ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

RULE BASED SYNTACTIC DISAMBIGUATION PARSER FOR AMHARIC SENTENCE

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of the requirements for the Degree of Master of Science in Information Science.

BY

WUBE ALEMAYEHU TUFFA

JANUARY 2005

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

RULE BASED SYNTACTIC DISAMBIGUATION PARSER FOR AMHARIC SENTENCE

BY
WUBE ALEMAYEHU TUFFA

Name and signature of members of the Examining Board:

Ato Getachew Jemaneh, Chairman, Examining Board: _____

Dr. Lina Zhou, Principal Advisor: _____

Ato Aklilu Yilma, Co-Advisor: _____

Ato Tesfaye Biru, External Examiner: _____

ACKNOWLEDGMENT

I am deeply indebted to my advisors Dr. Lina Zhou, Ato Getahun Amare and Ato Aklilu Yielma for their critical comments and suggestions, without whose close follow-up this study would have been incomplete. I am very much grateful to Kidane Mebrehatu who helped me a lot in the prototype development. I am also gratitude to Zemene Hadgo who assisted me in the development of the bi-gram count system.

I wish to express my thanks to Gezachw Brehanu and Mesele Solomon, Ethiopian language study students, who helped me in selecting and hand-parsing the sample sentences. My heartfelt thanks goes to my brothers Tedrows Alemayehu and Ermias Alemayehu, who helped me in stemming and feeding the affix into the database and the dictionary into the related files.

My deepest gratitude goes to my beloved friend, Azene Zenebe, for his unreserved, all-rounded support and invaluable inspiration. I am also indebted to Ato Mesfin Getahun and W/t Atelach Alemu, instructors at the department if Information Science for their assistance and valuable ideas.

While I am writing this acknowledgment, so many people – classmates, instructors, friends, family members, and colleagues come into my mined with their support, encouragements, and prays. May God bless all of them!

ABSTRACT

Parsing is important in Natural Language Processing. A parser allows a collection of Sentences to be analyzed in terms of its well- formedness according to a pre-defined grammar. A parser can be applied in Machine translation, spelling correction, grammar checking, summary, speech processing, question-answering systems. This paper presents a rule-based parser for Amharic sentence disambiguation.

The parser analyzes structurally ambiguous sentences with a rule base method. Using an active chart, the parser first generates all possible parses of a sentence according to given grammar rules. It employees depth-first search strategy in invoking rules and bottom up parsing strategy in constructing parses. In order to resolve structurally ambiguous sentences, the parser incorporates a dictionary that containing the headwords of noun phrases and verb phrases and their categories that is used to uniquely identify the groups of headwords. Structural rules are also provided for each category of the headwords in the dictionary.

The performance of the parser is measured with accuracy, which is evaluated by comparing the automatically parsed sentences against the hand-parsed sentences in the test set. The result shows that 86% of the sentences are parsed correctly. The result achieved is very encouraging. Given the small sample size available for rules induction, the parser can be improved by evaluating it on a large collection of Amharic sentences in future.

Abbreviations and symbols used

Symbols

( ) What is inside is optional except those used to indicate cite (source) of documents used

\ To separate words from their tags

Abbreviations

ADJ	adjective
ADV	adverb
AUX	Auxiliary verbs
CC	Coordinate conjunction
CN	Concrete Noun
CNo	Cardinal Number
COMP	Complement
CON	Compound Noun
CV	Complementized verbs
GNP	Genitive noun phrase
IJ	Interjections
JN	Joined Noun
MADJP	Main adjective phrase
MADVP	Main adverb phrase
MNP	Main noun phrase
MPP	Main prepositional phrase
MVP	Main Verb Phrase
NADV	Noun as an adverb
NL	Natural Language
NLP	Natural Language Processing
NLU	Natural Language Understanding
NP	Noun phrase

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1. Background

Although parsers have proved effective in the domain of programming languages, they suffer from inherent ambiguity in natural Languages processing (NLP). In analyzing ambiguous sentences, which are subject to two or more interpretations, humans intuitively employ various disambiguation heuristics to select the most likely one. In order for an NLP system to be useful, it should assign the same interpretation to a given sentence as a native speaker would, for that is exactly what a user would expect. For example, the sentence, "Time flies like an arrow" is structurally ambiguous. It can be interpreted in many ways including 'Does time fly as an arrow does?', 'Do you time flies as you would time an arrow?', 'Do time flies like arrows just as fruit flies like plums?', and so on. Native speakers of English show consistent preferences in parsing the above sentence. First, they saw "time as an arrow does", which means that their mental parser read "time" as a noun phrase and expected a verb to follow. The second meaning is less probable, for one has to parse "time" as a verb of imperative sentence. It is because "time" is most commonly used as a noun and less often as a verb, and least likely as an adjective describing a kind of fly, Kimball (1973); Frazier and Fodor (1978). A user would naturally expect an NLP system to reflect the above

preferences. Thus, such systems must model the *linguistic performance* as well as the *linguistic competence* of the native speaker in some way.

The computational "understanding" of a natural language can be thought of as a sequence of operations. The third operation (morphological analysis, POS tagging, parsing) corresponds to parsing, which is a procedure that takes a lexical from a tagger as input, more generally a fragment natural language, and outputs an explicit, unambiguous, structured representation of the sentence (Samad, 1986). However, it is often difficult to come up with such an efficient parser due to the prevalence of ambiguity in natural languages.

In fact, ambiguity is a major challenge for NLP systems. If the ambiguity is not resolved during parsing, it would reduce the effectiveness of the parser. Moreover, ambiguity may lead to misunderstanding, wrong interpretation and translation. To handle structurally *ambiguous* word sequences, it requires a parser to choose between two or more alternative structures via grammatical analysis.

1.2. Statement of the Problem And Its Justification

Amharic is the dominant language in Ethiopia. It has been extensively researched from a linguistic perspective, Analysis of Ambiguity in Amharic, Getahun Amare (2001); the syntax of functional category, Girma Halafom (1994); Amharic verb morphology, Bender, L. and Hailu

Fullas (1978); word formation, Girmaye Berhane (1992); Morphemic component of the conjugation forms of Amharic, Habtemariam Marcos (1994), and others. However, few attempts have been made with regard to investigating Amharic from an NLP point of view. It is only recently that Amharic language attracts the attention of NLP research community. Only limited studies have been conducted on Amharic language. They include, for instance, word stemming, Nega and Willett (2002); word parser that split different affixes and stem of a word Abiyot (2000); part-of-speech (POS) Tagging Mesfin (2001); morphological analyzer Tesfaye (2002); morphological synthesizer Kibur (2002); and sentence parser Atelach (2002) & Daniel (2003).

Amharic software currently available on the market suffers from lack of word processing facilities that provide deep analysis of the language. None of the software has language-specific support such as grammar checking, text retrieval, and spelling checker. The absence of these facilities makes it difficult to retrieve words from a text or to browse a text, letting alone automatically identify language structures to support intelligent applications such as machine translation and question and answer system.

Translating manually Amharic documents such as text books in to different local languages has been taking a lot of time to the extent of being bottle neck to the teaching learning process during the

introduction of the new educational policy. Still it is problem to translate Amharic legal documents into different regional state official languages (personal observation of experts at Ministry of Education and Ministry of Justice).

In addition to this, it is also important to make research on parser as one area of research.

- The thesis aims to develop an Amharic parser. Among all the NLP tasks such as morphological analysis and tagging, syntactic parsing deserves special attention, for understanding the structure of a sentence is crucial to interpreting its meaning. Natural language parsing is not a new topic. A variety of techniques have been developed for various languages to date. To the best of my knowledge, however, there are only two researches on Amharic reported in the extant literature Atelach (2002) Daniel (2003). Both of them tried to resolve structural ambiguity using statistical methods. The researchers are convinced a rule-based parser could be one of the accurate syntactic parser because it efficiently models the linguistic feature of a language.

- The purpose of this thesis is to address problems related to structural ambiguity in Amharic sentences and develop a parser for automatic structural disambiguation.

1.3. Objectives Of The Study

The followings are identified as general and specific objectives of the study.

1.3.1.General Objective

The general objective of the study is to investigate and develop a rule-based parser that can effectively resolve structural ambiguities of Amharic sentences.

1.3.2.Specific Objectives

The specific objectives of the study include:

- Study the types and nature of Amharic sentence ambiguity and review various parsing techniques suggested for automatic disambiguation.
- Collect relevant and representative Amharic sentences and parse them manually to create training and test data sets.
- Select a rule-based approach for parsing Amharic sentences.
- Develop a prototype Amharic parser for structural disambiguation
- Suggest directions for future study based on the evaluation results of the parser.

1.4. Methods

The following methods will be employed in conducting the proposed study.

1.4.1.Adapted Parsing Technique

Although there has been dramatic shift in computational linguistics from manually constructing grammars and knowledgebase to partially or totally automating the process by using statistical learning methods, this research adopts rule-based method. It is mainly due to the accuracy of the rule method than statistical method and also lack of large annotated corpora in Amharic, upon which the statistical method is highly dependent. In addition, there are some problems associated with the statistical method, which will be described in chapter five.

Prototyping approach will be applied in developing the parser due to time and others resource limitation. The parser is designed to reflect preferences among, structurally distinct parses of ambiguous sentences. In this approach a dictionary is build that provide knowledge about how best complements and modifiers of headwords of noun phrase and verb phrase can structurally be organized.

Visual C++ 6.0 and Visual Basic 6.0 are used to develop interfaces and the parser and MS-Access is used to develop morphological analyzer (stemmer database). The above programming languages are selected

because visual C++ has good string manipulation features and Visual basic has strong built in function to access database, and both are also user-friendly than the rest (Pascal, Perl, etc.) languages.

An active chart parsing method is selected to organize the passive arcs. Passive arcs are arcs, which are completely traversed according to the grammar rule. Generally, it is preferred for the following reasons:

1. It is possible to represent sentential ambiguity (i.e. sentences with more than one possible interpretation or syntactic structure).
2. It eliminates redundancy in the sub-structures generated during a parse by allowing sub-structures to be shared by subsuming structures.
3. It eliminates the duplication of effort in generating sub-structures which may be shared by more than one subsuming structure more than once, by storing completed sub structures independently of the substructure(s) which subsume(s) them, so that they may be shared as described in the above point;

A bottom-up rather than a top-down parsing strategy is adopted as it avoids left recursion problem. Depth-first search strategy is also preferred over the breadth-first search strategy, because it requires very modest memory space.

1.4.2.Literature and Data Collection Techniques

Parsing and searching state-of-the art techniques are reviewed to help select a parsing method. To better understand the features of sentence ambiguity in Amharic, both secondary and primary information sources are used. In addition, linguists in Amharic language provide consulting in analyzing the Amharic text.

Four hundred (400) sentences were collected to develop the disambiguation parser. Among them, 50 sentences were structurally ambiguous. The sentences were randomly selected from different books and newspapers, which also involved some judgment of the researchers. The sample sentences were first, manually analyzed, tagged, and parsed. The sentences were then passed to the linguistic advisor to receive feedback on the correctness of the manual parse.

1.5. Benefits and Application of the Disambiguation Parser

In Ethiopia more than 80 languages are spoken (Bender, 1976). Among them, Amharic is the official language of the Federal Government and five other Regional States. According to a census by ECSA[1], it is the first language for more than 17 million and second language for over five million people (Ethiopian Central Statistical Authority, 1998). Research

[1] ***ECSA*** stands for Ethiopian Central Statistical Authority of the Federal Democratic Republic of Ethiopia

work in Amharic language processing and understanding is still in its infancy. Developing an automatic disambiguation parser based on Amharic linguistic grammar will help to improve Amharic software currently available on the market.

A parser that can resolve ambiguities in Amharic sentences has both theoretical and practical implications. In theory, it extends the scope of parsing to Amharic language, meaning some theoretical framework of automatic parsing will be provided to Amharic language. In practice, it is important to many NLP applications. Thus, the beneficiaries of this study include researchers who are, or want to be, involved in increasing the capability of computer processing of Amharic. In general the following are few applications of the study:

- **Machine translation:** It could be possible to translate Amharic texts to different regional states languages and local languages.
- **Spelling correction, grammar checking:** It could be possible to provide spelling correction, and grammar checking facility to Amharic language with the help of morphological analysis component of the parser.
- **Better search engines**: It is possible to provide an intelligent search engine the retrieve Amharic texts by analyzing the meaning of texts and queries.
- **Information extraction and summary**: An intelligent system that models effectively the linguistic feature of Amharic can extract particular information from a give text and also summarize a text into few lines.

- **Speech processing**: In human computer interaction the system is expected to understand the speech made by the user. In order to understand the speech the system parses the sentence into its constituents.
- **Question-answering systems:** It is a system that answer for certain question presented to it based on specific domain in a text.

1.6. Scope of the Research

Among the various sources of ambiguity in Amharic such as lexical and semantic, the thesis focuses on the problems associated with structural ambiguity only. Both simple and complex sentences are considered in this research. However, the structure of the sentences is limited to the surface structure and doesn't take into account the deep structure (more complex). Amharic sentences can be classified into four types: declarative, interrogative, negative and imperative sentences (Baye, 1986). In this study we only examine the declarative sentences due to time and other resource limitation. In addition, the morphological analyzer does not contain all possible grammatical features of constituent affixes (e.g., subject markers, direct object markers, gender markers, tense markers).

1.7. Organization of the thesis

The thesis is organized into six chapters. The first chapter introduces the target problems and their potential solutions. In particular, it states the

problem and introduces the objectives of the study. Chapter two summarizes relevant literature on NLP and especially syntactic analysis. Different approaches and strategies for developing sentence parser and issues related to different methods to resolve structural ambiguities were also discussed. The third chapter describes the Amharic grammar, including various word classes, phrase categories, and complex sentence formalisms in the language. It also summarizes structural ambiguity pertaining to the language. Chapter four describes data preparation and the development of a morphological analyzer and a parts of speech tagger. It also explains how they are incorporated into the design of the parser. The algorithm for rule based structural disambiguation, the rule development, the experiment and the results are discussed in chapter five. Finally, conclusion and suggestions on future direction based on the findings of the study are presented in chapter six.

CHAPTER TWO

AUTOMATIC DISAMBIGUATION PARSER

As the main objective of constructing a parser is to understand the meaning of a sentence, the first section focuses on introducing sentence meaning and composition, and associated syntactic structural ambiguity. The classification of grammars and specific formalisms for linguistic grammars are presented in section 2.2 and section 2.3 respectively. Parsing methods and search strategies are described in section 2.4. Approaches to structural disambiguation are discussed in section 2.5.

2.1. Sentence Meaning and Composition

Meanings of words and sentences differ in one important aspect. Because meaning of words is known and learned, we can store words and their meanings. Stored meanings therefore are called lexical meanings (Formkin & Rodman, 1983). There is no need to store all the words, as the system can understand the meaning of some words from the stored lexicon by using lexical composition. For example, from *"believe"* in the stored lexicon the following words could be derived: *believed, unbelieved, unbelievable, unbelievably*. Complex expressions whose meanings are not stored in the lexicon are considered to have compositional meaning (Löbner, 2002).

A sentence comprises two types of meanings: lexical and structural meaning (Formkin & Rodman, 1983).

2.1.1.Lexical and Grammatical Meaning

The sentence:

(1) "The dog ate the yellow socks."

The examination of words in the above sentence including *the, dog, eat, yellow* and *sock* reveals that there are no larger units than lexical meaning. Thus, the interpretation of the sentence is the composition of lexical meanings. Nonetheless, we still observe that words occur in specific grammatical forms. The verb form *ate* is in past tense, the noun *socks* is in plural form, the adjective yellow is in its basic form, called 'positive'. The forms of the words matter directly to their meanings, and consequently to the meaning of the entire sentence. For example, the singular noun *dog* had a different meaning from the plural noun *dogs*; the meaning of present tense *eat(s)* is not the same as that of past tense *ate*. It is common practice that only one meaning of a word is stored in our lexicon, reasonably the singular meaning of nouns, a tenseless meaning of verbs and the 'positive' meaning of adjectives. Therefore, the meanings of the words in their given form must be derived from their lexical meanings by rules. These rules are part of the apparatus for composition (Löbner, 2002). The grammatical form itself, e.g. singular, plural, positive, comparative, simple past tense, progressive past tense, etc. has a meaning. Such meanings are called grammatical meaning (William & Michael. 1996).

2.1.2.Structural Meaning and Combination Rules

This lexical meaning of a sentence is revealed by the particular morphemes of which it is composed, but sentence meaning is more than the sum of the meaning of morphemes. How the constituents of the sentence are organized has a great influence over the meaning of the sentence. This is called structural meaning (Formkin & Rodman, 1983). To assign a structural meaning of a sentence one has to analyze the phrase structure of the sentence or parse the sentence into its constituents (Anderson, 2002).

(2) "The dentist hurt my teeth."

For example, one can get a whole range of meanings not expressed in the lexical records of the words themselves from sentence (2). For example, that the "dentist" performed the action, not the "teeth". However, if the same words were organized in a different structure, as shown in sentence (3), it would give us different meanings.

(3) "My teeth hurt the dentist."

Syntax is part of our linguistic knowledge, which concerns the structure of a sentence. Syntactic rules that determine the correct orders in a sentence are more than merely words placed one after another like beads on a string (Formkin & Rodman, 1983). Sentences organized into subgroups of words are called 'constituents'. Constituents have a tree-like structure, which uses to represent sentences as syntactic tree (Allen, 1995).

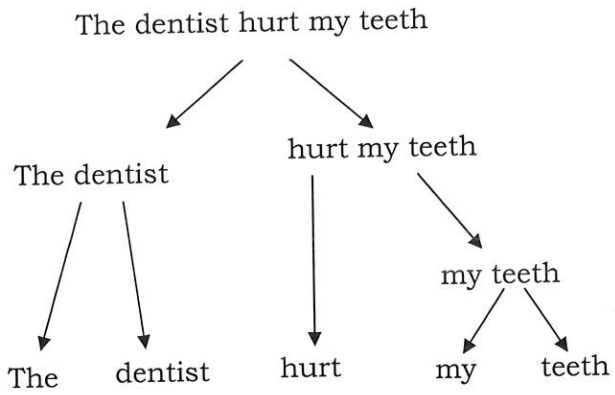


Figure 2.1 Syntactic Structure

The above structural diagram is correct, but it is redundant, and it can be simplified by employing phrase structures. *"The dentist hurt my teeth"* is a sentence that could be represented by 'S'. It also consists of two structural constituents: a noun phrases *"The dentist"* and a verb phrase *"hurt my teeth"*. Therefore *"The dentist"* is substituted by 'NP' and *"hurt my teeth"* is substituted by the 'VP". The verb phrase *"hurt my teeth"* consists of two structural sub constituents: The verb *'hurt'* and the noun phrase *"my teeth"*. The pronoun *'my'* and the noun *'teeth'* are sub constituents contained in a larger noun phrase constituent *"My teeth"*.

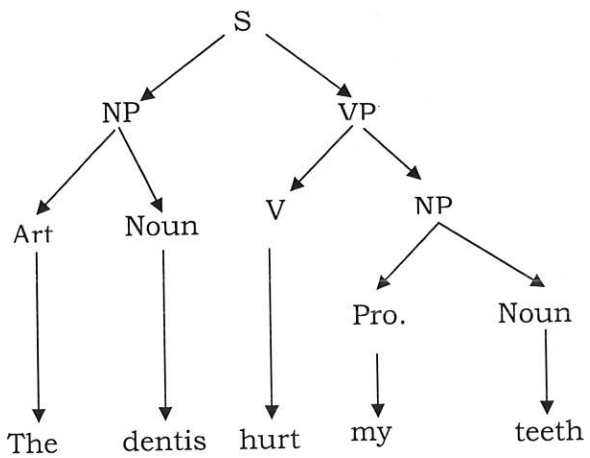


Figure 2.2 Phrase Structure

The combination of words into larger syntactic units is governed by the rules of grammar (Löbner, 2002). There is a rule for combining adjectives with nouns, and another rule for combining this adjective-noun combination or nouns alone, with an article (the article comes first). Given such rules for forming larger syntactic units, we need corresponding composition rules for meanings, for example:

- A rule for deriving the meaning of an article-noun NP *(the dentist)* from the meaning of the article and the meaning of the noun.
- A rule for deriving the meaning of a VP *(hurt my teeth)* from the meaning of the verb *(hurt)* and the meaning of the direct object NP *(my teeth);*
- A rule for deriving the meaning of a sentence *(the dentist hurt my teeth)* from the meaning of the subject NP *(the dentist)* and the meaning of the VP *(hurt my teeth).*

In sum, we can draw the following general results from the above examples. The syntactic rules of a language allow the formation of complex expressions from what will be called basic expressions. (Basic expressions are expressions with a lexical meaning). Therefore, the meaning of these expressions a sentence is drawn from three sources (Formkin & Rodman, 1983):

1. The **lexical meanings** of the basic expressions
2. The **grammatical forms** of the basic expressions
3. The **syntactic structure** of the complex expression

> "The meaning of a complex expression is determined by the lexical meanings of its components, their grammatical meanings and the syntactic structure of the whole" (Löbner, 2002, p.15).

The combination of these three sources is called principle of compositionality (Formkin & Rodman, 1983). The principle implies that the meanings of complex expressions are fully determined by the three sources mentioned, i.e. by the linguistic input alone.

2.1.3.Syntactic Structural Ambiguity

The principle of composition (Formkin & Rodman, 1983) emphasizes the importance of syntactic structure to the interpretation of a sentence. There are inherent difficulties associated with determining the syntactic structures of a sentence. Ambiguity is a term that refers to a concept where lexical or sentences have two or more readings or meanings (Allen, 1995). In dealing with meaning of sentences in the previous topics, expressions were treated as though they had only one meaning. This is, of course not the case. Some expressions are lexically ambiguous, some are structurally ambiguous and some are semantically ambiguous. Structural ambiguity occurs when the grammar assigns more than one possible parse to a sentence (Jurafsky & Martin, 2002).

(4) "I would like to see the synthetic buffalo hides."

The syntactic knowledge goes beyond our being able to decide which strings are grammatical and which are not (Formkin & Rodman, 1983). It accounts for the structural ambiguity of expression like the one in the

above sentence. The humor of the sentence is enhanced by the ambiguity of the phrase "synthetic buffalo hides." The ambiguity results from the fact that *synthetic* can modify *buffalo hides* or only *buffalo*. That is you may parse the phrase either as (*synthetic buffalo* and *hides*) or as (*synthetic* and *buffalo hides*).

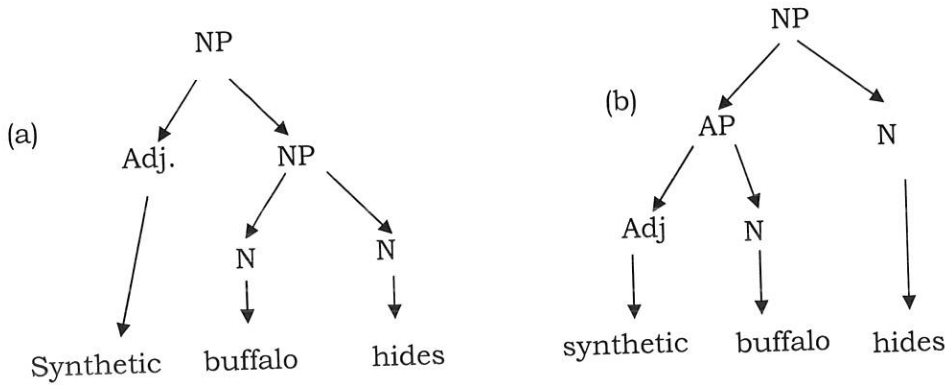


Figure 2.3 Two Ways of Ambiguity

Each parsing tree represents one of the possible meanings.

2.2. Grammar Classification

A wide range of grammar classes have been investigated in parsing and language modeling, depending on the nature of the application and particular insights on language structure and sentence distribution. Grammatical representations of meaningful relationships may be classified into three main classes: *linguistic grammars*, *task-oriented grammars* and *data-oriented grammars* (Löbner, 2002).

