Asian Journal of Information Technology 5(5) : 504-511, 2006
© Medwell Online, 2006

# Parsing Algorithms for Bengali Parser to Handle Affirmative Sentences

Mohammad Zakir Hossain Sarker, Shaila Rahman and M.A. Mottalib
Department of CSE, East West University, 43, Mohakhali, Dhaka-1212, Bangladesh
Department of CSE, The University of Asia Pacific, Road # 3/A, Dhanmondi, Dhaka-1209, Bangladesh
Department of CIT, Islamic University of Technology, Boardbazar, Gazipur, Dhaka, Bnagladesh

**Abstract:** This study embodies the design of parsing algorithms tangibly for a Bengali parser. To design parsing algorithms a detailed study on linguistics and grammar has been performed. A detailed study also has been made on the various techniques and algorithms of the parsers which have been designed for various languages such as English, Arabic, Hindi, Warlpiri(an Australian language) etc. Finally, Bengali sentences have been analyzed according to Chomskyan grammar. Bottom-up parsing technique and Context-sensitive rules have been used to design parsing algorithms. The designed parser can parse simple affirmative sentences for all types of tense i.e., Present, Past, Future and Habitual past tense together with all types of person i.e., First, Second and Third person. The algorithms can maintain the agreement of Person + Class between Verb form and Subject which is available in Bengali sentences. Words are stored into the dictionary (lexicon) together with lexical categories. Using these algorithms it would be possible to parse the Bengali sentences to let the users know whether those sentences are syntactically correct or not.

**Key words:** Bengali parser, bottom-up, context-sensitive, natural language processing, chomskyan grammar

## INTRODUCTION

According to D.W. Patterson[1], Artificial Intelligence (AI), the field of computer science that studies how computers can be made to act intelligently, is emerging from the laboratory and is beginning to take its place in human affairs. There are several sub-areas of AI research. Such as Robotics, Medical diagnosis, Character recognition, Natural Language Processing (NLP) etc. Natural Language Processing (NLP) which is one of the most precursory areas of AI research allows people to communicate with computers in a human language such as, English, Bengali, Hindi etc. as easily as it is to communicate with other people.

According to J. Allen[2], Natural Language (NL) is simply the language of human being. Developing programs that understands a natural language is a difficult problem. Natural language is the most versatile communication medium of mankind. Natural languages are very large. They contain infinity of different sentences. No matter how many sentences a person has heard or seen, new ones can always be produced. We speak to each other in a human language from our childhood and take most of its complex characteristics. Even, sometimes when we speak natural language incorrectly i.e., not strictly in accordance with rules of grammar and syntax, we can still make sense out of it. However, these situations create great problems for computers to understand people. This makes the creation of programs that understand a natural language, one of the most challenging tasks in AI.

In general, a language-understanding program must have considerable knowledge about the structure of that natural language including what the words are and how they combine into phrases and sentences. It must also know the meanings of the words and how they contribute to the meanings of sentences and to the context within which they are being used. Finally, a program must have some general world knowledge as well as knowledge of what a human knows. So, the component forms of knowledge needed for a natural language understanding program is sometimes classified as, Phonological, Morphological, Syntactic, Semantic, Pragmatic and World. This paper focuses on Syntactic knowledge. For example, the construction of the word *khaben* can be derived from *khabo* and *en* i.e., *khaben*= *khabo* + *en*. Syntactic knowledge relates to how words are put together or structured to form grammatically correct sentences in the language. To achieve this goal sentences are broken down into their component parts, (technically it can be said that, sentences are parsed into their component parts) and then the structures of those sentences are analyzed with the help of those component parts. Breaking down a sentence into its component parts is also

**Corresponding Author:** Mohammad Zakir Hossain Sarker, Department of CSE, East West University, 43, Mohakhali, Dhaka-1212, Bangladesh

known as parsing. To parse a sentence of a language it is needed to know the grammatical rules and grammar of that language. In this paper we have designed few parsing algorithms for Bengali parser. For doing so, we have to understand the Bengali grammar and its various rules very clearly. Few of the rules are also incorporated in this study.

