Compiladores — Folha laboratorial 4

Pedro Vasconcelos, DCC/FCUP

Outubro 2020

Análise sintática LR

Exercício 1

(a) Mostre que a gramática de parêntesis simples

$$A \to \texttt{(}A\texttt{)} \quad | \quad \texttt{a}$$

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

(b) Mostre que a gramática

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

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:

$$\begin{array}{lcl} E \to \texttt{(}L\texttt{)} & | & \texttt{a} \\ L \to L\texttt{,}E & | & E \end{array}$$

(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 (`,`) e identificadores (*ident*):

$$\begin{array}{l} \mathit{Decl} \rightarrow \mathit{Type}\ \mathit{Varlist} \\ \mathit{Type} \rightarrow \texttt{int} \mid \texttt{float} \\ \mathit{Varlist} \rightarrow \mathit{Varlist}\texttt{,}\mathit{ident} \mid \mathit{ident} \end{array}$$

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

Exercício 4

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

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

(a) 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 `*` em relação a `+`)
`1+2/3`	deve calcular $1 + (2/3)$	(prioridade de `/` em relação a `+`)
`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 deve apenas construir um *árvore sintática abstrata* (AST) durante a análise sintática e depois usar uma função de avaliação recursiva sobre a AST para efetuar os cálculos. O resultado deve ser calculado e apresentado em vírgula flutuante.

(b) Modifique a análise lexical, sintática e a função de avaliação para acrescentar algumas funções pré-definidas à calculadora:

$$\begin{array}{l} E \rightarrow \ \cdots \mid F\texttt{(}E\texttt{)} \\ F \rightarrow \texttt{sqrt} \mid \texttt{sin} \mid \texttt{cos} \mid \texttt{exp} \mid \texttt{log} \end{array}$$