Linguistic grammars are a general-purpose grammar that addresses the main issues in applying linguistic theory to the development of computational grammars. The issues are *coverage, predictive power* and *computational requirements.* Task-oriented grammars are used in most current applications such as text summarization, information retrieval and speech understanding, which specifies directly how relationships relevant to the task than general-purpose grammar. In a data-oriented framework, learning or *training* procedure tries to determine the evaluation function that produces best results on an appropriate training corpus (William & Michael, 1996). Data-oriented grammars is dated back to the beginning of statistical studies of language by Markov, but such grammars capable of representing meaningful relationships have started only recently.

2.3. Grammar Formalism

Linguistic grammar is used to model natural language. Hence, grammar in a language model or parser uses as the specification of a configuration space in which the configurations represent stages of constituent combination, and transitions between configurations describe how constituents are combined in deriving larger constituents (Pereira, Sentence Modeling and Parsing). For instance, in a more complex case of phrase-structure grammars, configurations represent sequences of phrases (*sentential forms*), and transition of the possible combinations of adjacent phrases into larger phrases. A *derivation* of a sentence according

to the grammar is a path in the configuration space of the grammar from an initial configuration to a final configuration in which all the elementary constituents are consumed (Pereira, Sentence Modeling and Parsing).

Any parse that models a sentence must include a generative mechanism or *grammar* that specifies how sentences are built from their parts, and how the information associated to the sentence derives from the information associated to its parts (Pereira, Sentence Modeling and Parsing).

There are a number of grammar formalisms proposed by different scholars. Among them, there are Transition Network Grammars, Context Free Grammar, Features and Unification Based Grammars and Context Dependent Grammar.

2.3.1. Transition Network Grammars

Transition network grammar is based on the notion of a transition network consisting of nodes and labeled arcs. The nodes are specified as the initial state or start state, transition state and final or accepting state (Allen, 1995).

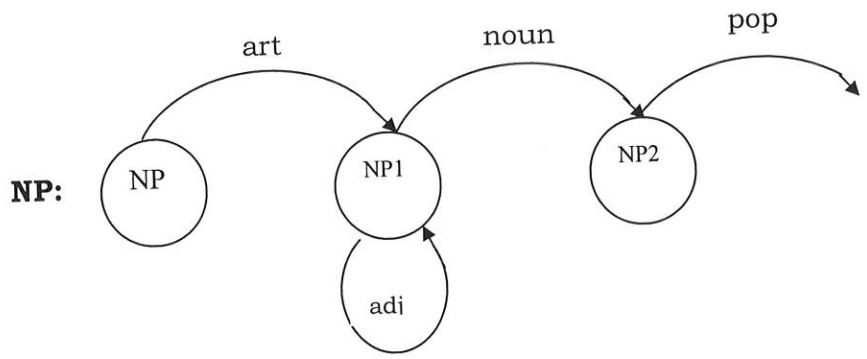


Figure 2.4 TNG (Allen, 1995)

Consider the above NP grammar network with the initial state; one can traverse an arc if the current word in the sentence belongs to the category on the arc. If the arc is followed, the current word is updated to the next word. A phrase is a legal NP if there is a path from the node NP to a POP arc that accounts for every word in the phrase. This network recognizes the same set of sentences with the following context-free grammar:

$$NP \rightarrow ART + NP1$$

$$NP1 \rightarrow ADJ + NP1$$

$$NP1 \rightarrow Noun$$

Simple transition networks are often called finite state machines (Jurafsky & Martin, 2002). Finite state machines are equivalent in expressive power to regular grammars, and thus are not powerful enough

to describe main features of a language. To get the descriptive power, it needs a notion of recursion in the network grammar.

2.3.1.1. Recursive Transition Network Grammars

A Recursive Transition Network (RTN) is like a simple transition network, except that it allows arc labels to refer to other networks as well as word categories (Allen, 1995). In recursive transition network two or more simple transition networks are interconnected with arc labels.

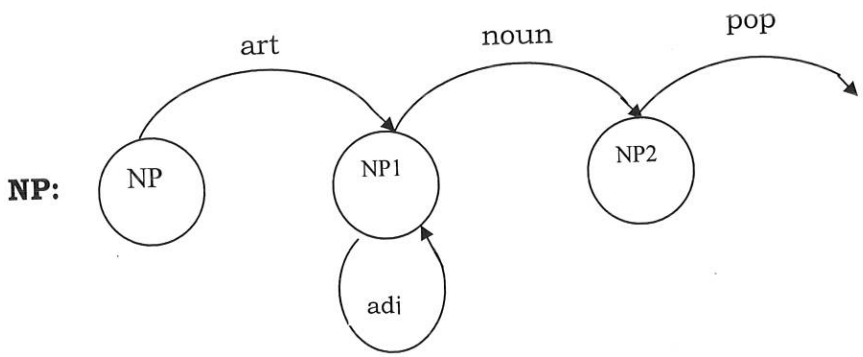


Figure 2.5 Simple TNG (Allen, 1995)

We can construct a recursive transition network that recognizes a sentence by using the above simple NP transition network in figure 2.5.

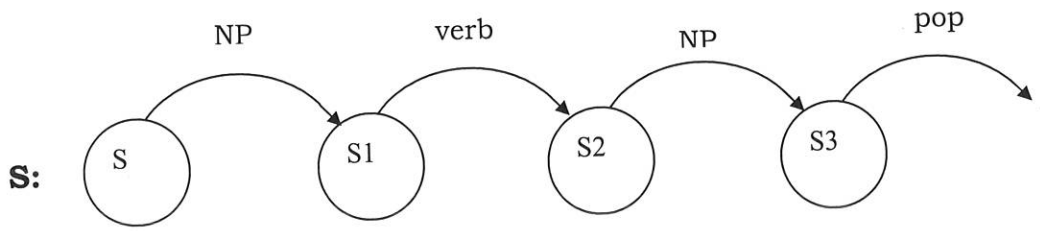


Figure 2.6 RTN (Allen, 1995)

The arc labeled as NP, in the sentence transition network in Figure 2.6, refers to the NP simple network. The arc from S to S1 can be followed only if the NP network can be successfully traversed to a pop arc.

2.3.1.2. Augmented Transition Networks

When a feature is added to a recursive transition network, it produces an Augmented Transition Network (ATN). A feature that will be added is an agreement restriction between words and phrases. The words and phrases restriction includes subject-verb agreement, gender agreement for pronouns, and restrictions between the head of a phrase and the form of its complement (Allen, 1995).

Features in an ATN are traditionally called Registers (Allen, 1995). When constituent structures are created, each network will contain a set of registers. Each time a new network is pushed, a new set of registers is created. As the network is traversed, these registers are set to values by

actions associated with each arc. When the network is popped, the registers are assembled to form a constituent structure, with the CAT slot indicating the network name (Harries, 1985).

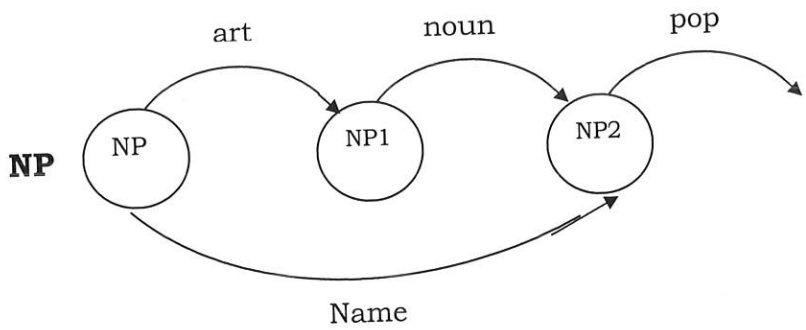


Figure 2.7 ATN (Allen, 1995)

(5) "The dog saw Jack."

In order to check the grammaticality of the subject NP in sentence (5) against the given ATN, we have to extract feature values of arcs 1 and 2. For instance, to record the value of arc 1 in the registers we have to follow a path that describes the sequence of the features structure, as shown in Figure 2.8(a). The path has two arcs, labeled with CAT and AGR, and three nodes, of which two are terminal nodes with values of **ART** and **SG, PL**. Each of the features has atomic-values.

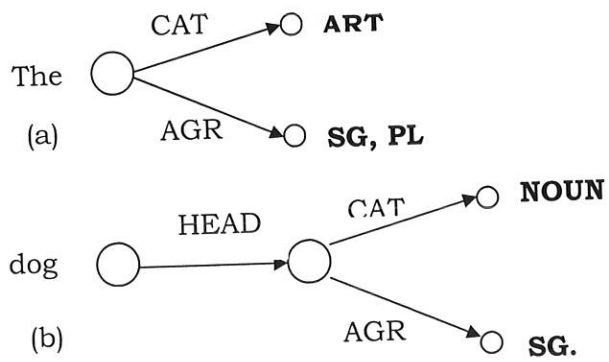


Figure 2.8 Sequence of the Features Structure

Agreement test will be conducted in the two AGR registers on whether there is intersection in their values. Therefore, the AGR value of the NP constituent in Figure 2.8 will be **SG**, which is the intersection between values of the two sub-constituents.

2.3.2.Context Free Grammar

Context free grammar (CFG) is grammar formalism where the syntax of each constituent is independent of the symbols occurring before and after it in a sentence (Harris, 1985). It is a mathematical system most commonly used for modeling constituent structure in natural language. Context-free grammars are also called Phrase-Structure Grammars, and the formalism is equivalent to what is also called Backus Naur Form or BNF (Jurafsky & Martin, 2002).

A context-free grammar consists of a set of rules or productions, each of which expresses the ways that symbols of the language can be grouped and ordered together into lexicon of words and symbols (Harries, 1985). For example:

S → NP + VP (2.1)

NP → Det + Nominal (2.2)

NP → ProperNoun (2.3)

Nominal → Noun | Noun Nominal (2.4)

VP → Verb + NP (2.5)

The above production rule expresses that an NP (or noun phrase), can be composed of either a ProperNoun or a determiner (Det) followed by a Nominal; a Nominal can be one or more Nouns.

Context free rules can be hierarchically embedded, so we could combine the pervious rule with the following, which express facts about the lexicon:

Det → A (2.6)

Det → the (2.7)

Noun → flight (2.8)

This formalism is powerful enough to describe most of the structure in natural languages, yet it is restricted enough so that efficient parsers can be built to analyze sentences. Symbols that cannot be further decomposed in a grammar, namely the words in the preceding example, are called terminal symbols. The other symbols, such as NP, VP, and S are called non-terminal symbols. The grammatical symbols such as N and V describe word categories are called lexical symbols (Allen, 1995).

A CFG is usually thought of in two ways: as a device for generating sentences, or as a device for assigning a structure to a given sentence (Jurafsky & Martin, 2002). As a generator, we could read the → arrow "rewrite the symbol on the left with the string of symbols on the right". So staring from the symbol NP, we could use rule 2.2 to rewrite NP as Det

Nominal. And then rule 2.3 rewrites Nominal as a Noun and finally via rule 2.6 and 2.8 we find the NP "a flight". In the other way the grammar assigning Det to an article 'a' and Noun to the word "flight" using rule 2.7 and rule 2.8. Rule 2.2 and 2.4 assign NP structure by to "a flight" combining the two parts of speech.

2.3.3.Features and Unification Based Grammars

Feature structures and unification provide computational implementation of a constraint-based formalism. A constrained based representation scheme will allow us to represent fine-grained information about number, person, and tense agreement, sub categorization, as well as semantic categories like mass/count (Jurafsky & Martin, 2002).

Feature structure is a method used to capture and represent the kind of grammatical properties of sentence constituents. One of the simplest ways to encode this kind of properties that we have in mind is through the use of feature structures. The encoding could possibly include constituent part of speech category (CAT), number, and person as follows (Backofen, 1995):

CAT	NP
NUMBER	SG
PERSON	3

The above feature structure must operate with feature structures of other constituents. Feature structures would be of little use without performing

reasonably efficient and powerful unification operations on them. The two principal operations are merging the information content of the two structures and rejecting the merger of structures that are incompatible. Fortunately, unification, a single computational technique, suffices for both of these purposes (Jurafsky & Martin, 2002). The unification results as follows:

[NUMBER SG] U [NUMBER PL] → Fails/Reject

[NUMBER SG] U [NUMBER [ ]] = [NUMBER SG]

After developing the feature and unification constraints, we associate it with the context free rules of the grammar and finally incorporate it into the parser (Joshi, 1996).

2.3.4.Context Dependent Grammar

The basic idea behind the context dependent grammar is that it generates strings with a special start symbol and then applies rules that specify how certain combination of symbols may be replaced with other combinations of symbols as a constraint. The constraint rules is that the length of the string on the left-hand side is at least one, and is less than or equal to the length of the string on the right-hand side (Warren, 1999).

It has rules that rewrite a non-terminal symbol A in the context $\alpha A \beta$ as any non-empty string of symbols. They can be either written in the form

$\alpha A\beta \rightarrow a\gamma A$ or in the form $A \rightarrow \gamma / a \_\_\_ \beta$ (Jurafsky & Martin, 2002). A linguistic model that is known to be context-sensitive is Tree-Adjoining Grammar (Joshi, 1985).

2.4. Parsing Methods and Search Strategies

Parsing is the process of assigning syntactic or (and) semantic representations to natural language expressions according to a grammar (Arnold, 2001). EAGLES Central Secretariat also defines *parsing* as the process of assigning structural descriptions to sequences of words in a natural language (or to sequences of symbols derived from word sequences). Just what sort of structural description is assigned and on what grounds depends either on *grammar* - a description language plus set of structural constraints or statistics that could be acquired from a corpus (EAGLES Central Secretariat, parsing).

Therefore, generally, there are two types of methods for parsing (Atelach, 2002):

- Rule based approach: pure rule reasoning driven by grammar rules;
- Statistical method: probabilistic reasoning driven by grammar rules with the association of statistical method.

2.4.1.Rule Based Method

For decades, the majority of NL parsers have been "rule-based." In such parsers, knowledge about the syntactic structure of a language is written

in the form of linguistic rules (grammar rules), and these rules are applied to input text segments in order to produce the resulting parse trees by the parser. Information about individual words, such as what parts-of-speech they may be, is usually stored in an online dictionary, or "lexicon," which is accessed by the parser for each word in the input text prior to applying the linguistic rules (Richardson, 1994).

2.4.2.Statistical Method

Although rule-based parsers are widely used, in working NLP systems, they have disadvantage that extensive amounts of dictionary data and labor (to write the rules) by highly skilled linguists are required in order to create, enhance, and maintain them. This is especially true if the parser is required to have "broad coverage", i.e., if it is to be able to parse NL text from many different domains (Wilms, no year, Web).

As a result in the last few years, there has been increasing activity in the computational linguistics community focused on making use of statistical methods to acquire information from large corpora of NL text, and on using that information in statistical NL parsers. Instead of being stored in the traditional form of dictionary data and grammatical rules, linguistic knowledge in these parsers is represented as statistical parameters, or probabilities. These probabilities are commonly used together with simpler, less specified, dictionary data and/or rules,

thereby taking the place of much of the information created by skilled labor in rule-based systems (Allen, 1995).

Advantages of the statistical approach that are claimed by its proponents include a significant decrease in the amount of rule coding required to create a parser that performs adequately, and the ability to "tune" a parser to a particular type of text simply by extracting statistical information from the same type of text. Perhaps the most significant disadvantage appears to be the requirement for large amounts of training data, often in the form of large NL text corpora that have been annotated with hand-coded tags specifying parts-of-speech, syntactic function, etc (Richardson, 1994).

2.4.2.1. Probabilistic Context Free Grammars (PCFG)

Probabilistic context-Free Grammar (PCFG) is the simplest augmentation of the context-free grammar. It is also known as the Stochastic Context-Free Grammar (SCFG), first proposed by Both in 1969 (Jurafsky & Martin, 2002). Probabilistic parsing specially used to disambiguate. A probabilistic grammar offers a solution to the problem by choosing the most-probable interpretation. Thus, due to the prevalence of ambiguity probabilistic parsers can play an important role in most parsing or natural language understanding tasks (Yao and Lua, 1998).

A PCFG is a 5-tuple G = (H_T, H_N, E^1, P, D) where H_T is the set of terminal symbols, H_N the set of non-terminal symbols, $E_1 \varepsilon H_N$ the start symbol, and P the set of productions. Productions take the form of $E \rightarrow \xi (P)$, with $E \varepsilon H_N$, $\xi \varepsilon (H_T \ U \ H_N)+$, and $P = PROB(E \rightarrow \xi)$, the probability that E is expanded into the string ξ. D is a function assigning probabilities to each rule in P. The probability of applying a particular production to an intermediate string is conditionally independent of what productions were previously applied to obtain the current string, or what productions will be applied to other symbols in the current string, given the presence of the left-hand symbol. Therefore, the probability of a given derivation is simply the product of the probabilities of the individual productions involved. The probability of a string in the language is the sum over all possible derivations (Jurafsky & Martin, 2002).

A probabilistic context-free grammar augments each rule in P with a conditional probability:

$$E \rightarrow \xi (P)$$

This function expresses the probability P that the given non-terminal E will be expanded to the sequence ξ. It is often referred to as:

$P(E \rightarrow \xi)$ Or as

$P(E \rightarrow \xi \mid E)$

Formally this is conditional probability of a given expansion given the left-hand-size non-terminal E. Thus if we consider all the possible expansions of a non-terminal, the sum of their probabilities must be one.

The independence assumptions between constituents make PCFG not to be better (Allen, 1995). Problems arise in many guises, but one critical issue is the handling of lexical items. For instance, the context-free model assumes that the probability of a particular verb being used in a VP rule is independent of the rule under consideration. This means that lexical preferences for certain rule cannot be handled in the basic framework. This problem then influences many other issues, such as attachment decisions (Jurafsky & Martin, 2002).

2.4.2.2. Best First Parsing

So far the PCFG doesn't contribute a lot to increase the efficiency of the parser. To increase the efficiency of the parser Best First search is introduced. The Best First Parsing algorithm attempts to explore the highest probability constituents first. The assumption is that the best parse can be found quickly as lower-rated possibilities are never explored in the search space (Allen, 1995). It adopted a heuristically guided search parsing strategy (Kay 1980).

The algorithm doesn't search from left to right in the search space rather it finds any word in the sentence that has a heights probability value.

2.4.2.3. Simple Context Dependent Best-First Parsing

Even though the Best-First algorithm helps to improvement the efficiency of the parser, by reducing searching space, it doesn't improve the accuracy problem of PCFG. The alternative solution to improve the accuracy is to make the probability rule based on the context dependent lexical information (Chilly, 2000).

The idea exploits the observation that the first word in a constituent is often the head and thus has a dramatic effect on the probabilities of rules that account for its complement. This suggests a new probability measure for rules that is relative to the first word, PROP (R | C, W). This is estimated as follows (Allen, 1995):

$$PROB\left(R/_{C,W}\right) = \frac{count\left(\#times\ rule\ R\ used\ \ for\ cat\ C\ sarting\ with\ W\right)}{count\left(\#times\ cat\ C\ starts\ with\ W\right)}$$

The effect of this modification is that probabilities are sensitive to the particular words used. Pronoun is relatively more common as a subject than as an object in a sentence. More importantly, the context-sensitive rules encode verb preference for a different sub categorization. For instance, given the rule VP → V + NP + PP definitely tell that the verb *"put"* will be used more times in this rule than the verb *"like"* which usually doesn't take a preposition (Allen, 1995).

2.4.3.Search Strategies in Parsing

Parsing is defined as a searching technique (Allen, 1995). In syntactic parsing, the parser can be viewed as searching through the space of all possible parse trees for the sentence. The search space of possible parse trees is defined by the grammar. Regardless of the algorithm implemented there are two constraints that should help to guide the search. These are the data or the input lexical and the grammar (Jurafsky & Martin, 2002). These two constraints give rise to the two search strategies, top-down and bottom-up.

2.4.3.1. Top-down Parsing

Top-down parsing is grammar-driven search, it starts with the start symbol of the grammar or at the top of the tree, tries to derive the input string. This is done by using the next input symbol(s) and the current state of the parser to properly "guess" the next derivation step (Jurafsky & Martin, 2002). Backtracking is important to top-down parsing. While searching, the parser could follow different strategies including depth-first strategy or breadth-first strategy (Knight, 1991).

In this goal oriented parsing technique, sub-goals that are already visited and recognized will be kept in a data structure known as chart. Algorithms that do this are called chart parsers (Russell & Norving, 1995).

2.4.3.2. Bottom-up Parsing

The basic idea of a bottom-up parser is that it starts with the words of the input, and tries to build a sentence or a tree from the words up by applying rules from the grammar one at a time (Jurafsky & Martin, 2002). The main operation is to take a sequence of symbols and match it to the right-hand side of the rules.

So far, various algorithms are developed to increase the efficiency of a parser. The goal is accomplished by avoiding repetitive matching using different types of data structure. As stated above, a chart data structure allows the parser to store the partial results of the matching it has done so far so that the work need not be reduplicated in chart parsing. In shift reduce parser a stack data structure will be used in order to store the already matched constituents of a sentence (Allen, 1995)

2.5. Approaches to structural Disambiguation

There are a number of methods used to resolve structural ambiguities. These methods could be classified into three types: syntactic approach, semantic approach and statistical approach (Frannz, 1996).

2.5.1.Syntactic Approach

Syntactic approach to ambiguity resolution uses structural property of the parse tree to choose a particular parse (Hang Li, 1996). This

approach usually is inspirited by psycho-linguistics. It involves structural heuristics to disambiguate. The information for disambiguation would be taken from the actual language expression and entered into a dictionary after being clarified (McRoy & Hirst, 1989).

In psycholinguistics, a number of principles have been proposed that attempt to modelize human disambiguation process. They include the Lexical Preference Rule (LPR)[2] (Ford et al., 1982); the Right Association Principle (RAP)[3] (Kimball 1973); the Attach Low and Parallel Principle (ALPP, an extension of RAP)[4] (Hobbs and Bear 1990); Minimal Attachment Principle (MAP)[5] and Late Closure (LC)[6] (Frazier and Fodor, 1978).

Syntactic factors clearly play a large role in ambiguity resolution; however it is widely acknowledged that many cases of structural ambiguity, including PP attachment, could not be resolved on the basis of structural properties of the sentence alone. Instead, semantic or pragmatic information is also required (Frannz, 1996).

---

[2] Give priority to the interpretations that are consistent with the strongest lexical form of the predicate.

[3] Attempt to attach to the lowest right non-terminal node in the parse tree.

[4] RAP prefers an interpretation attached to a nearer phrase, ALPP prefers interpretations with attachments that are low and in parallel.

[5] It is the preference for incorporating a new word into a parse tree using the fewest possible new nodes

[6] It is happen when grammatically permissible, attach new items into the clause or phrase currently being processed.

2.5.2.Semantic Approach

The semantic approach relies on the deployment of world or situation knowledge to assess alternative interpretations of a sentence. In order to realize an actual semantic analysis, such an approach requires semantic dictionary and interpretation systems, which generate fairly rich semantic analysis for sentences along with sophisticated inference mechanisms to operate on them (Hirst, 1986).

An extensive amount of work has been done on semantic features of disambiguation. It includes "selectional restriction"[7] (Katze and Foder, 1963), the framework of "conceptual parsing"[8] (Schank and Abelson, 1977; Schank, 1995), and so on. Parsing methods falling under this framework would be the case frame parser (Carbonell and Hayes, 1988) and expectation driven parsing (Riesbeck, 1987).

2.5.3.Statistical Approach

The dominant approach to syntactical ambiguity resolution at the moment involves the use of statistical techniques in one form or another. In particular, using statistical grammars, which take into consideration lexical dependencies, has proved a very sensitive way of capturing the kind of grammatical contexts that particular words or word combinations

[7] Restricting some expression which are sought inconvenient from analysis
[8] These parsers used domain- specific semantic knowledge to drive the parsing process, and to perform disambiguation.

typically appear in; this information can be effectively used to decide between alternative parses (Knott & Vlunter, 2002).

