

# V – Teoria de Parsing

## Termos Básicos:

- Parser  $\rightarrow$  Analisador Sintático
- Parsing  $\rightarrow$  Análise Sintática
- Parse  $\rightarrow$  Representação da análise efetuada
  - Ascendentes:  $S \xRightarrow[\text{dir}]{}^+ x$  (\* Seq. Invertida  $\equiv$  Redução \*)
  - Descendentes:  $S \xRightarrow[\text{esq}]{}^+ x$  (\* Seq. Normal  $\equiv$  derivação \*)

## Exemplo:

1:  $S \rightarrow AbC$

2, 3:  $A \rightarrow aA \mid a$

4, 5:  $C \rightarrow cC \mid c$

$x = abc$      $PAx =$   
                   $PDx =$

# FIRST

**Definição** → Conjunto de terminais que podem iniciar uma sequência de símbolos

## Exemplos:

- Se  $\alpha = a \beta$   $\therefore$   $\text{first}(\alpha) = \{a\}$
- Se  $\alpha = \epsilon$   $\therefore$   $\text{first}(\alpha) = \epsilon$
- Se  $\alpha = B \beta$   $\therefore$   $\text{first}(\alpha) = ?? ?$

## Algoritmo:

(\* para todo  $X \in V_n \cup V_t$  \*)

1 – Se  $X \in V_t \rightarrow \text{first}(X) = \{X\}$

2 – Se  $X \in V_n \wedge X \rightarrow a\alpha \in P \Rightarrow a \in \text{First}(X)$

Obs: Se  $X \rightarrow \epsilon \in P \Rightarrow \epsilon \in \text{First}(X)$

3 – Se  $X \in V_n \wedge X \rightarrow y_1 y_2 \dots y_k \in P$

$\Rightarrow \text{First}(y_1) \in \text{First}(X)$

- Se  $\epsilon \in \text{First}(y_1) \Rightarrow \text{First}(y_2)$  também  $\in \text{first}(X)$
- Se  $\epsilon \in \text{First}(y_2) \Rightarrow \dots$
- Se  $\epsilon \in \text{First}(y_k) \Rightarrow \epsilon$  também  $\in \text{First}(X)!!!$

## Exercícios:

1)  $S \rightarrow Ab \mid ABc$

$B \rightarrow bB \mid Ad \mid \epsilon$

$A \rightarrow aA \mid \epsilon$

2)  $S \rightarrow ABC$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid ACd$

$C \rightarrow cC \mid \epsilon$

## Follow

Definição: Seguidores validos de um símbolo!

Se  $S \Rightarrow^+ \alpha A a B \quad \therefore a \in \text{FOLLOW}(A)$

Se  $S \Rightarrow^+ \alpha A B c \gamma \quad \therefore \text{First}(Bc\gamma) \in \text{Follow}(A)$

Algoritmo:

(\* Para todo  $A \in V_n$  \*)

1 – Se A é o símbolo inicial da gramática

$\rightarrow \$ \in \text{Follow}(A)$

2 – Se  $A \rightarrow \alpha B \beta \in P \wedge \beta \neq \epsilon$

$\rightarrow$  adicione  $\text{first}(\beta)$  em  $\text{Follow}(B)$

3 – Se  $A \rightarrow \alpha B$  (ou  $A \rightarrow \alpha B \beta$ , onde  $\epsilon \in \text{First}(\beta)$ )  $\in P$

$\rightarrow$  adicione  $\text{Follow}(A)$  em  $\text{Follow}(B)$

Exemplos:

1)  $S \rightarrow ABC$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid ACd$

