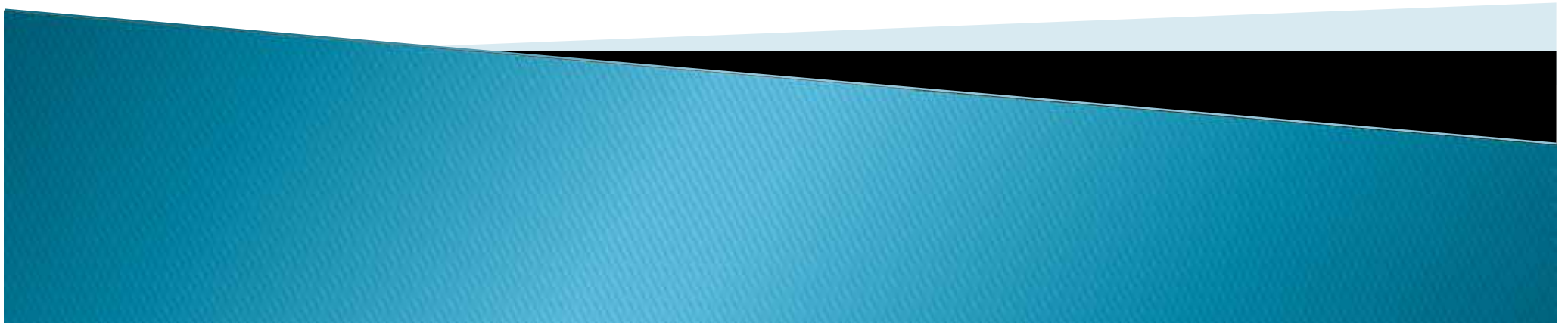


Compiladores

Análise Sintática – Parte 2

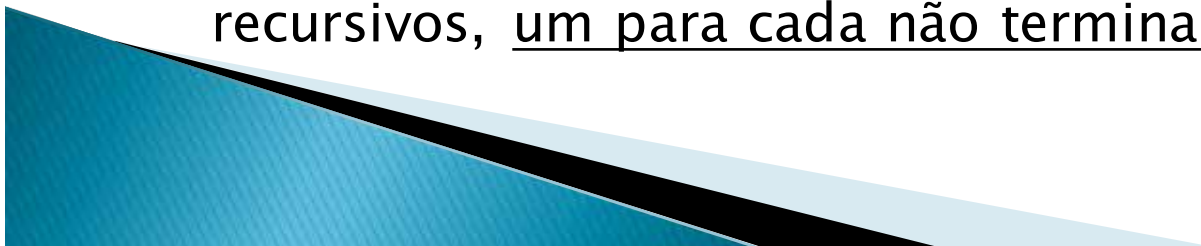
Prof. Tales Viegas

<https://fb.com/ProfessorTalesViegas>



Análise Descendente Recursiva

- ▶ Precisamos saber, dado o símbolo de entrada α e o não terminal A a ser expandido, qual das alternativas da produção
 - $A \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$é a única alternativa que deriva uma cadeia começando por α
- ▶ Se uma alternativa de A é ϵ , e nenhuma das outras alternativas deriva da cadeia começando com α , então podemos expandir
 - $A \rightarrow \epsilon$aceitando a entrada α .
- ▶ O parser recursivo descendente é um conjunto de procedimentos recursivos, um para cada não terminal a ser derivado.



Análise Descendente Recursiva – Restrições

- ▶ Gramáticas não-recursivas à esquerda, com produções do tipo $A \rightarrow A\alpha$
- ▶ Não possuir mais que um lado direito de um não terminal começando por um mesmo terminal;



Análise Descendente Recursiva

Implementação

- ▶ **Método:** Diagrama sintático → Procedimento
- ▶ **Consequência:** 1 Gramática → 1 Programa