This approach uses large database or corpora of natural language data as its base for disambiguation. The data allow the system to use statistically based techniques for automatically deriving the probabilities needed (Allen, 1995)

CHAPTER THREE

THE AMHARIC GRAMMAR

This chapter focuses on reviewing grammatical rules of Amharic. The chapter consists of six sections. Section 3.1 highlights the characteristics of Amharic alphabet and punctuation marks. The word class of Amharic is discussed in section 3.2. Section 3.3 describes word classes including Interjection, Determiners, Quantifiers, Modifiers, and Complements. The syntactic structure of Amharic is discussed in section 3.4. Section 3.5 is devoted to feature of Amharic sentence. The last section summarizes structural ambiguity of Amharic sentences.

3.1. The Amharic Alphabet and Punctuation Marks

The present writing system of Amharic is taken from Ge'ez alphabet, which was the language of literature in Ethiopia at the early time. The Amharic writing system consists of a core of 33 characters (fidels) each of which occur in one basic form and in six other forms known as orders. The seven orders represent different forms of a consonant. These forms are created by the combination of the consonants and vowels. The non-basic forms are derived from the basic forms by more-or–less regular modification (Abiyot, 2000). For example using the consonants /q/ and /b/ the orders are:

basic 1st	2nd	3rd	4th	5th	6th	7th
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
qä	qu	qi	qa	qe	qə	qo
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
bä	bu	bi	ba	be	bə	bo

In addition to the above forms, there are nearly 40 others, which usually contain a special feature representing labialisations. For example,

ቈ	ቍ	ቋ	ቌ	ቍ
q^{w}ä	q^{w}i	q^{w}a	q^{w}e	q^{w}

Without including the labiovelars the alphabet includes 241 letters. Each graphic symbol represents a consonant together with its vowel. The vocalic symbol cannot be detached from the consonant element. Thus, the Amharic script is a syllabic rather than an alphabet (Leslau, 1967)

Analysis of Amharic texts reveals that different Amharic punctuation marks are used in Amharic to serve for different purposes. Beletu (1982) as cited in Abiyot (2000), indicated that there are about 17 punctuation marks of which only a few of them are commonly used and have representations in Amharic software.

In this study, Phonetics script is used to represent the Amharic alphabet instead of/fidels/ as it is difficult for morphological analysis and under

the assumption that it is possible to convert the script to their equivalent Amharic texts using transliterations software.

3.2. Word Classes in Amharic

It makes the study of features of Amharic words very easy and efficient to classify words into groups according to their similarities (Getahun, 1995). Grouping words into classes also makes language processing possible.

Mersi'hazen (1934)[9] categorized Amharic words into eight classes, including noun, pronoun, verb, adverb, adjective, preposition, conjunction, and interjection. Baye (1987)[10] and Getahun (1995)[11] reduced them into five classes according to their forms and position in sentences, which are: noun, verb, adverb, adjective, and preposition. They grouped pronoun into noun and conjunction into preposition because they share the same form and position in the sentence. They totally removed interjection because it doesn't have any syntactic property.

In this study, we mainly followed the set of classes from Mersi'hazen, but also incorporated the idea of eliminating interjection from Baye and

[9] Date is according to the Ethiopian calendar
[10] Date is according to the Ethiopian calendar
[11] Date is according to the Ethiopian calendar

Getahun. It is because that we found it necessary to separate pronouns from nouns and separate conjunctions from prepositions in tagging and parsing Amharic sentences. Finally, we obtained the following word classes: noun, pronoun, adjective, verb, adverb, preposition, and conjunction.

3.2.1.The Noun Class

Like English, Amharic nouns are words used to name or identify any of a class of things, people, places or ideas (Attelach, 2002). Nouns are classified into primitive and derived nouns (Gethahun, 1995). The primitive nouns are those nouns not derived from any other roots, or stem like /sämay/ 'sky', /märet/ 'earth' , /dəngay/ 'stone' and etc. A long list of primitive nouns is given in Laselu (1967). Derived nouns are those nouns that are derived from other roots such as /məsəl/ 'picture', /mäls/ 'answer' , and /qärät'/ 'tax'. Different types of derived nouns are provided in Getahun (1995).

According to Getahun (1995), the Amharic noun has the following forms and positions in the sentence.

- It takes the morpheme -**wočč**, or -**očč** at the end to make it plural.

/bäre/ox	/bärewočč/oxen
/fəyyäl/goat	/fəyyäločč/ goats

- It serves as the subject of a complete sentence structure.

fɔ̀yäl sar yɔ̀bälal /goat eats grass/ (1)
goat grass eat

- It serves as the object of a transitive verb in a sentence.

dawit bäre gäzza /Dawit bought an ox/ (2)
Dawit ox bought

- It may be preceded with a quantifier or deictic adjective.

dawitɔ̀ bɔ̀zu bäre gäzza /Dawit bought many oxen/ (3)
Dawit many ox bought

yɔ̀h bäre yädawit näw /This is Dawit's ox/ (4)
This ox Dawit's is

- It takes type modifier as adjective.

dawit qäy bärre gäzza /Dawit bought a red ox/ (5)
Dawit read ox bought

Since Amharic is a highly inflectional language, nouns change their forms according to number; gender, definiteness and case (Baye, 1986 & Getahun, 1995).

Number

Number in Amharic grammar indicates the quantity of a certain things, events, idea, and so forth.

Singular	Plural
bäg /sheep	bägočč /sheeps
säw /man	säwočč /men
mäkina/ car	mäkinawočč /cars
kɔ̀t'äl /leaf	kɔ̀t'älakɔ̀t'äl /leafs

Gender

Based on the gender they refer, we can classify nouns into female type and male type.

Male	Female
antä /you	anəči /you
yəh /this	yəhəčč /this
ya /that	yačči / that

Definiteness

A noun is called definite whenever it is known or particularly defined beforehand.

Subject	gloss	Definiteness	gloss
bet	'house'	betu	'the house'
bäg	'sheep'	begu	the sheep'
ləj	'boy'	ləju	'the boy'

Case

It refers to the roles of nouns in a sentence. It could be subject of a sentence, object in a verb phrase and possession in a noun phrase.

Subject	gloss	Object	gloss	Possession	gloss
bet	'house'	betun	'the house'	bete	'my house'
bäg	'sheep'	bägun	'the sheep'	bäge	'my sheep'
ləj	'child'	ləjun	'the child'	ləje	'my child'

Nouns are used in object form whenever they take definiteness form.

In Amharic numerals can be classified as nouns. These are words representing numbers, which include cardinal or ordinal numbers. A list of the Amharic cardinal numbers is found in Dawkins (1960) and Laslu (1973). In Amharic, the ordinal numbers are formed from the cardinal numbers by attaching the suffix **–(ä)ňňa**. For Example

Cardinal	gloss	Ordinal	gloss
and	'one'	and-äňňa	'first'
assər	'ten'	assər-äňňa	'tenth'
haya	'twenty'	haya-ňňa	'twentyth'

Compound Amharic numerals are constructed by concatenating multiple numerals sequentially like those in English. For example,

hulät mäto sälasa /two hundred thirty one/

sosət mäto häya and- äňňa /321st/

In Amharic, there are also numerals that indicate distribution. These numerals are called distributive numerals. For example,

and and "One for each of them."

3.2.2.The pronoun class

The pronoun is separated from the noun class because it is important for a system to resolve reference ambiguity in complex sentences. The Amharic pronoun includes personal, demonstrative, interrogative, indefinite, impersonal, emphatic, reflexive, distributive, and relative pronouns. All types of pronoun are listed in Dawkins (1960).

The Amharic personal pronouns are classified into independent and dependent personal pronoun. The dependent personal pronoun is a pronoun that is found inherently in the noun and the verb. The independent personal pronoun is a pronoun that stands by itself.

The independent personal pronouns are expressed independently as separate words. That it is why they are referred as independent personal pronouns. For example:

∂ne/I	antä/you (m)	ančči/you (f)
∂rso/you (pol)	∂rsu/he, it	∂res^{w}a/she
∂ňňa/we	∂nnantä/you (pol)	∂nnarsu/they
∂resaččw/ he, she (pol)		

These pronouns are used especially for emphasis and contrast because the subject of the verb is already attached to the verb as illustrated below.

∂mmokk∂rallähu /I will tray. /
try – Aux - I

∂ne ∂mmokk∂ralähu /I will tray. /
I try – Aux - I

There are occasions where the personal pronouns where expressed as dependent to the noun or the verb. Among the dependent pronouns, possessive suffixes, object suffixes and subject suffixes are three of them.

The possessive suffixes are added into a noun to refer possession. If the noun is /abbat/ 'father' the following suffix will be attached.

-e/my	-h/your (m)	-š/your (f)
-wo/your(pol)	-u/his or its	-wa/her
-ačč∂n/our	-ačč∂hu/ your (pl)	-aččäw/their (pol)

The object suffixes are the equivalent of the English object personal pronouns, and are suffixed to the basic verbs of which they are direct or

indirect object (Dawkins, 1960). If the basic verb is /fälläga/ 'find' the following suffix will be attached to identify the object to which they refer.

-ňň /me	-h/you (m)	-š/you (f)
-wot/you (pol)	-w/him or it	-at/her
-n/us	-ačču/you (pl)	-aččw/them(him, her)

The subject of a sentence can be found attached itself to the verb. This suffix is called subject pronoun marker. Both the perfect and imperfect verbs have the subject affix. Considering the stem /fällag-/ 'find', the different subject affixes for both perfect and imperfect verb types are given in the table below.

	person	Singular subject suffix	example	Plural subject suffix	example
Perfect	1st	-hu	fällaghu	-ɔn	fällagɔn
	2nd musc.	-h	fällagh	-aččɔhu	fällagaččɔhu
	Female	-š	fällagš		
	Pol.	-u	fällagu		
	3rd musc.	-ä	fällagä	-u	fällagu
	Femal	-äč	fällagäč		
Imperfect	1st	ɔ-	ɔfällɔgallähu	ɔn-	ɔnfällɔgallän
	2nd musc.	tɔ-	tɔfällɔgalläh	tɔ-	tɔfällɔgallačhu
	Female	tɔ-	tɔfällɔgialläš		
	Pol.	yɔ-	yɔfällɔgallu		
	3rd musc.	yɔ-	yɔfällɔgall	yɔ-	yɔfällɔgallu
	Femal	tɔ-	tɔfällɔgalläčč		

Table 3.1 the Subject Marker

The perfect verb shows actions that are completed while the imperfect describe actions that are incomplete.

3.2.3.The Adjective Class

Adjectives can also be classified into primitive and derived adjective classes. Like English adjectives, Amharic adjectives precede the noun they modify.

t∂ll∂q bäre /big ox/ (6)
big ox

However, words that modify a noun might not be necessarily an adjective (Getahun, 1995), it could be determiner, relative clause and others. Consider the underlined words given below that precede the noun /bare/ 'ox':

y∂h bäre /this ox/ (7)
this ox

dawit yägäzzaw bäre /an ox that Dawit bought./ (8)
Dawit bought ox

In (7) it is a determiner, whereas in (8) it indicates a relative clause. Like nouns, adjectives also have inflections for gender, case and definite.

3.2.4.The Verb Class

A verb in Amharic is any word that can be placed at the end of a sentence and which could have subject or object pronoun affixes. Consider the following examples (Getahun, 1995):

sät'-ä (he gave)
sät'-äčč (she gave)
sät'-ähu (I gave)
sät'-äh /sät'-äk (you gave, male)

sät'-äš (you gave, female)

Verbs are inflected for person, number, gender and tense. The different inflection forms of the stem /säbbär-/ 'break' is given in appendix 1.

Amharic has six tense types as shown in the table found in appendix 1. These are simple perfect tense, simple imperfect tense, present perfect tense, present imperfect tense, past perfect tense, and past imperfect tense (Baye, 1986). The simple perfect and the simple imperfect tense are morphological tense type. However, the rest tens type are formed by combining two types of main verb stems and two other auxiliary verbs. The main stems are perfect and imperfect while the auxiliary verbs are /allä/ 'exist' and /näw/ 'is'. /näbär/ 'was' which is the perfect form of the two auxiliary can also be used in the tense formation.

3.2.5.The Adverb Class

Amharic adverbs modify a verb in terms of time, position and condition (circumstance) (Getahun, 1995).

dawit tolo alɔmät'am /Dawit didn't come quickly./ (9)
Dawit quickly didn't come

The underlined word /tolo/ is an adverb that modifies the verb /alɔmät't'am/.

Adverbs are classified into primitive and derived words. Some examples of the primitive words are /gäna/ 'not yet' and /tolo/ 'quickly' while the

derived words are /kəfuňa/ 'badly' and /mənəňňa/ 'seriously'. In a language that has only a few numbers of adverbs, however, the role of the adverb can be accomplished by the noun phrase, prepositional phrase and relative clause (Baye, 1986 & Getahun, 1995).

dawit bäfət'nät wädä gondär hedä /Dawit went to Gonder in hurry./ (10)
Dawit in hurry to Gondar went

The underlined words in the above sentence describe how Dawit went to Gonder. However, the constituent consists of /bä/ and /fət'nät/, which is a combination of preposition and noun. In this case, it is a prepositional phrase that plays the adverbial function.

aster dawit wädä gondär sihed aläqäsäčč /Aster cried when Dawit went to Gonder. / (11)
Aster Dawit to Gondar going cry

The underlined constituent in (11) is a dependent sentence (subordinate). It functions as an adverbial of reason because it explains why Aster cried.

dawit wädä gondär hedä /Dawit went to Gondar./ (12)
Dawit to Gonde went

In (12), the underline constituent is prepositional phrase that performs the adverbial function.

Adverbs of position, which is referred as noun adverbs, can be used either as a noun or as an adverb depending on the context. For example:

wädä häyl täläwät'ä /It is changed into force./ (13)
to force changed

/häyl/ is used as a noun in (13).

bä-häyl yəsäral /He works strongly./ (14)
by inside works (he)

/bä-häyl/ is used as an adverb in (14).

3.2.6.The Preposition Class

Prepositions in Amharic are words that can be combined with nouns. Words like /əndä/, /wädä/, /səlä/, /lä/, and /kä/ are prepositions.

dawit kägondär əndämät'a täňňa /Dawit sleeps as he came from Gondar./(15)
Dawit from Gondar as he came sleep

dawit wädä gondär hedä /Dawit went to gondar./ (16)
Dawit to Gondar went

dawit əndä kasa ayəčäkkulm /Dawit didn't hurry like Kassa./ (17)
Dawit like Kassa din't hury

dawit səlädäkkämä marräf aläbbät /Dawit rests as he tired. / (18)
Dawit as he tired rests

All the underlined words from (15)-(18) are prepositions.

According to Baye (1986) and Getahun (1995), prepositions have the following features.

- It is placed before a noun and acts as an adverb.
- It doesn't inflect.
- No word can be derived from it.

In the modern linguistic literature, conjunction is considered as a preposition because both of them share the same features.

3.2.7.The Conjunction Class

Often words used as conjunctions and prepositions are the same (Dawkins, 1960). However, conjunctions are prefixed to verbs while prepositions are prefixed to nouns. Conjunctions are classified into coordinate and subordinate categories. Coordinate conjunction coordinate words, phrases, clause and sentences. Some of the coordinate conjunctions are /ɔnna/ 'and', /mɔ/ 'and, too, also, even, either', /nägär gɔn/ 'however', /wäyɔm/ 'or'. The subordinate conjunction used to introduce a subordinate clause, subordinating it to the principal clause of the sentence. Some examples of the subordinate conjunction are /sɔlä/ 'becouse', /kä/ 'since', /bä/ 'if'. A list of coordinate and subordinate conjunctions can be found in Dawkins (1960) along with a detailed discussion of coordinate and subordinate conjunctions.

Even though preposition and conjunctions are not differentiated in the modern work due to their similar features, they are presented separately in this study as they have different roles in a sentence.

3.3. Interjections, Determiners, Quantifiers, Modifiers, and Complements

There are some phrases or numerals that can not be grouped into the above-mentioned word classes. These phrases or numerals can be classified into five categories, including interjections, determiners, quantifier, modifiers, and complements.

3.3.1.Interjections

Like English, Amharic has many words or phrases to express such emotion as sudden surprise, pleasure, annoyance and so on. Such kinds of Amharic words do not assume grammatical function and, therefore, are not grouped into word classes. These interjections can appear as singletons by themselves outside a sentence or appear any where in a sentence. For example,

goš !

abbate mät'a goš ! or goš abbate mät'a !

A long list of Amharic interjections can be found in Dawkins (1960)

3.3.2.Determiners, Quantifiers, Modifiers and Complements

This and the following sections discuss phrasal categories and sentences in Amharic. Baye (1986) and Getahun (1995) use the terms, determiner, modifier, and complement, in parsing Amharic sentences. These are not

word categories, but are used to specify the function of a certain word or phrase in a sentence.

Determiner: a construct used to specifically point out or differentiate a certain thing. For example

dawit [bät'am ∂ndä wänd∂mu säw färi] näw (19)
Dawit very much like his brother human afraid is

/Dawit is very much shy like his brother. /

In (19), /bät'am/ 'very much' specifically points out how the subject (i.e. he) is shy and thereby labeled as a determiner.

Quantifier: a constituent used to quantify certain thing. For example,

hulät gize t∂m∂h∂r∂t bet hedä /He went to school twice./ (20)
Twice education house he went

In (20), /hulät gize/ 'twice' is used to quantify the number of times the subject (i.e. he) went to school, and is therefore labeled as a quantifier.

Modifier: a constituent that builds a phrase or a sentence. Modifiers are found preceding the phrases or the sentences they modify. The modifiers could be a word or a phrase. Some of the modifiers are adjectives, adjectival phrases; prepositional phrases, and so on. For example in (21) the verb is modified by a prepositional phrase.

bä-mäkina wädä bet hedä /he went to school by car./ (21)
by car to house he went

Here, the prepositional phrase /bä-mäkina/ 'by car' is labeled as a modifier for the VP. /wädä bet hedä/ 'went home (he)' is a sentence by itself, and the function of /bä-mäkina/ is to build the sentence by giving more information about the instrument. More details on modifiers are provided in Baye (1986) and Getahun (1995).

Complement: a construct that is used to make a certain idea complete. For example,

*kasa gäddälä	/Kassa killed. /	(22)
kasa anbäsa gäddälä	/Kassa killed a lion. /	(23)

/anɔbäsa/ 'lion', which is a noun, is a complement in (23). It is considered as a complement because it is used to make (22) the ill formed sentence /kasa gäddälä/ 'Kassa killed' complete, by way of specifying what the object of the sentence is.

3.4. Syntactic Structure of Amharic

The syntactic structure is formed by combing different words. Since Amharic word formation follows its own structure, the syntax of the language also exhibit a unique structure. The syntactic structure of Amharic sentence is SOV (Subject-Object-Verb). The modifiers in such structure generally precede the word or the phrase they modify.

* ill formed or not acceptable construction

3.4.1.Phrase

There are two types of constituents in the syntactic structure: phrase and sentence. The phrase is a constituent in the language, which is composed of the head word, the complementary and the modifier. In Amharic there are five types of phrases, including noun phrase, verb phrase, adverbial phrase, adjectival phrase, and prepositional phrase (Baye, 1986; Getahun, 1995).

3.4.2.Noun Phrase

Pronouns and proper nouns are main constituents of Amharic noun phrases.

(24)

∂ne at'änalähu /I study. /
I study

(25)

dawit bäre gäza /Dawit bought an ox. /
Dawit ox buy

The pronoun /∂ne/ 'I' in (24) and the proper noun or personal pronoun /dawit/ 'Dawit' in (25) are noun phrases. We can formulate rule 1 and rule 2 from construct (24) and (25).

Rule (1) NP → IPP

Rule (2) NP → PERP

In Amharic noun phrases are constructed from sub phrase and main phrse (Baye, 1986). The sub noun phrase (SNP) is made up of the head word and/or one more noun as complementary.

(26)

[bet]
house

[yäsar bet] (27)
thatched house

The underlined words that are illustrated above are sub-phrases. What is given in (26) consists of one head word /bet/. This head word helps to uniquely identify the phrase type. However (27) is comprised of one head word /bet/ and one more noun phrase /yäsar/ that is complementizer of the head word. Hence, we can derive the following rules from these noun phrases:

Rule (3) SNP → N

Rule (4) SNP → NP + N

A main noun phrase (MNP) is composed of a sub-phrase and some additional words or phrases as modifier.

[tɔlləq bet] (28)
big house

The main phrase given in (28) is consists of a sub phrase /bet/ and an adjective /tɔlləq/ as a modifier. It can be represented with rule (5), as shown below.

Rule (5) MNP → ADJ + SNP

[tɔlləq yäsar bet] /Yesterday's big thatched house. / (29)
big yesterday's house

[yätɔnanɔtu tɔlləq yäsar bet] /Yesterday's big thatched house. / (30)
yestrady's big thatched house

[hulät tɔlləq yäsar bet] /Two big thatched houses. / (31)
Two big thatched house

In (29), (30) and (31) main phrases are indicated in side the brackets, of which the sub phrases are underlined. A main phrase has structures, which are shown in rules (6), (7), and (8).

[tɔlləq yäsar bet]

NP [$_{Adj}$ [tɔlləq $_{SNP}$ [yäsar bet]]

Rule (6) MNP → ADJ + SNP

[yätɔnanɔtu tɔlləq yäsar bet]

$_{NP}$ [$_{NP}$ [yätɔnanɔtu] $_{Adj}$ [tɔlləq] $_{SNP}$ [yäsar bet]]

Rule (7) MNP → NP + ADJ + SNP

[hulät tɔllalɔq yäsar bet]

$_{NP}$ [$_{Quant}$ [hulät] $_{Adj}$[tɔlləq] $_{NP}$[yäsar bet]]

Rule (8) MNP → Quant + ADJ + SNP

MNP → NP + ADJ + SNP

3.4.3.Verb Phrase

In Amharic, there are action verbs, which don't take any object as complementary to form a sub verb phrase (SVP). These verbs are called action verbs with out object complement (AVO).

dawit [aläqäsä] /Dawit cries./ (32)
Dawit cries

In (32), a sub verb phrase is both underlined and enclosed with brackets.

Rule (9) SVP → AVO

However, there are some action verbs that take complementary words or phrase as object of the verb. Those verbs, which take noun phrase as complementary, are called transitive verbs (TV).

dawit [mɔsawɔn bälla] /Dawit ate his lunch./ (33)
Dawit his lunch eat

The transitive verb /bälla/ in (33) takes the NP /mɔsawɔn/ as an object.

Rule (10) SVP→ NP + TV

Sometimes transitive verbs might take two objects as complement of the verb.

dawit [läkasa mäs'ɔhaf sät'äw] /Dawit gave a book to Kasa./ (34)
Dawit to Kassa book give him

In (34), a complement PP /läkasa/ which is the indirect object precedes the direct object NP /mäs'ɔhaf/.

Rule (11) SVP→ PP + NP + TV

Even though intransitive verbs (IV) don't take NP as an object, they can take PP as complement of the verb.

dawit [wädä sɔra hedä] /Dawit went to work. / (35)
dawit to work went

In (34) the sub phrase consists of /hedä/ as head word and PP /wädä sɔra/ as complement.

Rule (12) SVP → PP + IV

Like the other phrases the main verb phrase (MVP) will be formed from sub verb phrase and some additional words or phrases as a modifier.

dawit [zare hulät bäre gäza] /Dawit buys two oxen. / (36)
Dawit today two ox buy

Rule (13) MVP → N + N + SVP

The sup phrase in (36) is modified with the nouns 'zare' and 'hulät'.

Consider the following examples.

dawit [bätolo wädä sɔ̀ra hedä] /Dawit went to work quickly. / (37)
Dawit quickly to work went

Rule (14) MVP→ ADV + SVP

dawit [kasa sɔ̀lätamämä wädä sɔ̀ra alähedäm] (38)
Dawit Kassa as sick to work didn't go

/Dawit didn't go to work, as kassa is sick. /

Rule (15) MVP→ S' + SVP

lɔ̀jočču [tɔ̀nant bämkina [wädä gonɔ̀där hedu] (39)
The boys yesterday by car to Gondar went

/The boys went to Gondar on yesterday by car. /

Rule (16) MVP→ N + PP + SVP

In (37), (38) and (39) the sub-phrases are modified with noun phrases, adverbial phrases, dependent sentences, and noun phrases along with prepositional phrases respectively. Such syntactic structures are summarized in rule (14), (15) and (16).

