

– INF01147 –
Compiladores

Árvore Sintática Abstrata
Introdução à Análise Sintática Ascendente

Prof. Lucas M. Schnorr
– Universidade Federal do Rio Grande do Sul –



Gramáticas LL(1)

Análise Descendente Preditiva Tabular
(revisão da aula anterior)

Tabela Preditiva – Algoritmo de Construção

- ▶ Para cada produção $A \rightarrow \alpha$ da gramática
- ▶ Sendo M a tabela de não-terminais *versus* terminais
 - ▶ Para cada terminal t em $\text{First}(\alpha)$, inclua $(A \rightarrow \alpha)$ em $M[A, t]$
 - ▶ Se $(A \rightarrow \epsilon)$ inclua-a esta regra em todos os $M[A, b]$
 - ▶ b faz parte do $\text{Follow}(A)$
 - ▶ Se $(A \rightarrow \epsilon)$ e $\text{Follow}(A)$ contém $\$,$ inclua-a em $M[A, \$]$
- ▶ Ao fim do algoritmo
 - ▶ Células vazias na tabela são consideradas erros

Análise Preditiva Tabular – Funcionamento

- ▶ Reconhecer $(id+id)*id+id$
- ▶ Considerando a tabela preditiva
- ▶ A pilha começa com **E**

	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	$F \rightarrow id$			$F \rightarrow (E)$		

Plano da Aula de Hoje

- ▶ Árvore Sintática Abstrata
- ▶ Lançamento da Etapa 3

- ▶ Introdução à Análise Sintática Ascendente
 - ▶ Princípios gerais
 - ▶ Vocabulário
 - ▶ Exemplos
- ▶ Analisador com pilha (Empilhar/Reduzir)
 - ▶ Tabelas LR

Árvores de Derivação (Relembrando)

- ▶ **Representação** do processo de derivação
 - Mostra a estrutura sintática do programa
- ▶ Estrutura hierárquica
 - ▶ Raiz é o símbolo inicial da gramática
 - ▶ Vértices intermediários são não-terminais
 - ▶ Folhas são os terminais e palavras vazias (ϵ)

Árvores de Derivação

- Supondo a gramática de expressões aritméticas

$\text{exp} \rightarrow \text{exp op exp}$

$\text{exp} \rightarrow (\text{exp})$

$\text{exp} \rightarrow \text{número}$

$\text{op} \rightarrow + \mid - \mid \times$

- Como fica a árvore de derivação para $(34 - 3) \times 42$
- **Pergunta:** como podemos simplificar a árvore de derivação?
 - Considerando a geração de código intermediário

Árvore Sintática Abstrata (AST)

- ▶ Análise sintática é o centro do compilador
- ▶ Princípio da tradução direcionada por sintaxe
 - ▶ Significado da entrada deve ter relação direta com a sintaxe
- ▶ **Árvore Sintática Abstrata (AST)**
 - ▶ Simplificação da árvore de derivação
 - ▶ Semântica idêntica

AST – Exemplo 1

- Supondo a gramática de expressões aritméticas com atribuição

stmt \rightarrow ident = exp

ident \rightarrow **var**

exp \rightarrow exp op exp

exp \rightarrow (exp)

exp \rightarrow **número**

op \rightarrow + | - | ×

- Como fica a AST para **var** = (34 - 3) × 42 ?

AST – Exemplo 2

- Supondo a gramática para o comando **if**

stmt → if-stmt | **outra**

if-stmt → **if** (exp) stmt

if-stmt → **if** (exp) stmt **else** stmt

exp → **false** | **true**

- AST para **if (false) if (true) outra else outra ?**

AST – Exemplo 3

- Supondo a gramática para os comandos **do while** e **if**

stmt → while-stmt | if-stmt | **outra**

while-stmt → **do** stmt **while** (exp)

if-stmt → **if** (exp) stmt

if-stmt → **if** (exp) stmt **else** stmt

exp → **false** | **true**

- if (false) if (true) outra else do outra while (true) ?

AST – Exemplo 4

- Supondo a gramática de sequência de comandos separados por ;

seq-stmt \rightarrow stmt ; seq-stmt | stmt

stmt \rightarrow **comando**

- Qual a AST para **comando;comando;comando; ?**

AST – Exemplo 5

- Supondo a gramática

seq-stmt	→	stmt ; seq-stmt stmt
stmt	→	while-stmt if-stmt outra
while-stmt	→	do stmt while (exp)
if-stmt	→	if (exp) stmt
if-stmt	→	if (exp) stmt else stmt
exp	→	false true

- do if (true) outra while (false); if(false) outra ?

