Parsing Bangla using LFG: An Introduction

Muhammad Nasimul Haque and Mumit Khan
BRAC University,Dhaka, Bangladesh
mnasimh@yahoo.com, mumit@bracuniversity.ac.bd

Abstract

This paper is introduces LFG (Lexical Functional Grammar) formalism for parsing Bangla. The LFG formalism, which has evolved from extensive computational, linguistic, and psycholinguistic research, provides a simple set of devices for describing the common properties of all natural languages and the particular properties of individual languages. This paper tabulates a set of instructions for using the formulation of LFG rules to parse Bangla. With the information contained in this paper, linguists previously unfamiliar with this very expressive formalism of this theory should find it possible to interpret and to compose the kind of syntax rules and lexical items employed in LFG. In this paper we present successful parses of some simple Bangla sentences along with some unsuccessful parses of ungrammatical sentences.

1. Introduction

The Computer understanding of language has many practical applications, for example, easy to use interfaces, machine translation and intelligent question-answering systems. Parsing is a fundamental problem in language processing for both machines and humans. Most natural language applications, such as Information Extraction, Machine Translation, and Speech Recognition, would almost certainly benefit from high-accuracy parsing. From a scientific standpoint, there is the question of how people interpret language: what knowledge is used, and exactly how this knowledge is applied in practice. In its simplest form, the parsing problem involves the definition of an algorithm that maps any input sentence to its associated syntactic tree structure.

It is well established that parsing natural language text is much more difficult than strictly defined computer languages. One reason is that grammars for natural languages are often complex, ambiguous, and specified by collections of examples rather than complete formal rules. Another difficulty is that punctuation is used much more sparingly. For example, many sentences in Bangla consist of a sequence of words in which the only punctuation is the terminating period. Parsing is a process of transforming natural language into an internal system representation, which can be trees, dependency graphs, frames or some other structural representation. Syntactic-only parsing attempts to convert the natural language strings into either tree structures or dependency links representing the syntactic structure of the utterance. The syntactic structures can later be sent for a semantic interpreter for further processing. The most common syntactic parsers today are probabilistic context free grammar parsers, which combine a context free grammar with a probability model which determines the most likely parse out of a large number of syntactic trees consistent with a given utterance (see for example [1, 2].

The steps of the understanding process are parsing and semantic interpretation, and the formal knowledge representation suitable for computer processing. A core component necessary for parsing and semantic interpretation is the system lexicon. This is a data store that lists all words known to the system, and encodes their syntactic properties and the correspondences between words in the language and concepts in the computer knowledge representation. In this paper we will not be exploring the structures that efficiently represent the information needed for interpretation in the system lexicon, and the way parsing speed and semantic disambiguation accuracy can be improved with the use of semantic feature vectors and efficient integration of domain independent and domain specific information in the lexicon.

We are interested in syntactic parsing as the syntactic relationships in a sentence correspond to functional relationships in the underlying meaning representation. For example, in a sentence "*aamra bhaat kheyechhilaam*", '*bhaat*' is the object of *'kheyechhilaam',* which in the underlying meaning representation corresponds to the fact that *'bhaat'* is an argument (sometimes called THEME or PATIENT) of a '*khaowa*' action. This relationship has long been studied in linguistics, and it is well known that often there are many possible syntactic structures consistent with the same string. The correct syntactic parse is (informally) defined as the one that humans see as corresponding to the correct semantic interpretation of the utterance. It is the job of the semantic theory to select the correct parse and the corresponding interpretation from the set of all parses consistent with a sentence. This paper is concerned with the lexical information needed to solve ambiguity problems during parsing and semantic interpretation.

2. Lexical Functional Grammar (LFG)

The term Lexical Functional Grammar (LFG) was first introduced in print in the 1982 by Kaplan and Bresnan. Since then the formalism of LFG has been applied in the description of a wide range of linguistic phenomena. The basic features of the formalism are quite simple: the theory assigns two levels of syntactic representation to a structure, the constituent structure and functional structure [3]. The c-structure is a phrase-structure tree that serves as the basis for phonological interpretation while the f-structure is a hierarchical attribute-value matrix that represents underlying grammatical relations. The c-structure is assigned by the rules of a context-free phrase structure grammar. Functional annotations on those rules are instantiated to provide a formal description of the f-structure, and the smallest structure satisfying those constraints is the grammatically appropriate f-structure.

