Universidade Federal de Uberlândia
Faculdade de Computação

Análise Sintática (Métodos Preditivos)

Curso de Bacharelado em Ciência da Computação
GBC071 - Construção de Compiladores
Prof. Luiz Gustavo Almeida Martins

Analisador Sintático Preditivo

- Baseado em estratégias **descendentes ou *top-down***

- Árvore de derivação construída **da raiz para as folhas**
  - Construção baseada na **expansão da cabeça** da produção
    - Ponto de partida é o **símbolo sentencial**
  - Sentença é reconhecida se **casar com a forma sentencial derivada** após todas as substituições

- Métodos baseados em **recursão** ou em **tabela**
  - **Ex:** descida recursiva e tabelas de análise preditiva

- Eficiência exige **gramáticas determinísticas**
  - **Remoção da recursão à esquerda**
  - **Tratamento da ambiguidade (ex: fatoração)**

Recursão à Esquerda

- Uma gramática possui **recursão à esquerda** se ela tiver uma produção do tipo:
  - $A \Rightarrow A\alpha$, para qualquer cadeia **α**
- **Métodos preditivos** não lidam bem com esse tipo de gramática
- Diferentes tipos de recursão à esquerda devem ser eliminadas:
  - Remoção da **recursão à esquerda imediata**
  - Remoção da **recursão à esquerda indireta**

Recursão à Esquerda Imediata

- Ocorre quando existem produções do tipo:
  - $A \rightarrow A\alpha$, para qualquer cadeia α

- **Passos para a remoção:**
  - Agrupe as produções pelo não-terminal da cabeça:
    - $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$
    - Sendo que nenhum β_i começa com A e nenhum α_i é ε
  - Substitua as produções-*A* por:
    - $A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_m A' \mid \varepsilon$
    - $A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_n A' \mid \varepsilon$

Recursão à Esquerda Imediata

- **Ex:** expressões aritméticas com precedência de operadores

Gramática Ambígua

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

Elimina ambiguidade

Gramática recursiva a esquerda

- $E \rightarrow E$ **+** $T \mid E$ **-** $T \mid T$
- $T \rightarrow T$ ***** $F \mid T$ **/** $F \mid F$
- $F \rightarrow$ **(** E **)** $\mid$ **id**

Elimina recursão a esquerda

Gramática LL(1)

- $E \rightarrow TE'$
- $E' \rightarrow$ **+** $TE' \mid$ **-** $TE' \mid \varepsilon$
- $T \rightarrow FT'$
- $T' \rightarrow$ ***** $FT' \mid$ **/** $FT' \mid \varepsilon$
- $F \rightarrow$ **(** E **)** $\mid$ **id**

Recursão à Esquerda Indireta

- Ocorre quando a **recursão à esquerda** é obtida a partir de **2 ou mais passos de derivação**
  - **Exemplo:**

$$S \rightarrow Aa \mid b$$
$$A \rightarrow Ac \mid Sd \mid \varepsilon$$

    - Não-terminal ***S*** é recursivo à esquerda: $S \Rightarrow Aa \Rightarrow Sda$

- Existe uma **forma sistemática para remoção** deste tipo de recursão:
  - Garante o funcionamento se a gramática não contiver:
    - Derivações da forma $A \Rightarrow^* A$ (**ciclos**)
    - Produções da forma $A \rightarrow \varepsilon$ (**produções-ε**)
  - Gramática resultante pode conter **produções-ε**

Recursão à Esquerda Indireta

- **Entrada:** Gramática *G* sem ciclos ou produções-ε
- **Saída:** Gramática *G'* sem recursão à esquerda
- **Algoritmo:**

Organize os não-terminais em uma ordem crescente (A_1 , A_2 , … , A_n)

para (cada *j* de 1 até *n*) **faça**

para (cada *i* de 1 até *j-1*) **faça**

Substitua cada produção $A_j \rightarrow A_i \, \alpha$

por produções na forma $A_j \rightarrow \beta_1 \, \alpha \mid \beta_2 \, \alpha \mid \ldots \mid \beta_k \, \alpha$

sendo $A_i \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$ produções-*A*

fim_para

Elimine as recursões à esquerda imediatas nas produções-*A*

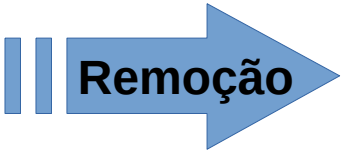
fim_para

Recursão à Esquerda Indireta

- **Exemplo:** Considere a Gramática *G* com as produções:
  - $S \rightarrow Aa \mid b$
  - $A \rightarrow Ac \mid Sd \mid \varepsilon$

