UNIVERSIDADE DE SÃO PAULO

Introdução a Gramáticas e Linguagens

versão 1.0

SOLANGE REZENDE RODRIGUES

Nº 10

NOTAS DIDÁTICAS

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2585

Introdução a Gramáticas e Linguagens
versão 1.0

SOLANGE REZENDE RODRIGUES

Nº 10

NOTAS DIDÁTICAS DO ICMSC

São Carlos
JUN. / 1993

Introdução a Gramáticas e Linguagens

Solange Rezende Rodrigues
Universidade de São Paulo / ILTC
Instituto de Ciências Matemáticas de São Carlos
Departamento de Ciências de Computação e Estatística

Versão 1.0

Maio 1993

Conteúdo

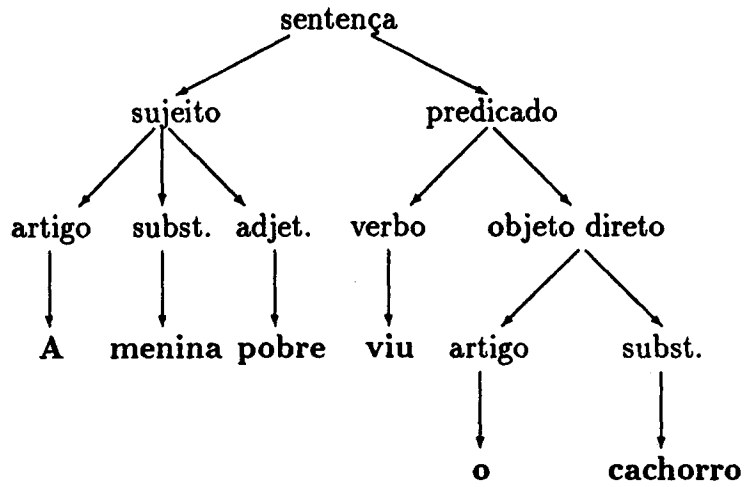
1 Introdução

A noção de gramática teve origem no estudo de linguagem natural. Os linguistas tentavam definir precisamente quais eram as sentenças válidas destas linguagens e encontrar formas estruturais de representá-las.

Mas uma linguagem natural é muito extensa e complexa, por isso não se conseguiu uma maneira de definí-la totalmente.

Por exemplo, observe a análise da sentença:

A menina pobre viu o cachorro



Com esta análise verifica-se que a sentença está correta, mas não é possível fazer isso com todas as sentenças da linguagem.

No caso de linguagem de programação isto é possível pois estas são mais simples e a representação segue a mesma idéia de análise sintática da linguagem natural.

O objetivo inicial dos pesquisadores ainda não foi atingido, ou seja não existe uma "gramática formal" definida para o inglês, ou qualquer linguagem natural (LN), mas os resultados dessa linha de pesquisa foram muito úteis na descrição de linguagem de computação — a linguagem **Algol-60**, por exemplo, foi descrita usando uma gramática livre de contexto na notação BNF (Backus Naur Form).

A idéia funcionou para Linguagens de Programação, apesar de não ter funcionado para LNs porque as primeiras podem ser definidas através de uma sintaxe rígida e uma semântica bem determinada, de forma a habilitar o tratamento computacional.

> Qual é a maneira sistemática de especificar a sintaxe de uma Linguagem de Programação?

Para isto é necessário:

- Um método que especifique meios de construir programas sintaticamente corretos (geração).
- Um método para analisar um programa e verificar se ele está sintaticamente correto (reconhecimento).

Este trabalho está organizado da seguinte forma: na seção 2 são apresentados conceitos básicos e a definição de gramáticas. Na seção 3 os tipos de gramáticas seguindo a hierarquia de Chomsky, as linguagens correspondentes e propriedades das gramáticas regulares envolvendo o processo de construção de uma gramática regular através de operações de união, concatenação e fechamento. Na seção 4 são apresentados alguns aspectos relacionados com ambiguidade nas linguagens de programação. A seção 5 é dedicada a precedência e associatividade de operadores e a seção 6 aos conjuntos e expressões regulares. Finalmente na seção 7 é apresentada a conclusão.

2 Noções de Gramáticas

Definição 1 - Alfabeto ou Vocabulário: *é um conjunto finito de símbolos.*

Ex.: $\{a, b\}$, $\{1, 2, 3\}$

Definição 2 - Cadeia: *qualquer concatenação de símbolos de um dado alfabeto.*

Ex.: aab, bab, λ - cadeia nula.

Definição 3 - Comprimento de Cadeia: *é o nº de símbolos de uma dada cadeia.*

Ex.: $|aab| = 3 \quad |\lambda| = 0$

Definição 4 - Concatenação de 2 ou mais Cadeias:

Ex.: $x = abaa$, $y = ba$, $z = \lambda$

então: $xy = abaaba$, $yz = zy = y = ba$,
$yx = baabaa$

$|x| = 4, \; |y| = 2, \; |xy| = |x| + |y| = 6$

Definição 5 - Produto de 2 alfabetos:

$V1 = \{a, b\}$

$\Rightarrow V1.V2 = V1 \times V2 = \{a1, a2, a3, b1, b2, b3\}$

$V2 = \{1, 2, 3\}$

portanto $V1.V2 \neq V2.V1$

Definição 6 - Exponenciação de Alfabetos:

$$V^0 = \{\lambda\}$$

$$V^1 = V$$

$$V^n = V^{n-1}.V \quad , \quad n > 1$$

Ex.: $V = \{0,1\}$
$V^3 = V^2.V = (V.V).V = \{00,01,10,11\}.\{0,1\}$
$= \{000,001,010,011,100,101,110,111\}$

V^n é o conjunto de todas cadeias de símbolos de V, de comprimento n.

Definição 7 - Fechamento (Clausura) de um Alfabeto

Seja A um alfabeto, então o fechamento de A é definido como

$A^* = A^0 \cup A^1 \cup A^2 \cup \ldots \cup A^n \cup \ldots$

Portanto A^* = conjunto das cadeias de qualquer comprimento sobre o alfabeto A.

Ex.: $A = \{1\}$
$A^* = \{\lambda, 1, 11, 111, \ldots\}$

Fechamento Positivo de A: $A^+ = A^* - \{\lambda\}$

Portanto $\forall$ que seja V, $V^* - V^+ = \{\lambda\}$

Definição 8 - Gramática

Uma gramática formal G é a quádrupla

$G = \{V_N, V_T, P, S\}$ onde:

- $V_T \rightarrow$ é um alfabeto finito conhecido como **vocabulário terminal**. Os símbolos de V_T são aqueles que aparecem nos programas de uma linguagem de programação ou as palavras da L. Portuguesa, por exemplo.
- $V_N \rightarrow$ é um alfabeto finito, conhecido como **vocabulário não-terminal**. Os símbolos de V_N são usados como nomes para categorias sintáticas.

  (Exemplos de categoria sintática:
  em Português: Sentença, Predicados, ..., Verbo;
  em Pascal, Programa, Bloco, Procedure)

- $P \rightarrow$ é um conjunto finito de **regras** ou **produções** da forma $W \rightarrow W'$ onde:

  1. $W \in (V_N \cup V_T)^+$; isto é o conjunto de cadeias não vazias sobre V_N e
  2. $W' \in (V_N \cup V_T)^*$.

  A interpretação de regra $W \rightarrow W'$ é que W pode ser substituído por W' sempre que W aparecer.

  Ex.: em Português:
  Sent_interrogativa → Sentença ?
  Sentença → Sn Sv Verb Predicado

- $S \rightarrow \in V_N$ e é chamado o **axioma** ou **símbolo inicial** de G. S é o nome da categoria sintática principal.

  Ex.: em Português: S = Sentença
  em Pascal: S = Programa

  Obs.:

  1. $\rightarrow V_N \cap V_T = \emptyset$; $V_N \cup V_T = V$
  2. $\rightarrow$ os elementos de V_T, são os terminais e por convenção são representados por letras minúsculas $a, b, c, d, \ldots$
  3. $\rightarrow$ os elementos de V_N são os não-terminais e representados por letras maíusculas $A, B, C, D, \ldots$
  4. $\rightarrow$ as **cadeias mistas**, ou seja as cadeias que $\in (V_N \cup V_T)^*$ são representadas por letras gregas $(\alpha, \beta, \gamma, \delta, \ldots)$

Ex.: 1 $G1 = (\underbrace{\{S, A, B\}}_{V_N}, \underbrace{\{a, b\}}_{V_T}, \underbrace{P}_{P}, \underbrace{S}_{S})$

onde: $P : \{$ 1) $S \rightarrow AB$
2) $A \rightarrow a$
3) $B \rightarrow b\}$

Portanto $S \xrightarrow{1)} AB \xrightarrow{2)} aB \xrightarrow{3)} \underline{ab}$
(Única cadeia gerada por esta gramática).

Ex.: 2 V_N = { Sentença,Sn,Sv,Artigo,Verbo,Substantivo,Complemento }
V_T = { peixe,isca,mordeu,o,a }
P = { 1) Sentença → Sn Sv
2) Sn → Artigo Substantivo
3) Sv → Verbo Complemento
4) Complemento → Artigo,Substantivo
5) Artigo → o
6) Artigo → a
7) Substantivo → peixe
8) Substantivo → isca
9) Verbo → mordeu }
S = Sentença

Definição 9 *Outra maneira de representar produções: BNF-(Forma Normal de Backus), que definiu o* **Algol 60.**

Neste caso, → *é substituído por* ::= *os não terminais são palavras ladeadas por* < >.