Projeto de Compilador

Lançamento da Etapa 3

Análise Sintática Ascendente

Descendente *versus* Ascendente

- Considerando a seguinte gramática e a entrada **ccbca**

$S \rightarrow AB$

$A \rightarrow c \mid \epsilon$

$B \rightarrow cbB \mid ca$

- Análise Descendente

$S \Rightarrow_{S \rightarrow AB} AB \Rightarrow_{A \rightarrow c} cB \Rightarrow_{B \rightarrow cbB} ccbB \Rightarrow_{B \rightarrow ca} \text{ccbca}$

- Análise Ascendente

$\text{ccbca} \Rightarrow_{A \rightarrow c} Acbca \Rightarrow_{B \rightarrow ca} AcbB \Rightarrow_{B \rightarrow cbB} AB \Rightarrow_{S \rightarrow AB} S$

- Mais sofisticada (aceita recursão à esquerda, por exemplo)
- Abrange um número maior de gramáticas

Ascendente – Processo de Redução

- ▶ **Redução** é a substituição do corpo pela cabeça (do lado direito pelo não-terminal correspondente)
 - ▶ Efetua-se várias reduções, até obter o símbolo inicial
- ▶ Principais decisões durante a análise ascendente
 - ▶ Quando reduzir
 - ▶ Qual produção utilizar na redução
- ▶ Considerando a seguinte gramática e a entrada **abbcde**

$$\begin{aligned} S &\rightarrow aABe \\ A &\rightarrow Abc \mid b \\ B &\rightarrow d \end{aligned}$$

- ▶ Processo de Redução

abbcde

a**b**bcde ($A \rightarrow b$)

a**Abc**de ($A \rightarrow Abc$)

aA**d**e ($B \rightarrow d$)

aABe ($S \rightarrow aABe$)

S

Ascendente – Processo de Redução

- Considerando a gramática aritmética e a entrada **id*id**

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid \text{id}$$

- Processo de Redução

$$id * id \quad F \rightarrow id$$

$$F * id \quad T \rightarrow F$$

$$T * id \quad F \rightarrow id$$

$$T * F \quad T \rightarrow T * F$$

$$T \quad E \rightarrow T$$

$$E$$

- Por definição
 - Uma redução é o inverso de um passo de derivação
 - Percebam a derivação **mais à direita** no exemplo acima (olhando debaixo para cima)

Ascendente – Poda do Handle

- ▶ Definição informal de um **Handle**
 - ▶ Subcadeia que casa com o corpo de uma produção
 - ▶ Sua redução **garante** a derivação **mais à direita**
- ▶ Definição formal (considerando derivações mais à direita)
$$S \Rightarrow^* \alpha A \omega \Rightarrow \alpha \beta \omega$$
- ▶ Então, β é um *handle* de $\alpha \beta \omega$
sendo que ω contém apenas símbolos terminais
- ▶ Se a gramática for ambígua \leadsto vários *handles* possíveis
(pois existem várias derivações mais à direita possíveis)
- ▶ **Poda do Handle**
 - ▶ Permite construir uma derivação mais à direita ao reverso
 - ▶ Considerando μ uma sentença da gramática
$$S = \gamma_0 \Rightarrow \gamma_1 \Rightarrow \dots \Rightarrow \gamma_{n-1} \Rightarrow \gamma_n = \mu$$
Localiza-se β_n em γ_n , escolhe-se uma $A_n \rightarrow \beta_n$ para obter γ_{n-1}

Ascendente – Poda do Handle (Exemplo)

- Considerando a gramática e a entrada $id * id$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid id$$

- A derivação mais à direita de $id * id$ é

$$E \Rightarrow T \Rightarrow T * F \Rightarrow T * id \Rightarrow F * id \Rightarrow id * id$$

- **Pergunta:** quais são os handles?
- Resposta

Forma Sentencial	Handle	Produção de Redução
$id_1 * id_2$	id_1	$F \rightarrow id$
$F * id$	F	$T \rightarrow F$
$T * id$	id	$F \rightarrow id$
$T * F$	$T * F$	$E \rightarrow T * F$

Ascendente – Empilha-Reduz

- ▶ Analisador Sintático **Empilha-Reduz** *shift-reduce*
 - ▶ Implementa uma forma de análise ascendente
 - ▶ Vários algoritmos o implementam
LR(0) – SLR(1) – LR(1) – LALR(1)
- ▶ Componentes: uma pilha e um buffer de entrada
- ▶ Operações
 - ▶ **Empilha** *shift*
Empilha um token da entrada
 - ▶ **Reduz** *reduce*
Realiza a poda do handle sempre no topo da pilha
 - ▶ **Aceita**
Reconhece a sentença de entrada
 - ▶ **Erro**
Ativa o tratamento de erros sintáticos

