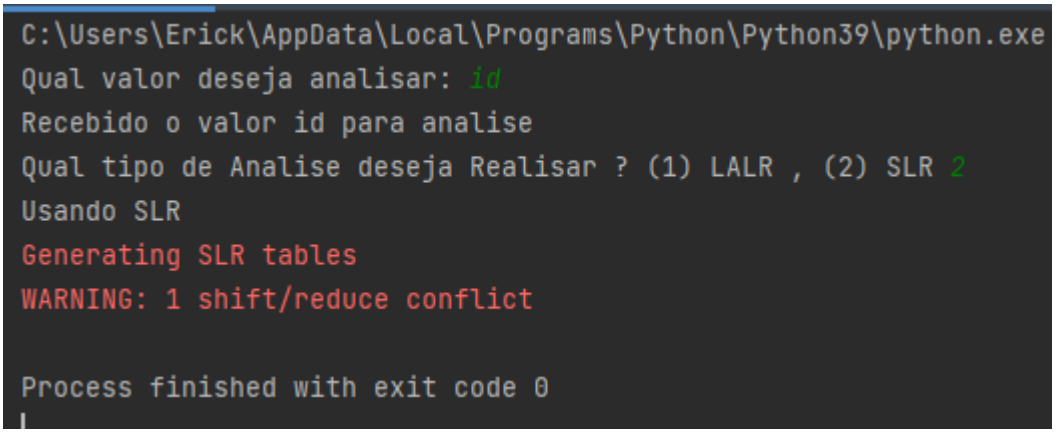
Erick Guimarães de Oliveira - 816031129 - 23/07/2021

Dada a gramática:

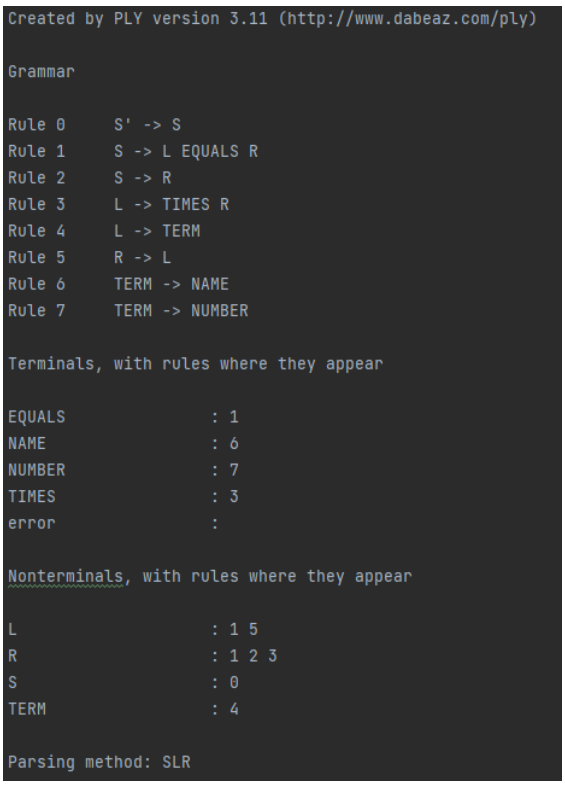
S -> L = R | R
L -> * R | id
R -> L

O problema deste trabalho consiste que a tabela de parsing SLR é ambígua para ela, gerando problemas de decisão. A LALR por sua vez não é.

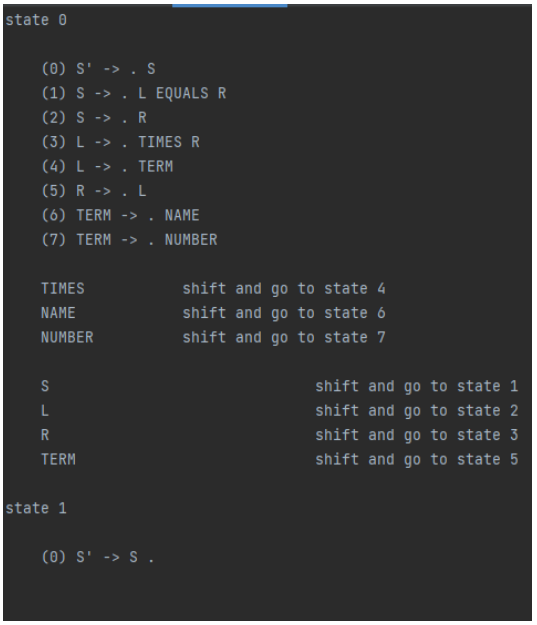
Ao gerar a tabela de de parsing utilizando SLR o programa PLY responde com o seguinte aviso. WARNING 1 shift/reduce conflict o que demonstra sua ambiguidade



