STUDIES ON A SYNTAX BASED APPROACH FOR TRANSLATION BETWEEN STRUCTURALLY DIFFERENT LANGUAGES AND THE DEVELOPMENT OF A PROTOTYPE FOR MALAYALAM TO ENGLISH TRANSLATION

Thesis submitted to

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

In partial fulfilment of the requirements

for the award of the degree of

DOCTOR OF PHILOSOPHY

Under the

Faculty of Technology

By

Latha R. Nair

DEPARTMENT OF COMPUTER SCIENCE

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY

KOCHI-22, INDIA

March 2013

Dr. S. David Peter
Professor
Division of Computer Engineering
School of Engineering
Cochin University of Science and Technology
Kochi-682022
Phone :91 484 556187
davidpeter@cusat.ac.in

March 2013

Certificate

Certified that the work presented in this thesis entitled "Studies on a Syntax based approach for translation between structurally different Languages and the development of a prototype for Malayalam to English translation" is based on the authentic record of research done by Smt.Latha R.Nair under my guidance in the Department of Computer Science, Cochin University of Science and Technology, Kochi-682022 and that no part there of has been included in any other thesis submitted for the award of any degree.

Dr. S.David Peter

(Supervising Guide)

Dr. Sumam Mary Idicula
Professor
Department of Computer Science
Cochin University of Science and Technology
Kochi-682022
Phone :91 484 2577126
sumam@cusat.ac.in

March 2013

Certificate

Certified that the work presented in this thesis entitled "Studies on a Syntax based approach for translation between structurally different Languages and the development of a prototype for Malayalam to English translation" is based on the authentic record of research done by Smt.Latha R..Nair under guidance of Dr.S. David Peter and myself in the Department of Computer Science, Cochin University of Science and Technology, Kochi-682022 and that no part there of has been included in any other thesis submitted for the award of any degree.

Dr. Sumam Mary Idicula

(Co-Guide)

DECLARATION

Certified that the work presented in this thesis entitled "*Studies on a Syntax based approach for translation between structurally different Languages and the development of a prototype for Malayalam to English translation*" is based on the original research work done by me under the supervision and guidance of Dr S. David Peter, Professor, Division of Computer Engineering, School of Engineering, Cochin University of Science and Technology, Kochi-682022 with Dr. Sumam Mary Idicula Professor, Department of Computer Science, Cochin University of Science and Technology, Kochi-682022 as co-guide and has not been included in any other thesis submitted previously for the award of any degree.

Kochi-22

March 2013 Latha R. Nair

ACKNOWLEDGEMENTS

I thank God almighty for blessing me with willpower and all qualities required for completion of my work as well as getting along with life.

The investigations in this thesis have been carried out under the supervision of Dr. S. David Peter, Professor, Division of Computer Engineering, School of Engineering, Cochin University of Science and Technology. I express my deep sense of gratitude for his excellent guidance, competent advice, keen observations and persistent encouragement as well as personal attention given to me during the entire course of work, without which the successful completion of this work would not have been possible. I am deeply indebted to him for all the above considerations.

It is with great pleasure that I acknowledge my co-guide Dr. Sumam Mary Idicula, Professor, Department of Computer Science, Cochin University of Science and Technology for her support and guidance during the entire course of my work.

I extend my sincere gratitude to Dr. Poulose Jacob, Head, Department of Computer Science for allowing me to use the facilities of the Department and for the support and guidance during the entire course of my work .

I wish to thank Dr. P. S. Sreejith, Principal, School of Engineering, for allowing me to use the facilities of the Department.

I am most thankful to Dr.. Rajiv Sangal, Dr. Sriram Venkatapathy, Dr. Srinivas Bangalore, Dr. Sobha L. and other faculty members at IIIT Hyderabad for helping with the contact angle studies.

I remember all the participants of the Winter School on Natural Language Processing held at IIIT Hyderabad especially Kartik Gali, Vasudevan, Vishal Goyal and Bhuvaneswari among others who have helped and supported me during my work.

I am indebted to my colleagues Dr. Sheena Mathew and Dr. G. Santhosh Kumar for helping me with various points during my work.

I express my sincere gratitude to all non teaching staff of CUSAT who have helped and supported me during the entire period of work.

I record my sincere and utmost gratitude to my parents and family for patience and tolerance during the entire period of my work

I thank all my well wishers.

.

Latha R. Nair

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MT	Machine translation
POS	Part of Speech
STAG	Synchronous tree adjoining grammar
LTAG	Lexicalised tree adjoining grammar
PL	Plural suffix
NA	Postposition
PA	Adjective
N	Noun
V	Verb
ADJA	Adjectival suffix
ADVA	Adverbial suffix
PAV	Adverb
VN	Verbal Noun
V RP	Verbal Relative participle
NCA	Noun clause suffix
ADVCA	Adverb clause suffix
INFA	Infinitive
DJ	Disjunction
C	Conjunction

LIST OF ABBREVIATIONS Contd..

LOC	Locatives
VA	Verbal suffix
NP	Noun Group
VG	Verb Group
NC1	Noun clause
ADVC	Adverb clause
ADJC	Adjective clause
NPC	Conjunct Noun
S	Sentence
CS	Compound sentence
CMPN	Compound noun
ADJCNP	Adjectival clause + Noun
ADJG	Adjective group
INFSG	Infinitive + verb group
INF	Infinitive
ADVG	Adverb group
VGC	Compound verb
VA	Verbal suffix
ADJLOC	Locative adjective

Studies on a syntax based approach for translation between structurally different languages and the development of a prototype for Malayalam to English translation

PhD thesis in the field of Natural Language Processing

Author:

Latha R Nair
Division of Computer Engineering
School of Engineering
Cochin University of Science and Technology
Kochi-682022,Kerala,India
Email :latharnair@cusat.ac.in

Supervisor:

Dr S. David Peter
Professor
Division of Computer Engineering
School of Engineering
Cochin University of Science and Technology
Kochi-682022, Kerala, India
Email:davidpeter@cusat.ac.in

Co-Guide

Dr. Sumam Mary Idicula
Professor
Department of Computer Science
Cochin University of Science and Technology
Kochi-682022, Kerala, India
Email:sumam@cusat.ac.in

March 2013

Dedicated to my parents

PREFACE

Machine processing of Natural (Human) Languages has a long tradition, benefiting from decades of manual and semi-automatic analysis by linguists, sociologists, psychologists and computer scientists among others. Machine translation (MT) is the name for computerized methods that automate all or part of the process of translating from one language to another. In a large multilingual society like India, there is a great demand for translation of documents from one language to another language. Though work in the area of machine translation has been going on for several decades, efficient methods for machine translation continues to be a challenging task and fully automatic high quality machine translation system is extremely difficult to build.

Various MT groups have used different formalisms best suited to their applications. Of them transfer based systems are found to be more flexible and it can be extended to language pairs in a multilingual environment. Direct translation is appropriate for translation between structurally similar languages. The interlingua based systems can be used for multilingual translation. The amount of analysis needed in interlingua approach is more than that in a transfer based approach. The universal networking language has been proposed as the interlingua by the United Nations University for overcoming the language barrier. Over the past decade data-driven approaches to machine translation have come to the fore of language processing research. The relative success in terms of robustness of example based and statistical approaches have given rise to a new optimism and an exploration of

other data-driven approaches such as Maximum Entropy language modeling. Performance of statistical techniques can be improved through large parallel corpus and usage of linguistic knowledge in the model. Hybrid systems are found to have better performance compared to the ones with the component technology. Though a number of Machine Translation systems between Indian and non-Indian languages have already been developed there are only very few systems developed for south Indian languages. The MT systems developed have many shortcomings in terms of rule set, dictionary and translation methodology. It is apparent from the survey that further work is needed in MT as a whole to produce meaningful translations. The research in machine translation will help to overcome the language barrier faced by India.

This thesis summarizes the results on the studies on a syntax based approach for translation between Malayalam, one of Dravidian languages and English and also on the development of the major modules in building a prototype machine translation system from Malayalam to English. The development of the system is a pioneering effort in Malayalam language unattempted by previous researchers. The computational models chosen for the system is first of its kind for Malayalam language.

An in depth study has been carried out in the design of the computational models and data structures needed for different modules: morphological analyzer , a parser, a syntactic structure transfer module and target language sentence generator required for the

prototype system. The generation of list of part of speech tags, chunk tags and the hierarchical dependencies among the chunks required for the translation process also has been done. In the development process, the major goals are: (a) accuracy of translation (b) speed and (c) space. Accuracy-wise, smart tools for handling transfer grammar and translation standards including equivalent words, expressions, phrases and styles in the target language are to be developed. The grammar should be optimized with a view to obtaining a single correct parse and hence a single translated output. Speed-wise, innovative use of corpus analysis, efficient parsing algorithm, design of efficient Data Structure and run-time frequency-based rearrangement of the grammar which substantially reduces the parsing and generation time are required. The space requirement also has to be minimised.

Chapter 1 gives an introduction to the need of a machine translation system for translating Malayalam to English. The chapter also mentions about the main objective of the research. It also discusses the language processing tools so far developed by various institutions for Malayalam language. The major research tasks are also briefed.

Chapter 2 describes in detail the various computational models tried for machine translation systems by various researchers. Three main models used are direct translation, rule based translation and corpus based translation. The chapter gives a detailed description of the various translation systems developed with each of these methods. The advantage and disadvantage of each method are also discussed.

Chapter 3 deals with the morphological variations found for nouns and verbs in Malayalam language and its use in morphological analysers. It details the various inflections and derivations found in the language. It also discusses the various word compounding rules in the formation of the morphological variations of root words. These rules are used for the design of the morphological analysis phase of the prototype machine translation system. The chapter also discusses the computational models used for the development of morphological analysers by various researchers.

Chapter 4 discusses the syntax of languages and the syntactic structure difference between languages. It discusses the hierarchical dependencies between sentence parts. It describes the different classes of sentences and clauses in a language with specific examples from Malayalam language. Finally the different models for parsing which uses language syntax for the syntax analysis are discussed.

Chapter 5 describes the design and development of the prototype machine translation system. The prototype uses synchronous tree adjoining grammar model for the translation process. It uses a morphological analyzer which finds the sequences of morphemes in the source sentence and parser which performs the word sense disambiguation of source words and creation of source and target parse trees. Data structure for bilingual dictionary and the steps in target sentence generation module are also discussed in this chapter.

Chapter 6 describes the development of the morphological analyzer module. The algorithm used by the analyzer is explained with

examples. The design of the splitting rule table is also discussed. The analyzer uses a depth first search algorithm with a learning component to find all sequences of morphemes in the sentence. The learning component helps to reduce the time for searching. It also gives the implementation of the analyzer tool and some sample screenshots from it.

Chapter 7 describes the development of the parser module of the MT system. The parse performs word sense disambiguation, chunking and syntactic structure transfer. The computational model and the algorithm for the parser module are discussed. It also discusses how the set of part of speech tags, chunk tags and also the chunk dependencies are derived to be used by the parser. Only a subset of the IIIT tagset was identified for our system. The tagsets and dependency rules can be extended to handle other sentence classes. The syntactic structure transfer rules required in the translation of Malayalam sentences to the corresponding English sentence are also discussed in this chapter. Creation of the source and target parse tree are explained with suitable examples.

Chapter 8 discusses the analysis of the outputs generated by the analyzer, parser and the translator as a whole along with examples. The prototype system outputs multiple translation results for some inputs. They are due to the deficiency in the analyzer module or the parser module. Only word sense disambiguation based on lexical category of the words in the sentence can be performed by the parser. The system also produces error outputs in some cases due to the difference in syntax of the source sentence. The system also lacks in considering the

agreements between chunks in the sentence. Some sample outputs of the translator are also given.

Chapter 9 presents the performance evaluation of the analyser, parser and prototype translation system as whole separately. The analyzer, parser and the full translator are tested with texts belonging to different domains. The analyser is found to generate more than one sequence in some cases. The parser accuracy was tested in the sentence level and also on the chunk level. The accuracy of the parser is tested on a sentence level. The precision and recall are calculated for the parser on a chunk level. The translator evaluation is done using the matrices for accuracy and word error rate. The evaluation reveals that the system performs best for children stories.

Chapter 10 summarizes the main tasks and the observations in the research work and the scope for future work. The major areas of enhancement required is in the generation of a unique correct translation for an input sentence of an unrestricted text. All the modules of the prototype are to be enhanced with suitable techniques to achieve this goal. The dictionary also has to be improved to handle words belonging to other domains. Since the computational model used is based on artificial intelligence techniques the development of translators from Malayalam to other languages is also considered as our future work.

CONTENTS

Chapter 1

INTRODUCTION

The various programmes developed for promoting use of IT in Indian languages have helped to generate expert manpower in diverse areas to offer solutions in Indian languages. As part of these programmes, work in the area of some of the foreign languages has also been carried out to provide solutions supporting those languages on personal computers. Research in language computing is progressing in many areas such as:

- machine translation
- speech processing
- optical character recognition (OCR)
- standards (character representation, fonts display, etc)
- localization
- development of applications like word processors, e-mail clients, etc.
- information extraction and retrieval
- search engines

In a large multilingual society like India, there is great demand for translation of documents from one language to another language. There are 22 constitutionally approved languages, which are officially used in different states. There are about 1650 dialects of these languages that are spoken by

different communities. There are 10 Indic scripts. All of these languages are well developed and are rich in content. They have similar scripts and grammars. The alphabetic order is also similar. Some languages use common script, especially Devanagari. Hindi written in the Devanagri script is the official language of the Union Government. English is also used for government notifications and communications. Even though India's average literacy level is 65.4 percent (Census 2001) less than 5 percent of people can either read or write English. As most of the state government works are in provincial languages whereas the central government's official documents and reports are in English or Hindi, these documents are to be translated from and to the respective provincial languages to have an appropriate communication. Moreover, over 95 percent of the population is normally deprived of the benefits of information technology due to language barrier [1]. All these make language translation a necessary one.

Work in the area of Machine Translation in India has been going on for several decades. During the early nineties, advanced research in the field of artificial intelligence and computational linguistics made a promising development of translation technology. This helped in the development of usable machine translation systems in certain well-defined domains. It is extremely difficult to build a fully automatic high quality machine translation system (FGH_MT). In fact there is no system in the world which qualifies to be called FGH_MT. Many institutions like IIT(Kanpur), CDAC(Mumbai), CDAC(Pune), IIIT(Hyderabad), etc. are engaged in the development of MT systems under projects sponsored by Department of Electronics, state governments etc. since 1990 [2]. Research on MT systems between various Indian and foreign languages and also between Indian languages are going on in these institutions. Translation between structurally similar languages like Hindi and Punjabi is

easier than that between language pairs that have wide structural difference like Hindi and English. Having many parts of their grammars and vocabularies in common, it is easy to develop translation systems between closely related languages [3].

Development of Machine translation (MT) system requires very close collaboration among linguists, professional translators and computer engineers. In the development process, there are two major goals; (a) accuracy of translation and (b) speed. Accuracy-wise, smart tools for handling transfer grammar and translation standards including equivalent words, expressions, phrases and styles in the target language are to be developed. The grammar should be optimized with a view to obtaining a single correct parse and hence a single translated output. Innovative use of corpus analysis, efficient parsing algorithm, design of efficient data structure and run-time frequency-based rearrangement of the grammar are required to reduce the parsing and generation time [4].

1.1 Earlier works in Malayalam Language processing

C-DAC, Thiruvananthapuram (formerly ER&DC) is a pioneer institution involved in Malayalam language tool development. It is one of the thirteen resource centers for Indian language technology solutions set up across the country by the Ministry of Communications and Information Technology, Govt. of India under the TDIL (Technology Development for Indian Languages) programme.

Some of the Tools for Malayalam which C-DAC could develop so far include:

1. Tools such as portal, fonts, morphological analyzer, ppell checker, text editor, search engine and code converters.

2. Knowledge resources like Malayalam corpora, trilingual (English-Hindi-Malayalam) online dictionary and knowledge bases for literature, art and culture of Kerala.
3. Human machine interface systems comprising of optical character recognition and text to speech systems.
4. Services like E-commerce application and E-mail server in Malayalam and
5. Language tutors for Malayalam and English.

1.2 Motivation for the work

Language translation between Malayalam and other languages can be said to be in very primitive stage now. There has been hardly any work done in the area of machine translation from Malayalam to English. It is also found that MT systems developed for other language pairs need improvements in:

- rule set
- dictionary design
- translation methodology

1.3 Research Objective

The main objective of this research is to study the issues in Malayalam to English translation and the development of a prototype system for Malayalam to English translation. The major goals in the design process are a) high accuracy of translation b) fast processing and c) low space requirement.

1.4 Research tasks

Major tasks include:

1. Study of the morphology of various lexical categories of Malayalam and English.
2. Identification of the part of speech tag set and chunk tag set for word sense disambiguation and structure transfer.
3. Derivation of syntactic structure of Malayalam and English sentences using morphology and word level dependencies.
4. Determination of the syntactic structure differences between Malayalam and English sentences.
5. Design of efficient lexicon in terms of speed of retrieval, space requirement and speeding up of the translation process.
6. Improving speed and accuracy of the translation process.

1.5 Conclusion

This chapter gave an introduction about the research work. The motivation behind the research work, the major objectives of the research and the tasks involved in the work were clearly described. The following chapter lists the various MT approaches and major MT systems developed in India using these approaches. Out of the various approaches listed one which is best suited for the language pair is chosen.

Chapter 2

APPROACHES FOR MACHINE TRANSLATION

2.1 Introduction

This chapter describes the various approaches used for machine translation. Researchers used different formalisms that are best suited to their applications. They are mainly i) Direct translation approach ii) Rule based approach and iii) Corpus based approach. Direct translation is appropriate for structurally similar languages. Among the rule based approaches transfer based systems are more flexible and it can be easily extended to language pairs in a multilingual environment. The interlingua based systems can be used for multilingual translation. The Universal Networking Language has been proposed as the interlingua by the United Nations University for overcoming the language barrier. Over the past decade data-driven approaches to machine translation have come to the fore of language processing research. Hybrid systems are found to have better performance compared to the ones with the component technology.

2.2 Direct machine translation systems.

As the name suggests, these systems provide direct translation, without using any intermediate representation. This is done on a word by word translation using a bilingual dictionary usually followed by some syntactic arrangement. The steps involved are :

a. Identification of root words by removing suffixes from source language words.

b. Dictionary look up to get the target language words/morphemes.

c. Rearrange the word order to match the target language. For English to Malayalam, this may be reordering of prepositions to postpositions and changing subject-verb-object to subject –object- verb structure.

A sample output from such a system for English to Hindi is given below:

Input(English): Rama played in the garden.

Output (Hindi) after word translation: Rama khela maim baag.

Output after syntactic rearrangement: Rama baag maim khela.

A direct translation system is appropriate for similar languages like Hindi and Punjabi [5, 6]. Vishal Goyal and Gurpreet Singh Lehel of PunjabUniversity have developed a web based Hindi to PunjabiMT system with 95% accuracy. Their system has additional modules for training the system for generating the lexicon using already existing corpus, input text font conversion into Unicode format to make the system free from specific font dependency, Hindi text normalization to handle spelling variations for the same word due to variation in dialects, replacement of collocations by keeping a lexicon for collocations, named entity recognition and replacement , word by word translation using bilingual dictionary and transliteration of unknown words. They also perform word sense disambiguation using a dictionary of ambiguous words. It uses a trigram approach with a sample corpus for word sense disambiguation.

A similar direct translation approach has been applied to Anusaraka systems which translate between two closely related Indian languages using the principles of paninian grammar [7]. The Anusaraka project started at IIT

Kanpur, by Prof. Rajeev Sangal and the research is continued at IIIT Hyderabad. Anusaarakas have been constructed from Telugu, Kannada, Bengali, Punjabi and Marathi to Hindi. Anusaraka systems or the language accessors are based on the principle of 'information preservation'. As a consequence, the Anusaaraka output follows the grammar of source language. Hence before using Anusaaraka systems to access the information, the reader has to undergo a short training to read and understand the output. Anusaaraka provides 'glosses' in target language for each meaningful lexical unit. There are cases where the meaning is too general or too specific. Such cases are handled by introducing some special notation to either narrow down or widen themeaning. An attempt is made to find the underlying thread (called '*shabda sutra*' or 'wordthread') that connects different senses of the polysemous word. A kind of formula ('sutra' also means a formula in Sanskrit) is then evolved that faithfully and unambiguously represents the connection between these different senses. The core Anusaaraka engine has four major modules viz.

(i) Word level substitution

(ii) Word sense disambiguation

(iii) Preposition placement, and

(iv) Target language generation

The output generated by the system may not be grammatically perfect and can be understood by a reader after some training. A sample output of an Anusaaraka system from Telugu to Hindi is shown below:

Input(Telugu): mIru pustakam caduvutunnArA

Output(Hindi): aap pusthak paTh raha [hai|thha] kya ?

The translation is done in a morpheme by morpheme basis. The tense, aspect and modality information of the verb from the source sentence is extracted and is translated as raha [hai|thha] kya?. The possible suffixes for tense information [hai/thha] are given and the user has to take the correct one based on the context.

Anusaraka based on a new architecture together with a user-friendly interface is convenient for a user-cum-developer. This new architecture makes a clear-cut distinction between the resources that are in principle reliable and those that are in principle probabilistic.

2.3 Rule based translation

Rule based machine translation (RBMT) systems parse the source text and produce an intermediate representation, which may be a parse tree or some abstract representation. The target language text is generated from the intermediate representation. These systems rely on specification of rules for morphology, syntax, lexical selection and transfer, semantic analysis and generation and hence are called rule based systems. Depending on the intermediate representation used, these systems are further categorized as Transfer based machine translation and Interlingua based machine translation.

2.3.1Transfer based machine translation

This method needs parsing of input text to get the structure of the input sentence. It has three modules: analysis module, transfer module and generation module [8,9]. The analysis module produces source language structure. The language grammar rules can be used to generate the hierarchical syntax tree for

the source language sentence. A set of hierarchical rules for forming the syntax tree for English sentences are given below:

S-> NP VP | VP

NP-> N| NP PP| Det N

VP ->V| VP PP| VP NP

The transfer module transfers the source language structure representation to a target language structure representation. This module needs the subtree rearrangement rules by which the source language sentence syntax tree can be transformed into target language sentence syntax tree.

A set of transfer rules for a English to HindiMT system are:

English structure	Hindi structure
VP->V NP	VP->NP V
PP-> P NP	PP-> NP P
VG->ADV V	VG->V ADV

The generation module generates target language text using target language structure. Syntactic structure transfer for verbs from Tamil to Hindi is discussed in [10,11]. It involves lexical transfer of verbs, transfer of auxiliary verb for tense, aspect and mood and transfer of gender, number and person information. Syntactic and lexical ambiguities are better resolved in this approach than in

direct translation approach. An example for structure transfer from English to Hindi is shown below:

Input(English): Ram sat on a chair.

Output(Hindi): Ram ne ek kursi mein baita.

We have used two transfer rules. One is to switch the preposition and noun in the PP chunk and the other is to place the verb at the end of the sentence. The structural transfer and Hindi generation are shown in Figure 2.1. Transfer systems are more realistic, flexible and adaptable in meeting the needs of different levels and depths of syntactic and semantic analysis.

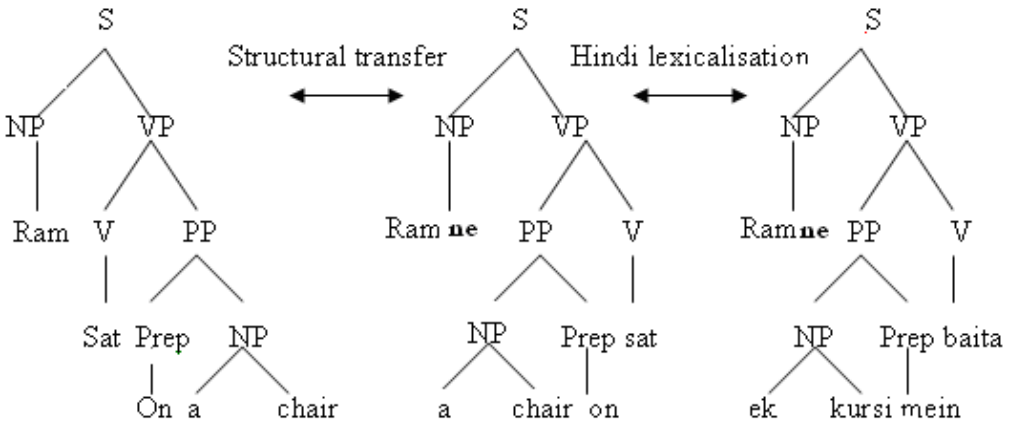


Figure 2.1 Structural transfer in a English-Hindi MT system

The English to Hindi MT system Mantra, developed by Applied Artificial Intelligence(AAI) group of CDAC, Bangalore, in 1999 uses transfer based

approach. The system translates domain specific documents in the field of personal administration; specifically gazette notifications, office orders, office memorandums and circulars. It is based on lexicalized tree adjoining grammar(LTAG) to represent English and Hindi grammar which are used to parse source English sentences and for structural transfer from English to Hindi [12,13,14,15,16]. This system also works well on other language pairs such as English-Bengali, English_Telugu, English_Gujarati and Hindi_English and also between Indian language pairs such as Hindi-Bengali and Hindi-Punjabi. The Mantra approach is general but the lexicon and grammar have been limited to the specific domain of personal Administration. The salient features of Mantra are :i) format retention ii) input can be in .rtf or .htm file or output of a speech recognition program or an optical character recognition package. It uses preprocessing tools like phrase marker, named entity recognizer, spell and grammatical checker. It uses Earley's style bottom up parsing algorithm for parsing . The system provides online addition of grammar rule. The system produces multiple translation results in the case of multiple correct parses. Mantra is one of CDAC's major achievements.

An English to Kannada MT system is developed at Resource centre for Indian Language Technology Solutions(RC_ILTS), University of Hyderabad by Dr. K. Narayan Murthy [12]. This also uses a transfer based approach and it can be applied to the domain of government circulars. The project is funded by Karnataka government. This system uses Universal Clause Structure Grammar (UCSG) formalism [17]. The technique is applied to English_ Telugu translation as well.

Other systems using this approach are : Matra - English to Hindi MTS developed by CDAC, Pune, Sakti- English to Marathi, Hindi and Telugu

developed by IISc Bangalore and IIIT, Hyderabad, Anubaad- English to Bengali developed by CDAC, Kolkata, English to Malayalam MTS developed by Amrita Institute of Technology. Some of the MT systems for non indan languages using this approach are Apertium, interNOSTRUM etc. [18,19,20,21]

2.3.2 Interlingua based machine translation

The interlingua approach was inspired by Chomsky's claim that regardless of varying surface syntactic structures, languages share a common deep structure [22]. In this approach, translation is a two step process: analysis and synthesis. During analysis, the source language text is converted into a language independent meaning representation called interlingua. In synthesis phase the interlingual representation is translated into any target language. Thus it can be used for multilingual translation. The amount of analysis needed in interlingual approach is more than that in a transfer based approach. This requires semantic analysis and the representation can be used for information retrieval.

For the sample sentence "Raman played in the garden", the interlingua generated is,

(play (tense: past)
(Mood declarative)
(subject (Raman(number singular))
(location(garden(reference definite)
(number singular))))

The major difficulty in using this approach is in defining a universal interlingua which preserves the meaning of a sentence.

Anglabharti is an MT system for translation from English to Indian languages which uses pseudo interlingua approach. It was developed in 1991 by Prof. R.M.KSinha and his team at IIT Kanpur. The system analyses English sentences and creates an intermediate structure called PLIL(Pseudo Lingua for Indian Languages). It performs most of the disambiguation. The effort required for analysis phase is 70% and the generation phase takes 30%. So with an additional effort of 30% a new translator for another Indian language could be built. A context free grammar like structure is used to create the PLIL structure. It also uses statistical analysis of a corpus to identify the movement rules for the PLIL structure. Its beta_version is Angla Hindi for English to Hindi translation and is available at http://anglahindi.iitk.ac.in. [23,24].

The World Wide Web contents are mostly in English and cannot be accessed without proficiency in this language. The universal networking language has been proposed by the United Nations University for overcoming the language barrier. An English to Hindi MT system which uses Universal Natural Language (UNL) as the interlingua has been developed by Pushpak Bhattacharya at IIT Bombay. This system has an English analyzer which converts the sentence into UNL form which is then given to a Hindi generator which generates the target sentence in Hindi. 95% of the UNL expressions were correctly converted to Hindi. It also does part of speech disambiguation and sense disambiguation for postposition markers and wh_pronouns. The system handles language divergence in a better way [22]. Currently on an MT system for English to Marati and Bengali is in progress.

2.4 Corpus based Machine Translation

Corpus based MT systems have gained much interest in recent years. The advantage of these systems are that they are fully automatic and require less human labour than rule based approaches. The disadvantage is that they need sentence aligned parallel text for each language pair and this method can not be employed where these corpora are not available. Corpus based systems are classified into statistical machine translation(SMT) and Example based Machine Translation(EBMT) [23,24,25].

2.4.1 Statistical Machine Translation

In this the input is considered as a distorted version of the target language sentence and the task is to find the most likely source language sentence given the translation.

The task involves three steps:

i) Estimating the language probability P(t)

ii) Estimating the translational model probability p(s/t)

iii) Devising an efficient search for the target text that maximizes their product.

We have to find the sentence T for which p(s,t) is maximum.

$$P(s,t)= \arg\max_{t} p(s,t) = \arg\max_{t} p(t)p(s/t)$$

In the above model s is the source language sentence and t is the target language sentence. The probabilities are to be calculated from the parallel corpus.

Smoothing techniques are required for handling data sparsity problem that occurs in any noisy channel model.

A phrase-based Hindi-English translation system was tried by Kamal Kuzhinjedathu and ShravyaShetty at Department of Computer Science, State University of New York. The translation model was generated using a Hindi-English parallel corpus. Since the parallel corpus is only sentence aligned the freely available online tool called GIZA++ was used to perform word alignment. The alignment produced were then processed to create the phrase based translation model. Two sets of parallel corpuses were used: EMILE (Enabling minority language engineering) corpus distributed by the European Language Resources Association and the Hindi and English bibles from www.Hindibible.org. Bilingual dictionaries available on the internet were used to to augment the statistical model. The SRI language model toolkit10 was used to obtain a statistical language model. After getting translation model and language model a phrase based decoder, Moses was used to translate the test sentences.

An English to Hindi MT system which combines RBMT and phrase-based SMT approach was developed at IIIT Hyderabad in 2010. Though SMT systems are able to handle local re-orderings by themselves, in case of long-distance transformations they benefit considerably from external guidance (the RBMT system in this case). The system works in two stages. In the first stage, the source analyzer performs extensive linguistic analysis by running Brill's POS tagger and the Stanford dependency parser on the input sentence. It then converts the source into a chunk-based unordered dependency tree. In the next stage, the Transfer Grammar performs local and long-distance re-orderings. By chunking the source sentences and converting them into a dependency structure,

the RBMT system separates local (intra-chunk) reordering decisions from global (inter-chunk) re-orderings. This allows for separate specifications of local and long-distance rules; thus, greatly reducing the number of rules that must be written into the grammar [26].

Translation of English into the Dravidian language, Malayalam SMT system has been tried in Cochin University of Science and Technology, Cochin, in 2010. By using a monolingual Malayalam corpus and a bilingual English/Malayalam corpus in the training phase, the machine automatically generates Malayalam translations of English sentences. The alignment model is improved by incorporating the parts of speech information into the bilingual corpus. Removing the insignificant alignments from the sentence pairs by this approach has ensured better training results. Pre-processing techniques like suffix separation from the Malayalam corpus and stop word elimination from the bilingual corpus are carried out for effective training. The structural difference between the English - Malayalam pair is resolved by a decoder using the order conversion rules [27].

2.4.2 Example Based Machine Translation (EBMT)

An Example based Machine translation (EBMT) system maintains a corpus consisting of translation examples between source and target languages. An EBMT system has two modules: Retrieval module and an adaptation module. The retrieval module retrieves a similar sentence and its translation from the corpus for the given source sentence. The adaptation module then adapts the retrieved translation to get the final corrected translation.

Consider the English to Hindi translation for the following sentence:

Rama sings a song.

The retrieval module retrieves the following sentence and its translation from a list of approximately matching sentences in the corpus. It uses some similarity measures based on word similarity or syntactic and semantic similarity to identify this set of approximately matching sentences. From these the system selects the sentence with closest match with the input sentence.

If the system selects "Rohit sings a song" and its translation "Rohit geet gaatha hai" as the closest one, it replaces Rohit with Rama and gaata with gaathi and finally forms the translation

Rama geet gaati hai

Here the adaptation is required to replace the word and suffix replacements. This method may not work in case of translation divergence where structurally similar sentences of the source language get translated into a different structure [28].

Anubharti, is an EBMT approach based MT system developed at IIT Kanpur by Prof. R.M.K Sinha and his associates. Along with basic EBMT it uses some grammatical analysis to reduce the size of the parallel corpus. This is done primarily by generalizing the constituents and replacing them with abstracted form achieved by identification of syntactic groups from the raw examples [29,30].

Vaasaanubaada is another system for translating bilingual Bengali-Assamese news texts using EBMT technique. The work involves machine translation of bilingual texts at sentence level. In addition, it also includes preprocessing and post-processing tasks. The work is unique because of the language pair that is

chosen for experimentation. The bilingual corpus was constructed and aligned manually by feeding real examples using pseudocode. The longer input sentence is fragmented at punctuations, which resulted in high quality translation. Backtracking is used when an exact match is not found at the sentence/fragment level, leading to further fragmentation of the sentence. Since bilingual Bengali-Assamese languages belong to the Magadha Prakrit group, the grammatical form of sentences is very similar and has no lexical word groups. The system gives quality translation [31].

Siva, an English to Hindi MT system developed jointly by Carneige Mellon University, USA, IISc Bangalore and IIIT Hyderabad is using EBMT approach. In addition to the hard coded linguistic rules it uses a statistical approach for learning new rules [15]. An English to Sanskrit EBMT has been tried at Banaras Hindu University. Many hybrid systems which combines rule based approach and corpus based approach has been tried and has produced encouraging results [32,33,34,35,36]. The systems developed for Indian languages are listed in Table 2.1.

Table 2.1 Machine translation systems for Indian languages

Method	Name of the MT system and Language Pair	Developer	Year
Direct	Anusaarak systems(Telugu,Kannada,Bengali,Marathi to Hindi)	Started in IIT Kanpur, continuing in IIIT, Hyderabad	1995
	Punjabi to Hindi MTS	PunjabUniversity, Patiala	2007
	Web based Hindi to Punjabi MTS	PunjabUniversity, Patiala	2010

Table 2.1 Machine translation systems for Indian languages contd..

Method		Name of the MT system and Language Pair	Developer	Year
Rule Based	Transfer based	Mantra-English to Hindi, Telugu,Gujarathi,Hindi-English	CDAC,Pune	1995
		Matra-English to Hindi MTS	CDAC,Pune	2004
		Shakti-English to Hindi,Marathi,Telugu MT	IIISc Bangalore and IIIT, Hyderabad	2004
		Anubaad-English-Bengali MTS(n-gram approach for pos tagging)	CDAC, Kolkata	2005
		English to Kannada MTS using UCSG	Unversity of Hyderabad	2006
		English – Malayalam MTS	Amrita Institute of Technology	2009
	Interlingua	Anglabharti, English-Hindi,Tamil MTS; uses pseudo interlinguaPLIL	IIT,Kanpur	1991
		AnglaHindi(combines example based approach and AnglaBharti approach)	IIT,Kanpur	1991
		English to Hindi MTS Using UNL	IIT, Mumbai	2001
		Hindi-English MTS (Additional layer over AnglabhartiII)	IIT,Kanpur	2005
Corpus based	SMT	Tamil-Sinhala MTS	Carnegie-Mellon University, Pittsburgh, USA	2002
		English-Hindi MTS	IIIT, Hyderabad	2010
		English- Malayalam MTS	Cochin University ,Cochin	2010
		Hindi-English MTS	StateUniversity of New York.	2010
	EBMT	VAASAANUBAADA - Bengali-Assamese	PondicheryUniversity	2002
		Anubharti; Hindi-English MTS	IIT,Kanpur	2004
		Siva- English-Hindi	IIScBangalore,IIITHyderabad	2004
		English-Sanskrit	I.T, Banaras Hindu University	2008

2.5 Conclusion

This chapter explains in detail the various approaches tried by different researchers in this field. A number of Machine Translation systems between Indian and non-Indian languages have already been developed. Most of the MT systems are for Hindi and there are only very few systems for south Indian languages. More research has to done in these areas to overcome the language barrier faced by India. The MT systems developed has many shortcomings in terms of rule set, dictionary, translation methodology and it is apparent from the survey that further work is needed in MT as a whole to produce intelligible translations. The amount of analysis needed in interlingual approach is more than that in a transfer based approach. From among the various approaches the transfer based approach has been chosen due to its flexibility and extensibility to other language pairs in a multilingual environment.

Chapter 3

LANGUAGE MORPHOLOGY AND MORPHOLOGICAL ANALYSIS

3.1 Introduction

The chapter discusses the different types of morphology found in Malayalam and the word compounding rules for concatenating two morphemes. The chapter discusses only concatenative morphology. This chapter also discusses the various types of models for morphological analysis. The morphological analyser requires the types of morphology in a language and the morphophonemic changes at morpheme boundary when two or more morphemes are concatenated.

3.2 Morphology of Malayalam

Malayalam is the mother language in the state of Kerala. It is one among the 22 official languages of India. The language is used by more than 35 million people spreading along the regions Kerala, Lakshadweep, and Pondichery.

Malayalam is one of the four major languages of this Dravidian family with a rich literary tradition. It is very close to Tamil, one of the major languages of the same family. This was due to the extensive cultural synthesis that took place between the users of the two languages. The origin of Malayalam as a distinct language may be traced to the last quarter of 9th Century A.D. Throughout its gradual evolution Malayalam has been influenced by the various circumstances prevailed during different periods.

Various types of morphological variations are found in Malayalam language. Morphological variations for words occur in Malayalam due to

i) Inflections
ii) Derivations
iii) Word compounding

In inflectional morphology a lexical category like noun or verb is attached to suffixes to generate words of the same category. In Derivational morphology categories like noun or verb with a suffix attached to it generates a word of new category. In word compounding new words are formed by combining a noun and a noun, noun and adjective, verb and noun, adverb and verb, adjective and noun and in some cases all the words of an entire sentence to reflect the semantics of the sentence [37].

3.2.1 Morphology for nouns

A lexical category like a noun or a verb is attached to suffixes to generate a word of the same category or of a different category. Nouns show inflectional morphology due to the addition of gender, number and case information. Nouns also show derivational morphology where modifiers and verbs are derived from nouns by adding proper suffixes.

3.2.1.1 Inflectional morphology for nouns

The morphological information that can occur with nouns are gender, number and case. Prepositions in English and postpositions in Hindi marking the cases are not attached to nouns. But in most of the Dravidian languages case suffixes are attached to nouns. There are three genders for nouns: masculine, feminine and neuter. The number can be singular or plural. For proper nouns and

abstract nouns, there are no plural forms. The case indicates the relation which a noun or pronoun has with verb of the sentence. The seven cases in Malayalam are nominative, accusative, sociative, dative, genitive and locative [38,39].

3.2.1.2 Derivational morphology for nouns

Adjectives, adverbs and verbs are derived from nouns by the addition of proper suffixes.

i) Adjectives

Modifiers are qualifiers and can be of three types. It is called *naamavisheshanam* (adjective) when it modifies a noun, *kriyaavisheshanam* (adverb) when it modifies adverb and *bhedakavisheshana* (modifier of modifier) when it modifies a modifier. The modifiers can be pure modifiers or those derived from nouns and verbs. Different types of pure modifiers are determinative adjectives, separative adjectives, interrogative adjectives, temporal adverbs, special adverbs and adverbs of manner.

Adjectives may be derived from another noun by the addition of ulla.

ആഴം (*azham*/depth)) + ഉള്ള (*ulla*/has) = ആഴമുള്ള *azhamulla (*deep).

An adjective can be derived by suffixing ഇലെ (*ile* / in) to the noun,

കാട് *(kaatz/forest*) + ഇലെ (*ile*/in) = ഇലെ (*kaattile/in forest).*

An adjective can be derived by adding the suffix '*aththe*' to the noun,

അന്ന് (*annz* / that day) + അത്തെ (*aththe/*on) = അന്നത്തെ (*annaththe*/that day's)

ii) Adverbs

Adverbs can be derived from a noun by suffixing adverbial markers.

The suffix – ആയി */aayi* is used with abstract nouns to make modifiers:

ഭംഗി (*bhangi*/nice) + ആയി (*aayi/ly)* = ഭംഗിയായി (*bhangiayi*/nicely).

To show direction, the suffix *ekku* is added to the noun,

പുറത്ത് (*puRathtz* / out) + ഏക്ക് (*ekku* / to) = പുറത്തേക്ക്
(*puRaththEkkz* / to outside).

iii) Verbs

The nouns with vowel ending takes suffixes ഇക്കുന്നു / *ikkunnu*, ഇച്ചു / *ichu* and ഇക്കും / *ikkum* for present, past and future respectively. The nouns with vowel ending takes the suffixes ക്കുന്നു / *kkunnu*, ച്ചു / *chu* and ക്കും / *kkum* for present, past and future tenses.

തടി (*thati* fat) + ക്കുന്നു (*kkunnu*/present tense marker) = തടിക്കുന്നു
(*thatikkunnu*/becoming fat)

കല്ല് (*kallz*/stone) + ഇക്കുന്നു (*ikkunnu/*tense marker) = കല്ലിക്കുന്നു
kallikkunnu (hardening)

The inflectional and derivational morphology for nouns commonly found in Malayalam are listed in Table 3.1.

Table 3.1 Morphology for nouns

		Type	suffix	Example
inflections	plural		കൾ / *kaL*	കുട്ടികൾ / *kuttikaL*
	case	nominative	No suffix	രാമൻ /*raaman*
		accusative	എ / e, ഇനെ / *ine*	രാമനെ / *raamane*, മധുവിനെ / *madhuvine*
		dative	ഓട് *Odu*, ഇനോട് *inOd*u	രവിയോട് / *raviyOdu*, മധുവിനോട് *madhuvinOdu*
		sociative	ഇക്ക് / *ikku*, ഇന് / *inu*	സതിക്ക് / *sathikku*, ന് / *madhuvinu*
		instrumental	ആൽ/ *aal*, ഇനാൽ / *inaal*	വടിയാൽ / *vatiyaal*, മധുവിനാൽ / *madhuvinaal*
		genitive	ഇന്റെ / *inte*, ഉടെ/ *ute*	സീതയുടെ / *seethayute*, മോഹന്റെ / *mOhante*
		locative	ഇൽ / *il*	കാട്ടിൽ / *Kaattil*, തറയിൽ / *thaRayil*
derivations	adjective	quality	ആയ / *aaya*	നല്ലവനായ / *nallavanaaya*
		quality	ഉള്ള / *ulla*	ഭംഗിയുള്ള / *bhangiyulla*
		place	ഇലെ *ile*	മരത്തിലെ / *marathile*
	adverb	manner	ആയി / *aayi*	ഭംഗിയായി / *bhangiyaayi*
		direction	ഏക്ക് / *ekkz*	കാട്ടിലേക്ക് / *kattilEkkz*

3.2.2 Morphology for verbs

A verb denotes the ‘state of’ or ‘action’ done by a substance. Various admissible forms of verbs can be generated by considering their tense, mood

and aspect information. There are 57 admissible forms of a verb, taking into account various forms of tense, mood and aspect [38,39].

3.2.2.1 Inflectional morphology for verbs.

Inflections for verbs occur due to i) tense, ii) aspect iii) mood information. In some languages like Hindi, the verbs have inflections corresponding to the gender, number and person information of the subject. In Malayalam the inflection due to gender and person information of the subject is absent. The inflectional suffixes for verbs are given in Table 3.2.

i) ***Tense***

Tense aspects considered are for present and future tenses. The morphology for past tense is not considered.

a) *Present tense*: The inflectional suffix considered is *unnu.*

പറ (*paRa*/tell) + ുന്നു (*unnu*/present tense marker) = പറയുന്നു/ *parayunnu* (is telling)

b) *Future tense*: The suffix which marks future tense is um.

പോക് (*poku*/go) + ും (*um*/future tense marker) = പോകും (*pokum*/ will go)

ii) ***Aspect***

The aspect factors considered are for a) perfective b) imperfective c) ingressive.

a) *Perfective*: The perfective aspect can be present perfect, past perfect, future perfect, present continuous or past continuous.

പോയി (*pOyi* / went) + കൊണ്ടിരുന്നു (*kontirunnu* / had been) =

പോയിക്കൊണ്ടിരുന്നു (*pOyikkontirunnu* / had been going)

പോയി(*pOyi/went)* + ഇരുന്നു (i*runnu/had)* = പോയിരുന്നു (*pOyirunnu* /had gone)

b) *Imperfective*: The imperfective suffixes considered are ഉന്നുണ്ട് / *unnuntu* and ഉകയാണ് / *ukayaaNz*

പോക് (*pokz*/go) +ഉന്നുണ്ട് (*unnuntz*/will be) = പോകുന്നുണ്ട്

pokunnuntz (will be going)

c) *Ingressive:* The ingressive feature indicates beginning of a situation. The suffix used is ആറായി / *aarayi.*

തുടങ്ങ് / *thutangz* +ആറായി / *aaRaayi* = തുടങ്ങാറായി / *thutangngaaRaayi*

iv) Mood

The mood features considered are a) optative b) intentional c) debititive d) ability e) permission f) degree of certainty g) authority of assertion.

a) *Optative* : This mood indicates a wish for something to happen. The suffix used is atte.

രക്ഷിക്ക് / *rakshikkz* +അട്ടെ / *atte* =രക്ഷിക്കട്ടെ / *rakshikkatte(*let them save*)*

b) *Intentional :* This indicates the speaker's willingness to carry out an action. This indicates a stronger commitment to a future course of action than simple future tense form. The suffix for this mood is ആം / *aam.* Here the subject should be first person.

പോക് (*pokz*/go) + ആം (*aam*/will) = പോകാം (*pOkaam*/will go)

c) *Debititive:* This expresses obligation. This is marked by suffixes *anam, eetheeroo.*

പോക് (*pokz*/go) + അണം (*aNam/*should) = പോകണം

(*pokaNam*/should go)

വന്ന് (*vannz*/come) + ഏതീരു (*Etheeroo/*should) = വന്നേ തീരു

(*vannEtheeroo*/should come)

d)*Ability:* This indicates the physical ability to perform an action. The suffix used for these are of two types: a verb with the suffix "*aam*" or an infinitive followed by forms of verbs *kazhiyuka, sadhikkuka and okkuka*. Both cases expect a dative subject.

കാണ് (*kaaNz*/see) + ആം (*aam/*can) = കാണാം (*kaaNaam/*can see*)*

പോക് (*pOkz*/go) + ആൻകഴിഞ്ഞു (*aankazhinjnju*/could) = പോകാൻ കഴിഞ്ഞു (*pOkaankazhinjnju/could go).*

e) *Permission:* This mood grants permission to the addressee. The verbal suffix used is *anam* with a second person dative subject.

പോക് (*pokz*/go) +ആം (*aam*/shall) =പോകാം (*pokaam/* shall go*)*

f) *Degree of certainity*: This indicates the degree of certainity with which the speaker makes an assertion. The suffixes used for this are: ഏക്കാം *Ekkam*, ഉമായിരിക്കും / *umaayirikkum*, വരുമായിരിക്കാം / *umaayirikkaam*.

വന്ന് (*vannz*/come) + ഏക്കാം (*Ekkaam*/may) = വന്നേക്കാം/

(*vannEkkaam* / may come*)*

Table 3.2 Inflectional morphology for verbs

Type of inflection		Imperative forms	Suffix
Tense		Past	ഉം / *um*
		Present	ഉന്നു / *unnu*
Aspect	Perfect	Present	ഇട്ടുണ്ട് / *ittuntu*, ഇരിക്കുന്നു / *irikkunnu*
		Past	ഇക്കഴിഞ്ഞിരുന്നു / i*kkazhinjirunnu*, ഇട്ടുണ്ടായിരുന്നു / *ittuntaayirunnu*
		Future	ഇരിക്കും / *irikkum*, ഇട്ടുണ്ടാകും / *ittuntakum*
		Present continuous	കൊണ്ടിരുന്നു / *kontirikkunnu*
		Past continuous	ഇരിക്കുകയായിരുന്നു / *irikkukayayirunnu*
	Imperfective		ഉന്നുണ്ട് / *unnuntu*, ഉകയാണ് / *ukayaaNu*
	Ingressive		ആറായി *aaraayi*
	Other Auxiliaries		പോയി / *POyi*, വിട്ടു / *vittu*, കളഞ്ഞു / *kalanjnju*, കൊടുത്തു / *kotuththu*
Mood	Optative		അട്ടെ / *atte*
	Intentional		ആം / *aam*
	Debititive		അണം / *aNam*, ഏതീരു / *Etheeru*
	Debititive(-ve)		അണ്ട / *anta*, ഇക്കൂട/ *ikkoota*, ആൻപാടില്ല/ *aanpaatilla*
	Ability		ആം/ *aam + dative subject*
			ആൻകഴിഞ്ഞു/*aankazhinjnju*, ആൻസാധിച്ചു / *aansadhichchu*, ആൻഒത്തു / *aanoththu*
	Permission(+ve)		ആം / *am* (only for dative subject)
	Permission(-ve)		അരുത് / *aruthu*
			ആട്ടെ /a*ate*, ഓളു / *Oloo* (only third person)
	Degree of certainity		ഏക്കാം/*Ekkaam*,ഉമായിരിക്കാം/ *umaayirikkaam*,ഉമായിരിക്കും /*umaayirikkum*
	Authority for assertion		അത്രെ/ *aththre*, എന്നു കേട്ടു / *ennukEttu*

3.2.2.2 Derivational morphology for Verbs

In this word categories like noun or verb with a suffix attached to it generates a word of new category. The derivations considered are i) participles ii) infinitives. Derivational morphology for verbs are given in Table 3.3.

i) ***Participles:*** The participles considered are verbal participle, conditional participle, concessive participle and relative participle.

പറഞ്ഞ (*paRanjnja* / said-relative participle), പോയാൽ (*pOyaal* / if goes-conditional participle)

ii) ***Infinitives***: The suffix taken by infinitives is "*aan*"

പറയ് *(paRayz* / say) + ആൻ (*aan* / to) = പറയാൻ (*paRayaan / to say)*

പോക് *(pOkz* / go) + ആൻ(*aan* / to) = പോകാൻ(*pOkaan* / to *go)*

3.2.3 Word compounding

In written text any letter can follow another letter. But in spoken language, it depends on the ability of sound generating organs. It is difficult to pronounce when some sounds come together. This is called hiatus (*vivruthi)*. In order to avoid this some changes are made to the sounds so that pronunciation becomes easy. Sometimes, for clarity of meaning or for beauty of sounds also these morphophonemic changes are made. The sound change when two words or suffixes join are called word compounding (*sandhi*). Malayalam is usually written in a way it is spoken. So Malayalam text contains a lot of compound

Table 3.3 Derivational morphology for verbs

Derived form	Derivation	Example
Verbal participle (+ve)	Past form+ ഉ / *u*	പറഞ്ഞു / *paRanjnju*
Verbal participle(-ve)	Past form + ആതെ / *aathe*	പറയാതെ / *paRayaathe*
Conditional participle(+ve)	Past form + ആൽ / *aal*	പറഞ്ഞാൽ / *paRanjnjaal*
Conditional participle(-ve)	Root+ആതിരുന്നാല് / *aathirunnal*	പറയാതിരുന്നാൽ / *paRayaathirunnaal*
Concessive participle(+ve)	Past form +ആലും / *aalum*	പറഞ്ഞാലും / *paRanjnjaalum*
Concessive participle(-ve)	Past form + ഇല്ലെങ്കിലും / *illenkilum*	പറഞ്ഞില്ലെങ്കിലും / *paRanjnjillenkilum*
Relative participle (+ve)	Past form – അ	പറഞ്ഞ / *paRanjnja*
Relative participle (-ve)	Root + ആത്ത / *aaththa*	പറയാത്ത / *paRayaaththa*
Infinitive	Root + ആൻ / *aan*	പറയാൻ / *paRayaan*

words. Formation of new words by combining a noun and a noun, noun and adjective, verb and noun, adverb and verb, adjective and noun and in some cases all the words of an entire sentence to reflect the semantics of the sentence are very common [39]. The complexity of compounding in Malayalam language can be understood from the following example:

Malayalam: ഞാനിന്നലെയൊരാനയെക്കണ്ടു

Transliteration: *njaaninnaleyoraanayekkandu*

English : I saw an elephant yesterday

Such sentences are common in Malayalam language. This kind of compounding is found in plenty in media passages and in poems. Sanskrit grammarians like Panini classified word compounding on the basis of the position in which the sound changes occur. According to this classification word compounding is classified into three types: word_medial (*padamadhyam*), word final (*padaanta*) and hybrid (*ubhaya*). In word medial, the compounding occurs between a stem and a suffix. Word final compounding occurs between two words. In hybrid both word medial and word final are involved. Malayalam compounding rules are also classified as: vowel *sandhi*, vowel-consonant *sandhi* and consonant-consonant *sandhi*.

മഴ (*mazha*/rain) +അല്ല (*alla*/not) =മഴയല്ല (*mazhayalla)* (vowel sandhi)

താമര (*thaamara*/lotus) + കുളം (kuLam/pond) = താമരക്കുളം *(thaamarakkulam /lotus pond)* (vowel-consonant sandhi)

നെല് (*nel* / rice) +മണി (*maNi*/seed) = നെൻമണി (*nenmaNi / rice seed)* (consonant-consonant sandhi)

Keralapanini has adopted a different classification for the compounding rules based on the changes occur during compounding. They are: *lOpa sandhi* (elison), *aagama sandhi* (addition), dvitva sandhi (germination or reduplication) and *aadEsa sandhi* (displacement or substitution). *lOpa sandhi* is that in which one of the sounds is lost, *aagama* is that in which new sound is added, dvitva is that in which one of the sounds geminates and *aadEsa* is that in which one of the sounds is displaced by another sound. The basis of all sound changes in s*andh*i is ease in pronunciation. It is difficult to pronounce consonants without vowels. When two consonants combine they are pronounced as one. Vowels

have well defined pronunciation. Consequently, sound changes are more essential in vowel combinations. The rules for compounding in each category are given in tables 3.4 to 3.7. The morphophonemic changes at the boundary depends on the ending vowel or consonant, the category of the first or the second word and the beginning vowel or consonant of the second word. The abbreviations used in the tables are: ss(v2)- vowel symbol corresponding to the beginning of the second word. Trans(x)- function which converts N,n,L,R etc. to Na,na,la,Ra respectively. Each entry in the substitution column has three fields. The first and second field gives the number of characters to be removed from the beginning of the first word and from the beginning of the second word. The third field gives the character to be placed at the boundary. The entry (1,1,ss(V2)) indicates when two words are compounded one character from the end of the first word, one character from the beginning of the second word are to be removed and the vowel symbol corresponding to the starting vowel of the second word has to be added at the boundary [40,41].

3.2.3.1 Elision (lopaSandhi)

Elision rules are given in Table 3.4. Elision of sounds occur in the following cases:

i) When followed by any Vowel, ് (unrounded u / *chandrakkala*) undergoes elision.

തണുപ്പ് (*thaNuppz* /chillness) + ഉണ്ട് (*uNtz* /is) = തണുപ്പുണ്ട് (*thaNuppuNtz* /there is chillness)

കാറ്റ് (*kaattz* /wind) +അടിക്കുന്നു (*atikkunnu* /blows) = കാറ്റടിക്കുന്നു (*kattatikkunnu* /wind blows)

ii) The ഉ / u occurring finally in verbs undergoes elision when followed by Vowel. The ഉ / u found in these forms, is by origin unrounded 'u', added for clarity in pronunciation. Unrounded 'u' is pronounced as rounded 'u' to provide emphasis to the sentence.

a) Regular

കണ്ടു (*kaNtu*/saw) +ഇല്ല (*illa*/no) = കണ്ടില്ല (*kaNtilla*/did not see)

കാണുന്നു (*kaaNunnu*/sees) +ഉണ്ട് (*uNTz*/have) = കാണുന്നുണ്ട്
(*kaaNunnuNtz*/is seen)

b) Irregular

കണ്ട് (*kaNtu* /saw) + ഓ (O / ?) = കണ്ടോ (*kaNtuvO* / *kaNtO* / saw?)

iii) The 'അ / a' at the end of അല്ല / *alla* and ഇല്ല / *illa*, and the the ഇ / i at the end of ആയി / *aayi* and പോയി / *pOyi* have elision when followed by Vowel.

അല്ല (*alla* /not) + എന്ന് (*ennu*/thus) = അല്ലെന്ന് (*allennu*/not thus)

ഇല്ല (*illa*/no) + എന്ന് (*ennu*/thus) = ഇല്ലെന്ന് (*illennu*/not thus)

iv) The mid verbal participle also means respectful persuasion. It's final അ/a also has elision.

വരിക (*varika*/come) + എടോ (*etO*/you) = വരികെടോ (*variketO/*
you come)

v) The final 'എ / e' of the permissive suffix അട്ടെ / *atte*, verbal participle ആതെ / a*athe*, possessive case marker ഉടെ/*ute* and conjunction ഊടെ / *uute* occasionally have lision when followed by vowel.

സാമിയുടെ (*saamiyute* / swamy's) + അനിയൻ

(aniyan / brother) = സാമിയുടനിയൻ (*saamiyutaniyan*/swamy's brother)

Table 3.4 Elision rules

	word1 ending	**word2 beginning v2**	**Substitution**	**Example**
1	് / z	Vowel	(1,1,ss(v2))	കാട് (*kaatz* /forest)+ ഇൽ (*il*/in) = *kaattil*
2	Word1=അല്ല/*alla*, ഇല്ല /*illa*	Vowel	(-,1,ss(v2))	അല്ല (*alla*/not) + എന്ന് (*ennz*/that) = അല്ലെന്ന്/*allennz*
3	Word1= ആയി/*aayi*, പോയി/ *pOyi*	Vowel	(-,1,ss(v2))	ആയി (*aayi*/became) + എന്ന് (*ennz*/that) = ആയെന്ന് *aayennz*
4	Word1= ഒരു /*oru*	Vowel	(1,1,ss(v2))	ഒരു(*oru*/a)+ആന (*aana*/ elephant) = ഒരാന / *oraana*
5	Word1= ഉ / *u*, ഉം/ *um*, Word1 category = verb	Vowel	(1,1,ss(v2)	വന്നു (*vannu*/came) + ഇല്ല (*illa*/not) = വന്നില്ല / *vannilla*

3.2.3.2 Addition (aagamasandhi)

It is inconvenient to pronounce two vowels together as the vowels have independent pronunciation. The difficulty is overcome by inserting യ / *ya* or വ / *va* in between them. If the preceding vowel is palatal അ, ആ, ഇ, ഈ, എ, ഏ,

ഐ (*a, aa, i, ii, e, ee or ai.)* 'യ */ya'* is inserted and if it is labial (ഉ, ഊ, ഒ, ഓ / *u, uu, o,O)* വ/*va* is inserted. Addition rules are given Table 3.5.

i) Compounding in Palatals

തീ (*thii*/fire) +ആട്ട് (*aattz*/dance) = തീയാട്ട് (*tiiyaattz*/fire dance)

കൈ (*kai*/hand) + ഉണ്ട് (*untz*/has) = കൈയുണ്ട് (*kaiyuntz*/has hand)

ii) Compounding in Labials

തട (*thata*/rub) + ഉന്നു (*unnu*/does) = തടവുന്നു (*thatavunnu*/rubs)

ചാ (*chaa*/die) + ഉന്നു (*unnu*/does) = ചാവുന്നു (*chaavunnu*/dies)

ii) In verbs ഏ/E may be substituted by ന /na. ന /na can also occur for making the utterance more pleasing.

കാട്ടി (*kaatti*/showed) + ഏൻ (*En* / I) = കാട്ടിനേൻ (*kaattiyEn* or *kaattinEn/*

I showed)

കരുതി (*karuthi)* +അ(*a)*=കരുതിന(*karuthina*/*karuthiya*)(that was thought)

iii) The demonstratives അ / *a,* ഇ / *i* and e are referred to as *chuttezhuththu* in Tamil. Though these three vowels are palatals വ / v instead of യ / y is added when followed by a vowel, thus providing exception to the earlier rule.

ഇ(i/this) + അൻ (*an* /he) = ഇവൻ (*ivan*/he)

It is v that gets added when this pronounce are used as qualifiers as well.

അ(a/that) + ഇടം (*itam* /place) = അവിടം (*avitam* /that place)

iv) The relative participle suffix അ / a also is demonstrative. So there too വ / va should be added.

ചെയ്ത (*cheyhtha*/which was done) + അൻ (*an*/he) = ചെയ്തവൻ

(*cheythavan*/he who did)

v) In words ending in the long vowels aa, ii, uu, ee, ai, oo, y or v is generally added. This is confined to a few words. There is no specific grammatical rule regarding this phenomenon.

കാ *kaa* - കായ് (*kaay*/fruit)

പാ *paa* - പായ് (*paay*/mat)

vi) യ/*y*a is added to palatal vowels then followed by suffixes beginning with ക/ *k*a. There is no addition of യ/ya if the ക്ക/*kk*a is not the beginning of a suffix or the preceding vowel is not palatal.

തല (*thala*/head) + ക്ക് (*kku*/to) = തലക്ക് (*thalaykku*/to head)

ചാടി (*chaati* /jump) + കടന്ന (*katannu*/crossed) = ചാടിക്കടന്ന

(c*haatikkatannu*/crossed jumping)

3.2.3.3 Reduplication (dvitvasandhi)

Vowels do not have cluster information. Each vowel is capable of free pronunciation with no obstructions by the tongue in the points of articulation. Even in combination, it is not marked as in the case of consonants. Instead of germination they have length; instead of clustering they have diphthongs. So

Table 3.5 Addition rules

	Word1 ending/word1	Word2 beginning v2	substitution	Example
1	Vowel	Vowel	(-,1,ya+ss(v2)	പന (*Pana* /palm)+ ഓല (*Ola* /leaf) = പനയോല panayola
2	ഉ /*U*, ഊ /*uu*	Vowel	(-,1,va+ss(v2))	മധു (*Madhu*/Madhu) + ആണ് (*aaNz*/only) = മധുവാണ് /*madhuvaanz*
3	Word1=അ /*a*, ഇ/ *i*, ഏ /*e* (*chuttezhuththz*)	Vowel	(-,1,va+ss(v2))	അ /a+അർ/ *ar* = അവർ (*avar*/they)
4	അ/ *a*	Consonant	(-,-,ss(aa))	കല (*Kala*/arts) + മേള (*mELa* /fesival) = കലാമേള *kalaamELa*

the rules of germination are restricted to consonants. In Malayalam, germination is more in tense consonants and less in lax consonants.

i) When two words combine in which the first is the qualifier and the qualified, the tense consonants initial to the second word geminates.

തല (*thala*/head) + കെട്ട് (*kettu*/tie) = തലക്കെട്ട് (*thalakkettu*/turban)

താമര (*thaamara*/lotus) + കുളം (*kuLam*/pond) = താമരക്കുളം (*taamarakkuLam* /lotus pond)

ii) *dvandvasamasam* does not involve combination of the qualifier and the qualified. So this gemination does not occur in it.

കൈ (*kai*/hand) + കാല് (*kaal*/leg) = കൈകാല് (*kaikaal*/hand and leg)

രാമ (*raama*/rama) + കൃഷ്ണന്മാർ (*krishNanmaar*/krishna) = രാമകൃഷ്ണന്മാർ (*raamakrishNanmaar*/Rama and Krishna)

iv) There will be gemination in sentences also, namely in the tense consonant after the three verbal participles, *mun, tan* and *paaksikam*, the pseudo case എ /e used in the locative sense and the case suffixes ആൽ / aal, ഇൽ / il, കൽ / kal.

പോയി (*pOyi/*went) + പറഞ്ഞു (*paRanjnju*/said) = പോയിപ്പറഞ്ഞു (*pooyippaRanjnju*/went and said)

മനസ്സാൽ (*manassaal*/by will) + കൊടുത്തു (*kotuththu*/gave) = മനസ്സാൽ ക്കൊടുത്തു *Manassaalkkotuththu*/gave by will)

v) The pronouns a/അ, ഇ/i and e/എ are called *chuttezhuththu*. The term is meaningful, they being the base for words that point out persons. അവൻ (a-*van* /he), ഇവൻ (*i-van*/he) and ഏവൻ (*ee-van*/who). All consonants that follow chuttezhuththu are geminated.

അ(*a* /that) +കാലം (*kaalam*/period) = അക്കാലം (*akkaalam*/that period)

ഇ (*i* /this) + കണ്ട (*kanta*/which was seen) = ഇക്കണ്ട (*ikkanta*/this which was seen)

vi) The nasals as well as യ/y and ള/la occurring finally in *Ekamaathra* (single beat) bases will geminate. There is no single beat word in Malayalam.

നെയ് (*Ney*/ghee) + ആറ് (*aaRu*/river) = നെയ്യാറ് / *neyyaaRu*

vii) To words ending in consonants, unrounded ഉ /u has to be added. If it is a *chillz*, this addition is optional.

വയസ്സ് (*vayasz* / age) + കുറഞ്ഞു (*kuRanjnju*/reduced) = വയസ്സുകുറഞ്ഞു

(*vayasukuRanjnju*/age reduced)

3.2.3.4 Substitution (aadeesasandhi)

i) When consonants of the t-class combine with those of the t class they become t-class. With the c-class and class they become c-class and # class ഞ്ച, ണ്ട, ന്ത (ncha, nta, ntha)etc. respectively. Substitution rules are given in Table 3.6.

തണ് (*thaN* /cool) + താർ (*thaar*/flower) =തണ്ടാർ (*thantaar*/cool flower)

ii) When l followed by consonants of the t-class, it is substituted by റ്റ / tt. Similarly when l followed by t-class consonants it is substituted by ട / t. This change is called *vinaamam* which means lowering in position.

വിൽ (*vil*/to sell) +തു (*tu*/did) = vit + tu = വിറ്റു (*vittu*/sold)

കേൾ (*keel*/to hear) +തു (*tu*/did) = keet + tu = കേട്ടു (*kEttu*/heard)

iii) The ൻ/ *n* in the words മുൻ/ mun, പിൻ / *pin* and പൊൻ / *pon* changes to ൽ when followed by voiceless stops.

പിൻ (*pin* /back) + പാട്ട് (*paattu*/song) = പിൽപ്പാട്ട് (*pilppaattu*/ background song)

പൊൻ (*pon*/gold) + കുടം (*kutam*/pot) = പൊൽക്കുടം (*polkkutam*/ golden pot)

iv) The voiceless stops initial to a suffix is substituted by a Nasal except when the suffix begins with ക/ka or ട/ta. The substitution by Nasal according to this rule is called Nasal Assimilation.

പറഞ്ഞ് (*paRan/say)* + ച്ചു (*chu/past tense)* = പറഞ്ഞു (*paRanjnju*/said)

ഉണ് (*Un*/to eat) + തു (*thu*/did) =ഉണ്ടു (*uNtu*/ate)

v) 1) *Anusvaaram*(ം / am) becomes clear when vowel follows. In combination with any vowel it changes to മ/ma.

മരം (*maram*/tree) + അല്ല (*alla*/not) = മരമല്ല (*maramalla*/nottree)

മരം (*uN*/to eat) + തു (*thu*/did) =ഉണ്ടു (*uNtu*/ate)

2) When *anusvaaram* is followed by class sounds changes to ങ, ഞ, ണ, ന, മ *(nka, ncha, nta, ntha, mpa)* respectively.

വരും(*Varum/*coming) + കാലം(*kaalam/*period) = വരുങ്കാലം/*varunkaalam* (coming period)

പെരും (*perum*/big) + പറ (*paRa*/drum)=പെരുമ്പറ(*perumpaRa*/huge drum)

3) *anusvaaram* changes to വ/v when it is followed by *samuchchaya* (additive) *nipaatham* ഉം /um. anusvaaram in the future tense marker ഉം /um also changes to v when followed by suffix.

കുലം (*kulam*/family) + ഉം (*um*/and) =കുലവും (*kulavum*/family and)

വാരം (*vaaram*/*week*) + ഉം (*um*/and) = വാരവും (*vaaravum*/week and)

4) anusvaaram changes to ത്ത/*ththa* when followed by any suffix beginning with a vowel. If the suffix following is the case marker ഓട് / *Otu* this substitution is not regular.

ധനം (*dhanam*/wealth) + എ (*e*/of) = ധനത്തെ (*dhanaththe*/of wealth)

തുലാം (*thulaam*/support) + ഇന്റെ (*inte*/of) = തുലാത്തിന്റെ

(*thulaaththinte*/support of)

5) ം / am changes to ന്/n when followed by plural marker കൾ / *kaL* , due to assimilation.

മരം (*maram*/tree) + കൾ (*kaL*/plural) =മരങ്ങൾ

(*marankaL*/*marangngaL* /trees)

The assimilation is found also in

കുളം (*kuLam*/pond) + കര (*kara*/shore) = കുളങ്ങര (*kuLangngara*/ pond's shore)

Table 3.6 Substitution rules

	Word1 ending	Word2 beginning	Substitution	Example
1	ം / am	Vowel(v2)	(1,1,മ/ma+ss(v2))	മരം+ അല്ല=മരമല്ല maram + alla= maramalla
2	ം / am	Vowel(v2), word class=case marker	(1,1,ത്ത/ththa +ss(v2))	ധനം+എ = ധനത്തെ dhanam+e=dhanaththe
3	ൺ / N	ട,ത,ന/ ta,tha,na	(1,1, ണ്ട/nta, ണ്ട/nta, NNa resp.)	ത ൺ +താർ =തണ്ടാർ thaN+thaar=thaNTaar
4	ം / am	ക ച ട ത പ ka, cha, ta, tha, pa	ങ്ക, ഞ്ച, ണ്ട, ന്ത, മ്പ, ങ്ങ (nka, ncha, nta, ntha, mpa, ngnga)	മരം+കൾ=മരങ്ങൾ maram+kaL=marangngaL
5	ൻൺൾ ൽ ർ (N, n, L, l, r)	Vowel(v2)	(-,1,ന, ണ, ള, ല, ര / Na, na, La, la, ra) resp.+ss(v2))	അവർ + ഇൽ = അവരിൽ avar +il= avaril

3.2.3.5 Sanskrit compounding

Since Malayalam had adopted many words from Sanskrit many of the compounding rules found in Sanskrit are common in Malayalam sentences. Sanskrit compounding is classified as vowel compounding *(swarasandhi)* which happens when two vowels join and consonant compounding *(vyanjanasandhi)* which happens when two consonants join. Vowel sandhi is

further classified into i)*deeghasandhi, ii)guNasandhi, iv)vridhisandhi , and iii) yaNsandhi*. Sanskrit compounding rules are given in Table 3.7.

i) *Deerghasandhi*

When the vowels അ /a, ഇ/ i, ഉ /u join with അ, ആ (a,aa), ഇ,ഈ (i, ii), ഉ, ഊ *(*u,uu) respectively, the vowels changes to ആ /aa, ഈ /ii, ഊ / uu respectively.

Most common for compound nouns which join two nouns or between an adjective and a noun.

കഥ (*kadha/*story*)* + അന്ത്യം (*anthyam/*end) = കഥാന്ത്യം

(*kadhaanthyam/story ending)*

കടം (*katam/loan)* + ആശ്വാസം (*aasvaasam/help)* = കടാശ്വാസം/

kataasvaasam(relief loan)

ii) GuNasandhi

When അ, ആ (a/aa) is followed by ഇ / *i* ,ഈ / *ii*, ഉ/ *u*, ഊ / *uu* it changes to ഏ / *E*, ഓ / *O* respectively.

മഹാ (*maha*/great) + ഇന്ദ്രൻ (*indran*/indra) =മഹേന്ദ്രൻ (*mahEndran*/great Indra)

കല (*kala*/art) + ഉപാസന (*upaasana*/devotion) = കലോപാസന

(*KalOpaasana*/devotion for art)

ii) Vridhisandhi

a) When അ/ *a*, ആ */aa* is followed by ഏ /*E* , both sounds join to form ഐ /*ai.*

b) When അ/ *a* , ആ */aa* is followed by ഊ/ *oo*, both sounds join to form ഔ/ *ou.*

സദാ (*sada*/always) + *Evam* (all) = സദൈവം (*sadaivam* /always)

iii) YaN sandhi

When എ *e* is followed by അ / a, ആ / aa, യ / *y* is added. When എ / *e* is followed by ഉ / *u*, ഊ / *uu*, വ / va is added. It is a form of addition compounding (*aagamasandhi*) in Malayalam.

അതി (*athi*/very) + ആവശ്യം (*aavaSyam*/need) = അത്യാവശ്യം

(*athyavaSyam*/most needed)

Table 3.7 Sanskrit compounding rules

	Word1 end	Word2 beg.	Substitution	Example	Sandhi
1	അ /a	അ/a, ആ /aa	(-,1,ss(aa))	കഥ + അന്ത്യം = കഥാന്ത്യം *kadha+anthyam=kadhaanthyam*	Deergha Sandhi
2	ഇ /i	ഇ /i, ഈ /ii	(1,1,ss(ii))	കവി + ഈശ്വരൻ =കവീശ്വരൻ *kavi+iisvaran=kaviisvaran*	
3	ഉ /u	ഉ/u,ഊ/uu	(1,1,ss(, ഊ/uu))	ഗുരു +ഉപദേശം = ഗുരുപദേശം *guru+upadeSam=guruupadeSam*	
4	അ /A	ഇ/i,ഉ/u	(,1,ss(ee/oo))	മഹാ+ ഇന്ദ്രൻ = മഹേന്ദ്രൻ *maha+indran=mahEndran*	Guna Sandhi
5	ആ /aa	ഏ/E	(1,1,ss(ai))	സദാ+ ഏവം = സദൈവം sada+eevam=sadaivam	Vridhi Sandhi
6	ഇ/i	Vowels except ഇ	(1,1,ss (യ / ya))	അതി+ആവശ്യം =അത്യാവശ്യം *athi+aavaSyam=athyavaSyam*	Vridhi Sandhi
7	ഉ/u	അ/ a, ആ /aa	(1,1,ss(വ/ va))	വധു + ആഗമനം = വധ്വാഗമനം v*adhu+aagamanam=vadhvaagamanam*	YaNa Sandhi

3.3 Morphological analysis

Morphological analysis is the first phase in any natural language processing application. The output of this phase is used by the syntax analysis phase following it. Morphological analyser should accept as input a surface form of a word in a language such as spies and return an underlying form divided into morphemes, namely spy+s, plus a gloss string such as N+PLURAL [37]. Here, N and PLURAL are the category information regarding the morphemes spy and s, the constituents of the input.

3.3.1 Two level morphology

KimmoKoskenniemi in 1983 proposed a two level morphology for word form recognition and generation [42,43]. The model was based on the traditional distinction that linguists make between morphotactics and morphophonemics. For example, the word chasedis analyzed morphotactically as the stem chase followed by the suffix -ed. The addition of the suffix -edapparently causes the loss of the final ‘e’ of chase; thus chase and chas are allomorphs or alternate forms of the same morpheme.

Two level morphology is based on three ideas:

i) Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules.

ii) The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time.

iii) Lexical lookup and morphological analysis are performed in tandem.

In Koskenniemi's two level model a word is represented as a direct, letter-for-letter correspondence between its lexical or underlying form and its surface form. For example, the word chased is given this two-level representation (where + is a morpheme boundary symbol and 0 is a null character):

Lexical form: c h a s e + e d

Surface form: c h a s 0 0 e d

Koskenniemi's two-level morphology was the first practical general model in the history of computational linguistics for the analysis of morphologically complex languages. Karttunen et.al completed the project and published a collection of papers on the topic, along with Lisp code. They called it the KIMMO system and it inspired many other KIMMO implementations. The most popular of these is PC-KIMMO, a free C implementation from the Summer Institute of Linguistics [44,45,46,47].

The KIMMO parser had two analytical components: the rules component and the lexical component, or lexicon. First, the rules component consisted of two-level rules that accounted for regular phonological or orthographic alternations, such as chase versus chas. Second, the lexicon listed all morphemes (stems and affixes) in their lexical form and specified morphotactic constraints. The Generator would accept as input a lexical form such as spy+sand return the surface form spies. The Recognizer would accept as input a surface form such as spies and return an underlying form divided into morphemes, namely spy+s, plus a gloss string such as N+PLURAL.

But it had a serious deficiency: it could not directly determine the part of speech of a word or its inflectional categories. For example, given the word enlargements, PC-KIMMO could tokenize it into the sequence of morphemes

en+large+ment+sand gloss each morpheme, but it could not determine that the entire word was a plural noun. This meant that PC-KIMMO was not adequate to act as a morphological front end to a syntactic parser its most desirable application.

3.3.2 Computational models for morphological analysis

Various NLP research groups have developed different algorithms and data structures for morphological analysis. Some of the algorithms are language dependent and some of them are language independent. The various techniques are:

1) Finite State Transducers (FST).

2) Suffix stripping approach

3) Corpus Based Approach

4) Paradigm Based Approach.

3.3.2.1 Finite state transducers

An FST is represented as a two tape automaton. We can combine lexicon, orthographic rules and spelling variations in the FST to build a morphological analyzer. The simplest finite state machine is a finite state automaton (FSA), which recognizes (o r generates) the well-formed strings of a regular language.

A Mealy machine extension FST can be formally defined as:

Q : a finite set of N states q0,q1,q2,..qn

Σ : a finite alphabet of complex symbols. Each complex symbol is composed of an input output pair i:o; one symbol i from an input

alphabet I and one symbol o from an output alphabet O. I and O may each include ε.

Q0: the start state

F : the set of final states, F C Q

δ (q,i:o): the transition function between states. δ is a relation from Q X Σ to Q

A sample transducer which can work as a morphological parser for English nominal number inflection is shown in Figure 3.1. When the states are traversed from start state to final state with an irregular noun like geese the output will be goose + N + PL.

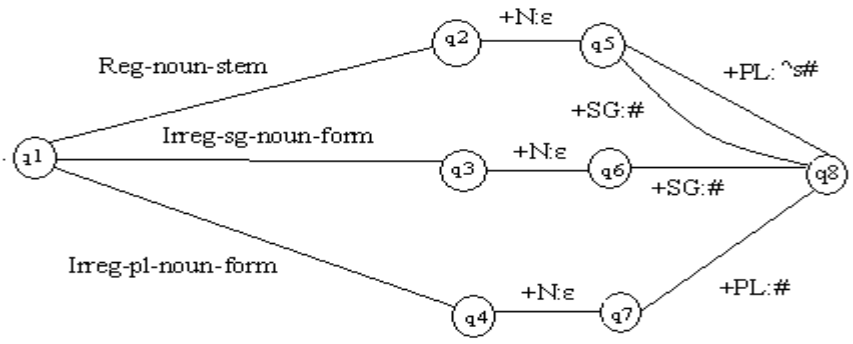


Figure. 3.1 Transducer for English nominal number inflection

A word grammar based morphological analyzer has been discussed [48]. The model integrates the two level method and a unification based formalism. The

system was tried for Basque texts. It separates the sequential and non sequential constraints. Sequential morphotactic constraints are applied in the segmentation phase and the non sequential constraints in the final feature recombination phase. The two level formalism takes care of the morphographemics and sequential morphotactics. Unification based word grammar combines the grammatical information defined in morphemes to tackle complex morphotactics. This design allowed full coverage analyzer that processes efficiently unrestricted texts in Basque.

A Unification based method has been discussed [49,50] for morpho syntactic parsing of agglutinative and inflectional languages. The system called Humor 99 has already been integrated with a variety of industrial applications. It can handle almost all agglutinative languages like Hungarian, Turkish, Estonian etc. and inflectional languages like Polish, Czech, German etc. very effectively.

Ramaswamyveerappan et. al. has developed a Kannada Morphological analyzer and generator using Trie [51,52,53]. The tool uses a trie for storing roots and suffixes. It combines paradigm approach with suffix stripping. No full fledged Morphological analyzer for Kannada has been developed and the performance of the tool is encouraging.

Girinath Jha has developed an inflectional Morphology analyzer for Sanskrit [54]. It uses sandhi free Sanskrit Unicode text as input. The system labels each word as subanta, tinanta and avyaya. There after each of the category is further categorized using a stem and suffix database.

Kemal Oflazer has discussed about a lenient morphologic analyser [55]. The system augments the two level morphology rule so that word forms with

violations of some of the two level constraints can be analysed and ranked. The problem was motivated by the languages Turkish and Basque. In Turkish, the imported words violate the assumption of one correspondence between pronunciation and orthography and Basque words usually have errors.

3.3.2.2 Corpus Based Approach

Corpus is a large collection of written text belong to a particular language . Raw corpus can be used for morphological analysis. It takes raw corpus as input and produces a segmentation of the word forms observed in the text. Such segmentation resembles morphological segmentation.

Morfessor1.0 developed in Helsinki University is a corpus based language independent morphological segmentation program.

The LTRC Hyderabad successfully developed a corpus based morphological analyzer [56]. The program combines paradigm based approach as well corpus based approach. It uses a low coverage morphological analyzer and combines unsupervised learning of paradigm of unknown words using a corpus. When the morph analyzer cannot analyze a given word the morph guessing package returns a pair of stem and paradigm. The guessing is based on suffixes. Thus it improves the performance of a morphological analyzer from 32% to 63%.

Unsupervised learning of morphology for building Lexicon for Assamese, a highly inflectional language [57,58,59,60]. It tries to create a dictionary and a morphological rule base using a text corpus which was developed for the purpose.

A statistical method using N gram approach has been tried for modeling morphologically rich languages using split words and unstructured dependencies [61,62]. The method assumes that n-1 tokens that determine the probability of a given token can be chosen anywhere in the sentence rather than preceding n-1 positions. The system reduces the perplexity of the standard N gram model by 24%. It was designed for Turkish.

3.3.2.3 Paradigm Approach

A paradigm defines all the word form of a given stem and also provides a feature structure with every word form. The paradigm based approach is efficient for inflectionally rich languages. The ANUSAARAKA research group has developed a language independent paradigm based morphological compiler program for Indian Languages.

This or a variant of this scheme has been used widely in NLP. The linguist or the language expert is asked to provide different tables of word forms covering the words in a language. Each word-forms table covers a set of roots which means that the roots follow the pattern (or paradigm) implicit in the table for generating their word forms. Almost all Indian language morphological analyzers are developed using this method. Based on paradigms the program generates add delete string for analysis. Paradigm approach rely on findings that the different types of word paradigms are based on their morphological behavior.

Words are categorized as nouns, verbs, adjectives, adverbs and postpositions. Each category will be classified into certain types of paradigms based on their morphophonemic behavior. For example noun '*maram*'(tree) belongs to a

paradigm class is different form '*vanitha*'(lady) which belongs to a different paradigm class as they behave differently morphophonemically.

•*maram, maraththe, maraththil, maraththinte, maraththOtu, maraththilninnum*

•*vanitha, vanithaye, vanithayil, vanithayute, vanithayOtu, vanithayilninnum*

There is a Dictionary of Roots, where along with the roots the types and grammatical information that is common to all the associated endings (that is, word forms), can be stored. This leads to efficient storage because there is only one paradigm table for a class of roots rather than separate word forms table for each root. A sample dictionary of roots is shown in Table 3.8. The paradigm for the word given in first column is given in the second column and its gender is given in the third column.

Table 3.8 Dictionary of roots

Root	Type	Gender
aabharaNam	*(n, maram)*	N
katam	*(n, maram)*	N
rama	*(n, vanitha)*	F

Every language has its own dictionary of indeclinable words. Using this dictionary, paradigm tables and dictionary of roots, one can generate a word form when a root and desired feature values are given. The dictionary of roots

should be kept sorted since searching for an item in a sorted list is much faster. It is set up so that given a root and the desired features; one can locate the right table and then lock up the right entry.

3.3.2.4 Suffix Stripping

Suffix Stripping is another one method used for analyzing the words in a language. In highly agglutinative languages such as Malayalam, a word is formed by adding suffixes to the root or stem. Morphologically highly complex words exist in such languages, which are formed by continuously adding suffixes to the stem. Suffix Stripping method make use of this property of the language, i.e., having complex suffixes attached to the stem. Once the suffix is identified, the stem of the whole word can be obtained by removing that suffix and applying proper morphophonemic rules.

Thus the general format of the morphological analyzer of Malayalam is Word stem + suffixes. The two main grammatical category of Malayalam are Noun and Verb. Stem is either a verb stem or a noun stem. The suffix stripping method makes use of a stem dictionary (for identifying a valid stem), a suffix dictionary, containing all possible suffixes that nouns/verbs in the language can have (to identify a valid suffix), morphotactics rules and morphophonemic rules or *sandhi* rules. Nouns can have case markers as their suffixes. Normally verbs inflect for tense, aspect and mood. Thus the verbal forms are stripped into suffixes which denote different tenses, moods and aspects. For example, the verbal form paaTikkoNTirunnuwill be analysed as follows:

i - ‘conjunctive particle’
koNtiru - ‘continuous aspect’
nnu - ‘past tense’
paaT - ‘sing’

C-DAC, Thiruvananthapuram (former ER&DC), has developed a morphological analyzer for Malayalam using suffix stripping approach.

Sumam, *et. al*., Cochin University of Science and Technology also has developed a morphological analyzer using the same approach. It has a morphological analyzer which categorises the words based on the suffixes. For nouns the morphology due to gender, number and case are considered. For verbs, the morphology due to tense, aspect and mood are considered. The system mainly handles inflectional morphology for verbs and nouns. Some derivational morphology for nouns and verbs are also considered [38].

It use separate tables for the lexical categories nouns, pronouns, verbs and modifiers. The information stored with each category are:

> Noun : root, gender, property (human, non human), type (abstract noun, collective noun, common noun)
>
> Pronoun: root, gender, number, person, type (personal, reflexive, interrogative, indefinite)
>
> Verb: root, *kaaritham / akaaritham*

The system has not considered compound nouns. It fails to represent linguistic generalization. Use of a separate table for suffixes saves lexicon space.

Rajeev R.R. *et. al.*, Kerala University, has developed a morphological analyzer for Malayalam as part of a Malayalam_ Tamil translator. They also have used a suffix stripping approach. It does not have a *sandhi* splitter to split into

morphemes when more than one lexical category is joined to form a word. It takes as input a Malayalam word and finds the root and suffix from the given word. It then converts the root and suffix into Tamil using a bilingual dictionary and runs a morphological generator for Tamil to find the corresponding word in Tamil. The output is generated in Tamil script using Unicode [63].

Other existing morphological analysers and generators are: TelMOre for Telugu [64], Bangla morph generator [65], Tamil morphological analyzer [66] and Arabic Morph generator from interlingua [67].

3.4 Conclusion

This chapter discussed the various approaches used for morphological analysis for natural language sentences. The survey revealed that very few works are reported on morphological analysis for Malayalam language. The morphological variations found in Malayalam language also were discussed in this chapter. The word compounding rules identified were used in the morphological analysis phase of the prototype machine translation system. The same set of compounding rules is used in forming inflections and derivations of a word. The following chapter discusses the computational models for parsers and the various activities of the parsing phase of a machine translation system.

Chapter 4

LANGUAGE SYNTAX AND PARSING

4.1 Introduction

The parser for a language needs the part of speech tag set for various words in the sentence, the groups of co-occurring words known as word chunks, the structure of sentences in a language and the hierarchical dependencies of chunks in sentences. A detailed description of various classes of sentences in a language is given with specific examples from Malayalam language. The need for syntactic structure transfer between two languages is also explained in this chapter. Finally, the different computational models for parsers are also discussed.

4.2 Syntactic structure of languages

In order to arrive at a computational grammar for the language the set of word classes (Part Of Speech tagset), chunk tagset and the hierarchical dependencies among the chunks are needed. This requires a careful analysis of the different classes of sentences in the language and the hierarchical dependencies among the word groups in a sentence. A study of the syntactic difference between the source and target language is also required for applications like machine translation to produce the target sentence in the correct structure.

4.2.1 Part of Speech

Words are divided into classes called parts of speech (POS) according to their use in a sentence. The significance of POS tags of words in the present day NLP is widely known. POS tagging is an important activity for investigators of

natural language processing, speech recognition and other related areas. It proves to be a basic building block for constructing statistical models and it needs an annotated corpora for automatic processing of natural languages [68,69]. Annotation of corpora can be done at various levels viz, part of speech, phrase/clause level, dependency level, etc. Part of speech tagging forms the basic step in any natural language processing application. Chunking can form the next level of tagging [70,71].

A 'word' in a text carries the linguistic knowledge such as a) grammatical category and b) grammatical features. The POS tag should be based on the 'category' of the word and the features such as gender, number, person etc. can be acquired from the morphological analyser.

POS tags are very important for words in language translation task. Same word can have different meanings in different contexts. Word sense disambiguation is required to find the correct meaning of the word in the context. POS tags of words can be used as a means for word sense disambiguation.

Malayalam : രാമു കരയുന്നു (Ramu is crying)

Transliteration: *raamu karayunnu*

Morphemes: രാമു / *Ramu* (Noun)+ കര / *kara* (Noun/verb) + ഉന്നു / *unnu* (Verbal suffix)

The word കര / *kara* can have two tags: noun and verb root. Depending on the tags the word can be translated as either land or cry. Parser uses these tags for word sense disambiguation. When it tries to build the parse tree for the sentence using the lexicalized grammar it will be able to identify the word കര / *kara* as a verb root and finally the correct meaning “cry” can be found out.

4.2.2 Selection criteria for POS tagset

The tagset is arrived at considering the following factors:

a) it should be comprehensive and complete
b) it should be simple.
c) the tags are to be used for word sense disambiguation for getting the correct translation by the parser,

Major issues that are encountered in arriving at a POS tagset are:

1. Fineness v/s Coarseness in linguistic analysis
2. New tags v/s tags from a standard tagger

i) Fineness v/s Coarseness

One of the issues while deciding a tagset for annotating the input text is the 'fineness' or 'coarseness' in linguistic analysis. A decision had to be taken whether the tags will account for finer distinctions of the various features of the parts of speech. It has to be decided if plurality, gender and such other information will be marked distinctly or only the lexical category will be marked.

It is preferred to come up with a tagset which avoids fine distinctions. The motivation behind this is to have less number of tags since less number of tags leads to simplicity. Accuracy of manual tagging is higher when we have lesser number of tags.

However, a matter of greater concern concern is regarding the grammatical and other relevant linguistic knowledge which is encoded in a word, particularly in agglutinating languages (which several Indian languages are). If tags are too coarse, some crucial information for further processing might be missed out.

Too coarse an analysis is also not of much use. A balance between fineness and coarseness is required.

ii) New tags v/s tags from a standard tagger

While deciding the tags for a tagger we can either come up with a totally new tag set or take any other standard tagger as a reference and make modifications in it according to the objective of the new tagger. The Penn tag set is a standard tag set used for English. Many tag sets designed after this have been a variant of this tag set (e.g. Lancaster tag set). While deciding the tags for our work, the IIIT tag set has been used as a benchmark. We have used our own tag names for convenience. New tags have been introduced wherever Penn tags have been found inadequate for our translation problem.

4.2.3 Clauses

A group of words that forms part of a sentence and has a subject and a predicate of its own is called a clause. Clauses are classified into :

1. Adverb clause
2. Adjective clause
3. Noun clause

The clause markers in Malayalam are given Table 4.1.

4.2.3.1 Adverb clause

An adverb clause is a group of words which contains a subject and a predicate of its own and does the work of an adverb. They are further classified into:

a) Adverb clause of time

Adverb clauses of time are introduced by the subordinating conjunctions അപ്പോഴൊക്കെ / *appOzhokke*, അപ്പോൾ / *appOL*, കഴിഞ്ഞു / *kazhinjnju*, അതിനു ശേഷം / *athinuSEzham*, അതിനുമുന്പ് / *athinumunpu*, അതുതൊട്ട് / *athuthottu*, അതുമുതൽ / *athumuthal* etc.

1. നീ വന്നപ്പോൾ മഴ പെയ്തു

nee vannappoL mazha peythu

(It rained when you came)

2. നീ വന്നത് മുതൽ മഴയാണ്

nee vannathu muthal mazhayaanu

(It had been raining since you came).

b) Adverb clause of place

Adverb clause of place are introduced by the subordinating conjunctions എവിടെ / *evite* and the verb ending ഉം/ um, എവിടെ / *evite* and ഓ / o suffix with the verb.

രാമു എവിടെ പോയാലും അവന്റെ പട്ടി കൂടെക്കാണും

raamu evitepoyalum avante patti kootekkaaNum

(Ramu's dog will be with him wherever he goes)

c) Adverb clause of purpose

Adverb clause of purpose are introduced by the subordinating conjunction ആൻ വേണ്ടി / *aakaanvEnti.*

രാജാവാകാൻ വേണ്ടി വീരൻ മകനെ കൊന്നു

raajavakaanvEnti veeran makane konnu.

(Veeran killed his son to become the king)

d)Adverbal clause of reason or consequence

Adverb clause of reason are introduced by the subordinating conjunction അതുകൊണ്ട് / *athukontu*, അതുകാരണം / *athukaaraNam.*

രാമു പഠിക്കാത്തത് കാരണം പരീക്ഷയിൽ തോറ്റു

Raamu paThikkathathu kaaraNam pareekshayil thOttu.

(Ramu failed in the exam because he didn't study)

e) Adverb clause of condition

Adverb clauses of condition are introduced by the subordinating conjunctions ആൽ / *aal*, ആലും / *aalum.*

നീ പഠിച്ചാൽ നിനക്ക് വിജയിക്കാം

nee paThichchaal ninakku vijayikkam

(If you study you can win)

f) ***Adverb clause of comparison***

Adverb clause of comparison is two types:

i) *Adverb clause of comparison of degree :* In this the subordinating conjunction അതിനെക്കാൾ / *athinEkkaL*, അതുപോലെ / *athupOle*.

നീ വിചാരിക്കുന്നതുപോലെ അവൻ മിടുക്കനല്ല

nee vicharikkunnathupOle avanmitukkanalla

ii) *Adverb clause of comparison of manner* : In this the subordinating conjunction used is അതുപോലെ / *athupOle*

ഞാൻ വിചാരിച്ചതുപോലെ കാര്യങ്ങൾ നടന്നു

njaanvichaarichchapOle kaaryangngaL natannu

(All happened as I planned)

g) Adverb clause of supposition or concession

Adverb clause of supposition or concession are introduced by the subordinating conjunctions ആലും / *aalum*, എങ്കിലും / *enkilum.*

രാമൻ നേരത്തെ പോയെങ്കിലും ഡോക്ടറെ കാണാൻ പറ്റിയില്ല

Raman nErathe pOyenkilum dOctoRe kaaNaan pattiyilla

. (Even though Raman went early he couldn't see the doctor)

4.2.3.2 Adjective clauses

An adjective clause in a complex sentence is a subordinate clause which does the work of an adjective and so qualifies some noun or pronoun in the main clause.

1. An adjective clause is introduced by a relative pronoun or by a relative adverb. The adjective clause is marked by a relative participle like ചെയ്ത /*cheytha*, പറഞ്ഞ / *paRanjnja* etc.

ജനല് പൊട്ടിച്ച കുട്ടി ഇവനാണ്

janaala pottichcha kutti ivanaaNz.
(It is this boy who broke the window)

അപകടം ഉണ്ടായ വീട് ഇവിടെ അടുത്താണ്

apakatam undaaya veetz ivite atuththaaNz.

(The house where the accident took place is nearby)

2. An infinitive with a suffix ആൻ/ aan is also often used as the equivalent of an adjective clause.

എനിക്ക് പഠിക്കാൻ പുസ്തകം വേണം

enikkz paThikkaan pusthakam vENam.

(I want a book to study)

4.2.3.3 Noun clause

Noun clause is a subordinate clause which does the work of a noun in a complex sentence. It is further classified into:

a) The subject of a verb: In this use the clause ends with a verb with the suffix അത് / *athu.*

ഞാൻ പറഞ്ഞത് സത്യമാണ്

njaan paRanjnjathu sathyamaaNu.

(what he told is true)

b) The object of a transitive verb: in this use of the noun clause it ends with a word എന്ന് / *ennu.*

മോഹൻ വരുമെന്ന് സീത പറഞ്ഞു

mOhan varumennu seethe paRanju.

(Seetha told that Mohan will come)

c) The object of a preposition

നീ പറയുന്നതിൽ ഒരു അര്ത്ഥവുമില്ല

nee paRayunnathil oru ardhavumilla

There is no meaning in what you say

d) In apposition to a noun or pronoun

നീ അവധിയിലായിരുന്നത് ദുഖകരമാണ്

nee avadhiyilayirunnuennathu dughakaramaanu.

(It is saddening that you were on leave.)

e) The complement of a verb of incomplete predication.

അവൻ വരുകില്ലെന്നതാണ് എന്റെ വിശ്വാസം

Avan varukillaennathanu ente viSwasam.

(It is my belief that he will not come)

4.2.4 Classification of Sentences

Sentences in a language are classified as

1. simple sentences
2. complex sentence
3. compound sentences

4.2.4.1 Simple sentence

A simple sentence (*choornika*) is one which has only one subject and one predicate. In another way, a simple sentence is one which has only one finite verb.

Table 4.1 Clauses and Clause markers in Malayalam

Type of Clause	Clause function	Clause marker/ suffix to verb
adjective	Adjective	Relative participle പറഞ്ഞ/ *paRanja*
	Adjective	Infinitive before noun
Noun	Subject of a verb	അത് /*athu*
	Object of a transitive verb	എന്ന് *ennu*
	Object of a preposition	അത് /*athu+preposition*
	In apposition to a noun or pronoun	എന്നത് /*ennathu*
	Complement of a verb of incomplete predication	എന്നതാണ് /*ennathaaNu*
Adverb	Time	അപ്പോൾ /*appOL*, കഴിഞ്ഞു /*kazhinju*, അതിനുശേഷം/ *athinuSEsham*, അതിനുമുന്പ് /*athinumunpu*, അതുതൊട്ട് /*athuthottu*, അതുമുതൽ/ *Athumuthal*
	Place	verb prefixed by എവിടെ/evite and suffixed by ഉം verb prefixed by എവിടെ/evite and suffixed by ഓ/ഠ
	purpose	അതിനുവേണ്ടി / *athinuvEnti*
	Reason/ consequence	അതുകൊണ്ട് *athukontu*, അതുകാരണം *athukaaraNam*
	condition	ആല് / *aal*
	Comparison of degree / manner	അതിനേക്കാൾ /*AthinekkaL*, അതുപോലെ /*athupOle*

1.സൂര്യൻ കിഴക്കുദിക്കുന്നു

Sooryan kizhakkuudikkunnu

(Sun rises in the east)

2.മോഹൻ മിടുക്കനാണ്

mOhan mitukkan aanz

(Mohan is intelligent)

4.2.4.2 Compound sentence

When two or more independent simple sentences are combined by a coordinating conjunction like ഉം/um we get a compound sentence (*mahaavakyam*). In a compound sentence each of the sentences is called a principal clause or main clause.

രാമൻ പാടുകയും മോഹൻ ആടുകയും ചെയ്തു

Raaman patukayum mohan aatukayum cheythu.

(Raman is singing and Mohan is dancing.)

4.2.4.3 Complex sentence

A complex sentence (*sangeernakam*) consists of a principal clause and one or more subordinate clauses.

രാജൻ അടിച്ചപ്പോൾ കുട്ടി കരഞ്ഞു

raajan atichchappOL kutti karannjnju.

(The child cried when Rajan hit.)

In the above sentence, കുട്ടി കരഞ്ഞു / *kutti karanjnju* is the principal clause, which can stand by itself and രാജൻ അടിച്ചപ്പോൾ / *raajan atichchappOL* cannot

stand by itself and make good sense. This clause is dependent on the clause കൂട്ടി കരഞ്ഞു / *kutti karanjnju*. Subordinate clauses are also called dependent clauses.

പഠിക്കാത്തതുകൊണ്ട് രാജു പരീക്ഷയിൽ തോറ്റു

paThikkaththathukondu raju pareekhayil thOttu

(Raju failed in the exam because he didn't study)

പഠിക്കാത്തതുകൊണ്ട് / *paThikkaththathukondu* is the subordinate clause and പരീക്ഷയിൽ തോറ്റു / *pareekshayil thOttu* is the main clause.

മോഹൻ വന്നപ്പോൾ സീതയെക്കണ്ടില്ലെന്ന് രാജു പറഞ്ഞു

mOhan vannapOL seethayekkandillennu raaju paRanjnju

(Raju told that when Mohan came he didn't see Seetha)

This sentence has two subordinate clauses and one main clause. The two subordinate clauses: a noun clause, മോഹൻ വന്നപ്പോൾ സീതയെക്കണ്ടില്ലെന്ന് / *mohan vannappOL seethayekandillennu* and an adverbal clause മോഹൻ വന്നപ്പോൾ / *Mohan vannappOL* which is part of the above subordinate clause. This shows the hierarchical dependencies of the clauses in a sentence. രാജു പറഞ്ഞു / *raaju paranjnju* is the main clause.

4.2.5 The Hierarchical structure

Clauses in a sentence can be nested one inside the other, resulting in a hierarchical or tree like structure. This aspect of structure is called the hierarchical structure [72,73,40,41]. Clauses in a sentence are not completely independent of one another but there are inter-clause dependencies. For example, a noun phrase being modified by a relative clause has two roles to play, one in the relative clause and the other in the outer clause.

According to Universal clause structure grammar (UCSG) all inter-clause dependencies systematically flow down the clause structure tree from the root towards the leaves [74]. Also, the constituents of a clause do not cross clause boundaries in scrambling. Violations of this principle can be viewed as exceptions to this general rule rather than as evidence for the invalidity of this principle. Verb groups and sentinels contain all the required information for recognizing clauses, for determining the nested or hierarchical structure of clauses and for determining the clause boundaries, although only partially. It is seen that every clause in a sentence except for the main clause has a sentinel which marks one of the boundaries of that clause. The sentinel marks either the beginning or the end of the clause depending upon the language in use. Also, by definition, every clause must have exactly one verb group. Thus verb groups and sentinels behave like brackets and impose very strong constraints – the brackets must match properly. Thus the total number of verb groups in a sentence must be exactly one more than the total number of sentinels.

Constraints on clause structure imposed by verb groups and sentinels are thus very strong yet very easy to apply. These constraints also help us in reducing lexical ambiguities to some extent, especially the more critical ambiguities such as noun/verb and sentinel/non-sentinel ambiguities. The hierarchical structure for sentences is used in language understanding and also in machine translation [75,76,77,78]. The job of syntactic analyzer is to accept a sentence in natural language and to produce a description of its internal structure in the light of the given grammar - a formal specification of all the valid structures in that language [79,80]. There are several aspects of structure that a syntactic theory is expected to deal with, including assignment of functional roles to the various constituents, analyzing the modifier-modified relationships, resolution of

anaphoric and other kinds of references, attachment of prepositional phrases and subordinate clauses, and analysis of emphasis, focus, topic etc.

4.2.6 Syntactic structure difference between languages

One of the crucial task in machine translation is the syntactic structural transfer, which is the conversion from a syntactic analysis structure of the source language to the structure of the target language. Structural information can be in the form of constituent transfer: for example, how is a noun phrase or a sentence constructed in a language, and how does the ordering of words and groups of words change when translated into another language, etc. [81,82]. Languages like Malayalam and English belong to two language families and their sentence structure differs a lot. So a simple morpheme to morpheme mapping will not give the correct translation of the input sentence. The parser has to perform the syntactic structure transfer also to get the target sentence in the correct structure.

Malayalam sentence: സീത രാമന്റെ കുട മോഹൻ കൊടുത്തെന്ന് രാധ പറഞ്ഞു

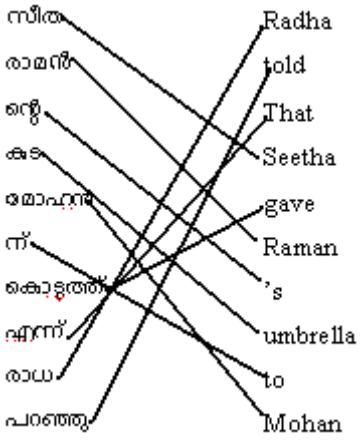
seetha raamante kuta mOhanu kotuththennu raadha paRanju

Sequence of morphemes: സീത / *seetha*, രാമൻ / *raaman*, ന്റെ / inte, കുട/ *kuta*, മോഹൻ / *mOhan*, ന് / inu, കൊടുത്ത് / *kotuththu*, എന്ന് / *ennu*, രാധ/ *raadha*, പറഞ്ഞു / *paRanjnju*

English translation: Radha told that Seetha gave Raman's umbrella to

Mohan

For the example given above the reordering required at the morpheme level is shown below.



From the above example it is clear that a lot of reordering of words is required to get the target sentence in the correct form for applications like machine translation. It is also found that the reordering occurs at the word group level. The reordering required is specific to the language pair under consideration.

4.3 Parsing

The process of generating the sentence through derivation using a set of grammar rules is called parsing and the generated hierarchical structure is called the parse tree of the sentence. Although a lot of work has gone into developing full syntactic parsers, high performance wide coverage syntactic parsing remained a difficult challenge. In recent times there has been an interest in shallow parsing or partial parsing. Shallow parsing is restricted to finding phrases in sentences. These phrases are called chunks. With partial parsing it is

possible to disambiguate words based on part of speech tags like verb/noun/suffix conflict. But for handling sentences like I turned off highway I-90 and I turned off my radio we need more world knowledge to distinguish between the two meanings of the phrase 'turned off'. More knowledge regarding the type of the object, like whether it is the name of a road or it is a device, is needed to find the correct meaning of 'turned off'. This kind of disambiguation can be done only by a global analysis, including syntactic analysis, discourse analysis and even world knowledge. It is possible that many different phrase structure trees derive the same sequence of words. If a sentence has multiple parses then the grammar is called ambiguous grammar.

4.3.1 Computational models for parsing

Finding the right syntactic structure is a search process. The search finds all possible parses for a sentence. Parsing with sub grammars reduces the problem of rapid growth in parsing table sizes as number of rules increase [83,84,85]. The following constraints guides the search process:

1. The first constraint comes from the words in the input sentence. A valid parse is one that covers all the words in a sentence. So these words will constitute the leaves of the final parse tree.

2. The second constraint comes from the grammar. The root of the final parse tree must be the start symbol of the grammar.

Using the two constraints two widely used search strategies are used:

a) Top down search or (goal directed search)

b) Bottom up search or (data directed search)

4.3.1.1 Top down parsing

As the name suggests, top down parsing starts its search from the root node S and works downwards towards the leaves. The parser needs the set of grammar rules represented in the context free form. The parser has to find all subtrees which can start with S. To generate the subtrees of the second level search, the root node is expanded using all the grammar rules with S on the left hand side. Each non terminal in the resultant sub trees is expanded next using the grammar rules having a matching nonterminal symbol on the left hand side. The right hand side of the grammar rules provides the nodes to be generated which are then expanded recursively. As the expansion continues the tree grows downward and eventually reaches a state where the bottom of the tree consist only of terminal symbols. The subtrees whose leaves do not match words in the input sentence are rejected and other options for its parent are tried.

4.3.1.2 Bottom-up parsing

A bottom up parser starts with the words in the input sentence and attempts to construct a parse tree in an upward direction towards the root. At each step the parser looks for rules in the grammar where the right hand side matches some of the portions in the parse tree constructed so far and reduces it using the left hand side of the production. The parse is considered successful if the parser reduces the tree to the start symbol of the grammar.

Both the methods have advantages and disadvantages. The top down search generates trees with the start symbol of the grammar. So it never wastes time exploring a tree leading to a different root. But it wastes time exploring trees that produce words that are inconsistent with the input. A bottom up parser

never explores a tree that does not match the input. It wastes time generating trees that have no chance of leading to the start state.

The top down parsing needs backtracking as a word can have more than one tag and the tree can be traversed in more than one way. So when the first choice is found to be false at a later stage it has to go back and try the other tag for the same word. The second problem with such algorithms is that the trees built may be discarded during backtracking. At a later time these may have to be rebuilt during subsequent steps in the parse.

4.3.2 Previous works

Both the above problems of top down depth first approach are solved by dynamic programming algorithms. The CYK algorithm, Graham Harrison Ruzzo (GHR) algorithm and the Earley algorithm are the three parsers which use dynamic programming. Earley parser is a top down search using dynamic programming. The CYK parser is a parser based on bottom up approach using dynamic programming.

Rajiv Sangal *et. al*. has proposed a karaka based approach to parsing Indian languages [86]. Their parser does karaka role assignment for verbs and word sense disambiguation for verbs and nouns in Hindi. For karaka role assignment they use integer programming techniques. For word sense disambiguation they use a merged lakshan chart for each verb which contains the different conditions to be tested to assign a particular meaning to a word.

The verb “jotha” has different meanings in different contexts. The disambiguation is done using a merged karaka chart.

to find the meaning of jotha as harnessed, hitched or ploughed, context dependent rules like the following are used. "if category of the karta is human and category of karma is animal then meaning is hitched" .

"if the karma is the word khet or jamin then the meaning is ploughed".

Partial parsing via finite state cascades has been described by Abney [87,88]. A finite state cascade consists of a sequence of levels. Phrases at one level are built on phrases at the previous level and there is no recursion. Phrases never contain same level or higher level phrases. Deterministic parsers specified by finite state cascades are fast and reliable. They can be extended to construct parse tree with finite feature structures.

Miles Osborne has proposed that shallow parsing can be used for part of speech tagging [89]. Zhou *et. al*. has proposed an HMM based chunk tagger with context dependent lexicon. Rob Koeling applied maximum entropy models for chunking [90].

Purely linguistic as well as purely machine learning approaches have proved to be impractical. Hand crafting rules in the linguistic approach can be very laborious and time consuming. Parsers tend to produce a large number of possible parses and in the absence of suitable ranking and rating mechanism selecting the right parse can be difficult. In addition to these, morpheme based parsing increases number of possible parses [91].

Murthy has proposed a method which combines a finite state chunker and a statistical chunker using HMM [17,92]. Here, instead of looking for a grammar that can capture all and only valid structures, a finite state chunker which captures all valid word groups without necessarily restricting to only those

word groups which are appropriate in the context of a given sentence. A separate statistical component is used to rank the word groups so produced. The system only rates the word groups produced. The system rates all the parses produced. The system is more than a shallow parser as it can do disambiguation in context to some extent.

Win Win Thant *et. al*. has proposed a Context Free Grammar Based Top-Down Parsing of Myanmar Sentences [93]. In this they describe a method to parse simple and complex sentences in Myanmar. The accuracy reported is 90% for simple sentence and 91% for complex sentences.

Mark A Jones *et. al.* has reported on the application of a probabilistic parser applied to software test documents [94,95]. They train a statistical parser from a bracketed corpus and its use in a software testing application that translates English specifications into automated testing language. No grammar rules are explicitly specified. The rules and contextual probabilities are generated from the corpus. The parser is successful in identifying the correct parse and deterministic in the number of parses it produces. With 211 sentences for training and 147 for testing, parses were found for 77% of test sentences. Of these the top ranked parse was correct 90% of the time and 99.3% of bracketing decisions were correct.

Brian Roark describes the functioning of a broad-coverage probabilistic top-down parser, and its application to the problem of language modeling for speech recognition [96].

Richard A. Frost describes a new top-down parsing algorithm to accommodate ambiguity and left recursion in polynomial time [97]. He discusses on how exponential complexity can be avoided by memorization using a table, and that

left-recursive productions can be accommodated through a technique like the length of input to stop recursive descent or calling nonterminals with how many terminals should be selected etc.

Gabor Proszeky has discussed a morphological analyzer as a syntactic parser [49]. The system was called Humor ESK (High speed unification morphology enhanced with syntactic knowledge). It consists of numerous meta lexicons. Each of them has the name of the syntactic category it describes. The categories like S, NP, VP etc. are stored in separate lexicons. Meta lexicons form a hierarchy. Parsing on a level can be realized as a lexicon lookup. No backtracking, look ahead, or the time consuming parser steps are needed for the analysis of a sentence.

Stuart M. Schieber describes Sentence disambiguation by a shift reduce parsing technique and a uniform architecture for parsing and language generation [98,99].

4.4 Conclusion

This chapter discussed the parsing methods, the syntactic structure of sentences in a language and also the syntactic differences between sentences of two languages. The different classes of sentences and the hierarchical structure in sentences were explained in this chapter. The need for part of speech tagging and the criteria for tagset selection were also discussed in the chapter. The next chapter discusses the design and development of the prototype translator.

Chapter 5

DESIGN AND DEVELOPMENT OF THE PROTOTYPE

5.1 Introduction

This chapter discusses the design and development of the prototype machine translation system. The computational model and the data structures selected are clearly described. The major criteria for the selection of these models and data structures were accuracy of translation, speed, space required and ease of future enhancement.

5.2 Selection of Translation model

The major difficulties observed in the development of a translation system for Malayalam to English translation were i) the highly agglutinative nature of Malayalam ii) the wide syntactic structure difference between Malayalam and English. To deal with the huge dictionary size required by agglutinative languages a morpheme based translator was chosen. The dictionary used consists of morphemes and their translations. The survey on the various approaches for machine translation revealed that a direct translation approach can be applicable to only closely related languages. Corpus based approaches also will be inefficient in the case of Malayalam to English translation system because of the wide syntactic difference between the language pairs. Moreover, corpus based approaches require sentence aligned parallel text for each language pair. The transfer based approaches works well for language pairs with wide syntactic difference. Transfer based systems are more reliable, flexible and adaptable in meeting the needs of different levels and depths of syntactic and semantic analysis. So it has been decided to use a syntax transfer

based approach using synchronous tree adjoining grammar(STAG) as the translation approach. The system to be developed is only a prototype and the main objective has been set to check the accuracy of the syntax rules and the transfer rules derived and the accuracy of the final translation.

5.3 Prototype Model

Translation with Synchronous TAGs is a three step process:

1. Derivation tree for source sentence is obtained by parsing the source sentence.
2. Source derivation tree is converted into target derivation tree or trees.
3. Once the target derivation is obtained from step 2, target parse tree is generated by traversing target elementary trees and listing the leaf nodes in the parse tree in order will give the required target string. [100,101,102,103].

The system modules selected are similar to that of SAMPARK system developed by IIIT Hyderabad [14,104]. The syntax structure formalism used is similar to the one used in UCSG based Kannada translator by Murthy. The parsing method is similar to the one in MANTRA system developed by CDAC Mumbai [105]. We have tried to combine modern artificial intelligence techniques with the classical Paninian framework based on Sanskrit grammar [106,107,108,109]. The computational models chosen for the translation, morphological analysis and parsing is a pioneering effort for Malayalam language un attempted by previous researchers. The models chosen are first of their kind in Malayalam language.

5.4 System modules

The major modules in the system are shown in Figure. 5.1. There are four main modules:

a)Morphological analyser

b)Morpheme based parser

c)Target sentence generator

d)Bilingual dictionary

5.4.1 The morphological analyzer

The analysis module, finds the sequences of morphemes in the input sentence. Since there are more than one way of splitting a word into morphemes it performs a depth first approach to find all possible sequence of morphemes for the given sentence. A Detailed description on the design and development of the module is given in chapter 6. The output of the analyzer module is fed to the parser to create source and target parse trees.

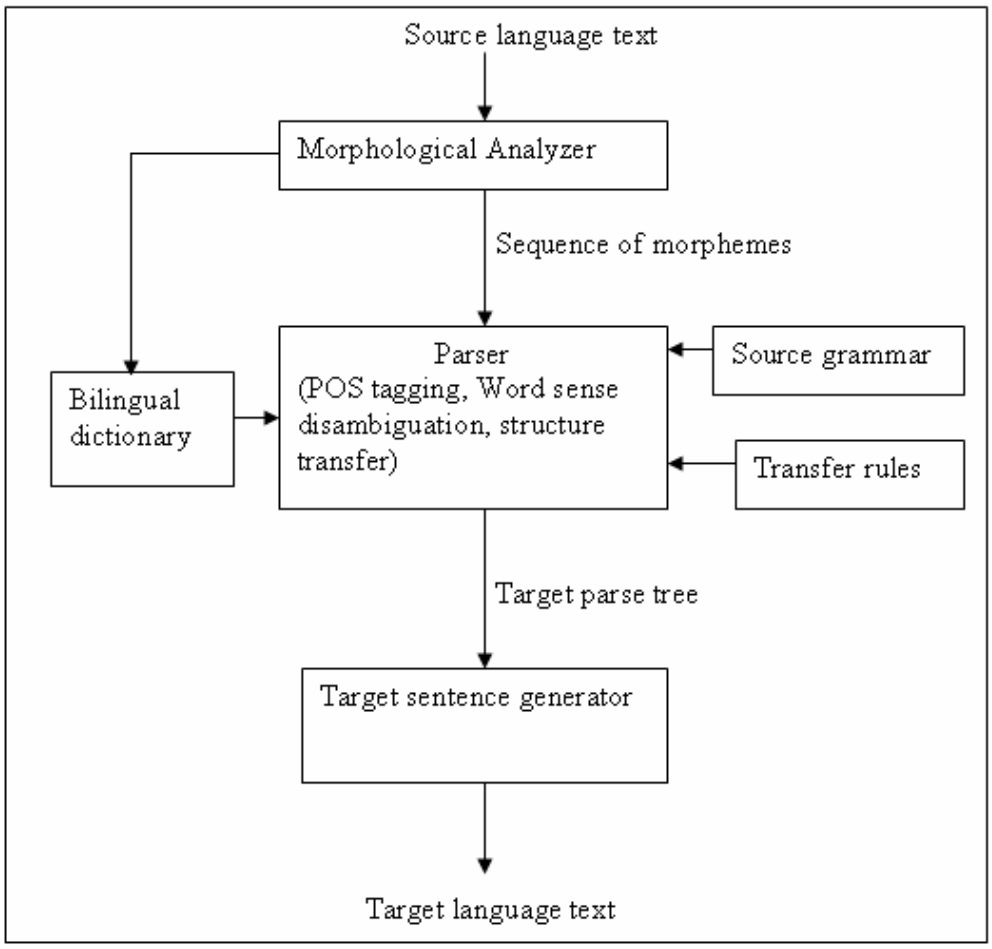


Figure 5.1 Block diagram of the prototype translator

5.4.2 The parser

The parser performs three functions. i) It finds the correct POS tag for each morpheme for word sense disambiguation. ii) It identifies the chunks in the sentence for reordering and creates the source parse tree. iii) It performs the reordering required in order to meet syntactic requirements of the target language and create the target parse tree.

A detailed description of the design and development of the parser module is given in chapter 7. The output of the parser is a set of valid parse trees for a sentence of the source language and their corresponding parse trees in the target language. The target language trees are fed to the target sentence generator module for the final sentence generation in the target language.

5.4.3 Target sentence generator

The function of the target generation module is to generate the target sentence from the target parse tree. It includes the morphological generation module for the target language. This module produces the final English sentence by performing a depth first traversal of the target parse tree created by the parser. It also performs surface form generation of English words using the morpheme sequences in the tree. It also uses additional set of rules for the generation of target sentences in the correct form. It performs two kinds of post processing on the target sentences.

1. It removes the dative case marker 'to' from the first noun group (i.e. subject).

Input: രമക്ക് കടയിൽ പോകേണ്ടിവന്നു (*ramakk katayil pOkEnti vannu*)

Rama had to go to the shop

The depth first traversal of the target parse tree creates the following sentence in English.

to Rama had to go to shop.

The initial 'to' has to be removed to form the correct sentence as:

Rama had to go to shop

2. *It removes the last 'and' in a conjunct noun*.

Input: രമയും സീതയും കൂട്ടുകാരാണ് (*ramayum seethayum koottukaaraaN*)

Rama and Seetha are friends

The depth first traversal of the target parse tree creates the following English sentence

Rama and Seetha and are friends

The last and in the noun group is eliminated forming the correct sentence as

Rama and Seetha are friends

5.4.4 Bilingual dictionary

The bilingual dictionary is the most essential part of the whole machine translation system. The first rule based translation system Systran uses for every source language two types of related dictionary: stem dictionary and expression dictionary. The stem dictionary contains the basic form of single words and expression dictionary contains idiom replace, collocation, conditional expression, etc. For the present system only one dictionary had been kept as it was designed for a prototype machine translator. The major concerns were to develop and test the functioning of the analyzer and the parser module which were unattempted so far for Malayalam language. The structure of the bilingual dictionary of our prototype system is shown in Table 5.1. Excerpts from the bilingual dictionary are given in Appendix 1. The bilingual dictionary contains the following information:

i)The dictionary includes most of the commonly occurring verbs, nouns, pronouns, adjectives, inflectional and derivational suffixes, clause suffixes etc.

ii) Each entry in the file has four fields: the root word (morpheme), the morpheme tag, category (human/nonhuman) and its translation. A root word can have more than one tag.

iii)The verbs in past tense have their root words stored along with them.

Table 5.1 Bilingual dictionary

Source word	Morpheme tag	Target word	Root word	Category
പൂച്ച	Noun (N)	cat		Non human
ഉടെ	Case suffix (NA)	‘s		
പോയി	Verb (V)	went	go	

It has been found that a trie model for dictionaries helps in fast retrieval of data. A prototype dictionary had been implemented using trie model. The trie had been implemented as a two dimensional array. Though the system gave 95% saving in retrieval time the space requirement was 93% higher than that of a linear storage of data in a text file. Since the main memory space is scarce more advanced data structures have to be used to implement the trie model. Taking these factors into consideration the dictionary was implemented as a sequential file containing the morphemes, their lexical categories and their translations.

5.4 Implementation

The modules were separately developed in Python language and tested separately for accuracy. After sufficient amount of testing all the modules were combined to build the prototype. The system can work in Linux or Windows operating system. The input sentences were stored in a text file. The set of grammar rules in the context free notation and bilingual dictionary are also stored as text files. The structure transfer rules were implemented in the source code of the parser.

5.5 Conclusion

This chapter discussed the design and development of the prototype machine translation system. The major modules used in the system were explained in the chapter. The post processing steps done by the target sentence generator module and the bilingual dictionary structure were also given. The next chapter gives the computational model chosen for the morphological analyzer, the working of the algorithm for the analyzer and the data structure used for the splitting rule table.

Chapter 6

MORPHOLOGICAL ANALYSER FOR MALAYALAM

6.1 Introduction

This chapter discusses the design and development of the morphological analyser module of the prototype machine translation system. The computational model and the data structures selected are clearly described. Algorithm used and the sample outputs generated by the module are also given.

6.2 Design of the morphological analyzer

The highly agglutinative nature of Malayalam language and the ambiguity in the word compounding rules were the major difficulties we faced while designing the morphological analyser. First of all, a suitable model for the morphological analyzer was needed. Also, an optimal set of word compounding rules was needed to reduce the ambiguity in forming multiple sequences of morphemes for the same compound word.

6.2.1 Design of the algorithm

The first step in the design of the morphological analyzer was to identify the different classes of morphophonemic changes that occur when morphemes are joined. The word compounding rules were carefully studied and it was observed that the changes occur at the boundary. The sound change depends on the ending sound of the first morpheme, the beginning sound of the second morpheme, on the morpheme categories of the two morphemes and in some cases it depends on the first word. It was also observed that a two level finite

automata is not enough for a highly agglutinative language like Malayalam. So a recursive depth first search with backtracking algorithm has been utilized for the morphological analyzer [37,110]. The algorithm uses backtracking to find all the sequences of morphemes in the sentence since there are multiple ways to split a given compound word. Unlike the morphological analysers described in chapter 3 the morphological analyzer of our system finds only the morpheme sequences in the given sentence as a morpheme can have multiple POS tags. The correct lexical category for each morpheme is found only during the parsing stage.

Input to the morphological analyser is a valid sentence of the source language and output is a sequence of morphemes for the sentence. The algorithm uses Table 6.1 which contains the replacement rules at each morpheme boundary and the bilingual dictionary which contains the words belonging to various lexical categories which is shared by other modules of our prototype translation system. It also uses an array to store partial results during the search process. When the input is scanned many times for getting all morpheme sequences, it is possible that the same string has to be split many times. So computation time can be reduced by storing partial results. This form of learning method by which we store the intermediate results to reduce search time is called rote learning [110].

6.2.2 Design of the compounding rule table

The format of the compounding rule table is shown in Table 6.1. The first entry shows the current syllable. The entry in the second column is the additional rule which must be satisfied by the first word or by the next syllable. The entry in the third column gives the character that must be added to the end of the first word.

The entry in the forth column gives the character to be added or deleted from the beginning of the second part of the string during splitting.

As an example, in rule no.3 the syllable is replaced by the characters given in column 3 and 4 only if the condition given in column 2 is true. i.e. Only if the first morpheme is a verb or the word ഒരു / a the search continues with the current replacements. If the first part is not a verb or the word ഒരു / a the current replacement will be cancelled and another applicable rule will be selected and search continues from there. The inclusion of additional conditions eliminates the possibility of wrong splits.

6.3 Algorithm for the analyzer

The Algorithm is detailed as follows:

Split (input: string)

Step1: If input is a valid word

Output=input

Return output.

Step2: While not end of input do steps 3-8

Step3: While there are unselected rules at the current syllable boundary do steps 3-7

Step 4: Select an unselected rule from the rule table which satisfies the preconditions at the syllable boundary.

Step5: Apply the rule and split the input into two parts first and second

Step6: If the first is a valid word and If second word is present in the partial array attach the split result with first to form output goto step 8 else goto step 7

Step7: Temp=split (second) // recursive call to the same function with the second part //

If there are valid splits in temp form partial results in output array by combining first and each entry of the temp array.

Step 8: Advance to next syllable.

Step 9: return output.

Step10: Stop.

6.4 Working and analysis of the algorithm

This algorithm works in a self recursive or depth first manner. First it checks whether the string can be accepted as such without splitting. In that case it is returned as such. Otherwise it tries to find all the splits for the string within the while loop in step2. The while loops in step 3 tries to apply multiple splits at the current syllable position. The string is split into two parts using the standard sandhi splitting rules of the language. If the first part is a valid word then the algorithm proceeds recursively with the second part as the input string, otherwise it moves to the next syllable. When the procedure ends the output array will contain all the possible splits.

Table 6.1. Rules for Morpheme Splitting

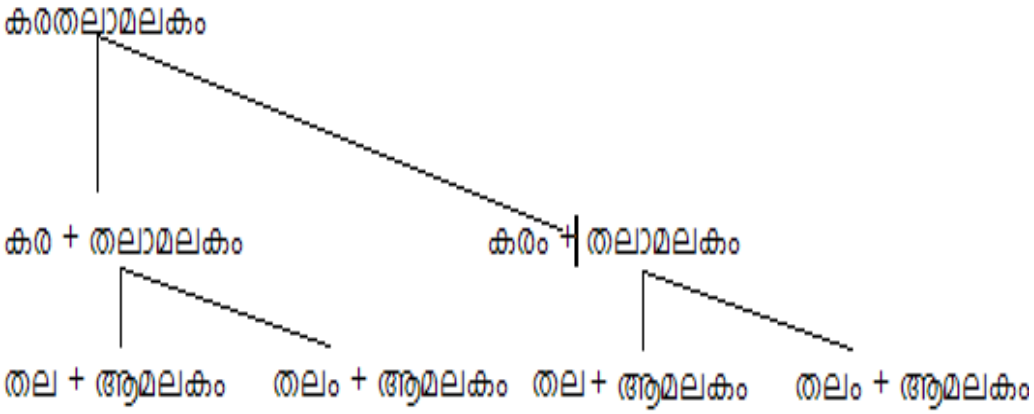
Sl no	Condition 1 On the current syllable	Condition2	String to be added to end of first word	beginning of the second word	Example
1	All (C+SS)		C + ്	Vowel(SS)	കാടായി= കാട്+ ആയി
2	ല്ല+ SS	W1=അല്ല/ഇല്ല	ല്ല	Vowel (SS)	അല്ലെന്ന്= അല്ല+ എന്ന്
3	All	WC1=Verb / W1=ഒരു	C+(ു/ൂം)	Vowel(SS)	ഒരാന= ഒരു+ ആന
4	യ+ SS	W1= ആയി/ പോയി	യ+ി	Vowel(SS)	ആയില്ല= ആയി+ ഇല്ല
5	C+ (ു/ൂ)	C2= വ	S1	Vowel(SS2)	മധുവിനെ= മധു+ ഇനെ
6	C+vowel sounds other than(ു/ൂ)	C2=യ	S1	Vowel(SS2)	സീതയുടെ= സീത+ ഉടെ
7	അ,ഇ,എ	C2=വ	S1	Vowel(SS2)	അവർ= അ+ അർ
8	C+ ാ		C	-	കലാവാസന= കല+ വാസന
9	C+SS	C2= മ	C + ം	Vowel(SS2)	പണമില്ല= പണം+ ഇല്ല
10	C+SS	C2=ത്ത	C + ം	Vowel(SS2)	മരത്തിൽ= മരം+ ഇൽ
11	C	Gem(TC2)		Sin(C2)	പടിപ്പുര= പടി+ പുര
12	C	Gem(C2)	C + ം	Sin(C2)	പണപ്പെട്ടി= പണം+ പെട്ടി
13	ങ,ഞ,ണ്ട,ന്ത,മ്പ,ങ്ങ		ം	ക,ച,ട,ത,പ,ക	വരുങ്കാലം= വരും+കാലം
14	C1/C1+SS	C2	C1/c1+SS	C2	കലമാൻ = കല+ മാ ന്
15	C+ ാ		C/C+ ം	C2=അ/ആ	
16	ഗ,ദ,ധ,ന,മയ,ര,ല,വ,ഹ+ ോ		C + സ്	S2	മനോദുഖം= മനസ്+ദുഖം
17	ന,ണ,ള,ല, ര+SS	All	ൻൺൾ ൽർ	Vowel(ss)	നീരസം= നി:+ രസം
18	ദു/ നി:	W1=ദു/നി: C2=ര		Vowel(ss2)	ദുര□ഥം=ദു: +അ□ഥം
19	C+ോ		C+സ്	S2	മനോദുഖം= മനസ്+ദുഖം
20	അ,ഇ,എ	Gem(C2)	S1	Sin(C2)	അക്കാലം= അ+ കാലം

SS (vowel symbol) – ാ, ി, ീ, ു, ൂ, ൃ, െ,േ, ൈ, ോ, ൗ, ം

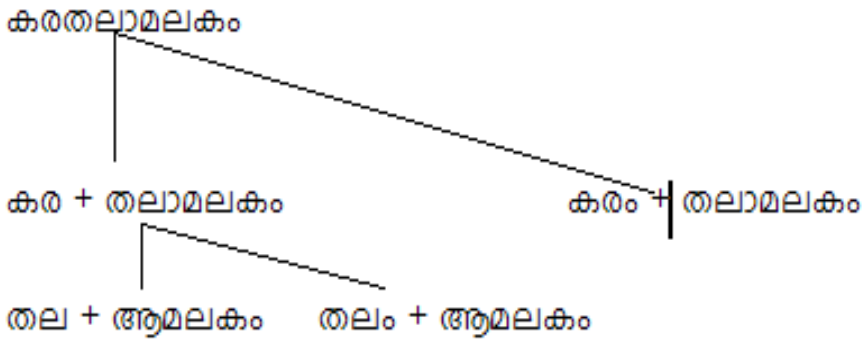
C- Consonant C1- Ending consonant of the first word
C2- Starting consonant of the second word.
Sin(C) – singular form of the geminated character. e.g.: sin (ക്ക) – ക

S1- the ending syllable of the first word
S2-the starting syllable of the second word
TC-Tense characters (ക, ച, ട, ത, പ)

The number of recursive calls is reduced by storing intermediate results. The split tree for the compound word കരതലാമലകം is shown in Figure. 6.1. The states in the tree with no intermediate storage is shown in (i) and the states in the tree with intermediate storage is shown in (ii). It can be seen that there are 7 states in the search tree without storing intermediate results and only 5 states with partial result storing. This definitely reduces the computing time.



(i)



(ii)

Figure 6.1 Depth first tree for the morpheme splitting

The analyser is basically a depth first search with backtracking. The time complexity of a depth first search is O(e) where e is the number of nodes in the search space tree. A node in the search space corresponds to a state resulting from the application of a valid rule. The rule can be splitting rule in the case of the analyser or a production rule in the case of a parser.

The best case time complexity of the analyser algorithm is O(1) when the input compound word is available as a whole in the dictionary and no splitting trials needed to be done on the input. The worst case time complexity for the algorithm is $O(r^n)$ where n is the number of syllables in the input and r is the number of splitting rules. The worst case occurs when at each syllable boundary all the splitting rules are applicable. The computation time has been drastically reduced because of the selection of intelligent splitting rules and by storing intermediate results. It has been found that an average of 16 states have been

generated for the words tested. Analysis of compound words belonging to different domains showed that the maximum number of syllables in a compound word is not exceeding 12. All these minimise the total number of splitting trials. The speed of the analyser critically depends on the number of words in the dictionary. Better storage techniques are required for this algorithm to work for a large dictionary. Finite automata models have proved to be efficient for storing words for the analyser. As each applicable rule corresponds to a dictionary search and since dictionary search time seriously affects the total time of execution of the analyser the total number of applicable splitting rules for a specific input forms a measure of the computation time of the module. The total number of applicable splitting rules for a set of inputs with and without partial result storage are given in Table 6.2. Column N1 represents the number of rule trials without intermediate storage and column N2 represents the number of rule trials with intermediate storage. In our experiment with 100 words it was found that a saving of 23% was achieved in computation time with intermediate result storage.

6.5 Sample outputs

Sample outputs from the morphological analyzer are shown in Table 6.2. The table lists the various categories of splits obtained using the algorithm. For some words (e.g.:-2) only one correct sequence was obtained. The screenshot for such a case is shown in Figure. 6.3. In some cases (e.g.:-9) multiple sequence some of which are not correct were observed. For cases 1 and 3 multiple sequences all of which are correct were observed. The screenshot for this kind of output is shown in Figure. 6.4.

Table 6.2. Sample outputs from the analyzer

Sl. No.	Compound word/splits	Rules applied	N1	N2
1	കരതലാമലകം		18	12
	കര(land)+ തല(head)+ ആമലകം(aamla)	14,15		
	കര(land)+തലം(head)+ആമലകം(amla)	14,15		
	കരം(hand)+തലം(plane)+ആമലകം(alma)	14,15		
	കരം(hand)+തല(head)+ആമലകം(amla)	14,15		
2	പകരമായി		10	10
	പകരം + ആയി	9		
3	അവനാരെന്ന്		13	13
	അവ(those) +നാര്(fibre)+ എന്ന്(that)	17,14,1		
	അവൻ(he)+ആര്(who) + എന്ന്(that)	14,14		
4	നല്ലതാളി		8	8
	നല്ല(good)+അത്(that)+ആളി(friend)	2,1		
	നല്ല(good)+താളി(shampoo)	14		
5	ആനയുമാടുഞ്ചാടി		14	14
	ആന(elephant)+ഉം(conjunct)+ആട്(goat)+ ഉം(conjunct)+ചാടി(jump)	6,9,6,14		
6	പടക്കത്തിന്		18	13
	പട(army) +കത്ത്(letter)+ഇന്(casesuffix)	11,1		
	പടം(picture)+കത്ത്(letter)+ ഇന്(case suf)	12,1		
	പടക്കം(crackers)+ ഇന്(case suffix)	10		

Sl. No.	Compound word/splits	Rules applied	N1	N2
7	പടച്ചട്ട		6	6
	പട (army) + ചട്ട (cover)	11		
	പടം (picture) + ചട്ട (cover)	12		
8	കലമാൻ		6	6
	കലം(pot)+ആൻ(verb suffix)	9		
	കല(mark)+മാൻ (deer)	14		
9	പറയാനായി		14	8
	പറ(measure)+ആന(elephant)+ആയി	6,15		
	പറ(measure)+ആൻ(verb suffix)+ആയി	6,17		
	പറ(talk)+ആന(elephant)+ആയി	1,15		
	പറ(talk)+ആൻ(verb suffix)+ആയി	1,17		
10	കരിമ്പനക്ക്		14	14
	കരിം(black) +പന(palm)+ക്ക്(suffix)	13,14		
	കരിമ്പ്(sugarcane)+ അനക്ക്(move)	1		

6.6 Merits of the model

The following merits have been observed for the developed algorithm.

1. It uses the minimum number of compounding rules which reduces the computing time.

3. Since intermediate split results are stored, the system takes less time for finding all splits. Here before a recursive call a check is made whether a split was generated for the substring. In that case the split is taken directly from the stored array. In this way the system implements rote learning which is used to cut down the search paths .

4. It takes into account of almost all morphophonemic changes in Malayalam and Sanskrit.

5. The use of morphemes instead of surface forms for the words reduces the dictionary size.

6. It generates all possible spits for a compound word.

7. It uses best set of splitting rules to minimize the number of splits.

6.7 Module Implementation

Two versions of analyzer module were developed. One was implemented in python language to be integrated with parser to form the translator. The other version was developed as a tool to study the word compounding in Malayalam language. The tool was developed as a web based application which needs a web server like Apache. We used HTML, Javascript and PHP for the development of the tool. The tool can run in any web browser. The tool can run in Windows or Linux and can run in any system with Mac or Intel architecture. Screen shot of the opening screen from the tool is given in Figure. 6.2. The input to the system can be given in three ways:

i) Using virtual keyboard - The keyboard layout is shown the screenshot. The keyboard contains separate keys for almost all single alphabets in Malayalam. Other alphabets are generated using combinations of keys.

ii) Using transliteration - Text can be input in the text box shown in the top left corner. In this each Malayalam alphabet is mapped to a combination of English alphabets. The API for transliteration was downloaded from google website. The transliteration scheme used is shown in Appendix 2. It is found that inputting using transliteration method is easier than inputting using virtual keyboard.

iii) Using a text file - The input can also be given from a text file. The file can be selected using the browse button shown in the top right corner of Figure 6.2. The continue button is used to start the analyser. The arrows in Figure 6.4 help to move through the words in the selected text file either in the forward direction or backward direction. The contents of the file will be shown in the text box provided in the top.

The output is shown the text window shown in the bottom. All the possible combinations of morphemes for a compound word are generated by the system. The reset button is used to clear the screen for the next input which can be done in any one of the three ways explained earlier.

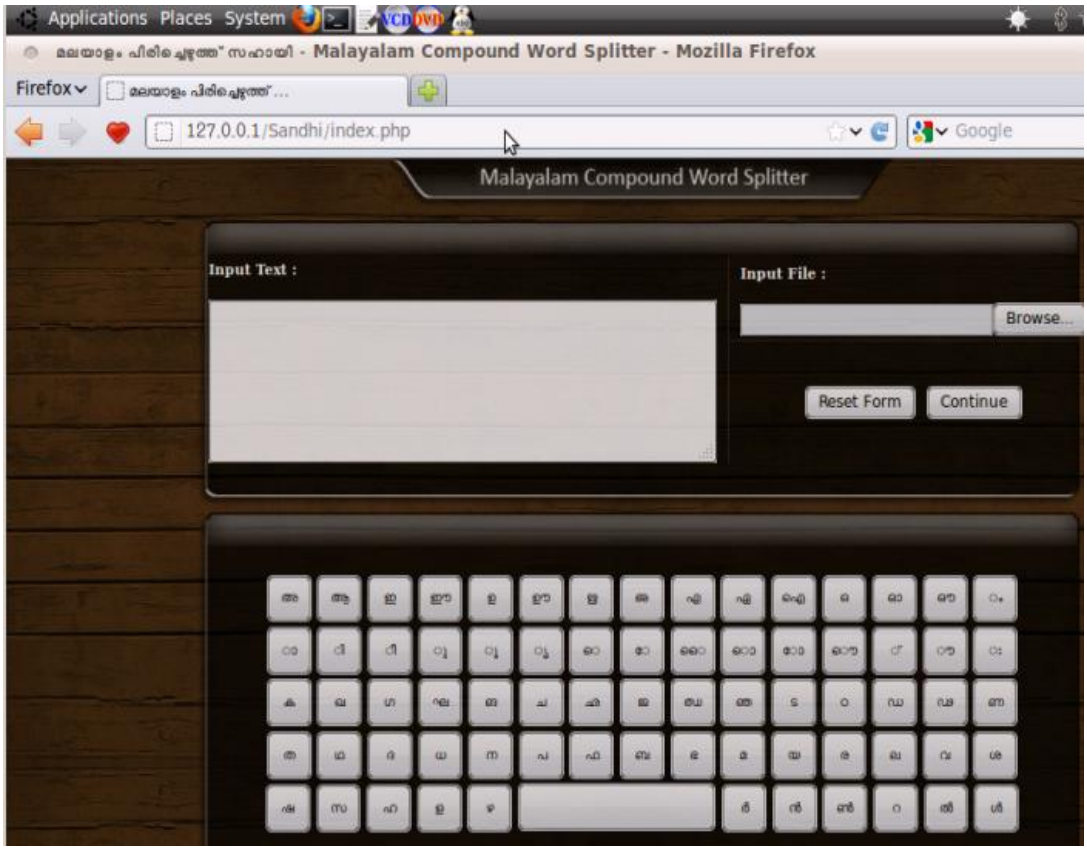


Figure 6.2 Screenshot1 from analyser

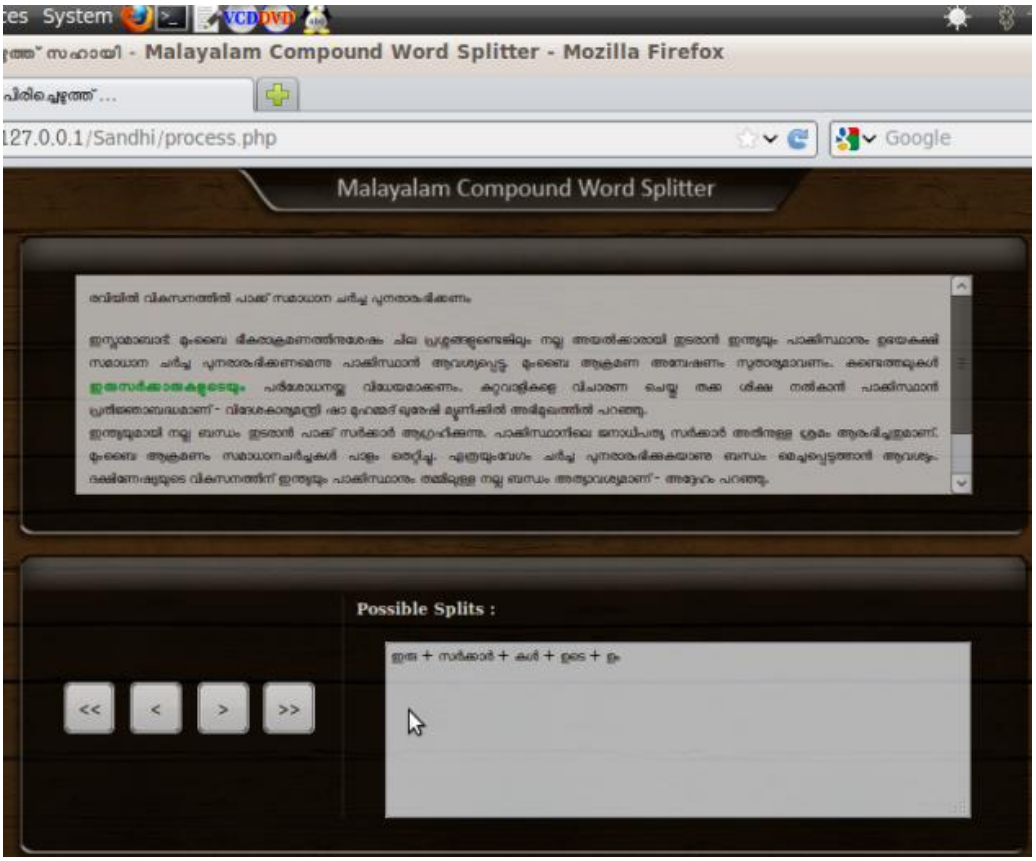


Figure 6.3 Screenshot2 from analyser

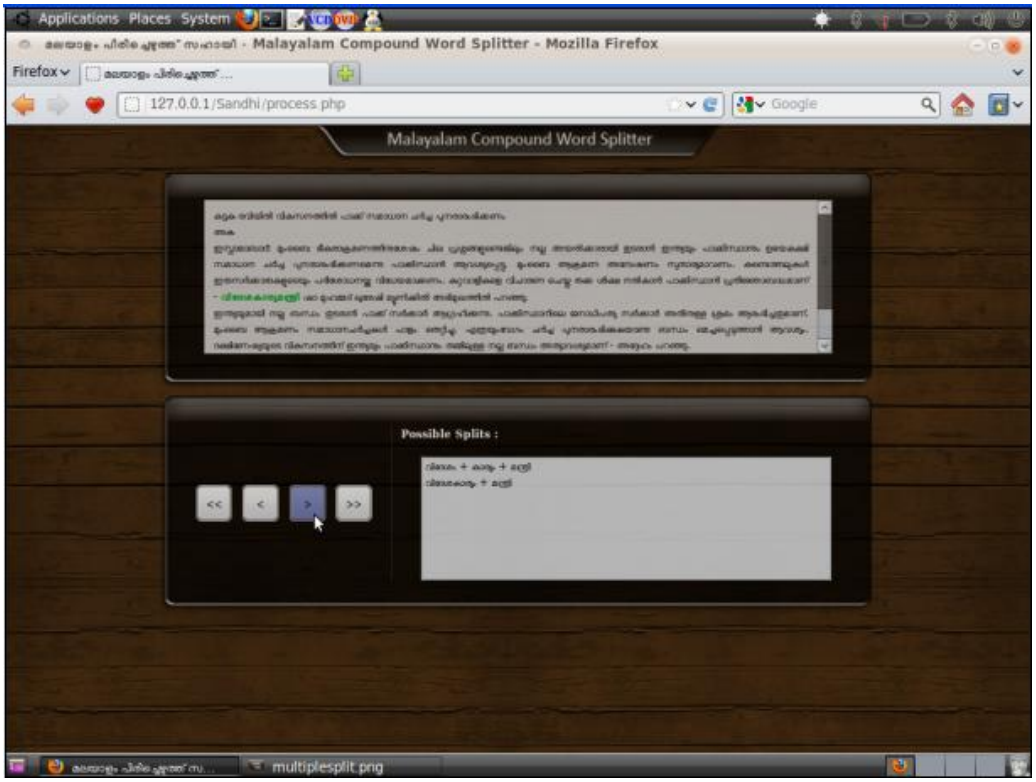


Figure 6.4 Screenshot3 from analyser

6.9 Conclusion

This chapter discussed the design and development of the morphological analyser module of the prototype translation system. The analyzer tries to find all the sequences of morphemes for the input words. The design of the algorithm and the rule table were clearly described in the chapter. The algorithm uses a depth first search approach with a rule based learning method to speed up the search process. Sample outputs and screen shots were also given in the chapter. The next chapter describes the design and development of the parser module.

Chapter 7

PARSER FOR MALAYALAM

7.1 Introduction

The chapter discusses the design and development of the parser module of the prototype translation system. It clearly describes the working of the parser algorithm with sample inputs. The chapter also discusses the set of part of speech tags, chunk tags and the chunk dependencies which were derived for the parser. The selected syntactic structure transfer rules required in the translation of Malayalam sentences to the corresponding English sentence are also discussed in this chapter.

7.2 Selection of the parser model

Of the different models available for parsing natural language sentences the bottom up parsers generate subtrees that have no hope of leading to the start state. But a top down approach never wastes time exploring trees that cannot result from the start state. The algorithms based on dynamic programming approaches like Earley parsing and CYK parsing algorithms overcome all problems of standard top down and bottom up parsing. Also there are parsers based on finite state cascades which improves the speed of the search process[37]. A top down approach has been chosen as the base model for the translation system considering many aspects. Frost, Hafiz and Callaghan [115] have proved that the time complexity of a top down parser to accommodate any form of CFG to produce all parses is polynomial time $\Theta(n^4)$ for left-recursive grammars and $\Theta(n^3)$ for non left-recursive grammars where n is the

number of input symbols. The top down parsers may fall into infinite loop for left recursive grammars. So the grammar rules are to be non left recursive. The derived grammar rules for Malayalam for the parser were non left recursive. The prototype system uses a small dictionary which further reduces the problems of multiple traversals in the case of standard top down approach.

7.3 Formation of rules used by the parser

It was decided to include the word sense disambiguation and the syntactic structure from source parse tree to target parse tree into the parser module. For this the parser uses two sets of rules: i) the syntax rules of source language for source parse tree creation ii) the syntactic structure transfer rules for target parse tree creation. The syntax of the source language was arrived at by the following steps i) selection of appropriate primitive tagset ii) selection of chunk tagset iii) identification of hierarchical dependencies between chunks.

7.3.1 Selection of POS Tags

First step in deriving the syntactic structure of Malayalam sentences was the identification of set of word categories in a Malayalam sentence called part of speech tags. Since we found that a morpheme based parsing was appropriate for a highly agglutinative language like Malayalam we decided to give a unique tag mane for each morpheme category. The inflectional and derivational suffixes were given separate tag names. The set of tags identified for our problem are listed in Table 7.1.

1. N - Nouns

Nouns are words used as the name of a person, place or thing. Nouns are

further classified into a) common nouns b) proper nouns c) collective nouns d) abstract nouns.

a) Common nouns - Common nouns are names given in common to every person.

കുട്ടി (*kutti*/child), മരം (*maram*/tree)

b) Proper nouns - Proper nouns is the name of some particular place or person.

സീത (*sita/*sita), ഡെല്ഹി (*delhi/*Delhi)

c) Collective nouns - Collective noun is the name of a number of persons or things taken together and spoken of as one whole.

ജനം (*janam*/people), കുടുംബം (*kutumbam/*family)

d) Abstract nouns – Abstract nouns is the name of a quality, action or state considered apart from the object to which it belongs.

Quality- ക്രൂരത (*krooratha*/cruelty), ധൈര്യം (*dhairyam*/bravery)

Action – ചിരി (*chiri*/smile), കരച്ചില് (*karachchil*/cry)

State – കുട്ടിക്കാലം (*kuttikkalam*/childhood)

Abstract nouns are usually formed from adjectives, verbs and common nouns with proper modifications to root.

കുട്ടിക്കാലം (*kuttikkalam* / childhood) formed from noun കുട്ടി (*kutt* /child)

കരച്ചിൽ (*karachchil*/cry) formed from the verb root കര (*kara*/verb root for cry)

Another classification for nouns are a) countable nouns b) uncountable nouns

a) Countable nouns

Countable nouns are names of objects, people etc. that we can count.

പേന (*pEna*/pen), കുട (*kuta*/umbrella)

b) Uncountable nouns

Uncountable nouns are names of things which we can not count.

പാല് (*paalz*/milk), എണ്ണ (*eNNa*/oil)

Nouns have gender, number and case information attached to them. The Penn tag NN has been adopted as such for common nouns. Penn tagset makes a distinction between noun singular and noun plural. As mentioned earlier, distinct tags based on grammatical information were avoided. This reduces the number of tags and hence helps achieve simplicity and consistency. The same tag N had been used for all kinds of nouns. The tagset derived at IIIT for Hindi language includes a separate tag for proper nouns. Such is not required in languages like English as the Proper nouns in the text are marked by capital letters. This is not the case with many of the Indian languages as:

a) Indian languages, unlike English, do not have any specific marker for proper nouns in orthographic conventions. English proper nouns begin with a capital letter which distinguishes them from common nouns.

b) all proper nouns in Indian languages are otherwise used as common nouns containing a lexical meaning. For example, the names വിശാലം / *viSaalam*, വേണു / *vEnu* etc. can occur as proper nouns and also common noun. This poses a problem during translation. The common nouns should be translated to their equivalent target language words whereas the proper nouns should be used as such.

1. രാജുവിന്റെ മനസ്സ് വിശാലം ആണ്

   raajuvinte manassu (viSaalam) aanu

   Correct translation: Raju's mind is (broad)

2. വിശാലം നല്ല കുട്ടിയാണ്

   (viSaalam) nallakuttiaanu

   Correct translation: (*ViSaalam)* is a good student.

In sentence 1, the word വിശാലം *viSaalam* occurs as a common noun and needs translation. In sentence 2, the same word occurs as proper noun and should not be translated. The identification of proper nouns can be better achieved by named entity filters. We have not used a separate tag for proper nouns as we have assumed that the proper nouns are distinct from common nouns. IIIT tag set includes a separate tag for proper nouns for manual annotation and ignores it for machine learning algorithms. This tag is also similar to the Penn tagset and IIIT tag set uses NNP as the tag for proper nouns.

2. V Verb Finite

V is used to mark a verb root. A verbal construction such as the following is finite :

കുട്ടികൾ വീട്ടില് കളി-ക്കുന്നു (*kuttikaL veettil kaLi-kkunnu)*

children are playing in the house

Here the verb root kali is given the tag verb.

Marking the finiteness or non-finiteness as in IIIT tag set was unnecessary for our problem. All verb roots are tagged as V whether it is part of a finite verb or non finite verb.

3. VA Verb Auxiliary

All auxiliary verbs which marks inflections for verb roots due to tense, aspect and mood will be marked as VA. VAUX is the tag which has been adopted in Penn tag set and IIT tagset.

ഉം (*um* / future tense marker), ഉന്നു (*unnu* / present tense marker)

4. VNN Gerund suffix

A separate tag, VNN, for gerunds has been kept as functionally they are more like nouns and take on the nominal postpositions. This distinction is made in order to preserve the information that this word is a form of a verb. Every verb is capable of taking its own arguments in a sentence, even when it occurs in a nominalised form. The suffix അത് / athu forms the gerund suffix.

a) എനിക്ക് നീന്തുന്നത് ഇഷ്ടമാണ് (enikku neenthunnathu ishatamaanu)

I like swimming

നീന്തുന്നത് (*neenthunnathu*/swimming) functions as a noun in the sentence.

b) എനിക്ക് പഴം കഴിക്കുന്നത് ഇഷ്ടമാണ് (I like eating banana)

enikku pazham kazhikkunnathu ishtamaanu

The gerund takes an argument in the above sentence. Noun പഴം / *pazham* is an object of the verb കഴിക്കുന്നത് / *kazhikkunnathu* and has no relation to the main verb. Therefore, in order to be able to show the exact verb-argument structure in the sentence, it is essential that this crucial information of a noun derived from a verb is preserved. The verbs having അവൻ / *avan* and അവൾ / *avaL* as suffix are also marked as VNN.

5. INFA verb infinitive suffix

This tag is to mark the infinitival verb form. Infinitive form of the verbs ends with aan. In Hindi the gerunds and infinitives end with –na. Since in Hindi both behave functionally in a similar manner, the distinction is not very clear.

രാമു രാജുവിനെ അടിക്ക് ആൻ പോയി (Ramu went to hit Raju)

ramu rajuvine atikk – aan- pOyi

Here verb root അടിക്ക് / *atikk* suffixed by ആൻ / *aan* forms the infinitive.

6. PA AdjectiveA word used with a noun to describe the person, animal, place or thing which the noun names or to tell the number or quantity is called an adjective. Adjectives are of following types:

i) *adjectives of quality*: this shows quality of a person or thing.

ഡെല്ഹി വലിയ നഗരമാണ് (*delhi valiya nagaramaaNz)*

Delhi is a big city

Here വലിയ (*valiya* / big*)* is the adjective of quality.

ii) *adjective of quantity* : This shows how much of a thing is meant.

രാമു കുറച്ച് ആഹാരം കഴിച്ചു (*raamu kuRachu aahaaram kazhichchu)*

Ramu had some food

In this sentence കുറച്ച് (*kuRachchu /* some*)* is the adjective of quantity.

iii) *Number adjectives* : this shows how many persons, things etc. are meant.

കൈയില് അഞ്ചു വിരലുകളുണ്ട്

kaiyil –anchu- viralukaluntu.

The word അഞ്ചു *anchu* is the number adjective here.

iv) *Demonstrative adjectives:* point out which person or thing is meant. They are ആ / *aa* and ഈ / *ee.*

ഈ സാരി പുതിയതാണ് *(ee) saari puthiyathaanu*

This sari is new

Here ഈ / *ee* is the demonstrative adjective which qualifies സാരി / *saari.*

In Malayalam the words for demonstratives are different from words for pronouns. Some languages like Hindi uses the same word as demonstratives and pronouns.

Hindi: *vaha ladakA merA bhAI hE*(vaha is used as demnostrative)

Hindi: *vaha merA bhAI hE* (vaha is used as pronoun)

It was decided to use the same tag PA to all these kinds of adjectives. This tag is named as JJ in Penn tagset and IIIT tagset. Penn tagset also makes a

distinction between comparative and superlative adjectives. This has not been considered for our problem.

7. PAV Adverb

A word that modifies a verb, an adjective or another adverb is called an adverb. Adverbs are further classified into:

a) *Adverbs of time:* This shows when the event has happened.

എന്റെ കാല് ഇന്നലെ മുറിഞ്ഞു (*ente kaal innale muRinjnju)*

My leg was hurt yesterday

ഇന്നലെ (*innale*/yesterday) is the adverb used here.

b) *Adverbs of frequency:* This shows how often the event happens.

കുട്ടി എപ്പോഴും കരയുന്നു (*kutti eppOzhum karayunnu)*

The child cries even now

എപ്പോഴും (*eppOzhum*/always) is the adverb of frequency.

c) *Adverbs of place:* It shows where it has happened.

രാജു ഇവിടെ വന്നു (*raaju ivite vannu)*

Raju came here

ഇവിടെ (*ivite*/here) is the adverb of place.

d) *Adverbs of manner*: This shows how the action is done.

അവൻ നന്നായി ഉറങ്ങി (*avan –nannaayi- uRangngi)*

He slept nicely

നന്നായി (*nannaayi*/nicely) is the adverb of manner.

e) *Adverbs of affirmation*

സീത തീ ർച്ചയായും വരും (Seetha will definitely come)

seetha –theerchchayaayum- varum.

തീ ർച്ച യായും Theerchayaayum (definitely) is the adverb of affirmation.

All these classes of adverbs are given the tag PAV. This is RB tag of Penn tagset and IIIT tagset. Penn tagset also makes a difference between comparative and superlative adverbs, which is not adopted in our case. This is in accordance with our philosophy of coarseness in linguistic analysis.

8. NA Postposition

All Indian languages have the phenomenon of postpositions. Some languages like Hindi separate the post positions from the noun. All postpositional suffixes are given the tag NA. All case markers are tagged as PSP by IIIT tagset.

രാജു മധുവിന് പണം കൊടുത്തു (*rajumadhuv-inu- panamkotuththu)*

Raju gave the cash to Madhu

Here the case marker ന് (*inu*/to) is attached to the noun preceding it.

9. C Conjuncts (co-ordinating and subordinating)

The tag C will be used for both co-ordinating and subordinating conjuncts. The Penn tagset has used IN tag for prepositions and subordinating conjuncts. The rationale behind this is that subordinating conjuncts and prepositions can be distinguished because subordinating conjuncts are followed by a clause and a prepositions by a noun phrase. All connectors other than prepositions will be marked as C. The conjucts are ഉം / *um* and ഓ /*oo.*

രാമനും മധുവും കടയിൽ പോയി (Raaman and Madhu went to shop)

raaman-um-madhuvum katayil pOyi

ഉം / *um* is the conjunct used here.

10. QW Question Words

The Penn tagset makes a distinction between various uses of 'wh-' words and marks them accordingly (WDT, WRB, WP, WQ etc). The 'wh' words in English can act as questions, as relative pronouns and as well as determiners. However, for Indian languages we need not keep this distinction. Therefore, we tag the question words as QW.

ആരാണ് മധുവിനെ അടിച്ചത് ? (*aaraaNu madhuvine atichchathu?)*

Who hit *madhu*?

ആരാണ് (*aaraaNu/*who) is the question word.

11. NEG Negative

The words under this tag are ഇല്ല / *illa*, അണ്ട / *anta*, അല്ല / *alla* etc.

In Hindi Negatives like 'nahIM', 'na', etc. will be marked as NEG. Penn tagset does have a separate tag for this.

രാജു വന്ന് - ഇല്ല (*raaju vann illa)*

(Raju did not come)

Here ഇല്ല (*illa*/did not) is the word in this category.

രാജു പോക് - അണ്ട (*raaju pok anta)*

(Raju should not go to shop)

Here അണ്ട (*anta*/should not) following പോക്/*poku* is the NEG.

12. PL Plural marker

The plural markers for nouns we have considered are കൾ/kaL and മാര്/maar.

കുട്ടി - കൾ (*kutti-kaL*/children), നാരി - മാര് (*naari-maar*/ladies)

13. NCA Noun clause marker

The noun clause marker considered are അത്/ *athu* and എന്ന്/ *ennu.*

രാമൻ പോയി എന്ന് സീത പറഞ്ഞു *(raaman pOyi ennz seetha paRanjnju)*

Seetha told that Raman had gone

Here രാമൻ പോയി എന്ന് (Raman *poyi ennu*) is the noun clause which is marked by the suffix എന്ന് (*ennz*/that).

രാമു പറഞ്ഞത് എനിക്കിഷ്ടപ്പെട്ടു (*(raamu paRanjnjathu enikkzishtappettu)*

I liked what Ramu told

Here രാമു പറഞ്ഞത് *raamu paranjnathu* is the noun clause which is marked by the suffix അത് (*athu*/what).

14. ADJA Adjectival suffix

The adjectival suffixes join nouns to form adjectives. The adjectival affixes considered are ഉടെ *ute*, /ആയ / *aaya* and ഉള്ള /*ulla*. The nouns with suffixes ഇലെ /*ile* is treated differently than the other suffixes during the final word alignment phase.

രാമു ദരിദ്രൻ-ആയ മോഹന് പണം കൊടുത്തു(*raamu daridran aaya mohanu paNam kotuththu*)

. Ramu gave money to poor Mohan

In this the suffix ആയ / *aaya* following the word ദരിദ്രൻ (*daridran*/poor) makes the word adjective which qualifies the noun മോഹൻ /*mohan* which follows it.

15. ADVA Adverbal suffix

These are suffixes which forms adverbs when joined with nouns. Two examples are ആയി / *aayi* and ഇലേക്ക് / *ilEkkz.*

രാമു തറ നന്ന് - ആയി കഴുകി (Ramu washed the floor neatly)

raamu thaRa(nannz–aayi-) kazhuki.

In this the suffix ആയി (*aayi* / ly) added to the noun നന്ന് (*nannz* / neat) forms the adverb which qualifies the verb കഴുകി/ *kazhuki.*

16. RP Relative participle

Relative participles form the clause markers for adjective clauses. Some of the words forming the relative participles are പറഞ്ഞ / *paranjnja,* എടുത്ത / *etuththa,* കൊടുത്ത / *kotuththa* etc.

രാജു കൊടുത്ത പഴം സീത കഴിച്ചു (Seetha ate the banana which Raju gave)

(raaju-kotuththa-) pazham seetha kazhichchu.

In the sentence കൊടുത്ത *kotuththa* is the word which belongs to this category which marks the adjective clause രാജു കൊടുത്ത / *raaju kotuththa* which qualifies the noun പഴം / *pazham.*

17. ADVCA Adverbal clause suffix

These form the marker suffixes in adverb clauses. The suffixes in this category are അപ്പോൾ / *appO*L, കഴിഞ്ഞു / kazhinju, അതിനുശേഷം / *athinuSesham*, അതിനുമുന്പ് / *athinumunpu,* അതുതൊട്ട് / *athuthottz,* അതുമുതല് / *athumuthal*, അതിനുവേണ്ടി / *athinuventi,* അതുകൊണ്ട് / *athukontu,* അതുകാരണം / *athukaaranam*, ആൽ / aal, അതിനേക്കാൾ / *Athinekkal* and അതുപോലെ / *athupOle.*

രാമു വിളിച്ചതുകൊണ്ട് ഞാൻ പോയി *(raamu viLichchathukontunjaan pOyi)*

I went because Ramu called

.

In the above sentence അതുകൊണ്ട് (*athukont/*because) marks the end of the adverb clause രാമു വിളിച്ചതുകൊണ്ട് / *raamu viLichathukontu.*

രാമു വിളിച്ചാൽ ഞാൻ പോകും *(raamu viLichchaal njaan pOkum).*

I will go if Ramu calls

In the above sentence the suffix ആൽ (*aal*/if) marks the end of the adverb clause രാമു വിളിച്ചാൽ / Raamu *viLichchaal.*

18. LOC Locative suffix

The locative suffix for nouns ലെ (ile / in) is given a separate tag as the noun with this suffix has to be treated differently compared to possessive suffix during the syntactic transfer.

7.3.2 Selection of Chunk Tags

After selection of POS tags in sentences the chunk tags were identified. The chunks that are to be rearranged for the translation from Malayalam to English were identified and given a unique tag name for each chunk. The tagset includes all of the tags in IIIT tagset and also some additional tags to handle

Table 7.1 POS tags

No.	Tag	Description
1	PL	Plural suffix
3	NA	Postposition
4	PA	Adjective
5	N	Noun
6	V	Verb
7	ADJA	Adjectival suffix
8	ADVA	Adverbial suffix
9	PAV	Adverb
10	VN	Verbal Noun
11	V RP	Verbal Relative participle
12	NCA	Noun clause suffix
13	ADVCA	Adverbal clause suffix
14	INFA	Infinitive
15	DJ	Disjunction
16	C	Conjunction
17	LOC	Locatives
18	VA	Verbal suffix

higher level constructs like clauses. The list of chunk tags is shown in Table 7.2. A chunk tag is allotted for each of the morpheme group found in the hierarchical structure. The tags were so chosen that it helps to identify the morpheme groups to be used in the reordering process to generate the target language parse tree.

Table 7.2 Chunk tags

No.	Tag	Description
1	NP	Noun Group
2	VG	Verb Group
3	NC1	Noun clause
4	ADVC	Adverb clause
5	ADJC	Adjective clause
6	NPC	Conjunct Noun
7	S	Sentence
8	CS	Compound sentence
9	CMPN	Compound noun
10	ADJCNP	Adjectival clause + Noun
11	ADJG	Adjective group
12	INFSG	Infinitive + verb group
13	INF	Infinitive
14	ADVG	Adverb group
15	VGC	Compound verb
16	VA	Verbal suffix
15	ADJLOC	Locative adjective

7.3.3 Hierarchical dependency rules between chunks

The parser needs the hierarchical dependencies among the chunks for the creation of parse tree and also for the final reordering required to build the target parse tree. The sentences of Malayalam language were carefully

analysed to identify these dependency rules and the syntactic structure of sentences in Malayalam were formulated.

Malayalam belongs to Indo- Dravidian family of languages and it is an relatively free word order language like other Dravidian languages. Malayalam is an S-O-V language. The default or unmarked order of constituents is Subject first, then the Object and finally the verb. However, Malayalam, being a relatively free word order language, permits substantial amount of freedom in the order of constituents although normally the verb remains in the sentence final position. Word order is less important mainly because noun groups are marked for cases and the verb agrees with the subject in gender, number and person. Subjects and objects are often dropped. The subject of a sentence is expressed by a noun group in the nominative case in most of the sentences [74]. Normally all modifiers precede the modified.

There are a variety of subordinate clauses. Subordinate clauses also precede the main clause. They typically involve special non-finite forms of verbs which occur invariably in the clause final position and mark the right hand boundary of the respective clauses. All these assertions were used to form the syntax rules. There are exceptional situations where deviations from these rules are possible. Also, most of these rules apply not only to Malayalam but to Dravidian languages in general.

1. ഞാൻ നാളെ വരും (*njaan naaLe varum)*

I will come tomorrow

2. രാമൻ കാട്ടില് പോയി (raaman kaattil pOyi)

Raman went to forest

The nominative case marker is empty in sentences 1 and 2. Malayalam also permits dative subject constructions where the understood subject is indicated by a noun group in dative case whereas the surface subject appears in the nominative case.

1. എനിക്ക് മിഠായി ഇഷ്ടമാണ് (I like sweets)

enikku mittayi ishtamaanz

2. സീതക്ക് നാല് പൂച്ചകൾ ഉണ്ട് (*seethakkz naalupoochakaL untz)*

Seetha has four cats

Malayalam has postpositions unlike prepositions in English. The genitive precedes the head noun in the genitive phrase and the complementiser follows the embedded clause. Adjectives, participial adjectives and free relatives precede the head noun. There is no person, number and gender(PNG) agreement between the subject and the verb as in the case of English. Based on the above facts the hierarchical dependencies among the chunks in a Malayalam sentence are shown in Figure 7.1. The rules are shown as a finite state diagram. The circles shown are the states and the transitions between states are POS tags or chunk tags.

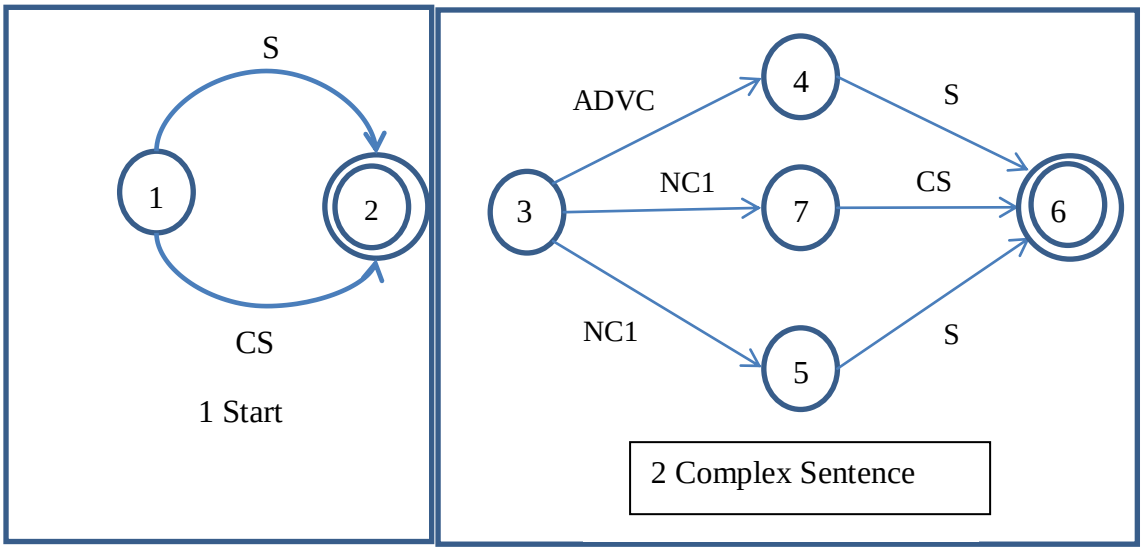
Rules for forming chunks are given below with examples.

1) Start - Highest level chunk

1. S - A simple sentence

രാമു പഠിക്കുന്നു (*raamu paThikkunnu)*

Ramu is studying



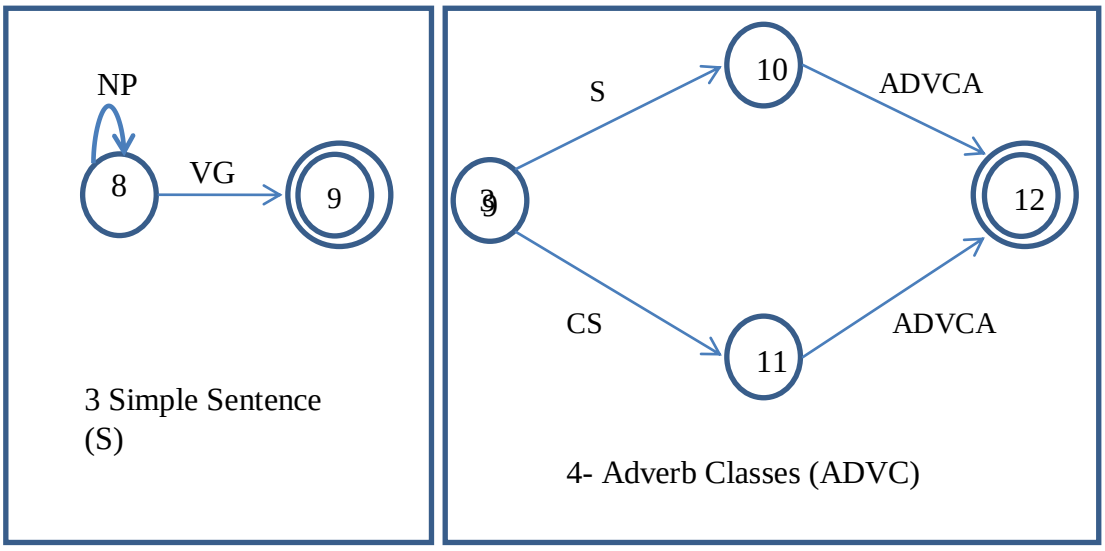


Figure 7.1 Hierarchical dependency rules

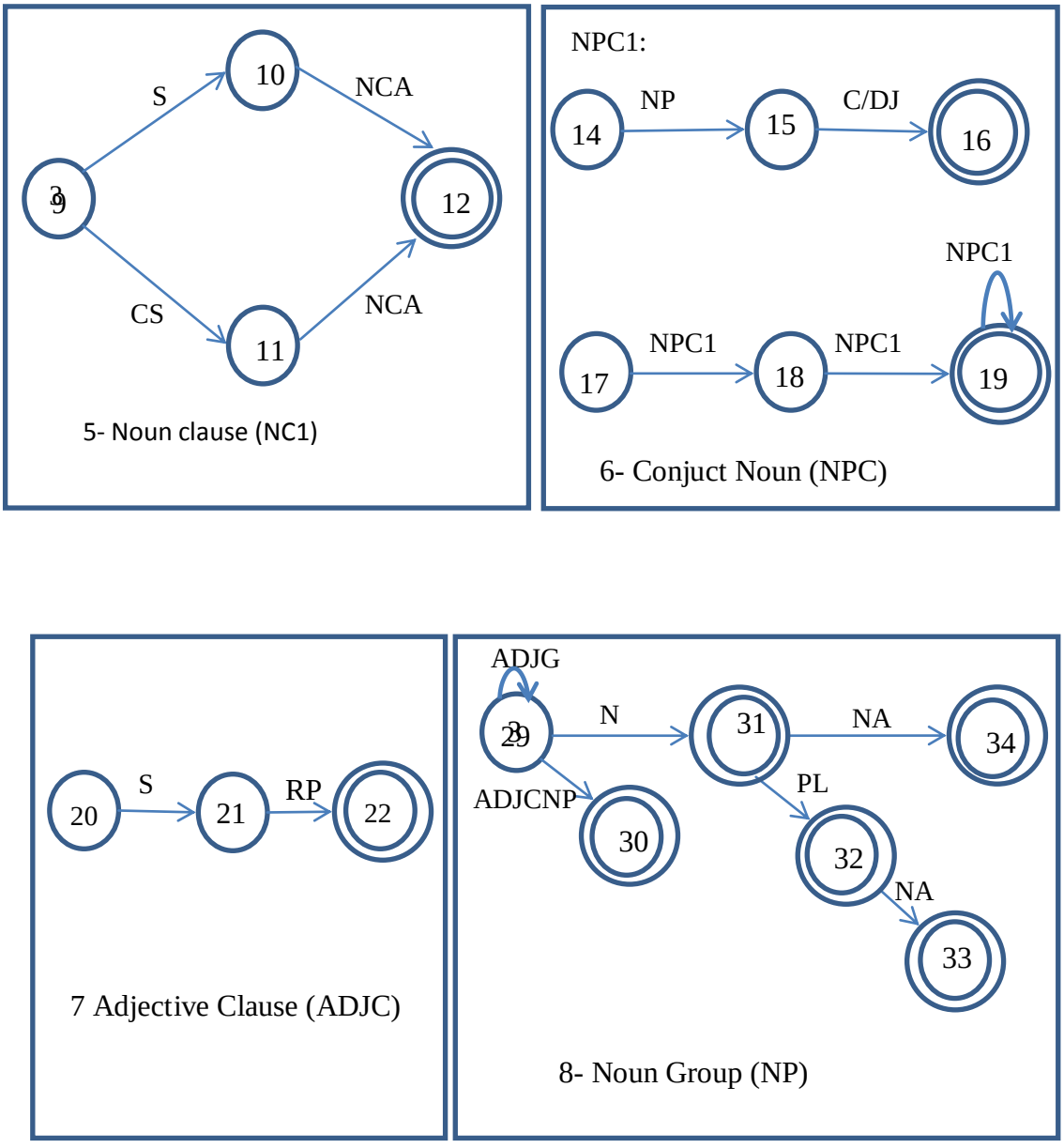


Figure 7.1 Hierarchical dependency rules Contd….

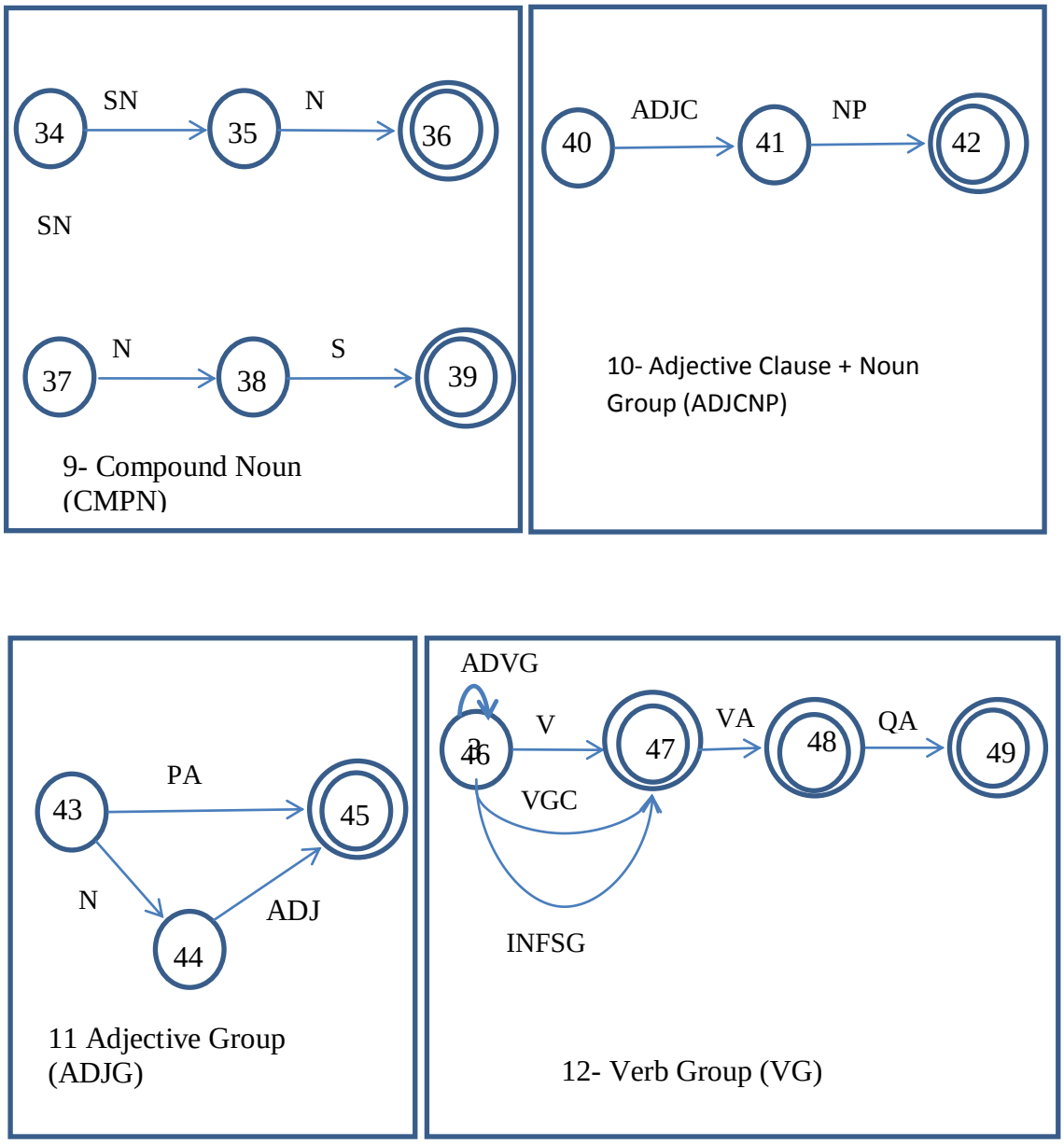


Figure 7.1 Hierarchical dependency rules Contd….

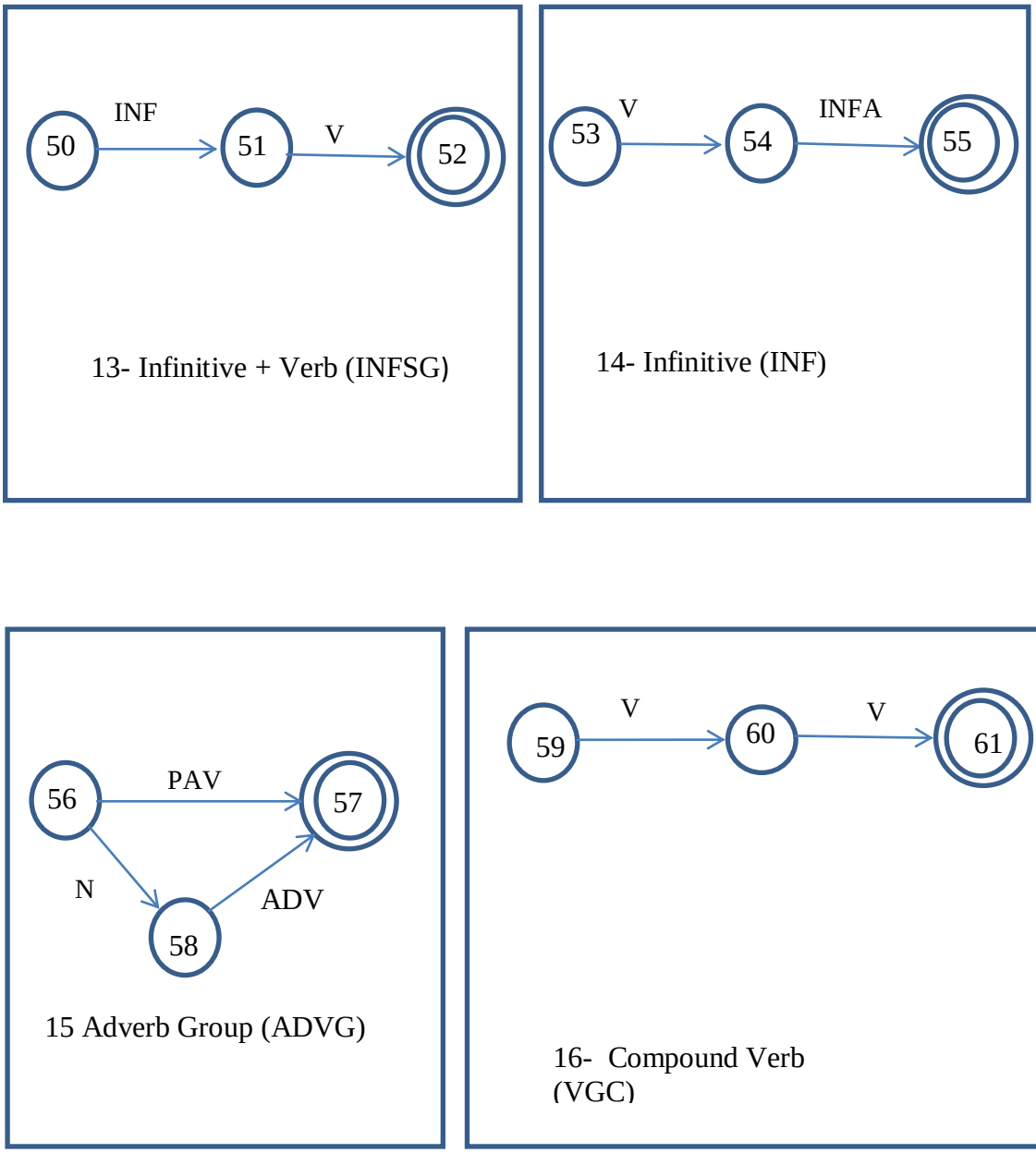


Figure 7.1 Hierarchical dependency rule Contd….

2. CS – Complex sentence

രാമു പഠിച്ചാൽ പരീക്ഷയിൽ വിജയിക്കും

(*Raamupadichaal*)(ADVC) *(pareekshayiljayikkum*)(S)

If Ramu studies he will pass in the examination

2) CS - Complex sentence

1. An adverb clause followed by a simple sentence

രാമു പഠിച്ചാൽ പരീക്ഷയിൽ വിജയിക്കും

(*Raamupadichaal*)(ADVC) *(pareekshayiljayikkum*)(S)

If Ramu studies he will pass in the examination

2. A noun clause followed by a complex sentence

(രാമൻ മോഹനെ അടിച്ചെന്ന്)NC(രമയെക്കണ്ടപ്പോൾ സീത പറഞ്ഞു)CS

(*Raaman Mohane adichchennu)*(NC) (*ramaye kandappOL seetha paRanjnju*)(CS)

When Seetha saw Rama she told that Raman hit Mohan

3. An adverb clause followed by a complex sentence

(രാമു പഠിച്ചാൽ) (ADVC) (പരീക്ഷയിൽ വിജയിക്കുമെന്ന് മോഹൻ പറഞ്ഞു)(CS)

(*raamu padichaal)*(ADVC)(*pareekshayil jayikkumennu mohan paranju*) (CS)

Mohan told that if Mohan studies he will pass in the examination

4. A noun clause followed by a simple sentence

(രാമൻ മോഹനെ അടിച്ചെന്ന്)(NC) (സീത പറഞ്ഞു)(S)

(*raaman mOhane adichuennu*)(NC) (*seethaparanju)*(S)

Seetha told that Raman ad hit Mohan

3) S - Simple sentence

One or more noun groups followed by a verb group.

NP(രാമൻ) NP(മോഹനെ)VG(അടിച്ചു)(Raman hit Mohan)

NP*(raaman*) NP(*mohane)* VG(*atichchu*)

കുട്ടികൾ ക്ലാസ്സില് പോകാന് പോകുന്ന (Children are about to go to class)

NP(*kuttikaL)* NP(*cLassil)* VG(*pOkaanpOkunnu)*

4) ADVC - Adverb clause

A simple sentence followed by adverb clause marker.

S(രാമു വന്ന്) CONDP(ആൽ) (S(*raamu vann*) CONDP(*aal))*

if *Ramu* comes

5) NC1 - Noun clause

A sentence followed by the clause marker എന്ന / ennz forms noun clause.

രമ വന്ന എന്ന മോഹൻ പറഞ്ഞു ((*rama vannu*)(S) *ennu*(NCE1) (*mOhan*

paRanjnju)(S))

(Mohan told that Rama had come)

6) ***NPC - Noun Conjunct***

A noun group followed by the conjunct suffix ഉം / *um* forms a conjunct noun.

രമ / rama(NP) – ഉം / *um*(C) രവി / ravi (NP)– ഉം / *um* (C) (Rama and Ravi)

7) ***ADJC - Adjective clause***

A sentence followed by relative participle forms an adjective clause.

(സീത പറഞ്ഞ)(ADJC) കഥ രമക്കിഷ്ടപ്പെട്ടു ((*seetha paRanjnja*)(ADJC) *kadha Ramakkuishtappettu*)S

(Rama liked the story which Seetha told)

8) ***NP - Noun chunk***

1.A noun alone.

രാമൻ (*raaman* / Raman)

2.A noun followed by a case marker

രാമൻ - ഓടു (*raaman-Odu* / to Raman)

3.A noun followed by a plural marker and a case suffix

കുട്ടി - കൾ - ഓടു (*kutti-kaL-Odu* / to children)

4.A noun preceded by an adjectival clause

രമ പറഞ്ഞ കഥ (the story which Raman told)

rama paRanjnja kaTha

9) CMPN - Compound noun

A noun followed by another noun.

വിവാഹ - മോതിരം (*vivaaha-mOthiram* / wedding ring)

10) ADJCNP - Noun preceded by an adjective clause

The adjective clause and the noun it qualifies are grouped as they are to be treated as a single unit during structure transfer from Malayalam to English.

11) ADJG - Adjective chunk

1.A pure adjective

നല്ല (*nalla* / good), കുറെ (*kure* / some)

2.A derived adjective formed by a noun followed by adjectival suffixes.

ഭംഗി (*bhangi* / beautiful) – ഉള്ള *(ulla*)

12) VG - verb group

1. Zero or more adverb group followed by a verb, verb and inflectional suffixes, verb, inflectional suffix and question tag.

പോയി (V)(*pOyi*/went), പോക് (V) (*pOk*)– ഉന്നു (VA)(*unnu* /is going)

2. A Compound verb i.e. a verb followed by another verb

ചാടി /*chaadi* (V) കയറി/ *kayar*i(V)(climbed jumping), ഓടി / *Odi*(V)

പോയി / *pOyi*(V)(went running)

3. Infinitive followed by a verb

പോക് / *pOk*(V)- ആൻ / *aan*-(INFA) പോയി *pOyi*(V) (went to go)

13) INFSG - Infinitive followed by a verb group

The infinitive and the verb following it are grouped.

പോകാൻ / *pOkaan*(INF) തുടങ്ങി / *thutangi*(V)(started to go),

വാങ്ങാൻ / *vaangaan*(INF) പോയി / *pOyi*(V)(went to by)

14) INF- Infinitive

A verb followed by the suffix ആൻ / *aan* is taken as infinitive.

പോക് / *pOk*(V) – ആൻ / *aan*(INFA), വര് / *var*(V)- ആൻ / *aan*(INFA)

15) ADVG - Adverb group

1. Pure adverb (PAV)

പതുക്കെ / *pathukke*(slowly), പെട്ടെന്ന് / *pettennu*(quickly)

2. Noun followed by adverbial suffix

ഭംഗി / *bhangi*(N)- ആയി / *aayi*(ADVA)(beautifully)

16) VGC- Compound verb

A verb followed by another verb are grouped to form a compound verb.

ചാടി / *chaati*(V) – കയറി / *kayaRi*(V), നടന്ന് / *natannu*(V) – പോയി/ *pOyi(V)*

7.3.4 Syntactic structure difference between Malayalam and English

The different classes of sentences in Malayalam and their English translation were carefully analysed and the set of reordering rules required in Malayalam to English translation process were derived. It was found that the reordering occurs in the word level and also in the chunk level. 13 rules wee identified for reordering which are given below along with examples. The set of chunk transfer rules are shown in Table 7.3.

1. When an adverb clause is encountered the adverb clause marker is moved to the beginning of the clause. In the following sentence the clause marker അപ്പോൾ (*appOL*/when) in the end of the clause is moved to the beginning of the clause.

മോഹൻ സീതയെ അടിച്ച് –(അപ്പോൾ)

mOhan seethaye adichch-(appOL)

(when) Mohan hit Seetha

2. When a noun clause is found, the noun clause marker is moved to the beginning of the clause. In the following example the clause marker എന്ന് (*ennu*/that) is moved to the beginning of the noun clause.

മോഹൻ വീട്ടിൽ പോയ് –(എന്ന്)

mOhan veettil pOy-(ennu)

(that) Mohan went to house

3. When a noun group with a noun and a case suffix is found position of both are interchanged. In the following example positions of വീട് (*veetz*/house) and the case marker ഇൽ (*il* / to) are interchanged.

വീട് ഇൽ

veett-il

to house

4. When a simple sentence is found the verb is placed after the subject. If the sentence does not have a subject it is placed before the object. In the example the verb പോയി (*pOyi*/went) is positioned after the subject മോഹൻ / mOhan.

മോഹൻ വീട്ടിൽ പോയി

mOhan veettil (pOyi).

Mohan (went) house to.

5. When an adjective clause is found the adjectival marker is placed before the clause. The marker is replaced by who or that depending on whether the noun it qualifies is human or nonhuman. In the following example the adjective clause marker അ / a (which/who/what) is moved to the beginning of the clause.

രാമൻ കൊടുത്ത് - അ

raman kotuthth-a

(which) *raaman* gave

6. When an adjective clause is followed by a noun group the noun group is placed in the beginning of the clause. In the example given below the noun following the adjective clause കുട (*kuta*/umbrella) is moved to the beginning of the clause.

രാമൻ കൊടുത്ത (കുട)

raaman kotuththa (kuta)

(umbrella) which raman gave

7.When complex sentence with one subordinate clause and main clause is found the main clause is written first. In the given example the main clause സീത പറഞ്ഞു / *seetha paRanju* is written first.

മോഹൻ വീട്ടിൽ പോയെന്ന് സീത പറഞ്ഞു

mOhan veettil pOyennu (seethe paRanjnju)

(Seetha told) that Mohan went to house

8. When an infinitive is found the infinitive stem and suffix are exchanged. The infinitive marker ആൻ (*aan*/to) and the verb root പഠിക്ക് (*paThikkz*/study) are interchanged.

പഠിക്ക് - ആ ന്

padhikk-aan

to study

9. When a verb group is encountered the inflection suffix and root are exchanged. Proper modifications are made to the verb root. In the following example the verb ഓട് /Otz(run) and the suffix ഇക്കൊണ്ടിരുന്ന / ikkontirunnu(had been) are interchanged.

ഓടി കൊണ്ടിരുന്ന

ot-ikkondirunnu.

had been running

10. When a verb follows an infinitive they are interchanged. In sentences with objects the infinitives are placed after the object. In the following example the infinitive അടിക്കാൻ(*adikkaan/*to hit) and the verb following it are interchanged.

അടിക്കാൻ പോയി

atikkan- pOyi.

went- to hit

11.The adjective group i..e. a noun with adjective derivation markers (ലെ / ile) and the noun group following it should be interchanged. In the following example മരത്തിലെ / maraththile is a derived adjective formed from മരം/ maram(tree) and the marker ലെ (*ile* / in). The adjective and the noun it qualifies are interchanged.

മരത്തിലെ കിളി

maraththile kiLi

bird in tree

12. Adverb and verb following it should be interchanged. In sentences with objects the adverb is placed after the object. In the following example the verb പോയി (*pOyi*/went) and the adverb പെട്ടെന്ന്(*pettennz*/quickly) are interchanged.

പെട്ടെന്ന് പോയി

pettennz pOyi

went quickly

7.4 Parser model

A top down depth first search method was chosen as the parsing strategy [93]. The parser tries to find all valid parser trees for the source sentence. The parser creates a source parse tree using the syntax rules given in Figure 7.1. It uses synchronous tree adjoining grammar (STAG) approach to create the target

parse tree. Synchronous grammars for TAGs were introduced by Shieber and Schabes to characterize correspondences between tree adjoining languages [111,112]. Synchronous TAGs are used for relating TAGs for two different

Table 7.3 Structure transfer rules

No	Source structure	Target structure
1	S ADVCA	ADVCA S
2	S NCA	NCA S
3	N NA	NA N
4	NP* VG	NP VG NP*
5	S ADJA	ADJA S
6	ADJC NP	NP ADJC
7	ADVC S	S ADVC
8	V INFA	INFA V
9	V VA	VA V
10	INF V	V INF
11	N LOC	LOC N
12	ADJLOCN	N ADJLOC
13	ADVG V	V ADVG

languages for the purpose of machine translation or generation or semantic analysis [113].

The model keeps both source and target tree pair, and performs operations simultaneously while traversing through the tree nodes. Thus the syntax

structure transfer from Malayalam to English is integrated into the parser [114]. The syntax rules were given in the regular expression form. The longest rules are listed first on the right hand side of each production rule so that longer chunks are recognized first. Regular expression notation helps for a compact representation of language syntax and also it provides easy modification of the syntax rules. Some examples regular expressions for language syntax and production rules are given below. The compete set of syntax rules in the context free grammar notation is given in Table 7.4.

1. S-> NP*VP

The rule states that a simple sentence is a sequence of noun chunks followed by a verb chunk.

2. NG-> ADJ*N

The rule states that a noun chunk consists of a set of adjectives followed by a noun.

3. VG ->ADV* V| INF V

The rule states that a verb chunk consist of a sequence of adverbs followed by a verb or an infinitive followed by a verb.

7.5 Algorithm of the parser

The algorithm works in a recursive manner. The search begins with the start symbol of the set of grammar rules and works in a top down manner. As each symbol is recognized the sub tree is added to the source parse tree and the subtree with the required change in order is added to target parse tree. Each

Table 7.4 Grammar rules in regular expression form

Sl. No.	Production rules
1	START=>S|CS
2	CS=>ADVC S|NC1 S
3	ADVC=>S ADVCA
4	NC1=>S NCE1
5	S=>NP^{+} VG
6	NPC1=>NP C NPC=>NPC1 NPC1|NPC1 NPC1 NPC1*
7	ADJC=>NP* VRP
8	ADJG=>PA|N ADJA | ADJLOC ADJLOC=>N LOC
9	NP=>ADJG* N|ADJG* N NA|ADJG* N PL NA|ADJG* N PL|ADJG* NPC|ADJG* NC2 NA|ADJC NP|ADJLOCN ADJLOCN=>ADJLOC N
10	VG=>ADVG* V NE|ADVG* VG1|ADVG* V|INFSG|INFG|ADVG* V QA| N CVA
11	ADVG=>PAV|N ADVA
12	INFSG=>INF V | INF V VA
13	INF=>V INFA

rule on the right side of the production is tried with the following algorithm.

1. Initialise two trees to store the parse trees for the source and target parse trees identified.

2. For each possible untried symbol in the grammar rule do steps 3-7

3. If the symbol is a primitive morpheme tag then if it matches with the word in input move to the next symbol in the rule and advance the input pointer to get the next morpheme. Create a node with the matched morpheme. Add it to current parse tree. Go to step2.

4. Else if there is a mismatch then the next rule for the nonterminal is tried. The input pointer is reset to the initial position.

5. Else if the transition is on a chunk tag then search for the symbol recursively. If it is a successful parse add the sub tree returned by the call to the current parse tree and the reordered tree to the target parse tree. go to step 2. If it is a failure then the next rule for the nonterminal is tried. The input pointer is reset to the initial position.

6. Else if all the symbols for the current rule are satisfied then the parse tree for the non terminal is returned.

7. Return failure.

8. Stop.

7.6 Creation of source parse tree

A sample grammar is given below. The working of the top down parsing algorithm for the sample input '*Mohan paatunnu'* is shown in Figure 7.2.

S-> NP VP | VP
NP->Adj Nominal | Noun
Nominal -> Noun | Noun Nominal

VP -> NP Verb | Verb
Verb ->uRangunnu | padhikkunnu |natakkunnu| paatunnu
Noun ->raam | mOhan

In level 3, in substitution 1 the first terminal adj does not match with the first input which is a noun Mohan. So the next alternative for NP is tried. Now noun matches with the first input. So the same procedure is repeated for the VP part of level 3. If at any level all the options fail then control goes back to the previous level and there other options are tried and a depth first search starts with the new substitution. A successful parse corresponds to a tree which matches exactly with the words in the input sequence.

7.7 Creation of target tree

A parse tree for a Malayalam sentence and its target tree in English created by the parser are shown in Figure. 7.3.

രവി പുസ്തകം വാങ്ങാൻ പോയി (ravi pusthakam vaangngaan pOyi)

Ravi went to buy book

More sample results are given in Appendix 3. The parser uses rule no. 8, 10 and 4 of the structure transfer rules described in this chapter for parsing the above sentence. The target parse tree is created along with the source tree in a depth first manner. When the subgroups are formed during parsing appropriate rearrangement rules are applied to the subgroup in the target parse tree. For

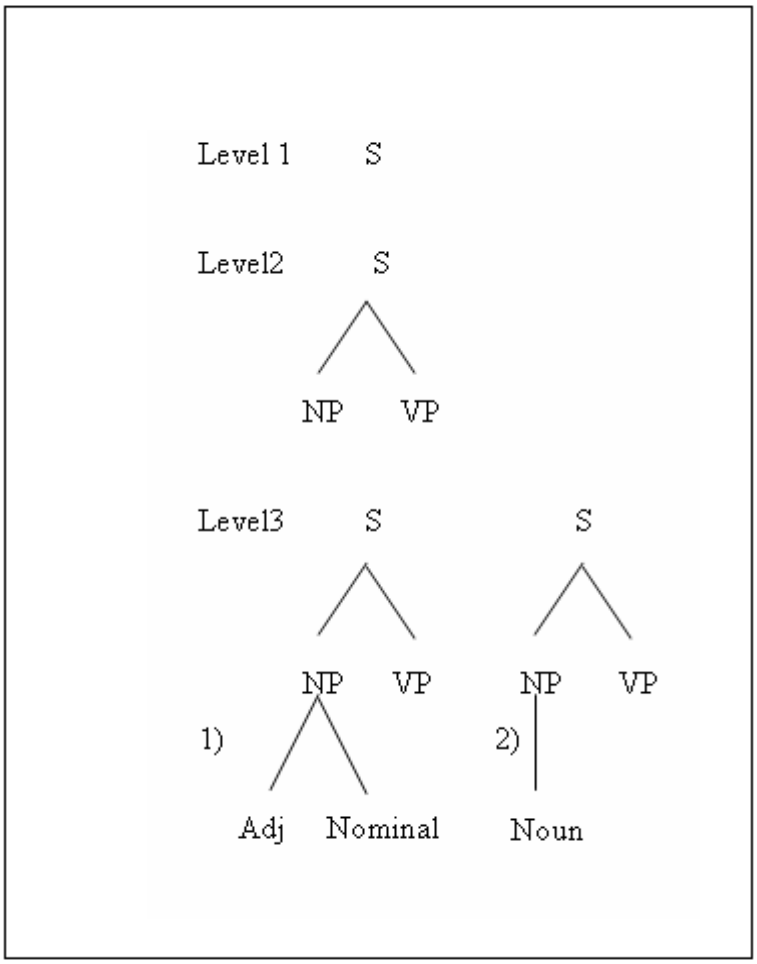


Figure 7.2 parse tree creation in top down parsing

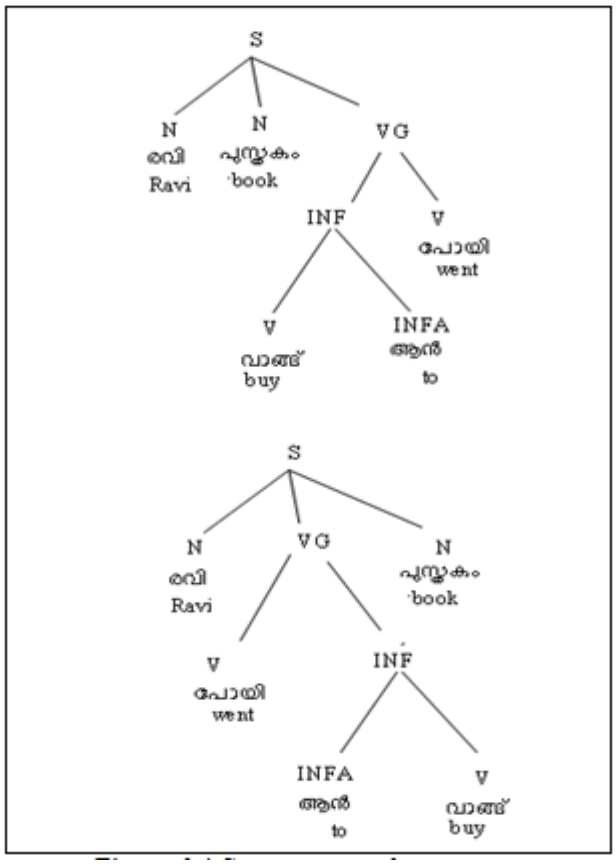


Figure 7.3 Source tree and target tree

example in the above sentence, when the group INF is recognized, reordering rule no. 8 is applied to interchange the suffix and the verb root. After that when VG group is found rule no.10 is applied to interchange infinitive and finite verb went. Finally when the simple sentence S is recognized, the sub tree for the verb group VG is placed after the first noun group(i.e. subject) according to rule 4. Thus when the parsing of the source sentence is complete the target parse tree is created for the target sentence generator module.

7.8 Conclusion

The chapter discussed the design and development of the parser module of the prototype translation system. It clearly described the working of the parser algorithm using sample inputs. The chapter also discussed the set of part of speech tags, chunk tags and the chunk dependencies derived to be used by the parser. Only a subset of the tagset developed by IIIT was identified for the application. The tagset can be extended to handle other sentence classes. The selected set of syntactic structure transfer rules required in the translation of Malayalam sentences to the corresponding English sentence were also discussed in this chapter.

Chapter 8

RESULTS AND DISCUSSIONS

8.1 Introduction

The outputs generated by each of the modules were analysed to find the various categories of outputs generated. This analysis helped to modify the rules used by each module and the dictionary content design for better results. The analysis was conducted in three levels: the analyzer level, parser level and the MT system as a whole. The analysis for each phase was carried out using a test corpora which was a collection of naturally occurring text from many domains. The test data was selected from randomly selected news, literature and children stories. Care had been taken to ensure that sentences used a variety of constructs. All possible constructs including simple as well as complex ones were incorporated in the test data set.

8.2 Morphological analyser

The morphological analyzer generated three kinds of output for the compound words in the tested texts.

1. ***Correct unique splits***: Here the analyser generated only one split which is the correct split for the compound word.

    ഭംഗിയായി (beautifully)= ഭംഗി + ആയി

2. ***Correct multiple splits***: Due to ambiguity in the splitting rules in Malayalam language some compound words are split into more than one set of morphemes. All of them are correct according to the syntax rules of the language. More

context knowledge is needed to arrive at the correct translation for the compound word.

The compound word പടക്കത്തിന് has three splits :

a) പട (army) + കത്ത് (letter) + ഇന് (case suffix)

b) പടം (picture) + കത്ത് (letter) + ഇന് (case suffix)

c) പടക്കം (crackers) + ഇന് (case suffix)

3. ***Multiple splits with incorrect split***: For some compound words the system generated multiple splits. But the correct one could be identified using the syntax rules of the language.

The compound word കരിമ്പനക്ക് has two splits:

a) കരിം (black) +പന (palm) + ക്ക് (suffix)

b) കരിമ്പ് (sugarcane) + അനക്ക് (move)

The result (b) is syntactically wrong and the parser will be able to identify the correct split from the set of splits. Because according to the syntax rules identified the verb root will have a attached suffix and it never appears in the free form in sentences.

8.3 Parser

The parser generated three kinds of output:

1. *Unique correct parse*

In this the parser generated a single parse tree which is syntactically correct.

രവി പുസ്തകം വാങ്ങാൻ പോയി

(Ravi went to buy book)

Source parse tree: S(N(രവി/Ravi) N (പുസ്തകം/book) VG(INF(V (വാങ്ങാന്/buy) INFA (ആൻ/to)) V (പോയി/went)))

Target parser tree: S(N(Ravi) VG(V(went)INF(INFA(to) V(buy)) N(book))

2. ***Wrong parse***

The correct translation of the following sentence could not be found with our present set of syntax rules and the structure transfer rules.

Malayalam: മോഹൻ പറഞ്ഞ കഥ രാമൻ കേട്ടു

Correct translation: Raman heard the story which Mohan told

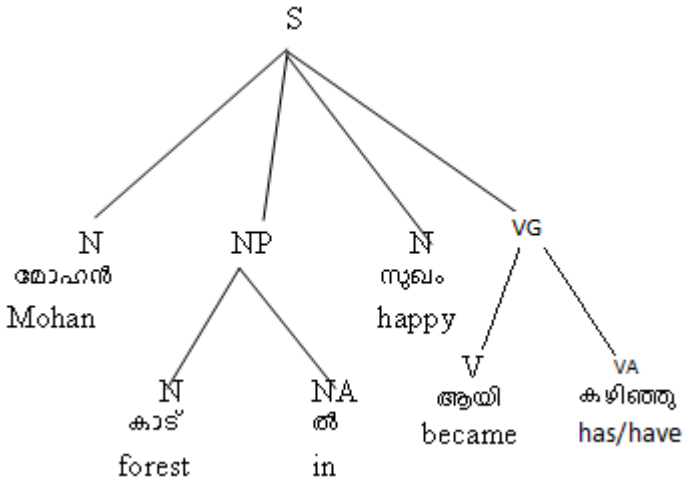
Source Parse tree: (S(ADJCNP(ADJC(N(മോഹൻ/Mohan)V(പറഞ്ഞ/told)) RP(a/which))N (കഥ/story)) N(രാമൻ/Raman) V(കേട്ടു/heard))

Target parse tree: (S(ADJCNP(N (story) ADJC(RP(which)S(N(Mohan) V(told)))))V(heard) N(Raman)
Output from the translator : Story which Mohan told heard Raman

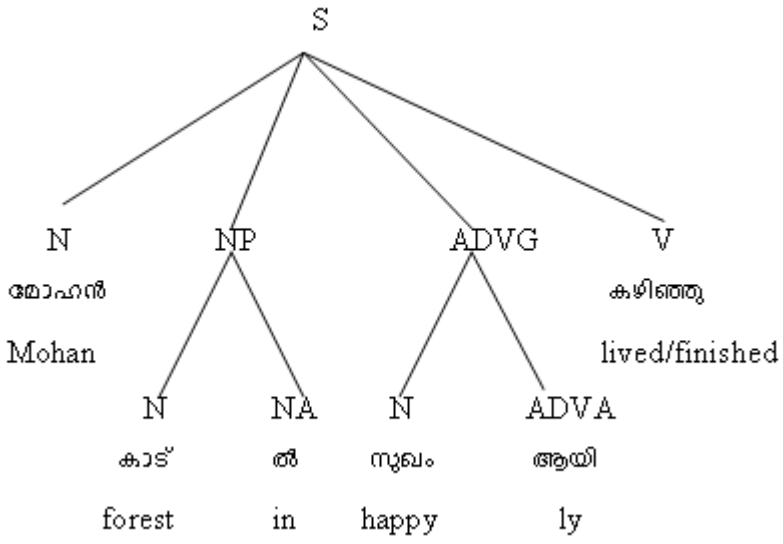
3. ***Multiple parses which includes correct parse***

An example for this case is given below. The parse trees generated for the two parses are given in Figure. 8.2 and Figure 8.3. In this the correct target parse tree is the second one shown in Figure 8.3.

മോഹൻ കാട്ടിൽ സുഖമായിക്കഴിഞ്ഞു (*mOhan kaattil sughamaayi kazhinjnju*)

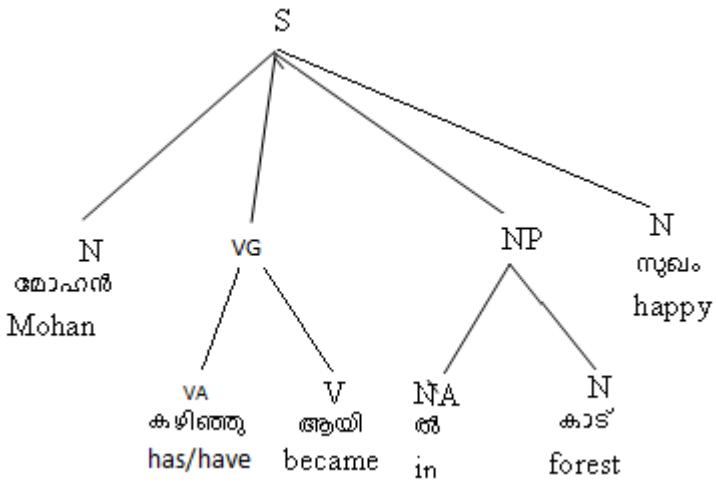


(a)

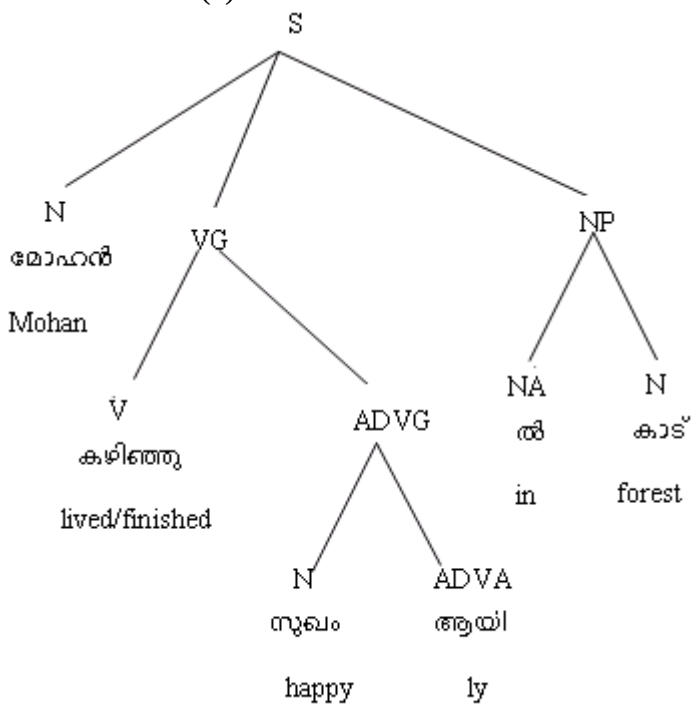


(b)

Figure. 8.1 Multiple source parse trees



(a)



(b)

Figure 8.2 Multiple target parse trees

Two more examples for multiple parse trees are given below. In example 1, b is the correct parse. In example 2 b is the correct parse.

1. കടൽകരയോട്ടുചേർന്നു (The sea joined the land)
a. Source parse : S(N(കട ൽ/sea) N(കര/land) N(ഓട്/tile) V(ചേർന്നു/joined))
Target parse: S(N(കട ൽ/sea) V(ചേർന്നു/joined) N(കര/land) N(ഓട്/tile))
b. Source parse: S(N(കടൽ/sea) NG(N(കര/land) NA(ഓട് /to)) V(ചേർന്നു/joined))
Target parse: S(N(കടൽ/sea) V(ചേർന്നു/joined) NG(NA(ഓട് /to) N(കര/land)))
c.Source parse : S(N(കടൽ /sea) N(കര/land) VG(V(ഓട്/run)V(ചേർന്നു/joined)))
Target parse : S(N(കടൽ /sea) VG(V(ഓട്/run)V(ചേർന്നു/joined)) N(കര/land))

2.നീമാറിയാൽഞാൻവരും
a. Source parse : S(N(നീ/you) N(മാറ് /chest) N(ആൽ/banyan) N(ഞാൻ/I) V(വരും/will come))
Target parse: S(N(നീ/you) V(വരും/will come) N(മാറ് /chest) N(ആൽ/banyan) N(ഞാൻ/I))

b. Source parse: CS(ADVC(S(N(നീ/you) V(മാറ്/move))CONDP(ആൽ/if)) S(N(ഞാൻ/I) V(വരും/will come)))
Target parse : CS(ADVC(CONDP(ആൽ/if) S(N(നീ/you) V(മാറ്/move))) S(N(ഞാൻ/I) V(വരും/will come)))

The system has been implemented in such a way that the parser finds all parses valid for the current input and the translator creates the translations corresponding to each of these parses. The selection of the correct translation is left to the user. This has been one of the drawbacks of the system which should be rectified using appropriate techniques. Use of statistical approaches for parsing with a corpus of correctly parsed sentences can reduce multiple parses greatly. Hidden Markov models have been used for the purpose[17]. The number of wrong parses can be eliminated through the use of better tag set and unambiguous grammar rules. Preprocessing of the morpheme sequence to

eliminate wrong tags for a word examining adjacent tags also can reduce the wrong parses greatly. Keeping agglutination minimum in the source language sentences can also reduce wrong parses.

8.4 Prototype Translator

The following three kinds of translations are obtained:

1) ***Incorrect translation due to syntax limitation***: There are many cases where the output is incorrect due to limitations in the syntax. Many sentences do not fall in the structure we have derived. We have assumed that clauses can not overlap. This creates problems in translation of following type of sentence which is very common in spoken language. One of the cases is shown below:

Malayalam: രാമന് മോഹൻപറഞ്ഞ കഥ കേട്ടു

Parsed output : (S (NG (ADJC (രാമന് മോഹൻപറഞ്ഞ) കഥ) V (കേട്ടു))

Translated output : story which Raman told Mohan heard.

Correct translation: Raman heard the story which Mohan told

Here the error occurred because രാമൻ കേട്ടു is the principal clause and according to our syntax rule this should come after the subordinate clause മോഹൻപറഞ്ഞ (adjective clause) and കഥ, the noun it qualifies. The system will not be able to generate the correct translation for the following word order for the input sentence as it violates the rule for the input that the subject should come in the first place in the sentence:

Malayalam: മോഹന് പറഞ്ഞ കഥ രാമന് കേട്ടു

Translated output: the story which Mohan told heard Raman.

2) ***Correct unique translation:*** In many cases it generates a single translation which is correct in structure and meaning.

Malayalam: രാമൻ ഒരു കുട്ടിയെ അടിച്ചു

Translated output: Raman hit a boy

3) ***Multiple translation with correct translation***: The system generates multiple translations due to two reasons a) the splitter create more than one split for a sentence. b) The parser generates more than one parse tree for a sentence since words can have multiple tags.

a) due to error from the splitter

പണപ്പെട്ടി will be split into

പണം (money) + പെട്ടി (box) or പണ (field) + പെട്ടി (box)

In the above case there are two ways to split the compound word. This creates multiple translations only one of which will be true in a particular context. More context information is required to find the correct translation.

b) multiple parse tree

Malayalam: മോഹ ൻ കാട് ഇൽ സുഖമായി കഴിഞ്ഞു

Morpheme sequence: മോഹ ൻ കാട് ഇൽ സുഖം ആയി കഴിഞ്ഞു

Translated output: i) Mohan happily lived/finished in forest

ii) Mohan became in forest happy

The word ആയി has two tags. An adverb suffix(ADVA) and a verb(V). The parser creates two parse trees in each of the two cases. ആയി is treated as an adverbal suffix in the first sentence and a verb in the second sentence.

8.5 Sample Outputs from the translator

Sample outputs generated by the translator for the input story given below are shown in Table 8.1. A screenshot from the prototype translator is shown in Figure 8.3.

Input story in Malayalam: ഒരു നല്ലവനായ രാജാവ് ഒരിടത്തൊരിടത്ത് ഉണ്ടായിരുന്നു. രാജാവിന്റെ മകൾ അതിസുന്ദരി ആയിരുന്നു. ആ രാജകുമാരിക്ക് സൂര്യനെപ്പോലെ തിളങ്ങുന്ന ഒരു സ്വർണ്ണപന്തുണ്ടായിരുന്നു. രാജകുമാരിക്ക് പൂന്തോട്ടത്തിൽ പന്തു കളിക്കാന് ഇഷ്ടമായിരുന്നു. രാജകുമാരി ഒരു ദിവസം കളിച്ചു കൊണ്ടിരുന്നപ്പോ ൾ സ്വർണ്ണപന്ത് അടുത്തുള്ള കിണറ്റിൽ വിണു. രാജകുമാരി കരയാൻതുടങ്ങി. കിണർന്വക്കിലിരുന്ന ഒരു തവള ഇതെല്ലാം കാണുന്നുണ്ടായിരുന്നു. തവള രാജകുമാരിയുടെ അടുത്തേക്ക് ചാടിച്ചെന്നു. തവള പന്ത് എടുത്തുതരാം എന്ന് തവള രാജകുമാരിയോട് പറഞ്ഞു. രാജകുമാരി പകരം എന്ത് തരും എന്ന് തവള രാജകുമാരിയോട് ചോദിച്ചു. രാജകുമാരി നൂറ് സ്വർണ്ണനാണയങ്ങ ൾ കൊടുക്കാം എന്ന് തവളയോട് പറഞ്ഞു. എനിക്ക് രാജകുമാരിയുടെ കൂടെ എപ്പോഴും ഇരിക്കണം. രാജകുമാരിക്കു തവളയുടെ ആവശ്യം ഇഷ്ടപ്പെട്ടില്ല. രാജകുമാരി എങ്കിലും തവളയെ കൂടെനടക്കാൻ സമ്മതിച്ചു. പെട്ടെന്നു തവള കിണറ്റിലേക്ക് ചാടി. തവള പന്ത് കൊടുത്തപ്പോൾ രാജകുമാരിയുടെ ഭാവം മാറി. രാജകുമാരി തവളയെ കൂടെ കൊണ്ടുപോകാ ൻ തയ്യാറായില്ല. രാജകുമാരി രാത്രിയിൽ ആഹാരം കഴിക്കാന് ഇരുന്നപ്പോ ൾ തവള കതകില് തട്ടി. രാജകുമാരി അച്ഛനോട് തവളയെപ്പറ്റി പറഞ്ഞു. അവൾ തവളയുടെകൂടെ ഇരിക്കണം എന്ന് രാജാവ് രാജകുമാരിയോട് പറഞ്ഞു. രാജകുമാരി തവളയെ അകത്ത് വരാൻ സമ്മതിച്ചു. അങ്ങനെ തവള രാജാവിന്റെയും രാജകുമാരിയുടെയും കൂടെയിരുന്ന് കഴിച്ചു. രാജകുമാരി രാത്രി ഉറങ്ങാൻ കിടന്നപ്പോ ൾ തവളയും കൂടെകിടന്നു. രാജകുമാരി എന്നെ പതുക്കെ തടവണം എന്ന് തവള രാജകുമാരിയോട് പറഞ്ഞു. രാജകുമാരി തവളയെ കൈയില് എടുത്തു തലോടിയപ്പോൾ തവള സുന്ദരനായ ഒരു രാജകുമാരനായിത്തീർന്നു. തവള അടുത്ത രാജ്യത്തെ രാജകുമാരനായിരുന്നു. ഒരു മന്ത്രവാദി രാജകുമാരനെ ശപിച്ചതു കൊണ്ട് രാജകുമാരൻ ഒരു തവളയായിത്തീർന്നു. രാജകുമാരി തൊട്ടപ്പോള് തവളക്ക് പഴയ രൂപം തിരിച്ചു കിട്ടി. രൂപം തിരിച്ചുകിട്ടിയ രാജകുമാരൻ രാജകുമാരിയെ വിവാഹം കഴിച്ചു.

Table 8.1 Sample outputs from the prototype translator

System Input and Output	Remarks
1. Input: ഒരു നല്ലവനായ രാജാവ് ഒരിടത്തൊരിടത്ത് ഉണ്ടായിരുന്നു. **Word to word translation:** a kind king in a place was. **English version of the sentence:** There was a kind king in a place. **System Output:** A kind king was in a place.	System output is in line with the English version except for the positioning of the article .Meaningfully correct sentence
2.Input:. രാജാവിന്റെ മകൾ അതിസുന്ദരി ആയിരുന്നു. **Word to word translation:** King's daughter very beautiful was **English version:** The king's daughter was very beautiful. **System Output:** King's daughter was very beautiful.	System output is exactly in line with the English version except for the positioning of article to the noun king since same word is not there in input language. Meaningful correct translation
3.Input: ആ രാജകുമാരിക്ക് സൂര്യനെപ്പോലെ തിളങ്ങുന്ന ഒരു സ്വർണ്ണപന്തുണ്ടായിരുന്നു. **Word to word translation:** That princess sun like shining is a goldenball. English **version** : The princess had a golden ball which was shining like the sun. **System Output:** That princess had a ball which is shining like sun.	System output gives translation without positioning of article for the nouns. The variations in translations for ആ have not been considered. System output is meaningful correcttranslation
4.Input: രാജകുമാരിക്ക് പൂന്തോട്ടത്തിൽ പന്തു കളിക്കാൻ ഇഷ്ടമായിരുന്നു. **Word to word translation:** Princess garden in ball play to liked **English Version** : The princess liked to play in the garden **System Output:** Princess liked to play in garden.	Output gives translation without positioning of preposition since same word is not there in input language Meaningful correct translation
5.Input: രാജകുമാരി ഒരു ദിവസം കളിച്ചുകൊണ്ടിരുന്നപ്പോൾ സ്വർണ്ണപന്ത് അടുത്തുള്ള കിണറ്റിൽ വീണു. **Word to word translation:** princess a day play was when golden ball nearby well in fell. **English Version** : When the princess was playing one day the golden ball fell into a nearby well. **System Output:** When princess was playing a day golden ball fell in nearby well.	Output translation is without positioning of presposition since same word is not there in input language. Multiple translations of ഒരു has not been considered. Meaningful correct translation
6.Input: രാജകുമാരി കരയാൻ തുടങ്ങി. **Word to word translation** :Princess cry to started **English version** : The princess started to cry **System Output:** Princess started to cry.	System output gives exact English version of input sentence
7.Input: കിണറ്റുവക്കിലിരുന്ന ഒരു തവള ഇതെല്ലാം കാണുന്നുണ്ടായിരുന്നു **Word to word translation:** well side in sat a frog all these see was **English version:** A frog sitting beside the well was seeing all these. **System Output:** frog which was sitting in side of well was seeing all these.	Meaningful correct translation which slightly varies from the English version.
8.Input: തവള രാജകുമാരിയുടെ അടുത്തേക്ക് ചാടി ചെന്നു. **Word to word translation:** frog princess's side jumped went **English version** : The frog jumped to the princess. System Output: frog went jumping to princess's side.	Meaningful correct translation which slightly varies from the English version.

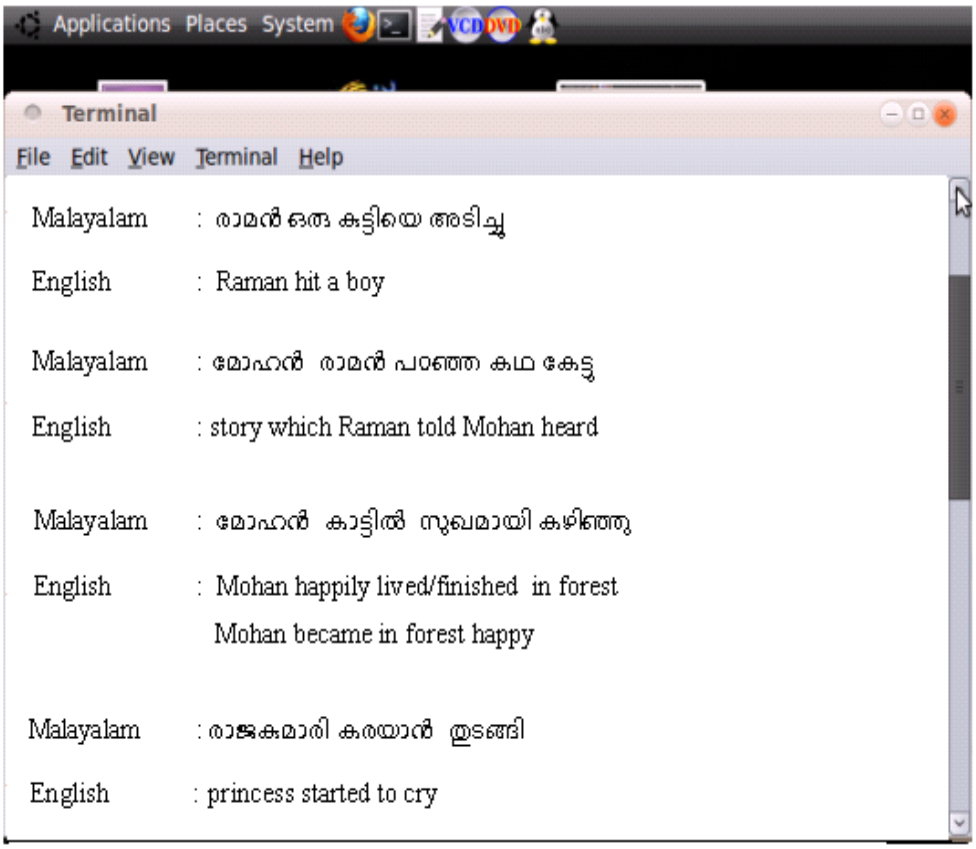


Figure 8.3 Screenshot from translator

8.6 Merits of the prototype translator

1. The translator could handle both simple and complex sentences. Correct translations for complex sentences with two adverb or noun clauses could be generated.

2. The model adopted for the parser makes it possible to extend the parser to handle words of any depth by enhancing the syntax rules.

3. The system design was based on artificial intelligence techniques. The parser developed was a general one and the same could parse sentences in any language if the syntax rule were replaced with that of the new language.

4. The translation systems for other language pairs could easily be generated by replacing the syntax rules and structure transfer rules with those of the new language pair.

5. The modular design helps to reuse the modules for other applications. The analyzer and parser can be integrated into the present day Malayalam spell checkers to enhance their performance. They also can be integrated into Malayalam language understanding systems.

6. All the modules work with morphemes. This helps to reduce the space required by the dictionary.

8.7 Limitations of the MT system

The translator could not find the correct translations of sentences which are not of the syntax rules selected as give in an example shown above. In addition to this the following limitations were identified in the outputs generated by it.

1) Limitations in agreements and surface form generation

Agreements like subject verb agreements or morphological generation for surface forms for plural etc. are missing.

Malayalam: കുട്ടികള്കളിക്കുന്നു

Translated output: Chlild(s) is/are playing

The system was unable to find the correct translation "are playing" and also the correct plural form for the word child as children.

2) Limitations in word sense disambiguation

The parser could perform word sense disambiguation only based on morpheme tags. So the system could not find the correct translation for polysemous words. Multiple meanings for the morpheme category are produced.

i) The system could not distinguish between the two translations for the locative case suffix ഇൽ in the following sentences

a) Malayalam: രമ തറയി ൽഇരുന്നു (rama thaRayil irunnu)

Translated output: Rama sat on/to the floor

b) Malayalam: രാജു ഡെല്ഹിയി ൽ പോയി (raaju delhiyil pOyi)

Translated output: Raju went on/to Delhi

ii) In the following set of sentences the word ആയി has to be treated differently which is not done by the system. In sentence a) the ആയി suffix should be translated as 'as' whereas in sentence b) it should be translated as the suffix–ly

which should be attached to the morpheme beautiful forming the adverb beautifully.

a) Malayalam: രാമു മോഹന് കുറച്ചു പണം കടമായി നൽകി

Translated output: Ramu gave some money as/ly loan to Mohan

b)Malayalam: രാമു പടം ഭംഗിയായി വരച്ചു

Translated output: Ramu drew beautiful as/ly the picture

3) Absence of a transliteration module and named entity recognizer

The system does not possess a named entity recognizer and a transliteration module. Their addition would make handling of out of vocabulary words in an efficient way.

8.8 Conclusion

This chapter discussed the analysis of the outputs generated by the system to find the various categories of outputs generated. This analysis helped to modify the rules used by each module and the dictionary content design for improving the performance of the system. The analysis was conducted in three levels: the analyzer level, parser level and the MT system as a whole. Even for sentences with more than two subordinate clauses the system returned meaningful translations. All the modules generated multiple outputs which reduces the accuracy of the system. The prototype system also produced multiple translation results for some inputs. They were due to the deficiency in the analyzer module or the parser module. Only word sense disambiguation based on lexical category of the words in the sentence could be performed by the parser. The system also produced error outputs in some cases due to the

difference in syntax. The system also lacks in considering the agreements between chunks in the sentence. More rules can be added to make the system to give exact translation of input sentences in all cases. Additional modules like finding and replacing collocations, finding and replacing named entities can also be added to the basic translator. The next chapter discusses the performance evaluation of the prototype system.

Chapter 9

PERFORMANCE EVALUATION

9.1 Introduction

This chapter discusses the evaluation results of the developed prototype system. First the storage space requirement for the dictionary is presented to show the saving in storage space achieved by storing morphemes other than the inflected forms for words. The evaluation of each module and also the prototype as a whole are discussed. Results of the comparative study of the performance of other existing systems which were developed for Indian language translation are also discussed.

9.2 Analysis of space requirement for dictionary

The space efficiency achieved by a morpheme based dictionary can be shown as follows. Only noun and verb inflections are considered for the present calculation. Also it is assumed that a noun or a verb can have only one suffix. In fact Malayalam is highly agglutinative and any number of morphemes can join to form a word. In those cases the space saving achieved will be much higher compared to the one shown here.

Number of nouns in the dictionary - N
Number of verbs in the dictionary - V
Total space required for the nouns - Sn bytes
Total space required for the verbs - Sv bytes
Total space required by noun suffixes - Ns bytes
Total space required by verb suffixes - Nv bytes
Total number of noun suffixes - n
Total number of verb suffixes - m

Space Saving = Space required with inflected words - Space required with morphemes (Sn*n+Sv*m+Ns*N+Nv*V) – (Sn+Sv+Ns+Nv)

The sample dictionary used consists of:

Total 1500 words

N 775

V 128

Sn 13298 bytes

Sv 2735 bytes

Ns 34 bytes

Nv 96 bytes

n 17 (case and number suffixes)

m 12 (Tense, aspect and mood suffixes)

Saving in storage space =(13298*17+2735*12+34*775+96*128) – (13298+2735+34+96) =281361 bytes

i.e. saving% = 94.5%

In this study it was assumed that one verb or one noun can have only one suffix at a time. But in actual case in Malayalam any number of morphemes can join to form a word. In that case the saving in space will be much higher.

9.3 Test data set

MT evaluations typically include evaluation of the quality of the unedited translations, which includes intelligibility, accuracy, fidelity, appropriateness of style, the usability of facilities for creating and updating dictionaries, the

extendibility to new language pairs and/or new subject domains and cost-benefit comparisons with human translation performance. The performance evaluation of the prototype translation system was conducted during various stages of its development. The performance of each of the phases was separately carried out. The initial evaluation was carried out using test suites which was a collection of artificially constructed inputs, where each input was designed to probe a system's treatment of a specific phenomenon or set of phenomena. Inputs were in the form of sentences, sentence fragments, or even sequences of sentences. The feedback from the tests was used to modify the system for better performance. The final evaluation for each phase was carried out using a test corpora which was a collection of naturally occurring text from many domains. The test data was selected from randomly selected news (articles sports, politics, world, regional, entertainment, travel etc., literature text and children stories. Care had been taken to ensure that sentences used a variety of constructs. All possible constructs including simple as well as complex ones were incorporated in the set. The sentence set also contained all types of sentences such as declarative, interrogative, imperative and exclamatory. Sentence length was not restricted although care had been taken that single sentences were not too long. The test data set is shown in Table 9.1.

Table 9.1 Test data set for evaluation of prototype system modules and whole MT system

	Children stories	**News articles**	**Literature**
Number of documents	30	20	10
Number of sentences	1542	423	2021
Number of words	15431	2811	14356

The performance evaluation of the system was conducted in three levels: In the analyzer level, parser level and the MT system as a whole. Out of the different metrics for performance evaluation we had chosen accuracy test and error analysis for the MT system and the different modules. Accuracy Test was conducted to measure how much information the translated sentence retained compared to the original. In error analysis mainly word error rate was calculated.

9.4 Performance evaluation of morphological analyser

The outputs from the morphological analysis was analysed and the accuracy of the module was calculated. The performance of the analyser for the three kinds of texts is shown in Table 9.2 and Figure 9.1. The compound words which fall into the first category of correct unique split and the number of constituents in a compound word were found to be more in news articles than children stories. The analyzer is found to be working with 90.7% accuracy.

Table 9.2 Performance of morphological analyser

Type of output	Children stories	News articles	Literature
Correct unique split	90%	94%	88%
Correct multiple split	8%	4%	8%
Multiple splits with incorrect split	2%	2%	4%

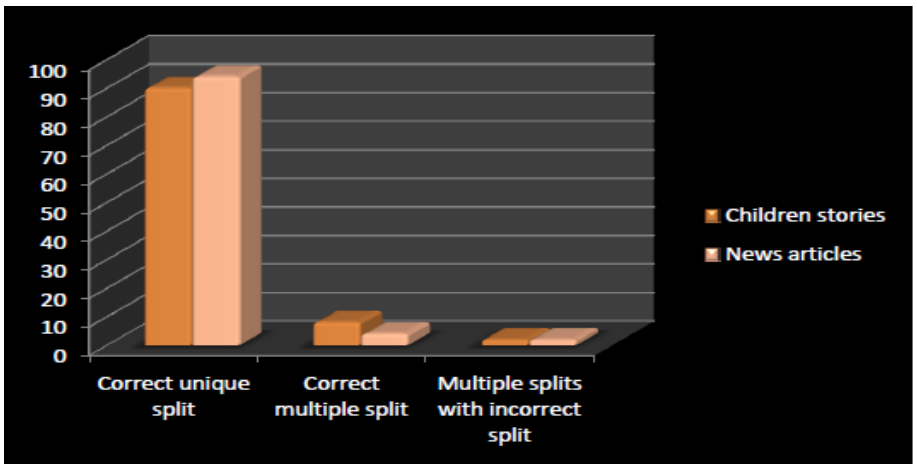


Figure 9.1: Performance of the morphological analyzer

9.5 Performance evaluation of parser module

The parser was tested with test suits initially for arriving at the correct syntax of sentences and also for identifying the correct chunks for reordering. Finally the module was tested with a test corpus from three domains children stories, news articles and literature texts. Two kinds of evaluation were done with parser output. Based on the type of output sentences produced and based on the chunks it generated. The precision and recall percentages were used for evaluation based on chunks.

9.5.1 Performance evaluation on a sentence level

The results are tabulated in Table 9.3 and shown in Figure 9.2. The analysis of the performance showed that the parser works well for children stories as the syntax rules derived closely matches with that of children stories.

Table 9.3 Performance of parser module

Sl. No.		Children stories	News	Literature
1.	Unique correct parse	85	78	70
2.	Wrong parse	5	12	15
3.	Multiple parses which includes correct parse	10	10	15

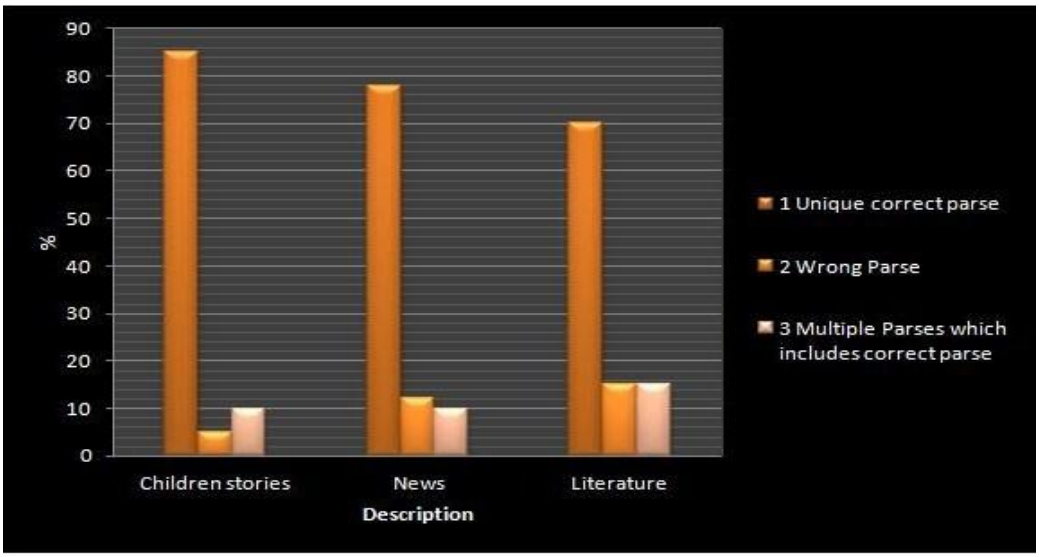


Figure 9.2 Performance of parser module

9.5.2 Performance evaluation on a chunk level

The precision for the parser was based on the kind of chunks it produced. Only the F-measures for noun chunks, verb chunks and clause chunks are shown in Table 9.4. Similar calculations were done for other kinds of chunks. Average precision for the chunking process was found to be 91.2% and the recall was found to be 89.3%.

$$\text{Precision} = \frac{\text{Number of correct chunks in output}}{\text{Total number of generated chunks}}$$

$$\text{Recall} = \frac{\text{Number of correct chunks in output}}{\text{Total number of chunks in the correct translation.}}$$

The precision and recall were used to calculate the F- measure.

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Table 9.4 Precision and Recall from the parser module

Chunk type	Precision	Recall	F-measure
Noun chunk	95%	96%	95.4
Verb chunk	93%	94%	93.49
Clauses	90%	93%	91.47

9.6 performance evaluation of the prototype MT system

Out of the different metrics for performance evaluation we had chosen accuracy test and error analysis for the MT system. Accuracy Test was conducted to measure how much information the translated sentence retained compared to the original. In error analysis mainly word error rate was calculated. A study was also conducted to compare the performance of our system with that of four other translation systems for Indian languages.

9.6.1 Accuracy testing

The performance of the prototype system for various types of documents is shown in Table 9.5 and Figure 9.3. A careful analysis of the table shows that the system is most suited for translation of children stories. This is because word compounding is less in children stories and the syntax structure closely

matches with that of children stories. The system gives an average accuracy rate of 65%.

Table 9.5 Performance of prototype system

Type of output	Children stories	News articles	Literature Text
1. Incorrect translation due to syntax limitation	25%	50%	30%
2. Correct unique translation	55%	30%	40%
3. Multiple translation with correct translation	20%	20%	30%
a) multiple parses	8%	5%	15%
b) limitations in word sense disambiguation	8%	15%	15%

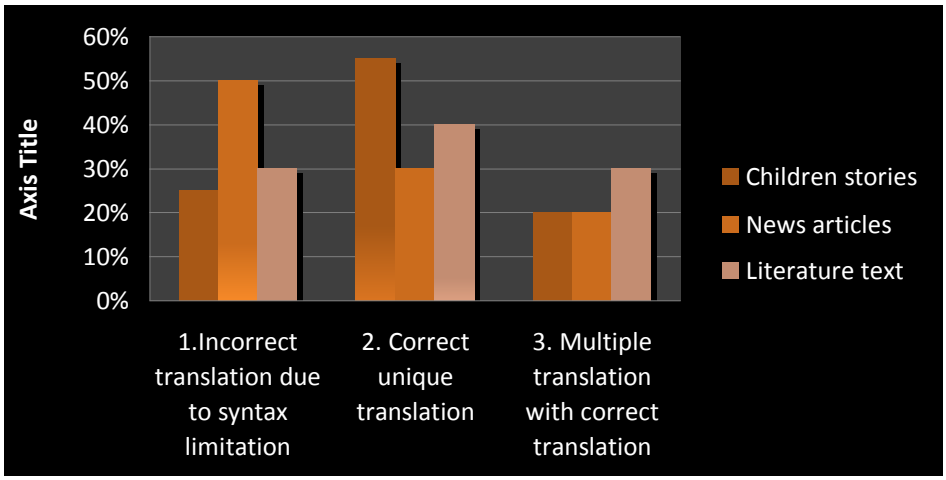


Figure. 9.3 Graph depicting performance of prototype MT system

9.6.2 Comparison with existing systems

The performance of the system had been compared with that of some of the existing MTsystems for Indian languages which use the same approach as that of the developed prototype system. Three MT systems had been taken for the comparative study. The results are tabulated in Table 9.6.

Table 9.6 Comparison with existing systems

MT system	Structure Transfer approach	Sentence Types	Accuracy of chunking	Accuracy of translation
Mantra	LTAG parser	Office administration	Not available	93%
Punjabi-English	Hand crafted chunk rules	Simple sentences in Legal documents	86.57%	70.54%
English-Malayalam [117]	Stanford parser	Simple sentences with max. of 6 words	Not available	53.63%
Malayalam-English	Top down parser	Complex sentences Children stories	85%	75%

A detailed Comparison of the prototype system developed with Punjabi to English machine translation system had been conducted since both use transfer

based approach as the basic approach for translation. Both the systems use parsing, chunking and structure transfer for the generation of translated sentences[116].

The English to Punjabi translation system developed by Punjabi University [116] is based on the transfer approach, with three main components: an analyzer, a transfer component, and a generation component. The analysis component is composed of preprocessing, morph analyzer and tagger, phrase chunker and the module which adds semantic information, assigns morph information and karaka roles to the phrases by means of Punjabi grammatical rules. The target language translation is produced by means of a combination of synthesis and post processing modules. It also includes a transliteration module. The morph database used for the system includes words in Punjabi along with the information about its gender, number, person, case, tense etc. Every inflected word also contains the root word from where it is derived. The database also contains the grammatical category of each word and also the inflected words it can form. Only noun phrases, adjective phrases, postpositional phrases and verb phrases have been considered in the system. The system can handle only simple sentences. Reported translation accuracy is 60.33%, Tagging accuracy is 75.54% and chunker accuracy is 86.57%.

The Malayalam to English translation system developed contains only base morphemes, the lexical category and the translation. The system uses only one bilingual dictionary which is shared by morphological analyser, parser and generation component. The dictionary presently contains only a small set of commonly found words in Malayalam. The dictionary size required is small compared to the former one as it contains only the root words. The system uses the top down parser to do word sense disambiguation based on lexical category,

chunking and structure transfer. The system can handle simple and complex sentences with any number of adjective clauses. The system can recognise more chunks compared to the previous one. A transliteration module is absent in the present system and it is assumed that all words including proper nouns are present in the dictionary. The present prototype system takes care of complex sentences with utmost two adverb clauses or noun clauses. Chunking accuracy of 93% has been obtained and the translation accuracy including simple and complex sentences was found to be 75%.

9.6.3 Analysis of word Error rate

The Word error rate for different types of input is shown in Figure 9.4. It is found that word error rate is more for news articles compared to other types of documents. After robust analysis, Word Error rate was found to be 8% which was comparably lower than that of general systems, where it ranges from 9.5 to 12. Figure 9.5 shows the percentage type of errors out of the word errors found. The addition or omission of words like omission of articles was the most frequently occurring word error.

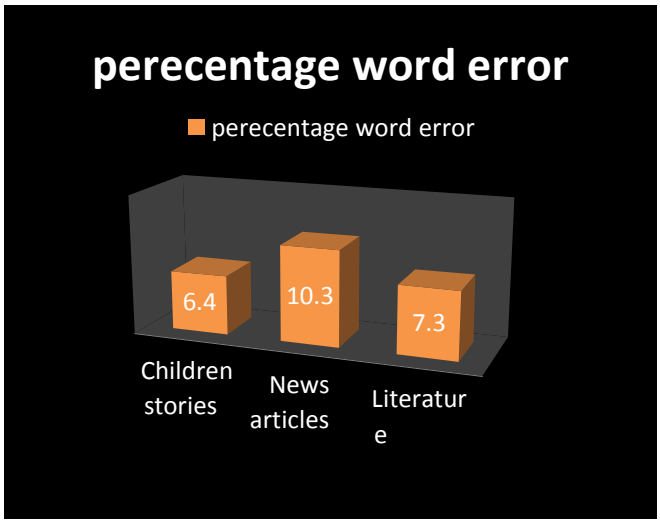


Figure 9.4 Word error rate for various texts

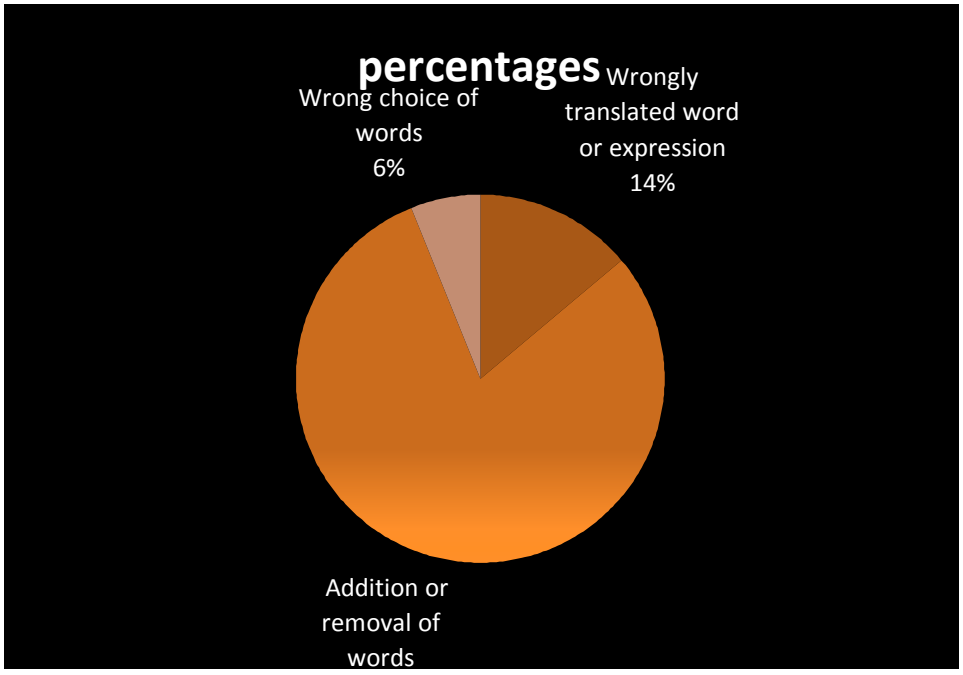


Figure 9.5 Word error rate

9.7 Conclusion

The chapter discussed the performance evaluation of the prototype translation system. The performance of the morphological analyzer, the parser and the prototype system as a whole were evaluated separately using texts belonging to different domains. The MT system was evaluated based on accuracy rate, word error rate. The average accuracy rate for the system was found to be 65% and 75% for children stories. The results of the comparison of performance of our prototype with other existing translation systems also were discussed in the chapter.

Chapter 10

EPILOGUE

10.1 Introduction

Work in the area of Machine Translation in India has been going on for several decades. During the early nineties, advanced research in the field of Artificial Intelligence and Computational Linguistics made a promising development of translation technology. This helped in the development of usable Machine Translation Systems in certain well-defined domains. It is extremely difficult to build a fully automatic high quality machine translation system(FGH_MT). In fact there is no system in the world which qualifies to be called FGH_MT. Many institutions like IIT(Kanpur), CDAC(Mumbai), CDAC(Pune), IIIT(Hyderabad), etc. are engaged in development of MT systems under projects sponsored by Department of Electronics, state governments etc. since 1990. Research on MT systems between various Indian and foreign languages and also between Indian languages are going on in these institutions. Translation between structurally similar languages like Hindi and Punjabi is easier than that between language pairs that have wide structural difference like Hindi and English. Having many parts of their grammars and vocabularies in common, it is easy to develop translation systems between closely related languages.

Development of Machine translation (MT) system requires very close collaboration among linguists, professional translators and computer engineers. In the development process, there are two major goals; (a) accuracy of translation and (b) speed. Accuracy-wise, smart tools for handling transfer grammar and translation standards including equivalent words, expressions,

phrases and styles in the target language are to be developed. The grammar should be optimized with a view to obtaining a single correct parse and hence a single translated output. Innovative use of corpus analysis, efficient parsing algorithm, design of efficient data structure and run-time frequency-based rearrangement of the grammar are required to reduce the parsing and generation time.

Language translation between Malayalam and other languages can be said to be in very primitive stage now. Turning to Machine translation from Malayalam to English, it can be seen that there has been hardly any work done in the area. This was probably the first of its kind with respect to Malayalam language. It was also observed that machine translation systems developed for other language pairs needs improvements in rule set, dictionary design and translation methodology.

The main objective of this research was to study the issues in Malayalam to English translation and the development of a prototype system for Malayalam to English translation. Major tasks of the research work included: i) Study of the morphology of various lexical categories of Malayalam and English ii) Identification of the part of speech tag set and chunk tag set for word sense disambiguation and structure transfer iii) Derivation of syntactic structure of Malayalam and English sentences using morphology and word level dependencies iv) Determination of the syntactic structure differences between Malayalam and English sentences v) Design of efficient lexicon in terms of speed of retrieval, space requirement and speeding up of the translation process vi) Improving speed and accuracy of the translation process by choosing the best models for the different modules for the prototype translator like morphological analysis, parsing etc.

10.2 Design and development of the prototype translator

As a first step we conducted a survey on the various approaches tried by various researchers for machine translation between two languages including both Indian and non Indian languages. Researchers used different formalisms that were best suited to their applications. They were mainly i) Direct translation approach ii) Rule based approach and iii) Corpus based approach. Direct translation is appropriate for structurally similar languages. Among the rule based approaches transfer based systems based on syntactic structure transfer are more flexible and it can be easily extended to language pairs in a multilingual environment. The interlingua based systems can be used for multilingual translation. Recently, the Universal Networking Language has been proposed as the interlingua by the United Nations University for overcoming the language barrier. Over the past decade data-driven approaches to machine translation have widely been used in language processing research. The relative success in terms of robustness of Example Based and Statistical approaches have given rise to a new optimism and an exploration of other data-driven approaches such as Maximum Entropy language modeling. Performance of statistical techniques can be improved through large parallel corpus and usage of linguistic knowledge in the model. Hybrid systems are found to have better performance compared to the ones with the component technology.

It was found that there were only very few systems developed for south Indian languages. More research has to done in these areas to overcome the language barrier faced by India. The MT systems developed had many shortcomings in terms of rule set, dictionary, translation methodology and it was apparent from the survey that further work is needed in MT as a whole to produce intelligible translations. From among the various approaches the transfer based approach

has been chosen due to its flexibility and extensibility to other language pairs in a multilingual environment.

The following tasks were performed for the design of the various modules of the prototype systems like morphological analyser, parser etc. and for the development of the translator as a whole.

a) Study of the morphology of various lexical categories of Malayalam and English:

The morphological variations for words in Malayalam were studied. Malayalam belongs to the category of highly agglutinative languages in which formation of new words by combining a noun and a noun, noun and adjective, verb and noun, adverb and verb, adjective and noun, and in some cases all the words of an entire sentence to reflect the semantics of the sentence are very common. In addition to this nouns and verbs show inflectional morphology. The nouns are inflected because of gender, number and case. Verbs show inflections due to tense, aspect and mood information. Derivational morphology is also found for verbs and nouns in which other lexical categories like adjectives, adverbs and infinitives etc. are formed from verbs and nouns by adding proper suffixes. The prototype system makes use of most of the commonly found rules for inflection, derivation and word compounding for the purpose of reducing the dictionary space used.

b) Identification of the part of speech tag set and chunk tag set for word sense disambiguation and structure transfer.

POS tagging is an important activity for investigators of natural language processing, speech recognition and other related areas. It proves to be a basic building block for constructing statistical models which needs an annotated corpora for automatic processing of natural languages. The words in a sentence

occur as groups called chunks. In problems like machine translation or language understanding these chunks play a very important role. We used IIIT tag set as a benchmark. A set of primitive morpheme tags and chunk tags required for the translator were derived.

c) Derivation of syntactic structure of Malayalam and English sentences using morphology and word level dependencies.

According to Universal clause structure grammar clauses in a sentence can be nested one inside the other, resulting in a hierarchical or tree like structure. Verb groups and sentinels contain all the required information for recognizing clauses, for determining the nested or hierarchical structure of clauses and for determining the clause boundaries. It is seen that every clause in a sentence except for the main clause has a sentinel which marks one of the boundaries of that clause. A careful analysis of the different sentence classes of Malayalam and English was required for deriving the hierarchical dependency rules and also the syntactic structure transfer rules of the prototype translator. Malayalam is an S-O-V language. The default or unmarked order of constituents is Subject first, then the Object and finally the verb. However, Malayalam, being a relatively free word order language, permits substantial amount of freedom in the order of constituents although normally the verb remains in the sentence final position. Word order becomes less important mainly because noun groups are marked for cases and the verb agrees with the subject in gender, number and person. In fact, subjects and objects are often dropped. Normally all modifiers precede the modified. There are a variety of subordinate clauses. Subordinate clauses also precede the main clause. They typically involve special non-finite forms of verbs which occur invariably in the clause final position and mark the

right hand boundary of the respective clauses. All these assertions were taken as rules.

d) Determination of the syntactic structure differences between Malayalam and English sentences.

13 rules were identified and implemented in the prototype system. These rules cover most of the commonly found structure difference between Malayalam and English.

e) Selection of a suitable model for the translator.

The survey on the various approaches for machine translation revealed that a direct translation approach and corpus based approaches will be inefficient in the case of Malayalam to English translation system. So we have decided to use a syntax transfer based approach using synchronous tree adjoining grammar (STAG). Translation with Synchronous TAGs is a three step process:

1. Derivation tree for source sentence is obtained by parsing the source sentence.

2. Source derivation tree is converted into target derivation tree or trees.

3. The target sentence is obtained by listing the leaf nodes in the target parse tree in a depth first order.

The system modules selected are like that of SAMPARK system developed by IIIT Hyderabad. The syntax structure formalism used is similar to the one used in UCSG based Kannada translator by Murthy. The parsing method is similar to the one in MANTRA system developed by CDAC Mumbai. We had tried to

combine modern artificial intelligence techniques with the classical Paninian framework based on Sanskrit grammar.

There are four main modules for the system: i) Morphological analyzer ii) Morpheme based parser iii) Target sentence generator and iv) Bilingual dictionary.

***i) Morphological analyzer*:** It finds the sequences of morphemes in the input sentence. Since there are more than one way of splitting a word into morphemes it performs a depth first approach to find all possible sequence of morphemes for the given sentence. The morphological analyser takes a valid sentence of the source language as input and it produces a sequence of morphemes in the sentence as output. It was noted that a two level finite automata was not enough for highly agglutinative language like Malayalam. So a recursive depth first search with backtracking algorithm has been utilized to find all the sequences of morphemes for the given sentence. The algorithm uses rules for morphophonemic changes at each morpheme boundary to find the constituent morphemes. Unlike the previously developed morphological analysers for Malayalam, the morphological analyzer of our system finds only the morpheme sequences in the given sentence. A morpheme can have more than one lexical category and the appropriate category for each morpheme is found only during the parsing stage.

ii) ***Morpheme based parser:*** It performs three functions. i) It finds the correct POS tag for each morpheme for word sense disambiguation. ii) It identifies the chunks in the sentence for reordering and creates the source parse tree. iii) It performs the reordering required to meet syntactic requirements of the target

language and to create the target parse tree. The parser model selected was a top down parser based on Synchronous TAG. The model keeps both source and target tree pair, and performs operations simultaneously while traversing through the tree nodes. Thus the syntax structure transfer from Malayalam to English is integrated into the parser. The syntax rules were given in the regular expression form. The longest rules are listed first on the right hand side of each production rule so that longer chunks are recognized first. Regular expression notation helps for a compact representation of language syntax and also it provides easy modification of the syntax rules.

iii) Target sentence generator: The function of the target generation module is to generate the target sentence by performing a depth first traversal of the target parse tree created by the parser. It also performs surface form generation of English words using the morpheme sequences in the tree. It also uses additional set of rules for the generation of target language sentences in the correct form.

iv) Bilingual dictionary: The dictionary is the most essential part of the whole machine translation system. The space required for the dictionary was minimized by storing morphemes. The bilingual dictionary contains the following information:

i) Most of the commonly occurring verbs, nouns, pronouns, adjectives, inflectional and derivational suffixes, clause suffixes etc.

ii) Each entry in the file has four fields: the root word in Malayalam (morpheme), the lexical category of the morpheme, category (human/nonhuman) information (for nouns only) and its translation. A root word can have more than one tag.

iii) The verbs in past tense have their root words stored along with them.

10.3 Performance Evaluation

The performance evaluation was conducted in three levels: In the analyzer level the parser level and the MT system as a whole. The evaluation was conducted with different classes of texts like news articles, stories etc. The analyser generated three categories of output namely unique correct sequence, multiple sequences with incorrect sequence and multiple correct sequences. The parser also generated three categories of output. The evaluation on the prototype system revealed that the system gives correct answers for most of the sentences belonging to the derived syntax. It generated three kinds of translation: unique correct translation, multiple translations with incorrect translations and incorrect translation. The incorrect translation was due to the limitation of the analyser, the parser or the target sentence generator. The parser needs more syntax rules and more semantic level analysis is required for achieving better results.

10.4 Future enhancements

1. Currently the morphological analyser and parser are two separate modules. The morphological analyser can be integrated into parser thus reducing the number of wrong sequences of morphemes generated by the analyser module.
2. The rules for morphology, the lexical categories for words, the syntax rules and the structure transfer rules are to be enhanced for the system to handle texts belonging to all domains.
3.The parser could perform word sense disambiguation based on lexical category only. To handle other types of sense disambiguation more context dependant rules or statistical approaches are to be incorporated into our system.

4.The system has been implemented in such a way that the parser finds all parses valid for the current input and the translator creates the translations corresponding to each of these parses. The selection of the correct translation is left to the user. This has been one of the drawbacks of the system which should be rectified using appropriate techniques. Use of statistical approaches for parsing with a corpus of correctly parsed sentences can reduce multiple parses greatly. Hidden Markov models have been used for the purpose[17]. The number of wrong parses can be eliminated through the use of unambiguous grammar rules. Preprocessing of the morpheme sequence to eliminate wrong tags for a word examining adjacent tags also can reduce the wrong parses greatly. The sentences with blanks to mark the boundaries of semantically related units also help to reduce wrong parses.

5. Also the bilingual dictionary has to be enhanced by including more words.

10.5 Conclusion

The objective of the research work was to conduct a study on the various issues in a syntax based translation method and selection of appropriate models for the various modules for the system and also to develop a prototype Malayalam to English translator based on the selected models. Encouraging results were obtained even with a small set of resources. Weeding out the limitations of the present work will help to shape the prototype into a full fledged system for machine translation. The method adopted was based on artificial intelligence techniques and can easily be extended to build MT systems for other language pairs.

References:

[1] Dubey P., Overcoming the Digital Divide through Machine Translation, Translation Journal, Volume 15, 2011, http://translation journal.net /journal/55mt_india. htm [Dec 12, 2011].

[2] Murthy, B. K., Deshpande W. R., Language technology in India: past, present and future, 1998, http://www.cicc.or.jp/english/hyoujyunka /mlit3/7-12.html.

[3] V Goyal, G S Lehal, Advances in Machine Translation Systems, Language In India, Vol. 9, No. 11, pp. 138-150, 2009.

[4] Alifarghaly, KhaledShaalan, Arabic Natural Language Processing: Challenges and Solutions, ACM Transactions on Asian Language Information Processing, Vol. 8, No. 4, Article 14, 2009.

[5] G.S.Josan , G.S. Lehal, Evaluation of Hindi to Punjabi Machine Translation System, International Journal of Computer Science Issues, vol4 no1, pp. 243-257, 2009.

[6] Goyal V., Lehal G.S., Web Based Hindi to Punjabi Machine Translation System, Journal of Emerging Technologies in Web Intelligence, Vol.2, pp.148-151, 2010.

[7] Bharati A., Chaitanya V., Kulkarni A.P., and Sangal R., Anusaaraka: Machine Translation in Stages, Vivek: A Quarterly in Artificial Intelligence, Vol. 10, No.3, pp. 22-25, 1997.

[8] GoutamKumar S., The EB-Anubad translator: A hybrid scheme, Journal of Zhejiang University SCIENCE 2005, ISSN 1009-3095, pp.1047-1050.

[9] Bandyopadhyay S., ANUBAAD - The T ranslator from English to Indian Languages, VIIth State Science and Technology Congress, Calcutta, India, 2000.

[10] Lalithadevi S., Pralayankar P and Kavitha V., Case Marking Pattern from Hindi to Tamil MT, 3rd National Conference on Recent Advances and Future Trends in IT (RAFIT),Punjabi University, Patiala, Punjab, 2009.

[11] Lalitha Devi S., Pralayankar P., et. al., Verb Transfer in a Tamil to Hindi Machine Translation System, International Conference of Asian Language Processing, Harbin, China, pp.261-264, 2010.

[12] Bandyopadhyay S., State and Role of Machine Translation in India. Machine Translation Review, pp.11: 25, 2000.

[13] Bandyopadhyay S., Use of Machine Translation in India-current status. AAMT Journal, 36, pp. 25-31, 2004.

[14] Bandyopadhyay S., Teaching MT: an Indian perspective, Sixth EAMT Workshop, UMIST, Manchester, England, pp.13-22, 2002.

[15] Dwivedi S.K. et. al., Machine translation systems in Indian perspective, Journal of computer science, pp.1082-1087, 2010.

[16] Gary Anthes, Automated translation of Indian languages, communications of ACM, vol.53, no,1, pp.24-26, January 2010.

[17] Kumar G.B., Murthy K.N., UCSG Shallow Parser, Proceedings of CICLing-2006, Lecture Notes in Computer Science, Volume 3878 , pp 156-167, Springer-Verlag.

[18] Joanna Ruth, Jimmy O'Regan, Shallow-transfer rule-based machine translation from Czech to Polish (apertium), Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation, p.p. 69-76, January 2011.

[19] Marote R. C, Guillen E et.al.., The Spanish-Catalan machine translation system interNOSTRM, In proceedings of MT Summit VIII, pp.18-22, 2001.

[20] M. Corbí-Bellot, Mikel L. et.al., An open-source shallow-transfer machine translation engine for the Romance languages of Spain., In Proceedings of the Tenth Conference of the European Association for Machine Translation, Hungary, pp. 79-86, 2005.

[21] Scannell K.P., Machine Translation for Closely Related language Pair, Proceedings of the Workshop on Strategies for developing machine translation for minority languages, pp103-107, 2006.

[22] Dave S., Parikh J. and Bhattacharyya P., Interlingua Based English Hindi Machine Translation and Language Divergence, Machine Translation, Volume 16, pp. 251-304, 2002.

[23] R.M.K. Sinha, A hybridized EBMT system for Hindi to English Translation, CSI Journal, volume 37, no. 4, pp.3-9, 2007.

[24] R.M.K. Sinha, et. al., ANGLABHARTI: A Multi-lingual Machine Aided Translation Project on Translation from English to Hindi, 1995 IEEE International Conference on Systems, Man and Cybernetics, Canada, pp.1609-1614, 1995.

[25] Sinha R.M.K, Jain A., AnglaHindi: an English to Hindi machine-aided translation system, MT Summit IX, New Orleans, USA, 23-27, pp.494-497, 2003.

[26] Ahsan A., Kolachina P., et. al., Coupling Statistical Machine Translation with Rule-based Transfer and Generation, AMTA- The Ninth Conference of the Association for Machine Translation in the Americas. Denver, Colorado,.2010.

[27] Sebastian M.P., Kurien S.K., Kumar G.S., English to Malayalam translation – a statistical approach, A2CWiC-10, First Amrita ACM – W celebration in women in Computing in India, article 64, pp.1-5, 2010.

[28] Gupta D., N. Chatterjee., Identification of Divergence for English to Hindi EBMT , Proceedings of MT SUMMIT IX, p.p.141-148, 2003.

[29] Sinha R.M.K, An Engineering Perspective of Machine Translation: AnglaBharti-II and AnuBharti-II Architectures, Proceedings of International Symposium on Machine Translation, NLP and Translation Support System (iSTRANS- 2004), November 17-19, 2004.

[30] JR.M.K. Sinha, Designing Multi-lingual Machine- Translation System: Some Perspectives, International Conference on Machine Learning: Models, Technologies &Applications (MLMTA 2007), pp. 244-249, 2007.

[31] Vijayanand K., Choudhury S.I., Ratna P., VAASAANUBAADA - Automatic Machine Translation of Bilingual Bengali-Assamese News Texts, Language Engineering Conference, Hyderabad, India, pp.183-188, 2002.

[32] Sriram Venkatapathy, Srinivas Bangalore, Discriminative Machine Translation Using Global Lexical selection, ACM Transactions on Asian Language Information Processing, Vol. 8, No. 2, Article 8, May 2009.

[33] Alon Lavie, Stephan Vogel, et.al., Experiments with a Hindi-to-English Transfer-Based MT System Under a Miserly Data Scenario, ACM Transactions on Asian Language Processing, Vol. 2, No. 2, p.p.143–163, 2003.

[34] Wajdi Zaghouani, RENAR: A Rule-Based Arabic Named Entity Recognition System University of Pennsylvania, ACM Transactions on Asian Language Information Processing, Vol. 11, No. 1, Article 2, 2012.

[35] Chung-Hsien Wu and Hung-Yu Su , Transfer-Based Statistical Translation of Taiwanese Sign Language Using PCFG, ACM Transactions on Asian Language Information Processing, Vol. 6, No. 1,Article 1, April 2007.

[36] Bonnie Dorr and David Zajic, Cross-Language Headline Generation for Hindi, ACM Transactions on Asian Language Information Processing, Vol. 2, No. 3, p.p. 270–289, 2003.

[37] Jurafsky D., Martin H.M., "Speech and natural language processing", Prentice Hall, 2003.

[38] Sumam Mary Idicula, Peter S. David, A morphological processor for Malayalam language, South Asia Research, vol.27 (2), p.p.173-186, 2007.

[39] Varma A.R.R.R, Kerala Paniniyam, D.C Books, Kerala, India, 2000.

[40] E.V.N.Namboothiri, VakyaGhatana, Kerala bhasha institute, third edition, 1997.

[41] Prof. Gopikkuttan, MalayalaVyakaranam, Current Books, Kottayam, 2008, ISBN 81_240_1093_5.

[42] Frances Karttunen, Romald M Kaplan, et. al., Two-level morphology with composition, Coling-92, France, pp.141-148, 1992.

[43] Karttunen, Constructing lexical transducers, Proceedings of the 15th International Conference on Computational Linguistics, Coling94, pp.406-411, 1994.

[44] E. Antworth, PC-KIMMO: a two level processor for morphological analysis, No. 16, Occasional publications in academic computing, Dallas: Summer institute of Linguistics, 1990.

[45] John Bear, A morphological recognizer with syntactic and phonological rules, COLING86, Proceedings of 11th congress on computational linguistics, p.p.272-276, 1986.

[46] Gilbert Krube, Two level representation for natural language, Research on Language and Social Interaction, volume 19, issue 2, p.p.205-286, 1986.

[47] Koskenniemi K., Two level model for morphological analysis, IJCAI-83, p.p.683-685, 1983.

[48] Aduriz I, Agirre E, et. al., A word grammar based morphological analyzer for agglutinative languages, COLING '00, Proceedings of the 18th conference on Computational linguistics, Volume 1, p.p.1-7.

[49] Gabor Proszeky, et. al., A unification based approach to morpho syntactic parsing of agglutinative and other highly inflectional languages, ACL99, Computational linguistics, p.p. 261-268, 1999.

[50] Gabor Proszeky, Morphological analyzer as syntactic parser, COLING 96, Proceedings of 16th conference on computational linguistics, Volume 2, 1996.

[51] Shambhavi B.R, Ramakanth Kumar P, Kannada Morphological Analyser and Generator using Trie, International Journal of Computer Science and Network Security, vol.11, no.1, 2011.

[52] RamasamyVeerappan, Antony P J, S Saravanan and Soman K P, A Rule based Kannada Morphological Analyzer and Generator using Finite State Transducer, International Journal of Computer Applications, 27(10):45-52, 2011.

[53] J. Hankamer, Finite state morphology and left to right phonology, Proceedings of the fifth west coast conference on formal linguistics, Stanford, CA, p.p.29-34., 1986.

[54] GirinathJha, MuktanandAgrawal, et. al., Inflectional morphology analyzer for Sanskrit, Sanskrit computational linguistics, volume 5402, p.p. 219-238, 2009

[55] Kemal Oflazer, Lenient morphological analysis, Natural language engineering 9(1): pp. 87-99, 2003.

[56] Akshar Bharti, Rajeev Sangal, et. al., Unsupervised improvement of morphological analyser for inflectionally rich languages, Proceedings of NLPRS-2001, Tokyo, pp. 27-30, Nov 2001.

[57] Utpal Sarma, Jugal Kalita, et. al., Unsupervised learning of morphology for building lexicon for a highly inflectional language, SIGPHON, Association of Computational linguistics, Philadelphia, p.p.1-10, July 2002.

[58] Creutz M., Lagus K., Unsupervised discovery of morphemes, Proceedings of workshop on Morphological and Phonological Learning of ACL 2002, p.p. 21-30, 2002.

[59] Jual K. Kalita, Utpal Sharma, et. al., Acquisition of Morphology of an Indic Language from Text Corpus, ACM Transactions on Asian Language Information Processing, Vol. 7, No. 3, Article 9, p.p.1-33, 2009.

[60] Honglan Jin, Kam-Fai Wong, A Chinese Dictionary Construction Algorithm for Information Retrieval, ACM Transactions on Asian Language Information Processing, Vol. 1, No. 4, p.p. 281-296, 2002.

[61] Deniz Yuret, Ergun Bicici, Modelling morphologically rich languages using split words and unstructured dependencies, Proceedings of the ACL- IJCNLP2009, p.p.345-348, Singapore, 2009.

[62] Masaki Murata, Qing Ma, et. al., Comparison of Three Machine-Learning Methods for Thai Part-of-Speech Tagging, ACM Transactions on Asian Language Information Processing, Vol. 1, No. 2, p.p.145-158, 2002.

[63] Rajeev R.R., Rajendran N, et. al., A suffix stripping Based morph analyzer for Malayalam Language, Science congress, 2007.

[64] Ganapathiraju, Madhavi and Lori Levin, TelMore: Morphological Generator for Telugu Nouns and Verbs, Proceedings of Second International Conference on Universal Digital Library, p.p.17-19, 2006.

[65] Dasgupta, Sajib and Vincent, Unsupervised morphological parsing of Bengali, Language Resources and Evaluation, Netherlands: Springer, Volume 40, Number 3-4, p.p. 311-330, 2006.

[66] S.Viswanathan, S. Ramesh Kumar, et. al., A Tamil Morphological Analyser, ICON, p.p. 31-39, 2003.

[67] Shaalan, Khaledet. al., Arabic morphological generation from Interlingua: A rule-based approach, Intelligent Information Processing III, p.p. 441-451, 2006.

[68] P.P.Ray, Harish, V.S.Sarkar and A.Basu, Part of speech tagging and Local word grouping techniques for natural language parsing in Hindi, Proceedings of international conference on natural language processing (ICON 2003).

[69] S. LakshmanaPandian, T.V.Geetha, Morpheme based Language Model for Tamil Part of Speech Tagging, Polibits, 38, p.p.19-26, 2008.

[70] Dipanjan Das, Monojit Choudhury, Sudeshna Sarkar and Anupam Basu, An affinity based Greedy approach towards Chunking for Indian languages, Proceedings of ICON-2005, p.p.55-62.

[71] GuohongFu,Chunyu Kit and Jonathan J. Webster, Chinese word segmentation as morpheme based lexical chunking, Sciences Volume 178 , Issue 9, p.p. 2282-2296, 2008.

[72] Noam Chomsky, Logical syntax and semantics: their linguistic relevance, vol.31, No.1, p.p.36-45, 1955.

[73] Noam Chomsky, Aspects on the theory of syntax, p.p.4-5, 1965.

[74] Murthy K.N, UCSG and the syntax of relatively free word order languages, South Asian Language Review VII, 1997.

[75] Aravind K. Joshi, L. Levy and M. Takahashi,Tree Adjunct Grammars, Journal of Computer and System Sciences, volume10, issue1, p.p.136-163, 1975.

[76] K. Narayana Murthy, A. Sivasankara Reddy, Universal Clause Structure Grammar, Computer Science and Informatics, Vol. 27, No 1, Special Issue on Natural Language Processing and Machine Learning, p.p. 26-38, 1997.

[77] Sergei Nirenburg, et.al., A knowledge-based machine translation, volume4, issue1, p.p. 5-24, 1989.

[78] Martha Palmer, Rebecca Passonneau, Carl Weir and Tim Finin**,** The KERNEL text understanding system, Artificial Intelligence, volume63, issue 1-2, p.p.17-68, 1993.

[79] Noam Chomsky, On Certain Formal Properties of Grammars, Information and Control, Vol. 9, p.p.137-167, 1959.

[80] Noam Chomsky, Syntactic structures, 2nd edition, ISBN_3_11_0 17279_8, 1957.

[81] S.Lalithadevi, P.Pralayankar , V.Kavitha, Translation of Hindi se to Tamil in a MT System, Information systems for Indian languages, Berlin Heidelberg: Springer-Verlag, p.p. 246–249, 2011.

[82] Devi S.L., Ram V.S., et. al., Syntactic structure transfer in a Tamil to Hindi MT system - A hybrid approach, Computational linguistics and intelligent text processing, volume6008, p.p. 438-450, 2010.

[83] Helen Meng, PO-Chui Luk, GLR Parsing with Multiple Grammars for Natural Language Queries, ACM Transactions on Asian Language Information Processing, Vol. 1, No. 2., p.p. 123-144, 2002.

[84] Weng F., Stolke A., Partitioning grammar and composing parsers, In Proceedings of the 4th International Workshop on Parsing Technologies, 1995.

[85] Tomita M., An efficient augmented-context-free parsing algorithm, Computational Linguistics archive, Volume 13, Issue 1-2, p.p. 31-46, 1987.

[86] Akshar Bharthi, Rajeev Sangal, A Karaka Based approach to Parsing Indian Languages, Foundations of Software Technology and Theoretical Computer Science Lecture Notes in Computer Science, Vol.472, p.p. 410-420, 1990.

[87] S. Abney, Parsing by chunks, Principle based parsing, Kluwer Academic Publishers, 1991.

[88] Abney .S, Partial parsing via finite state cascades, Natural Language Engineering, Cambridge University Press, p.p.1-8, 1995.

[89] Miles Osborne, Shallow parsing as part of speech tagging, CoNLL-2000 and LLL-2000, p.p.145-147, Portugal, 2000.

[90] Rob Koeling, Chunking with maximum entropy models, CoNLL-2000 and LLL-2000, p.p.139-141, Portugal, 2000.

[91] Philip Brooks, SCP: Simple chunk parser, 2005, www.ai.uga.edu /mc /ProNTo/Brooks.pdf.

[92] A.Gelbukh, UCSG Shallow parser, CICLin 2006, LNCS3878, Springer -Verlag, p.p.156-167, 2006.

[93] Win Win Thant, Tin Myat Htwe et. al., Context Free Grammar Based Top-Down Parsing of Myanmar Sentences, International conference on computer science and information technology, Pattaya, p.p. 71-75, 2011.

[94] Mark A Jones et. al., A Probabilistic parser applied to software testing documents, Proceedings of national conference on Artificial Intelligence, San Jose, p.p. 322-328, 1992.

[95] Mark A Jones, A probabilistic parser and its application, Statistically based natural language processing techniques workshop, p.p. 20-27, 1992.

[96] Brian Roark, Probabilistic top down parsing and language modeling, Computational linguistics, volume 27, p.p. 249-276, 2001.

[97] Richard A. Frost, Rahmatullah Hafiz, A new top-down parsing algorithm to accommodate ambiguity and left recursion in polynomial time, ACM SIGPLAN, volume41, issue5, p.p. 46-54, 2006.

[98] Stuart M Scheiber, Sentence disambiguation by a shift reduce parsing technique, 8th international Joint conference on artificial intelligence, p.p. 699-703, West Germany, 1983.

[99] Stuart M Scheiber, A uniform architecture for parsing and generation, 12th International conference on computational linguistics, p.p.614-619, Hungary, 1988.

[100] A.Abeille, et. al., Using lexicalized tags for machine translation, 13th International conference on computational linguistics, volume 3, Finland, p.p. 1-6, 1990.

[101] Yves Schabes, Stuart M Shieber, An alternative conception of tree adjoining derivation, 30th annual meeting of the association of computational linguistics, p.p. 167-176, Newark, 1992.

[102] Steve Deneefe, Kevin Knight, Synchronous tree adjoining machine translation, EMNLP-2009: Proceedings of the 2009 Conference on Empirical methods in natural language processing, Singapore, p.p. 727-736, 2009.

[103] Stuart M Shieber, Yves Schabes, Generation and synchronous tree adjoining grammars, Computational intelligence, 1992, p.p. 220-228.

[104] Murthy. K. 2002. MAT: A Machine Assisted Translation System. In Proceedings of Symposium on Translation Support Systems, STRANS-2002, IIT Kanpur, India,. p.p. 134-139, 2002.

[105] MANTRA- Part of Smithsonian Institution's National Museum of American History, ttp://pune.cdac.in/html/about/success/mantra.aspx.

[106] Cristina Bonet, et. al., Discriminative Phrase-Based Models for Arabic Machine Translation, ACM Transactions on Asian Language Information Processing, Vol. 8, No. 4, Article 15, 2009.

[107] Tong Xiao, Jingbo Zhu, et. al., Language Modeling for Syntax-Based Machine Translation Using Tree Substitution Grammars: A Case Study on Chinese-English Translation, ACM Transactions on Asian Language Information Processing, Vol. 10, No. 4, Article 18, 2011.

[108] Chung-Chi Huang and Ho-Ching Yen, Translation Using Sub lexical Translations to Handle the OOV Problem in Machine Translation, ACM Transactions on Asian Language Information Processing, Vol. 10, No. 3, Article 16, Publication date: September 2011.

[109] Yufeng Chen, ChengqingZong, A Structure-Based Model for Chinese Organization Name Translation ACM Transactions on Asian Language Information Processing, Vol. 7, No. 1, Article 1, 2008.

[110] Elaine Rich, Kevin Knight, Shivashankar B. Nair, Artificial Intelligence, The McGraw Hill Companies , Third Edition, p.p.295-300, 2009.

[111] Stuart Shieber and Yves Schabe**s,** Synchronous Tree Adjoining Grammars, *13th* International Conference on Computational Linguistics (COLING'90), volume3, p.p.253-258, 1990.

[112] Yves Schabes, Anne Abeille, and Aravind K. Joshi, Parsing strategies with `lexicalized' grammars: Application to Tree Adjoining Grammars, 12th International Conference on Computational Linguistics (COLING'88), Budapest, Hungary, August, 1988.

[113] Dania Egedi, Hyun Seok Park, Martha Palmer, and Aravind K. Joshi, Korean to English Translation Using Synchronous TAGs. Manuscript, University of Pennsylvania, 1994.

[114] Victoria L. Fossum, Integrating parsing and word alignment in syntax based machine translation, ISBN : 10:1244577723,2010

[115] Richard A. Frost1, Rahmatullah Hafiz1, and Paul Callaghan2 Parser Combinators for Ambiguous Left-Recursive Grammars LNCS 4902, Springer-Verlag, Berlin, pp. 167–181, 2008.

[116] A Punjabi to English Machine Translation System for Legal Documents, Ph.D. thesis,Available at : http://www.amazon.com /Punjabi-English- Machine-Translation- Documents/dp/3659506036.

[117] K.P Soman et.al., Rule Based Machine Translation from English to Malayalam, International Conference on Advances in Computing, Control, and Telecommunication Technologies,2009, pp.439-441.

Publications of the researcher related to this work

International Journals

1. Latha R. Nair, David Peter S., Machine Translation System for Indian Languages, International Journal of Computer Applications, Volume 39, No.1, February 2012.

2. Latha R. Nair, David peter S. et. al., System for Syntactic Structure Transfer from Malayalam to English, The International Journal of computer applications, volume iRAFIT, pp. 26-30, 2012.

3. Latha R. Nair, David Peter S., et. al., Design and development of a Malayalam to English translator- A transfer based approach, International journal of Computational Linguistics(IJCL), Volume(3), issue(1), 2012.

International conferences

1. Latha R Nair, Sumam Mary Idicula, Design and development of a compound word synthesizer for Malayalam language, International Conference on Advances in Optoelectronics, Information and Communication Technologies (ICOICT), February, 2009, p.p.561-565 .

2. Latha R. Nair, Sumam Mary Idicula, Augmented Transition Network Based Chunking for Malayalam Language, 2nd International Conference on Advanced Computing and Communication Technologies for High Performance Applications, Dec. 2010.

3. Latha R. Nair, David Peter S., Development of a Rule Based Learning System for Splitting Compound Words in Malayalam Language, IEEE Recent advances in Intelligent Computational systems (RAICS), 2011, p.p. 751-755, 10.1109 /raics. 2011.

4. Latha R. Nair, David Peter S., Shallow Parser for Malayalam Language using Finite State Cascades, 4th International Congress on Image and signal processing, China, 2011, p.p. 2464-2467.

National Conferences

1. Latha R Nair, Sumam Mary Idicula, An efficient morphological processor for Malayalam Language, 37th All India conference on Dravidian Linguistics, International School of Dravidian Linguistics, June, 2009.

2. Latha R Nair, Sumam Mary Idicula, Hidden Markov Model Part of Speech Tagger for Malayalam Language, 38th National Conference on Dravidian Linguistics, International School of Dravidian Linguistics, June 2010.

Appendix 1

Excerpts from Bilingual Dictionary

["അഗ്നി", "n",”fire”],
["അഗ്രം", "n",”edge”],
["അംഗം", "n",”part”],
[" അങ്കം", "n",”fight”],
[" അംഗന", "n",”lady”,H],
["അംഗുലി", "n",”finger”],
["അങ്ങാടി", "n",”market”],
["അച്ചടക്കം", "n",”discipline”],
["അച്ചിങ്ങ", "n",”beans”],
["അച്ചാർ", "n",”pickle”],
["അച്ച്", "n",”mold”],
["അച്ചൻ", "n",”father”,H],
["അമ്മ", "n",”mother”,H],
["അട", "n",”cake”],
["അടക്ക്", "v",”close”],
["അടക്കി", "v",”suppressed”],
["അടപ്പ്", "n",”lid”],
["അടയ്", "v",”close”],
["അടക്ക", "v",”arecanut”],
["അടർ", "v",”peel”],
["അടിമ", "n",”slave”,H],
["അടിച്ചു", "v",”beat”],
["അടിക്ക്", "v",”beat”],
["അടുക്കള", "n",”kitchen”],
["അടുപ്പ്", "n",”stove”],
["അടുപ്പം", "n",”closeness”],
["അട്ട", "n",”millipede”],
["അട്ടഹാസം", "n",”laughter”],
["അണയ്", "v",”put off”],
["അണി", "n",”army”],
["അണിയറ", "n",”room”],
["അണു", "n",”atom”],

```
[ " അണ്ണാൻ", "n",”squirrel”],
[ "അതിക്രമം", "n",”attack”],
[“അപ്പോൾ”,”condp”,”when”]
[ "അത്തർ", "n",”perfume”],
[ "അത്യാഗ്രഹം", "n",”greed”],
[ "അവിടെ", "n",”there”],
[ "അദ്ദേഹം", "n",”he”,H],
[ "അധികാരം", "n",”power”],
[ "അനത്ത്", "v",”heat”],
[ "അനന്തരം", "n",”afterwrds],
[ "അനലൻ", "n",”fire”],
[ "അനാഥ", "n",”orphan”,H],
[ "അനിലൻ", "n",”breeze”],
[ "ആർപ്പ്", "n",”shouting],
[ "ആല", "n",”workshop”],
[ "ആലപിക്ക്", "v",”sing”],
[ "ആലിംഗനം", "n",”hug”],
[ "ആലേപം", "n",”apply”],
[ "ആലോചന", "n",”thought”],
[ "ആവരണം", "n",”covering”],
[ "ആവാഹിക്ക്", "v",”catch”],
[ "ആവേശം", "n",”spirit”],
[ "ആശങ്ക", "n",”doubt],
[ "ആശയം", "n",”idea”],
[ "ആശാൻ", "n",”teacher”,H],
[ "ആശ്ചര്യം", "n",”surprise”],
[ "ആശ്രയം", "n",”support”],
[ "ആഹാരം", "n",”food”],
[ "അസുരൻ", "n",”asura”,H],
[ "ആറ്", "n",”river”],
[ "ഇക്കിളി",”n”, "tickle"],
[ "ഇംഗിതം",”n”, "wish"],
[ "ഇഞ്ചി", “n”,"ginger"],
[ "ഇട",”n”, "gap"],
[ "ഇടനാഴി", “n”,"corridoor"],
[ "ഇടയൻ",”n”, "shepherd",H],
[ "ഇടറ്",”v”,"break"],
[ "ഇടിക്ക്", "v",”hit”],
```

["ഇണ", "n","pair"],
["ഇന്ധനം", "n","fuel"],
["ഇര", "n","prey"],
["ഇരുട്ട്", "n","darkness"],
["ഇല", "n","leaf"],
["ഇഷ്ടം", "n","liking"],
["ഇളക്", "n","moving"],
["ഇളവ്", "n","concession"],
["ഇഴയ്", "n","mov"],
["ഇറക്കം", "n","descend"],
["ഇറക്ക്", "v",down"],
["ഇറങ്ങ്", "v","start"],
["ഈട്", "n","strong"],
["ഈട്ടി", "n","rosewood"],
["ഈണം", "n","tune"],
["ഈച്ച", "n","fly"],
["ഈർപ്പം", "n","wetness"],
["ഈർച്ച", "n","anger"],
["ഈശൻ", "n","god",H],
["ഉക്തി", "n","idea"],
["ഉഗ്രം", "n","powerful"],
["ഉടൻ", "pav","now"],
["ഉടക്ക്", "v","quarrel"],
["ഉണക്ക്", "v","dry"],
["ഉടമസ്ഥൻ", "n","owner"],
["ഉടുപ്പ്", "n","dress"],
["ഉണർത്ത്", "v","wake"],
["ഉടുക്ക്", "v","wear"],
["ഉണ്ണി", "n","child"],
["ഉത്തമം", "n","perfect"],
["ഉത്തരവാദം", "n","responsible"],
["ഉല്പാദകം", "n","product"],
["ഉത്ഭവം", "n","origin"],
["ഉത്സാഹം", "n","spirit"],
["ഉദരം", "n","stomach"],
["ഉദാഹരണം", "n","example"],
["ഉദ്യോഗം", "n","job"],
["ഉന്തി", "v","pushed"],

```
[ "ഉന്നം", "n",”aim”],
[ "ഉന്നതം", "n",”top”],
[ "ഉപകാരി", "n",”helpful”],
[ "ഉപദേശം", "n",”advice”],
[ "ഉരുള", "n",”scoup”]
```

Appendix 2
Transliteration table

അ	ആ	ഇ	ഈ	ഉ	ഊ	എ	ഏ	ഐ	ഒ	ഓ	ഔ
a	aa	I	ee	u	oo	e	E	ai	o	O	ou

അം
m

ക	ഖ	ഗ	ഘ	ങ
ka	**kha**	**ga**	**gha**	**nga**
ച	ഛ	ജ	ഝ	ഞ
cha	chha	ja	jha	nja
ട	ഠ	ഡ	ഢ	ണ
Ta	**Tha**	**Da**	**Dha**	**Na**
ത	ഥ	ദ	ധ	ന
Tha	**thha**	**da**	**dha**	**na**
പ	ഫ	ബ	ഭ	മ
Pa	**pha**	**ba**	**bha**	**ma**

യ	ര	ല	വ	ശ	ഷ	സ	ഹ	ള	ഴ	റ
ya	**ra**	**la**	**va**	**Sa**	**sha**	**sa**	**ha**	**La**	**zha**	**Ra**

ങ്ക	ഞ്ച	ണ്ട	ന്ത	മ്പ	ൻ	ൽ	ർ	ൾ	ൺ
nka	**ncha**	**nta**	**ntha**	**mpa**	**n**	**l**	**R**	**L**	**N**

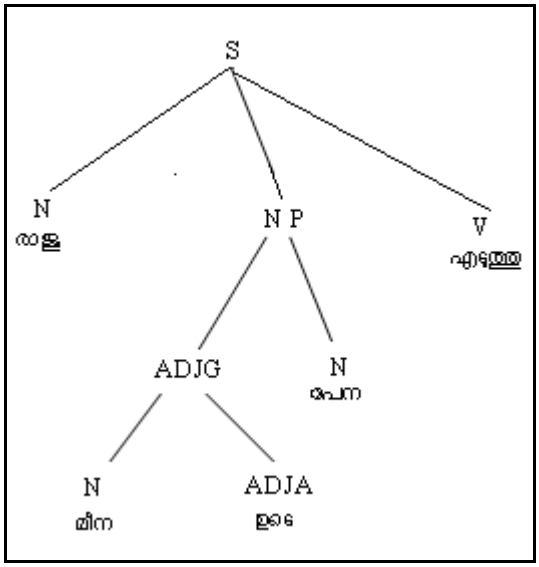
The letters like ത്യ, ത്വ ,ത്ര are generated by the sequences thya, thva and thra respectively. ് is marked by z.

The geminated letters ക്ക, ച്ച, ത്ത are generated by the sequences kka, chcha, ththa respectively.

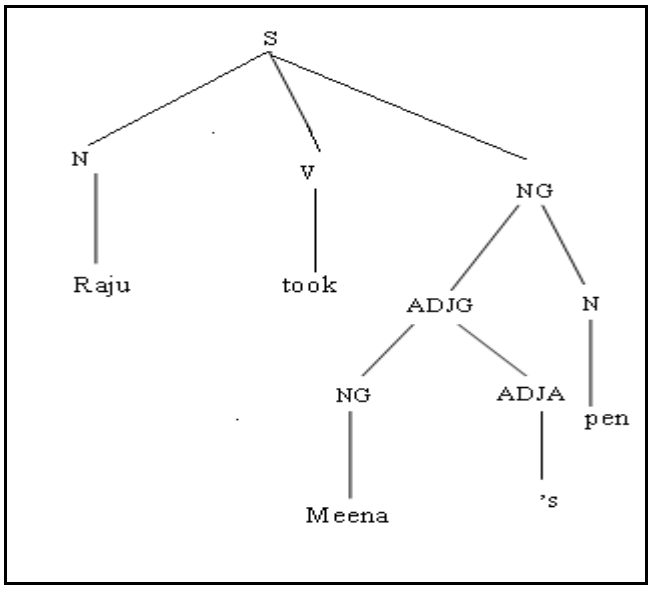
Appendix 3
Sample outputs from the Parser

1. Input: മീനയുടെ പേന രാജു എടുത്തു (Raju took Meena's pen)

   Source Parse Tree:



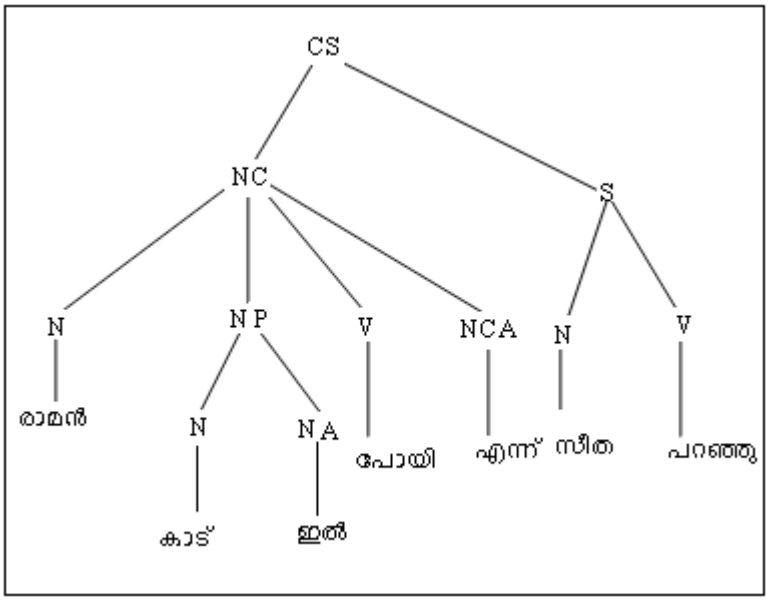
   Target parse tree:



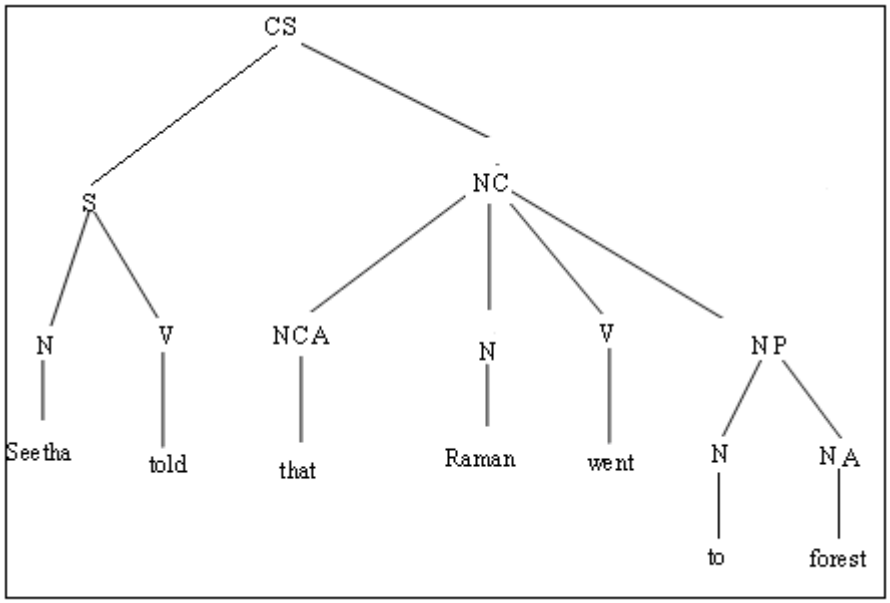
In the source parse tree the noun group(NG) chunk (മീന, ഉടെ) is treated as an adjective which depends on the noun പേന and not as NG which depends on the verb. This sentence also uses the transfer rule 4 to place the verb in position.

2. Input: രാമൻ കാട്ടില് പോയി എന്ന് സീത പറഞ്ഞു (Seetha told that Raman had gone to forest)

Source parse tree :



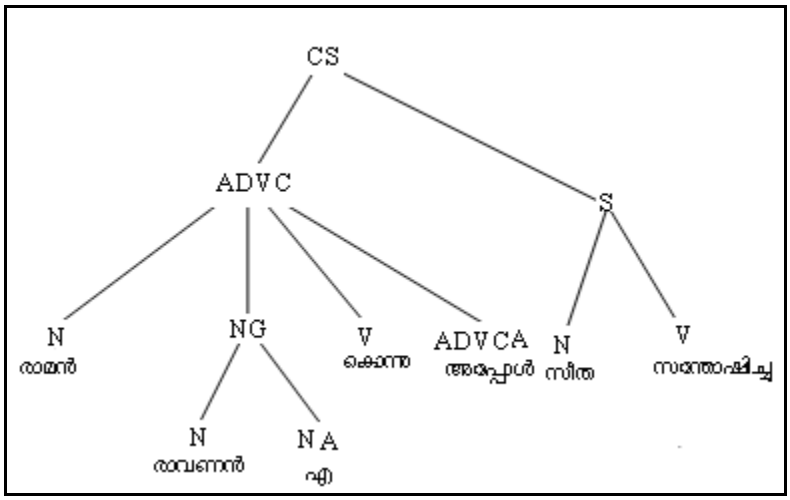
Target parse tree:



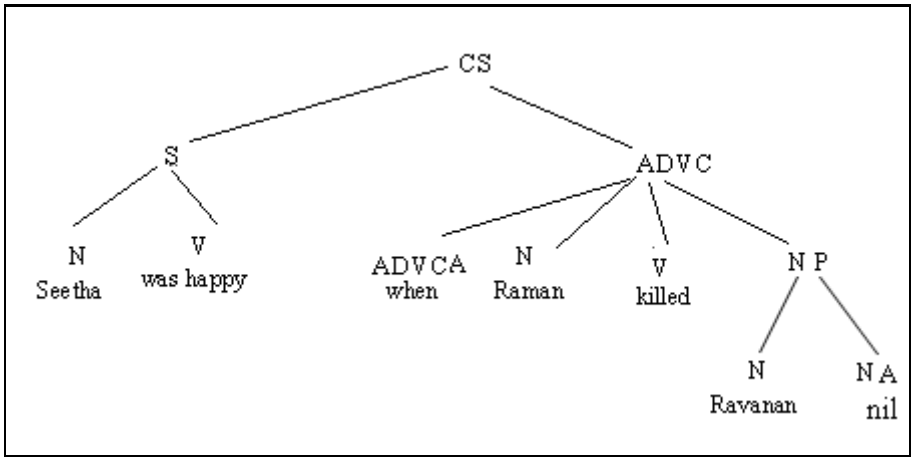
This uses 3 transfer rules. 1)Rule no.2 to move the noun clause suffix marker എന്ന് to the beginning of the noun clause. 2)rule no.7 to move the principal clause to the front. 3) Place the verb in the correct position in the noun clause.

3. Input: രാമൻ രാവണനെ കൊന്നപ്പോൾ സീത സന്തോഷിച്ചു (Seetha was happy when Raman killed Ravanan)

Source parse tree:



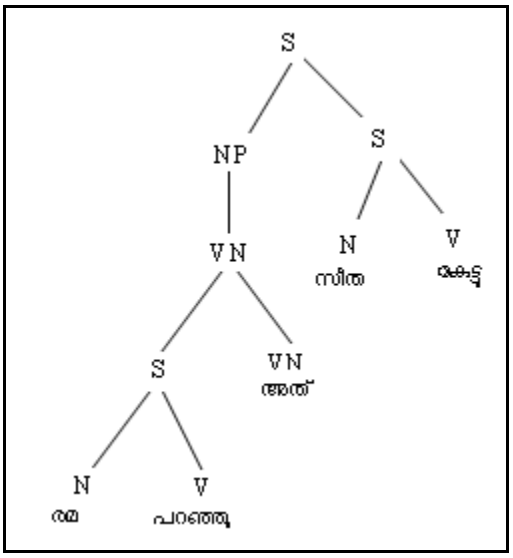
Target parse tree:



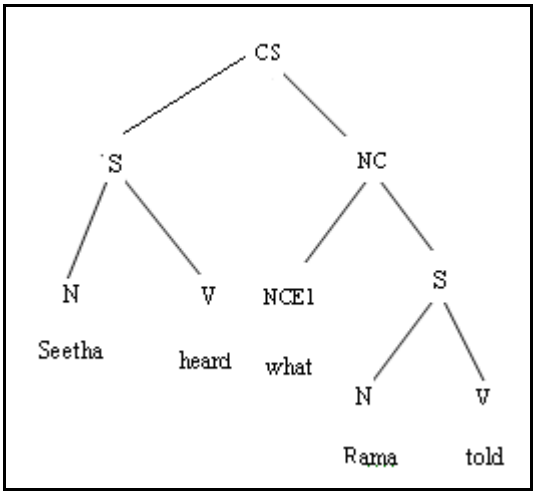
This sentence needs 3 transfer rules 1) rule 1 to place the adverb clause marker to the front of the clause 2) rule 7 to place the main clause in the front 3) rule 4 to place verb in position in the adverb clause.

4. Input: രമ പറഞ്ഞത് സീത കേട്ടു (Seetha heard what Rama told)

Source parse tree:



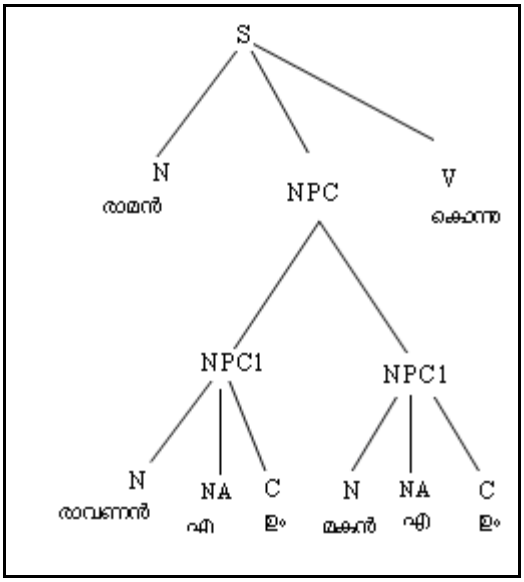
Target parse tree:



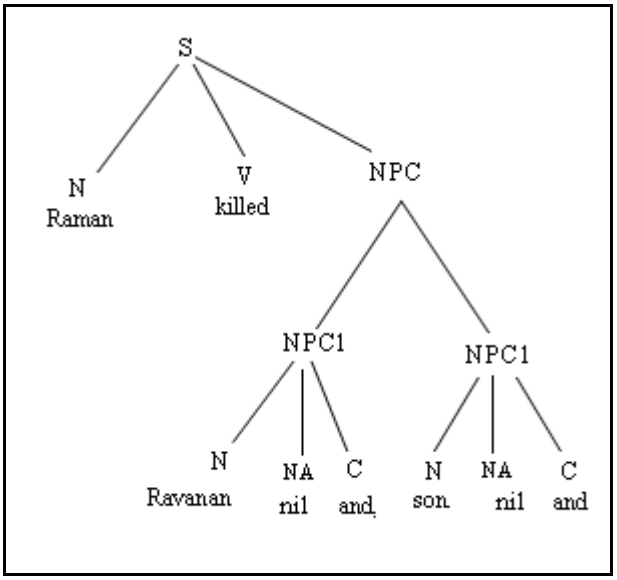
This uses 2 rules: 1) rule 1 to place the adverb clause marker to the beginning of the clause 2) rule 7 to move the main clause to the front.

5. Input: രാമൻ രാവണനെയും മകനെയും കൊന്നു (Raman killed Ravanan and his son)

Source parse tree:



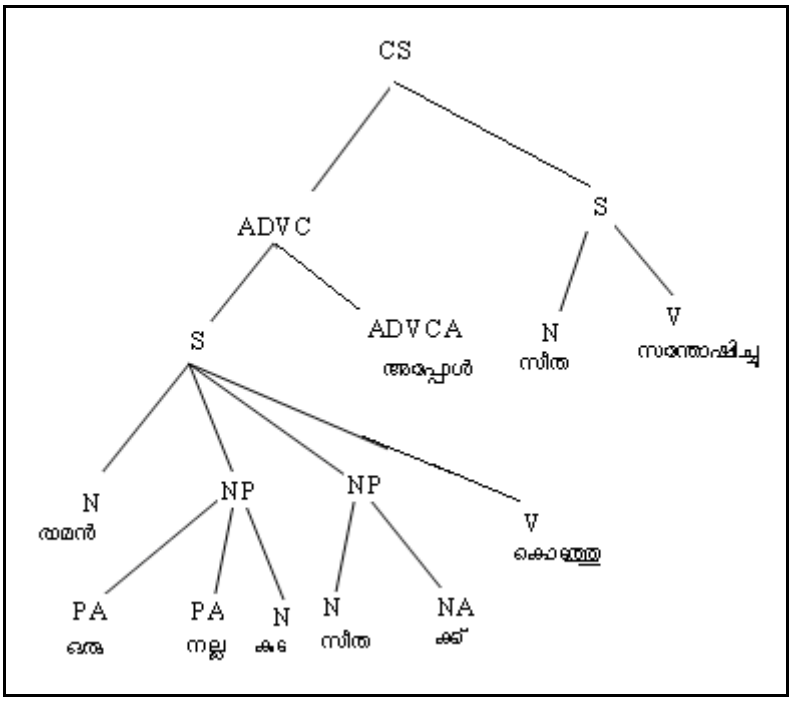
Target parse tree:



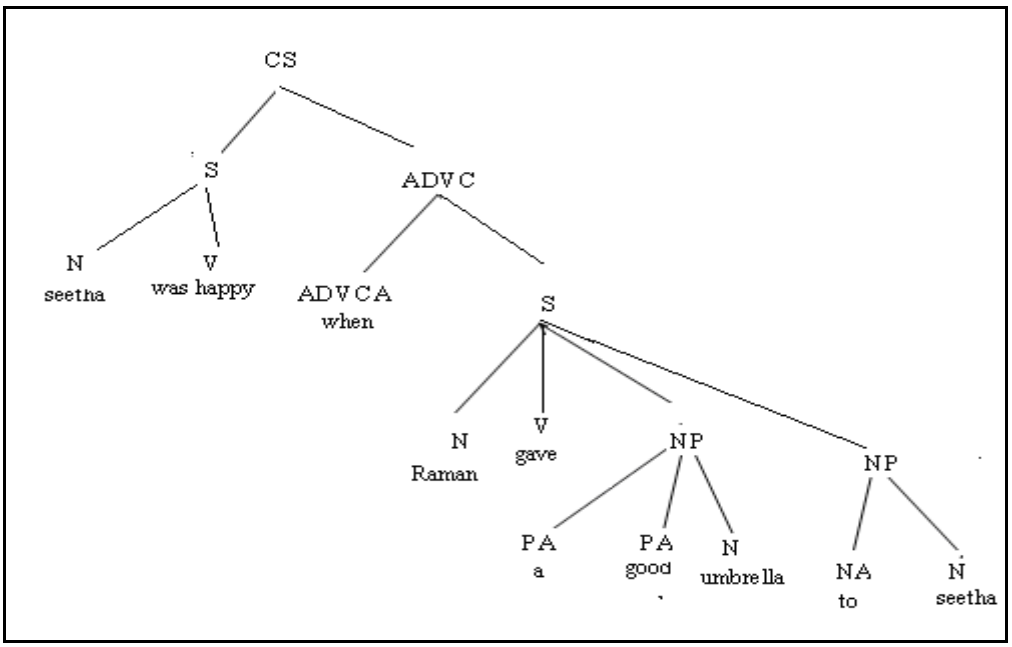
This uses rule 4 to place the verb in position.

6. Input: രാമൻ ഒരു നല്ല കുട സീതക്ക് കൊടുത്തപ്പോൾ സീത സന്തോഷിച്ചു (Seetha was happy when Raman gave an umbrella to Seetha)

Source parse tree:



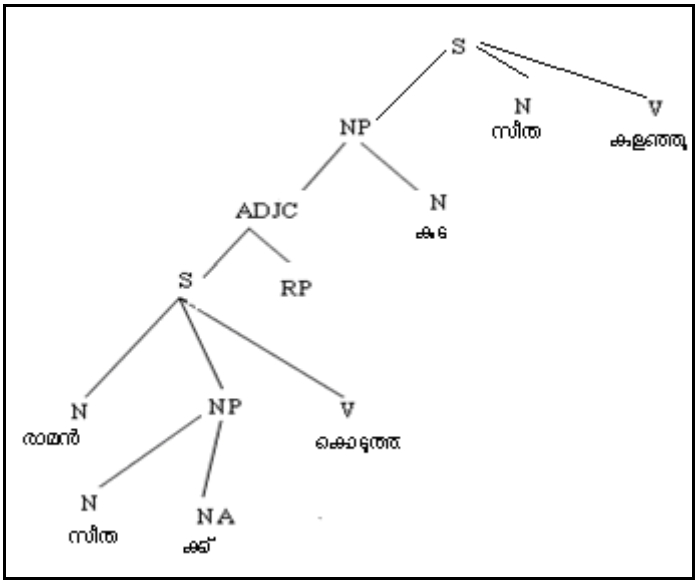
Target parse tree:



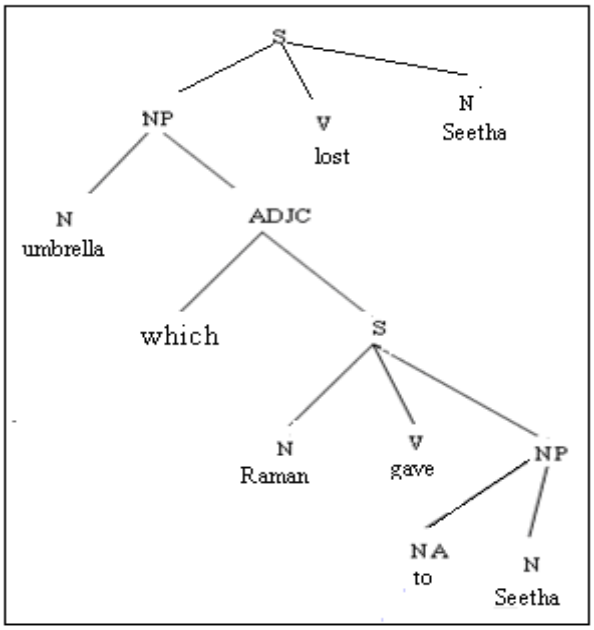
This uses 2 rules: 1) rule 1 to place the adverb clause marker to the beginning of the clause 2) rule 6 to move the main clause to the front.

7. Input: രാമൻ സീതക്ക് കൊടുത്ത കുട സീത കളഞ്ഞു (Seetha lost the umbrella which Raman gave)

Source parse tree:



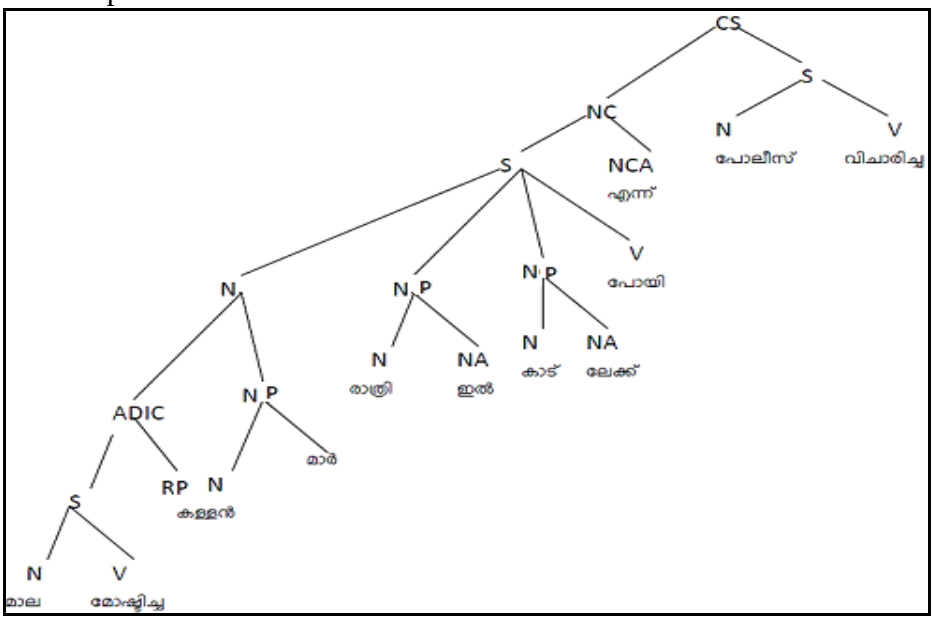
Target parse tree:



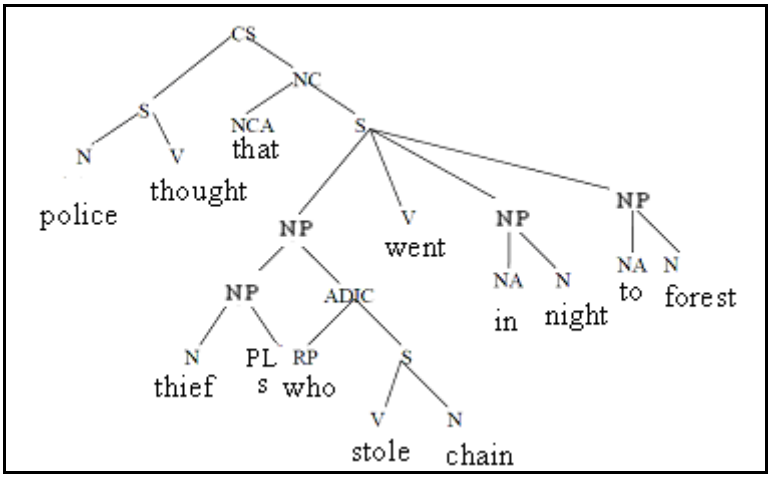
The input is parsed erroneously as follows. The error occurs due to the condition that clauses should not overlap. Here the subject of the main clause overlaps with the adjective clause.

8. Input:മാല മോഷ്ടിച്ച കള്ളന്മാര് രാത്രിയില് കാട്ടിലേക്ക് പോയെന്ന പോലീസ് വിചാരിച്ചു (The police thought that the thieves who stole the chain had gone to forest in night)

Source parse tree:



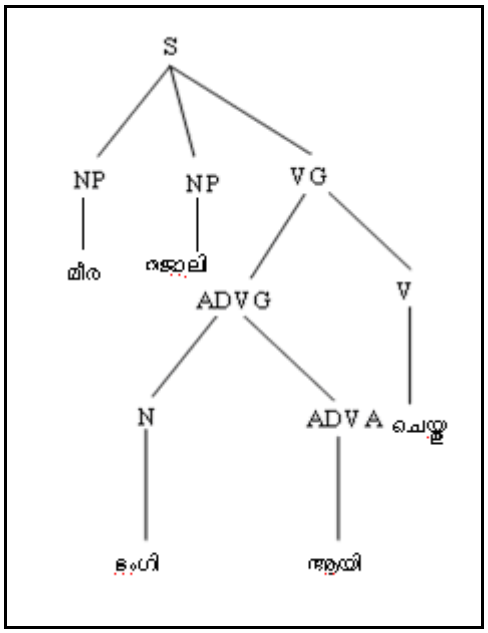
Target parse tree:



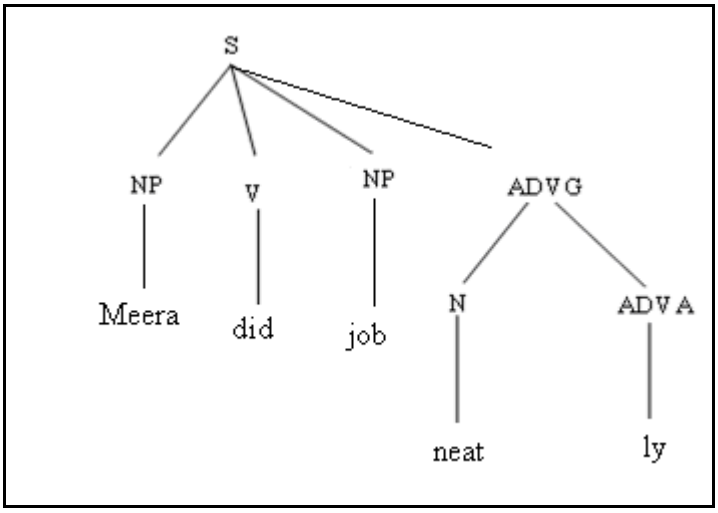
This uses 5 transfer rules: 1) rule 2 to move the adjective clause marker to the beginning of the clause. 2) rule 6 to move the noun following the adjective clause in front of the adjective clause. 3) rule 7 to place the main clause in the front 4) rule 3 to exchange noun and case suffix 5) rule 4 to place the verb after subject in the main clause and the subordinate clause.

9. Input: മീര ജോലി ഭംഗിയായി ചെയ്തു (Meera did the job neatlyt

Source parse tree:



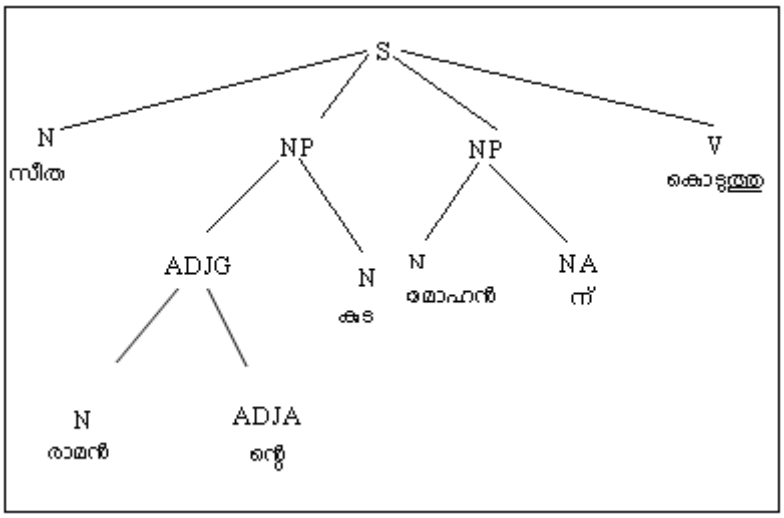
Target parse tree:



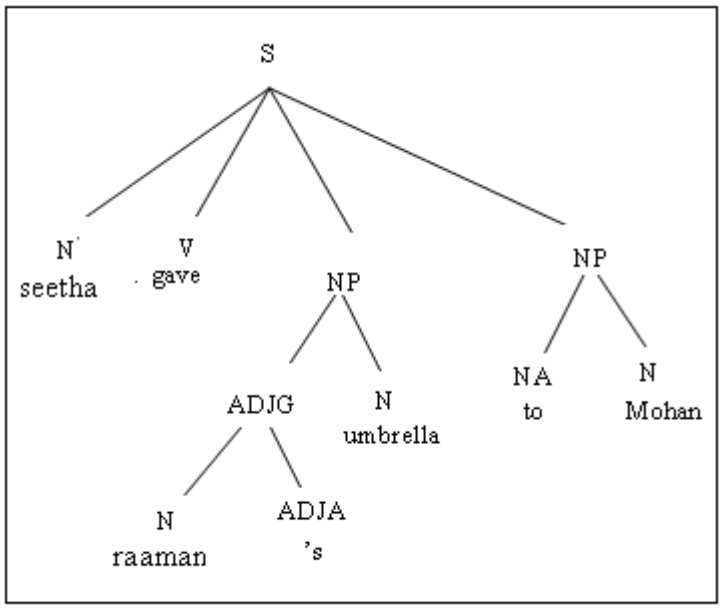
This example uses rule 13 to place the verb of a verb group before the object in the sentence.

10. Input: സീത രാമന്റെ കുട മോഹന് കൊടുത്തു (Seetha gave Raman's umbrella to Mohan)

Sourse parse tree:



Target parse tree:

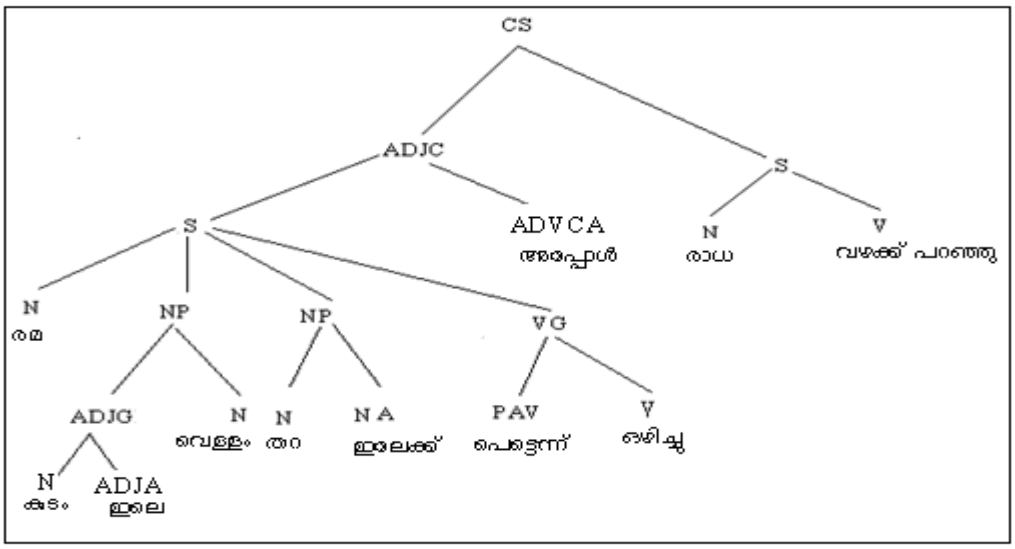


This uses two rules: 1) rule 4 to place the verb after the subject. 2) rule 3 to interchange the position of noun and case suffix.

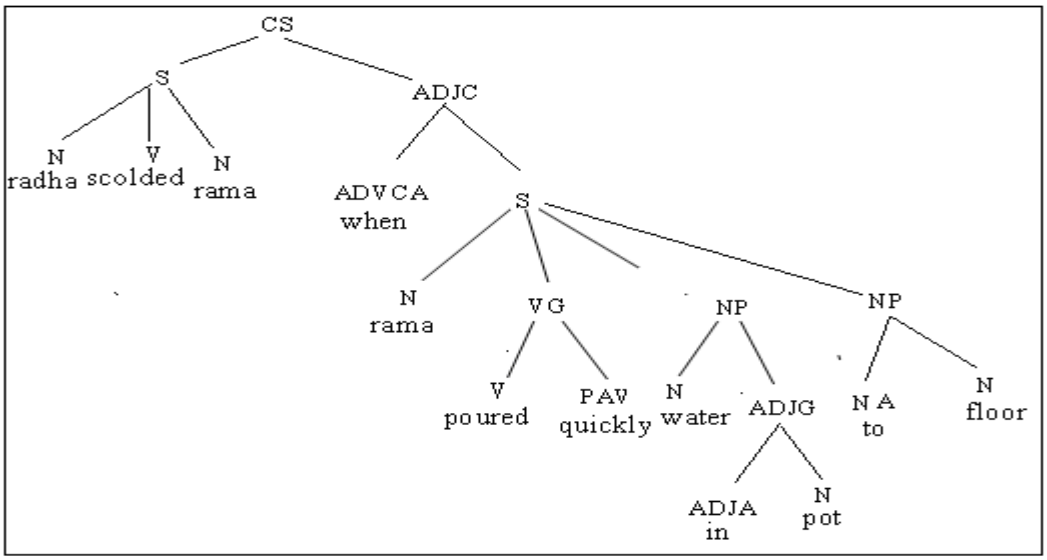
11. Input: രമ കൂടത്തിലെ വെള്ളം തറയിലേക്ക് പെട്ടെന്ന് ഒഴിച്ചപ്പോൾ രാധ വഴക്ക് പറഞ്ഞു (

Radha scolded Rama when she poured quickly the water in the pot to the floor)

Source parse tree:



Target parse tree:



This uses 6 rules 1) rule 3 to exchange noun and adjectival marker. 2) rule 11 to exchange adjective with suffix “ile” and noun following it 3) rule 12 to exchange the verb and the adverb. 4) rule 1 to place the adverb clause marker to the front of the clause. 5) rule 4 to place the verb in position in the main clause and the subordinate clause 6) rule 6 to move the main clause to the front.