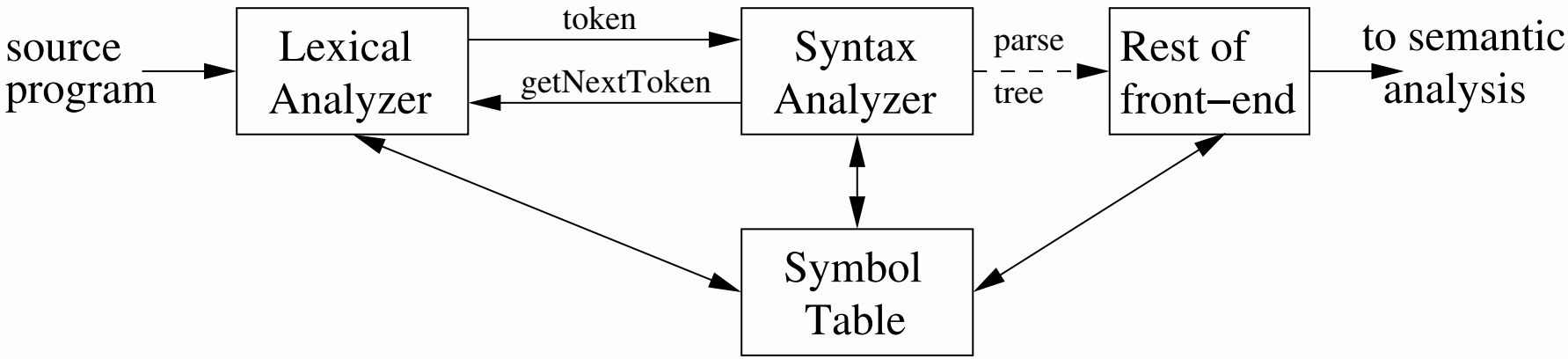
Introdução à Análise Sintáctica



Para além da construção da árvore sintáctica, é tarefa do compilador:

- Identificar a presença de erros de forma clara e precisa
- Recuperar rapidamente de um erro para detectar potenciais erros seguintes
- Adicionar overhead mínimo no processamento de programas correctos

Para além da construção da árvore sintáctica, é tarefa do compilador:

- Identificar a presença de erros de forma clara e precisa
- Recuperar rapidamente de um erro para detectar potenciais erros seguintes
- Adicionar overhead mínimo no processamento de programas correctos

Estratégia simples - reportar a linha (e a posição) no programa onde o erro foi detectado.

Outline

Gramáticas

Motivação

- Gramática serve para produzir/derivar palavras a partir de outras palavras, **recursivamente**, começando no símbolo inicial:

$$S \to 0S1$$
$$S \to \epsilon$$

- As gramáticas são uma notação formal para representar **definições recursivas** de linguagens

Vantagens:

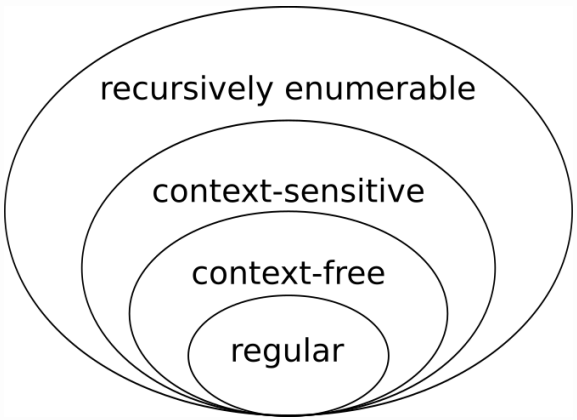
- Descrição precisa e fácil de compreender da sintaxe de uma linguagem
- Extensão e actualização fácil
- Possibilidade de gerar automaticamente um analisador sintáctico eficiente
- Relação directa com a estrutura da linguagem usada
- ...

Backus-Naur Form (BNF) é uma notação para gramáticas livres de contexto com regras na forma

$$\text{Símbolo} \rightarrow \text{expressão1} \mid \ldots \mid \text{expressãoN}$$

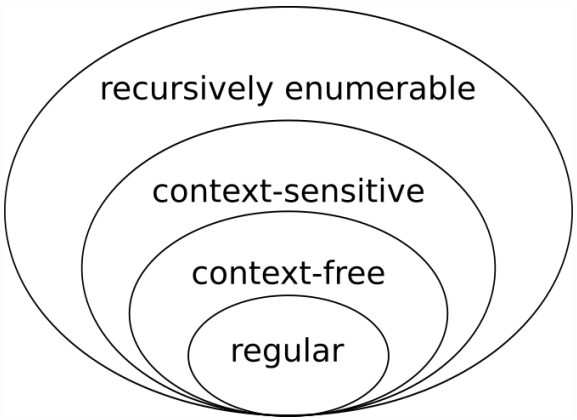
Gramáticas

Hierarquia de Chomsky



Gramáticas

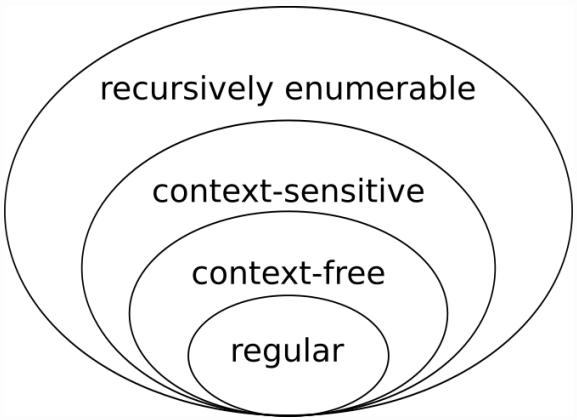
Hierarquia de Chomsky



- **regulares** - representam linguagens regulares. Podem ser representada por expressões regulares ou um autómato finito. forma: $A \rightarrow aB$

Gramáticas

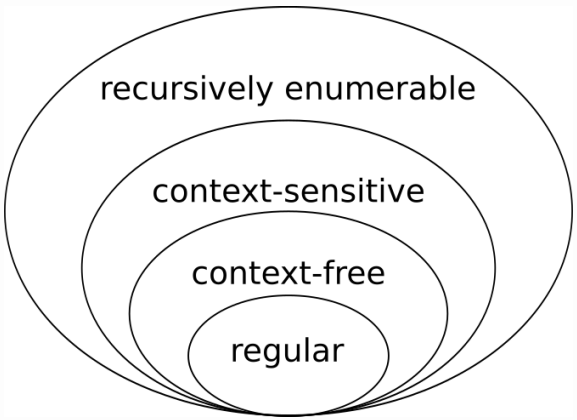
Hierarquia de Chomsky



- **regulares** - representam linguagens regulares. Podem ser representada por expressões regulares ou um autómato finito. forma: $A \rightarrow aB$
- **livres de contexto** - representam linguagens livres de contexto. Podem ser representadas por um autómato de pilha forma: $A \rightarrow \gamma$

