Proceedings of 13th International Conference on Computer and Information Technology (ICCIT 2010)
23-25 December, 2010, Dhaka, Bangladesh

A Context Free Grammar and its Predictive Parser for Bangla Grammar Recognition

K. M. Azharul Hasan, Amit Mondal, Amit Saha
Department of Computer Science and Engineering (CSE)
Khulna University of Engineering and Technology (KUET)
Khulna-9203, Bangladesh
azhasan@cse.kuet.ac.bd, amit.kuet@gmail.com, amit_kuet@yahoo.com

Abstract
Parsing is a process of transforming natural language into an internal system representation, which can be trees, dependency graphs, frames or some other structural representations. If a natural language be successfully parsed then grammar checking from this language becomes easy. In this paper we describe a context free grammar for Bangla language and hence we develop a Bangla parser based on the grammar. Our approach is very much general to apply in Bangla Sentences and the method is well accepted for parsing a language of a grammar. The scheme is based on Top down parsing method and to avoid the left recursion the idea of left factoring is adopted.

Keywords: Context Free Grammar, Predictive Parser, Bangla Language processing, Parse Table, Top down and Bottom up Parser.

I. INTRODUCTION

Parsing is a fundamental problem in language processing for both machines and humans. In general, the parsing problem includes the definition of an algorithm to map any input sentence to its associated syntactic tree structure[1]. A parser analyses the sequence of symbols presented to it based on the grammar[2]. Natural language applications namely Information Extraction, Machine Translation, and Speech Recognition, need to have an accurate parser[3]. Parsing natural language text is more difficult than the computer languages such as compiler and word processor because the grammars for natural languages are complex, ambiguous and infinity number of vocabulary. For a syntax based grammar checking the sentence is completely parsed to check the correctness of it. If the syntactic parsing fails, the text is considered incorrect. On the other hand, for statistics based approach, Parts Of Speech (POS) tag sequences are prepared from an annotated corpus, and hence the frequency and the probability[4]. The text is considered correct if the POS-tagged text contains POS sequences with frequencies higher than some threshold[5].
Although Bangla is the fourth largest language of the world having over 200 million native speakers but still now Bangla language does not have a full fazed computerized grammar checker for a given Bangla sentence. In this paper, we proposed a context free grammar for the Bangla language and hence we proposed a predictive Bangla parser constructing a parse table. We have adopted the top down parsing scheme and avoided the problem of left recursion using left factoring[6] for the proposed grammar. We implemented the corresponding Bangla dictionary in XML format using the corresponding word as tag name and it's POS as value. It helps to search the dictionary very fast. We believe the proposed grammar and parser can be applicable to any types of Bangla sentences and can be used for grammar checker.

II. RELTED WORKS

A rule based Bangla parser have proposed in [1] that handles semantics as well as POS identification from Bangla sentences and ease the task of handling semantic issues in machine translation. An open-source morphological analyzer for Bangla Language using finite state technology is described in [7]. They developed the monolingual dictionary called monodix and stored in XML file. The work [5] addresses a method to analyse syntactically Bangla sentence using context-sensitive grammar and interpret the input Bangla sentence to English using the NLP conversion unit. The system is based on analyzing an input sentence and converting into a structural representation. A parsing methodology for Bangla natural language sentences is proposed in [8] and show how phrase structure rules can be implemented by top-down and bottom-up parsing approach to parse simple sentences of Bangla. A comprehensive approach for Bangla syntax analysis was developed [9] where a formal language is defined as a set of strings. Each string is a concatenation of terminal symbols. Some other approaches such as Lexical Functional Grammar (LFG) [4] and Context Sensitive Grammar (CSG) [10]–[12] have also been developed for parsing Bangla sentences. Some developers devoloped Bangla parser using SQL to check the correctness of sentence; but its space complexity is inefficient. Besides it take more time for executing SQL command.As a result those Parser becomes slower.
In this paper we store the Bangla word as tag and its corresponding constituents or POS are used as value. Hence to the POS or constituents of a word it will be speedy to get the result. More over we have applied the idea of left facoring to remove the left recursion and ambiguity of the grammar and hence according to our proposed grammar, the parser can

detect the errors in a Bangla sentence.All the schemes mentioned in this section does not create the parse table to parse a sentence. Hence it would be difficult to check the errors in a given sentence if it is not a string of the grammar. But in our approach, the errors can be detected by by means of the empty entry in the parse table.

