A Computational Grammar of Bangla using the HPSG formalism: Developing the First Phase

Naira Khan[*]
Dr. Mumit Khan[**]

In this article we describe an endeavor to develop a computational grammar of Bangla in the Head-Driven Phrase Structure Grammar (HPSG) formalism and the resultant first phase of such an undertaking. Although simplistic in its makeup the HPSG formalism is a highly developed framework which allows us to describe languages in terms of hierarchical feature structures making it a powerful tool for computational as well as psycholinguistic research whereby features specific to a language can be captured and simultaneously cross-linguistic features common to all languages i.e. language universals may be explored. To implement our grammar we used the Linguistic Knowledge Building (LKB) system - a powerful tool comprising both a syntactic and a semantic level that allows the user to parse as well generate by using the formalism to code in linguistic rules through feature structures and feature unification.

1. Introduction

Developing a computational grammar for a natural language is undoubtedly a complicated task. A computational grammar is a grammatical description of a natural language in a computational framework – one that is inherently important in various applications of Natural Language Processing (NLP), ranging from low-end applications such as a grammar checkers or spelling-checkers to high-end ones such as dialogues systems, machine translation, etc. These involve parsing human sentences which pose considerable problems due to the ambiguous nature of natural languages. Hence, grammar development, also known as grammar engineering, presents a formidable task and requires a formalism that mirrors a very high-level programming language and skills in developing an implementable account of linguistic phenomena (Copestake, 2002). As described in the following, we have attempted to embark on such an endeavor for Bangla, using HPSG - a formalism that is striking in its own right; and an implementation platform known as LKB also unique in that it is

[*] Lecturer, Dept. of Linguistics, University of Dhaka
[**] Head, Centre for Research on Bangla Language Processing, BRAC University

linguistically motivated, has semantic representation in the form of Minimal Recursive Semantics (MRS), and most importantly, it is bidirectional being equipped with both a parser and a generator (Copestake and Flickinger, 2000).

1.1 The Formalism: Head-driven Phrase Structure Grammar (HPSG)

Developed by Pollard and Sag (1987, 1994), the Head-driven Phrase Structure Grammar (HPSG) is a non-derivational generative grammar theory. It evolved directly from attempts to modify its immediate predecessor - Generalized Phrase Structure Grammar (GPSG). The name 'Head-driven Phrase Structure Grammar' was chosen to reflect the increasingly recognized importance of information encoded in the lexical heads of syntactic phrases. In other words, the formalism is based on 'lexicalism' i.e. the lexicon is more than just a list of entries in that it is in itself richly structured with individual entries being marked with types and the types form a hierarchy. HPSG uses a uniform formalism and it is its modular organization that makes it so attractive for natural language processing. (Sag and Wasow, 1999)

1.2 The Implementation Platform: LKB and the LinGO Grammar Matrix

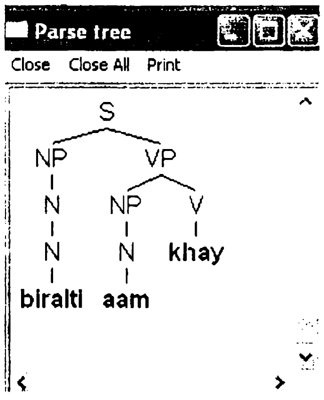
An ongoing collaboration, the Linguistic Grammars Online (LinGO) initiative at Stanford University makes available a number of open-source, HPSG related computational resources, and includes partners in the US, Europe and Asia. Freely available online, these resources include grammars, lexicons, and the Linguistic Knowledge Based (LKB) Grammar Engineering Platform. (Sag and Wasow, 1999)
Used most extensively for HPSG, the LKB system is a grammar and lexicon development environment for typed feature structure grammars. The LKB source is freely available and implemented in Common Lisp. Typed feature structure languages are essentially based on one data structure and one operation – the typed feature structure and unification, respectively. The constraints on the type system provide a way of capturing linguistic generalizations. This is a powerful combination in that it allows a grammar developer to write grammars and lexicons that can be used to parse and generate natural languages (Copestake, 2002).

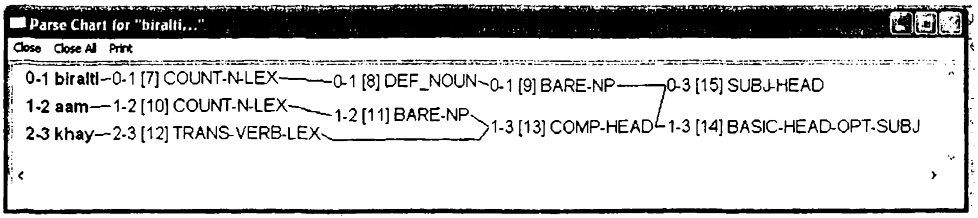
The largest grammar developed on the LKB is the LinGo project's English Resource Grammar (ERG) (Copestake, 2002). The LKB was also used as the test-bed for a number of teaching grammars and smaller-scale grammars for other languages including Japanese (JACY) and Greek (Modern Greek Resource Grammar). This grammar writing experience across a variety of languages was sublimated into devising the LinGO Grammar Matrix -- a framework to aid in the development of broad-coverage, precision, implemented grammars for diverse natural languages. These resources can be accessed online at the following URL: http://lingo.stanford.edu/