$C \rightarrow cC \mid \epsilon$

2)  $E \rightarrow T E'$

$E' \rightarrow + T E' \mid \epsilon$

$T \rightarrow F T'$

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

$F \rightarrow ( E ) \mid id$

3)  $S \rightarrow AbCD \mid EF$

$A \rightarrow aA \mid \epsilon$

$C \rightarrow ECF \mid c$

$D \rightarrow CD \mid dDd \mid \epsilon$

$E \rightarrow eE \mid \epsilon$

$F \rightarrow FS \mid fF \mid g$

4)  $S \rightarrow AC \mid CeB \mid Ba$

$A \rightarrow aA \mid BC$

$C \rightarrow cC \mid \epsilon$

$B \rightarrow bB \mid AB \mid \epsilon$

# Classes de Analisadores

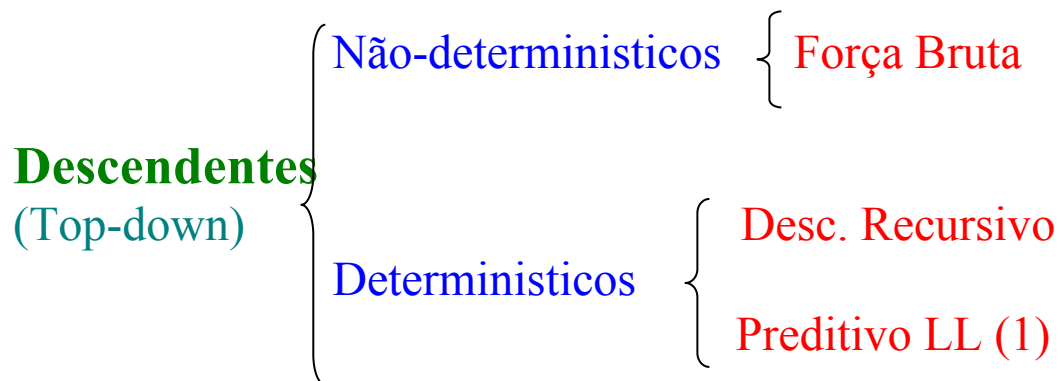
- Ascendentes (Bottom-up)
  - Sentença → Símbolo inicial
  - Uso de redução

- Principais técnicas



- Descendentes (Top-down)
  - Símbolo inicial → Sentença
  - Uso de derivação

- Principais técnicas



- **Técnicas Não-Determinísticas**

- exigem implementação com backtracking
- complexidade exponencial
- não limitam a classe de GLC que pode ser analisadas

**Exemplos:**

Asc. – Alg. geral SHIFT-REDUCE

Desc. – Algoritmo da Força Bruta

- **Técnicas Determinísticas**

- Limitam a classe de GLC que pode ser analisada
- Implementação sem backtracking (determ.)
- Algoritmos eficientes - complexidade linear (espaço requerido proporcional ao tamanho da gramática e tempo de análise proporcional ao tamanho da sentença)
- Parser's automatizáveis
- Principais Técnicas:

**Descendentes (Top-Down)**

- Desecendente Recursiva
- Preditiva (LL(1))

**Ascendente (Bottom-Up)**

- Precedência de Operadores

- LR

- LR(1),LALR(1),SLR(1)

