To Design a English Language Recognizer by using Nondeterministic Pushdown Automata (ELR-NPDA)

Madiha Khurram Pasha
Department of Computer Science (UBIT)
University of Karachi

M. Sadiq Ali Khan, Ph.D.
Department of Computer Science (UBIT)
University of Karachi

ABSTRACT
Natural language recognization is a popular topic of research as it covers many areas such as computer science, artificial intelligence, theory of computation, and machine leaning etc. Many of the techniques are used for natural language recognization by the researchers, parsing is one of them.

The aim to propose this paper is to implement nondeterministic pushdown automata (NPDA) for the English Language (ELR-NPDA) that can modernize Context Free Grammar (CFG) for English language and then refurbish into Nondeterministic Pushdown Automata (NPDA). This converting procedure can uncomplicatedly parse legitimate English language sentences. Parsing can be organized by Nondeterministic Pushdown Automata (NPDA) that used push down stack and input tape for recognizing English language sentences. To formulate this NPDA convertor we have to exchange Context Free Grammar into Chomsky Normal Form (CNF). The move toward this is more appropriate because it uses nondeterministic approach of PDA that can improve language recognizing capabilities as compare to other parsing approach.

General Terms
Natural Language Recognizer, Computer Science, Artificial Intelligence, Theory of Computation, Machine Learning, Pushdown Automata (PDA), Nondeterministic pushdown automata (NPDA), Context Free Grammar (CFG), Chomsky Normal Form (CNF).

Keywords
English Language Recognizer - Nondeterministic Pushdown Automata (ELR-NPDA)

1. INTRODUCTION
The main purpose of parsing is to find the syntax error in any natural language or in computer language. Every language has some set of rules whether it is a Computer language or a natural language. The arrangement of words in a sentence and punctuation are called syntax. Our focus here is to parse the English Language.

Parsing has two major types: Top-down parsing and Bottom-up parsing. But, our approach is different from these types. We designed a Nondeterministic Push down automata (NPDA) for parsing. We prefer Nondeterministic PDA as compare to Deterministic PDA because DPDA can recognize only the sub-sets of CFG while on the other hand NPDA can store unbounded amount of information and recognize the whole CFG due to it nondeterministic nature. In this approach first is to take left factored CFG for English language [1], and then convert this CFG into Chomsky normal form (CNF) and then CNF convert into the NPDA.

CFG is a collection of four things: a starting non terminal, a finite set of non terminal, a finite set of terminal and a finite set of production. For parsing we need a left factored Context free Grammar [8] [1]. To achieve goal this CFG must be converted into CNF. Before this conversion there are two more phases. First one is to remove all null productions (Non-terminal→ €) and get a new GFG. Now, second is to remove all unit productions (Non terminal → one non-terminal). The resulting CFG is now capable to make Chomsky normal form (CNF). In CNF the whole CFG only has these two types of productions:

1. Non terminal → exactly two non terminals
2. Non terminals → one terminals

The resultant CFG is required Chomsky Normal Form (CNF) that can be converted into Nondeterministic Push Down Automata (NPDA) .The Nondeterministic Push down automata consist of following elements: the set of input letter, an input tape, a stack, and set of states (START, ACCEPT, REJECT, PUSH, POP, READ).

2. LITRATURE REVIEW
Parsing is a technique that checks the syntactical structure of any given input, if the given input is matched with the syntactical structure of parser then, it can be passed from it otherwise it will reject. Parsing having two types: [22], [23] Top-down and Bottom-up parsing. Top-down parsing technique first finds the highest level of the parse tree that parses from top to bottom, whereas, the bottom-up parsing does the opposite. Top-down parsing has few types, Recursive descent parser and LL parser.

[10] used chart top-down parsing [11] mechanism to parse the Arabic language sentences. This proposed parser can parse Arabic sentences from real documents and also capable for identifying conjunctions, exceptive particles, preposition etc in the Arabic language. [12] proposed a new top down parsing algorithm. This algorithm accommodates ambiguity and left recursion in polynomial time. Another modular and efficient top-down parsing technique [13] for same ambiguous left-recursive grammars has been proposed. The Parsing Expression Grammars (PEGs) [14] have been projected. The PEGs integrate with lexical and parsing phase.

A survey paper [15] for all types of parsing have been proposed. In which all types of parsing technique have been discussed. Another survey paper [16] describes the different methodologies for context free grammar and provides an introduction to main concept and latest approaches in Natural Language Learning research. Also, [17] a semi supervised relation extraction mechanism by using CFG has been proposed. While, [18], [19], [20] using the bottom-up parsing.