BNF é usada para definir gramáticas com a seguinte característica: o lado esquerdo de cada regra é composto por **um único símbolo não-terminal.**

No caso de repetições de lado esquerdo:

<A> ::= α_1
<A> ::= α_2
: :
<A> ::= α_n

escreve-se: $A ::= \alpha_1|\alpha_2|\ldots|\alpha_n$

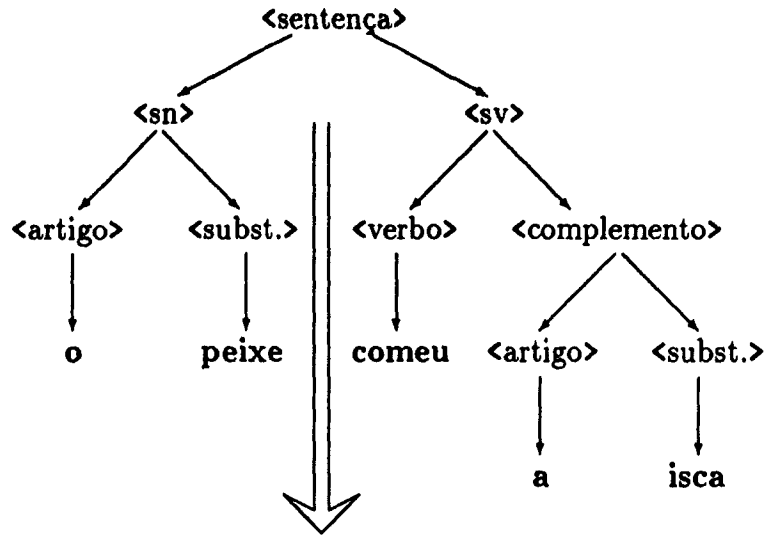
Os símbolos <, >, :, =, | *formam a* **meta-linguagem**, *ou seja, são símbolos que não fazem parte da linguagem mas ajudam a descrevê-la.*

Ex.: 3 - G = { V_N, V_T, P, S} onde:

V_N = { <sentença>,<sujeito>, <predicado>,<artigo>,
<substantivo>,<verbo>, <complemento> }
V_T = { o,a,peixe,comeu,isca }
S = <sentença>
P = {

1. <sentença> ::= <sn> <sv>
2. <sn> ::= <artigo> <substantivo>
3. <sv> ::= <verbo> <complemento>
4. <complemento> ::= <artigo> <substantivo>
5. <artigo> ::= o|a
6. <verbo> ::= comeu
7. <substantivo> ::= peixe|isca }

Gerar uma sentença sintaticamente correta:



árvore de derivação sintática

Com esta gramática, pode-se gerar as seguintes frases sintaticamente corretas:

1. o peixe comeu a isca
2. a isca comeu o peixe
3. o peixe comeu o peixe
4. a isca comeu a isca
   etc. ...

Para as definições a seguir considere $G = (V_N, V_T, P, S)$ e $\alpha, \beta, \gamma, \delta \in (V_N \cup V_T)^*$

Definição 10 *Uma cadeia* $\alpha\gamma\beta$ **gera diretamente** $(\rightarrow)$ *uma cadeia* $\alpha\delta\beta$ *sse*

$$(\gamma \rightarrow \delta) \in P$$

$$\begin{array}{lccccccccccc} & \alpha & \gamma & \beta & & \alpha & \delta & \beta & & \gamma & & \delta \\ & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow & & & & \\ \textbf{No Ex.: 1} & a & B & & \rightarrow & a & b & & \text{pois} & B & \rightarrow & b \end{array}$$

$$\begin{array}{ccccccc} & & S & & & \overbrace{\quad\quad} & \\ & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ & \alpha & \gamma & \beta & \rightarrow & \alpha & \delta & \beta \end{array}$$

Definição 11 *Uma cadeia* α *gera* $(\stackrel{*}{\rightarrow})$ *uma cadeia* β *sse* $\exists\ \gamma_1, \gamma_2 \ldots \gamma_n$ *tal que* $\alpha \rightarrow \gamma_1 \rightarrow \gamma_2 \rightarrow \ldots \rightarrow \gamma_n \rightarrow \beta \quad n \geq 0.$

$$n = 0 \Rightarrow \alpha \equiv \beta \text{ portanto } \alpha \stackrel{*}{\rightarrow} \alpha \; para \forall \alpha.$$

No Ex.: 1 -

$S \xrightarrow{*} ab$
$S \xrightarrow{*} aB$
$AB \xrightarrow{*} ab$
$ab \xrightarrow{*} ab$ (zero passos)

No Ex.: 3 -

<sentença> $\xrightarrow{*}$ o peixe mordeu a <substantivo>
<complemento> $\xrightarrow{*}$ o <substantivo>

Definição 12 *Uma cadeia* $\alpha \in (V_N \cup V_T)^*$ *é uma* **forma sentencial** *de* G *sse* $S \xrightarrow{*} \alpha$ *ou seja,* α *é um embrião para uma cadeia gerada pela gramática.*

No Ex.: 1 -

- aB, AB, S, ab são formas sentenciais;
- já $bA \in (V_N \cup V_T)^*$ mas não é forma sentencial.

Definição 13 *Uma forma sentencial,* α*, é uma* **sentença** *de* G *sse* $\alpha \in V_T^*$
Portanto as cadeias geradas pela gramática são as sentenças de G*.*

Ex.: 4 - $G_1 =$ $(\{A, B\}, \{a, b, c\}, P, A)$ onde

$P = \{$ 1) $A \rightarrow aB$
2) $B \rightarrow bB$
3) $B \rightarrow c\}$

$x = abbbc$ é uma sentença de $G1$? **sim**

$A \xrightarrow{1} aB \xrightarrow{2} abB \xrightarrow{2} abbB \xrightarrow{2} abbbB \xrightarrow{3} abbbc$

Exercício: Verifique se $x = ac$ e $y = aabc$ são sentenças de $G1$.
O conjunto de sentenças geradas por $G1$ pode ser expresso como $\{ab^n c, n \geq 0\}$?

Definição 14 $L(G) = \{\alpha \mid \alpha$ *é sentença de* $G\}$ *é a* **linguagem gerada** *por* G*, ou seja,* $L(G) = \{\alpha \in V_T^* \mid S \xrightarrow{*} \alpha\}$

Ex.: $L(G1) = \{ab^n c \ , \ n \geq 0\}$

Definição 15 *Duas gramáticas* $G1$ *e* $G2$ *são equivalentes sse* $L(G1) = L(G2)$

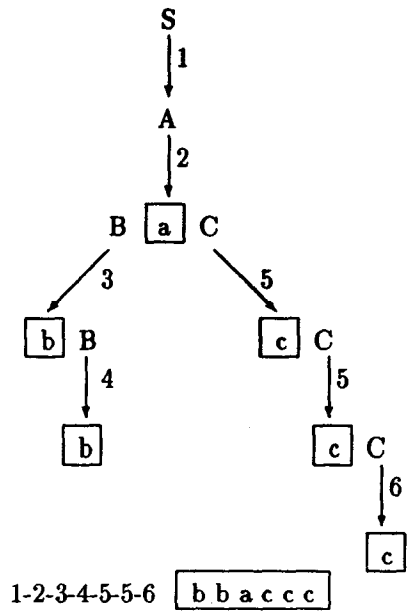
Ex.: 5 - $G2 = (\{S, A, B, C, \}, \{a, b, c\}, P, S)$

onde $P = \{$ 1) $S \rightarrow A$
2) $A \rightarrow Bac$
3) $B \rightarrow bB$
4) $B \rightarrow b$
5) $C \rightarrow cc$
6) $C \rightarrow c\ \}$

Portanto a linguagem gerada por $G2$ é:

bac
bbac
bacc $\Rightarrow$ ou seja,
bbbaccc $L(G2) = \{b^i a c^j | i, j \geq 1\}$
etc.

