DESIGN OF
HEAD-DRIVEN PHRASE STRUCTURE GRAMMAR
FOR BANGLA

A Thesis

Submitted to the Department of Computer Science and Engineering

of

BRAC University

by

M. Hammad Ali

Student ID: 02201076

In Partial Fulfillment of the

Requirements for the Degree

of

Bachelor of Science in Computer Science and Engineering

Fall 2006

BRAC University, Dhaka, Bangladesh

DECLARATION

I hereby declare that this thesis is based on the results found by myself and my team members. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

---------- ----------

Signature of Supervisor Signature of Author

ACKNOWLEDGMENTS

Special thanks to my supervisor Mumit Khan without whom this work would have been very difficult. Thanks to Naira Khan for providing all the linguistic knowledge that was required for this work. Also special thanks to Dr. Emily Bender, who has managed to take out some time from her busy schedule to give feedback about the implementation of this work all the way from Washington.

ABSTRACT

A considerable amount of work is being done in recent years in the arena of Computational Linguistics. However, little has been done in developing a generic computational grammar for the Bangla language that could be used for advanced language processing. Context Free Grammar (CFG) has been attempted for English, and it has been realized that this does not have the descriptive power required to model a natural language. The chief reason behind this is the large number of rules that are needed in order to capture the vast array of natural language phenomenon. Our target is to try and implement a Head-driven Phrase Structure Grammar (HPSG) for Bangla. We would like to find out a pattern behind the various phenomenon of the language and try to capture these in the form of generalizations for the different entities of the language. We would also like to explore the underlying patterns governing the successful unification of feature structures. Eventually, we want to end up with a small scale working implementation of a Bangla HPSG that can recognize and generate a set of representative sentences from the Bangla language, accurately capturing relevant linguistic phenomenon.

To the martyrs of The Language Movement of 1952…thank you

Table of Contents

List of Figures

Introduction

Bangla is the language of nearly 210 million people, and the 4th most widely spoken language in the world. It is the primary language of the people of Bangladesh and that of the Indian State of West Bengal. It belongs to the Aryan family of languages and is written in the Brahmi-derived Bangla script. Bangla underwent a period of vigorous Sanskritization that started in the 12th century and continued throughout the middle ages [13]. Bangla lexicon today consists of *tatsama* (Sanskrit words that have changed pronunciation, but have retained the original spelling), *tadbhava* (Sanskrit words that have changed at least twice in the process of becoming Bangla), and a fairly large number of "loan-words" from Persian, Arabic, Portugese, English and other languages. There are also a large number of words of unknown etymology. Despite this wide usage and rich diversity of words in the Bangla language, Bangla is yet in its infant stage at least as far as work in the area of computational linguistics is concerned. While languages like English and even Hindi are rapidly progressing in terms of work done in processing by computers, Bangla lags behind in crucial areas of research like parts-of-speech tagging, text summarization and categorization, information retrieval and most importantly in the area of computational grammar.

The computational grammar for a language has a wide variety of applications. It can be used to implement a grammar checker in word processing software, like Microsoft word currently has for English. It can also be used as a platform upon which to try and develop text summarization and information extraction tools. These types of tools can greatly benefit from knowledge of what constituents of a language can co-occur and where the most important information in a sentence is usually encoded. In future, computational grammars can also be a big help in the design of software that will be able to edit natural language written by humans, as envisioned by Google. On a similar note, a computational grammar can also be useful in work involving machine translation. In such cases, a grammar could be used to check whether or not the sentences generated by the translation mechanism should be considered well-formed for the language.

Background

As already mentioned, little or no work has been done for a computational grammar for the Bangla language. In a course conducted by Dr. Emily Bender at the University of Washington, students have to pick a language and design a *Head-driven Phrase Structure Grammar* for it (HPSG). Last Spring, one of the graduate students in the class, Jeremy Kahn, designed an HPSG for Hindi, which is a language that has several similarities with Bangla. We also tried to implement a context-free grammar (CFG) for the Bangla language as the project for a course on Natural Language Processing. However, continuing work with CFG did not seem feasible due to reasons we will discuss later in this text. So, we moved to working on an HPSG for Bangla, about which we will have more to say in the later sections.

Section 1: *Generalized Overview*

Primary contribution

by

Ayesha Binte Mosaddeque

Section 1: Generalized Overview

1.1 Grammar

The mental system of rules and categories that allows humans to form and interpret the words and sentences of their language is called a grammar [2]. The sounds and sound patterns, the basic units of meaning, such as words, and the rules to combine them to form new sentences constitute the grammar of a language. We use the term "grammar" with a systematic ambiguity. On one hand, the term refers to the explicit theory constructed by a linguist and proposed as a description of the speaker's competence. On the other hand, it refers to this competence itself [1]. A grammar that seeks to describe human linguistic ability and knowledge and not to prescribe one system in preference to another is called a descriptive grammar. A prescriptive grammar, on the other hand, aims to state the linguistic facts in terms of how they should be.

Grammars are used to:

- Categorize speaker's social and economic class and also their stage of learning
- Recognize the level of formality of a particular usage
- Understand how people construct and comprehend language
- Identify resemblance and divergence across speeches
- Gauge the progress of language technologies

1.2 Syntactic category

Syntactic category is the grouping into which a component is located depending on the type of the meaning that it conveys, the type of affixes it can take and the type of formation in which it can occur. An elementary fact about words in all human language is that they can be assembled together into a relatively small number of classes, called syntactic categories. This classification reflects a variety of factors, including those mentioned above about their meaning, affixes and where they can or cannot occur.

Table 1 provides examples of the word-level categories that are most central to the study of syntax.

Table 1

Lexical categories	Examples
Noun	ছেলে/cʰele, আয়শা/ajʃa
Pronoun	সে/ʃe, তিনি/t̪ini
Verb	বলা/bɔla,
Adjective	ভালো/bʰalo, বৃদ্ধ/bri̪d̪d̪ʰo
Non lexical categories	Examples
Determiner	এই/ei, এটা/eta, টি/ti
Qualifier	সবসময়/ʃɔbʃomoj, হয়তো/hɔj̪to
Conjunction	এবং/eboŋ, অথবা/ɔt̪ʰoba, কিন্তু/kint̪u

The syntactic categories of Bangla grammar are explained below.

1.2.1 Noun [বিশেষ্য]

A lexical category that typically names entities; can usually be inflected for number and possession, and functions as the head of a noun phrase. The noun class is further divided into the following subclasses [5]-

1.2.1.1 Proper Noun [নামবাচক]

Proper noun is a noun that names a person, a geographical place or definition and books. For example, রহিম/rohim, ঢাকা/dʰaka etc.

1.2.1.2 Common Noun [জাতিবাচক]

A common noun is a noun referring to a person, place, or thing in a general sense. A common noun is the opposite of a proper noun. For example, নদী/no̪di, মানুষ/manuʃ etc.

1.2.1.3 Material Noun [বস্তুবাচক]

A concrete noun is a noun which names any entity that we can perceive through our physical senses: touch, sight, taste, hearing, or smell. For example, বই/boi, পানি/pani etc.

1.2.1.4 Collective Noun [সমষ্টিবাচক]

A collective noun is a noun naming a group of things, animals, or persons. We can count the individual members of the group, but we usually think of the group as one unit. For example, দল/d̪ɔl, সমিতি/ʃɔmiti etc.

1.2.1.5 Verbal Noun [ভাববাচক]

A noun that expresses any work being done is named as a Verbal Noun. For example, যাওয়া/ɟaoja (to go), দেখা/d̪ækʰa (to see) etc.

1.2.1.6 Abstract Noun [গুণবাচক]

An abstract noun is a noun which names anything which we can not perceive through the five physical senses. For example, সুখ/ʃukʰ, বীরত্ব/biro̪t̪to etc.

1.2.2 Pronoun [সর্বনাম]

Pronoun is a lexical category whose members can replace a noun phrase and look to another element for their interpretation. The latter element is known as the antecedent of the pronoun. The pronoun class is further divided into the following subclasses –

1.2.2.1 Personal Pronouns

A personal pronoun refers to a specific person or thing and changes its form to indicate person, number, gender, and case. Divisions of personal pronouns are as follows-

1.2.2.2 Subjective Personal Pronouns [সামীপ্যবাচক]

A subjective personal pronoun indicates that the pronoun is acting as the subject of the sentence For example, আমি/ami, তারা/t̪ara etc.

1.2.2.3 Objective Personal Pronouns [দূরত্ববাচক]

An objective personal pronoun indicates that the pronoun is acting as an object of a verb, compound verb, preposition, or infinitive phrase. For example, তাকে/t̪ake, তার/t̪ar etc.

1.2.2.4 Possessive Personal Pronouns [আত্মবাচক]

A possessive pronoun indicates that the pronoun is acting as a marker of possession and defines who owns a particular object or person. For example, স্বয়ং/ʃɔjoŋ, খোদ/kʰod̪ etc.

1.2.2.5 Demonstrative Pronouns [সাকল্যবাচক]

A demonstrative pronoun points to and identifies a noun or a pronoun. For example, সব/ʃɔb , সকল/ʃɔkol etc.

1.2.2.6 Interrogative Pronouns [প্রশ্নবাচক]

An interrogative pronoun is used to ask questions. For example, কে/ke, কার/kar etc.

1.2.2.7 Relative Pronouns [সংযোগজ্ঞাপক]

We use a relative pronoun to link one phrase or clause to another phrase or clause. For example, যে/ɟe, যারা/ɟara etc.

1.2.2.8 Indefinite Pronouns [অনির্দিষ্টতাজ্ঞাপক]

An indefinite pronoun is a pronoun referring to an identifiable but not specified person or thing. An indefinite pronoun conveys the idea of all, any, none, or some. For example, কেউ/keu, কিছু/kicʰu etc.

1.2.2.9 Intensive Pronouns [ব্যতিহার]

An intensive pronoun is a pronoun used to emphasize its antecedent. Intensive pronouns are identical in form to reflexive pronouns. For example, নিজে/nije নিজে/niɟe, আপনাআপনি/apnapni etc.

1.2.3 Verb [ক্রিয়া]

A lexical category that typically designates actions, sensations, and states; can usually be inflected for tense; and functions as the head of a verb phrase. The verb class is divided into the following subclasses with regard to the meaning expressed.

1.2.3.1 Finite Verb [সমাপিকা ক্রিয়া]

A finite verb is a verb that is inflected for person and for tense according to the rules and categories of the language in which it occurs. Finite verbs can form independent clauses, which can stand by their own as complete sentences. For example, নাফিদ/napʰid̪ বই/boi পড়ে/pɔɾe.

1.2.3.2 Non-finite Verb [অসমাপিকা ক্রিয়া]

In linguistics, a non-finite verb (or a verbal) is a verb form that is not limited by a subject; and more generally is not fully inflected by categories that are marked inflectionally in language, such as tense, aspect, mood, number, gender, and person. As a result, a non-finite verb cannot generally serve as the main verb in an independent clause; rather, it heads a non-finite clause. For example, আমরা/amra খেলতে/kʰelt̪e, here the verb "খেলতে/kʰelt̪e" does not complete the sentence. It requires some other word to make a complete sense of the sentence. For example, if we rewrite the sentence as, আমরা/amra খেলতে/kʰelt̪e গেলাম/gelam. Now this sentence has a complete meaning. Depending on whether a verb takes an object or not the verb class has the following subclasses.

1.2.3.3 Transitive verb [সকর্মক ক্রিয়া]

In syntax, a transitive verb is a verb that requires both a subject and one or more objects. For example, নাফিদ/napʰid̪ বই/boi পড়ে/pɔɽe. Here the transitive verb "পড়ে/pɔɽe" has object "বই/boi". Another example, বাবা/baba আমাকে/amake কলম/kɔlom দিয়েছেন/dijecʰen. Here the verb "দিয়েছেন/dijecʰen" takes two objects- "কলম/kɔlom" is the direct object and "আমাকে/amake" is the indirect object. Transitive verbs that are able to take both a direct object and an indirect object are called ditransitive verbs.

