

Context-Free Grammar for Bangla

Ayesha Binte Mosaddeque & Nafid Haque
Department of Computer Science & Engineering
BRAC University
66 Mohakhali, Dhaka-1212.
Bangladesh
Email: {lunaticbd, nafid99}@yahoo.com

Abstract: This paper proposes a way of producing context-free grammar for Bangla Language. The grammar proposed by this paper can successfully parse a number of sentences having seven to eight words of length, extracted randomly from a newspaper article. We have defined a tag set and according to the tag set, we tagged the corpus. In the next step of producing CFG, we defined the constituents and finally the rules were generated. We encountered difficulties choosing the parser. Bottom up parser had the trouble of wrong categorization. On the other hand top down parser had the problem of infinite loop caused by a left recursive rule. However we have finally used top down parser to parse the sentences with our generated grammar.

1. Introduction

A context-free grammar (CFG) is a set of recursive rewriting rules called productions used to generate string patterns. A CFG has four components. And they are 1) a set of terminal symbols that are characters that appear in the strings generated by the grammar, 2) a set of non-terminal symbols that are placeholders for patterns of terminal symbols that can be generated by the non-terminal symbols, 3) a set of productions that are used to replace the non-terminals with other non-terminals or terminals and 4) a start symbol for the grammar [1]. A formal language is context-free if there is a context-free grammar that generates it.

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 [2]. And parsing is a method where a parser algorithm is used to determine whether a given input string is grammatical or not for a given grammar.

Bangla, being the successor to Sanskrit, Pali and Prakrit, is an Indo-Aryan language of South Asia. With over 200 million native speakers, Bangla is the fourth largest language of the world [3]. Even being in the top end of the rank list Bangla Language does not yet have computerized grammar checking methods for a given Bangla sentence. Thus, this paper highlights the process of generating context free grammar for the Bangla language from scratch.

2. Previous work

There has been no organized formal initiative for generating context-free grammars for the 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 [5, 6].

3. Proposed CFG for Bangla

As there were no formal initiatives in generating context-free grammars for Bangla, we had to start from the beginning. We took a corpus of 10 sentences with an average length of the sentences being eight words. The corpus can be found in the appendix. The sentences of our corpus were taken randomly from an article of the newspaper and were not specific to a certain type of sentence. After we selected our corpus we tagged words of the corpus with their respective parts-of-speech (POS) tags. The (POS) tag set used to tag the corpus is given below:

a. Tag set Description

#	Level 1	Level 2	Definition	Symbol	Examples
1	Noun	Proper Noun	A proper noun is a noun that is the name of a specific individual, place or object.	np	মণিসিংহ, আজাদ
2		Common Noun	A noun that can be preceded by the definite article and that represents one or all of the members of a class	nc	জীবন, টক্ক
3		Proper Noun Genitive	A proper noun that is marked with the genitive case markers র/এর	np _{gen}	মণিসিংহের
4		Common Noun Genitive	A common noun that is marked with the genitive case markers – র/এর	nc _{gen}	জমির

5		Proper Noun Accusative	A proper noun that is marked with the accusative case marker – কে	npacc	নাফিদকে
6		Common Noun Accusative	A common noun that is marked with the genitive case marker – কে	ncacc	ভাইকে
7		Proper Noun Locative	A proper noun that is marked with the prototypical locative case markers – এ/য়/তে/য়ে	nploc	
8		Common Noun Locative	A common noun that is marked with the prototypical locative case markers – এ/য়/তে/য়ে	ncloc	পুরুষে
9	Pronoun	Pronoun	A function word that is used in place of a noun in a sentence	pro	তিনি
10		Pronoun Accusative	Personal pronoun in accusative case	proacc	তাকে
11		Pronoun Possessive	Personal pronoun in genitive (possessive) case	proposs	তাঁর, এঁর
12	Verb	Finite Verb	A finite verb is a verb form that occurs in an independent clause.	vf	করেছিল, জানালাম
13		Non Finite Verb	A nonfinite verb is a verb that is not fully inflected for categories that are marked inflectionally in a language.	vnf	করে, জানতে
14		Existential Verb		ve	ছিল, হলো
15	Adjective	Adjective		adj	প্রবাদতুল্য, পরিণত
16	Adverb	Temporal Adverb	Adverb of time	advt	সালে
17	Postposition	Postposition		postp	পরে, থেকে

18	Indeclinable	Negative Particle	Word that denotes negative	prtn	না
19	Ordinal	Ordinal	A numeral belonging to class whose member designate positions in a sequence	ord	প্রথম
20		Ordinal Locative	An ordinal marked by the locative case marker – ৗ	ordloc	প্রথমে
21	Cardinal	Number		num	১৯৫০

