Building a Foundation of HPSG-based Treebank on Bangla Language

Altaf Mahmud[*], Mumit Khan[†]
[*] CRBLP, BRAC University, Dhaka, Bangladesh, e-mail: *altaf.mahmud@gmail.com*
[†] CRBLP, BRAC University, Dhaka, Bangladesh, e-mail: *mumit@bracuniversity.ac.bd*

***Abstract*—Now a day, the importance of a large annotated corpus for NLP researchers is widely known. In this paper, we describe an initial phase of developing a linguistically annotated corpus for non-configurational 'Bangla' language. Since, the formalism differs from those posited for configurational languages; several features have been added for constraint based parsing through HPSG-based formalism. We propose an outline of a semi-automated process by applying both case marking approach and some morphological analysis to constraint the parsing of a relatively free word order language for creating a linguistically rich, highly-lexicalized annotated corpus.**

***Keywords*—treebank, hpsg, parsing, non-configurational, treebanking**

I. Introduction

A large and syntactically annotated corpus has now become an important research tool for investigators in Natural Language Processing to develop stochastic model to automatically process natural languages. But, free word order languages always have long posed significant problems for standard parsing algorithms to build such a syntactically annotated corpus. Since, Bangla is a relatively free word order language; developing standard parsing algorithms can be a complicated endeavor. So far, a considerable amount of work has been done for developing a computational grammar for Bangla using the Head-driven Phrase Structure Grammar (HPSG) formalism [1], but it assumed that the word ordering is predominantly Subject-Object-Verb (SOV). Although, word-order is not completely free in that all possible arrangements of words within a sentence is not grammatical in Bangla, incorporating only those sentences into a corpus, which match that fixed word order pattern is inadequate, because it would be a lack of expressiveness of the built corpus for pragmatically free word order language by discarding variable or flexible grammatical phenomena. Therefore, we have been motivated to include all possible variations of word-order within a sentence, and (un)grammatical syntactical constructions will be accepted / rejected with manual interpretations. This is a semi-automated process. Now, considering the background of the 'Bangla' language, not much work has been carried out in the front of automatic processing of the language. The main obstacle, of course, is being unavailability of a large annotated corpus to experiment statistical algorithms. The work reported in this paper aims at providing such a syntactically annotated corpus ('treebank') using HPSG for Bangla language. In particular, we focus on two syntactical issues for case-based parsing of a non-configurational language: case and morphological analysis. Our implementation platform is LKB written in emacs lisp, which has semantic representation in the form of Minimal Recursion Semantics (MRS), and is bi-directional – equipped with both a parser and a generator [2] (for more information: www.delph-in.net/lkb).

A. Why HPSG Treebank?

It has been observed by the recent deep linguistic investigations in the existing hand-built treebank that, the depth of linguistic information in any current hand-built syntactical corpus is comparatively shallow and these static treebanks tend to fall behind the theoretical advances in formal linguistics and grammatical representations, because the design and format of linguistic representation of in the treebank hard-wires a small, predefined range of ways in which information can be extracted [3]. On the other hand, HPSG treebanks are both rich and dynamic, and the ways of retrieving linguistic data from the treebank in varying granularity. Therefore, flows of deep linguistic information make it more committed to acquire preferred syntactical constructions of any non-configurational grammatical phenomena.

II. Related Work Done

HPSG treebank was first initiated for English by LinGO Redwoods HPSG Treebank [3]. For other free word order languages such as: German, Bulgarian and Polish [4, 5, 6], HPSG based treebanks have already been developed with a competitive performance compare to other hand-parsed treebanks. Among them, the annotation scheme of 'BulTreeBank' project is precisely published [7]. They organized the language data into several levels: **Text Corpus, Treebank** and **Core Set of Sentences.** Briefly, the parsing process is split down into two major steps: **partial parsing** and **HPSG step.** During partial parsing, sentences are extracted from grammar books and corpus, each sentence is morphosyntactically tagged by assigning all possible tags to each word using a morphological analyzer, then part of speech is disambiguated by predicting most probable part of speech. Then names, numerical expressions, dates, special tokens are recognized. And finally recursive constituents are identified by chunk parsing. At next step, this output from the pre-processing step is encoded into HPSG compatible representations and is encoded as feature graphs. Another one is **resolution step**, which is basically in order to minimize the incoming possibilities from the HPSG step.