1.2.3.4 Intransitive verb [অকর্মক ক্রিয়া]

An intransitive verb is a verb that has only one argument, that is, a verb with valency equal to one. In more familiar terms, an intransitive verb has a subject but does not have an object. For example, ঘুমায়/gʰumaj, হাসে/haʃe etc.

1.2.4 Adjective [বিশেষণ]

A lexical category that typically designates a property that is applicable to the entities named by nouns, can often take comparative and superlative endings, and functions as the head of an adjective phrase. For example, লাল/lal ফুল/pʰul.

1.2.5 Conjunction [অব্যয়]

Conjunction is the syntactic category that joins two parts of a sentence. For example, এবং/eboŋ, কিন্তু/kin̪tu etc.

1.2.6 Case

Case is usually manifested through inflection on the nouns and pronouns. Since case provides a way to mark the roles played by elements in a sentence, languages with a rich case system have relatively free word order. Bangla would be a good example. For instance,

"নাফিদ/napʰid̪ হাম্মাদ/hammad̪-কে/ke বই/boi দেয়/dej"

and

“হাম্মাদ/hammaḍ-কে/ke নাফিদ/napⁿiḍ বই/boi দেয়/dej”

convey the same sense, since the roles played by the nouns (napⁿiḍ and hammaḍ) are indicated by the case affixes that they take and do not rely on the ordering of the words.

In Bangla grammar, there are six different cases:

1.2.6.1 Nominative (কর্তৃকারক):

The nominative case is the usual, natural form of certain parts of speech, such as nouns, adjectives, pronouns and less frequently numerals and participles, and sometimes does not indicate any special relationship with other parts of speech. Therefore, in some languages the nominative case is unmarked, that is, the nominative word is the base form or stem, with no inflection; alternatively, it may be said to have been marked by a zero morpheme. Moreover, in most languages with a nominative case, the nominative form is the lemma; that is, it is the one used to cite a word, to list it as a dictionary entry, etc.

Possible case markers are- +0, কে/ke, দ্বারা/ḍara, র/r, ে /e , য়/j, তে/ṭe.

1.2.6.2 Accusative (কর্মকারক):

The accusative case of a noun is the grammatical case used to mark the direct object of a transitive verb. The same case is used in many languages for the objects of (some or all) prepositions.

Possible case markers are- +0, কে/ke, রে/re, র/r, ে /e

1.2.6.3 Dative (সম্প্রদানকারক):

The dative case is a grammatical case generally used to indicate the noun to whom something is given. The dative generally marks the indirect object of a verb, although in some instances the dative is used for the direct object of a verb pertaining directly to an act of giving something.

Possible case markers are- কে/ke, ে /e

1.2.6.4 Locative (অধিকরণকারক):

Locative is a case which indicates a location. It corresponds vaguely to the English prepositions "in", "on", "at", and "by". The case is often used to distinguish between describing a place and going there.

Possible case markers are- +0, দিয়ে/dije, থেকে/t̪ʰeke, তে/t̪e, ে /e

Instrumental Locative (করণ কারক):

Possible case markers are- +0, দ্বারা/d̪ara, দিয়া/d̪ija, ে /e , য়/j, তে/t̪e

1.2.6.5 Ablative (অপাদানকারক):

In linguistics, ablative case is the name given to the case in various languages whose common thread is that they mark motion away from something, though the details in each language may differ. The name "ablative" is derived from a Latin verb meaning "to carry away".

Possible case markers are- +0, কে/ke, এর/er, ে /e, য়/j

1.2.6.6 Genitive (সম্বন্ধপদ):

In grammar, the genitive case or possessive case is the case that marks a noun as being the possessor of another noun. The genitive case typically has other uses as well, which can vary from language to language: it can typically indicate various relationships other than possession; certain verbs may take arguments in the genitive case; and it may have adverbial uses.

Possible case marker is - র/r.

Section *2: More into Linguistics*

Primary contribution

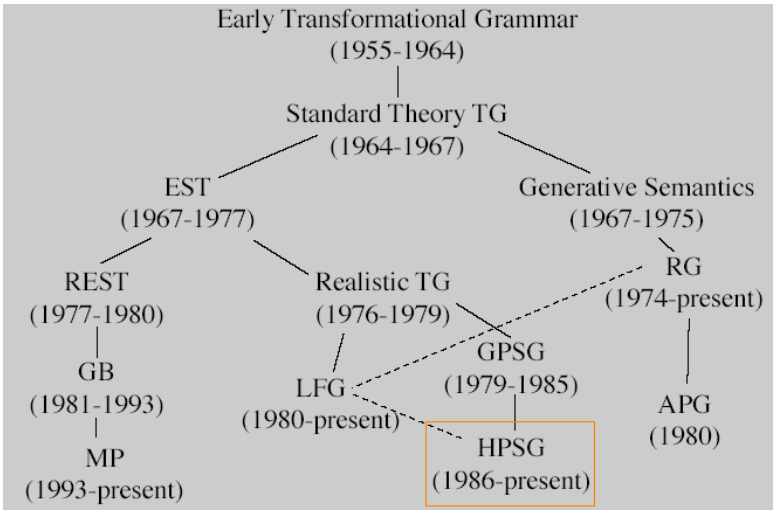
by

Ayesha Binte Mosaddeque

Section 2: More into Linguistics

Writings on grammar started at least 3000 years ago. Until 200 years ago, almost all of it was prescriptive. Until 50 years back, most linguistic works were about sound systems (phonology), word structure (morphology), and the historical relationships among languages. Noam Chomsky's work in the 1950s drastically changed linguistics, making syntax the central component of grammar. Chomsky has been the leading figure in linguistics ever since. The hierarchy of the syntactic theories is shown in Figure 1 [8].

Figure 1 Family Tree of Syntactic Theories



2.1 Phrase

A phrase is a unit of syntactic structure that is built by combining words together so that the phrase consists of a head and an optional specifier and/or complement. A phrase or a constituent is made up of one or more words. Sentences have a hierarchical design in which words are grouped together into successively larger structural units. These structural units are called the 'Phrase Structure'. These phrases are built around the largest unit of syntactic analysis which is the sentence (S).

Head is the lexical category around which a phrasal category is built. A head of a phrase is the driving word of that phrase. A noun phrase can be thought of to be revolving around a head, the central noun in the noun phrase [3]. So a head is a noun in the case of

noun phrases, a verb in the case of verb phrases, and so on. Although phrases usually consist of two or more words, a head may form a phrase all by itself. For example, books (NP), eat (VP) etc. This traditional analysis assumes that S is special in not having an internal structure like other phrases.

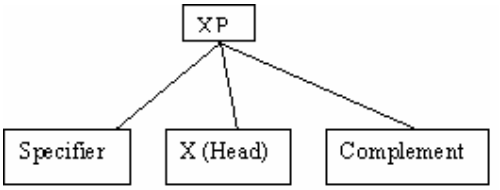
2.2 Phrase Structure Grammar

A grammar that is driven by the phrase structure rules is called a phrase structure grammar. One or more words that make up a syntactic unit are called a constituent. Phrase structure rules specify how a syntactic constituent is formed out of other smaller syntactic constituents. In many languages, groups of consecutive words act as a group or a constituent, which can be modeled by Phrase Structure Grammar.

The XP rule:

XP -> (Specifier) X (Complement), where X stands for N, V, A, or P

Figure 2 The Phrase Structure Schema



A specifier is a word that makes the meaning of the head more precise. The complement is one or more words that convey more information about the entities and location of the head of a phrase. In a phrase structure grammar the specifier and complement are optional but the head is mandatory.

2.3 Conventional CFG

The most commonly used and mathematical system for modeling constituent structure in natural languages is the Context-Free Grammar, or CFG. Context-free grammars are also called Phrase Structure Grammars. A context-free grammar (CFG) is a set of recursive rewriting rules called productions used to generate string patterns. A CFG has four components. They are 1) a set of terminal symbols that are characters that appear in the strings generated by the grammar, 2) a set of non-terminal symbols that are placeholders

for patterns of terminal symbols that can be generated by the non-terminal symbols, 3) a set of productions that are used to replace the non-terminals with other non-terminals or terminals and 4) a start symbol for the grammar [7]. A formal language is context-free if there is a context-free grammar that generates it. For example, the following productions expresses that an NP (or Noun Phrase), can be composed of an NP followed by a Verb Phrase (VP) or a Pronoun followed by a NP or just a Noun. A VP is composed of a Verb.

NP -> NP VP

NP -> Pronoun NP | Noun

VP -> Verb

Context-free rules can be hierarchically embedded, so we have to combine the previous rules with others like these which express facts about the lexicon:

Noun -> ভাত/b$^{\text{fi}}$at

Pronoun -> আমি/ami

Verb -> খাই/k$^{\text{fi}}$ai

The symbols that are used in a CFG are divided into two classes. The symbols that correspond to words in the language are called terminal symbols. The lexicon is the set of rules that introduce these terminal symbols. The symbols that express clusters or generalizations of these are called non terminals.

A parse tree for a grammar is a tree where the root of is the start symbol for the grammar, the interior nodes are the non-terminals of the grammar, the leaf nodes are the terminals of the grammar and the children of a node starting from the left to the right correspond to the symbols on the right hand side of some production for the node in the grammar. Every valid parse tree represents a string generated by the grammar [6]. Parsing is a method where a parser algorithm is used to determine whether a given input string is grammatical or not for a given grammar.

A CFG is thought of in two ways: as device for generating sentences, or as a device for assigning a structure to a given sentence. While designing our Bangla grammar using

CFG, we found some shortcomings of CFG. (These problems have been discussed in the "background" section.) Below is a brief description of the problem.

Let us consider the above grammar. This grammar can parse the sentence "আমি/ami ভাত/b$^{\text{fi}}$at খাই/k$^{\text{fi}}$ai". If we want to increase the coverage of the grammar, we might want to parse a few more sentences. Look at the following sentences:

তুমি/tumi ভাত/b$^{\text{fi}}$at খাও/k$^{\text{fi}}$ao

সে/ʃe ভাত/b$^{\text{fi}}$at খায়/k$^{\text{fi}}$aj

To parse these sentences first we need to provide the lexicon support.

Pronoun -> আমি/ami | তুমি/tumi | সে/ʃe

Verb -> খাই/k$^{\text{fi}}$ai | খাও/k$^{\text{fi}}$ao | খায়/k$^{\text{fi}}$aj

However this would still not be sufficient because in Bangla the verb changes its form according to the person of the noun. The example grammar will parse sentences like *তুমি/tumi ভাত/b$^{\text{fi}}$at খাই/k$^{\text{fi}}$ai. So we have to put in additional constraints. In CFG the only way to do it is by adding more rules. So for this example grammar we can modify the grammar as follows.

NP1 -> Pronoun1 NP| Noun

NP2 -> Pronoun2 NP| Noun

NP3 -> Pronoun3 NP| Noun

VP1 -> Verb1

VP2 -> Verb2

VP3 -> Verb3

NP1 -> NP1 VP1

NP2 -> NP2 VP2

NP3 -> NP3 VP3

We now need to provide lexicon support to parse the two other sentences.

Pronoun1 -> আমি/ami

Pronoun2 -> তুমি/tumi

Pronoun3 -> সে/ʃe

Verb1 -> খাই/k^{h}ai

Verb2 -> খাও/k^{h}ao

Verb3 -> খায়/k^{h}aj

So we find out that the number of rules increases tremendously. There are simpler, more elegant solutions that take us out of the CFG framework. We will use feature structures and the constraint-based unification formalism.