III. A Parsing Scheme for Bangla Grammar Recognition

A. Storing Words in XML

The Extensible Markup Language (XML) is now really used almost everywhere. It is a good format to store any kind of data. The tasks behind using XML are always the same: reading data from XML and writing data into it. The general format of XML tag is: <tag_name>value</tag_name>. To design the dictionary in XML we consider *Bangla word* as tag_name and *value* as its corresponding POS. An example of XML file is shown in Figure 1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpyΠ-->
<WORD>
                    < আমি >pronoun</আমি>
                    <খাই>verb</খাই>
                <একটি>modifier</একটি>
                <এবং>conjunction</এবং>
                    <আমরা>pronoun</আমরা>
                    <না>neg</না>
</WORD>
```

Figure 1: Data Format of XML File

B. Developing the Tag Set for Bangla Grammar using XML

Each sentence is composed of one or more phrases. So if we can identify the syntactic constituents of sentences, it will be easier for us to obtain the structural representation of the sentence [8,9]. We tagged Bangla words with their respective parts-of-speech(POS), modifier, pattern, number [3] and stored in XML file. Table 1 shows the tag set used in XML file for storing Bangla words. Each sentence is partitioned into its constituent. A constituent expresses the complete meaning for a specific context. The constituents of the sentences found out are shown in the table 2.

C. Bangla Grammar Design

Once constituents have been identified, the productions for Context Free Grammar (CFG) are developed for Bangla sentence structures. Figure 2 shows the production rules for the proposed CFG. In the following we call the CFG as only G.

Table 1: Tag set Description for Bangla grammar

Tag Name (Symbol)	Examples
Noun (noun)	রহিম, বিদ্যালয়, ছেলে, বই,ভাল, টাকা, ঘাস।
pronoun(pronoun)	আমি, আম্রা,তুমি , সে, তারা।
Adjective (adjective)	ভাল, বলবান, স্মুবান , একটু, দশ, অনেক, শখ, সর্বদা, অনেক, দদ্রুত।
verb(verb)	খাই, খেলে, করবে, পড়া, চলে, যাই।
Conjunction(conjunction)	এবং, ও, চেয়ে।
Negative Description(neg)	না, নয়।
Modifier(modifier)	এ, একটি, একদিন, এই।

Table 2: Constituents for developing the grammar

Constituents (Symbol)	Productions	Examples
Noun Phrase (NP)	NP -> noun NP -> pronoun NP ->modifier noun etc.	রহিম আমি এ পথে।
Verb Phrase(VP)	VP -> noun verb VP -> noun verb verb VP -> noun verb ptrn etc.	বই পড়ছে খাওয়া হল না।
Adjective Phrase(AP)	AP -> adjective AP -> adjective noun etc .	দশ ভাল ছেলে।

Left Factoring

The grammar shown in Figure 2 is ambiguous. The parser generated from this kind of grammar is not efficient as it requires backtracking. To remove the ambiguity from the grammar we have used the idea of left factoring and reconstruct the grammar productions. Left factoring is a grammar transformation useful for producing a grammar suitable for predictive parsing. The basic idea is that when it is not clear which of the productions are to use to expand a non terminal then it can defer to take decision until we get an input to expand it. In general if we have productions of form

