Niterói, RJ - 22/07/2021
NOME: VICTORIA VELASCO TATE
MATRÍCULA: 217031118

EXERCÍCIO DE ANÁLISE DE CONFLITOS SHIFT-REDUCE COM O MÉTODO SLR EM COMPARAÇÃO COM O MÉTODO LALR.

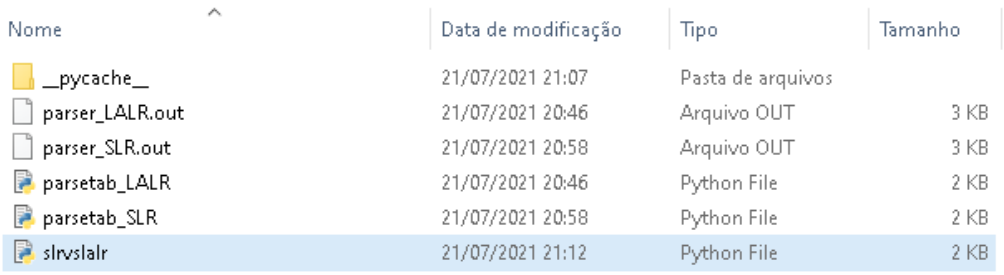
UTILIZANDO *PYTHON* E A BIBLIOTECA *PLY*.

- IMPLEMENTE ANALISADOR LÉXICO E SINTÁTICO DA GRAMÁTICA

  S -> L = R | R
  L -> * R | id
  R -> L

  NO ARQUIVO “slrvslalr.py”:

slrvslalr.py



Nome	Data de modificação	Tipo	Tamanho
__pycache__	21/07/2021 21:07	Pasta de arquivos	
parser_LALR.out	21/07/2021 20:46	Arquivo OUT	3 KB
parser_SLR.out	21/07/2021 20:58	Arquivo OUT	3 KB
parsetab_LALR	21/07/2021 20:46	Python File	2 KB
parsetab_SLR	21/07/2021 20:58	Python File	2 KB
slrvslalr	21/07/2021 21:12	Python File	2 KB

```
13    #   ANÁLISE LÉXICA ----------------------------------------------
14    |
15    #   TOKENS
16    tokens = ('ID','EQUALS', 'STAR')
17
18    t_STAR   = r'\*'
19    t_EQUALS = r'='
20    t_ID     = r'id'
21
22    #   IGNORANDO ESPAÇOS E TABS
23    t_ignore  = ' \t'
24
25    def t_newline(t):
26        r'\n+'
27        t.lexer.lineno += t.value.count("\n")
28
29    #   VALIDANDO CARACTERES LIDOS DE ACORDO COM OS TOKENS MAPEADOS
30    def t_error(t):
31        print("Illegal character '%s'" % t.value[0])
32        t.lexer.skip(1)
33        |
34
35
36
37    #   ANÁLISE SINTÁTICA -------------------------------------------
38
39    #   DEFININDO A GRAMÁTICA
40    def p_expression(t):
41        '''
42            S : L EQUALS R
43              | R
44            L : STAR R
45              | ID
46            R : L
47            |
48        '''
49        if   t[1] == '='  : t[0] = t[2]
50        elif t[1] == '*'  : t[0] = t[2]
51        elif t[1] == 'id' : t[0] = ''
52        else              : t[0] = t[1]
53
```

- ESCOLHER ENTRE SLR OU LALR COMO ALGORITMO PARA CONSTRUÇÃO DA TABELA DE PARSING

```
62    lexer   = lex.lex()
63
64    #   DEFININDO MÉTODO UTILIZADO
65    op      = input('Escolha o método: \n [1] SLR \n [2] LALR \n -> ')
66    if op == 1  :   parser = yacc.yacc(method='SLR')
67    else        :   parser = yacc.yacc()
68
69    #   RECEBENDO INPUT PARA ANÁLISE
70    while True:
71        try:
72            s = input('S -> ')   # Use raw_input on Python 2
73        except EOFError:
74            break
75        parser.parse(s)
76
```