2.4 Later Techniques

Generalized phrase structure grammar (GPSG) is a framework for describing the syntax and semantics of natural languages. GPSG was initially developed in the late 1970s by Gerald Gazdar. Other contributors include Ewan Klein, Ivan Sag, and Geoffrey Pullum. Their book, Generalized Phrase Structure Grammar, published in 1985, is the main monograph on GPSG. One of the chief goals of GPSG is to show that the syntax of natural languages can be described by context-free grammars, with some suitable conventions intended to make writing such grammars easier for syntacticians. Among these conventions are a sophisticated feature structure system and so-called "meta-rules", which are rules generating the productions of a context-free grammar. GPSG further augments syntactic descriptions with semantic annotations that can be used to compute the compositional meaning of a sentence from its syntactic derivation tree. However, it has been argued that these extensions require parsing algorithms of a higher order of computational complexity than those used for basic CFG. Most of the syntactic innovations of GPSG were subsequently incorporated into head-driven phrase structure grammar [9].

2.5 Finally HPSG

HPSG is a constraint-based, lexicalist approach to grammatical theory that seeks to model human languages as a system of constraints. Typed feature structures play a central role in this modeling. The Head-driven phrase structure grammar (HPSG) is a non-derivational generative grammar theory developed by Carl Pollard and Ivan Sag

(1985). It is the immediate successor to Generalized Phrase Structure Grammar. HPSG draws from other fields such as computer science (data type theory and knowledge representation) and uses the notion of sign (Ferdinand de Saussure). It uses a uniform formalism and is organized in a modular way which makes it attractive for natural language processing.

Few Basics about HPSG:

2.5.1 Feature Structure

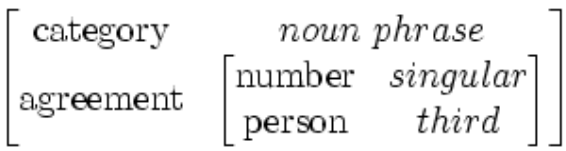
In phrase structure grammars, such as generalized phrase structure grammar, head-driven phrase structure grammar and lexical functional grammar, a feature structure is essentially a set of attribute-value pairs. We can encode these properties by associating what are called Feature Structures with grammatical constituents.

A feature structure can be represented as a directed acyclic graph (DAG), with the nodes corresponding to the variable values and the paths to the variable names. Operations defined on feature structures, e.g. unification, are used extensively in phrase structure grammars. In most theories (e.g. HPSG), operations are, strictly speaking, defined over equations describing feature structures and not over the feature structures themselves, though feature structures are usually used for the sake of informal exposition.

2.5.2 Attribute Value Matrix (AVM)

Often, feature structures are written as shown Figure 3. This particular notation is called attribute value matrix (AVM). An attribute value matrix is a descriptive device employed in HPSG to display the grammatical properties of the class of feature structures meeting that description, consisting of a column of feature name on the left and the value- possibly complex values taking the form of AVMs themselves – on the right [10].

Figure 3 An AVM



Here we have the two features category and agreement. Category has the value noun phrase whereas the value of agreement is indicated by another feature structure with the features number and person being singular and third.

The matrix has two columns - one for the feature names and the other for the values. In this sense a feature structure is a list of key-value pairs. The value might be atomic or another feature structure.

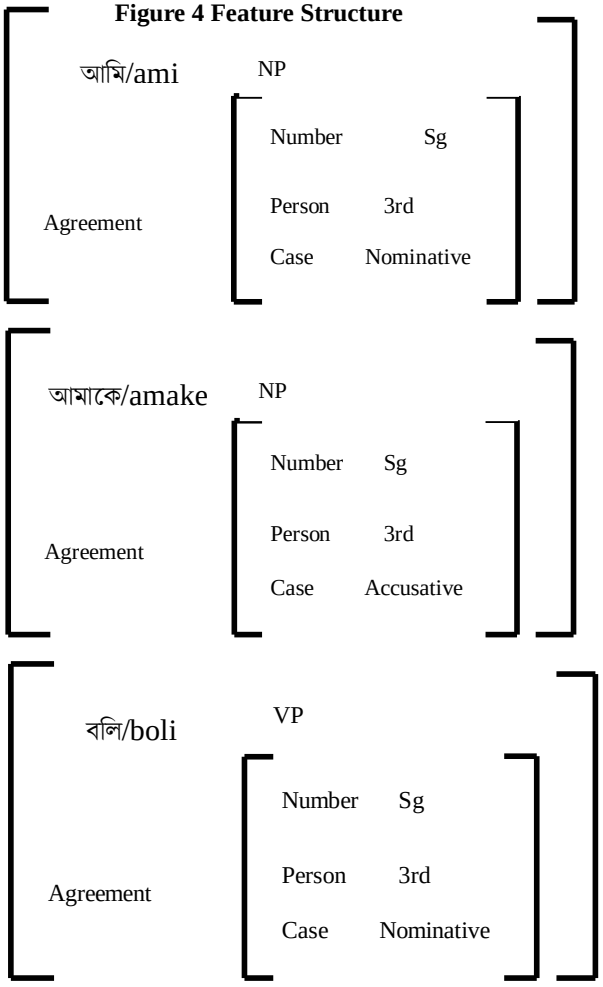
2.5.3 Unification Rule

Unification is an operation that allows us to-

- Check the compatibility of two structures
- Merge the information in two structures

Merging two feature structures produces a new feature structure that is more specific (has more information) than, or is identical to, each of the input feature structures. We say two feature structures can be unified if the component features that make them up are compatible. Structures are compatible if they contain no features that are incompatible. If successful, unification returns the union of all feature/value pairs. For example, let us consider the feature structures of the words “আমি/ami”, “আমাকে/amake”, and “বলি/boli”.

Figure 4 Feature Structure



The following constraints use the "Agreement" feature to make the unification.

S -> NP VP

(NP Agreement) = (VP Agreement)

Now, if we want to unify "আমি/ami" with "বলি/boli" then it unifies, because all the attributes of "আমি/ami" match with the attributes of "বলি/boli". But if we try to unify "আমাকে/amake" with "বলি/boli" then the unification fails because the given constraint does not hold in that the feature "case" is not compatible for "আমাকে/amake" and "বলি/boli".

2.5.4 Head Features

The features for most grammatical categories are copied from one of the children to the parent. The child that provides the features is called the head of the phrase, and the features copied are referred to as head features. For example, the agreement of NP will be copied from its head Noun.

NP -> Noun NP

(NP Agreement) = (Noun Agreement)

Section 3: *Analysis of the Bangla Language*

Primary contribution

by

M. Hammad Ali

Section 3: Analysis of the Bangla Language

Every language has it own grammatical phenomenon that describes the correct usage of the language. We have studied the important grammatical phenomenon of Bangla. Here you will be presented with the phenomenon we came across along with sets of correct and incorrect examples for each one of them.

3.1 Basic Word Order

For a language with subject, verb and object, there can be six possible ways in which these elements can be ordered. These possibilities are: (Subject, Verb, Object), (Subject, Object, Verb), (Verb, Subject, Object), (Verb, Object, Subject), (Object, Subject, Verb) and lastly (Object, Verb, Subject). Bangla is essentially a language with free word order, meaning that every possible ordering carries similar sense, and none of these can be dismissed as being practically impossible. This situation can be slightly controlled if we consider only formal written Bangla, but even then some flexibility still remains.

However, in order to use the LKB platform we would have to decide on one basic word ordering. As such, we chose the order (Subject, Object, Verb), since this is the ordering most commonly used in writing and everyday verbal use of the language. Due to this limiting choice, we had to settle for doing away with some possible re-arrangements of the ordering that are often made in order to provide added emphasis.

Some correct examples using the SOV ordering are:

আমি/ami ভাত/b^{fi}at খাই/k^{fi}ai

তুমি/t̪umi বই/boi পড়/pɔɾo

Some incorrect examples of the other possible word orderings are:

আমি/*ami* খাই/*k^{fi}ai* ভাত/*b^{fi}at* (SVO)

খাই/*k^{fi}ai* আমি/*ami* ভাত/*b^{fi}at* (VSO)

ভাত/*b^{fi}at* আমি/*ami* খাই/*k^{fi}ai* (OSV)

ভাত/*b^{fi}at* খাই/*k^{fi}ai* আমি/*ami* (OVS)

খাই/*kʰai* ভাত/*bʰat* আমি/*ami* (VOS)

It should be mentioned here that some of the “incorrect” example are incorrect only in the sense that they are not considered well-formed by the grammar that we intend to design. In actual Bangla, they might carry a certain amount of information which would be perfectly intelligible, and would be considered meaningful by a native speaker of the language.

3.2 Pronouns

For a formal discussion about pronouns, their types and the roles they play in general, refer to [Section 2]. Pronouns are an integral part of any language. Bangla has two number features: singular and plural, three persons: first, second and third, and three honorifics for each non-first person: pejorative, non-honorific and honorific. As such, the language depends largely on the use of pronouns.

In this part of the lab, we were asked to find out which properties of the subject can cause the pronoun to change form. We report our findings below:

Bangla pronouns do vary by person. This means that there are three different sets of pronouns for each of the three different persons allowed in the language. A listing of the different persons and their corresponding pronouns are given later in the section.

Bangla pronouns also change form depending on the number property of the subject. Bangla has just two number variations: singular and plural. A listing of the number variations is provided later in this section.

Bangla pronouns do not vary with respect to gender. This means that other properties being equal, the pronoun for the masculine gender would be the same as that for the feminine gender. This is unlike the case of Hindi, which has separate pronouns for the different genders.

Next, we consider the distribution that pronouns have in our language. This means whether or not a pronoun can completely replace any noun phrase and still preserve the complete meaning of the sentence. Let us consider some examples, with a sentence first being presented with a full noun phrase and then with a pronoun to take its place.

কেউ/Keu কি/ki আমার/amar খোঁজ/kʰoj করতে/korte এসেছিল/eʃecʰilo?

ক্লাস/klas এর/er একটি/ekti ছেলে/cʰele এসেছিল/eʃecʰilo.

সে/ʃe কিছু/kicʰu বলেছে/bolecʰe?

বলেছে/bolecʰe পরে/pɔre আবার/abar আসবে/aʃbe.

In the example outlined above, in line 2 the noun phrase "ক্লাস/klas এর/er একটি/ekti ছেলে/cʰele" could not be replaced with a pronoun right away without losing information. However, this case depends on the discourse structure and would not be enough to make a conclusive decision. Let us consider another example:

আমাকে/amake তোমার/tomar ভাই/bʰai ফোন/pʰone করেছিল/korecʰilo.

আমাকে/amake সে/ʃe ফোন/pʰone করেছিল/korecʰilo.

Again, both of these sentences would be considered grammatically well-formed and would be deemed acceptable by a native speaker of Bangla, but the problem remains that the latter cannot completely replace the former, unless the order of discourse makes sure that no vagueness or ambiguity in meaning will be introduced in the process. With this is mind, we conclude that pronouns do not have the same distribution as noun phrases, since this decision will depend on the context, something that is as yet beyond the scope of our work.