The context free grammar that defines the syntax of English language sentences was proposed in [1]. Basically it is "English grammar predictive parser" that portrays a left factored context-free grammar for English language and used top down predictive parsing for syntax checking. Same approach of top down parsing is used is [2] that projected the Bangla grammar parser that also based on predictive parsing approach for parsing the Bangla language.

A "natural language analyzer" has been proposed in [3] that takes a general class of context-free grammars as drivers and also used the conception of non-deterministic pushdown transducer to compute the computational efficiency. Also in [4] an algorithm has been planned that offer conversion from "acceptor model" to "generator model". In this pushdown automata used as an acceptor model and context-free grammar used as a generator model.

Another technique that based on this conversion was developed in [5] is a context-free grammar that used as a meta-language. This language is then joint with Genetic programming to develop Deterministic Push-Down Automata (D-PDAs). Also Formalization of context-free grammars and pushdown automata using HOL4 was projected in [6]. Through formalization is to create a CFG from a PDA and vice versa. See also the [9]. Another paper [23] verifies the theorem that every Context Free Grammar (CFG) is accepted by a Pushdown Automata (PDA).

JFLAP [7] is a potent tool for execution of theory of computation. It supports approximately all topics of machine computation such as finite state machine, nondeterministic finite automata, nondeterministic pushdown automata, multi-tape Turing machines, several types of grammars, and parsing etc. By using this software CFG to NPDA translation is pretty trouble-free.

3. METHODOLOGY

The block diagram shows the steps to convert English language Context Free Grammar (CFG) into Nondeterministic Pushdown Automata (NPDA). (See Figure 1)

The first step for this conversion is to make left factored CFG for the English language [1]. Second is to remove all null, third to kill all unit productions from the CFG and in the forth step convert this CFG into CNF, in the second last step this CNF is capable for the NPDA. Finally, this ELR-NPDA can be tested through parsing of different English sentences.

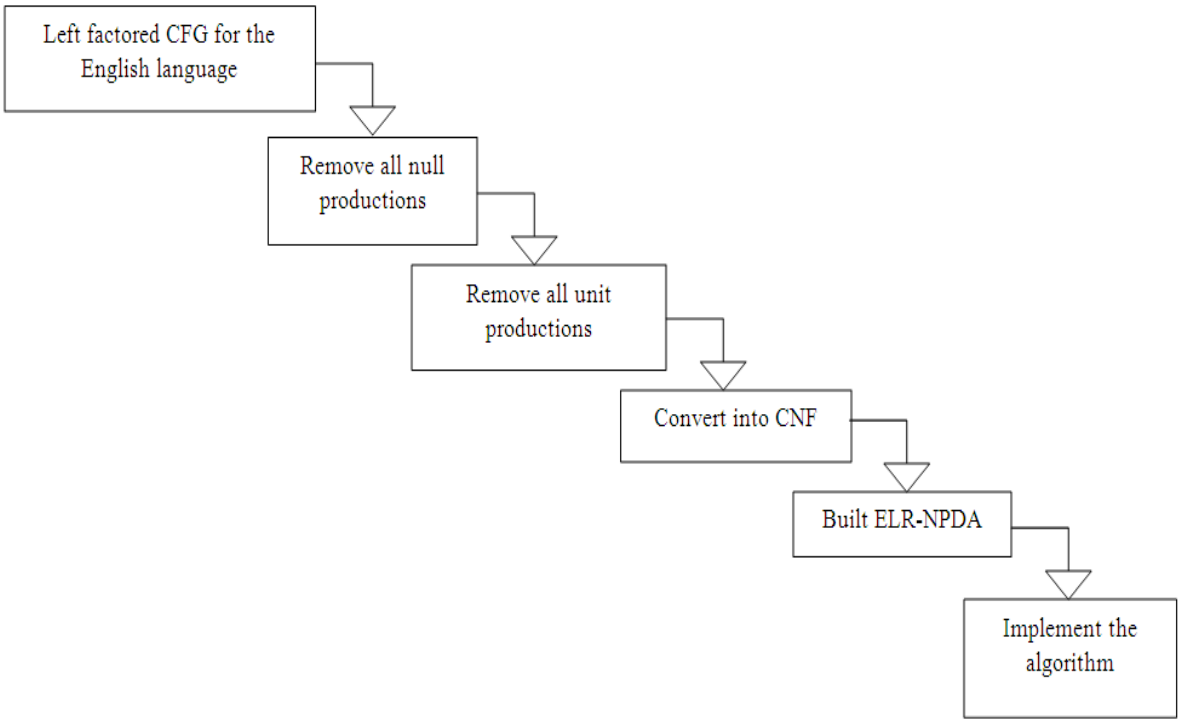

Fig 1: The ELR-NPDA Model

3.1 Left Factored CFG for English Language

The left factored context free grammar extract from [1]. (See Table 1**)**

Table 1. Left factored grammar [1]