III. Methodology

Starter Grammar: The Lingo Grammar Matrix (available at: www.delph-in.net/matrix/) provides a starter grammar instead of writing from the scratch, with bare minimum lexical entries and some rudimentary rule types by customizing certain cross-linguistically common typological properties for the target language. The starter grammar consists of 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 grammar for Bangla is implemented on LKB system by developing on the matrix starter grammar. We are modifying or enhancing some linguistic phenomena of the previous skeleton version [1] to make it consistent with the target corpus.

Basic Word Order: Since, Bangla is a non-configurational language, we permit the word-order to be completely free. Thus, the starter grammar provides a series of phrase structure rules that allow major sentence constituents (Subject, Verb, First Complement, and Second Complement) to be realized in any order, without any spurious extra parses.

Coordination strategy: The coordination strategy has been selected as ‘Monosyndeton’ that supports coordination structure like: “A, B and C”, which also permits “A and B and C”. Co-ordinands are typically nouns, noun phrases (NPs), verbs and verb phrases (VPs).

Agreement and Relations: The previous formalism [1] has proposed an extensive design regarding this issue. Built on instances on the matrix, following type definitions for agreements have already been defined there as noun’s features: png :+ [PER person, NUM number, GRD grade, TPC topical, GEN gender].

Case Relations: For any pragmatically free word order Indian language like Bangla, case relations are syntactico-semantic or semantico-syntactic relations between the verbals and related constituents [8]. They by themselves do not give the semantics. Instead they specify relations, which mediate between inflectors of nominals and verb form on one hand and semantic relations on the other. But, in most of the cases, the problem arises when case-inflections are appeared to be ambiguous. Two examples of case ambiguities are: “shikshhak chhaatraderke parhacchhen” and “shikshhake chaatrader parhacchhen” (Teacher is teaching students). At the above two examples, the subject ‘shikshhak’ (teacher) appeared with both nominal (0 case marker) and locative (case marker ‘e’). Similarly, the object ‘chaatra’ (student) can take two case markers accusative (case marker ‘derke’) and genitive-plural (case marker ‘der’). In order to disambiguate the related constituents of a verb, the simple straight-forward approach would be to allow a particular constituent (subject or object) to take a group of case-markers instead of a particular one, where different case-markers can morphologically assign different cases for a single constituent, but still we will be able to identify its role. So far, for the beginning of the first phase of this treebank project, we resolve this ambiguity problem using the type hierarchy system of LKB as following Fig. 1:

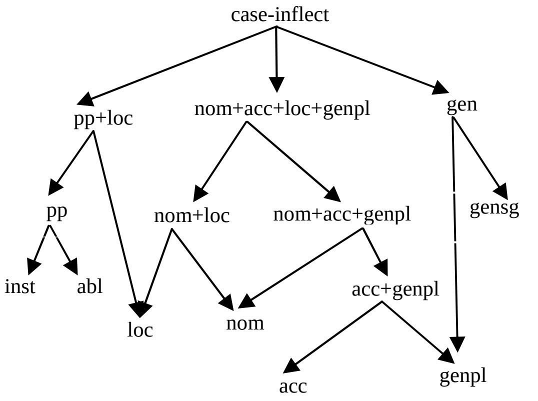


Fig. 1. Proposed case markers’ hierarchy for ‘Bangla’.

