Capítulo II – Gramáticas

II.1 – Motivação

- **O que é uma Gramática?**
  - **Um sistema gerador de linguagens;**
  - **Um sistema de reescrita;**
  - **Uma maneira finita de representar uma linguagem;**
  - **Um dispositivo formal usado para especificar de maneira finita e precisa uma linguagem potencialmente infinita.**

- **Qual a finalidade de uma gramática?**
  - **Definir o subconjunto de V* que forma uma determinada linguagem.**

- **Exemplo intuitivo de uma Gramática**

  (um subconjunto da gramática da língua portuguesa)

<sentença> :: = <sujeito> <verbo> <objeto>
<sujeito> :: = <substantivo>
| <artigo> <substantivo>
| <artigo> <adjetivo> <substantivo>
<substantivo> :: = compilador | tradutor | codigo
| interpretador | programa | resultado
<artigo> :: = o | um
<adjetivo> :: = eficiente | bom | mau | melhor | pior
<verbo> :: = gera | traduz | compila | interpreta
<objeto> :: = <substantivo>
| <artigo> <substantivo>
| <artigo> <adjetivo> <substantivo>
| ε (* vazio *)

II.2 – Definição Formal de Gramática

$$G = (Vn, Vt, P, S)$$

onde:

Vn – conjunto finito de símbolos **não-terminais**.

Vt – conjunto finito de símbolos **terminais**.

convenções: $\mathbf{Vn} \cap \mathbf{Vt} = \varphi$ e $\mathbf{Vn} \cup \mathbf{Vt} = \mathbf{V}$

P – conjunto finito de pares (α, β) denominados **produções** (ou regras gramaticais ou de sintaxe).

$$P = \{ \alpha ::= \beta \mid \alpha \in V^{*}VnV^{*} \wedge \beta \in V^{*} \}$$

S – símbolo $\in$ Vn, é o **símbolo inicial** da gramática.

Exemplo: Formalizando o subconjunto da gramática da língua portuguesa apresentado, teríamos:

Gportugues = (Vn, Vt, P, S), onde:

Vn = { <sentença> , <sujeito> , <substantivo>, <artigo>, <adjetivo>, <verbo>, <objeto>}

Vt = { compilador, tradutor, código, interpretador | programa, resultado, o, um, eficiente, bom , mau, melhor, pior, gera, traduz, compila, interpreta }

P = é o conjunto das regras gramaticais apresentado

S = <sentença>

Exemplos e exercícios:

1 - Construa gramáticas que representem as seguintes linguagens:

a) Inteiros positivos menores que 1000
b) Inteiros positivos
c) Inteiros positivos pares
d) Números reais

2 – Dado VT = { a, b }, construa uma gramática cuja linguagem gerada seja:

a) O conjunto de sentenças cujo último símbolo seja igual o primeiro
b) O conjunto de sentenças de tamanho ímpar
c) O conjunto de sentenças com número par de b´s
d) O conjunto de sentenças com #a´s divisível por 3
e) O conjunto de sentenças que não possuam a's consecutivos
f) O conjunto de sentenças onde o #a's seja igual o #b's e todos os a's precedam todos os b's
g) O conjunto de sentenças onde o #a's = #b's

Notação a ser utilizada neste curso:

::= - $\rightarrow$

V_N – Letras de **"A" a "T"** e palavras escritas com letras maiúsculas

V_T – Letras de **"a" a "t"**, palavras escritas com letras minúsculas, dígitos e caracteres especiais

V_T^* - **u, v, x, y, w, z**

$\{V_N \cup V_T\}$ – **U, V, X, Y, W, Z**

$\{V_N \cup V_T\}^*$ - $\alpha, \beta, \gamma, \delta, \ldots, \omega$ (exceto ε)

II.3 – Derivação e Redução

Definição: São operações de substituição que formalizam o uso de gramáticas

Seja $G = (Vn, Vt, P, S)$ uma gramática $\forall \wedge$ Seja $\delta\alpha\gamma \in (Vn \cup Vt)^*$

- **Derivação em um passo (ou direta):**

$$\delta\alpha\gamma \Rightarrow \delta\beta\gamma \iff \alpha \rightarrow \beta \in P$$

- **Derivação em zero ou mais passos**

$$\alpha \overset{*}{\Rightarrow} \beta \iff$$
$$\alpha \Rightarrow \alpha 1 \Rightarrow \alpha 2 \Rightarrow \ldots \Rightarrow \alpha n = \beta;\ p/\ n \geq 0$$

obs. Se $\alpha \overset{*}{\Rightarrow} \beta$ em 0 (zero) passos, então $\alpha = \beta$.