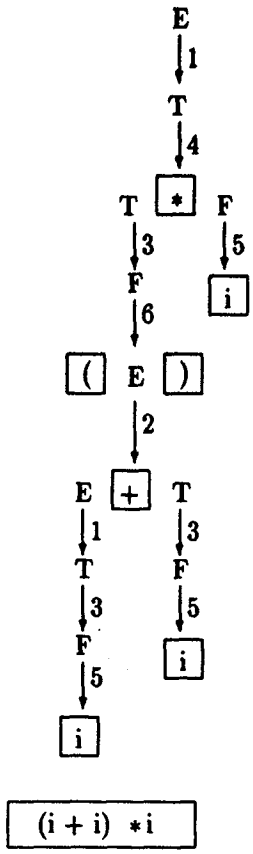
Árvore de derivação sintática para a cadeia bbaccc



Ex.: 6 - $G3 = (\{E, T, F\}, \{+, *, (,), i\}, P, E)$

$P = \{$ $E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow i\}$

Árvore de derivação sintática para a cadeia $(i + i) * i$:



Definição 16 Ambiguidade de Gramáticas

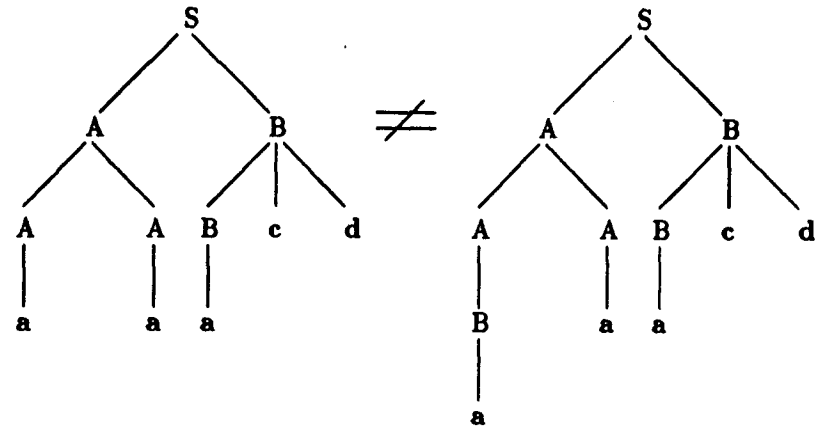
Uma **sentença** *é ambígua se* $\exists$ *duas ou mais árvores sintáticas que a definem.*

Uma **gramática** *é ambígua se possui alguma sentença ambígua.*

Ex.: 7 -

$S \rightarrow AB$
$A \rightarrow AA$
$A \rightarrow B$
$A \rightarrow a$
$B \rightarrow Bcd$
$B \rightarrow a$

Considere a cadeia $x = aaacd$



Ex.: 8 -

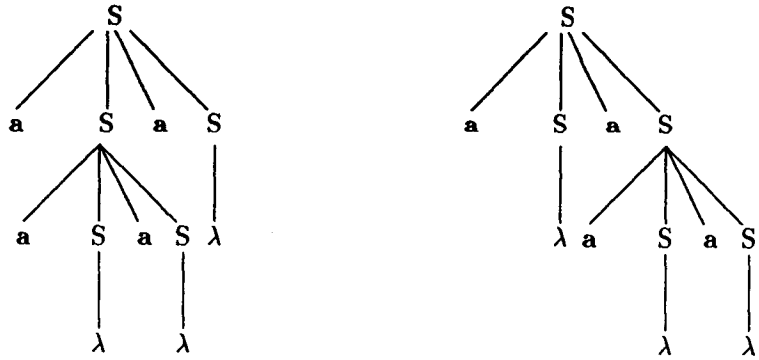
$$E \rightarrow E + E$$
$$E \rightarrow E * E$$
$$E \rightarrow (E)$$
$$E \rightarrow a$$

Considere a cadeia $x = a + a + a$

Ex.: 9 -

$$S \rightarrow aSaS$$
$$S \rightarrow \lambda$$

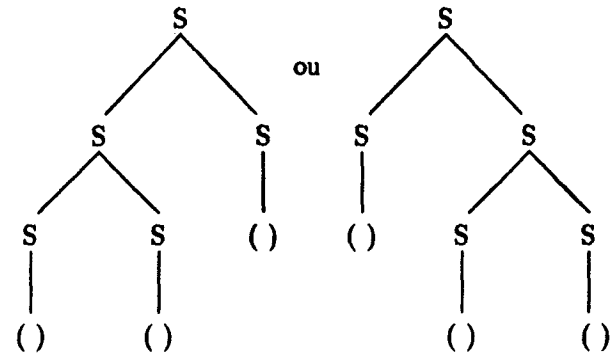
X = aaaa



Ex.: 10 -

$$S \rightarrow (S)$$
$$S \rightarrow (\,)SE$$
$$E \rightarrow (\,)$$

X = ( ) ( ) ( )



Ex: 11 - Construir gramáticas para a definição da linguagem

$L(G) = \{a^{2n} \mid n \geq 1\}$ por ex: aa aa aa $\in L(G)$

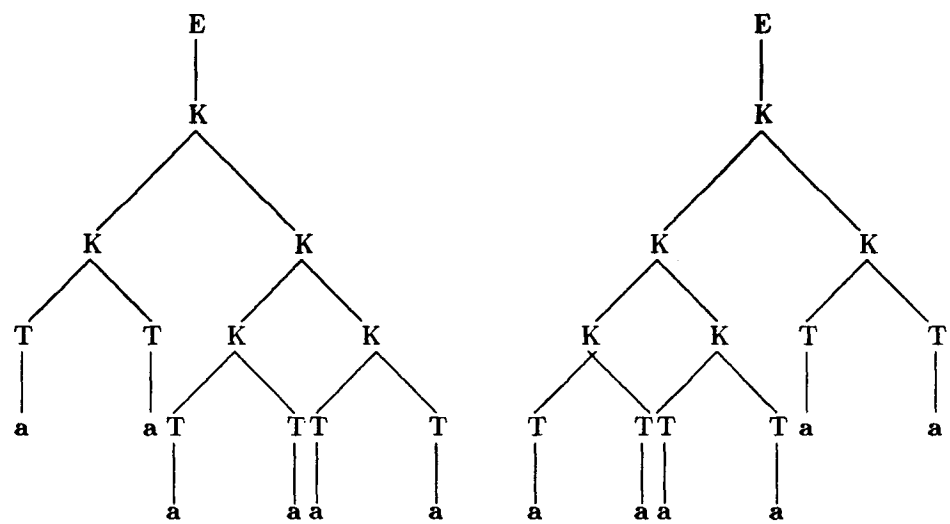
Exemplos

1º) $E \rightarrow K$

$K \rightarrow TT$

$K \rightarrow KK$

$T \rightarrow a$



* Esta solução não interessa pois a gramática é ambígua.

2º) $E \rightarrow T$
$T \rightarrow TF$
$T \rightarrow F$
$F \rightarrow aa$

3º) $E \rightarrow aa\ F$
$F \rightarrow E$
$F \rightarrow \lambda$

4º) $E \rightarrow ET$
$T \rightarrow aa$
$E \rightarrow aa\ |\lambda$

5º) $E \rightarrow aa$
$E \rightarrow E\ aa$

Obs.: - as 4 últimas soluções são consideradas gramáticas equivalentes.

3 Tipos de Gramáticas

Vamos considerar agora 4 tipos de gramáticas, que constituem a chamada "hierarquia de Chomsky" (gramáticas gerativas).

3.1 Gramáticas com Estrutura de Frase — GEF

Gramáticas - tipo 0 -

As gramáticas com Estrutura de Frase ou Gramáticas não Restritivas são aquelas cujas produções P são do tipo:

$$\alpha \rightarrow \beta \quad \text{onde} \quad \beta \in (V_N \cup V_T)^* \qquad \alpha \in (V_N \cup V_T)^+$$

Pode-se restringir P e obter as gramáticas dos tipos 1,2 e 3.

3.2 Gramáticas Sensíveis ao Contexto — GSC

Gramáticas - tipo 1 -

Seja $G = (V_N, V_T, P, S)$ onde toda produção de P é da forma

$\alpha \rightarrow \beta$ tal que $|\beta| \geq |\alpha|$ exceção quando $\beta = \lambda$

onde:

$\alpha \in (V_N \cup V_T)^+$

$\beta \in (V_N \cup V_T)^*$

Ex.: $G = (V_N, V_T, P, S)$.

$V_N = \{S, B, C\}$, $V_T = \{a, b, c\}$

$P = \{$ 1) $S \rightarrow aSBC$
2) $S \rightarrow aBC$
3) $CB \rightarrow BC$
4) $aB \rightarrow ab$
5) $bB \rightarrow bb$
6) $bC \rightarrow bc$
7) $cC \rightarrow cc\}$

Exercício: Mostrar que $L(G) = \{a^n b^n c^n \mid n \geq 1\}$

→ Usa-se a produção 1 n-1 vezes, obtemos

$$S \xrightarrow{*} a^{n-1} S (BC)^{n-1}$$

→ Usa-se a produção 2 para obter $S \xrightarrow{*} a^n (BC)^n$

→ Usa-se a produção 3 (n vezes) para arranjar os $B's$ e $C's$ tal que todos os $B's$ precedam todos os $C's$.

Por ex.: $n = 3$ $aaaB\mathbf{CB}CBC \xrightarrow{3} aaaBBC\mathbf{CB}C \xrightarrow{3}$

$aaaBB\mathbf{CB}CC \xrightarrow{3} aaaBBBCCC$

Portanto $S \xrightarrow{*} a^n B^n C^n$

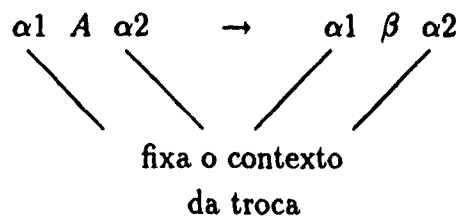
→ A seguir, usamos a produção 4 uma vez para obter $S \xrightarrow{*} a^n b B^{n-1} C^n$

→ Então usa-se a produção 5 (n-1) vezes para obter $S \xrightarrow{*} a^n b^n C^n$.

