Parsing Bangla Grammar Using Context Free Grammar

Chapter · January 2014

DOI: 10.4018/978-1-4666-6042-7.ch044

4 authors, including:

Bishnu Sarker
National Institute for Research in Computer Science and Control
11 PUBLICATIONS **12** CITATIONS
SEE PROFILE

K. M. Azharul Hasan
Khulna University of Engineering and Technology
69 PUBLICATIONS **317** CITATIONS
SEE PROFILE

Chapter 7
Parsing Bangla Grammar Using Context Free Grammar

Al-Mahmud
Khulna University of Engineering and Technology (KUET), Bangladesh

Bishnu Sarker
Khulna University of Engineering and Technology (KUET), Bangladesh

K. M. Azharul Hasan
Khulna University of Engineering and Technology (KUET), Bangladesh

ABSTRACT

Parsing plays a very prominent role in computational linguistics. Parsing a Bangla sentence is a primary need in Bangla language processing. This chapter describes the Context Free Grammar (CFG) for parsing Bangla language, and hence, a Bangla parser is proposed based on the Bangla grammar. This approach is very simple to apply in Bangla sentences, and the method is well accepted for parsing grammar. This chapter introduces a parser for Bangla language, which is, by nature, a predictive parser, and the parse table is constructed for recognizing Bangla grammar. Parse table is an important tool to recognize syntactical mistakes of Bangla sentences when there is no entry for a terminal in the parse table. If a natural language can be successfully parsed then grammar checking of this language becomes possible. The parsing scheme in this chapter works based on a top-down parsing method. CFG suffers from a major problem called left recursion. The technique of left factoring is applied to avoid the problem.

INTRODUCTION

Language plays the most important role in human communication. Human communication is based on exchange of feelings, sharing of knowledge, etc. Feelings can be exchanged through voice, symbols and signs. Language provides the most convenient way of expressing the expressions of feelings through providing those necessary phonetics that could be spoken and the symbols that could be written to be preserved. Language is the driving force to human communication. Language is the most important medium to represent and express human knowledge and human communication.

DOI: 10.4018/978-1-4666-3970-6.ch007

Bangla (or Bengali) is one of the more important Indo-Iranian languages, is the sixth most popular in the world and spoken by a population that now exceeds 250 million. Geographically, Bangla-speaking population percentages are as follows: Bangladesh (over 95%), and the Indian States of Andaman & Nicobar Islands (26%), Assam (28%), Tripura (67%), and West Bengal (85%). The global total includes those who are now in diaspora in Canada, Malawi, Nepal, Pakistan, Saudi Arabia, Singapore, United Arab Emirates, United Kingdom, and United States.

Bangla is still in degraded stage at least as far as work in the area of computational linguistics is concerned. Natural languages like English and even Hindi is rapidly progressing as far as work done in processing by computers is concerned. Unfortunately, Bangla lags more or less behind in some crucial areas of research like parts of speech tagging, text summarization and categorization, information retrieval and most importantly in the area of grammar checking. The grammar checking for a language has a wide variety of applications.

The activity of breaking down a sentence into its constituent parts is known as parsing. Parsing is an earlier term for the diagramming of sentences of natural languages, and is still used for the diagramming. Parsing a sentence involves the use of linguistic knowledge of a language to discover the way in which a sentence is structured. Exactly how this linguistic knowledge is represented and can be used to understand sentences is one of the questions that has engaged the interest of psycholinguists, linguists, computational linguists, and computer scientists. Bangla parsing is a challenging task. This chapter has a detail discussion over Bangla parsing using Context Free Grammar.

BACKGROUND

In computing, a parser is one of the components in an interpreter or compiler that checks for correct syntax and builds a data structure (often some kind of parse tree, abstract syntax tree or other hierarchical structure) implicitly in the input tokens. Parsing can be defined as a method where a parser algorithm is used to determine whether a given input string is grammatically correct or not for a given grammar. 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 (Saha, 2006). The parser often uses a separate lexical analyzer to create tokens from the sequence of input characters. Parsers may be programmed by hand or may be automatically or semi-automatically generated (in some programming languages) by a tool.

A parse tree for a grammar is a tree where the root of the tree is the start symbol for the grammar, the interior nodes are the non-terminals of the grammar, the leaf nodes are the terminals of the grammar and the children of a node starting from the left to the right correspond to the symbols on the right hand side of some production for the node in the grammar. Every valid parse tree represents a string generated by the grammar (Yarowsky, 1995).

A parser analyzes the sequence of symbols presented to it based on the grammar (Yarowsky, 1995). Natural language applications namely Information Extraction, Machine Translation, and Speech Recognition, need to have an accurate parser (Haque & Khan, 2005). 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 infinite in number of vocabulary. It is difficult to prepare formal rules to describe informal behavior even though it is clear that some rules are being followed. 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 (Sengupta & Chaudhuri, 1997). The text is considered correct if the POS-tagged text contains POS sequences with frequencies higher than some threshold (Anwar, Anwar & Bhuiyan, 2009)

In this chapter, we implemented Bangla dictionary in XML format using the corresponding word as tag name and it's POS as value. It is a very useful technique because it needs less time to store data as well as data retrieval from this data storage is very easy and fast compared to other popular forms of data storage. It helps to search the dictionary very fast. Bangla grammar has a large number of forms and rules. The aim of this chapter is to describe a parser that parses a Bangla sentence, rather than generating sentences. A parse tree is displayed if the Bangla parser decides that sentence is correct otherwise parse tree are not generated. This technique of parsing system with XML checks whether every sentence of given Bangla passage is grammatically correct or not correct. It also indicates the location of problems. If any sentence of the passage is found accepted or correct then it generates a parse tree.

There has been a structural difference between the Bangla and English grammar. It is challenging to parse Bangla and English sentence with the same grammatical structure. If we match word to word with Bangla and English grammar structure then the parser will not give the correct answer.

