Developing a Computational Grammar for Bengali Using the HPSG Formalism

Naira Khan and Mumit Khan
Center for Research on Bangla Language Processing, BRAC University, Dhaka, Bangladesh
naira@bracu.ac.bd, mumit@bracu.ac.bd

Abstract

This paper describes the first phase of developing a computational grammar for Bengali using the Head-Driven Phrase Structure Grammar (HPSG) formalism. The HPSG formalism is a highly developed framework that combines computational and psycholinguistic research to provide a tool with which the features particular to a language can be captured and simultaneously provide information about language universals i.e. linguistic phenomena common to diverse languages. The grammar is implemented on a Linguistic Knowledge Building (LKB) system that has both a syntactic and a semantic level. The system is a powerful tool that allows the user to build a parser along with a generator by using the formalism to code in linguistic rules through feature structures and feature unification. This paper provides a set of instructions for using the formulation of HPSG to parse as well as generate grammatical sentences of Bengali. This paper will enable linguists to interpret and formulate features and types with which Bengali or a structurally similar language can be coded in HPSG. It provides a stepping-stone towards the development of a full computational grammar for Bengali which will also provide useful information as a computational description for its sister languages.

1. Introduction

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) such as machine translation, question-answering systems etc. These involve parsing human sentences that pose considerable problems due to the ambiguous nature of natural languages. Therefore, developing a computational grammar for a natural language can be a complicated endeavor. Grammar development, also known as grammar engineering, is 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. [1] In this paper we attempt to mark the beginnings of such an endeavor for Bengali, using the HPSG formalism and an implementation platform known as LKB. The LKB system is also unique in that it is linguistically motivated, has semantic representation in the form of Minimal Recursive Semantics (MRS), and most importantly, is bi-directional – equipped with both a parser and a generator. [2]

1.1. The Head-driven Phrase Structure Grammar (HPSG) formalism

The Head-driven Phrase Structure Grammar (HPSG) is a non-derivational generative grammar theory developed by [3, 4]. 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. The lexicon is not simple a list of entries, it is richly structured in that individual entries are marked with types forming a hierarchy in themselves. The uniform and modular organization of HPSG makes it attractive for natural language processing. [5]

1.2. Implementation: LKB and the LinGO Grammar Matrix

The Linguistic Grammars Online (LinGO) initiative at Stanford University is an ongoing collaboration that makes available a number of open-source, HPSG related computational resources and includes partners in the US, Europe and Asia. These resources, freely available online, include grammars, lexicons, and the Linguistic Knowledge Based (LKB) Grammar Engineering Platform. [5]

The LKB system is a grammar and lexicon development environment for typed feature structure grammars, used most extensively for HPSG. The LKB source is freely available and implemented in Common Lisp. Typed Feature Structure (TFS) languages are essentially based on one data structure – the typed

feature structure – and one operation –unification – on the typed feature structure. The constraints on the type system provide a way of capturing linguistic generalizations. This powerful combination allows the creation of grammars and lexica used to parse and generate a wide range of natural languages. [1]

LinGo project's English Resource Grammar (ERG) is the largest published grammar developed to date on the LKB to date. [1] The LKB system was also used to develop grammars for other languages such as Japanese and Spanish. This cross-language grammar writing experience resulted in the LinGO Grammar Matrix, available online at the following URL: http://lingo.stanford.edu/.

1.3. Previous work

To our knowledge the development of a grammar of Bengali in a computational framework is still unexplored territory. Although the HPSG formalism itself is not new to Bengali as we find that HPSG and LKB have been used extensively for the implementation of compound verbs of Bengali. [6] Other formalisms such as LFG [7, 8] and CSG [9, 10, 11] have also been used for the parsing of some Bengali sentences. Apart from these, parsing in Bengali has primarily been statistical or based on shallow parsing with little or no semantic representation. [12, 13, 14] In fact grammar engineering is still in its infancy in terms of South Asian languages.

2. Developing a Computational Grammar for Bengali

In the following sections we describe the various linguistic phenomena relevant to a computational description of Bengali and the method in which they were implemented.

2.1. Methodology

2.1.1. 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.

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

Basic Word Order and Phrase Structure Rules: Although Bengali 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 is not grammatical. Although it can be said that Bengali has a pragmatically free word order, nevertheless word ordering is predominantly Subject-Object-Verb (SOV).

E.g.: *ami* *bhat* *kha-i*
I.1.sg rice eat.1.pres-hab
S O V

Hence phrase structure rules in Bengali have been coded accordingly treating it as Head-Final Specifier-Initial with the following rule instances and type definitions from the Matrix.