$$A \rightarrow \alpha\beta_1 | \alpha\beta_2$$

We left factored it by getting the input α and break it as follows

$$A \rightarrow \alpha A'$$

$$A' \rightarrow \beta_1 | \beta_2$$

The above grammar is correct and is free from conflicts. After left factoring, the reconstructed grammar productions of Figure 2 are shown in Figure 3.

D. Parser Design

The idea of predictive parser design is well understood in a compiler design[11]. To construct a predictive parser for grammar G two functions namely FIRST() and FOLLOW() are important. These functions allow the entries of a predictive parse table for G. Once the parse table has been constructed we can verify any string whether it satisfy the grammar G or not. The FIRST() and FOLLOW() determines the entries in the parse table. Any other entries in the parse table are error entries.

S→NP VP
NP -> noun | pronoun | modifier noun | AP | pronoun conjunction pronoun | noun conjunction noun.
AP -> adjective noun | adjective AP | adjective adjective ptrn | adjective | adjective pronoun | adjective conjunction AP.
VP -> noun verb | noun verb verb | noun verb ptrn | noun adjective noun |
verb verb | verb verb ptrn | AP noun | AP verb | AP | AP adjective verb |
AP adjective | adjective verb | pronoun AP verb.

Figure 2: The proposed Bangla grammar G

S -> NP VP
NP -> noun NP1 | pronoun NP2 | modifier noun | AP
NP1 -> conjunction noun | ε
NP2 -> conjunction pronoun | ε
AP -> adjective AP1
AP1 -> noun | adjective AP2 | pronoun | conjunction AP | ε
AP2 -> ptrn | ε
VP -> noun VP1 | verb verb VP2 | AP VP3 | pronoun AP verb
VP1 -> verb VP4 | adjective noun
VP2 -> ptrn | ε
VP3 -> noun | verb | adjective VP5 | ε
VP4 -> verb | ptrn | ε
VP5 -> verb | adjective verb

Figure 3: Grammar after left factoring

Computing FIRST of variable

FIRST(α) be the set of terminals that begin the strings derived from α. If α → ε then ε is also included in FIRST(α). According to the rules[6] of computing FIRST(), the values are computed for the grammar G and are shown in Figure 4.

Computing FOLLOW of a Non terminal

FOLLOW(A) of a non terminal A is the set of terminals *a* that can appear immediately to the right of A. If A is the right most symbol in the sentential form then $ is added to FOLLW(A). According to the rules[6] of computing FOLLOW, the values are computed for the grammar G and are shown in Figure 5.

FIRST(S) ={noun, pronoun, modifier, adjective}
FIRST(NP) = {noun, pronoun, modifier, adjective}
FIRST(NP1) ={conjunction, ε }
FIRST(NP2) = {conjunction, ε }
FIRST(AP) ={adjective}
FIRST(AP1) = {noun, adjective, pronoun, conjunction, ε }
FIRST(AP2) = {ptrn, ε }
FIRST(VP) = {noun, verb, adjective, pronoun}
FIRST(VP1) = {verb, adjective}
FIRST(VP2) = {ptrn, ε }
FIRST(VP3) = {noun, verb, adjective, ε }
FIRST(VP4)={verb, ptrn, ε }
FIRST(VP5)={verb, adjective}

Figure 4: Computing FIRST for the grammar G

FOLLOW(S) = {$}
FOLLOW(NP) ={noun,verb,adjective,pronoun}
FOLLOW(NP1) = {noun,verb,adjective,pronoun}
FOLLOW(NP2) = {noun,verb,adjective,pronoun}
FOLLOW(AP) = {noun,verb,adjective,pronoun,$}
FOLLOW(AP1) = {noun,verb,adjective,pronoun,$}
FOLLOW(AP2) = {noun,verb,adjective,pronoun,$}
FOLLOW(VP) ={$}
FOLLOW(VP1) ={$}
FOLLOW(VP2) ={$}
FOLLOW(VP3) ={$}
FOLLOW(VP4) ={$}
FOLLOW(VP5) ={$}

Figure 5: Computing FOLLOW for the grammar G

Parse Table Construction

Let M[m, n] be a matrix where m is the number of non terminals in grammar G and n is the number of distinct input symbols that may occur in a sentence of grammar G. Table 3 shows the resulting parse table for the grammar G constructed by applying the following rules [6]

1. for each production of the form A → α of the grammar G
   for each terminal *a* in first(α), add A → α to M[A,a].
2. If ε is in first(α) then
   for each terminal in FOLLOW(A) add A → α to M[A,a]
3. All other undefined entries of the parsing table are error entries.

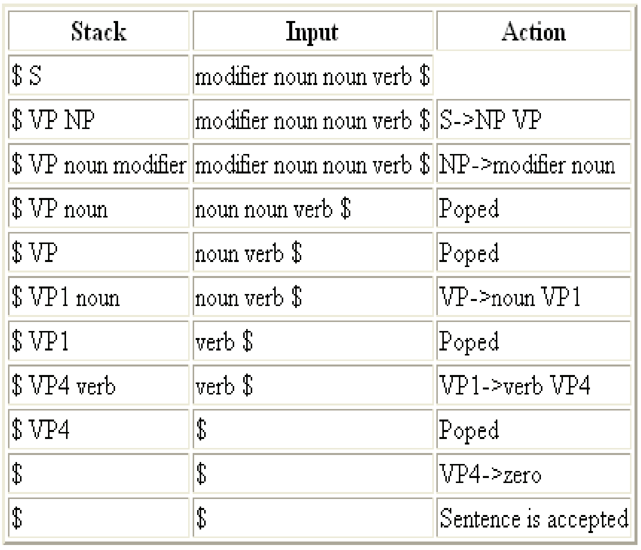
Table 3: Predictive Parsing Table for the developed Bangla grammar G

Non terminal	INPUT SYMBOL							
	Noun	Pronoun	Modifier	Adjective	Conjunction	Ptrn	Verb	$
S	S->NPVP	S->NPVP	S->NPVP	S->NPVP				
NP	NP->noun NP1	NP-> pronoun NP2	NP-> modifier noun	NP->AP				
NP1	NP1 -> ε	NP1 -> ε		NP1 -> ε	NP1 -> conjunction noun		NP1 ->ε	
NP2	NP2 -> ε	NP2 -> ε		NP2-> ε	NP2 - > conjunction pronoun		NP2-> ε	
AP				AP ->adjective AP1				
AP1	AP1-> noun	AP1-> pronoun		AP1->adjective AP2	AP1 -> conjunction AP			ε
AP2	AP2->ε	AP2->ε		AP2->ε		AP2->ptrn	AP2->ε	ε
VP	VP -> noun VP1	VP -> pronoun AP verb		VP -> AP VP3			VP-> verb verb VP2	
VP1				VP1 ->adjective noun			VP1 -> verb VP4	
VP2						VP2 ->ptrn		ε
VP3	VP3-> noun			VP3 ->adjective VP5			VP3-> Verb	ε
VP4						VP4->ptrn	VP4 -> Verb	ε
VP5				VP5 ->adjective verb			VP5 -> Verb	

Parse Tree Generation

We store the parse table M using a two-dimensional array. To read an element from a two-dimensional array, we must identify the subscript of the corresponding *row* and then identify the subscript of the corresponding *column*. For example, the production "*NP→ modifier noun*" is in row 2, column 3, (see table 3) so it is identified as M[2][3].

Let us explain the grammar for the sentence "একটি ছেলে বই পড়ছে"। Using the XML data file we get the tags "*modifier noun noun verb*". Using the production *S→ NPVP* of the grammar G (Figure 3) the sentence matches to *NP noun verb*; in the second iteration the *noun verb* part matches to *noun VP1*. The *VP1* in turn matches to *verb VP4* and *VP4* produces ε. Figure 6 shows a snap shot of the moves of our implementation using a stack.
Initially, the parser is in a configuration with S$ in the input buffer and the start symbol S on top of the stack and then $.



Stack	Input	Action
$ S	modifier noun noun verb $	
$ VP NP	modifier noun noun verb $	S->NP VP
$ VP noun modifier	modifier noun noun verb $	NP->modifier noun
$ VP noun	noun noun verb $	Poped
$ VP	noun verb $	Poped
$ VP1 noun	noun verb $	VP->noun VP1
$ VP1	verb $	Poped
$ VP4 verb	verb $	VP1->verb VP4
$ VP4	$	Poped
$	$	VP4->zero
$	$	Sentence is accepted

Figure 6: Moves made by a Bangla parser on input "modifier noun noun verb" for correct sentence.

```
1.  set Input Pointer(IP) to point to the first word of w;
2.  set X to the top stack word;
3.  while ( X!= $ ) { /* stack is not empty */
      a.  if  X is a terminal,  pop the stack and advance
          IP;
      b.  if X is a Nonterminal and  M[X,IP] has  the
          production  X → Y1Y2…Yk  output the
          production X -> YlY2 -Yk;
         pop the stack;
         push Yk, Yk-1,. . . , Yl onto the stack, with Yl
      on top;

      c.  if X=$ ,Sentence is Accepted.
   }
```

Figure 7: Bangla parsing algorithm

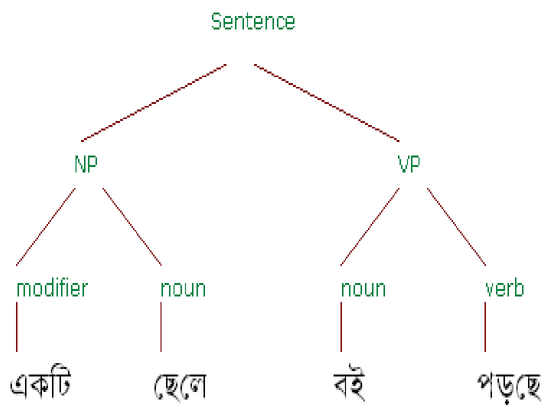


Figure 8: Parse tree for Bangla parser on input *"modifier noun noun verb"*

Figure 7 shows parsing algorithm which uses the parse table *M* of Table 3 to produce a Bangla parser for the input of a Bangla sentence.
Figure 8 shows the parse table generated by the grammar G for the input "একটি ছেলে বই পড়ছে" of the form *modifier noun noun verb*. Figure 9 shows another parse tree for the Bangla sentence "রোগী পথ্য সেবন করে" of the form *" noun noun verb verb"*. Figure 10 shows the moves for an incorrect sentence "রহিম বিদ্যালয়ে" of the form *"noun noun"*

IV. Conclusion and Future Works

The aim of this paper was to Design and Develop algorithms for Bangla Parser. It is amazing that a lot of works have been done in designing the parsers for English language, but unfortunately no satisfactory work has been done in this field for Bangla language so far. If a natural language be successfully parsed then grammar checking from this language becomes easy. We believe the proposed grammar can be applied for the most of the Bangla sentences and can be extended to any format of Bangla sentence. Our Bangla Parser fail to indicate structural similarity. For example:

১. আমি ভাত খাই।
২. খাই আমি ভাত।

Above two sentences are approximately similar. Further modifying the CFG rule, we can design a more powerful parser for Bangla sentences. By combining the bottom-up approach and top-down approach together, we can also design a more efficient algorithm for parser. As a future improvement we want to organize the order of the productions in the parser then we can hope that more sentences of different format can be parsed automatically. However, we intend to extend this grammar to recognize and also to generate generic Bangla sentences.

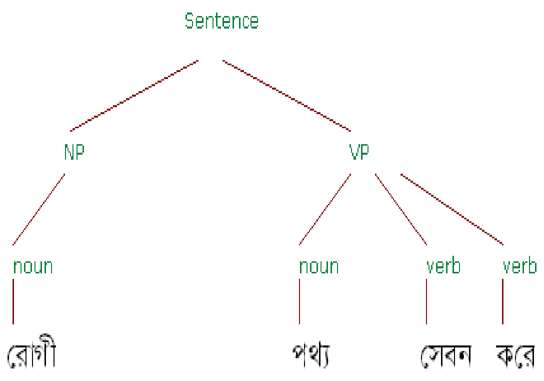
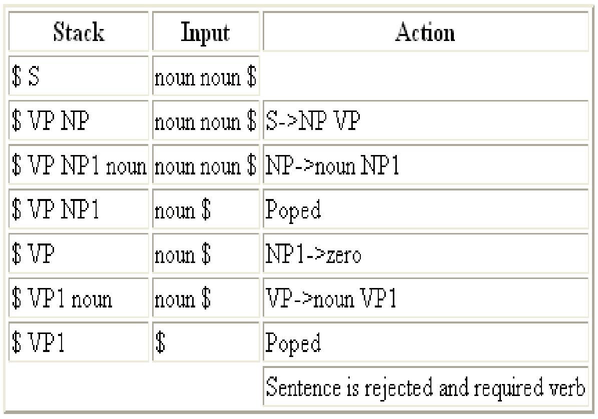


Figure 9: Parse tree for Bangla parser on input *" noun noun verb verb"*



Stack	Input	Action
$ S	noun noun $	
$ VP NP	noun noun $	S->NP VP
$ VP NP1 noun	noun noun $	NP->noun NP1
$ VP NP1	noun $	Poped
$ VP	noun $	NP1->zero
$ VP1 noun	noun $	VP->noun VP1
$ VP1	$	Poped
		Sentence is rejected and required verb

Figure 10: Moves made by a Bangla parser on input "noun noun" for incorrect sentence

References

[1] G.K. Saha, "Parsing Bengali Text: An Intelligent Approach", ACM Ubiquity, 7(13), 2006.
[2] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods", Proc. Of 33rd Annual meeting of the ACL, Cambridge, MA, USA, pp 189-196, 1995.
[3] M.N. Haque and M. Khan, "Parsing Bangla Using LFG: An Introduction", BRAC University Journal, 2(1), pp. 105-110, 2005.
[4] P. Sengupta and B. B. Chaudhuri, "A Delayed Syntactic-Encoding-based LFG Parsing Strategy for an Indian Language – Bangla" Computational Linguistics, 23(2), pp. 345-351, 1997.
[5] Md. Musfique Anwar, Mohammad Zabed Anwar and Md. Al-Amin Bhuiyan, "Syntax Analysis and Machine Translation of Bangla Sentences", International Journal of Computer Science and Network Security, VOL.9 No.8, pp-317-326, 2009.
[6] A. V. Aho, R. Sethi and J. D. Ullman " Compilers Principles, Techniques and Tolls", Pearson Education, 2002.
[7] Abu Zaher, Md. Faridee, Francis M. Tyers, "Development of a morphological analyser for Bengali", Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation, p. 43-50 Alacant, Spain, November 2009.
[8] M. M. Hoque and M. M. Ali, "A Parsing Methodology for Bangla Natural Language Sentences", Proceedings of International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, pp. 277-282, 2003.
[9] L. Mehedy, N. Arifin and M. Kaykobad, "Bangla Syntax Analysis: A Comprehensive Approach", Proceedings of International Conference on Computer and Information Technology (ICCIT), Dhaka, Bangladesh, pp. 287-293, 2003.
[10] 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, pp. 615-620, 2004.
[11] M. M. Murshed, "Parsing of Bengali Natural Language Sentences", In Proceedings of ICCIT, Dhaka, pp. 185-189, 1998.
[12] M. R. Selim and M. Z. Iqbal, "Syntax Analysis of Phrases and Different Types of Sentences in Bangla", In Proceedings of ICCIT, Dhaka, pp. 175-186, 1999.