After tagging the corpus, each sentence was broken down into their constituent parts. The constituents were identified keeping in mind that each constituent should represent a complete meaning for a context. The constituents of the sentences found out are shown in the table below:

b. Sentence Constituents

#	Constituent	Symbol	Definition	Productions	Examples
1	Noun Phrase	NOUNPH	A noun phrase is a constituent that has a noun as the head word. It may have a word too. It might contain a modifier or an adjective preceding the head noun.	<i>NOUNPH -> MOD ADJ NC MOD NC ADJ NC np propooss NC np NACC PRO</i>	কমরেড মণিসিংহের বর্ণবহুল জীবন, আজাদ
2	Verb Phrase	VERBPH	Verb Phrase is a constituent that has a verb as the head word. It might contain a VCR*, finite verb following a noun phrase, an accusative pronoun followed by a locative noun followed by a finite verb etc.	<i>VERBPH -> PROACC NLOC VF PROACC NLOC VCR ORDLOC NOUNPH NC VC NACC NC VNF NC VF NOUNPH VF VCR POSTPH VCR PRTN NC PRTN VNF</i>	তাঁর চিরকুট পেলাম, প্রথমে নিজেদের জমির টুক মাফ করে দেন

3	Postpositional Phrase	POSTPH	A postpositional phrase is a constituent that has a postposition as the head word. It may contain a common noun or a possessive pronoun followed by a postposition.	<i>POSTPH -> NC POSTP proposs POSTP</i>	কয়েকদিন পরে, এর থেকে
4	Adverbial Phrase	ADVPH	An adverbial phrase is a constituent that has an adverb as the head word. It may contain a numeral string denoting a year followed by a temporal adverb.	<i>ADVPH -> NUM ADVT</i>	১৯৫০ সালে
5	Locative Noun	NLOC	It contains all the nouns (both common and proper) that end with 'ট' and also adjective followed by NLOC	<i>NLOC -> NCLOC NPLOC ADJ NLOC</i>	পুরুষে
6	Accusative Noun	NACC	It contains all the nouns (both common and proper) that end with 'ক' and also a proper noun followed by another NACC.	<i>NACC -> NCACC NPACC np NACC</i>	ভাইকে
7	Modifier	MOD	Modifier is usually used in noun phrases followed by a noun or an adjective. It modifies a noun or an adjective. Modifier may contain a genitive noun, a noun followed by another modifier, a possessive pronoun followed by another modifier, a noun phrase followed by another modifier, a genitive noun followed by a possessive pronoun, or a common noun followed by a genitive common noun.	<i>MOD -> NCGEN NPGEN np MOD proposs MOD NOUNPH MOD nc MOD npgen proposs NC NCGEN</i>	নিজদের জমির, জমিদার বংশের
8	Compound Verb	VC	It contains a non finite verb followed by a finite verb.	<i>VC -> VNF VF</i>	করে দেন
9	Compound Root Verb	VCR	It contains a finite verb preceded by an adjective or a common noun.	<i>VCR -> ADJ VF NC VF</i>	পরিণত করেছিল

After identifying the constituents for each sentence, separate context-free grammar productions were generated for each sentence. When all the production rules were found out, they were all merged together to form an optimized generic context-free grammar for the corpus. The final context-free grammar produced is shown below:

Grammatical productions.

S -> NOUNPH VERBPH | VERBPH | POSTPH S | ADVPH S | VERBPH
VERBPH

NOUNPH -> MOD ADJ NC | MOD NC | ADJ NC | np | proposs NC | np
NACC | PRO

VERBPH -> PROACC NLOC VF | PROACC NLOC VCR | ORDLOC NOUNPH
NC VC | NACC NC VNF NC VF | NOUNPH VF | VCR | POSTPH VCR PRTN | NC
PRTN VNF

POSTPH -> NC POSTP | proposs POSTP

ADVPH -> NUM ADVT

NLOC -> NCLOC | NPLOC | ADJ NLOC

NACC -> NCACC | NPACC | np NACC

MOD -> NCGEN | NPGEN | np MOD | proposs MOD | NOUNPH MOD |
nc MOD | npgen proposs

NC -> nc

VF -> vf | ve

VC -> VNF VF

VCR -> ADJ VF | NC VF

VNF -> vnf

ADJ -> adj

ADVT -> advt
NCLOC -> ncloc
NPLOC -> nploc
NCACC -> ncacc
NPACC -> npacc
PROACC -> proacc
NCGEN -> ncgen
NPGEN -> npgen
ORDLOC -> ordloc
POSTP -> postp
PRO -> pro
NUM -> num
PRTN -> prtn

Lexical productions.

nc -> 'জীবন' | 'টস্ক' | 'মাফ' | 'চিঠি' | 'বিষয়টা' | 'চিরকুট' | 'কয়েকদিন' | 'গ্রেপ্তার' | 'ফসল' | 'রেহাই' | 'পরিবারও' | 'মালিক'

np -> 'কমরেড' | 'মণিসিংহ' | 'আজাদ'

npgen -> 'মণিসিংহের'

ncgen -> 'poribarar' | 'জমির'

ncacc -> 'jomiderke' | 'ভাইকে'

npacc -> 'o'

ncloc -> 'পুরুষে'

nploc -> 'o'

proacc -> 'তাকে'

postp -> 'পরে' | 'থেকে'

pro -> 'তিনি'

proposs -> 'নিজেদের' | 'তঁরা' | 'এরা' | 'নিজের'

vnf -> 'করে' | 'লিখে' | 'ফলেও'

vf -> 'করেছিল' | 'pari' | 'shunechhila' | 'korechhilen' | 'দেন' | 'জানালাম' | 'পেলাম'

ve -> 'হন' | 'ছিল'

adj -> 'প্রবাদতুল্য' | 'duroborti' | 'বর্ণবহুল' | 'পরিণত'

adv -> 'সালে'

ord -> 'প্রথম'

ordloc -> 'প্রথমে'

num -> '১৯৩০' | '১৯৫০'

prtn -> 'না'