The person and number distinctions for the language are given below:

Person distinction

First person – আমি/ami, আমাকে/amake, আমাদেরকে/amaḍerke, নিজেকে/niɟeke, নিজেদেরকে/niɟeḍerke

Second person – তুমি/ṭumi, তোমাকে/ṭomake, তোমাদেরকে/ṭomaḍerke, আপনি/apni, আপনাকে/apnake, আপনাদেরকে/apnaḍerke

Third person – সে/ʃ, তিনি/ṭini, তাদেরকে/ṭaḍerke, তাকে/ṭake

Table 2 Number Distinction

Singular	Plural
আমি/ami, তুমি/ṭumi, সে/ʃe, আপনি/apni, তিনি/ṭini	আমরা/amra, তোমরা/ṭomra, তাঁরা/ṭara, ওনারা/onara

3.3 The rest of the Noun Phrase (NP)

This section addresses the obligations that a noun phrase (NP) in Bangla must satisfy. Some examples of such obligations could be rules regarding the use of determiners, such as how nouns must or must not pair up with certain determiners. Determiner-noun issues would include whether determiners are optional or mandatory, which type of nouns can take determiners and which cannot take determiners and the relative order of the determiner with respect to the noun. Another aspect of this part would be to find out whether determiners are mandatory, optional or always missing for each class of noun. In Bangla, determiners as independent words are optional. There are no nouns in Bangla that always require a mandatory determiner. Proper nouns and pronouns never take determiners, whether before or after the noun. Common nouns can take determiners either before or after the noun that it qualifies. In some instances, common nouns may not take determiners, but then the meaning of the sentence is changed. “কুকুর/kukur ডাকছে/dakc^{fi}e” (dogs are barking) and “কুকুর/kukur-টা/ta ডাকছে/dakc^{fi}e”(the dog is barking)

are two different sentences that illustrate this fact. Furthermore, in some cases one of the sentences may not carry any meaning at all, and will not be considered well-formed. For instance, "মানুষ/manuʃ-টা/ta ব্যস্ত/bæs̪to" (the man is busy) makes sense, but "মানুষ/manuʃ ব্যস্ত/bæs̪to" (man is busy) does not seem to convey any useful information. Considering this, we might say that common nouns always need determiners, since omitting the determiner results in changing the meaning of the sentence, which is not something we want. In Bangla determiners are not just independent adverbs. Some determiners can also occur as affixes after the noun it modifies. This class of determiners displays some variation depending on the number property of the noun it qualifies, for instance in the case ছেলে/c^{ɦ}ele-টি/ti (the boy) VS ছেলে/c^{ɦ}ele-গুলো/gulo (the boys). Notice that the noun itself does not change form with the change in number property. It is the determiner that manifests this change in number property, and hence in the meaning of the sentence.

Some correct and incorrect examples of determiner-noun relations are given below:

Correct examples:

হাম্মাদ/hammad̪ বই/boi পড়ছে/poɾc^{ɦ}e.

নাফিদ/nap^{ɦ}id̪ ফোন/p^{ɦ}one-ে/e কথা/kot̪^{ɦ}a বলছে/bolc^{ɦ}e.

Wrong examples:

হাম্মাদ/hammad̪ -গুলো/gulo বাইরে/baire যাবে/ɟabe.

নাফিদ/nap^{ɦ}id̪ -টা/ta ঘুমাচ্ছে/g^{ɦ}umacc^{ɦ}e.

3.4 Agreement

Agreement is co-variation in form between multiple terms (typically a head and a dependent) of a sentence. Either the head or the dependent changes form, depending on the properties of the head. Languages vary greatly in terms of how much agreement they display. There are two ways to categorize agreement systems for any language: Elements that are involved in the agreement relation. The typical cases are subject and verb, object and verb, determiner and noun and lastly, adjective and noun. Features that are involved. Typical cases are person, number, case and gender.

For Bangla, the elements involved in the agreement are subject and verb. This means that usually, the verb has to change form depending on change in form of the subject. In addition, adjectives and nouns in Bangla display a certain degree of agreement in connection to gender. The latter case is mostly because in Bangla, most adjectives have different forms for the masculine and feminine gender. An example is given below:

Masculine: "বুদ্ধিমান/buḍḍʱiman ছেলে/cʱele" – Intelligent boy

Feminine: "বুদ্ধিমতি/buḍḍʱimoti মেয়ে/meje" – Intelligent girl

So if the gender of the noun were to change, the adjective would also have to change form accordingly, in order for the sentence to remain well-formed. However, it should be mentioned here that this form of agreement is not grammatical but based on semantics. Implementing it in the grammar would not require semantic knowledge though, since it could be done through incorporating a person property with the noun and the adjective qualifying the noun. The person property of the noun and the adjective would then take care that the right form of the adjective occurs with the noun.

For Bangla, there is no gender agreement like there is in the case of Hindi. Both masculine and feminine genders use the same form of a verb. There is no number agreement in Bangla either, meaning that the verb does not change form depending on whether the subject of the sentence is singular or plural. There is person agreement, examples of which are given below:

First person – আমি/ami বললাম/bollam

Second person – তুমি/ṭumi বললে/bolle

Third person – সে/ʃe বলল/bollo

3.5 Case

Case is a category that encodes information about the grammatical role played by an element in a sentence. The element can be subject, direct object or any other component of the sentence. An alternative way of denoting grammatical roles is by use of word

ordering, which is something that English does. This is why languages with strong case features can be relatively free ordered, since changes in order will not in anyway affect the roles played by the different constituents of the sentence. Bangla has a moderately strong case system, which is why it is more free order than most other languages. In Bangla, case marking is mostly done with the use of inflection on nouns. For more general discussion about cases, refer to [Ayesha]. The general discussion found in that section is basically what we were asked to determine about our language. We will discuss some more on that later in this section. Some correct and incorrect examples are given below. Two things should be mentioned here: we are assuming a SOV word order, and by convention, incorrect examples are preceded with the asterisk (*) sign.

Noun-nom intransvrb: নাফিদ/napⁿiḍ ঘুমায়/gⁿumaj

*noun-acc intransvrb: নাফিদকে/ napⁿiḍke ঘুমায়/gⁿumaj

Noun-nom noun-acc transvrb: নাফিদ/napⁿiḍ হাম্মাদকে/hammaḍke মারে/mare

*noun-acc transvrb: নাফিদকে/ napⁿiḍke হাম্মাদকে/hammaḍke মারে/mare

*noun-nom noun-nom transvrb: নাফিদ/napⁿiḍ হাম্মাদ/hammaḍ মারে/mare

noun-nom noun-acc2 noun-acc1 ditransvrb: নাফিদ/napⁿiḍ হাম্মাদকে/hammaḍke বই/boi দেয়/ḍej

noun-acc1 noun-nom noun-acc2 ditransvrb: বইটা/boita নাফিদ/napⁿiḍ হাম্মাদকে/ hammaḍke দিল/ḍilo

*noun-acc1 noun-acc2 noun-nom ditransvrb: বইটা/biota হাম্মাদকে/hammaḍke নাফিদ/nafiḍ দিল/ḍilo

noun-acc2 noun-nom noun-acc1 ditransvrb: হাম্মাদকে/hammaḍke নাফিদ/napⁿiḍ বই/boita দিল/ḍilo

*noun-acc2 noun-acc1 noun-nom ditransvrb: হাম্মাদকে/hammaḍke বই/boita নাফিদ/napⁿiḍ দিল/ḍilo

Something that can be inferred from the list of sentences above is that the elements in noun-acc1 and noun-acc2 are freely ordered to some extent.

Table 3 Set of case markers

Case	***Case Marker***
Nominative	+0
Dative-Accusative	+0, কে
Locative	ে, য়, য়ে, তে
Instrumental Locative	ে, তে + post-positives
Ablative	র+ post-positives
Genitive	র

3.6 Negation

Negation, or more formally sentential negation, refers to how the meaning of sentences can be changed from positive to negative. To be more specific, this refers to what change needs to be made to a sentence structure so that the sentence now means the opposite of what it originally did. In most languages, there are two ways of doing this – through the use of inflection on some component of the sentence, usually the verb, or through the use of separate adverbs. In Bangla, both are relevant to some extent. Inflection on verbs is manifested in cases such as "করিনা/korina" to mean "I do not", whereas "করি/kori" would mean "I do". However, we again had to pick one and thus we chose the independent adverb "না/na", which when appended to the end of a sentence, negates its meaning. In future, we might be able to try and incorporate both the use of independent adverbs and affixes that occur with verbs. Below, we provide some examples of how this independent adverb can perform negation on a sentence, as well as some examples of incorrect constructs.

Correct

আমি/ami ভাত/bʱat খাই/kʱai না/na. (I do not eat rice)

Incorrect

আমি/ami ভাত/bʱaṭ না/na খাই/kʱai

আমি/ami না/na ভাত/bʱaṭ খাই/kʱai

না/na আমি/ami ভাত/bʱaṭ খাই/kʱai

3.7 Matrix yes-no questions

As is evident from the title, this part of the work deals with questions that can be answered by a simple affirmative or negative response. The clause matrix simply means that the question is not embedded (more on that later in this section). In this phase, we were to determine how questions are phrased in Bangla. Now, in Bangla, questions are phrased in multiple ways, such as change of intonation and re-ordering of the words in a declarative sentence. For example. "অর্নব/arnab বই/boi পড়ে/pɔre", which is normally a declarative sentence, can be phrased as a question by changing intonation patterns. In verbal communication, this change in intonation would be enough to turn the sentence into a question, without the inclusion of any other words or affixes. However, intonation by itself would not always be enough to pose a question, although in this case it works fine. Since we are mostly working with written language, some rearrangement of the words, or addition of words, would still have to be done. For the purpose of our implementation, we chose to use the independent question adverb "কি/ki", appended to the end of the sentence, to phrase questions. Of course, there are also cases where the adverb "কি/ki" could be placed elsewhere in the sentence but would still pose a valid yes-no question.

For the purpose of this rudimentary grammar, we did not take these cases into consideration. This would not be a problem, since any question that can be formed by placing the question article elsewhere could also be formed by placing it at the end of the sentence. The latter form would probably be considered more "literary" by a native speaker, but no meaning would be lost. Below, we give a sentence, the way it is converted into a question by using the question particle and how the question particle can occur at some position other than the end of the sentence.

অর্নব/arnab বই/boi পড়ে/pɔɾe (Arnab reads books)

অর্নব/arnab বই/boi পড়ে/pɔɾe কি/ki? (Does Arnab read books?)

অর্নব/arnab কি/ki বই/boi পড়ে/pɔɾe? (Does Arnab read books?)

Through the examples shown above, we can see one more thing – the latter way of phrasing a question can lead to ambiguity, since the exact same ordering is also used to express the wh-question "What books does Arnab read?" Thus, through our choice of the first method, we are also avoiding this problem.

3.8 Imperatives

Imperatives are sentences that are meant to specify orders or directions. In English, imperatives are achieved through the use of intonation. Often, subjects in English imperatives can be dropped. In general, imperatives can be indicated through the use of word order variation, use of a special particle or by morphologically marking the verb. Bangla uses this last strategy – morphologically marking the verb. The subject of the imperative is optional. The strategy remains the same in case of negative imperatives.

3.9 Embedded clauses (declarative, negative)

Embedded clauses are sentences that have a clause, and another complete sentence embedded within itself through use of this clause. Bangla embedded declarative clauses are marked through use of the special complementizer "যে/ʄe". Bangla does not mark the clauses morphologically. The word order for matrix and embedded clauses is the same, namely, SOV. For embedded interrogative clauses, we also use a clause-final "kina" in addition to the basic complementizer "যে/ʄe".

Examples

Embedded declarative clauses

আমি/ami জানি/ʄani যে/ʄe তুমি/ʈumi ঢাকা/ɖʱaka যাবে/ʄabe

সে/ʃe বলল/bollo যে/ʄe আজকে/aʄke তার/ʈar সময়/ʃomoj হবে/hɔbe না/na

Embedded interrogative clauses

আমি/ami তাকে/ʈake প্রশ্ন/proʃno করলাম/korlam যে/ɟe সে/ʃe কাজটা/kaɟta করবে/korbe কিনা/kina?

সে/ʃe আমাকে/amake জানাবে/ɟanabe যে/ɟe তিনি/ʈini আসবেন/aʃben কিনা/kina?

Section 4: *Implementation*

Primary contribution

by

Nafid Haque

Section 4: Implementation

After detailed analysis of the grammatical phenomenon of our language, Bangla, we need to implement them to check the effectiveness of our grammar. Linguistic Knowledge Builder (LKB) is a grammar and lexicon development environment developed by DELPH-IN [15], a collaborative effort of several computational linguists. LKB has been developed using LISP and is available for free. Thus we chose LKB to be the environment where we test our grammar.

4.1 More about LKB

Linguistic Knowledge Builder (LKB) is an open source tool to model grammars [11]. Different versions are available for different platforms. We have used the Linux version of LKB for our convenience. The install script given with the Linux version is sufficient to install all the associated packages required with LKB. For the Windows platform, each of those associated packages need to be installed individually and then linked together. Emacs is a well-known editor and is very useful for debugging while working with LKB.