→ Finalmente, usa-se 6 uma vez e 7 (n-1) vezes para obter $S \xrightarrow{*} a^n b^n c^n$

$$L(G) = \{a^n b^n c^n \mid n \geq 1\}$$

Obs.: Alguns autores colocam as produções de uma GSC (GDC) como:



onde

- $\alpha 1,\ \alpha 2,\ \beta \in V^*$ e $\beta \neq \lambda$
- $A \in V_N$

para motivar o nome sensível ao contexto desde que a produção $\alpha 1\ A\ \alpha 2 \rightarrow \alpha 1\ \beta\ \alpha 2$ permita que A seja trocado por β.

3.3 Gramáticas Livres de Contexto — GLC

Gramática - tipo 2 -

Toda produção P, é da forma:

$A \rightarrow \beta$ onde:

$A \in V_N$ e $\beta \in (V_N \cup V_T)^*$

Ex.: 1 $G = (V_N, V_T, P, S)$ onde

$V_N = \{S, A, B\}$
$V_T = \{a, b\}$

$P = \{$
1) $S \rightarrow aB$
2) $S \rightarrow bA$
3) $A \rightarrow a$
4) $A \rightarrow aS$
5) $A \rightarrow bAA$
6) $B \rightarrow b$
7) $B \rightarrow bS$
8) $B \rightarrow aBB\}$

$L(G) = \{w \in \{a,b\}^+ \mid w$ contém números de a's igual ao número de b's $\}$

$S \xrightarrow{1} B \xrightarrow{7} abs \xrightarrow{2} abbA \xrightarrow{3} abba$
$S \xrightarrow{1} aB \xrightarrow{8} aaBB \xrightarrow{6} aabb$
$S \xrightarrow{2} bA \xrightarrow{4} baS \xrightarrow{1} baaB \xrightarrow{6} baab$
$\xrightarrow{2} babA \xrightarrow{3} baba$

$L(G)$ é o conjunto de todas as combinações de cadeias em V_T^+ com números de $a's$ = números de $b's$.

Prova: por indução sobre o comprimento de uma palavra (sentença).

Hipótese Indutiva (devemos provar para todo comprimento de W)

Para todo $W \in V_T^+$,

1) $S \xrightarrow{*} W$ sse W consiste de $n^{\underline{o}}de\ a's$ igual $n^{\underline{o}}de\ b's$.
2) $A \xrightarrow{*} W$ sse W tem um **a** a mais que **b**$'s$.
3) $B \xrightarrow{*} W$ sse W tem um **b** a mais que **a**$'s$

$\rightarrow$ se $|W| = 1$

temos que

2) $A \xrightarrow{*} a$,
3) $B \xrightarrow{*} b$ e
1) S não gera nenhuma sentença de terminais de comprimento = 1.

$\rightarrow$ Suponha que a hipótese seja verdadeira para todo $W||W| < k$

$\rightarrow$ Deve-se mostrar que é também válida para $|W| = k$

1) Se $S \xrightarrow{*} W$, então a derivação iniciou com $S \rightarrow aB$ ou $S \rightarrow bA$

(i) se $S \rightarrow aB, W$ é da forma aW_1 onde $|W_1| = k - 1$ e $B \xrightarrow{*} W_1$. Pela hipótese, o nº de $b's$ em W_1 é 1+ nº de $a's$.

Portanto $aW_1 = W$ contém nº $a's$ = nº $b's$

(ii) se $S \rightarrow bA \Rightarrow$ análogo.

Deve-se mostrar a volta de 1), isto é, se $|W| = k$, e W contém o mesmo número de $a's$ e $b's$, então $S \xrightarrow{*} W$.

Neste caso, o 1º símbolo de W é **a** ou **b**.

Assuma que $W = aW_1$. Então $|W_1| = k - 1$, e W_1 tem um **b** a mais que $a's$. Pela hipótese, $B \xrightarrow{*} W_1$. Mas, então:

$S \rightarrow aB \xrightarrow{*} aW_1 = W$

Portanto $S \xrightarrow{*} W$.

No caso de $W = bW_1 \Rightarrow$ análogo.

A prova ainda não está completa. Fica como exercício mostrar 2) e 3) para $|W| = k$.

Ex.: 2 - Podemos pensar num processo inverso, isto é, dada $L(G)$, determinar G, que gera $L(G)$.

$$L(G) = \{0^n\ 1^{2n}\ 0^m\}\ ,\ n \geq 0,\ m \geq 0$$
$$L(G) = \{\lambda, 0110, 01100, 0011110, \ldots\}$$

1) $S \rightarrow AB$

2) $A \rightarrow 0A11$

3) $A \rightarrow \lambda$

4) $B \rightarrow 0B$

5) $B \rightarrow \lambda$

$$S \xrightarrow{1} AB \xrightarrow{2} 0A11B \xrightarrow{2} 00A1111B \xrightarrow{3} 001111B \xrightarrow{4} 0011110B \xrightarrow{5} 0011110.$$

Ex.: 3:

$$L(G) = \{a^m\ b^n\ ,\ m > 0,\ \ n > 0\}$$

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA \\ A &\rightarrow a \\ B &\rightarrow bB \\ B &\rightarrow b \end{aligned}$$

Teorema 1 Forma Normal de Chomsky

Toda Linguagem Livre de Contexto pode ser gerada por uma gramática na qual todas as produções são da forma:

$A \rightarrow BC$

$A \rightarrow a$

onde $A, B, C \in V_N$ *e* $a \in V_T$

Teorema 2 Forma Normal de Greibach

Toda Linguagem Livre de Contexto pode ser gerada por uma gramática na qual todas as produções são da forma:

$A \rightarrow b\alpha$

onde $A \in V_N$, $b \in V_T$ *e* $\alpha \in ((V_N \cup V_T)^* \cup \{\lambda\})$.

(Ou seja, é o conjunto de todas as produções com o lado direito começando com um símbolo termimal)

3.4 Gramáticas Regulares — GR

Gramática - tipo 3 -

São as gramáticas com produções do tipo:

$$\begin{array}{lll} A \rightarrow aB & & A, B \in V_T \\ \quad \text{ou} & & a \in V_T \\ A \rightarrow b & & b \in V_T \cup \{\lambda\} \end{array}$$

Ex.: 1

$$S \rightarrow aS$$
$$S \rightarrow bA$$
$$A \rightarrow c$$

Qual é a $L(G)$? $L(G) = \{a^n bc |\ n \geq 0\}$

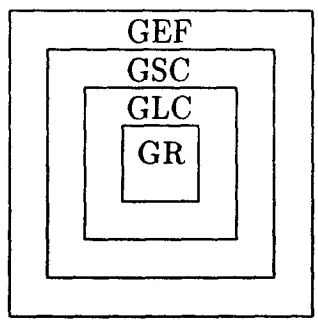
Ex.: 2 - Vamos escrever uma gramática regular que gere os números inteiros com sinal:

$G = (\{< inteiro >, < digitos >\}\ ,\ \{0, 1, \ldots, 9\}, P, < inteiro >)$
$< inteiro > ::= + < digitos > | - < digitos >$
$< digitos > ::= 0|1|2|3|4|5|6|7|8|9|$
$0 < digitos > | 1 < digitos > \ldots | 9 < digitos >$
$L(G) = \{\pm 0, \pm 1, \ldots, \pm 9, \pm 10, \ldots \pm 99, \ldots\}$

3.5 Hierarquia de Chomsky e as Linguagens

Como uma linguagem pode ser gerada por várias gramáticas que podem ser de tipos distintos; o que interessa na verdade é saber qual é o tipo da linguagem.

Hierarquia de Chomsky



Uma linguagem gerada por uma GSC, GLC ou GR é uma linguagem LSC, LLC ou LR.

Assim, como uma linguagem do tipo i é também do tipo j interessa saber qual é o maior i para o qual a linguagem é do tipo i.

3.6 Propriedades de Gramáticas Regulares

Dadas duas gramáticas regulares $G_1 = (V_{N1}, V_{T1}, P_1, S_1)$ e $G_2 = (V_{N2}, V_{T2}, P_2, S_2)$:

(a) Pode-se construir uma nova gramática **regular** $G3$, tal que

$$L(G3) = L(G_1).L(G_2) = \{xy \mid x \in L(G_1), \text{ e } y \in L(G_2)\}$$

Processo de Construção

$G_3 = (V_{N1} \cup V_{N2}, V_{T1} \cup V_{T2}, P_3, S_1)$
onde P_3 :

1. Se $A \rightarrow aB \in P_1$, então $A \rightarrow aB \in P_3$;
2. Se $A \rightarrow a \in P_1$, então $A \rightarrow aS_2 \in P_3$;
3. Todas as produções pertencentes a $P_2 \in P_3$.

Prova em "The Theory of Parsing, Translation and Compiling", Volume 1, Aho e Ullman.

Obs.:
Se $a = \lambda \Rightarrow A \rightarrow S_2$ (não é GR) - cuidado.

Ex.:

G_1
$S_1 \rightarrow 0A$
$S_1 \rightarrow 1B$
$A \rightarrow 1$
$B \rightarrow 2$

G_2
$S_2 \rightarrow 0$
$S_2 \rightarrow 1C$
$S_2 \rightarrow 0S_2$
$C \rightarrow 0$

$L(G_1) = \{01, 12\}$ $\qquad$ $L(G_2) = \{0, 0, \ldots, 10, 0, \ldots 010\}$

$G_3 =$
$S_1 \rightarrow 0A$
$S_1 \rightarrow 1B$
$*$ $A \rightarrow 1S_2$
$*$ $B \rightarrow 2S_2$
$S_2 \rightarrow 0$
$S_2 \rightarrow 1C$
$S_2 \rightarrow 0S_2$
$C \rightarrow 0$

$L(G_3) = \{010, 0110, 0100, \ldots 120, 1210, 1200, \ldots\}$

(*b*) Pode-se construir G_3 tal que

$$L(G_3) = L(G_1) \cup L(G_2) = \{w \mid w \in L(G_1), \text{ ou } w \in L(G_2)\}$$

Processo de Construção

$G_3 = (V_{N1} \cup V_{N2} \cup \{S3\}, V_{T1} \cup V_{T2}, P_3, S3)$

P_3 consiste das produções de P_1 e P_2, menos $S_1 \to \lambda$ ou $S_2 \to \lambda$, mais para todas as produções da forma $S_1 \to \alpha$ ou $S_2 \to \beta$, adicionamos a produção $S3 \to \alpha$ ou $S3 \to \beta$.
Somente repete-se as regras de S_1 e S_2 se elas aparecerem do lado direito das produções.

Obs: Prova em Hopcroft

Sejam G_1 e G_2 do Ex. anterior

G_3 :

$S \to 0A$
$S \to 1B$
$S \to \lambda$
$A \to 1$
$B \to 2$
$S \to 0$
$S \to 1C$
$S \to 0S_1$
$S_2 \to 0$
$S_2 \to 1C$
$S_2 \to 0S_2$
$C \to 0$

as produções:

$$\left\{ \begin{array}{l} S_1 \to 0A \\ S_1 \to 1B \\ S_1 \to \lambda \end{array} \right\}$$

são necessárias apenas quando S_1 aparece do lado direito.
$L(G_3) = \{01, 12, \lambda, 0, 10, 00, 010, 0, \ldots 10\} = L(G_1) \cup L(G_2)$

(*c*) **Definimos** $L^* = \{\lambda\} \cup L \cup L.L \cup L.L.L \cup \ldots$.
Isto é, se $x \in L^*$ então x consiste da concatenação de zero ou mais cadeias de L. (analogo à definição de V^*).

Dada uma GR, G_1, podemos construir uma GR, G, tal que $L(G) = L(G_1)^*$

Processo de Construção

$G = (V_{N1} \cup \{S\}, V_{T1}, P, S)$
onde P :

1. Se A $\rightarrow aB \in P_1$, então $A \rightarrow aB \in P$;
2. Se $A \rightarrow a \in P_1$, então $A \rightarrow aS \in P$ e $A \rightarrow a \in P$;
3. Se $S_1 \rightarrow \alpha \in P_1$, então $S \rightarrow \alpha \in P$ e
4. $S \rightarrow \lambda \in P$.

Ex.: G_1: $S_1 \rightarrow aS_1$
$S_1 \rightarrow bA$
$A \rightarrow c$

$L(G_1) = \{a^n b\, c |\ n \geq 0\}$

G: $S \rightarrow aS$
$S \rightarrow bA$
$A \rightarrow cS$
$A \rightarrow c$
$S \rightarrow \lambda$

$L(G) = \{(a^n bc)^* \ /\ n \geq 0\}$

$$L(G) = \{\ \underset{L_1^0}{\lambda}\ ,\ \underset{L_1^1}{bc}\ ,\ \underset{L_1^1}{abc}\ ,\ \underset{L_1^1}{aa\ldots bc}\ ,\ \underset{L_1^2}{bcbc}\ ,\ \underset{L_1^2}{ababc},\ldots\}$$

Resultado: As linguagens regulares são fechadas sob as operações de união, concatenação e clausura$^{(*)}$.

Isto é, a união, concatenação e clausura de *LRs* resulta numa *LR*.

Isto significa que $\forall LR$ não vazia pode ser construida a partir de um número finito de cadeias simples e um número finito de operações de união, concatenação e clausura.

Relação de correspondência entre as classes de linguagens, gramáticas e reconhecedores.				
Linguagem		Gramática		Reconhecedor
tipo 0:	Conjuntos Recursivamente Enumeráveis	tipo 0 :	Gramática com Estrutura de Frase	Máquinas de Turing
tipo 1:	Sensíveis ao Contexto	tipo 1:	Gramática Sensíveis ao Contexto	Máquinas de Turing com Memória Limitada
tipo 2:	Livres de Contexto	tipo 2:	Gramáticas Livres de Contexto	Autômatos à Pilha
tipo 3:	Conjuntos Regulares	tipo 3:	Gramáticas Regulares	Autômatos Finitos

4 Ambiguidade nas Linguagens de Programação

Um requisito importante de uma LP é que ela não seja ambígua, ou no mínimo, qualquer ambiguidade na linguagem seja facilmente evitada.

O mais famoso caso de ambíguidade é o **else pendente**

Seja a gramática:

$$\begin{cases} C \rightarrow \textbf{if } b \textbf{ then } C \textbf{ else } C \\ C \rightarrow \textbf{if } b \textbf{ then } C \\ C \rightarrow s \end{cases}$$

A gramática é ambígua desde que a cadeia

if b **then if** b **then** s **else** s

pode ser interpretada como

if b **then** (**if** b **then** s **else** s)

ou

if b **then** (**if** b **then** s) **else** s

Nas regras que definem o **Algol 60** a sequência:

if ... then if ... não é permitida pois um comando condicional é:

<cc>:: <if com>|
<if com> else <comando>
<if com>:: = if <exp> then <comando inc>

Mas os compiladores como **Pascal** e **Algol** permitem, **if then ... if** escolhendo a interpretação que associa o **else** com o if mais próximo.

Teorema 3 *O problema de decidir se uma dada gramática (GLC) é ambígua é* **insolúvel.**

Mas em casos particulares pode-se reconhecer a ambiguidade como propriedade da gramática mais do que da linguagem. É sempre possível reescrever a definição de uma gramática para que não seja ambígua, ou seja, encontrar uma gramática equivalente não ambígua.

Mas há linguagems que são inerentemente ambíguas, isto é, não existe uma GLC não ambígua que as definam. Mas problemas de linguagem inerentemente ambíguas não existem em L.P.

Seja $G1 = (\{< E >, < T >, < F >, < U >\}, \{0, 1, 2, \ldots, 9, +, *, (,)\}, P, < E >)$

onde:

$$\begin{aligned} P = \; & \{< E > ::= < E > + < T > \mid < T > \\ & < T > ::= < T > * < F > \mid < F > \\ & < F > ::= < U > \mid (< E >) \\ & < U > ::= 0 \mid 1 \mid \ldots \mid 9 \end{aligned}$$

$G1$ é a gramática usada para gerar expressões aritméticas com $(+ \; e \; *)$ para inteiros em PASCAL.

Seja $G2 = (\{< E >, < F >\}, \{0, 1, 2 \ldots 9, +, *, (,), \}, P, < E >)$

onde:

$$\begin{aligned} P = \; & \{< E > ::= < E > + < E > \mid < E > * < E > \mid (< E >) \mid < F > \\ & < F > ::= 0 \mid 1 \mid \ldots \mid 9\} \end{aligned}$$

equivalente a $G1$.

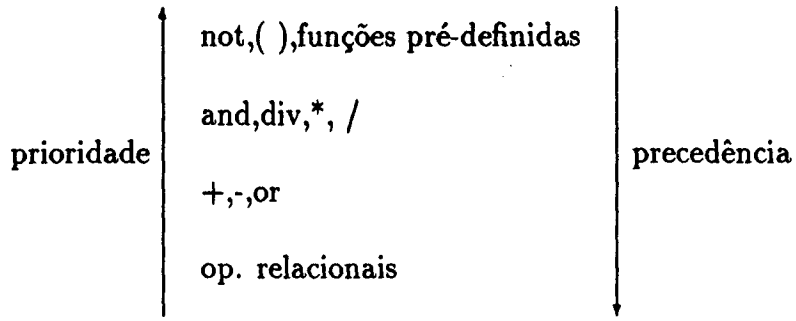
$G2$ contém um menor número de produções e símbolos que a gramática $G1$. Então porque as Linguagens como PASCAL não usam $G2$?

razão: **G2** é ambígua **G1** não é.

5 Precedência e Associatividade de Operadores

Precedência de operadores decide a correta interpretação das expressões

Quanto maior a precedência do operador, maior é o escopo que o operador atinge.



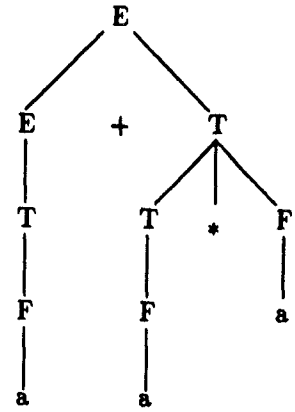
Por exemplo, na gramática

$E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow a$
$F \rightarrow 2$
$F \rightarrow b$
$F \rightarrow c$

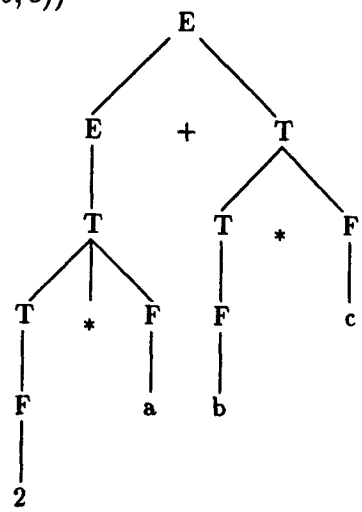
+ tem maior precedência que * pois está definido mais acima na árvore e no juntor.