4. Tools used to test the grammar automatically

After generating the context-free grammar given in the previous section, the Shift-Reduce Parser of NLTK_LITE was used to test the grammar. Shift-reduce parsing is the simplest kind of bottom-up parsing. Here the parser takes in 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 [4]. An example is shown below:

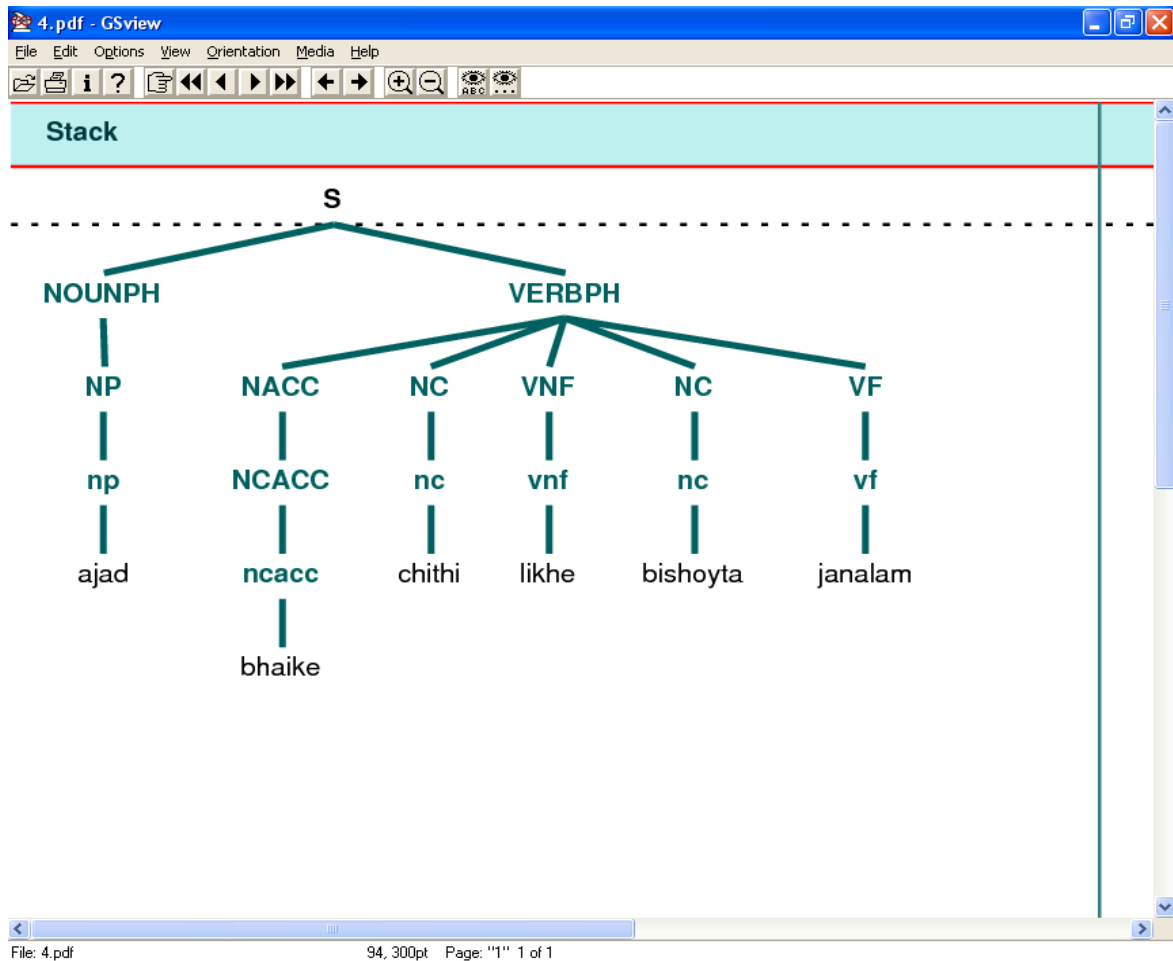


Fig 1: A wrong categorization of bottom up parser

Recursive Descent Parsing is another way to produce parse trees for an input sentence. Recursive Descent Parser of NLTK_LITE shows graphically how recursive descent parsing is done. 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 [4]. The problem is highlighted below:

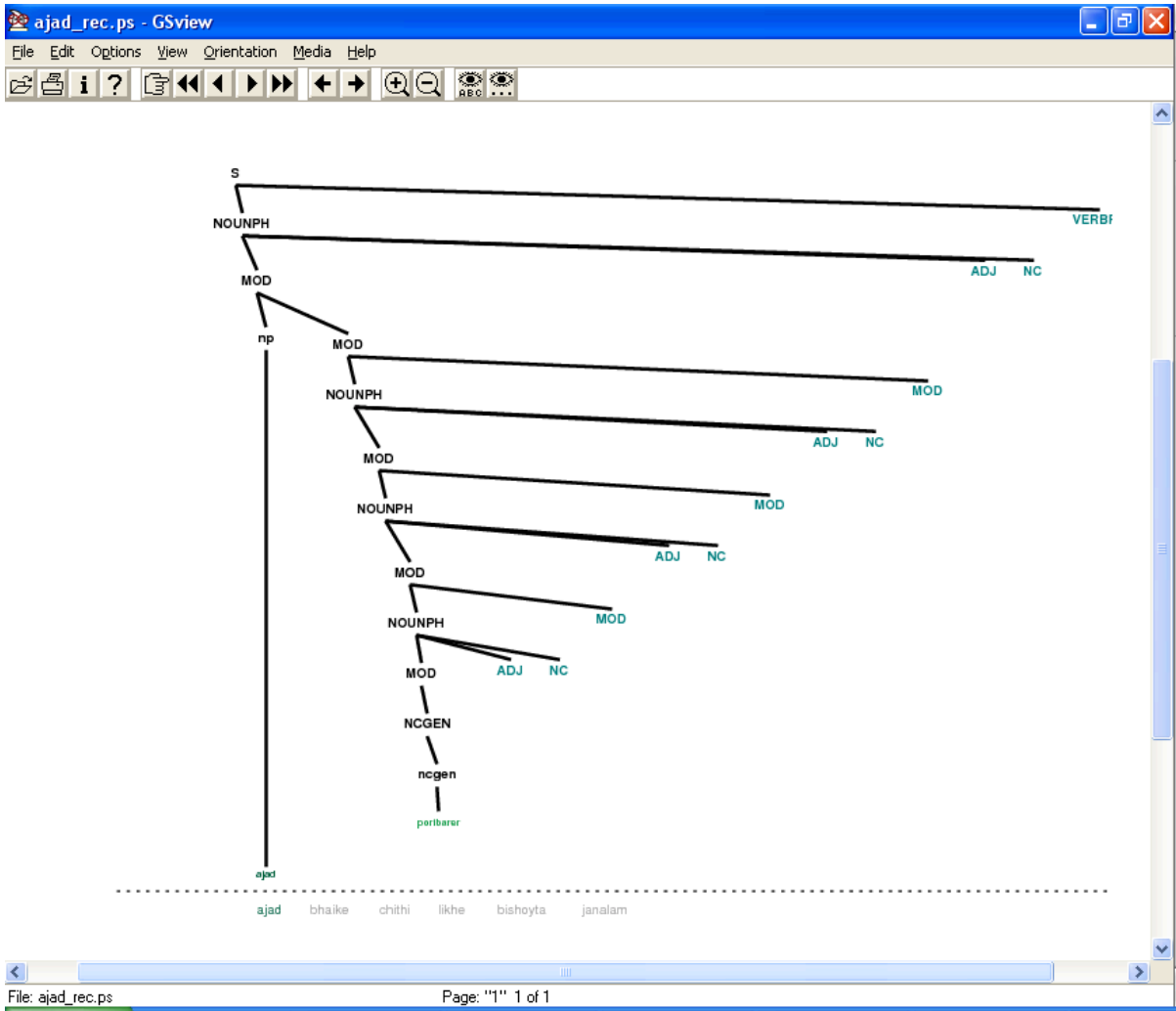
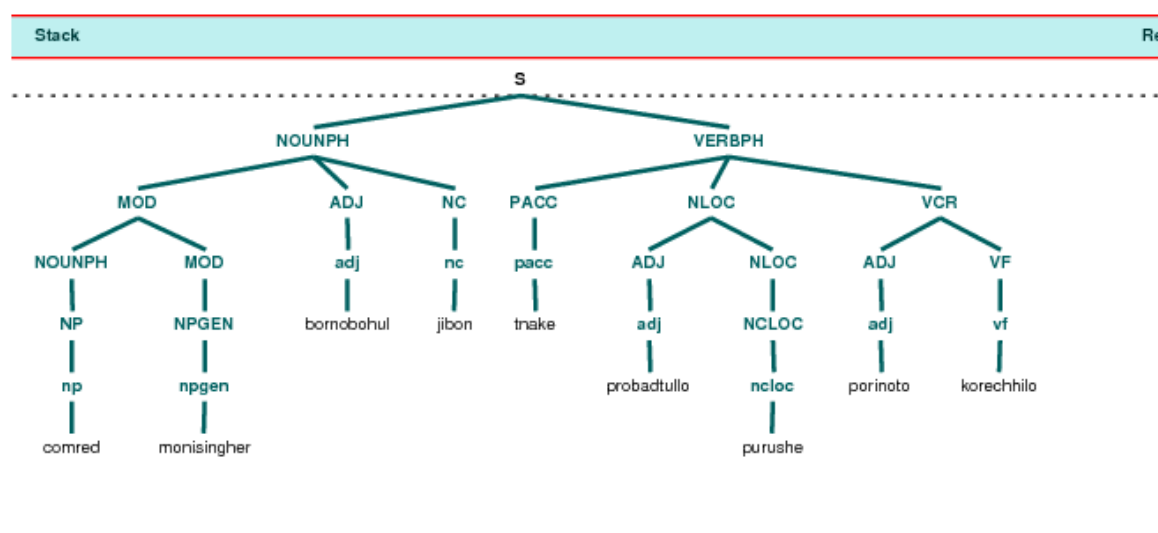


