Departamento de Informática - UFPR

Segunda prova (Soluções)

Construção de Compiladores - CI211 - 2018/1

Prof. André Luiz Pires Guedes

22 de junho de 2018

PROVA SEM CONSULTA

A prova tem duração de 1:30 horas.

A interpretação faz parte da prova. Pode fazer a lápis (contanto que seja possível ler). Pode ficar com a folha de questões.

Considere as gramáticas abaixo, onde os símbolos em minúsculas são terminais:

G_1			
1	P	$\rightarrow$	L
2	L	$\rightarrow$	$D\ L$
3		$\mid$	ϵ
4	D	$\rightarrow$	x
5		$\mid$	y

G_2			
1	S	$\rightarrow$	E
2	E	$\rightarrow$	$while\ E\ do\ E$
3		$\mid$	$id := E$
4		$\mid$	$E + E$
5		$\mid$	id

(20 pts) **1.** Dada a gramática G_2 acima, que ações (usando tradução dirigida por sintaxe) devem ser associadas a cada regra da da gramática para podermos construir a árvore de derivação?

Resposta: A função *newnode* cria um nó novo. O primeiro parâmetro é o nome do nó. Os parâmetros seguintes são colocados como filhos do nó recém criado. Quando a regra tem mais de uma ocorrência de um símbolo, estes são numerados da esquerda para a direita.

1 $S.tree \leftarrow newnode(\text{"}S\text{"}, E.tree)$

2 $E_1.tree \leftarrow newnode(\text{"}E\text{"}, newnode(\text{"}while\text{"}), E_2.tree, newnode(\text{"}do\text{"}), E_3.tree)$

3 $E_1.tree \leftarrow newnode(\text{"}E\text{"}, newnode(\text{"}id\text{"}), newnode(\text{" := "}), E_2.tree)$

4 $E_1.tree \leftarrow newnode(\text{"}E\text{"}, E_2.tree, newnode(\text{" + "}), E_3.tree)$

5 $E.tree \leftarrow newnode(\text{"}E\text{"}, newnode(\text{"}id\text{"}))$

(20 pts) **2.** Apresente uma gramática qualquer (simples) que não seja SLR(1) e seja LALR(1).

Resposta: Seja a gramática:

G			
1	S	$\rightarrow$	$L = R$
2		$\mid$	R
3	L	$\rightarrow$	$* R$
4		$\mid$	id
5	R	$\rightarrow$	L

Note que $FOLLOW(R) = \{\$, =\}$ e que teremos um estado SLR(1) com os ítens $[S \rightarrow L \cdot\ =\ R]$ e $[R \rightarrow L \cdot]$. Neste estado tem um shift com = e uma redução pela regra 5 com \$ e =. Portanto temos um conflito e G não é SLR(1).

Neste mesmo estado, considerado como LALR(1), o *lookahead* do ítem responsável pela redução é somente {\$}. E o conflito não existe mais.

(20 pts) **3.** Para a gramática G_1 acima, elabore as ações semânticas (usando tradução dirigida por sintaxe) para que seja verificado que o número de ocorrências de "x" e de "y" sejam iguais.

Resposta: Vamos usar um atributo numérico para manter a diferença entre o número de ocorrências de x e de y. Este atributo tem o nome "dif"e é usado nos símbolos L e D. O símbolo inicial P tem um atributo "resultado"de tipo boleano, sendo verdadeiro quando o número de ocorrências de x e de y são iguais. Quando a regra tem mais de uma ocorrência de um símbolo, estes são numerados da esquerda para a direita.

1 $P.resultado \leftarrow (L.dif = 0)$

2 $L_1.diff \leftarrow D.dif + L_2.dif$

3 $L.dif \leftarrow 0$

4 $D.dif \leftarrow 1$

5 $D.dif \leftarrow -1$

(20 pts) **4.** Para a gramática G_2 acima, apresente o primeiro conjunto de ítens LALR(1), I_0, construa o estado $I_1 = GOTO(I_0, while)$. Em seguida construa os estados derivados do I_1.

Resposta:

I_0:

$[S \to \cdot\ E, \{\$\}]$
$[E \to \cdot\ while\ E\ do\ E, \{\$, +\}]$
$[E \to \cdot\ id\ :=\ E, \{\$, +\}]$
$[E \to \cdot\ E\ +\ E, \{\$, +\}]$
$[E \to \cdot\ id, \{\$, +\}]$

$I_1 = GOTO(I_0, while)$:

$[E \to while\ \cdot\ E\ do\ E, \{\$, +\}]$
$[E \to \cdot\ while\ E\ do\ E, \{do, +\}]$
$[E \to \cdot\ id\ :=\ E, \{do, +\}]$
$[E \to \cdot\ E\ +\ E, \{do, +\}]$
$[E \to \cdot\ id, \{do, +\}]$

$I_2 = GOTO(I_1, E)$:

$[E \to while\ E\ \cdot\ do\ E, \{\$, +\}]$
$[E \to E\ \cdot\ +\ E, \{do, +\}]$

$I_3 = GOTO(I_1, while)$:

$[E \to while\ \cdot\ E\ do\ E, \{do, +\}]$
$[E \to \cdot\ while\ E\ do\ E, \{do, +\}]$
$[E \to \cdot\ id\ :=\ E, \{do, +\}]$
$[E \to \cdot\ E\ +\ E, \{do, +\}]$
$[E \to \cdot\ id, \{do, +\}]$

$I_4 = GOTO(I_1, id)$:

$[E \to id\ \cdot\ :=\ E, \{do, +\}]$
$[E \to id\ \cdot, \{do, +\}]$

Os estados I_1 e I_3 são unificados no estado I'_1 abaixo: $I'_1 = I_1 \cup I_3$:

$[E \to while\ \cdot\ E\ do\ E, \{\$, do, +\}]$
$[E \to \cdot\ while\ E\ do\ E, \{do, +\}]$
$[E \to \cdot\ id\ :=\ E, \{do, +\}]$
$[E \to \cdot\ E\ +\ E, \{do, +\}]$
$[E \to \cdot\ id, \{do, +\}]$

(20 pts) **5.** O que é otimização de código? Por que é necessária?

Resposta: A otimização de código é um conjunto de ações para melhorar o desempenho do código gerado pelo compilador, removendo ou alterando instruções redundantes ou desnecessárias. É necessária pois os compiladores geram código em blocos associados aos comandos de mais alto nível das linguagens fonte. Estes blocos, ao serem colocados em sequência, nem sempre se encaixam da melhor forma.