Agreement and Relations: Bengali is a fusional language and has agreement primarily in terms of Subject-Verb agreement where the verb is marked for person, grade, tense, aspect and modal information. Apart from this, certain adjectives have gender agreement with nouns and number agreement is relevant for certain determiners and nouns.

These agreement issues are dealt with the following type definition built on instances of the Matrix:

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

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. The second and third person can be further subdivided into three grades. 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.

A new subtype TPC is introduced as Bengali nouns take different plural markers based on topicality levels. It has the values of 'high' and 'non-high' which usually correspond to topicality issues of animacy, human, non-human, etc.

Although gender agreement is not prolific as Bengali 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.

Bengali 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: Bengali 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. This is a very simplified view of case-marking in Bengali 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]. And 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.

Types and Lexical Entries: Lexical Entries consist of the following major word classes 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 a noun with or without a 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 and mass nouns. Count nouns are further constrained as singular in number (`NUM sg`) and according to topicality as nouns in Bengali 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.

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 'cat':

```
biral := count-n-lex &
 [ STEM < "biral" >,
   SYNSEM.LKEYS.KEYREL.PRED
"_biral_n_rel" ].
```

Pronouns: Pronouns are types that inherit from the basic noun-lex but are constrained to be determiner-less 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:

```
1sg-pronoun-lex := pronoun-lex &
  [SYNSEM.LOCAL.CONT.HOOK.INDEX.PNG
    [ PER first, NUM sg ] ].
1sg-nom_pron-lex := 1sg-pronoun-lex.
```

Lexical entry for the first person singular pronoun:

```
ami := 1sg-nom_pron-lex &
  [ STEM < "ami" >,
  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.
proximal+dem_q_rel:=
demonstrative_q_rel.
```

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

```
ei := determiner-lex &
 [ STEM < "ei" >,
   SYNSEM.LKEYS.KEYREL.PRED
     "_ei_proximal+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 Bengali 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 that 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 Bengali verbs minimally consist of a root and its inflection where the inflection carries agreement constraints, the lexical entries can be made different 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:

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

Only the root form:

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

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

Adjectives: In order to code in adjectives, it is necessary to have head-modifier rules. In Bengali 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 adjective and its respective PS rules.

The type definition for a simple attributive adjective for Bengali 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 - ]]].
```

In Bengali, 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. 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'.

Adverbs: By default the adverb type definition inherits from matrix supertypes and constrains the modified constituent to be verbal. The adverb then has to be constrained to modify the V, VP or S. In Bengali 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 appropriate:

```
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'.

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.

Lexical Rules: 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, as defined by the Matrix. There are other infl* rule types in the Matrix for various kinds of lexical changes. For the first phase we've used the aforementioned two. 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 morphoph onemic 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. [1]

Nominal Inflection: In the first phase nominal inflection is limited to plural and definiteness markers. In terms of pluralizers, nouns in Bengali 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 high topical nouns. 'ra' on the other hand can only be added to high topical nouns. Hence TPC value constraints (high vs. non-high) prevent inappropriate marking. Some nouns can have both plural markers. In these cases the TPC value has been left undefined.

The type definition for plural markers:

```
;;; for noun + '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 noun + '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) (!a
!agulo)
  plur_noun1-lex-rule.
plur-noun2 := %suffix (* ra) (!a
!ara)
  plur_noun2-lex-rule.
```

Here the letter set is:

```
%(letter-set (!a
abcdefghijklmnopqrstuvwxyz))
```

Definiteness markers can be added concatenatively to any common or count noun. Mass

nouns are constrained through NUM values to prevent them from inflecting.

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

ROOT + [aspect+tense+person-grade] / mode

kor + *ch* + *il* *e - n*

do.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:

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.

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

Here the 'i' of 'dekhi' 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-pastprog-verb := %suffix
  (* chilam) (!ti !tchilam) 1p-
past_verb.
1p-pluperf-verb := %suffix
  (* echilam) (!ti !techilam) 1p-
past_verb.
```

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

```
dekhi > dekhchilam

dekhi > dekhechilam
```

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 '`infl-add-only-no-ccont-ltol-rule`'. This allows us to keep only the root form in the lexicon and generate all else with lexical rules.

3. Results

With the above grammar it is possible to recognize and parse a considerable number of grammatical sentences of Bengali as well as generate various other inflected forms form it.

The result of parsing the sentence 'tini amake bhalo aamti diechilen' (he had given me the nice mango) is shown in Figure 1.

