International Journal of Computer and Communication Technology

Volume 1 | Issue 2 Article 6

April 2010

# Context Free Grammar (CFG) Analysis for simple Kannada sentences

B M. Sagar
*Asst Prof, Information Science, RVCE Bangalore, India*, sagar.bm@gmail.com

Dr. Ramakanth Kumar P
*Professor & Head, RVCE Bangalore, India*, ramakanthkp@rvce.edu.in

Dr. Shobha G
*Professor, RVCE, Bangalore, India*, shobhag@rvce.edu.in

Recommended Citation
Sagar, B M.; P, Dr. Ramakanth Kumar; and G, Dr. Shobha (2010) "Context Free Grammar (CFG) Analysis for simple Kannada sentences," *International Journal of Computer and Communication Technology*: Vol. 1 : Iss. 2 , Article 6.
DOI: 10.47893/IJCCT.2010.1033
Available at: https://www.interscience.in/ijcct/vol1/iss2/6

# Context Free Grammar (CFG) Analysis for simple Kannada sentences

B M Sagar
Asst Prof, Information Science, RVCE
Bangalore, India
sagar.bm@gmail.com

Dr. Shobha G1, Dr. Ramakanth Kumar P2
Dean PG (CS & ISE1, Professor & Head2, RVCE
Bangalore, India

*Abstract*—When Computational Linguistic is concerns Kannada is lagging far behind compared to Telugu and Tamil. Writing the grammar production for any south Indian language is bit difficult. Because the languages are highly inflected with three gender forms and two number forms. This paper is an effort to write Context Free Grammar for simple Kannada sentences. Kannada Language being one of the major Dravidian languages of India and it has 27th place in most spoken language in the world. But still it does not yet have computerized grammar checking methods for a given Kannada sentence. Thus, this paper highlights the process of generating context free grammar for simple Kannada sentences.

Keywords- Context Free Grammar, Syntactical Analysis

## INTRODUCTION

Formalization of natural languages had been a topic of interest among linguistic researchers for years. But such formal representations depend much on the nature of the language **[1].**

In Natural Language Processing (NLP) Syntax analysis is very much required. Natural Language Processing is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things **[2]**. Natural Language Processing is the computerized approach to analyzing text that is based on both a set of theories and a set of technologies.

Applications of Natural Language Processing include machine translation, Text Processing, Multilingual and cross language information retrieval, speech recognition and so on **[11]**.

Natural Language has an underlying structure usually referred to under the heading of Syntax **[3]**. The idea of Syntax is that words group together to form a phrase which behaves as a single unit called sentence. A commonly used mathematical system for modeling structure in Natural Language is Context Free Grammar.

In Speech to Text translation the output will not be having 100% accuracy. In order to increase the accuracy Syntactic Analyzer is introduced. This module extracts each sentence and check for the syntax according to the grammar that is selected.

With the advantages of Natural language processing here is an attempt to write context free grammar (CFG) for the simple Kannada sentences **[3]**. In order to write the CFG for the set of sentences the analysis of the grammar is a must.

Section II describes the Context Free Grammar, Section III refers to Kannada CFG, Section IV about the parsers, Section V concludes the paper.

## CONTEXT FREE GRAMMAR

Context Free Grammars is a notation that has been used extensively for defining the syntax of languages. Even with the regular expression grammar can be represented but it will be inadequate to characterize the syntax of natural languages, i.e. to make precise grammatical distinction is difficult in Regular expression. Even when language is formally regular, CFG will give correct interpretation. CFG is also known as Phrase-Structure Grammar (PSG) and which is equivalent to BNF (Backus-Naur form).

A context-free grammar is a formal system that describes a language by specifying how any legal text can be derived from a distinguished symbol called the sentence symbol. It consists of a set of productions, each of which states that a given symbol can be replaced by a given sequence of symbols **[16]**.