- **Descendentes (Top-down)**

- **Força-Bruta**

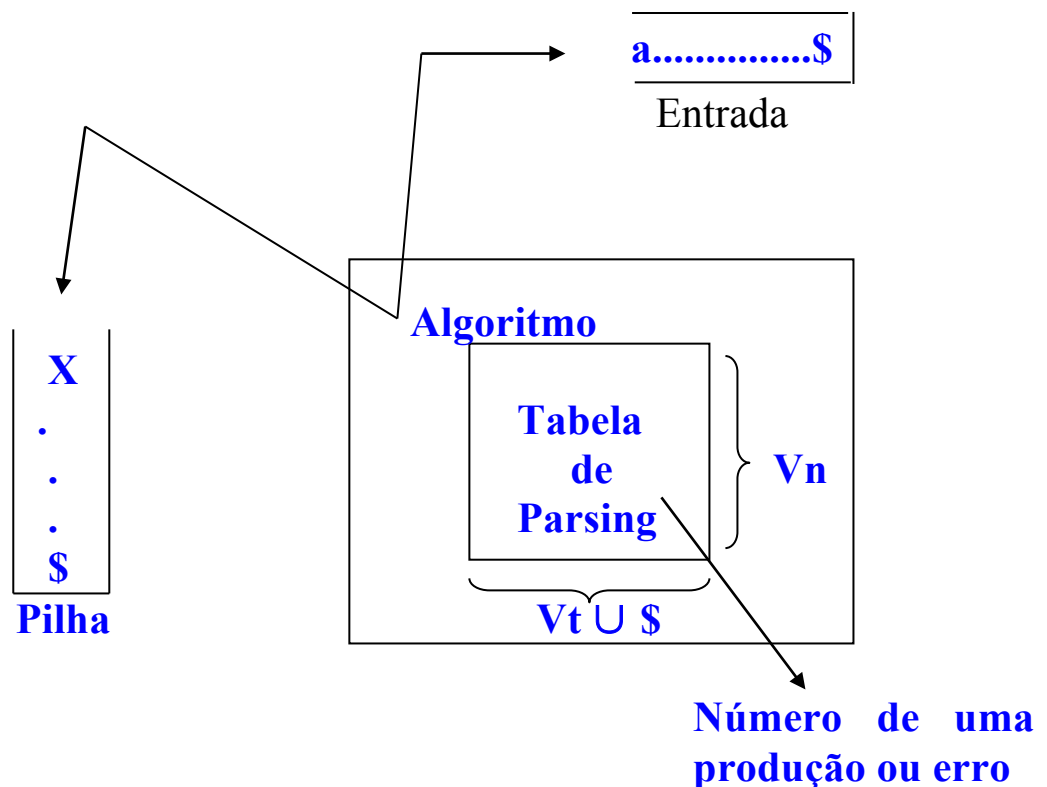
- Não-determinística – com backtracking

- **Descendente recursivo**

- Determinística – sem backtracking
- Procedures mutuamente recursivas
  - (uma para cada não-terminal)
- Vantagens X Desvantagens
- Exemplo:  $S \rightarrow aBc$   
 $B \rightarrow bB \mid d \mid \epsilon$

- **Preditivo (LL(1))**

- **Estrutura Geral**



## • Algoritmo Base

Se  $x \in V_t$

Se  $x = a = \$ \rightarrow$  fim

Se  $x = a \neq \$ \rightarrow$  reconhece a

Se  $x \neq a \rightarrow$  erro sintático

Se  $x \in V_n$

Se  $TP(x,a) = N^\circ$  de produção  $\rightarrow$  deriva!

Se  $TP(x,a) = \text{erro} \rightarrow$  erro sintático!

### Exemplo:

G: 1:  $E \rightarrow TE'$

2,3:  $E' \rightarrow +TE' | \epsilon$

4:  $T \rightarrow FT'$

5,6:  $T' \rightarrow *FT' | \epsilon$

7,8:  $F \rightarrow (E) | id$

TP:	id	(	)	+	*	\$
E	1	1	-	-	-	-
E'	-	-	3	2	-	3
T	4	4	-	-	-	-
T'	-	-	6	6	5	6
F	8	7	-	-	-	-

### • Exercício - Analisar as seguintes sentenças:

▪  $x = id + id \$$

▪  $Y = id * (id id) \$$

## Construção da tabela de parsing LL(1)

- Condição LL(1)

1) Não possuir recursão à esquerda

2) Estar fatorada

3) Para todo  $A \in V_n \mid A \Rightarrow^* \varepsilon$ ,  $\text{First}(A) \cap \text{Follow}(A) = \varnothing$

**$\therefore$  Somente GLC que satisfazem estas condições podem ser analisadas (deterministicamente) pelos Analisadores Descendentes LL(1).**

