

Analísadores Ascendentes ou Empilha-Reduz

Marcelo Johann

Conteúdo da aula

Analísadores Ascendentes

- Funcionamento
- Analísadores de Precedência de Operadores
- Analísadores LR(k)
- SLR - Simple LR (estudaremos SLR(1))
- LR Canônicos
- LALR - Look Ahead LR (yacc)

Trabalho

Atributos no YACC

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 2

Lembrando: tabelas LL(1)

- Como fazer ?
 - Re-escrever gramática para satisfazer condições de LL(1)
 - Calcular conjuntos First e Follow
 - Para cada produção $A \rightarrow \alpha$
 1. Para cada $a \in \text{First}(\alpha)$
 - incluir $A \rightarrow \alpha$ em $M[A,a]$
 2. Se $\epsilon \in \text{First}(\alpha)$
 - incluir $A \rightarrow \alpha$ em $M[A,b]$ para cada b em $\text{Follow}(A)$
 3. Se $\epsilon \in \text{First}(\alpha)$ e $\$ \in \text{Follow}(A)$
 - incluir $A \rightarrow \alpha$ to $M[A,\$]$
 - Todas entradas não definidas são erros

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 3

Mais um exemplo...

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \epsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \epsilon$
$F \rightarrow (E) \mid \text{Id}$

Símbolo	First	Follow
E	{(, id}	{\$, }
E'	{+, ϵ }	{\$, }
T	{(, id}	{+, \$,)}
T'	{*, ϵ }	{+, \$,)}
F	{(, id}	{*, +, \$,)}

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 4

Mais um exemplo...

	Símbolo	First	Follow
$E \rightarrow TE'$	E	{(, id}	{\$, }
$E' \rightarrow +TE' \mid \epsilon$	E'	{+, ϵ }	{\$, }
$T \rightarrow FT'$	T	{(, id}	{+, \$,)}
$T' \rightarrow *FT' \mid \epsilon$	T'	{*, ϵ }	{+, \$,)}
$F \rightarrow (E) \mid \text{Id}$	F	{(, id}	{*, +, \$,)}

	*	id	(+)	\$
F		$F \rightarrow \text{id}$	$F \rightarrow (E)$			
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'	$T' \rightarrow *FT'$			$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$	$T' \rightarrow \epsilon$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 5

Top-Down x Bottom Up

Gramática: $S \rightarrow AB$ Entrada: $cbca$
 $A \rightarrow c \mid \epsilon$
 $B \rightarrow cbB \mid ca$

Top-Down/Esquerda	Bottom-Up/Direita
$S \Rightarrow AB$	$cbca \leftarrow Acbca$
$\Rightarrow cB$	$\leftarrow AcbB$
$\Rightarrow ccbB$	$\leftarrow AB$
$\Rightarrow ccbca$	$\leftarrow S$
$S \rightarrow AB$	$A \rightarrow c$
$A \rightarrow c$	$B \rightarrow ca$
$B \rightarrow cbB$	$B \rightarrow cbB$
$B \rightarrow ca$	$S \rightarrow AB$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 6

Redução – exemplo

$S \rightarrow aABe$ $abcde$
 $A \rightarrow Abc \mid b$
 $B \rightarrow d$

Redução = substituição do lado direito de uma produção pelo não terminal correspondente (lado esquerdo)

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 7

Redução – exemplo

$S \rightarrow aABe$ $abcde$
 $A \rightarrow Abc \mid b$ $aAbcde$
 $B \rightarrow d$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 8

Redução – exemplo

$S \rightarrow aABe$ $abcde$
 $A \rightarrow Abc \mid b$ $aAbcde$
 $B \rightarrow d$

handle = seqüência de símbolos do lado direito da produção, tais que suas reduções levam, no final, ao símbolo inicial da gramática

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 9

Redução – exemplo

$S \rightarrow aABe$ $abcde$
 $A \rightarrow Abc \mid b$ $aAbcde$
 $B \rightarrow d$ $aAde$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 10

Redução – exemplo

$S \rightarrow aABe$ $abcde$
 $A \rightarrow Abc \mid b$ $aAbcde$
 $B \rightarrow d$ $aAde$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 11

Redução – exemplo

$S \rightarrow aABe$ $abcde$
 $A \rightarrow Abc \mid b$ $aAbcde$
 $B \rightarrow d$ $aAde$
 $aABe$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 12

Redução – exemplo

$S \rightarrow aABe$ **abcde**
 $A \rightarrow Abc \mid b$ **aAbcde**
 $B \rightarrow d$ **aAde**
 aABe
 S

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 13

Análise bottom-up: o que fazer?

- Como decidir qual lado direito de produção trocar pelo lado esquerdo (redução) ?
 - Lê-se a entrada da esquerda para direita
 - Em algumas situações uma escolha aparece:
 1. ler mais um caractere da entrada (shift) ou
 2. aplicar redução (reduce)

Análise	Entrada	Ação
\$	abc\$	Ler
a\$	bc\$	Ler
ab\$	c\$	Ler
abc\$		Redução
S\$	\$	Aceitar

$S \rightarrow abc \mid a$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 14

Ações bottom-up (empilha-reduz)

- O *parser* Bottom-Up vai necessitar:
 - De uma pilha para guardar os símbolos
 - De um buffer de entrada para a sentença (seqüência de símbolos) *w* a ser reconhecida.
- Operações do *parser*:
 - **empilha (shift)**:
 - coloca no topo da pilha o símbolo que está sendo lido e lê o próximo token da sentença *w*.
 - **reduz (reduce)**:
 - substitui o handle no topo da pilha pelo não terminal correspondente
 - **aceita**:
 - reconhece que a sentença foi gerada pela gramática
 - **erro**:
 - ocorrendo erro de sintaxe, chama uma subrotina de atendimento a erros

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 15

Vários *parsers* Bottom-Up

- O parsing bottom-up é mais poderoso do que o parsing Top-Down
 - Regras aplicadas em reverso
 - Pode "adiar decisões" de reduções
 - Pode usar mais de um símbolo na entrada para tomar a decisão.
- Existe vários algoritmos para o parsing **Shift-Reduce** (empilha-reduz):
 - LR(0)
 - SLR(1)
 - LR(1)
 - LALR(1)

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 16

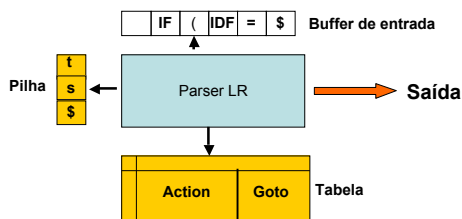
Parsing LR

Parser baseado em tabelas

- Lê a entrada de esquerda para direita (L)
- Cria derivações mais a direita (R)

Usa um autômato de estados finitos com pilha

- Cada estado representa handlers
- Transições são feitas a cada terminal/não terminal



INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 17

$E \rightarrow E + E \mid E * E \mid (E) \mid id$

Pilha	Entrada	Ação
\$	id + id * id	shift
\$id	+ id * id	Reduce $E \rightarrow id$
\$E	+ id * id	shift
\$E+	id * id	shift
\$E+id	* id	Reduce $E \rightarrow id$
\$E+E	* id	Reduce $E \rightarrow E + E$
\$E	* id	shift
\$E*	id	shift
\$E*id	\$	Reduce $E \rightarrow id$
\$E*E	\$	Reduce $E \rightarrow E * E$
\$E	\$	aceita!

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 18

Estruturas de dados

- Pilha de estados:
 - Cada estado representa duas informações:
 - O símbolo resultando da redução (ex.: “S” – start),
 - O *handle* a ser reduzido (“lado direito”).
- Tabela de ações, a partir de um estado s
 - Action[s,t]; $t \in T$
 - Transições: Goto[s,X]; $X \in N$

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 19

Tabela Ação/Transição (LR)

- **Ação (action)**: A partir de um estado s e de um terminal t , Ação[s, t] indica:
 - Se se faz um **shift** S (empilha) ou um **Reduce** R (reduz) ao ler o símbolo ‘a’ na entrada.
 - Caso Shift: indica também o estado a empilhar;
 - Fazer:
 - empilhar o estado e avançar na leitura.
 - Caso Reduce: indica também a regra a reduzir e, **a seguir, aplica uma Transição (goto)**.
 - Fazer:
 - depilhar tantos estados como símboloS reduzidoS;
 - empilhar o estado alvo do Goto(topo Pilha, Símbolo reduzido).
- **Transição (goto)**: a partir de um estado s e de um não-terminal X , Goto[s, X] indica o próximo estado a empilhar.

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 20

Exemplo de uso de tabela Ação/Transição

- Gramática usada:

$$S \rightarrow T$$

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

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

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 21

Tabela Ações/Transições

Ações Goto

	*	()	id	\$	E	T	F
0		S5		S8			2	1
1	R1	R1	R1	R1	R1			
2	S3				Ok!			
3		S5		S8				4
4	R2	R2	R2	R2	R2			
5		S5		S8			6	1
6	S3		S7					
7	R4	R4	R4	R4	R4			
8	R3	R3	R3	R3	R3			

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 22

Tabela Ações/Transições

Ações Terminals Goto Não-Terminals

	*	()	id	\$	E	T	F
0		S5		S8			2	1
1	R1	R1	R1	R1	R1			
2	S3				Ok!			
3		S5		S8				4
4	R2	R2	R2	R2	R2			
5		S5		S8			6	1
6	S3		S7					
7	R4	R4	R4	R4	R4			
8	R3	R3	R3	R3	R3			

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 23

Tabela Ações/Transições

Terminals Não-Terminals

	*	()	id	\$	E	T	F
0		S5		S8			2	1
1	R1	R1	R1	R1	R1			
2	S3				Ok!			
3		S5		S8				4
4	R2	R2	R2	R2	R2			
5		S5		S8			6	1
6	S3		S7					
7	R4	R4	R4	R4	R4			
8	R3	R3	R3	R3	R3			

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 24

Tabela Ações/Transições

	*	()	id	\$	E	T	F
0		S5		S8			2	1
1	R1	R1	R1	R1	R1			
2	S3				Ok!			
3		S5		S8				4
4	R2	R2	R2	R2	R2			
5		S5		S8			6	1
6	S3		S7					
7	R4	R4	R4	R4	R4			
8	R3	R3	R3	R3	R3			

Terminals (id, \$, E, T, F)
Não-Terminals (*, (,)
Estados (0-8)
Shift e empilha 8 (row 5, col 5)
Reduce a 4a regra (row 7, col 4)

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 25

Análise de "(id)*id" (1/2)

Stack	Input	Action
0	(id) * id \$	Shift S5
0 5	id) * id \$	Shift S8
0 5 8) * id \$	Reduz 3 F→id, pop 8, goto [5,F]=1
0 5 1) * id \$	Reduz 1 T→F, pop 1, goto [5,T]=6
0 5 6) * id \$	Shift S7
0 5 6 7	* id \$	Reduz 4 F→(T), pop 7 6 5, goto [0,F]=1
0 1	* id \$	Reduz 1 T→F pop 1, goto [0,T]=2

Productions	
1	T → F
2	T → T * F
3	F → id
4	F → (T)

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 26

Análise de "(id)*id" (2/2)

Stack	Input	Action
0 1	* id \$	Reduz 1 T→F, pop 1, goto [0,T]=2
0 2	* id \$	Shift S3
0 2 3	id \$	Shift S8
0 2 3 8	\$	Reduz 3 F→id, pop 8, goto [3,F]=4
0 2 3 4	\$	Reduz 2 T→T * F pop 4 3 2, goto [0,T]=2
0 2	\$	Aceitar

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 27

Leituras e Tarefas sugeridas

Ler capítulo 3 do livre da série didática até o início da seção 3.3.3

Fazer pequenos exemplos yacc

Fazer trabalho

INF01033 - Compiladores B - Marcelo Johann - 2010/2

Aula 09 : Slide 28