

Aula 3 - Descrevendo a Sintaxe e a Semântica

Introdução:

- Descrição compreensível, de uma Linguagem de Programação é difícil e essencial;
- Capacidade de determinar como as expressões, instruções e unidades são formadas e suas intenções de efeito quando executadas;
- Pode ser dividida em:
 - Sintaxe -> forma
 - Semântica -> significado

Sintaxe:

- Linguagem:
 - Conjunto de cadeias de caracteres de algum alfabeto.
- As cadeias são chamadas sentenças ou instruções;
- Regras sintáticas de uma Linguagem especificam quais cadeias do alfabeto pertencem a Linguagem;
- Lexema:
 - Identificadores, constantes, operadores e palavras especiais.
- Token:
 - Categoria de lexemas: identificadores, operação de adição, ...

Exemplo:

Resp = 2 * cont + 17;

Lexemas	Tokens
---------	--------

Resp	identificador
------	---------------

=	operador de atribuição
---	------------------------

2	constante numérica inteira
---	----------------------------

*	operador de adição
---	--------------------

17	constante numérica inteira
----	----------------------------

;	ponto e vírgula
---	-----------------

Reconhecedores:

Suponha:

- Linguagem L sobre um alfabeto Σ

Para definir L usando o método de reconhecimento é preciso construir um mecanismo R de tal forma que quando uma cadeia for fornecida para este mecanismo, este diga se esta cadeia pertence ou não a L.

É como um filtro que separa as sentenças corretas das erradas;

Reconhecimento não é usado para enumerar todas as sentenças de uma Linguagem;

A Análise Sintática de um compilador é um reconhecedor, determinando apenas se um programa esta na Linguagem ou não.

Geradores

Dispositivo usado para gerar sentenças de uma Linguagem;

É como um botão que, quando pressionado, produz uma sentença da Linguagem.

Forma de Backus-Naur (BNF)

Popularmente conhecida com BNF;

É um método formal para descrição da sintaxe;

Origem:

- Algol 58 e Algol 60.

É uma notação natural de descrever a sintaxe.

Uma metalinguagem é uma Linguagem usada para descrever outra Linguagem.

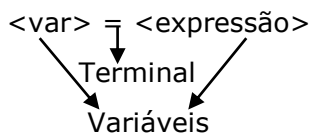
BNF é uma metalinguagem para Linguagens de Programação.

BNF utiliza abstrações para representar estruturas sintáticas.

As abstrações são geralmente chamadas de símbolos não-terminais (variáveis).

Os lexemas e tokens são chamados símbolos terminais.

As definições são chamadas de regras.



Uma Gramática é uma coleção de regras.

Símbolos não-terminais podem ter mais de uma definição:

<if_stmt> **if** (<expressão_lógica>) **then** <comando>
| **if**(<expressão_lógica>) **then** <comando> **else** <comando>

Listas de tamanhos variáveis:

<ident_Lista> → <identificador>
| <identificador>, <ident_Lista>

BNF é um dispositivo de geração de Linguagens.

Inicia com um conjunto não-terminal especial:

- start Symbol

Esta geração de sentença é chamada de derivação.

O Start Symbol representa um programa completo.

Exemplo 1:

$\langle \text{program} \rangle \rightarrow \text{Begin } \langle \text{ident_list} \rangle \text{ end}$
 $\langle \text{ident_list} \rangle \rightarrow \langle \text{ident} \rangle ; \mid \langle \text{ident} \rangle ; \langle \text{ident_list} \rangle$
 $\langle \text{ident} \rangle \rightarrow \langle \text{var} \rangle = \langle \text{expressão} \rangle$
 $\langle \text{var} \rangle \rightarrow \mathbf{A} \mid \mathbf{B} \mid \mathbf{C}$
 $\langle \text{expressão} \rangle \rightarrow \langle \text{var} \rangle + \langle \text{var} \rangle \mid \langle \text{var} \rangle - \langle \text{var} \rangle \mid \langle \text{var} \rangle$

Uma derivação de um Programa nesta Linguagem:

$\langle \text{program} \rangle$
 $\text{begin } \langle \text{ident_list} \rangle \text{ end}$
 $\text{begin } \langle \text{ident} \rangle ; \langle \text{ident_list} \rangle \text{ end}$
 $\text{begin } \langle \text{var} \rangle = \langle \text{expressão} \rangle ; \langle \text{ident_list} \rangle \text{ end}$
 $\text{begin } \mathbf{A} = \langle \text{expressão} \rangle ; \langle \text{ident_list} \rangle \text{ end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \langle \text{var} \rangle ; \langle \text{ident_list} \rangle \text{ end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \mathbf{C}; \langle \text{ident_list} \rangle \text{ end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \mathbf{C}; \langle \text{ident} \rangle ; \text{end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \mathbf{C}; \langle \text{var} \rangle = \langle \text{expressão} \rangle ; \text{end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \mathbf{C}; \mathbf{B} = \langle \text{expressão} \rangle ; \text{end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \mathbf{C}; \mathbf{B} = \langle \text{var} \rangle ; \text{end}$
 $\text{begin } \mathbf{A} = \mathbf{B} + \mathbf{C}; \mathbf{B} = \mathbf{C}; \text{end}$

Exemplo 2:

$\langle \text{atribua} \rangle \rightarrow \langle \text{id} \rangle = \langle \text{expr} \rangle ;$
 $\langle \text{id} \rangle \rightarrow \mathbf{A} \mid \mathbf{B} \mid \mathbf{C}$
 $\langle \text{expr} \rangle \rightarrow \langle \text{id} \rangle + \langle \text{expr} \rangle \mid \langle \text{id} \rangle * \langle \text{expr} \rangle \mid (\langle \text{expr} \rangle) \mid \langle \text{id} \rangle$

