Trabalho de Compiladores 2021.1

Aluno: Eduardo Teixeira Carneiro
Matrícula: 319.031.001
Professor: Christiano Braga
Universidade Federal Fluminense
Data: 23/07/2021

Objetivo

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

Dados Iniciais

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

A gramática acima não é ambígua. É LALR mas não SLR. Sendo assim, podemos utilizar os arquivos gerados por PLY, para mostrar que a tabela SLR possui um conflito e a LALR não.

O Código

O código, que pode ser encontrado no arquivo slrvslalr.py, é composto por tokens, que seguem o padrão PLY, definição de funções que descrevem a gramática e os parsers, tanto para LALR quanto para SLR.

A Execução

Na execução de cada analisador, um arquivo de saída é criado, e através desses é possível comparar e analisar as diferenças e, neste caso, o conflito causado pela ambiguidade.

Ao executar o SLR, o terminal informa a seguinte mensagem:

```
WARNING: 1 shift/reduce conflict
```

Olhando o arquivo de saída *parser_slr.out*, podemos encontrar as seguintes linhas no estado 2:

```
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 .) ]
```

Além disso, no fim do arquivo, encontra-se a mensagem:

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

Já a execução do analisador LALR, não resulta em nenhuma mensagem de erro, o arquivo de saída não possui nenhum conflito.

Conclusão

Por mais que a gramática não seja ambígua, a forma como cada analisador executa gera diferença na interpretação.

Na execução do analisador SLR, a ambiguidade está no estado 2, onde é detectado um conflito para o símbolo de “igualdade”, pois há a possibilidade de se fazer um shift para o estado 6 e também de se fazer um *reduce* usando a regra R -> L. Ainda sim, ele executa o shift para o estado 6.

Já o analisador LALR, reconhece a redução inválida de R -> L no estado 2, então executa apenas o shift para o estado 6 sem gerar conflito.