progress over a period of time that will end in the future. In Hindi this is formed by the below rule:

Simple Present form of root + रहा/ रहे/ रही + future of होना

The changes in the auxiliary with the number, and the person are shown below:

Gender	Person	Number	Hindi Sentence	English Sentence
male	first	Singular	मैं जाता रहा हूँगा	I would have been going
female	first	Singular	मैं जाती रही हूँगी	I would have been going
male	first	Plural	हम जाते रहे होंगे	We would have been going
female	first	Plural	हम जाती रही होंगी	We would have been going
male	second	Singular	तू जाता रहा होगा	You would have been going
female	second	Singular	तू जाती रही होगी	You would have been going
male	second	Singular	तुम जाते रहे होंगे	You would have been going
female	second	Singular	तुम जाती रही होगी	You would have been going

male	second	Plural	तुम जाते रहे होंगे	You would have been going
female	second	Plural	तुम जाती रही होंगी	You would have been going
male	second	Singular	आप जाते रहे होंगे	You would have been going
female	second	Singular	आप जाती रही होंगी	You would have been going
male	second	Plural	आप जाते रहे होंगे	You would have been going
female	second	Plural	आप जाती रही होंगी	You would have been going
male	third	Singular	वह जाता रहा होगा	He would have been going
female	third	Singular	वह जाती रही होंगी	She would have been going
male	third	Plural	वे जाते रहे होंगे	They would have been going
female	third	Plural	वे जाती रही होंगी	They would have been going

3.2.2 The Overall Structure of the Hindi Nouns

Hindi nouns have two genders, masculine and feminine and two numbers, singular and plural. It has eight cases which are indicated by the presence of postpositions that immediately follow the nouns. Postpositions are analogous to the prepositions of English. Hindi nominal forms are classified as direct or oblique. Hindi nouns followed by postpositions are said to be in their oblique forms. Otherwise they are said to be in their direct form..

3.2.2.1 Masculine nouns: There are two classes of masculine nouns in Hindi, class I and class II. Class I nouns end in आ in their singular direct, ए in their singular oblique and in plural direct, ओ in their plural oblique form as shown in Table 3.1a and Table 3.1b.

Class I	Singular	Plural
Direct	लड़का	लड़के
oblique	लड़के	लड़कों

Table 3.1a

Class I	Singular	Plural
Direct	कुआँ	कुएँ
oblique	कुएँ	कुओँ

Table 3.1b

Masculine class II nouns have no distinct endings in their direct singular, direct plurals and oblique singular forms. They add ओँ to form the plural oblique. See Table 3.2.

Class II	Singular	Plural
Direct	घर	घर
oblique	घर	घरों

Table 3.2

Masculine class II nouns ending with ई shorten this vowel to इ and insert a य before the oblique plural termination ओँ See Table 3.3

Class II	Singular	Plural
Direct	आदमी	आदमी
oblique	आदमी	आदमियों

Table 3.3

Masculine class II nouns ending in ऊ shorten this vowel to उ before the oblique plural termination. See Table 3.3.

Class II	Singular	Plural
Direct	चाकू	चाकू
Oblique	चाकू	चाकुओ

Table 3.4

Few of the masculine nouns ending in आ are declined according to the pattern of class II as given in Table 3.5.

Class II	Singular	Plural
Direct	राजा	राजा
Oblique	राजा	राजाओ

Table 3.5

3.2.2.2 Feminine Nouns: Feminine nouns also possess two classes, class I and class II. Class I nouns end in ई in their singular direct, ई in their singular oblique, इयाँ in their plural direct, and इयों in their oblique plural forms. See Table 3.6.

Class I	Singular	Plural
Direct	लडकी	लडकियाँ
oblique	लडकी	लडकियों

Table 3.6

Few class I feminine nouns show forms similar to लड़की, except that इ or इया appear in place of ई in the singular forms as shown in Table 3.7.

Class I	Singular	Plural
Direct	चिड़िया	चिड़ियाँ
oblique	चिड़िया	चिड़ियों

Table 3.7

The feminine class II nouns form their plural direct forms by means of the suffix एँ and their plural oblique with ओँ. The singular forms, both direct and oblique, may end in virtually any sound, excepting ई, इ and इया which are characteristic of class I feminine nouns. See Table 3.8.

Class II	Singular	Plural
Direct	पुस्तक	पुस्तकें
	वस्तु	वस्तुएँ
	माता	माताएँ
oblique	पुस्तक	पुस्तकों
	वस्तु	वस्तुओँ
	माता	माताओँ

Table 3.8

Class II nouns having singular direct forms in ऊ; shorten this vowel to उ before the plural direct termination ऐ and the plural oblique termination ओ See Table 3.9.

Class II	Singular	Plural
Direct	बहू	बहुऐँ
oblique	बहू	बहुओँ

Table 3.9

3.2.3 Adjective forms for Hindi

Hindi adjectives are of two kinds, declinable and indeclinable. Declinable adjectives agree with the nouns they modify in gender (masculine and feminine), number (singular and plural), and case (direct and oblique).

The masculine forms of declinable adjectives end in आँ in the singular direct, and ए in the singular oblique, plural direct and plural oblique cases. See Table 3.10.

Masculine	Singular	Plural
Direct	खड़ा लड़का	खड़े लड़के
oblique	खड़े लड़के	खड़े लड़कों

Table 3.10

Declinable objectives always show ई when modifying feminine nouns, whether singular or plural, direct or oblique. See Table 3.11.

Feminine	Singular	Plural
Direct	खड़ी लड़की	खड़ी लड़कियाँ
oblique	खड़ी लड़की	खड़ी लड़कियों

Table 3.11

Indeclinable adjectives possess a single form when modifying nouns of different genders, numbers or cases. These adjectives do not end in any characteristic sounds.

Hindi adjectives may be used either predicatively (to make a statement about nominal entity) or attributively (to restrict the meaning of nominal entity). See the following examples.

Predicative: 'The man is standing there.'

आदमी वहाँ खड़ा है।

Attributive: 'The standing man is madan.'

खड़ा आदमी मदन है।

3.2.4 Postposition Forms for Hindi

Indirect object, etc. are indicated by a class of words called Post positions. These words are similar in function to English prepositions, but stand after the nouns with which they are linked. Hindi postpositions are either simple or compound. Some important Simple Postpositions are:

में, पर, तक, से, को, का,

Vast majority of Hindi Postpositions are compound and consists of two or more words.

के लिए, के बाद, के बारे में

3.2.5 Adverb Forms for Hindi

Adverbs can be defined as words that qualify or modify verbs, adjectives, or other adverbs. Hindi adverbs can be divided into adverbs of time, place, manner, degree, affirmation or negation. Adverbs of time specify the time at which a verbal activity or state of affairs takes place. Some common adverbs of time are

कल, आज, तबी, सुबह

Adverbs of place specify the location at which verbal actions or states of affairs take place.

यहाँ, इधर, वहा से

Adverbs of manner specify the manner in which some activity is carried out.

जल्दी, देर से, मुश्किल से

Adverbs of degree specify the extent to which some adjectival quality pertains to qualified noun.

बहुत

3.3 HINDI PHRASE STRUCTURES

We can refer to Hindi and other SOV languages are **head-final** and **specifier-initial languages**. Since the specifier comes before X' and the head comes after its complements. This generalization holds in all phrases in Hindi. The various Phrase Structures for English have been described in detail in Chapter 2. Now in this section, we are giving various Phrase Structures for Hindi. The rules forms of Hindi Phrase Structures are given in Appendix B. The basic phrase structure rules are:

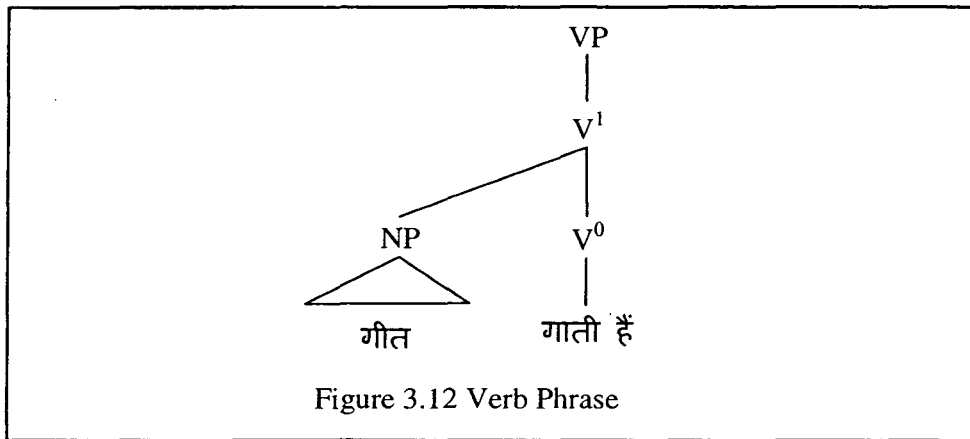
$XP \rightarrow \text{Specifier } X'$ -- (1) (The Specifier Rule)

$X' \rightarrow \text{Complements } X^0$ -- (2) (The Adjunct Rule)

$X' \rightarrow \text{Adjuncts } X'$ -- (3) (The Complement Rule)

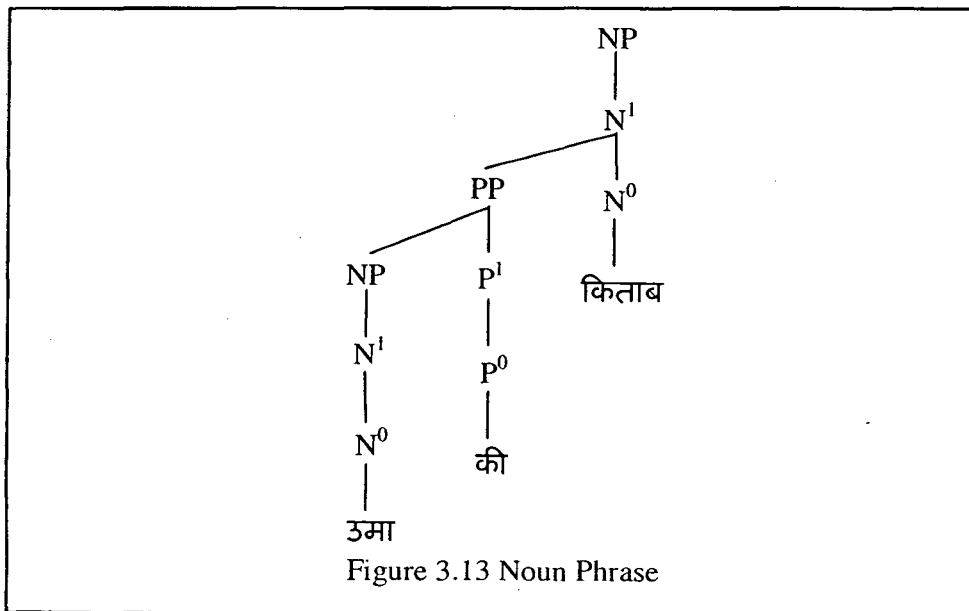
3.3.1 A Verb Phrase

The tree representation of Verb Phrase “गीत गाती हूँ” is shown in Figure 3.12.



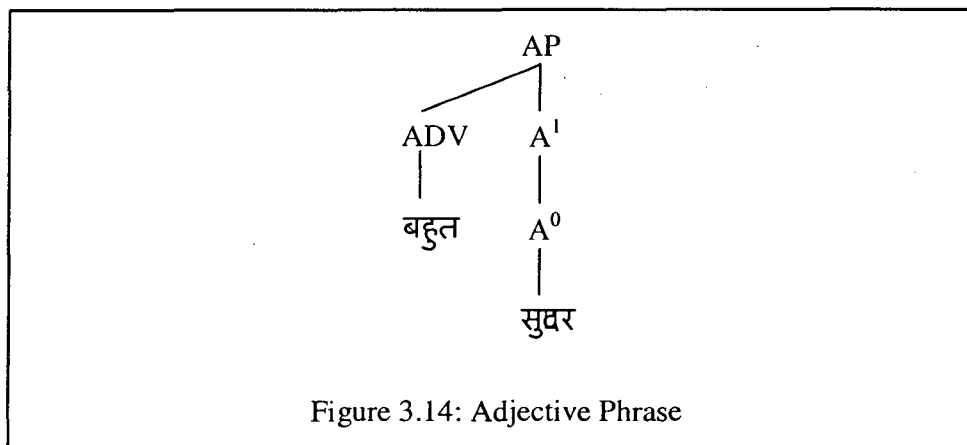
3.3.2 A Noun Phrase

The tree representation for a Noun Phrase “उमा की किताब” is shown in Figure 3.13.