O PLY, gera dois arquivos na saída um chamado de parse.out e outro com a tabela de parsing que é o parsetab.py e o conteúdo do parse.out é:

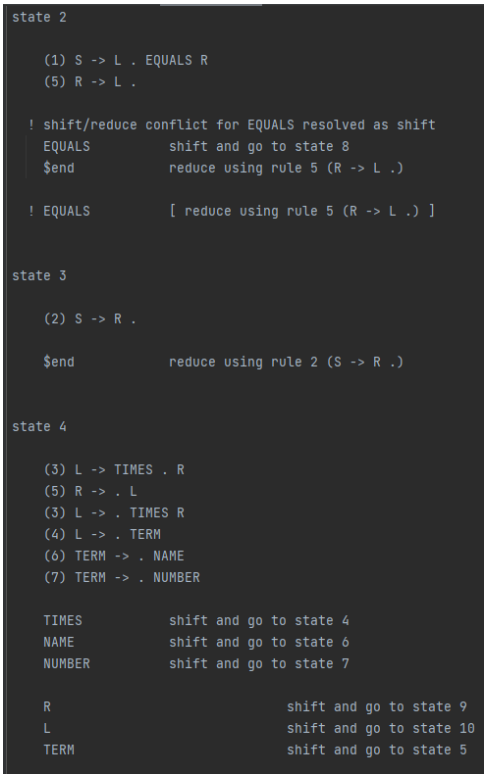


```
Created by PLY version 3.11 (http://www.dabeaz.com/ply)

Grammar

Rule 0     S' -> S
Rule 1     S -> L EQUALS R
Rule 2     S -> R
Rule 3     L -> TIMES R
Rule 4     L -> TERM
Rule 5     R -> L
Rule 6     TERM -> NAME
Rule 7     TERM -> NUMBER

Terminals, with rules where they appear

EQUALS               : 1
NAME                 : 6
NUMBER               : 7
TIMES                : 3
error                :

Nonterminals, with rules where they appear

L                    : 1 5
R                    : 1 2 3
S                    : 0
TERM                 : 4

Parsing method: SLR
```

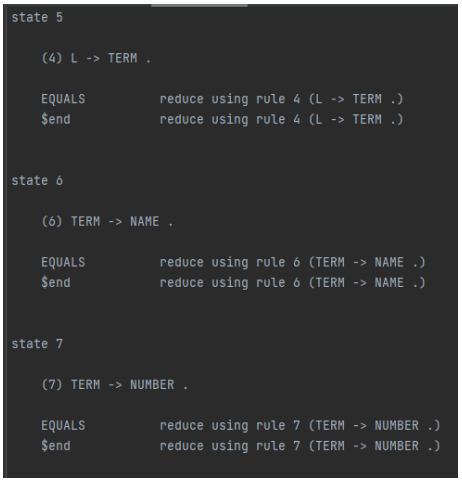


```
state 0

    (0) S' -> . S
    (1) S -> . L EQUALS R
    (2) S -> . R
    (3) L -> . TIMES R
    (4) L -> . TERM
    (5) R -> . L
    (6) TERM -> . NAME
    (7) TERM -> . NUMBER

    TIMES           shift and go to state 4
    NAME            shift and go to state 6
    NUMBER          shift and go to state 7

    S                               shift and go to state 1
    L                               shift and go to state 2
    R                               shift and go to state 3
    TERM                            shift and go to state 5

state 1

    (0) S' -> S .
```

Pode-se notar o método de parsing na primeira imagem. (SLR)



```
state 2

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

  ! shift/reduce conflict for EQUALS resolved as shift
    EQUALS          shift and go to state 8
    $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 -> TIMES . R
    (5) R -> . L
    (3) L -> . TIMES R
    (4) L -> . TERM
    (6) TERM -> . NAME
    (7) TERM -> . NUMBER

    TIMES           shift and go to state 4
    NAME            shift and go to state 6
    NUMBER          shift and go to state 7

    R                           shift and go to state 9
    L                           shift and go to state 10
    TERM                        shift and go to state 5
```



```
state 5

    (4) L -> TERM .

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


state 6

    (6) TERM -> NAME .

    EQUALS          reduce using rule 6 (TERM -> NAME .)
    $end            reduce using rule 6 (TERM -> NAME .)


state 7

    (7) TERM -> NUMBER .

    EQUALS          reduce using rule 7 (TERM -> NUMBER .)
    $end            reduce using rule 7 (TERM -> NUMBER .)
```

```
state 8

    (1) S -> L EQUALS . R
    (5) R -> . L
    (3) L -> . TIMES R
    (4) L -> . TERM
    (6) TERM -> . NAME
    (7) TERM -> . NUMBER

    TIMES           shift and go to state 4
    NAME            shift and go to state 6
    NUMBER          shift and go to state 7

    L                           shift and go to state 10
    R                           shift and go to state 11
    TERM                        shift and go to state 5

state 9

    (3) L -> TIMES R .

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


state 10

    (5) R -> L .

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

```
state 11

    (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
```

pode-se ver aqui também, que no estado 11 o programa gera o conflito e avisa que resolveu com um shift.

Utilizando o LALR por sua vez, o resultado se dá como

```
C:\Users\Erick\AppData\Local\Programs\Python\Python39\python.exe
Arquivo de saída já existe, removendo arquivo
parsetable já existe, removendo arquivo
Qual valor deseja analisar: id
Recebido o valor id para analise
Qual tipo de Analise deseja Realisar ? (1) LALR , (2) SLR 1
Usando LALR
Generating LALR tables

Process finished with exit code 0
```

e o conteúdo da parse.out é o seguinte:

```
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 -> TIMES R
Rule 4     L -> TERM
Rule 5     R -> L
Rule 6     TERM -> NAME
Rule 7     TERM -> NUMBER

Terminals, with rules where they appear

EQUALS               : 1
NAME                 : 6
NUMBER               : 7
TIMES                : 3
error                :

Nonterminals, with rules where they appear

L                    : 1 5
R                    : 1 2 3
S                    : 0
TERM                 : 4

Parsing method: LALR
```

```
state 0

    (0) S' -> . S
    (1) S -> . L EQUALS R
    (2) S -> . R
    (3) L -> . TIMES R
    (4) L -> . TERM
    (5) R -> . L
    (6) TERM -> . NAME
    (7) TERM -> . NUMBER

    TIMES           shift and go to state 4
    NAME            shift and go to state 6
    NUMBER          shift and go to state 7

    S                               shift and go to state 1
    L                               shift and go to state 2
    R                               shift and go to state 3
    TERM                            shift and go to state 5

state 1

    (0) S' -> S .



state 2

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

    EQUALS          shift and go to state 8
    $end            reduce using rule 5 (R -> L .)
```

Pode-se notar o método de parsing na primeira imagem. (LALR)

```
state 3

    (2) S -> R .

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


state 4

    (3) L -> TIMES . R
    (5) R -> . L
    (3) L -> . TIMES R
    (4) L -> . TERM
    (6) TERM -> . NAME
    (7) TERM -> . NUMBER

    TIMES           shift and go to state 4
    NAME            shift and go to state 6
    NUMBER          shift and go to state 7

    R                           shift and go to state 9
    L                           shift and go to state 10
    TERM                        shift and go to state 5

state 5

    (4) L -> TERM .

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


state 6

    (6) TERM -> NAME .

    EQUALS          reduce using rule 6 (TERM -> NAME .)
    $end            reduce using rule 6 (TERM -> NAME .)
```

```
state 7

    (7) TERM -> NUMBER .

    EQUALS          reduce using rule 7 (TERM -> NUMBER .)
    $end            reduce using rule 7 (TERM -> NUMBER .)


state 8

    (1) S -> L EQUALS . R
    (5) R -> . L
    (3) L -> . TIMES R
    (4) L -> . TERM
    (6) TERM -> . NAME
    (7) TERM -> . NUMBER

    TIMES           shift and go to state 4
    NAME            shift and go to state 6
    NUMBER          shift and go to state 7

    L                           shift and go to state 10
    R                           shift and go to state 11
    TERM                        shift and go to state 5

state 9

    (3) L -> TIMES R .

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


state 10

    (5) R -> L .

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

```
state 11

    (1) S -> L EQUALS R .

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

Não identificando assim, nenhum problema ao gerar a tabela de parsing.

Ou seja, o PLY identifica com WARNINGS quando há problemas de parsing. e não mostra nada quando não houve nenhum conflito, na saída do programa e gera arquivo de saída coms as etapas falando onde o conflito ocorreu e como foi resolvido.