S → NP.VP
NP→ a NP1.VP4 |pronoun.NP4|the.NP6|an.NP7 | propernoun.NP3 | I| noun
NP1 → noun|adjective.NP2
NP2→ noun
NP3→ conjuction.NP5|€
NP4→ conjuction.NP5|noun|€
NP5→ noun |pronoun | propernoun
NP6→propernoun.NP4|adjective NP2
NP7→ adjective1.NP2
VP→ verb1.vp'|verb2.vp' |aux31.VP3 |aux32.VP6 | aux21.VP4| aux22.VP9|aux11.VP5
VP→aux12.VP7|adverb.VP6
VP'→ NP1.VP2|adverb.VP2|PP.NP|€|pronoun
VP1→ adjective.NP2
VP2→PP.NP|€
VP3→ verb4.VP'|adverb.VP6|pronoun.VP1
VP4→verb1.VP'|be.VP6|aux.11.VP7|have.VP8
VP5→verb3.VP'|BEEN.VP6
VP6→verb4.VP'
VP7→ verb3.VP'
VP8→ been.VP6
VP9→ be.VP6
PP→ preposition

3.2 Remove All Null Productions

After removing the €-productions the CFG become: (see Table 2). Only four non-terminals are having the null productions i.e. NP3, NP4, VP' and VP2.

Table 2. Remove all null productions

S → NP .VP
NP→ a NP 1.VP4 |pronoun |the.NP6|an.NP7 |propernoun |I| noun| pronoun.NP4|proper noun.NP3
NP1 → noun|adjective.NP2
NP2→ noun
NP3→ conjuction.NP5
NP4→ conjuction.NP5|noun
NP5→ noun |pronoun | propernoun
NP6→proper noun.NP4|propernoun.| adjective.NP2
NP7→ adjective1.NP2
VP→ verb1.vp'| verb1.|verb2.vp' |verb2. |aux31.VP3 | aux32.VP6|aux21.VP4|aux22.VP9|aux11.VP5
VP→aux12.VP7 |adverb2.VP6
VP'→NP1.VP2|NP1.|adverb.VP2|adverb.|PP.NP|pronoun
VP1→ adjective.NP2
VP2→PP.NP
VP3→ verb4.VP'|verb4.|adverb.VP6|pronoun.VP1
VP4→verb1.VP'|verb1.|be.VP6|aux.11.VP7|have.VP8
VP5→verb3.VP'|verb3.|been.VP6
VP6→verb4.VP'|verb4
VP7→ verb3.VP'|verb3.
VP8→ been.VP6
VP9→ be.VP6
PP→ preposition

3.3 Remove All Unit Productions

There is only a single unit production

- VP'→NP1

So, it can be replace by

- VP' → noun
- VP'→ adjective.NP2

3.4 Convert into Chomsky Normal Form (CNF)

As mention above there are two possibilities for CNF:

1. Non terminal → exactly two non terminals
2. Non terminals → one terminals

3.4.1 Possibility 1 for CNF

The result is shown in Table 3.

Table 3. Possibility 1 for CNF

S→NP.VP
NP→A'NP1.VP4|pronoun|C'.NP6|B'.NP7|propernoun|I| noun| F'.NP4|G'.NP3
NP1→noun|H'.NP2
NP2→noun
NP3→L'NP5
NP4→L'.NP5|noun
NP5→noun| propernoun| pronoun
NP6→G'.NP4| propernoun| H'.NP2
NP7→I'.NP2
VP→M'.VP'| verb1| N'.VP'| verb2 | U'.VP3| V'.VP6 | S'.VP4| T'.VP9| Q'.VP5 |R'.VP7| J'.VP6
VP'→NP1.VP2| noun| H'.NP2| J'.VP2| adverb| PP.NP | pronoun
VP1→H'.NP2
VP2→PP.NP
VP3→P'.VP | verb4 | J'.VP6 | F'.VP1
VP4→M'.VP' | verb1|Y'.VP6| Q'.VP7| W'.VP8
VP5→O'.VP' | verb3 | X'.VP6
VP6→P'.VP'| verb4
VP7→O'.VP'| verb3
VP8→X'.VP6
VP9→Y'.VP6
PP→preposition
A' →a
B'→an
C'→ the
D'→noun
F'→pronoun
G'→proper noun
H' → adjective
I → adjective1
J'→ adverb
K'→preposition
L'→Conjunction
M'→verb1
N'→verb2
O'→verb3
P'→verb4
Q'→aux11
R'→aux12
S'→aux21
T'→aux22
U'→aux31
V'→aux32
W'→have
X'→been
Y'→be

3.4.2 Possibility 2 for CNF

The result of second possibility of CNF is shown in Table 4.

Table 4. Possibility 2 for CNF

1. S→NP.VP
2. NP→A'R1
3. R1→NP1.VP4
4. NP→pronoun | C'.NP6 | B'.NP7 | propernoun | I| noun| F'.NP4| G'.NP3
5. NP1→noun|H'.NP2
6. NP2→noun
7. NP3→L'NP5
8. NP4→L'.NP5|noun
9. NP5→noun| propernoun| pronoun
10. NP6→G'.NP4| propernoun| H'.NP2
11. NP7→I'.NP2
12. VP→M'.VP'| verb1| N'.VP'| verb2 | U'.VP3| V'.VP6 |S'.VP4| T'.VP9| Q'.VP5 |R'.VP7| J'.VP6
13. VP'→NP1.VP2| noun| H'.NP2| J'.VP2| adverb| PP.NP | pronoun
14. VP1→H'.NP2
15. VP2→PP.NP
16. VP3→P'.VP | verb4 | J'.VP6 | F'.VP1
17. VP4→M'.VP' | verb1|Y'.VP6| Q'.VP7| W'.VP8
18. VP5→O'.VP' | verb3 | X'.VP6
19. VP6→P'.VP'| verb4
20. VP7→O'.VP'| verb3
21. VP8→X'.VP6
22. VP9→Y'.VP6
23. PP→preposition
24. A' →a
25. B'→an
26. C'→ the
27. D'→noun
28. F'→pronoun
29. G'→proper noun
30. H' → adjective
31. I'→ adjective1
32. J'→ adverb
33. K'→preposition
34. L'→Conjunction
35. M'→verb1
36. N'→verb2
37. O'→verb3
38. P'→verb4
39. Q'→aux11
40. R'→aux12
41. S'→aux21
42. T'→aux22
43. U'→aux31
44. V'→aux32
45. W'→have
46. X'→been
47. Y'→be

3.5 Construct English Language Pushdown Automata (ELR-NPDA)

The diagrammatical view of ELR-NPDA is exposed in figures 2, 3, 4 and 5. Essentially these figures (2, 3, 4, and 5) showing the single nondeterministic pushdown automata for the English language. It means that all figures (2, 3, 4, and 5) are connected with each other. It has been separated into parts for appropriate view and better understanding. The all diamond shapes symbol in ELR-NPDA symbolize the READ states and all rectangular shapes shows the PUSH states of stack.

3.6 Algorithm for ELR-NPDA

The algorithm of ELR-NPDA that can recognize syntactically correct sentences of the English language is to be proposed here.

```
1. Put starting non-terminal symbol on the empty stack.
2. Set variable 'x' to the top of stack.
3. while (1)
   {
    if(x == non_ternimal)       // x = top of stack
            {
            Nondeterministically choose a production and
            replace non-terminal with the production's rule.
            }
   else if(x== terminal)
            {
            Compare this terminal with the next input symbol
            from input tape.
            if (terminal != input symbol)
                    goto Reject state
            else
                    Advance input tape head to read next
                    symbol
            }
    else     //codition if top of stack points to start of stack
            {
            if (input symbol==end marker)  //input ends
                    goto accept state
            else
                    goto reject state.
            }
    }
```

4. RESULTS

The above mention algorithm of EL-NPDA can easily parse valid English sentences. For parse any sentence initially stack is empty and tape having input sentences which we want to parse. We begin by pushing the starting non-terminal onto the empty stack. This starting symbol is now pop form the stack and nondeterministically chooses a production which start form starting symbol and push onto the stack. The process continues until the end of input tape. The dry run of algorithm is shown in table 5 and 6.

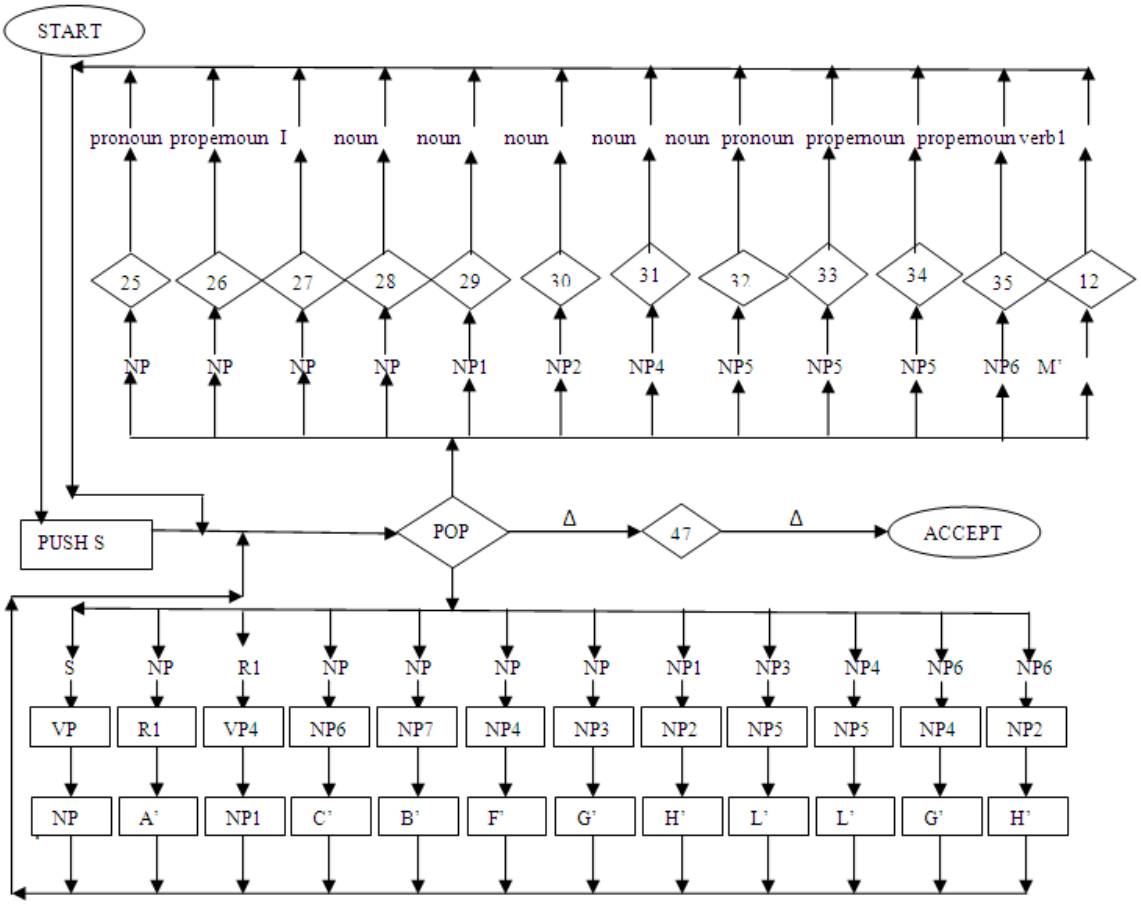

Fig 2: ELR-NPDA diagrammatical view 1

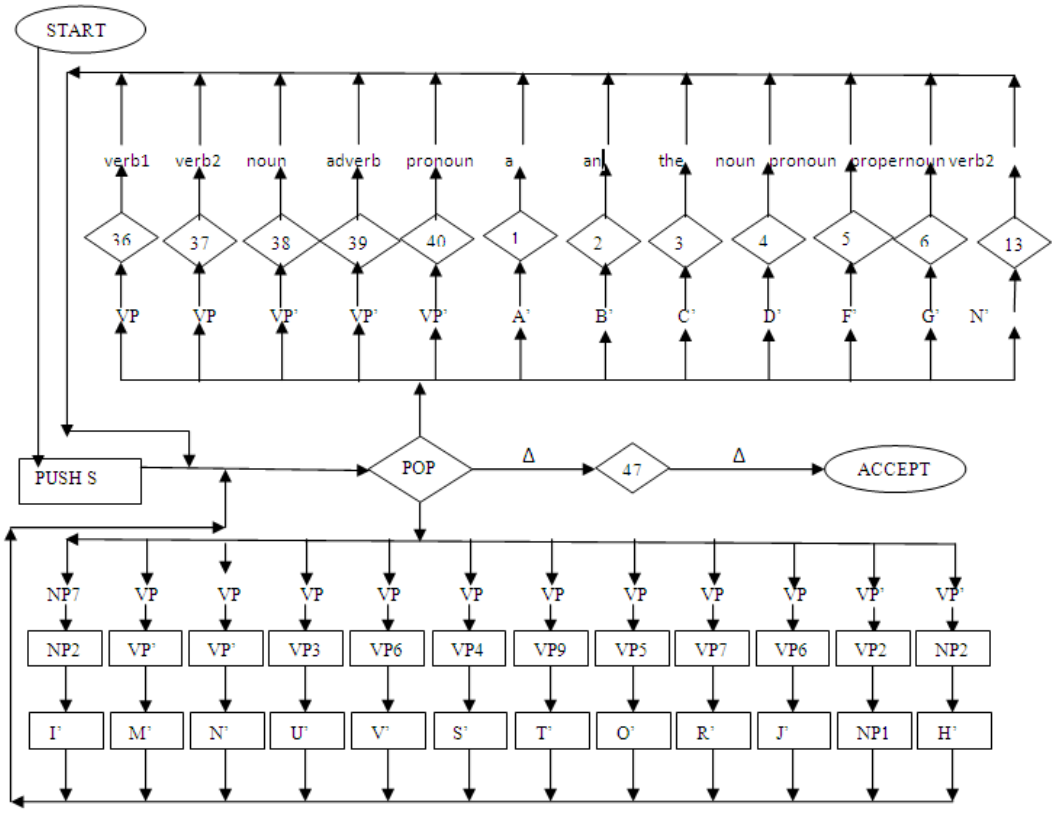

Fig 3: ELR-NPDA diagrammatical view 2

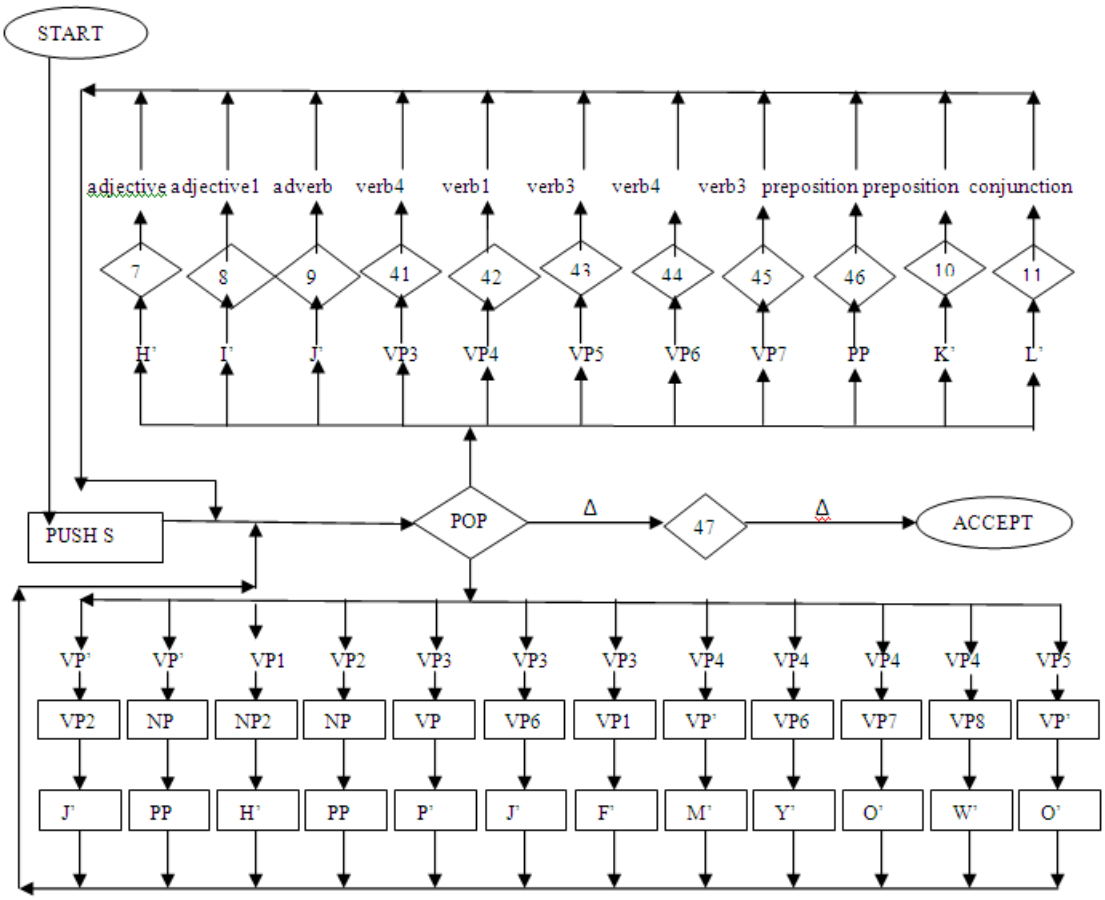

Fig 4: ELR-NPDA diagrammatical view 3

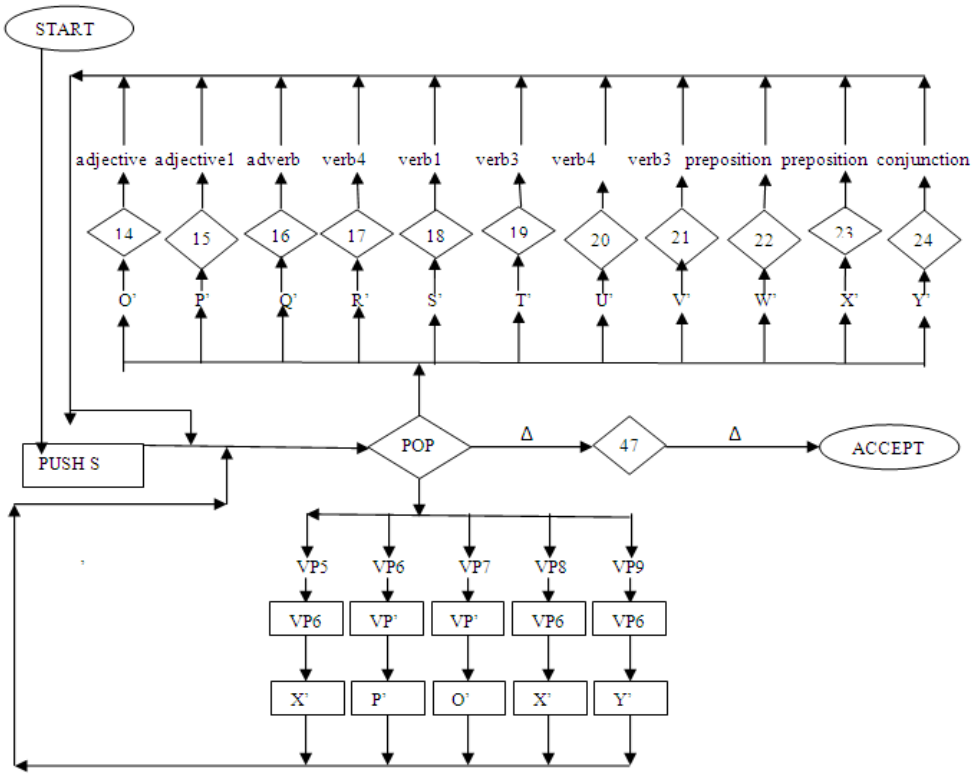

Fig 5: ELR-NPDA diagrammatical view 4

Table 5: example 1 "John was going to school"

STATE	STACK	TAPE
START	Δ	~~John~~ was going to school
PUSH S	S	~~John~~ was going to school
POP	Δ	~~John~~ was going to school
PUSH VP	VP	~~John~~ was going to school
PUSH NP	~~NP~~.VP	~~John~~ was going to school
POP	VP	~~John~~ was going to school
READ 23	VP	~~John~~ was going to school
POP	~~VP~~	~~John~~ was going to school
PUSH VP6	VP6	~~John~~ was going to school
PUSH v'	~~V'~~.VP6	~~John was~~ going to school
READ 21	VP6	~~John was~~ going to school