Fig 2: An infinite loop caused by recursive rules in RD parser

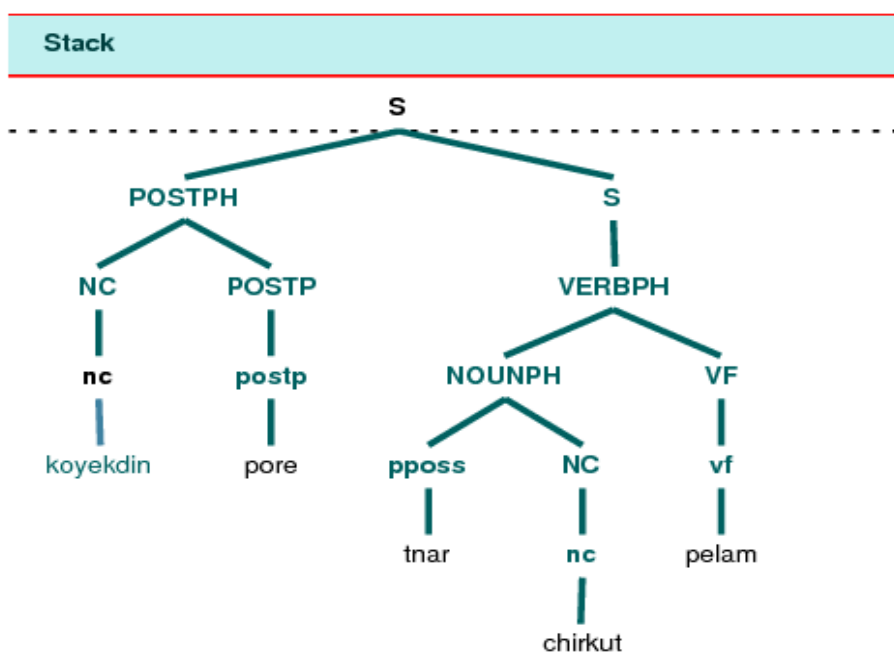
An advantage for Recursive Decent method over the Shift-reduce method is that, the parser can be forced to choose a specific production to continue parsing. Thus, this makes it easier to produce correct parse tree of a sentence in a given grammar which may not be possible with a shift-reduce parser for the same grammar.

5. The Parse trees for the sentences of the corpus

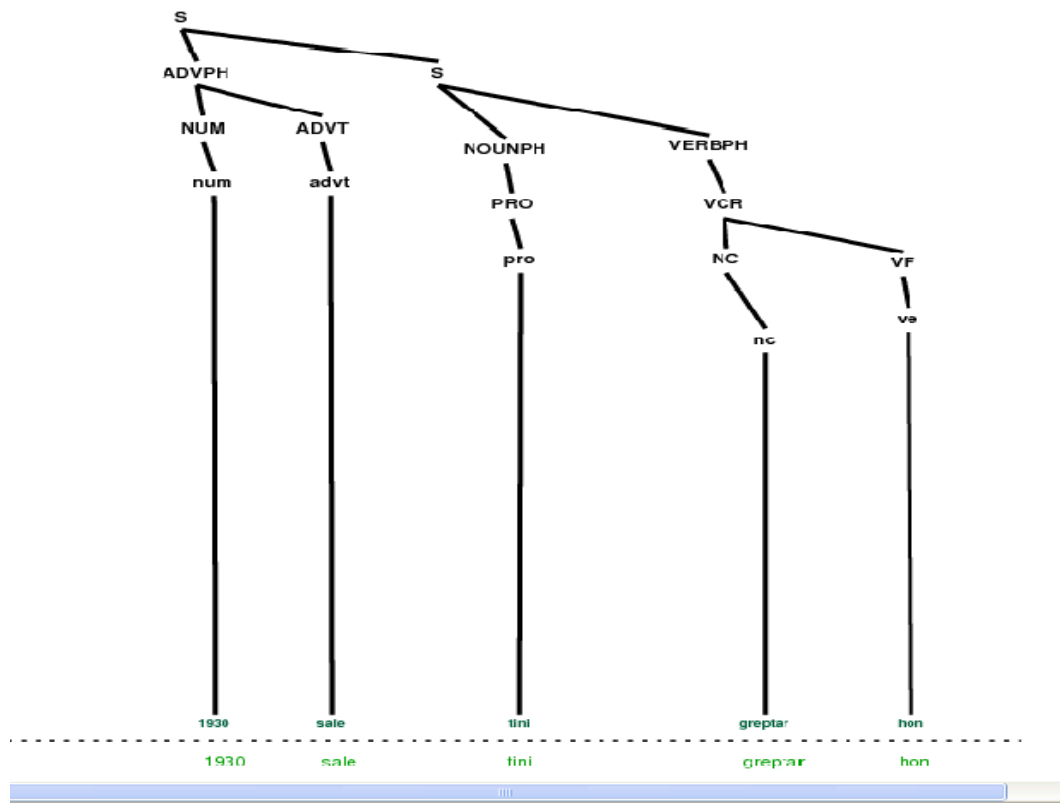
All the parse trees shown below have been produced using the Recursive Decent parser of the NLTK_LITE package. The parser has been modified to support Bangla Font and the CFG produced in section 3.



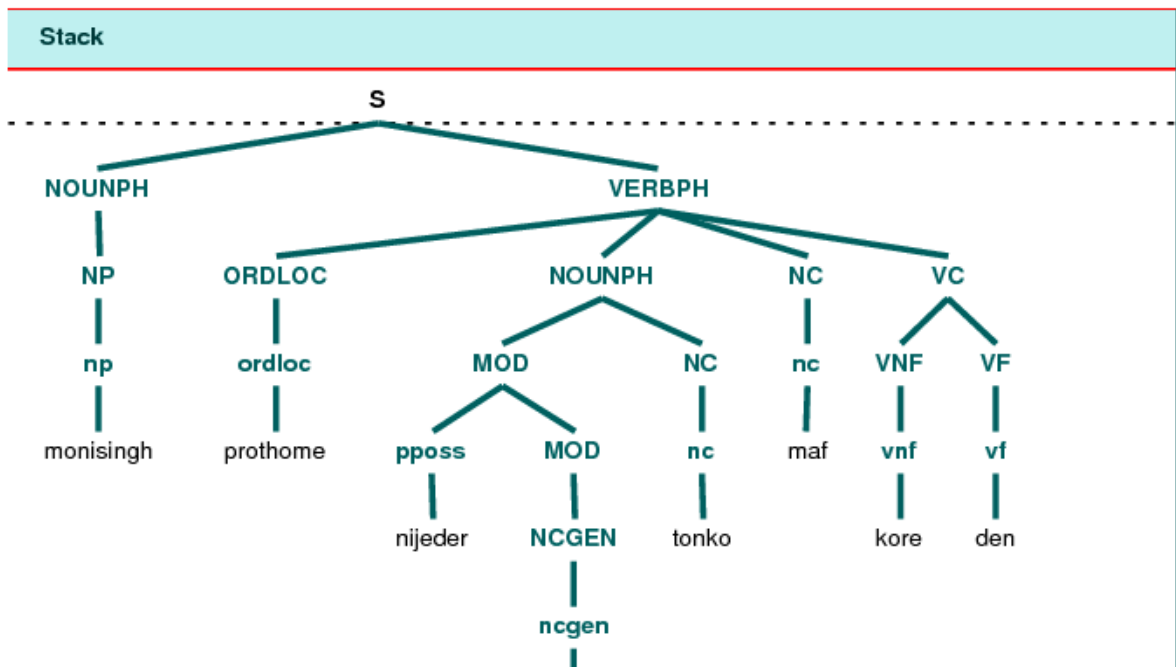
Tree 1



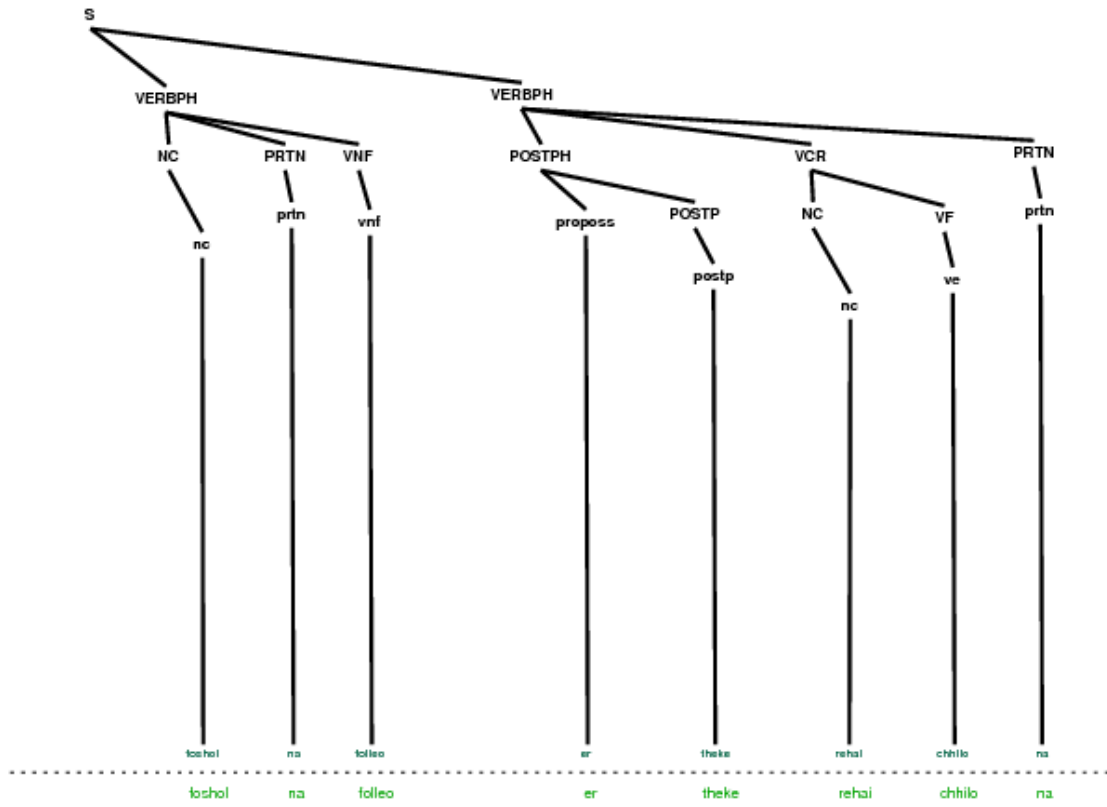
Tree 2



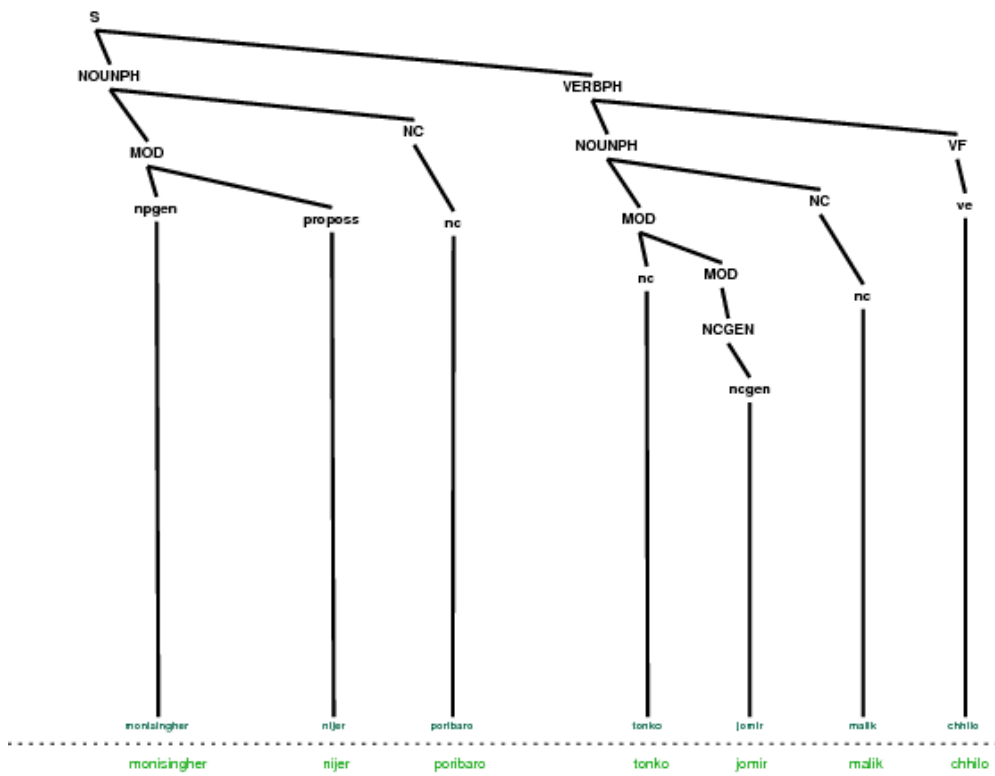
Tree 3



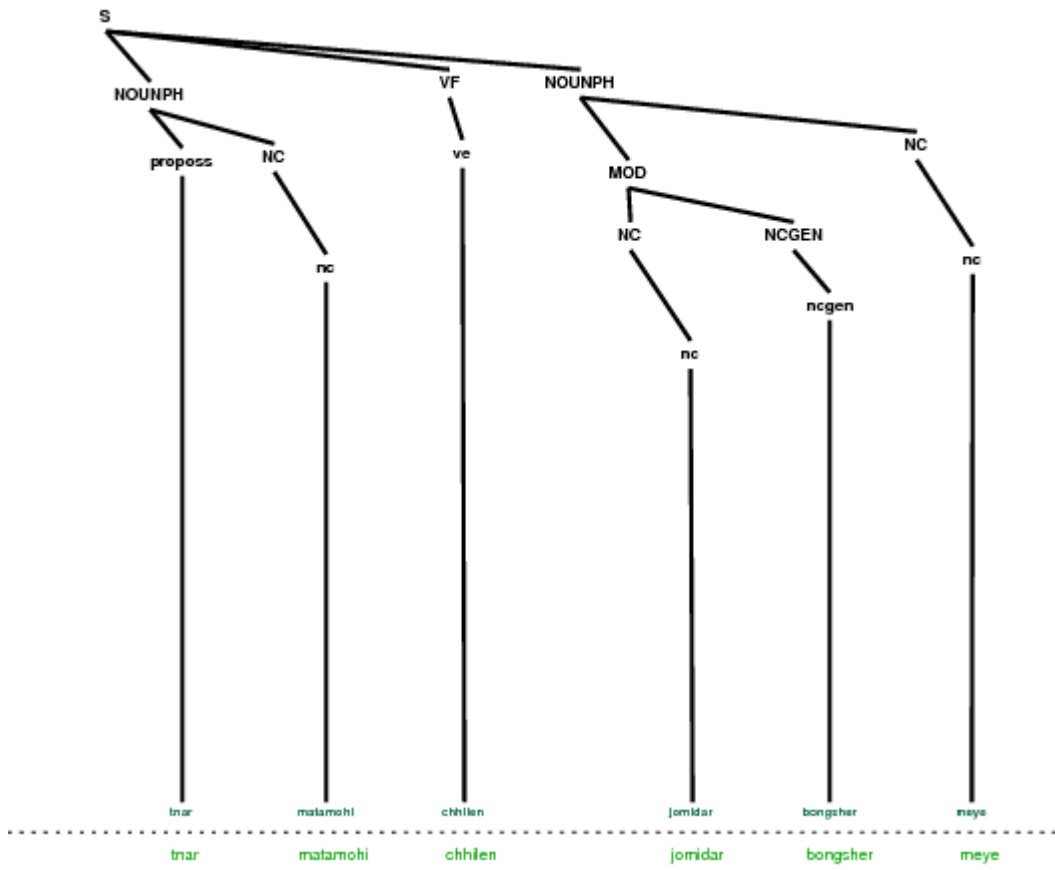
Tree 4



Tree 5



Tree 6



Tree 7

6. Conclusion

The grammar proposed by this paper is limited to the scope of a specific corpus. The grammar that this paper proposes has another potential problem of not providing an automated parsing. We have to supervise the way that a sentence should be parsed in, using ‘step’ and ‘backtrack’ methods of the parser. Among the 10 sentences of our corpus only one can be parsed automatically by the parser. As a future improvement if we can organize the order of the productions in the parser then we can hope that more sentences can be parsed automatically. However, we intend to extend this grammar to recognize and also to generate generic Bangla sentences.

7. Appendix

a. Corpus

1. কমরেড মণিসিংহের বর্ণবহুল জীবন তাঁকে প্রবাদতুল্য পুরুষে পরিণত করেছিল
2. আজাদ ভাইকে চিঠি লিখে বিষয়টা জানালাম
3. কয়েকদিন পরে তাঁর চিরকুট পেলাম
4. ১৯৩০ সালে তিনি গ্রেপ্তার হন
5. মণিসিংহ প্রথমে নিজেদের জমির টক্ক মাফ করে দেন
6. ফসল না ফললেও এর থেকে রেহাই ছিল না
