

# Compiladores: Parsing ascendente

Christiano Braga

Out. 2020

# Parsing ascendente

- ▶ Das folhas até o topo: redução de uma string ao símbolo inicial da gramática.
- ▶ Gramática 4.1, livro-texto, pg. 244

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

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

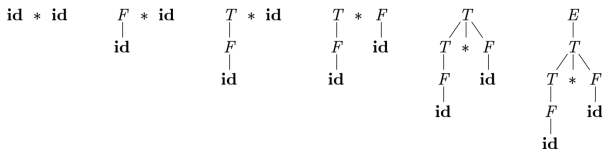


Figure 1: Parsing ascendente para id \* id

## Shift-reduce

- *Shift-reduce parsing* (empilha-reduz) é uma forma de parsing ascendente. Uma pilha guarda os símbolos gramaticais e um *buffer* guarda (o resto da) string sendo analisada.

STACK	INPUT	ACTION
\$	<b>id<sub>1</sub></b> * <b>id<sub>2</sub></b> \$	shift
\$ <b>id<sub>1</sub></b>	* <b>id<sub>2</sub></b> \$	reduce by $F \rightarrow \mathbf{id}$
\$ $F$	* <b>id<sub>2</sub></b> \$	reduce by $T \rightarrow F$
\$ $T$	* <b>id<sub>2</sub></b> \$	shift
\$ $T$ *	<b>id<sub>2</sub></b> \$	shift
\$ $T$ * <b>id<sub>2</sub></b>	\$	reduce by $F \rightarrow \mathbf{id}$
\$ $T$ * $F$	\$	reduce by $T \rightarrow T * F$
\$ $T$	\$	reduce by $E \rightarrow T$
\$ $E$	\$	accept

Figure 2: Configurações do parsing shift-reduce para  $\mathbf{id} * \mathbf{id}$

# Items

- Como decidir entre shift e reduce? Itens.

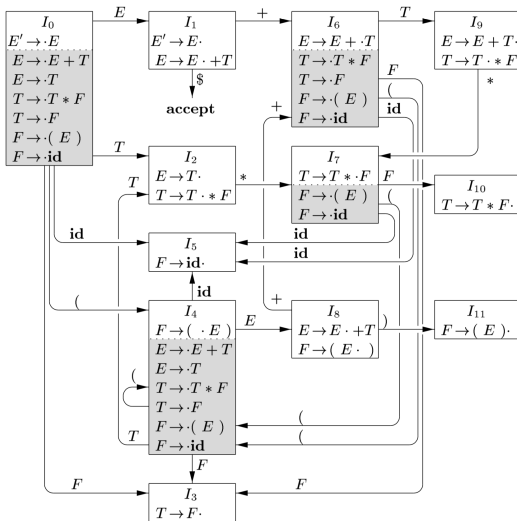


Figure 3: Autômato LR(0) para a gramática 4.1

# Closure de um conjunto de itens

- Closure (fecho) de um conjunto de itens representa (parte do) o estado do autômato LR(0).

```
SetOfItems CLOSURE( $I$ ) {  
     $J = I$ ;  
    repeat  
        for ( each item  $A \rightarrow \alpha \cdot B \beta$  in  $J$  )  
            for ( each production  $B \rightarrow \gamma$  of  $G$  )  
                if (  $B \rightarrow \cdot \gamma$  is not in  $J$  )  
                    add  $B \rightarrow \cdot \gamma$  to  $J$ ;  
    until no more items are added to  $J$  on one round;  
    return  $J$ ;  
}
```

Figure 4: Algoritmo para o cálculo da closure de um conjunto de itens

# Goto

- ▶ A função goto define as transições do autômato LR(0).
- ▶ A função  $\text{GOTO}(I, X)$  é definida como a closure do conjunto de todos os itens  $A \rightarrow \alpha X \cdot \beta$  desde que  $A \rightarrow \alpha \cdot X \beta$  esteja em  $I$ .

# Canonical items

- Canonical items (itens canônicos) representam os estados do autômato LR(0).

```
void items( $G'$ ) {  
     $C = \{\text{CLOSURE}(\{[S' \rightarrow \cdot S]\})\};$   
    repeat  
        for ( each set of items  $I$  in  $C$  )  
            for ( each grammar symbol  $X$  )  
                if (  $\text{GOTO}(I, X)$  is not empty and not in  $C$  )  
                    add  $\text{GOTO}(I, X)$  to  $C$ ;  
    until no new sets of items are added to  $C$  on a round;  
}
```

Figure 5: Algoritmo para o cálculo dos itens canônicos

# Parsing LR

- ▶ As tabelas ACTION e GOTO formam a tabela de parsing utilizada pelo driver do parsing.

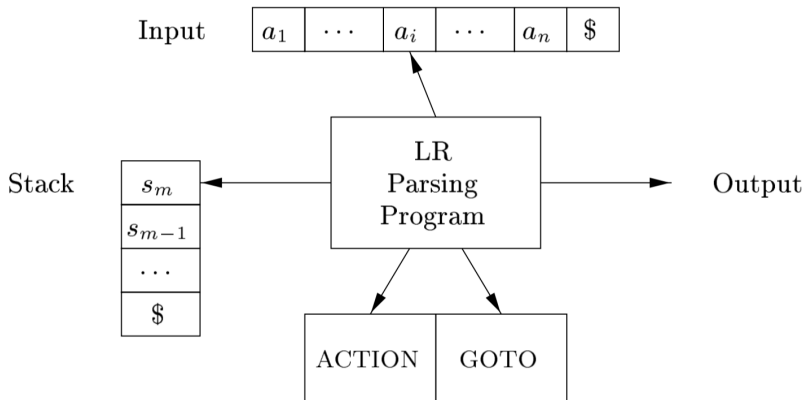


Figure 6: Modelo de um parser LR



# Tabelas de parsing SLR

STATE	ACTION						GOTO		
	id	+	*	(	)	\$	<i>E</i>	<i>T</i>	<i>F</i>
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Figure 7: Tabelas de parsing SLR para a gramática 4.1

# Parsing SLR

1. Construa  $C = \{I_0, I_1, \dots, I_n\}$
2. O  $i$ -ésimo estado é construído a partir de  $I_i$ . As ações associadas ao  $i$ -ésimo estado são:
  - 2.1 Se  $A \rightarrow \alpha \cdot a\beta$  está em  $I_i$  e  $GOTO(I_i, a) = I_j$  então  $ACTION[i, a] = shiftj$ , onde  $a$  é um terminal.
  - 2.2 Se  $A \rightarrow \alpha \cdot$  está em  $I_i$ , então  $ACTION[i, a] = reduce A \rightarrow \alpha$  para todo  $a$  no  $FOLLOW(A)$ , com  $A \neq S'$ .
  - 2.3 Se  $S' \rightarrow S \cdot$  está em  $I_i$ , então  $ACTION[i, \$] = accept$ .
3. As transições para o  $i$ -ésimo estado são construídas para todos os não terminais  $A$  utilizando a regra se  $GOTO(I_i, A) = I_j$  então  $GOTO(i, A) = j$ .