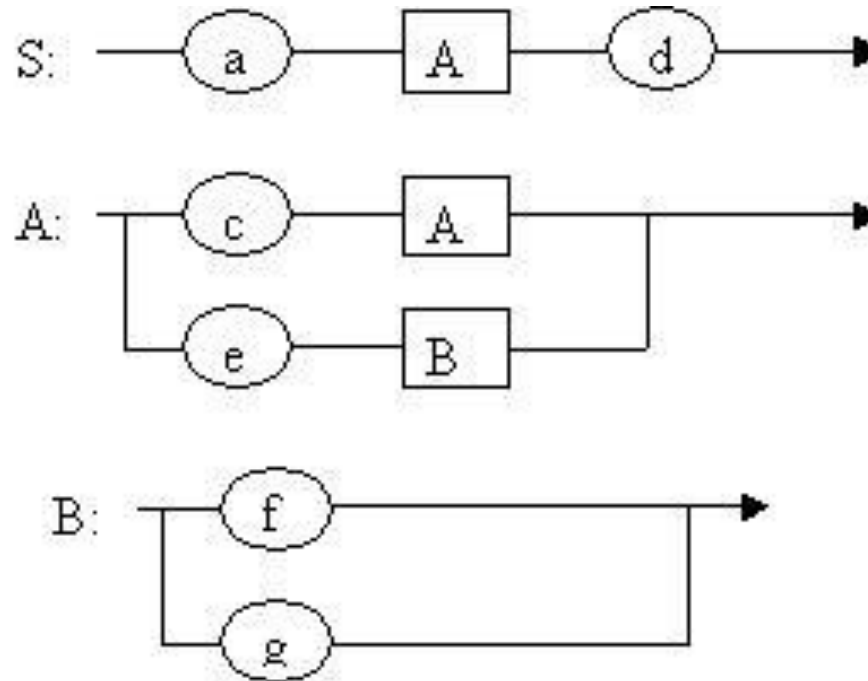


Análise Descendente Recursiva

Implementação – Exemplo

► Seja a gramática dada por:

- $S \rightarrow aAd$
- $A \rightarrow cA$
- $A \rightarrow eB$
- $B \rightarrow f$
- $B \rightarrow g$



Eliminação de Recursão à Esquerda

- ▶ Suponhamos o par de produções gramaticais, com recursão à esquerda:

$$A \rightarrow A\alpha \quad A \rightarrow \beta$$

- ▶ A recursão à esquerda pode ser eliminada por novas produções do tipo:

$$A \rightarrow \beta A' \quad A' \rightarrow \alpha A' \quad A' \rightarrow \epsilon$$



Exercício

- ▶ Elimine a recursão à esquerda para a seguinte gramática:

$$E \rightarrow T$$

$$T \rightarrow F$$

$$F \rightarrow a$$

$$E \rightarrow E+T$$

$$T \rightarrow T * F$$

$$F \rightarrow b$$

$$F \rightarrow (E)$$



Análise LL(k)

- ▶ LL(k) *Left to right, leftmost derivation with k lookahead symbols*
- ▶ Ideia básica: basta olharmos “no máximo k” símbolos da cadeia a frente do ponto que estamos para decidir sobre qual regra devemos aplicar.
- ▶ Ideal: apenas um símbolo ($k=1$) é necessário



Gramáticas LL(1)

- ▶ Condições necessárias
 - Sem ambiguidade
 - Sem recursão à esquerda



Gramáticas LL(1)

- ▶ Uma gramática G é LL(1) sempre que, sendo $A \rightarrow \alpha | \beta$ forem duas produções distintas de G , vigorarem as seguintes condições:
 - Não existem terminais comuns nos conjuntos $\text{FIRST}(\alpha)$ e $\text{FIRST}(\beta)$
 - No máximo um dos dois, α ou β derivam ϵ
 - Se $\alpha \rightarrow^* \epsilon$, então não devem existir terminais comuns nos conjuntos $\text{FIRST}(\beta)$ e $\text{FOLLOW}(A)$

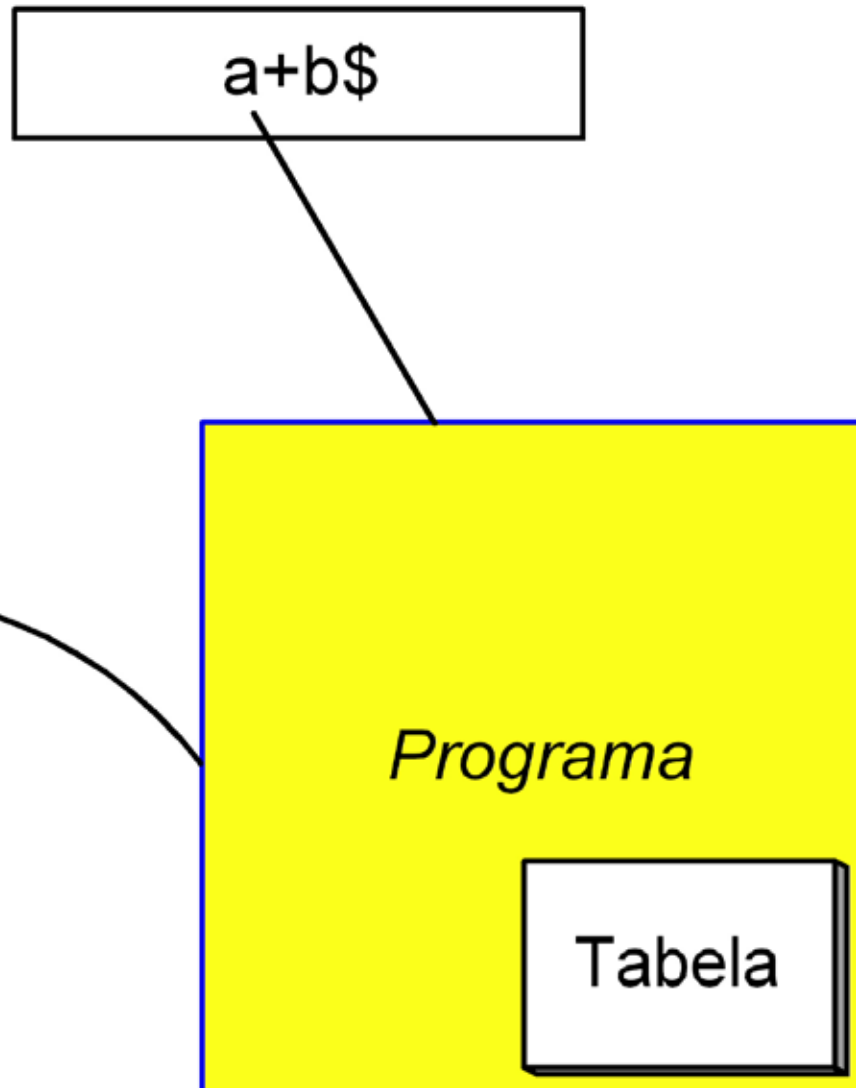


Implementação de um Parser

- ▶ A partir de uma gramática do tipo LL(1)
- ▶ A implementação contém uma pilha e a cadeia de entrada, consultando os conjuntos FIRST() para tomar a decisão do que empilhar/desempilhar
- ▶ Quando o topo da pilha contiver um não-terminal, ele será substituído na pilha pela cadeia de símbolos da opção escolhida, de acordo com os conjuntos FIRST()



Implementação de um Parser



Exemplo

- ▶ Considere a gramática dada pelas produções

$$\begin{array}{lll} E \rightarrow T & T \rightarrow F & F \rightarrow \text{id} \\ E \rightarrow E + T & T \rightarrow T * F & F \rightarrow (E) \end{array}$$

- a) Elimine a recursão à esquerda, obtendo novas regras gramaticais
- b) Obtenha os conjuntos FIRST() e FOLLOW()
- c) Como poderia, de acordo com as especificações anteriores, funcionar um parser no processo de reconhecimento da cadeia (id+id) \$



Implementação de um Parser

- ▶ Item a

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \epsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \epsilon$

$F \rightarrow (E)$

$F \rightarrow id$



Implementação de um Parser

► Item b

$\text{FIRST}(E) = \{ (, \text{id} \}$

$\text{FIRST}(E') = \{ +, \epsilon \}$

$\text{FIRST}(T) = \{ (, \text{id} \}$

$\text{FIRST}(T') = \{ *, \epsilon \}$

$\text{FIRST}(F) = \{ (, \text{id} \}$

$\text{FOLLOW}(E) = \{ \$,) \}$

$\text{FOLLOW}(E') = \{ +, \$,) \}$

$\text{FOLLOW}(T) = \{ +, \$,) \}$

$\text{FOLLOW}(T') = \{ +, \$,) \}$

$\text{FOLLOW}(F) = \{ +, \$,), * \}$



Implementação de um Parser

- ▶ O parser é implementado por um autômato de pilha, controlado por uma tabela de análise
- ▶ O analisador busca a produção a ser aplicada na tabela, levando em conta o não-terminal no topo da pilha e o token que está sendo analisado




Implementação de um Parser

- ▶ Considerando X como o símbolo no topo da pilha e α como o token que está sendo analisado, o analisador executa uma das 3 ações:
- ▶ 1 – Se $X = \alpha = \$$, o analisador pára, aceitando a sentença
- ▶ 2 – Se $X \neq \alpha \neq \$$, o analisador desempilha α e avança a análise para o próximo token



Implementação de um Parser

- ▶ Se X é um símbolo não-terminal, o analisador consulta a entrada $M[X, \alpha]$ da tabela de análise.
 - ▶ Esta entrada poderá conter uma produção da gramática ou ser vazia
 - ▶ Supondo $M[X, \alpha] = \{X \rightarrow UVW\}$, o analisador substitui X (que está no topo da pilha) por WVU e retorna a produção aplicada.
 - ▶ Se $M[X, \alpha]$ é vazio, isto corresponde a uma situação de erro. Neste caso, o analisador deve chamar a rotina de tratamento de erros.
- 

Implementação de um Parser

	+	*	()	Id	\$
E			$E \rightarrow TE'$		$E \rightarrow TE'$	
E'	$E' \rightarrow +TE'$			$E' \rightarrow \epsilon$		$E' \rightarrow \epsilon$
T			$T \rightarrow FT'$		$T \rightarrow FT'$	
T'	$T' \rightarrow \epsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \epsilon$		$T' \rightarrow \epsilon$
F			$F \rightarrow (E)$		$F \rightarrow id$	



Implementação de um Parser

Pilha	Leitura	Ação
\$E	(id+id)\$	$E \rightarrow TE'$
\$E'T	(id+id)\$	$T \rightarrow FT'$
\$E'T'F	(id+id)\$	$F \rightarrow (E)$
\$E'T')E((id+id)\$	Desempilha e avança
\$E'T')E	id+id)\$	$E \rightarrow TE'$
\$E'T')E'T	id+id)\$	$T \rightarrow FT'$
\$E'T')E'T'F	id+id)\$	$F \rightarrow id$
\$E'T')E'T'id	id+id)\$	Desempilha e avança
\$E'T')E'T'	+id)\$	$T' \rightarrow \epsilon$
\$E'T')E'	+id)\$	$E' \rightarrow +TE'$
\$E'T')E'T+	+id)\$	Desempilha e avança
\$E'T')E'T	id)\$	$T \rightarrow FT'$

Implementação de um Parser

Pilha	Leitura	Ação
\$E'T')E'T'F	id)\$	$F \rightarrow id$
\$E'T')E'T'id	id)\$	Desempilha e avança
\$E'T')E'T')\$	$T' \rightarrow \epsilon$
\$E'T')E')\$	$E' \rightarrow \epsilon$
\$E'T'))\$	Desempilha e avança
\$E'T'	\$	$T' \rightarrow \epsilon$
\$E'	\$	$E' \rightarrow \epsilon$
\$	\$	Sentença aceita



Exercício

- ▶ Comprove se as sentenças abaixo fazem parte da gramática do exemplo, realizando a análise LL
- ▶ $(id + id) * id$
- ▶ id
- ▶ (id)
- ▶ $id + id * id$



Análise LL(k)

- ▶ Supondo a gramática dada por:

$S \rightarrow AS \mid BA$

$A \rightarrow aB \mid C$

$B \rightarrow bA \mid d$

$C \rightarrow c$

A gramática acima é do tipo LL(1)?