A very striking aspect of LFG is its stability as a framework. The fundamental architecture of the theory has remained constant since the late 1970s. A very important facet of LFG syntax, which signals it out from many other syntactic theories, is the representation of different dimensions of the syntax (c-structure and f-structure, or external and internal syntax) by means of different formal entities: the architecture combines a context free grammar formalism (for c-structure) with attribute value structure (for f-structure) [4].

LFG is a monotonic theory of syntax; instead of postulating different derivational levels represented in the same formal language, it incorporates different parallel levels of information, which can all potentially access each other, each with its own formal language. The assumption about parallel levels of information extends even to non-syntactic aspects of grammar. Thus, for example, semantic information is assumed to be available to various levels of syntax, and syntactic levels can input into phonology [5].

3. Parsing Bangla

We now focus on writing a simple LFG for parsing simple sentences in Bangla. We use here a simple sentence with an object to the verb. Let the sentence be

(S1) *'aamra bhaat kheyechhilaam'*

Here we see that *'kheyechhilaam'* is the THEME with a PATIENT *'bhaat'*. To parse this sentence using CFG (Context Free Grammar) we need the following rules:

(R1)	S →	NP	VP
(R2)	NP →	Pro	
(R3)	NP →	N	
(R4)	VP →	NP	VP
(R5)	VP →	V	
(R6)	Pro →	*aamra*	
(R7)	N →	*bhaat*	
(R8)	V →	*kheyechhilaam*	

These rules made the parsing of the sentence straightforward. The tree view of the solution is as in Figure 1.

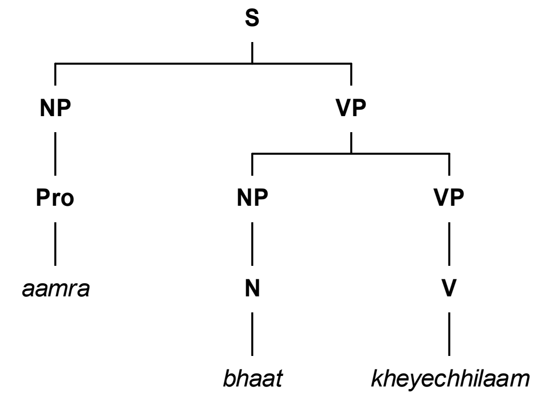

Figure 1: Tree of the CFG parse of (S1)

Now, let us add two more pronouns *'aamader'* and *'tomraa'* in the list.

(R9)	Pro	→	*aamader*
(R10)	Pro	→	*tomraa*

Then we get successful parses (along with tree view, Figure 2 and Figure 3) for grammatically incorrect sentences like

(S2) **'aamader bhaat kheyechilaam'*

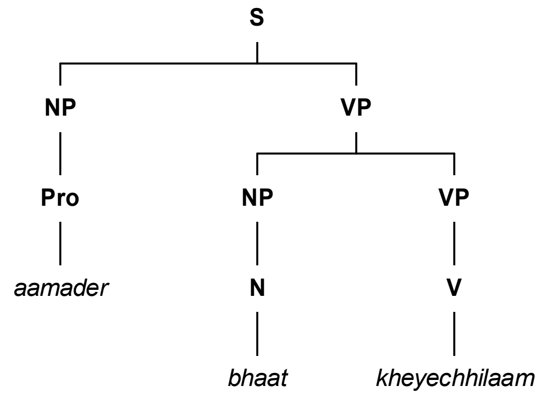

Figure 2: Tree of the CFG parse of (S2)

