# Sintaxe e Semântica

Paradigmas de Programação – BCC/UFRPE
Lucas Albertins – lucas.albertins@deinfo.ufrpe.br

## Agenda

- Introdução
- Descrevendo Sintaxe
- Análise léxica
- Gramáticas
- Descrevendo Semânticas
- Semântica Operacional
- Semântica Denotacional
- Semântica Axiomática

## Introdução

*Como descrever/definir uma linguagem de programação?*

## Introdução

+ Definição das características das linguagens
  + Sintaxe
  + Semântica

## Sintaxe

+ Descreve a **FORMA** ou estrutura da linguagem
+ Como **arranjar** / **organizar** / **ordenar** expressões, comandos, declarações e outras construções
+ Como os programas são escritos pelo programador, lidos por outros programadores e quebrados/divididos pelo computador.

## Semântica

+ Descreve o **SIGNIFICADO** da linguagem
+ Como deve ser o comportamento do programa especificado
+ Como os programas são compostos pelo programador, entendidos por outros programadores e interpretados pelo computador

## Exemplo Sintaxe e Semântica

- Comando **While**
- Sintaxe:
  - while (<boolean_expr>) <statement>
- Semântica:
  - Enquanto o valor de boolean_expr for verdadeiro, execute statement
- Sintaxe e semântica relacionadas: semântica deriva da sintaxe

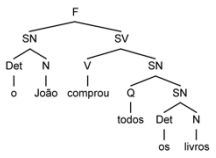
## Descrevendo Sintaxe

## Descrevendo Sintaxe

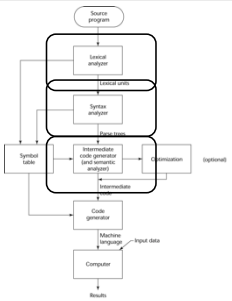
- Linguagens são compostas por strings de caracteres (palavras): sentenças ou expressões
  - Português, Inglês, Java
- Linguagens de programação são muito mais simples do que as naturais
  - Menos palavras, combinações, expressões

## Exemplo Português

- Uma frase (F) é formada por Sintagma Nominal (SN) e sintagma verbal (SV).
- Sintagma Nominal possui determinante (Det) e Nome, ou quantificador (Q) e outro sintagma nominal
- Sintagma Verbal possui um verbo (V) e um sintagma nominal




## Etapas da Compilação




## Descrevendo a Sintaxe

- 1. Leitura do programa e armazenamento dos lexemas - Análise Léxica
- 2. Definição da gramática

## Descrevendo a Sintaxe

- **1. Leitura do programa e armazenamento dos lexemas - Análise Léxica**
- 2. Definição da gramática

## Análise Léxica

- Lexema: unidade sintática de menor nível de uma linguagem
- Token: Categoria do lexema
- Armazenamento na
- tabela de símbolos ->
- Exemplo:

```
index = 2 * count + 17;
```

| Lexema | Token |
|---|---|
| index | identificador |
| = | igualdade |
| 2 | int_literal |
| * | mult_op |
| count | identificador |
| + | plus_op |
| 17 | int_literal |
| ; | semicolon |



## Descrevendo a Sintaxe

- 1. Leitura do programa e armazenamento dos lexemas - Análise Léxica
- **2. Definição da gramática**

## Definição de Gramática

- Gramáticas Livres de Contexto
  - Desenvolvidas por Noam Chomsky (metade dos anos 1950)
  - Feitas para descrever a sintaxe de linguagens naturais
  - Define uma classe de linguagens chamadas linguagens livre de contexto
- Backus-Naur Form – BNF (1959)
  - Usada para descrever a sintaxe de Algol 58
  - BNF é equivalente a gramáticas livre de contexto
  - Forma:
    - Lado esquerdo: abstração sendo definida
    - Lado direito: significado

## Exemplo de gramática BNF

```
<program> → <stmts>
<stmts> → <stmt> | <stmt> ; <stmts>
<stmt> → <var> = <expr>
<var> → a | b | c | d
<expr> → <term> + <term> | <term> - <term>
<term> → <var> | const
...
```

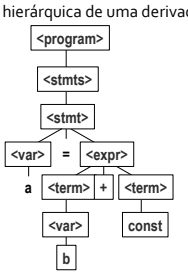
## Exemplo de Derivação

Derivação é a aplicação de regras começando com o símbolo inicial e terminando com uma sentença com todos os símbolos terminais

```
<program> => <stmts> => <stmt>
          => <var> = <expr>
          => a = <expr>
          => a = <term> + <term>
          => a = <var> + <term>
          => a = b + <term>
          => a = b + const
```

## Árvore de derivação

- Uma representação hierárquica de uma derivação



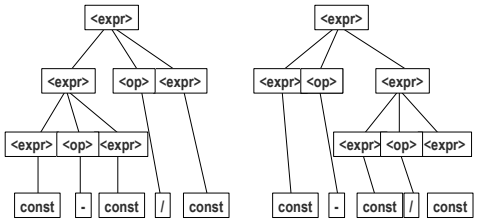

## Exercício

- Crie a árvore de derivação das seguintes expressões
  - `c = a - d`
  - `a = c; b = d`
  - `d = a + c; b = b - d`

## Ambiguidade em Gramáticas

- Uma gramática é ambígua se, e somente se, ela gera uma forma sentencial que tem duas ou mais árvores de derivação

```
<expr> → <expr> <op> <expr>  |  const
<op> → /  |  -
```




## Outros operadores de BNFs

- Partes opcionais entre colchetes
- `<proc_call> -> ident [(<expr_list>)]`
- Repetições entre chaves
- `<ident> → letter {letter|digit}`