Recursão à Esquerda Indireta

- **Exemplo:** Considere a Gramática *G* com as produções:
  - $S \rightarrow Aa \mid b$
  - $A \rightarrow Ac \mid Sd \mid \varepsilon$

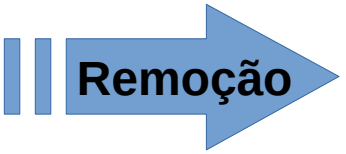


$S \rightarrow Aa \mid b$

- **Passos:**
  - Ordenação dos não-terminais de *G*: {***S*** , ***A***}
  - Para ***j* = 1** (***S***), nada muda nas produções de ***S*** (**1 iteração**)
    - Não existe **recursão à esquerda imediata** em ***S***

Recursão à Esquerda Indireta

- **Exemplo:** Considere a Gramática *G* com as produções:
  - $S \rightarrow Aa \mid b$
  - $A \rightarrow Ac \mid Sd \mid \varepsilon$



$S \rightarrow Aa \mid b$
$A \rightarrow bdA' \mid A'$
$A' \rightarrow cA' \mid adA' \mid \varepsilon$

- **Passos:**
  - Ordenação dos não-terminais de *G*: {*S* , *A*}
  - Para ***j* = 1** (***S***), nada muda nas produções de ***S*** (**1 iteração**)
    - Não existe **recursão à esquerda imediata** em ***S***
  - Para ***j* = 2** (***A***) e ***i* = 1** (***S***), substitui-se ***S*** em $A \rightarrow Sd$
    - **Loop + interno:** $A \rightarrow Ac \mid Aad \mid bd \mid \varepsilon$
    - **Remoção da recursão à esquerda imediata em *A*:**
      - $A \rightarrow bdA' \mid A'$
      - $A' \rightarrow cA' \mid adA' \mid \varepsilon$

Fatoração à Esquerda

- Transformação usada quando não é possível escolher entre 2 ou mais produções para um mesmo símbolo não terminal (produções-*A*)
  - **Produções começam com a mesma forma sentencial α**
  - **Ex: $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$** , sendo **α uma cadeia não-vazia**

- **Ideia:** reescrever as produções a fim de **adiar a decisão** até se obter **um prefixo de entrada que permita realizar a escolha correta**
  - **Ex: $A \rightarrow \alpha A'$ $A' \rightarrow \beta_1 \mid \beta_2$**

- Útil na **construção de gramáticas adequadas** para um **reconhecedor sintático preditivo**

Fatoração à Esquerda

- **Entrada:** Gramática *G*
- **Saída:** Gramática *G'* fatorada à esquerda
- **Algoritmo:**

para (cada símbolo não-terminal *A*) **faça**

enquanto (houver produções-*A* com prefixo comum) **faça**

Encontre o **prefixo mais longo α** comum a 2 ou mais produções-*A*

se (**α ≠ ε**) **então**

- Mantenha as produções-*A* (***A*** → **ω**), onde **ω** não começa com **α**

- Substitua todas as produções-*A* que comecem com **α** :

A → **α** β_1 **|** **α** β_2 **| … |** **α** β_n , por uma única produção: ***A*** → **α** ***A'***

- Crie **produções-*A'*** com o complemento das produções-*A* :

$$A' \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

fim_se

fim_enquanto

fim_para

Fatoração à Esquerda

- **Exemplo:** problema do “else vazio”
  - $S \rightarrow$ **if** E **then** S | **if** E **then** S **else** S | **cmd**
  - $E \rightarrow$ **expr**

- **Passos:**
  - Encontrar o **prefixo mais longo em comum** (**α**)

Fatoração à Esquerda

- **Exemplo:** problema do “else vazio”
  - *S* → **if** *E* **then** *S* | **if** *E* **then** *S* **else** *S* | **cmd**
  - *E* → **expr**

- **Passos:**
  - Encontrar o **prefixo mais longo em comum** (**α**)
  - Substituir as produções-*S* que começam com **α**:
    - *S* → **if** *E* **then** *S* ***S′***

Fatoração à Esquerda

- **Exemplo:** problema do “else vazio”
  - $S \rightarrow$ **if** E **then** S | **if** E **then** S **else** S | **cmd**
  - $E \rightarrow$ **expr**

- **Passos:**
  - Encontrar o **prefixo mais longo em comum** (**α**)
  - Substituir as produções-S que começam com **α**:
    - $S \rightarrow$ **if** E **then** S S'
  - Criar produções-S' com o complemento:
    - $S' \rightarrow$ **else** S | **ε**