Postposition and locative case markers are to be constrained through the head-modifier rules and placed under the sub-hierarchy ‘pp+loc’, where the sub-tree under ‘nom+acc+loc+genpl’ is to constraint the arguments of the verb types. The case-marker singular genitive (gensg) is only used for possessive case, where genitive-plural (genpl) is used for both possessive and accusative case. The ‘genpl’ case marker is inherited from ‘acc+genpl’ and ‘gen’, therefore, it will be subsumed by both of its parent types. Case information is included to valance features of all kind of verb types. Inflected nouns and pronouns go through some morphological rules to exhibit their case values.

Word classes and Lexical Entries: Lexical entries are written in lexicon.tdl file and their respective word classes are encoded into bangla.tdl file:

Nouns: To implement lexical or morphological rules in LKB appropriately, nouns are constrained to be categorized into ‘singular’ and ‘non-singular’ nouns at first level. Singular nouns are sub-divided into pronoun, proper noun and count noun. Non-singular nouns are divided into ‘nominal inflected noun’ and ‘locative inflected noun’. Nominal inflected nouns are further sub-divided into ‘locative common noun’ and ‘mass noun’, which have case-inflect to be nominal. The other sub-type of non singular noun ‘locative inflected noun’ are constrained to be locative case-inflect value and has only one sub-type ‘locative proper noun’, which has topicality value ‘high’ (animated). The person, gender, topicality and grade information are coded into lexical entries, where number and case-inflect values are kept open or ‘underspecified’ for lexical rules. An example of word class:

```
count-noun-lex := sg-nouns & single-rel-lex-item.
```

An example of a lexical entry ‘biraal’ (cat):

```
biraal := count-noun-lex &
[ STEM < "biraal" >,
  SYNSEM [ LOCAL.CONT.HOOK.INDEX.PNG [
           PER third,TPC medium,GRD non-hon ],
                     LKEYS.KEYREL.PRED "_cat_n_rel" ]].
```

Pronouns: Pronouns are constrained to have no determiner and further constrained into general pronoun or personal pronoun and demonstrative pronoun. Demonstrative pronouns introduce two relations: ‘proximal+dem_q_rel’ (closer to speaker) and ‘distal+dem_q_rel’ (away from speaker). These two relations are two sub-types of ‘demonstrative_q_rel’ in bangla.tdl file. An example of demonstrative pronoun’s type definition:

```
demo-pronoun-lex := pronoun-lex &
[ SYNSEM [ LOCAL.CONT.HOOK.INDEX.COG-ST
activ+fam,
        LKEYS [KEYREL noun-relation & [LBL #lbl ],
          ALTKEYREL event-relation & [LBL #lbl ]]]].
```

A lexical entry for demonstrative pronoun:

```
ei_demo := demo-pronoun-lex &
[ STEM < "ei" >,
  SYNSEM [ LOCAL.CONT.HOOK.INDEX.PNG [
                     PER third, GRD non-hon, TPC low ],
   LKEYS.ALTKEYREL.PRED proximal+dem_q_rel ]].
```