Gramáticas

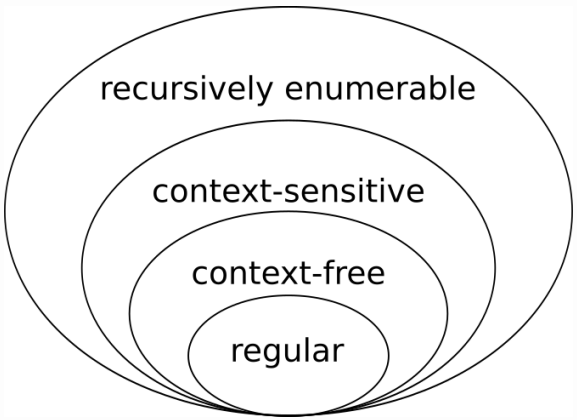
Hierarquia de Chomsky



- **regulares** - representam linguagens regulares. Podem ser representada por expressões regulares ou um autómato finito. forma: $A \rightarrow aB$
- **livres de contexto** - representam linguagens livres de contexto. Podem ser representadas por um autómato de pilha forma: $A \rightarrow \gamma$
- outras ...
  - expressividade
  - facilidade fazer parsing

Gramáticas

Hierarquia de Chomsky



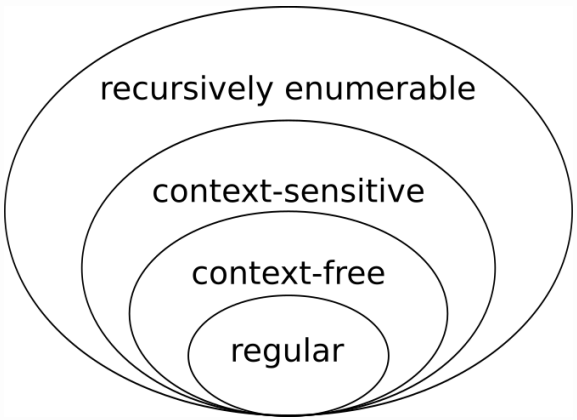
- **regulares** - representam linguagens regulares. Podem ser representada por expressões regulares ou um autómato finito. forma: $A \rightarrow aB$
- **livres de contexto** - representam linguagens livres de contexto. Podem ser representadas por um autómato de pilha forma: $A \rightarrow \gamma$
- outras ...
  - expressividade
  - facilidade fazer parsing

Exemplo: $L = \{0^n 1^n | n \geq 0\}$

- L não é regular. Porquê?

Gramáticas

Hierarquia de Chomsky



- **regulares** - representam linguagens regulares. Podem ser representada por expressões regulares ou um autómato finito. forma: $A \rightarrow aB$
- **livres de contexto** - representam linguagens livres de contexto. Podem ser representadas por um autómato de pilha forma: $A \rightarrow \gamma$
- outras ...
  - expressividade
  - facilidade fazer parsing

Exemplo: $L = \{0^n 1^n | n \geq 0\}$

- L não é regular. Porquê?
- DFA não permite memorizar o número de 0's

Gramáticas livres de contexto

Definição

Uma **gramática livre de contexto** $G = (V, \Sigma, R, S)$ é um tuplo onde:

- V é o conjunto de **símbolos não terminais**
- Σ é o conjunto de **símbolos terminais**
- R é o conjunto finito de **produções** ou **regras**, com a forma $X \to Y_1 Y_2 \ldots Y_n$, onde:
  - X é o símbolo não terminal a ser definido
  - $\to$ é o símbolo de produção
  - $Y_1 Y_2 \cdots Y_n$ é o corpo da produção
- $S \in V$ é o **símbolo inicial**

Hopcroft and Ullman, 1979

Gramáticas livres de contexto

Exemplo

Gramática para a definição de expressões aritméticas. **Exemplo**:

- $V = \{expr, op\}$
- $\Sigma = \{id, +, -, *, /, (,)\}$
- Regras:

  $expr \rightarrow expr\ op\ expr$
  $expr \rightarrow (\ expr\ )$
  $expr \rightarrow$ - $expr$
  $expr \rightarrow id$
  $op \rightarrow + \mid$ - $\mid$ * $\mid /$
- $S = expr$

Gramáticas livres de contexto

Derivações

Considere uma **gramática livre de contexto** $G = (V, \Sigma, R, S)$:

Gramáticas livres de contexto

Derivações

Considere uma **gramática livre de contexto** $G = (V, \Sigma, R, S)$:

- **Passo de Derivação**:
  - Palavra actual: $\alpha A \beta$
  - Seja $A \to \gamma$ uma produção de G
  - $\alpha A \beta \underset{G}{\Rightarrow} \alpha \gamma \beta$ representa um passo de derivação da palavra $\alpha \gamma \beta$ a partir $\alpha A \beta$

Gramáticas livres de contexto

Derivações

Considere uma **gramática livre de contexto** $G = (V, \Sigma, R, S)$:

- **Passo de Derivação**:
  - Palavra actual: $\alpha A \beta$
  - Seja $A \to \gamma$ uma produção de G
  - $\alpha A \beta \underset{G}{\Rightarrow} \alpha \gamma \beta$ representa um passo de derivação da palavra $\alpha \gamma \beta$ a partir $\alpha A \beta$
- **Derivações**: * representa 0 ou mais passos de derivação:
  - **Caso base**: Para qualquer palavra α de terminais e não terminais, $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$
  - **Indução**: Se $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$ e $\beta \underset{G}{\Rightarrow} \gamma$, então $\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma$

Gramáticas livres de contexto

Derivações

Considere uma **gramática livre de contexto** $G = (V, \Sigma, R, S)$:

- **Passo de Derivação**:
  - Palavra actual: $\alpha A \beta$
  - Seja $A \to \gamma$ uma produção de G
  - $\alpha A \beta \underset{G}{\Rightarrow} \alpha \gamma \beta$ representa um passo de derivação da palavra $\alpha \gamma \beta$ a partir $\alpha A \beta$
- **Derivações**: * representa 0 ou mais passos de derivação:
  - **Caso base**: Para qualquer palavra α de terminais e não terminais, $\alpha \overset{*}{\underset{G}{\Rightarrow}} \alpha$
  - **Indução**: Se $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$ e $\beta \underset{G}{\Rightarrow} \gamma$, então $\alpha \overset{*}{\underset{G}{\Rightarrow}} \gamma$
- Nota: pode utilizar-se $\Rightarrow$ e $\overset{*}{\Rightarrow}$ como alternativa a $\underset{G}{\Rightarrow}$ e $\overset{*}{\underset{G}{\Rightarrow}}$, respectivamente

