

Linguagens de Programação

André Moraes

Conteúdos:

Sintaxe e Semântica

Sintaxe

- ▶ Sintaxe é a forma de suas expressões, suas instruções e de suas unidades de programa
 - ▶ É muito mais fácil de se descrever do que a semântica
 - ▶ Existe uma notação concisa e universalmente aceita para a sintaxe
 - ▶ A semântica não possui notações deste gênero.

Sintaxe

- ▶ **Problemas de descrever a sintaxe**
 - ▶ Diversas pessoas devem entender a descrição
 - ▶ Linguagens possuem muitos dialetos diferentes
 - ▶ Toda a linguagem deve permitir sua implementação através de um manual de referência

Descrevendo a sintaxe

- ▶ Linguagens funcionam como conjuntos de sequências de caracteres de algum alfabeto.
- ▶ Estas sequências são denominadas **sentenças** ou **instruções**
 - ▶ Na linguagem existem regras de sintaxe que especificam quais sentenças de caracteres do alfabeto da linguagem estão nela.

Descrevendo a sintaxe

- ▶ A sintaxe possui dois elementos que a descrevem de maneira mais específica:
 - ▶ Lexemas
 - ▶ é o conjunto literal de identificadores, operadores e palavras especiais
 - ▶ Token (símbolo)
 - ▶ Consistem em categorias de lexemas, que podem, em alguns casos, consistir em apenas um único significado

Descrevendo a sintaxe

▶ Exemplo

- ▶ Seja a expressão:
 - ▶ `Index = 2 * cont + 17;`
- ▶ Quais são os lexemas e os símbolos desta expressão?

LEXEMAS	TOKENS
Index	identificador
=	sinal_igual
2	int_literal
*	op_multiplic
cont	identificador
+	Op_soma
17	Int_literal
;	Ponto_e_virgula

Métodos formais de descrever sintaxe

- ▶ Forma de backus-aur
- ▶ Gramática livre de contexto
- ▶ Gramáticas e derivações

Gramática Livre de Contexto

- ▶ São bastante usadas para descrever a maioria das linguagens de programação
 - ▶ A sintaxe da maioria das linguagens é produzida utilizando esta gramática
 - ▶ O nome Livre de contexto, indica que esta é definida sem uma definição exata das palavras que compõem a linguagem
 - ▶ É determinada uma regra geral e as palavras da linguagem são produzidas em função desta regra
 - ▶ Desenvolvida pelo Linguista Chomsky na década de 50

Descrição de Backus-aur

- ▶ **Apresentado por John Backus, em 1959**
 - ▶ Introduziu uma nova notação formal para se descrever a sintaxe das linguagens de programação
 - ▶ Modificada em 1960 por Peter Naur
 - ▶ Esta descrição ficou conhecida mais tarde como forma de Backus-Naur ou BNF.
- ▶ **Tornou-se o método mais natural para se descrever a sintaxe de linguagens de programação**

Gramática de Backus Naur

- ▶ A BNF é conhecida como uma metaliguagem
 - ▶ Ou seja, é uma linguagem para usada para descrever outra linguagem
 - ▶ Utiliza abstrações para estruturas sintáticas
 - ▶ Exemplo:
 - ▶ $\langle \text{atribuição} \rangle \rightarrow \langle \text{var} \rangle = \langle \text{expressão} \rangle$
 - Aqui estamos definindo que a expressão $\langle \text{atribuição} \rangle$ terá obrigatoriamente que ser formado com uma abstração $\langle \text{var} \rangle$, seguida pelo lexema $=$ e seguida pela abstração $\langle \text{expressão} \rangle$
 - ▶ O que poderia resultar em:
 - Total = A + B



Gramática de Backus Naur

- ▶ A expressão anterior é chamada de **regra ou produção**
 - ▶ Os elementos **<var>** e **<expressão>** também são considerados abstrações
 - ▶ O símbolo do lado esquerdo é chamado de **lado esquerdo (LE)** e contém a abstração a ser definida
 - ▶ O texto na direita da seta é considerado a definição do **LE**, sendo chamado de **lado direito**, e sempre irá conter símbolos, lexemas e referências a outras abstrações
- ▶ Analisando as operações na gramática, temos ainda os símbolos chamados de terminais e não-terminais