Fatoração à Esquerda

- **Exemplo:** problema do “else vazio”
  - *S* → **if** *E* **then** *S* | **if** *E* **then** *S* **else** *S* | **cmd**
  - *E* → **expr**

- **Passos:**
  - Encontrar o **prefixo mais longo em comum** (**α**)
  - Substituir as produções-*S* que começam com **α**:
    - *S* → **if** *E* **then** *S* ***S′***
  - Criar produções-*S′* com o complemento:
    - ***S′*** → **else** *S* | **ε**
  - Manter produções-*S* sem o **α**:
    - *S* → **cmd**

Fatoração à Esquerda

- **Exemplo:** problema do “else vazio”
  - $S \rightarrow$ **if** E **then** S | **if** E **then** S **else** S | **cmd**
  - $E \rightarrow$ **expr**

- Símbolo não-terminal ***E*** não é fatorado

- **Gramática fatorada *G′*:**
  - $S \rightarrow$ **if** E **then** S S' | **cmd**
  - $S' \rightarrow$ **else** S | **ε**
  - $E \rightarrow$ **expr**

Fatoração à Esquerda

- **Exemplo:** problema do “else vazio”
  - $S \rightarrow$ **if** E **then** S | **if** E **then** S **else** S | **cmd**
  - $E \rightarrow$ **expr**

- Símbolo não-terminal ***E*** não é fatorado

- **Gramática fatorada *G′*:**
  - $S \rightarrow$ **if** E **then** S S' | **cmd**
  - $S' \rightarrow$ **else** S | **ε**
  - $E \rightarrow$ **expr**

Ambiguidade permanece, mas pode ser tratada se sempre atribuirmos o **else** para o **then** mais próximo (**ε** só quando não há outra produção possível - ***default***)

FIRST e FOLLOW

- Funções que auxiliam na construção de analisadores sintáticos
  - Ajudam escolher **qual produção aplicar** com base no próximo *token*
  - Podem ser usadas para determinar os ***tokens* de sincronismo** no tratamento de erro (**modo pânico**)

- **FIRST(α)** retorna o **conjunto de símbolos terminais** que iniciam as cadeias derivadas de **α**
  - **α** é qualquer cadeia de símbolos da gramática
  - Se **α** $\Rightarrow^*$ **ε**, então **ε** também está em **FIRST(α)**

- **FOLLOW(*A*)** retorna o **conjunto de símbolos terminais** que podem aparecer imediatamente à direita de ***A*** em alguma forma sentencial
  - ***A*** é um símbolo não-terminal
  - se existe a derivação ***S*** $\Rightarrow$ **α*Aa*β**, então **a** está em **FOLLOW(*A*)**
  - Se ***A*** é o **não-terminal mais à direita** em alguma forma sentencial, então **$** está em **FOLLOW(*A*)** (**$** é o marcador de fim da entrada ou fim do arquivo)

Cálculo da Função FIRST

- Para computar o **FIRST(*X*)** dos símbolos *X* da gramática **(terminais e não-terminais)**, aplique as seguintes regras:
  - Se ***X* é terminal**, então **FIRST(*X*) ← {*X*}**
  - Se ***X* é não-terminal** e ***X*** $\rightarrow$ **ε**, inclua **ε** em **FIRST(*X*)**
  - Se ***X* é não-terminal** e ***X*** $\rightarrow$ $\mathbf{Y_1\ Y_2 \ldots Y_n}$, então coloque ***a*** no **FIRST(*X*)**, se:
    - ***a*** está em $\mathbf{FIRST(Y_1)}$
    - ***a*** está em $\mathbf{FIRST(Y_i)}$ e **ε** está em todos os símbolos não-terminais precedentes $\mathbf{FIRST(Y_1)}$, …, $\mathbf{FIRST(Y_{i-1})}$, ou seja, $\mathbf{Y_1\ Y_2 \ldots Y_{i-1}} \Rightarrow$ **ε**
      - **Ex:** $\mathbf{FIRST(Y_2)}$ só estará em **FIRST(*X*)**, se $\mathbf{FIRST(Y_1)}$ tenha **ε**
    - Se **ε** está em **todos** $\mathbf{FIRST(Y_k)}$, para **k = 1, 2, …, n**, então **ε** também estará em **FIRST(*X*)**

- As regras são aplicadas até **não haver mais símbolos (terminais ou ε) que possam ser incluídos** em nenhum dos conjuntos **FIRST**