Verbs: The starter grammar provides intransitive and transitive verb types inherited from a basic verb type. Another type, di-transitive verb is also created. The transitive verb types are further categorized according to argument optionality: definite and indefinite null instantiation. Usually, all transitive verbs in Bangla allow arguments to be unspecified. The transitive verb types with definite null instantiation will allow the pro-dropped subject or object to be recoverable from the context, but the indefinite null instantiated transitive verbs will not. We instantiate two rules in ‘rules.tdl’ file: ‘basic-head-opt-subj’ when complement is missing, and ‘decl-head-opt-subj’ when subject is missing. In this case, the parser will produce extra parse trees which are licensed by the written phrase structure rules. The human annotators can decide whether a verb has left its related constituents or not depending on the context. Case relations are incorporated from the type hierarchy of ‘case-inflect’ type to all nominal arguments of all types of verb. Below some examples of verb types:

```
intransitive-verb-lex := verb-lex & intransitive-lex-item & [
SYNSEM.LOCAL.CAT.VAL.COMPS < >,
      ARG-ST < [ LOCAL.CAT.HEAD noun & [
                    CASE-INFLECT nom+loc ]] > ].
```

```
transitive-verb-lex := verb-lex & transitive-lex-item &
[ SYNSEM.LOCAL.CAT.VAL.COMPS  <#comps>,
  ARG-ST < [ LOCAL.CAT.HEAD noun &
                                  [ CASE-INFLECT nom+loc ]],
              #comps & [ LOCAL.CAT [ VAL [ SPR < >,
                                           COMPS < > ],
                        HEAD+np & [CASE-INFLECT
                             nom+acc+genpl ]]] > ].
```

```
ditransitive-verb-lex := verb-lex & ditransitive-lex-item &
[ SYNSEM.LOCAL.CAT.VAL.COMPS < #comps1,
                                          #comps2 >,
ARG-ST < [ LOCAL.CAT.HEAD noun &
                                [ CASE-INFLECT nom+loc ]],
        #comps1 & [ LOCAL [ CAT [ HEAD noun, VAL [
                                SPR < >, COMPS < > ],
               HEAD +np & [ CASE-INFLECT acc+genpl ]],
           CONT.HOOK.INDEX.PNG.TPC high+medium ]],
        #comps2 & [ LOCAL [ CAT [ HEAD noun, VAL [
                                      SPR < >, COMPS < > ],
                     HEAD +np & [ CASE-INFLECT nom ]],
        CONT.HOOK.INDEX.PNG.TPC medium+low ]] > ].
```

```
definite-pro-drop-trans-verb-lex := transitive-verb-lex &
[SYNSEM.LOCAL.CAT.VAL.COMPS.FIRST.OPT-CS
                                          activ-or-more].
```

```
indefinite-pro-drop-trans-verb-lex := transitive-verb-lex.
```

The di-transitive verb type can also be constrained according to the argument optionality. Lexical entries for verbs in ‘lexicon.tdl’ file contain only root forms:

```
khaa := indefinite-pro-drop-trans-verb-lex &
[ STEM < "khaa" >,
SYNSEM [ LOCAL.CAT.VAL.SUBJ <
                 [LOCAL.CONT.HOOK.INDEX.PNG.TPC
           [high+medium ] >,
            LKEYS.KEYREL.PRED "_eat_v_rel" ]].
```

Adjectives: Because we taking free word order into account, we instantiated both ‘head-adj-int-phrase’ and ‘adj-head-int-phrase’ to allow adjectives appear as both pre-head and post-head modifiers. For now, we are only caring about intersective adjectives, scopal or predicative adjective types have not been implemented yet.

Lexical Rules: In starter grammar, lexical rules are cross-classified into two dimensions: inflecting v. constant and ltol v. ltow. A rule is inflecting if it is associated with spelling change rule (i.e. the rule instance has %prefix and %suffix line), should be instantiated in ‘irules.tdl’ file. A rule is constant if the mother and the daughter have the same form (i.e. the rule doesn’t have %prefix or %suffix line), and is instantiated in ‘lrules.tdl’ file. On the other hand, the ltol/ltow distinction has to do with when a word is morphologically completed and thus eligible to be a daughter in a phrase structure rule. The feature, which is used to track this, is called INFLECTED. Signs that are INFLECTED +, are considered fully inflected. As mentioned earlier, two features of noun types are left underspecified: case and number. For the basic noun forms, we still need morphological rules to assign specific number and case values to root forms and make it to be morphologically completed. The daughter is constrained to be INFLECTED -, to make it incompatible with its mother and to prevent a rule to apply further to its own output. An example for type definition of a constant rule:

```
nom-sg-nouns-lex-rule := const-ltow-rule &
[ SYNSEM.LOCAL [ CAT.HEAD noun & [
                                CASE-INFLECT nom ],
             CONT.HOOK.INDEX.PNG.NUM sg ],
  DTR sg-nouns &
     [INFLECTED -,  SYNSEM.LOCAL.CAT.HEAD noun ]].
```