Símbolos Não-Terminais

- ▶ São as abstrações existentes na gramática
 - ▶ Exemplo:
 - ▶ <atribuição>
- ▶ Podem ter duas ou mais definições distintas
 - ▶ Para representar duas ou mais formas sintáticas possíveis na linguagem
 - ▶ Várias definições podem ser escritas na mesma regra, separadas pelo símbolo | (significando ou)
 - ▶ Exemplo:

```
<inst_if> → if <expr_logica> then <inst>  
          | if <expr_logica> then <inst> else <inst>
```

```
<inst_if> → if <expr_logica> then <inst>  
<inst_if> → if <expr_logica> then <inst> else <inst>
```

Símbolos Terminais

- ▶ São os símbolos comuns da gramática
 - ▶ Os lexemas
 - ▶ Os símbolos de regras
- ▶ Exemplos:
 - ▶ -
 - ▶ +
 - ▶ =
 - ▶ Index
 - ▶ Cont
 - ▶ 55

Gramáticas e Derivações

- ▶ A BNF é um dispositivo para gerar linguagens
 - ▶ As sentenças são produzidas através de uma sequência da aplicação das regras
 - ▶ Inicia com um símbolo não-terminal denominado símbolo de início
 - ▶ A geração de uma sentença denomina-se **derivação**
 - ▶ A derivação é o resultado de executar a linguagem obedecendo-se a todas as regras, o que dependendo da gramática, poderá gerar várias sentenças diferentes
 - ▶ O símbolo de início representa um programa completo e usualmente é chamado de <programa>

Gramática - Exemplo 1

```
<programa> → begin <lista_instr> end  
<lista_instr> → <inst>  
                | <inst> ; <lista_instr>  
<inst> → <var> = <expressão>  
<var> → A | B | C  
<expressão> → <var> + <var>  
                | <var> - <var>  
                | <var>
```

- ▶ A gramática deste exemplo possui apenas uma forma de instrução: a atribuição
 - Um programa consiste na palavra inicial **Begin**,
 - seguida de uma listagem de instruções separadas por ponto e vírgula,
 - seguida da palavra especial **End**
 - Uma expressão é uma variável única ou duas variáveis separadas pelo operador + ou pelo –
 - Os únicos nomes de variáveis desta linguagem são A, B e C.

Gramática – Exemplo 1

- ▶ Agora seja um exemplo de derivação do exemplo anterior:

```
<programa> => begin <list_inst> end
=> begin <inst>; <list_inst> end
=> begin A = <expressão>; <list_inst> end
=> begin A = <var> + <var>; <list_inst> end
=> begin A = B + <var>; <list_inst> end
=> begin A = B + C; <list_inst> end
=> begin A = B + C; <inst> end
=> begin A = B + C; <var> := <expressão> end
=> begin A = B + C; B = <expressão> end
=> begin A = B + C; B = <var> end
=> begin A = B + C; B = C end
```

Gramática – Exemplo 1 (comentários)

- ▶ A derivação anterior começa com o símbolo de iniciar <programa>
 - ▶ O símbolo = > é lido como “deriva”
 - ▶ Cada cadeia sucessiva é derivada da cadeia anterior substituindo um dos não-terminais por uma das definições do mesmo
 - ▶ Cada cadeia de derivação incluindo <programa> é chamado de **forma sentencial**.
 - O não-terminal substituído é o da extrema esquerda na forma sentencial anterior
 - Derivações desta ordem recebem o nome de **Derivações à Extrema Esquerda**

Gramática – Exemplo 1 (comentários)

- ▶ A derivação será executada até a não-existência de símbolos não-terminais
 - Note que a forma sentencial gerada é composta apenas de lexemas

```
=> begin A = B + C; B = C end
```

Próximos Conteúdos:

- ▶ Árvore de análise (parse trees)
- ▶ Associatividade de operadores
- ▶ BNF extendida