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$$S \stackrel{*}{\Rightarrow} 11011 \qquad \text{Sim}$$

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$$S \stackrel{*}{\Rightarrow} 11011 \qquad \text{Sim}$$

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$$S \stackrel{*}{\Rightarrow} 11011 \qquad \text{Sim}$$
$$S \stackrel{*}{\Rightarrow} 1001$$

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$$S \stackrel{*}{\Rightarrow} 11011 \qquad \text{Sim}$$
$$S \stackrel{*}{\Rightarrow} 1001 \qquad \text{Sim}$$

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$$S \stackrel{*}{\Rightarrow} 11011 \qquad \text{Sim}$$
$$S \stackrel{*}{\Rightarrow} 1001 \qquad \text{Sim}$$
$$S \stackrel{*}{\Rightarrow} 1011$$

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$S \stackrel{*}{\Rightarrow} 11011$	Sim
$S \stackrel{*}{\Rightarrow} 1001$	Sim
$S \stackrel{*}{\Rightarrow} 1011$	Não

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$S \stackrel{*}{\Rightarrow} 11011$	Sim
$S \stackrel{*}{\Rightarrow} 1001$	Sim
$S \stackrel{*}{\Rightarrow} 1011$	Não
$S \stackrel{*}{\Rightarrow} 010010$	

Gramáticas livres de contexto

Derivações

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$S \stackrel{*}{\Rightarrow} 11011$	Sim
$S \stackrel{*}{\Rightarrow} 1001$	Sim
$S \stackrel{*}{\Rightarrow} 1011$	Não
$S \stackrel{*}{\Rightarrow} 010010$	Sim

Considere a **gramática livre de contexto** G:

$$S \to 1S1 \,|\, 0S0 \,|\, 1 \,|\, 0 \,|\, \epsilon$$

- Quais as palavras que G gera?
- Exemplos:

$S \stackrel{*}{\Rightarrow} 11011$	Sim
$S \stackrel{*}{\Rightarrow} 1001$	Sim
$S \stackrel{*}{\Rightarrow} 1011$	Não
$S \stackrel{*}{\Rightarrow} 010010$	Sim

- Resposta: G gera palíndromos sobre o alfabeto $\{0, 1\}$

Gramáticas livres de contexto

Linguagem gerada

Dada uma **gramática livre de contexto** $G = (N, \Sigma, R, S)$:

- linguagem de G, L(G) é o conjunto de palavras com derivações a partir do símbolo inicial

$$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

Gramáticas livres de contexto

Derivações: à esquerda e à direita

- Derivação à esquerda:
  Substituir símbolo não terminal mais à esquerda a cada passo de derivação

Gramáticas livres de contexto

Derivações: à esquerda e à direita

- Derivação à esquerda:
  Substituir símbolo não terminal mais à esquerda a cada passo de derivação
  - Notação: $\underset{lm}{\Longrightarrow}$ e $\overset{*}{\underset{lm}{\Longrightarrow}}$

Gramáticas livres de contexto

Derivações: à esquerda e à direita

- Derivação à esquerda:
  Substituir símbolo não terminal mais à esquerda a cada passo de derivação
  - Notação: $\underset{lm}{\Longrightarrow}$ e $\overset{*}{\underset{lm}{\Longrightarrow}}$
  - Exemplo. Derivação à esquerda da *string* $id * (-id)$:

$$\begin{aligned} expr &\underset{lm}{\Longrightarrow} expr\ op\ expr \underset{lm}{\Longrightarrow} id\ op\ expr \underset{lm}{\Longrightarrow} id * expr \\ &\underset{lm}{\Longrightarrow} id * (expr) \underset{lm}{\Longrightarrow} id * (-expr) \underset{lm}{\Longrightarrow} id * (-id) \end{aligned}$$

Gramáticas livres de contexto

Derivações: à esquerda e à direita

- Derivação à esquerda:
  Substituir símbolo não terminal mais à esquerda a cada passo de derivação
  - Notação: $\underset{lm}{\Longrightarrow}$ e $\overset{*}{\underset{lm}{\Longrightarrow}}$
  - Exemplo. Derivação à esquerda da *string* $id * (-id)$:

$$\begin{aligned} expr &\underset{lm}{\Longrightarrow} expr\ op\ expr \underset{lm}{\Longrightarrow} id\ op\ expr \underset{lm}{\Longrightarrow} id * expr \\ &\underset{lm}{\Longrightarrow} id * (expr) \underset{lm}{\Longrightarrow} id * (-expr) \underset{lm}{\Longrightarrow} id * (-id) \end{aligned}$$

- Derivação à direita:
  Substituir símbolo não terminal mais à direita a cada passo de derivação

Gramáticas livres de contexto

Derivações: à esquerda e à direita

- Derivação à esquerda:
  Substituir símbolo não terminal mais à esquerda a cada passo de derivação
  - Notação: $\underset{lm}{\Longrightarrow}$ e $\overset{*}{\underset{lm}{\Longrightarrow}}$
  - Exemplo. Derivação à esquerda da *string* $id * (-id)$:

$$\begin{aligned} expr &\underset{lm}{\Longrightarrow} expr\ op\ expr \underset{lm}{\Longrightarrow} id\ op\ expr \underset{lm}{\Longrightarrow} id * expr \\ &\underset{lm}{\Longrightarrow} id * (expr) \underset{lm}{\Longrightarrow} id * (-expr) \underset{lm}{\Longrightarrow} id * (-id) \end{aligned}$$

- Derivação à direita:
  Substituir símbolo não terminal mais à direita a cada passo de derivação
  - Notação: $\underset{rm}{\Longrightarrow}$ e $\overset{*}{\underset{rm}{\Longrightarrow}}$

Gramáticas livres de contexto

Derivações: à esquerda e à direita

- Derivação à esquerda:
  Substituir símbolo não terminal mais à esquerda a cada passo de derivação
  - Notação: $\underset{lm}{\Longrightarrow}$ e $\overset{*}{\underset{lm}{\Longrightarrow}}$
  - Exemplo. Derivação à esquerda da *string* $id * (-id)$:

$$\begin{aligned} expr &\underset{lm}{\Longrightarrow} expr\ op\ expr \underset{lm}{\Longrightarrow} id\ op\ expr \underset{lm}{\Longrightarrow} id * expr \\ &\underset{lm}{\Longrightarrow} id * (expr) \underset{lm}{\Longrightarrow} id * (-expr) \underset{lm}{\Longrightarrow} id * (-id) \end{aligned}$$

- Derivação à direita:
  Substituir símbolo não terminal mais à direita a cada passo de derivação
  - Notação: $\underset{rm}{\Longrightarrow}$ e $\overset{*}{\underset{rm}{\Longrightarrow}}$
  - Exemplo. Derivação à direita da *string* $id * (-id)$:

$$\begin{aligned} expr &\underset{rm}{\Longrightarrow} expr\ op\ expr \underset{rm}{\Longrightarrow} expr\ op\ (expr) \underset{rm}{\Longrightarrow} expr\ op\ (-expr) \\ &\underset{rm}{\Longrightarrow} expr\ op\ (-id) \underset{rm}{\Longrightarrow} expr\ *\ (-id) \underset{rm}{\Longrightarrow} id * (-id) \end{aligned}$$

Árvore sintáctica

Definição

Uma árvore sintáctica (ou de parsing) é uma representação gráfica de uma derivação.