Ascendente – Exemplo

- Considerando a gramática aritmética

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid id$$

- Aplicar as ações de analisador Empilha-Reduz para **id*id**
- Solução

Pilha	Entrada	Ação
\$	$id_1 * id_2$	\$ empilha
\$ id_1	$*id_2$	\$ reduz $F \rightarrow id$
\$ F	$*id_2$	\$ reduz $T \rightarrow F$
\$ T	$*id_2$	\$ empilha
\$ $T*$	id_2	\$ empilha
\$ $T * id_2$		\$ reduz $F \rightarrow id$
\$ $T * F$		\$ reduz $T \rightarrow T * F$
\$ T		\$ reduz $E \rightarrow T$
\$ E		\$ aceita

Ascendente – Conflitos Empilha-Reduz

- ▶ Duas situações onde não funciona
 - ▶ Conflito **Reduz-Reduz**
→ mais de uma redução possível
 - ▶ Conflito **Empilha-Reduz**
→ gramática ambígua
- ▶ Exemplo

stmt → if expr then stmt
if expr then stmt else stmt
other

- ▶ O que fazer nesta situação?

Pilha	Entrada
\$... if expr then stmt	else ... \$

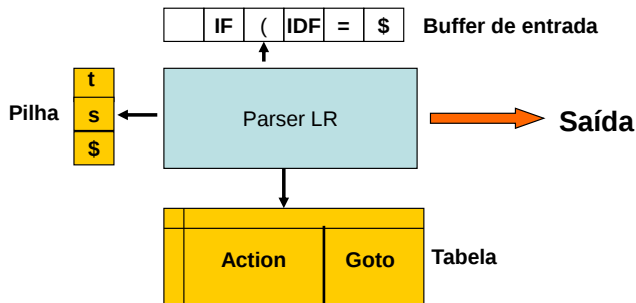
Análise Ascendente (Relembrando)

- ▶ Mais sofisticado que a abordagem descendente
 - ▶ Regras são aplicadas em reverso
 - ▶ Pode “adiar decisões” de redução
- ▶ Existem vários algoritmos para análise Empilha-Reduz
 - ▶ LR(0)
 - ▶ SLR(1)
 - ▶ LR(1) ou LR
 - ▶ LALR(1)

Gramáticas e Análise LR

Gramáticas e Análise LR

- ▶ LR, uma classe de gramáticas onde
 - ▶ Entrada lida da esquerda para a direita L de *left-to-right*
 - ▶ Aplica-se uma derivação mais à direita R de *rightmost*
- ▶ Análise LR: um **autômato de estados finitos** com pilha
- ▶ Componentes
 - ▶ Pilha contém **estados** (ao invés de símbolos)
 - ▶ Tabela de ações/transições (com terminais e não-terminais)



Análise LR – Tabela de Ação/Transição

- ▶ Tabela de ações/transições, a partir de um estado s
 - ▶ Ação $[s, t]$ sendo t um terminal
 - ▶ Transição $[s, X]$ sendo X um não-terminal
- ▶ Ação $[s, t]$ pode indicar
 - ▶ (Empilha e), onde e é um estado para empilhar
 - ▶ (Reduz p), onde p é a regra de produção para a redução
 - ▶ (Aceita)
- ▶ Transição $[s, X]$ pode indicar
 - ▶ (Move para e), onde e é um estado

Análise LR – Exemplo com entrada **id*id+id**

(1) $E \rightarrow E + T$	(3) $T \rightarrow T * F$	(5) $F \rightarrow (E)$
(2) $E \rightarrow T$	(4) $T \rightarrow F$	(6) $F \rightarrow id$

Estado	id	+	*	()	\$	E	T	F
0	e5			e4			1	2	3
1		e6				a			
2		r2	e7		r2	r2			
3		r4	r4		r4	r4			
4	e5			e4			8	2	3
5		r6	r6		r6	r6			
6	e5			e4				9	3
7	e5			e4					10
8		e6			e11				
9		r1	e7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Sumário da Aula de Hoje

- ▶ Árvore Sintática Abstrata
- ▶ Análise Ascendente
 - ▶ *Handles*, para determinar qual cadeia reduzir
 - ▶ Tabelas LR (com *goto* e transições) para determinais quais reduções e *shifts* realizar
- ▶ Autômato de Estados Finitos com pilha, para reconhecer
 - ▶ Construído a partir dos itens canônicos
 - . Fechamento
 - + Propagação

Conclusão

- ▶ Leituras Recomendadas
 - ▶ Livro do Dragão
 - ▶ Seções 4.5
- ▶ Próxima Aula
Análise Sintática Ascendente