## MATERIALS AND METHODS

This research is basically based on literature study. A detailed study on the existing techniques and algorithms for Bengali parser has been carried out from M.M. Hoque and M.M. Ali[5], Z.H. Sarker, S. Rahman and M.A. Mottalib[6], Z.H. Sarker, S. Rahman and M.A. Mottalib[7], Z. Khan[8]. A detailed study also has been made on the various techniques and algorithms of the parsers which have been designed for various languages such as English, Arabic, Hindi, Warlpiri (an Australian language) etc. from C. Huyck[9], D.I. Feriozi[10], M. Kashket[11], R. Sangal and V. Chaitanya[12]. After analyzing various parsing techniques Bottom up parsing technique has been used to design the parser in this dissertation. An extensive study on the background of linguistics and languages, on Chomskyan grammar (grammars that have been developed by Noam Chomsky, one of the greatest linguist of this century) and on non-Chomskyan grammar (grammars that have been developed by linguists excluding Chomsky) from A. Aho, J. Hopcroft and J. Ullman[13], C. Hockett[14], H. Andrew[15], N.A. Chomsky[16], N.A. Chomsky and H. Allen[17], P. Culicover[18], V.J. Cook[19]. Considering all, Chomskyan grammars have been used to design the present parser. But Chomsky described his grammars with respect to English language and English sentences have been analyzed according to these grammars. As a Bengali parser is going to be designed, it is needed to analyze Bengali sentences. For this reason, Chomskyan grammar and grammatical rules have been converted for analyzing Bengali sentences. For doing a detail study on Bengali grammar and grammatical rules has been made from H. Azad[20-23], L. Mehedy, N. Arifin and M. Kaykobad[24], M.A. Islam, M.A. Mottalib and L. Rahman[25], M.Z. Iqbal and M.R. Selim[26], P. Sengupta[27], S. Chatterji[28], S. Monir[29], Z. Khan and A. Radhakrishna[30], Z. Khan and R. Berwick[31]. Finally various algorithms have been designed using Context sensitive rules. As far as literature is concerned, these rules have not been used to design such algorithms before. As bottom up parsing technique and context sensitive rules are used, the various algorithms, which have been designed to develop the Bengali parser, are called Bottom up context sensitive algorithms. These algorithms are generally called parsing algorithms.

The workhorse of a natural language parser is the lexicon (dictionary). There is no way to use a natural language parser without a lexicon. A lexicon has been designed to store the Bengali words after studying P. Sengupta[18], R.A. Hudson[16]. A Bengali text editor is needed to accept Bengali sentences to be parsed. For the convenience of the work, a simple Bengali text editor has also been designed.

**Parsing and parsing techniques:** The activity of breaking down a sentence into its constituent parts is known as parsing. It is found that most parsing methods fall into one of these two classes, named as Top-down parsing method and Bottom-up parsing method. These terms refer to the order in which nodes in the parse tree are constructed. In Top-down parsing method, construction starts at the root and proceeds towards the leaves i.e., this method works from sentence symbol to the sentence. In case Bottom-up parsing method, construction starts at the leaves and proceeds towards the root i.e., this method works from the sentence to sentence symbol. Parsers, which are designed using Top-down parsing method, are known as Top-down parser and Parsers, which are designed using Bottom-up parsing method, are known as Bottom-up parser. The aim of this work is to parse given sentences, not to generate sentences. It can easily be found that, for this purpose Bottom-up parsing method is more appropriate than Top-down parsing method. Because, Bottom-up parsing method deals with the given sentences, it does not generate sentences. For this reason, Bottom-up parsing technique is used in this present work.