1.3 Previous work

To our knowledge no complete or extensive endeavor has yet been undertaken for the development of a grammar of Bangla in a computational framework. However, the HPSG formalism itself is not new to Bangla, as we find that both HPSG and LKB have been used extensively for the implementation of Bangla compound verbs (Paul, 2004). This article builds on a rudimentary grammar (Khan and Khan, 2006) including new enhancements (Mahmud and Khan, 2007) and some new proposals. The primary difference here is the inclusion of case marking and case relations, which the previous grammar was completely devoid of. Other differences include a new proposal for encoding compound verbs and addressing the issue of adverb movement within the parameters of grammaticality. Although other formalisms such as Lexical Functional Grammar (LFG) (Sengupta and Chaudhuri, 1997; Haque and Khan, 2005) and Context Sensitive Grammar (CSG) (Hoque and Ali, 2004; Murshed, 1998; Selim and Ikbal, 1999) have also been used in order to parse some simple sentences of Bangla, it should be mentioned that these endeavors were always an attempt to parse a few sentences rather than build a complete computational grammar for Bangla. Apart from these, parsing in Bangla has primarily been statistical or based on shallow parsing with little or no semantic representation (Billah and Sikder, 2004; Saha, 2006; Naskar and Bandyopadhyay, 2005). In fact grammar engineering is still in its infancy in terms of South Asian languages. The only example so far would be an unpublished rudimentary grammar of Hindi in HPSG/LKB by Jeremy Kahn, produced as an output of a Grammar Engineering course (Ling 567) at the University of Washington. Apart from that, some sparse amount of work on certain aspects such as code-switching (Goyal et al, 2003); and considerable amount of work in LFG on Hindi can be found.

2. A Computational Grammar for Bangla: The First Phase




Parse Chart for "biralti..."
Close Close All Print
0-1 biralti—0-1 [7] COUNT-N-LEX—0-1 [8] DEF_NOUN—0-1 [9] BARE-NP—0-3 [15] SUBJ-HEAD
1-2 aam—1-2 [10] COUNT-N-LEX—1-2 [11] BARE-NP
2-3 khay—2-3 [12] TRANS-VERB-LEX—1-3 [13] COMP-HEAD—1-3 [14] BASIC-HEAD-OPT-SUBJ

It should be mentioned here that at this stage the Bangla grammar is in LKB proper written in LISP and hence has been encoded in basic Latin with a transcription system devised by us based on phonetic makeup. However, LKB also has a wrapper program known as LKBTrollet that allows Unicode support for better multilingual support and a cleaner, more logical interface (http://wiki.delph-in.net/moin/LkbTrollet). With this program we will ultimately be able to parse Bangla sentences in the Bangla script, and we have marked this as one of our short term goals.

In the following sections we have addressed various linguistic phenomena relevant to developing the first phase of the computational grammar for Bangla in HPSG, along with the implementation methodology.

2. Methodology

The Starter Grammar: One of the best tools that the LinGO project provides is the starter grammar. Instead of writing up a grammar from scratch, the Lingo Grammar Matrix (available at: http://www.delph-in.net/matrix/) allows you to automatically generate a 'starter grammar' consisting of the bare minimum lexical entries and rudimentary rule types by configuring certain cross-linguistically common typological properties in accordance with the grammar of the target language. The starter grammar consists of lexical entries comprising two nouns, two verbs, rules for basic word order, types and constraints for determiners, basic sentential negation, yes-no question strategies and co-ordination strategies.

The Bangla Grammar: The Bangla HPSG grammar written and implemented in the LKB system by building on the Matrix starter grammar comprises the following linguistic phenomena:

Basic Word Order and Phrase Structure Rules: Although Bangla is a non-configurational language we must be careful to note that word-order is not completely free in that all possible arrangements of words within a sentence are not grammatical. Although it can be said that Bangla has a pragmatically free word order, nevertheless word ordering is predominantly Subject-Object-Verb (SOV).

The Lingo Grammar Matrix allows you both options in that you can begin with a starter grammar with:

a. an SOV word order, whereby other orders will then be considered ungrammatical but can be included through extra rules.
b. a pragmatically-free word order, whereby the structures that are ungrammatical can subsequently be constrained to prevent the grammar from over-generating.

For our grammar we decided to begin with an SOV ordering as described below:

Eg:	*ami*	*aam*	*kha-i*
	I.1.sg	mango	eat.1.pres-hab
	S	O	V

Hence phrase structure rules in Bangla have been coded accordingly treating it as Head-Final Specifier-Initial with the following rule instance:

```
comp-head := comp-head-phrase.
subj-head := subj-head-phrase.
```

And the following type definitions from the Matrix:

```
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 < > ].
```

Agreement and Relations: Bangla is a fusional language with agreement issues primarily pertaining to Subject-Verb agreement where the verb is marked for person, grade, tense, aspect and modal information.

The prototypical structure of the Bangla verb can be shown as:

ROOT+[tense+aspect+person-grade]/imp {2nd/obl 2nd/1st incl}

Apart from this, certain adjectives have gender agreement with nouns and number agreement is relevant for certain determiners and nouns.