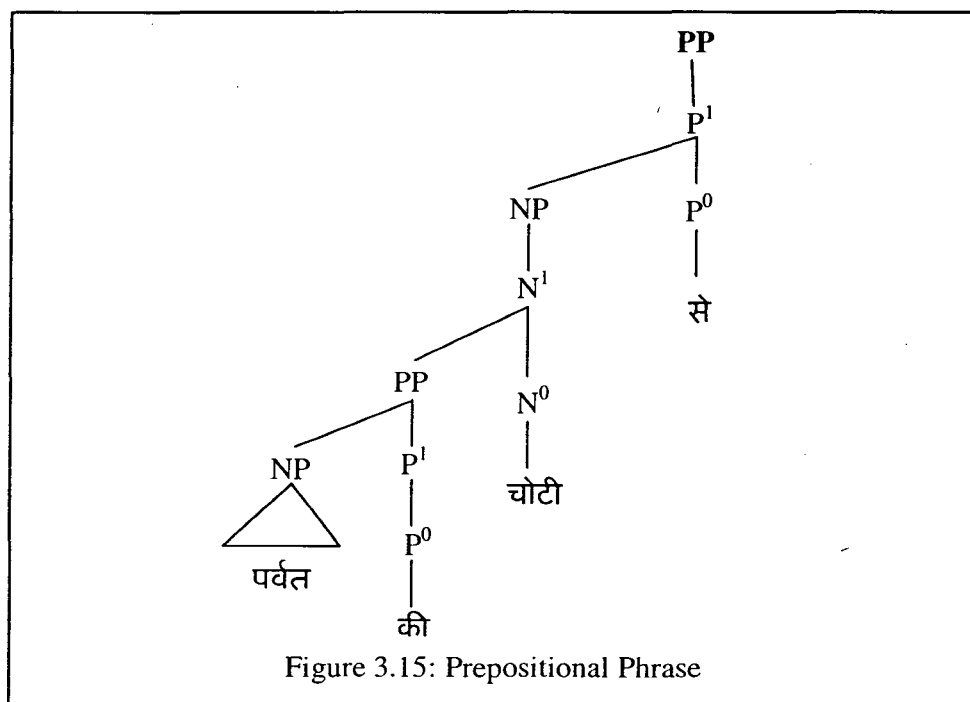
3.3.3 An Adjective Phrase

The tree representation for an Adjective Phrase “बहुत सुंदर” is shown in Figure 3.14.



3.3.4 A Prepositional Phrase

The tree representation for the Prepositional Phrase “पर्वत की चोटी से” is shown in Figure 3.15.



3.3.5 An Inflection Phrase

The tree representation for an Inflection Phrase “राम सुघर कपो पहनता ह” is shown in Figure 3.16.

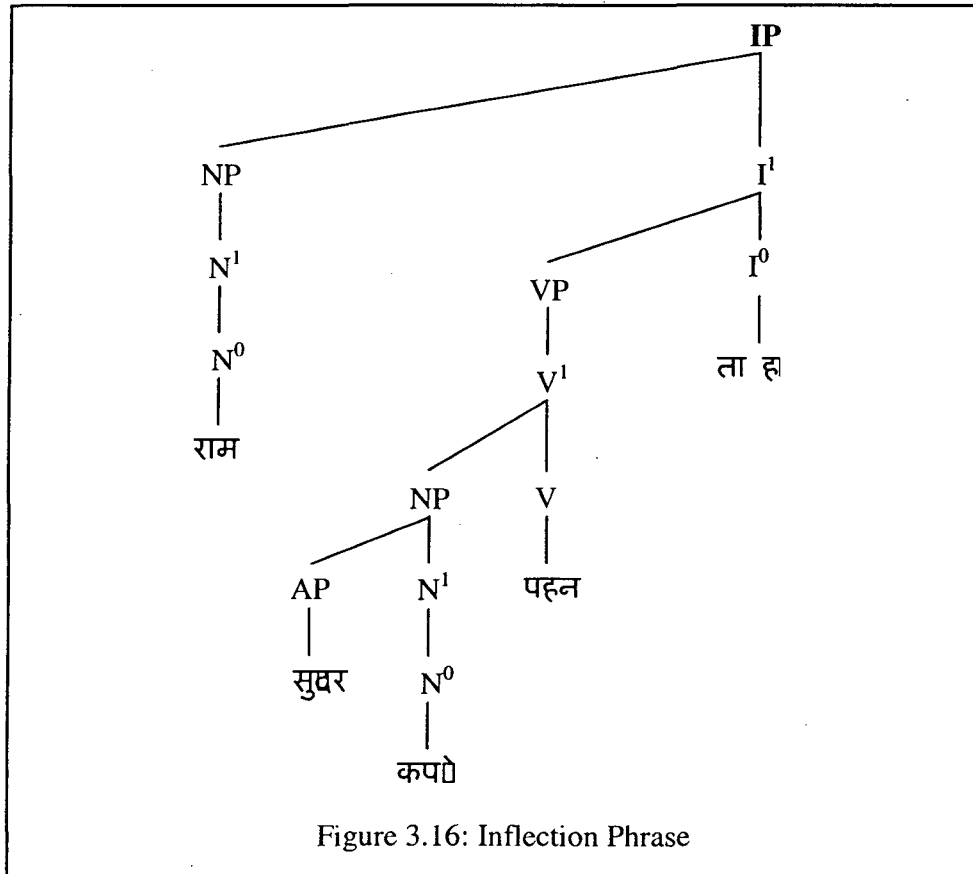


Figure 3.16: Inflection Phrase

3.3.6 A Complementizer Phrase

The tree representation for a Complementizer Phrase is shown in Figure 3.17.

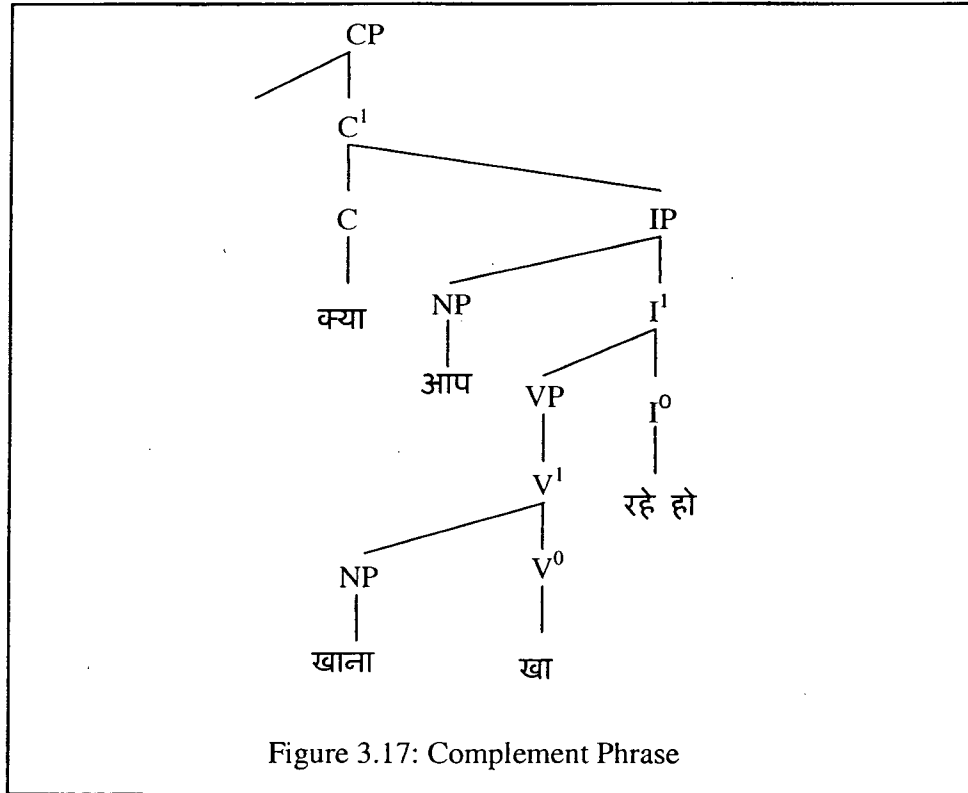


Figure 3.17: Complement Phrase

3.4 SUMMARY

This chapter briefly explains how Government and Binding rules are used for an SOV ordered language like Hindi and various phrasal projections for Nouns, Verbs, Adjectives and Adverbs. This chapter also covers grammar rules of Hindi 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

Lexicon

4.1 INTRODUCTION

Lexicon is usually a list of words together with additional word-specific information. *Lexicon* is a word of Greek origin, meaning vocabulary. When linguists study the lexicon, they study such things as what words are, how the vocabulary in a language is structured, how people use and store words, how they learn words, the history and evolution of words, types of relationships between words as well as how words were created. The term is also sometimes used in the title of an encyclopedic dictionary or an encyclopedia, especially for 19th century works and those written in German (*lexikon*).

In linguistics, *lexicon* has a slightly more specialized definition, as it includes the lexemes used to actualize words. Lexemes are formed according to morpho-syntactic rules and express sememes. In this sense, a lexicon organizes the mental vocabulary in a speaker's mind: First, it organizes the vocabulary of a language according to certain principles (for instance, all verbs of motion may be linked in a lexical network) and second, it contains a generative device producing (new) simple and complex words according to certain lexical rules. For example, the suffix '-able' can be added to transitive verbs only such that we get 'read-able' but not '*cry-able'. (Though exceptions exist to this rule: one can certainly imagine a 'sleepable mattress' or the expression, 'Sure, that's workable.

4.2 STRUCTURE OF HINDI LEXICON

4.2.1 Introduction

The vocabulary of Modern Standard Hindi is both rich and diverse. It draws from the vast lexical resources of Sanskrit, Arabic, Persian, Turkish, Portuguese, English, and Other languages with which Hindi has come into contact. Indian grammarians have found it useful to classify some of the different types of vocabulary items that coexist in the language. Those words that are borrowed directly from Sanskrit with little or no phonetic alteration are classified as tatsama: e.g., paksi, jal, karya, agni. Items that are ultimately of Sanskrit origin but that have undergone continual phonetic change in the course of their historical evolution are designated as tadbhava e.g., ag, sab. The Indian Grammarians also recognize a class of vocabulary items intermediate between tatsama and tadbhava forms. These words categorized as arddha-tatsama are direct borrowings from Sanskrit but show degree of phonetic modification e.g., agni, saicar. The tatsama, arddha-tatsama, and tadbhava vocabularies of Hindi are Historically Indo-Aryan, owing their origins to Sanskrit in one way or another.

Hindi Lexicon contains information of each and every word as per the word type we classify these words into different categories. Hindi words are mainly classified as Nouns, Pronouns, Verbs, and Auxiliary verbs, Adverbs, Postpositions and Adjective as per the grammar. In the database we had maintained different table for each of the above categories. Along with these tables we had subcategory information tables, which provide the sub-category information for each of the words which were in the main table. The sub categorization tables were designed in such a manner that redundancy doesn't exist.

4.2.1.1 Noun Information

Mainly the nouns in Hindi have following attributes

- Number
- Gender
- Nominal form

All the information regarding nouns is stored in a different table.

Word	Number	Gender	Nominal form
लडका	Singular	Male	Direct
घरों	Plural	Male	Oblique
लडकियाँ	Plural	feminine	Direct

Table 4.1

4.2.1.2 Pronoun Information

Pronouns in Hindi have the following attributes.

- Number
- Gender
- Personal
- Nominal form

All the information regarding Pronouns is stored in a different table.

Word	Number	Gender	Person	Nominal form
मैं	Singular	Both	First	Direct
तुम	Plural	Both	Second	Direct
वह	Singular	Both	Third	Direct
कुछ	Both	Both	Third	Both

Table 4.2

4.2.1.3 Verb Information

Verbs in Hindi have the following attributes.

- Tense
- Aspect
- Number

- Gender

All the information regarding primary verbs is stored in a different table

Word	Tense	Aspect	Number	Gender
जा	Present	Simple	Both	Both
गया	Past	Simple	Singular	Masculine
गयी	Past	Simple	Plural	Feminine

Table 4.3

4.2.1.4 Auxiliary Verb Information

Auxiliary Verbs in Hindi have the following attributes.