Cálculo da Função FOLLOW

- Para computar o **FOLLOW(*A*) de todos os símbolos não-terminais *A* da gramática**, aplique as seguintes regras:
  - Se ***A* é o símbolo inicial** (***S***), então coloque **$** em **FOLLOW(*A*)**
  - Se ***B*** → **α*A*β**, então **FOLLOW(*A*)** ← **FIRST(β) -** {**ε**}
  - Se ***B*** → **α*A*** ou (***B*** → **α*A*β** e **FIRST(β)** contém **ε**), então **FOLLOW(*A*)** ← **FOLLOW(*B*)**

- As regras são aplicadas até **não haver mais símbolos que possam ser incluídos** em nenhum dos conjuntos **FOLLOW**

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = **FIRST(*F*)** = { **(** , **id** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** } **(*E* é o símbolo inicial da gramática)**

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow +TE' \mid \varepsilon$
  - $T \rightarrow FT'$
  - $T' \rightarrow *FT' \mid \varepsilon$
  - $F \rightarrow (E) \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*E'*) = **FOLLOW(*E*)** = { **)** , **$** }

(*E'* é o símbolo mais a direita da produção-*E*)

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow +TE' \mid \varepsilon$
  - $T \rightarrow FT'$
  - $T' \rightarrow *FT' \mid \varepsilon$
  - $F \rightarrow (E) \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*E'*) = FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*T*) = (**FIRST(*E'*) – ε**) ∪ FOLLOW(*E'*) = { **+** , **)** , **$** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow +TE' \mid \varepsilon$
  - $T \rightarrow FT'$
  - $T' \rightarrow *FT' \mid \varepsilon$
  - $F \rightarrow (E) \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*E'*) = FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*T*) = (FIRST(*E'*) – **ε**) ∪ **FOLLOW(*E'*)** = { **+** , **)** , **$** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*E'*) = FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*T*) = (FIRST(*E'*) – **ε**) ∪ FOLLOW(*E'*) = { **+** , **)** , **$** }

FOLLOW(*T'*) = **FOLLOW(*T*)** = { **+** , **)** , **$** }

(*T'* é o símbolo mais a direita da produção-*T*)

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*E'*) = FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*T*) = (FIRST(*E'*) – **ε**) ∪ FOLLOW(*E'*) = { **+** , **)** , **$** }

FOLLOW(*T'*) = FOLLOW(*T*) = { **+** , **)** , **$** }

FOLLOW(*F*) = (**FIRST(*T'*) – ε**) ∪ FOLLOW(*T'*) = { ***** , **+** , **)** , **$** }

Exemplo

- Considere a gramática sem recursão à esquerda:
  - $E \rightarrow TE'$
  - $E' \rightarrow \mathbf{+}TE' \mid \boldsymbol{\varepsilon}$
  - $T \rightarrow FT'$
  - $T' \rightarrow \mathbf{*}FT' \mid \boldsymbol{\varepsilon}$
  - $F \rightarrow \mathbf{(}\ E\ \mathbf{)} \mid \mathbf{id}$

FIRST(*F*) = { **(** , **id** }

FIRST(*T'*) = { ***** , **ε** }

FIRST(*T*) = FIRST(*F*) = { **(** , **id** }

FIRST(*E'*) = { **+** , **ε** }

FIRST(*E*) = **FIRST(*T*)** = **FIRST(*F*)** = { **(** , **id** }

FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*E'*) = FOLLOW(*E*) = { **)** , **$** }

FOLLOW(*T*) = (FIRST(*E'*) – **ε**) ∪ FOLLOW(*E'*) = { **+** , **)** , **$** }

FOLLOW(*T'*) = FOLLOW(*T*) = { **+** , **)** , **$** }

FOLLOW(*F*) = (FIRST(*T'*) – **ε**) ∪ **FOLLOW(*T'*)** = { ***** , **+** , **)** , **$** }

Referências Bibliográficas

- Aho, A.V.; Lam, M.S.; Sethi, R.; Ullman, J.D. Compiladores: Princípios, técnicas e ferramentas, 2 ed., Pearson, 2008
- Alexandre, E.S.M. Livro de Introdução a Compiladores, UFPB, 2014
- Aluisio, S. material da disciplina “Teoria da Computação e Compiladores”, ICMC/USP, 2011
- Dubach, C. material da disciplina “*Compiling Techniques*”, University of Edinburgh, 2018
- Freitas, R. L. notas de aula - Compiladores, PUC Campinas, 2000
- Menezes, P.B. Linguagens Formais e Autômatos, 6 ed., Bookman, 2011
- Ricarte, I. Introdução à Compilação, Elsevier, 2008