7. মণিসিংহের নিজের পরিবারও টক্ক জমির মালিক ছিল
8. তাঁর মাতামহী ছিলেন জমিদার বংশের মেয়ে
9. নবাবপুর রোডে ক্যাপিটাল রেস্টুরেন্টে বসে চা খাচ্ছি
10. ১৯৫০ সালে এ প্রথা রদ হয়

b. Tagged Corpus

1. কমরেড/**np** মণিসিংহের/**np**gen বর্ণবহুল/**adj** জীবন/**nc** তাঁকে/**proacc** প্রবাদতুল্য/**adj** পুরুষে/**ncloc** পরিণত/**adj** করেছিল/**vf**
2. আজাদ/**np** ভাইকে/**ncacc** চিঠি/**nc** লিখে/**vnf** বিষয়টা/**nc** জানালাম/**vf**
3. কয়েকদিন/**nc** পরে/**postp** তাঁর/**proposs** চিরকুট/**nc** পেলাম/**vf**
4. ১৯৩০/**num** সালে/**adv**t তিনি/**pro** গ্রেপ্তার/**nc** হন/**ve**
5. মণিসিংহ/**np** প্রথমে/**ordloc** নিজেদের/**proposs** জমির/**nc**gen টক্ক/**nc** মাফ/**nc** করে/**vnf** দেন/**vf**
6. ফসল/**nc** না/**prtn** ফললেও/**vnf** এর/**proposs** থেকে/**postp** রেহাই/**nc** ছিল/**ve** না/**prtn**
7. মণিসিংহের/**np**gen নিজের/**proposs** পরিবারও/**nc** টক্ক/**nc** জমির/**nc**gen মালিক/**nc** ছিল/**ve**
8. তাঁর/**proposs** মাতামহী/**nc** ছিলেন/**vnf** জমিদার/**nc** বংশের/**nc**gen মেয়ে/**nc**
9. নবাবপুর/**np** রোডে/**ncloc** ক্যাপিটাল/**np** রেস্টুরেন্টে/**ncloc** বসে/**vnf** চা/**nc** খাচ্ছি/**vf**
10. ১৯৫০/**num** সালে/**adv**t এ/**dp** প্রথা/**nc** রদ/**adj** হয়/**ve**

c. Tutorial to use the Bangla supported Recursive Decent parser

To generate the parse tree of the sentences in the corpus using the produced grammar, the Recursive Decent Parser has been used. The Recursive Decent Parser of NLTK_LITE has been modified to support Unicode characters. The Recursive Decent Parser of NTK_LITE can be started by the following command:

```
>>> from nltk_lite.draw.rdparser import demo
>>> demo()
```

After the above command a window will open where the parse tree for a given sentence will be shown. From the Edit menu of the window, using the Edit Text option the user can enter a new sentence to be parsed by the parser. The Edit Grammar option can be used to put a new grammar for the parser.

8. Reference:

1. Lecture Note CSC173, available online at http://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/cfg.html
2. Lecture Note CSC173, available online at http://www.cs.rochester.edu/~nelson/courses/csc_173/grammars/parsetrees.html
3. Wikipedia entry of Bangla Language, available online at http://en.wikipedia.org/wiki/Bangla_Language
4. Steven Bird, Ewan Klein and Edward Loper, Parsing, NLTK_LITE Documentation.
5. হুমায়ূন আজাদ, বাকত্বেব
6. রফিকুল ইসলাম, ভাষাত্বেব