* tem maior prioridade que + pois é resolvido primeiro que +.

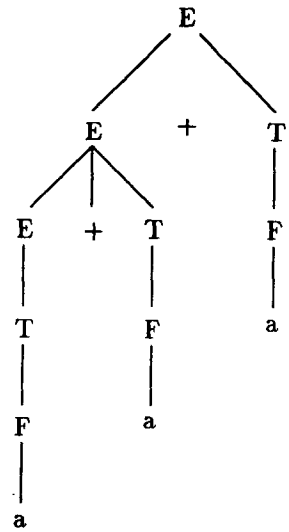
Ex.: 1 - $a + a * a$
$+(a, *(a, a))$



Ex.: 2 - $2 * a + b * c$
$+(*(2,a), *(b,c))$



Ex.: 3 - $a + a + a$
$+(+(a,a),a)$



Associatividade - decide a correta interpretação das expressões com operadores de mesma precedência.

$$G = \{ \begin{array}{l} E \rightarrow E + E \\ E \rightarrow E * E \\ E \rightarrow (E) \\ E \rightarrow a \} \end{array}$$

Essa gramática possui duas características não desejáveis:

- G é ambígua, mas esta ambiguidade pode ser removida

$$G1 = \{ \begin{array}{l} E \rightarrow E + T \\ E \rightarrow E * T \\ E \rightarrow T \\ T \rightarrow (E) \\ T \rightarrow a \} \end{array}$$

- em G e em $G1 \Rightarrow +$ e $*$ tem a mesma precedência

$$\left.\begin{array}{lcl} a + a * a & & (a + a) * a \\ & \Rightarrow & \\ a * a + a & & (a * a) + a \end{array}\right\} \text{ pela associatividade}$$

Voltamos à gramática G_0 para obter a precedência convencional.

$$G_0 = \{ \begin{array}{l} E \rightarrow E + T \\ E \rightarrow T \\ T \rightarrow T * F \\ T \rightarrow F \\ F \rightarrow (E) \\ F \rightarrow a \} \end{array}$$

Exemplo:

Construir uma gramática que gere todas as expressões Booleanas válidas, envolvendo os seguintes operadores em ordem crescente de precedência,

$\equiv$ (equivalência)

$\rightarrow$ (implicação)

$\vee$ (ou)

$\wedge$ (e)

$\neg$ (negação)

e os operadores **a, b e c**, de modo que a gramática não gere nenhuma sentença com parenteses supérfluos.

Exemplo de sentença válida: $\neg(a \vee b) \equiv \neg a \wedge \neg b$

Exemplo de sentença inválida: $a \equiv (b \wedge c)$

Solução

<E>:: = <E>≡ <T>| <T>
<T>:: = <T>→ <F>| (<T>≡ <F>) | <F>
<F>:: = <F>∨ <G>| (<F>→ <G>) | (<F>≡ <G>) | <G>
<G>:: = <G>∧ <H>| (<G>∨ <H>) | (<G>→ <H>) | (<G>≡ <H>) | <H>

6 Linguagens Regulares

Os métodos para expressar as linguagens regulares são:

1) Gramáticas Regulares;
2) Expressões Regulares.

Um estudo da teoria das linguagens regulares é justificado pelo fato de que estas modelam a Análise Léxica de um Compilador; o uso de Autômatos Finitos modelam certos aspectos do funcionamento do cérebro (Redes Neurais); outra aplicação é no casamento de padrão (procura de uma palavra em um texto, muito usada em editores), entre outros.

Vários utilitários do UNIX usam expressões regulares como especificação de entrada como por exemplo:

- LEX → Gerador de Analisadores Léxicos,
- AWK → Ling. de propósito específico, é utilizada como filtro que fornece facilidades para processamento de textos,
- e outros como ED, GREP, EX, VI, TROFF, NROFF.

6.1 Conjuntos Regulares

Definição 17 *Seja V um alfabeto finito. Definimos um conjunto regular sobre V da seguinte forma:*

a) *o conjunto vazio é um conjunto regular sobre V*

b) $\{\lambda\}$ *conjunto com cadeia vazia - é um conjunto regular sobre V*

c) $\{\mathbf{a}\}$ *é um conjunto regular sobre V para todo* **a** *pertencente a V*

d) *Se P e Q são conjuntos regulares sobre V, então também são*

- $P \cup Q$ *(união)*
- PQ *(concatenação)*
- P^* *(fechamento)*

e) *nada mais é um conjunto regular*

Obs.: O conjunto é regular se pode ser obtido por um número finito de aplicações das operações de união, concatenação e fechamento.

O método para denotar conjuntos regulares são as **Expressões Regulares.**

6.2 Expressões Regulares

Definição 18 *Expressões Regulares sobre V são definidas como:*

a) $\emptyset$ *é uma expressão regular denotando o conjunto regular* $\emptyset$ *(vazio)*

b) λ *é uma expressão regular denotando o conjunto regular* $\{\lambda\}$

c) a em V é uma expressão regular denotando o conjunto regular $\{a\}$

d) Se p e q são expressões regulares denotando os conjuntos regulares P e Q respectivamente então:

- $(p+q)$ *é uma expressão regular denotando* $P \cup Q$
- (pq) *é uma expressão regular denotando* PQ
- $(p)^*$ *é uma expressão regular denotando* P^*

e) nada mais é uma expressão regular.

Obs.:

- Usa-se a notação p^+ para denotar a expressão regular pp^*;
- Pode-se remover os parênteses quando não houver ambiguidade.

Precedência: $*$ |
conc |
$+$ ↓

Ex.: $0 + 10^*$ significa $(0 + (1(0^*)))$

Ex.:

1) 01 denota a Linguagem Regular $\{01\}$;

2) 0^* denota a Linguagem Regular $\{0^*\}$;

3) $(0+1)$ denota a Linguagem Regular $\{0,1\}$;

4) $(0^*1^*)^*$ conjunto de todas as cadeias de 0 e 1 incluindo a cadeia nula;

5) $(0+1)^*011$ denota o conjunto de cadeias de 0's e 1's terminadas por 011;

6) $(a+b)(a+b+0+1)^*$ denota o conjunto de cadeias em $\{0,1,a,b\}^*$ começando por a ou b;

7) $(ab+c)$ denota a linguagem $\{ab,c\}$;

$$\begin{array}{llll} (\text{ab} & + & \text{c}) & \\ G_1' & & G_1'' & \\ G_1' & = & S_1 \to aB & \\ & & B \to b & \end{array}$$

$$\begin{array}{ll} G_1'' & S_2 \to c \\ L(G_1') \cup L(G_1'') & = \quad S \to aB \\ & \quad\;\; S \to c \\ & \quad\;\; B \to b \end{array}$$

8) $(a+b+c)^*$ denota a linguagem $\{\lambda, a, b, c, aa, ab, ac, ba, bb, ca, cb, cc, aaa, aab,$ isto é, todas as cadeias sobre $\{a,b,c\}$;

$$\begin{array}{lll} (G_2) & = & S_2 \to aS_2 \\ & & S_2 \to bS_2 \\ & & S_2 \to cS_2 \\ & & S_2 \to \lambda \end{array}$$

9) $(aa^*+c)b$ denota $\{cb, ab, aab, aa\ldots ab, \ldots\}$;

$$\begin{array}{lll} (G_3) & = & S_3 \to cB \\ & & B \to b \\ & & S_3 \to aB \\ & & S_3 \to aA \\ & & A \to aB \\ & & A \to aA \end{array}$$

$$\begin{array}{llllll} (a & a^* & + & c) & b \\ G_1' & G_2' & & G_3' & G_4' \end{array}$$

$$G_1' : \quad S_1 \to a$$

$$\begin{array}{ll} G_2' : & S_2 \to aS_2 \\ & S_2 \to \lambda \end{array}$$

$$G_3' : \quad S_3 \to c$$

$$G_4' : \quad S_4 \to b$$

$$
\begin{array}{lllll}
L(G_1).L(G_2) & = & G_1' & = & S_1 \rightarrow aS_2 \\
 & & & & S_2 \rightarrow aS_2 \\
 & & & & S_2 \rightarrow \lambda \\
L(G_1') \cup L(G_3) & = & G & = & S \rightarrow aS_2 \\
 & & & & S \rightarrow c \\
 & & & & S_2 \rightarrow aS_2
\end{array}
$$

$$
\begin{array}{ll}
L(G).L(G_4) & S \rightarrow aS_2 \\
 & S \rightarrow cS_4 \\
 & S_2 \rightarrow aS_2 \\
 & S_4 \rightarrow b
\end{array}
$$

O processo usado para encontrar uma expressão regular denotando a $L(G)$ para uma dada gramática regular é o seguinte:

Toma-se as produções da gramática G como definindo um conjunto de equações:

$$
\begin{array}{l}
\{S \rightarrow aS \\
\ S \rightarrow bR \\
\ R \rightarrow aS\}
\end{array}
$$

Em lugar das 3 produções escrevemos 2 equações em $L(R)$ e $L(S)$:

$$
\begin{array}{lll}
L(S) & = & \{a\}.L(S) \cup \{b\}.L(R) \\
L(R) & = & \{a\}.L(S)
\end{array}
$$

$L(G) = L(S)$ por definição (G=gramática S=símbolo inicial)

Com expressões regulares dispensa-se os { } e abrevia-se $L(S)$ e $L(R)$ por S e R

$$
\begin{array}{l}
S = aS + bR \\
R = aS
\end{array}
$$

Substitui-se o valor de R em S :
$S = aS + baS \rightarrow S = (a + ba)S$ pelo axioma $A6$.