- Tense
- Aspect
- Number
- Gender

All the information regarding Auxiliary verbs is stored in a different table

Word	Tense	Aspect	Number	Gender
ता हू	Present	Simple	Singular	Male
रहा था	Past	Continuous	Singular	Male
चुकी थी	Past	Perfect	plural	Female

Table 4.4

4.2.1.5 Post position Information

Post positions in Hindi have the following attributes.

- Number
- Gender

All the information regarding Post positions is stored in a different table

Word	Number	Gender
का	Singular	Male
की	Both	Female
के	Both	Male

Table 4.5

4.2.1.6 Adverb Information

Adverbs in Hindi have the following attributes.

- Number
- Gender

All the information regarding Adverbs is stored in a different table

Word	Number	Gender
कल	Both	Male
आज	Both	Male
देर से	Both	Both

Table 4.6

4.2.1.7 Adjective Information

Adjectives in Hindi have the following attributes.

- Number
- Gender

All the information regarding Adjectives is stored in a different table

Word	Number	Gender
खडा	Singular	Male
खड़ी	Both	Feminine
खड़े	Both	Masculine

Table 4.7

4.3 DESIGN OF LEXICON

The tables listed above explain different categories of words and their concerned attributes that plays a major role in parsing the sentence. Current section discusses with the organization of lexicon for Hindi. After the through study of the Hindi 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.

4.3.1 Word table

In the word table we have following fields.

- Word
- category
- Category index
- Sub-category index

Word	Category	Category index	Sub-category index
------	----------	----------------	--------------------

Table 4.8

4.3.2 Subcategory table

We have the following fields in subcategory table

- Sub-category index
- No of complements
- 1st complement
- 2nd complement
- 3rd complement

Sub-category index	No of complements	1 st complement	2 nd complement	3 rd complement
--------------------	-------------------	----------------------------	----------------------------	----------------------------

Table 4.9

4.3.3 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.

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.

The screenshot shows a software window titled "Hindi Lexicon Data Entry". The interface includes the following elements:

- Word:** A text input field.
- Category:** A dropdown menu.
- Attributes:** A section containing nine dropdown menus arranged in two columns:
 - Row 1: Number, Gender, Person
 - Row 2: NominalForm, Mood
 - Row 3: Tense, Aspect
 - Row 4: Voice, AdverbCase
- SubCategory:** A dropdown menu labeled "NoOfSubCategory".
- Complement1, Complement2, Complement3:** Three dropdown menus.
- Save:** A button located at the bottom right of the window.

Fig 4.1 Screenshot of Hindi Lexicon Data Entry

4.4 SUMMARY

This chapter defines lexicon, how the vocabulary in a language is structured, how people use and store words, how they learn words, the history and evolution of words, types of relationships between words as well as how words were created and importance of Lexicon in parsing. Further it explains the different attributes associated with each grammatical category of Hindi. The last section explains the use of interface for storing the words in the 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.

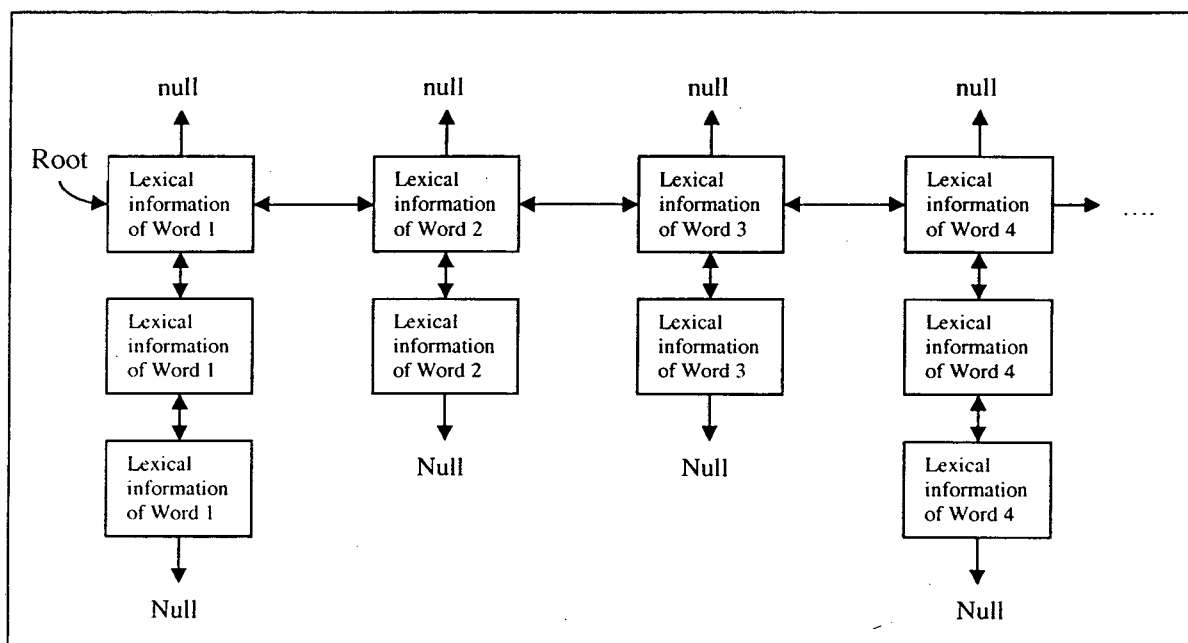


Fig. 5.2 Linked List Structure for sentence

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 be 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 Hindi 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.

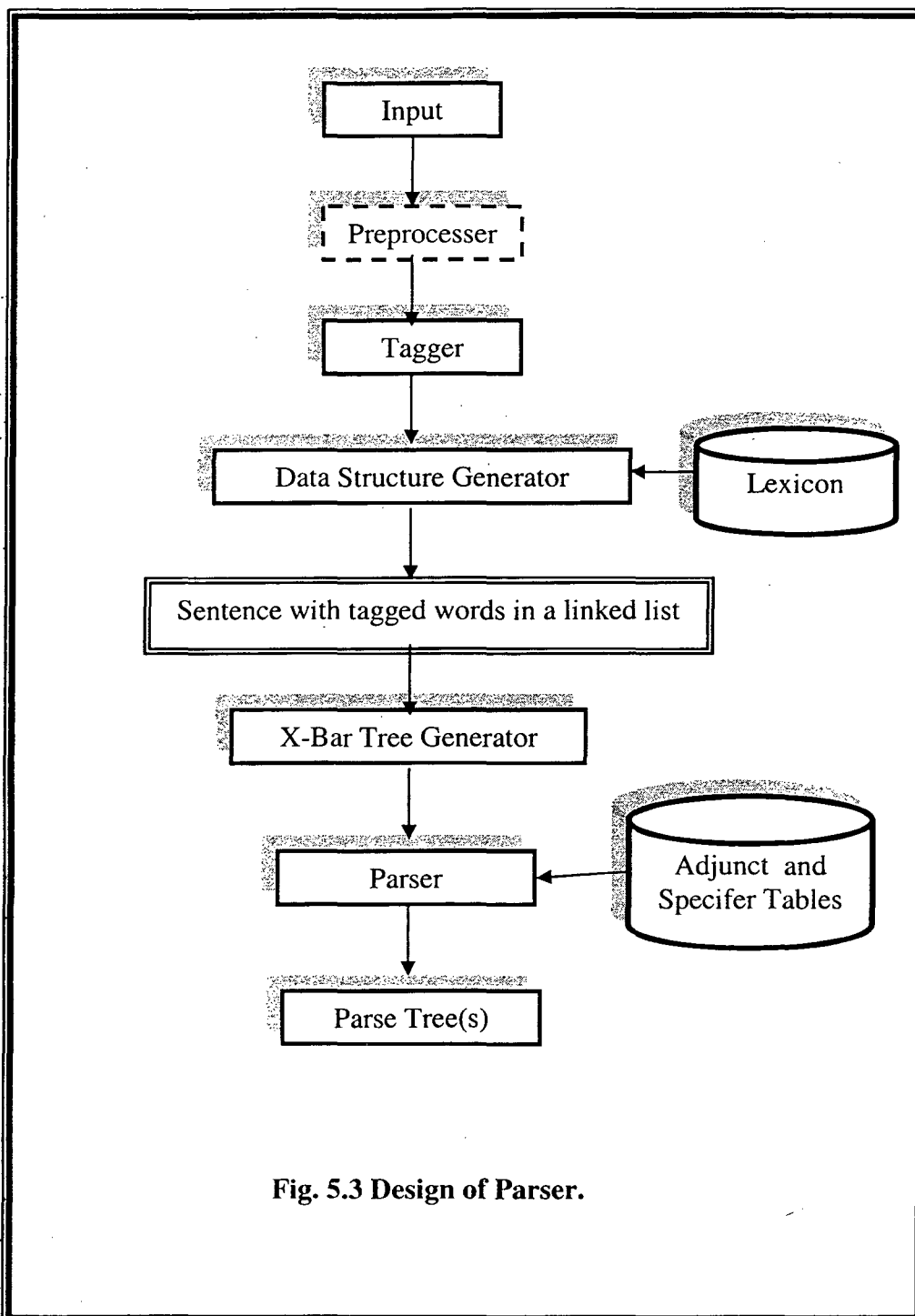


Fig. 5.3 Design of Parser.

REPRESENTATIONS:

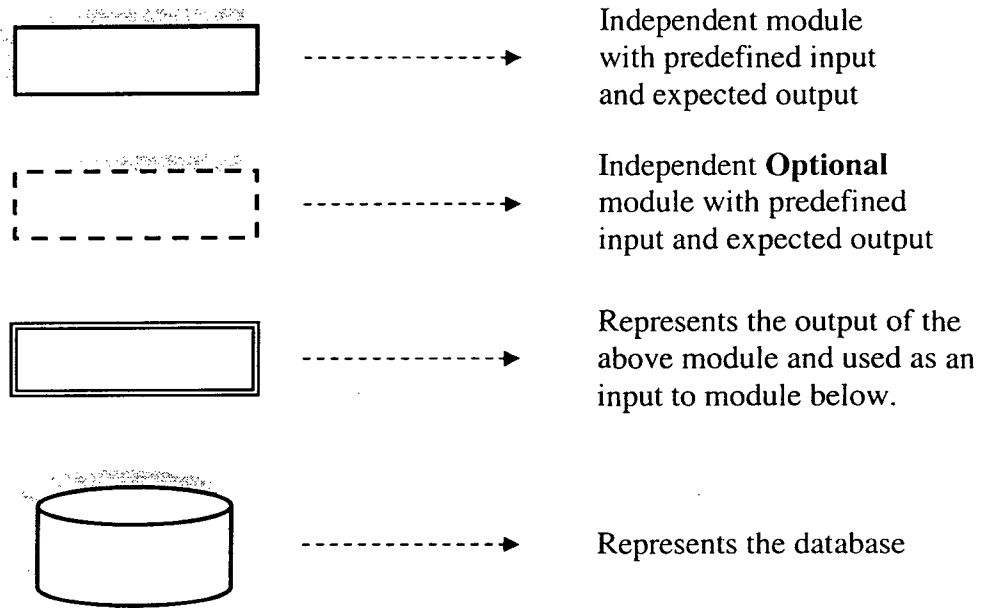


Figure 5.3

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 lexicon, which depends solely on word.

5.2.2.5 Data Structure Generator

In run time, all the words are stored in specially designed data structure along with all its attributes. This data structure basically a doubly linked list, where every node has a pointer to previous and next nodes. Here nodes contain lexical information of words. Every word may have one or more entries in Lexicon, so for a particular word, if it contains more than one entry in Lexicon, 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.