Nominal Inflection: Inflections on nouns change its two features: case and number. In addition to the previous work [1], each inflection rules are considered against each noun classes separately, because different noun classes can be inflected with different markers for same feature. Nouns are inflected with determinative makers in Bangla, but locative noun, mass noun and personal pronouns are restricted. Mass nouns and locative nouns are constrained to be 'non-sg' to take any further plural markers. Example of a type definition of nominal inflections:

```
;;; plural marker 'raa'
high-tpc-pl-nom-noun-lex-rule := infl-ltow-rule &
[ SYNSEM.LOCAL [ CAT.HEAD noun &
                               [ CASE-INFLECT nom ],
            CONT.HOOK.INDEX.PNG.NUM pl ],
  DTR sg-nouns &
     [ INFLECTED -,
      SYNSEM.LOCAL [ CAT.HEAD noun,
         CONT.HOOK.INDEX.PNG.TPC high ]]].
```

And inflection rule is:

```
high-tpc-pl-inflect-noun :=
%suffix (* raa) (* eraa) (aami aamraa) (tumi tomraa) (tui
toraa) (se taaraa) (aapni aapnaaraa) (tini tnaaraa) (uni
unaaraa)
high-tpc-pl-nom-noun-lex-rule.
```

Verbal Inflection: Like nouns, verbs are placed at root form into lexicon.tdl. The previous computational grammar [1] describes that, root forms are constrained to have 2nd person pejorative nominal subject, and in all other cases verbs are inflected with person, tense, aspect and modal information. In case of subject-verb agreement in Bangla, verb agrees with nominal subject only with person variation. Therefore, verbal inflection types are sub-categorized for person, grade and finiteness or non-finiteness. As verb lexicons are coded at root form, a constant rule is needed to pump a root verb into morphologically completed and is compatible with 2nd person pejorative nominal subject. Verbs are constrained to have compatible nominal subject through the valence feature SUBJ < >. An inflected 3rd person verb type definition:

```
3p-verb-lex-rule := infl-ltow-rule &
[SYNSEM.LOCAL.CAT[HEAD verb & [ FORM fin ],
        VAL.SUBJ < #subj &
        [LOCAL.CONT.HOOK.INDEX.PNG [
                PER third, GRD non-hon ]] > ],
  DTR [ INFLECTED -,
        SYNSEM.LOCAL.CAT [ HEAD verb,
                     VAL.SUBJ < #subj > ]]].
```

And the inflection rule is:

```
3p-verb :=
%suffix (* e) (* chhe) (* chhila) (* chhilo) (* echhe) (*
echhila) (* be) (a ay) (a acchhe) (aa eyechhe) (jaa
giechhe) (aa eyechhila) (aa eyechhilo) (jaa giechhila) (jaa
giechhilo) (e ey) (e icchhe) (e icchhila) (e icchhilo) (e
iechhe) (e iechhila) (e iechhilo) (e ibe)
3p-verb-lex-rule.
```

The above type definition for 3rd person verb and inflection rules would accept and can produce following 3rd person verb forms from the root verb 'khaa': khaay, khaacchhe, khaabe, kheyechhe, kheyechhila, kheyechhilo. Although, still this inflection rule will also accept and generate some unacceptable verb forms, such as 'khaae'. While we are building the basic foundation, we are not yet presenting any full analysis for any module or part of it. Our next further review will fix up these pitfalls.