(S3) **'tomraa bhaat kheyechilaam'*

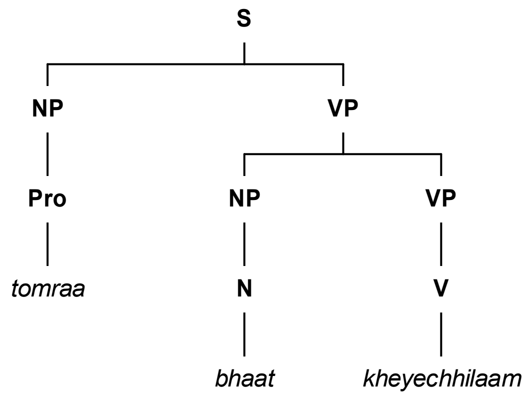

Figure 3: Tree of the CFG parse of (S3)

To avoid this kind of inefficiency the LFG adds another level to CFG that is known as c-structure. The rules of a Lexical Functional Grammar contain expressions known as FUNCTIONAL SCHEMATA, which are associated with the symbols that appear on the right hand side of the arrow →. Figure 4 shows the usual format for writing rules in LFG.

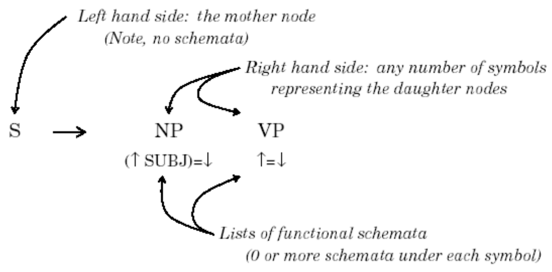
Left hand side: the mother node
(Note, no schemata)
Right hand side: any number of symbols representing the daughter nodes
S → NP VP
(↑ SUBJ)=↓ ↑=↓
Lists of functional schemata
(0 or more schemata under each symbol)

Figure 4: Format of LFG rules

LFG uses two meta-variables ↑ and ↓ arrows in its functional schemata. The symbol ↑, known as the EGO or SELF meta-variable, abbreviates the composition of the structural correspondence with the mother function, and ↓, known as the MOTHER meta-variable, stands for the f-structure corresponding to the matching node. Thus the annotation on the NP, i.e., (↑ SUBJ) = ↓, can be read as "the subject of the f-structure of the matching NP node's mother is the matching node's f-structure". Following are the rewritten rules of (R1) to (R4) in LFG:

(R11)	S →	NP	VP
		(↑ SUBJ) = ↓	↑ = ↓
(R12)	NP →	N	
		↑ = ↓	
(R13)	VP →	NP	VP
		(↑ OBJ) = ↓	↑ = ↓
(R14)	VP →	V	
		↑ = ↓	

along with the lexical entries

(R15)	N →	*aamra*
		(↑ PRED) = *'pro'*
		(↑ PERS) = 1
		(↑ NUM) = PL
		(↑ CASE) = NULL
		(↑ ANIM) = '+'
(R16)	N →	*aamader*
		(↑ PRED) = *'pro'*
		(↑ PERS) = 1
		(↑ NUM) = PL
		(↑ CASE) = GEN
		(↑ ANIM) = '+'
(R17)	N →	*tomraa*
		(↑ PRED) = *'pro'*
		(↑ PERS) = 2
		(↑ NUM) = PL
		(↑ CASE) = NULL
		(↑ ANIM) = '+'
(R18)	N →	*bhaat*
		(↑ PRED) = *'rice'*
		(↑ PERS) = 3
		(↑ CASE) = NULL
		(↑ ANIM) = '-'
(R19)	V →	*kheyechhilaam*
		(↑ PRED) = *'eat*<(↑ SUBJ), (↑ OBJ)>*'*
		(↑ TENSE) = PAST
		(↑ SUBJ PERS) = 1
		(↑ SUBJ CASE) = NULL
		(↑ SUBJ ANIM) = '+'

The annotated tree of the sentence (S1) is shown in Figures 5 and 6.