### Algoritmo para construção da T.P. LL(1)

1) Para cada produção

$$A \rightarrow \alpha \in P$$

Execute os passos 2 e 3:

2) Para todo  $\underline{a} \in \text{First}(\alpha)$ , exceto  $\varepsilon$ ,  
coloque o número da produção  $A \rightarrow \alpha$  em  $\text{TP}(A, a)$

3) Se  $\varepsilon \in \text{first}(\alpha)$   
coloque o número da produção  $A \rightarrow \alpha$   
em  $\text{TP}(A, b)$ , para todo  $\underline{b} \in \text{follow}(A)$

4) Coloque “erro” nas posições da TP que ficaram indefinidas.



## Exercícios

1 -  $C \rightarrow \text{if } E \text{ then } C \text{ else } C$   
     $\quad \mid \text{ if } E \text{ then } C$   
     $\quad \mid \text{ com}$   
 $E \rightarrow \text{exp}$

2 -  $S \rightarrow bCDE \mid Cef$   
     $C \rightarrow cC \mid \varepsilon$   
     $E \rightarrow eSf \mid \varepsilon$   
     $D \rightarrow dDA \mid aDd \mid b$   
     $A \rightarrow cAa \mid aa$

3 -  $P \rightarrow \text{begin } D \text{ } C \text{ end}$   
     $D \rightarrow \text{int id } I$   
     $I \rightarrow , \text{ id } I \mid \varepsilon$   
     $C \rightarrow C; T = E$   
         $\mid T = E$   
         $\mid \text{ com}$   
     $E \rightarrow E + T \mid T$   
     $T \rightarrow \text{id} \mid \text{id } [E]$

- **Ascendentes (Bottom-Up)**

- **Algoritmo Geral Shift-Reduce**

- Não-determinístico – uso de back-track

- Exemplo:

**$S \rightarrow AB$**

**$A \rightarrow ab$**

**$B \rightarrow aba$**

- Vantagens X Desvantagens

- **Analísadores de Precedência de Operadores**

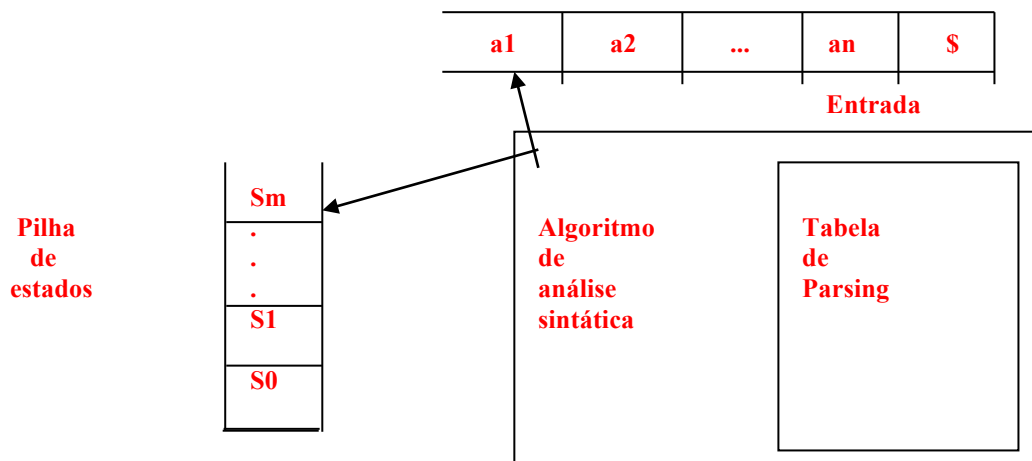
- Determinísticos

- Baseados no Shift-Reduce, acrescido de relações de precedência entre símbolos terminais

- Abrangência limitada para LP

- **Analísadores LR**

- Estrutura Geral de um analisador LR



- **Algoritmo de Análise Sintática “LR”**

$$TP[Si, aj] = \begin{cases} \text{Shift S} \\ \text{Reduce R} \\ \text{Erro} \\ \text{Halt (accept)} \end{cases}$$