- A ordem em que as produções são aplicadas não é representada

Árvore sintáctica

Definição

Uma árvore sintáctica (ou de parsing) é uma representação gráfica de uma derivação.

- A ordem em que as produções são aplicadas não é representada

Árvore de sintáctica para $G = (V, \Sigma, R, S)$:

- Cada nó interior etiquetado com um símbolo não terminal de V
- Cada nó folha etiquetado com um símbolo terminal, não terminal, ou ϵ
  - Se um nó folha é etiquetado com ϵ, então é o único filho do nó pai
- Se nó interior é etiquetado com A e os filhos são etiquetados com $X_1, X_2, \ldots, X_k$ (a partir da esquerda), então $A \to X_1 X_2 \ldots X_k$ é uma produção de G

Árvore sintáctica

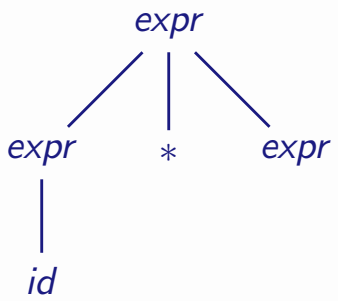
Exemplo

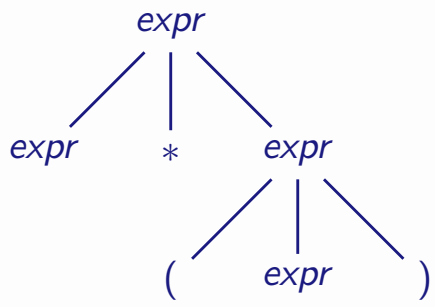
- Derivação de $id * (-id)$ a partir de *expr*

Árvore sintáctica

Exemplo

- Derivação de $id * (-id)$ a partir de *expr*

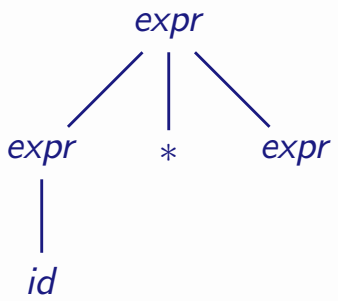


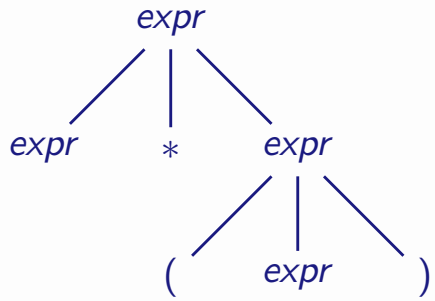


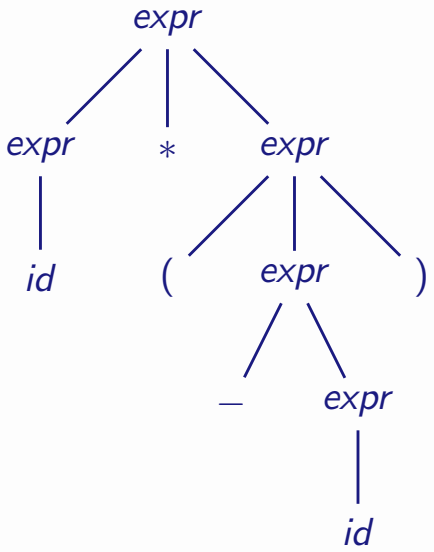
Árvore sintáctica

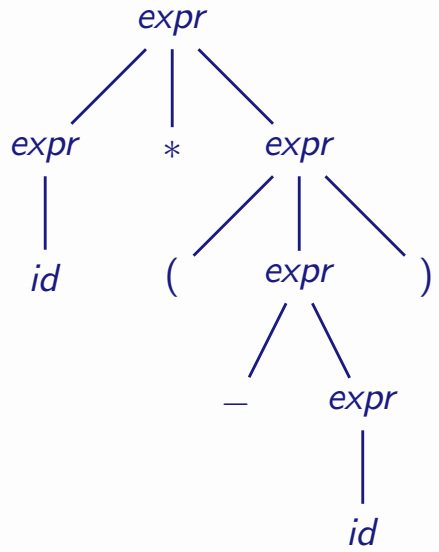
Exemplo

- Derivação de $id * (-id)$ a partir de *expr*









Árvore sintáctica

Resultado

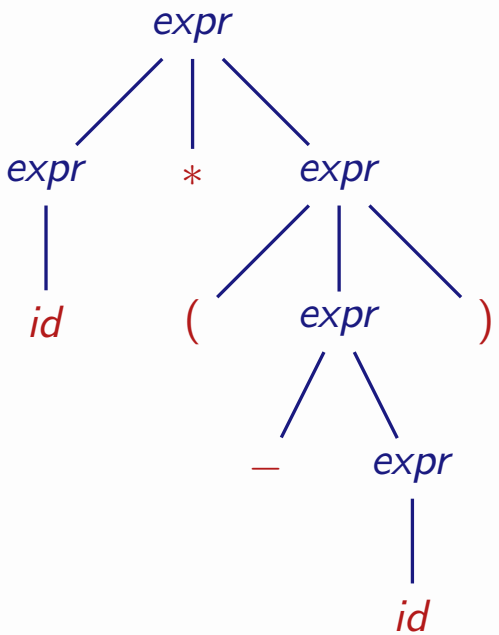
Resultado: nós folha da árvore sintáctica concatenados da esquerda para a direita

Árvore sintáctica

Resultado

Resultado: nós folha da árvore sintáctica concatenados da esquerda para a direita

Exemplo:

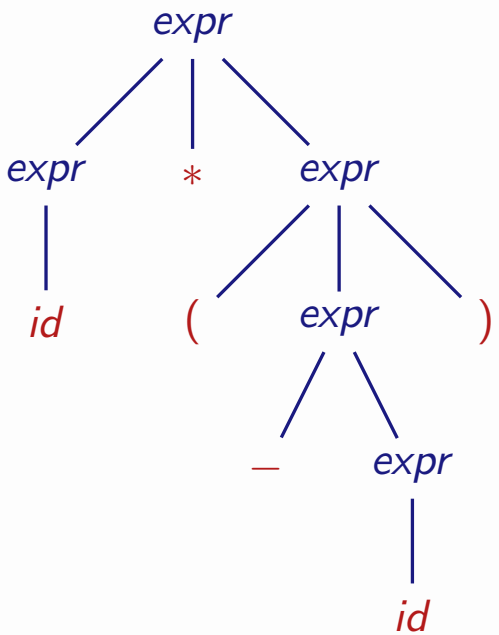


Árvore sintáctica

Resultado

Resultado: nós folha da árvore sintáctica concatenados da esquerda para a direita

Exemplo:



Resultado é: id * (- id)

Propriedades:

- Ambiguidade
- Precedência
- Associatividade

Gramáticas livres de contexto

Propriedades - Ambiguidade

Uma gramática $G = V, \Sigma, R, S)$ diz-se ambígua se existe pelo menos uma palavra em Σ^* que tem duas árvores sintácticas diferentes, ambas com raíz S

Gramáticas livres de contexto

Propriedades - Ambiguidade

Uma gramática $G = V, \Sigma, R, S)$ diz-se ambígua se existe pelo menos uma palavra em Σ^* que tem duas árvores sintácticas diferentes, ambas com raíz S

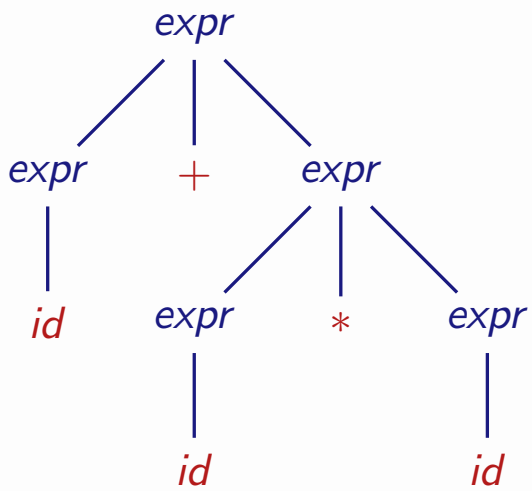
Exemplo: id + id * id

Gramáticas livres de contexto

Propriedades - Ambiguidade

Uma gramática $G = V, \Sigma, R, S)$ diz-se ambígua se existe pelo menos uma palavra em Σ^* que tem duas árvores sintácticas diferentes, ambas com raíz S

Exemplo: id + id * id

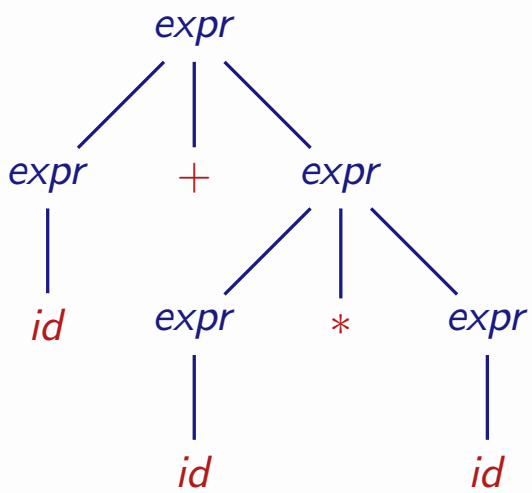


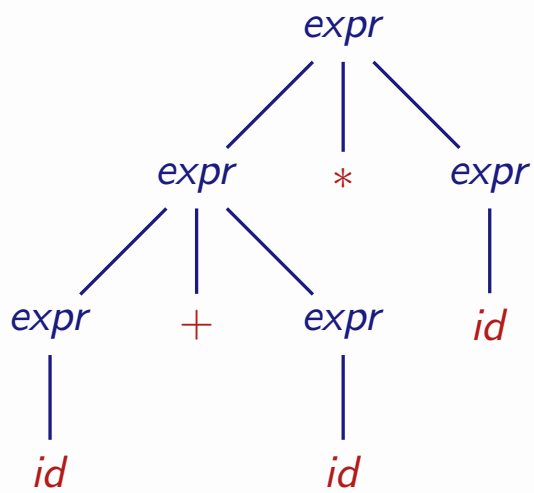
Gramáticas livres de contexto

Propriedades - Ambiguidade

Uma gramática $G = V, \Sigma, R, S)$ diz-se ambígua se existe pelo menos uma palavra em Σ^* que tem duas árvores sintácticas diferentes, ambas com raíz S

Exemplo: id + id * id



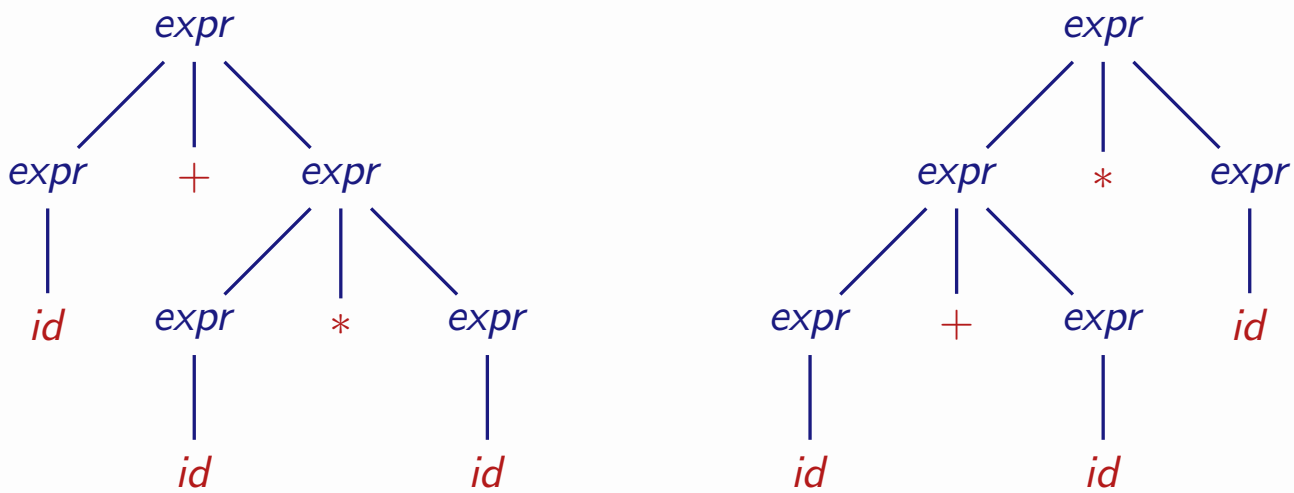


Gramáticas livres de contexto

Propriedades - Ambiguidade

Uma gramática $G = V, \Sigma, R, S)$ diz-se ambígua se existe pelo menos uma palavra em Σ^* que tem duas árvores sintácticas diferentes, ambas com raíz S

Exemplo: id + id * id



Nota: Não é possível processar gramáticas ambíguas de forma eficiente!

