

Atividade Compiladores 2021.1

Aluno: Victor Ferreira Teixeira da Silva

Matrícula: 114031056

data: 23/07/2021

Objetivo

Mostre na prática que a tabela de parsing SLR é ambígua e que a LALR não é utilizando o gerador de analisador sintático PLY, uma implementação em Python da ferramenta Yacc. Dada a gramática:

$S \rightarrow L = R \mid R$
 $L \rightarrow * R \mid id$
 $R \rightarrow L$

Iniciamos com a implementação do lexer que é necessária para realizar a interpretação dos tokens:

```
tokens = ('id', 'MULT', 'EQ')

t_EQ = r'='
t_MULT = r'\*'
t_ignore = '\t'

def t_error(t):
    print("Caractere '%s' inválido" % t.value[0])
    t.lexer.skip(1)
```

o formato das variáveis utilizadas para denotar cada token é segue o formato padrão da biblioteca PLY, sendo este ***t_NOMETOKEN***

As funções a seguir definem e representam a gramática inicialmente definida pela linguagem, também seguindo o formato padrão PLY de nomenclatura.

```
def p_equality(p):
    """
    S : L EQ R
    """
    return p

def p_initial_simple_transform(p):
    """
    S : R
    """
    return p

def p_multiplication(p):
    """
    L : MULT R
    """
    return p

def p_define_id(p):
    """
    L : id
    """
    return p

def p_simple_transform(p):
    """
    R : L
    """
    return p

def p_error(p):
    print(f"Erro de sintaxe {p.value!r}")
```

You, 8 hours ago • implementacao da verificacao de amb

Por fim os parsers são gerados, um aplicando SLR e outro LALR permitindo validar evidenciar a ambiguidade da no parser SLR.

```
if __name__ == '__main__':  
    yacc(method='SLR', debugfile='slr_parser.out', tabmodule='slr_parsetab')  
    yacc(method='LALR', debugfile='lalr_parser.out', tabmodule='lalr_parsetab')
```

O output ao executar o programa é

```
> python3 slrvslalr.py  
Generating SLR tables  
WARNING: 1 shift/reduce conflict  
Generating LALR tables
```

Este conflito **shift reduce** evidencia o problema ao aplicar o parser SLR, ao inspecionar a saída gerada no arquivo **slr_parser.out** temos:

```
48  
49 state 2  
50  
51 (1) S -> L . EQ R  
52 (5) R -> L .  
53  
54 ! shift/reduce conflict for EQ resolved as shift  
55 EQ shift and go to state 6  
56 $end reduce using rule 5 (R -> L .)  
57  
58 ! EQ [ reduce using rule 5 (R -> L .) ]  
59  
60
```

Como a possibilidade de executar o **shift** leva a um estado, e o executar o **reduce** leva a outro temos a evidência do conflito. Verificando o este mesmo estado, porém no parser

LALR (*arquivo lalr_parser.out*) temos:

```
state 2

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

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

Aqui a ambiguidade oriunda de caminhos distintos ao executar o **shift** ou **reduce** não está presente. Deste modo podemos evidenciar que a utilização do parser SLR apresenta ambiguidade enquanto esta gramática é interpretada corretamente pelo parser LALR.