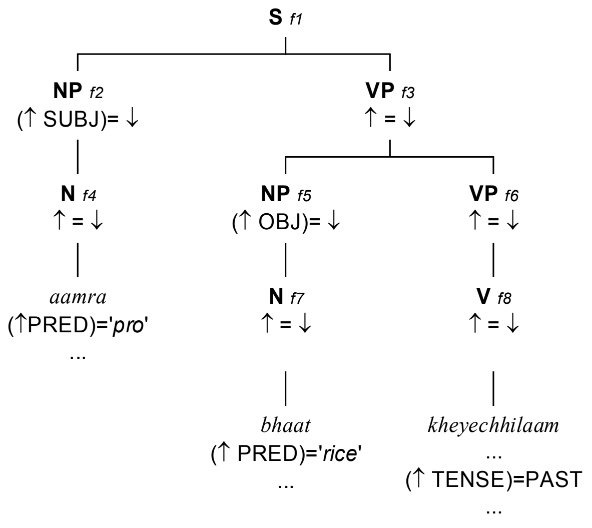

Figure 5: Initial annotated tree for (S1) in LFG

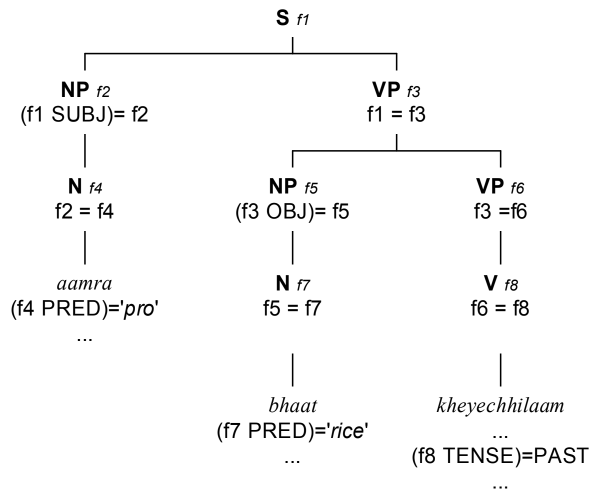

Figure 6: Completed annotated tree for (S1)

After completing the annotated tree the unification process begins, i.e., the f-structure formation starts. A simple unification of the annotated tree in Figure 6 at the functional *f*3 is shown in Figure 7.

$$
f3\left[\begin{array}{ll} OBJ & \begin{array}{l} f5 \\ f7 \end{array}\left[\begin{array}{ll} PRED & 'pro' \\ PERS & 3 \\ CASE & NULL \\ ANIM & - \end{array}\right] \end{array}\right] \cup
$$

$$
\begin{array}{l} f3 \\ f6 \\ f8 \end{array}\left[\begin{array}{ll} PRED & 'eat<(f8\,SUBJ),(f8\,OBJ)>' \\ TENSE & PAST \\ SUBJ & \left[\begin{array}{ll} PERS & 1 \\ CASE & NULL \\ ANIM & + \end{array}\right] \end{array}\right]
$$

$$
\begin{array}{l} f3 \\ =f6 \\ f8 \end{array}\left[\begin{array}{ll} PRED & 'eat<(f8\,SUBJ),(f8\,OBJ)>' \\ TENSE & PAST \\ SUBJ & \left[\begin{array}{ll} PERS & 1 \\ CASE & NULL \\ ANIM & + \end{array}\right] \\ OBJ & \begin{array}{l} f5 \\ f7 \end{array}\left[\begin{array}{ll} PRED & 'rice' \\ PERS & 3 \\ CASE & NULL \\ ANIM & - \end{array}\right] \end{array}\right]
$$

Figure 7: Unification process at functional *f*3

Thus the unification of functional gives the total solution of successful parse of the sentence, which is given by the f-structure given in Figure 8. Therefore f-structure is an attribute-value matrix that holds all the syntactic and even semantic information of the sentence.

$$
\begin{array}{l} f1 \\ f3 \\ f6 \\ f8 \end{array}\left[\begin{array}{ll} PRED & 'eat<(f8\,SUBJ),(f8\,OBJ)>' \\ TENSE & PAST \\ SUBJ & \begin{array}{l} f2 \\ f4 \end{array}\left[\begin{array}{ll} PRED & 'pro' \\ PERS & 1 \\ NUM & PL \\ CASE & NULL \\ ANIM & + \end{array}\right] \\ OBJ & \begin{array}{l} f5 \\ f7 \end{array}\left[\begin{array}{ll} PRED & 'rice' \\ PERS & 3 \\ ANIM & - \\ CASE & NULL \end{array}\right] \end{array}\right]
$$

Figure 8: f-structure of the sentence (S1)

But when we try to parse the sentence (S2) we are left with an invalid unification, and this is because the attribute CASE of the pronoun '*aamader*' has value GEN, but the head verb '*kheyechhilaam*' is associated with subject having CASE value NULL. Hence parsing fails, which is desired, as shown in Figure 9.

