# V – Teoria de Parsing

## Termos Básicos:

- Parser → Analisador Sintático
- Parsing → Analise Sintática
- Parse → Representação da analise efetuada
  - **Ascendentes: $S \Rightarrow^{+} x$ (* Seq. Invertida ≡ Reducao *)**
  - **Descendentes: $S \Rightarrow^{+} x$ (* Seq. Normal ≡ derivação *)**

## Exemplo:

**1: S → AbC**
**2, 3: A → aA | a**
**4, 5: C → cC | c**

**x = abc PAx =**
**PDx =**

# FIRST

Definição → Conjunto de terminais que podem iniciar uma seqüência de símbolos

**Exemplos:**

- **Se α = a β ∴ first (α) = {a}**
- **Se α = ε ∴ first (α) = ε**
- **Se α = B β ∴ first (α) = ?? ?**

**Algoritmo:**

**(* para todo X ∈ Vn ∪ Vt *)**

**1 – Se X ∈ Vt → first (X) = {X}**

**2 – Se X ∈ Vn ∧ X → aα ∈ P ⇒ a ∈ First (X)**

**Obs: Se X → ε ∈ P ⇒ ε ∈ First (X)**

**3 – Se X → y1 y2 ... yk ∈ P ⇒ First (y1) ∈ First (X)**

- **Se ε ∈ First (y1) ⇒ First (y2) também ∈ first(X)**
- **Se ε ∈ First (y2) ⇒ ...**
- **Se ε ∈ First (yk) ⇒ ε também ∈ First(X)!!!**

**Exercícios:**

**1) S → Ab | ABc**
**B → bB | Ad | ε**
**A → aA | ε**

**2) S → ABC**
**A → aA | ε**
**B → bB | ACd**
**C → cC | ε**

# Follow

**Definição: Seguidores validos de um símbolo!**

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

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

**Algoritmo:**

**(* Para todo A ∈ Vn *)**

**1 – Se A é o símbolo inicial da gramática**

**→ $ ∈ Follow(A)**

**2 – Se A → αBβ ∈ P ∧ β ≠ ε**

**→ adicione first(β) em Follow(B)**

**3 – Se A → αB (ou A→αBβ, onde ε ∈ First(β)) ∈ P**

**→ adicione Follow(A) em Follow(B)**

Exemplos:

1) S → ABC
A → aA | ε
B → bB | ACd
C → cC | ε

2) E → T E'
E' → + T E' | ε
T → F T'
T' → * F T' | ε
F → ( E ) | id

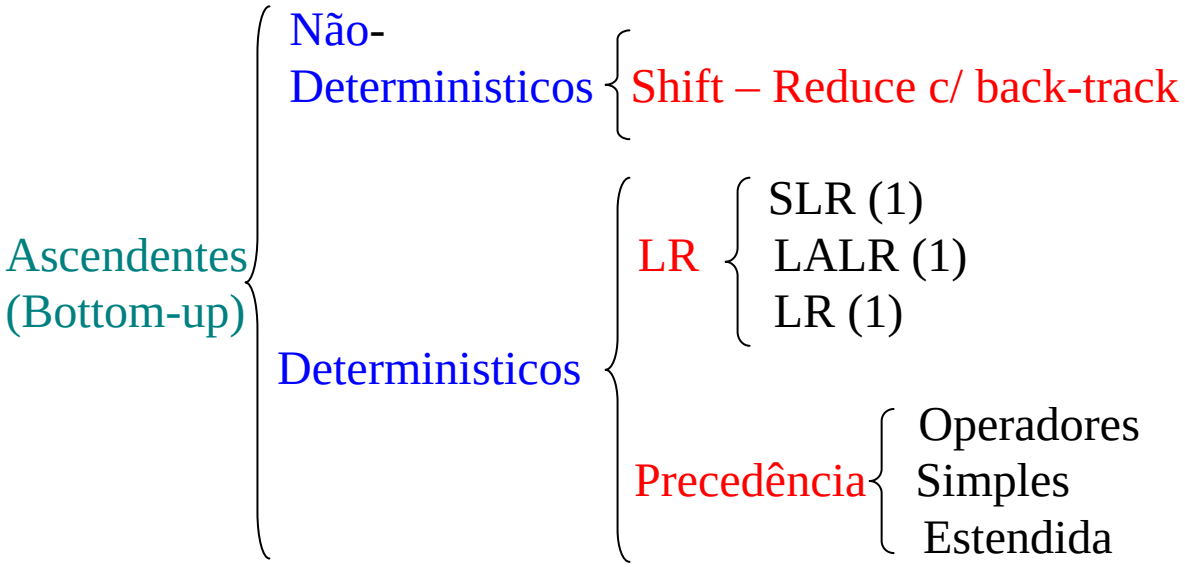
3) S → AbCD | EF
A → aA | ε
C → ECF | c
D → CD | dDd | ε
E → eE | ε
F → FS | fF | g

4) S → AC | CeB | Ba
A → aA | BC
C → cC | ε
B → bB | AB | ε

# Classes de Analisadores

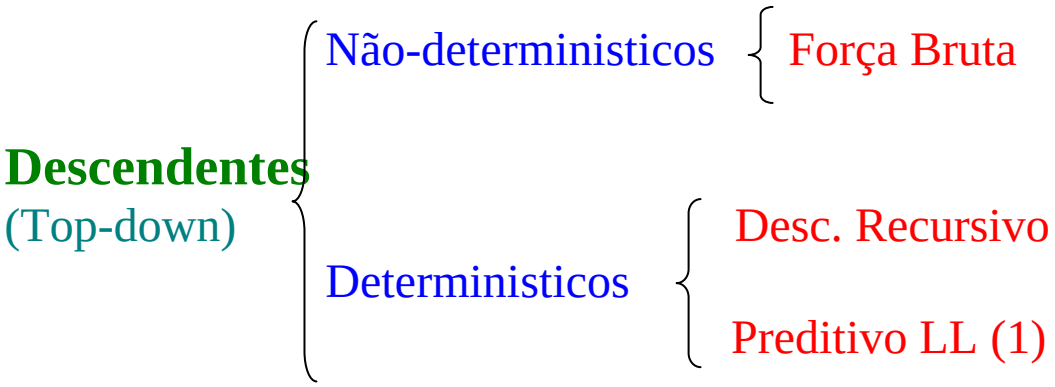
- **Ascendentes (Bottom-up)**
  - **Sentença → Símbolo inicial**
  - **Uso de redução**

- **Principais técnicas**




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

- **Principais técnicas**



- **Técnicas Não-Determinísticas**
  - **exigem implementação com back-track**
  - **complexidade exponencial**
  - **não limitam a classe de GLC que pode ser analisadas**

    **Exemplos:**

    **Asc. – Alg. geral SHIFT-REDUCE**

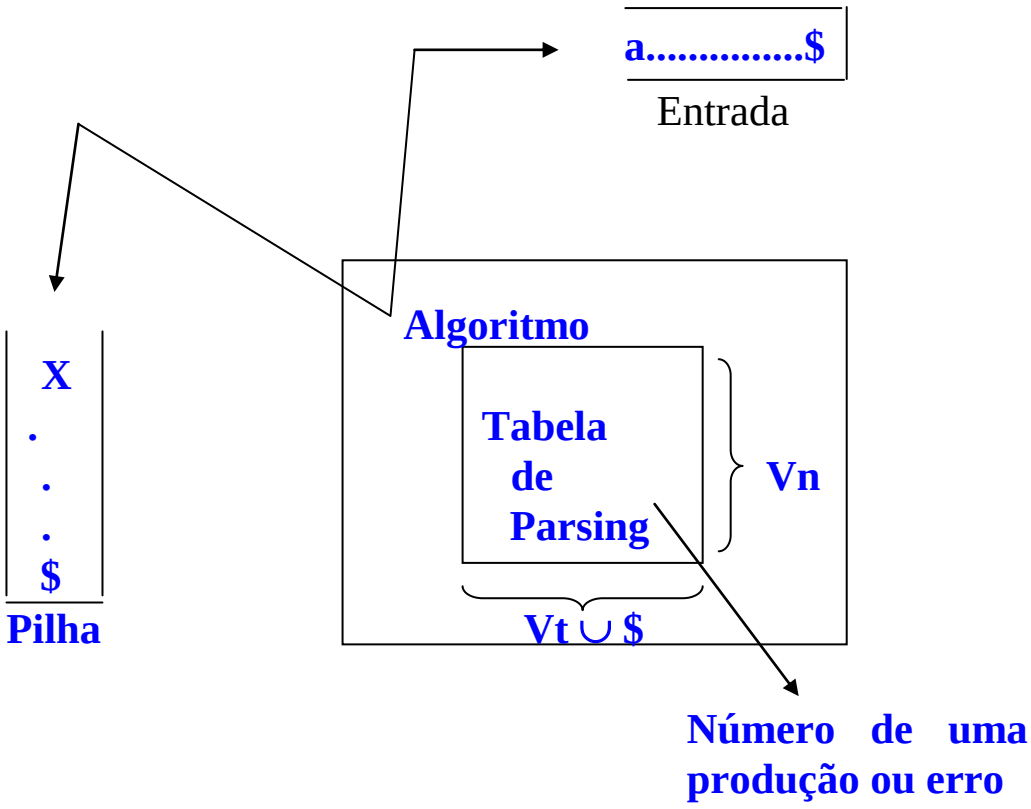
    **Desc. – Algoritmo da Força Bruta**

- **Técnicas Determinísticas**
  - **Limitam a classe de GLC que pode ser analisada**
  - **Implementação sem back-track (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**
      - **Simples, de Operadores, Estendida**
    - **LR**
      - **LR(1),LALR(1),SLR(1)**

- **Descendentes (Top-down)**
  - **Força-Bruta (Não-deterministica–c/ back-track)**
  - **Descendente recursivo**
    - **Procedures mutuamente recursivas**
      - **(uma para cada não-terminal)**
    - **Vantagens X Desvantagens**
    - **Exemplo: S → aBc**
      **B → bB | d | ε**
  - **Preditivo (LL(1))**
    - **Estrutura Geral**



- **Algoritmo Base**

**Se x ∈ Vt**

**Se x = a = $ → fim**
**Se x = a ≠ $ → reconhece a**
**Se x ≠ a → erro sintático**

**Se x ∈ Vn**

Se TP(x,a) = Nº de produção → deriva!
Se TP(x,a) = erro → erro sintático!

**Exemplo:**

**G: 1: E → TE'**
**2,3: E' → +TE'|ε**
**4: T → FT'**
**5,6: T' → * FT'|ε**
**7,8: F → (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 $A \in Vn \mid A \Rightarrow^{*} \varepsilon$, $First(A) \cap Follow(A) = \varphi$**

**$\therefore$ 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 a $\in$ First($\alpha$), exceto $\varepsilon$,**
**coloque o número da produção $A \rightarrow \alpha$ em TP(A,a)**

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

4) **Coloque "erro" mas posições da TP que ficaram indefinidas.**

## Exercícios

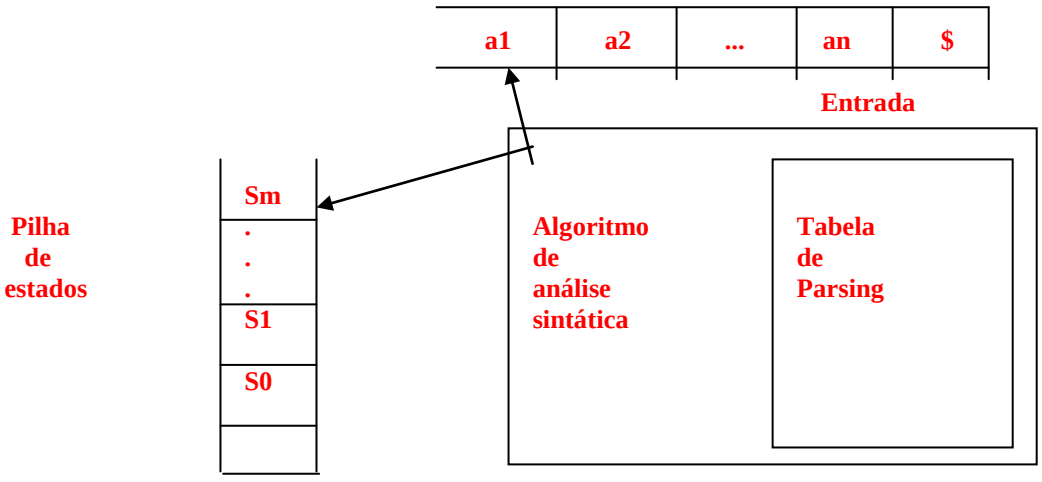
**1 - C → if E then C else C**
**| if E then C**
**| com**
**E → exp**

**2 - S → bCDE | Cef**
**C → cC | ε**
**E → eSf | ε**
**D → dDA | aDd | b**
**A → cAa | aa**

**3 - P → begin D C end**
**D → int id I**
**I → , id I | ε**
**C → C; T = E**
**| T = E**
**| com**
**E → E + T | T**
**T → id | id [E]**

- **Ascendentes (Bottom-Up)**
  - **Algoritmo Geral Shift-Reduce**
    - **Não-deterministico – uso de back-track**
    - **Exemplo:**

      **S → AB**
      **A → ab**
      **B → aba**
    - **Vantagens X Desvantagens**
  - **Analisadores de Precedência**
    - **Precedência Simples, Estendida, de Operadores**
    - **Determinísticos**
    - **Baseados no Shift-Reduce, acrescido de relações de precedência entre Vn e Vt**
    - **Abrangência limitada para LP**
  - **Analisadores 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"**

$$(S0_{x1}, S1_{x2}, S2x3... Sm, a_i\ a_{i+1} ... a_m\ \$)$$

- **Principais Técnicas da Familia 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' → E$

  1,2: E → E + T | T

  3,4: T → T * F | F

  5,6: F → (E) | 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 = id * id

Y = (id + id*) id

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

  - **Definições gerais:**

    - **Item LR(0):**

      **Seja G: S → Sa | b**

      **Os itens LR(0) de G são:**

      **S → . S a (1,0)**
      **S → S.a (1,1)**
      **S → Sa. (1,2)**
      **S → .b (2,0)**
      **S → b. (2,1)**

      **posição da marca de análise**
      **nº de uma produção**

    - **Item Completo:**

      **Exemplos: S → As. e S → b.**

    - **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→ .S$**

- ## 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' → S $

c) Defina o núcleo do estado inicial como sendo S' → .S$

### 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

  - Indentifica todos os itens equivalentes
  - Exemplos:

Item S → .Sa { S → .Sa ; S → .b }

Item S → S.a → { S → S.a }

- ## 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 (Vn ∪ Vt ∪ {$})**

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 ∈ Follow(A), onde A é o símbolo do lado esquerdo da produção (N) correspondente ao item completo em questão (N: A → d.)**

4) **Coloque HALT na linha correspondente no estado que contenha o item S' → 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!!!**

- **Exemplos de GLC para construção da tabela SLR(1)**

**1 – E → E + T | T**
**T → T * F | F**
**F → ( E ) | id**

**2 – B → A B'**
**B' → or A B' | ε**
**A → C A'**
**A' → and C A' | ε**
**C → exp | ( B ) | not C**

**3 - C → if E then C else C**
**| if E then C**
**| com**
**E → exp**

**4 - S → bCDE | Cef**
**C → cC | ε**
**E → eSf | ε**
**D → dDA | aDd | b**
**A → cAa | aa**

**5 - P → begin D C end**
**D → int id I**
**I → , id I | ε**
**C → C; T = E**
**| T = E**
**| com**
**E → E + T | T**
**T → id | 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**

LR(1) $\supset$ LALR(1) $\supset$ SLR(1) $\cong$ LL(1)

**Facilidade para escrever gramáticas**

LR > LL > Precedência

**Facilidade para depurar gramáticas**

LL > Precedência > LR

**Detecção 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**

LR(1) > LALR(1) $\cong$ SLR(1) $\cong$ LL(1)

**Tempo de análise**

proporcional ao tamanho da sentença

**Integração com analisador semântico**

possibilitam inclusão de AÇÕES SEMÂNTICAS

**Disponibilidade de Geradores automáticos**

possível para todas as técnicas
qual(is) está(ão) disponível(is)?