A context-free grammar (CFG) has four components: **[4]**

1. A set of tokens called terminals.
2. A set of variable called nonterminals.
3. A set of production rules.
4. A designation of one of the nonterminals as the start symbol.

*Recursive nesting of phrases can be easily done in CFG. All formal languages that can be generated by a CFG.*

*A grammar is a precise, understandable specification of language syntax (but not semantics). Grammar is also Collection of rules that describes well-informed sentences in a language.*

## KANNADA CFG

Designing a grammar for the entire Kannada language is a daunting, difficult task **[5]**; for the sake of simplicity, in this paper we will work with simple grammars that can generate only a subset of Kannada by writing grammatical productions with CFG.

*Following is the Kannada Grammatical Productions for a Robot to explain simple instructions like:*

ಕೆಂಪು ಬಾಲು ಕೊಡು (1)
Give red ball

ಬಿಳಿ ಪುಸ್ತಕ ಹಿಡಿದುಕೊ(2)
Hold white book

ಕಪ್ಪು ಬಾಗಿಲು ಎಳೆ (3)
Pull black door

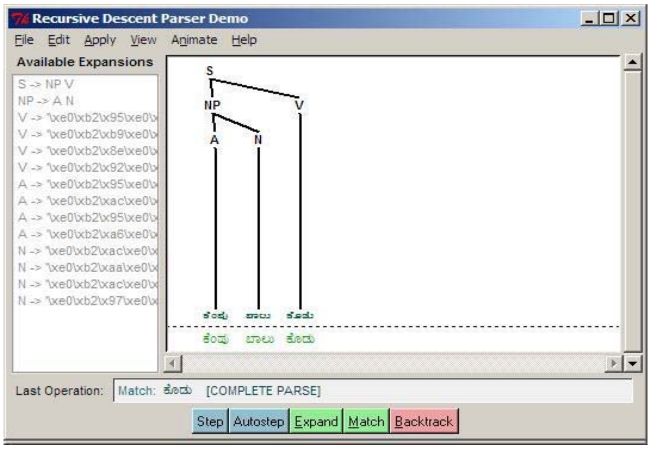
ದಪ್ಪ ಗುಂಡು ಒತ್ತು (4)
Press the big button

S → NP V
NP → A N
V → ಕೊಡು| ಹಿಡಿದುಕೊ| ಎಳೆ| ಒತ್ತು
N → ಬಾಲು| ಪುಸ್ತಕ| ಬಾಗಿಲು| ಗುಂಡು
A → ಕೆಂಪು |ಬಿಳಿ| ಕಪ್ಪು| ದಪ್ಪ

Interpretation of the above grammar requires syntactic analysis or parsing. There are two types of parsers available to parse the given grammar that is Top-Down and Bottom-Up.

A parse tree is a graphical representation of a derivation that shows hierarchical structure of the language. Parse tree interpretation for the simple instructions.

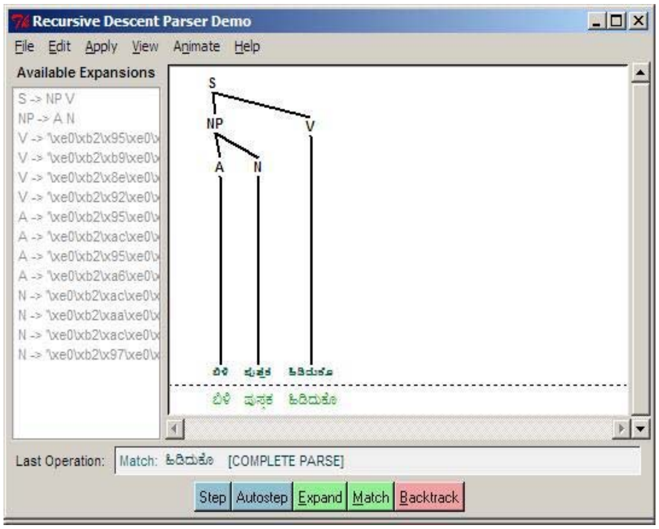
The following section gives the Parse tree for the above sentences. Parse tree is generated by using NLTK recursive descent parser. The recursive descent parser is a top-down parser.