3.4.4.Adjectival Phrase

Like other phrases that we have discussed above, adjectival phrases can be made from sub and main phrases. The sub phrase consists of either the head adjective alone or the head word with other constituent.

dawit [sännäf] näw /Dawit is lazy. / (40)
Dawit lazy is

dawit [säw färi] näw /Dawit is afraied of human being./ (41)
Dawit human afraid is

The underlined words in (40) and (41) are sub adjectival phrases (SADJP). They include an adjective /sännäf/ in sentence (40) and a noun and an adjective /säw färi/ in (41). The structures are summarized in rule (17) and (18).

Rule (17) SADJP → ADJ

Rule (18) SADJP → N + ADJ

The main adjective phrase (MADJP) is comprised of sub phrase and prepositional phrases, as shown in (42). The structure is formulated in rule (19).

dawit [ɔndäwändɔmu säw färi] näw (42)
Dawit like his brother human afraid is
/Dawit is lazy like his brother. /

Rule (19) MADJP → PP + SADJP

Adjectival phrase can be constructed with a sentence and an adjective, as shown in (43).

dawit [käwändɔmmu yɔbälɔt' säw färi] näw (43)
Dawit from his brother greater human afraid is
/Dawit is shy than his brother. /

The structure of (43) and the rule derived is given below.

dawit MADJP[S'[käwändɔmu yɔbälɔt] SADJP[säwu färi]] näw

Rule (20) MADJP → S' + SADJP

Sometimes an adjectival phrase may be preceded by a determiner, which is represented in rule (21).

dawit [bät'am ∂ndä wänd∂mu säw färi] näw

Rule (21) MADJP → Determiner + PP + SADJP

3.4.5.Prepositional Phrase

In order to construct a sub prepositional phrase (SPP), it requires at least an NP as the complement in addition to the preposition.

dawit [läkassa] näggäräw /Dawit told (to) Kassa./ (44)
Dawit to Kassa tells

dawit [wädä bet] head /Dawit went (to) home. / (45)
Dawit to home went

In (44) and (45), the PPs in brackets consist of a preposition (i.e. /wädä/ or /lä/) and an NP (i.e. /bet/ or /kasa/) respectively. The structure is formalized as rule (22).

Rule (22) SPP→ PRE + N

There are cases where a preposition appears after an NP, as shown in (46) below. It is called post position (POP).

dawit [bet w∂st'] allä /There is Dawit inside the house. / (46)
Dawit house inside there is

The preposition /w∂st'/ comes after the NP /bet/.

Rule (23) SPP→ N + POP

Sometimes a PP is preceded by another preposition as a complement.

dawit [käwänd∂mu gar] head /Dawit went together with his brother. / (47)
Dawit from his brother with went

The sub-phrase, which is underlined in sentence (47), takes a preposition /kä/ as a complement.

Rule (24) SPP→ PP + POP

In order to construct a main PP (MPP), the sub phrase will be modified with another preposition.

[käangät bälay] /above the neck. / (48)
from neck above

[ɔndä qäbäro bägudgʷad] / like a fox in a hole. / (49)
like a fox in the hole

The underlined words in (48) and (49) are main phrases, which are modified by PPs that precede them.

Rule (25) MPP→ PP + SPP

3.4.6.Adverbial phrase

In adverbial phrases, the sub phrase consists of only a head word, which is an adverb.

dawit [kɔfuňňa] tammämä /Dawit is seriously sick. / (50)
Dawit seriously sick

The underlined word in brackets is a sub phrase consisting of an adjective.

Rule (26) SADVP→ ADV

The main phrase in an adverbial phrase is constructed by modifying the sub phrase with a PP, as shown in sentence (51) and Rule (27).

dawit [ɔndä abbatu kɔfuñña] tammämä (51)
Dawit like his father seriously sick
/Dawit is seriously sick like his father. /

/ɔndä abbatu/ is a PP that modifies the sub phrase /kɔfuñña/.

Rule (27) MADVP→ PP + SADVP

The main phrase could be further modified by preceding the PP with a determiner.

dawit [bät'am ɔndä abbatu käfuñña] tammämä (52)
Dawit very much like his father seriously sick
/Dawit, very much like his father, is seriously sick. /

The determiner /bät'am/ is inserted before the PP /ɔndä abatu/ in (52).

Rule (28) MADVP → Determiner + PP + SADVP

3.5. Sentence in Amharic

A sentence in Amharic is a combination of two types of phrases: noun phrases and verb phrases as shown in (53).

S [NP [Quant [hulät] Adj[tɔllalɔq] N[lɔjočč]]
MVP [N [tɔnanɔt] SVP [MPP [PP [bɔmɔkina] SPP [wädä gonɔdär] IV [hedu]]]] (53)

Rule (29) S→ NP + VP

Amharic sentences are classified into simple and complex sentences. When the sentence only has one verb, it is called a simple sentence (Getahun, 1995). For example,

S [NP [wänbäru] TV [täsäbbärä]] (54)

S [NP [dawit] VP [PP [läwändɔmu] NP [gänzäb] VP [lakällät]]] (55)

S [NP [dawit] VP [PP [käwänd∂mu] POP [gar]] IV [hedä]]] (56)

Complex sentences contain complex noun phrases and/or complex verb phrases. A complex noun phrases comprise of dependent sentence as complement and sub noun phrase (Getahun, 1995).

NP [S' [NP [dawit] VP [CV yäsäbäräw]] SNP [yäšäk∂la b∂r∂č'∂qo]] (57)

NP [S' [NP [dawit] VP [NP [t∂nant] VP [CV yäsäbäräw]]
SNP [yäšäk∂la b∂r∂č'∂qo]] (58)

In (57) and (58) the subordinate or the dependent sentence is made up of a noun phrase and complementized verb.

Rule (30) MNP→ S' + SNP

A verb phrase is considered as a complex phrase when it comprises more than one verb, or one or more sentences. Like a complex noun phrases, the sentences are dependent (Getahun, 1995).

SVP[S' [NP [dawit] VP [b∂r∂č'∂qo ∂ndsäbbärä]] TV [sämmačč]] (59)

The dependent sentence found in (59) uses as a complement of the transitive verb. Even though the example in (59) is a sub phrase, it is a complex verb phrase as the transitive verb is complemented by a sentence.

Rule (31) SVP→ S' + TV

In Amharic, there are times when we find two dependent sentences within one verb phrase (Getahun, 1995). For example,

MVP [S' [NP [dawit] VP [PP [kägondär] VP [∂ndämät'a]]]

SVP[S'[NP[aläm]VP[PP[wädä nazz∂ret]VP[∂ndähedačč]]]VP[sämma]] (60)

The two dependent sentences in (60) are the subordinate sentence that modify the sub verb phrase and the bound sentence that complements the transitive verb.

Rule (32) MVP→ S' + SVP

In the above case, the NP in the subordinate sentence modifying the sub verb phrase can be deleted because it is already described in the VP /∂n∂dämät'a/. For example:

VP [S' [VP [PP [kägondär] VP [∂ndämät'a]]]

VP[S[NP[aläm]VP[PP[wädä nazz∂ret]VP[∂ndähedačč]]]TV[sämma]](61)

Rule (33) S'→ PP + VP

It is also possible to construct complex noun phrases and complex verb phrases by using complex adjectival phrases. An adjectival phrase becomes complex whenever it contains a relative clause (Getahun, 1995).

MADJP[S' [NP [ ] VP [PP [käaläm] VP [yäbällät'äčč]]] ADJ [kon∂jo]] (62)

What is given in (62) is complex adjectival phrase that consists of a sentence and adjective.

Rule (34) MADJP→ S' + ADJ

As a complex adjectival clause is not a complex sentence by itself, it must be attached to the noun phrase or the verb phrase (Getahun, 1995).

[käaläm yäbällät'äčč konəjo] ləj kägondär mät'ačč (63)

aləmaz [käaläm yäbällätäčč konəjo] näčč (64)

The complex adjective in brackets attaches itself to the noun phrase /ləj/ in (63) and verb phrase /näčč/ in (64).

As we stated earlier, a complex sentence can be comprised of a complex noun phrase.

S[NP[S'[N[dawit] CV[yägäzzaw]] N[mäkina]]
SVP [N [zare] TV [tägälläbät'ä]]] (65)

For example, (65) is a complex sentence consisting of one sub verb phrase and one complex noun phrase.

The second alternative to construct a complex sentence is by making the verb phrase complex.

S [NP [dawit]
SVP[S' [NP [aläm] VP [NP [bet] CV [əndäsäračč]] TV [sämətowal]] (66)

For example, the complex sentence given in (66) consists of one complex verb phrase and one sub noun phrase.

The last alternative is to make both phrases complex.

S [NP[S' [NP [ ] SVP [PP [kägondär] CV [yämät'ačəw]]] N [ləj]]
SVP[S' [NP [dawit] CV [əndäwädädat]] TV [awäqäčč]]] (67)

The illustration given in (67) contains both complex noun phrases and complex verb phrases.

Baye (1986) classifies Amharic sentences into declarative, interrogative, negative and imperative sentences. However, in this study we only examined the declarative sentence.

3.6. Amharic Structural Ambiguity

Structural Ambiguity is the most common type of ambiguity in Amharic sentences. Structural ambiguity occurs when a constituent of a structure has more than one possible position. Getahun (2002) identified some sources for structural ambiguities, including locative genitive, noun phrases with adjectival modifiers, scope of determiner, conjoined structures, different relativized position, and different deep structures of PP's. Based on Getahun's work (2002), which is the only source that is found relevant to our research problem, we summarize the following categories of sources of structural ambiguities.

1) Locative genitive

Locative genitive becomes structurally ambiguous whenever the genitive noun phrase refers either only the next word only or the whole noun phrase.

1. yä – abäša tarik astämari (68)
   of – Abyssinia – history teacher

(a)'A person who teaches Abyssinian history.'
(b) 'An Abyssinian who teaches history.'

There is ambiguity in sentence (68) because the constituent /tarik/ 'history' can be grouped in more than one way such as 2a and 2b.

2a. NP [NP [yä – abäša tarik] NP [astämari]] 'Abyssinian history teacher'

2b. NP [yä – abäša [Ñ [NP [tarik] NP [astämari]]] 'An Abyssinian who teaches history'

Thus, /tarik/ 'history' can occurs as the head of the genitive NP / yä – abäša tarik/ or as complement of /astämari/ 'teacher' in /tarik astämari/ 'history teacher'.

The above examples illustrate that genitive NPs are one source of structural ambiguity, for every genitive NP of the type: [yä NP NP NP] can have either of the following realization:

a. [yä NP_1 NP_2] NP_3]
b. [yä NP_1 [NP_2 NP_3]

In case of (a) NP_3 is considered as head of the phrase while NP_1 and NP_2 are considered as modifier of the head. The modifiers form rule 35

Rule (35) NP→ GNP + NP_2

And the main noun phrase would be:

Rule (36) MNP→ NP + NP_3

In order to realize the second (type b) structure, the second NP (NP_2) must be genitive NP.

(b). [yä NP_1 [yä NP_2 NP_3]

Hence, NP_3 is considered as head of the phrase but NP_2 is its complements while NP_1 its modifier.

Rule (37) SNP→ GNP + NP_3

Rule (38) MNP→ GNP + SNP

2) Noun phrases with adjectival modifiers.

fät't'an färäs galabi (69)
fast horse rider

(a) One who rides a fast horse.
(b) A fast person who rides a horse.

In (69), the adjective /fät't'an/ 'fast' can modify /färäs/ 'horse' or /färäs galabi/ 'horse rider'. In the former case, the phrase has the following structure.

NP [NP2 [fät't'an färäs] NP1 [galabi]] 'One who rides a fast horse'

Rule (39) NP→ NP_2 + NP_1

In the latter case, if the adjective modifies the noun phrase /färäs galabi/ the structure becomes:

NP [ADJ [fät't'an] NP [färäs galabi]] 'A fast person who rides a horse'

Rule (40) NP→ADJ + NP

3) The scope of Determiner

The scope in which a determiner relates to the constituents in adjective phrase can cause different interpretations.

(70)

1. bät'am tɔllɔq säw färi
very big person fearful

(a) One who is afraid of a very big man.
(b) One who is afraid of a big man very much

In the constituent [bät'am ADJ NP_1 NP_2], the head of NP_2 is a participle, /bat'am/ 'very' can modify [$AdjP_1$], as in (2a).

2a. SADJP [NP [ADJP [bät'am tɔllɔq] säw] färi]] 'One who is afraid of a very big man.'

Rule (41) MNP→ ADJP + N

Rule (42) SADJP→ MNP + ADJ

When /bät'am/ 'very' is considered as the determiner of the entire structure [ADJP NP1 NP2], it is interpreted as (2b).

2b. SADJP [bät'am [ADJP [NP [tɔillɔq säw] färi]] 'One who is afraid of a big man very much.'

Rule (43) ADJP→ NP + ADJ

Rule (44) SADJP→ Determiner + ADJP

The ambiguity in constituent (70) is resulted from the scope of the determiner /bät'am /, which may specify the entire NP or the ADJP in it.

4) Conjoined structures

Take the following conjoined structures as illustration.

(71)

1. wät't'at wand-očč ɔnna set-očč
young man - pl and women-pl

(a) young boy and young girls
(b) young boys and women

When the adjective /wät'at/ modifies the entire phrase, we have the following structure:

NP [wät't'at Ñ [wand-očč ∂nna set-očč]] 'Young boy and young girls'

Rule (45) NP→ ADJ + [NP_1 + NP_2]

But if the adjective only modifies the first conjunct, we will get the following structure:

NP [NP [wät't'at wänd–očč] ∂nna NP [set–očč]] 'Young boys and women'

Rule (46) NP→ NP1 + NP2

Therefore, a conjoined NP can have either of the following representation and the associated readings.

(a) [Modifier [NP_1 ∂nna] NP2]]
(b) [Modifier NP_1] ∂nna [NP_2]]

5) Different relativized position

It is another cause of a sentence's having two different readings.

anbäsa yä – gäddäl–ä–w gäbäre (72).
lion comp killed-3mss-3mso farmer

(a) A lion that has killed the farmer
(b) The farmer that killed the lion.

Two possible interpretations of sentence (72) are illustrated in Figure 3.1.

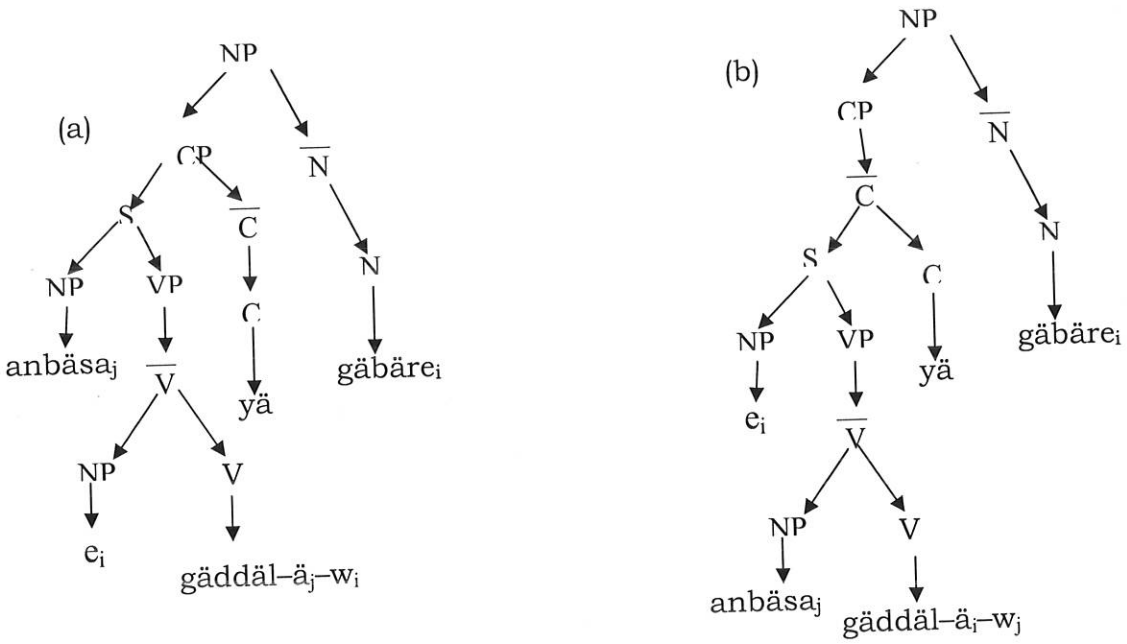


Figure 3.1 Different Relativised Position

As shown in Figure 3.1, the relativised position is different between reading (a) and reading (b), since it is a complement that is relativized in (a), and a subject that is relativzed in (b).

6) Different deep structures of PP's

The last category of ambiguity relates to structures derived from different deep structures of PP's with a simple NP or a genitive NP complement. For example,

kä – fäqadu yə – bält' yä – getaččäw lə̂j gobäz näw (73)
from – Fedadu 3ms–better of - Getachew son cleaver is

(a) Getachew's son is cleverer than Fekadu.
(b) Getachew's son is cleverer than Fekadu's (son)

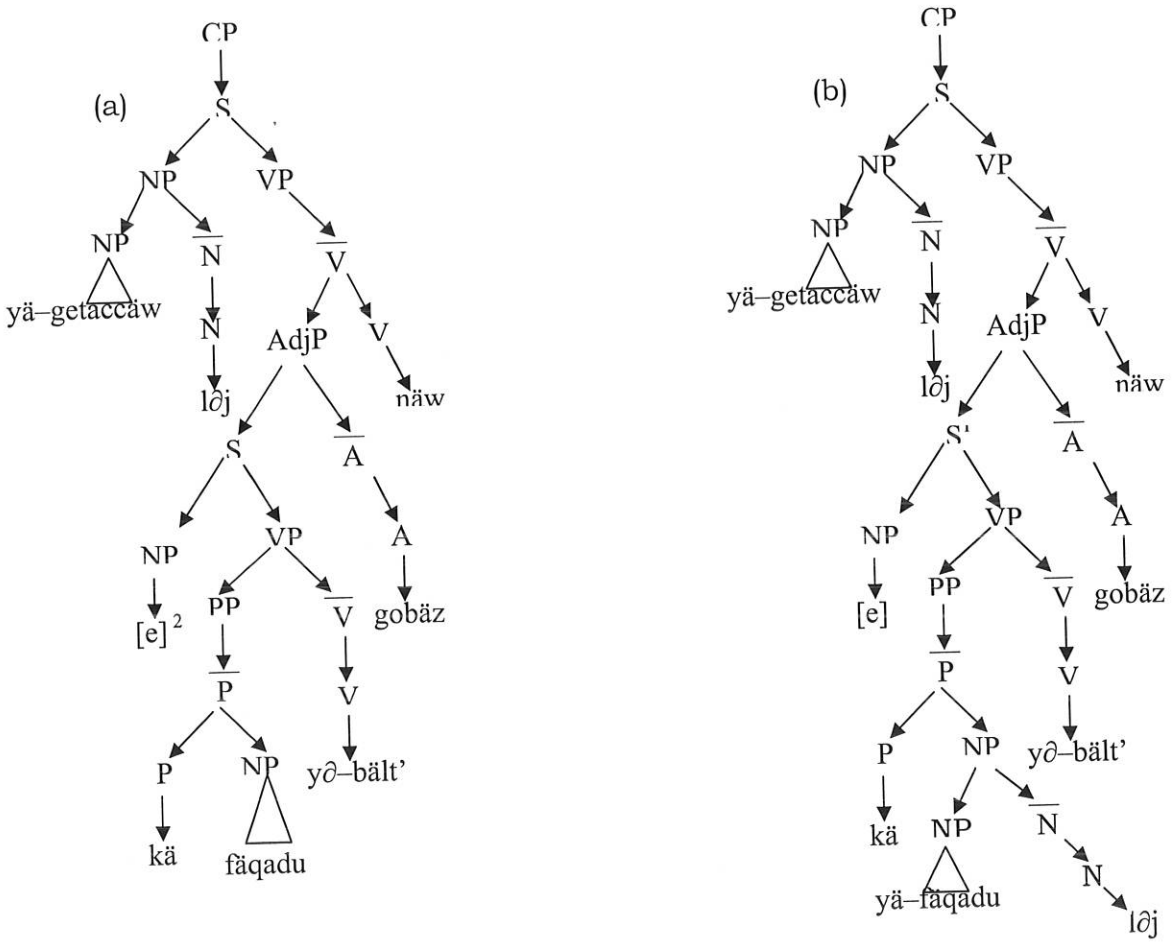


Figure 3.2 Different Structure of Preposition Attachment

Figure 3.2 exhibits two possible structures of sentence (73). The preposition /ka/ takes a simple NP complement in (a) whereas it takes an NP complement consisting of a head and a genitive NP determiner in (b).

CHAPTER FOUR

DATA PREPARATION AND MORPHOLOGICAL ANALYSIS

This chapter describes data preparation and morphological analysis to support the development of a parser. It first introduces the data preparation process in detail; then describes the algorithm and resources for morphological analysis; and finally presents a simple morphological analyzer developed for this study.

4.1. Data Preparation

4.1.1.The Sample Sentences

For the purpose of this study, 400 Amharic sentences were collected from three widely used grammar books of the Amharic language. Among them, 50 were structurally ambiguous sentences. They were selected from the following books: "yä amarigna säwasäw" /Amharic Grammar/ by Baye (1987), "yä amariña säwasäw bäqälal aqäraräb" /Amharic Grammar simplified Approach/ by Getahun (1995) and "Amharic for Beginners" by Frydenlund and Svensen (1998). All of the books were prepared with the purpose of serving as references for teaching Amharic language at tertiary and secondary levels, respectively. The books were selected as our references because they claimed to cover all the grammatical rules of Amharic. Ambiguous sentences were constructed based on patterns discovered by Getahun. The sentences were selected randomly but with careful judgment by the researcher.

The sample sentences were first transcribed into Phonemic symbols according to the standard set for representing Amharic fidels (Appendix 2). They were then tagged manually in order to develop a HMM (Hidden Marcove Model) tagger. As the tagger only accepts text files (ANSII code), some characters in the transcribed sentences would be expected to reflect some changes. Amharic Phonemic symbols listed in the first column of Table 4.1 are not text supported. For example, the symbol '*ə*' stands for palatal sound; '**čč**' stands for a more or less unique character of Amharic, which represents a velar explosive sound and '**ň**' has already been used for representing the usual nasal sound '**n**'.

Original Phonemic symbols	Transcribed Phonemic symbols
'*ə*'	'**?**'
'**č**'	'**ç**'
'**čč**'	'**çç**'
'**ň**'	'**ñ**'
'**š**'	'**sh**'
'**ž**'	'**zs**'

Table 4.1 Mapping between Amharic Character and Latin Characters

The text file would change the palatal symbol, which is '*ə*' while the rest are changed deliberately by the researcher with some other symbols in the right side of the table that are text and Microsoft visual studio supported. The parses of some sample sentences were extracted from Baye (1987) and Getahun (1995) and other sentences were parsed manually by the researcher and two students from Ethiopian Language

Studies to identify phrase structure rules of the language. The parses were finally revised based on comments and suggestions from the linguistic advisor of the thesis. Disambiguation rules were developed based on 320 sentences (80% of the sample sentences). These sentences were selected randomly as the training set and the remaining 80 sentences were used as the test set (Appendix 3).

4.1.2. Pre-processing

A pre-processor prepares a text file for the morphological analysis. It incorporates a sentence and word splitter, which operates by reading a text from a file, determining sentence boundaries of the text and verticalising the sentences in the text. The sentence extraction routine is described in Figure 4.1.