Built on instances of the Matrix these agreement issues are dealt with the following type definition:

```
png :+ [ PER person,
         NUM number,
         GRD grade,
         TPC topical,
        GEN gender ].
```

The subtypes of **png** and their respective values are described in the following:

Person [PER]

For person agreement we have the subtype PER with values 'first', 'non-first', 'second' and 'third'. The value 'non-first' is introduced as the honorific forms of the second person and third person have identical verbforms:

tini dekhen

apni dekhen

This is done through the following code in the Type file:

```
person := *top*.
first := person.
non-first := person.
second := non-first.
third := non-first.
```

Grade [GRD]

In Bangla the second and third person can be further subdivided into grades:

	2P	3P
Honorific	*apni*	*tini*
Non-Honorific	*tumi*	*she*
Pejorative	*tui*	

These are introduced through the subtype GRD and the values 'hon' for honorific, 'non-hon' for non-honorific and 'pej' for pejorative. An alternative way of coding grade would be to split the values of the second and third person within the subtype of PER rather than introducing a new subtype.

```
grade := *top*.
hon := grade.
non-hon := grade.
pej := grade.
```

Topicality [TPC]

A new subtype TPC is introduced as Bangla nouns take different plural markers based on topicality levels.

ra – a plural marker that typically collocates with nouns of high topicality and not with those of low topicality.

gulo - a plural marker that typically collocates with high as well as low topical nouns.

It has the values of 'high' and 'non-high' which usually correspond to topicality issues of animacy, human, non-human, etc.

```
topical := *top*.
high := topical.
non-high := topical.
```

Gender [GEN]
Although gender agreement is not prolific as Bangla only has lexical gender, certain adjectives and nouns have gender agreement only in the feminine form and hence the subtype GEN has values 'fem' and 'non-fem'. As all nouns whether masculine or feminine can be modified by the masculine adjective the 'masc' value is kept undefined.

Number [NUM]
Bangla verbs do not have number agreement. However, the subtype NUM serves the purpose of defining pluralizers and definiteness markers between count and mass nouns. It has the values 'sg', 'non-sg' and 'pl'.

Case Relations: Bangla has 6 morphosyntactic cases where 5 of them are analytic and one periphrastic. Amongst the analytic case markers, accusative and dative have identical markers and hence the two comprise a single value 'acc-dat'. The instrumental and locative cases are a similar issue. In case of the absence of case markers in transitive situations we have encoded intrinsic case information on the arguments of verbs which will match the protopycal nominative and accusative nouns in terms of topicality. It is due to the limited size of the lexicon that we were able to do this. This is an oversimplified view of case-marking in Bangla because there are several issues of morphosyntactic ambiguity which are not dealt with in the first phase of this grammar.
The feature CASE is defined for the type noun and adpositions.

```
noun :+ [CASE case].
np :+ [CASE case].
case:= *top*.
nom := case.
acc-dat := case.
gen := case.
instr-loc:= case.
abl := case.
```

Adpositions are constrained through the head-complement rules to exclude adpositions as head daughters in order to make them post-positions rather than prepositions.

The case information is added to the valence features of the transitive verb types. Noun and pronoun types are also modified to reflect case values.

It should be denoted that the Bangla case marking system is extremely ambiguous from a morphosyntactic perspective due to interchangeable case markers and case relations being determined through ordering of constituents as well as topicality. In our grammar we have kept the most simplistic system of case relations, however, in another rudimentary grammar (Mahmud and Khan, 2007) case marking has been implemented in the following manner:

A simple straight-forward approach was taken whereby, the related constituents of a verb was disambiguated by allowing a particular constituent (subject or object) to take a group of case-markers instead of a particular one, where different case-markers could morphologically assign different cases for a single constituent, but its role would still be identifiable. The ambiguity problem was resolved using the type hierarchy system of LKB as shown in the following figure (Mahmud and Khan, 2007):

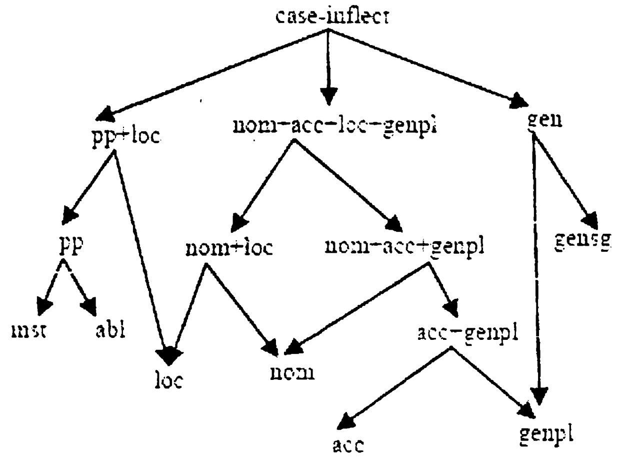


Fig. 1. Proposed case markers' hierarchy for 'Bangla'.

Types and Lexical Entries: For our first phase we have encoded the following major word classes as lexical entries and their respective types written into the lexicon.tdl and bangla.tdl files respectively, in the starter grammar:

Nouns: The nouns consist of, at the very first level, a basic noun form that has determiner optionality in that a Noun Phrase (NP) can consist of [noun ±

determiner]. The basic noun-lex is then classified into various types of common noun and proper noun that inherit from the basic noun form but have constraints particular to each type. Common nouns are constrained to be in the 3rd person and are subdivided into:

- Common nouns that don't take determiners.
- Count nouns.
- Mass nouns.

Count nouns are further constrained as singular in number (NUM sg) and according to topicality as nouns in Bangla have different plural markers based on differences in levels of topicality. Mass nouns are constrained for number information (NUM non-sg) in order to prevent it from taking any plural markers. Proper nouns are constrained to be in the 3rd person and don't take determiners. Proper nouns in the vocative case have not been addressed at this level.

An example of a type:

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

A typical nominal lexeme for 'cow':

```
;cow
goru := count-n-lex &
   [ STEM < "goru" >,
     SYNSEM.LKEYS.KEYREL.PRED "_goru_n_rel" ].
```

Pronouns: Pronouns are types that inherit from the basic noun-lex but are constrained to be determinerless and do not inflect.

Pronouns are constrained for number, person and case information in agreement with its respective verb form due to pronominal marking on the verb. The constraints have been split in that the number and person information is coded in the type file while the case information comes from the lexical entry.

Type definitions:

```
2sg-pronoun-lex := pronoun-lex &
  [ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG [ PER second,
                                       NUM sg ] ].
2sg-nh-pronoun-lex := 2sg-pronoun-lex &
  [ SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG [ GRD non-hon] ].
```

```
2sg-nh-nom_pron-lex := 2sg-nh-pronoun-lex.
```

Lexical entry for the second person singular non-honorific pronoun:

```
;2p.sg (you)
tumi := 2sg-nh-nom_pron-lex &
   [ STEM < "tumi" >,
     SYNSEM.LOCAL.CAT.HEAD.CASE nom ].
```

Determiners: The basic determiner type is constrained to have SPR value that is compatible with nouns that take determiners. The determiners are treated as quantifying or demonstrative pronouns and are subdivided into demonstratives and non-demonstratives. The non-demonstratives are then further divided into proximal and distal demonstratives. Further distinctions can be made by building on this principle.

Examples of type definitions:

```
demonstrative_q_rel := reg_quant_rel.
non+demonstrative_q_rel := reg_quant_rel.
proximal+dem_q_rel := demonstrative_q_rel.
distal+dem_q_rel := demonstrative_q_rel.
```

The lexical entry inherits from 'determiner-lex' with the type definition defining the predicate relation:

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

Verbs: The starter grammar provides a basic verb type and two subtypes as transitive and intransitive forms that inherit from the basic verb.

For Bangla we created a new transitive verb (trans-verb-lex) which inherits from the transitive verb as well as the basic verb and which was constrained for case agreement with the nouns that act as the arguments as word ordering can mark case relations when no overt marker is present. The case information was coded in accordance to a definite order to prevent ambiguities. Argument order must follow head-final ordering.

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

A basic ditransitive verb was created by building on the principle of the transitive verb which inherits from the transitive verb and basic verb and is constrained for case information.

```
ditrans-verb-lex := verb-lex & ditransitive-lex-item &
   [ SYNSEM.LOCAL.CAT [VAL [ SUBJ < #subj >,
                             COMPS < #comp1 , #comp2 > ]],
    ARG-ST < #subj &
              [ LOCAL.CAT [ VAL [ SPR < >,
                                  COMPS < > ],
                HEAD noun & [CASE nom]]] ,
               #comp1 &
              [ LOCAL.CAT [ VAL [ SPR < >,
                                  COMPS < > ],
                            HEAD noun & [CASE acc-dat]]],
               #comp2 &
              [ LOCAL.CAT [ VAL [ SPR < >,
                                  COMPS < > ],
                        HEAD noun & [CASE acc-dat]]] > ].
```

Building on this instance it is possible to create different types of transitive and ditransitive verbs which differ in terms of case information of its arguments. In such cases it is more economical to create a generic type devoid of case constraints and to create specific supertypes that inherit form it.

Lexical entry for verbs:

As Bangla verbs minimally consist of a root and its inflection where the inflection carries agreement constraints, hence the lexical entries can be written in two ways depending on the way the inflectional rules are coded. Hence the lexicon may consist of full verb forms for each person variation or the root forms only.

Full verb form:

```
hasho := intransitive-verb-lex &
```

```
  [ STEM < "hasho" >,
    SYNSEM [ LOCAL.CAT.VAL.SUBJ
<[LOCAL.CONT.HOOK.INDEX.PNG [ PER second,

GRD non-hon] ] >,
                  LKEYS.KEYREL.PRED "_hasho_v_rel" ] ].
```

Only the root form:

```
hash := intransitive-verb-lex &
  [ STEM < "hash" >,
    SYNSEM.LKEYS.KEYREL.PRED "_hash_v_rel" ] ].
```

This is discussed in more detail in the 'Verbal Inflection' section.

A Proposal for Implementing Compound Verbs:

Any analysis of a Bangla corpus will reveal that about half the verbs appear in larger complexes namely compound verbs. The prototypical structure of a compound verb is (Paul, 2004):

V1-conjunctive participial + V2, where V1=matrix verb; V2=light verb
eg. *kin - e* + *dilo*

The conjunctive participials as well the number of V2s are fixed in Bangla whereby the V1 remains static and only the V2 is inflected.

Ideally we want to be able to parse both the verbs with type definitions that encode the V1 verbal as well as the V2 light verb. However at this stage we base the parse on entire compound as one entity by treating the space between the two verbs as a character itself with the verb beginning at V1 and ending at V2 with the inflection of the V2 acting as the input for forms to be generated. This idea is based on the LKB's ability to treat multiword lexemes (eg. ice cream) with one affixation site. This is specified on a per-entry basis, via the user-definable function find-infl-pos which allows affixation on the rightmost element (Copestake, 2002).

Adjectives: In order to encode adjectives, it is necessary to have head-modifier rules. The Matrix distinguishes between intersective and scopal modification and has instances for both. In Bangla adjectives are of two types: attributive and predicative. Attributive adjectives are pre-head modifiers while predicative adjectives have a different syntactic structure with an invisible copula that manifests when the tense changes. In the first phase of this grammar, we have not dealt with the predicative adjectives and its respective PS rules.

The type definition for a simple attributive adjective for Bangla is:

```
adjective-lex := basic-adjective-lex & intersective-
mod-lex &
  [ SYNSEM [ LOCAL.CAT [ HEAD.MOD < [ LOCAL.CAT [ HEAD
noun,

VAL [SPR < > ]]]>,
                      VAL [ SPR < >,
                          SUBJ < >,
                          COMPS < >,
                          SPEC < > ],
                      POSTHEAD - ]]].
```

Bangla adjectives don't have person, number or case agreement. However, some adjectives do have gender marking in that the feminine counterpart may be overtly marked and only appropriate for lexically feminine nouns. Eg- *shundori.* i.e. *shundori meye ; *shundori chhele.* This can be very easily coded by adding an extra subtype for gender (GEN fem) and adding constraints to the MOD value of an adjectival subtype and corresponding constraints to particular nouns. Lexically feminine nouns can also combine with masculine adjectives and this can be achieved by leaving the masculine value undefined so that the masculine adjectives can combine with all nouns.

Lexical entry for an adjective will inherit from 'adjective-lex':

```
;bad/rotten
pocha := adjective-lex &
   [ STEM < "pocha" >,
     SYNSEM.LKEYS.KEYREL.PRED "_pocha_n_rel" ].
```

Adverbs: The adverb type definition inherits from matrix supertypes by default and constrains the modified constituent to be verbal. The adverb then has to be constrained to modify the V, VP or S. In Bangla, adverb movement is prolific in that if it is moved around within the VP, the sentence and its syntactic tree remain grammatical.

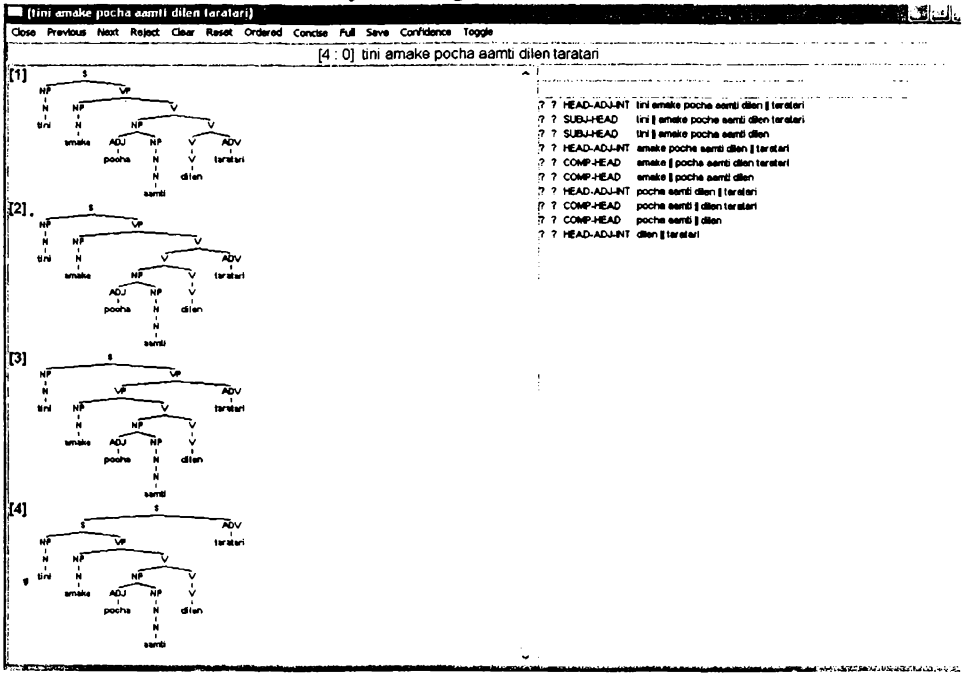
Hence the following type definition for adverbs is adequate at this stage:

```
adverb-lex := basic-adverb-lex & intersective-mod-lex
&
  [ SYNSEM [ LOCAL.CAT [ HEAD.MOD < [ LOCAL.CAT.HEAD
verb ]>,
                         VAL [ SPR < >,
                            SUBJ < >,
                            COMPS < >,
                            SPEC < > ]]]].
```

The lexical entries for adverbs inherit from 'adverb-lex':

```
;slow
aste := adverb-lex &
   [ STEM < "aste" >,
     SYNSEM.LKEYS.KEYREL.PRED "_aste_n_rel" ].
```

Due to adverb movement there will be multiple parse trees from a single sentence. The LKB system allows us to compare these in terms of adverb placement on particular nodes of the tree. These will however, have to be constrained as all the trees may not be grammatical.



(tini amake pocha aamti dilen taratari)

[4 : 0] tini amake pocha aamti dilen taratari

```
? ? HEAD-ADJ-INT  tini amake pocha aamti dilen | taratari
? ? SUBJ-HEAD     tini | amake pocha aamti dilen taratari
? ? SUBJ-HEAD     tini | amake pocha aamti dilen
? ? HEAD-ADJ-INT  amake pocha aamti dilen | taratari
? ? COMP-HEAD     amake | pocha aamti dilen taratari
? ? COMP-HEAD     amake | pocha aamti dilen
? ? HEAD-ADJ-INT  pocha aamti dilen | taratari
? ? COMP-HEAD     pocha aamti | dilen taratari
? ? COMP-HEAD     pocha aamti | dilen
? ? HEAD-ADJ-INT  dilen | taratari
```

Lexical Rules: As defined by the Matrix, lexical rules for inflection are written in the 'irules.tdl' and the 'lrules.tdl' files and inherit from the types 'infl-ltow-rule' or the 'infl-ltol-rule', depending on whether the form is fully inflected or not. There are other infl* rule types in the Matrix for various kinds of lexical changes. For the first phase however we've used only the two mentioned above.

The inflection is defined through instances of rule types in the irules.tdl where spelling rule subrules are denoted on a line beginning with %suffix. After %suffix there is a list of pairs in which the first member matches the input form and the second member describes the output form and thus can handle complex morphophonemic changes. The * matches an empty string and the ! signals a letter set. More specific rules are placed at the right and full forms can be written for suppletive forms. (Copestake, 2002).

In the following we will describe the implementation of lexical rules in more details through nominal and verbal instantiations:

Nominal Inflection: In the first phase nominal inflection is limited to plural and definiteness markers. In terms of pluralizers, nouns in Bangla differ in which plural marker can be added on the basis of topicality. The two most prolific plural markers are 'gula' and 'ra'. The marker 'gula' can be added to almost anything except for a few highly topical nouns. 'ra' on the other hand can only be added to nouns of high topicality. Hence TPC value constraints (high vs. non-high) prevent inappropriate marking. In cases of nouns that can have both plural markers the TPC value has been left undefined.

The type definition for plural markers:

```
;;; for nouns + 'gulo'
plur_noun1-lex-rule := infl-ltow-rule &
  [ SYNSEM.LOCAL [ CAT.VAL.SPR
                          < [ LOCAL.CONT.RELS < ! [PRED
reg_quant_rel] ! > ] >,
            CONT.HOOK.INDEX.PNG [ PER third,
                                    NUM sg,
                                 TPC non-high ] ] ].
;;; for nouns + 'ra'                                    .
plur_noun2-lex-rule := infl-ltow-rule &
  [ SYNSEM.LOCAL [ CAT.VAL.SPR
                          < [ LOCAL.CONT.RELS < ! [PRED
reg_quant_rel] ! > ] >,
            CONT.HOOK.INDEX.PNG [ PER third,
                                    NUM sg,
                                     TPC high] ] ].
```

And the suffix rules are:

```
plur-noun1 :=
%suffix (* gulo) (!t !tgulo)
```

```
plur_noun1-lex-rule.

plur-noun2 :=
%suffix (* ra) (!t !tra)
plur_noun2-lex-rule.
```

Here the letter set is:

```
%(letter-set (!t bcdfghjklmnpqrstvwxz))
```

Definiteness markers can be added concatenatively to any common or count noun. Mass nouns are constrained through NUM values to prevent them from taking plural markers.

Verbal Inflection: In Bangla the prototypical verb has the following structure:

ROOT +	[aspect+tense+person-grade] / mode			
hash +	*ch* +	*il* +	*e - n*	
laugh.root	prog.	past.	3P.hon	

The root, stripped of all inflection can only function as a stem in the second person pejorative imperative form. In all other forms the verb carries person, tense, aspect and modal information in the form of inflections and the root cannot occur on its own.

Verbal inflection can be coded in two ways:

a) Each person form of the verb root is added as a lexical entry whereby the person ending can act as the input for the suffix change rule for all other TAM forms.

```
hashi := intransitive-verb-lex &
   [ STEM < "hashi" >,
     SYNSEM [ LOCAL.CAT.VAL.SUBJ
<[LOCAL.CONT.HOOK.INDEX.PNG [ PER first,
GRD non-hon]]>,
                 LKEYS.KEYREL.PRED "_hashi_v_rel" ] ].
```

Here the ‘i’ of ‘hashi acts as the input for other forms of TAM inflection introduced by the following type definitions:

```
1p_verb-lex-rule := infl-ltow-rule &
  [ SYNSEM.LOCAL.CAT.VAL.SUBJ < [
LOCAL.CONT.HOOK.INDEX.PNG [ PER first ]] >,
    DTR.SYNSEM.LOCAL.CAT.HEAD verb ].

1p-past_verb := 1p_verb-lex-rule.
```

And the following suffix rule:

```
1p-past-verb :=
%suffix (* lam) (!ti !tlam) (ai alam) (jai gelam)
(khai khelam) (pai pelam) (gai gelam) (ei ilam) ;
(ei=dei/nei)
```

```
1p-past_verb.
1p-pastprog-verb :=
%suffix (* chilam) (!ti !tchilam) (ai acchilam) (ei
icchilam)
1p-past_verb.
```

Thus the the above two rules can generate the following forms from 'dekhi':

hashi > hashlam
hashi > hashchilam

b) An alternative to this would be to use the lrules.tdl file to create forms from the roots by using an instance of an infl* rule such as:
type definitions:

```
1p_v-lex-rule := infl-add-only-no-ccont-ltol-rule &
   [ SYNSEM.LOCAL.CAT.VAL.SUBJ <[
LOCAL.CONT.HOOK.INDEX.PNG [ PER first ]]>,
    DTR.SYNSEM.LOCAL.CAT.HEAD verb ].
```

1p-past-verb := 1p_v-lex-rule.

