9.3 Gramáticas Irrestritas

Para investigar a conexão entre as linguagens recursivamente enumeráveis e as gramáticas, retornaremos à definição geral de uma gramática. Na definição 1.2.11, as regras de produção poderiam ser de qualquer forma, mas várias restrições foram feitas, depois, para se obter gramáticas específicas. Se tomarmos a forma geral não impondo restrições, obteremos uma **gramática irrestrita**. Assim, numa gramática irrestrita, qualquer número de variáveis ou terminais podem ocorrer em qualquer ordem, dentro de uma produção. Existe somente uma ligeira restrição: λ não é permitido no lado esquerdo da produção.

Exemplo 9.3.1 *Uma gramática irrestrita para gerar a linguagem*

$$\mathcal{L} = \{a^n b^{2n} a^n \ / \ n \geq 1\}$$

é descrita a seguir.

$S \longrightarrow aAbba$
$aAb \longrightarrow aabbbA \ | \ ab$
$bAb \longrightarrow bbA$
$bAa \longrightarrow Bbaa$
$bB \longrightarrow Bb$
$aB \longrightarrow aA$

A cadeia aaabbbbbbaaa pode ser derivada dessa gramática como segue

$$\begin{array}{llll} S & \Rightarrow aAbba & \Rightarrow aabbbAba & \Rightarrow aabbbbAa & \Rightarrow aabbbBbaa \\ & \Rightarrow aabbBbbaa & \Rightarrow aabBbbbaa & \Rightarrow aaBbbbbaa & \Rightarrow aaabbbAbbbaa \\ & \Rightarrow aaabbbbAbbaa & \Rightarrow aaabbbbbAbaa & \Rightarrow aaabbbbbbAaa & \Rightarrow aaabbbbbBbaaa \\ & \Rightarrow aaabbbbBbbaaa & \Rightarrow aaabbbBbbbaaa & \Rightarrow aaabbBbbbbaaa & \Rightarrow aaabBbbbbbaaa \\ & \Rightarrow aaaBbbbbbbaaa & \Rightarrow aaaAbbbbbbaaa & \Rightarrow aaabbbbbbaaa & \end{array}$$

Exemplo 9.3.2 *Uma gramática irrestrita para gerar a linguagem*

$$\mathcal{L} = \{a^n b^n a^n b^n \ / \ n \geq 1\}$$

é descrita a seguir.

$S \longrightarrow aAbab$
$aAb \longrightarrow aabbbA \ | \ ab$
$bAb \longrightarrow bbA$
$bAa \longrightarrow baB$
$aBa \longrightarrow aaB$
$aBb \longrightarrow Caabb$
$aCa \longrightarrow Caa$
$bC \longrightarrow Cb$
$aCb \longrightarrow aAb$

Exemplo 9.3.3 *Uma gramática irrestrita para gerar a linguagem*

$$\mathcal{L} = \{a^n \ / \ n \text{ é uma potência de } 2\}$$

é descrita a seguir.

$S \longrightarrow ISaaF \mid aa$
$Sa \longrightarrow aaS$
$aSF \longrightarrow BaF \mid a$
$aB \longrightarrow Ba$
$IB \longrightarrow IS$
$Ia \longrightarrow a$

As linguagens dos exemplos acima, são todas linguagens recursivas que não são livres de contexto. Exemplos de linguagens recursivamente enumerável que não são recursivas são difíceis de serem descritos (veja por exemplo o teorema 9.2.3) e no entanto, existem gramáticas irrestritas para tais linguagens, elas são altamente complexas. Como teremos oportunidade de vêr, as gramáticas irrestritas são muito mais poderosas do que outras formas mais restritas, como gramáticas regulares e livres de contexto. Realmente, as gramáticas irrestritas correspondem à maior família de linguagens que esperamos reconhecer por meios mecânicos. Ou seja, as gramáticas irrestritas geram exatamente a família das linguagens recursivamente enumeráveis.

Teorema 9.3.4 *Qualquer linguagem gerada por uma gramática irrestrita é recursivamente enumerável.*

Demonstração: A gramática define um procedimento de enumeração para toda cadeia na linguagem. Por exemplo, podemos listar toda $w \in \mathcal{L}$ tal que

$$S \Rightarrow w,$$

isto é, w é derivado em uma etapa. Como o conjunto de produções da gramática é finito, haverá um número finito de tais cadeias. Em seguida, listamos todas $w \in \mathcal{L}$ que podem ser derivadas em duas etapas

$$S \Rightarrow x \Rightarrow w,$$

e assim por diante. Podemos simular essas derivações numa máquina de Turing, e portanto temos um procedimento de enumeração para a linguagem. Portanto, ela é recursivamente enumerável. ■

Esta parte da correspondência entre linguagens recursivamente enumeráveis e gramáticas irrestritas não é surprendente. A gramática gera cadeias por um processo algorítmico bem definido. Portanto as derivações podem ser dadas por uma máquina de Turing. Para mostrar a recíproca descrevemos como qualquer máquina de Turing pode ser imitada por uma gramática irrestrita. Esta parte não é fácil. A prova é longa e muito técnica, embora a idéia subjacente seja bastante simples. Por isso, daremos somente um esboço da prova.