For example, considering the grammatical structure:

Subject + verb + object (I + eat+ rice)

In Bangla it appears as:

আমি + খাই + ভাত.

But it is a non-traditional form. The traditional form is:

আমি + ভাত + খাই.

If we transfer it as word to word then the English form will become:

I + rice + eat.

Here, in English object is appeared after the verb whereas in Bangla the object is appeared before the verb. It's clearly states that how difficult to parse Bangla sentences rather than English sentences.

Roughly speaking, phrases and idioms are expressions whose meaning cannot be completely understood from the meanings of the component parts. For example, whereas it is possible to work out the meaning of (1a) on the basis of knowledge of English grammar and the meaning of words, this would not be sufficient to work out that (1b) can mean something like 'If Sam dies, her children will be rich'. This is because *kick the bucket* is an idiom.

1a: *If Sam mends the bucket, her children will be rich.*

1b: *If Sam kicks the bucket, her children will be rich.*

The problem with phrases and idioms, in a parsing context, is that it is not usually possible to parse them using the normal rules.

Throughout the chapter there will be a detailed discussion over Context Free Grammar for parsing Bangla Text and predictive Bangla parser for constructing parse table. Here a top down parsing scheme is adopted and the problem of left recursion is avoided applying left factoring over the proposed CFG.

RELATED WORKS

Modern Bangla morphology is very productive, especially for verbs, with the root verbs takes around 168 different forms. Bangla lexicon also has a very large number of compound words

(words that have more than one root), which can be created from at most any combination of nouns, pronouns and adjectives. An effort is made at building a complex morphological parser for Bangla, where it can only handle simple words with a single root. Though the addition of inflectional suffixes in Bangla compound word is fairly complex, the compound word's individual root words may retain their inflectional suffixes. Such a compound word morphological parser is developed which can efficiently parse compound words having inflectional suffix and also resolves ambiguities. Some rules are used to develop the morphological analysis of simple and compound Bangla words that can be used to make Universal Natural Language (UNL)-Bangla dictionary for converting the natural Bangla sentences to UNL documents and vice versa.

There has been no organized formal initiative for generating a context-free grammar for Bangla Language. There are some books available on context-free grammars for Bangla but most of them have been limited to the writers' way of generating the grammar rather than producing a generic grammar to cover the whole language (Anwar, Anwar & Bhuiyan, 2009; Aho, Sethi & Ullman, 2002).

Shift-reduce parsing is the simplest kind of bottom-up parsing. Here the parser takes an input string (sentence) and repeatedly pushes the next input word onto a stack which is the shift operation. If the top n items on the stack match the n items on the right-hand side of some production, then they are popped off the stack and the item on the left-hand side of the production is pushed on the stack. This replacement operation is the reduce part of the parsing. The parser ends its work when all the input is consumed and there is only one item remaining on the stack which is a parse tree with the start symbol of the grammar as its root. The parser fails to parse a sentence when there are no remaining input words to shift or when there is no way to reduce the remaining items on the stack. The second problem occurs because the parser blindly reaches the root of terminal using the productions (Sengupta & Chaudhuri, 1997).

Recursive descent Parsing is another way to produce parse trees for an input sentence. This parser interprets the grammar as a specification of how to break a high-level goal into several lower-level sub-goals. The top-level goal is to find the start symbol of the grammar and sub-goals are the production that breaks down the start symbols of the grammar. Such sub-goals are matched with an input sentence and succeed to its next state if the next word is matched. If there is no match the parser backtracks to a previous node and tries a different alternative. One problem with this recursive decent method is that when a production is recursive, the parser keeps on generating a never-ending parse tree when it selects the recursive production (Sengupta & Chaudhuri, 1997). An advantage for recursive decent over the shift-reduce method is that the parser can be forced to choose a specific production to continue parsing. Thus, this makes it easier to produce correct parse tree of a sentence in a given grammar which may not be possible with a shift-reduce parser for the same grammar.

A rule-based Bangla parser has been proposed in (Saha, 2006), 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 using finite state technology is described in (Faridee & Tyers, 2009). They developed the monolingual dictionary called Monodix, stored in XML file.

Anwar, Anwar & Bhuiyan (2009) address a method to analyze 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.

There have been several parsing technique proposed for Bangla Language. A parsing methodology for Bangla natural language sentences

is proposed by Hoque and Ali (2003) and shows how phrase structure rules can be implemented by top-down and bottom-up parsing approach to parse simple sentences of Bangla. An approach named Predicate Preserving Parser (PPP) is described in (Ali, Ripon and Allayear 2012) which maps Bangla text in UNL which then can be translated to any other natural languages. A technique of unsupervised morphological learning for Bengali language is introduced in (Dasgupta & Ng, 2006). A Bengali Dependency Parser based on statistical data driven parsing system followed by a rule based post processing is presented in (Ghosh et. al., 2010). A comprehensive approach for Bangla syntax analysis was developed (Mehedy, Arifin & Kaykobad, 2003) 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) (Sengupta & Chaudhuri, 1997), and Context Sensitive Grammar (CSG) (Hoque & Ali, 2004; Murshed, 1998; Selim & Iqbal, 1999) have also been developed for parsing Bangla sentences. LFG is a monotonic theory of syntax. Instead of postulating different derivational levels represented in the same formal language, it incorporates different parallel levels of information, which can all potentially access each other, each with its own formal language. The assumption about parallel levels of information extends even to non-syntactic aspects of grammar. Thus, for example, semantic information is assumed to be available to various levels of syntax, and syntactic levels can input into phonology (Joshi, 1993).