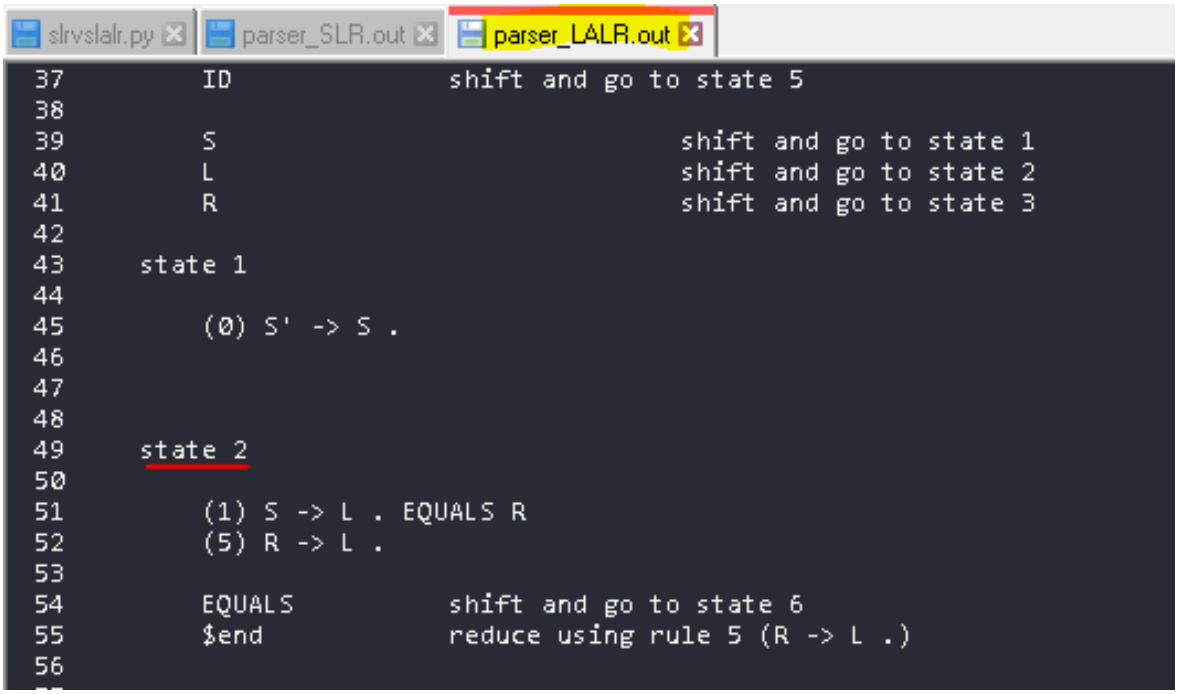
__pycache__	21/07/2021 21:07	Pasta de arquivos	
parser_LALR.out	21/07/2021 20:46	Arquivo OUT	3 KB
parser_SLR.out	21/07/2021 20:58	Arquivo OUT	3 KB
parsetab_LALR	21/07/2021 20:46	Python File	2 KB
parsetab_SLR	21/07/2021 20:58	Python File	2 KB
slrvslalr	21/07/2021 21:45	Python File	2 KB

DEMONSTRAR, UTILIZANDO OS ARQUIVOS GERADOS POR PLY, QUE A TABELA SLR POSSUI UM CONFLITO E A LALR NÃO.

```
victa@LAPTOP-ASVVMRA8 MINGW64 ~/Documents/UFF/UFF 20211/COMPILADORES/slr-vs-lalr-victate/slrvslalr (main)
$ python slrvslalr.py
Escolha o método:
 [1] SLR
 [2] LALR
 -> 1
Generating SLR tables
WARNING: 1 shift/reduce conflict
S -> |
```

slrvslalr.py | parser_SLR.out | parser_LALR.out

```
33          (4) L -> . ID
34          (5) R -> . L
35
36          STAR            shift and go to state 4
37          ID              shift and go to state 5
38
39          S                           shift and go to state 1
40          L                           shift and go to state 2
41          R                           shift and go to state 3
42
43      state 1
44
45          (0) S' -> S .
46
47
48
49      state 2
50
51          (1) S -> L . EQUALS R
52          (5) R -> L .
53
54        ! shift/reduce conflict for EQUALS resolved as shift
55          EQUALS          shift and go to state 6
56          $end            reduce using rule 5 (R -> L .)
57
58        ! EQUALS          [ reduce using rule 5 (R -> L .) ]
59
60
61      state 3
62
63          (2) S -> R .
64
65          $end            reduce using rule 2 (S -> R .)
66
```




parser_SLR.out


parser_LALR.out

PROBLEMA DE CONFLITO NO MÉTODO SLR:

NA GRAMÁTICA CONSIDERADA, CADA MÉTODO IRÁ EXECUTAR O PARSER DE UMA FORMA DIFERENTE.
Na gramática considerada, cada método irá executar o parser de uma forma diferente.

Para o caso do 2° estado, como demonstrado na imagem, o método de LALR, só elenca o “SHIFT AND GO TO STATE 6”.

Já para o método SLR, são considerados para o mesmo ACTION:

- “SHIFT AND GO PAR O STATE 6”
- REDUCE “L → R”

Isso ocasiona um conflito de ambiguidade para o método SLR, embora a gramática não seja ambígua. Com o método de LALR, isso não ocorre pois ocorre um agrupamento de estados que possuem o mesmo CORE (conjunto de elementos que são os primeiros componentes em um LR(1). Isso faz com que ocorra, portanto, um agrupamento de GOTO, resolvendo conflitos de SHIFT-REDUCE.

```
# -----------------------------------------------------------------------------
# slrvslalr.py
#
# VICTORIA VELASCO TATE
# 217031118
# -----------------------------------------------------------------------------

#   ANÁLISE LÉXICA ----------------------------------------------

#   TOKENS
tokens = ('ID','EQUALS', 'STAR')

t_STAR   = r'\*'
t_EQUALS = r'='
t_ID     = r'id'

#   IGNORANDO ESPAÇOS E TABS
t_ignore  = ' \t'

def t_newline(t):
   r'\n+'
   t.lexer.lineno += t.value.count("\n")

#   VALIDANDO CARACTERES LIDOS DE ACORDO COM OS TOKENS MAPEADOS
def t_error(t):
   print("Illegal character '%s'" % t.value[0])
   t.lexer.skip(1)




#   ANÁLISE SINTÁTICA -------------------------------------------

#   DEFININDO A GRAMÁTICA
def p_expression(t):
   '''
      S : L EQUALS R
        | R
      L : STAR R
        | ID
      R : L

   '''
   if   t[1] == '='  : t[0] = t[2]
   elif t[1] == '*'  : t[0] = t[2]
   elif t[1] == 'id' : t[0] = ''
   else              : t[0] = t[1]
```

```
#   VALIDANDO PROBLEMAS DE SINTAXE
def p_error(t):
    print("Syntax error at '%s'" % t.value)


import ply.yacc as yacc
import ply.lex as lex

lexer   = lex.lex()

#   DEFININDO MÉTODO UTILIZADO
op = input('Escolha o método: \n [1] SLR \n [2] LALR \n -> ')
if op == '1'  :   parser = yacc.yacc(method='SLR')
else          :   parser = yacc.yacc(method)


#   RECEBENDO INPUT PARA ANÁLISE
while True:
    try:
        s = input('S -> ')   # Use raw_input on Python 2
    except EOFError:
        break
    parser.parse(s)
```

```
# --------------------------------------------------------------------------
# parser_LALR.out
#
# --------------------------------------------------------------------------

Created by PLY version 3.11 (http://www.dabeaz.com/ply)

Grammar

Rule 0     S' -> S
Rule 1     S -> L EQUALS R
Rule 2     S -> R
Rule 3     L -> STAR R
Rule 4     L -> ID
Rule 5     R -> L

Terminals, with rules where they appear

EQUALS             : 1
ID                 : 4
STAR               : 3
error              :

Nonterminals, with rules where they appear

L                  : 1 5
R                  : 1 2 3
S                  : 0

Parsing method: LALR

state 0

    (0) S' -> . S
    (1) S -> . L EQUALS R
    (2) S -> . R
    (3) L -> . STAR R
    (4) L -> . ID
    (5) R -> . L

    STAR           shift and go to state 4
    ID             shift and go to state 5

    S                          shift and go to state 1
    L                          shift and go to state 2
    R                          shift and go to state 3
```

state 1

(0) S' -> S .

state 2

(1) S -> L . EQUALS R
(5) R -> L .

EQUALS shift and go to state 6
$end reduce using rule 5 (R -> L .)

state 3

(2) S -> R .

$end reduce using rule 2 (S -> R .)

state 4

(3) L -> STAR . R
(5) R -> . L
(3) L -> . STAR R
(4) L -> . ID

STAR shift and go to state 4
ID shift and go to state 5

R shift and go to state 7
L shift and go to state 8

state 5

(4) L -> ID .

EQUALS reduce using rule 4 (L -> ID .)
$end reduce using rule 4 (L -> ID .)

state 6

(1) S -> L EQUALS . R
(5) R -> . L

(3) L -> . STAR R
(4) L -> . ID

STAR shift and go to state 4
ID shift and go to state 5

L shift and go to state 8
R shift and go to state 9

state 7

(3) L -> STAR R .

EQUALS reduce using rule 3 (L -> STAR R .)
$end reduce using rule 3 (L -> STAR R .)

state 8

(5) R -> L .

EQUALS reduce using rule 5 (R -> L .)
$end reduce using rule 5 (R -> L .)

state 9

(1) S -> L EQUALS R .

$end reduce using rule 1 (S -> L EQUALS R .)

```
# --------------------------------------------------------------------------
# parser_SLR.out
#
# --------------------------------------------------------------------------

Created by PLY version 3.11 (http://www.dabeaz.com/ply)

Grammar

Rule 0     S' -> S
Rule 1     S -> L EQUALS R
Rule 2     S -> R
Rule 3     L -> STAR R
Rule 4     L -> ID
Rule 5     R -> L

Terminals, with rules where they appear

EQUALS             : 1
ID                 : 4
STAR               : 3
error              :

Nonterminals, with rules where they appear

L                  : 1 5
R                  : 1 2 3
S                  : 0

Parsing method: SLR

state 0

    (0) S' -> . S
    (1) S -> . L EQUALS R
    (2) S -> . R
    (3) L -> . STAR R
    (4) L -> . ID
    (5) R -> . L

    STAR           shift and go to state 4
    ID             shift and go to state 5

    S                       shift and go to state 1
    L                       shift and go to state 2
    R                       shift and go to state 3
```

state 1

(0) S' -> S .

state 2

(1) S -> L . EQUALS R
(5) R -> L .

! shift/reduce conflict for EQUALS resolved as shift
EQUALS shift and go to state 6
$end reduce using rule 5 (R -> L .)

! EQUALS [reduce using rule 5 (R -> L .)]

state 3

(2) S -> R .

$end reduce using rule 2 (S -> R .)

state 4

(3) L -> STAR . R
(5) R -> . L
(3) L -> . STAR R
(4) L -> . ID

STAR shift and go to state 4
ID shift and go to state 5

R shift and go to state 7
L shift and go to state 8

state 5

(4) L -> ID .

EQUALS reduce using rule 4 (L -> ID .)
$end reduce using rule 4 (L -> ID .)

state 6

(1) S -> L EQUALS . R
(5) R -> . L
(3) L -> . STAR R
(4) L -> . ID

STAR shift and go to state 4
ID shift and go to state 5

L shift and go to state 8
R shift and go to state 9

state 7

(3) L -> STAR R .

EQUALS reduce using rule 3 (L -> STAR R .)
$end reduce using rule 3 (L -> STAR R .)

state 8

(5) R -> L .

$end reduce using rule 5 (R -> L .)
EQUALS reduce using rule 5 (R -> L .)

state 9

(1) S -> L EQUALS R .

$end reduce using rule 1 (S -> L EQUALS R .)

WARNING:
WARNING: Conflicts:
WARNING:
WARNING: shift/reduce conflict for EQUALS in state 2 resolved as shift