4.2 Type Description Language

TDL is a typed feature-based representation language and inference system, specifically designed to support highly lexicalized grammar theories like HPSG. Type definitions in TDL consist of type and feature constraints over the Boolean connectives. TDL supports open- and closed-world reasoning over types and allows for partitions and incompatible types. Working with partially as well as with fully expanded types is possible. [12]

4.3 Implementation of the Bangla Grammar

To implement a grammar we need four things:

- The type system
- Start structure
- Lexical entries

- Grammar rules

In this section, we offer a more detailed discussion of each of these four constituents and provide relevant examples for each.

4.3.1 The type system

It acts as the defining framework for the grammar. The type system determines the structures that are mutually compatible and allows inheritance of types. [14].

Example:

```
;;;Types
syn-struc := *top* & [CATEG cat].
```

4.3.2 Start structure

It is the starting point and is very similar to the start symbol of a standard Context Free Grammar.

Example:

```
;;;Start
start := phrase & [CATEG s].
```

4.3.3 Lexical entries

These are the information about the words of the language being used in the grammar. Lexical entries define relationship between a string representing the characters in a word and some linguistic description of the word [14].

Example:

```
;;;Lexicon
cat := word & [ORTH “cat”, CATEG n].
```

4.3.4 Grammar rules

These are the typed feature structures that describe how to combine lexical entries and phrases to make further phrases.

Example:

```
;;;Grammar
np_rule := phrase & [CATEG np,
                     ARGS [FIRST [CATEG det],
                           REST [FIRST [CATEG n],
                                 REST *null*]]].
```

All of these must be declared in two different TDL files to be used by LKB. They are the *lexicon.tdl* file and the *grammar.tdl* file. In the *lexicon.tdl* [Appendix B] file, only the lexical entries are stored. All the other rules must be stated in another TDL file and for our case it is the *bangla.tdl* [Appendix A] file.

4.4 Our approach towards implementation in LKB

Dr. Emily Bender conducts a graduate level course in linguistics at the University of Washington [details], where students have to pick one language and develop an HPSG for it as the lab project for the course. In order to get started on our work, we decided to follow the structure of the lab for this course, with Bangla as our language of choice. We went through from Lab 1 to Lab 5, limited initially by our lack of understanding of the more linguistic aspect of the work. Throughout these labs, we had to do a lot of linguistic analysis, find out information about properties of the Bangla language and determine where Bangla stands in terms of how certain linguistic properties are manifested in the language. We have already provided our findings in the previous sections. Here we state the requirements of each of the Lab work and how we implemented that in LKB.

4.4.1 UWashington Linguistics567 Lab 1

This lab was not really very integral to the work of developing a grammar for the chosen language. The main objective of this lab was to get familiar with the LKB environment. We were given a starter grammar for English, with several redundant entries in the lexicon and type files. First we were asked to try parsing a few sentences and check how they are being parsed, if at all. Then we were asked to rewrite these portions in a manner so as to remove this redundancy and use the inheritance property of TDL optimally. After having done this, we were asked to check the same set of sentences again to ensure that we had not lost any coverage in the process of trying to optimize code. This lab was thus intended to let us experiment with LKB and writing TDL code, as well as teaching us some of the basics of how work on a grammar should be done.

4.4.2 UWashington Linguistics567 Lab 2

In this lab, we were given a list of grammatical phenomena that we would be expected to do some background work on as part of the design process. Our task was basically to find out which of these phenomena are relevant for our language and which are not. For the ones that are relevant, we would have to do detailed analysis of how these phenomena are manifested, what properties and quirks they display for Bangla, and how they can be summarized into a small set of general rules which can then be used to generate a grammar that supports these properties. We were provided with a list of thirteen phenomena. All of them have been discussed in Section 3.

4.4.3 UWashington Linguistics567 Lab 3

In this lab, we continued working with the list of phenomena given to us during Lab 2.

4.4.4 UWashington Linguistics567 Lab 4

After having completed the knowledge-gathering phase for the phenomenon mentioned above, we had to fill out a customization form that would give us a lead by generating a starter grammar for our language. This customization page was available along with the specifications given for lab 4. Below, we give a quick review of the choices we made while filling out this form.

Before we move on to what choices we made, however, some points should be emphasized. We would need to input some lexical entries that the starter grammar would be able to work with. As per the lab instructions, these entries would have to be entered in their full form. What this means is that to mean “boy” we would have to input “chele”, and to incorporate “boys” our input would have to be “chele-gulo”, that is, the fully inflected form. This is because the rudimentary grammar does not have any lexical rules that would enable us to provide the root word and the number property and end up with the plural form of the root word. Also, while some of the questions are mandatory and some answer had to be selected (even if only an approximation), certain other sections may not be very relevant or simple for our language and we had the liberty to skip over them if we wanted to do so. That being said, we now highlight our answers to the questions asked in the customization page.

Language name – Bangla

Word order – SOV. Bangla does have determiners as independent words, and the order with respect to nouns is Determiner-noun.

Sentential Negation – this is achieved through use of an adverb. This adverb is an independent modifier of S and appears to the right of it. Since we chose only adverbial negation, the later parts of the section is not relevant to us.

Coordination – Monosyndeton best describes our language. We chose to skip over the later sections, and this would be an area that we would want to come back to and brush up a little.

Matrix yes-no questions – this is done through use of a separate question particle, written “ki” and occurring as a sentence final.

Once these questions were done, we had to enter a few lexicon entries. Some of the lexical types were not relevant for our grammar so we skipped those sections. Finally, we had to input two test sentences using only the lexical entries input in the previous

step. For details about the lexicon entries, refer to Section 4.3.3. After filling out this form we were able to download the starter grammar for Bangla, with the lexical entries we specified and ability to parse the sentences that we entered.

At this point, we were asked to setup a testsuite database that would enable us to automatically test our grammar. After having a lot of difficulty with this, we decided to do without it. The only functionality we lost in the process is that we had to test the sentences manually, but this was not a very big inconvenience. For details about the coverage of the grammar at this stage, refer to Section 4.5.

4.4.5 UWashington Linguistics567 Lab 5

At this stage, we were expected to implement some more "packages" from a list available in the specifications page for this lab. The students for the course got to choose a subset of all the packages mentioned, but we decided to implement the packages one by one and see how far we got.

The first package we started working on was pronouns, person and number distinctions. The instructions in this portion of the lab are very specific, even to the point of giving the exact code that will need to be implemented for the particular case evident in our language. The main reason behind this is that at this stage, we were expected to spend more time on the linguistic analysis and finding out details about the Bangla language. Once we had this information, we would simply have to choose the code that would best serve our needs and that phase of the work would be done.

As part of the first package, we had to find out the relation between nouns and determiners in Bangla. To be more specific, we would need to find out the determiners in Bangla and how they are used along with each of common nouns, proper nouns and pronouns. There are three possibilities as to how determiners and nouns can relate to each other: the determiner can be mandatory, always missing or optional. We report out findings below.

In Bangla, as in all other languages, the proper noun never takes a determiner. Similarly, pronouns do not take a determiner either. Common nouns can optionally take a determiner, but they do not have to take one. Knowing this, we simply had to find the code for this particular case and add it to our grammar. This then enabled us to parse some more sentences, about which more discussion is given in Section 4.5. Before these sentences could be parsed, however, we needed to add in the relevant lexicon entries. For adding in determiner entries, we had to find out more information about determiners in Bangla, and found out that Bangla has determiners in two forms, as shown below:

Independent adverbs – ei, shei, oi
Affixes – ti, ta, gulo

In the examples above, “ei” and “shei” are proximal determiners while “oi” is a distal determiner. Among the affixes, “ti” and “ta” are used in conjunction with nouns in singular form, while “gulo” and “guli” are used with nouns in the plural form. At this point, we were ready to parse a new class of sentences with the changes made in the grammar.

Next, we moved on to the part about implementing the case system for Bangla. Initially, we were supposed to work on case system being implemented through the use of inflection. This is the point where we got stuck again, since this was basically the first time we had to write an AVM starting from scratch. All the types we had created up until this point were basically done through re-using code that was already provided in the starter grammar. We had to take a look at the syntax for the Hindi grammar we got from our supervisor, and through help from it eventually managed to get the case system implementation right. At this point though, the nouns still have to be input in their fully inflected form, so that the root word “ami” in accusative case would have to be added to the lexicon as the word “amake”. The next logical step in our work would be to try and design lexical rules such that given a root word and the desired case property we would be able to generate the inflected form of the word. This is as far we got in lab 5, and thus as far as we got with out work on this Head-Driven Phrase Structure Grammar for Bangla.

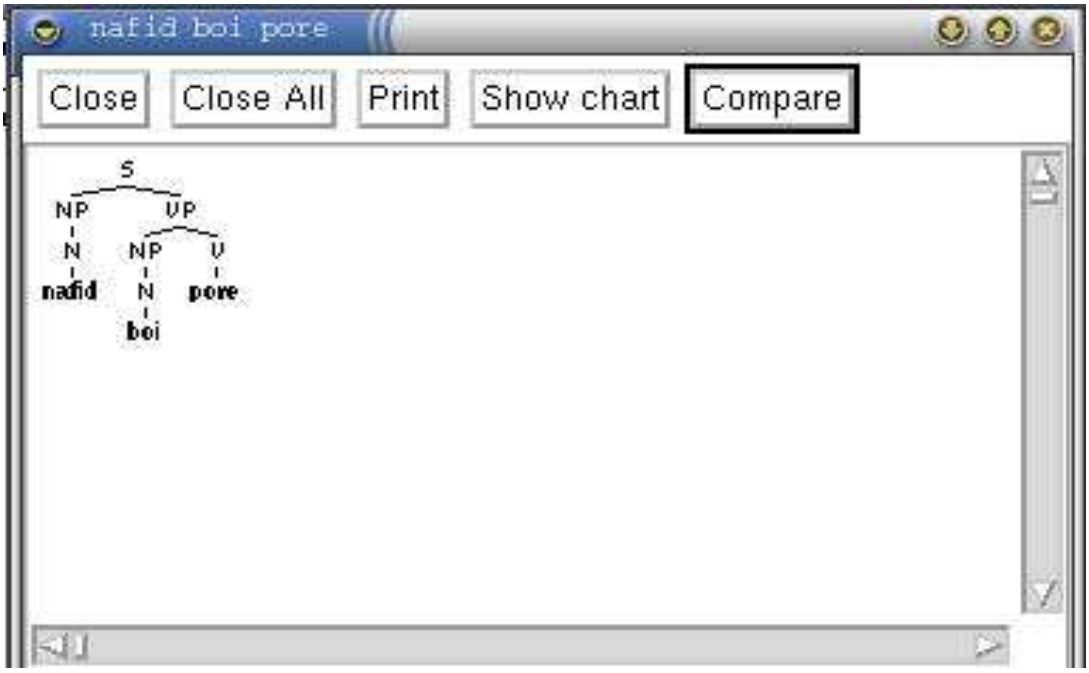
4.5 The Bangla Grammar in LKB

After working with the UWashington Linguistics567 Labs, we got a working grammar that could show some basic structure of the Bangla grammar. In this section some features of LKB is shown with respect to the Bangla grammar we have prepared.