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
ಕೆಂಪು ಬಾಲು ಕೊಡು
ಕೆಂಪು ಬಾಲು ಕೊಡು
Step Autostep Expand Match Backtrack

S -> NP V
NP -> A N
V -> '\xe0\xb2\x95\xe0\x
V -> '\xe0\xb2\xb9\xe0\x
V -> '\xe0\xb2\x8e\xe0\x
V -> '\xe0\xb2\x92\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xac\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xa6\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\xaa\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\x97\xe0\x

Last Operation: Match: ಕೊಡು [COMPLETE PARSE]

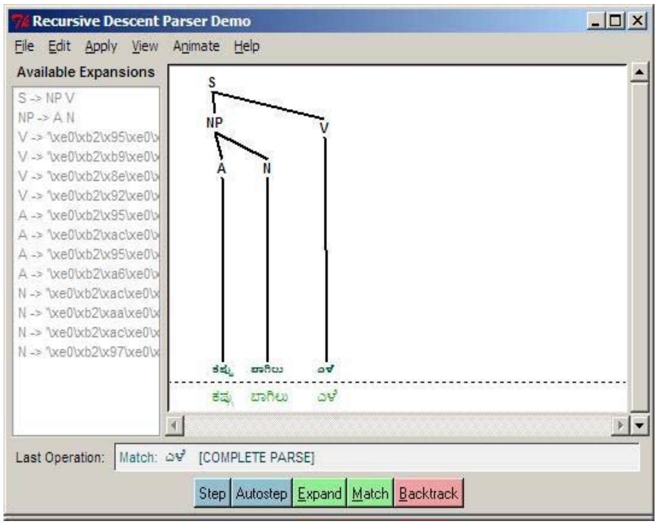
Figure 1. Parse tree for the sentence (1)


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
ಬಿಳಿ ಪುಸ್ತಕ ಹಿಡಿದುಕೊ
ಬಿಳಿ ಪುಸ್ತಕ ಹಿಡಿದುಕೊ
Step Autostep Expand Match Backtrack

S -> NP V
NP -> A N
V -> '\xe0\xb2\x95\xe0\x
V -> '\xe0\xb2\xb9\xe0\x
V -> '\xe0\xb2\x8e\xe0\x
V -> '\xe0\xb2\x92\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xac\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xa6\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\xaa\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\x97\xe0\x

Last Operation: Match: ಹಿಡಿದುಕೊ [COMPLETE PARSE]

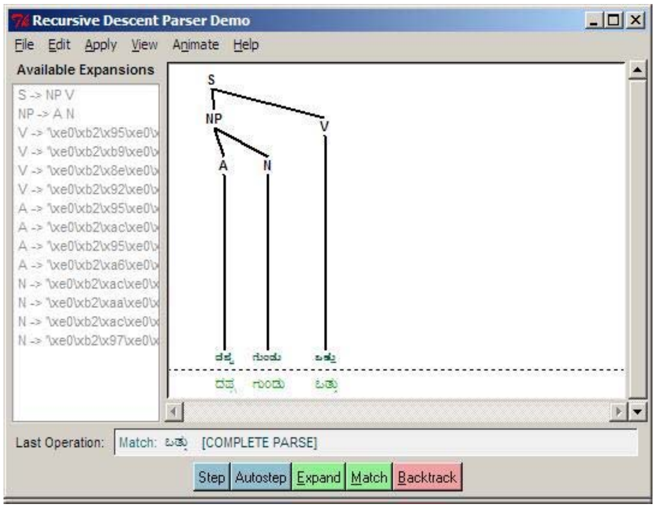
Figure 2. Parse tree for the sentence (2)


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
ಕಪ್ಪು ಬಾಗಿಲು ಎಳೆ
ಕಪ್ಪು ಬಾಗಿಲು ಎಳೆ
Step Autostep Expand Match Backtrack