Some developed Bangla parser using SQL to check the correctness of sentence; but its space complexity is inefficient. Besides, it takes more time for executing SQL command. As a result those parsers becomes slower. A technique is implemented to perform structural analysis of Bangla sentences of different tenses using Context-Free Grammar (CFG) rule (Anwar, Shume & Bhuiyan, 2010). A methodology for analyzing the Bangla sentences in semantic manner is presented in (Hoque, Rahman & Dhar, 2007) and (Hasan, Mondal & Saha, 2010) presents a technique for detecting the named entity based on classifier for Bangla documents. But these papers (Anwar, Shume & Bhuiyan, 2010; Hoque, Rahman & KumarDhar, 2007; Hasan, Mondal & Saha, 2010) do not deal with the detail grammar recognition for Bangla sentences.

PARSING BANGLA SENTENCES WITH CONTEXT FREE GRAMMAR

Context Free Grammar

The most commonly used and mathematical system for modeling constituent structure in natural languages is the Context-Free Grammar, or CFG. Context-free grammars are also called Phrase Structure Grammars. A Context-Free Grammar (CFG) is a set of recursive rewriting rules called productions used to generate string patterns. A CFG consists of the following components (Aho, Sethi & Ullman, 2002):

1. A start symbol for the grammar.
2. A set of terminal symbols that are characters that appear in the strings generated by the grammar.
3. A set of non-terminal symbols that are placeholders for patterns of terminal symbols that can be generated by the non-terminal symbols.
4. A set of productions that are used to replace the non-terminals with other non-terminals or terminals.

A formal language is context-free if there is a context-free grammar that generates it.

This chapter describes a way of producing CFG for Bangla Language and designing a Bangla parser. The grammar discussed throughout the chapter can successfully parse a number of sen-

tences. We have defined a tag set and according to the tag set, we tagged the word. In the next step of producing CFG, we defined the constituents and finally the rules were generated. We used top-down parsing techniques for designing Bangla parser by using XML. Our parser decides which sentence is correct and which is to be rejected. And finally generate a parse tree for this sentence by using Bangla Parser.

There are a number of important subclasses of the context-free grammars:

- LR(*k*) grammars (also known as deterministic context-free grammars) allow parsing (string recognition) with deterministic pushdown automata, but they can only describe deterministic context-free languages.
- Simple LR, Look-Ahead LR grammars are subclasses that allow further simplification of parsing.
- LL(*k*) and LL(*) grammars allow parsing by direct construction of a leftmost derivation as described above, and describe even fewer languages.
- Simple grammars are a subclass of the LL(1) grammars mostly interesting for its theoretical property that language equality of simple grammars is decidable, while language inclusion is not.
- Bracketed grammars have the property that the terminal symbols are divided into left and right bracket pairs that always match up in rules.
- Linear grammars have no rules with more than one non terminal in the right hand side.
- Regular grammars are a subclass of the linear grammars and describe the regular languages, i.e., they correspond to finite automata and regular expressions.

Context-free grammars play a central role in the description and design of programming languages and compilers. They are also used for analyzing the syntax of natural languages. Noam Chomsky (1956) has posited that all human languages are based on context-free grammars at their core, with additional processes that can manipulate the output of the context-free component (the transformations of early Chomskyan theory).

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. Parsers designed using top-down parsing method are known as top-down parsers.

On the contrary, in case of 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. And parsers which are designed using bottom-up parsing method are known as bottom-up parser.

There are some well-recognized problems when concerning with bottom up parsing. Bottom up parsing suffers from termination problem. It is inefficient when dealing with empty categories. Bottom up parsing is data directed as it attempts to parse the words that are there and inefficient when there is great lexical ambiguity (grammar-driven control might help here).

Parsing of Bangla sentences by using grammar rules is still in a rudimentary stage. Very little research has been conducted regarding parsing of Bangla sentences but a significant number of research activities have been conducted on the recognition of Bangla characters. Some developers developed Bangla parser using SQL to check the correctness of sentence. But it takes more space on our computers, meaning its space complexity

is inefficient. Besides, it takes more time for executing SQL command. As a result those parsers becomes slower. To decrease time and space, XML files re used for storing data which is more speedy and takes up less space.

The extensions and refinements presented in this chapter over previous works especially in Hasan et al. (2011) are as follows:

- A set of CFG rules.
- Store data into XML file.
- Design a Bangla Parser by using table-driven method with top-down approach.
- Remove grammar ambiguity by using left factoring.
- Construct a parse tree.

Bangla words are stored as tag and its corresponding constituents or POS are used as value. More over the idea of left factoring have been applied to remove the left recursion and ambiguity of the grammar and hence according to the designed grammar, the parser can detect the errors in a Bangla sentence.

SOLUTION AND RECOMMENDATIONS

A Parsing Scheme for Bangla Grammar Recognition

A predictive parser is an efficient way of implementing recursive decent parsing by handling the stack of activation record.

The predictive parser is composed of four elements namely:

1. **Input:** Contains the string to be parsed or checked, followed by a $, the right end marker.
2. **Stack:** Contains a sequence of grammar symbols.
3. **Parse Table:** A two dimensional array M[A,*n*] where A is nonterminal and n is a terminal or $ sign.
4. **Output.**

Building the Word Repository/ Dictionary Using XML

XML (eXtensible Markup Language) has been designed as a markup language and a textual format. It provides for a description of a document's contents, with non-predefined tags, and does not provide for any presentational characteristics. The eXtensible Markup Language (XML) is now used almost everywhere for its simplicity and ease of use. 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. XML have several advantages. Some of them are:

1. Ease of access.
2. Efficient in response to access time.
3. Easy to format the grammar structure.
4. No need to use any space consuming DBMS.
5. Ease of defining the category of words.
6. Easy to implement and integrate.

The general format of XML tag is

<tag_name>value</tag_name>

A dictionary is a very basic Natural Language Processing (NLP) tool used to get the meaning, parts of speech, and usage of a word and can also be used as a spell-checker to detect errors in a sentence and correct them by providing a set of correct alternatives which includes the intended word. 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.