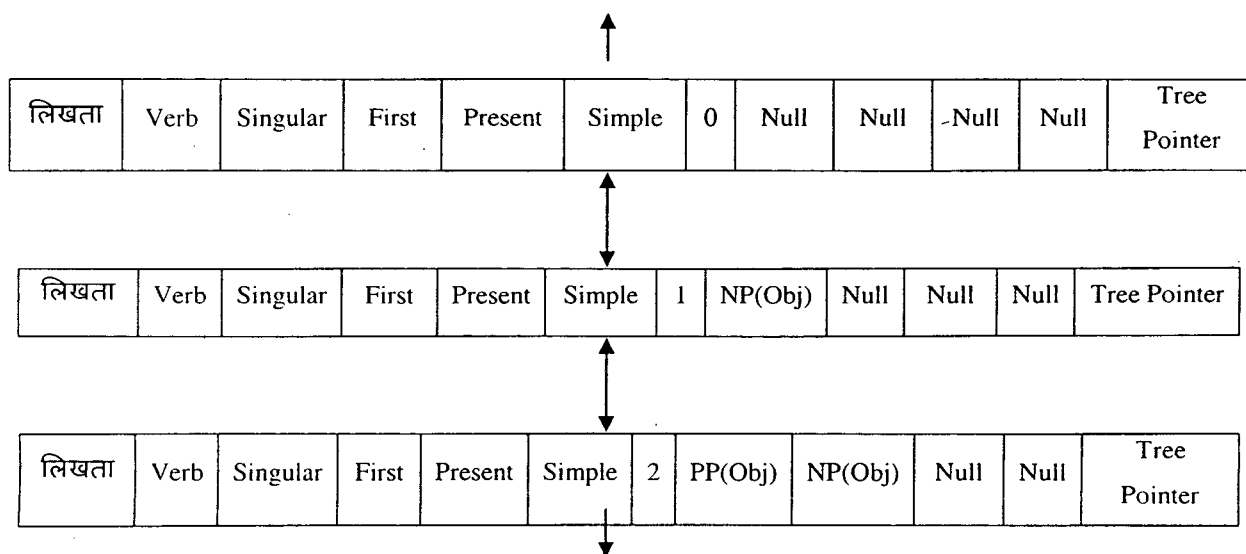


Figure 5.5 Data Structure for word लिखता

Example: if word लिखता 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.

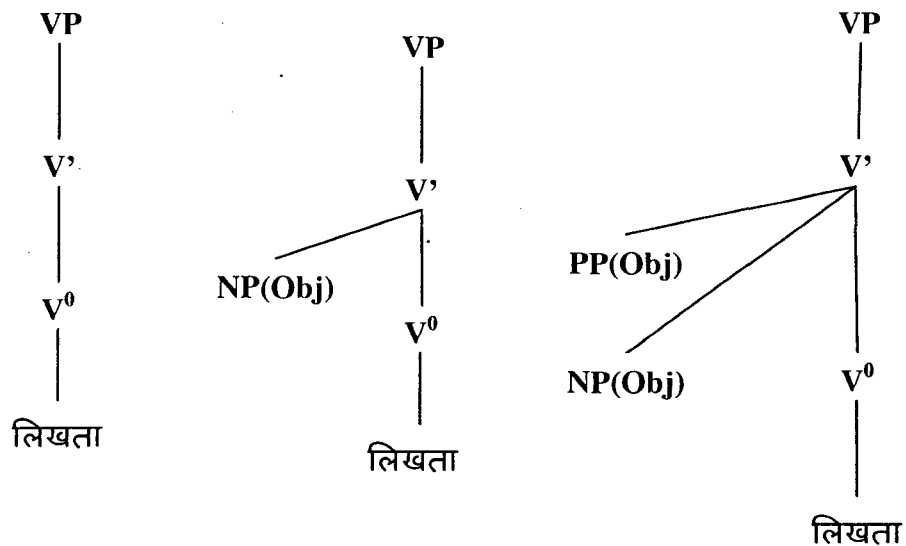


Figure 5.6 X-Bar Elementary Trees

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 Hindi 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 Hindi sentences given to parser at different instances will be:

राम लिखता है

राम मोहन को पत्र लिखता है

राम कलम से सुंदर पत्र लिखता है

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:

राम लिखता है

राम मोहन को पत्र लिखता है

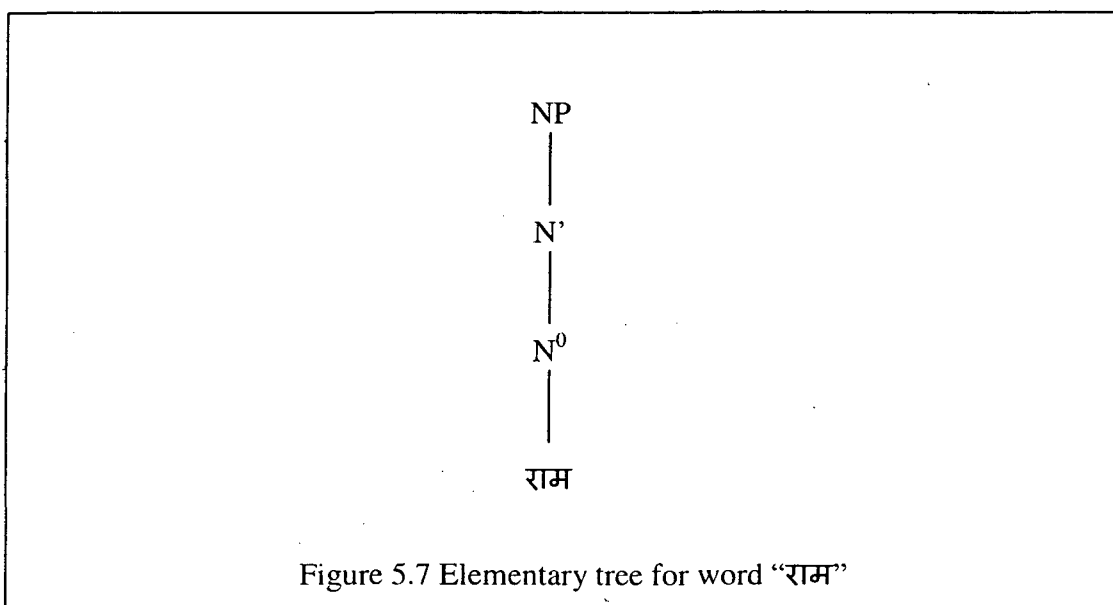
राम कलम से सुंदर पत्र लिखता है

Next, the normalized sentence is sent to Tagger. The Tagger first divides the sentence into lexical items राम, लिखता and है for first sentence, and राम, मोहन, को, सुंदर, पत्र, लिखता and है for second sentence and राम, कलम, से, पत्र, लिखता and है 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.

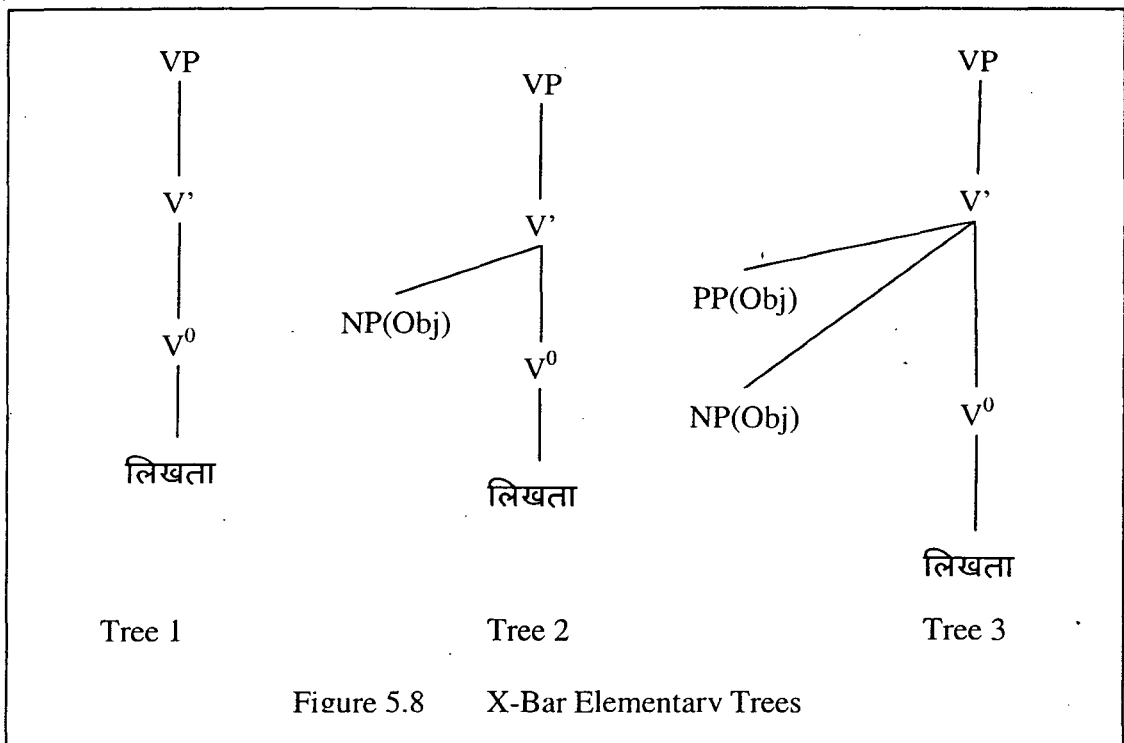
Now for every sentence parser reacts differently to build the complete parse tree. First we will see how parser will work for the sentence “राम लिखता है”.

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

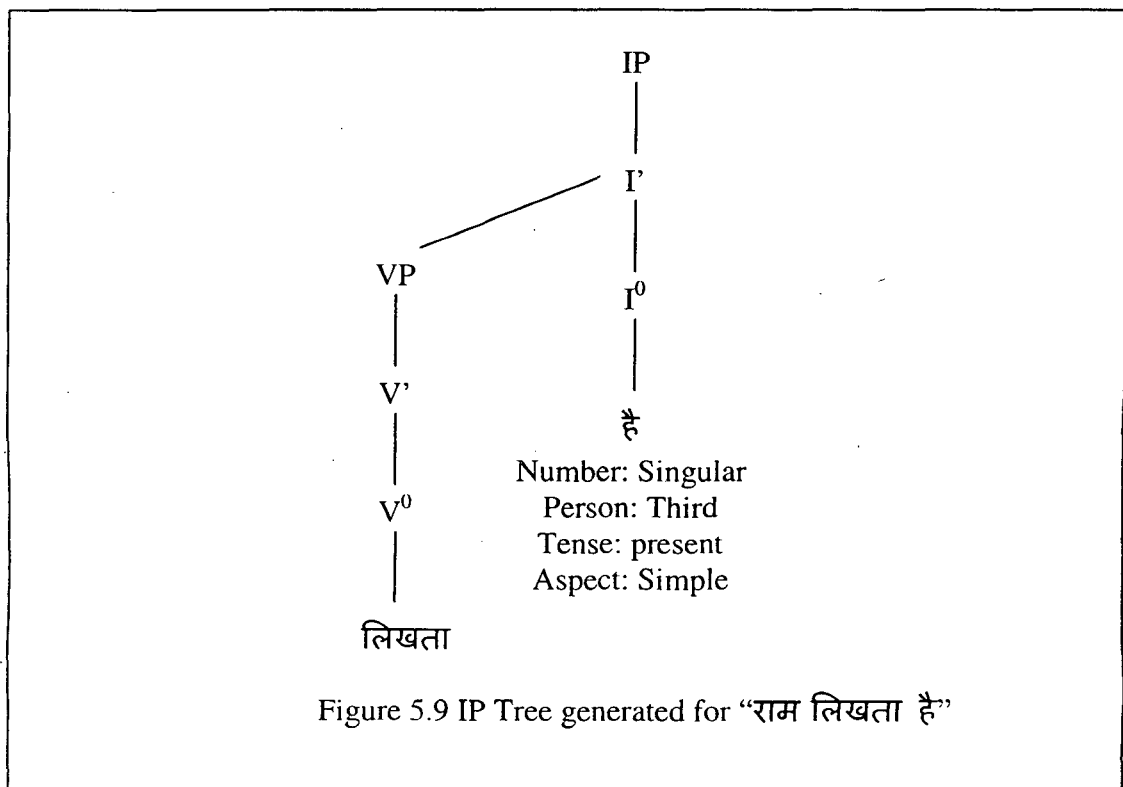


As it does not require any complement, parser not searches for complements. Now parser will check for the adjuncts with using the information stored in adjunct table in sentence for word राम, as no adjunct is available, tree will remain same.

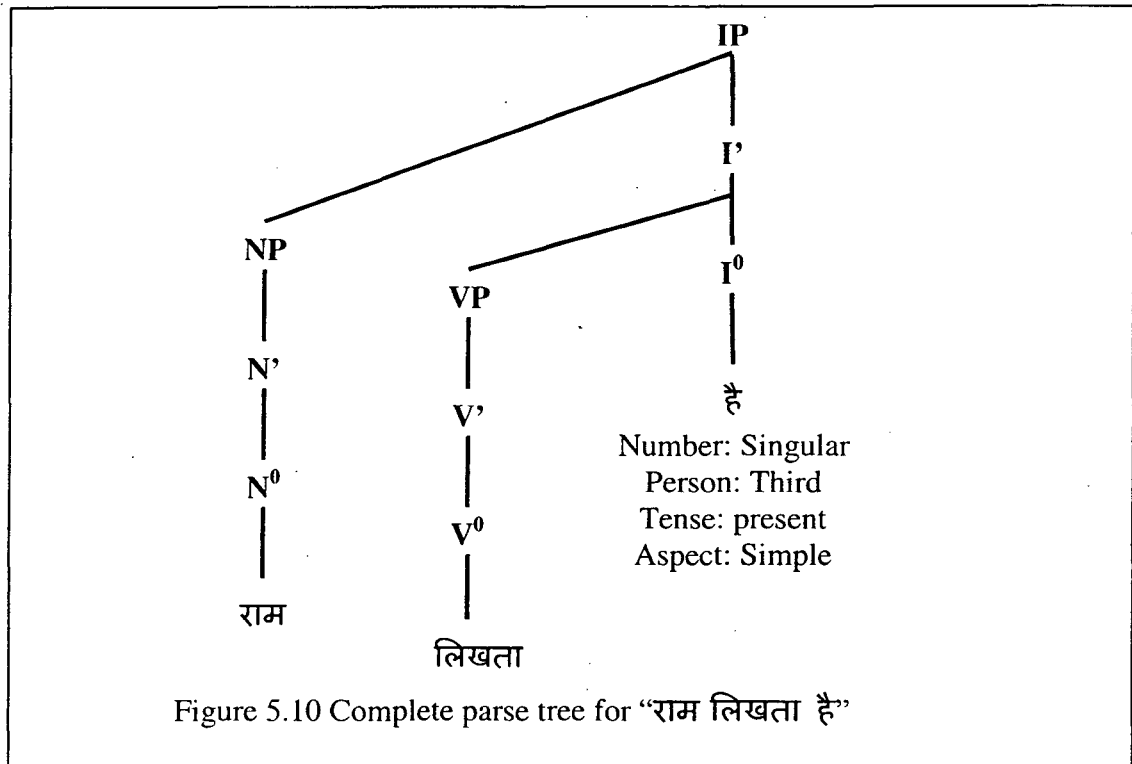
Now the word लिखता will be encountered by parser. This word has three entries in the lexicon, so three elementary trees will be generated as shown in Figure 5.8.



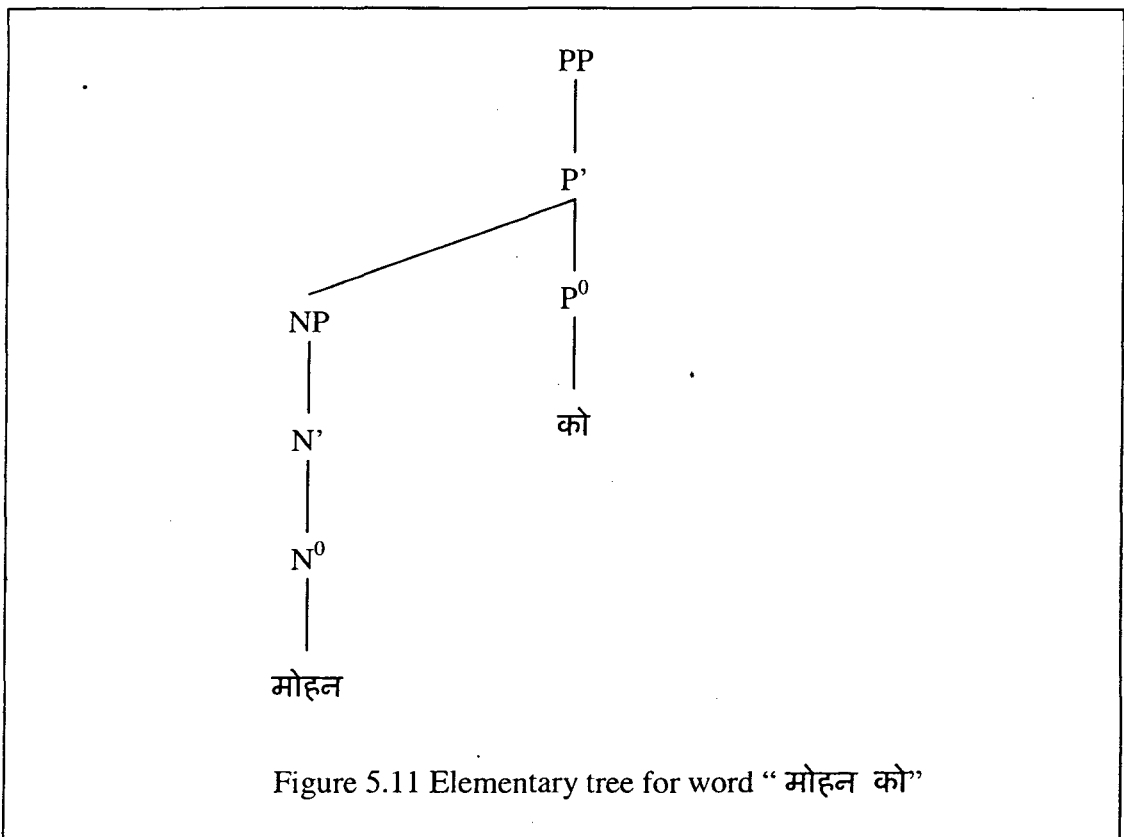
The tree 2 and 3 will be deleted, as these trees can not find required complements. So, only tree 1 will remain. As parser encounters "है", it generates an IP and connects VP as complements for it. So, resulting tree after generating IP is shown in Figure 5.9.



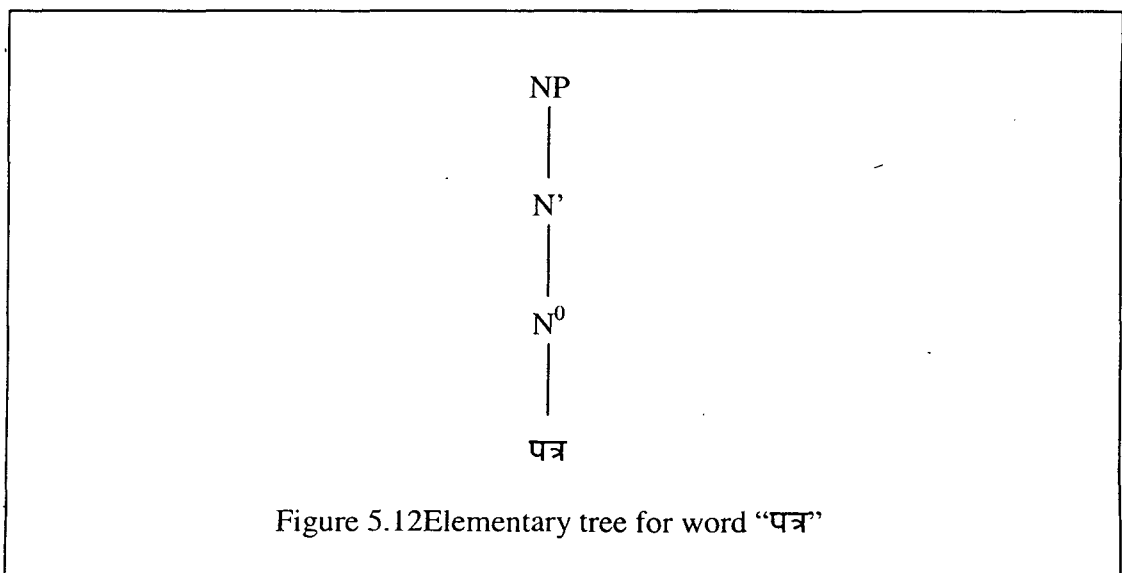
Now parser will check for the specifier for the IP, specifier is to be known from specifier table. Specifier of IP always a NP, with a Nominative case marker. So the final tree will be generated as shown in figure 5.10.



The next sentence “राम मोहन को पत्र लिखता है” the parser acts differently. For word “राम” parser acts similarly as acted for “राम लिखता है” and generates tree shown in figure 5.7.

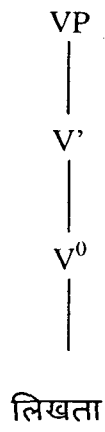


The next word is मोहन of sentence can have no complements. As word “मोहन” also did not get any adjunct, elementary tree generated will not be modified, as it followed by a Post Position “को”, which takes NP as complement, the tree for “मोहन को” generates a tree shown in Figure 5.11.

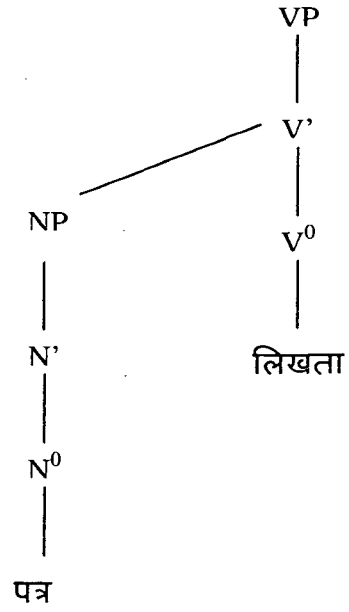


The next word is पत्र of sentence can have no complements and can not get any adjuncts. The elementary tree is shown in Figure 5.12.

Now parser encounters word “लिखता” 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 “पत्र” and third tree requires one PP(Obj) and one NP(Obj) as complements and can get these in form of मोहन को, पत्र. So resulted trees looks like in Figure 5.13.