POP	~~VP6~~	~~John was~~ going to school
PUSH VP'	VP'	~~John was~~ going to school
PUSH P'	P'.VP'	~~John was~~ going to school
Pop	~~P~~' .VP'	~~John was~~ going to school
READ 15	VP'	~~John was going~~ to school
POP	~~VP'~~	~~John was going~~ to school
PUSH NP	NP	~~John was going~~ to school
PUSH PP	PP.NP	~~John was going to~~ school
POP	~~PP~~.NP	~~John was going to~~ school
READ 46	NP	~~John was going to~~ school
POP	~~NP~~	~~John was going to~~ school
READ 28	Δ	~~John was going to school~~
ACCEPT	Δ	~~John was going to school~~

Table 6: example 2 "The young man is laughing"

STATE	STACK	TAPE
START	Δ	The young man is laughing
PUSH S	S	The young man is laughing
POP	Δ	The young man is laughing
PUSH VP	VP	The young man is laughing
PUSH NP	~~NP~~.VP	The young man is laughing
PUSH NP6	VP	The young man is laughing
PUSH C'	~~C'~~.NP6.VP	The young man is laughing
POP	NP6.VP	The young man is laughing
READ 3	NP6.VP	~~The~~ young man is laughing
POP	VP	~~The~~ young man is laughing
PUSH NP2	NP2.VP	~~The~~ young man is laughing
PUSH H'	H'.NP2.VP	~~The~~ young man is laughing
POP	NP2.VP	~~The~~ young man is laughing
READ 7	NP2.VP	~~The~~ young man is laughing
POP	VP	~~The young~~ man is laughing
READ 30	VP	~~The young man~~ is laughing
READ 30	VP	~~The young man~~ is laughing
POP	Δ	~~The young man~~ is laughing
PUSH VP6	VP6	~~The young man~~ is laughing
PUSH V'	V'.VP6	~~The young man~~ is laughing
POP	VP6	~~The young man~~ is laughing
READ 21	VP6	~~The young man~~ is laughing
POP	Δ	~~The young man is~~ laughing
PUSH VP'	VP'	~~The young man is~~ laughing
PUSH P'	P'.VP'	~~The young man is~~ laughing
POP	VP'	~~The young man is~~ laughing
READ 15	VP'	~~The young man is laughing~~

