Assignment no. B8
Roll no. 4202

1 Title

SLR Parsing Algorithm.

2 Problem Definition

Write a program to implement SLR Parsing algorithm using python for the ordered input set in XML
$P \rightarrow E$,
$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow i$

3 Objective

- To understand various parsing techniques.
- To implement SLR parsing using python.

4 Pre-requisite

- Knowledge of parsing techniques.
- Knowledge of python programming.

5 Software and Hardware Requirements

1. 64 bit Machine i3/i5/i7
2. 64-bit open source Linux OS Fedora 20
3. Python
4. Eclipse

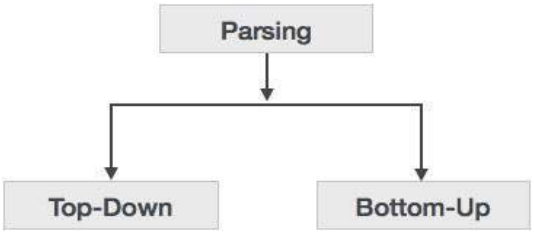
Theory

Parsing :

- Every programming languages has a set of rules that must be followed by the programmer and it is called as the syntax of the language.
- So its the compiler duty to check if the programmer has followed the rule of the language. The second phase of the compiler is assigned the task of checking the syntax and thus it is called syntax analyser and parser.
- It takes the input from a lexical analyser in the form of token streams.
- The parser analyzes the source code (token stream) against the production rules to detect any errors in the code. The output of this phase is a parse tree.
- This way, the parser accomplishes two tasks, i.e., parsing the code, looking for errors and generating a parse tree as the output of the phase.

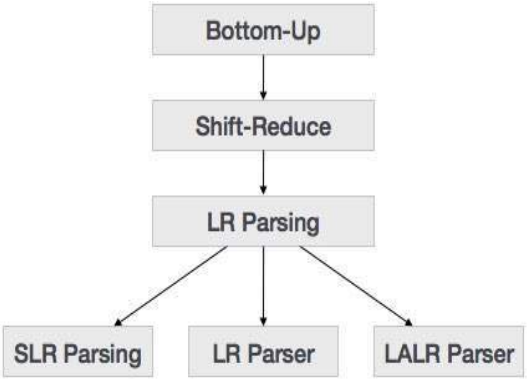
Parsing Strategies :

Syntax analyzers follow production rules defined by means of context-free grammar.
The way the production rules are implemented (derivation) divides parsing into two types : top-down parsing and bottom-up parsing.



Bottom up Parsing technique :

- In bottom parsing start with the input string and try to obtain the starting symbol of the grammar given using successive reductions.
- Bottom-up parser considers the right most derivation of the input string.
- In bottom-up parsing we can create a parse tree from leaves and proceeds towards the root, which is attempting to construct the parse tree in the bottom-up.



SLR Parsing Techniques :

SLR(1) are the Simple LR easiest to implement but weak in terms of the members of the grammar in which it exceeds.
Parsing tables are constructed by this method is called as SLR table.

LR(0) items :

- A LR(0)item of a grammar G is the production of G eith the dot at some position of the right side. Thus the production A $\rightarrow$ XYZ yields the four items.
  A $\rightarrow$. XYZ
  A $\rightarrow$ X . YZ
  A $\rightarrow$ XY . Z
  A $\rightarrow$ XYZ .

  The production A $\rightarrow \epsilon$ generates A $\rightarrow$ . .

- To find of the states of the SLR parser, we group items into sets called canonical LR(0). We define an augmented grammar and two functions, closure and goto.
- Augmented grammar is the grammar G with new introduced non terminal S' and the production S' $\rightarrow$ S. The production S' $\rightarrow$ S is introduced to indicate the parsing and announce the acceptance of input.

Algorithm for SLR Parsing table :

Input : An augmented grammar G'.

Output : The SLR parsing table functions action and goto for G'.

Method :

1. Construct C = { I_0,I_1.....I_n } collection of set of LR(0) items for G'.
2. State I constructed from I_i . The parsing actions for state I are determined as follows:
   (a) If [A $\rightarrow \alpha$.aβ] is in I_i and goto(I_i ,a)=I_j then set action [i,a] to "shift j " .
   (b) If [A $\rightarrow \alpha$.aβ] is in I_i then set action [i,a] to reduce A $\rightarrow \alpha$" for all a in FOLLOW(A) A may not be S'.
   (c) If S'$\rightarrow$ S is in I_i then set action [i,$] to accept.
3. The goto transition for state i are constructed for all non-terminal A using the rule; If goto(I_i ,a)=I_j then goto [i,A]=j .
4. All the entries not defined by the rule (2) and (3) are error.
5. The initial state of the parser is the one constructed from the set of items obtaining : [S' $\rightarrow$ S].

Limitations of Syntax Analyzers :

Syntax analyzers receive their inputs, in the form of tokens, from lexical analyzers. Lexical analyzers are responsible for the validity of a token supplied by the syntax analyzer. Syntax analyzers have the following drawbacks -

- it cannot determine if a token is valid,
- it cannot determine if a token is declared before it is being used,
- it cannot determine if a token is initialized before it is being used,
- it cannot determine if an operation performed on a token type is valid or not.

These tasks are accomplished by the semantic analyzer, which we shall study in Semantic Analysis.

6 Mathematical Model :

Let S be the solution perspective of the system such that,
S ={S_t,E_t,I,O,DD,NDD,F_{me},Sc,Fc}
where,

S_t = start state represents grammar productions.

I represents set of input
I = {parsing table,terminals,non- terminals, production rules,input string }

production rules = lhs -¿ rhs input string = [a-z]* ; terminals ∈ [a-z] nonterminals ∈ [A-Z]
O represents set of output
O = status
where, status= accept v error

F_{me}=set of functions.
F_{me}={f1,f2,f3}
where,
f1= f1 represents the function to accept parse table.
f2= f2 represents the function to parse string
f3= f3 to display the result of acceptance.

E_t=display result.

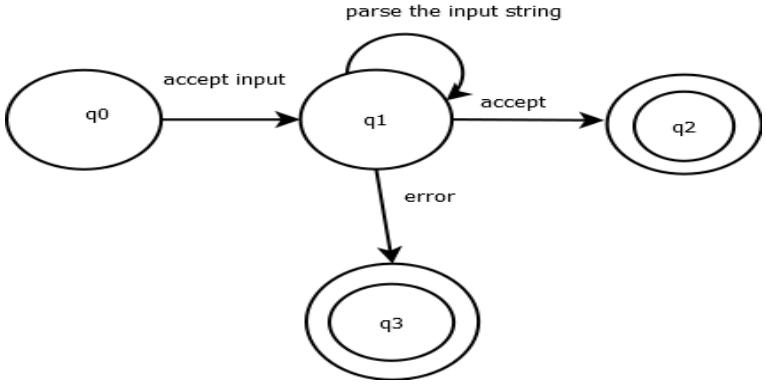
DD = Deterministic Data
= parsing table

NDD (Non Deterministic Data)
= result of parsing

Sc = status=input string is accepted

Fc = status= parse table does not have entry while parsing string

7 State Diagram :



8 Conclusion

We have thus successfully implemented SLR parsing technique using Python.