Teorema 9.3.5 *Para toda linguagem recursivamente enumerável, $\mathcal{L}$, existe uma gramática irrestrita G, tal que*

$$\mathcal{L} = L(G).$$

Demonstração: (Esboço) Se $\mathcal{L}$ é recursivamente enumerável, então, por definição, existe uma máquina de Turing, M, que a reconhece. Pretendemos construir uma gramática, G, tal que $L(G) = L(M)$. Isto significa que queremos ter

$$S \stackrel{*}{\Longrightarrow} w_G,$$

para exatamente aqueles w para os quais

$$q_0 w \vdash^*_M \ xq_f y.$$

Obteremos isso construindo G de tal modo que exista uma correspondência entre a sequência de descrições instantâneas de M e as formas sentenciais de G. Como M começa com w, as derivações de G devem terminar nesta cadeia. De fato, a derivação será a reversa da computação de M. Além disso, como a derivação começa com S, devemos primeiro derivar dela uma forma sentencial que represente a configuração de parada $xq_f y$, da qual, por seu turno, devemos derivar $q_0 w$. O que queremos, então, é construir G, tal que

$$S \stackrel{*}{\Longrightarrow} xq_f y \stackrel{*}{\Longrightarrow} q_0 w$$

se e somente se $w \in L(M)$. Existem algumas complicações aqui. Uma delas é que, realmente, queremos $S \stackrel{*}{\Longrightarrow} w$, e devemos nos livrar de q_0. Para isto introduzimos um símbolo especial $\#$, o qual é associado com a variável de estado na descrição instantânea, de tal modo que ela possa ser eliminada na última etapa. Uma outra complicação pode provir dos possíveis brancos nas descrições instantâneas intermediárias. Esses também devem ser eliminados pela gramática. Quando colocamos todas essas condições juntas, descobrimos que a gramática G terá de ter vários conjuntos de produções:

1. Um conjunto que permite todas as derivações da forma

   $$S \stackrel{*}{\Longrightarrow} \#xq_f y\#,$$

   para todos os x e y possíveis e todo estado final q_f, com brancos possíveis à direita e à esquerda de $\#xq_f y\#$,

2. Um conjunto que permite a derivação

   $$\#xq_f y\# \stackrel{*}{\Longrightarrow} \#q_0 w\#,$$

   sempre que $q_0 w \vdash^*_M \ xq_f y$,

3. Um conjunto para se livrar de $\#q_0$ e quaisquer possíveis brancos. ■

Exemplo 9.3.6 *Seja $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \Box, F\rangle$ uma máquina de Turing com*

- $Q = \{q_0, q_1\}$,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, 1, \Box\}$,
- $F = \{q_1\}$ *e*
- $\delta(q_0, 0) = (q_0, 0, D)$ *e* $\delta(q_0, \Box) = (q_1, \Box, E)$.

Esta máquina aceita a linguagem $L(00^*)$. Seguindo as sugestões poderemos constatar que as exigências serão satisfeitas pela gramática $G = \langle V, T, S, P\rangle$, com

- $V = \{S, A, \Box, \#, q_0, q_1\}$,
- $T = \{0, 1\}$.

O primeiro subconjunto das produções de P é

$S \longrightarrow \square S \mid S\square \mid \#A,$

$A \longrightarrow 0A \mid 1A \mid A0 \mid A1 \mid q_1.$

É fácil vêr que podemos usar essas produções para gerar qualquer cadeia $\#xq_1y$, com $x, y \in \{0,1\}^*$, precedida e seguida por um número arbitrário de brancos. O segundo conjunto de produções é

$0q_0 \longrightarrow q_00,$

$q_10\square \longrightarrow 0q_0\square,$

$q_11\square \longrightarrow 1q_0\square,$

$q_1\square\square \longrightarrow \square q_0\square.$

Essas produções refletem mudanças feitas na descrição instantânea efetuadas por δ. Por exemplo, a subcadeia q_00 toma o lugar de $0q_0$ quando a transição $\delta(q_0, 0) = (q_0, 0, D)$ é usada. Poucos testes devem nos convencer de que essas regras tornam possível a derivação $x \stackrel{*}{\Longrightarrow} y$, sempre que $y \vdash^* x$.

Finalmente, o último conjunto

$\#q_0 \longrightarrow \lambda,$

$\square \longrightarrow \lambda,$

é suficiente para se livra de $\#q_0$ e quaisquer brancos.

Olhemos, agora, a computação, em M, que aceita a cadeia 00.

$$q_000 \vdash 0q_00 \vdash 00q_0\square \vdash 0q_10\square.$$

Uma derivação correspondente, em G, dessa cadeia é

$S \Longrightarrow S\square \Longrightarrow \#A\square \Longrightarrow \#0A\square \Longrightarrow \#0A0\square \Longrightarrow \#0q_10\square \Longrightarrow \#00q_0\square \Longrightarrow$

$\#0q_00\square \Longrightarrow \#q_000\square \Longrightarrow \#q_000 \Longrightarrow 00.$

Ignorando a marca especial # e brancos, vemos que parte dessa derivação é, como alegamos, a sequência de descrições instantâneas da máquina de Turing em ordem reversa.

9.4 Linguagens Sensíveis ao Contexto

Podem serem definidas diversas famílias de linguagens formais mais abrangentes que a família das linguagens livres de contexto, porém menos geral que a família das linguagens recursivas. Cada uma destas famílias podem ser vistas como uma classe de linguagens geradas por gramáticas com tipos de produções mais gerais que as produções das gramáticas livres de contexto, mas com algum tipo de restrição ou, do ponto de vista de autômatos, como certo tipo de máquinas de Turing que seja mais poderosa que os autômatos a pilha não determinísticos. A mais conhecida e interessante destas famílias de linguagens são as **linguagens sensíveis ao contexto**, as quais são geradas pelas gramáticas sensíveis ao contexto ou, equivalentemente, reconhecidas por autômatos limitados linearmente.