A lexical rule:

```
1p-past-verb := 1p_v-lex-rule.
```

And a simpler suffix rule:

```
1p-verb :=
%suffix (* i)
1p_v-lex-rule.
1p-past-verb :=
%suffix (* lam)
1p_v-lex-rule.
```

This not only simplifies the irules file, it allows us to keep only the root form in the lexicon and generate all else with lexical rules.

```
hash := intransitive-verb-lex &
   [ STEM < "hash" >,
     SYNSEM.LKEYS.KEYREL.PRED "_hash_v_rel" ] ].
```

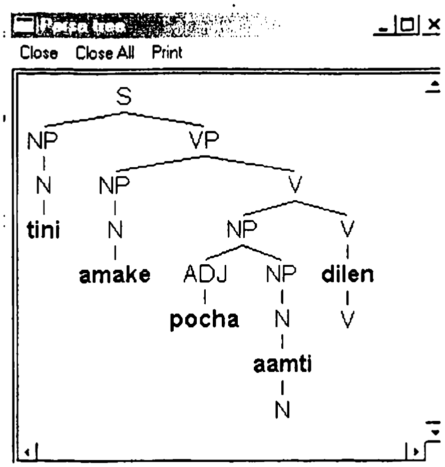
Generated forms:	hash > hashi
	hash > hashlam

3. Results

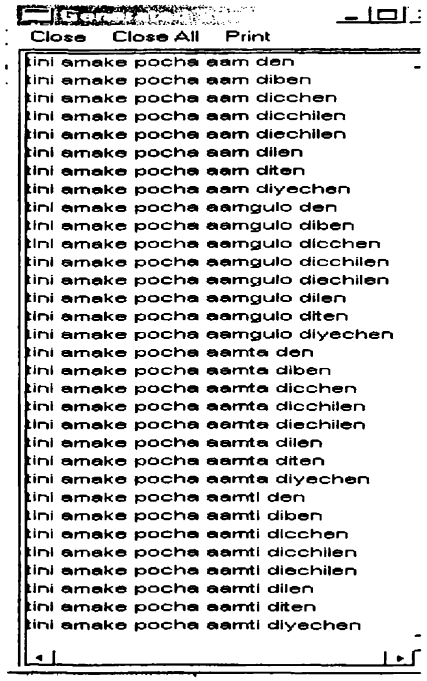
With the above grammar it is possible to

a) recognize and parse a considerable number of grammatical sentences of Bangla
b) generate various inflected forms from each parse.

An example of a parsing and generating a sentence is given in the following:
The sentence 'tini amake pocha aamti dilen' (s/he gave me the rotten mango) is parsed below:

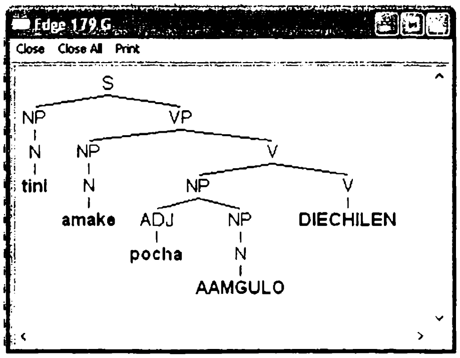


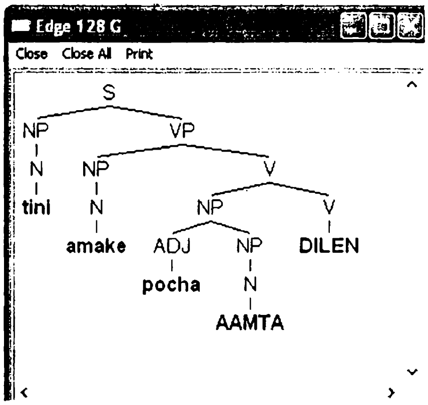
And forms generated from the above sentence is given in the following:

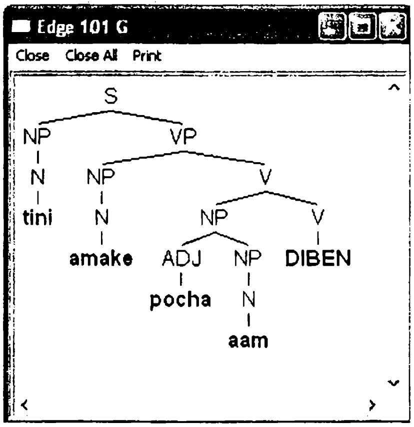


```
tini amake pocha aam den
tini amake pocha aam diben
tini amake pocha aam dicchen
tini amake pocha aam dicchilen
tini amake pocha aam diechilen
tini amake pocha aam dilen
tini amake pocha aam diten
tini amake pocha aam diyechen
tini amake pocha aamgulo den
tini amake pocha aamgulo diben
tini amake pocha aamgulo dicchen
tini amake pocha aamgulo dicchilen
tini amake pocha aamgulo diechilen
tini amake pocha aamgulo dilen
tini amake pocha aamgulo diten
tini amake pocha aamgulo diyechen
tini amake pocha aamta den
tini amake pocha aamta diben
tini amake pocha aamta dicchen
tini amake pocha aamta dicchilen
tini amake pocha aamta diechilen
tini amake pocha aamta dilen
tini amake pocha aamta diten
tini amake pocha aamta diyechen
tini amake pocha aamti den
tini amake pocha aamti diben
tini amake pocha aamti dicchen
tini amake pocha aamti dicchilen
tini amake pocha aamti diechilen
tini amake pocha aamti dilen
tini amake pocha aamti diten
tini amake pocha aamti diyechen
```

Each of the generated sentences can be expanded into a tree as well, as shown from the edges 179, 128 and 101 respectively.