S -> NP V
NP -> A N
V -> '\xe0\xb2\x95\xe0\x
V -> '\xe0\xb2\xb9\xe0\x
V -> '\xe0\xb2\x8e\xe0\x
V -> '\xe0\xb2\x92\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xac\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xa6\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\xaa\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\x97\xe0\x

Last Operation: Match: ಎಳೆ [COMPLETE PARSE]

Figure 3. Parse tree for the sentence (3)


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
ದಪ್ಪ
ಗುಂಡು
ಒತ್ತು
ದಪ್ಪ
ಗುಂಡು
ಒತ್ತು
Step Autostep Expand Match Backtrack

```
S -> NP V
NP -> A N
V -> '\xe0\xb2\x95\xe0\x
V -> '\xe0\xb2\xb9\xe0\x
V -> '\xe0\xb2\x8e\xe0\x
V -> '\xe0\xb2\x92\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xac\xe0\x
A -> '\xe0\xb2\x95\xe0\x
A -> '\xe0\xb2\xa6\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\xaa\xe0\x
N -> '\xe0\xb2\xac\xe0\x
N -> '\xe0\xb2\x97\xe0\x
```

Last Operation: Match: ಒತ್ತು [COMPLETE PARSE]

Figure 4. Parse tree for the sentence (4)

With the above robot instruction we will take another example with different sentences.

**ದೊಡ್ಡ ನಾಯಿ ಬಂದಿತು. (1)**

**Big dog came**

**ಶೂರ ಹನುಮಂತ ಸಮುದ್ರ ದಾಟಿದನು. (2)**

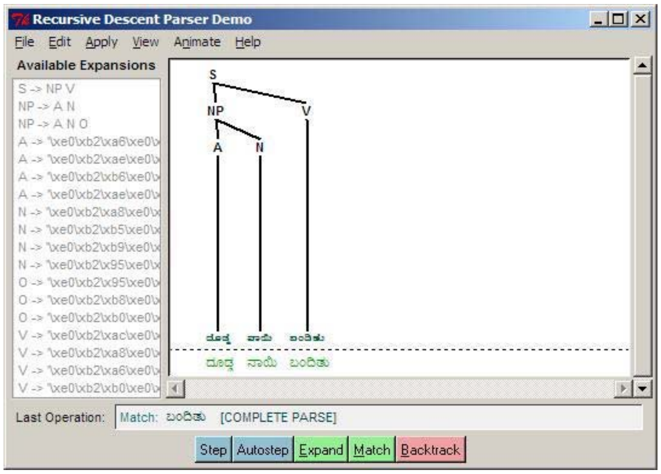
**Brave Hanumantha crossed the sea.**

**ಮಹಾಕವಿ ಕುವೆಂಪು ರಾಮಾಯಣದರ್ಶನಂ ರಚಿಸಿದರು. (3)**

**Great Poet Kuvempu written Ramayanadarshanam**

For the above sentences the grammatical productions are as follows:

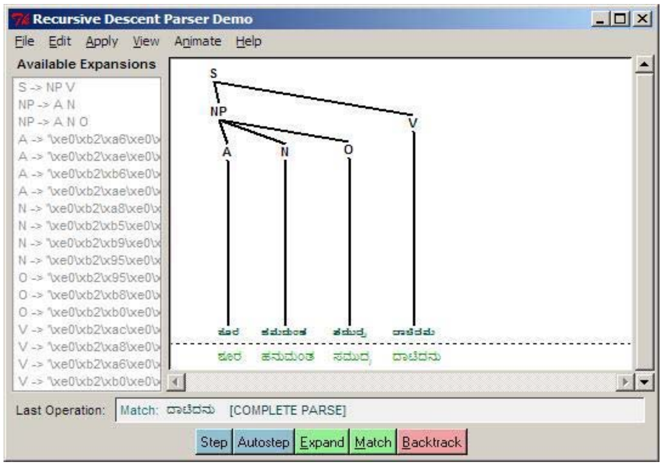
S → NP V
NP → A N | A N O
A → ದೊಡ್ಡ | ಮಹಾವಿಜ್ಞಾನಿ |ಶೂರ | ಮಹಾಕವಿ
N → ನಾಯಿ | ವಿಶೇಶ್ವರಯ್ಯ | ಹನುಮಂತ | ಕುವೆಂಪು
O → ಕನ್ನಂಬಾಡಿ | ಸಮುದ್ರ | ರಾಮಾಯಣದರ್ಶನಂ
V → ಬಂದಿತು | ನಿರ್ಮಿಸಿದರು | ದಾಟಿದನು | ರಚಿಸಿದರು