Sentence: nafid boi pore

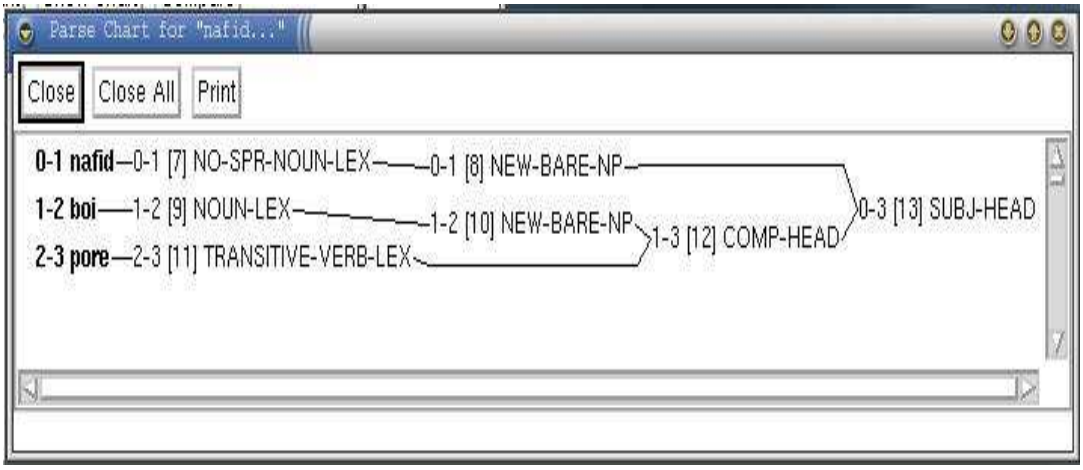
The parse tree generated by the LKB is shown below. On clicking the terminals and non-terminals of the parse tree the entire feature structure of that particular constituent can be viewed.

Figure 5 Parse Tree



A parse chart for the same sentence is shown below. The parse chart shows the raw head types and how they have been inherited.

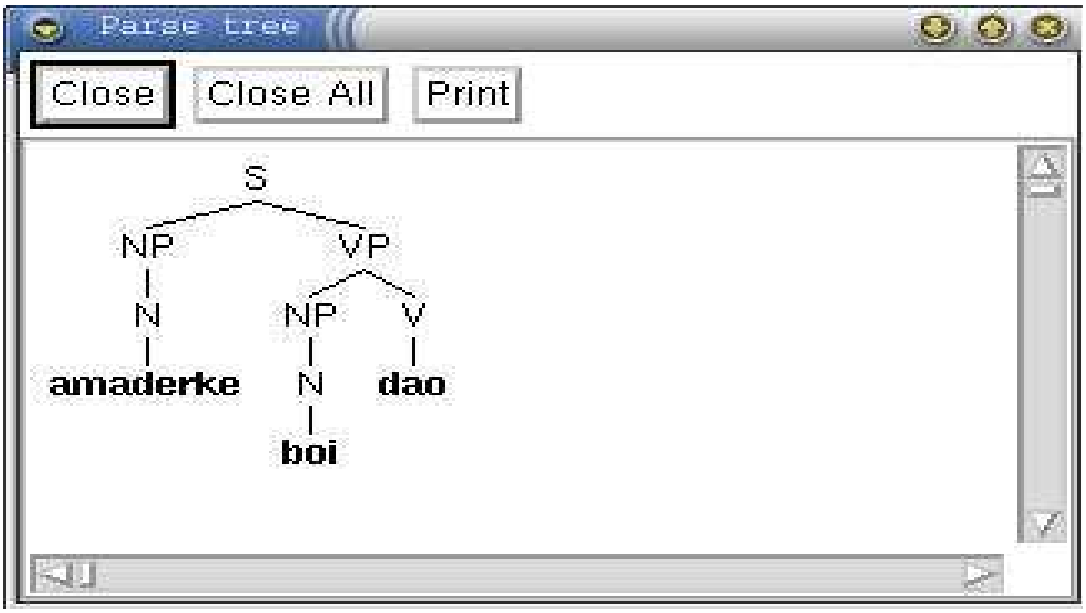
Figure 6 Parse Chart



Sentence: amaderke boi dao

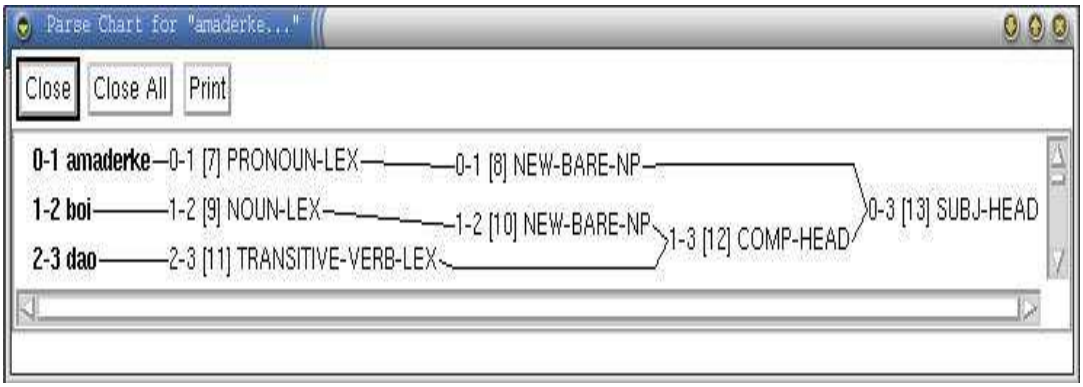
The parse tree for the above sentence is shown below. This sentence highlights the implementation of the accusative case.

Figure 7 Parse Tree



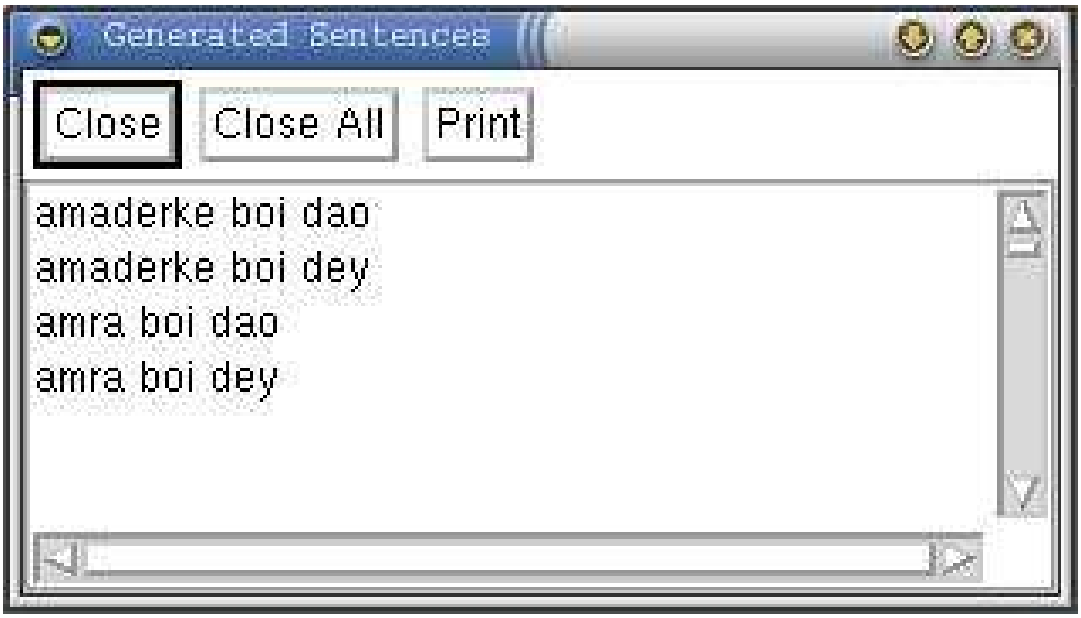
A parse chart for the sentence is shown below.

Figure 8 Parse Chart



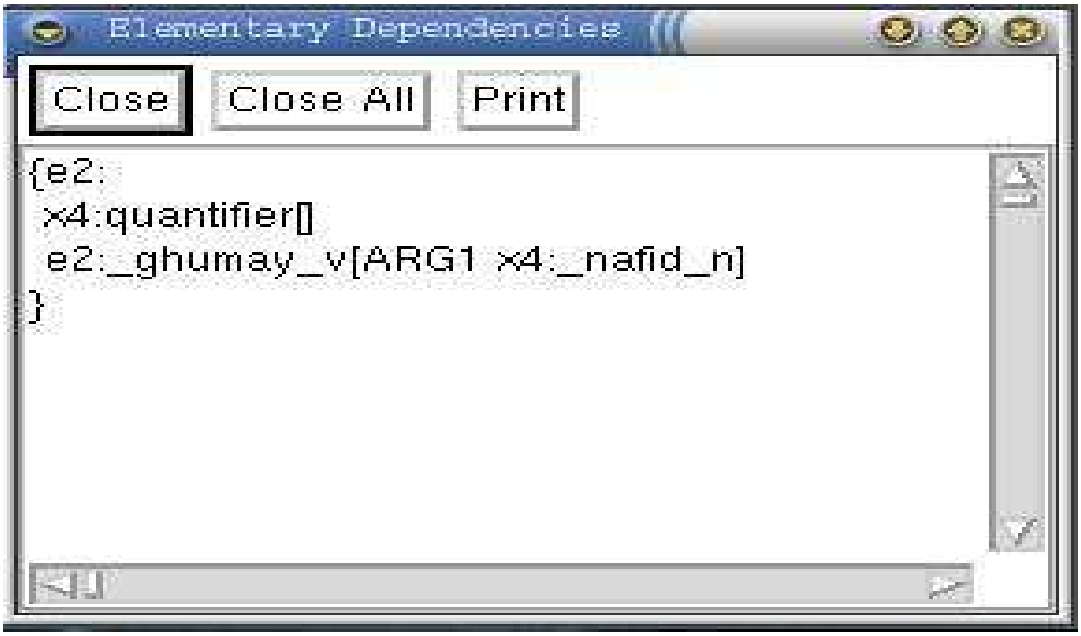
Some sentences generated by the LKB is shown below. It can be seen that among the sentences that have been generated, some are grammatically incorrect. Here "over-generation" took place. This could have been avoided by adding more constraints with the associated head-types.

Figure 9 Generated Sentences



An elementary dependency chart is shown below. This chart highlights the dependencies of the head-types.

Figure 10 Elementary Dependencies



Some feature structures generated by the LKB for the Bangla Grammar is shown below.

