Compiladores

Prof. Marc Antonio Vieira de Queiroz

Ciência da Computação - UNIFIL
LAB 7
marc.queiroz@unifil.br

31/07/2013

Roteiro I

Análise Sintática Descendente I

Analisadores sintáticos preditivos podem ser construídos para uma classe de gramáticas chamada LL(1).

A sigla LL, refere-se L (left-to-right), onde na qual faz-se a varredura da palavra, da esquerda para a direita. O segundo L significa derivação mais a esquerda. A análise do número **(**1) a frente significa a quantidade de símbolos necessários para tomada de decisão.

A classe LL(1) é rica o suficiente para reconhecer a maioria das construções presentes nas linguagens de programação. Porém não é tão simples escrever uma gramática deste tipo, pois não pode conter recursão à esquerda, nem ser ambígua.

Uma gramática G é LL(1) se se somente se, sempre que $A \rightarrow \alpha|\beta$ forem duas produções distintas de G, as seguintes condições forem verdadeiras:

Análise Sintática Descendente II

1. Para um terminal **a**, tanto α quanto β não derivam cadeias começando com a.
2. No máximo um dos dois, α ou β, pode derivar a cadeia vazia.
3. Se $\beta \overset{\Rightarrow}{*} \epsilon$, então α não deriva nenhuma cadeia começando com um terminal em FOLLOW(A)

1 e 2 são condições equivalente a dizer que FIRST(α) e FIRST(β) são conjuntos disjuntos. A interseção destes conjuntos é vazia.

Construções de fluxo de controle, com suas distintas palavras-chave, geralmente satisfazem as restrições das gramáticas LL(1). Exemplo:

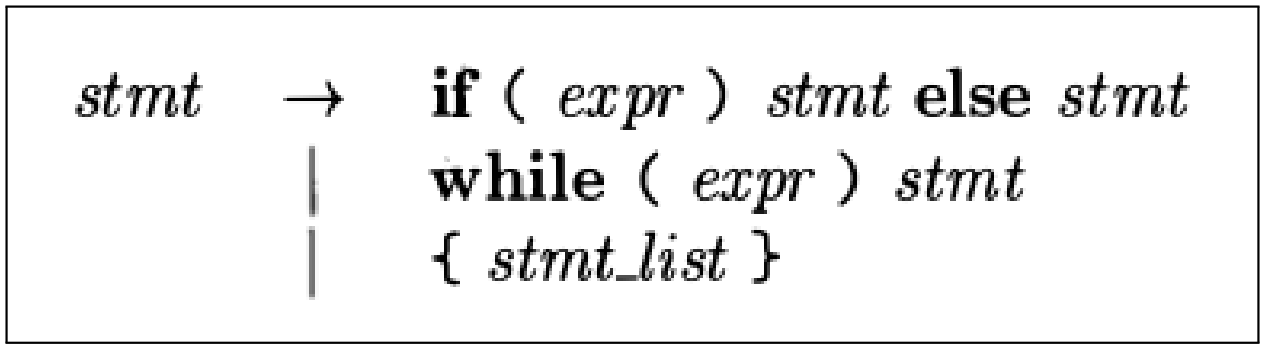

Figura: 1

Então as palavras-chave if, while e o símbolo $\{$ nos dizem quais as alternativas bem-sucedidas se estivermos procurando por um stmt.

Tabela para reconhecer preditivo I

Utiliza os conjuntos FIRST e FOLLOW em uma tabela M[A,a], onde:

- A é um não-terminal
- a é um terminal ou $

Para montar a tabela a produção $A \rightarrow \alpha$ é escolhida se o próximo símbolo de entrada **(**a) estiver em FIRST(α). A única complicação ocorre quando $\alpha = \epsilon$ ou ($\alpha \overset{*}{\underset{Rightarrow}{}} \epsilon$). Nesse caso, deve-se escolher $A \rightarrow \alpha$, se o símbolo corrente da entrada estiver em FOLLOW(A), ou seja $ foi alcançado e $ está em FOLLOW(A).

Algoritmo: Construção da tabela preditiva
entrada: Gramática
saída: tabela M
Método: para cada produção $A \rightarrow \alpha$, faça:

1. Para cada a em FIRST(α) inclua $A \rightarrow \alpha$ em M[A,a]
2. Se ϵ pertence a FIRST(α), inclua $A \rightarrow \alpha$ em M[A,b] para cada terminal b em FOLLOW(A).
3. Se ϵ pertence a FIRST(α) e \$ pertence a FOLLOW(A) acrescente $A \rightarrow \alpha$ em M[A,\$].

Se depois de realizar estes passos, não houver produções em M[A,a] então M[A,a] é definida como error (entrada em branco).

Exemplo

Para a expressão da gramática G (figura abaixo), utilize o algoritmo 4.31 para produzir uma tabela de parsing. Onde entradas em branco representam erros, entradas não nulas indicam uma produção na qual deve expandir o não terminal.

$$
\begin{array}{lcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathbf{id}
\end{array}
$$

Tabela para reconhecedor preditivo:

NON-TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to \mathbf{id}$			$F \to (E)$		