Regra para eliminar S do lado direito:

A equação $X = \alpha X + \beta$ tem como solução $X = \alpha^* \beta$

Logo $S = (a + ba)S \rightarrow S = (a + ba)^*$

Axiomas:		
	$A1$	$(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$
	$A2$	$(\alpha.\beta).\gamma = \alpha.(\beta.\gamma)$
	$A3$	$\alpha + \beta = \beta + \alpha$
	$A4$	$\alpha + \alpha = \alpha$
	$A5$	$\alpha.(\beta + \gamma) = (\alpha.\beta) + (\alpha + \gamma)$
	$A6$	$(\beta + \gamma).\alpha = (\beta.\alpha) + (\gamma.\alpha)$
	$A7$	$\alpha + \emptyset = \alpha$
	$A8$	$\alpha.\emptyset = \emptyset.\alpha$
	$A9$	$\lambda.\alpha = \alpha = \alpha.\lambda$
	$A10$	$\alpha^* = \lambda + \alpha.\alpha^*$
	$A11$	$\alpha^* = (\lambda + \alpha)^*$

Obs. adicionais:

- $\rightarrow$ a^* fechamento;
- $\rightarrow$ a^+ fechamento positivo;
- $\rightarrow$ classes $[a-z0-9]$ de a á z e de 0 á 9 (range);
- $\rightarrow$ complemento $[\hat{}a-z]$ tudo menos letras minúsculas;
- $\rightarrow$ $(ab|cd) = ab$ ou $cd = (ab + cd)$;
- $\rightarrow$ $[A-Za-z][A-Za-z0-9]^*$ identificador;
- $\rightarrow$ $[0-9]^+$ números inteiros.

Exercícios:

1) Seja $G1$ uma gramática cujas produções são:

$\{S_1 \rightarrow aS_1$
$S_1 \rightarrow bS_1$
$S_1 \rightarrow aS_2$
$S_2 \rightarrow b\}$

Qual é $L(G1)$ em expressões regulares? mostre sua resposta.

2) Seja $G2$ uma gramática cujas produções são:

$\{S_2 \rightarrow lR$
$S_2 \rightarrow l$
$R \rightarrow d$
$R \rightarrow dR$
$R \rightarrow l$
$R \rightarrow lR$

Qual é $L(G2)$ em expressões regulares? mostre sua resposta.

6.3 Exercícios

1. Descrever formalmente o conjunto fechamento do conjunto $A = \{00\}$
2. Seja $A = \{+\}$ e seja $z = +$.
   Escrever as cadeis abaixo declarando o tamanho de cada uma delas;
   z, zz, z^2, z^5, z^0
   Escrever o fechamento de A.
3. Seja $A = \{0,1\}, x = 01$ e $y = 110$
   Escrever as cadeias $xy, yx, xyx, (x^2)(xy)^3, (xx)^3$.
   Dar também o fechamento positivo de A.
4. Construir gramáticas não ambíguas para a definição de linguagens cujas sentenças sejam:

   a) $\underbrace{a\ldots a}_{n}\underbrace{b\ldots b}_{n}$, $n > 0$

   b) $\underbrace{a\ldots a}_{2n-1}\underbrace{b\ldots b}_{n}$, $n > 0$

   c) $\underbrace{a\ldots a}_{n}\underbrace{b\ldots b}_{n}\underbrace{a\ldots a}_{n}$, $n > 0$

   d) $\underbrace{a\ldots a}_{m}\underbrace{b\ldots b}_{n}\underbrace{a\ldots a}_{m}$, $m > 0, n > 0$

   e) $\underbrace{a\ldots a}_{m}\underbrace{b\ldots b}_{n}\underbrace{a\ldots a}_{m}\underbrace{b\ldots b}_{n}$, $m > 0, n > 0$

   Obs.: Duas entre as linguagens acima não poder ser descritas por uma gramática em notação BNF. Quais são elas?
5. Descrever as linguagens geradas pelas gramáticas abaixo e classificá-las segundo a hierarquia de Chomsky.

   a) $S \to A0$
      $A \to 1A$
      $A0 \to 10$

   b) $S \to 1S$
      $S \to 1A$
      $A \to 0a$
      $A \to 0$

   c) $S \to S0$
      $S \to A1$
      $A \to 0A0$
      $A \to 1$

   d) $S \to 0S1$
      $S \to 01$

   e) $S \to 0A$
      $S \to 1B$
      $A \to 0A$
      $A \to 0$
      $B \to 1B$
      $B \to 1$

   f) $S \to aXa$
      $X \to bX$
      $aX \to aC$
      $Xa \to ca$
      $X \to XB$
6. Escreva uma gramática em BNF cujas sentenças sejam números inteiros positivos, na base 10 e múltiplos de 5.
7. Escrever uma gramática regular cujas sentenças sejam números positivos em base decimal, múltiplos de 4. Permitir os zeros não significativos.

8. Mostre qual a linguagem gerada pelas produções da GLC abaixo e construa uma GR equivalente:

   $S \rightarrow AB$
   $A \rightarrow aA$
   $A \rightarrow aBB$
   $B \rightarrow Bb$
   $B \rightarrow b$

9. Seja $G2$ uma gramática descrita pelo seguinte conjunto de produções:
   $< id > ::= a \mid b \mid c < id > a \mid < id > 0 \mid < id > 1$

   Escrever V_N, V_T.
   Mostrar as derivações possíveis para gerar as sentenças $a, ab0, a0c01, 0a, 11, aaa$

10. Escrever uma gramática cuja linguagem seja o conjunto dos números inteiros pares positivos sem zeros à esquerda.

11. Construir uma gramática que gere a linguagem $\{a\,(b^n)\,a \mid n \geq 0\}$

12. A gramática $G3$ abaixo é frequentemente usada para expressões aritméticas com operadores binários:

    $< exp > \;::=\; < termo > \mid < exp > + < termo > \mid < exp > - < termo >$
    $< termo > \;::=\; < fator > \mid < termo > * < fator > \mid < termo > \mid < fator >$
    $< fator > \;::=\; (< exp >) \mid i$

    Dar as árvores de derivação sintática das seguintes expressões aritméticas:

    $i,\ (i),\ i * i,\ i * (i + i)$

13. Construir uma gramática cujas sentenças sejam cadeias com números de $0's$ igual ao $1's$.

14. Seja $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
    Pede-se:

    a) Descrever (com palavras) o conjunto fechamento positivo de A.
    b) Dizer se em A^+ os elementos 0015 e 15 são os mesmos e porquê.

15. Seja $G4$ a gramática:
    $< cad > \;::=\; ab \mid a < meio > b$
    $< meio > \;::=\; a < meio > \mid < meio > b \mid a \mid b$

    Pergunta-se:

    Trata-se de uma gramática ambígua ou não? Justifique. Qual a linguagem gerada por $G4$?

16. Dadas as seguintes gramáticas $G5$ e $G6$.

$G5$	$G6$
$E \rightarrow E + E$	$E \rightarrow E + T$
$E \rightarrow E * E$	$E \rightarrow T$
$E \rightarrow (E)$	$T \rightarrow T * F$
$E \rightarrow a$	$T \rightarrow F$
$E \rightarrow b$	$F \rightarrow a$
	$F \rightarrow b$
	$F \rightarrow (E)$

a) Enumere todas as derivações distintas da cadeia $b * a + a$ usando ambas as gramáticas.

b) Mostre que as gramáticas $G1$ e $G2$ geram a mesma linguagem.

c) Mostre que a gramática $G2$ não é ambígua.

17. Quais as linguagens geradas pelas seguintes gramáticas?

a) $S \rightarrow 0S1 \quad S \rightarrow 01$

b) $S \rightarrow +SS \quad S \rightarrow -SS$
$S \rightarrow a$

c) $S \rightarrow S(S)S \quad S \rightarrow \&$

d) $S \rightarrow aSbS \quad S \rightarrow bSaS$
$S \rightarrow \&$

e) $S \rightarrow a \quad S \rightarrow S + S$
$S \rightarrow SS$
$S \rightarrow S*$
$S \rightarrow (S)$

Quais destas gramáticas são ambíguas?

18. Considere a gramática livre de contexto

$S \rightarrow +$
$S \rightarrow SS*$
$S \rightarrow a$

a) Mostre como é gerada a sentença $(aa+a*)$ e construa a árvore de derivação sintática correspondente.

b) Qual é a linguagem gerada por esta gramática?

