SLR vs LALR

Analisador léxico e sintático em PLY

Rafael Duarte Campbell (117031036) – 23 de Julho de 2021

Objetivo

Em contrapartida ao ganho em simplicidade sobre o método LR, os analisadores SLR[1] podem encontrar conflitos ao analisar algumas gramáticas. Dito isso, uma gramática SLR é aquela que, quando analisada por um *parser* SLR, é convertida em uma tabela sem conflitos *shift/reduce* ou *reduce/reduce*. No objetivo de contornar esses conflitos, o método LALR[2] considera um *lookahead*.

Desse modo, algumas linguagens não ambíguas podem ser descritas como não-SLR, gerando tabelas coerentes com o método LALR e encontrando conflito quando analisadas por um parser SLR. Um exemplo desse conjunto de gramáticas é o descrito abaixo.

$$
\begin{aligned}
S &\rightarrow L = R \mid R \\
L &\rightarrow *R \mid id \\
R &\rightarrow L
\end{aligned}
$$

O objetivo do experimento é construir, com ajuda do framework PLY, um analisador léxico e sintático para a gramática usando ambos os métodos SLR e LALR. Deste modo, deve-se observar que a tabela SLR será ambígua e a LALR, não; logo, a gramática é não-ambígua e não-SLR.

Problema

A gramática indicada, ao ser convertida em tabela de parsing, resultará em nove estados (em ambos os métodos). O problema real ocorre no segundo estado, que é dado por:

$$
\begin{aligned}
&(1)\ S \rightarrow L\ .\ EQUAL\ R \\
&(5)\ R \rightarrow L\ .
\end{aligned}
$$

Neste estado, o *parser SLR* encontra duas opções: reduzir (pela regra 5) ou empilhar (para reduzir, depois, pela regra 1). Não contando com um mecanismo de decisão, é indicado um conflito *switch/reduce*.[3]

Código

No geral, serão usadas as implementações *lex.py* e *yacc.py*. Para a construção do analisador léxico, basta definir os *tokens* e as expressões regulares que os descrevem; ao fim, chama-se o método lex.lex().

[1] SLR é a abreviação de *Simple LR*.

[2] LALR é a abreviação de *Look-Ahead LR parser*.

[3] Conforme será indicado mais a frente, o framework PLY "resolve"o conflito optando por empilhar.

```
# lista de tokens
tokens = (
'EQUAL',
'TIMES',
'ID'
)

# Expressões regulares
t_EQUAL   = r'='
t_TIMES   = r'\*'
t_ID      = r'[A-z][A-z0-9]*'
t_ignore  = ' \t\n' #expressões a ignorar

# construindo o lexer
lexer = lex.lex()
```

Para o analisador sintático, descrevem-se as regras de produção na forma de funções. Ao fim, chama-se o método yacc.yacc()[4] com o parâmetro opicional *method* – que usa LALR por padrão, mas aceita SLR também.

```
# Definindo as regras para S
def p_S(p):
    '''S : L EQUAL R
         | R '''
    if len(p) == 4:
        p[0] = p[1] + p[2] + p[3]
    elif len(p) == 2:
        p[0] = p[1]

# Definindo as regras para L
def p_L(p):
    '''L : TIMES R
         | ID '''
    if len(p) == 3:
        p[0] = p[1] + p[2]
    elif len(p) == 2:
        p[0] = p[1]

# Definindo as regras para R
def p_R(p):
    'R : L'
    p[0] = p[1]

#construindo o parser LALR
parserLALR = yacc.yacc(method='LALR')

#construindo o parser SLR
parserSLR = yacc.yacc(method='SLR')
```

Com base na regra de produção, são gerados dois arquivos: parsetab.py e parse.out – que é o que nos interessa. Enquanto o primeiro gera uma implementação do analisador, o segundo conta com a descrição dos estados e da tabela de parser.

[4]Para evitar que a saída seja sobrescrita, é feita a renomeação dos arquivos.

Resultado

Conforme esperado, a saída resultante do método SLR indica um conflito de *switch/reduce* no segundo estado, conforme descrito abaixo.

```
[...]
    state 2

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

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

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

Já o mesmo estado, no caso do LALR, não indica conflito algum.

```
[...]
    state 2

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

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

Dessa forma, fica demonstrado que a gramática é não-ambigua (dado a aceitação pelo parser LALR) e não-SLR (dado o conflito no parser SLR).