**Parse tree:** A parse tree is a structural representation of the sentences being parsed. In Figure 1 a parse tree is shown for the sentence Bipul Boi Porche. A parse tree represents the sentence in a hierarchical fashion, moving from a general description of the sentence (at the root of the tree) down to the specific sentence being parsed (the actual tokens) at the leaves.

## HOW THE SYSTEM WORKS TO PARSE SENTENCES

To parse a sentence the system accepts that sentence with help of a text editor. A Bengali text editor has been used in this thesis to accept a Bengali sentence. Then the first phase(a programme module) of the system which is called lexical analyzer (also called scanner or tokenizer) reads and converts the input sentences into a stream of tokens i.e., words. A lexicon (dictionary) is used

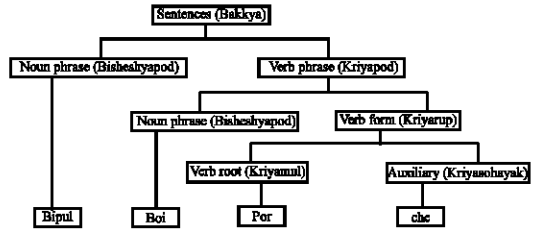


Fig. 1: Parse tree

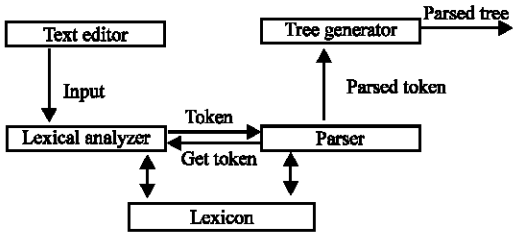


Fig. 2: Position of various program modules

here that contains these words along with their lexical categories (noun or pronoun or verb or adjective etc.). The lexical analyzer or scanner scans each token to test whether it is a member of the lexicon or not, if so then what lexical categories it belongs to. So the output of the lexical analyzer is a list structure called the token list. Then the parser accepts this token list and verifies the validity of its syntactic structure. At last a tree generator generates a parse tree if the sentence has the valid construction. Figure 2 shows the entire process graphically.

## INTRODUCTION TO LANGUAGE AND LINGUISTICS

The question what is language? is comparable with-and, some would say, hardly less profound than-what is life? the presuppositions of which circumscribe and unify the biological science. Great Noam Chomsky[13] said that: When we study human language, we are approaching what some might call the human essence the distinctive qualities of mind that are, so far as we know, unique to man. According to P. Culicover[15] Language is a purely human and non-instinctive method of communicating ideas, emotions and desires by means of voluntarily produced symbols. V.J. Cook[16] said that A language is a system of arbitrary vocal symbol by means of which a social group co-operates. These definitions of language quoted above are concerned with natural language. Linguistics is usually defined as the science of language or, alternatively, as the scientific study of language. C. Hockett[11] described linguistics as a science and used the term linguistic science instead of linguistics.

**Brief history of grammar:** Several break-through have occurred several times in the history of grammar. According to V.J. Cook[16] the different periods are:

[A] Traditional grammar
[B] Structural grammar
[C] Generative grammar
  [C.1] Finite state grammar
  [C.2] Phrase structure grammar
  [C.3] Transformational grammar
[D] Transformational Generative grammar
[E] Transformational grammar of Aspects model
  [E.1] Standard theory
  [E.2] Extended Standard theory
[F] Government and Binding (GB) theory
[G] Universal grammar

All grammars in between [C] and [G] have been developed and analyzed mainly by Noam Chomsky, one of the greatest linguists of the century. So this era (from the Generative grammar to Universal grammar) can be considered as Chomskyan era. Besides, some other linguists developed some other grammars after the development of Chomsky's Extended Standard theory[16]. These are:

[a] Generative semantics
[b] Transformational case grammar

After the development of Chomsky's Extended Standard theory some other linguists, like George Lakoff, M. Postal, Robert Ross, D.J. McCawly, Stockwell introduced another kinds of grammar, like Generative Semantics, Transformational Case Grammar etc[16]. Not only these grammars, there are some other types of grammar by which natural language can be analyzed for designing of parsers. But among so many different kinds of grammar Chomskyan grammar is used in this thesis to analyze Bengali sentences. Because, it is found and observed that, Chomskyan grammar is more powerful, universality (capability of analyzing all kinds of languages of this world) is better than any grammar. It is easy to understand, learn and use. Sentences can be analyzed without complexities. Under these considerations, Chomskyan grammar is used in this present work to analyze Bengali sentences for developing parsing algorithms.

## AN INTRODUCTION TO BENGALI GRAMMAR

Since the principle objective of this paper is to bring out a Bengali language parser thus tenses, verbs, persons etc. of Bengali grammar should be described. All of these are going to be described next.

**Tense(*kal*):** The word tense comes from the Latin word tempus which is meant time. So the tenses show the time of action and its degree of completeness. In fact, the tense is a form of the verb showing the time of the happening of an action. There are four kinds of tenses in Bengali grammar. These are-1.Present Tense (Bortoman Kal) 2. Past Tense (Otit Kal) 3. Future Tense (Bobhishot Kal) 4. Habitual Past Tense (Nittobritto Otit Kal). Again each of these tenses has three forms. These are-1. Simple (Sorol) 2. Continuous (Ghotoman) 3. Perfect (Ghotito)

**Person *(Purus*h) and class *(Shreni*):** There are three kinds of person in Bengali grammar. These are given below with examples

- First Person (Prothom Purush) -Ami, Amra
- Second Person (Ditiya Purush) -Apni, Tui, Tumi, Tora
- Third Person (Tritiyo Purush) -Se, Tara, Tini

There are three types of classes in Bengali grammar. These are given below with examples-

- Honorable (Sommanito) -Apni, Tini
- General (Shadharon) -Tumi, Tomra
- Negligible (Hino) -Tui, Tora

Classes are not applicable for all kinds of person. Such as-

- First person does not have any class.
- Second person consists of all three classes.
  Second Person Honorable -Apni, Apnara
  Second Person General -Tumi, Tomra
  Second Person Negligible -Tui, Tora
- Third person consists of two classes
  Third Person Honorable -Tini
  Third Person General -Se

**Verb forms (Kriyarup):** The word verb comes from the Latin word verbum. it is so called because it is the most important word in a sentence. In fact, it is the core or backbone of every sentence. It is the word used for stating something about a person or a thing. The Bengali verb form can be segmented into two parts. These are-

1. Verb roots (Kriyamul)
2. Auxiliary (Kriyasohayak)

Verb roots can not be analyzed further and the other part of verb forms that can be analyzed further is known as Auxiliary. Such as- Korechi → Kor (Kriyamul) + Echi (Kriyasohayak)

**More about auxiliary:** Auxiliary can represent-Tense, Aspect and Person + Class. For example, from Echi→Ech +I it is found that, Ech represents Tense (here, it is present tense) and Aspect (here it is perfect aspect). On the other hand i represents Person + class (here it is First person + No class).

**Agreement of person+class between verb form and subject:** It is said that, In a Bengali sentence there is an agreement of person+class between verb form and subject. It is already known that auxiliary can represent person + class of a Bengali sentence. Keeping this in mind we can say that if subject (Korta) is Apni then person+class of auxiliary should be en or for Tumi it should be 'o' or 'a' and so on. Such as, Apni Korchen→Apni + Kor + ch + en. But it is not possible to write Apni Korcho→Apni + Kor + ch + o Because o does not agree with the subject Apni. This agreement of person+class between subject and verb form can be represented only by Context-sensitive Phrase Structure. It is not possible to represent this agreement by Context-free Phrase Structure rules. So, Context-sensitive Phrase Structure rules have been used to design the algorithms for the present work.