Figure 11 TFS 1

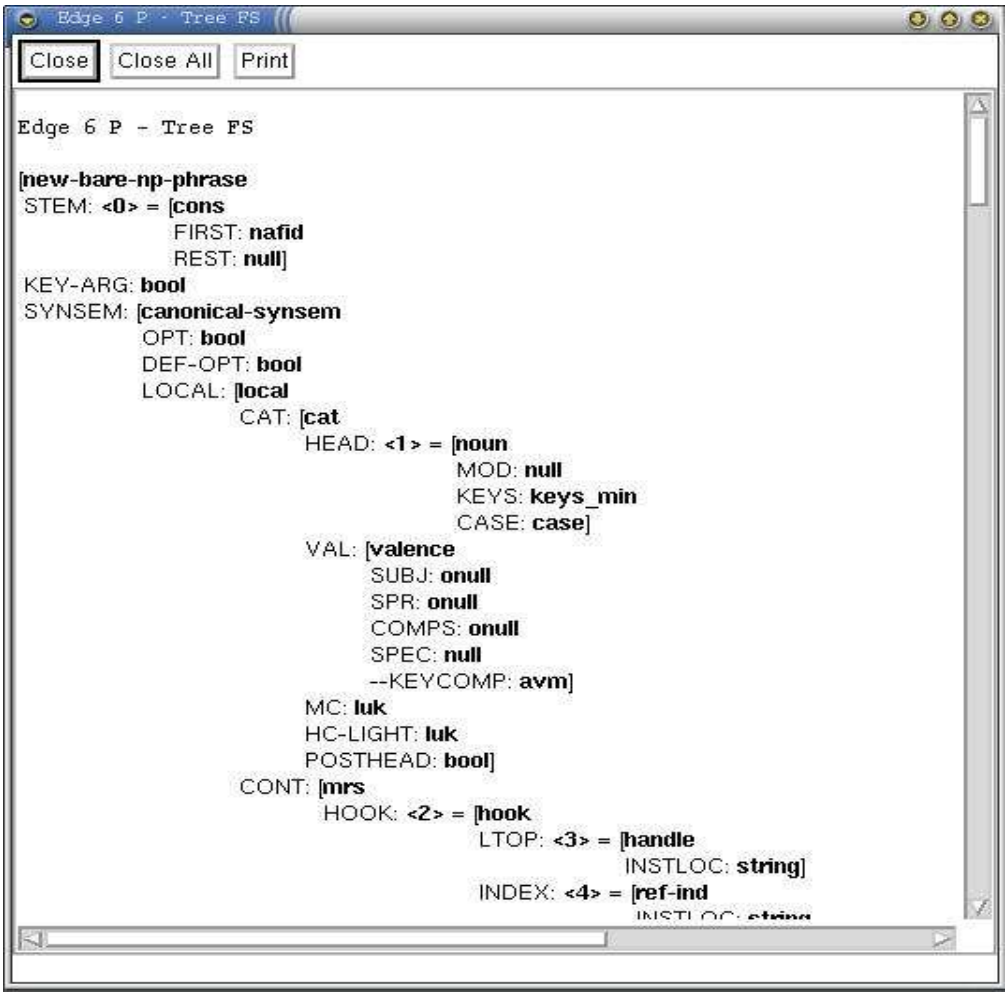


Figure 12 TFS 2

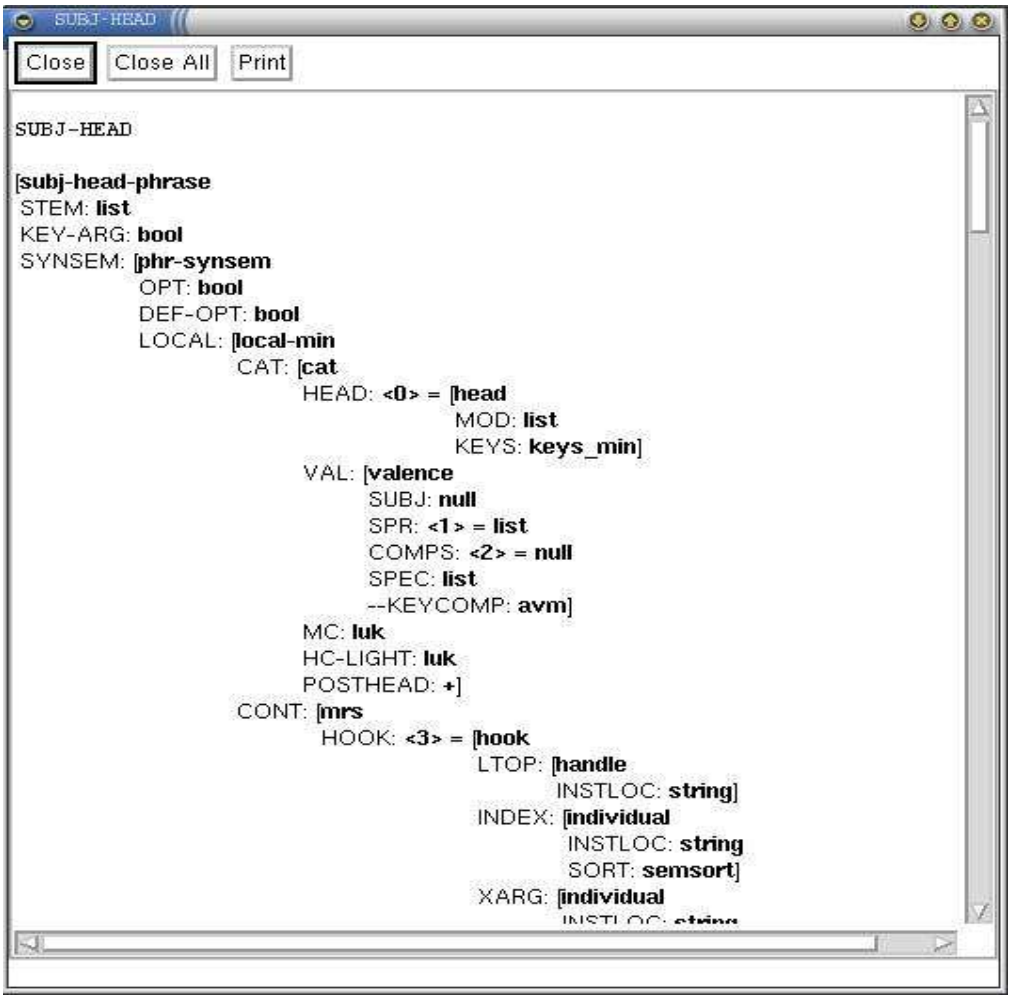
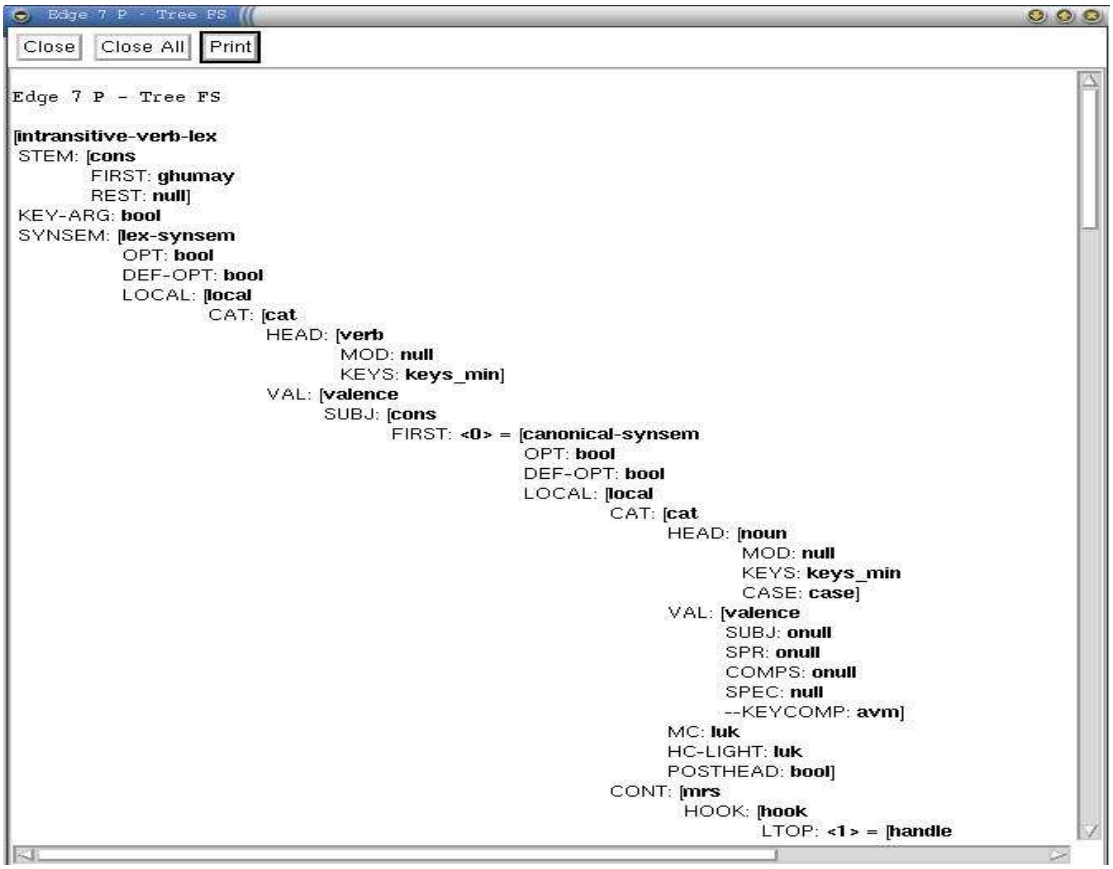


Figure 13 TFS 3



Future Work

In this paper, we discussed the background of the Bangla grammar, the linguistic analysis done by us and a rudimentary implementation of the grammar. In terms of coverage and amount of linguistic knowledge, this grammar is of course at a very basic stage. In the future, we would like to increase the coverage of this grammar through experimentation and through background analysis of some more linguistic phenomena. We would also like to try a more generic approach towards incorporating more sentences in our list of coverage. At present, we start with some sentence in mind and then add in the features that would be necessary to be able to parse this sentence. In future, we would like to be able to start from a more abstract level and then arrive at a whole class of sentences instead of just one or two sentences. Lastly, we would also like to try and add semantic knowledge to go with our grammar and also maybe use a corpus in order to implement a probabilistic parser.

Conclusion

In this paper, we gave a considerable amount of background information of the Bangla language and its different linguistic components. We also discussed how CFG has been attempted to capture the Bangla language and the reasons why it fails to perform adequately for Bangla or for that matter any other natural language rich in agreement issues. We provided some background information of the different entities of Bangla language and that of natural languages in general, and discussed the knowledge base behind our decision to use a head-driven grammar for our purpose. Then we discussed the basics of HPSG, and presented an analysis of a list of different phenomena for Bangla. Lastly, we presented a sample implementation of the grammar, built upon an auto-generated grammar and with certain other features added in the course of our work. The grammar theory and implementation is at a basic stage, but it is designed in a way so that it should be easy to build upon this grammar and capture a host of other linguistic components. Another important product of our work would be this documentation itself, highlighting the basics of Bangla language and a summary of a good number of linguistic issues and how they are manifested. With this basic implementation to work with in the future and a good documentation that can serve as a good starter resource, we trust that our work will be able to make a positive difference in the efforts of anyone willing to continue work on the design of a computational grammar for Bangla.

References:

1. N. Chomsky and M. Halle, The Sound Pattern of English.
2. Grady *et Al*, Contemporary Linguistics.
3. Daniel Jurafsky and James H. Martin, Chap-9, Speech and Language Processing
4. Bangla Bhashar Bekoron
5. Online notes collected from: http://www.arts.uottawa.ca/writcent/hypergrammar
6. Lecture Note CSC173, available online at http://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/parsetrees.html
7. Lecture Note CSC173, available online at http://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html
8. Lecture Note Linguistics 120, Autumn 2003, collected from the Internet
9. Wikipedia entry of GPSG, available online at http://en.wikipedia.org/wiki/GPSG
10. Robert D. Levine, Encyclopedia of Cognitive Science – Head Driven Phrase Structure Grammar.
11. Online notes collected from: http://en.wikipedia.org/wiki/LKB
12. Online notes collected from: http://www.ims.uni-stuttgart.de/ftp/pub/Users/jochen/Esslli95/tdl_flyer.html
13. Naushad UzZaman, Undergraduate Thesis, Spring 2005. Phonetic Encoding for Bangla and its application to spelling checker, transliteration, cross language information retrieval and name searching.
14. Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars*. CSLI Publications, Stanford.
15. DELPH-IN website.

