SLR vs LALR

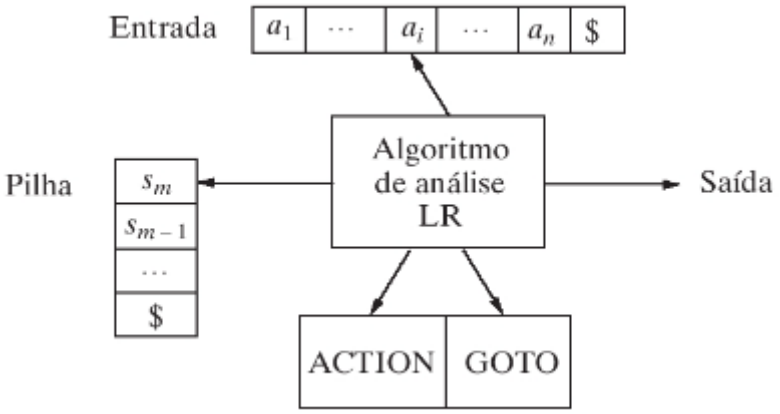
Arthur Almeida Vianna
23/07/2021

Introdução

A tarefa tem como objetivo reproduzir, na prática, um exemplo de **conflito shift/reduce** visto em sala de aula. No exemplo utiliza-se dois tipos de parser LR(*Left-to-Right with Rightmost Derivation*), o **SLR** e o **LALR**, e nele vemos que uma gramática não-ambígua gera um conflito shift/reduce quando utilizamos o SLR.

Parser LR

Na figura abaixo vemos o modelo de um parser LR, ele consiste em uma entrada, uma saída, uma pilha, o algoritmo de análise sintática e uma tabela de análise constituída de duas partes (ACTION E GOTO). O algoritmo é o mesmo para todos os analisadores sintáticos LR, apenas a tabela de análise é diferente de um analisador para outro.



Como mencionado, a tabela de análise é constituída de duas partes: ACTION e GOTO, que são funções que possuem as seguintes características.

1. ACTION: é uma **função de ação de análise,** ela recebe como argumentos um estado **i** e um terminal **a**. O valor de ACTION[i,a] pode ter um dos quatro formatos:
    a. *Shift* **j**: onde **j** é um estado. A ação tomada pelo analisador transfere a entrada **a** para a pilha, mas usa o estado **j** para representar **a**.
    b. Reduce A➡B: A ação do analisador sintático efetivamente reduz B no topo da pilha para A, o lado esquerdo da produção.
    c. Accept: O analisador sintático aceita a entrada e termina a análise
    d. Error: O analisador sintático descobre um erro em sua entrada.
2. GOTO: é uma **função de transição**, para a qual: se GOTO[I_i, A] = I_j, então GOTO também mapeia um estado **i** e um não-terminal **A** para o estado **j**.

PLY

Sobre o PLY

O PLY é uma biblioteca python para prototipação de compiladores bastante completa e de fácil uso, nela temos basicamente dois módulos, **lex** e **yacc**(*Yet Another Compiler Compiler*). O módulo lex é usado para análise léxica, o módulo yacc é responsável pela análise sintática.
Primeiramente definimos os tokens usando o módulo lex, em seguida utilizamos o módulo yacc para construir o parser, feito isso será gerado dois arquivos, o **parsertab.py** que é uma implementação em python da tabela de análise de uma dada gramática, e o **parser.out** que representa a coleção canônica(ou autômato canônico) e informa possíveis erros.

Caso de Estudo

O caso de estudo faz uso da gramática abaixo:

S ➡ L = R| R
L ➡ *R| id
R ➡ L

Esta gramática deve produzir a seguinte coleção canônica:

I_0: $S' \to \cdot S$
$S \to \cdot L = R$
$S \to \cdot R$
$L \to \cdot * R$
$L \to \cdot \mathbf{id}$
$R \to \cdot L$

I_1: $S' \to S\cdot$

I_2: $S \to L\cdot = R$
$R \to L\cdot$

I_3: $S \to R\cdot$

I_4: $L \to *\cdot R$
$R \to \cdot L$
$L \to \cdot * R$
$L \to \cdot \mathbf{id}$

I_5: $L \to \mathbf{id}\cdot$

I_6: $S \to L = \cdot R$
$R \to \cdot L$
$L \to \cdot * R$
$L \to \cdot \mathbf{id}$

I_7: $L \to * R\cdot$

I_8: $R \to L \cdot$

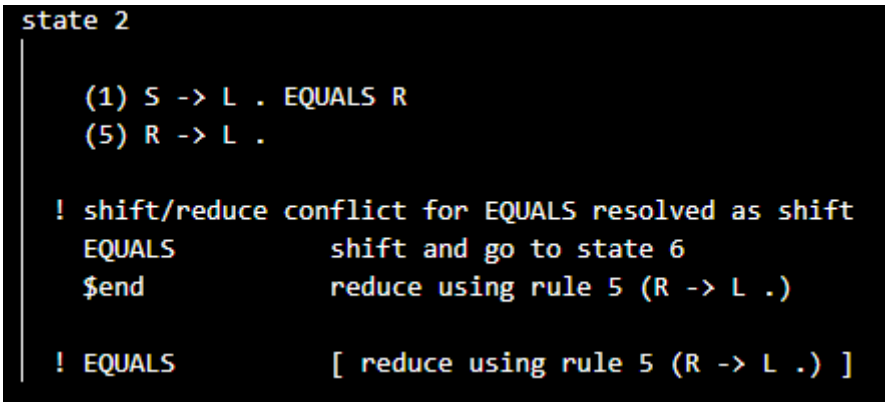
I_9: $S \to L = R\cdot$

Executando

O programa funcionou da forma prevista e gerou uma coleção canônica igual à que foi vista em sala de aula, os arquivos **parser.out** foram renomeados para **SLR_parser.out** e **LALR_parser.out** para facilitar a identificação do método.

SLR_parser.out

Utilizanndo o método de parser SLR, tal como visto em sala é acusado o conflito shift/reduce pela ferramenta python.



No caso do parser SLR este conflito ocorre pois o primeiro item do conjunto 2 faz com que ACTION[2,=] seja "*shift 6*". Como FOLLOW(R) contém "=", o segundo item define ACTION[2,=] como "reduce segundo a produção "R ➡ L". Como existe uma ação de shift e reduce para o estado 2 e a entrada "=" em ACTION[2,=], o estado 2 caracteriza um conflito shift/reduce sob o símbolo de entrada =.
Esse conflito shift/reduce ocorre porque o método SLR não é poderoso o suficiente para lembrar o contexto à esquerda a fim de decidir que ação o reconhecedor deve tomar quando "=" aparece na entrada, tendo visto uma cadeia redutível para L.

LALR_parser.out

Já o método de parser LALR é mais poderoso e é capaz de reconhecer a gramática sem produzir conflitos, e a ferramenta confirmou isso como podemos observar na figura abaixo que contém o mesmo estado que no parser SLR gerou um conflito shift/reduce.