## ANALYZING BENGALI SENTENCES USING CHOMSKYAN GRAMMAR

In this section Bengali sentences have been analyzed according to the Chomskyan grammar. Only Context-sensitive phrase structure grammar has been used to analyze Bengali sentences.

**Context-sensitive phrase structure grammar:** As per the discussion, it is considered that verb forms in Bengali language are divided into two parts Verb roots and Auxiliary. No further division is allowed here. Auxiliary indicates tense-aspects-person+ class as a unit. But the most important characteristics of Context-sensitive phrase structure grammar, which actually makes this grammar different from Context-free phrase structure grammar is those auxiliaries, are depended on subjects. The category of person+class for both the auxiliary and subject should be the same i.e., there is an agreement of person+class between the auxiliary and subject. Such as if the subject is Ami then Korchi or Porchi etc should be used, there are no ptions to use korcho or Porcho etc. Now considering all these conditions Phrase Structure Rules for Context-Sensitive Phrase Structure Grammar are written below for Present Tense only. In the same way it could written for other tenses also.

1. Sentence →BisheshyaPod + KriyaPod
2. KriyaPod →Kriyarup
3. Kriyarup→Kriyamul + Kriyasohayak

4. BisheshyaPod→Purush
5. Purush→ Prothom Purush (PP)
   Ditiya Purush (DP)
   Tritiyo Purush (TP)
6. Ditiya Purush →DP-Somman
   DP- Shadharon
   DP- Hino
7. Tritiyo Purush →TP-Somman
   TP- Shadharon
8. Kriyasohayak →Kriyariti + Kal + Songoti
9. Kal →Bortoman

10. Kriyariti Ghotoman→Sorol
    Ghotito
11. PP →Ami, Amra
12. DP-Somman →Apni, Apnara
13. DP-Shadharon →Tumi, Tomra
14. DP-Hino →Tui, Tora
15. TP-Somman →Tini
16. TP-Shadharon →Se
17. Kriyamul →Kor, Por, Likh ……
18. Bortoman →Φ
19. Sorol →Φ
20. Ghotoman →Ch
21. Ghotito →Ech
22. Songoti -PP→ i
23. Songoti - DP-Somman →En
24. Songoti - DP-Shadharon→ o
25. Songoti –DP-Hino →Ish
26. Songoti - TP-Somman →Een
27. Songoti - TP-Shadharon →a

An algorithm can be written according to this Context-sensitive phrase structure grammar which is given below. It is for Present Tense only. In the same way it could be written for other tenses also.

## ALGORITHM

This algorithm can handle verb, noun and person and this can be implemented for present tense. It works under a menu-driven system. A function Menu () returns the integer value to an integer variable Pochonder Rong, according to which either function Parser () or function Tree Generator () works. Parser () uses a lot of variables like, FP, SPH, SPG, SPN, TPH, TPG, T, Q, Tn, Tv, Ta, Tp, Tr, VP, NP, NP1 etc. FP, SPH, SPG, SPN, TPH, TPG are used to confirm the agreement of Person + class between Subject and Verb form. T is used to store the lexical categories of the words. Q is used to keep the word from the top of stack. VP, NP, NP1 are used to check Verb phrase and Noun phrase. Tn, TV, Tp, TR, Ta are actually used by the function Tree Generator (). A stack S is used to store the words and lexical categories. The editor mode is set to Bengali at the beginning and again set to English just before closing the program. This algorithm is developed according to the grammar described above.

- Set the Editor mode in Bengali
- Clear the screen
- Do Menu
- If Pochonder Rong =1 then Do Parser ()
- Else If Pochonder Rong =2 then Do Tree Generator ()
- Else If Pochonder Rong =3 then Set the Editor mode in English and Exit(0)
- Else Print Sothik no. din
- Set the Editor mode in English
- Exit

**Menu (): This function returns an integer value which selects the function to be executed**

- Print 1. Bakkya Bislesion
- Print 2. Tree Tairikoron
- Print 3. Ber Hon"
- Print Sathik no. din
- If Pochonder Rong < 1 or Pochonder Rong > 3 then goto Step 4
- Return

**Parser (): This function returns the message whether sentences are correct or not**

- FP='F', SPH='F', SPG='F', SPN='F', TPH='F', TPG='F'
- Push "NULL" to the stack S
- Load the Dictionary
- Tokenize the Input sentence
- Repeat steps until all the tokens are taken
- Search the Token into the dictionary
- a) If found then store the lexical categories in the variable T

{If T= *Bisheshya* then push *(Bisheshya*) to the stack S and store the Token in Tn

Else If T= Kriya then push (Kriya) to the stack S and store the Token in Tv
Else If T= i then push *(*i) to the stack S and store *i* in Ta
Else If T= en then push (en) to the stack S and store en in Ta
Else If T= ish then push (ish) to the stack S and store ish in Ta

Else If T= o then push (o) to the stack S and store o in Ta
Else If T= a then push (a) to the stack S and store a in Ta
Else If T= Prothom Purush then push (Prothom Purush) to the stack S and FP='T', Tp= FP
Else If T= Ditiya Purush Somman then push (Ditiya Purush Somman) to the stack S and SPH='T', Tp= SPH
Else If T= Ditiya Purush Shadharon push (Ditiya Purush Shadharon) to the stack S and SPG='T', Tp= SPG
Else If T= Ditiya Purush Hino then push (Ditiya Purush Hino) to the stack S and SPN='T', Tp= SPN
Else If T= TritiyoPurush Somman then push (Tritiyo Purush Somman) to the stack S and TPH='T', Tp= TPH
Else If T= Tritiyo Purush Shadharon then push (Tritiyo Purush Shadharon) to the stack S and TPG='T', Tp= TPG
Else If T= F then print Shadharon Bortoman Kal and Tr= F
Else If T= ch then print Ghotoman Bortoman Kal and Tr= ch
Else If T= ech then print Ghotito Bortoman Kal and Tr= ech}

b) Else [i] Print Dictionary-te Nai and Exit(0)
[ii] Enter the word into the lexicon and start again

- VP='F', NP='F', NP1='F', Flag='F', Te='F' and store the top element of stack S into Q
- Repeat steps while(Q!='NULL' && (VP && NP != 'T'))
- If Q= *Bisheshya* then NP1='T'
- Else If Q= i then If FP!='T' then print Bakkyati Sathik Na and Exit(0) Else Flag='T'
- Else If Q= en then If SPH or TPH !='T' then print Bakkyati Sathik Na and Exit(0) Else Flag='T'
- Else If Q= *o* then If SPG !='T' then print Bakkyati Sathik Na and Exit(0) Else Flag='T'
- Else If Q= ish then If SPN !='T" then print Bakkyati Sathik Na and Exit(0) Else Flag='T'
- Else If Q= a then If TPG !='T' then print Bakkyati Sathik Na and Exit(0) Else Flag='T'
- Else If Q= Kriya then VP='T'
- Else If Q= Prothom Purush then NP='T'
- Else If Q= Ditiya Purush Somman then NP='T'
- Else If Q= Ditiya Purush Shadharon then NP='T'
- Else If Q= Ditiya Purush Hino then NP='T'
- Else If Q= Tritiyo Purush Somman then NP='T'
- Else If Q= Tritiyo Purush Shadharon then NP='T'
- Store the top element of the stack S in the variable Q
- If VP && NP='T' then print Bakkyatir ghoton sathik na and Te='T'
  Else If VP !='T' && NP ='T' then print Bakkyatte Kriya nei
  Else Print Bakkyatir ghoton sathik na
- Return

**Tree Generator (): This function will work if the variable *Te*='T'; Otherwise Exit (0)**

- Move the cursor to (2, 24) and print Bakkya
- Move the cursor to (7, 11) and print BisheshyaPod
- Move the cursor to (7, 41) and print KriyaPod
- Move the cursor to (10, 11) and print Bisheshya
- Move the cursor to (10, 33) and print BisheshyaPod
- Move the cursor to (10, 53) and print Kriyarup
- Move the cursor to (13, 11) and print Purush
- Move the cursor to (13, 33) and print Bisheshya
- Move the cursor to (13, 43) and print Kriyamul
- Move the cursor to (13, 62) and print Kriyasohayak
- Move the cursor to (15, 55) and print Kriyariti
- Move the cursor to (15, 62) and print Kal
- Move the cursor to (15, 69) and print Songoti
- Move the cursor to (17, 11) and If Tp= FP/SPH/SPG/SPN... then print Purush Purush/Ditiya Purush- Somman.. ..
- Move the cursor to (17, 55) and If Tr= F/ch/ech then print F/ Ghotoman/ Ghotito
- Move the cursor to (17, 62) and print Bortoman
- Move the cursor to (17, 69) and If Ta= i/ish/a.. .. then print Ditiya Purush-Hino / Tritiyo Purush-Shadharon.. .. ..
- Move the cursor to (23, 11) and print Token of Purush
- Move the cursor to (23, 33) and print Tn
- Move the cursor to (23, 44) and print Tv
- Move the cursor to (23, 55) and print Tr
- Move the cursor to (23, 69) and print Ta
- Join all of the necessary points
- Return

## CONCLUSION

The aim of this study was to design and develop algorithms for the Bengali parser. It is amazing that a lot of works have been done in designing the parsers for English language, but unfortunately (as far as literature is concerned) no satisfactory work has been done in this field for Bengali language so far. The aim of the entire work was to design parser for Bengali language since it's

almost an untouched field. For doing so an extensive knowledge on linguistics and languages (especially on Bengali language) was required. The summary of the knowledge which was gathered is discussed in section 5, 6 and 7. Finally algorithms are developed for Bengali parser and discussed in section 8. To complete the natural language processing system a few more steps such as, Semantic, pragmatic, Natural Language Generation (NLG) etc. should be accomplished and this makes the whole process very difficult to develop. In fact it is proved that NLP is the most difficult task for AI technology. To many AI researchers, NLP is the most important and crucial of all AI goals and developing a complete computational model for NLP will be the final AI task to be performed. Since it's only a starting point of designing and implementing various algorithms for a Bengali parser there are ample scope for improvement. The algorithms which are discussed in this paper are only applicable for simple affirmative sentences. At present we are trying to develop algorithms for Interrogative, Negative, Exclamatory sentences of the Bengali language. After doing so we will be hoping to design more algorithms for Compound, Complex sentences.