Appendices

Appendix A

The bangla.tdl file

1. bangla.tdl file

```
;;; -*- Mode: TDL; Package: LKB -*-
;;;
;;; Language-specific types and constraints for Bangla

;;; Type addenda adding constraints to head types

;;; This grammar includes head-modifier rules. To keep
;;; out extraneous parses, constrain the value of MOD on
;;; various subtypes of head. This may need to be loosened later.
;;; This constraint says that only adverbs, adjectives,
;;; and adpositions can be modifiers.

+nvcdmo :+ [ MOD < > ].

;;; Types for values of additional features.

;;; Phrase structure rule types

;;; Types for SOV word order.

comp-head-phrase := basic-head-1st-comp-phrase & head-final.
subj-head-phrase := basic-head-subj-phrase & head-final &
[ HEAD-DTR.SYNSEM.LOCAL.CAT.VAL.COMPS < > ].



;;; Rules for building NPs. Note that the Matrix uses SPR for
;;; the specifier of nouns and SUBJ for the subject (specifier) of
verbs.

spec-head-phrase := basic-head-spec-phrase & head-final.

;;; Bare NP phrase. Consider modifying the PRED value of the quantifier
relation
;;; introduced to match the semantic effect of bare NPs in your
language.

;;;bare-np-phrase := basic-bare-np-phrase &
;;;[ C-CONT.RELS <! [ PRED "unspec_q_rel" ] !> ].
```

```
new-bare-np-phrase := basic-bare-np-phrase &
[ HEAD-DTR.SYNSEM.LOCAL.CAT.VAL.SPR
< [ LOCAL.CONT.RELS < ! [ PRED #pred ] ! > ] >,
C-CONT.RELS < ! [ PRED #pred & quantifier_rel ] ! > ].
;;; Lexical types

;;; Nouns

noun-lex := basic-noun-lex & basic-one-arg &
[ SYNSEM.LOCAL [ CAT.VAL [ SPR < #spr & [ LOCAL.CAT.HEAD det ] >,
COMPS < >,
SUBJ < >,
SPEC < > ]],
ARG-ST < #spr > ].

;;;just added to test
pronoun-lex := noun-lex &
[ SYNSEM [ LOCAL.CAT.VAL.SPR
< [ LOCAL.CONT.RELS < ! [PRED pronoun_q_rel] ! > ] >,
LKEYS.KEYREL.PRED 'pronoun_n_rel ] ].

common-noun-lex := noun-lex &
[ SYNSEM.LOCAL [ CAT.VAL.SPR
< [ LOCAL.CONT.RELS < ! [PRED reg_quant_rel] ! > ] >,
CONT.HOOK.INDEX.PNG [ PER third ] ] ].

proper-noun-lex := noun-lex &
[ SYNSEM.LOCAL [ CAT.VAL.SPR
< [ LOCAL.CONT.RELS < ! [PRED proper_q_rel] ! > ] >,
CONT.HOOK.INDEX.PNG [ PER third ] ] ].

no-spr-noun-lex := noun-lex &
[ SYNSEM.LOCAL.CAT.VAL.SPR < [ OPT + ] > ].

;;; Verbs

verb-lex := basic-verb-lex &
[ SYNSEM.LOCAL [ CAT [ VAL [ SPR < >,
SPEC < >,
SUBJ < #subj > ]],
CONT.HOOK.XARG #xarg ],
```

```
ARG-ST < #subj &
[ LOCAL [ CAT [ HEAD noun,
VAL [ SPR < >,
COMPS < > ]],
CONT.HOOK.INDEX #xarg ]], ... > ].

intransitive-verb-lex := verb-lex & intransitive-lex-item &
[ SYNSEM.LOCAL.CAT.VAL.COMPS < > ].

transitive-verb-lex := verb-lex & transitive-lex-item &
[ SYNSEM.LOCAL.CAT.VAL.COMPS < #comps >,
ARG-ST < [ ],
#comps &
[ LOCAL.CAT [ VAL [ SPR < >,
COMPS < > ],
HEAD noun ]] > ].

;;; Case-marking adpositions
;;; Case marking adpositions are constrained not to
;;; be modifiers.

case-marker-p-lex := basic-one-arg & raise-sem-lex-item &
[ SYNSEM.LOCAL.CAT [ HEAD adp & [ MOD < > ],
VAL [ SPR < >,
SUBJ < >,
COMPS < #comps >,
SPEC < > ]],
ARG-ST < #comps & [ LOCAL.CAT [ HEAD noun,
VAL.SPR < > ]] > ].

;;; Determiners
;;; SPEC is non-empty, and already specified by basic-determiner-lex.

determiner-lex := basic-determiner-lex & basic-zero-arg &
[ SYNSEM.LOCAL.CAT.VAL [ SPR < >,
COMPS < >,
SUBJ < > ]].

;;; Adverbs

;;; Negative adverb
```

```
neg-adv-lex := basic-scopal-adverb-lex &
[ SYNSEM.LOCAL.CAT [ POSTHEAD +,
VAL [ SPR < >,
COMPS < >,
SUBJ < > ],
HEAD.MOD < [ LOCAL.CAT [ HEAD verb,
VAL [ SUBJ cons,
COMPS null ]]] > ]].


qpart-le := basic-scopal-adverb-lex &
[ SYNSEM [ LOCAL.CAT [ HEAD.MOD < [ LOCAL.CAT [ HEAD verb,
VAL [ SUBJ < >,
COMPS < > ]]]>,
VAL [ SUBJ < >,
SPR < >,
COMPS < > ],
POSTHEAD + ],
LKEYS.KEYREL.PRED question_m_rel ]].


;;; Coordination


n1-top-coord-rule := basic-n-top-coord-rule & monopoly-top-coord-rule &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
n1-mid-coord-rule := basic-n-mid-coord-rule & monopoly-mid-coord-rule &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
n1-bottom-coord-rule := conj-last-bottom-coord-rule & n-bottom-coord-
phrase &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].


np1-top-coord-rule := basic-np-top-coord-rule & monopoly-top-coord-rule
&
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
np1-mid-coord-rule := basic-np-mid-coord-rule & monopoly-mid-coord-rule
&
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
np1-bottom-coord-rule := conj-last-bottom-coord-rule & np-bottom-coord-
phrase &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].


vp1-top-coord-rule := basic-vp-top-coord-rule & monopoly-top-coord-rule
&
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
vp1-mid-coord-rule := basic-vp-mid-coord-rule & monopoly-mid-coord-rule
&
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
```

```
vp1-bottom-coord-rule := conj-last-bottom-coord-rule & vp-bottom-coord-
phrase &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].


s1-top-coord-rule := basic-s-top-coord-rule & monopoly-top-coord-rule &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
s1-mid-coord-rule := basic-s-mid-coord-rule & monopoly-mid-coord-rule &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].
s1-bottom-coord-rule := conj-last-bottom-coord-rule & s-bottom-coord-
phrase &
[ SYNSEM.LOCAL.COORD-STRAT "1" ].


;;; This part added on 16th September.
;;; Lab 5 Number Distinction


png :+ [ PER person,
NUM number ].


person := *top*.
first := person.
second := person.
third := person.


number := *top*.
sg := number.
non-sg := number. ; use this one if your language only has sg-pl
dual := non-sg. ; at these two if your language has sg-du-pl
pl := non-sg.


;;; This part added on 8th October.
;;; Lab 5 Pronouns cover-det-rule


quantifier_rel := predsort.
pronoun_q_rel := quantifier_rel.
proper_q_rel := quantifier_rel.
reg_quant_rel := quantifier_rel.


;;; Lab 5 Determiners


demonstrative_q_rel := reg_quant_rel.
non+demonstrative_q_rel := reg_quant_rel.
proximal+dem_q_rel := demonstrative_q_rel. ; close to speaker
distal+dem_q_rel := demonstrative_q_rel. ; away from speaker
```

```
remote+dem_q_rel := distal+dem_q_rel. ; away from speaker and hearer
hearer+dem_q_rel := distal+dem_q_rel. ; near hearer
def_q_rel := non+demonstrative_q_rel. ; definite
indef_q_rel := non+demonstrative_q_rel. ; indefinite



;;; Lab 5 Case Inflection (17th November)
noun :+ [ CASE case].

case := *top*.

nom := case.
acc := case.

trans-verb-lex := basic-verb-lex & transitive-lex-item &
[ SYNSEM.LOCAL [ CAT [ HEAD verb,
VAL [ SPR < >,
SUBJ < #subj & synsem
& [ LOCAL.CAT [ HEAD noun &
[ CASE nom ],
VAL.SPR <> ]] >,
COMPS < #comps
& [ LOCAL.CAT [ HEAD noun &
[ CASE acc ],
VAL.SPR <> ]]>,
SPEC < > ]]],
ARG-S < #subj, #comps > ].



;;; Lab 5 Case Inflection (2nd December)

;;;adp :+ [ CASE case ].
+np :+ [ CASE case ].
```

Appendix B

The lexicon.tdl file

2. lexicon.tdl file

```
;;; -*- Mode: TDL; Package: LKB -*-

;;; Nouns

nafid := no-spr-noun-lex &
[ STEM < "nafid" >,
SYNSEM.LKEYS.KEYREL.PRED "_nafid_n_rel" ].

boi := noun-lex &
[ STEM < "boi" >,
SYNSEM.LKEYS.KEYREL.PRED "_boi_n_rel" ].

gyan := noun-lex &
[ STEM < "gyan" >,
SYNSEM.LKEYS.KEYREL.PRED "_gyan_n_rel" ].



;;;just adding to test 12th October
hammad := no-spr-noun-lex &
[ STEM < "hammad" >,
SYNSEM.LKEYS.KEYREL.PRED "_hammad_n_rel" ].

paper := noun-lex &
[ STEM < "paper" >,
SYNSEM.LKEYS.KEYREL.PRED "_paper_n_rel" ].



;;; Verbs

ghumay := intransitive-verb-lex &
[ STEM < "ghumay" >,
SYNSEM.LKEYS.KEYREL.PRED "_ghumay_v_rel" ].

pore := transitive-verb-lex &
[ STEM < "pore" >,
SYNSEM.LKEYS.KEYREL.PRED "_pore_v_rel" ].

orjon := transitive-verb-lex &
[ STEM < "orjon" >,
SYNSEM.LKEYS.KEYREL.PRED "_orjon_v_rel" ].
```

```
kore := transitive-verb-lex &
[ STEM < "kore" >,
SYNSEM.LKEYS.KEYREL.PRED "_kore_v_rel" ].



;;;added on 25th november
dey := transitive-verb-lex &
[ STEM < "dey" >,
SYNSEM.LKEYS.KEYREL.PRED "_dey_v_rel" ].

dao := transitive-verb-lex &
[ STEM < "dao" >,
SYNSEM.LKEYS.KEYREL.PRED "_dey_v_rel" ].

;;;added to test 12th oct
pori := transitive-verb-lex &
[ STEM < "pori" >,
SYNSEM.LKEYS.KEYREL.PRED "_pori_v_rel" ].

poro := transitive-verb-lex &
[ STEM < "poro" >,
SYNSEM.LKEYS.KEYREL.PRED "_poro_v_rel" ].

;;; Other

;;; Case-marking adpositions

subj-marker := case-marker-p-lex &
[ STEM < "Ayesha" > ].

;;; Determiners

ei := determiner-lex &
[ STEM < "ei" >,
SYNSEM.LKEYS.KEYREL.PRED proximal+dem_q_rel ].

shei := determiner-lex &
[ STEM < "shei" >,
SYNSEM.LKEYS.KEYREL.PRED proximal+dem_q_rel ].
```

```
oi := determiner-lex &
[ STEM < "oi" >,
SYNSEM.LKEYS.KEYREL.PRED distal+dem_q_rel ].

;;;era := determiner-lex &
;;;[ STEM < "era" >,
;;;SYNSEM.LKEYS.KEYREL.PRED "_ei_q_rel" ].

na := neg-adv-lex &
[ STEM < "na" >,
SYNSEM.LKEYS.KEYREL.PRED "_neg_r_rel" ].

ki := qpart-le &
[ STEM < "ki" > ].

;;;ke := qpart-le &
;;;[ STEM < "ke" > ].

ebong_1 := conj-lex &
[ STEM < "ebong" >,
SYNSEM.LKEYS.KEYREL.PRED "_and_coord_rel",
CFORM "1" ].

;;; Pronouns added on 8th october2006

amra := pronoun-lex &
[ STEM <"amra">,
SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG [ PER first,
NUM non-sg] ].

;;;Case Inflection added on 11th November
amra := pronoun-lex &
[ STEM < "amra" >,
SYNSEM.LOCAL [ CAT.HEAD.CASE nom,
CONT.HOOK.INDEX.PNG [ PER first,
NUM non-sg ] ] ].

amaderke := pronoun-lex &
[ STEM < "amaderke" >,
SYNSEM.LOCAL [ CAT.HEAD.CASE acc,
CONT.HOOK.INDEX.PNG [ PER first,
```

```
NUM non-sg ] ] ].
;;;added to test case inflection on 25th november
ke := noun-lex &
[ STEM < "ke" >,
SYNSEM.LOCAL [ CAT.HEAD.CASE acc,
CONT.HOOK.INDEX.PNG [ PER first,
NUM non-sg ] ] ].

;;;adposition
hote := case-marker-adp &
[ STEM < "hote" >,
SYNSEM.LOCAL.CAT.HEAD.CASE []].
```