Figure 1. XML word repository

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpyII-->

<WORD>
        <আমি>pronoun</ আমি >
        <খাই> verb</খাই >
        <একটি >modifier</একটি >
        <এবং >conjunction</এবং >
        <আমরা > pronoun</আমরা >
        <না > neg</ না >
</WORD>
```

Categorization of Words

Each sentence is composed of one or more phrases. So if the syntactic constituents of sentences are identified, it will be easier to obtain the structural representation of the sentence (Hoque & Ali, 2003; Mehedy, Arifin & Kaykobad, 2003). Bangla words are tagged with their respective Parts-Of-Speech (POS), modifier, pattern, number (Haque & Khan, 2005), and stored in XML file.

Table 1 shows the tag sets used in XML file for storing Bangla words. Every word is tagged with the appropriate member of the tag set and stored in the XML file in a definite format.

Table 1. Tag set description for Bangla grammar

Tag Name (Symbol)	Examples
Noun (noun)	রহিম, বিদ্যালয়ে, ছেলে, বই, ভাত,টাকা|
Pronoun (pronoun)	আমি, আমরা, তুমি, সে, তারা।
Adjective (adjective)	ভাল, দশ, অনেক, শখ।
Verb (verb)	খাই, খেলে, পড়ছে,পড়া।
Conjunction (conjunction)	এবং, ও, চেয়ে|
Negative Description (neg)	না , নয়|
Modifier (modifier)	এ, একটি, একদিন|

Defining Constituents and Phrases

Each sentence is to be partitioned into its constituents. A constituent expresses the complete meaning for a specific context. After tagging the words, each sentence is to be broken down into their constituent parts. The constituents of the sentences found out are shown in the Table 2.

Grammar Design with CFG

Once constituents have been identified, the productions for Context-Free Grammar (CFG) are developed for Bangla sentence structures. Table 3 shows the productions along with their respective constituents. As Bangla grammar has different forms, the same production term can be used only by reorganizing the in the grammar. For example, following three forms can be applied by reorganizing the production terms.

1. আমি ভাত খাই |
2. আমি খাই ভাত |
3. ভাত আমি খাই |

CFG Productions for Bangla Grammar

The CFG productions for the Bangla grammar G is defined as follows:

S –> NP VP

NP –> noun conjunction noun | noun ip| noun pronoun conjunction pronoun| noun pronoun ip| noun pronoun noun| noun pronoun adjective| noun pronoun pronoun| noun pronoun tp| noun adjective| noun noun conjunction pronoun| noun noun aw| noun noun| noun| noun pronoun| pronoun conjunction pronoun| pronoun ip| pronoun noun conjunction noun| pronoun noun ip| pronoun noun adjective| pronoun noun conjunction pronoun| pronoun noun aw| pronoun noun| pronoun adjective| pronoun pronoun conjunction pronoun|

Table 2. Defining constituents

No.	Constituents	Symbol	Definitions	Examples
1	Noun Phrase	NP	Noun phrase may be consists of only noun or pronoun. It may be modified by quantifier, post position, specifier, plural marker etc.	রহিম, আমি, এ পথে।
2	Verb Phrase	VP	A group of words including a verb and its complements, objects, or other modifiers that functions syntactically as a verb. In English a verb phrase combines with a noun or noun phrase acting as subject to form a simple sentence.	বই পড়ছে, পথ্য সেবন করে, খাওয়া হল না।
3	Adjective Phrase	AP	A group of words including an adjective and its complements or modifiers that functions as an adjective, as too openly critical of the administration.	দশ, ভাল ছেলে।

Table 3. 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, পথ্য সেবন করে, খাওয়া হল না।
Adjective Phrase (AP)	AP –> adjective AP –> adjective noun etc .	দশ, ভাল ছেলে।

pronoun pronoun aw| pronoun pronoun| pronoun tp| pronoun| modifier noun| modifier adjective ptrn| modifier adjective| modifier pronoun| modifier conjunction adjective| modifier| adjective noun| adjective pronoun| adjective conjunction adjective| adjective ptrn| adjective| ip| tp| xp ip| xp pronoun conjunction pronoun| xp pronoun ip| xp pronoun noun| xp pronoun adjective| xp pronoun tp| xp adjective| xp noun conjunction pronoun| xp noun aw| xp noun| xp tp| xp aw| tp pronoun| tp adjective| tp ip| tp pronoun conjunction pronoun| tp pronoun noun| tp pronoun adjective| tp pronoun tp| tp noun conjunction pronoun| tp noun aw| tp aw.

AP –> adjective noun| adjective pronoun| adjective ptrn

VP –> noun verb| noun verb verb ptrn| noun verb verb adjective verb| noun verb verb adjective noun| noun verb verb adjective pronoun| noun verb verb adjective| noun verb verb noun adjective verb| noun verb verb noun ptrn| noun verb verb noun aw| noun verb verb noun adjective verb| noun verb verb noun verb ptrn| noun verb verb noun verb aw| noun verb verb noun verb adjective| noun verb verb noun ptrn| nounverb verb noun pronoun| noun verb verb| pronoun adjective verb| pronoun verb verb ptrn| pronoun verb verb adjective verb| pronoun verb verb adjective noun| pronoun verb verb adjective pronoun| pronoun verb ptrn| pronoun verb aw| pronoun verb adjective| pronoun verb pronoun| pronoun verb noun| pronoun ptrn | verb verb verb adjective| verb verb verb adjective verb| verb verb verb ptrn| verb verb verb pronoun| verb verb ptrn| verb verb adjective verb| verb verb adjective noun| verb verb adjective pronoun| verb verb noun adjective verb| verb verb noun verb ptrn| verb verb noun verb aw| verb verb noun verb adjective| verb verb noun ptrn| verb verb noun verb pronoun| verb adjective| verb adjective verb ptrn| verb adjective noun| verb adjective verb noun| verb adjective verb noun verb | adjective noun verb adjective verb| adjective noun verb ptrn| adjective noun verb pronoun| adjective noun adjective ptrn| adjective noun adjective aw| adjective noun adjective | adjective pronoun| adjective ptrn | conjunction