Note that the terminal generated forms appear in the uppercase as opposed to the non-generated forms.

4. Limitations:

Being the first phase this grammar has a number of limitations typical of any rudimentary grammar, that need to be overcome and are marked as our short-term immediate goals:

- It has a limited lexicon. As the lexicon will be expanded subsequently the irules and grammar rules will have to rewritten to a certain extent.
- The grammar is simple, in that it deals primarily with transitive, intransitive and ditransitive situations.
- Case marking needs to be disambiguated and incorporated into the work.
- A proper framework is needed to deal with compound verbs where both the verbs are parsed separately.
- Sentence situations with zero copula have not been addressed and need to be encoded before the grammar complexifies.
- At this point the grammar has been written in a transcription system using basic Latin, however, ultimately with the help of LKBTrollet we want to be able to parse Bangla sentences in the Bangla script.

5. Future work

The goal of developing a computational grammar for Bangla will inevitably coincide with the goals of any grammar development endeavor. Our first phase grammar is an attempt to trigger the beginnings of writing up a full-scale computational grammar of Bangla with certain long term goals outlined below:

- ♦ To write a computational description of Bangla.
- ♦ To test various linguistic hypotheses of Bangla using HPSG.
- ♦ To be able to include Bangla in various practical applications of NLP systems.
- ♦ To create a resource of computational information for languages of similar grammatical structure.
- ♦ To test hypotheses about linguistic universals that cut across languages.
- ♦ To facilitate the exchange of data and analyses of a wide range of phenomena across diverse languages.

6. Conclusion

This article is based on the first phase of the development of a computational grammar for Bangla – a gargantuan task that requires not only a description of a natural language in a computational framework but an implementation tool suited to the framework. The HPSG formalism is a rich linguistic framework built for the mammoth task of grammar engineering and combined with LKB, provides a suitable platform for the implementation of this formidable task. As shown in this article, various aspects of linguistic phenomena of Bangla have been accounted for and reinterpreted in an implementable framework. As an end result the rudimentary grammar that we have built is capable of parsing and generating a large number of sentences of Bangla. It is thus a point of beginning for further development of a large scale Bangla grammar and consequently a computational description that will prove extremely useful both from a computational and a linguistic perspective.

References

Billah, Md. M and Shikder, M R. 2004. Syntax Analysis of Bangla Phrases. In *Proceedings of skjdfhasilf*, pages 669-673, Dhaka.

Copestake, A and Flickinger, D. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000).* Athens, Greece.

Copestake, A. 2002. *Implementing Typed Feature Structure Grammars.* CSLI Publications, Stanford.

Haque, M N and Khan, M. 2005. Parsing Bangla Using LFG: An Introduction. *BRAC University Journal*, 2(1):105-110.

Hoque, M M and Ali, M M. 2004. Context-Sensitive Phrase Structure Rule for Structural Representation of Bangla Natural Language Sentences. In *Proceedings of International Conference on Computer and Information Technology,* pages 615-620, Dhaka.

Khan, N and Khan, M. 2006. Developing a Computational Grammar for Bengali using the HPSG Formalism. In *Proc. of 9th International Conference on Computer and Information Technology (ICCIT),* Dhaka, Bangladesh.

Kumar, S N and Bandyopadhyay, S. 2005. A Phrasal EBMT System for Translating English to Bengali. In *Conference Proceedings: the tenth Machine Translation Summit,* pages 372-379, Phuket, Thailand.

Mahmud, A and Khan, M. 2007. Building a Foundation of HPSG-based Treebank on Bangla Language (415). In *Proceedings of the* ICCIT, Dhaka.

Manav P. G, Mital R., Mukerjee A., Raina A. M., Sharma D, Shukla P. et al. 2003. Saarthaka - A Bilingual Parser for Hindi, English and code-switching structures. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest.

Murshed, M. M. 1998. Parsing of Bengali Natural Language Sentences. In *Proceedings of International Conference on Computer and Information Technology,* pages 185-189, Dhaka.

Paul, Soma. 2004. *An HPSG Account of Bangla Compound Verbs with LKB Implementation.* Ph.D. thesis, University of Hyderabad, Hyderabad.

Pollard, C and Sag, I. 1987. *Information-based Syntax and Semantics, Volume 1: Fundamentals.* CSLI Publications, Stanford.

Pollard, C and Sag, I. 1994. *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago.

Sag, I and Wasow, T. 1999. *Syntactic Theory: A Formal Introduction.* CSLI Publications, Stanford.

Saha, G K. 2006. Parsing Bangla Text: An Intelligent Approach. *ACM Ubiquity,* 7(13): - USA.

Selim, M. R. and Iqbal, M. Z.. 1999. Syntax Analysis of Phrases and Different Types of Sentences in Bangla. In *Proceedings of International Conference on Computer and Information Technology,* pages 175-186, Dhaka.

Sengupta, P and Chaudhuri, B. B. 1997. A Delayed Syntactic-Encoding-based LFG Parsing Strategy for an Indian Language – Bangla. *Computational Linguistics,* 23(2): 345-351.

Spencer, A. 2005. Case in Hindi. In *Proceedings of the LFG05 Conference,* Bergen.