Derivação a Extrema Esquerda e a Extrema Direita:

Derivação a Extrema Esquerda (DEE):

$\mathbf{A} = \mathbf{B} * (\mathbf{A} + \mathbf{C});$
 $\langle \text{atribua} \rangle$
 $\langle \text{id} \rangle = \langle \text{expr} \rangle ;$
 $\mathbf{A} = \langle \text{expr} \rangle ;$
 $\mathbf{A} = \langle \text{id} \rangle * \langle \text{expr} \rangle ;$
 $\mathbf{A} = \mathbf{B} * \langle \text{expr} \rangle ;$
 $\mathbf{A} = \mathbf{B} * (\langle \text{expr} \rangle);$
 $\mathbf{A} = \mathbf{B} * (\langle \text{id} \rangle + \langle \text{expr} \rangle);$
 $\mathbf{A} = \mathbf{B} * (\mathbf{A} + \langle \text{expr} \rangle);$
 $\mathbf{A} = \mathbf{B} * (\mathbf{A} + \langle \text{id} \rangle);$
 $\mathbf{A} = \mathbf{B} * (\mathbf{A} + \mathbf{C});$

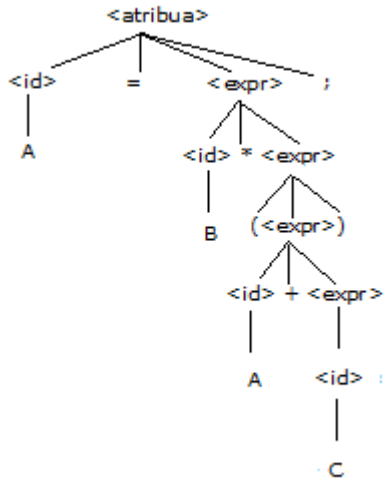
Derivação a Extrema Direita (DED):

$\mathbf{A} = \mathbf{B} * (\mathbf{A} + \mathbf{C});$
 $\langle \text{atribua} \rangle$
 $\langle \text{id} \rangle = \langle \text{expr} \rangle ;$
 $\langle \text{id} \rangle = \langle \text{id} \rangle * \langle \text{expr} \rangle ;$
 $\langle \text{id} \rangle = \langle \text{id} \rangle * (\langle \text{expr} \rangle);$
 $\langle \text{id} \rangle = \langle \text{id} \rangle * (\langle \text{id} \rangle + \langle \text{expr} \rangle);$
 $\langle \text{id} \rangle = \langle \text{id} \rangle * (\langle \text{id} \rangle + \langle \text{id} \rangle);$
 $\langle \text{id} \rangle = \langle \text{id} \rangle * (\langle \text{id} \rangle + \mathbf{C});$
 $\langle \text{id} \rangle = \langle \text{id} \rangle * (\mathbf{A} + \mathbf{C});$
 $\langle \text{id} \rangle = \mathbf{B} * (\mathbf{A} + \mathbf{C});$
 $\mathbf{A} = \mathbf{B} * (\mathbf{A} + \mathbf{C});$

Árvores de Derivação

Gramáticas descrevem uma estrutura hierárquica das sentenças.

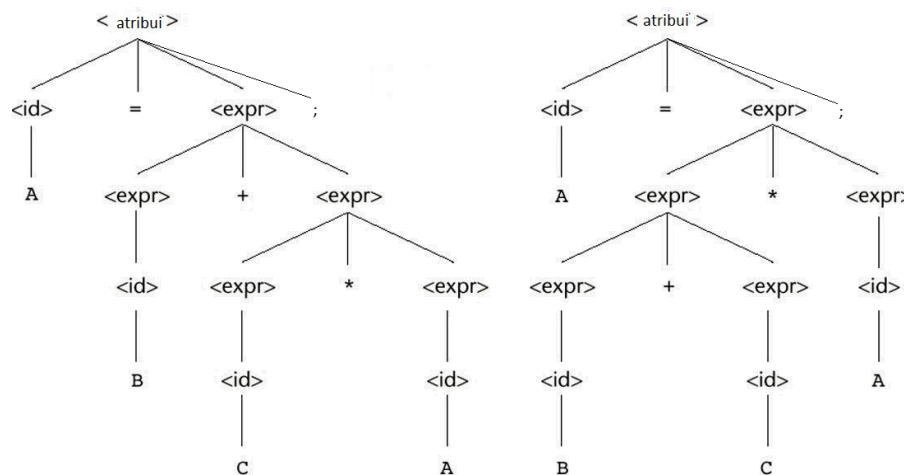
Essa estrutura hierárquica é chamada de *parse tree* (árvore de derivação).



Uma Gramática que gera uma sentença para a qual existem duas ou mais árvores de derivação é chamada de ambígua.

Exemplo:

$A = B + C * A$



Exercícios:

1. [Sebesta, 2000] Usando a Gramática a seguir, mostre a derivação e a árvore de Derivação à Extrema Esquerda para as seguintes instruções:

$\langle \text{atribuição} \rangle \rightarrow \langle \text{id} \rangle := \langle \text{expr} \rangle$

$\langle \text{id} \rangle \rightarrow \mathbf{A} \mid \mathbf{B} \mid \mathbf{C}$

$\langle \text{expr} \rangle \rightarrow \langle \text{id} \rangle + \langle \text{expr} \rangle \mid \langle \text{id} \rangle * \langle \text{expr} \rangle \mid (\langle \text{expr} \rangle) \mid \langle \text{id} \rangle$

a) $A := A * (B + (C * A))$

b) $B := C * (A * C + B)$