## Descrevendo semântica

## Semântica

- Descreve o **SIGNIFICADO** da linguagem
- Como deve ser o comportamento do programa especificado
- Como os programas são **compostos** pelo programador, **entendidos** por outros programadores e **interpretados** pelo computador

## Semântica

- Verificar a consistência em relação à definição da linguagem
- Usa a árvore sintática e tabela de símbolos
  - Acrescenta e testa informações
- Checagem de tipos
  - Cada operador tem operandos compatíveis
- Verificação de erros, escopo, ambiguidade

## Semântica das Linguagens

- Sem notação ou formalismo extensamente aceito
- Necessidades de uma metodologia/notação
  - Programadores precisam saber significados dos comandos
  - Quem programa o compilador precisa saber o que os construtores das linguagens fazem
  - Provas de corretude
  - Geração de compiladores
  - Projetistas poderiam detectar ambiguidades e inconsistências
- Três tipos de Semântica
  - Operacional,
  - Denotacional e
  - Axiomática

## Semântica Operacional

- Mais utilizada
- Descreve ***Como*** ocorre a execução do comando
- A semântica é o que ocorre na máquina de estados do programa quando uma instrução é realizada
- Mudança de estado é definida por algoritmos codificados

## Semântica Operacional

- O comando for em C

| *C Statement* | *Meaning* |
|---|---|
| `for (expr1; expr2; expr3) {` | `expr1;` |
| `...` | loop: `if expr2 == 0 goto out` |
| `}` | `...` |
| | `expr3;` |
| | `goto loop` |
| | out: `...` |

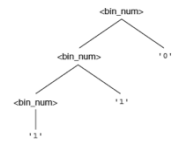
## Semântica Denotacional

- Descreve **O Efeito** da execução do comando
- Mudanças de estado definidas por rigorosas funções matemáticas
- Cada entidade da linguagem define:
  - objetos matemáticos
  - funções que mapeiam entidade -> objeto matemático

## Semântica Denotacional

- Dada uma sintaxe

```
<bin_num> → '0'
          | '1'
          | <bin_num> '0'
          | <bin_num> '1'
```




- Funções Semânticas para mapeamento em inteiros:

$M_{bin}$('0') = 0
$M_{bin}$('1') = 1
$M_{bin}$(<bin_num> '0') = 2 * $M_{bin}$(<bin_num>)
$M_{bin}$(<bin_num> '1') = 2 * $M_{bin}$(<bin_num>) + 1

## Exemplo de Mapeamento

$M_{bin}$('0') = 0
$M_{bin}$('1') = 1
$M_{bin}$(<bin_num> '0') = 2 * $M_{bin}$(<bin_num>)
$M_{bin}$(<bin_num> '1') = 2 * $M_{bin}$(<bin_num>) + 1

Mbin(10) = Mbin ( <1> '0' )
2 * Mbin(1)
2 * 1 = 2
Mbin(11) = Mbin ( <1> '1' )
2 * Mbin(1) + 1
2 * 1 + 1 = 3
Mbin(100) = Mbin( <10> '0')
2 * (Mbin(10))
2 * (2) = 4

## Semântica Denotacional

- Podem ser utilizadas para provar corretude de programas
- Fornecem uma forma rigorosa de pensar sobre programas
- Podem ajudar no projeto de linguagens
- Tem sido utilizada em sistemas geradores de compiladores
- Devido a sua complexidade, tem pouca utilidade para usuários da linguagem

## Semântica Denotacional vs Operactional

- Na semântica operacional, as mudanças de estado são definidas por algoritmos codificados
- Na semântica denotacional, as mudanças de estado são definidas por rigorosas funções matemáticas

## Semântica Axiomática

- **Quais proposições lógicas são válidas para um programa?**
- Desenvolvida como um método para verificação formal de programas (provas)
- Baseado em lógica matemática formal (cálculo de predicado)
- A prova consiste que se ela puder ser construída, o programa é compatível com sua especificação

## Semântica Axiomática

- **Quais proposições lógicas são válidas para um programa?**
- Pensamento semelhante a TDD: Test Driven Development
- Os testes são escritos antes do programa
- A corretude do programa é baseada na satisfação dos testes

## Semântica Axiomática

- Uso de ***asserções***: são sentenças da lógica definidas sobre os valores das variáveis do programa
- Por exemplo, em {P}Q{R}, P e R são asserções.
  - P é a *pré-condição*: uma sentença que é verdadeira antes da execução do comando Q
  - R é a *pós-condição*: uma sentença que é verdadeira após a execução do comando Q.

ex.: `{x = 4} x+1 {x = 5}`

## Exemplo: Axioma para Seleção

- Uma regra de inferência para seleção
  - **`if`** B **`then`** S1 **`else`** S2

$$\frac{\{B \text{ and } P\}\ S1\ \{Q\},\ \{(\text{not } B) \text{ and } P\}\ S2\ \{Q\}}{\{P\}\ \texttt{if}\ B\ \texttt{then}\ S1\ \texttt{else}\ S2\ \{Q\}}$$

## Semântica Axiomática

- Para usar a semântica axiomática numa LP, dever ser definida uma regra de inferência para cada tipo de instrução da linguagem
- Desenvolvimento de axiomas ou regras de inferência para todas as instruções de uma linguagem é uma tarefa difícil.
- Útil para pesquisar provas de exatidão, mas pouco útil para desenvolvedores da linguagem e desenvolvedores de compiladores

## Leitura Adicional

- Capítulo 3 – Descrevendo sintaxe e semântica. SEBESTA, R. W. Conceitos de Linguagens de Programação. 10ª ED. BOOKMAN, 2012.
- Próxima aula
  - Capítulo 5 – Nomes, Vinculações e Escopo. SEBESTA, R. W. Conceitos de Linguagens de Programação. 10ª ED. BOOKMAN, 2012