Tree 1



Tree 2

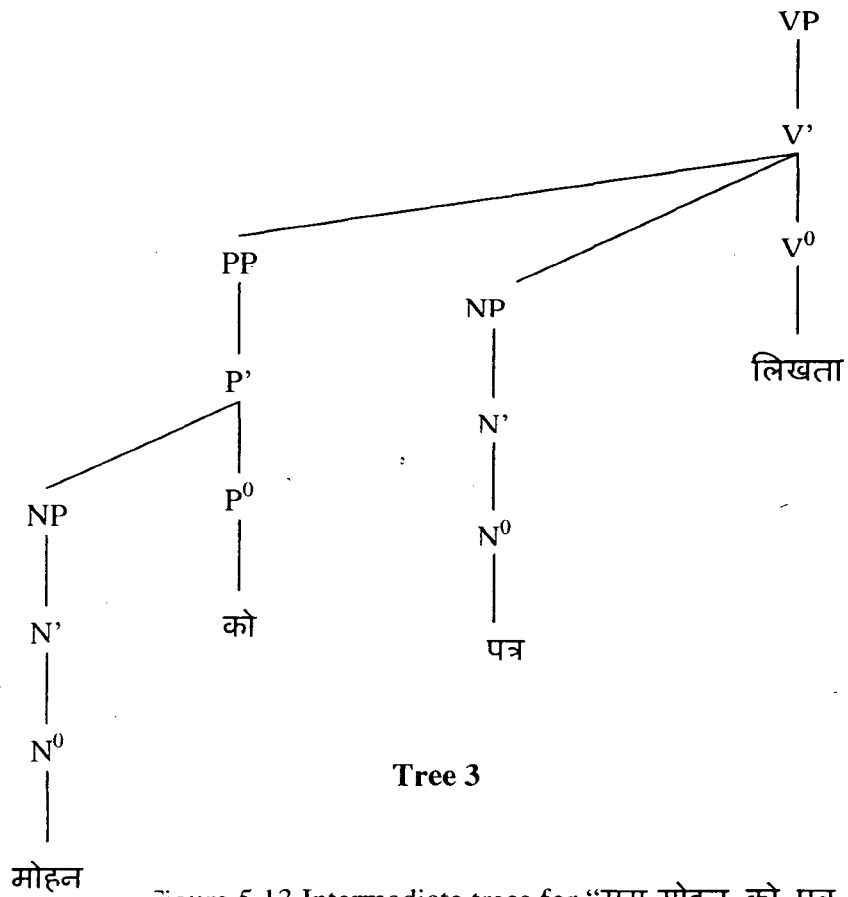
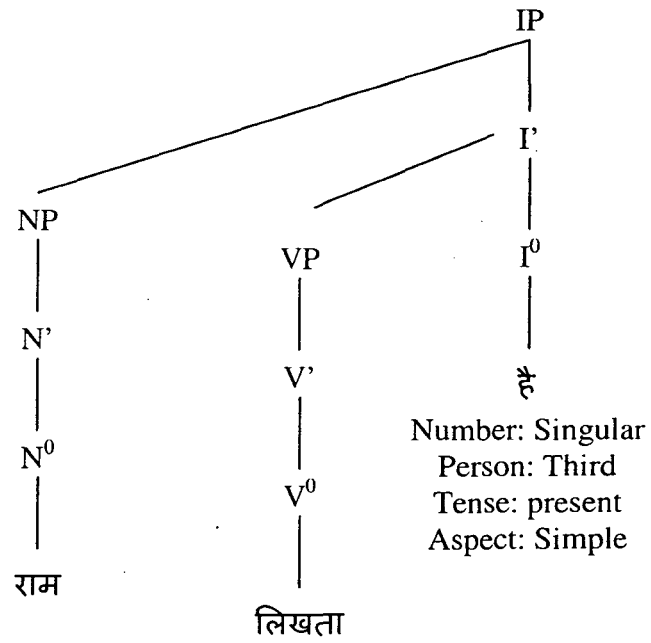
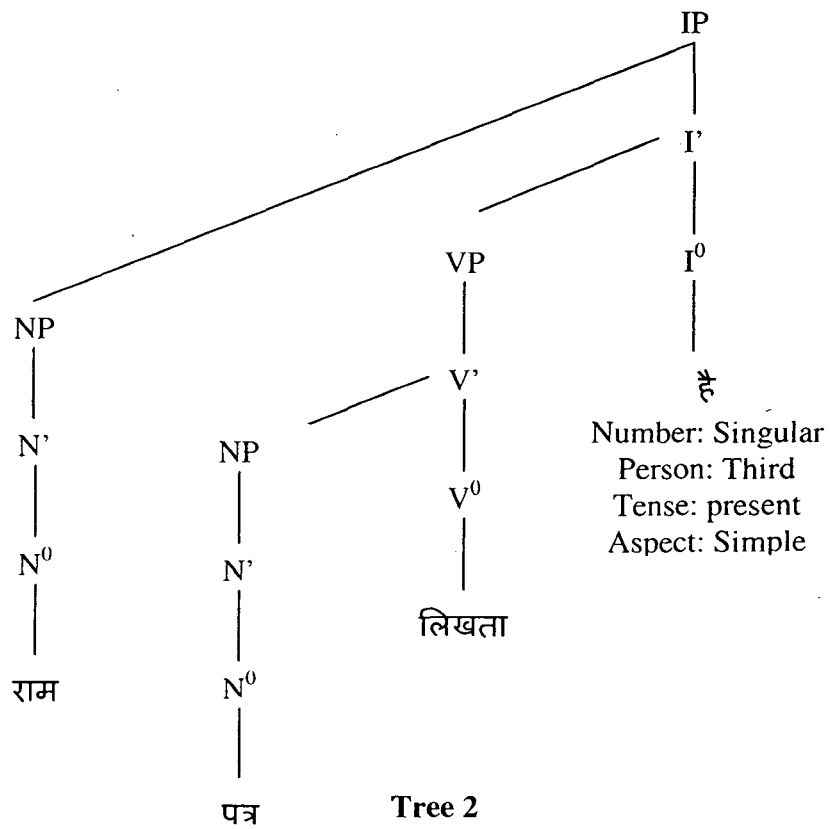


Figure 5.13 Intermediate trees for "राम मोहन को पत्र लिखता है"

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

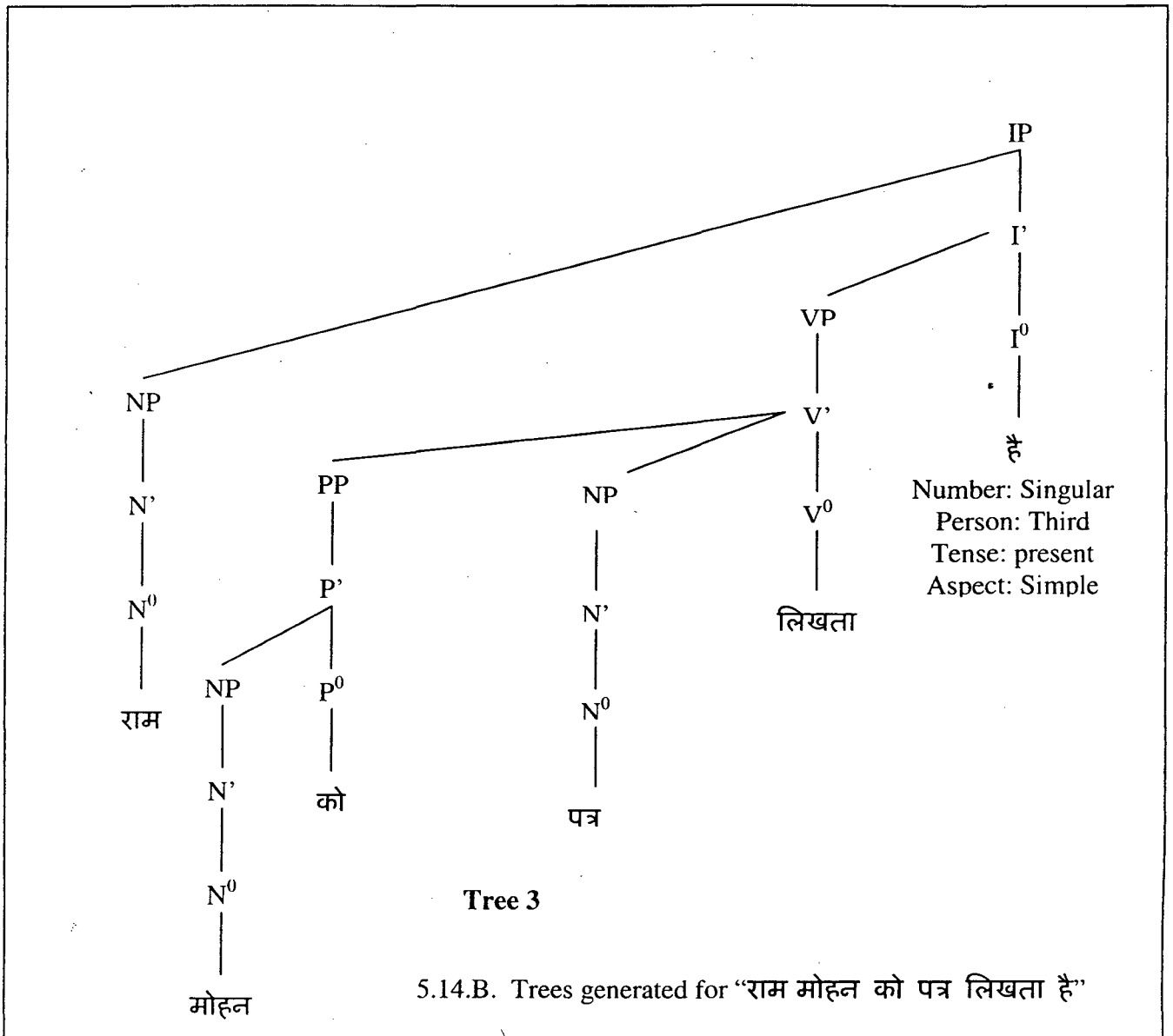


Tree 1



Tree 2

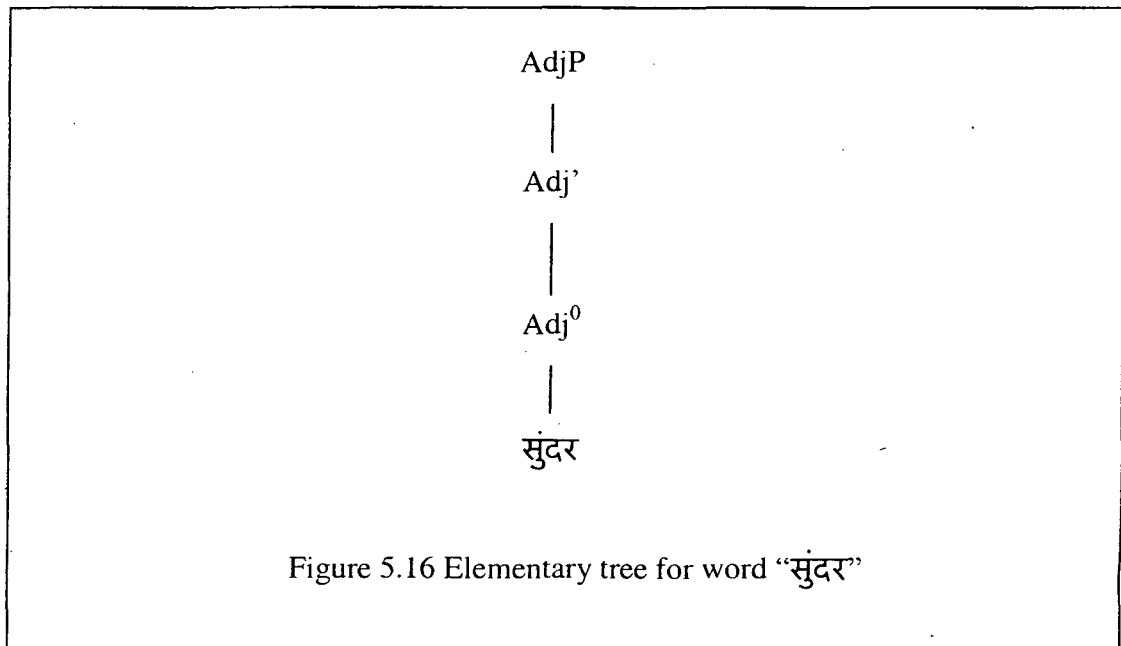
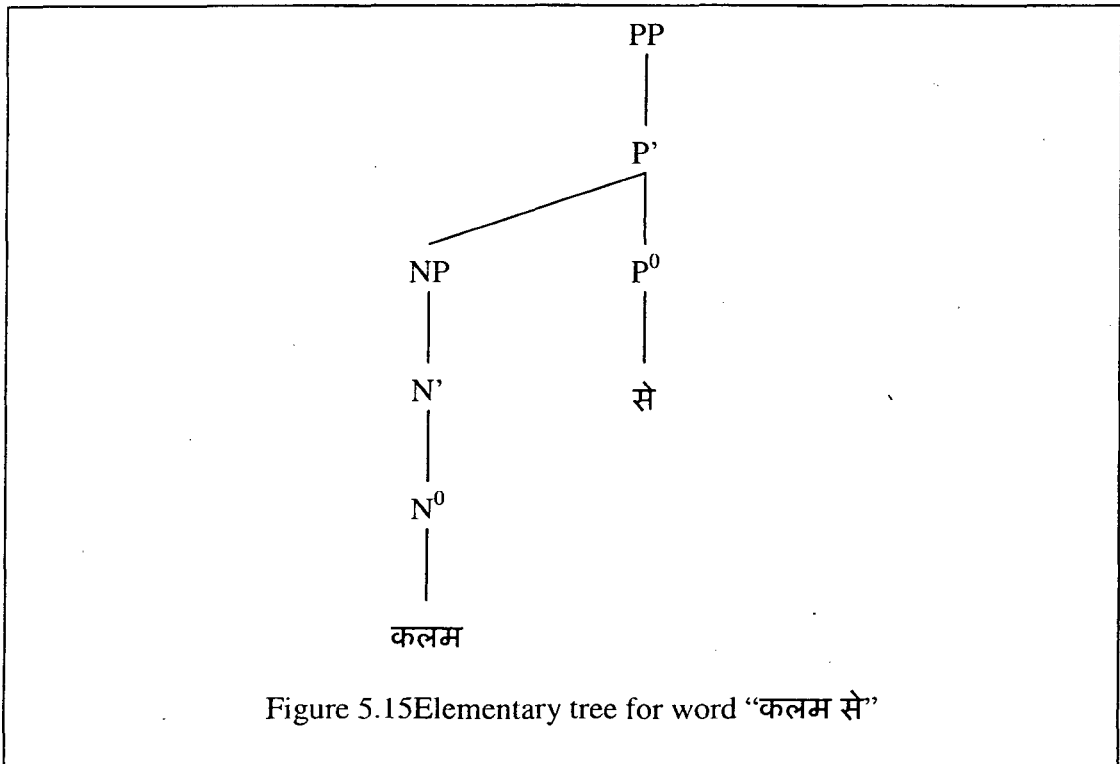
5.14. A. Trees generated for "राम मोहन को पत्र लिखता है"