9.4.1 Gramáticas Sensíveis ao Contexto

No capítulo 1, dimos a noção geral de gramática, e nos capítulos posteriores fomos restringindo a forma das produções na gramática até não se ter nenhuma restrição (gramáticas irrestritas). Uma outra possibilidade de se restringir produções numa gramática, de forma que elas sejam menos permissiva que as produções das gramáticas livres de contexto, é por um contexto.

Definição 9.4.1 *Seja $G = \langle V, T, S, P\rangle$ uma gramática irrestrita. G é uma* **gramática sensível ao contexto** *se toda produção $v \longrightarrow w$ em P, tem a seguinte propriedade: $\mid v \mid \leq \mid w \mid$, isto é, a forma sentencial w tem comprimento maior ou igual a cadeia v.*

Observe que se eliminamos λ-produções de uma gramática livre de contexto, a gramática resultante sempre será sensível ao contexto. Assim, podemos dizer que toda linguagem livre de contexto (que não contenha λ) é sensível ao contexto. Por outro lado existem gramáticas que são sensíveis ao contexto mas que não são livres de contexto. A seguir, um exemplo de gramática sensível ao contexto que gera uma linguagem que não é livre de contexto.

Exemplo 9.4.2 *Uma gramática sensível ao contexto $G = \langle V, T, S, P\rangle$, para gerar a linguagem*

$$\mathcal{L} = \{a^n b^n a^n \ / \ n \geq 1\}$$

é descrita a seguir.

1. $V = \{S, A, B, C\}$
2. $T = \{a, b\}$
3. *P é o seguinte conjunto de produções:*

$S \longrightarrow aAba \mid aba$
$aAb \longrightarrow aabbB$
$Bb \longrightarrow bB$
$Ba \longrightarrow Caa \mid aa$
$bCa \longrightarrow CbabC \longrightarrow Cb$
$aCb \longrightarrow aAb$

Observe que a gramática irrestrita do exemplo 9.3.3 é sensível ao contexto, mas as gramáticas irrestritas dos exemplos 9.3.1 e 9.3.2 não são sensíveis ao contexto, pois elas contém uma produção ($aAb \longrightarrow ab$) a qual o lado direito da produção é menor que o lado esquerdo. Porém, isso não significa que as respectivas linguagens geradas por essas gramáticas não sejam sensíveis ao contexto. De fato, as linguagens

$$\mathcal{L}_1 = \{a^n b^{2n} a^n \ / \ n \geq 1\}$$

e

$$\mathcal{L}_2 = \{a^n b^n a^n b^n \ / \ n \geq 1\}$$

são sensíveis ao contexto.

Uma gramática sensível ao contexto que gera a linguagem $\mathcal{L}_1$ é a seguinte:

$S \longrightarrow aAbba \mid abba$
$aAb \longrightarrow aabbbB$
$Bb \longrightarrow bB$
$Ba \longrightarrow Caa \mid aa$
$bCa \longrightarrow Cba$
$bC \longrightarrow Cb$
$aCb \longrightarrow aAb$

Uma gramática sensível ao contexto que gera a linguagem $\mathcal{L}_2$ é a seguinte:

$S \longrightarrow aAbab \mid abab$
$aAb \longrightarrow aabbB$
$Bb \longrightarrow bB$
$Ba \longrightarrow aC$
$aCb \longrightarrow Daabb \mid aabb$
$Da \longrightarrow aD$
$bDa \longrightarrow Eba$
$bE \longrightarrow Eb$
$aE \longrightarrow aA$

Observe que nas gramáticas livres de contexto as produções tem a seguinte forma: $A \longrightarrow w$, onde $A \in V$ e $w \in (V \cup T)^*$, a qual significava que uma ocorrência, numa forma sentencial, da variável A pode ser substituida pela cadeia w, independente do resto das componentes da forma sentencial. As gramáticas sensíveis ao contexto têm uma forma normal, daí seu nome, nas quais esta troca só será possível de realizar quando parte do que estiver de junto (na direita e na esquerda) da variável A na forma sentencial for uma determinada cadeia (seu contexto).

Definição 9.4.3 *Seja $G = \langle V, T, S, P\rangle$ uma gramática sensível ao contexto. G está na* **forma normal SC** *se todas as produções em P têm a seguinte forma:*

$$x_1 A x_2 \longrightarrow x_1 w x_2,$$

onde $A \in V$, $x_1, x_2 \in (V \cup T)^*$ e $w \in (V \cup T)^+$, ou seja w é uma forma sentencial não vazia.

Toda gramática sensível ao contexto tem uma gramática sensível ao contexto na forma normal SC.

Teorema 9.4.4 *Se $G = \langle V, T, S, P\rangle$ é uma gramática sensível ao contexto, então existe uma gramática sensível ao contexto $\widehat{G} = \langle \widehat{V}, T, S, \widehat{P}\rangle$ na forma normal SC tal que $L(G) = L(\widehat{G_1})$.*

Demonstração: A construção de $\widehat{G_1}$ será dada em duas etapas.

Etapa1: Esta etapa é similar à etapa 1 da forma normal de Chomsky. Construir a partir de G uma gramática, $G_1 = \langle V_1, T, S, P_1\rangle$, introduzindo, para cada $a \in T$, uma nova variável B_a e a produção $B_a \longrightarrow a$. Para cada produção, cujo lado esquerdo ou direito seja de comprimento maior do que um, substitua cada ocorrência de um símbolo terminal, digamos a, por sua respectiva variável associada (no caso, B_a). Faça o mesmo com as eventuais produções cujo lado esquerdo for um símbolo terminal. No fim dessa etapa, temos uma gramática G_1, cujas produções têm a forma

$$X \longrightarrow Y, \tag{9.3}$$

com $X \in V$ e $Y \in V \cup T$, ou

$$B_1 \cdots B_m \longrightarrow C_1 \cdots C_n, \tag{9.4}$$

com $m \leq n$, e $B_i, C_j \in V_1$, para cada $i = 1, \ldots, m$ e $j = 1, \ldots, n$. É fácil vêr que $L(G_1) = L(G)$ e G_1 continua sendo sensível ao contexto.