Left Factoring

The above grammar G is ambiguous. The parser generated from this kind of grammar is not efficient as it requires backtracking. To remove the ambiguity from the grammar the grammar productions are reconstructed by left factoring.

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:

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$$

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

$$A \rightarrow \alpha A'$$
$$A' \rightarrow \beta_1 \mid \beta_2$$

The above grammar is correct and is free from conflicts. After left factoring the grammar G, the reconstructed left factored grammar productions are as below:

S –> NP VP

NP –> noun NP1|pronoun NP2|modifier AP1|conjunction NP1| AP | NP2 |xp NP1 |tp NP1

NP1 –> conjunction noun |ip | pronoun NP2 | adjective | noun NP3 | tp | aw |ε

NP2 –> conjunction pronoun | ip | noun NP1 | adjective | pronoun NP3 | tp | ε

NP3 –> conjunction pronoun| aw | ε

AP –> adjective AP1

AP1 –> noun | adjective AP2 | pronoun | conjunction AP | ε

AP2 –> ptrn | ε

VP –> noun VP1| pronoun VP4 | verb VP2 | AP VP3 | conjunction

VP1 –> verb VP2 | adjective VP3 | pronoun noun |noun pronoun

VP2 –> verb VP3 | ptrn | aw | AP VP3 | ε

VP3 –> verb VP4 | ptrn | adjective VP5 | noun VP4 | ε

VP4 –> AP verb | verb VP2| ptrn | pronoun

VP5 –> verb| noun VP4| pronoun AP1

There have been lexical productions which are as follows:

noun –> রহিম । বিদ্যালয়ে। ছেলে। বই। ভাত। টাকা । কণে । সিংহ । ঘাস

pronoun –> আমি । আমরা । তুমি । সে । তারা

verb –> খাই । খেলে । করবে । পড়া । চলে । যাই

adjective –> ভাল । বলবান । গুণবান । একটু । দশ । অনেক । শখ । দ্রুত

conjunction –> এবং । ও । চেয়ে

modifier –> এ । একটি । একদিন । এই

ptrn –> না । নয়

Designing Bangla Parser

A parser for a grammar G is a program that takes a string as input and produces a parse tree as output if the string is a sentence of G or produces an error message indicating that the sentence is not according to the grammar G. The idea of predictive parser design is well understood in compiler design (Murshed, 1998). A predictive parser is proposed in (Hasan et al., 2011) that can recognize mistakes of Bengali sentences when there is no entry for terminal in parse table.

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 any string could be verified 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 considered invalid.

Computation of FIRST and FOLLOW

FIRST(α) be the set of terminals that begin the strings derived from α. If $\alpha \rightarrow \varepsilon$ then ε is also included in FIRST(α). The rules of computing FIRST (Aho, Sethi & Ullman, 2002) are as follows:

- If X is a terminal symbol ⇒ first(X) ={X}
- If X is a non-terminal symbol and $X \rightarrow \varepsilon$ is a production rule ⇒ ε is in first(X).
- If X is a non-terminal symbol and $X \rightarrow Y^1Y^2.Y^n$ is a production rule ⇒ first(X) = first(Y1).

According to the rules of computing FIRST(α) (Aho, Sethi & Ullman, 2002) the values for FIRST() are computed for the grammar G and are shown here:

FIRST(S) = {noun, pronoun, modifier, adjective, conjunction, xp, tp}

FIRST(NP) = {noun, pronoun, modifier, adjective, conjunction, xp, tp}

FIRST(NP1) = {conjunction, ip, pronoun, adjective, noun, tp, aw, ε }

FIRST(NP2) = {conjunction, ip, pronoun, adjective, noun, tp, ε }

FIRST(NP3) = {conjunction, aw, ε}

FIRST(AP) = {adjective}

FIRST(AP1) = {noun, adjective, pronoun, conjunction, ε}

FIRST(AP2) = {ptrn, ε }

FIRST(VP) = {noun, pronoun, verb, adjective, conjunction}

FIRST(VP1) = {verb, adjective, pronoun, noun}

FIRST(VP2) = {verb, ptrn, aw, adjective, ε}

FIRST(VP3) = {noun, verb, adjective, ptrn, ε}

FIRST(VP4) = {verb, ptrn, pronoun, adjective, ε}

FIRST(VP5) = {verb, noun, pronoun}

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). The rules for computing FOLLOW (Aho, Sethi & Ullman, 2002) of a non terminal *A* is as follows:

- If S is the start symbol ⇒ $ is in follow(S).
- If $A \rightarrow \alpha B\beta$ is a production rule ⇒ everything in first(β) is follow(B) except ε.
- If ($A \rightarrow \alpha B$ is a production rule) or ($A \rightarrow \alpha B\beta$ is a production rule and ε is in

first(β))⇒ everything in follow(A) is in follow(B).

According to the rules of computing FOLLOW(S) the values for FOLLOWS() are computed for the grammar G and are shown as follows:

FOLLOW(S) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction}

FOLLOW(NP) ={noun, verb, adjective, pronoun}

FOLLOW(NP1) = {noun, verb, adjective, pronoun}

FOLLOW(NP2) = {noun, verb, adjective, pronoun}

FOLLOW(NP3) = {noun, verb, adjective, pronoun}

FOLLOW(AP) = {noun, verb, adjective, pronoun, ptrn, modifier, ip, xp, tp, conjunction}

FOLLOW(AP1) = {noun, verb, adjective, pronoun, ptrn, modifier, ip, xp, tp, conjunction}

FOLLOW(AP2) = {noun, verb, adjective, pronoun, ptrn, modifier, ip, xp, tp, conjunction}

FOLLOW(VP) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction, $}

FOLLOW(VP1) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction, $}

FOLLOW(VP2) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction, $}

FOLLOW(VP3) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction, $}

FOLLOW(VP4) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction, $}