Parsing the sentence (S3) is also unsuccessful. As the head verb suggests that subject must have the value '1' for the attribute PERS, while the pronoun '*tomraa*' has value '2' for the attribute PERS, as shown in Figure 10.

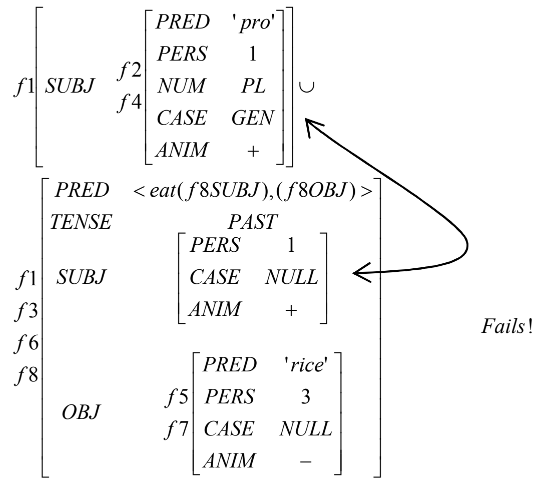

Figure 9: Failed unification of (S2) and (S3)

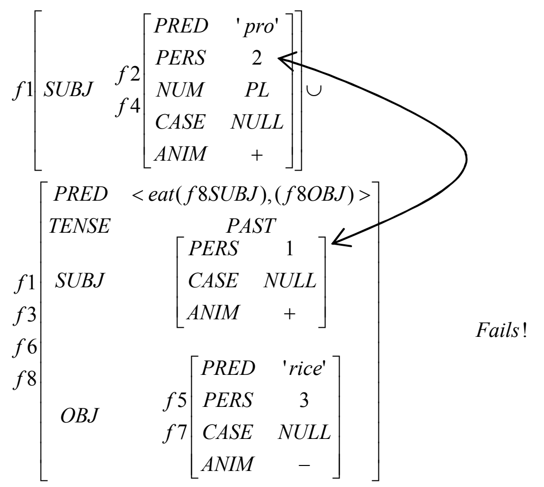

Figure 10: Failed unification of (S2) and (S3)

4. Discussion

We present a very simple framework for parsing Bangla using LFG, with some examples of successful parses of grammatically correct simple sentences and unsuccessful parses of ungrammatical sentences. This is just the beginning however, and we are still a long way from creating a usable computational grammar for Bangla [4] L. Complete Bangla Lexicon with the information needed for LFG, and a systematic study of Bangla grammar required to formulate the syntactic rules.

5. References

[1] E. Charniak, "*Statistical Parsing with a Context-Free Grammar and Word Statistics*," In Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97), pages 598-603, Menlo Park, July 27-31 1997. AAAI Press.

[2] M. Collins, "*Three Generative, Lexicalized Models for Statistical Parsing*," In Philip R. Cohen and Wolfgang Wahlster, editors, Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, pages 16-23, Somerset, New Jersey, 1997.

[3] R.M. Kaplan, *"The Formal Architecture of Lexical Functional Grammar",* Journal of Information Science and Engineering, pp 305-322, vol. 5, 1989.

[4] L. Sadler, "*New Developments of Lexical Functional Grammar*", 1996.

[5] S. Joshi, *"Selection of Grammatical and Logical Functions in Marathi",* PhD thesis, 1993.

[6] P. Sengupta and B.B. Chaudhuri, *"A Delayed Syntactic-Encoding-based LFG Parsing Strategy for an Indian Language – Bangla",* Association for Computational Linguistics, 1997.