JNIESTRT'S

SMT. INDIRA GANDHI COLLEGE OF ENGINEERING

GHANSOLI, NAVI MUMBAI – 400701

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)

COMPUTER ENGINEERING DEPARTMENT

ACADEMIC YEAR :- 2021-22(EVEN SEM)

NAME- DEEPAK H CHOURASIYA
ROLL NO - 77
YEAR - TE SEM - VI
BRANCH - COMPUTER

LAB NO : 07

TITLE-: IMPLEMENTING FIRST ORDER LOGIC USING - PARSING OF CONTEXT FREE GRAMMAR USING PROLOG

Date of Performance	Date of Evaluation	Marks (10)					Sign / Remark
		A	B	C	D	E	
		2	3	2	2	1	
08/03/22	15/03/22						
		Total Marks					

Roll No: 77
Deepak S

Date: 08-3-22

Date	Experiment	Grade:
08-03-22	Exp 7- Implementing First Order logic using Prolog: Parsing Of Context Free Grammar	Sign:

Aim: To implement first order logic - Parsing of context free grammar using prolog.

Theory:

- Prolog is a programming language used in creating AI.
- Prolog is classified as a logic programming language
- We will use list in prolog.
- For each non-terminal we will construct a Prolog procedure to recognize strings generated by that non-terminal.
- Each procedure will have two arguments.
- The first will be an input parameter consisting of the representing the input string.
- The second will be an output argument, and will be set by the procedure to

Date:_____

the remainder of the input string from an initial segment matched by the non-terminal has been removed.

Problem Statement:

Consider the grammar for a small subset of English sentences [stated in Backus Naur Form].

<s> ::= <np>, <vp>
<np> ::= <det>, <n>
<vp> ::= <tv>, <np> / <v>
<det> ::= the / a / an / every
<n> ::= boy / elephant / plants
<tv> ::= loves / likes
<v> ::= walks

Predicate:

s(stringlist, emptylist).

↓

i.e. non-terminals

Date:_____

Proof tree / Search tree:

s([a, boy, likes, an, elephant], []).

↓

true

s([a, boy, likes, an, elephant], [])

① ↓

np([a, boy, likes, an, elephant], [])

② ↓

det([a, boy, likes, an, elephant], [])

④ ↓

det([boy, likes, an, elephant], [])

⑤ ↓

n([boy, likes, an, elephant], [])

⑤ ↓

n([likes, an, elephant], [])

③ ↓

vp([likes, an, elephant], [])

⑥ ↓

tv([an, elephant], [])

② ↓

np([an, elephant], [])

④ ↓

det([an, elephant], [])

④ ↓

det([elephant], [])

⑤ ↓

n([elephant], [])

⑤ ↓

n([], [])

true (succeed)

Date:_____

s([a, boy, liker, an, elpnt], []).

↓

false

s([a, boy, liker, an, elpnt], [])

① ↓

np([a, boy, liker, an, elpnt], [])

② ↓

det([a, boy, liker, an, elpnt], [])

④ ↓

det([boy, liker, an, elpnt], [])

⑤ ↓

n([boy, liker, an, elpnt], [])

⑤ ↓

n([liker, an, elpnt], [])

③ ↓

vp([liker, an, elpnt], [])

⑥ ↓

tv([liker, an, elpnt], [])

⑥ ↓

tv([an, elpnt], [])

② ↓

np([an, elpnt], [])

④ ↓

det([an, elpnt], [])

④ ↓

det([elpnt], [])

⑤ ↓

n([elpnt], [])

↓

fails (false)

SWI-Prolog -- d:/Prolog/Parsing-of_CFG.pl

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% d:/prolog/parsing-of_cfg compiled 0.00 sec, -2 clauses
% d:/prolog/parsing-of_cfg compiled 0.02 sec, 0 clauses
?- s([a, boy, likes, an ,elephant], []).
true .

?- s([the, park, loves, every, elephant], []).
true .

?- s([every, elephant, likes, an, apple], []).
true .

?-  s([every, elephant, walks], []).
true.

?- s([a, boy, lks, an, elpnt], []).
false.

?-
```

parsing-of_cfg.pl [modified]

File Edit Browse Compile Prolog Pce Help

parsing-of_cfg.pl [modified]

```
/*
 s([a, boy, likes, an ,elephant], []).
 s([the, park, loves, every, elephant], []).
 s([every, elephant, likes, an, apple], []).
 s([every, elephant, walks], []).
*/


s(S0,S) :- np(S0,S1), vp(S1,S).
np(S0,S) :- det(S0,S1), n(S1,S).
vp(S0,S) :- tv(S0,S1), np(S1,S).
vp(S0,S) :- v(S0,S).
det(S0,S) :- S0=[the|S].
det(S0,S) :- S0=[a|S].
det(S0,S) :- S0=[an|S].
det(S0,S) :- S0=[every|S].
n(S0,S) :- S0=[boy|S].
n(S0,S) :- S0=[elephant|S].
n(S0,S) :- S0=[park|S].
n(S0,S) :- S0=[apple|S].
tv(S0,S) :- S0=[loves|S].
tv(S0,S) :- S0=[likes|S].
v(S0,S) :- S0=[walks|S].
```

Date:_____

Conclusion: Hence, I have learned how to implement first order logic - Parsing of context free grammar using prolog.