19. Produzir uma gramática G tal que:

a) $L(G) = \{ab^n c \text{ com } n \geq 0\}$

b) $L(G) = \{a^n b^m \text{ com } n \leq m, n, m \geq 1\}$

c) $L(G) = \{a^{2^n} | n \geq 0\}$

d) $L(G) = \{a^{n^2} | n \geq 0\}$

e) $L(G) = a^n b^m | n \leq m \leq 2n \text{ onde } n, m > 0\}$

20. Considere a GLC com as seguintes produções

<expressão> :: = <expressão> + <termo>| <termo>
<termo> :: = <termo> * <fator>| <fator>
<fator> :: = (<expressão>) | id

Para esta gramática, encontre:

1) derivações mais à esquerda
2) derivações mais à direita
3) derivações que não são nem mais à esquerda nem mais à direita para as sentenças:

a) $rd + id + id$
b) $(id + id) * (id + id)$
c) $((id + id) * id) + id$

21. Considere a gramática descrita pelas regras de produção abaixo:

$S \rightarrow aAS$ $\qquad$ $S \rightarrow a$
$A \rightarrow SbA$ $\qquad$ $A \rightarrow ba$
$A \rightarrow SS$

construa derivações mais à esquerda para as cadeias:

a) *aaa ba baaaabb aa baa*
b) *aaa baba aaa bb aa ba aa bbaa*

22. Escrever um algorítmo que, dadas duas cadeias α_1 e α_2 e uma $G.L.C$ $G = (V_N, V_T, P, S)$, determine se $\alpha_1 \overset{*}{\underset{G}{\Longrightarrow}} \alpha_2$.

23. Considere a gramática

$$G = (\ \{S,T,L\}\ ,\ \{a,b,+,-,*,/,[,],\},P,S)$$

onde $P =$
$\{S \rightarrow T + S$
$S \rightarrow T - S$
$S \rightarrow T$
$T \rightarrow L * T$
$T \rightarrow L/T$
$T \rightarrow L$
$L \rightarrow [S]$
$L \rightarrow a$
$L \rightarrow b\}$

Descreva informalmente qual é $L(G)$. Encontre uma gramática na forma normal de Chomsky que gera $L(G)$.

24. Considere a gramática $G(\{A,B,C\}\ ,\ \{0,1\},P,A)$ onde P consiste das produções:

$A \rightarrow 0A1$ $\qquad$ $B \rightarrow 1C0$
$A \rightarrow 0AC$ $\qquad$ $B \rightarrow AC$
$A \rightarrow 0$ $\qquad$ $C \rightarrow 1CB$
$A \rightarrow 1B0$ $\qquad$ $C \rightarrow AB$

Encontre uma gramática G_1 equivalente a G.

25. Se G é uma gramática, escrita na FNC onde $W \in L(G)$ e há uma derivação de W usando P passos, qual o comprimento de W? Prove sua resposta.

26. Considere a gramática

$$G = (\{S\}\ ,\ \{p, [,], n, c\}, P, S)$$

onde P consiste das produções:

$$\begin{aligned} P = \quad & \{S \to p \\ & S \to nS \\ & S \to [ScS]\} \end{aligned}$$

Descreva $L(G)$ informalmente. Encontre uma gramática equivalente escrita na forma normal de Chomsky. A partir da gramática escrita na FNC encontre uma gramática equivalente escrita na forma normal de Greibach.

27. Dê uma GLC que gera a linguagem:

$$L = \{0^i 1^i 2^k \mid i = j \text{ ou } j = k, \text{ onde } i, j, k > 0\}$$

28. Dada a $GLC, G = (V_N, V_T, P, S)$ onde

$$\begin{aligned} V_N = \quad & \{S, A, B, C, D\} \\ V_T = \quad & \{0, 1\} \\ P = \quad & \{S \to 0S1|0A|D \\ & A \to ASC|1C|1 \\ & B \to 0B1|01 \\ & C \to 0D1 \\ & C \to 01C\} \end{aligned}$$

Encontrar uma GLC equivalente sem simbolos estéreis e inacessíveis.

29. Encontre uma gramática livre do contexto, sem símbolos inúteis, equivalente à gramática descrita pelas regras abaixo:

$$\begin{aligned} & S \to AB|CA && A \to AA|C|D \\ & A \to a && C \to aB|b \end{aligned}$$

30. Considere a gramática abaixo:

$$\begin{aligned} G = & (\{S, X, Y, B, A, C, D\},\ \{0, 1\}, P, S) \\ P = \quad & S \to XY \quad A \to AA|C|D \\ & X \to B \quad C \to 0 \\ & Y \to \lambda \quad D \to 1 \\ & B \to YA \quad E \to X \end{aligned}$$

Construa uma gramática G_1 equivalente a G tal que G_1 não tenha produções do tipo $A \to \varepsilon$ nem $A \to B$ e não tenha símbolos estéreis ou inacessíveis. Se G_1 for amígua, construir G_2 equivalente a G_1 tal que G_2 seja não ambígua.

31. Encontre gramáticas **não ambíguas** que gerem os seguintes conjuntos:

    a) números binários multiplos de 4, sem zeros não significativos.

    b) $L_1 = \{X | X \in \{0, 1\}^*$ não possui três, 1's consecutivos $\}$

c) números inteiros pares, positivos ou negativos, com sinal (exceto o zero), sem zeros não significativos.

d) $L_3 = \{W \mid W \in \{(,)\}^*$ e W está balanceada $\}$

32. Considere a gramática $G = (\{S, A, B\}, \{0, 1\}, P, S)$ onde P consiste das produções

$S \to AB \quad A \to a \quad B \to \lambda$
$S \to aBB \quad A \to aB \quad B \to bB$
$S \to aSBa \quad A \to aSb \quad B \to aSB$

Encontre uma gramática equivalente,onde o símbolo reservado (inicial) S não aparece do lado direito das produções, e $S \to \lambda$ é a única produção que possui λ do lado direito.

33. Passar para a **forma normal de Chomsky** e para a **forma normal de Greibach** as GLC abaixo:

1) $S \to AS$
$S \to b$
$A \to aB$
$B \to aB$
$B \to b$

2) $S \to AB$
$A \to a|b$
$B \to AC$
$C \to D$
$D \to a$

3) $S \to A|B$
$A \to aaA|aa$
$B \to aaB|a$

4) $S \to Ab|Ac$
$A \to AB|a$
$B \to a$

5) $S \to Ab|Bc$
$A \to Aa|a$
$B \to Ba|a$

6) $S \to BAb|CAc$
$A \to BA|a$
$B \to a$
$C \to a$

34. Dizer se a gramática abaixo possui a propriedade do encasulamento. Justificar sua resposta.

1. $A \rightarrow CB$
2. $A \rightarrow b$
3. $B \rightarrow CA$
4. $C \rightarrow AB$
5. $C \rightarrow a$

Obs.: Uma gramática G_1 é encasulada se $\exists$ variável A tal que $A \rightarrow \alpha_1 A \alpha_2$ onde α_1 e α_2 são cadeias não vazias.

35. Encontre uma gramática regular equivalente à gramática $G = (V_N, V_T, P, S)$, onde

$$\begin{aligned} V_N = &\ \{S, A, B\} \\ V_T = &\ \{a, b\} \\ P = &\ \{S \rightarrow abbaA|bB \\ &\ A \rightarrow aA|aB \\ &\ B \rightarrow bA|aba|b\} \end{aligned}$$

36. Escreva uma gramática cuja linguagem seja $\{a^n b^n | \ n \geq 1\}$

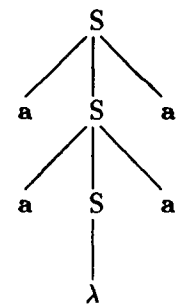
37. Determine pelo menos 10 sentenças geradas pelas produções da gramática:

<S>:: = <B>| se <S>então <S>senão <S>
<B>:: = <C>| <C>+ <D>| + <C>
<C>:: = <D>| <C>* <D>| * <D>
<D>:: = x | (<S>) | - <D> .

38. Dizer se são ambíguas as gramáticas e determinar a $L(G)$:

a) $S \rightarrow aSa$
$S \rightarrow \lambda$

X = aaaa



b) E → E + E
E → E * E
E → (E)
E → Identificador
Identificador → a

c) $S \rightarrow SE$
$S \rightarrow E$

$E \rightarrow (\ )$
$E \rightarrow (S)$

d) $S \rightarrow (S)$
$S \rightarrow (\ )S(\ )$

7 Conclusões

Como comentado anteriormente esta Nota Didática tem como objetivo familiarizar o leitor com os conceitos básicos das Gramáticas enfatizando aqueles que subsidiam a definição de Linguagens de Programação.

Esta Nota Didática terá continuidade em uma próxima nota relacionada com os Reconhecedores das Gramáticas Regulares e Livres de Contexto, a ser publicada futuramente.

Agadecimentos: as Profas. Maria das Graças Volpe Nunes, Rosely Sanches e Sandra Caldeira pela contribuição com material didático; a Carmen Lucia Pagadigorria pela digitação deste documento.

Referências

[Backhouse 79] Backhouse, R.C. *Sintax of Programming Languages - Theory and Pratice.* Englewood Cliffs, Prentice Hall, 1979.

[Hopcroft 69] Hopcroft, J.E.; Ullman, J.D. *Formal Languages and their Relation to Automata.* Reading, Addison-Wesley, 1969.

[Hopcroft 79] Hopcroft, J.E.; Ullman, J.D. *Introduction to Automata Theory, Languages and Computation.* Reading, Addison-Wesley, 1979.

[Suonio 71] Suonio, R. K. *Computability and Formal Languages.* AUERBACH Publishers Inc., Princeton, N.J., 1971.