FOLLOW(VP5) = {noun, adjective, pronoun, modifier, ip, xp, tp, conjunction, $}

Parse Table Construction

Parse table is a two-dimensional array where each row is leveled with non terminal symbol and each column is marked with terminal or special symbol $. Each cell holds a production rule.

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 4 shows the resulting parse table for the grammar G constructed by applying the following rules:

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].

The table shows all the valid entries for the parse table. Entries not in this table are error entries. All other undefined entries of the parsing table are error entries.

Parse Tree Generation

A parse tree for a grammar G is a tree where the root is the start symbol for G, the interior nodes are the non terminals of G and the leaf nodes are the terminal symbols of G. The children of a node T (from left to right) correspond to the symbols on the right hand side of some production for T in G. Every terminal string generated by a grammar has a corresponding parse tree and every valid parse tree represents a string generated by the grammar. We store the parse table M using a

Table 4. Predictive parse table for the grammar G

	Noun	Pronoun	Modifier	Adjective	Verb	Conjunction	ptrn	ip	aw	xp	tp	$
S	NP VP	NP VP	NP VP	NP VP		NP VP		NP VP		NP VP	NP VP	
NP	noun NP1	pronoun NP2	modifier noun	AP		conjunction NP1		NP2	aw	xp NP1	tp NP1	
NP1	noun NP3	pronoun NP2	NP	Adjective	ε	conjunction VP2		ip			tp	
NP2	noun NP1	pronoun NP3		Adjective	ε	conjunction pronoun		ip			tp	
NP3	ε	ε		ε	ε	conjunction AP			aw			
AP				adjective AP1								
AP1	noun	pronoun	ε	adjective AP2	ε	conjunction AP	ε	ε		ε	ε	ε
AP2	ε	ε	ε	ε	ε	ε	ptrn		ε	ε	ε	ε
VP	noun VP1	pronoun VP4		AP VP3	verb VP2	conjunction						
VP1	noun pronoun	pronoun noun		adjective noun	verb VP2							
VP2	noun VP3	ε	ε	AP VP3	verb VP3	ε	ptrn	ε	aw	ε	ε	ε
VP3	noun VP4	ε	ε	adjective VP5	verb VP4	ε	ptrn	ε		ε	ε	ε
VP4		pronoun		AP verb	verb VP2	ε	ptrn	ε		ε	ε	ε
VP5	noun VP4	pronoun	AP1		Verb							

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 4) 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 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. Table 5 shows 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 $.

Managing Non-Traditional Forms

The structure of Bangla language may change in a sentence although the meaning does not change. For example, the sentence আমি ভাত খাই can also be written as আমি খাই ভাত or ভাত আমি খাই. The latter two forms are also correct and have the same meaning. Hence it is sometimes difficult to detect such reorganized non traditional forms in a single grammar. The proposed grammar can detect the the non traditional forms. For example, the grammar G detects the forms "আমি খাই ভাত" and "ভাত আমি খাই" are shown in Table 6 and

Table 5. Moves made by a Bangla parser on input "modifier noun noun verb" for correct sentence

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
$ VP2 verb	verb $	VP1-> verb VP2
$ VP2	$	Poped
$	$	VP2-> ε
$	$	Sentence is accepted

*Table 6. Moves made by a Bangla parser on input "pronoun verb noun" for correct sentence "*আমি খাই ভাত*"*

Stack	Input	Action
$ S	pronoun verb noun $	
$ VP NP	pronoun verb noun $	S->NP VP
$ VP NP2 pronoun	pronoun verb noun $	NP->pronoun NP2
$ VP NP2	verb noun $	poped
$ VP	verb noun $	NP2->zero
$ VP3 verb	verb noun $	VP->verb VP3
$ VP3	noun $	poped
$ noun	noun $	VP3->noun
$	$	poped
$	$	Sentence is accepted

*Table 7. Moves made by a Bangla parser on input "noun pronoun verb" for correct sentence "*ভাত আমি খাই*"*

Stack	Input	Action
$ S	noun pronoun verb $	
$ VP NP	noun pronoun verb $	S->NP VP
$ VP NP1 noun	noun pronoun verb $	NP->noun NP1
$ VP NP1	pronoun verb $	Poped
$ VP	pronoun verb $	NP1->zero
$ verb AP pronoun	pronoun verb $	VP->pronoun AP verb
$ verb AP	verb $	Poped
$ verb	verb $	AP->zero
$	$	Poped
$	$	Sentence is accepted

Figure 2. Parse tree for Bangla parser on input "modifier noun noun verb"

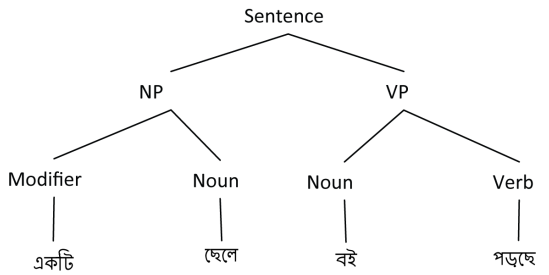

Figure 3. Parse tree for Bangla parser on input "pronoun verb noun"

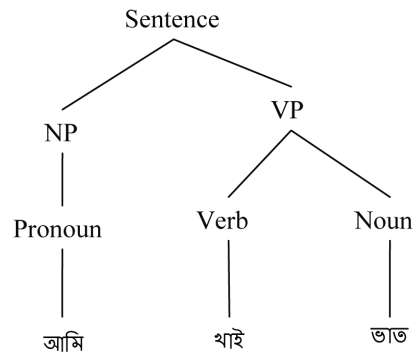

Figure 4. Parse tree for Bangla parser on input "noun pronoun verb"

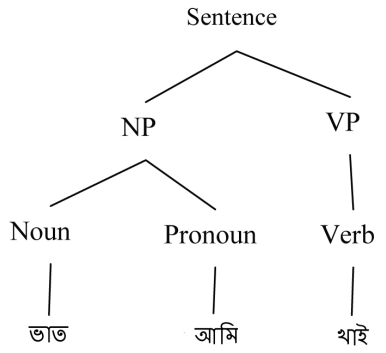

Table 8. Moves made by a Bangla parser on input "noun noun" for incorrect sentence

Stack	Input	Action
$ S	noun noun $	
$ VP NP	noun noun $	S->NP VP
$ VP NP1 noun	noun noun $	NP->noun NP1
$ VP NP1	noun $	Popped
$ VP NP3 noun	noun $	NP1->noun NP3
$ VP NP3	$	Popped
		Sentence is rejected

Table 7 respectively. Figure 3 and Figure 4 show the corresponding parse tree. Table 8 shows the moves for an incorrect sentence "রহিম বিদ্যালয়ে" of the form *"noun noun"*. The sentence is rejected because there is no verb in the sentence and such types of productions are not presented in the grammar G (see Algorithm 1).

Algorithm 1 is for parsing which uses the parse table *M (*Table 4) to produce a Bangla parser for the input of a Bangla sentence. Figure 2 shows the parse tree generated by the grammar G for the input একটি ছেলে বই পড়ছে of the form modifier *noun noun verb*.

PERFORMANCE ANALYSIS OF DESIGNED BANGLA PARSER

Some input sentences are considered to be used for performance analysis of the designed parser. There are three types of sentences used for this purpose:

1. Simple and traditional form
2. Some non-traditional form
3. Paragraphs

Algorithm 1. Parsing with table M