- **Derivação em um ou mais passos**

$$\alpha \overset{+}{\Rightarrow} \beta \iff$$
$$\alpha \Rightarrow \alpha 1 \Rightarrow \alpha 2 \Rightarrow \ldots \Rightarrow \alpha n = \beta;\ p/\ n \geq 1$$

- **Redução em um passo (ou direta)**

$$\delta\alpha\gamma \Leftarrow \delta\beta\gamma$$

- **Redução em zero ou mais passos**

$$\delta\alpha\gamma \overset{*}{\Leftarrow} \delta\beta\gamma$$

- **Redução em um ou mais passos**

$$\delta\alpha\gamma \overset{+}{\Leftarrow} \delta\beta\gamma$$

II.4 - Sentença, Forma Sentencial e Linguagem

Seja G = (Vn, Vt, P, S) :

Sentença – seqüência de terminais produzida a partir do símbolo inicial de G

$$x \mid x \in Vt^* \wedge S \overset{+}{\Rightarrow} x$$

Forma Sentencial – seqüência de terminais e/ou não-terminais produzida a partir do símbolo inicial de G

$$\text{Se } S \Rightarrow \alpha \Rightarrow \beta \Rightarrow \ldots \Rightarrow \gamma \Rightarrow \ldots$$

então $\alpha, \beta, \ldots, \gamma, \ldots$ são formas sentenciais de G.

$$\alpha \mid \alpha \in V^* \wedge S \overset{*}{\Rightarrow} \alpha$$

Linguagem – conjunto de sentenças derivadas a partir do símbolo inicial de G

$$L(G) = \{ x \mid x \in Vt^* \wedge S \overset{+}{\Rightarrow} x \}$$

Gramáticas Equivalentes – Duas gramáticas **G1** e **G2** são equivalentes entre si, se e somente se L(G1) = L(G2).

$$G1 \equiv G2 \Leftrightarrow L(G1) = L(G2)$$

II.5 – Tipos de Gramáticas

(Classificação ou hierarquia de CHOMSKY)

Gramática Tipo 0:

(ou gramática sem restrições)

G = (Vn, Vt, P, S), onde:

$$P = \{\alpha \rightarrow \beta \mid \alpha \in V^*VnV^* \wedge \beta \in V^*\}$$

Gramática Tipo 1:

(ou Gramática Sensível ao Contexto – G.S.C.)

G = (Vn, Vt, P, S), onde:

$$P = \{\alpha \rightarrow \beta \mid |\alpha| \leq |\beta|, \alpha \in V^*VnV^* \wedge \beta \in V^+\}$$

Gramática Tipo 2:

(ou Gramática Livre de Contexto – G.L.C.)

G = (Vn, Vt, P, S), onde:

$$P = \{A \rightarrow \beta \mid A \in Vn \wedge \beta \in V^+\}$$

Gramática Tipo 3:

(ou Gramática Regular – G.R.)

G = (Vn, Vt, P, S), onde:

$$P = \{A \rightarrow a\,X \mid A \in Vn, a \in Vt \wedge X \in \{Vn \cup \{\varepsilon\}\}\}$$

Observação: As linguagens representadas por G.S.C., G.L.C. e G.R. são denominadas, respectivamente, Linguagens Sensíveis ao Contexto (L.S.C.) Linguagens Livres de Contexto (L.L.C.) e Linguagens Regulares (L.R.).

II.6 –Sentença Vazia

Considerações:

1 - A motivação para o estudo de gramáticas foi à necessidade de se encontrar representações finitas para as linguagens.

2 - Se uma linguagem L possui uma descrição finita, então L1 = L ∪ {ε}, também deveria possuir descrição finita.

3 - Pela definição dada, as G.S.C., G.L.C. e G.R. não aceitam produções da forma S → ε - logo, a sentença vazia (ε) não pode pertencer as L.S.C., L.L.C. ou L.R..

Redefinição de G.S.C. G.L.C. e G.R.

G.S.C., G.L.C. e G.R., podem ter a produção S → ε , desde que:

> **1 – S seja o símbolo inicial da gramática;**
> **2 – S não apareça no lado direito de nenhuma produção da gramática em questão.**

Observação: segundo esta redefinição, a produção S → ε só poderá ser usada na derivação de ε (a sentença vazia).

Teorema II.1: "Se L é SC, LC ou REGULAR, então $L_1 = L \cup \{\varepsilon\}$ e $L_2 = L - \{\varepsilon\}$ serão do mesmo tipo".

Lema II.1: **"Se $G = (Vn, Vt, P, S)$ é uma GSC, então $\exists$ G_1 SC | $L(G_1) = L(G) \wedge$ o símbolo inicial de G_1 não apareça no lado direito de nenhuma produção de G_1".**

Prova: Seja $G = (Vn, Vt, P, S)$ uma G.S.C.;

Seja $G_1 = (Vn \cup \{S_1\}, Vt, P1, S1)$, onde:

$P_1 = \{S_1 \rightarrow \alpha \mid S_1 \rightarrow \alpha \ \exists \leftrightarrow S \rightarrow \alpha \in P\} \cup P$ e

S_1 é o símbolo inicial de G_1.