1. *Initialize a variable, input String, with all the text in a file;*
2. *Initialize Sentence Counter variable with 0;*

3. *Do until the end of a file*
   Do until the end of a sentence
   i. *Starting from the first character in Input String, extract all the characters up to and including the hash mark (#), which is a sentence delimiter.*
   ii. *Assign the extracted substring to a variable, input sentence.*
   iii. *Increase the Sentence Counter by one*
   iv. *Display the extracted sentence.*

Figure 4.1 Sentence Extraction Routine

A verticalised sentence is further processed, during which word boundaries are identified and each word of the sentence is verticalised. For example, the verticalsed result of the sentence:

dawit kasa yäyazäwn addis borsa adänäqä #

is as follows:

dawit
kasa
yäyazäwn
addis
borsa
adänäqä
#

The word extraction routine is illustrated in Figure 4.2. After words are lexicalized they are passed to a morphological analyser.

1. *Initialize an array variable to hold multiple words*
2. *Initialize a string variable with the text in a file*
3. *Do until a sentence delimiter is found*
    a. *Do until a word delimiter is found*
        i. *Extract one character*
    b. *Put a word in the array declared.*
    c. *Write the extracted words to an output file*
4. *Perform stemming on the extracted words.*

Figure 4.2 Word Extraction Routine

4.2. The Morphological Analysis

Morphological analysis is one of the most essential NLP components in the development of a sentence parser. There are a number of approaches

employed in computational morphology. As discussed in Kazakov and Munandhar (2000), some of these approaches are based on concepts in automata theory, probability, principle of analogy, and information theory. Kazakov and Munanadhar (2000) broadly classify morphological analysis methods into rule-based and corpus-based approaches.

A rule-based approach is adopted in this study. The rules were compiled based on the theory of morphology developed by Amharic language experts. The approach incorporate sophisticated linguistic theories into the process of computational morphology. Therefore, the system is expected to be more accurate and produce a better result than those applying little or no linguistic theories.

The morphological analyzer runs at the back-end of the parser. Figure 4.3 describes the morphological analysis algorithm. The algorithm was developed based on findings from previous studies on the Amharic language.

1. Call the stemmer Function for each word in the sentence
2. Get the stem of each word
3. Call the POS tagger
4. Get the POS tag and the prefix, infix and suffix of a word
5. Call the updating function to update the category output of the tagger

Figure 4.3 An algorithm for Morphological Analysis

The morphological analysis algorithm incorporates three components, including a stemmer, a part of speech tagger and a category updating function.

4.2.1. The stemmer

The function of a stemmer is to extract a free morpheme from a word. Then the free morpheme stem passes to a morphological part-of-speech tagger, which is the second component of the morphological analyzer.

Amharic belongs to the inflectional language, in which most of the words consist of a stem and an affix. Therefore, it is highly recommended to incorporate a stemmer in the system to increase the performance of a tagger. For instance, a word consisting of a single morpheme e.g., /bäre/ 'an ox', as mentioned previously, is very likely to be attached with affixes such as /ɔndä-/, /-wočč/, /-wočču/, /-wočče/. If the word was not stemmed, the lexicon of a tagger would have to list all the variants of /bäre/, including /ɔndäbäre/, /bärewočč/, /bärewočču/, and /bärewočče/. With the help of a stemmer, there would be only one morpheme (stem) /bare/ 'ox'. As a result, the lexicon only needs to store the stem entry (i.e., /bäre/).

Another potential advantage of performing word stemming is to reduce unknown words to a tagger. For instance, a tagger that can recognize the

word /yazä/ '(he) holds' as a verb 'V' may tag /yäyazäwun/ 'that (he) holds' as an unknown category 'UNC', which is actually an inflected form of the former word.

The stemmer component incorporated in this study is based on algorithm developed by Nega and Willett (2001). It was the intention of the researchers to incorporate Nega's and Willett's stemmer as it is. However, the text they used is transcribed using Serra and also there is incompatibility with our system as they are used Pascal programming language.

The stemmer component (Nega Alemayhu and Peter Willett, 2001) incorporated into our system is context-sensitive. Before it removes an affix string, it checks whether the string is part of the stem by considering the remaining string. There may be ambiguities in stemming a word. Some strings that seem like an affix may not function as prefix or suffix in a word. For example, 'pre' from 'present' and 'y' from 'lady' in English. In Amharic, removing the prefix /lä/ 'to' from /lämläm/ 'green' and the suffix /ta/ from /amtata/ '(he) makes confusion' would generate over-stemming result.

In order to make the stemmer context sensitive, Nega and Willet developed two action codes and five conditional rules. The two contexts sensitive actions are:

- A1: do not perform any affix removal;
- A2: remove a shorter form affix;

The five conditional rules are:

- Cl: list characters that follow a specified affix. In the case of prefixing, this condition helps to find a place where a vowel elision occurs at morpheme boundaries, and to protect the terminal consonants from being removed.
- C2: list characters that precede a specified affix. The condition helps to protect the terminal consonant from being removed, for it is not a genuine suffix.
- C3: list consonant characters of words whose beginning characters belong to the list of prefix. This condition helps to protect the removal of non-genuine affixes from specific words and their variants.
- C4: consider a minimum stem-length condition for a specified affix. The stem length involves both vowels and consonants. The condition is used to reduce over-stemming.
- C5: The condition combines any of the above.

Affixes that do not have any associated condition(s) are removed; otherwise, conditions C1-C5 are checked and one of the appropriate actions, A1 or A2, would be taken. Some examples of the context-sensitive rules that were developed are listed in table 4.2.

Affix	Context	Actions	Sample words
'ya'-	C1: 'wuna' Else	A1 A2:remove 'y'	yawuna yat'äbkutn, yamät'ahutn
'la'-	C1: 'b' Else	A1 A2: remove 'l'	lab labäbä, lamät'a, lamt'a
'as'-, 'at'-, 'st'-, 'ay'-	C1: vowels (ä, u, i, a, e, ∂, o)	A1	asa, ato, sto, ayä,ayu
-'at'	C2: l∂m, säb, s∂r	A1	l∂mat, säbat, s∂rat,
'kä'-	C3: klkl, ktm, k'sk's	A1	käläkälä, kätäma, qäsäqäsä
'bä'-	C3: bll	A1	bälläw, bälla,
-'n'	C3: hs'n, c'kn, mn, wn	A1	h∂s'an, c'∂k'un, ∂mun, ∂wun
-'nät'	C3: mnt, wnt	A2: remove 'ät'	∂mnät, ∂wunät
-'at'	C4: length < 5	A2: remove 't'	t∂gat,
-'at', -'t'	C5:[(C2 a vowel) or (length < 5)or (C3: hlt, sst, rt, mst, sbt, smnt)]	A1	s∂rat, tgat, hulät, sost, art, amst, säbat, s∂mnt

Table 4.2 Example of Context-Sensitive Rules

When checking for condition three (C3), only the root (consonant) of a word is considered, for different forms of the same word vary in their vowels. For instance, verbs in the following sentences differ in their vowels only: /qäsäqäsä/ 'he awake', /qəsqäsa/ 'awakeness', /qäsäqäso/ 'after he awake', /qäsqəsa/ 'after she awake', /qäsäqäsu/ 'They awake'. Such a method allows detecting many words with similar leading strings. The stemmer is comprised of three subroutine components: prefix remover subroutine, infix remover subroutine, and suffix remover subroutine. An algorithm shown in figure 4.4 depicts how the subroutines interact with one another.

1. Do until end of a sentence
- *a. Get a word*
- *b. Pass it to prefix removal subroutine component*
- *c. Get a string from the prefix remover subroutine component*
- *d. Pass the string to infix remover subroutine component*
- *e. Get a string from the infix remover subroutine component*
- *f. Pass the string to suffix remover subroutine component*
- *g. Get the stem of a word*

2. Tag the word with a part-of-speech.

Figure 4.4 The Stemming Algorithm

The Prefix removal subroutine algorithm described in figure 4.5 removes the prefix of a word by referring to a list of prefixes and context sensitive rules. For example, the output of the following verticalised words with their prefixes being removed is listed as follows:

INPUT	OUTPUT
dawit	dawit
kasa	kasa
yäyazäwn	yazäwun
addis	addis
borsa	borsa
adänäqä	adänäqä
#	#

1. Get a WORD and count the number of radicals (or consonant) in WORD;
2. If the number of radicals< 3 THEN stop and return WORD;
3. If length (WORD) <= (MAXPP+2) THEN assign the first length (WORD)-2 Characters to a temporary prefix, TEMPP;

ELSE assign the first MAXPL characters to TEMPP (were MAXPL is the length of the longest prefix in the list);

4. Find TEMPP in the prefix list;
5. IF found THEN
IF context sensitive THEN check the context-sensitive rules c1-c5, apply a1 or a2 and GO TO 7

ELSE GO TO 6

ELSE

IF length (TEMPP)>1 THEN assign length (TEMPP)-1 characters to

TEMPP and GO TO 4

ELSE return WORD

6. Remove Prefix

7. IF number of radicals of WORD >3 or length (WORD) > 4 THEN GO TO 1 ELSE GO TO 2

Figure 4.5 The Prefix Removal algorithm (Developed by Nega and Willett)

The infix removal subroutine algorithm given in figure 4.6 removes the infix of reiterative verbs like /täsäbabbärä/ 'broken again and again'.

1. Get the word from prefix removal component and cheek whether the third and the fifth consonants are identical;

2. If they are identical, remove the third and the fourth characters;
3. Pass the word to the suffix removal function

Figure 4.6 The Infix Removal Algorithm Developed by Nega and Willett

Iterative verbs are verbs that double the initial character ($s_1b_2b_2r_3$) of the two identical consonants found in the middle and place a vowel 'a' between the doubled consonants ($s_1b_2ab_2b_2r_3$) (Nega and Willett). The prefix removal function first removes the prefix /tä/ and passes it to the infix function. The infix removal function then returns the basic verb form /säbbärä/ '(he) broke' by removing the first consonant of the doubled consonants along with the fourth vowel, which is /ba/. This basic verb form will be submitted to the suffix removal subroutine, given in figure 4.7.

The suffix removal routine removes the suffix and passes the final stem to the part of speech tagger. For example, for the previous output of the prefix routine, the suffix routine generates the following output:

INPUT	OUTPUT
dawit	dawit
kasa	kas
yazäwn	yaz
addis	addis
borsa	borsa
adänäqä	adänäq
#	#

1. Get WORD and count the number of radicals (or consonant)
2. IF number of radicals< 3 THEN stop and return WORD
3. IF length (WORD) < (MAXSL+2) THEN assign the last length (WORD)-2 Character to a temporary suffix, TEMPS
ELSE assign the last MAXSL characters to TEMPS (were MAXSL is the length of the longest suffix in the list)
4. Find TEMPS in the suffix list
5. IF found THEN
IF the ending requires recoding THEN recode and GOTO 1
ELSE
IF context sensitive THEN checks the context-sensitive rules c1-c5, apply a1 or a2 and GO TO 7
ELSE GO TO 6
ELSE
IF length (TEMPS)>1 THEN assign length (TEMPS)-1 characters to TEMPS and GO TO 4
ELSE return WORD
6. Remove suffix
7. IF number of radicals of WORD >2 or length THEN GO TO 1
ELSE GO TO 2

Figure 4.7 the Suffix Removal Algorithm Developed by Nega and Willett

4.2.2.The Part-of-Speech Tagger

It is not an easy task to find an appropriate tagger that can be used as the back-end of the parser, for there is only one part of speech (POS) tagger developed by Mesfin (2001) that is publicly available. In his study, he used a tag set of 24 tags (categories) including unknown category. To handle unknown words and to identify the category of each word, he used the Viterbi algorithm. The prototype tagger was developed using the Stochastic Hidden Markov Model approach. It relies on four tables in a

database, namely word code table, category code table, lexical probability table, and transitional probability table. The performance of the tagger on a small sample Amharic text (a page long), was high, reaching 97% on the training set and approximately 90% on the test set.

In this study, it was the intention of the researchers to adopt Mesfin's tagger. The tagger was developed at the word level. It requires the system to include all forms of a root word in the lexicon, which entails a large vocabulary. Accordingly, inflectional language such as Amharic would consume a large amount of space. The parts of speech identified were sometimes found to be inconsistent with the parser. The aforementioned problems were addressed in a modified tagger, which was incorporated into the system.

The required format of input to the tagger was different from the output of the stemmer selected in this study. Thus, the content of the database used by the tagger was replaced with new data. The word code table was changed into a stem code table; the probability values of the lexical and transitional tables were also recalculated using new data from the stem code table. After making the above changes on Mesfine's tagger, it is incorporated into the system. The changes made to each of these tables and the functions that they perform are discussed in detail in the next section.

4.2.2.1. Category Code Table

The category code table, as shown in table 4.3, stores the stem categories (POS) as well as the corresponding category code. Totally, 31 categories were identified in this study (appendix 4). The 30th and the 31st tags stand for punctuation marks and unidentified words respectively. Most of the categories (POS) identified by Mesfine were changed by new once and the researchers believe they are as per the grammar rule of the language.

Category Code	Category
1	IPP
⋮	⋮
29	DR

Table 4.3 Category Code Table

4.2.2.2. Stem Code Table

The stem code table, as shown in table 4.4, was used to store stems extracted from the sample texts. Each stem was assigned with a unique in sequence. The table contained 972 stems extracted from 320 sentences. The tagger can only recognize any stems that are listed in the stem code table.

Stem Code	Stem
1	abäb
⋮	⋮
972	zär

Table 4.4 Stem Code Table

4.2.2.3. Lexical Probabilities Table

This table stores the probabilities of a stem given categories (tags) of each stem in a given corpora. The probability was written as P(stem\tag) or P (word\category) or in short as $P(S_i \backslash C_i)$. For instance, P(t'älla\CN) denotes the (lexical) probability of /t'älla/, a common drink, being a concrete noun, and P(t'älla\TV) denotes the (lexical) probability of /t'älla/ 'hate' being a transit verb. The lexical generation probability was estimated simply by counting the number of occurrence of each word for a word category. Mathematically, the probability was computed as:

$$p\left(\frac{S_i}{C_i}\right) = \frac{number\ of\ times\ S_i\ appears\ in\ category\ C_i}{total\ number\ of\ stems\ with\ category\ C_i} \quad (1)$$

Such probability values in the lexical tables were calculated using new data entered into the stem code table. The lexical probability table was shown in table 4.5.

Stem Code \ Category Code	IPP	DDP	----- ----- -----	CC	DR
1	0.010	0.012	----- ----- -----	0.022	0.036
⋮	⋮	⋮		⋮	⋮
972	0.021	0.013	----- ----- -----	0.028	0.014

Table 4.5 Lexical Probability Table

4.2.2.4. Transition Probabilities Table

Transition probabilities are denoted by P ($C_i \backslash C_{i-1} \ldots C_n$) and refer to the probability of a category given one or more previous category. For instance, the probability of a noun to be preceded by an adjective is given by P (C_i = Noun$\backslash C_{i-1}$ = Adjective). Depending on the value of n (which tells us the maximum number of categories being considered), we can have bigram (n = 2), trigram (n = 3) or, in general, an n-gram transitional probabilities. If (n = 2), the model is a bigram model and it is denoted by P ($C_i \backslash C_{i-1}$). If (n = 3), the model is a trigram model and it is denoted by P ($C_i \backslash C_{i-1}C_{i-2}$). These models assume that the probability of the occurrence of a particular category depends solely on the one or more categories immediately preceding it. In practice, given a database of texts tagged with parts of speech, the bi-gram or transitional probabilities can be estimated simply by counting the number of times each pair of categories

occurs compared to individual category counts. Mathematically, this is written as:

$$P\left(\frac{C_i}{C_{i-1}}\right) = \frac{Count\ of\ the\ number\ of\ times\ C_i\ and\ C_{i-1}\ occurs\ together\ in\ the\ corpus}{Number\ of\ times\ C_{i-1}\ occur\ in\ the\ copus} \quad (2)$$

Such values in the transitional probability table were calculated using the categorical sequence of the data in the stem code table. The transitional probability table is shown in table 4.6.

Category Code / **Category Code**	IPP	DDP	----- ----- -----	DR	Ø
IPP	0.013	0.003	----- ----- -----	0.012	0.062
⋮	⋮	⋮		⋮	⋮
Ø	0.024	0.022	----- ----- -----	0.14	0.034

Table 4.6 Transition Probability Table

After a word is stemmed, the tagger starts to tag the stem of a word by executing the Viterbi algorithm. The algorithm consists of three steps: initialization, iteration, and sequence identification, which is illustrated in Figure 4.8.

Given word sequence $W_1, ..., W_T$, lexical categories $L_1, ..., L_N$, lexical probabilities $PROB\left(W_i / L_i\right)$, and bigram probabilities $PROB\left(L_i / L_{i-1}\right)$, find the most likely sequence of lexical categories $C_1, ..., C_T$ for the word sequence. Where L_i is a word category that is preceded by (L_{i-1}) word category.

Initialization Step

For $i = 1$ *to* N *do*

$SEQSCORE(i,1) = PROB\left(W_1 / L_1\right) \times PROB\left(L_1 / \emptyset\right)$, where L_1 is a word category at an initial position that is not preceded by any word $(\emptyset)$

$BACKPTER(i,1) = 0$

Iteration Step

For $t = 2$ *to* T

For $i = 1$ *to* N

$$SEQSCORE(i,t) = MAX_{j=1,N}\left(\begin{matrix} SEQSCORE(j,t-1) \times \\ PROB\left(L_i / L_j\right) \end{matrix}\right) \times PROB\left(W_i / L_i\right)$$

$BACKPTR(i,1) =$ *index of* j *that gave the* max *above*

Sequence Identification Step

$C(T) = I$ *that* max *imizes* $SEQSCORE(i,T)$

For $i = T - 1$ *to* 1 *do*

$C(i) = BACKPTR(C(i+1)\, i+1)$

Figure 4.8 the Viterbi Algorithm from (Allen, 1995)

The algorithm helps to track the probability of the best sequence leading to each possible category at each position using an $(N \times T)$ array, where N is the number of lexical categories $(L_1, ... L_N)$ and T is the number of words in the sentence $(W_1, ... W_T)$. The array $SEQSCORE(n,t)$, records the probability for the best sequence up to position t that ends with a word

in category L_n. To record the actual best sequence for each category at each position it suffices to record only the one preceding category (L_{i-1}) for each category and position. Another $(N \times T)$ array, BACKPTR, will indicate for each category in each position what the preceding category is in the best sequence at position $(t-1)$.

4.2.3. Updating the Category (POS)

Following the identification of part of speech of a stem, each stem will be passed to category updating function to check whether its category changes when it is reunited with the affix. After a new word category is identified, the affix will be attached to the stem and assigned to the previous word.

In the updating category function, six additional POS are included that were not used in the tagger. These POS happens as result of the affixes. The whole POS used in the morphological analysis becomes 36 (appendix 5).

Stem Cat.	Prefix	Suffix	Word Cat.	Sample Word
CN	E	E	CN	lə̈bs
TV	E	a	DN	säbbära
TV	as	i	CN	astämari
CN	ə̈ndä	E	PP	ə̈ndäbäre
TV	E	i	CN	tämari
TV	E	ähut	TV	ayähut
TV	yä	äwun	CV	yäyazäwun
UNC	yä	E	GNP	yä + CN yä + PN yä +PERP

Table 4.7 A Table for Categorical Changes of Inflected Words

In table 4.7, E stands for empty affix position. For each word composed of a tagged stem and an affix, if the stem category and the affix match the information in the stem category and affix columns of the table in a given row, the stem category will be updated with the corresponding new word category, which becomes the resulting category of the stem and the affix(s). When a transitive verb is affixed with one of the verbal prefix such as 'yä' and the suffix 'äwn' (e.g. yäyazäwun), its part of speech tag will be updated into a complementized verb and tagged as a CV (yäyazäwn\CV). Finally, an input sentence is processed into the following format, which will be passed to the sentence-parsing module:

dawit\PERP
kasa\PERP
yäyazäwun\CV
addis\ADJ
borsa\CN
adänäqä\TV
#\PUNC

Infixes are not incorporated into the Category Updating function, as they don't cause change to the category of a stem.

The pre-processing and the different components of the morphological analyser developed for this study have the following relationship shown in Figure 4.9.

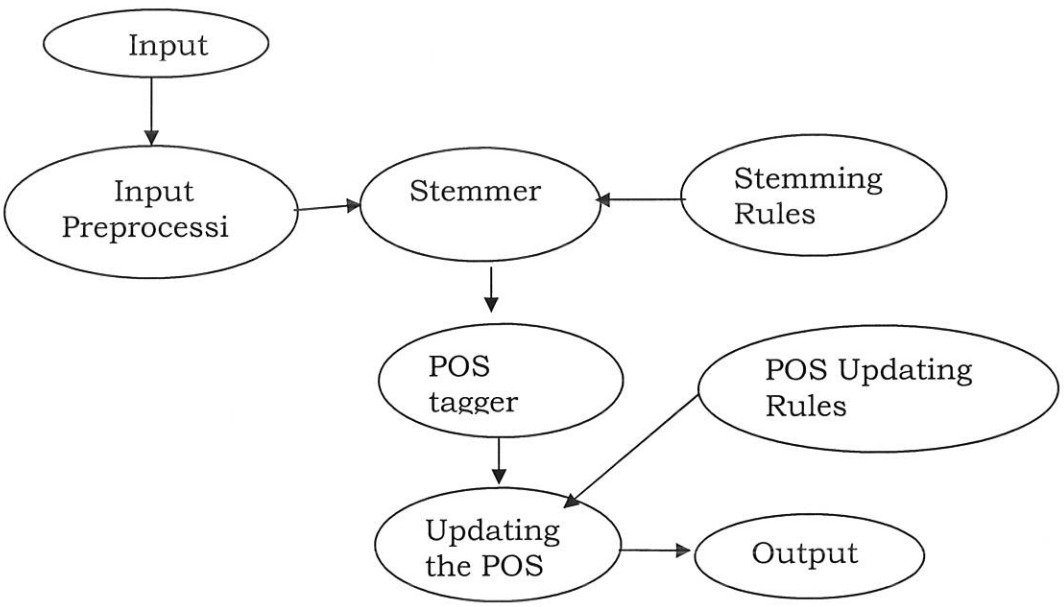


Figure 4.9 Diagrammatic Representation of the Pre-Processing and the Morphological Analysis

The input text breaks down into a sentence and then word level by the pre-processor and passes it to the morphological analyzer. The components of the morphological analyzer first stem and then tag the stem part of speech and finally check whether the POS should be updated or not due to the reunited of the affixes.

CHAPTER FIVE

A RULE BASED PARSER AND THE EXPERIMENT

This chapter is the core of this thesis research. Based on the grammar rules of Amharic discussed in chapter three and disambiguation rules, a rule-based parser is developed. This chapter begins by discussing the approaches to designing the parser, and then explains rules used to disambiguate structural ambiguities. Finally, it describes experiments conducted to evaluate the parser, data analyses, and discussion of the findings of the experiments.

5.1. The Parser

Words that are morphologically analyzed or output of the Morphological analysis system are input of the parser. Parsing a sentence can be considered as a procedure that searches for different ways of combining grammar rules to find a combination, which represents the structure of the sentence. One disambiguation method may return all possible parses for a given sentence; we preferred to provide all possible interpretation of a sentence and finally we present the correct parse. The parser is built based on Amharic grammar rules and a dictionary containing the possible grammatical senses of each word. In the following section we describe the disambiguation algorithm, the searching and parsing strategies, and data structures.

5.1.1.The Active Chart Parsing Algorithm

Chart parsing is an algorithmic schema based on context-free grammars. The basic operation of the chart-based parser involves combining an active arc with a completed constituent (inactive arc) in the chart, or combining completed constituent with active constituent in the chart. The result is either a new completed constituent or a new active arc that is an extension of the original active arcs. The active and inactive (completed constituent) edges are not added directly to the chart but to the agenda. The agenda stores the order of edges waiting to be added to the chart. Given that the stack is selected as the data structure for agenda, an edge is added to the top of the agenda and consequently removed from the top of and added to the chart. The complete algorithm is given in figure 5.1

1. Repeat until there is no input left:
   A. If the agenda is empty, look up the interpretations for the next word in the input and add them to the agenda. (initializing the agenda)
   B. Select the top constituent from the agenda (Let's call it constituent C from vertex P_0 *to* P_2)
   C. Add the selected constituent into the chart
      a. IF the constituent is not in the chart, Then for each grammar rule in the form of $X \rightarrow CX_1 \ldots X_n$, Then add an active arc of from $X \rightarrow \circ CX_1 \ldots X_n$, from vertex P_0 *to* P_1
         i. find any matching inactive edge $C \rightarrow X_1 \cdots X_n \circ$ from the chart
         ii. Combine the active and inactive edges into $X \rightarrow C \circ X_1 \ldots X_n$
      b. For each grammar rule in the form of $C \rightarrow X_1 \ldots X_n$, Then add inactive arc of from $C \rightarrow X_1 \ldots X_n \circ$, from vertex P_0 *to* P_2
         i. find any matching active edge $X \rightarrow \circ CX_1 \ldots X_n$ in the chart
         ii. Combine the inactive and active edges into $X \rightarrow C \circ X_1 \ldots X_n$
   D. Add any new arcs into the agenda.
2. Check if the chart contains an inactive edge from the first node to the last node that is labeled as "S". IF "yes", succeed. Otherwise, fail.

Figure 5.1 Bottom-up Active Chart Parsing (Allen, 1995)

In this algorithm, we only need to specify an initial agenda and a set of derivation rules. To process the first edge on the agenda, we invoke the grammar rule by following fundamental rules, which are described as follows:

- when adding an active edge from the agenda to the chart
  - take an active edge from the agenda:
    edge $(0,0,S \rightarrow \circ NP\ VP)$
  - find any matching inactive edge from the chart:
    edge $(0,2,NP \rightarrow PERP\ SNP \circ)$
  - combine the active and inactive edges and add the new edge to the agenda:
    edge $(0,2,S \rightarrow NP \circ\ VP)$

- when adding an inactive edge from the agenda to the chart
  - take an inactive edge from the agenda:
    edge $(0,2,NP \rightarrow PERP\ SNP \circ)$
  - find any matching active edges in the chart:
    edge $(0,0,S \rightarrow \circ NP\ VP)$
  - combine the active and inactive edges and add the new edge to the agenda:
    edge $(0,2,S \rightarrow NP \circ\ VP)$

New edges are continuously being added and then removed from the agenda for processing until the agenda becomes empty. When this happens, the agenda takes the next edge from the input stream. If the input stream is also empty, the parsing is terminated. If one or more inactive edges S spanning the whole chart, the given input has been fully parsed. In this case, each spanning edge in the chart corresponds to a parse.

5.1.2.Searching and Parsing Strategies

A search strategy aims to determine which state or node in a tree to expand first. Even though there are some limitations like getting stuck in an infinite loop, we employed depth-first search strategy in the parser, because it requires moderate working space and it tends to minimize the number of backup states,. In this strategy children nodes are expand prior to sibling nodes. Correspondingly, the collection of nodes is implemented as a stack, in which nodes are added or removed from the top of the stack.

A bottom-up parsing strategy is adopted in designing the parser. In this strategy parsing starts from the initial states that are words of the sentence and attempts to reach in to the goal state (S) which is a state of recognizing the sentence. It takes a sequence of symbols and matches it to the right-hand side of rules. In doing so left recursion problem will be avoided. The parser is built by formulating the matching process as a search process within the grammar rules (state space). A state simply consists of a list of symbols, starting with the words in the sentence. Successor states are generated by exploring all the possible ways to:

- Rewrite a word with its possible lexical categories
- Replace a sequence of symbols that matches the right-hand side of a grammar rule with its left-hand side symbol

Such an implementation could be extremely expensive because the parser would try the same matches repetitively. This problem can be avoided by using chart parsing algorithms, which introduces a data structure called chart together with agenda. The chart contains partial results of the matching that is already done (active or incomplete edges) and the agenda stores an order in which tasks are carried out.

5.1.3.The Active Chart Parser

As shown in the previous section, we need to add some active edges to the chart after applying the fundamental rule. These active edges correspond to the rules in the grammar. The rules are invoked and the active edges are added in the following order: once an inactive edge is added to the chart, find a grammar rule that contains the category as the leftmost element on the right-hand side of the rule and add it to the chart.

1. $S \rightarrow NP \quad VP$	8. $MVP \rightarrow SVP$
2. $NP \rightarrow MNP$	9. $MPP \rightarrow SPP$
3. $MNP \rightarrow SNP$	10. $SPP \rightarrow PP$
4. $SNP \rightarrow PERP$	11. $SVP \rightarrow MNP\ TV$
5. $SNP \rightarrow CN$	12. $SVP \rightarrow MPP\ MNP\ TV$
6. $VP \rightarrow MVP$	13. $SVP \rightarrow TV$
7. $MVP \rightarrow MPP\ SVP$	

Figure 5.2 A Simple Context-Free Grammar

Such an implementation could be extremely expensive because the parser would try the same matches repetitively. This problem can be avoided by using chart parsing algorithms, which introduces a data structure called chart together with agenda. The chart contains partial results of the matching that is already done (active or incomplete edges) and the agenda stores an order in which tasks are carried out.

5.1.3.The Active Chart Parser

As shown in the previous section, we need to add some active edges to the chart after applying the fundamental rule. These active edges correspond to the rules in the grammar. The rules are invoked and the active edges are added in the following order: once an inactive edge is added to the chart, find a grammar rule that contains the category as the leftmost element on the right-hand side of the rule and add it to the chart.

1. $S \rightarrow NP \quad VP$
2. $NP \rightarrow MNP$
3. $MNP \rightarrow SNP$
4. $SNP \rightarrow PERP$
5. $SNP \rightarrow CN$
6. $VP \rightarrow MVP$
7. $MVP \rightarrow MPP\ SVP$
8. $MVP \rightarrow SVP$
9. $MPP \rightarrow SPP$
10. $SPP \rightarrow PP$
11. $SVP \rightarrow MNP\ TV$
12. $SVP \rightarrow MPP\ MNP\ TV$
13. $SVP \rightarrow TV$

Figure 5.2 A Simple Context-Free Grammar

For instance, given a grammar as shown in Figure 5.2, we parse the following sentence (annotated with part of speech).

aster\PERP läkasa\PP mäs'haf \CN sät'ăččw\TV # (1)

To better illustrate the parsing process, we display the updated chart as it is extended after each step of the algorithm. The agenda is initially empty, so the word "aster" from position P_1 to P_2 is read and the constituent PERP (personal pronoun) placed onto the agenda (step 1.A). We then check whether it is already recorded in the chart (step 1.C.a). That is not the case, so we move the constituent PERP from the agenda into the chart. As a result, rule 4 $(SNP \rightarrow \circ PERP)$ which is an active edge is invoked and added into the chart. Next, we try to apply the fundamental rule to the new arc (step 1.C.a.i, ii). The fundamental rule combines an active arc with a passive arc that is immediately to the right of the active one. Since the arc we are working with is active, we have to find inactive arc to the right of, or more precisely, we have to find an inactive arc that ends in position 1. There is one; namely, $(1,2, aster \rightarrow PERP \circ)$. We can therefore build the edge $(1,2, SNP \rightarrow PERP \circ)$. Completing the arc $SNP \rightarrow PERP \circ$ causes adding SNP into the agenda (step D). Likewise, SNP is added to the chart by selecting it from the agenda; consequently the 3rd rule $(MNP \rightarrow \circ SNP)$ is added into the chart. Completing arc $MNP \rightarrow SNP \circ$ causes moving MNP (main noun phrase) into the chart. As a result, rule 2 $(NP \rightarrow \circ MNP)$ and rule 11

$(SVP \rightarrow \circ MNP\ TV)$ are invoked and added into the chart as an active edge. The completion of arc $NP \rightarrow MNP\circ$ adds NP to the agenda. Finally, NP is pushed into the chart, which invokes rule 1 $(S \rightarrow \circ NP\ VP)$ and adds it to the chart as an active arc $(S \rightarrow NP\circ\ VP)$.

The next word "läkasa" is read from position p2 to p3 and the constituent 'PP' is entered into the agenda. This constituent is pushed from the agenda into the chart. This in turn invokes rule 10 $(SPP \rightarrow \circ PP)$ which is entered into the chart. Entering rule 10 into the chart completes arc $(SPP \rightarrow PP\circ)$. Then the SPP (sub prepositional phrase) constituent is a added into the agenda. SPP will be pushed from the agenda and entered into the chart. As a result, rule 9 $(MPP \rightarrow \circ SPP)$ is activated and entered into the chart. The completion of arc $MPP \rightarrow SPP\circ$ causes MPP (main prepositional phrase) to be entered into the chart. The addition of MPP activates rule 7 $(MVP \rightarrow MPP\circ\ SVP)$ and rule 12 $(SVP \rightarrow MPP\circ MNP\ TV)$. What has been completed in the chart is shown in Figure 5.3. The chart contains two completed constituents: PERP from position 1 to 2 and PP from position 2 to 3.

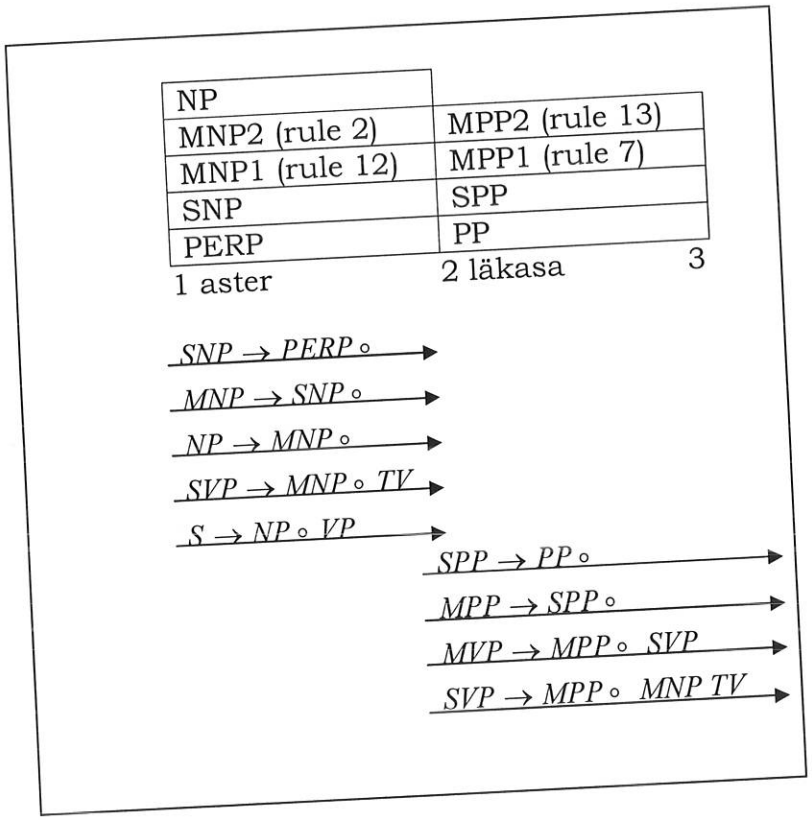


Figure 5.3 the Chart after Seeing PP in Position 2

For the next word "mäs'haf", constituent CN is entered into the agenda. Entering CN into the chart invokes rule 5 $(SNP \rightarrow \circ CN)$ and ultimately the arc $SNP \rightarrow CN \circ$ is completed. The completion of the arc will add SNP into the agenda, which will be pushed into the chart. Entering SNP into the chart activates new arc $(MNP \rightarrow \circ SNP)$in rule 3 and places it into the chart. The completion and entering of the inactive MNP edge into the chart calls for the application of the fundamental rule. To apply the fundamental rule (step 1.C. b. i, ii) we have to find an active edge in the

chart that starts in position 2. The edge $(SVP \rightarrow MPP \circ MNP\ TV)$ meets the requirement. As a result, the active edge is extended $(SVP \rightarrow MPP\ MNP \circ\ TV)$. The completion of rule $(MNP \rightarrow SNP \circ)$ also enters rule 12 $(SVP \rightarrow MNP \circ\ TV\ )$ and rule 2 $(NP \rightarrow MNP \circ)$ into the chart. Finally, the completion of arc $NP \rightarrow MNP \circ$ invokes rule $1\,(S \rightarrow NP \circ\ TV)$ and enters it into the chart.

Similarly the final word is read and constituent TV is entered into the chart after being selected from the agenda. This extends the active arcs $SVP \rightarrow MPP\ MNP \circ TV$ and $SVP \rightarrow MNP \circ TV$. Such extension completes the arcs and enters SVP1 and SVP2 into the agenda using step (1.C.b.i, ii). Completing SVP2 (sub verb phrase) and selecting it from the agenda extends and completes the active arc $(MVP \rightarrow MPP \circ SVP)$. By the same way completing and entering SVP1 into the chart invokes and enters rule $8\,(MVP \rightarrow \circ SVP)$ into the agenda. Entering this edge into the chart completes arc $(MVP \rightarrow SVP \circ)$. The completion of arcs $MVP \rightarrow MPP\ SVP \circ$ and $(MVP \rightarrow SVP \circ)$ enters MVP1 (main verb phrase) and MVP2 into the agenda. The insertion of MVP1 and MVP2 into the chart causes rule 9 $(VP \rightarrow MVP \circ)$ to be invoked and placed onto the chart. Finally, once VP1 and VP2 are entered, no active arcs are added in step B, The two arcs $S \rightarrow NP \circ\ VP$ will be completed in step (b, ii) by the arc extension

algorithm, which produces two Ss being added to the agenda. The complete chart is shown in Figure 5.4.

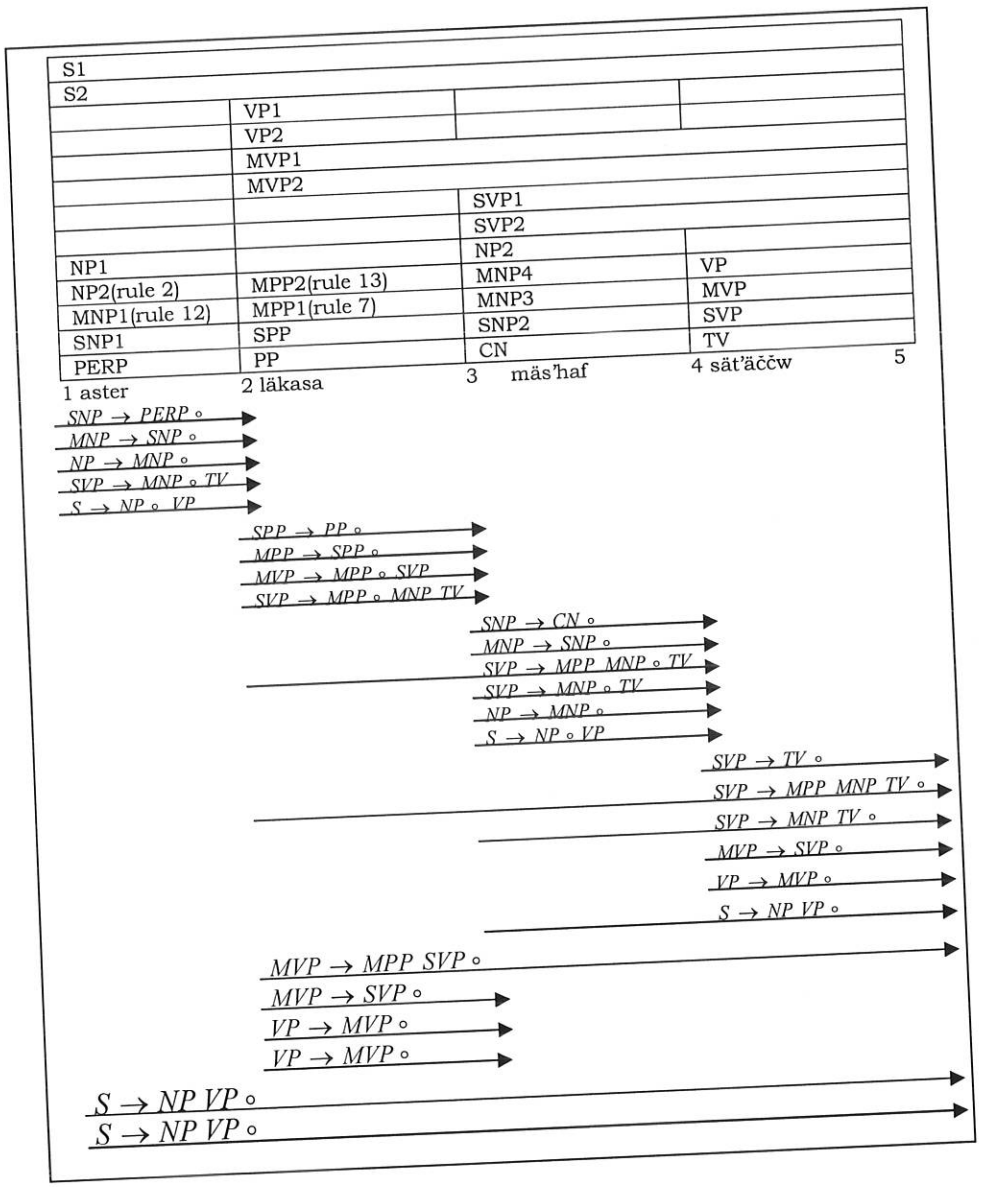


Figure 5.4 the Final Chart

The agenda is now empty, so we exit the loop. This takes us to step 2 of the algorithm. As is shown in the chart, we have two passive "S" nodes from the first to the last word in the sentence.

We have succeeded in recognizing the sentence by obtaining two complete structures of the sentence. To better illustrate the parsing results, we extract the parse trees from the final chart, which are displayed in Figure 5.5.

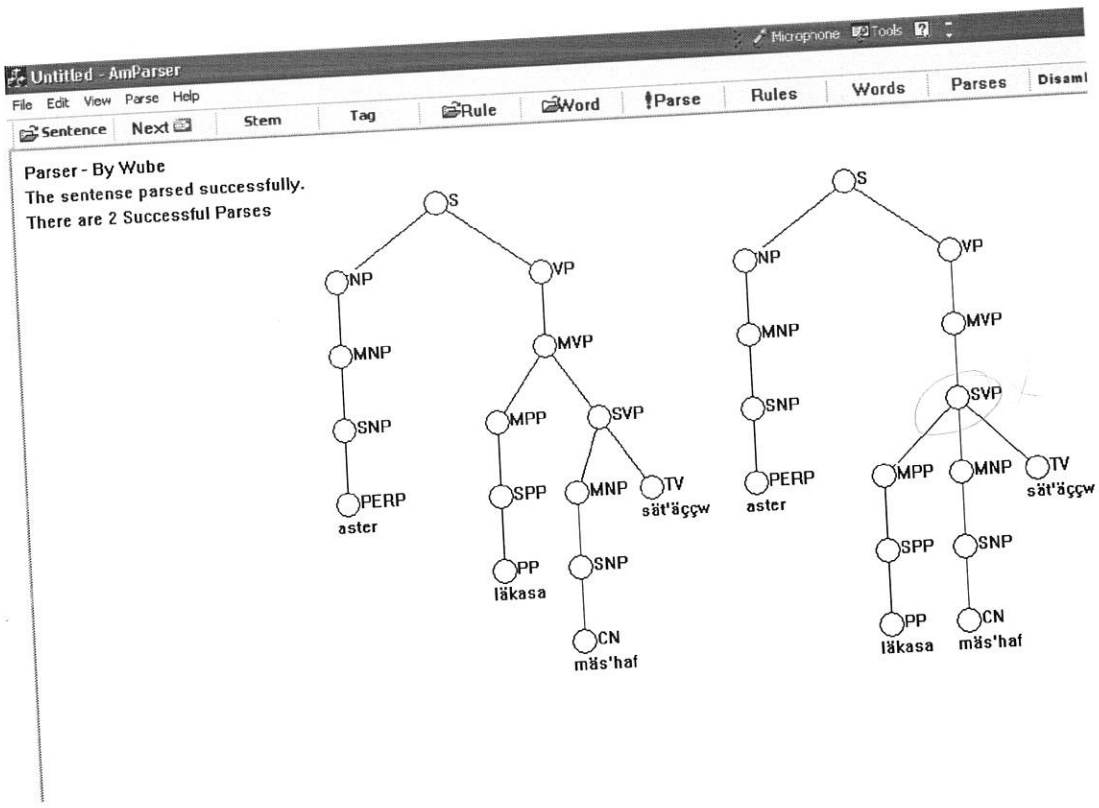


Figure 5.5 Parse Structure of the Final Chart

The above parse trees record the syntactic structure of the input sentence. Each tree consists of a parent node and several child trees.

5.2. Disambiguation Rules

In resolving structural ambiguities, a rule-based method is preferred than a statistical method in the design of the parser. As stated in Chapter two, rule-based parser is powerful enough to describe most of the structures in natural language. A probabilistic or statistic with grammar rule approach, such as PCFG, selects the best parse tree with the highest probability using the inside and outside algorithm, which is generated by production rules. However, ignoring the linguistic features of a language may sometimes end up with selecting a wrong parse. An illustration is shown in Figure 5.6

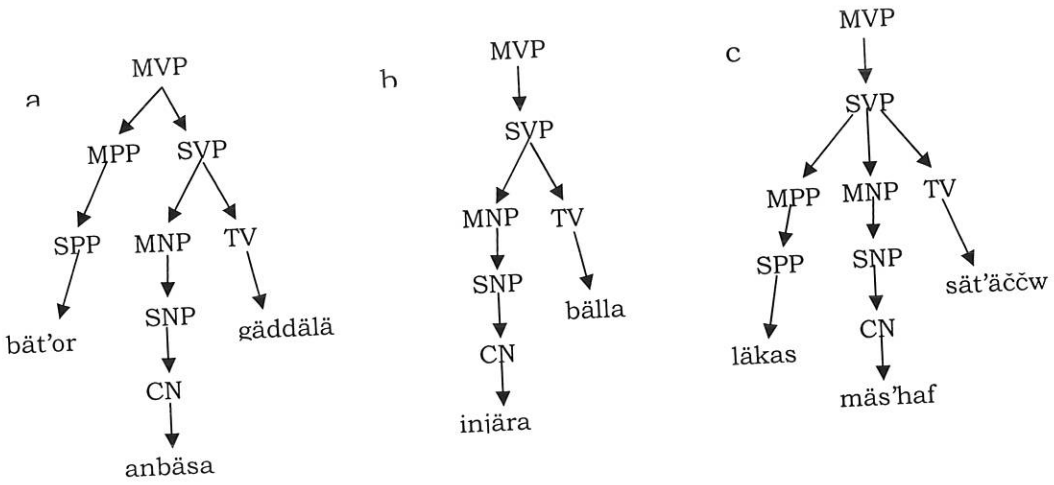


Figure 5.6 Syntactic Ambiguity of Verb Phrase

of a phrase (gäddälä, sät't'äččw) into a dictionary. The dictionary is used to select one of the correct parses.

As discussed in Chapter two, the Amharic verb phrases and noun phrases are composed of complements, modifiers, and the head word. According to Getahun (1995) and A Baye (1986) Amharic has five basic types of verbs:

- Action Verbs that are Transitive (bälla, säbbrä, lakä)
- Action Verbs that are Intransitive (hedä, gäba)
- Action Verbs without complement (aläqäs, c'c'oh)
- Auxiliary Verbs of be (näw, yəmäsəlal)
- Verbs of Existence (allä, norä).

The above verbs are headwords of verb phrases. The headword is always located at the right-end of a phrase. It is preceded by complements which together constitute a sub verb phrase (SVP). Modifiers precede a sub verb phrase in order to form a main verb phrase (MVP). For instance, consider the following sentence.

abäbä hulät gize läaster bäbank gänzäb ləkolatal # (3)
Abeab two times to Aster through bank money he has sent

'Abebe has sent money to Aster through bank two times.'

/lɔkolatal/ 'he has sent' is the head of the verb phrase, which is a transitive verb. The headword is complemented with an object /gänzäb'/, thus /gänzäb lɔkolatal/ 'has sent money' is a sub verb phrase. /hulät gize läaster bäbank/ 'two times through bank' is the modifier of the sub verb phrase. Therefore, the main verb phrase of the sentence is:

/hulät gize läaster bäbank gänzäb lɔkolatal/

One main structural ambiguity encountered in applying the grammar rules in this thesis is caused by misplacing modifiers as complements and vice versa as it is shown in Figure 5.5. Another ambiguity is due to the inability to identify appropriate word groups of complements and modifiers. It has great impact on disambiguation to identify the boundaries of complements and modifiers, for sub phrase (SP) and main phrase (MP) structures are essentially determined by boundaries.

The head of a noun phrase is located at the end of the phrase. The headword is a noun that can have complements and modifiers. The sub noun phrase (SNP) is formed by the headword and complements. When an SNP is combined with modifiers, they form a main noun phrase (MNP).

5.2.1.The Dictionary

The dictionary provides knowledge about a list of headwords including their inflected forms. It contains list of headwords, the grammatical

sense category of each headword with its part of speech, and the structures of complements and modifiers of each headword category.

The dictionary is stored in a related file. Table 5.1 lists category of headwords of both noun and verb phrases. Words belonging to the same grammatical sense category at least share similar complement structures.

ID	Category	Meaning	Explanation
1	SN	Source noun	Stands for noun phrase headword that takes object complement.
2	FN	Free noun	Stands for noun phrase headword that doesn't take object complement.
3	SOTV	Single object transitive verb	Stands for transitive verb that takes only one object complement.
4	DOTV	Double object transitive verb	Stands for transitive verb that takes two objects complement.
5	DIV	Different intransitive verb	Intransitive verb that shows movement from or to different place
6	SIV	Same intransitive verb	Intransitive verb that shows movement on the same place

Table 5.1 Different Categories of Headwords

The list of headwords along with their appropriate grammatical sense categories is given in table 5.2.

Category	words
1	bet
4	sät't'ä
4	sät't'äččw
4	wäsädä
5	hedä
5	mät'a
3	gäddälä
2	kokäb
6	qomä

Table 5.2 List of Headwords against Category

The structural relationship of complements and modifiers to the headwords of the verb phrase and the noun phrases are stored in a flat text file. It consists of four files, including verb complements, verb modifiers, noun complements and noun modifiers.

The complements and modifiers are organized in a separate file in order to determine their boundaries easily. The rules in the files are organized in a parent and children format taking the part of speech of the constituent as a unit, which is enclosed with curly brackets. This will help to identify the appropriate word groups for both of the complements and modifiers.

The verb complement rules describe the appropriate word groups of complements for each grammatical sense category of the headword with in each type of verbs. The noun modifier rules also determine the

appropriate word group of the modifier. In the same way, the noun complement and modifier rules describe the appropriate word groups for each of them. As an illustration, the appropriate word group of a verb complement is given in Figure 5.8.

```
SVP { MNP { SNP { CN } } TV
SVP { MNP { SNP { CN } } TV { SOTV } }
SVP { VE { BVE } }
SVP { MPP { SPP { PP } } MNP { SNP { CN } } TV
SVP { MPP { SPP { PP } } MNP { SNP { CN } } TV { DOTV } }
SVP { MNP { SNP { CN } } AUX { SCA } }
SVP { MNP { SNP { PN } } VE }
SVP { MPP { SPP { PP } IV { DIV } }
SVP { MPP { SPP { PP { PRE } } MNP { SNP { PN } } IV {  DIV } }
SVP { MADJP { SADJP { ADJ } } AUX { NCA } }
SVP { MPP { PP } SPP { PP { PRE } MNP { SNP { PN } } IV { DIV } }
SVP { GV AUX { NCA } }
SVP { S' { MNP { SNP { PERP } } MVP { SVP { CV } } } TV { SOTV } }
SVP { MNP { SNP { AN } } AUX { NCA } }
```

Figure 5.8 Organization of the Verb Complement Disambiguation Rule

In Figure 5.8, the third rule describes the appropriate complements' word group of the headword /sät't'äččw/ 'She gave him' that was illustrated in sentence (1).

In order to select the appropriate parse from all possible parses that are generated, the disambiguation rules in the dictionary are applied. When

the rules are invoked, they are converted into a small tree-structured format using the algorithm given in figure 5.9.

```
While input not finished
      Begin
      Find the next input string
         IF the string is not " } " THEN
            Add the component to the stack
         Else
            1. Take all elements up to the opening brackets "{ "

            2. Take the next element (part of speech) as parent
            of the elements (part of speech) that are taken
            previously.

            3. Add in to the stack the element (part of speech)
            removed in step 2 as child of the elements that will
            be extracted in the future.
       End
End
```

Figure 5.9 Algorithm for Converting a Disambiguation Rule into Tree Structure.

The disambiguation rules chooses the correct parse from all the possible parses on the chart by comparing each parse against the disambiguation rules in the dictionary. The disambiguation starts from right to left, for the headword is located on the right side of all type of phrases in Amharic. The algorithm for applying the disambiguation rule is given in figure 5.10.

1. Open all disambiguation rules files.
2. Generate a tree from a rules
3. Go to the verb complement file
4. Do until the end of the file.
    a. Take the headword of a phrase from the parse chart and find its grammatical sense category in the dictionary.
    b. Put the category as the last leaf node next to the POS of the headword.
    c. Take only leaf node (the parts of speech) of the complements structure rule in the dictionary and arrange in the ascending order.
    d. Compare it with the part of speech of a constituent of the parse in the chart from right to left.
    e. If found THEN
        i. Select the appropriate tree structure of the complement (SVP) from the alternate parses.
        ii. If the sentence parsed is free of ambiguities then terminate the process.
            1. Else go to step 5
    f. Else report failure in disambiguation.
5. Go to verb modifiers rules file.
6. Do until the end of the file.
    a. Take only leaf node (the part of speech) of the modifiers structure rule in the dictionary and arrange in the ascending order.
    b. Compare it with the part of speech of the constituent of the parse in the chart from right to left.
    c. If found THEN
        i. Select the appropriate tree structure of the complement (MVP) from the alternate parses.
        ii. If the sentence parsed is free of ambiguities then terminate the process.
            1. Else go to step 7
    d. Else report failure in disambiguation.
7. Go to noun complements rules file.
8. Do until the end of the file.
    a. Take the headword of the phrase from the pars chart and find its grammatical sense category in the dictionary.
    b. Put the category as the last leaf node next to the POS of the headword.
    c. Take only leaf node (the part of speech) of the complements structure rule in the dictionary and arrange in ascending order.
    d. Compare it with the part of speech of the constituent of the parse in the chart from right to left.
    e. If found THEN
        i. Select the appropriate tree structure of the complement (SNP) from the alternate parses.
        ii. If the sentence parsed is free of ambiguities then terminate the process.
            1. Else go to step 9
    f. Else report failure in disambiguation.
9. Go to noun modifiers rules file
10. Do until the end of the file.
    a. Take only leaf node (the part of speech) of the complements structure rule in the dictionary and arrange in ascending order.
    b. Compare it with the part of speech of the constituent of the parse in the chart from right to left.
    c. If found THEN
        i. Select the appropriate tree structure of the modifier (MNP) from the alternate parses.
        ii. If the sentence parsed is free of ambiguities then terminate the process.
    d. Else report failure in disambiguation.

Figure 5.10 the Algorithm for Structural Disambiguation

Two structural parses are generated for a sentence (1), as shown in figure 5.4, for the constituent 'PP' can be classified either as a complement or as a modifier. By applying the fifth line of the verb A complement disambiguation rule, we assign 'PP' as a complement.

The disambiguation rule disambiguates the two alternative structural parses of sentence (1) at the first loop of the algorithm, or at the verb complement stage (step 4.e.ii) as shown in figure 5.11

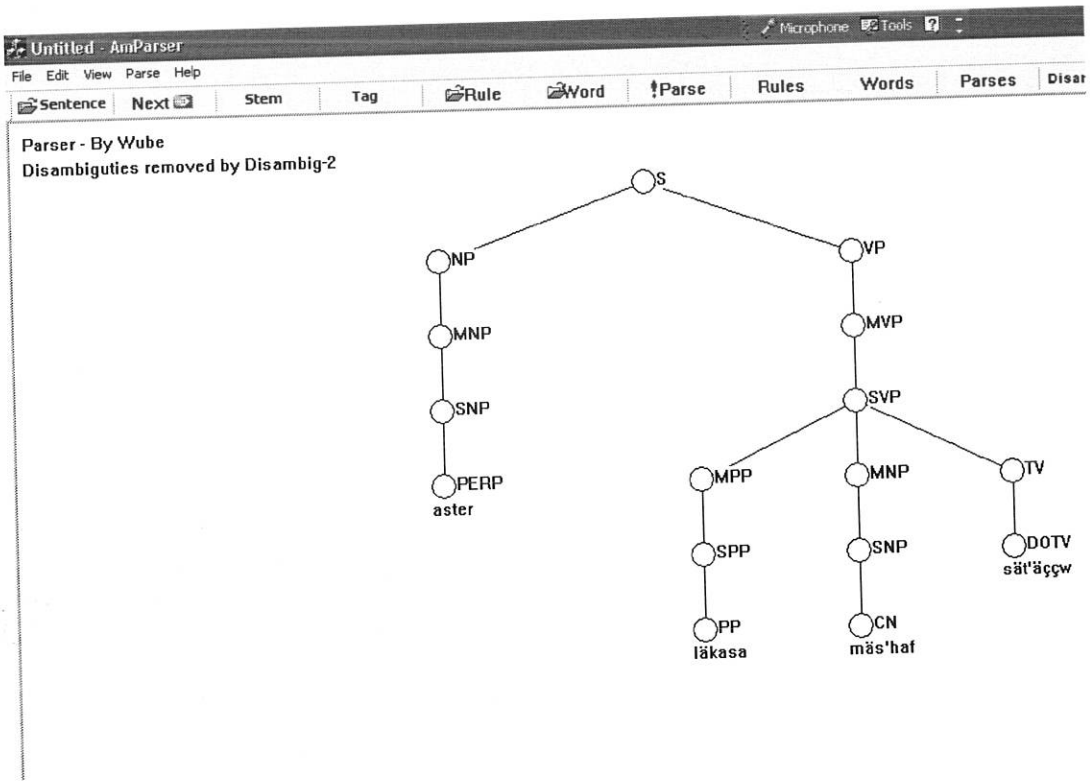


Figure 5.11 the Correct Parse of Sentence (1)

The above figure shows the correct parse of sentence (1) after the disambiguation rule is applied.

The process flow of the parser is displayed in figure 5.12.

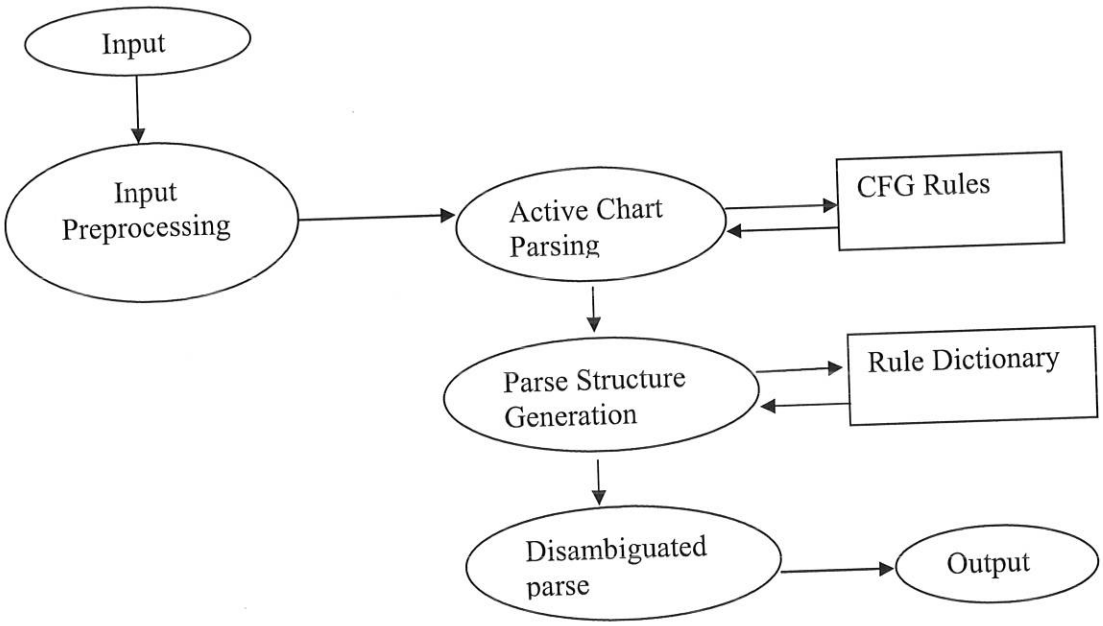


Figure 5.12 Diagrammatic Representation of the Parser

5.3. The Experiment

The text selected for the evaluation was discussed in chapter four. Each sentence in the text was, stemmed, tagged and parsed manually. Comments and suggestions are sought from the linguistic advisor when necessary.

From the manually labeled corpus, 320 sentences were randomly selected as the training set. They were used to derive grammar rules.

The remaining 80 sentences (573 words) were used for testing. The evaluation metrics for the stemmer or the tagger was accuracy, which is defined as the percentage of words that are correctly stemmed or tagged. The parser is also evaluated in terms of accuracy, which is the percentage of sentences that are parsed correctly.

$$Accuracy = \frac{number\ of\ correctly\ parssed\ sentence\ by\ the\ parser\ in\ the\ test\ set}{total\ number\ of\ hand\ parsed\ sentences\ in\ the\ test\ set} \times 100$$

The evaluation should have been approved by at least two independent parties separately however due to time limitation it is made by only the researcher.

5.3.1.Evaluation of the Stemmer and Tagger

Manual assessment of the stemming results showed that 541 stems (94.4 %) were linguistically meaningful and 32 (5.6 %) were meaningless. The errors consist of 12 (2.1 %) over-stemming and 20 (3.2 %) under-stemming cases. When it is compared with Nega and Willett it is less by 1.5%. Some stem requires deep structure analysis in order to attain the actual stem. Sample cases of over stemming, under stemming and stem that require deep structure analysis are shown in appendix 6.

The first test of the tagger achieved 89.7% accuracy. However, most of the errors were caused due to error committed during manual tagging (wrong manual tagging). After making necessary corrections to the baseline, the tagger attained 96.2% accuracy. It is better than Mesfine's by 5.2%.

5.3.2.The parser Evaluation

The evaluation of the parser based on the given metrics shows encouraging result. It is shown in Table 5.3.

Data/ set	**No of sentences**	**No of erroneously parsed sentences**	**Accuracy**
Test Set	80	11	86.25%

Table 5.3 Parsing Result on the Test Set

The accuracy was over 86%, which was satisfactory. Even though the corpus contains both simple and complex sentences, most of them are grammatically more structured than the usual sentences that one encounters (Some times it is acceptable if one miss the subject or object of the sentence as it is found in the verb a in the form of a subject or object affixes, even though it is not grammatical, exchanging the position of direct and indirect object of a sentence is acceptable). Thus, taking into account the sample sentences, the accuracy result of the parser seemed acceptable. Sample output of the parser is listed in Appendix 7.

5.3.3.Solution to Identified Problems

Due to the lack of resource in corpus and tree bank, we have encountered various problems in developing the parser. For example, human errors introduced during morphological analysis like incorrect stemming, tagging, and parsing and other sources of errors led to wrong induction of grammar rules, inconsistency between grammar rules and disambiguation rules during the time of training.

It was difficult to fined actual affixes of some words in order to develop context sensitive rules. Some times it becomes difficult to find the actual stem of a word after its affixes are removed. For example the word /mäsrat/ 'doing something' has a prefix of /mä/ and suffix of /at/. Finally we left with /sr/ which is assumed stem of the word. However it is neither the actual stem of the word nor the root of the stem word. Therefore, it requires finding the actual root of the word which is /srr/ and template of the word which is /cäcc/ and builds the actual stem which is /särr/. There were problem encounter with updating category function. The same part of speech when they are affix with the same suffix the system assigns one of the parts of speech which ever it finds first, usually TV.

Eg.

Eg. Stem	POS	Suffix	Word	Actual POS
säbbr	TV	a	säbbra	Derived Noun
bäll	TV	a	bälla	TV

Table 5.4 Problem of category updating function

With regard to stemming there is proble of developing context sensitive rule to words which have the same Phonemic representation and pronunciation, but which have different parts of speech. For example t'älla/TV '(he) hate' and t'älla/noun 'common drink' are two words, which have the same Phonemic representation. It becomes difficult to remove the last affix 'a' from the transitive verb and leave from the noun, as it is part of the steem.

There were also problem associated with manual stemming of words during training set preparation to the tagger. Some words were under stemmed while others were over stemmed.

Moreover, there was unknown word problem when words that did not have their stems in the database were tagged as unknown. Due to the insufficiency of the bi-gram count, some words were wrongly tagged.

To find the actual affixes of a word, the words are classified into original and derived with in each class of words (noun, verb, etc), then it become possible to find the actual affixes. Some words, which are difficult to find their actual stem, are taken as it is.

The human errors were corrected by iterative action during the time of training. Attempts were also made to guess parts of speech that are unknown to the tagger by evaluating their affixes. However, it was difficult to correct mis-tagged words.

The system trapped into infinite loop whenever it encounters dependent sentence. The problem was solved by introducing different rules for independent sentence **(S→NP VP)** and dependent sentence **(S'→MNP MVP)** separately. The major causes of parsing errors were over-stemming or under-stemming and mis-tagged and untagged stems. Another limitation is inherent in rule-based method, which is not exhaustive.

CHAPTER SIX

SUMMARY, CONCLUSION AND RECOMMENDATION

This chapter summarizes the major tasks accomplished and the contributions of the thesis, and suggestions for future work.

6.1. Summary and Conclusion

The objective of this thesis is developing a rule-based syntactic disambiguation parser for Amharic language. In chapter one, we introduced the importance and the one of challenges of developing a parser, i.e. sentence disambiguation. In chapter two and chapter three, we provided an overview of important concepts related to parsing and various strategies that can be used to address the challenging issues in parsing. In addition, we briefly reviewed Amharic grammar.

We described methods for data preparation and morphological analysis in chapter four. In Chapter five, we introduce the design and development of a parser. An evaluation of the parser showed that it attained 86.25% accuracy.

This thesis research represents the first study to build a rule-based parser for Amharic language. The performance of the parser was very promising, which improved the accuracy by 4.25% over that of the parser developed by Daniel Gochel. We hope that the results and findings of this

thesis will attract more researchers to study Amharic language and will enable more advanced analysis (e.g., semantic and discourse analyses) of Amharic language. The disambiguation method developed in this study can help improve the performance of other Amharic parsers. Cautions should be taken in generalizing the findings of this study. Due to the lack of tree bank for the language, it is difficult to compare the performance of different Amharic parsers both knowledge-based and corpus-based on the common ground.

The thesis integrates previous NLP studies on Amharic, which serves as the foundation for building high-level language analyzers and practical applications such as machine translations.

We can find several limitations with this research. First, the data size used for rules induction was small. Therefore, the parser may not have considered all possible features of Amharic sentences. Second, it didn't find the actual stem of all words in the sample sentence. Third, the morphological analysis was not comprehensive, for it only analyzes the stem morphemes. Forth, as the parser generates all possible parses and it completely depends on rules has a problem with cost and efficiency. Finally, Phonemic symbols are used to transcribe the text rather than Amharic 'fidel'. The Amharic 'fidel' was not used, because it is difficult to analyze the morphological features of every word.

6.2. Future Directions

There are several directions for improving the current parser in future.

1. The above limitations can be addressed in the following ways. First, enlarging the size of training and testing data sizes for rule induction and evaluation. With the prototype parser available, we can improve the efficiency in labeling training and testing data by parsing the text with the parse first and then validating the parsing results manually. Second, enabling the system to find root and template of the stem word. Third, extending the scope of morphological analysis to include grammatical features of affixes of a stem. This will help to develop feature-based parsers. Fourth, integrating a morphological analyzer and a POS taggar with the parser. Fifth, increasing the efficiency of the parser by developing a hybrid model for disambiguating Amharic sentences, which integrates rule-based and statistical methods. In this approach, generating a single and correct parse can also reduce the searching space. To generate the correct parse, first, the possible word groups of complements and modifiers will be generated first. Then, the next subsequent higher level of parses (parent node) can be generated by using probabilistic context free grammar. Finally, semantic knowledge is required to disambiguate structural ambiguities that arise from the scope of adjective over two nouns that are connected with conjunction and structures derived from different deep structures of PP's with a simple NP or genitive NP complement that are described in chapter three.

2. In this study, the bi-gram lexical co-occurrence assisted by a simple morphological analyzer was developed to determine the lexical category of a stem morpheme. The technique worked well for various inflections of a given stem in the database. Although this was one step further than earlier studies, some words were tagged wrongly. Thus, future researches could explore a better result by combining the tri gram lexical co-occurrences.

The thesis research can be extended in a number of ways.

1. Further researches can be conducted in applying the Amharic parser in variety of application such as, machine translation and spell and grammar checker could be conducted.
2. It would be benefit the entire Amharic NLP community by building a corpus in which both simple and complex Amharic sentences have proportional representation. The corpus can be used to extract both the grammar and disambiguation rules.
3. Future research can be pursued to develop a tree bank of Amharic. The bank can not only supports the development of statistical parser but also serve as a common ground to evaluate the performance of parsers.

REFERENCE

1. *Abiyot Bayou. 2000. Developing Automatic Word Parser for Amharic Verbs and Their Derivation, Master Thesis at School of Information Studies for Africa, Addis Ababa*

2. *Allen, James, Natural Language understanding, Redwood City, Benjamin/Cummings, 1995.*

3. *Anderson D.,* Syntax & Parsing: Basic Concepts, 2002 http://kiri.ling.cam.ac.uk/psych/docs/syn.doc

4. *Arnold, Doug, (October 9, 2001), Computational Linguistics I: Parsing and Generation, University of Essex. http://courses.essex.ac.uk/lg/LG511/0-Outline/index.html*

5. *Atelach Alemu, Automatic sentence Parsing for Amharic Text: An Experiment Using Probabilistic Context Free Grammars, Masters Thesis, Addis Ababa University, 2002*

6. *Backofen, Rolf, (1995), Feature Descriptions and Constraint-Based Grammars,* http://www.bio.inf.uni-jena.de/Publications/Backofen:PhD-Thesis:94/node6.html

7. *Baye Yemam, Amharic Grammar (Amharic), Educational Material Production and Distribution Enterprise, 1986*

8. *Bender, M. L., Sydney, W. H. and Roger Cowley. The Ethiopian Writing System, In Bender et al (Eds.) Language in Ethiopia. London: Oxford University Press, 1976.*

9. *Dawkins, C. H., The Fundamentals of Amharic, A.A Sudan interior mission, 1960*

10. *Ethiopian Central Statistical Authority (ECSA), The 1994 Population and Housing Census of Ethiopia: Results at Country Level. Vol. I Statistical Report, Addis Ababa, 1998.*

11. *EAGLES Central Secretariat, Parsing* http://www.ilc.cnr.it/EAGLES96/rep2/node41.html#SECTION0 43510000000000000

12. Formkin, Victoria & Rodman, An introduction to language, 3rd ed. Holt, Rinehart and Winston, 1983.

13. Frazier, L. & Fodor, J. (1978). The sausage machine: A new two stage parsing model. Cognition, 6, 291-325.

14. Frydenlund, M. and Svensen, K. Amharic for Beginners, Norwegian, Lutheran mission language school, SIM Press, Addis Ababa, 1998.

15. Getahun Amare, Amharic Grammar Simplified Approach (Amharic), Commercial Printing Press, 1995.

16. Getahun Amare, Towards the Analysis of Ambiguity in Amharic, "Journal of Ethiopian Studies", Vol. XXXIV, No.2, December 2001.

17. Harries, Mary Dee, Natural Language Processing, Reston Publishing company, Inc., Reston, 1985.

18. Hang Li, (1996) A Probabilistic Disambiguation Method Based on Psycholinguistic Principles, C&C Research Laboratories, NEC Corporation, lihang@sbl.cl.nec.co.jp
http://acl.ldc.upenn.edu/W/W96/W96-0112.pdf

19. Hepple, M.; Kevitt, Mc.; Wilks, Y., Novel-parsing methods
http://www.dcs.shef.ac.uk/nlp/parsing.html

20. Heris, Graeme, Semanic interpretation and the resolution of ambiguity, Cambridge University press, (1986).

21. Joshi, Aravind, 1996, Parsing Techniques, Pennsylvania: University of Pennsylvania.
http://cslu.cse.ogi.edu/HLTsurvey/ch11node6.html
On, Survey of the State of the Art in Human Language Technology
http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html

22. Joshi, A.K., Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In Dowty, D.R. Karttunen, L., and Zwicky, A. (Eds.), Natural Language Parsing, 1985, PP 206-250, Cambridge University Press, Cambridge.

23. jurafsky, Daniel & Martin, James H., SPEECH AND LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2000, Pearson Education Inc., Delhi.

24. Kay, Martin Kay, Algorithm Schemata and Data Structures in Syntactic Processing, 1980. Report CSL-80-12, Palo Alto, CA: XEROX PARC.

25. Kazakove, Dimater & Mnanadhar, Suresh (2000) Unsupervised Learning for Word Segmentation Rules with Genetic Algorithms and Inductive Logic Programming, Available at http://Citeseer.nj.nec.com/kuzakov00unsupervised.html

26. Kibur Lisanu. 2002. Design and Development of Automatic Morphological Synthesizer for Amhairc Perfective Verb Forms, Master Thesis at School of Information Studies for Africa, Addis Ababa.

27. Kimball, d., 1973: "Seven Principles of Surface Structure Parsing in Natural Language," Cognition, Volume 2, Number 1, pp. 15-47.

28. Knight, Rich, Artificial Intelligence, 2nd, McGraw_Hill, Inc., New Yourk, 1991.

29. Kupiec, J., and J. Maxwell. 1992, Training stochastic grammars from unlabelled text corpora, In AAAI-92 Workshop Program on Statistically-Based NLP Techniques (San Jose, CA), 14-19.

30. Leslau, Wolf, Amharic Text Book, California: Berkeley University, 1986

31. Löbner, Sebastian Understanding Semantics., London: Arnold Publications, 2002.

32. McRoy, Susan Wever and Hirst, Graeme, July 1989, Running Head: race-Based parsing.

33. Mesfin Getachew. 2001. Automatic Part of Speech Tagging for Amharic Language: An Experiment Using Stochastic Hidden Markov (HMM) Approach, Master Thesis at School of Information Studies for Africa, Addis Ababa.

34. Nega Alemayehu and Willett, Petter, 2002, Stemming of Amharic Words for Information Retrieval, University of Sheffield, Sheffield, Uk http://llc.oupjournals.org/cgi/reprint/17/1/1.pdf

35. Pereira, Fernando, Sentence Modeling and Parsing, AT&T Bell Labs, Murray Hill, New Jersey, USA. http://cslu.cse.ogi.edu/HLTsurvey/ch3node8.html

36. Richardson, Stephen D., (1994), Bootstrapping Statistical Processing into a Rule-based Natural Language Parser, http://www.lirmm.fr/~lafourcade/ML-pool/TR-95-48.doc

37. Russell, Stuart & Norving, Peter, Artificial Intelligence a Modern Approach,

38. Samad, Tariq, The Kluwer international series in engineering and computer science; natural language rocessing and machine translation, Carnegie-Mellon University, Boston: Kluwer Academic Publishers, 1986.

39. Simmons, Robert F. and Yu, Yeong-Ho, The Acquisition and Application of Context Sensitive Grammar for English, http://acl.ldc.upenn.edu/P/P91/P91-1016.pdf

40. Tesfaye Bayou. 2002. Automatic Morphological Analyzer: An Experiment Using Unsupervised and Autosegmental Approach. Masters Thesis. Addis Ababa University.

41. Warren, David S., Parsing of Context Sensitive Grammars, 1999 http://www.cs.sunysb.edu/~warren/xsbbook/node31.html

42. William, O'Grady & Michael, Dobrovolsky (editor), Contemporary Linguistic Analysis: An Introduction, 3rd ed., Copp Clark Ltd, 1996.

43. Wilms, G. Jan, Using Corpus-Based Techniques to Customize a Lexicon for a New Domain http://matcscserver.uu.edu/faculty/jwilms/papers/acm93/acm93.pdf

44. Yao, Yuan; Lua, KimTeng, (1998), A probabilistic Context-Free Grammar Parser for Chinese, Department of Information Systems & Computer Science, National University of Singapore, http://WWW.comp.nus.edu.sg/~luakt/paper/199805.doc

APPENDIX

APPENDIX 1: The Inflectional form of the verb /säbbär/ 'break'

Tense	Person	Perfect		Imperfect	
		Singular	Plural	Singular	Plural
Simple Past	1st	-hu (simple perfect) /säbbärhu/'I broke'	-∂n (simple perf.) /säbbär∂n/'we broke'		
	2nd mus.	-k, -h (simple perfect) /säbbärk, säbbärh/ 'You broke'	-ačč∂hu (simple perfect) /säbbäračč∂hu/ 'You broke'		
	Fem	-š (simple perfect) /säbbärš / 'You broke'			
	Pol.	-u (simple perfect) /säbbäru / 'You broke'			
	3rd mus.	-ä (simple perfect) / säbbärä / 'He broke'	-u (simple perfect) /säbbäru/ 'They broke'		
	Fem	-äč (simple perfect) /säbbäräč / 'She broke'			
Present time with auxiliary verb /allä/	1st	-iallähu (present perfect) /säbbriallähu / 'I have broken'	-änall (present perfect) /säbbränall/ 'We have broken'	∂-allähu (present imperfect) /∂säbbrallähu/ 'I will break'	∂n- allän (present imperfect) /∂nsäbbrallän/ 'we will brake'
	2nd mus.	-oall (present perfect) /säbbroall / 'you have broken'	-ačwall (present perfect) /säbbračwall/ 'You have broken'	t∂-alläh (present imperfect) /t∂säbbralläh/'You will break'	t∂- allačhu (present imperfect) /t∂säbbrallačhu/ 'You will break'
	Fem	-äšall (present perfect) /säbbräšall/ 'you have broken'		t∂-ialläš (present imperfect) /t∂säbbrialläš/'You will break'	
	Pol.	-äwall (present perfect) /säbbräwall/ 'You have broken'		y∂-allu (present imperfect) /y∂säbbrallu/'You will break'	
	3rd musc.	-owall (present perfect) /säbbrowall/ 'He has broken'	-äwall (present perfect) /säbbräwall/ 'They have broken'	y∂-all (present imperfect) /y∂säbbrall/'He will break'	y∂- allu (present imperfect) /y∂säbbrallu/'The y will break'
	Fem	-allč (present perfect) /säbbrallč/ 'She has broken'		t∂-allč (present imperfect) /t∂säbbrallč/'She will break'	

Tense	Person	Perfect		Imperfect	
		Singular	Plural	Singular	Plural
Past time with auxiliary verb /näbär/	1st	-e näbär (past perfect) /säbbre näbbär/ 'I had broken'	-än näbär (past perfect) /säbbrän näbbär/ 'We had broken'		
	2nd mus.	-äh näbbär (past perfect) /säbbräh näbbär/ 'You had broken'	-ačhu näbär (past perfect) /säbbračhu näbbär/ 'You had broken'		
	Fem	-š näbär (past perfect) /säbbräš näbbär / 'You had broken'			
	Pol.	-äw näbär (past perfect) /säbbräw näbbär/ 'You had broken'			
	3rd musc.	-o näbär (past perfect) /säbbro näbbär / 'He had broken'	-äw näbär (past perfect) /säbbräw näbbär / 'They had broken'		
	Fem	-a näbär (past perfect) /säbbra näbbär / 'She had broken			

Tense	Person	Present		Past	
		Singular	Plural	Singular	Plural
continuous time	1st	∂yä-ku näw (Present contin.) /∂yäsäbbärku näw/ 'I am breaking'	∂yä-n näw (Present contin.) /∂yäsäbbärn näw/ 'We are breaking'	∂yä-ku näbbär (past continuous) /∂yäsäbbärku näbbär/ 'I was breaking'	∂yä-n näbbär (past continuous) /∂yäsäbbärn näbbär/ 'we were braking'
	2nd mus.	∂yä-k näw (Present contin.) /∂yäsäbbärk näw/ 'You had broken'	∂yä-aču näw (Present contin.) /∂yäsäbbäraču näw/ 'You are breaking'	t∂- näbär (past continuous) /∂yäsäbbärk näbbär/ 'You were breaking'	∂yä-aču näbbär (past continuous) /∂yäsäbbäraču näbbär/ 'You were breaking'
	Fem	∂yä-š näw (Present contin.) /∂yäsäbbärš näw/ 'You had broken'		∂yä-š näbär (past continuous) /∂yäsäbbärš näbbär/ 'You were breaking'	
	Pol.	∂yä-u näw (Present contin.) /∂yäsäbbäru näw/ 'You had broken'		∂yä-u näbbär (past continuous) /∂yäsäbbäru näbbär/ 'You were breaking'	
	3rd musc.	∂yä-ä näw (Present contin.) /∂yäsäbbärä näw/ 'He is breaking'	∂yä-u näw (Present contin.) /∂yäsäbbäru näw/ 'They are breaking'	∂yä-ä näbbär (past continuous) /∂yäsäbbärä näbbär/ 'He was breaking'	∂yä-u näbbär (past continuous) /∂yäsäbbäru näbbär/ 'They were breaking'
	Fem	∂yä-č näw (Present contin.) /∂yäsäbbäräč näw/ 'She had broken		∂yä-č näbbär (past continuous) /∂yäsäbbärč näbbär/ 'She was breaking'	

APPENDIX 2: Phonemic symbols used for Amharic fidel.

Basic letters	ä	u	i	a	e	ə	o
ሀ	hä	hu	hi	ha	he	hə	ho
ለ	lä	lu	li	la	le	lə	lo
መ	mä	mu	mi	ma	me	mə	mo
ሰ	sä	su	si	sa	se	sə	so
ረ	rä	ru	ri	ra	re	rə	ro
ሸ	šä	šu	ši	ša	še	šə	šo
ቀ	qä	qu	qi	qa	qe	qə	qo
በ	bä	bu	bi	ba	be	bə	bo
ተ	tä	tu	ti	ta	te	tə	to
ቸ	čä	ču	či	ča	če	čə	čo
ነ	nä	nu	ni	na	ne	nə	no
ኘ	ňä	ňu	ňi	ňa	ňe	ňə	ňo
አ	ʔä	ʔu	ʔi	ʔa	ʔe	ʔə	ʔo
ከ	kä	ku	ki	ka	ke	kə	ko
ወ	wä	wu	wi	wa	we	wə	wo
ዘ	zä	zu	zi	za	ze	zə	zo
ዠ	žä	žu	ži	ža	že	žə	ž o
የ	yä	yu	yi	ya	ye	yə	yo
ደ	dä	du	di	da	de	də	do
ጀ	jä	ju	ji	ja	je	jə	jo
ገ	gä	gu	gi	ga	ge	gə	go
ጠ	ťä	ťu	ťi	ta	ťe	ťə	ťo
ጨ	čä	ču	či	ča	če	čə	čo
ጰ	pä	pu	pi	pa	pe	pə	po
ጸ	s'ä	s'u	s'i	s'a	s'e	s'ə	s'o
ፈ	fä	fu	fi	fa	fe	fə	fo

APPENDIX 3: Sample Sentences

Since the size of this thesis exceeds the maximum size, 150 pages, Graduate School of AAU allows, Appendix 3 is compiled separately and made available at the Bibliographic Laboratory of Informatics Faculty, Addis Ababa University.

APPENDIX 4: Parts of speech used in the taggar.

IPP	Independent Personal Pronouns	1
DDP	Demonstrative Determiner Pronoun	2
SDP	Selective Determiner Pronoun	3
IP	Interrogative pronoun	4
IDA	Indefinite Pronouns as adjective	5
PN	Proper Noun	6
PERP	Personal pronoun	7
CN	Concrete Noun	8
CON	Compound Noun	9
AN	Abstract Noun	10
NADV	Noun as an adverb	11
NQ	Noun as quantifier	12
VN	Verbal Noun	13
CNo	Cardinal Number	14
AUX	Auxiliary Verbs of be	15
VE	Verbs of Existence	16
TV	Action Verbs that are Transitive	17
IV	Action Verbs that are Intransitive	18
AVO	Action Verbs with out Complement	19
DV	Derivational verb	20
ADJ	Adjective	21
ADV	Adverb	22
PRE	Preposition	23
POP	Postposition	24
COMP	Complementary	25
IDP	Indefinite Article	26
IJ	Inter junction	27
CC	Coordinate Conjunction	28
DR	Determiner	29

APPENDIX 5: Total parts of speeches used in the morphological analysis.

WORD CLASS			TAG SET	DESCRIPTION	EXAMPLE
	NOUNS	PRONOUN	IPP	Independent Personal Pronouns	∂ne, ∂su, ∂rsu, anči, ∂rsuwa,
			DDP	Demonstrative Determiner Pronoun	y∂h, y∂he, y∂h∂n, y∂čči, ∂čč ∂zziya, ∂näzziya
			SDP	Selective Determiner Pronoun	y∂hňaw, yahañaw, ∂näzziyañoču
			IP	Interrogative pronoun	man, ma, m∂n, m∂nd∂r, m∂nd∂n
			IDP	Indefinite Pronouns as adjective	mann∂mm, yät∂ňňaw∂mm
			PN	Proper Noun	gojam, gondar, addis ab∂ba, medrok
			PERP	Personal pronoun	abäbä, kasa, aster, dawit
			AN	Abstract Noun	m∂sgana, kisara, ekonomi,
			CN	Concrete Noun	l∂j, färäs, lam, ∂nsäsa, bare, bäg
			*CON	Compound Noun	awudäry,
			DN	Derived Noun	säbbära,
			NADV	Noun as an adverb	zare, t∂lan∂t,
			NQ	Noun as quantifier	hulät, sost gize
			VN	Verbal Noun	mäsrak', mät'∂fat, mätäw
			*CCN	Coordinate Conjunction Noun	wändočča, setčča
		NUMERALS	CNo	Cardinal Number	and, ass∂r
			*ON	Ordinal Number	and-äňňa, ass∂r-äňňa
	VERB		*GV	Gerund Verbs	säbb∂ro, näg∂ro, säbb∂rä, sät't'∂ta, näg∂ra
			*DV	Derived verb	asäbbär, asabbär, säbabbär, täsäbbär, täsäbabbär, täsabbär, täläyayu
			AUX	Auxiliary Verbs of be	näbbär, jämmär, näw, y∂mäs∂lal, hon
			VE	Verbs of Existence	allä, alläňň, alläw, norä, walä

* New part of speech added during morphological analysis

		TAG SET	DESCRIPTION	EXAMPLE
		TV	Action Verbs that are Transitive	säbbrä, bälla, gäzza, bäl
		IV	Action Verbs that are Intransitive	gäba, hedä, wät'a
		AVO	Action Verbs with out Complement	aläqäs, c'c'oh,
		ADJ	Adjective	t∂ll∂k', räjm
		ADV	Adverb	tollo, gäna, k∂fuňa,
		PRE	Preposition	lä, kä, bä, wädä,
		POP	Postposition	gar∂, dräs, w∂st'
		COMP	Complementary	s∂lä, ∂ski, ∂ndä, ∂ndäy∂,
		CC	Coordinate Conjunction	∂nna, na,
		DR	Determiner	bat'am, l∂k,
		*GNP	Genitive noun phrase	yäsar, yät∂nant, yäktäma, yägonder, yäne
		*CV	Complementized Verb	s∂lähedä, ∂ndämät'a
		*PP	Prepositional Phrase	käbet, läkäsa, ∂ndäsäw

* New part of speech added during morphological analysis

APPENDIX 6: Sample output of the stemmer.

Word	Stemmer Output	Actual Stem	Error Type
mäsrat	mäsrat	sra	under stemming
bäkäftäña	käft	käfta	over stemming
lämäftat	mäftat	fät	require deep structure & under stemming
täqäbaynät	täqäbay	täqäbäl	require deep structure analysis
yawqal	awq	awäq	require deep structure analysis
f?llagot	f?llagot	fälläg	require deep structure analysis
däng?t'o	däng?t'	dänägät'	require deep structure analysis
mänoria	mänor	nor	under stemming
tatari	tatar	tatär	require deep structure analysis
?yat'äbä	t'äb	at'äb	over stemming
shäbäla	shäbäl	shäbäla	over stemming
yäsamrawit	samra	samrawit	over stemming
zäbäña	zäbä	zäbäña	over stemming
y?mäslal	mäsl	mäsäl	require deep structure analysis
akbari	akbar	akäbär	require deep structure
l?tt'äy?qäw	t'äy?q	t'äyäq	require deep structure
?ndämit'?ru	t'?r	t'ar	require deep structure
asälät'änu	asälät'än	sälät'	under stemming
?ndiad?g	ad?g	adäg	require deep structure analysis
adärägä	adäräg	däräg	under stemming

APPENDIX 7: Sample output of the prototype parser.

S[NP[MNP(SNP((PERP ayälä)))]VP[MVP(MADJP(SADJP(ADJ t?n?sh))SVP(MNP(SNP(CN injära)) (TV (SOTV bälla)))]]

S[NP[MNP(SNP((PERP aster)))]VP[MVP(SVP(MPP(SPP((PP läkasa)))MNP(SNP((CN mäs'haf)))TV ((DOTV sät'äççw)))]]

S[NP[S'(MNP(SNP(GNP yäkasa))MVP(SVP(MNP(SNP(CN bet))((CV yämäsrat))))SNP(AN alama)))]VP[MVP(SVP(TV täsaka))]]

S[NP[S'(MNP(SNP((PERP kasa)))MVP(SVP(MNP(SNP(CN bäg))((CV yarädäbät))))SNP(AN bilawa)))]VP[MVP(DR bät'am)(SVP(MADJP (SADJP(ADJ dänäz))(AUX näw)))]

S[NP[MNP(SNP((PERP kasa)))]VP[MVP(S'(MPP(SPP((PP kägojam))) MVP(SVP((CV ?ndämät'a))))SVP(S'(MNP(SNP((PERP aster))) MVP(MPP(SPP((PRE wädä)MNP(SNP((PN nazz?ret)))))SVP((CV ?ndähedäçç))))((TV säma)))))]]

S[NP[MNP(SNP((PERP kasa)))]VP[MVP(S'(MPP(SPP((PP bäççkola))) MVP(MNP(SNP((CN qursun))) SVP((GV ballto))))SVP(MPP(SPP((PRE wädä) MNP(SNP(JN((AN t?m?h?rt))(CN bet))(IV hedä)))))]]

S[NP[MNP(SNP((PERP aster)))]VP[MVP(SVP(MPP(SPP((PP läayälä))) MNP(SNP((AN ?rdata)))TV((DOTV adärägäççlät))]]

S[NP[MNP(SNP((PERP aster)))]VP[MVP(S'(MNP(SNP((CN l?joççu))) MVP(MPP(SPP((PRE wädä)MNP(SNP(JN((AN t?m?h?rt)))(CN bet)) SVP((CV ?ndähedu))))SVP(S'(MNP(SNP((PERP kasa)))MVP(SVP((VN mäshomun))))(TV sämaçç))))]]

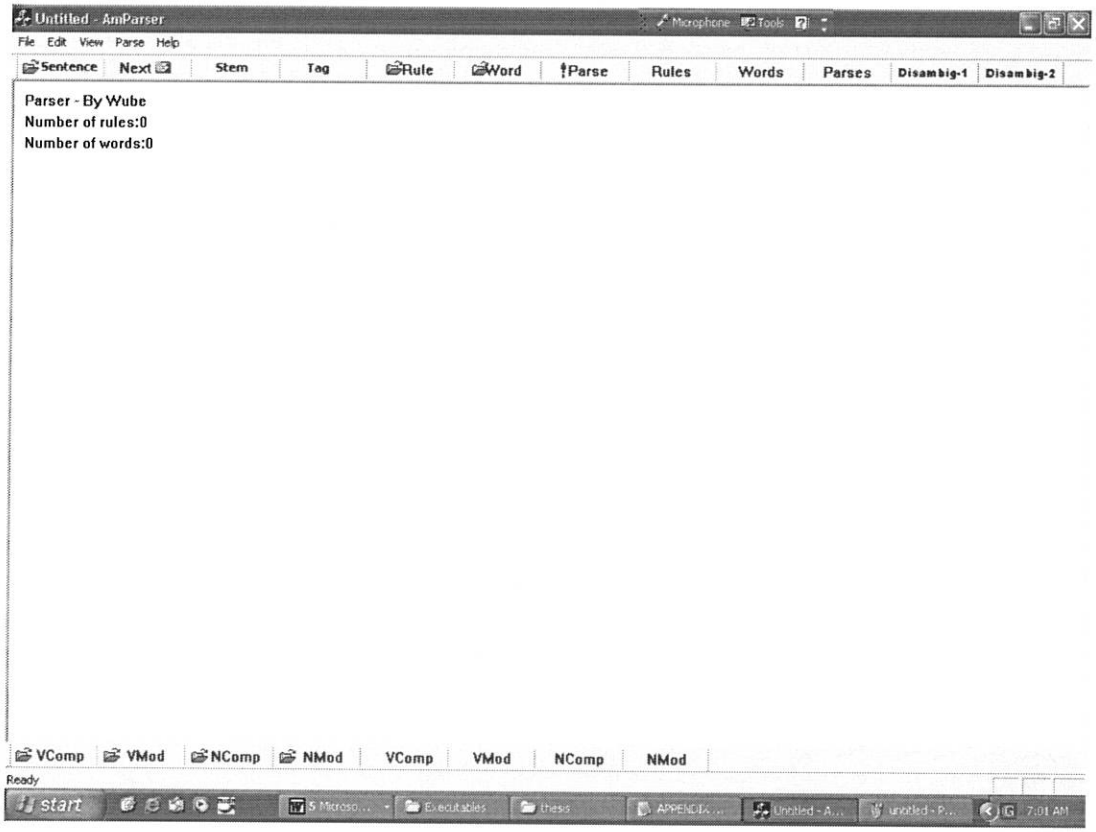
S[NP[S'(MNP(SNP((PN ityopia)))MVP(SVP((CV yaqäräbäççw))))((GNP yäw?y?y?t)SNP((AN häsab)))))]VP(MVP(SVP(MNP(SNP((VN täqäbaynät)))((alläw)))))]]

*S[NP[MNP(MPP(SPP((PP bäa?mro)))SNP((VN f?lsät))))]VP[MVP (S'(MNP(SNP((PN af?rica)))MVP(SVP(MPP(SPP((PP bäyäamätu) SPP((PP käarat)))((CNo biliyon)SNP((CN b?r)))SPP((PP bälay))))((CV ?yat'açç)))))SVP((AUX näw)))]]

S[NP[MNP(S'(MNP(SNP((AN mäng?st)))MVP(MNP(SNP((AN ççg?roççn))) (MNP(MPP(SPP((PP lämäftat)))SVP((CV yämikätäläw))))SNP((CN mängäd))))] VP(MVP(MPP((PP kähulum)SPP((PP bälay)))SVP(MNP(SNP((AN asasabi))((CC ?nna))((AN asafari)))((AUX näw)))))]]

* wrong parse

APPENDIX 8: A Screen Snapshot of the Parser.



Declaration

The thesis is my original work, has not been presented in any other universities and that all sources of material used in the thesis have been duly acknowledged.

Wube Alemayehu Tuffa

January 2005

The thesis has been submitted for examination with our approval as university advisor.

Dr. Lina Zhou

Ato Getahun Amare

Ato Aklilu Yielma

Declaration

The thesis is my original work, has not been presented in any other universities and that all sources of material used in the thesis have been duly acknowledged.

Wube Alemayehu Tuffa

January 2005

The thesis has been submitted for examination with our approval as university advisor.

Dr. Lina Zhou

Ato Getahun Amare

Ato Aklilu Yielma