IV. Results and Discussions

The grammar described here, is able to produce syntactical constructions of considerable number of Bangla sentences. The rudimentary grammatical features for lexical information we have just discussed, and the initial level of syntactico-semantic analysis regarding HPSG formalism would show up a large number of possibilities of syntactic trees for a given sentence. But, the rudimentary grammatical nature of the rules applied would help an annotator to select the preferred syntactical tree through a large parse forest. For an example, while the parser currently produces more than one syntactical construction for a sentence like this:

shikshhak aamaader balechhilen byaakaran baitaa bhaalavaabe parhte
"Teacher us told grammar book-the attentively read-to"
(Teacher told us to read the grammar book attentively)

Annotators can quickly navigate through the parse forest and identify the correct or preferred analysis in the current context (or, in rare cases reject all analysis proposed by the grammar) using LKB's treebanking option [9]. This tree selection tool shown at Fig. 2, presents users, who need little expert knowledge of the underlying grammar, with a range of basic properties that distinguish competing analyses and that are relatively easy to judge [3].

V. Future Work

We are currently reviewing for further syntactical and morphological analysis to minimize the number of ambiguous parses for an input sentence, and to increase the flow of linguistic information through the syntactical constructions of the treebank. The immediate next step should be defining a morphosyntactic tag set and then collecting core set of sentences from Bangla grammar books, and from newspapers. Our goal is to evaluate performances of any statistical Bangla parser trained on the corpora.

VI. Conclusion

The basic foundation presented here is designed not only for building the corpus, but will serve as a guideline for the annotators. We expect that the preliminary design of this platform will be suitable for complicated and massive amount of work to develop a large and linguistically rich annotated corpora for any pragmatically free word order language.

REFERENCES

[1] N. Khan and M. Khan, “Developing a Computational Grammar for Bengali Using the HPSG Formalism,” in *Proceedings of the 9th International Conference on Computer and information Technology (ICCIT ’06)*, Dhaka, Bangladesh, 2006.

[2] A. Copestake and D. Flickinger, “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, 2000.

[3] S. Oepen, K. Toutanava, S. Sheiber, C. Manning, D. Flickinger, and T. Brants, “The LinGO Redwoods Treebank. Motivation and Preliminary Applications,” in *Proceedings of the 19thInternational Conference on Computational Linguistics.* Taipei, Taiwan, 2002.

[4] G. Neumann and B. Crysmann, “Exploring HPSG-based Treebanks for Probabilistic Parsing,” in *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2006),* Genoa, Italy, 2006.