5. CONCLUSION AND FUTURE WORK:

After comparative study of Nondeterministic Pushdown Automata (NPDA) with the other parsing techniques (top-down and bottom-up) the conclusion is that NPDA is more appropriate than any other techniques. NPDA is able to count letters. Also it uses pushdown stack and by using this stack, the language recognizing capabilities are increased considerably as compare to other parsing approaches. The PDA can hold unlimited amount of information. The only drawback is that it uses more memory because of stack data structure. The comparative results of NPDA with the other are shown in table 7.

Table 7: Parsing Techniques Comparisons

Parsing Techniques	Recognizing Capabilities	Memory Usage	Time
PDA	It cannot recognized natural language which based on CFG	Large memory as compare to top-down and bottom up, and less memory as compare to NPDA	Fast for LR(1) based language
Top-down parsing	Slow as compare to NPDA and PDA	Less memory as compare to PDA and NPDA	Take too much time to find possible sentences
Bottom-up parsing	Slow as compare to NPDA and PDA	Less memory as compare to PDA and NPDA	Take less time as compare to bottom-up. parsing
NPDA	Excellent recognizing capability as compare to others.	Large memory as compare to all others.	Fast for natural language

In the future, we will design and implement the NL-PDA for many other Natural languages. The ELR-PDA has some restrictions that it cannot parse idioms and poetry text, but in future we will cope up with this limitation too.

6. REFERENCES

[1] Ratnagiri, Ichalkanji, Recognizing English grammar using predictive parser, IJERA, Vol. 3, Issuen4, Jul-Aug 2013, pp.306-312

[2] K. M. Azharul Hasan, Al-Mahmud, Amit Mondal, Amit Saha, Recognizing Bangla grammar using predictive parser, IJCSIT, Vol 3, No 6, Dec 2011, pp-61-73.

[3] Manuel Vilares Ferro, An efficient context-free backbone for natural language analyzers.

[4] Stefan Andrei, Hikyoo Koh, A fixed-point approach towards efficient models conversion, No. 2, June 2008.

[5] Afra Zomorodian, Context-Free Language Induction by Evolution of Deterministic Push-Down Automata Using Genetic Programming.

[6] Aditi Barthwal1, Michael Norrish, Mechanisation of PDA and Grammar Equivalence for Context-Free Languages.

[7] www.jflap.org

[8] Robert C. Moore, Removing Left Recursion Form context Free Grammar.

[9] http://hol.sourceforge.net.

[10] Ahmed Al-Taani, Mohammed Msallan and Sana Wedian, A top-Down chart Parser for Analyzing Arabic Sentences.

[11] Dick G. and Ceriel H., Paring Techniques, a Practical Guide, Technical Report, England, 1990.

[12] Richard A. Frost and Rahmatullah Hafiz, A New Top-Down Parsing Algorithm to Accommodate Ambiguity and left Recursion in Polynomial Time.

[13] Richard A. Frost, Rahmatullah Hafiz and Paul C-Callaghan, Modular and Efficient Top-down Parsing for Ambiguous Left-Recursive Grammars, Proceedings of the 10th Conference on Parsing Technologies, pages 109–120,

[14] Laurence Tratt, Direct Left-Recursive Parsing Expression Grammars.

[15] Pankaj Sharma, Naveen Malik, Naeem Akhter, Rahul, Hardep Rohilla, Parsing Techniques: A Review.

[16] Arianna D Ulizia, Fernando Ferri, and Patrizia Grifoni, A Survey of Grammatical inference Methods for Natural Language Learning.

[17] Georgios Petaris, Vangelis Karkaletsis, Georgios Paliouras and Contantine D. Spyropoulos, Learning Context-Free Grammars to Extract Relations Form text".

[18] Robert Moore and John Dowding, Efficient Bottom-up Parsing.

[19] Nazir Ahmad Zafar, LR(K) Parser Constraction Using Bottom Up Formal Analysis.

[20] David Carter, Efficient Disjunctive Unification for Bottom-Up Parsing.

[21] J.C.M. Beaten, P.J.L. Cuipers and P.J.A Van Tilburg, A Context-Free Process as a Pushdown Automata.

[22] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. compilers: principles, techniques, and tools: (Addison-Wesley Publishing Company, 1988).

[23] Kenneth C. Louden. compiler construction principles and practice. (San Jose State University: Galgotia Publication pvt.ltd, 2002)