A WEB-BASED
SIMPLE SENTENCE LEVEL
GB TRANSLATOR FROM SANSKRIT TO ENGLISH

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

Master of Technology

In

Computer Science and Technology

By

Pulipati Vijay Nirmallathma

Under the Supervision

of

Prof. G. V. Singh

SCHOOL OF COMPUTER & SYSTEMS SCIENCES

JAWARLAL NEHRU UNIVERSITY

NEW DELHI -110067

July 2005

JAWAHALAL NEHRU UNIVERSITY
School of Computer & Systems Sciences
NEW DELHI- 110067, INDIA

CERTIFICATE

This is to certify that the project entitled "**A WEB-BASED SIMPLE SENTENCE LEVEL GB TRANSLATOR FROM SANSKRIT TO ENGLISH**" being submitted by **Pulipati Vijay Nirmallathma** 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.

12/8/05

Prof.G.V.Singh 28/6/05
Professor, SC&SS,
JNU, New Delhi-67

Dean, SC&SS,
JNU, New Delhi-67

Dedicated to

My beloved

Parents

ACKNOWLEDGEMENTS

I would like to pay obeisance of 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 System Sciences, Jawaharlal Nehru University for the help, encouragement and support extended by him in successful completion of this project. His ideas were innovative and the valuable discussions we had were very much helpful in keeping the thesis work on the right track.

I would like to record my science my sincere thanks to Dean, Prof.Karmeshu for providing the necessary facilities. I will fail my duty if I forget my appreciation and thanks to my friends: Mr. Venkateswara Rao Katuri, Mr. Satish Addanki, Mr. Vijay Kumar Raja Injeti, and Mr. Nagaraju D. I 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.

P. Vijay

(Pulipati Vijay Nirmallathma)

CONTENTS

CHAPTER 2

CHAPTER 5

CHAPTER 6

CHAPTER 7

ABSTRACT

A web-based simple sentence GB translation system with Sanskrit as source language and English as target language has been developed. While various translations systems are being developed across the world using conventional approaches like Ruled- based or Exampled-based, we have adopted Government and Binding (GB) approach in our translation 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 important modules of GB are X-Bar levels and phrase structures, Theta assignment module, Case assignment module, and Binding module.

The Phrase Structures are developed for Sanskrit and English. These include Verb Phrase Structure, Noun Phrase Structure, Adjective Phrase Structure, Preposition Phrase Structure, Inflection Phrase Structure and Complementizer Phrase Structure. These Phrase Structures have been obtained after a thorough analysis of various phrases in both source and target languages. The analysis includes determining the complements for each lexical type, determining adjuncts and specifiers for each type of Phrase Structure

A bilingual lexicon has been developed for the translation system which contains category and subcategory information for the words in both source and target language sentences. The category and subcategory information include the attributes like tense, aspect, number, gender, person etc.

Morphological processes for source and target language are also developed as part of the translation system. The morphological processes include rules for Noun morphology, Adjective morphology and Verb morphology for both source and target languages.

An LR Parser based on Tomita's approach has been developed as part of the translation system. Our Parser is based on the GB phrase structure rules for both source and target languages. The Parser developed here can handle any context-free grammars, including ambiguous grammars.

Chapter 1

INTRODUCTION

1.1 NATURAL LANGUAGE PROCESSING

A Natural language is any language used by humans to communicate between them (e.g. English, Hindi, Japanese, Telugu, etc.). Programming Languages such as C, C++, Java, etc. are known as artificial languages. Natural Language Processing refers to descriptions 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:

1.1.1 Speech Recognition

Speech Recognition is basically the reduction of continuous sound waves to discrete words by a computer. 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 process of generating Natural Language utterances from any type written or hand written text. As almost every individual's utterance is different, to generate a speech to match the frequency and a style of individuals becomes if not impossible but a very difficult task. The problem of recognizing a written text in itself is quite difficult. For this reason designing and developing systems like Optical Character Recognition (OCR) in itself is a complex task. Thus the synthesis of natural-sounding speech is technically complex and almost certainly requires some 'understanding' of what is being spoken to ensure, for example, correct intonation.

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 refers to generating appropriate natural language responses to unpredictable inputs.

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 is a computer application to the task of analyzing the source text in one human language and producing an equivalent text called 'translated text' or 'target text' in the other language with or without human assistance as it may require a pre-editing and a post-editing phase.

The term does not include computer-based translation tools which support translators by providing access to dictionaries and remote terminology databases, facilitating the transmission and reception of machine-readable texts, or interacting with word processing, text editing or printing equipment. However, it includes systems in which translators assist computers in the production of translations, including various combinations of text preparation, on-line interactions and subsequent revisions of the output. Most of the Machine translation systems today produce a rough copy of the translated text to be often improved by human translators. However, in fields with highly limited ranges of vocabulary and simple sentence structures, for example the domain of weather reporting, Machine translation can deliver high quality results.

1.2.1 Brief History of MT

The first attempts at Machine translation were conducted after World War II. It was assumed at that time that the newly invented computers would have no trouble in translating texts. The logic was that computers were able to do complex mathematics quickly, something that humans did with difficulty. This logic was extended for translation by the computers. Further as even young children were able to learn to understand human language; therefore it was assumed that computers could do the same without much problem. However, this belief was soon shown to be incorrect.

On 7 January 1954, the first public demonstration of a MT system was held in New York at the head office of IBM. The demonstration was widely reported in the newspapers and received much public interest. The system itself, however, was no more than what today would be called a "toy" system, having just 250 words and translating just 49 carefully selected Russian sentences into English - mainly in the field of chemistry. Nevertheless it encouraged the view that MT was imminent - and in particular stimulated the financing of MT research, not just in the US but worldwide [17].

The first serious MT systems were used during the Cold War to parse texts in Russian scientific journals [17]. The rough translations produced were sufficient to understand the "gist" of the articles. If an article discussed a subject deemed to be of security interest, it was sent to a human translator for a complete translation; if not, it was discarded.

The advent of low-cost and more powerful computers towards the end of the 20th century brought MT to the masses, as did the availability of sites on the Internet. Much of the effort previously spent on MT research, however, has shifted to the development of computer-assisted translation (CAT) systems, such as translation memories, which are seen to be more successful and profitable.

1.2.2 Aims of MT

The humanity's dream of automatic translation is now being realized. Man has made computers capable of translating a wide variety of texts form one language into other. However, though the computers are able to translate but they are not yet perfect as they still can not handle the high level of ambiguities in the natural language which are easily handled by humans. There is no 'translating machines' which, at the touch of a few buttons, can take any text in any language and produce a perfect translation in any other language without human intervention or assistance. These touch-button machines are ideal for the future.

Now a days that the demand for translations of scientific and technical documents, commercial and business transactions, administrative memoranda, legal documentation, instruction manuals, agricultural and medical text books, industrial patents, publicity leaflets, newspaper reports, etc is increasing. There is lot of professional translators employed to meet these needs. This work is quite difficult and also challenging. But much of it is repetitive, while at the same time requiring accuracy and consistency. The demand for such translations is increasing at a rate far beyond the capacity of the translation profession. Translators are seeking the assistance of machine. The computer community has given immediate responses towards these challenges. The practical usefulness of an MT system is determined ultimately by the quality of its output. But what counts as a 'good' translation, whether produced by human or machine, is an extremely difficult concept to define precisely. It depends on the particular circumstances in which it is made and the recipient for whom it is intended. The ultimate aim of the MT is to make computer simulate the humans in terms of understanding and interpreting the natural languages i.e. it should be intelligent enough to translate according to the real world environment, which the humans actually do. This may need huge amount of data and lot of parameters to be considered to produce a very high quality translation.

1.2.3 Need for Translation of Indian Languages:

"India has 18 major regional languages written in 10 different scripts. However, English, though spoken by a minuscule 3 percent of the population, is still the de-facto link language for administration, business and control. All grass root information of land, agriculture, health and education needs to be disseminated in the respective regional languages for effective communication and understanding. Hence, translation is as important as basic and necessary infrastructure like roads, water and transportation for a country like India."

--- **R M K Sinha**, professor, Computer Science and Engineering, IIT Kanpur.

"India is a linguistically rich area-we have Hindi, English, and fourteen other official languages, each of which is spoken by millions of people. Since most information is generated in English, Machine translation has emerged as a critical technology that can help communicate and share information more effectively."

---**Durgesh Rao**, research scientist, Knowledge based Computer Systems Division, NCST.

The importance and need for development of Machine translation systems for Indian languages can be clearly seen from the above statements made by eminent scholars.

1.2.4 Need for the Current Work

People both in India and abroad are surely and steadily realizing the importance of ancient scripts in the diverse fields of science, commerce and arts. Also, as spiritual awareness sweeps the world, great efforts are being made to present the Vedic scriptures in different languages to the common people. In IIT Kanpur, a group of scholars is involved in a noble cause of putting up Shrimad Bhagavad Gita and its translation on to the internet. The Vedic texts are mainly in the Sanskrit language. Since Sanskrit is an ancient language and is no more in vogue as an easily understandable language, many communities are constantly working towards the translation of Sanskrit texts to popular languages. As English is popularly used language, a translator which can translate Sanskrit sentences into English will prove very useful. Researchers have developed a number of grammatical frameworks, semantic frameworks and parsing techniques to be employed in the task of Machine translation. The selection of these tools and techniques defines the translation strategy. In the next section we will briefly describe some of the Machine translation techniques.

1.3 TECHNIQUES OF MACHINE TRANSLATION

The Machine translation has two generations to be considered during its development. The first generation Machine translations are those which were done in 1960s and are called direct MT. They used the direct approach of translation which was based on word-to-word and/or phrase-to-phrase translations. Simple word-to-word translation cannot resolve the ambiguities arising in MT. A more thorough analysis of source language text is required to produce better translation. As the major problem of the first generation MT was the lack of linguistic information about source text, researchers therefore moved onto finding ways to capture this information. This gave rise to the development of the indirect MT systems which are generally regarded as second generation MT systems. The Figure 1.0 depicts a brief summary of relationship between these MT systems. Subsequent subsections give the details of the Direct and the Indirect MT systems.

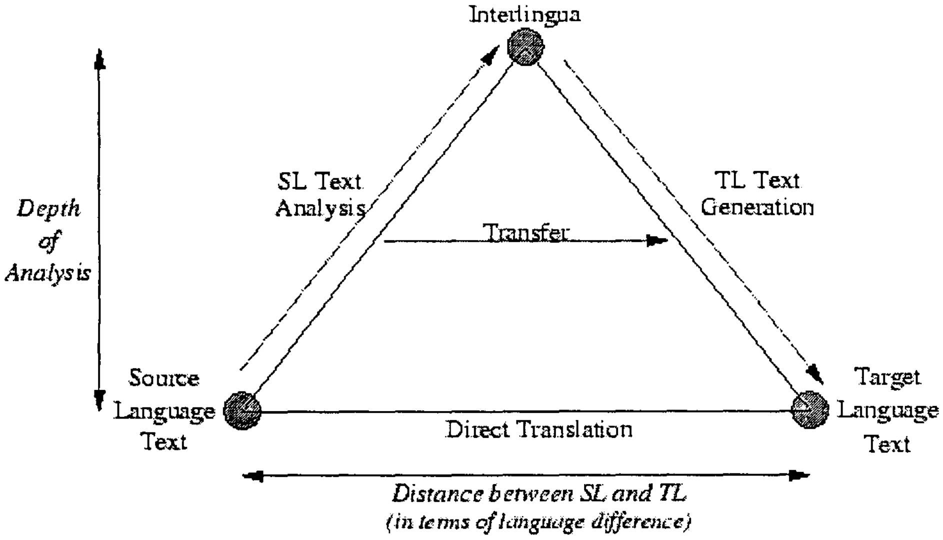


Figure 1.0: A brief summary of the relationship between MT systems.

1.3.1 Direct MT System

The direct MT systems simply translate the source language texts in to the corresponding word-to-word or phrase-to-phrase manner by using the bilingual lexicon. They also perform some morphological analysis before looking into the bilingual lexicon for the root words. They will then perform the necessary reordering of the words as according to the target language sentence format. The morphological processes may improve the quality of the translation but they don't really analyze the structure of the source language. An example of the direct MT system is SYSTRAN [18]. The Figure 1.1 below shows the translation process for the direct MT

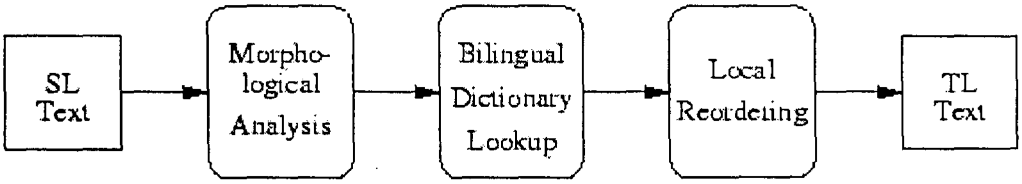


Figure 1.1 Direct MT System

The Direct Machine Translation was the technique developed in 1950s where the computers were in an early stage of technical development with very less speed which resulted in long processing time. Due to these constraints the direct MT used a very straightforward and easy to implement approach. It supports the translation of source language sentences to the sentences of the target language having structures similar to the structure in the source language. As very little effort is made to disambiguate the source language, this technique cannot translate highly ambiguous sentences. The main problem of the direct MT approach is that it does not analyze the linguistic information or the meaning of source sentences before performing the translation. Without this information, the resulting MT system cannot always resolve the ambiguities that arise in the source sentence and/or during the translation. As a result, the Direct MT systems cannot provide a quality translation of the source language text.

1.3.2 Indirect MT System

Owing to the fact that linguistic information helps an MT system to disambiguate source language sentences and to produce better quality target language translation, with the advance of computing technology, MT researchers started to develop methods to capture and use the linguistic information in the translation process. Hutchins & Somers (1992) [13] identified two kinds of Indirect MT systems: transfer-based and interlingual systems. Figures 1.2 and 1.3 depict the processes in these two types of translation systems.

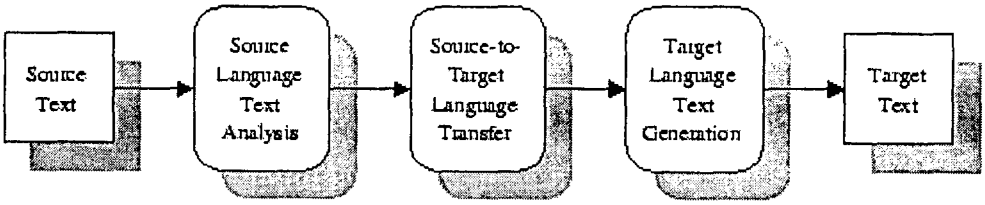

Source Text
Source Language Text Analysis
Source-to-Target Language Transfer
Target Language Text Generation
Target Text

Figure 1.2 Schematic representation of Transfer Based System

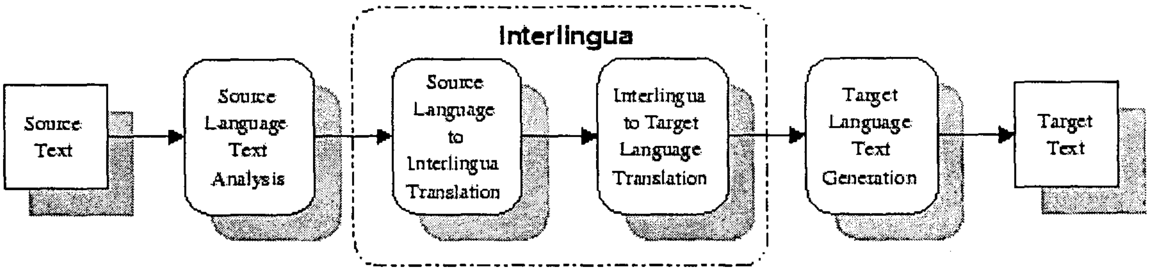

Interlingua
Source Text
Source Language Text Analysis
Source Language to Interlingua Translation
Interlingua to Target Language Translation
Target Language Text Generation
Target Text

Figure 1.3 Schematic representation of Interlingua System

The module 'Source Text Analysis' aims at capturing the required linguistic information about the source language sentences for aiding the translation. The transfer-based approach uses the information obtained from the analysis module directly to lookup the corresponding target language words. The Interlingua approach involves the use of an intermediate language (i.e. an Interlingua) for the transfer -- with the source language text translated to the Interlingua and the Interlingua translated to the target language text. As suggested by Hutchins & Somers (1992) [13], an Interlingua is an intermediate 'meaning' representation and this representation: "includes all information necessary for the generation of the target text without looking back to the original text. The representation is thus a projection from the source text and at the same time acts as the basis for the generation of the target text; it is an abstract representation of the target text as well as a representation of the source text."

According to the developers of MT using this approach they believe that the Interlingua is more regular and consistent, both lexically and structurally, than natural languages and could capture the characteristics of any natural language in a relatively precise way. In addition, if these artificial languages had already been developed, they can be incorporated to an interlingual MT system directly. No additional effort is required to define the Interlingua. After translating the source language words to their target language forms, the job of the 'Target Text Generation' module is to synthesize the resulting target language words to form the target sentences. One advantage of the transfer-based approach is that it allows the source language text to be analyzed according to what is required for facilitating its translation to a target language. Thus, much less effort, if any, would be wasted in analyzing the unnecessary features of the source language sentences. In addition, this approach also facilitates a close examination of the differences between a language pair. This, in turn, will facilitate the design and implementation of the required MT system. The interlingual approach, however, is more time-consuming as a lot of processing time is consumed in the 'double-transfer'. However, if a multilingual MT system is to be built, this approach would reduce the time and effort needed to produce a transfer module for each language pair as shown in Figure 1.4.

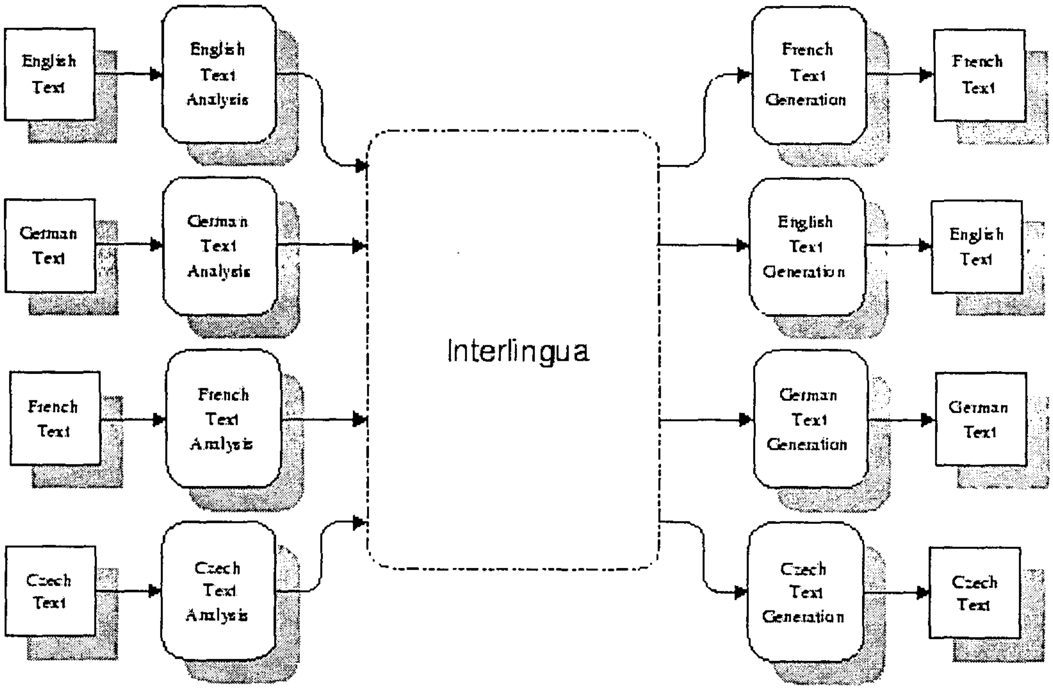


Figure 1.4: Building blocks of a multilingual MT system using the interlingual approach

1.3.3 Sublanguage MT

The researchers are still working to generalize the natural languages. However, due the presence of high level of ambiguities in the natural language it is very difficult to generalize the whole domain of the language. By restricting the scope of the language to domain-specific areas, e.g. language used in computing journals, manuals for specific products or weather reports it is possible to generalize the language. This process of

Machine translation in which we restrict the scope of the language to particular domain specific areas is called the sublanguage MT. It can help in narrowing down the scope of translation so that the size of the resulting MT system will be relatively small and of a manageable size. The term sublanguage, according to Arnold et al. (1994) [4], "refers to the specialized language used in certain fields of knowledge" and this specialized language is characterized by a specialized vocabulary which tends to be understandable by specialists only. An example of a well-known sublanguage MT system is 'Meteo' which was developed for translating weather reports from English to French only [4]. The sublanguage approach seems a more realistic way to develop MT systems for commercial purposes (especially for translating technical manuals and product labeling). This is because the input text is relatively regular and unambiguous. A relatively high quality translation is therefore made possible by a restricted domain of translation.

1.3.4 Example Based Machine Translation (EBMT)

Example based machine transition is basically translation by analogy. Given a set of sentences of source language and their corresponding translations in the target language the example based machine translation, uses these translations to translate the similar source sentences to target sentences. This technique is also considered as a case-based reasoning approach to MT, where previously resolved translation cases are reused to translate new source language text. The basic premise is that, if a previously translated sentence occurs again, the same translation is likely to be correct again.

The basic idea in this translation system is to collect a bilingual corpus of translation pairs and then use a best match algorithm to find the closest example to the source phrase in question. This gives a translation template, which can then be filled in by word-for-word translation. Instead of using explicit mapping rules for translating sentences from one language to another, the translation process is basically a procedure of matching the input sentence against the stored example translations. These bilingual translation pairs are repository called the translation memory. As the user translates the source sentences, the translations are added to the translation memory for further use. When the EBMT system gets a sentence for translation, it finds the closest match for the sentence with the previous translations in the translation memory using some matching algorithms, which can then be filled using word-for-word translation.

This translation system is not restricted to a specific domain. As the example translation set become more complete, the quality of the translation improves increasingly. When a sentence similar to the previous translation is given it will simply use word for word translation, avoiding the complex rule applications. This will save time wasted in re-translating it.

However, this approach demands huge collection of good bilingual data to make the translation reasonable. The calculation of the best match might be a complicated and lengthy process. However, in some cases, especially when an input sentence is relatively less ambiguous, a simple rule-based system which analyses the linguistic information about the input sentence would be less complicated and thus more efficient.

1.4 DEVELOPMENTS TO PRESENT DATE

There has been a lot of progress and renewed interest in Machine translation since the early 1980s. Some of the efficient MT systems used around the world are:

- **METEO** used at Canadian Meteorological Centre which does translation of over 45,000 words in weather bulletins. The system is operational since 1977.
- **EUROTRA** is a multilingual translation system developed in 1990. It is used for translation among European languages
- **MU** for Japanese and English, and **KMBT** for translation between English and Japanese are developed at Carnegie Mellon University.
- **SYSTRAN** is a MT system used by the AltaVista search engine.

India too has active groups in Machine translation. The earliest published work was by Chakraborthy in the year 1966 [19]. More recently, work has been undertaken in Tamil University in Thanjavur, National Center for Software Technology (NCST) in Mumbai, Center for Development of Advanced Computing (C-DAC) in Pune and IIT Kanpur [19].

A research group at Thanjavur attempted Russian to Tamil translation based on the direct approach in 1985 and the first system translated simple sentences. A group at NCST did some work on English to Hindi translation but did not develop a working system. Translation from Hindi to English is going on at the IIT Kanpur. However, except for some attempts concerning processing of Sanskrit texts performed by C-DAC, actual attempt towards translation from Sanskrit to English has not been made till date.

1.5 PROBLEM IDENTIFICATION

For the reasons stated earlier in this Chapter we have decided to engage ourselves in the translation work from Sanskrit to English. The thesis aims at building a web based simple sentence level Government and Binding (GB) theory based translator from Sanskrit to English language. The GB framework has been adopted for generating the target language sentences too. The key modules required in developing a translation system are: building a Lexicon, writing computational grammar for both source and target languages and design and implementation of a parser. Tomita's approach will be the basis for the design and implementation of the parser. The transfer approach, along with the lexical information has been adopted for converting the source language sentence to an equivalent target language sentences. Since the GB framework forms the heart of the overall system, it also requires developing GB based grammars both for source as well as target language. As each of these modules takes considerable amount of time and no previous work is available for the support of this work, we have confined our translation system to translate simple sentences only. Finally the overall system design and implementation is aimed to make the system available for translating source language sentences to target language sentences by submitting the source language sentences on the web. In the section 6 of this Chapter we present the organization of this thesis.

1.6 ORGANISATION OF THE THESIS

The thesis consists of 7 Chapters. Chapter 1 overviews the field of Natural Language Processing, gives an overview of Machine translation and also gives a brief introduction to various approaches for the task of Machine translation. This Chapter also gives an overview on the techniques of Machine translation. Then we discuss the developments to present date in the field of Natural Language Processing. Finally we explain the problem identification.

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. Then we focus on the Lexicon organization, Theta Theory, Case theory and Binding Theory.

Chapter 3 gives the GB based Phrase structures for English and Sanskrit, developed by us. Chapter 4 describes the morphological processes both for Sanskrit and English. The Interface between the Morphology with Lexicon is also described in this Chapter.

Chapter 5 gives the overview of various parsing strategies, and then describes Bottom-up parsing in detail. Finally the Chapter shows the parsing process with examples. Chapter 6 gives the complete design of our Translation System, which includes Parsing and Generation Modules. We illustrate here the Translation Process in detail with an example.

Finally, the Chapter 7 concludes the work with suggestions for future. The Appendix A gives the Phrase Structure Rules for Sanskrit and the Appendix B gives the Phrase Structure Rules for English. References are places at the end of the thesis.

Chapter 2

GOVERNMENT AND BINDING THEORY

2.1 INTRODUCTION

The shift from programming language grammars to NLP grammars seriously increases complexity and requires ways to handle the ambiguities inherent in every human language. While most programming languages are expressed by subclasses of well-understood context-free grammars (CFGs), no grammatical formalism has yet been accepted by the linguistic community for the description of human languages. On the contrary, new formalisms (or variants of older ones) appear constantly. These include, for instance, Linear Indexed Grammar (LIG) [11], Range Concatenation Grammar (RCG)[6], Definite Clause Grammar (DCG) etc. More recent formalisms, like Head-Driven Phrasal Structure Grammar (HPSG) [7], Lexical Functional Grammar (LFG) [16], which relies on more expressive Typed Feature Structures (TFS) [5] or constraints, and Tree Adjoining Grammar (TAG) with trees as elementary structures [2] & [3] have been more widely used by the Natural Language Processing community. However of all the grammar formalisms mentioned above, the Government and Binding (GB) Theory, with its emphasis on Universal Grammar, has had the most impact in NLP. Because of its very different approach in characterizing a language, its universality in handling Natural Languages and its interesting computational properties, we have adopted GB based approach for our work.

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

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

2.2 X-BAR LEVELS AND PHRASE STRUCTURES

For phrase structures we adopt symmetrical X-bar analysis. The same set of rules will be applicable to all phrases thus leading to uniformity and computational efficiency in terms of efforts, time and space. Further, we need exactly three levels of projection, namely X^0, X', and X". For any head X^0, we require the level X' to take care of constituents larger than X^0. Both the compulsory (Complements) and optional (Adjuncts) phrases can be joined at X', one at a time, by Adjunct Rule (refer Rule (2) below) are the Complement Rule (refer Rule (3) below), respectively. The highest X-bar which is not dominated by any other X projection is the maximal projection of X^0 denoted by X" or XP. The Specifier is attached at the X" level by the Specifier Rule (refer Rule (1) below). Thus we need only three levels namely X^0, X', and X". This three level hierarchical structure 'do so' substitution and 'one' substation constructs which can be taken care by two level flat structure (Hageman [12], Radford [2000]). Thus the three level symmetrical X-bar analysis is more suitable both linguistically as well as computationally.

Accordingly, an X-bar phrase in a language is defined by the following rule schemata: For any lexical category X such as Verb, Noun, Adjective, Preposition etc, X^0=Head

X'' (XP) → (XSpecier) ; X' -- (1) (The Specifier Rule)
*X' → (ZP) ; X' -- (2) (The Adjunct Rule)
X' → (YP) ; X^0 -- (3) (The Complement Rule)

The terms included in the parenthesis indicates that these terms are optional, '*' indicates that the Adjunct rule is optional, and a semicolon in the rules indicates that the terms on the right hand side are not taken as ordered. Each of ZP and YP are full phrases like XP. Maximal projections for categories Verb(V), Noun(N), Adjective(A), Preposion(P), etc, are denoted as VP, NP, AP, PP, etc, respectively. The tree representation defined by phrasal rules is given in Figure 2.1 below.

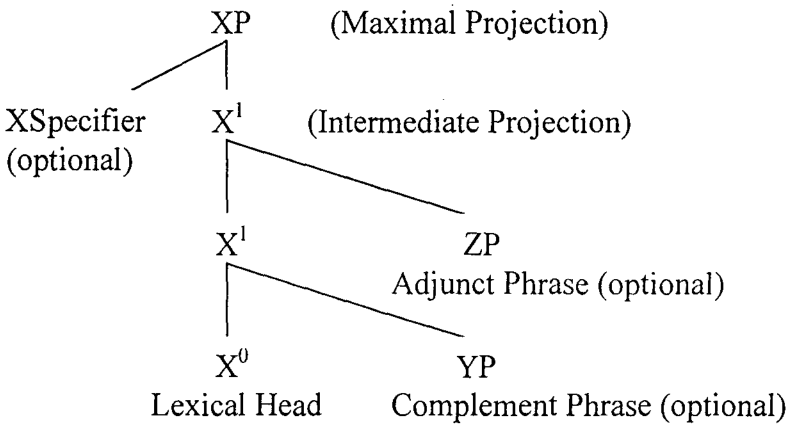

XP
(Maximal Projection)
XSpecifier
(optional)
X^1
(Intermediate Projection)
X^1
ZP
Adjunct Phrase (optional)
X^0
Lexical Head
YP
Complement Phrase (optional)

Figure 2.1: The pictorial representation of X-bar phrase rule schemata

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

2.2.1 The Phrasal Projection for a Verb

A Verb Phrase (VP) is a phrase having a verb as its head. Depending on a particular head complement(s) may be present or absent. The complement in a VP may be a NP, PP, AP, IP, or a CP. For each verb a specifier NP, according to the latest theory, is present which then moves to the specifier position IP as described later in the section. Adjunct(s) in a VP may be NPs, PPs, APs, or ADVP. A Verb Phrase typically functions as a

Complement in an Inflection Phrase (IP). For example, the VP 'talked to sue' in IP 'she talked to sue'. The tree representation for VP 'talked to sue' is given in Figure 2.2 below.

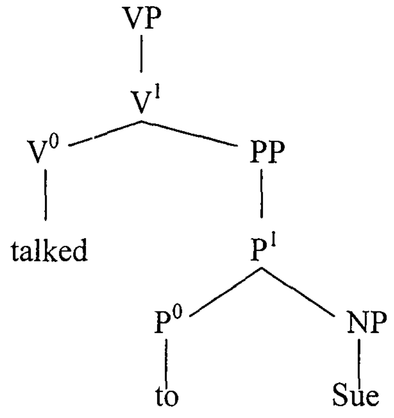


Figure 2.2 : Verb Pharase

2.2.2 The Phrasal Projection for a Noun

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

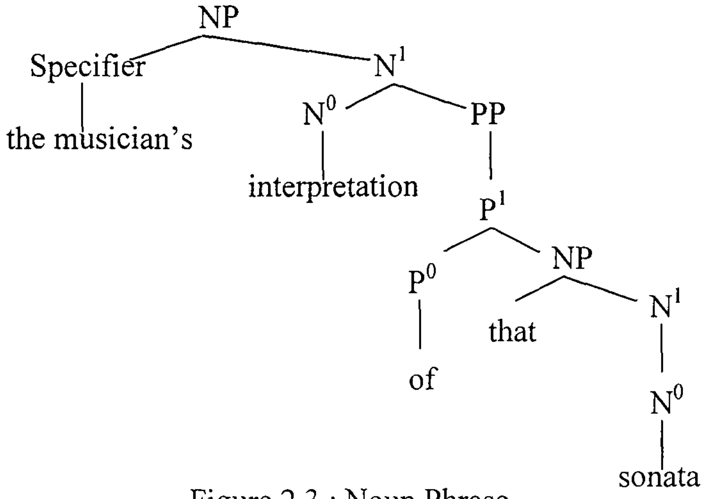


Figure 2.3 : Noun Phrase

2.2.3 The Phrasal Projection for an Adjective

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

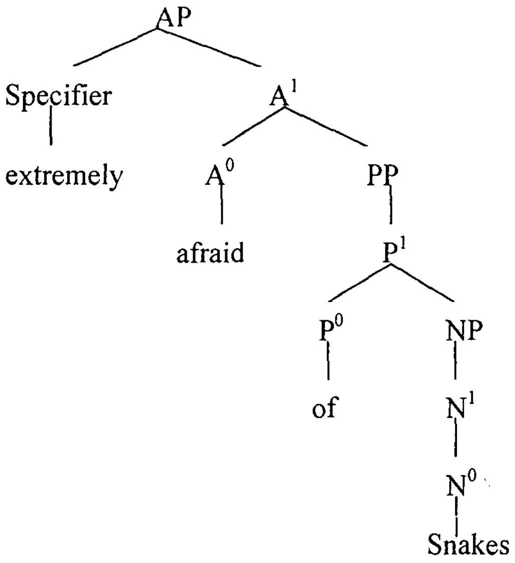


Figure 2.4 : Adjective Phrase

2.2.4 The Phrasal Projection for a Preposition

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

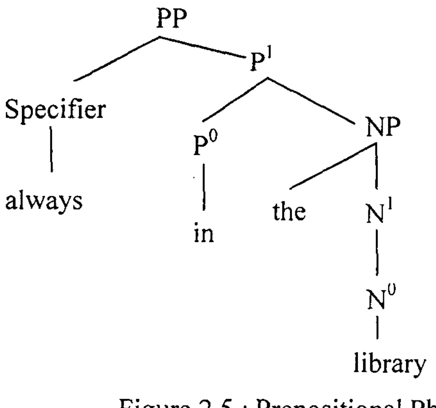


Figure 2.5 : Prepositional Phrase

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

2.2.5 The Phrasal Projection for a Sentence (Inflection Phrase)

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

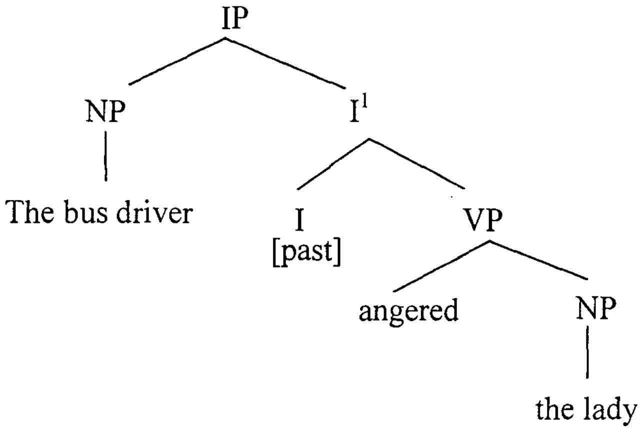


Figure 2.6 : Inflection Phrase

2.2.6 The Phrasal Projection for Complementizer

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

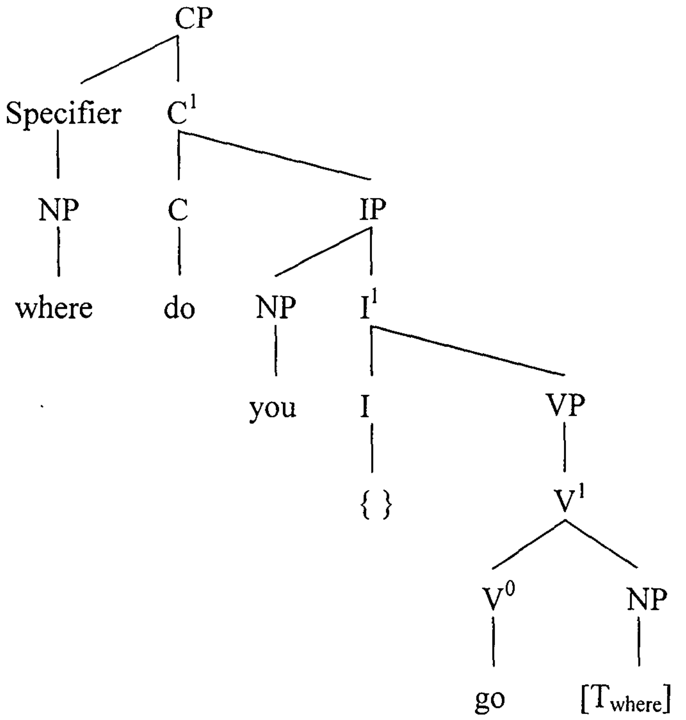


Figure 2.7 : Complement Phrase

2.3. LEXICON ORGANIZATION

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

2.3.1 Categorical Information

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

college: noun (N)
lecture: N
student: N
sleep: verb (V)
write: V
in: preposition (P)
slowly: adverb (Adv)
difficult: adjective (A)

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

2.3.2 Subcategorization Information

A **subcategorization frame** specifies in what syntactic environment a category a lexical item can be inserted. It is called "subcategorization" because we can distinguish subcategories (or subclasses) of categories on the basis of the context in which they appear, e.g.:

(1) a. sleep: V, [--] The boy is sleeping.
b. imitate: V, [-- NP] The boy is imitating [**the cat**].
c. give: V, [-- NP, PP] The boy gives [**the ball**] [**to the man**].
d. give: V, [-- NP, NP] The boy gives [**the man**] [**the ball**].

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

2.3.3 Thematic Information

As described in the section above, each lexical entry is specified for the number and type of arguments it requires. The **arguments** are the participants that are minimally involved in the activity or state expressed by the predicate, for instance: The verb imitate in (1)b requires two participants: one person who does the imitating (the **agent**), and someone who is being imitated (the **patient**). The verb gives in (1)c and d requires three participants: someone who does the giving (the **agent**), something that is given (the **theme**) and someone who receives the given entity (the **recipient**). These participants are called **arguments**, their roles (such as agent, theme, recipient) are called **thematic roles** (or **theta-roles, 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. sleep: V; [1] The boy is sleeping.
b. imitate: V; [1 2] The boy is imitating the cat.
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. sleep:	V; [1]		
	b. imitate:	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 below.

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

	Categorical Information	Subcategorization Information	Thematic Information
listen	v	[--PP] (listen to someone)	[1 2]
		[--] (listen)	[1]
send	v	[--NP] (send a letter)	[1 2]
		[--NP PP] (send a letter to some one)	[1 2 3]
		[--NP NP] (send someone a letter)	[1 2 3]

In the Machine Translation process the word lexicon is used more often than the word dictionary as this information is stored in a machine readable form. So the lexicon can be called as a "*computational dictionary*". In natural languages one word can have several forms and it is not wise enough to store all those forms in the lexicon as it simply increase the size of the lexicon. In MT, the lexicon usually contains the root words of the language and the morphological processes use the information present in the lexicon to generate the other forms of the words. Usually this information is stored in the files or as tables in the databases which can be easily retrieved for the computational purpose. However, the way you organize your data is more important as it would directly affect the speed of the computation.

2.4 THETA THEORY

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

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

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

Government:

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

1) A is a governor.
2) A m-commands B and
3) No barrier (some YP) intervenes between A and B

- Maximal projections are barriers to government.
- Governors are heads.

There are two types of *Governors:*

1) **Lexical governors**
    - E.g. V (verb), P (preposition), N (noun), A (adjective).
2) **Functional governors**
    - E.g. I (agreement), C (complementizer).

2.4.1 The Argument structure of other syntactic categories

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

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

Consider the examples:

The Romans destroyed the city.

The Romans' destruction of the city.

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

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

For example compare:

"The Romans destroyed the city." with "Destroyed the city." and

"The Romans' destruction of the city." with " the destruction of the city"

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

For example compare:

"Poirot will analyze the data." with "Poirot will analyze." and

"Poirot's analysis of the data." with "Poirot's analysis"

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

Poirot envies Bertie.

Poirot is envious of Bertie.

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

Peter is **in** London.

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

2.4.2 The Theta Criterion

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

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

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

2.4.3 The Projection Principle

We have said so far that the mental lexicon contains a lexical entry for each lexical item in the language. This lexical entry in turn contains phonological, morphological, semantic and syntactic information about that lexical item. The syntactic information contains 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 sates that: *"Lexical information is syntactically represented.*"

2.4.4 The Extended Projection Principle

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

So far we have presented in brief the requirements, according to the theory, the analysis and design of Lexicon and how the lexicon is to be organized, so that we can capture all the required syntactic information about lexical items of words in a given sentence. Detailed information about all these concepts can be found in Hageman [12] and Fromkin [18]. In the following section we are going to present some other important concepts like Case Theory and Binding Theory in brief.

2.5 OTHER CONCEPTS

2.5.1 Case Theory

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

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

In English, overt morphological realization of case in full lexical noun phrases is restricted to the GENITIVE case. NOMINATIVE and ACCUSATIVE case can only be seen on pronouns. Heads of some categories (specifically V, P, I, but not A and N) have case-assigning abilities. In English, Verbs assign ACCUSATIVE CASE to their complements, whereas nouns and adjectives don't. Further, the passive participles cannot assign ACCUSATIVE case to their complements and that therefore the internal argument moves to the subject position to receive NOMINATIVE case. This in turn is possible because the passive participle does not assign an external 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.

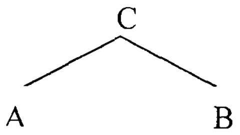
2.5.2 Binding Theory

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:

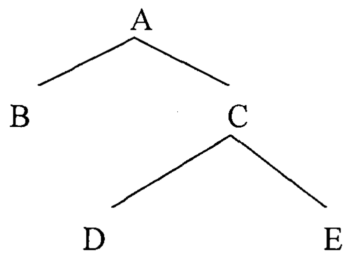
C-Command:

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.

2.6 BILINGUAL LEXICON

The lexicon that contains words and their attributes of two languages is called the '*bilingual lexicon*'. In machine translation these two languages are usually the source and the target languages. For a translation the bilingual lexicon serves as the main source for word to word mapping for source language to the target language. This will also contain the required information for both the source and the target language morphology. For each word in the source language the corresponding words in the target language may be stored in an organized manner as given in the Table 2.2 , so as to decrease the retrieval time. Note that the mapping between a source language word and target language words is one to many.

Table 2.2 : Structure for a Bilingual Lexicon.

Source Language Word (SLW)	Sub-categorization Information for the SLW	Argument Structure for SLW.	Equivalent Target Language Word (TLW)	Sub-categorization Information for the TLW	Argument Structure for TLW.
SW1	……	…..	TW11	……	……..
SW1	……..	…….	TW12	……..	…….
SW1	……..	…….	TW13	……..	……….
SW2	…….	…….	TW21	……..	……….
SW2	……..	……..	TW22	……..	……….
…					
…					
…					

Chapter 3

GOVERNMENT AND BINDING BASED GRAMMAR FOR SANSKRIT AND ENGLISH

3.1 INTRODUCTION

In this Chapter we are going to present the GB based phrase structure rules for the Sanskrit and English languages. We have seen in Chapter 2 that X-bar Theory can account for the phrase structure of lexical categories, sentences, and clauses of any language, using only the three basic rules represented below.

i) XP → Specifier X′

ii) X′ → X0 Complements

iii) X′ → X0 Adjuncts

In the following sections we represent the Phrase Structures for English (the Target Language) and Sanskrit (the Source Language)

3.2 ENGLISH PHRASE STRUCTURES

We can refer to English and other SVO languages as **head-initial** and **specifier-initial**, since the specifier comes before X′ and the head comes before its complements. This generalization holds in all phrases in English. For example, in sentences the subject is initial in the specifier position and the VP complement follows the head containing nonfinite *to* or the inflection features. Within the VP, the head V precedes the NP object complement. The various Phrase Structures for English have been described in detail in Chapter 2. For the sake of completeness these are repeated in the following. The rules forms of English Phrase Structures are given in Appendix B.

3.2.1 A Verb Phrase

A sample Verb Phrase Structure as shown in Figure 2.2, is repeated here in Figure 3.1.

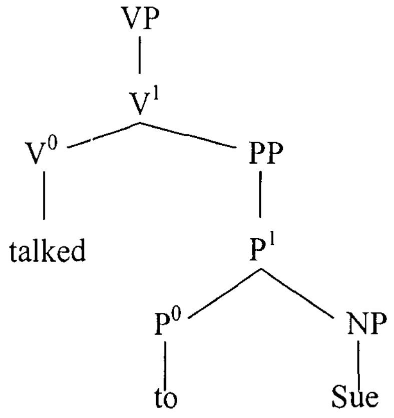


Figure 3.1 Verb Phrase

3.2.2 A Noun Phrase

A sample Noun Phrase Structure as shown in Figure 3.3, is repeated here in Figure 3.2.

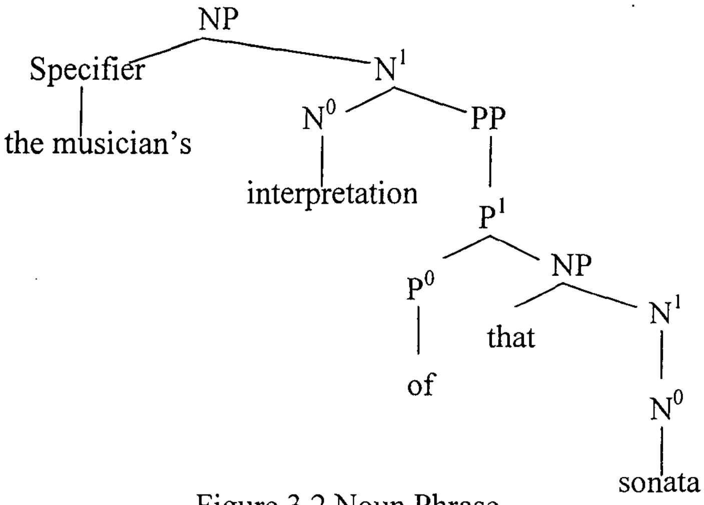


Figure 3.2 Noun Phrase

3.2.3 A Adjective Phrase

A sample Noun Phrase Structure as shown in Figure 2.4, is repeated here in Figure 3.3.

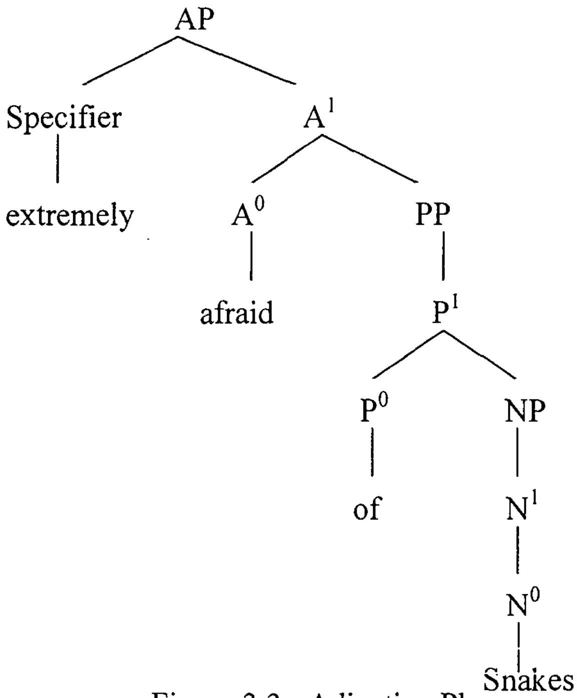


Figure 3.3 : Adjective Phrase

3.2.4 A Prepositional Phrase

A sample Noun Phrase Structure as shown in Figure 2.5 is repeated here in Figure 3.4.

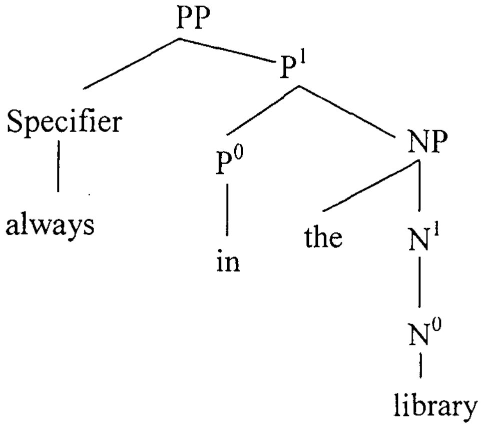


Figure 3.4 : Prepositional Phrase

3.2.5 An Inflection Phrase

A sample Noun Phrase Structure as shown in Figure 2.6 is repeated here in Figure 3.5.

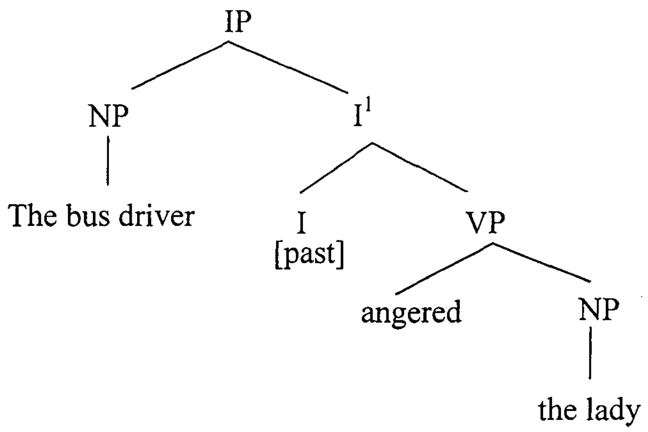


Figure 3.5 : Inflection Phrase

3.2.6 A Complementizer Phrase

A sample Noun Phrase Structure as shown in Figure 2.7 is repeated here in Figure 3.6.

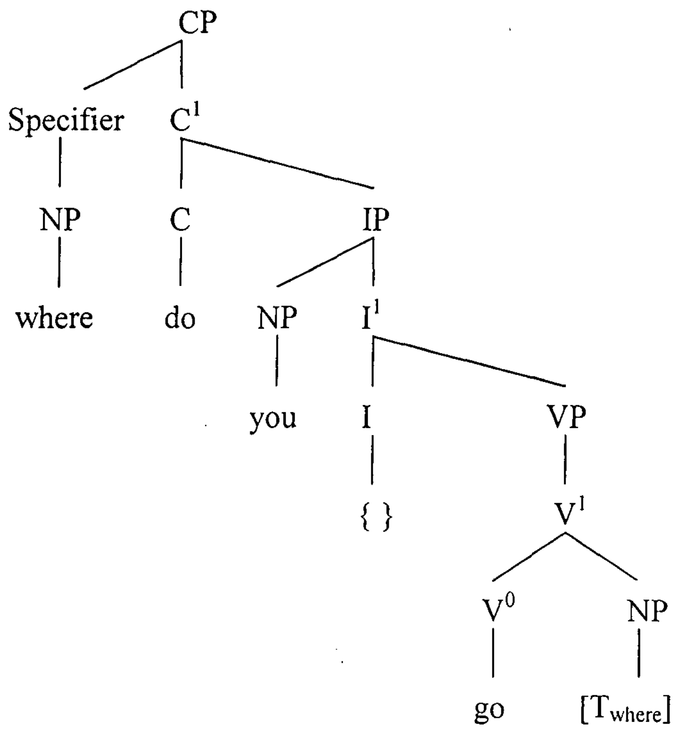


Figure 3.6 Complement Phrase

3.3 SANSKRIT PHRASE STRUCTURES

Sanskrit is a highly inflectional language and due to this property it is a completely word-order free language. To fit Sanskrit in the GB frame work we have superimposed a Hindi like word order on Sanskrit. So we assume that Sanskrit is a SOV language. SOV languages are **head-final** and **specifier-initial languages.** For this category of languages the basic phrase structure rules are:

XP → Specifier X′
X′ → Complements X0
X′ → Adjuncts X0

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

3.3.1 A Verb Phrase

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

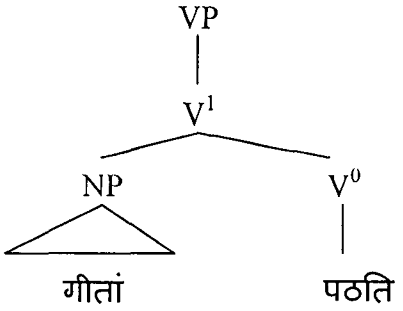


Figure 3.7 Verb Phrase

3.3.2 A Noun Phrase

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

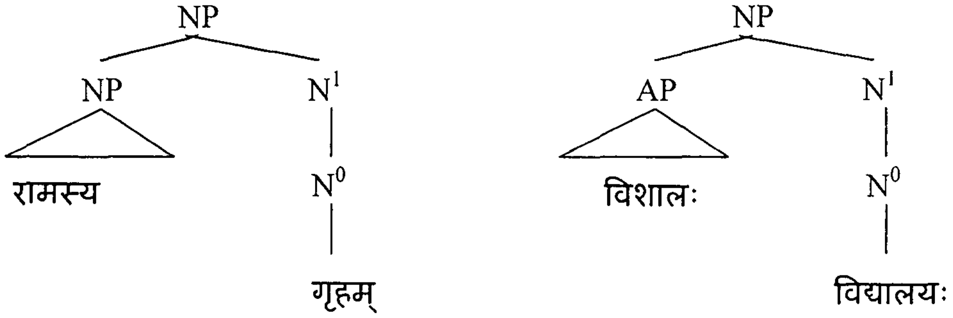


Figure 3.8 The Noun Phrases

3.3.3 An Inflection Phrase

The tree representation for the Sentence "रामः विद्यालयम् अगमम्" is shown in Figure 3.9.

In the example the specifier is an NP. However it could be an IP or a CP.

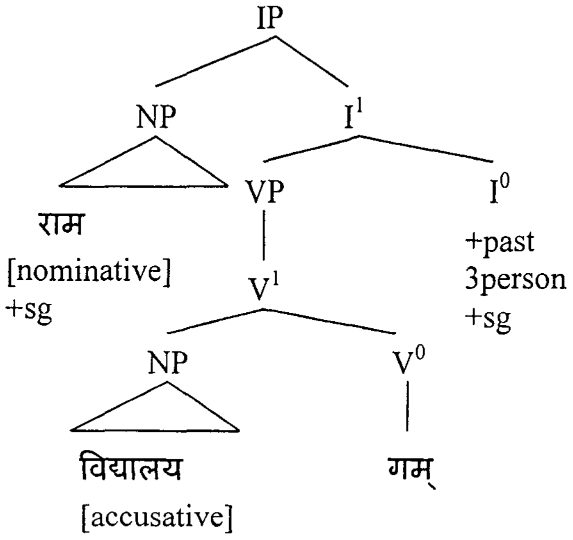


Figure 3.9 The Inflection Phrase

Chapter 4

SANSKRIT AND ENGLISH MORPHOLOGY

4.1 INTRODUCTION

In any language at word level two types of processes operate. The first of these processes is referred to as the Lexical process, where one type of lexical unit is converted into another type of lexical unit. For example adjective “good” is converted into the noun “goodness” by suffixing morpheme “ness” to “good”. The second of processes is referred to as the morphological process where a lexical type is not converted into other lexical type but its grammatical attributes are modified or changed. For example the verb “go”, which is the base form, is rendered as “going” in its present continuous form. The elements that are combining to form words are called morphemes. A morpheme is the smallest unit of meaning you can have in a language. In the above examples the morphemes present are “good”, “ness”, “go” and “ing”. Further, in the above examples “good” and “go” are free morphemes, having meanings and “ness” and “ing” are bound morphemes having no independent meaning attached to them, but they carry grammatical functions along with them. In the language words such as “good” and “go” are known as root words and words such as “goodness” and “going” are derived words. Therefore morphology refers to obtaining derived words from the root words and vice versa. The process of obtaining derived words from the root words is known as Forward Morphology and the process of obtaining the root words given the derived word is known as Reverse Morphology. The set of root words (free morphemes) in any language is an open set, where as set of bound morphemes is a closed set. The morphology plays an important role in analyzing the sentences in source language and also in the generation of target language sentences. We explain below in detail the morphology of Nouns, Adjectives and Verbs for source and target languages, that is, for Sanskrit and English.

4.2 MORPHOLOGY OF SANSKRIT LANGUAGE

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

In our domain of source as well as the Target Language, we consider eight cases of a Noun, these are: Nominative, Accusative, Instrumental, Dative, Ablative, Possessive, Locative and Vocative. In English, the cases are determined by the position of a Nominal Phrase in a sentence or by the presence of the preposition used with the nouns. However in Sanskrit language a noun, as stated above, is modified with a specific suffix for each case.

4.2.1 NOUN MORPHOLOGY IN SANSKRIT (DECLENSIONS OF NOUNS)

A nominal morpheme, referred to as Sup in Sanskrit, has 24 forms depending on the case and the number of the desired final form of the nominal Pada. Each of these 24 forms of Sup is further modified depending on the gender and the ending of the Pratipadik with which it has to be attached. Though most of the generation is regular in nature, in certain cases irregular processes also operate [Asthadhyayi]. Due the limitation of the space it is not possible here to give the full description of the Noun morphology. Therefore only some typical case(s) or example(s) are presented here. Table 3.1 gives the complete morphology of the Pratipadik राम. Table 3.2 gives the complete morphology of the

Pratipadik रमा. Similarly Table 3.3 gives the complete morphology of the Pratipadik ज्ञान. The root राम is अ ending masculine, the root रमा is आ ending feminine, and the root ज्ञान is अ ending neuter.

Table 4.1: Morphology of राम

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

Table 4.2: Morphology of रमा

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

Table 4.3: Morphology of ज्ञान

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

4.2.2 ADJECTIVE MORPHOLOGY

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

एतस्मात् मधुरात् फलात् अहं रसं प्राप्नोमि ।

मधुरां कथां श्रुत्वा, निद्रां करोति बालकः ।

मधुरस्य कृष्णस्य वाणी मधरा ।

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

4.2.3 VERB MORPHOLOGY (VERB CONJUGATION)

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

1.	लट्	Present tense
2.	लङ्	Past tense - imperfect
3.	लुङ्	Past tense - aorist
4.	लिट्	Past tense - perfect
5.	लुट्	Future tense - likely

6.	लृट्	Future tense – certain

Four Lakaras representing moods are:

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

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

A Pada of a verb form determines whether the activity specified in the verb applies to the person himself or whether it applies to someone other than the subject of the verb. Verbs referring to the activity for the self are said to be "**Atmanepada**" आत्मनेपद verbs. Verbs referring to the activity for others are said to be "**Parasmaipada**" परस्मैपद verbs. Verbs which can take both forms are known as "**Ubhayapada**" उभयपद verbs.

4.2.3.1 Simple Present Tense: There is only one form for the present tense (लट्). In simple present tense, we add suffixes to the root form of the verb before the terminations are added. The suffix आ is added in first person, while suffix अ is added in second and third persons. The terminations for the verbs in "Parasmaipada" are stated in Table 4.4:

Table 4.4

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

The different forms for the verb पठ् in present tense are given in Table 4.5:

Table 4.5

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

The terminations for the verbs in "Atmanepada" are stated in Table 4.6:

Table 4.6

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

4.2.3.2 Simple Past Tense: Past tense has three forms associated with it

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

For the simple past tense, अ is the prefix. The terminations for the verbs in "Parasmaipada" are stated below in Table 4.7:

Table 4.7

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

The forms for the past tense of the verb can be obtained from the above table. The root form of the verb is गच्छ् and the infix corresponding to the root is अ for the second and third person but आ for the first person. The different forms for the verb गच्छ् in past tense are given in Table 4.8:

Table 4.8

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

The terminations for the verbs in "Atmanepada" are stated in Table 4.9:

Table 4.9

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

4.2.3.3 Simple Future Tense: Future tense has two forms associated with it.

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

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

गम्, गच्छ् are the two root forms for गच्छति

पा, पिब् are the two root forms for पिबति

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

The terminations for the future tense in "Parasmaipada" are given in Table 4.10:

Table 4.10

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

The forms for the verb गच्छति in the future tense are given in Table 4.11:

Table 4.11

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

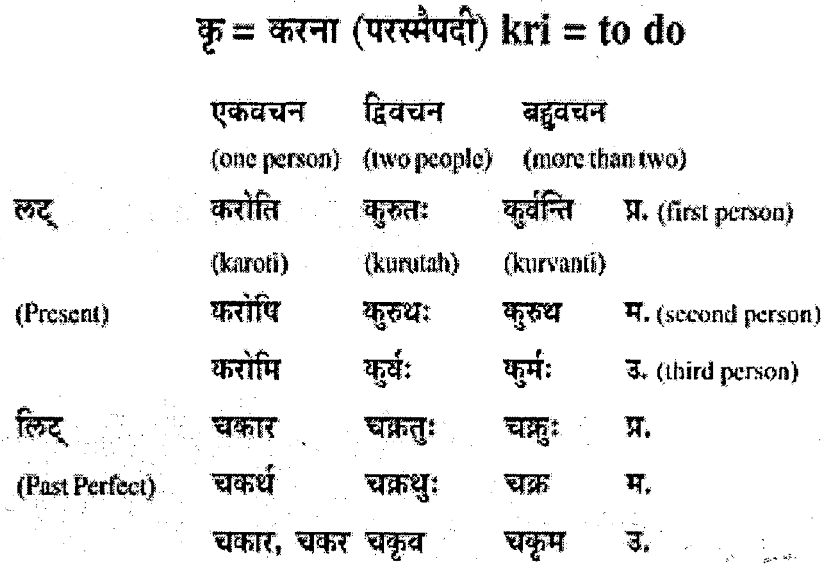
The terminations for the future tense in “Atmanepada” are given in Table 4.12:

Table 4.12

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

In our work we have considered only Present tense, past tense – Aorist, and Future tense – certain. Parasmaipada conjugation of verb “kri” is given in the Table 4.13.

Table 4.13: Conjugation of Verb “kri”



कृ = करना (परस्मैपदी) kri = to do

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

लुट्	कर्ता	कर्तारौ	कर्तारः	प्र.
(First future)	कर्तासि	कर्तास्थः	कर्तास्थ	म.
	कर्तास्मि	कर्तास्वः	कर्तास्मः	उ.
लृट्	करिष्यति	करिष्यतः	करिष्यन्ति	प्र.
(Simple future)	करिष्यसि	करिष्यथः	करिष्यथ	म.
	करिष्यामि	करिष्यावः	करिष्यामः	उ.
लोट्	करोतु,कुरुतात्	कुरुताम्	कुर्वन्तु	प्र.
(Imperative mood)	कुरु, कुरुतात्	कुरुतम्	कुरुत	म.
	करवाणि	करवाव	करवाम	उ.
लङ्	अकरोत्	अकुरुताम्	अकुर्वन्	प्र.
(Past imperfect)	अकरोः	अकुरुतम्	अकुरुत	म.
	अकरवम्	अकुर्व	अकुर्म	उ.
वि. लिं.	कुर्यात्	कुर्याताम्	कुर्युः	प्र.
(Potential mood)	कुर्याः	कुर्यातम्	कुर्यात	म.
	कुर्याम्	कुर्याव	कुर्याम	उ.
आ. लिं.	क्रियात्	क्रियास्ताम्	क्रियासुः	प्र.
(Benedictive)	क्रियाः	क्रियास्तम्	क्रियास्त	म.
	क्रियासम्	क्रियास्व	क्रियास्म	उ.
लुङ्	अकार्षीत्	अकार्ष्टाम्	अकार्षुः	प्र.
(Aorist)	अकार्षीः	अकार्ष्टम्	अकार्ष्ट	म.
	अकार्षम्	अकार्ष्व	अलार्ष्म	उ.
लृङ्	अकरिष्यत्	अकरिष्यताम्	अकरिष्यन्	प्र.
(Conditional)	अकरिष्यः	अकरिष्यतम्	अकरिष्यत	म.
	अकरिष्यम्	अकरिष्याव	अकरिष्याम	उ.

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

4.3. MORPHOLOGY FOR ENGLISH

As English is a highly used and well known language we just try to explain in brief some morphological rules that we have implemented in our translation system. Our English morphology mainly deals with the nouns and the verbs. As our system translates only simple sentences we are not implementing the entire English morphology. Except verbs and nouns, all the other types of words are assumed to be in their root form. For English, the nouns morphology is limited to only number transformation (singular, plural). The morphology for verbs also includes the auxiliaries. The following sub sections describe in brief the English morphology for Nouns and Verbs.

4.3.1 Noun Morphology

Plurals for English nouns are formed by adding –s to the root noun, except in the following cases:

- When a noun ends in -ch, -s, -sh, -ss or -x the plural is formed by adding -es to the root.
  E.g. benches, gases, dishes, crosses, taxes.

- When a noun ends in -y preceded by a consonant the plural form is formed by adding -ies to the root.
  E.g. parties, bodies, policies

- When a noun ends in -y preceded by a vowel the plural is formed by adding -s to the root.
  E.g. trays, joys, keys

- When a noun ends in -o then the more common plural ending is -oes
  E.g. tomatoes, potatoes, zeroes, heroes.

- But in less familiar nouns or when the final -o preceded by a vowel the plural ending is -os
  E.g. avocados, armadillos, studios, cameos

- When a noun ends in -f the plural is formed either by adding -s
  E.g. beliefs, cuffs, whiffs
  or by changing the -f to -v and adding –es
  E.g. wives, thieves, loves.

  Some nouns may take both forms, E.g. scarf, wharf
  In the cases where both endings can be used, the -ves ending is usually the more formal one. The -s ending is more "casual" and is a bit more of a "slang" form.

- When a noun ends in -ex or -ix the more formal plural ending is -ices. In more general contexts -es is used
  E.g. appendices, appendixes, indices, indexes

- When a noun form Latin ends in -is the plural form is obtained by adding -es
  E.g. egcrises, analyses

- When a noun form Latin ends in -us the plural form is obtained by adding -i
  E.g. succubi, nuclei, syllabi, radii
  exception: viruses

- In a compound noun (like court-martial), it is usually the most important part which is pluralized
  E.g. courts-martial, lord-justices, mothers-in-law

- In certain cases the plural form of a noun is the same as the singular form.
  E.g. deer, sheep, grouse
  and for some nouns both forms end in -s

E.g. measles, corps, mews

4.3.2 Verb Morphology

The morphology of English verb depends on Tense, Aspect, Person, and Number it represents in a sentence used. English has three tenses, present, past, and future and four aspects, simple, continuous, perfect, and perfect continuous. It has three persons, first, second and third and two numbers, singular and plural. English verbs do not change with the gender. English also uses Auxiliaries (helping verbs) along with verbs and are conjugated like main verbs. English auxiliaries are placed before the main verb.

Our translation system limits the translation to only active voices. We do not explain how the verb changes accordingly in passive voice. The Table 3.8 shows how the verb and the auxiliary change according to the tense and aspect. We may further note that the changes in the final verb forms due to the addition of morphological suffixes follow the same pattern as described in Section 4.3.1 under Noun Morphology.

4.3.2.1 Simple Present Tense

In simple present the verb will be in its base form except for the third person singular. The verb for the third singular are formed by adding -s to the base form except for the following:

- If the base form verb ends with -s, -h, -x, -z, -o then its simple present is formed by adding -es its base form. Eg kisses, washes, teaches, mixes, buzzes, goes, etc.

- If the base form verb ends with -y and if there is a consonant before it then its simple present is formed by removing -y and adding -ies. Eg. tries

- If the base form verb ends with -y and if there is a vowel before it then its simple present is formed by adding -s. Eg. Plays

4.3.2.2 Perfect

The verb in present perfect, past perfect and future perfect take its simple past form. However the auxiliary before the verb in these three differs. For example look at how the auxiliary differs in the verb 'learn' in the example below

E.g.	Present perfect	have learned
	Past perfect	had learned
	Future perfect	will have learned

4.3.2.3 Continuous and Perfect Continuous

The verbs in continuous (present, past, future) and perfect continuous (present, past, future) forms are obtained by adding -ing to their base form except for the following:

- If the base form of verb ends with -ve and then its continuous form is obtained by removing -e at the end and then adding -ing. Eg. believing

- If the base form of the verb ends with -t and then its continuous form is obtained by first adding -t and then adding -ing. E.g. setting

4.3.2.4 Simple Past Tense

The simple past form of the verb is formed by adding -ed to its base form except for the following

- If the base form verb ends with -y and if there is a consonant before it then its simple past is formed by removing -y and adding -ied. E.g. tried, denied, etc.

- If the base form verb ends with -y and if there is a vowel before it then its simple present is formed by adding -s. E.g. Plays.

- If the base form verb ends with -e then its past form is obtained by adding –d to it. E.g. moved, dyed, etc

The Table 4.14 shows the auxiliaries these 12 tenses take.

Table 4.14

	First singular	**Second singular**	**Third singular**	**First plural**	**Second plural**	**Third plural**
Simple present						
Present continuous	am	are	is	are	are	are
Present perfect	have	have	has	have	have	have
Present perfect continuous	have been	have been	has been	have been	have been	have been
Simple past						
Past continuous	was	were	was	were	were	were
Past perfect	had	had	had	had	had	had
Past perfect continuous	had been	had been	had been	had been	had been	had been
Simple future	will/shall	will/shall	will/shall	will/shall	will/shall	will/shall
Simple continuous	will be	will be	will be	will be	will be	will be
Future perfect	will have	will have	will have	will have	will have	will have
Future perfect continuous	will have been	will have been	will have been	will have been	will have been	will have been

The verb forms which are not obtained by employing the morphological process described above are referred to as irregular forms. Unlike regular forms, irregular verb forms have to be stored in the Lexicon. For example, irregular form "went" of verb "go" has to be stored in the Lexicon while regular form "going" is generated using morphological processes.

4.4 INTERFACING MORPHOLOGY WITH LEXICON

The morphological processes relate to obtaining the final word form (in generation) given its root form or obtaining the root form given its final form (in parsing). For obtaining the final form of a word, morphology involves deleting certain end portion of the root word and thereafter adding a suffix depending upon the grammatical attributes of the final form of the word. The process is reversed while obtaining the root form along with the attributes given the final form of the word. We, therefore, require an auxiliary Lexicon where we only store for each grammatical category attributes (like tense, aspect, etc.), the end string to be deleted from the root word and the string to be added to thereafter. This process needs creating an Auxiliary Data Base and associated processes.

Chapter 5

PARSING STRATEGIES – AN OVERVIEW

5.1 INTRODUCTION

The present section of 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 (1956), has enjoyed wide use in a variety of fields. CFGs have been used to model the structure of Programming languages and Natural languages (Aho, Sethi, and Ullman, 1986; Jurafsky and Martin, 2000; Durbin et al., 1998). Canonical methods for general CFG parsing are the CKY algorithm (Kasami, 1965; Younger, 1967) and Earley's algorithm (Earley, 1970). Both have a worst-case running time of O(gn^3) for a CFG of size g and string of length n (Graham, Harrison, and Ruzzo, 1980), 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 (1980) give a variant of Earley's algorithm that is based on the so-called 'four Russians' algorithm (Arlazarov et al., 1970) for Boolean matrix multiplication (BMM); it runs in time O(gn3/log n). Rytter (1985) further modifies this parser by a compression technique, improving the dependence on the string length to O(n3/log2 n). But Valiant's (1975) 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 X m Boolean matrices together. In the section we are going to explain the bottom-up parsers.

5.2 BOTTOM-UP PARSER (LR PARSING ALGORITHM)

Now we will describe the bottom-up parsing method. In bottom-up parsing we have various parsing algorithms like Shift-Reduce parsing, SLR, CLR and LALR. But we will concentrate on LR parsing algorithm.

5.2.1 Overview

The basic idea of a bottom-up parser is that we use grammar productions in the opposite way (from right to left). We use a stack to push symbols. If the first few symbols at the top of the stack match the RHS 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. If there is no handle on the top of the stack, we push one more terminal in the stack from the input stream and check again for a handle. This is called shifting. There two actions only: shift the current input token in the stack and read the next token, and 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. So we will construct a parsing table, by which we can easily take a decision each time when we scan an input symbol, weather to use shift action or to use reduce action.

5.2.2 Shift-Reduce Parsing Using the ACTION/GOTO Tables

As we said in the previous section that, we will precompute the parsing table and is given as input to the parser. This precomputed table will help the parser to take quick decision on each input word, thus improves the execution time. The other inputs to the parser are the lexical all words with attributes of an input sentence and the GB phrase rules. Let us now see the process of constructing the parse table with the help of an example.

Consider the following grammar:

0) [S] → [R] [$]

1) [R] → [R] [b]

2) [R] → [a]

which parses the R.E. *ab**$.

The DFA that recognizes the handles for this grammar is given in Figure 2.6 below.

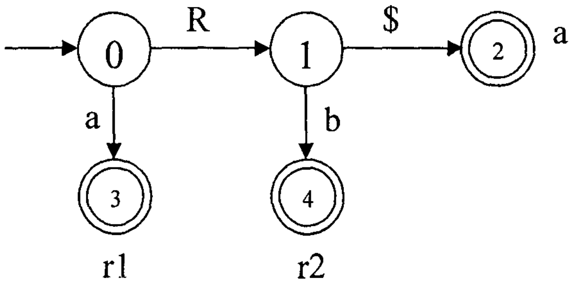


Figure 2.1 DFA

where 'r2' means reduce by rule 2 (ie, by [R] → [a]) and 'a' means accept. The transition from 0 to 3 is done when the current state is 0 and the current input token is 'a'. If we are in state 0 and have completed a reduction by some rule for R (either rule 1 or 2), then we jump to state 1.

The ACTION and GOTO tables that correspond to this DFA are:

state	action			goto	
	a	b	$	S	R
0	s3				1
1		s4	s2		
2	a	a	a		
3	r2	r2	r2		
4	r3	r3	r3		

where for example s3 means shift a token into the stack and go to state 3. That is, transitions over terminals become shifts in the ACTION table while transitions over non-terminals are used in the GOTO table.

Now let us consider the parser operations on stack with the help of the above parsing table are given below, with the help of an example. For example, for the input abb$, we have:

Stack	rest-of-input	Action
0	abb$	s3
0 3	bb$	r2 (pop(), push GOTO[0,R] since [R] → [a])
0 1	bb$	s4
0 1 4	b$	r1 (pop(), pop(), push GOTO[0,R] since [R] → [R] [b])
0 1	b$	s4
0 1 4	$	r1 (pop(), pop(), push GOTO[0,R] since [R] → [R] [b])
0 1	$	s2
0 1 2		accept

So far we have described the features of LR parsing Algorithm, which is nothing but rule-based parser. Now we will see the features of rule-based parser in brief.

5.3 RULE-BASED NATURAL LANGUAGE PARSER

The output of rule-based system is a syntactic feature structure corresponding to the input sentence. Providing coverage for casual language with conversational features requires that a grammar should handle syntactic constructions found in 'standard' language, as well as the many variations found mostly in spoken language. By increasing grammar coverage with the addition of rules, more ambiguity is introduced, causing the search for the correct analysis to be more difficult. This tradeoff between recall and precision was addressed in the rule-based system with a constraint relaxation approach, supported by robustness features of the parser, and implemented as a multi-pass parsing strategy.

The rule-based parser that suffers from coverage limitations that can result in complete parsing failure. A key observation is that while the rule-based parser can occasionally fail to parse, the analyses found when parsing succeeds are of high precision, Kenji Sagae and Alon Lavie [Carnegie Mellon University].

Since our main focus is to design a parser, which is working for all kinds sentences. After studying various parsing approaches, we decided to implement the LR Parsing algorithm using Tomita's approach, the one which supports ambiguous CFG's and which is easy for implementation. The LR parsing approach uses bottom-up approach. In the next section we are going to explain the parser structure, which is used in our translation process, in brief, with the help of an example.

5.4 EXAMPLE

We will now show parse trees for the Sanskrit sentence "अहम् विद्यालयम् गच्छामि".

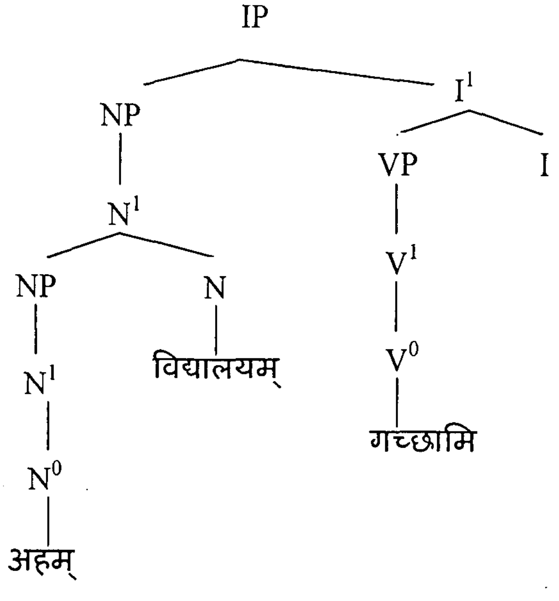


Figure 5.2 Parse tree for the Sanskrit sentence "अहम् विद्यालयम् गच्छामि".

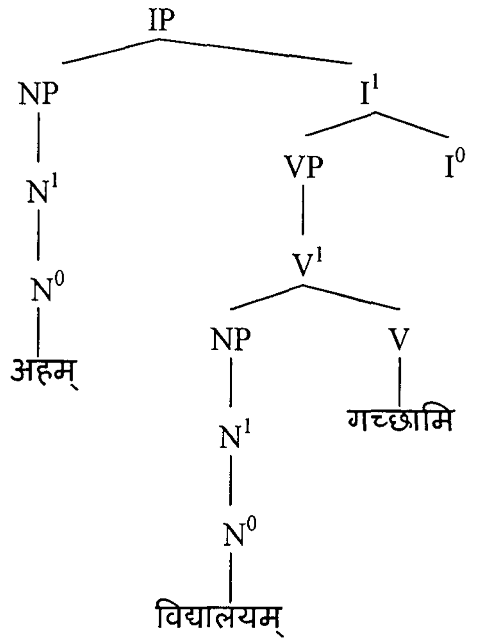


Figure 5.3 Parse tree for the Sanskrit sentence "अहम् विद्यालयम् गच्छामि".

The output tree given in Figure 5.2 will be filtered out if Verb "गम् "is taken as transitive. The output tree given in Figure 5.3 will be filtered out if Verb "गम् "is taken as intransitive. The outputs given in Figure 5.2 and in Figure 5.3 are both valid if "गम् "is taken as intransitive as well as transitive.

Chapter 6

DESIGN OF THE TRANSLATION SYSTEM

6.1. INTRODUCTION

The analysis of the problem and its possible solutions led to the following design of the system. The overall design of the translation system, in terms of the main modules and the inter-connection between them is shown in Figure 6.1. At the top level the Translation System consists of two modules namely, Parsing Module and the Generating Module. The Parsing Module consists of 4 modules namely, Input Module, Preprocessor, Tagger Module consisting of The Lexicon and Morphological Analyzer as sub modules, and Parser Module. The Generating Module consists of 3 modules namely, Phrase Mapper, Morphological Synthesizer, and the Output Generation Module. The description of each of these modules and sub-modules is given in following sections.

6.2 PARSING MODULE

As described above, the parsing module consists of the following sub modules: Input Module, Preprocessor (needed in case of Sanskrit only), Tagger Module, and Parser Module. These modules are described in following sub-sections.

6.2.1 Input Module

The translation system is made web enabled using the Client-Server technology, where the input given by the client system is transferred to the server for translation. The user is provided with a textbox where he can give the input sentence and a submit button. When the submit button is pressed the input sentence reading module reads the input and sends it to the server which executes the Translator.

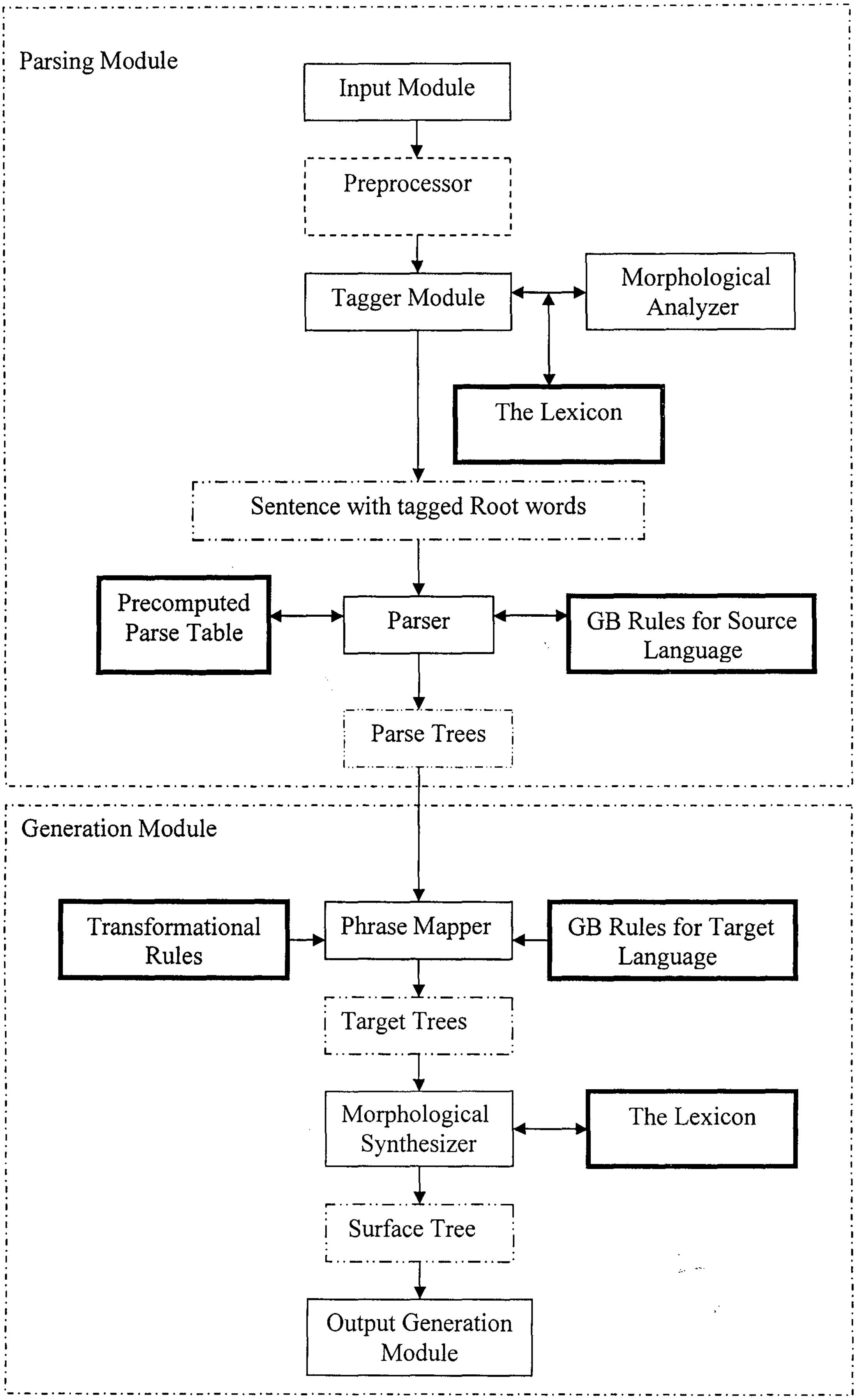

Parsing Module
Input Module
Preprocessor
Tagger Module
Morphological Analyzer
The Lexicon
Sentence with tagged Root words
Precomputed Parse Table
Parser
GB Rules for Source Language
Parse Trees
Generation Module
Transformational Rules
Phrase Mapper
GB Rules for Target Language
Target Trees
Morphological Synthesizer
The Lexicon
Surface Tree
Output Generation Module

Figure 6.1

6.2.2 Preprocessor

A part of the Preprocessor module converts a Sanskrit sentence into a SOV form. Since this conversion is done manually in our system, this module has really not been implemented. The other part of the Preprocessor module removes the string consisting of auxiliaries from the sentences, collects the associated attributes to be used in further processing. The module also removes redundant spaces, if existing between words in the given sentence. In the Figure 6.1, 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.

6.2.3 Tagger

The normalized input (the output of the Preprocessor) is subdivided into lexical items. Lexical items are not necessarily single words. More than one word in the input sentence may form a lexical item (e.g. Give up, put off, etc). 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. Once the lexical items are obtained, the next task of the Tagger is to obtain the category and subcategory information for each of these items. As the mapping between the word and the Lexical items is one to many the Tagger, in general produces more than one array of Lexical items. The Tagger uses the Morphological Analyzer and the Lexicon to obtain the category and subcategory information for the lexical items. The Morphological Analyzer and Lexicon are described in the following subsections.

6.2.3.1 The Lexicon

One of the important tasks in the translation process is development of a bilingual Lexicon treating Sanskrit as the source language and English target language. The structure of the Lexicon has been described in Table 2.2.

We may note that a given word may carry different category and subcategory information both in the source as well as in the target language. Further there may not be exact matching of attributes from the source language to the target language. For e.g. the nouns in Sanskrit language has three numbers (singular, dual, plural) and three genders (masculine, feminine and neutral) whereas English language has only two numbers (singular and plural) and two genders (masculine and feminine).

6.2.3.2 Morphological Analyzer

In Chapter 4, we have discussed the importance of the morphological processes in analyzing the input sentences of languages like Sanskrit, and in English. For each of the lexical items the category and subcategory information is found as follows:

The lexical item is first searched in the lexicon. If the lexical item is present in its root form then we directly obtain the required information from the lexicon. Otherwise, the morphology is to be done. The morphological rules differ for nouns, verbs, adjectives etc. So, when a word is sent to morphological analyzer it first assumes the word to be a noun and performs the noun morphology. If any of the rules is applicable, the rule is applied and the resultant word is searched in the lexicon. If the search is successful the module returns that the word is a noun and its root and various other attributes. If the noun morphology fails, we do the verb morphology, and then finally adjective morphology. If the morphology module is unsuccessful, the system specifies that the morphology module is unsuccessful, an error message is displayed and then it provides a provision for the user to enter the information himself. Since the process involves obtaining the root and their attributes for the lexical items, the morphology module is known as Morphological Analyzer. The output of this module is becomes the input to the Parser.

6.2.4 Parser

Once the attributes are obtained for each one of the words in the sentence , we check for the validity of the sentence i.e. whether the sentence is grammatical or not. The sentence is said to be grammatical if it follows GB Phrasal rules i.e. the computational grammar specification for the source language.

We have constructed an LR Parser using Tomita's approach, 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.

6.3 GENERATING MODULE

As described above, the Generating module consists of the following sub modules: Phrase Mapper, Morphological Synthesizer, and the Output Generation Module. These modules are described in following sub-sections.

6.3.1 Phrase Mapper

Once the source language sentence is found to be grammatical by the Parser, the next step in the translation process is the generation of sentence in target language. The Phrase Mapper receives parse trees generated by the parser module as input. The task of this module is to map each phrase generated by the parser to an equivalent phrase in the target language checking their grammatical compatibility while combining various phrases. This modification is necessary if the source and target languages have different word order as seen in Chapter 3. Since English has different word order from that of Sanskrit, this module maps the source language phrase structures to the target language phrase structures with the help of transformational rules employing GB grammars for Source as well as the target language. The result of this module is generation of parse trees in the target language.

Let us see how the Pharase Mapper works. At sentence level, the Phrase Mapper maps the phrase as shown in Figure 6.2a to a phrase shown in Figure in 6.2b. Then the Verb Phrase shown in Figure 6.3a is transformed into phrase shown in Figure 6.3b, where XP and XP1 represent the Complement and YP represent the Adjunct. The change of XP into XP1 is governed by the subcategorization entries in the Lexicon of the head verb for the Source and the Target language. For example the verb "go" subcategorizes for a PP complement where as the equivalent "गम्" subcategorizes for a NP complement, i.e. in the Target Language the PP complement is replaced by an NP complement. Depending upon the attributes under node I, in figure 6.2 the equivalent of root V (in the Source language) a V1 is the final form (or the Pada) in the Target language. This completes the translation up to the Verb Phrase level with phrases XP1 and YP to be translated next. Note three main actions of this process. The first, the rearrangement of phrasal components, the second, converting a phrase like XP into a phrase XP1, and the third, generating the final form of the head word. This process is recursive and continuous till all the phrases have been taken care of with the head words translated into their final forms. The last thing which has to be mentioned is that if it was a translation from Sanskrit to English then depending upon the attributes under node I, appropriate auxiliaries have to be generated and placed under node I.

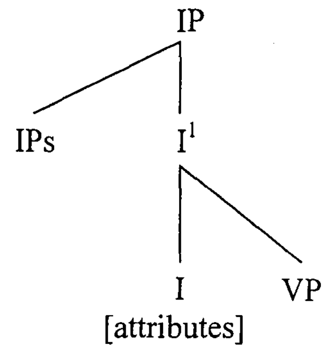


Figure 6.2a

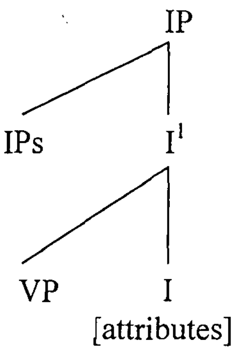


Figure 6.2b

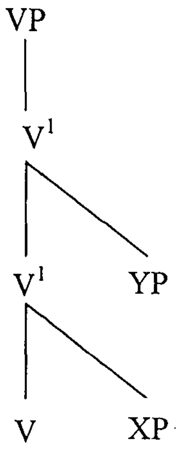


Figure 6.3a

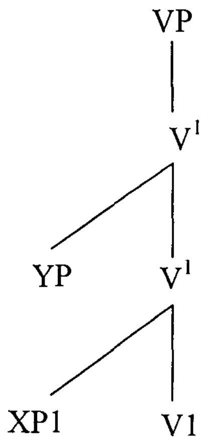


Figure 6.3b

6.3.2 Morphological Synthesizer

Now, that we have the phrase structures for the target language, our next step are to find equivalent word for each lexical item in the target language satisfying attributes. We first search the lexicon for the equivalent root in the target language. Once the root word is obtained from the lexicon, with the help of attributes, the correct form of the word in the target language is generated. Since the process involves in going from the root to the form, the morphology module here is known as Morphological Synthesizer.

6.3.3 Output Generation Module

Once the morphology is done for the words in the target language the output generation module generates the linear form of the target sentence by traversing the target trees in pre-order. Once the linear output is generated the server sends back the translated text to the client and thus the translation process is complete.

6.4 EXAMPLE

In this section we explain the working of the translator with the help of an example. Let the Sanskrit sentence presented to the translator be:

गजः   जलं   पिबति ।

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 sentence into a normalized form as shown below:

गजः जलं पिबति ।

Next, the normalized sentence is sent to Tagger. The Tagger first divides the sentence into lexical items गजः, जलं, and पिबति. Then it obtains from the Lexicon (using morphological analyzer if needed) the root form of each lexical item along with the corresponding attributes as shown below:

गजः (गज, singular, masculine, third person, nominative),

जलं (जल, singular, neuter, accusative),

पिबति (पिब्, singular, third person, simple present)

This sentence in the form of root words and associated attributes is then sent to the Parser. The parser generates a parse tree as given in Figure 6.4. Note that in general the Parser will generate more than one parse tree for a given sentence.

The parse tree generated by the Parser is given as an input to the Generator, which converts it in to an output tree. The process of conversion, as described under Section 6.3.1 is shown in Figures 6.5a to 6.5f.

The final output (i.e. given in Figure 6.5f) is given as input to the Output Module, which generates the output as given below:

Elephant drinks water

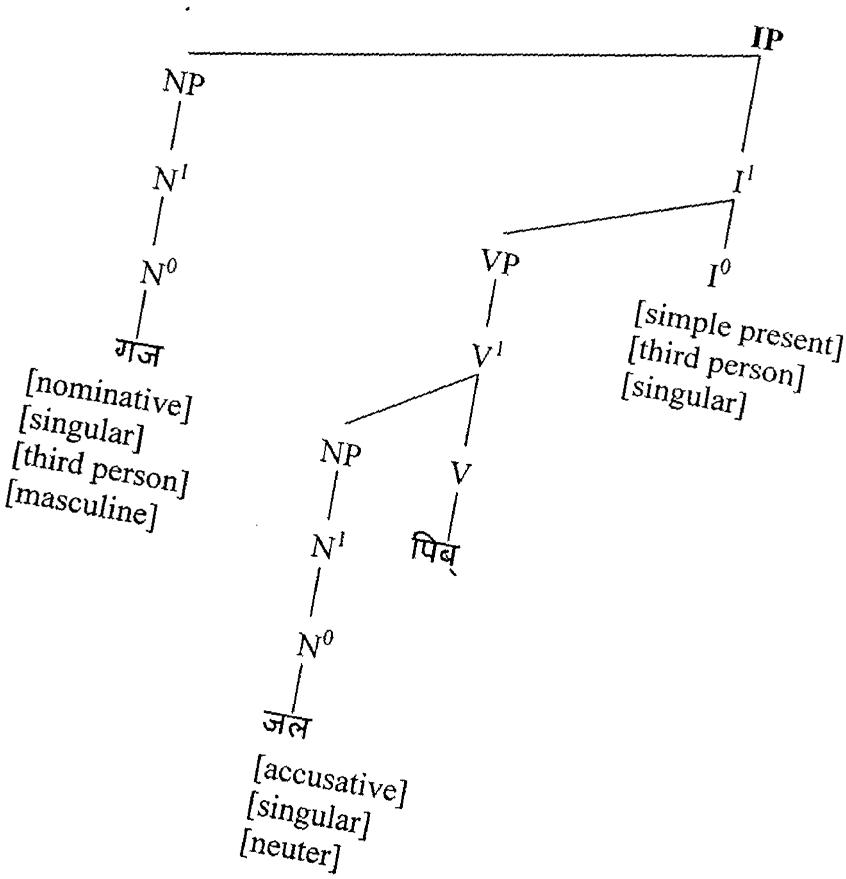

IP
NP
N^1
N^0
गज
[nominative]
[singular]
[third person]
[masculine]
I^1
VP
I^0
[simple present]
[third person]
[singular]
V^1
NP
V
N^1
पिब्
N^0
जल
[accusative]
[singular]
[neuter]

Figure 6.5a

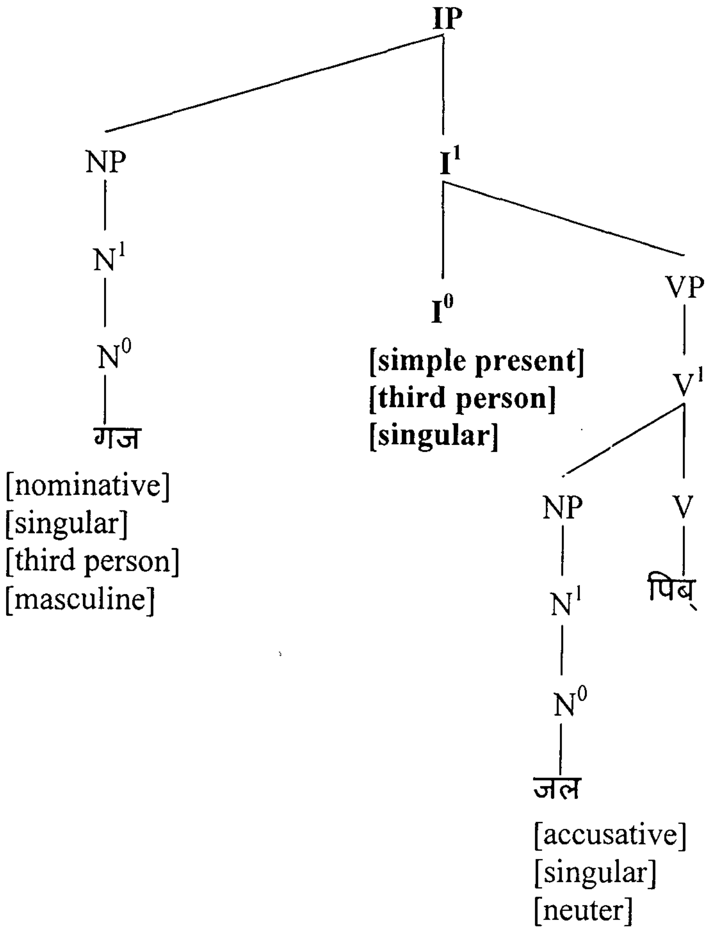

IP
NP
N¹
N⁰
गज
[nominative]
[singular]
[third person]
[masculine]
I¹
I⁰
[simple present]
[third person]
[singular]
VP
V¹
NP
N¹
N⁰
जल
[accusative]
[singular]
[neuter]
V
पिब्

Figure 6.5b

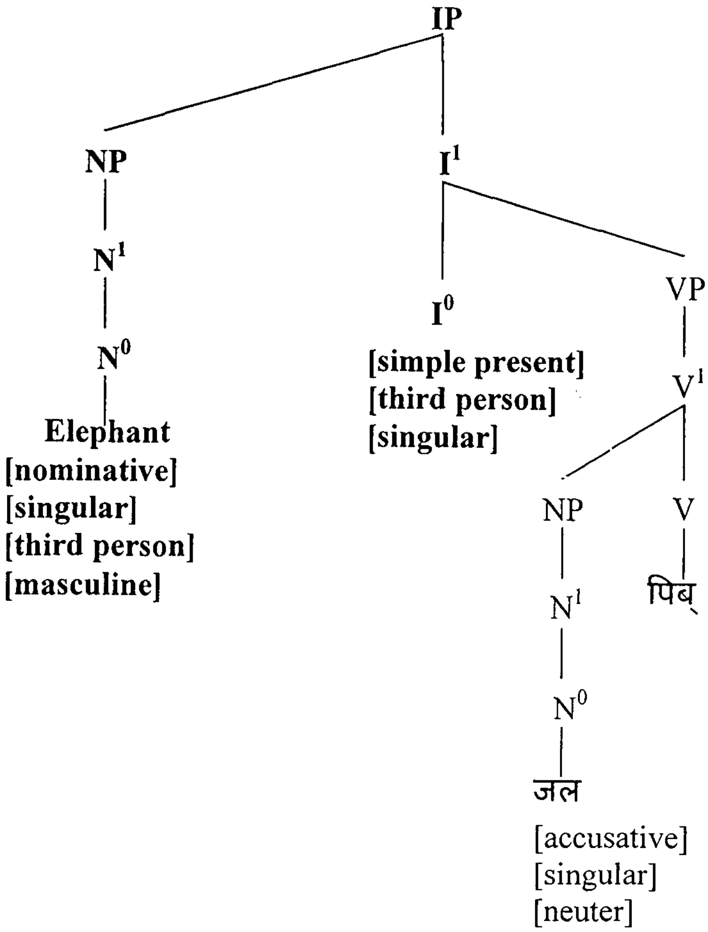

IP
NP
I1
N1
I0
VP
N0
[simple present]
[third person]
[singular]
V1
Elephant
[nominative]
[singular]
[third person]
[masculine]
NP
V
N1
पिब्
N0
जल
[accusative]
[singular]
[neuter]

Figure 6.5c

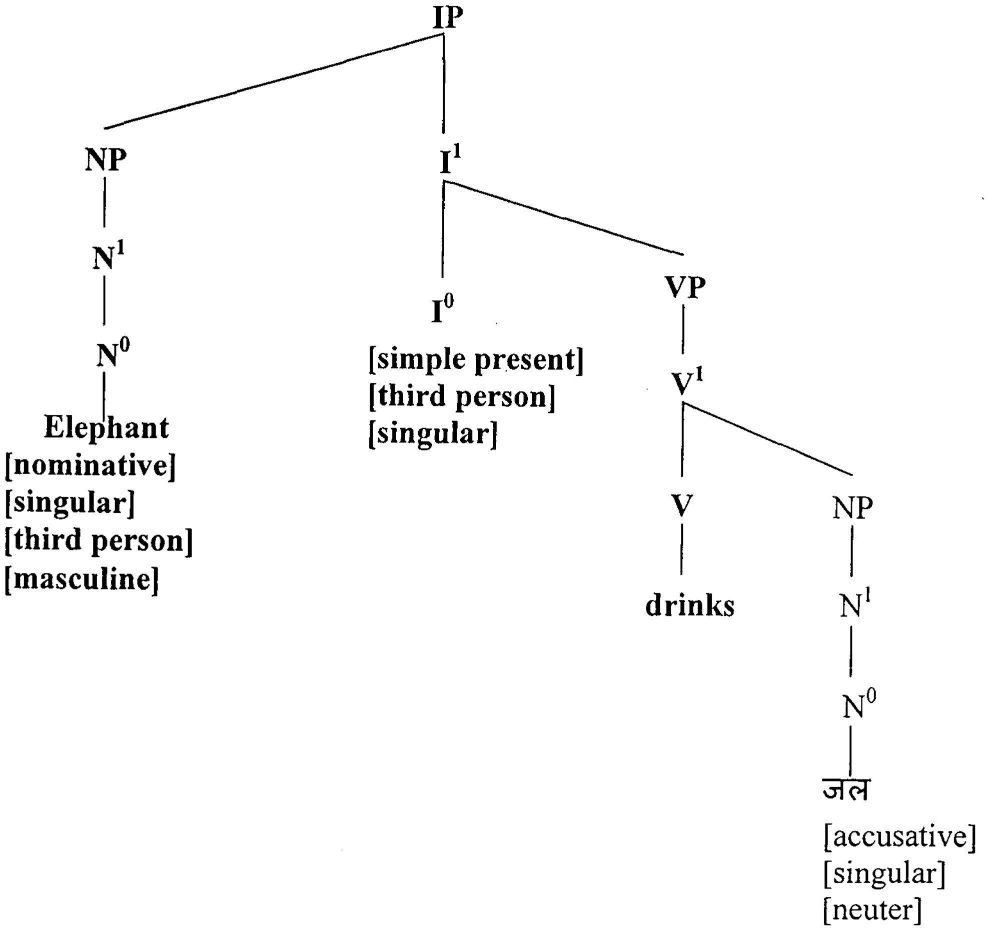

IP
NP
I1
N1
I0
VP
N0
[simple present]
[third person]
[singular]
V1
Elephant
[nominative]
[singular]
[third person]
[masculine]
V
NP
drinks
N1
N0
जल
[accusative]
[singular]
[neuter]

Figure 6.5d

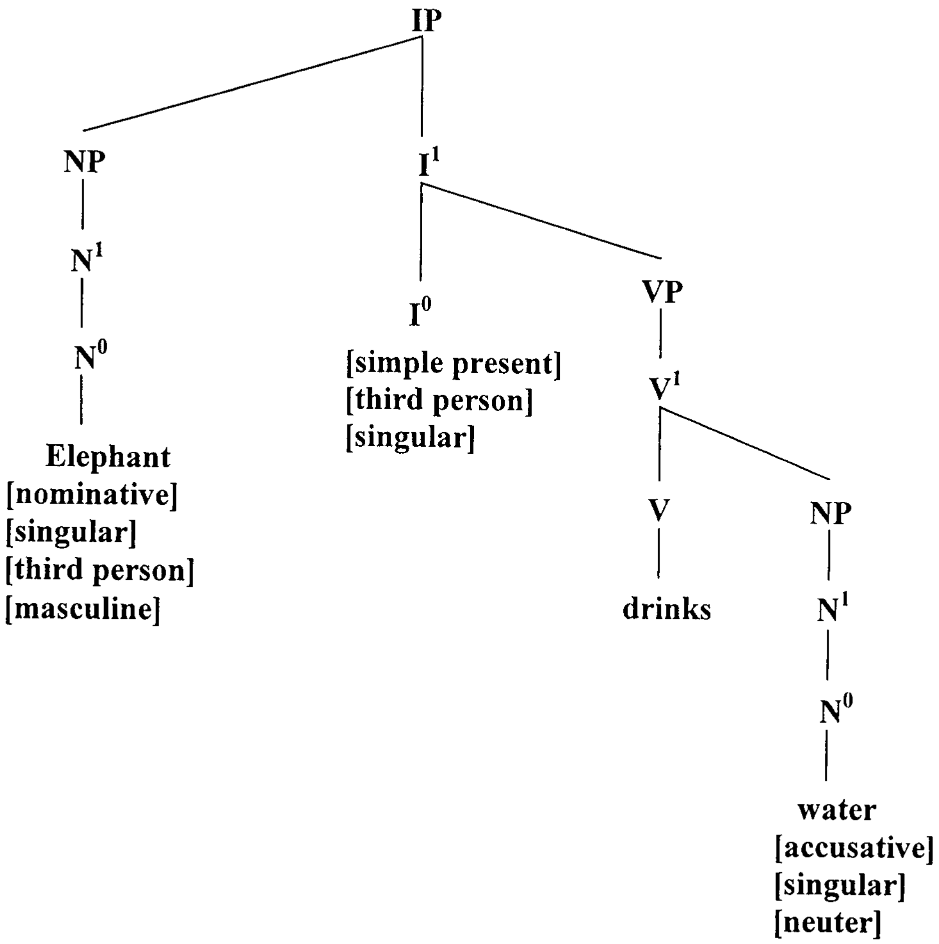

IP
NP
I1
N1
I0
VP
N0
[simple present]
[third person]
[singular]
V1
Elephant
[nominative]
[singular]
[third person]
[masculine]
V
NP
drinks
N1
N0
water
[accusative]
[singular]
[neuter]

Figure 6.5e

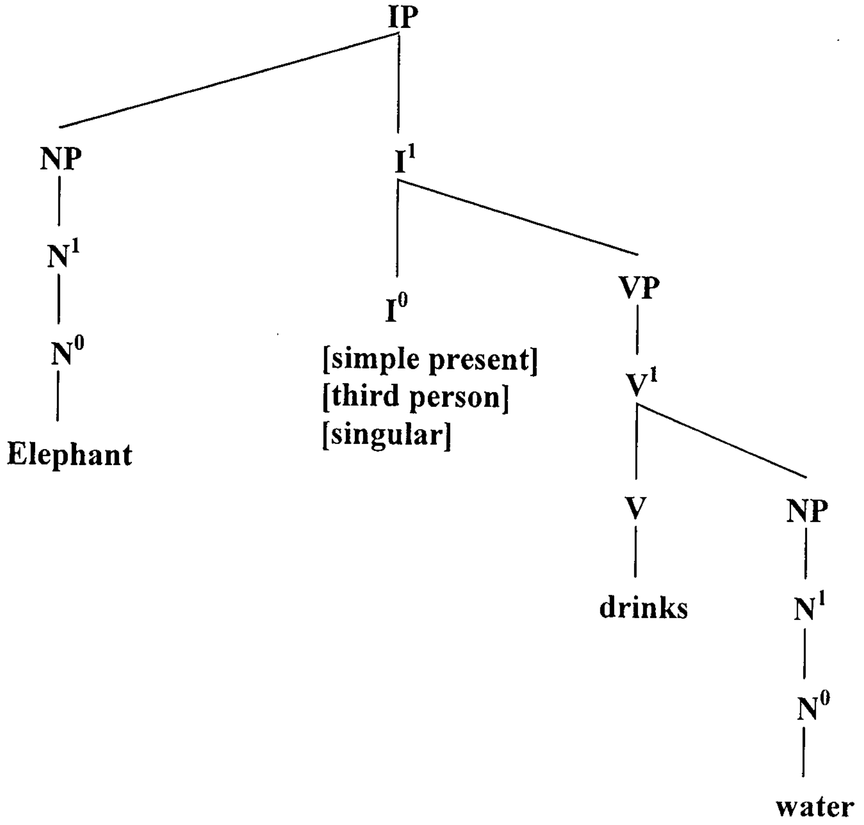

IP
NP
N1
N0
Elephant
I1
I0
[simple present]
[third person]
[singular]
VP
V1
V
drinks
NP
N1
N0
water

Figure 6.5f

In English the auxiliaries play an important role in determining the tense and aspect of the verb. We have seen that the English has three tenses and 4 aspects giving 12 different combinations, where as Sanskrit have only six tenses. That means that the 12 combinations of tense and aspect are mapped to 6 combinations in Sanskrit and vice versa. This means that for a simple present tense sentence they will be more than one English output. For example the sentence handled above and repeated below in (1) results into outputs given in (2). That means depending on the attributes we obtain more than one output tree from the tree given in Figure 6.5f, one tree corresponding to each output given in (2).

(1) गजः जलं पिबति ।

(2) The Elephant drinks water
The Elephant is drinking water
The Elephant has drunk water
The Elephant has been drinking water

One may however question the last two outputs given in (2). However, the discussion on this problem is not in the present scope of the thesis.

Chapter 7

CONCLUSIONS AND FUTURE ENHANCEMENTS

7.1 CONCLUSIONS

We have developed a web-based simple sentence GB translation system with Sanskrit as source language and English as target language. While various translations systems are being developed across the world using conventional approaches like Ruled- based or Exampled-based, we have adopted Government and Binding (GB) approach in our translation 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, Theta assignment module, Case assignment module, and Binding module. We have developed the X-Bar phrase structures for both Sanskrit and English.

The phrase structure rules are developed by us both for Sanskrit and English include Verb Phrase Structure, Noun Phrase Structure, Adjective Phrase Structure, Preposition Phrase Structure, Inflection Phrase Structure and Complementizer Phrase Structure. These Phrase Structures have been obtained after a thorough analysis of various phrases in both source and target languages. 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 Phrase. In our study we have analyzed the Verb Phrase in depth. We have only handled simple Noun Phrases. The Noun Phrase is much more complex than Verb Phrase. The complex Noun Phrase, therefore, has been left for future analysis. As an Adjective Phrase is always a part of a Noun Phrase, the analysis of this has also has been reserved for future. Once the Noun Phrase analysis has been taken care of, handling Prepositional Phrases becomes a trivial task. The grammar developed by us therefore, handles all simple sentences only in both languages. Our grammar has to be augmented in order to cover complex sentences.

A bilingual lexicon has been developed for the translation system. In general, the Lexicon contains the category and subcategory information for the words, the phonetic information (relating to speech sounds), thematic information, and other useful information such as morphological processes. However, in our case we are not using phonetic and thematic information. The lexicon developed by us for the translation system can be further improved by defining and adding finer subcategorization and thematic information.

Morphological processes widely differ from one language to another. For example Sanskrit is a highly inflected language but is more regular in nature. Whereas English is not so inflected language but has substantial amount of irregularity. We have written morphological processes for obtaining Sanskrit root and final word forms covering almost entire domain. For English the task however is not so complete because of irregular forms. To handle this situation we have to define finer sub categorizations both for verbs and nouns.

An LR Parser using Tomita's approach is developed as part of the translation system. A Tomita parser is particularly efficient when few conflicts are encountered in the LR states. Since all we need to do is recognizing the input (i.e. syntactic structure of the input), then Tomita parser is the best method of choice. Our parser is able to parse all kinds of simple sentences, which may be ambiguous at any level. Even for parsing, the complex sentences we need not change the basic parsing module, but simply build some sub-modules and can be made as a part of the existing module. The modules that we would like to develop and made as sub modules for our existing Parser are: the Case Checking Module, Trace Module, the Binding Module, and Bounding Module. Due to time limitations, we couldn't be able to extend our existing Parsing Module. So we suggest the above mentioned extensions for our Parser.

The system is implemented using VISUAL BASIC 6.0. The rich GUI, easy connectivity with the databases, and its user-friendly nature led to the choice of VB over other languages. The Lexicon and the morphological rules are stored in databases. We thus have SQL Server 2000 functioning at the back-end of our translation system. The server-side programming has been done in ASP. The UNICODE has been used for storing the information.

7.2 FUTURE ENHANCEMENTS:

As pointed out above enhancements are needed in the area of grammar for handling complex sentences. Finer sub-categorization is needed for making the morphology easier. It will also make the parser efficient as it will help in selecting more appropriate rules. As stated above, 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 as well as in the Generator. This therefore will be the next task we would like to take up.

We have already said that Sanskrit is a completely word order free language. There is a need therefore to convert a word order free sentence into a structured sentence needed for GB frame work. This is quite a heavy task. Further, in Sanskrit Sandhi plays a very important as it is a common practise in Sanskrit to present combinations of words as a single word. What this means is that in a sentence two or more words may be written together as a single combination replacing the original individual words. This problem in itself is quite complicated and needs lot of thinking along with developing Sandhi and Sandhi-Vichedi modules.

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

APPENDIX A

Here we are going to present the phrase structure rules adopted for Sanskrit language, which are required for parsing simple Sanskrit sentences by Parser. When complex sentences are to be considered the grammar rules can be augmented with out any change in our translator.

Augmentation:

[S] → [CP]

Complementizer Phrase:

[CP]→[CPS][C1]

[CP]→[C1]

[CPS]→[ADV]

[C1]→[C][CCOMP]

[C1] →[CCOMP]

[CCOMP]→[IP]

Inflection Phrase:

[IP]→[IPS][I1]

[IP]→[I1]

[IPS]→[NP]

[I1]→[ICOMP] [I]

[ICOMP]→[VP]

Verb Phrase:

[VP]→[V1]

[V1]→[VCOMP][V]

[V1]→[VCOMP]

[V1]→[V]

[VCOMP]→[NP]

[VCOMP]→[PP]
[VCOMP]→[CP]
[VCOMP]→[IP]

Noun Phrase:

[NP]→[NPS][N1]
[NP]→[N1]
[NPS]→[AP]
[N1]→[NCOMP][N]
[N1]→[N]
[NCOMP]→[NP]
[NCOMP]→[PP]

Prepositional Phrase:

[PP]→[PPS][P1]
[PP]→[P1]
[PPS]→[ADV]
[P1]→[PCOMP][P]
[P1]→[P]
[PCOMP]→[NP]

Adjective Phrase:

[AP]→[APS][A1]
[AP]→[A1]
[APS]→[ADV]
[A1]→[ACOMP][A]
[A1]→[A]
[ACOMP]→[NP]
[ACOMP]→[AP]

In the above rules

CP stands for Complementizer Phrase
NP stands for Noun Phrase
VP stands for Verb Phrase
PP stands for Prepositional Phrase
AP stands for Adjective Phrase
ADV stands for Adverb
NPS stands for Noun Specifier
VCOMP stands for Verb Complement

APPENDIX B

Here we are going to present the phrase structure rules adopted for English language, which are required for parsing simple English sentences by Parser. When complex sentences are to be considered the grammar rules can be augmented with out any change in our translator.

Augmentation:

[S] → [CP]

Complementizer Phrase:

[CP]→[CPS][C1]
[CP]→[C1]
[CPS]→[ADV]
[C1]→[C][CCOMP]
[C1] →[CCOMP]
[CCOMP]→[IP]

Inflection Phrase:

[IP]→[IPS][I1]
[IP]→[I1]
[IPS]→[NP]
[I1]→ [I][ICOMP]
[ICOMP]→[VP]

Verb Phrase:

[VP]→[V1]
[V1]→ [V][VCOMP]
[V1]→[V]

[VCOMP]→[NP]

[VCOMP]→[PP]

[VCOMP]→[CP]

[VCOMP]→[VP]

[VCOMP]→[IP]

Noun Phrase:

[NP]→[NPS][N1]

[NP]→[N1]

[NPS]→[AP]

[N1]→ [N][NCOMP]

[N1]→[N]

[NCOMP]→[NP]

[NCOMP]→[PP]

Prepositional Phrase:

[PP]→[PPS][P1]

[PP]→[P1]

[PPS]→[ADV]

[P1]→ [P][PCOMP]

[P1]→[P]

[PCOMP]→[NP]

Adjective Phrase:

[AP]→[APS][A1]

[AP]→[A1]

[APS]→[ADV]

[A1]→ [A][ACOMP]

[A1]→[A]

[ACOMP]→[PP]

[ACOMP]→[AP]

In the above rules

CP stands for Complementizer Phrase
NP stands for Noun Phrase
VP stands for Verb Phrase
PP stands for Prepositional Phrase
AP stands for Adjective Phrase
ADV stands for Adverb
NPS stands for Noun Specifier
VCOMP stands for Verb Complement

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] http://www.languageinindia.com/jan2005/aparnasanskritdissertation1.html

[20] Panini, *The Ashtadhyayi*

[21] Ullman J. D. and Hopecroft J. E. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979.

[21] Wren & Martin, 2001, *High School English Grammar &Composition.*