Por vezes é necessário transformar a gramática de forma a remover as ambiguidades.

A associatividade é uma propriedade de operadores binários.

$$\forall x, y, z \quad (x \Delta y) \Delta z = x \Delta (y \Delta z)$$

Mas nem todos os operadores binários são associativos.

- e.g. subtracção não é associativa $(5 - 2) - 1 \neq 5 - (2 - 1)$

A associatividade é uma propriedade de operadores binários.

$$\forall x, y, z \quad (x \Delta y) \Delta z = x \Delta (y \Delta z)$$

Mas nem todos os operadores binários são associativos.

- e.g. subtracção não é associativa $(5 - 2) - 1 \neq 5 - (2 - 1)$

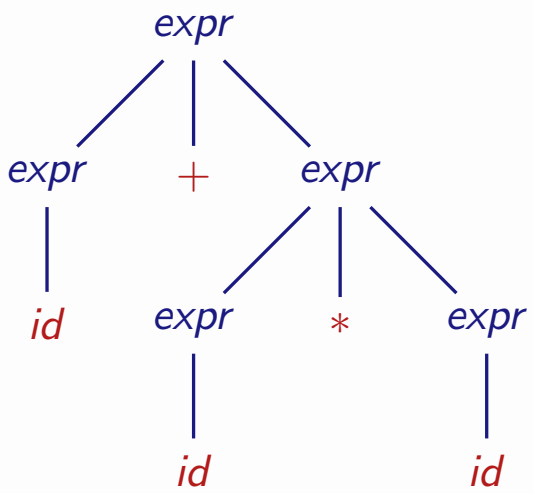
Solução: usar um símbolo diferente à esquerda e à direita do operador

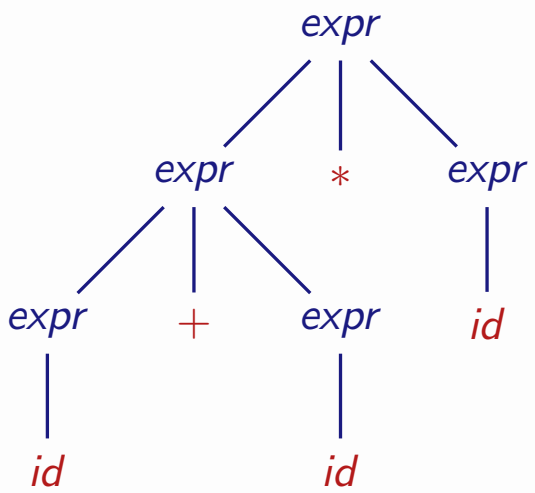
$$\begin{array}{l} E \rightarrow E \ - \ E \\ E \rightarrow E \ * \ E \\ E \rightarrow id \\ E \rightarrow (E) \end{array} \quad \Rightarrow \quad \begin{array}{l} E \rightarrow E \ - \ F \\ E \rightarrow E \ * \ F \\ F \rightarrow id \\ F \rightarrow (E) \end{array}$$

Gramáticas livres de contexto

Propriedades - Precedência

Muitas linguagens têm regras de precedência entre operadores.
Exemplo (árvores sintácticas anteriores):

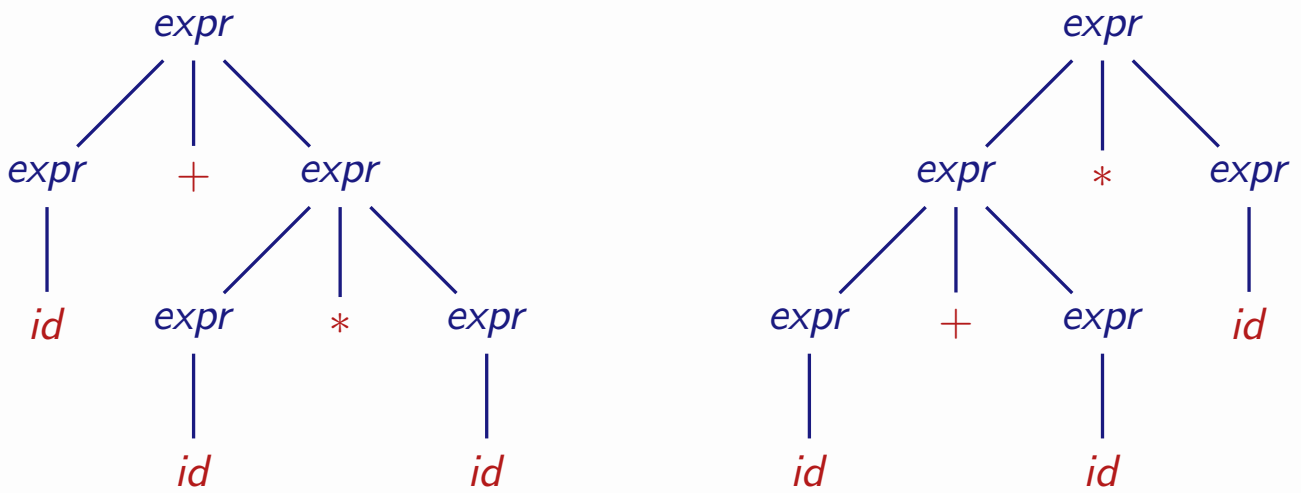




Gramáticas livres de contexto

Propriedades - Precedência

Muitas linguagens têm regras de precedência entre operadores.
Exemplo (árvores sintácticas anteriores):



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

As precedências podem ser ditadas pela estrutura das regras! afectando a profundidade a que os operadores ficam na árvore.
Quanto maior a precedência, mais profundos ficam!

Gramáticas livres de contexto

Recursão à esquerda

Uma gramática diz-se recursiva à esquerda se tem pelo menos um símbolo não terminal A com uma derivação contendo A como o símbolo mais à esquerda.

- e.g., $E \rightarrow E + T$

Uma gramática diz-se recursiva à direita se tem pelo menos um símbolo não terminal A com uma derivação contendo A como o símbolo mais à direita.

- e.g., $E \rightarrow T + E$

Outline

Conjuntos de análise

Os algoritmos de análise sintáctica usam conjuntos de símbolos terminais que ajudam a determinar as regras da gramática.

Conjuntos de análise

Os algoritmos de análise sintáctica usam conjuntos de símbolos terminais que ajudam a determinar as regras da gramática.

- **FIRST** - conjunto de símbolos terminais que iniciam sequências derivadas a partir de X

Conjuntos de análise

Os algoritmos de análise sintáctica usam conjuntos de símbolos terminais que ajudam a determinar as regras da gramática.

- **FIRST** - conjunto de símbolos terminais que iniciam sequências derivadas a partir de X
- **FOLLOW** - conjunto de símbolos terminais a que podem aparecer imediatamente a seguir a X

Conjuntos de análise

Os algoritmos de análise sintáctica usam conjuntos de símbolos terminais que ajudam a determinar as regras da gramática.

- **FIRST** - conjunto de símbolos terminais que iniciam sequências derivadas a partir de X
- **FOLLOW** - conjunto de símbolos terminais a que podem aparecer imediatamente a seguir a X
- **LOOKAHEAD** - conjunto de antevisão

FIRST(X) - conjunto de símbolos terminais que iniciam sequências derivadas a partir de X

Algoritmo:

- se α é um símbolo terminal, então $FIRST(\alpha) = \{\alpha\}$
- se X é um símbolo não terminal:
  - se $X \to \epsilon$ é uma produção, então $\epsilon \in FIRST(X)$
  - se $X \to Y_1 \dots Y_n$ é uma produção, então:
    - se $\epsilon \notin FIRST(Y_1)$, $FIRST(Y_1) \subseteq FIRST(X)$
    - se $\epsilon \in FIRST(Y_1)$, $FIRST(Y_1) \setminus \{\epsilon\} \cup FIRST(Y_2 \dots Y_n) \subseteq FIRST(X)$
    - se $\epsilon \in FIRST(Y_1) \wedge \dots \wedge \epsilon \in FIRST(Y_n)$, $\epsilon \in FIRST(X)$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \mathrm{id}
\end{aligned}
$$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \mathsf{id}
\end{aligned}
$$

- $\forall_{\alpha \in \Sigma}, \mathsf{FIRST}(\alpha) = \{\alpha\}$, $\Sigma = \{(,), +, *, \mathsf{id}\}$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathsf{id}
\end{array}
$$

- $\forall_{\alpha \in \Sigma}, \mathsf{FIRST}(\alpha) = \{\alpha\}$, $\Sigma = \{(,), +, *, \mathsf{id}\}$
- $\mathsf{FIRST}(F) = \{(, \mathsf{id}\}$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathsf{id}
\end{array}
$$

- $\forall_{\alpha \in \Sigma}, \mathsf{FIRST}(\alpha) = \{\alpha\}$, $\Sigma = \{(,), +, *, \mathsf{id}\}$
- $\mathsf{FIRST}(F) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(T) = \{(, \mathsf{id}\}$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathsf{id}
\end{array}
$$

- $\forall_{\alpha \in \Sigma}, \mathsf{FIRST}(\alpha) = \{\alpha\}$, $\Sigma = \{(,), +, *, \mathsf{id}\}$
- $\mathsf{FIRST}(F) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(T) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(E) = \{(, \mathsf{id}\}$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathsf{id}
\end{array}
$$

- $\forall_{\alpha \in \Sigma}, \mathsf{FIRST}(\alpha) = \{\alpha\}$, $\Sigma = \{(,), +, *, \mathsf{id}\}$
- $\mathsf{FIRST}(F) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(T) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(E) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(E') = \{+, \epsilon\}$

Conjuntos de análise

FIRST - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \mathsf{id}
\end{aligned}
$$

- $\forall_{\alpha \in \Sigma}, \mathsf{FIRST}(\alpha) = \{\alpha\}$, $\Sigma = \{(,), +, *, \mathsf{id}\}$
- $\mathsf{FIRST}(F) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(T) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(E) = \{(, \mathsf{id}\}$
- $\mathsf{FIRST}(E') = \{+, \epsilon\}$
- $\mathsf{FIRST}(T') = \{*, \epsilon\}$

FOLLOW - conjunto de símbolos terminais *a* que podem aparecer imediatamente a seguir a X. Calculado apenas para símbolos não terminais.

Algoritmo:

- se X é o símbolo inicial, então $\$ \in \mathit{FOLLOW}(X)$
- se $A \to \alpha X \beta$ é uma produção, então $\mathit{FIRST}(\beta) \setminus \{\epsilon\} \subseteq \mathit{FOLLOW}(X)$
- se $A \to \alpha X$ ou $A \to \alpha X \beta$ ($\beta \stackrel{*}{\Rightarrow} \epsilon$), então $\mathit{FOLLOW}(A) \subseteq \mathit{FOLLOW}(X)$
- Repetir até que os conjuntos FOLLOW não sejam alterados

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathrm{id}
\end{array}
$$

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathrm{id}
\end{array}
$$

- FOLLOW(E) = {\$}

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathrm{id}
\end{array}
$$

- $\mathrm{FOLLOW}(E) = \{\$\}$
- $\mathrm{FOLLOW}(T) = \mathrm{FIRST}(E') \setminus \{\epsilon\} = \{+\}$

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \text{id}
\end{aligned}
$$

- $\text{FOLLOW}(E) = \{\$\}$
- $\text{FOLLOW}(T) = \text{FIRST}(E') \setminus \{\epsilon\} = \{+\}$
- $\text{FOLLOW}(E) = \text{FOLLOW}(E) \cup \text{FIRST}()) = \{), \$\}$

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \text{id}
\end{aligned}
$$

- $\text{FOLLOW}(E) = \{\$\}$
- $\text{FOLLOW}(T) = \text{FIRST}(E') \setminus \{\epsilon\} = \{+\}$
- $\text{FOLLOW}(E) = \text{FOLLOW}(E) \cup \text{FIRST}()) = \{), \$\}$
- $\text{FOLLOW}(E') = \text{FOLLOW}(E) = \{), \$\}$

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{array}{rcl}
E & \rightarrow & T\ E' \\
E' & \rightarrow & +\ T\ E' \mid \epsilon \\
T & \rightarrow & F\ T' \\
T' & \rightarrow & *\ F\ T' \mid \epsilon \\
F & \rightarrow & (\ E\ ) \mid \mathrm{id}
\end{array}
$$

- $\mathrm{FOLLOW}(E) = \{\$\}$
- $\mathrm{FOLLOW}(T) = \mathrm{FIRST}(E') \setminus \{\epsilon\} = \{+\}$
- $\mathrm{FOLLOW}(E) = \mathrm{FOLLOW}(E) \cup \mathrm{FIRST}()) = \{), \$\}$
- $\mathrm{FOLLOW}(E') = \mathrm{FOLLOW}(E) = \{), \$\}$
- $\mathrm{FOLLOW}(T) = \mathrm{FIRST}(E') \setminus \{\epsilon\} \cup \mathrm{FOLLOW}(E) \cup \mathrm{FOLLOW}(E') = \{+,), \$\}$

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \text{id}
\end{aligned}
$$

- $\text{FOLLOW}(E) = \{\$\}$
- $\text{FOLLOW}(T) = \text{FIRST}(E') \setminus \{\epsilon\} = \{+\}$
- $\text{FOLLOW}(E) = \text{FOLLOW}(E) \cup \text{FIRST}()) = \{), \$\}$
- $\text{FOLLOW}(E') = \text{FOLLOW}(E) = \{), \$\}$
- $\text{FOLLOW}(T) = \text{FIRST}(E') \setminus \{\epsilon\} \cup \text{FOLLOW}(E) \cup \text{FOLLOW}(E') = \{+,), \$\}$
- $\text{FOLLOW}(T') = \text{FOLLOW}(T) = \{+,), \$\}$

Conjuntos de análise

FOLLOW - Exemplo

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \text{id}
\end{aligned}
$$

- $\text{FOLLOW}(E) = \{\$\}$
- $\text{FOLLOW}(T) = \text{FIRST}(E') \setminus \{\epsilon\} = \{+\}$
- $\text{FOLLOW}(E) = \text{FOLLOW}(E) \cup \text{FIRST}()) = \{), \$\}$
- $\text{FOLLOW}(E') = \text{FOLLOW}(E) = \{), \$\}$
- $\text{FOLLOW}(T) = \text{FIRST}(E') \setminus \{\epsilon\} \cup \text{FOLLOW}(E) \cup \text{FOLLOW}(E') = \{+,), \$\}$
- $\text{FOLLOW}(T') = \text{FOLLOW}(T) = \{+,), \$\}$
- $\text{FOLLOW}(F) = \text{FIRST}(T') \setminus \{\epsilon\} \cup \text{FOLLOW}(T) \cup \text{FOLLOW}(T') = \{*, +,), \$\}$

LOOKAHEAD - conjunto de antevisão

Dada uma regra $A \rightarrow \alpha$, o $LOOKAHEAD(A \rightarrow \alpha)$ é:

- $FIRST(\alpha)$, se $\alpha \stackrel{*}{\not\Rightarrow} \epsilon$
- $FIRST(\alpha) \bigcup FOLLOW(A)$, se $\alpha \stackrel{*}{\Rightarrow} \epsilon$

Outline

Algoritmos de parsing

- **Objectivo**: decidir se $w \in L(G)$?

Algoritmos de parsing

- **Objectivo**: decidir se $w \in L(G)$?

- Universais
  - Algoritmo de Earley
  - Algoritmo de Cocke-Younger-Kasami (CYK)
    - Gramática representada em CNF (Chomsky Normal Form)
  - Ineficientes, não utilizados em compiladores

Algoritmos de parsing

- **Objectivo**: decidir se $w \in L(G)$?

- Universais
  - Algoritmo de Earley
  - Algoritmo de Cocke-Younger-Kasami (CYK)
    - Gramática representada em CNF (Chomsky Normal Form)
  - Ineficientes, não utilizados em compiladores
- Específicos para gramáticas LL(k)/LR(k)
  - Eficientes

Algoritmos de parsing

Gramáticas livres de contexto divididas em:

- **LL(k)** parsing
  - Top-down - análise descendente
  - Considera sempre derivações mais à esquerda

Algoritmos de parsing

Gramáticas livres de contexto divididas em:

- **LL(k)** parsing
  - Top-down - análise descendente
  - Considera sempre derivações mais à esquerda
- **LR(k)** parsing
  - Bottom-up - análise ascendente
  - Considera sempre derivações mais à direita

Algoritmos de parsing

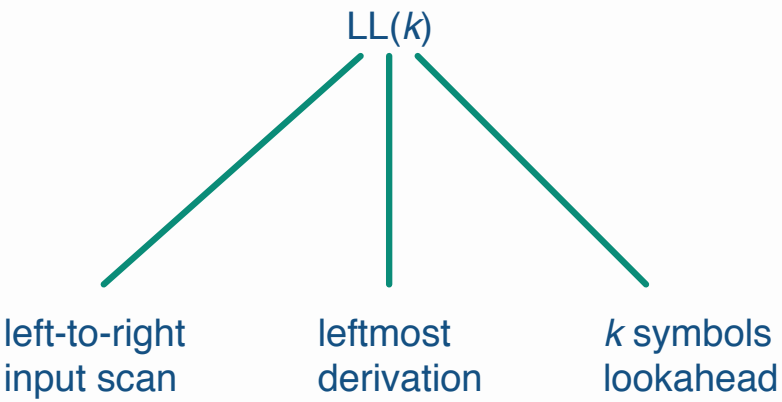
Gramáticas livres de contexto divididas em:

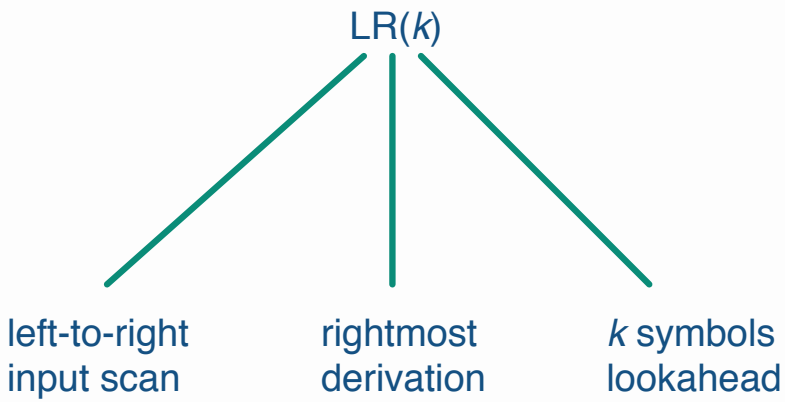
- **LL(k)** parsing
  - Top-down - análise descendente
  - Considera sempre derivações mais à esquerda
- **LR(k)** parsing
  - Bottom-up - análise ascendente
  - Considera sempre derivações mais à direita

k - indica o número de símbolos de LOOKAHEAD

LL vs LR parsing

- Parsers LL vs. parsers LR:





Questões?

Dúvidas?