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
ದೊಡ್ಡ
ನಾಯಿ
ಬಂದಿತು
ದೊಡ್ಡ
ನಾಯಿ
ಬಂದಿತು
Step Autostep Expand Match Backtrack

```
S -> NP V
NP -> A N
NP -> A N O
A -> '\xe0\xb2\xa6\xe0\x
A -> '\xe0\xb2\xae\xe0\x
A -> '\xe0\xb2\xb6\xe0\x
A -> '\xe0\xb2\xae\xe0\x
N -> '\xe0\xb2\xa8\xe0\x
N -> '\xe0\xb2\xb5\xe0\x
N -> '\xe0\xb2\xb9\xe0\x
N -> '\xe0\xb2\x95\xe0\x
O -> '\xe0\xb2\x95\xe0\x
O -> '\xe0\xb2\xb8\xe0\x
O -> '\xe0\xb2\xb0\xe0\x
V -> '\xe0\xb2\xac\xe0\x
V -> '\xe0\xb2\xa8\xe0\x
V -> '\xe0\xb2\xa6\xe0\x
V -> '\xe0\xb2\xb0\xe0\x
```

Last Operation: Match: ಬಂದಿತು [COMPLETE PARSE]

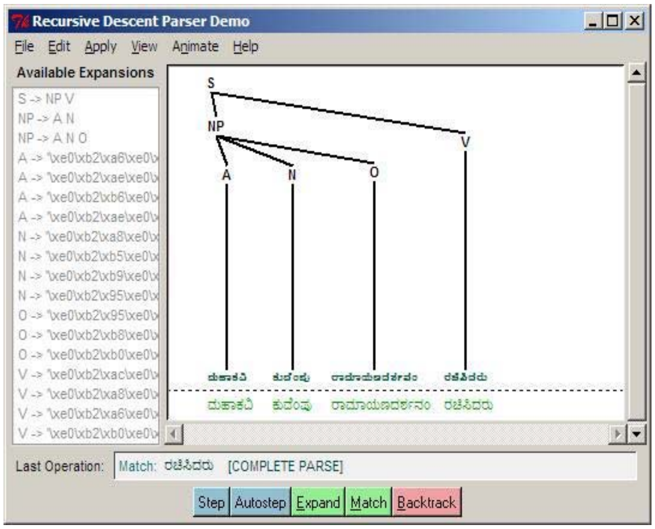
Figure 5. Parse tree for the sentence (1)


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
O
ಶೂರ
ಹನುಮಂತ
ಸಮುದ್ರ
ದಾಟಿದನು
ಶೂರ
ಹನುಮಂತ
ಸಮುದ್ರ
ದಾಟಿದನು
Step Autostep Expand Match Backtrack

```
S -> NP V
NP -> A N
NP -> A N O
A -> '\xe0\xb2\xa6\xe0\x
A -> '\xe0\xb2\xae\xe0\x
A -> '\xe0\xb2\xb6\xe0\x
A -> '\xe0\xb2\xae\xe0\x
N -> '\xe0\xb2\xa8\xe0\x
N -> '\xe0\xb2\xb5\xe0\x
N -> '\xe0\xb2\xb9\xe0\x
N -> '\xe0\xb2\x95\xe0\x
O -> '\xe0\xb2\x95\xe0\x
O -> '\xe0\xb2\xb8\xe0\x
O -> '\xe0\xb2\xb0\xe0\x
V -> '\xe0\xb2\xac\xe0\x
V -> '\xe0\xb2\xa8\xe0\x
V -> '\xe0\xb2\xa6\xe0\x
V -> '\xe0\xb2\xb0\xe0\x
```

Last Operation: Match: ದಾಟಿದನು [COMPLETE PARSE]

Figure 6. Parse tree for the sentence (2)


Recursive Descent Parser Demo
File Edit Apply View Animate Help
Available Expansions
S
NP
V
A
N
O
ಮಹಾಕವಿ
ಕುವೆಂಪು
ರಾಮಾಯಣದರ್ಶನಂ
ರಚಿಸಿದರು
ಮಹಾಕವಿ
ಕುವೆಂಪು
ರಾಮಾಯಣದರ್ಶನಂ
ರಚಿಸಿದರು
Step Autostep Expand Match Backtrack

```
S -> NP V
NP -> A N
NP -> A N O
A -> '\xe0\xb2\xa6\xe0\x
A -> '\xe0\xb2\xae\xe0\x
A -> '\xe0\xb2\xb6\xe0\x
A -> '\xe0\xb2\xae\xe0\x
N -> '\xe0\xb2\xa8\xe0\x
N -> '\xe0\xb2\xb5\xe0\x
N -> '\xe0\xb2\xb9\xe0\x
N -> '\xe0\xb2\x95\xe0\x
O -> '\xe0\xb2\x95\xe0\x
O -> '\xe0\xb2\xb8\xe0\x
O -> '\xe0\xb2\xb0\xe0\x
V -> '\xe0\xb2\xac\xe0\x
V -> '\xe0\xb2\xa8\xe0\x
V -> '\xe0\xb2\xa6\xe0\x
V -> '\xe0\xb2\xb0\xe0\x
```

Last Operation: Match: ರಚಿಸಿದರು [COMPLETE PARSE]

Figure 7. Parse tree for the sentence (3)

*Syntactic Analysis*

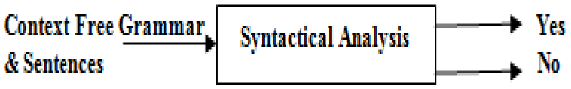


*Figure 8. Syntactical Analysis*

Syntactical Analysis means whether an input sentence can be generated by a given grammar **[11]**. When the answer is yes, the program should also output a parse tree for the sentence otherwise the sentence is syntactically wrong.

PARSER

A CFG only defines a language. It does not say how to determine whether a given string belongs to the language it defines. To do this, a parser can be used whose task is to map a string of words to its parse tree **[16]**.

Parsers are generally are of two kinds of parser, Top-Down Parser (Recursive Descent Parser) and Bottom-up parser (Shift-Reduce Parser).

*Top – Down Parsing*

Top down parsing is one strategy that build parse from the start Symbol (S). Top Down parsing is goal oriented. The goal is towards to parse the sentence according to the grammar production **[7]**.

Construct parse tree by starting at the start symbol and "guessing" at derivation step. It often uses next input symbol to guide "guessing".
A top-down parser starts with the root of the parse tree. It is labeled with the start symbol or goal symbol of the grammar. It picks a production and tries to match the input. It may require backtracking. Top-down parsers cannot handle left-recursion in a grammar. Top-Down Parsing is an attempt to find a left-most derivation for an input string.

To build a parse, it repeats the following steps until the fringe of the parse tree matches the input string **[14]**.

1. At the Start node S, Select a production with S on its *left hand* side and for each symbol on its *right hand* side, construct the appropriate child.
2. When a terminal is added to the fringe that doesn't match the input string, then backtrack.
3. Find the next node to be expanded.

If the parse tree did not match the input string then it means that input string is wrong. In the Kannada sentence if the parse tree did not match the input sentence then it means that there is a Syntax error in the sentence which is with respect to grammatical production that is whitened **[4]**.

Here is the Top-Down parsing for the sentence
ಕೆಂಪು ಬಟ್ಟೆಯನ್ನು ತಂದನು
.

S => NP VP
=> A N VP
=> A N V

This matches the input sequence.

Top down parsing occasionally requires backtracking. For example, suppose we used the derivation NP => N instead of the first derivation. Then, later we would have to backtrack because the derived symbols will not match the input tokens.

**Grammatical Production**

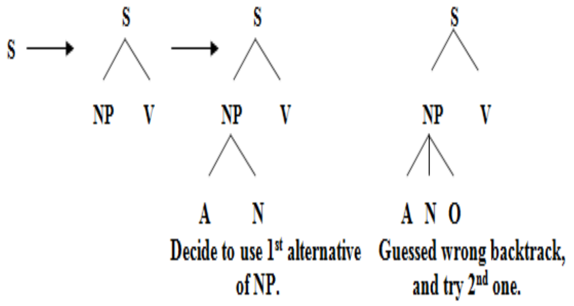
S → NP V
NP → A N | A N O

**Lexical Production**

A → ದೊಡ್ಡ | ಶೂರ
N → ನಾಯಿ | ಹನುಮಂತ
O → ಸಮುದ್ರ
V → ಬಂದಿತು | ದಾಟಿದನು

**Input Sentence**

ಶೂರ ಹನುಮಂತ ಸಮುದ್ರ ದಾಟಿದನು.
Brave Hanumanth sea crossed.
**Brave Hanumantha crossed sea.**


S
S
NP
V
S
NP
V
A
N
S
NP
V
A
N
O
Decide to use 1st alternative of NP.
Guessed wrong backtrack, and try 2nd one.

*Bottom-Up parser*

Parsing algorithms which proceed from the bottom of the derivation tree and apply grammar rules are called bottom up parsing algorithms. These algorithms will begin with an empty stack. One or more input symbols (words) are moved onto the stack, which are then replaced by nonterminals according to the grammar rules **[14]**. When all the input symbols have been read, the algorithm terminates with the starting nonterminal, alone on the stack, if the input string (Sentence) is acceptable.

Bottom up parsing involves two fundamental operations. The process of moving an input symbol (word) to the stack is called a shift operation, and the process of replacing symbols on the top of the stack with a nonterminal is called a reduce operation **[14]**. Most bottom up parsers are called shift reduce parsers because they use these two operations.

Main Idea of Bottom-Up parser is to look for substrings (words) that match right hand side (r.h.s) of a production.

Example:
Consider the following grammar parsed using Shift-Reduce Parser (bottom-up parser).

1. S → NP VP
2. NP → A N
3. NP → N
4. VP → V
5. A → ಶೂರನಾದ
6. N → ರಾಮು
7. V → ಬಂದನು

**Table 1. Sequence of Stack Frames Parsing ಶೂರನಾದ ರಾಮು ಬಂದನು**

| | |
|---|---|
| $ | ಶೂರನಾದ ರಾಮು ಬಂದನು<br>**Shift** |
| $ ಶೂರನಾದ | Reduce by rule 5 |
| $ A | ರಾಮು ಬಂದನು<br>**Shift** |
| $ A ರಾಮು | Reduce by Rule 6 |
| $ A N | Reduce by Rule 2 |
| $ NP | □<br>Shift |
| □ | Reduce by Rule 7 |
| $ NP VP | Reduce by Rule 1 |
| $ S | Accept |

## CONCLUSION

This paper sets the stage to develop a computerized grammar checking methods for a given Kannada sentence. Two sets of example is taken to explain the writing of Context Free Grammar (CFG) for a simple Kannada sentences. Grammar is parsed with Top-Down and Bottom-Up parser. Bottom-Up parser has two conflicts such as Shift-Reduce and Reduce-Reduce. Top-Down parser is more suitable to parse the given grammatical production.

## ACKNOWLEDGMENT

I owe my sincere feelings of gratitude to Dr. S.C. Sharma for his valuable guidance and suggestions which helped me a lot to write this paper. It gives us great pleasure to express my feelings of gratitude to Dr. G. Shobha & Dr. Ramakanth Kumar P for valuable guidance support and encouragement.

## REFERENCES

**[1]** Bala Sundara Raman L, Ishwar S, Context Free Grammar for Natural Language constructs - An implementation for Venpa class of Tamil Poetry Tamil Internet 2003, Chennai, Tamilnadu, India

**[2]** Gobinda G. Chowdhury, Natural Language Processing Dept. of Computer and Information Sciences University of Strathclyde, Glasgow G1 1XH, UK

**[3]** Jhon, Parsing Natural Language with Context Free Grammars (2005)

**[4]** Ayesha Binte Mosaddeque & Nafid Haque, Context-Free Grammar for Bangla, Dhaka, Bangladesh

[5] Vishweswariah, H.S.K. 2008 Edition Standard Kannada – English Grammar

[6] Roark B. Probabilistic Top–Down Parsing and Language Modeling, Association for Computational Linguist, 2001

[7] Rama Sree, R.J. , Kusuma Kumari P., Combining Pos Taggers For Improved Accuracy To Create Telugu Annotated Texts For Information Retrieval, Dept. of Telugu Studies, Tirupathi, India

[8] Kannada madhyama Vyakarana by T.N. Sreekantaiya, 2008 Edition

[9] Kannada prathama Vyakarana by Dr. A.N. Narasimhia, 2008 Edition, ISBN: 81-89818-91-0

[10] www.bookrags.com/eb/kannada-language-eb

[11] Rao, Durgesh, Pushpak Bhattacharya and Radhika Mamidi, "Natural Language Generation for English to Hindi Human-Aided Machine Translation", pp. 179-189, in KBCS-98, NCST, Mumbai.

[12] Vikram T N, Chidananada Gowda K and Shalini R Urs, Symbolic representation of Kannada characters for Recognition, International School of Information Management, Manasagangotri, Mysore, 2005

[13] Dr. A.N. Narasimhia, Kannada prathama Vyakarana, 2008 Edition, ISBN: 81-89818-91-0

[14] Bala sundara Raman L, Ishwar S, Sanjeeth Kumar Ravindranath, Context Free Grammar for Natural Language Constructs - An Implementation for Venpa class of Tamil Poetry, Tamil Internet 2003, Chennai, India

[16] G.V. Singh And D.K. Lobiyal , A Computational Grammar For Hindi Verb Phrase, IEEE transactions, 1994

**About the authors**

B.M. Sagar, Assistant Professor of Department of Information Science and Engineering. He obtained his Master’s Degree in Computer Science & Engineering from VTU and B.E. in Computer Science & Engineering from Kuvempu University. His research interests are Pattern Recognition. He has guided more than 25 under graduate projects and 3 post graduate projects. He has presented and published papers at national conference / International Conference and 2 Research Publication.

Dr. Shobha G., Dean PG(CSE,ISE), RVCE. She has been awarded Ph.D for her thesis titled “Knowledge Discovery in Transactional Database Systems” from Mangalore University, Mangalore. She obtained her M.S. degree in Software Systems from BITS, Pillani and BE in Computer Science from Gulbarga University. Her research interests are Data Mining, DBMS, and Operating Systems & Networking. She has guided more than 45 undergraduate and 09 post graduate projects.

Dr. Ramakanta Kumar, P was awarded Doctorate from Mangalore University, has teaching experience of around 14 years in academics and Industry. His area of research is on Artificial Intelligence, Pattern recognition. He has to his credits 03 National Journals, 02 International Journals, 12 Conferences and 15 Research Publications. He is guiding 14 M.Tech students and 03 PhD students.