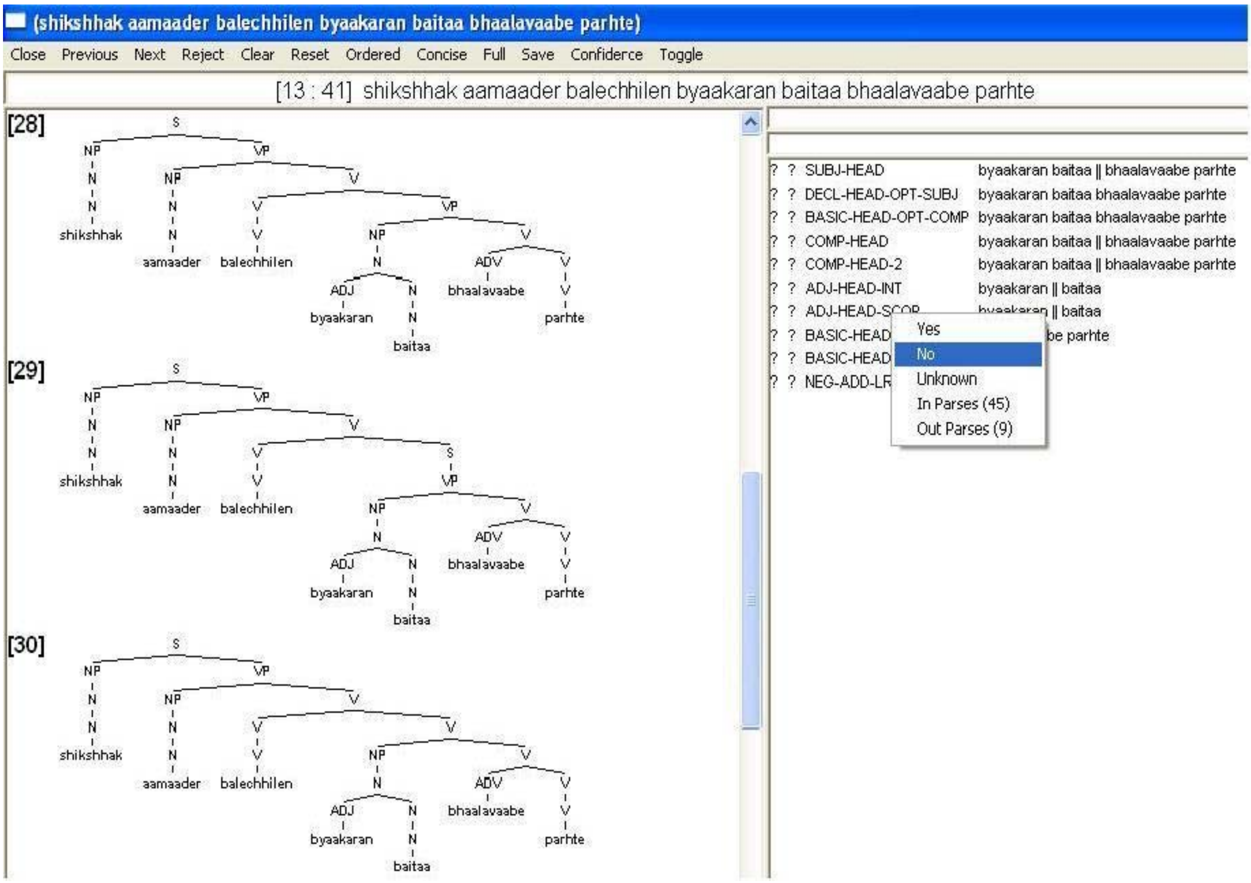
[5] K. Simov, G. Popova and P. Osenova, “HPSG-based Syntactic Treebank of Bulgarian (BulTreeBank),” in *A Raindow of Corpora: Corpus Linguistics and the Languages of the World,* edited by A. Wilson, P. Rayson, and T. McEnery. Lincom-Europa, Munich, Germany, 2002.

[6] M. Marciniak, A. Mykowiecka, K. Anna and A. Przepiórkowski, “An HPSG-Annotated Test Suite for Polish,” in *Anne Abeillé (editor), Treebanks. Building and Using Parsed Corpora,* Kluwer Academic Publishers, 2003, pp 129-146.

[7] K. Simov and P. Osenova, “Practical Annotation Scheme for an HPSG Treebank of Bulgarian,” in *Proceedings of the 4th Workshop on Linguistically Interpreted Corpora,* Budapest, Hungary, 2003.

[8] A. Bharati and R. Sangal, “Parsing Free Word Order Languages in the Paninian Framework,” in *31st annual meeting on Association for Computational Linguistics (ACL).* Ohio, U.S.A., 1993.

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



[13 : 41] shikshhak aamaader balechhilen byaakaran baitaa bhaalavaabe parhte

```
? ? SUBJ-HEAD              byaakaran baitaa || bhaalavaabe parhte
? ? DECL-HEAD-OPT-SUBJ     byaakaran baitaa bhaalavaabe parhte
? ? BASIC-HEAD-OPT-COMP    byaakaran baitaa bhaalavaabe parhte
? ? COMP-HEAD              byaakaran baitaa || bhaalavaabe parhte
? ? COMP-HEAD-2            byaakaran baitaa || bhaalavaabe parhte
? ? ADJ-HEAD-INT           byaakaran || baitaa
? ? ADJ-HEAD-SCOP          byaakaran || baitaa
? ? BASIC-HEAD                        be parhte
? ? BASIC-HEAD
? ? NEG-ADD-LR
```

Fig. 2. A screen dump of LKB treebanking tool.