Etapa 2: Construir a partir de G uma nova gramática, $\widehat{G}$, substituindo cada produção $B_1 \cdots B_m \longrightarrow C_1 \cdots C_n$ em P_1, com $m \leq n$ que não esteja na forma normal SC, pode ser substituida pelo seguinte conjunto de produções:

$B_1 \ldots B_m \longrightarrow Z_1 B_2 \ldots B_m$

$Z_1 B_2 \ldots B_m \longrightarrow Z_1 Z_2 B_3 \ldots B_m$

$\vdots$

$Z_1 \ldots Z_{m-1} B_m \longrightarrow Z_1 \ldots Z_{m-1} C_m \ldots C_n$

$Z_1 \ldots Z_{m-1} C_m \ldots C_n \longrightarrow C_1 Z_2 \ldots Z_{m-1} C_m \ldots C_n$

$C_1 Z_2 \ldots Z_{m-1} C_m \ldots C_n \longrightarrow C_1 C_2 Z_3 \ldots Z_{m-1} C_m \ldots C_n$

$\vdots$

$C_1 \ldots C_{m-2} Z_{m-1} C_m \ldots C_n \longrightarrow C_1 \ldots C_n$

onde $Z_1, \ldots, Z_{m-1}$ são variáveis novas. Assim, claramente, estas novas produções satisfazem a forma normal SC. Logo, a gramática $\widehat{G}$, resultante, está na forma normal SC e, claramente, $L(\widehat{G}) = L(G_1) = L(G)$. ■

Observe, que uma gramática sensível ao contexto, obtida usando o algoritmo acima, tem como propriedade extra que o lado esquerdo das produções são sempre cadeias de variáveis.

Exemplo 9.4.5 *Seja a gramática sensível ao contexto do exemplo 9.4.2.*

Etapa 1: *Construção de G_1*

1. $V_1 = \{S, A, B, C, B_a, B_b\}$
2. $T = \{a, b\}$
3. *P é o seguinte conjunto de produções:*

$S \longrightarrow B_a A b B_b \mid B_a B_b B_a$
$B_a A b \longrightarrow B_a B_a B_b B_b B$
$B B_b \longrightarrow B_b B$
$B B_a \longrightarrow C B_a B_a \mid B_a B_a$
$B_b C B_a \longrightarrow C B_b B_a$
$B_b C \longrightarrow C B_b$
$B_a C B_b \longrightarrow B_a A B_b$

Etapa 2: *Construção de* $\widehat{G}$

1. $\widehat{V} = \{S, A, B, C, B_a, B_b, \}$
2. $T = \{a, b\}$
3. *P é o seguinte conjunto de produções:*

$S \longrightarrow B_aAbB_b \mid B_aB_bB_a$ *já está na forma normal SC*

$B_aAB_b \longrightarrow Z_1AB_b$ *transformando a regra:* $B_aAB_b \longrightarrow B_aB_aB_bB_b$
$Z_1AB_b \longrightarrow Z_1Z_2B_b$
$Z_1Z_2B_b \longrightarrow Z_1Z_2B_bB_b$
$Z_1Z_2B_bB_b \longrightarrow B_aZ_2B_bB_b$
$B_aZ_2B_bB_b \longrightarrow B_aB_aB_bB_b$

$BB_b \longrightarrow Z_3B_b$ *transformando a regra:* $BB_b \longrightarrow B_bB$
$Z_3B_b \longrightarrow Z_3B$
$Z_3B \longrightarrow B_bB$

$BB_a \longrightarrow CB_aB_a \mid B_aB_a$ *já estão na forma normal SC*

$B_bCB_a \longrightarrow Z_4CB_a$ *transformando a regra:* $B_bCB_a \longrightarrow CB_bB_a$
$Z_4CB_a \longrightarrow Z_4Z_5B_a$
$Z_4Z_5B_a \longrightarrow Z_4Z_5B_a$
$Z_4Z_5B_a \longrightarrow CZ_5B_a$
$CZ_5B_a \longrightarrow CB_bB_a$

$B_bC \longrightarrow Z_6C$ *transformando a regra:* $B_bC \longrightarrow CB_b$
$Z_6C \longrightarrow Z_6B_b$
$Z_6B_b \longrightarrow CB_b$

$B_aCB_b \longrightarrow B_aAB_b$ *já está na forma normal SC*

9.4.2 Autômatos Limitados Linearmente

Um autômato limitado linearmente é, simplesmente, uma máquina de Turing não determinística a qual no lugar de ter uma fita potencialmente infinita sob a qual computa, ela é restrita à porção da fita que contém a entrada mais dois quadrados extras, um em cada extremo, contendo marcadores de início e fim de fita, respectivamente. Assim, um **autômato limitado linearmente (ALL)** é uma máquina de Turing satisfazendo as seguintes consições:

1. O alfabeto de entrada contém dois símbolos especiais $*$ e \$, usados como marcadores do início e fim da fita, respectivamente.
2. Não existem movimentos à esquerda de $*$ nem à direita de \$
3. Nada pode ser escrito sobre estes símbolos.