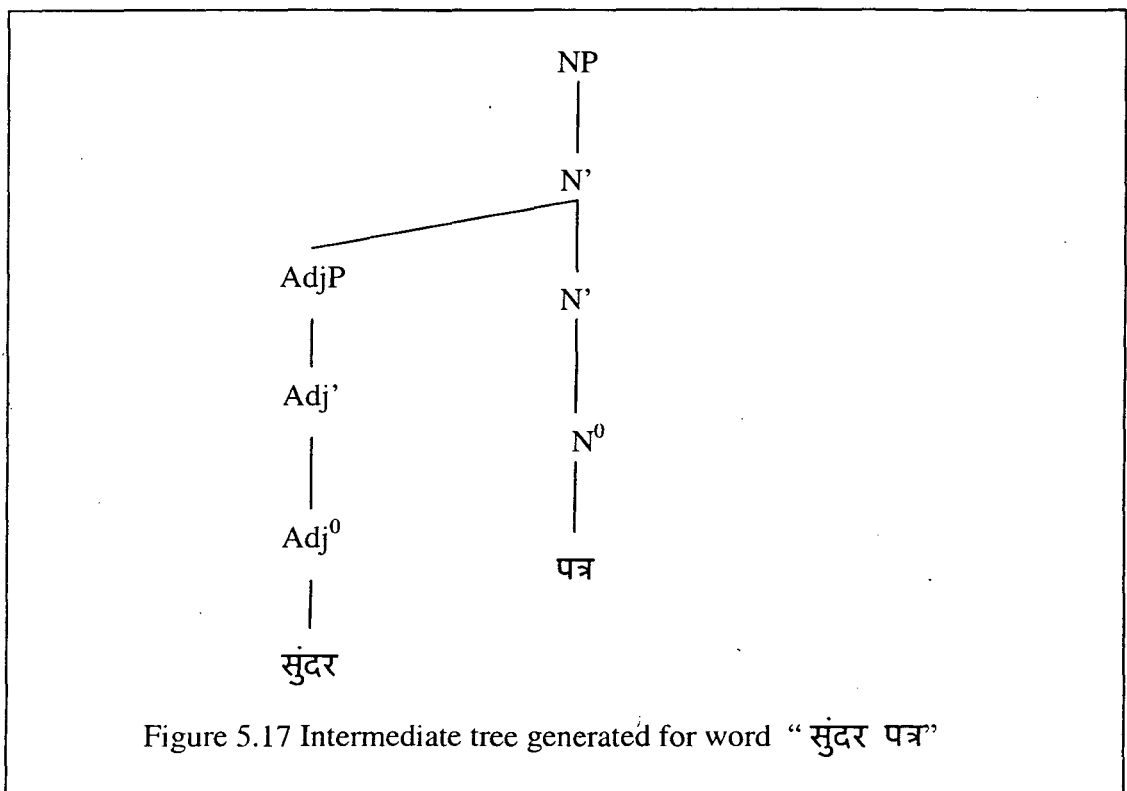
Here three final trees are generated, which are shown in figure 5.14. But only tree 3 represents all words in the sentence. So, remaining trees will be deleted. Tree 3 will become final tree.

The next is sentence “राम कलम से सुंदर पत्र लिखता है”. In case “राम”, it is similar to earlier cases. So, it generates a tree shown in 5.7. The next word “कलम” also not have any complements and its adjunct details does not match with any word in sentence, it have a Post position word “से”so a tree shown in figure 5.15 will be created. The next word is सुंदर 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 पत्र can have no complements and can find a matched adjunct to it as “सुंदर”. The elementary tree will be altered and tree shown in figure 5.17 will be created.



Now parser encounters “लिखता”, 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 लिखता, parser can get “कलम” as the adjunct for tree 2 of figure 5.18. “कलम से”, “सुन्दरम” will be adjuncts for tree 1 of figure 5.18. So trees will be generated as shown in figure 5.19.

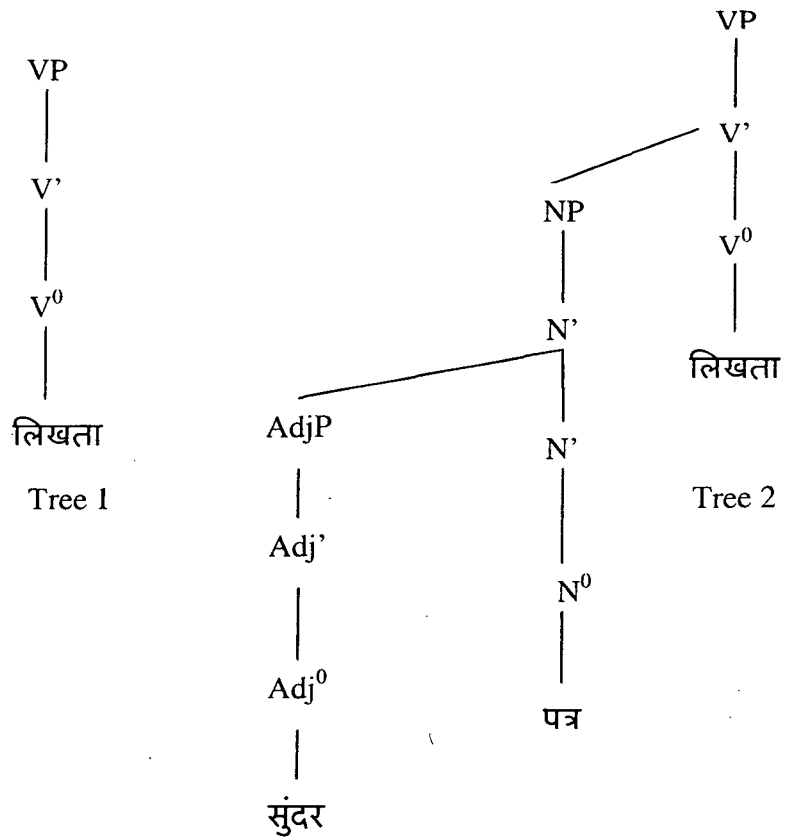


Figure 5.18 Intermediate trees for "राम कलम से सुंदर पत्र लिखता है"

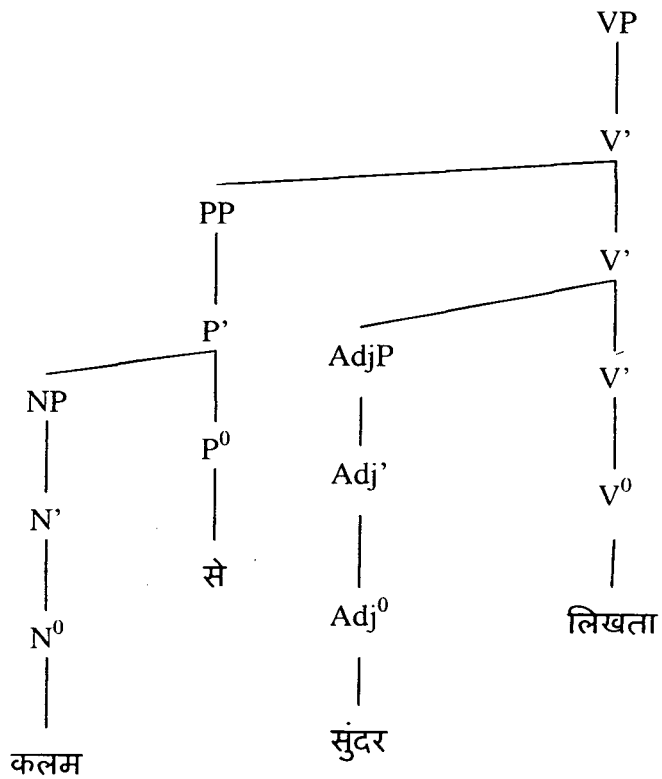
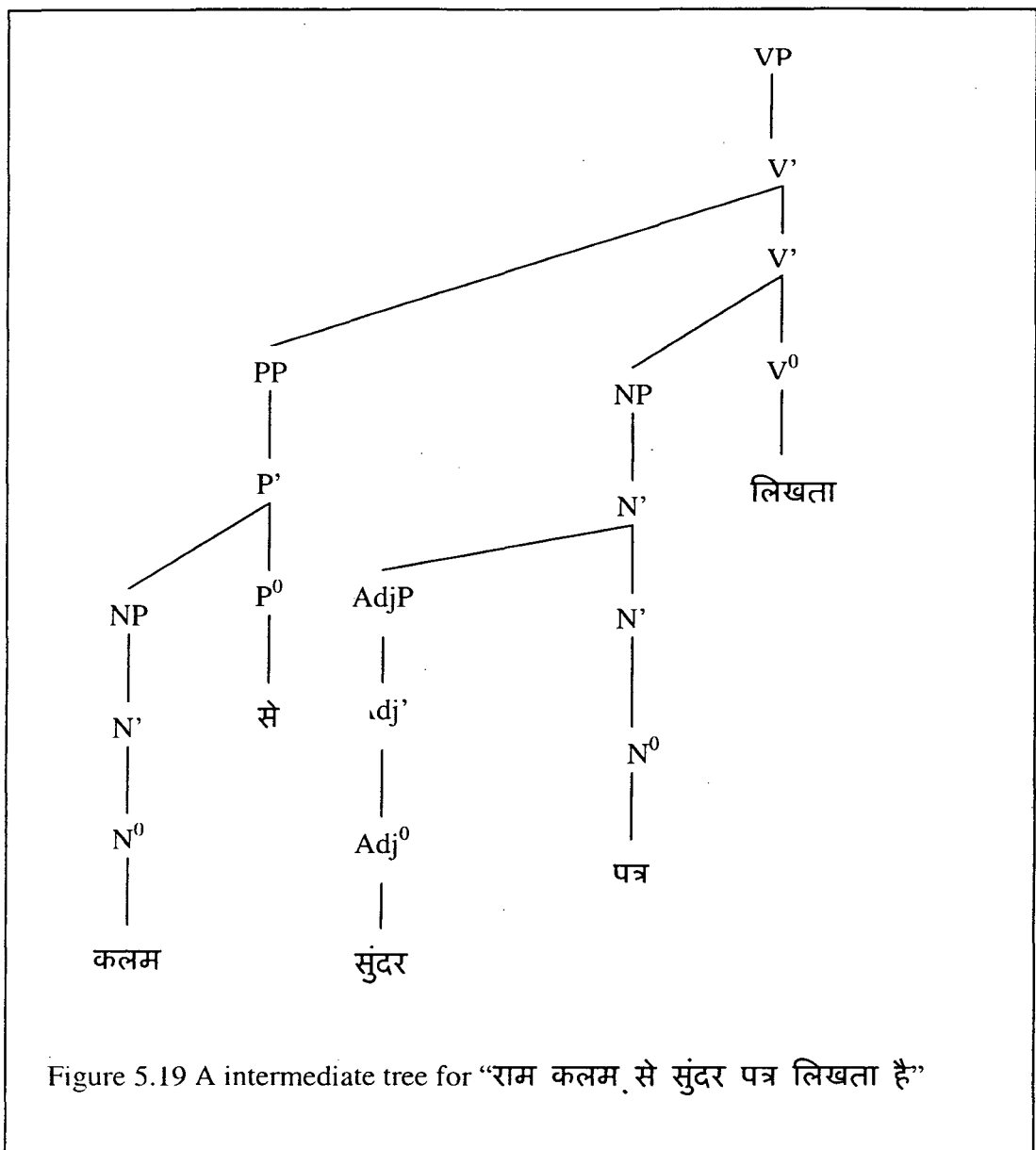
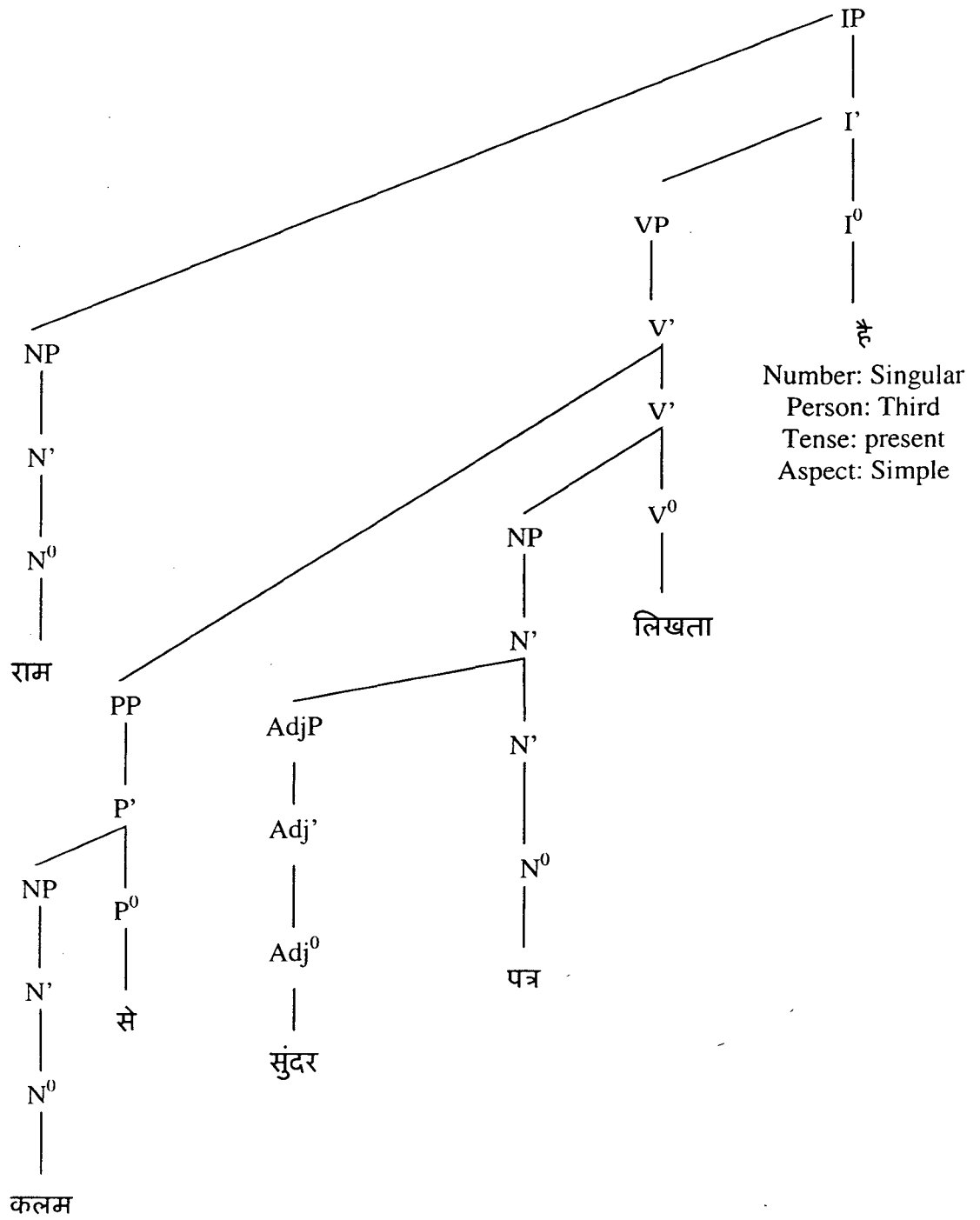


