

Compiladores — Folha laboratorial 4

Pedro Vasconcelos, DCC/FCUP

Outubro 2020

Análise sintática LR

Exercício 1

- (a) Mostre que a gramática de parêntesis simples

$$A \rightarrow (A) \mid a$$

é $LR(0)$ construindo o autómato e a tabela de *parsing* LR.

- (b) Mostre que a gramática

$$\begin{array}{ll} T \rightarrow R & R \rightarrow \varepsilon \\ T \rightarrow aTc & R \rightarrow bR \end{array}$$

é $SLR(1)$ construindo o autómato e a tabela de *parsing* LR. Justifique que esta não é gramática $LR(0)$ mostrando que existem conflitos na tabela de *parsing* $LR(0)$.

Exercício 2

Considere a seguinte extensão à gramática de parêntesis:

$$\begin{array}{ll} E \rightarrow (L) & \mid a \\ L \rightarrow L, E & \mid E \end{array}$$

- (a) Mostre uma derivação para a palavra $((a), a, (a, a))$
- (b) Construa o autómato de *parsing* $LR(0)$ da gramática
- (c) Será que a gramática é $LR(0)$? Se não, mostre qual o conflito; se sim, construa a tabela de *parsing*.

Exercício 3

Considere a seguinte gramática simplificada de declarações de variáveis na linguagem C; os terminais são `int`, `float`, vírgula (`,`), ponto-e-vírgula (`;`) e identificadores (*ident*):

$$\begin{aligned} Decl &\rightarrow Type\ Varlist\ ; \\ Type &\rightarrow \text{int} \mid \text{float} \\ Varlist &\rightarrow Varlist, ident \mid ident \end{aligned}$$

- Construa o autômato LR e a tabela de parsing $LR(0)$
- Simule os passos de execução do autômato para a entrada `int x,y,z;`

Exercício 4

Considere a seguinte gramática (ambígua) para expressões aritméticas simples:

$$\begin{array}{lll} E \rightarrow E+E & E \rightarrow E * E & E \rightarrow (E) \\ E \rightarrow E-E & E \rightarrow E/E & E \rightarrow num \end{array}$$

- Implemente uma calculadora para expressões desta gramática usando geradores de analisadores léxicais e sintáticos. Use o *Alex/Happy* com a linguagem Haskell ou o *Flex/Bison* com a linguagem C.

Resolva as ambiguidades usando diretivas de associatividade e precedência. Tenha o cuidado de verificar se a resolução respeita as convenções algébricas usuais. Exemplos:

1+2*3	deve calcular $1 + (2 \times 3)$	(prioridade de <code>*</code> em relação a <code>+</code>)
1+2/3	deve calcular $1 + (2/3)$	(prioridade de <code>/</code> em relação a <code>+</code>)
1-2-3	deve calcular $(1 - 2) - 3$	(associatividade à esquerda)
1/2/3	deve calcular $(1/2)/3$	(associatividade à esquerda)

O seu programa não necessita de construir a *árvore sintática abstrata*; pode efetuar os cálculos durante a análise sintática. O resultado deve ser calculado em vírgula flutuante de precisão simples.

- Modifique a análise lexical, sintática e a função de avaliação para acrescentar algumas funções matemáticas à calculadora:

$$\begin{aligned} E &\rightarrow \dots \mid F(E) \\ F &\rightarrow \text{sqrt} \mid \text{exp} \mid \text{log} \end{aligned}$$