Definição 9.4.6 *Um* **autômato limitado linearmente** *(ALL), é uma oito-tupla* $M = \langle Q, \Sigma, \Gamma, \delta, q_0, *, \$, F\rangle$*, tal que*

- $\langle Q, \Sigma, \Gamma, \delta, q_0, \Box, F\rangle$ *é uma máquina de Turing padrão (definição 8.1.1),*
- $*$ *e* \$ *são símbolos especiais de* Σ*, chamados* **marcador do início da fita** *e* **marcador do final da fita**, *respectivamente, e*
- *A função de transição* $\delta : Q \times \Sigma \longleftrightarrow \wp(Q \times \Sigma \times \{D, E\})$ *tem as seguintes restrições para todo* $q \in Q$ *e* $a \in \Sigma - \{*, \$\}$*:*
  - $\delta(q, *) \subseteq Q \times \{*\} \times \{D\}$
  - $\delta(q, \$) \subseteq Q \times \{\$\} \times \{E\}$
  - $\delta(q, a) \subseteq Q \times (\Sigma - \{*\$\}) \times \{D, E\}$

Como todo autômato, um ALL reconhece uma linguagem, mas neste caso a linguagem reconhecida será um subconjunto de $\Sigma - \{*, \$\}$ em vez de um subconjunto de Σ, pois os símbolos $*$ e \$ são meros marcadores. Assim, a linguagem reconhecida por um ALL $M = \langle Q, \Sigma, \Gamma, \delta, q_0, *, \$, F\rangle$ é o conjunto

$$L(M) = \{w \in (\Sigma - \{*, \$\})^* \ / \ q_0 * w\$ \vdash^* uq_f v \text{ para algum } q_f \in F\}$$

Exemplo 9.4.7 *Um ALL que reconhece a linguagem* $\mathcal{L} = \{a^n b^n a^n \ / \ n \geq 1\}$*, é a oito-tupla* $M = \langle Q, \Sigma, \Gamma, \delta, q_0, *, \$, F\rangle$*, tal que*

- $Q = \{q_0, \ldots, q_8\}$
- $\Sigma = \{a, b, *, \$, \Box\}$
- $\Gamma = \{a, b, , \$, \Box, x, y, z\}$
- $F = \{q_8\}$
- δ *é definido através da seguinte tabela:*

	a	b	$*$	\$	x	y	z	$\Box$
q_0			$(q_1, *, D)$					
q_1	(q_2, x, D)	(q_5, x, D)			(q_1, x, D)		(q_8, Z, E)	
q_2	(q_2, a, D)	(q_2, b, D)		$(q_3, \$, E)$		(q_3, y, E)		
q_3	(q_4, y, E)							
q_4	(q_4, a, E)	(q_4, b, E)			(q_1, x, D)			
q_5		(q_5, b, D)		$(q_6, \$, E)$		(q_5, y, D)	(q_6, z, E)	
q_6						(q_7, z, E)		
q_7		(q_7, b, E)			(q_1, x, D)	(q_7, y, E)		
q_8								

Observe que os marcadores estão no início da fita de entrada, mas não são considerados como partes das palavras aceitas ou rejeitadas pelo autômato. Por outro lado, ALL's não podem acessar células à esquerda de $*$ nem à direita de \$, tornando inúteis os símbolos em brancos à esquerda e à direita do conteúdo inicial da fita de entrada, usuais em máquinas de Turing padrão. Por esse motivo, $\Box$ não foi considerado explicitamente como componente da oito-tupla na definição de ALL (embora implicitamente seja considerado ao se requerer que uma certa sete-tupla seja uma máquina de Turing) e, portanto, não temos necessidade de supor que existe uma fita em branco à direita de \$ nem à esquerda de $*$. Logo, podemos considerar que a fita de entrada começa em $*$ e termina em \$ e eliminar completamente da definição de ALL a ocorrência do símbolo branco ($\Box$).

Exemplo 9.4.8 *Um ALL que reconhece a linguagem* $\mathcal{L}_1 = \{a^n b^n a^n b^n \ / \ n \geq 1\}$ *é descrito na seguinte tabela:*

	a	b	$*$	\$	x
q_0			$(q_1, *, D)$		
q_1	(q_2, x, D)	(q_5, x, D)			(q_1, x, D)
q_2	(q_2, a, D)	(q_2, b, D)		$(q_3, \$, E)$	(q_3, x, E)
q_3		(q_4, x, E)			
q_4	(q_4, a, E)	(q_4, b, E)			(q_1, x, D)
q_5	(q_5, a, D)	(q_5, b, D)			(q_6, x, E)
q_6	(q_7, x, E)				
q_7	(q_7, a, E)	(q_7, b, E)			(q_8, x, D)
q_8		(q_5, x, D)			(q_9, X, D)
q_9					

O estado final deste ALL é o estado q_9.

Proposição 9.4.9 *Seja* $M = \langle Q, \Sigma, \Gamma, \delta, q_0, *, \$, F\rangle$ *um ALL. Então existe um ALL* $M' = \langle Q', \Sigma, \Gamma', F'\rangle$ *tal que* $L(M) = L(M')$ *e*

$$w \in L(M') \text{ se, e somente se, } q_0 * w\$ \vdash^*_{M'} q_f * w\$$$

Demonstração: (Argumento) Como a fita é limitada ao tamanho da entrada, nunca adicionaremos símbolos extras à fita, o máximo que pode acontecer é substituir os símbolos da entrada por outros. Se ao fizermos isto, tivermos o cuidado de substituir cada símbolo $a \in \Sigma - \{*, \$\}$ por um símbolo especial associado a ele, então ao final da computação, poderemos re-escrever a entrada original na fita e nos posicionar ao início da mesma. ■

Exemplo 9.4.10 *O ALL do exemplo anterior, pode ser re-escrito nesta forma normal, da seguinte maneira:*

	a	b	$*$	\$	x	y
q_0			$(q_1, *, D)$			
q_1	(q_2, x, D)	(q_5, y, D)			(q_1, x, D)	
q_2	(q_2, a, D)	(q_2, b, D)		$(q_3, \$, E)$		(q_3, y, E)
q_3		(q_4, y, E)				
q_4	(q_4, a, E)	(q_4, b, E)			(q_1, x, D)	
q_5	(q_5, a, D)	(q_5, b, D)			(q_6, x, E)	(q_6, y, E)
q_6	(q_7, x, E)					
q_7	(q_7, a, E)	(q_7, b, E)				(q_8, y, D)
q_8		(q_5, y, D)			(q_9, x, D)	
q_9				$(q_{10}, \$, E)$	(q_9, x, D)	(q_9, y, D)
q_{10}			$(q_{11}, *, D)$		(q_{10}, a, E)	(q_{10}, b, E)
q_{11}	(q_{12}, a, E)	(q_{12}, b, E)				
q_{12}						

O estado final deste ALL é o estado q_{12}.

9.4.3 Relação entre Linguagens Recursivas e Linguagens Sensíveis ao Contexto

Como toda gramática sensível ao contexto é uma gramática irrestrita, então toda linguagem sensível ao contexto é uma linguagem recursiva enumerável. Menos evidente é mostrar que toda linguagem sensível ao contexto é recursiva, pois ALL podem não parar para toda entrada.

Teorema 9.4.11 *Toda linguagem sensível ao contexto é recursiva.*

Demonstração: (Esboço) Seja $\mathcal{L}$ uma linguagem sensível ao contexto e G uma gramática sensível ao contexto, tal que $L(G) = \mathcal{L}$. Analogamente ao teorema 9.3.4, podemos simular as derivações de G numa máquina de Turing, definindo um procedimento de enumeração para toda cadeia na linguagem. Por exemplo, podemos listar toda $w \in \mathcal{L}$ tal que

$$S \Rightarrow w,$$

isto é, w é derivado em uma etapa. Como o conjunto de produções da gramática é finito, haverá um número finito de tais cadeias. Em seguida, listamos todas $w \in \mathcal{L}$ que podem ser derivadas em duas etapas

$$S \Rightarrow x \Rightarrow w,$$

e assim por diante. Como as gramáticas sensíveis ao contexto têm a propriedade que a cada passo o tamanho da cadeia gerada aumenta, então para qualquer $u \in T^*$, este processo sempre pára, ou gerando u, neste caso $u \in \mathcal{L}$, ou gerando uma forma sentencial maior que $|\, u \,|$, o qual significa que a gramática nunca gerará u e, portanto, $u \notin \mathcal{L}$. Logo, como, temos uma máquina de Turing que sempre pára e reconhece $\mathcal{L}$, então $\mathcal{L}$ é recursiva. ■

Por outro lado, nem toda linguagem recursiva é sensível ao contexto, para demonstrar isto precisaríamos descrever uma linguagem e mostrar que ela é, efetivamente, recursiva mas não é sensível ao contexto. Linguagens recursivas que não sejam sensíveis ao contexto, e que sejam intuitivas são difíceis de se encontrarem, embora existam infinitas de tais linguagens. Uma demonstração elegante da existência de uma tal linguagem, pode ser encontrada em [Sal73, HU79]. No entanto, nestas demonstrações se constroe tal linguagem recursiva a partir de uma enumeração de um conjunto de máquinas de Turing que sempre param. Um exemplo mais tangível de uma tal linguagem, pode ser obtida via uma codificação do gráfico da função de Ackermann $A : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, definida por

$A(0, y) = y + 1$

$A(x + 1, 0) = A(x, 1)$

$A(x + 1, y + 1) = A(x, A(x + 1, y))$.

Esta função de dupla recursão, foi introduzida por Wilhelm Ackermann no intuito de demonstrar, construtivamente, a existência de uma função recursiva[1] que não é primitiva recursiva[2], isto é, uma função que seja computada por uma máquina de Turing que sempre pára, mas que não é computada por uma máquina limitada linearmente. Existem diversas versões da função de Ackermann, a dada aqui é a mais usual e pode ser encontrada em [BL74, Cut80] entre outros, outras definições podem ser encontradas em [HU79, Bir96].

[1] As funções recursivas são a sub-classe das funções parciais recursivas, ou analogamente as funções Turing-computáveis, que são totais, ou noutras palavras para as quais existe uma máquina de Turing que a computa e que sempre pára [BL74, Cut80, Smi94, CO98].

[2] A classe das funções primitivas recursivas são uma subclasse própria das funções recursivas.

Uma tal codificação poderia ser a linguagem $L_{Ack} \subseteq \{0,1\}^*$ descrita por

$$L_{Ack} = \{w_1 0 w_2 0 w_3 \ / \ , w_1, w_2, w_3 \in \{1\}^* \text{ e } A(|\,w_1\,|, |\,w_2\,|) = |\,w_3\,|\},$$

onde $|\,w\,|$ é o tamanho da cadeia w.

9.4.4 Equivalência entre ALL's e Gramáticas Sensíveis ao Contexto

Mostraremos que a não ser pelo fato que um ALL pode aceitar a palavra vazia, λ, enquanto uma gramática sensível ao contexto não, ambos (ALL's e gramáticas sensíveis ao contexto) aceitão, exatamente, a mesma classe de linguagens, ou seja as linguagens sensíveis ao contexto.