```
1.  Set Input Pointer(IP) to point to the first word of w;
2.  Set X to the top stack word;
3.  While (X! = $)
     Begin   /* stack is not empty */
       a.  If  X is a terminal,  pop the stack and advance IP;
       b.  If X is a Non terminal and  M[X,IP] has  the production  X → Y1Y2…Yk
             Output the production X -> YlY2 -Yk;
               Pop the stack;
               Push Yk, Yk-1,. . ., Yl onto the stack, with Yl on top;
       c.  If X=$, Sentence is Accepted.
     End
```

The paragraphs are collected from newspaper. By non-traditional form we mean the same meaning of another sentence having structural similarity. For example, "আমি ভাত খাই" and "খাই আমি ভাত" sentences are approximately similar but later on is rarely used. Table 4 shows the success rate of our proposed grammar.

- **Input Sentences:**

তিনি কি ভাল না?, আমি ভাত খাই । আমি কি ভাত খাই?, আমি কি ভাত খাই না? "বাহ! পাখিটি তো খুব সুন্দর"। শিতে আমরা খুব কষ্ট পাই । তোমরা আগামীকাল এসো । আমি, তুমি ও সে বাড়ি যাই । আগামীকাল কি তুমি আসবে?

- **Input Paragraphs:**

মাঝরাতে হঠাৎ করেই আপনার অসুখ করল। চিকিৎসক বা হাসপাতাল আশপাশে নেই।ভয় পাওয়ার কোন কারণ নেই । যদি আপনার হাতের কাছেই মোবাইল পান,তবে ডায়াল করুন হেল্প লাইনে । শীঘ্রই আপনি পেয়ে যাবেন দরকারি পরামর্শ।

আমরাই একমাত্র জাতি যারা ভাষার জন্য যুদ্ধ করেছি । মায়ের ভাষা বাংলাকে ছিনিয়ে এনেছি । ভাষার এই যুদ্ধ এখানেই বন্ধ হয়নি । বাংলাকে আলাদা করে প্রতিষ্ঠিত করার জন্য শুরু হয়েছে প্রযুক্তির ব্যাবহার । সেই যুদ্ধে ঝাঁপিয়ে পড়েছে মেধাবী জনতা।

একসময় বাংলা ভাষা যে হুমকির মুখে ছিল, তার মূলে ছিল উপনিবেশিক শাসন।যা ছিল সংখ্যাগুরুর ওপর চাপিয়ে দেওয়া সংখ্যালঘুর শাসন । তার বিরুদ্ধে লড়াই করেছিল বাংলা ভাষাভাষী বেশির ভাগ মানুষ । কিন্তু একটি ভাষার শক্তি এর ব্যাবহারকারীদের মোট শক্তির সমান । ফলে বাংলা ভাষার বিরুদ্ধকারীরা পরাজিত হয়েছে । যেহেতু তারা সংখ্যায় বেশি ছিল না।

From Table 9, it is clearly viewed that the parser is about 70-80% correct.

FUTURE RESEARCH DIRECTIONS

In this chapter, a context free grammar for Bangla language is described and hence a Bangla parser based on the grammar is developed. The approach is very much general to apply in Bangla Sentences and the method is well accepted for parsing a language of a grammar. The structural representation that has been built can cover the maximum simple, complex, and compound sentences. But the sentences composed of idioms and phrases are beyond the scope of this chapter.

There are lots of research scopes on the fields of Bangla Natural Language Parser. The future development may include the following.

1. More extensive CFGs with the addition of more constituents for parsing complex and compound sentences of Bangla NL more efficiently.
2. Efficient algorithm for Bangla NL Parser to solve the problem of ambiguity.
3. Parser for semantic analysis of Bangla natural sentences.

Fusion of top-down and bottom-up parsing technique could result in a more robust and powerful parsing technique.

CONCLUSION

It is well established that parsing natural language text is much more difficult than strictly defined computer languages. One reason is that grammars for natural languages are often com-

Table 9. Success rate for different Bangla sentences

Types of Sentences	Total No. of Sentences per Paragraph (I)	Correctly Detected (D)	Acceptance Rate A=(D/I)*100%
Traditional	120	100	83.33%
Nontraditional	80	58	72.5%
Paragraph	25	18	72%

plex, ambiguous, and specified by collections of examples rather than complete formal rules. The aim of this Chapter was to Design and Develop algorithms for Bangla Parser, which can parse Bangla language. Parsing is a process of transforming natural language into an internal system representation, which can be trees, dependency graphs, frames or some other SR (Structural Representation). If a natural language is successfully parse then grammar checking from this language becomes easy. Parsing a sentence that involves finding a possible legal structure for sentence and finally gets an SR. An SR generally graphic object and this representation cannot be deals with computer. The standard representation of an SR is a list that are one of the data structure that can be implemented and manipulated very easily within a computer. Our proposed CFG rules can be assigned all types of Bangla sentences into an SR. To represents a complex and a compound sentence into an SR, we used the decomposition technique. Further modify and extending the CFG rules (with the addition of sentences consisting of idioms and phrases, double word, change of voice, and narration), we can able to represents the all kinds of Bangla sentences which are used as an input of an MT engine to produced other equivalent sentences.

REFERENCES

Aho, A. V., Sethi, R., & Ullman, J. D. (2002). *Compilers principles, techniques and tools*. New York: Pearson Education.

