

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 segue a convenção padrão da biblioteca PLY, sendo este ***t_NOMETOKEN***

As funções a seguir definem/representam a gramática inicialmente proposta, também seguindo o formato padrão PLY de nomenclatura ***p_descricao_producao***.

```
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 e evidenciar a ambiguidade na tabela do parser SLR.

```
# gerando parsers SLR e LALR e salvando saidas na pasta outputs
if __name__ == '__main__':
    yacc(method='SLR', debugfile='slr_parser.out', tabmodule='slr_parsetab')
    yacc(method='LALR', debugfile='lalr_parser.out', tabmodule='lalr_parsetab')

    # criando pasta de outputs
    output_folder = "outputs"
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    # movendo os arquivos gerados para pasta outputs
    output_files = ["slr_parser.out", "lalr_parser.out", "slr_parsetab.py", "lalr_parsetab.py"]
    for file in output_files:
        shutil.move(file, f"{output_folder}/{file}")
```

victorl2, 3 minutes ago • atualizando localizacao arquivos

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 **outputs/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 é possível realizar o **shift** ou o **reduce**, não há uma maneira determinista de inferir qual operação deve ser realizada, um conflito sobre qual "transição" deve ser

aplicada. Porém no parser LALR (*arquivo outputs/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 .)
```

A ambiguidade oriunda de caminhos ao executar o **shift** ou **reduce** vista no parser SLR não está presente no parser LALR. Deste modo podemos evidenciar que a utilização do parser SLR apresenta ambiguidade, enquanto a gramática é interpretada corretamente sem qualquer ambiguidade pelo parser LALR.