- **Configuração de um analisador “LR”**

$(S0_{x1}, S1_{x2}, S2_{x3} \dots S_m, a_i a_{i+1} \dots a_m \$)$

- **Principais Técnicas da Família LR**

**Em ordem crescente de abrangência e complexidade de implementação**

**LR(0)**

**SLR(1)**

**LALR(1)**

**LR(1)**

- **Gramáticas LR e gramaticas LR(K)**

**→ Gramáticas X Técnicas**

- **Exemplo: 0:  $E' \rightarrow ES$**

**1,2:  $E \rightarrow E + T \mid T$**

**3,4:  $T \rightarrow T * F \mid F$**

**5,6:  $F \rightarrow (E) \mid id$**

- Temos a seguinte tabela de parsing

	id	(	)	+	*	\$	E	T	F
0	S5	S4					1	2	3
1				S6		halt			
2			R2	R2	S7	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			S11	S6					
9			R1	R1	S7	R1			
10			R3	R3	R3	R3			
11			R5	R5	R5	R5			

Action

Goto

Exemplo de utilização:

$X = \underline{id * id}$

$Y = \underline{(id + id*) id}$

- **Construção da Tabela de Parsing SLR(1)**

- **Definições gerais:**

- **Item LR(0):**

Seja  $G: S \rightarrow Sa \mid b$

Os itens LR(0) de  $G$  são:

$S \rightarrow \cdot Sa$  (1,0)

$S \rightarrow S \cdot a$  (1,1)

$S \rightarrow Sa \cdot$  (1,2)

$S \rightarrow \cdot b$  (2,0)

$S \rightarrow b \cdot$  (2,1)

└─ posição da marca de análise  
└─ nº de uma produção

- **Item Completo:**

Exemplos:  $S \rightarrow As \cdot$  e  $S \rightarrow b \cdot$

- **Estado:**

- Coleção de itens
- Denota uma situação particular do processo de análise

- **Núcleo de um estado**

- Conjunto de itens que deu origem ao estado
- Ex.: O núcleo do estado inicial de  $G$  seria:

$S \rightarrow \cdot SS$

- **Algoritmo de construção da coleção LR(0)**

- 1) Inicialização**

- a) Numere as produções de G de 1 a p
- b) Inclua a produção 0:  $S' \rightarrow S \$$
- c) Defina o núcleo do estado inicial como sendo  $S' \rightarrow .SS$

- 2) Construção dos estados**

- a) Faça o fechamento do núcleo.
- b) Faça o fechamento dos itens criados no item anterior
- c) Determine o núcleo dos estados sucessores
- d) Repita a, b e c até que todos os estados criados estejam completos.

- **Fechamento**

- Identifica todos os itens equivalentes
- Exemplos:

$$\text{Item } S \rightarrow .Sa \rightarrow \begin{cases} S \rightarrow .Sa \\ S \rightarrow .b \end{cases}$$

$$\text{Item } S \rightarrow S.a \rightarrow \begin{cases} S \rightarrow S.a \end{cases}$$

- **Estados Sucessores**

- Estados resultantes da movimentação da marca de análise sobre um determinado símbolo.

- **Algoritmo para construção da T.P. SLR(1)**

- 1) Crie uma linha para cada estado e uma coluna para cada símbolo ( $V_n \cup V_t \cup \{\$ \}$ )
- 2) Coloque **SHIFT** (ou **GOTO**) nos estados que deram origem a estados sucessores (na coluna do símbolo que deu origem)
- 3) Para cada estado  $i$  com itens completos,  
Para cada **ITEM COMPLETO**,  
Coloque **REDUCE N** na linha do estado  $i$  e nas colunas dos símbolos  $\in \text{Follow}(A)$ , onde  $A$  é o símbolo do lado esquerdo da produção ( $N$ ) correspondente ao item completo em questão ( $N: A \rightarrow d.$ )
- 4) Coloque **HALT** na linha correspondente no estado que contenha o item  $S' \rightarrow S.\$$
- 5) Coloque **ERRO** nas posições que ficarem vazias após a execução dos passos 2, 3 e 4.

- **Condição SLR(1)**

- Para que uma GLC  $G$  seja SLR(1), a tabela construída pelo algoritmo acima não deverá possuir nenhum conflito!!!

- **Exercícios:** Construa a tabela de parsing SLR(1) para as seguintes GLC:

1 –  $E \rightarrow E + T \mid T$   
 $T \rightarrow T * F \mid F$   
 $F \rightarrow ( E ) \mid \text{id}$

2 –  $B \rightarrow A B'$   
 $B' \rightarrow \text{or } A B' \mid \epsilon$   
 $A \rightarrow C A'$   
 $A' \rightarrow \text{and } C A' \mid \epsilon$   
 $C \rightarrow \text{exp} \mid ( B ) \mid \text{not } C$

3 -  $C \rightarrow \text{if } E \text{ then } C \text{ else } C$   
 $\quad \mid \text{if } E \text{ then } C$   
 $\quad \mid \text{com}$   
 $E \rightarrow \text{exp}$

4 -  $S \rightarrow bCDE \mid Cef$   
 $C \rightarrow cC \mid \epsilon$   
 $E \rightarrow eSf \mid \epsilon$   
 $D \rightarrow dDA \mid aDd \mid b$   
 $A \rightarrow cAa \mid aa$

5 -  $P \rightarrow \text{begin } D \ C \ \text{end}$   
 $D \rightarrow \text{int id } I$   
 $I \rightarrow , \text{id } I \mid \epsilon$   
 $C \rightarrow C; T = E$   
 $\quad \mid T = E$   
 $\quad \mid \text{com}$   
 $E \rightarrow E + T \mid T$   
 $T \rightarrow \text{id} \mid \text{id } [E]$



# **Vantagens da Especificação Formal da Sintaxe de Linguagens de Programação**

- **Precisão**
- **Legibilidade**
- **Algoritmos de análise eficientes**
- **Implementação automática de PARSER's**
- **Verificação automática da adequação da especificação para a técnica escolhida**
  - **Detecção de ambiguidades**
    - **conflitos na tabela de parsing**
  - **Recursão à esquerda, Não-fatoração**
- **Facilidade para teste da Consistência da especificação**
  - **presença de símbolos inúteis**
  - **construções reconhecidas X const. desejadas**
    - **uso de simuladores, casos de teste**
- **Facilidade para detecção/recuperação de erros**
- **Possibilidade de implementação de esquemas de tradução dirigidos pela sintaxe**
- **Favorece a extensibilidade de Linguagens**
- **Favorece a corretude do compilador como um todo**

# **Critérios para comparação de técnicas de PARSING**

## **Generalidade**

$$\text{LR}(1) \supset \text{LALR}(1) \supset \text{SLR}(1) \cong \text{LL}(1)$$

## **Facilidade para escrever gramáticas**

$$\text{LR} > \text{LL} > \text{Precedência}$$

## **Facilidade para depurar gramáticas**

$$\text{LL} > \text{Precedência} > \text{LR}$$

## **Deteccão de erros sintáticos**

Todas as técnicas formais possuem a propriedade dos prefixos corretos

## **Suporte para diagnóstico e recuperação de erros**

Todas as técnicas formais suportam

## **Espaço requerido**

$$\text{LR}(1) > \text{LALR}(1) \cong \text{SLR}(1) \cong \text{LL}(1)$$

## **Tempo de análise**

proporcional ao tamanho da sentença

## **Integração com analisador semântico**

possibilitam inclusão de AÇÕES SEMÂNTICAS

## **Geração automática**

possível para todas as técnicas