Teorema 9.4.12 *Se $\mathcal{L}$ é uma linguagem sensível ao contexto, então existe um ALL M que aceita $\mathcal{L}$, isto é $L(M) = \mathcal{L}$.*

Demonstração: (Veja [HU79]). ■

Observe que, linguagens sensíveis ao contexto não consideram λ, pois para uma gramática gerar a cadeia vazia λ, precisaria da produção $S \longrightarrow \lambda$ a qual fere a propriedade que o lado esquerdo seja menor que lado direito de uma produção , e por tanto não seria sensível ao contexto. Por outro lado linguagens aceitas por ALL's não têm esta restrição. Isto não é um grande problema, pois se quisessemos considerar linguagens com a cadeia vazia, bastaria flexibilizar a definição de gramática sensível ao contexto, impondo que toda produção $w \longrightarrow v$, tenha $|\,w\,| \geq |\,v\,|$ ou $|\,v\,| = 0$.

Teorema 9.4.13 *Seja M um ALL, então $L(M) - \{\lambda\}$ é uma linguagem sensível ao contexto.*

Demonstração: (Veja [HU79]). ■

9.4.5 Propriedades de Fecho das Linguagens Sensíveis ao Contexto

As linguagens sensíveis ao contexto são fechadas sobre a maioria das operações usuais sobre linguagens: união, interseção, concatenção, fecho positivo e imagem homomorfa.

Proposição 9.4.14 *Sejam $\mathcal{L}_1$ $\mathcal{L}_2$ linguagens sensíveis ao contexto. Então as linguagens:*

1. $\mathcal{L}_1 \cup \mathcal{L}_2$,
2. $\mathcal{L}_1 \cap \mathcal{L}_2$,
3. $\mathcal{L}_1 \circ \mathcal{L}_2$,
4. $\mathcal{L}_1^+ = \bigcup_{i=1} \mathcal{L}_1^i$
5. $h(\mathcal{L}_1)$, *para qualquer homomorfismo* $h : T_1 \longrightarrow T_2^*$

são sensíveis ao contexto.

Demonstração: Sejam $G_1 = \langle V_1, T_1, S_1, P_1 \rangle$ e $G_2 = \langle V_2, T_2, S_2, P_2 \rangle$ gramáticas sensíveis ao contexto tais que $\mathcal{L}_1 = L(G_1)$ e $\mathcal{L}_2 = L(G_2)$ e $V_1 \cap V_2 = \emptyset$.

1. Análogo ao caso das linguagens regulares e livres de contexto, a união das linguagens $\mathcal{L}_1$ e $\mathcal{L}_2$ pode ser obtida simplesmente juntando as duas gramáticas e adicionando uma nova variável de início S e as produções:

$$S \longrightarrow S_1 \mid S_2.$$

   Assim, a nova gramática

$$G_1 \cup G_2 = \langle V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1, S \longrightarrow S_2\}\rangle,$$

   claramente, é sensível ao contexto e gera a linguagem:

$$L(G_1 \cup G_2) = \mathcal{L}_1 \cup \mathcal{L}_2.$$

2. Sejam $M_1 = \langle Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^1, *, \$, F_1\rangle$ e $M_2 = \langle Q_2, \Sigma_2, \Gamma_2, \delta_2, q_0^2, *, \$, F_2\rangle$ dois ALL's, tais que $L(M_1) = \mathcal{L}_1$ e $L(M_2) = \mathcal{L}_2$. Pela proposição 9.4.9, é possível construir ALL's M_1' e M_2', tais que $L(M_1') = L(M_1)$, $L(M_2') = L(M_2)$, para cada $w \in L(M_1')$ se, e somente se, $q_0 * w\$ \vdash^*_{M_1'} q_f * w\$$, para algum $q_f \in F_1'$ e, analogamente, para cada $w \in L(M_2')$ se, e somente se, $q_0 * w\$ \vdash^*_{M_2'} q_f * w\$$, para algum $q_f \in F_2'$. Sem perca alguma, podemos pensar que $Q_1' \cap Q_2' = \emptyset$. A partir, dessas duas máquinas, podemos construir uma terceira $M_3 = \langle Q_1' \cup Q_2', \Sigma_1 \cup \Sigma_2, \Gamma_1 \cup \Gamma_2, \delta_3, q_0^1', *, \$, F_2\rangle$, onde $\delta_3 : (Q_1' \cup Q_2') \times (\Gamma_1 \cup \Gamma_2) \longrightarrow (Q_1' \cup Q_2') \times (\Gamma_1 \cup \Gamma_2) \times \{D, E\}$ é definida por

$$\delta_3(q,a) = \begin{cases} \delta_1(q,a) & \text{, se } q \in Q_1' \\ \delta_2(q,a) & \text{, se } q \in Q_2' \end{cases}$$

   Claramente M_3 aceitará um w se, e somente se, w é aceito por M_1' e por M_2', assim,

$$\begin{aligned} L(M_3) &= L(M_1') \cap L(M_2') \\ &= L(M_1) \cap L(M_2) \\ &= \mathcal{L}_1 \cap \mathcal{L}_2 \end{aligned}$$

3. Sejam $\widehat{G_1}$ e $\widehat{G_2}$ as gramáticas sensíveis ao contexto na forma normal SC, obtidas a partir de G_1 e G_2, respectivamente. Construa a gramática sensível ao contexto G_3 como

$$G_3 = \langle \widehat{V_1} \cup \widehat{V_2} \cup \{S\}, T_1 \cup T_2, \widehat{P_1} \cup \widehat{P_2} \cup \{S \longrightarrow S_1S_2\}, S\rangle.$$

   Claramente, G_3 é uma gramática sensível ao contexto na forma normal SC tal que $L(G_3) = L(G_1) \circ L(G_2)$. Observe, que as restrições colocadas por [HU79] para construir esta gramática não acontecem aqui, pois nos assumimos que as gramáticas G_1 e G_2 estão na forma normal SC via o algoritmo apresentado na demonstração do teorema 9.4.4, no qual sempre o lado esquerdo de uma produção é constituída só de variáveis.