Ali, M. N. Y., Ripon, S., & Allayear, S. M. (2012). UNL based Bangla natural text conversion: Predicate preserving parser approach. *International Journal of Computer Science Issues*, *9*(3), 259–265.

Anwar, M. M., Anwar, M. Z., & Bhuiyan, M. A. A. (2009). Syntax analysis and machine translation of Bangla sentences. *International Journal of Computer Science and Network Security*, *9*(8), 317–326.

Anwar, M. M., Shume, N. S., & Bhuiyan, M. A. A. (2010). Structural analysis of Bangla sentences of different tenses for automatic Bangla machine translator. *International Journal of Computer Science and Information Security*, *8*(9).

Dasgupta, S., & Ng, V. (2006). Unsupervised morphological parsing of Bengali. *Language Resources and Evaluation*, *40*, 311–330. doi:10.1007/s10579-007-9031-y.

Faridee, A. Z. M., & Tyers, F. M. (2009). Development of a morphological analyzer for Bengali. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation,* (pp. 43-50). IEEE.

Ghosh, A., Das, A., Bhaskar, P., & Bandyopadhyay, S. (2010). *Bengali parsing system*. Paper presented at ICON NLP Tool Contest 2010.

Haque, M. N., & Khan, M. (2005). Parsing Bangla using LFG: An introduction. *BRAC University Journal*, *2*(1), 105–110.

Hasan, K. M. A., Mahmud, A., Mondal, A., & Saha, A. (2011). Recognizing Bangla grammar using predictive parser. *International Journal of Computer Science & Information Technology*, *3*(6). doi: doi:10.5121/ijcsit.2011.3605.

Hasan, K. M. A., Mondal, A., & Saha, A. (2010). A context free grammar and its predictive parser for Bangla grammar recognition. In *Proceedings of International Conference on Computer and Information Technology,* (pp. 87 – 91). IEEE.

Hoque, M. M., & Ali, M. M. (2003). A parsing methodology for Bangla natural language sentences. In *Proceedings of International Conference on Computer and Information Technology*, (277-282). IEEE.

Hoque, M. M., & Ali, M. M. (2004). Context-sensitive phrase structure rule for structural representation of Bangla natural language sentences. In *Proceedings of International Conference on Computer and Information Technology*, (pp. 615-620). IEEE.

Hoque, M. M., Rahman, M. J., & Dhar, P. K. (2007). Lexical semantics: A New approach to analyze the Bangla sentence with semantic features. In *Proceedings of the International Conference on Information and Communication Technology*, (pp. 87 – 91). IEEE.

Joshi, S. (n.d.). *Selection of grammatical and logical functions in Marathi*. (PhD Thesis). Stanford University. Palo Alto, CA.

Mehedy, L., Arifin, N., & Kaykobad, M. (2003). Bangla syntax analysis: A comprehensive approach. In *Proceedings of International Conference on Computer and Information Technology (ICCIT),* (pp. 287-293). ICCIT.

Murshed, M. M. (1998). Parsing of Bengali natural language sentences. In *Proceedings of International Conference on Computer and Information Technology (ICCIT),* (pp. 185-189). ICCIT.

Saha, G. K. (2006). Parsing Bengali text: An intelligent approach. *ACM Ubiquity*, *7*(13), 1–5. doi:10.1145/1132512.1127026.

Selim, M. R., & Iqbal, M. Z. (1999). Syntax analysis of phrases and different types of sentences in Bangla. In *Proceedings of International Conference on Computer and Information Technology (ICCIT)*, (pp. 175-186). ICCIT.

Sengupta, P., & Chaudhuri, B. B. (1997). A delayed syntactic-encoding-based LFG parsing strategy for an Indian language – Bangla. *Computational Linguistics*, *23*(2), 345–351.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of 33rd Annual Meeting of the ACL*, (pp. 189-196). ACL.

KEY TERMS AND DEFINITIONS

Bottom-Up Parsing: Construction starts at the leaves and proceeds towards the root, i.e. this method works from the sentence to sentence symbol.

Context-Free Grammar (CFG): A set of recursive rewriting rules called productions used to generate string patterns.

FIRST(): FIRST(α) be the set of terminals that begin the strings derived from α. If $\alpha \rightarrow \varepsilon$ then ε is also included in FIRST(α).

FOLLOW(): 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 FOLLOW(A).

Parse Tree: A tree where the root of the tree is the start symbol for the grammar, the interior nodes are the non-terminals of the grammar, the leaf nodes are the terminals of the grammar and the children of a node starting from the left to the right correspond to the symbols on the right hand side of some production for the node in the grammar.

Parser: A parser analyzes the sequence of symbols presented to it based on the grammar.

Parsing: A method where a parser algorithm is used to determine whether a given input string is grammatically acceptable or not for a particular language.

Top-Down Parsing: 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. Parsers, which are designed using Top-down parsing method, are known as top-down parser.