Figure 5.19 A Intermediate trees for "राम कलम से सुंदर पत्र लिखता है"



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.



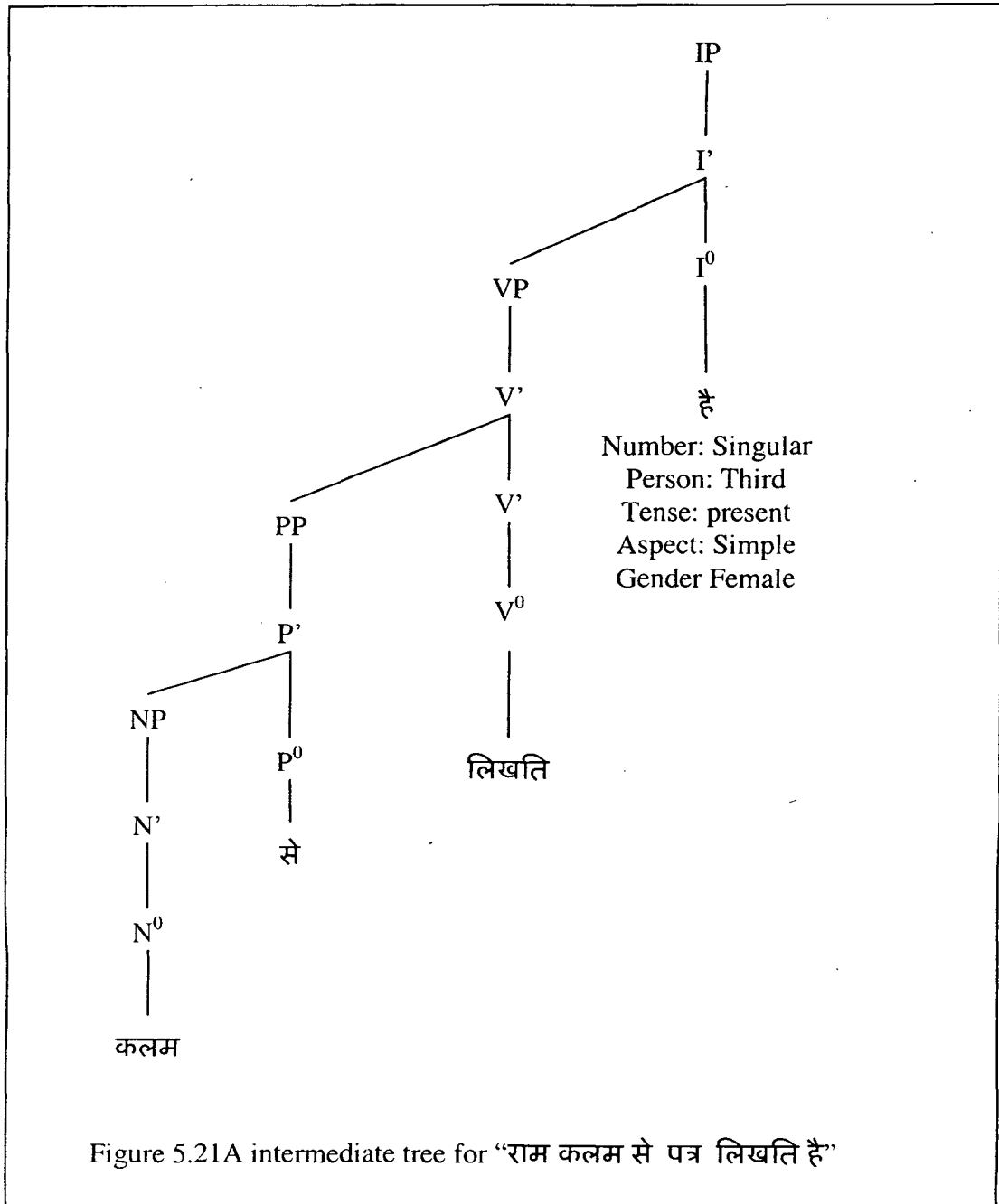
Number: Singular
 Person: Third
 Tense: present
 Aspect: Simple

5.20 Complete parse tree generated for "राम कलम से सुंदर पत्र लिखता है"

5.3.2 with a Wrong Sentence

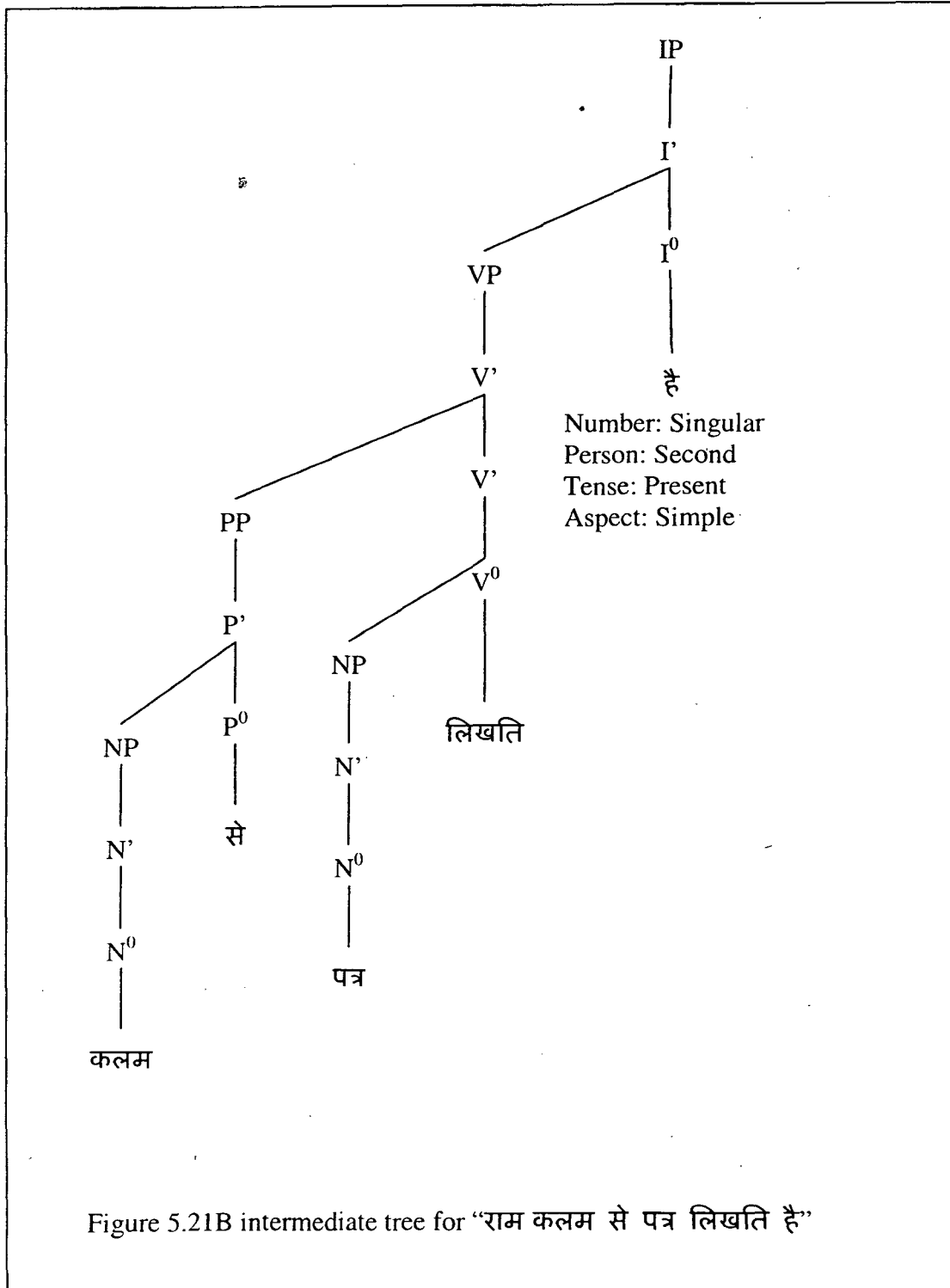
This section explains how parser will reject a wrong sentence, when it given as input to the parser. If the following sentence given as input to parser,

राम कलम से पत्र लिखति है



राम, कलम and पत्र will generate elementary trees as explained in previous section.

The word “लिखति” is the verb. Which have the same entries in lexicon as of “लिखता”, but only one attribute is different. That is, gender is “female”. The IP will be generated as shown in figure 5.21 for this sentence.



Both trees need a specifier which is having a gender attribute as female only, but word “राम” 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

CONCLUSIONS AND FUTURE ENHANCEMENTS

6.1 CONCLUSIONS

We have developed Elementary trees based parsing for Natural Language with Hindi 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 Hindi 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 Hindi 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.

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

We have already said that Hindi 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. 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

- [18] www.systransoft.com
- [19] Ullman J. D. and Hopcroft J. E. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [20] Wren & Martin, 2001, *High School English Grammar & Composition*.
- [21] <http://www.it-c.dk/people/pfw/hindi>