Para completar a prova, precisamos mostrar que $L(G_1) = L(G)$:

$$L(G_1) = L(G) \Leftrightarrow \begin{array}{l} 1 - L(G) \subseteq L(G_1) \\ 2 - L(G_1) \subseteq L(G) \end{array}$$

1 – Seja $w \in L(G) \therefore S \stackrel{+}{\Rightarrow} w$

se $S \stackrel{+}{\Rightarrow} w$

então $S \Rightarrow \alpha \stackrel{*}{\Rightarrow} w$

se $S \Rightarrow \alpha$

então $S \rightarrow \alpha \in P$ e, por definição, $S_1 \rightarrow \alpha \in P_1$

Logo, $S_1 \Rightarrow \alpha \stackrel{*}{\Rightarrow} w \therefore S_1 \stackrel{+}{\Rightarrow} w \therefore L(G) \subseteq L(G_1)$

2 – Seja $w \in L(G_1) \therefore S_1 \stackrel{+}{\Rightarrow} w$

se $S_1 \stackrel{+}{\Rightarrow} w$

então $S_1 \Rightarrow \alpha \stackrel{*}{\Rightarrow} w$

se $S_1 \Rightarrow \alpha$

então $S_1 \rightarrow \alpha \in P_1$ e, para que esta produção exista, P contem a produção $S \rightarrow \alpha$,

portanto:

$S \Rightarrow \alpha \stackrel{*}{\Rightarrow} w \therefore S \stackrel{+}{\Rightarrow} w \therefore L(G_1) \subseteq L(G)$

Conclusão: De 1 e 2, temos que $L(G_1) = L(G)$.

II.7 – Recursividade das G.S.C.

Definição: Uma gramática G é ***recursiva*** se existe um ***algoritmo*** que determine, para qualquer sequência w, se w ***é ou não gerada*** por G.

Teorema II.2: "Se G = (Vn, Vt, P, S) é uma G.S.C., então G é RECURSIVA".

Prova: (* através de algoritmo *)

Seja G = (Vn, Vt, P, S) uma G.S.C. | $S \rightarrow \varepsilon \notin P$;

Seja w uma seqüência $\forall$ $|w| = n$;

Seja T_M o conjunto de formas sentenciais α |

$|\alpha| \leq N \wedge S \rightarrow \alpha$ em M passos.

Algoritmo II.1:

1 – $T_0 = \{ S \}$

2 – $M \leftarrow 1$

3 – $T_M \leftarrow T_{M-1} \cup \{\alpha | \beta \rightarrow \alpha$, onde $\beta \in T_{M-1} \wedge |\alpha| \leq n\}$

4 – se $T_M \neq T_{M-1}$

então $M \leftarrow M+1$;

calcule novo T_M (volte ao passo 3).

Senão se $w \in T_M$

então $w \in L(G)$

senão $w \notin L(G)$

II.8 – Lema do Bombeamento para Linguagens Regulares - "Pumping Lemma"

Objetivo: demonstrar que algumas Linguagens não são Regulares.

Lema do Bombeamento : Se L é uma LR, então existe uma constante **$n \geq 1$** | para todo **$w \in L$, $|w| \geq n$** , podemos escrever **w** como **x y z** onde:

- **$|xy| \leq n$**
- **$y \neq \varepsilon$**
- **$xy^iz \in L$ para qualquer $i \geq 0$**

Idéia geral: O "**Lema do Bombeamento**", ou "Pumping Lemma", nos diz que qualquer sentença **w** de uma **linguagem regular** pode ser decomposta em três partes: **w = xyz**, de maneira que a repetição (o *bombeamento*) de **y**, qualquer número de vezes, resulta em sentenças **xy^iz** que também pertencem à linguagem

Para mostrar que uma linguagem **não é regular**, basta encontrar uma sentença w qualquer pertencente à linguagem, que não satisfaça o lema do bombeamento – isto é, não possa ser decomposta em xyz de forma que seja possível *bombear* **y** e **xy^iz** continue na linguagem.

Exemplos:

II.9 – Lema do Bombeamento para Linguagens Livres de Contexto - "Pumping Lemma”

Objetivo: demonstrar que algumas Linguagens não são Livres de Contexto.

Lema do Bombeamento : Se L é uma LLC, então existe uma constante **$n \geq 1$** | para todo **$w \in L$, $|w| \geq n$** , podemos escrever **w** como **u v x y z** onde:

- **$| v y | \geq 0$**
- **$| v x y | \leq n$**
- **$u v^i x y^i z \in L$ para qualquer $i \geq 0$**

Idéia geral: O "**Lema do Bombeamento**", ou "Pumping Lemma", nos diz que qualquer sentença **w** de uma **LLC** pode ser decomposta em cinco partes: **w = u v x y z**, de maneira que, respeitadas as duas primeiras condições, o bombeamento (repetição) de **v** e **y**, qualquer número de vezes, resulta em sentenças **$u\ v^i\ x\ y^i\ z$** que também pertencem à linguagem.

Para mostrar que uma Linguagem **não é** Livre de Contexto, basta encontrar uma sentença w qualquer pertencente à linguagem, que não satisfaça o lema do bombeamento – ou seja, não possa ser decomposta em u v x y z de forma que seja possível bombear **v e y** e **$u\ v^i\ x\ y^i\ z$** continue na linguagem.

Exemplos: