Análise sintática

- Verificar se um código fonte é bem-formado

Ex.:
```
for (i = 1; i <= n, i++) {
    soma = soma + i;
    mult = mult * i;
}
```
bem-formado

Ex.:
```
for i = 1, i <= n; i++
```
mal-formado

A sintaxe de linguagem de programação
* pode ser especificada através de uma gramática livre-de-contexto

Notação: BNF (Backus-Naur Form)

- $A \to aA \mid \xi \iff A \to a^*$?
- $A \to aA \mid b \iff A \to a^*b$ ~~a^*b~~ $a^*b \mid b$
- $A \to aA \mid a \iff A \to a^+$

O papel do analisador sintático

- Obter uma sequência de tokens do analisador léxico e verificar se tal sequência pode ser gerada pela gramática
- Reportar erros sintáticos (o principais) | recuperar erros comuns e continuar o processamento do entrada.

Importante: * Pode construir uma árvore de derivação para códigos bem-formados.
* Não é necessário a construção da árvore de derivação.

Tipos de analisadores sintáticos

* Universal
  - algoritmo de Cocke-Younger-Kasami
  - algoritmo de Earley
  - por serem métodos gerais (servem para qualquer gramática) são ineficientes
* Top-down simula (ou constrói) uma árvore de derivação de cima para baixo (da raiz para as folhas)
* Bottom-up simula (ou constrói) " de baixo para cima (das folhas para a raiz)

Realizar análise semântica durante análise sintática? Sim

* Coletar tipos de identificadores
* Verificar de tipos

Gramática livre de contexto

* Consiste de terminais, não terminais, um símbolo inicial (não terminal) e produções.

Exemplo de produção: $stmt \rightarrow$ if (expr) stmt else stmt

Terminais: { if, (,), else } => saída tokens (devolvidos pelo analisador léxico)

Não-terminais { stmt, expr } => são símbolos que derivam para algo

Derivação: $E \rightarrow E+E \mid E*E \mid -E \mid (E) \mid id$

O não terminal E significa expressão.

sentença: $-(id)$ é derivada pela gramática

$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(id)$

$E \overset{*}{\Rightarrow} -(id)$

$E \overset{+}{\Rightarrow} -(id)$

$-(id+id)$ é sentença?

$E \underset{lm}{\Rightarrow} -E \underset{lm}{\Rightarrow} -(E) \underset{lm}{\Rightarrow} -(E+E) \underset{lm}{\Rightarrow} -(id+E) \underset{lm}{\Rightarrow} -(id+id)$

Definições: $S \overset{*}{\Rightarrow} \alpha$, S é símbolo inicial de uma gramática, dizemos que α é forma sentencial da gramática.

Se α não possui símbolos não terminais dizemos que α é sentença.

A linguagem gerada por uma gramática é o seu conjunto de sentenças

Derivação mais à esquerda: o não-terminal mais à esquerda de uma forma sentencial é escolhido para ser substituído. Notação $\underset{lm}{\Rightarrow}$

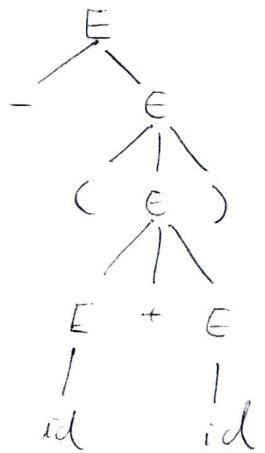
Derivação mais à direita: " " direita

" $\underset{rm}{\Rightarrow}$

Árvore de derivação: é uma representação gráfica de uma derivação

$|-(id + id)|$

$E \Rightarrow -E$
$\Rightarrow -(E)$
$\Rightarrow -(E + E)$
$\Rightarrow -(id + E)$
$\Rightarrow -(id + id)$



Importante: Existe um mapeamento um-para-um entre árvores de derivação e derivações mais à esquerda (ou mais à direita)

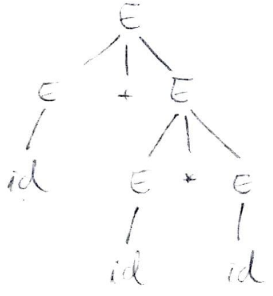
Ambiguidade

Uma gramática é <u>ambígua</u> se ela produz mais que uma árvore de derivação para alguma sentença.

Ex.1 $E \to E+E \mid E*E \mid (E) \mid id$

sentença: $id + id * id$

$$E \underset{lm}{\Rightarrow} E+E \underset{lm}{\Rightarrow} id+E \underset{lm}{\Rightarrow} id+E*E \underset{lm}{\Rightarrow} id+id*E \underset{lm}{\Rightarrow} id+id*id$$



$$E \underset{lm}{\Rightarrow} E*E \underset{lm}{\Rightarrow} E+E*E \underset{lm}{\Rightarrow} id+E*E \underset{lm}{\Rightarrow} id+id*E \underset{lm}{\Rightarrow} id+id*id$$