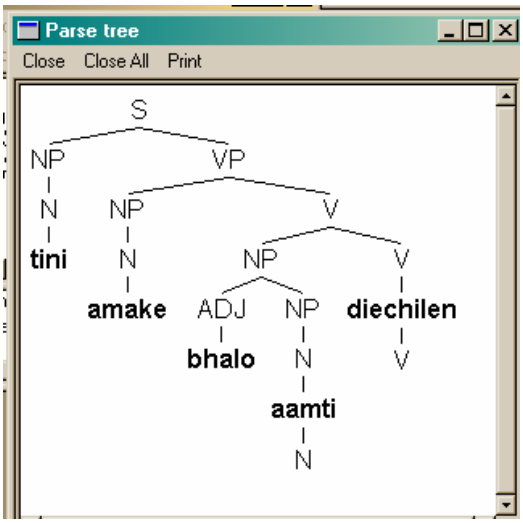

Figure 1: Parse tree for "tini amake bhalo aamti diechilen"

And the forms generated from the above sentence are given in Figure 2.

4. Future Work

The goal of developing a computational grammar for Bengali will inevitably coincide with the goals of any grammar development endeavor. This paper attempts to open up the avenue for a full-scale grammar development with the following long-term goals in mind:

- To write a computational description of Bengali.
- To test various linguistic hypotheses of Bengali using HPSG.
- To make available a computational description of Bengali in order to aid various practical applications in NLP systems.
- To create a resource of computational information for languages of similar grammatical makeup.

- 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.

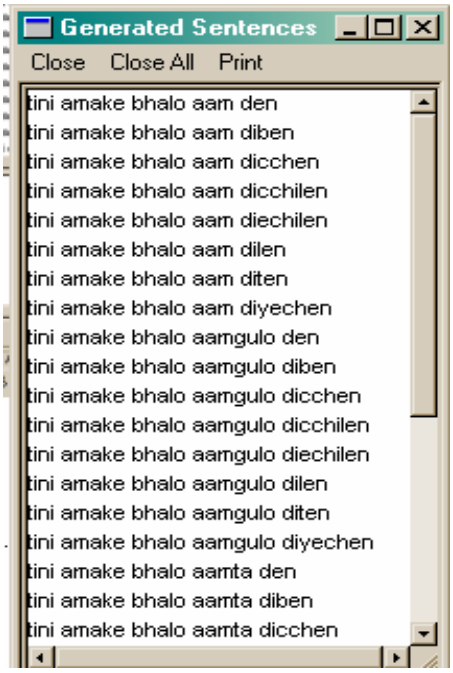

Figure 2: Generated forms for "tini amake bhalo aamti diechilen"

5. Conclusion

This paper serves as the beginning phase of the development of a computational grammar for Bengali. As we have seen the HPSG formalism is a rich linguistic framework suited for the mammoth task of grammar engineering and combined with LKB, provides a suitable platform for the implementation of this formidable task. The various aspects of linguistic phenomena of Bengali have been accounted for and reinterpreted in an implementable framework. As an end result the rudimentary grammar is capable of parsing and generating a large number of sentences of Bengali. It is thus a point of beginning for further development of a large scale Bengali grammar and consequently a computational description that will prove extremely useful both from a computational and a linguistic perspective.

6. Acknowledgement

This work has been supported in part by the PAN Localization Project (www.panl10n.net), grant from the International Development Research Center, Ottawa, Canada, administrated through Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

7. References

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

[2] A. Copestake and D. Flickinger, "An open-source grammar development environment and broad-coverage English grammar using HPSG", In *Proceedings LREC*, Athens, Greece, 2000.

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

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

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

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

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

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

[9] M.M. Hoque and M.M. Ali, "Context-Sensitive Phrase Structure Rule for Structural Representation of Bangla Natural Language Sentences", In *Proceedings of ICCIT,* Dhaka, 2004, pp. 615-620.

[10] M. M. Murshed, "Parsing of Bengali Natural Language Sentences", In *Proceedings of ICCIT,* Dhaka, 1998, pp. 185-189.

[11] M. R. Selim and M. Z. Iqbal, “Syntax Analysis of Phrases and Different Types of Sentences in Bangla”, In *Proceedings of ICCIT,* Dhaka, 1999, pp. 175-186.

[12] M.M. Billah and M.R. Shikder. 2004. Syntax Analysis of Bangla Phrases. In *Proceedings of ICCIT,* Dhaka, pp. 669-673.

[13] G.K. Saha, “Parsing Bengali Text: An Intelligent Approach”, *ACM Ubiquity, 7(13)*, 2006.

[14] S.K. Naskar and S. Bandyopadhyay, A Phrasal EBMT System for Translating English to Bengali. In *Conference Proceedings: the tenth Machine Translation Summit,* Phuket, Thailand, 2005, pp. 372-379.