## REFERENCES


1. Patterson, D.W., 1990. Introduction to Artificial Intelligence and Expert System, Prentice-Hall of India Pvt. Ltd., New Delhi, India.
2. Allen, J., 1990. Natural Language Understanding, 1 and 2, Benjamin Publishing Company.
3. Murshed, M.M., 1997. Design and Implementation of a bilingual natural language Parser (For Bengali and English sentences), M.Sc. Thesis, Department of Computer Science, University of Dhaka.
4. Murshed, M.M., 1998. Parsing of Bengali Natural Language Sentences, Proceedings of International Conference on Computer and Information Technology (ICCIT), Dhaka.
5. Hoque, M.M. and M.M. Ali, 2003. A Parsing Methodology for Bangla Natural Language Sentences, Proceedings of International Conference on Computer and Information Technology (ICCIT)- Jahangirnagar, Dhaka, 1: 277-282.
6. Sarker, Z.H., S. Rahman and M.A. Mottalib, 2003. Design and Implementation of Bottom-up Context-sensitive Algorithms for Bengali Parser in Natural Language Processing, Proceedings of International Conference on Computer and Information Technology (ICCIT), Jahangirnagar, Dhaka, 1: 332-337.
7. Sarker, Z.H., S. Rahman and M.A. Mottalib, 2005. Bottom-Up Parsing Algorithms for Bengali Parser to Maintain the Freeness of Word Order, Proceedings of National Conference on Computer Processing of Bangla (NCCPB), Dhaka, pp: 65-75.
8. Khan, Z., 1995. Parsing system design in Bengali using Pappi's grammar, M. Eng. Thesis, MIT.
9. Huyck, C., 1996. PLINK: An intelligence natural language parser, Ph.D. Thesis, University of Michigan, USA.
10. Feriozi, D.I., 1998. A practical methodology storing LL(k) parsing, Ph.D. Thesis, Florida Atlantic University, USA.
11. Kashket, M., 1996. A parameterized parser for English and Warlpiri. Ph.D. Thesis, University of Maryland College Park.
12. Sangal, R. and V. Chaitanya, 1999. Natural Language Processing: A Paninian Perspective, Prentice-Hall of India Pvt. Ltd., New Delhi, India.
13. Aho, A., J. Hopcroft and J. Ullman, 1980. The Theory of Parsing, Translation and Compiling, Prentice-Hall, Englewood, Cliffs NJ, USA.
14. Hass, A., 1997. A parsing algorithm for unification grammar, Computational Grammar, 15: 219-232.
15. Andrew, H., 1998. A parsing algorithm for unification grammar, Computational Grammar, 15: 219-232.
16. Chomsky, N.A., 1972. Language and Mind, NewYork, Harcourt Brace, USA.
17. Chomsky, N.A. and H. Allen, 1977. Some methodological remarks on Generative grammar, Readings in Applied English Linguistics, Indian Edn., Amerind publishing Company, New Delhi, India, pp: 36-45.
18. Culicover, P., 1976. Syntax, 3rd Edn., Academic Press, New York, USA.
20. Azad, H., 1978. Reciprocal Structures in Bengali, Jahangirnagar Rev., 2: 171-190.
21. Azad, H., 1979. Bakkya Sorbonamiyokoron, Bhasha SahittyoPatro, Dhaka, Year 4: 107-138.
22. Azad, H., 1980. Rupantormuluk Sristyshil Bakoron, Borsha, Dhaka, pp: 23-196.
23. Azad, H., 1994. BakkyaTatto, 2nd Edn., Dhaka University, Dhaka.
24. Mehedy, L., N. Arifin and M. Kaykobad, 2003. Bangla Syntax Analysis: A Comprehensive Approach, Proceedings of International Conference on Computer and Information Technology (ICCIT)-Jahangirnagar, Dhaka. 1: 287-293.
25. Islam, M.A., M.A. Mottalib and L. Rahman, 1997. Development of Machine Translation: An overview, Proceedings of National Conference on Computer and Information System-Dhaka, pp: 67-72.

26. Iqbal, M.Z. and M.R. Selim, 1999. Syntax Analysis of Pharases and Different Types of Sentences in Bangla, Proceedings of International Conference on Computer and Information Technology (ICCIT), Sylhet, pp: 183-186.
27. Sengupta, P., 1993. On lexical and syntactic processing of Bangla language by computer, Ph.D. Thesis, Indian Statistical Unit, Calcutta, India.
28. Chatterji, S. The Origin and Development of Bangla Language, Calcutta, India.
29. Monir, S., 1989. Bangla Bakoron, Students Publications, Dhaka.
30. Khan, Z. and A. Radhakrishna, 1996. Verb classes and alternations in Bengali, MIT AI Memo.
31. Khan, Z. and R. Berwick, 1998. A computational linguistics analysis of Bangla using GB theory, Proceedings of the International Conference on Computational Linguistics, Speech and Document processing, Calcutta, India.