4. Seja gramática $G = \langle V_1 \cup \{S\}, T_1, S, P_1 \cup \{S \longrightarrow S_1S \mid S_1\}\rangle$, para algum $S \notin V_1$. Trivialmente, G_1 é uma gramática sensível ao contexto tal que $L(G) = L(G_1)$

5. Seja a gramática $G = \langle V_1, T_2, S_1, P\rangle$, onde P é obtido substituindo em cada produção $w \longrightarrow v$ os símbolos terminais por sua imagem homomorfa, isto é, se o símbolo terminal a ocorre em w ou em v, então substituimos cada ocorrência de a, por $h(a)$. Claramente, G contínua sendo sensível ao contexto e $L(G) = L(G_1)$.

■

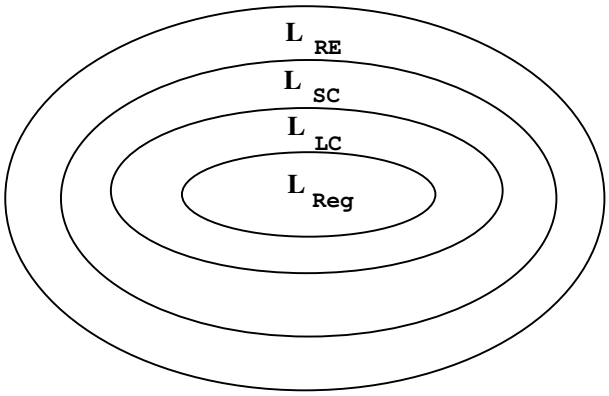

Figura 9.2: Hierarquia de Chomsky original.

9.5 A Hierarquia de Chomsky

Vimos a família das linguagens recursivamente enumeráveis ($\mathcal{L}_{RE}$), a família das linguagens recursivas ($\mathcal{L}_{Rec}$), a família das linguagens sensíveis ao contexto ($\mathcal{L}_{SC}$), a família das linguagens livres de contexto ($\mathcal{L}_{LC}$) e a família das linguagens regulares ($\mathcal{L}_{Reg}$). Uma maneira de exibir as relações entre essas famílias é através da **hierarquia de Chomsky**. Noam Chomsky, um dos fundadores da teoria das linguagens formais, forneceu uma classificação inicial em quatro tipos de linguagens do tipo 0 ao tipo 3. Essa terminologia inicial persistiu e se encontra frequentemente referências a ela. O tipo 0, o mais abrangente, é a família das linguagens geradas pelas gramáticas irrestritas, isto é, recursivamente enumeráveis. O tipo 1 consiste das linguagens sensíveis ao contexto ($\mathcal{L}_{SC}$) que são geradas por gramática sensíveis ao contexto. O tipo 2 consiste das linguagens livres de contexto e o tipo 3 das linguagens regulares. Cada família de tipo i é um subconjunto próprio da família de tipo $i-1$. O diagrama da figura 9.2 exibe, claramente, essa relação.

Podemos, também, acrescentar a essa hierarquia otras famílias de linguagens formais, apresentadas aqui de modo superficial, tais como: a família das linguagens recursivas ($\mathcal{L}_{Rec}$), a família das linguagens lineares, ($\mathcal{L}_{Lin}$), isto é as linguagens geradas por gramáticas lineares, e a família das linguagens livres de contexto determinísticas ($\mathcal{L}_{LCD}$), isto é, linguagens livres de contextos reconhecidas por autômatos a pilha determinísticos. Os níveis no diagrama de Venn da figura 9.3 indicam que a respectiva classe está contida propriamente na outra. Assim, a família das linguagens recursivamente enumeráveis contém propriamente a família das linguagens recursivas, um exemplo de uma linguagem recursivamente enumerável não recursiva é a linguagem descrita no teorema 9.2.3; a família das linguagens recursivas contém propriamente a família das linguagens sensíveis ao contexto, um exemplo de uma linguagem recursiva que não é sensível ao contexto é a linguagem L_{Ack}; a família das linguagens sensíveis ao contexto contém propriamente as linguagens livres de contexto, um exemplo de uma linguagem sensível ao contexto que não é livre de contexto é $\mathcal{L} = \{a^n b^n a^n \ / \ n \geq 0\}$; a família das linguagens livres de contexto contém propriamente a família das linguagens linguagens lineares, por exemplo a linguagem $\mathcal{L} = \{w \in \{a,b\}^* \ / \ \mathcal{N}_a(w) = \mathcal{N}_b(w)\}$ é livre de contexto mais não é linear; a família das linguagens lineares contém a família das linguagens livres de contexto deterministicas, um exemplo de uma linguagem linear não livre de contexto determinística é a linguagem dos palíndromos; e finalmente a família das linguagens livres de contexto determinísticas contém a família das linguagens regulares, um exemplo de uma linguagem livre de contexto determinística não regular é a linguagem $\mathcal{L} = \{a^n b^n \ / \ n \geq 0\}$.

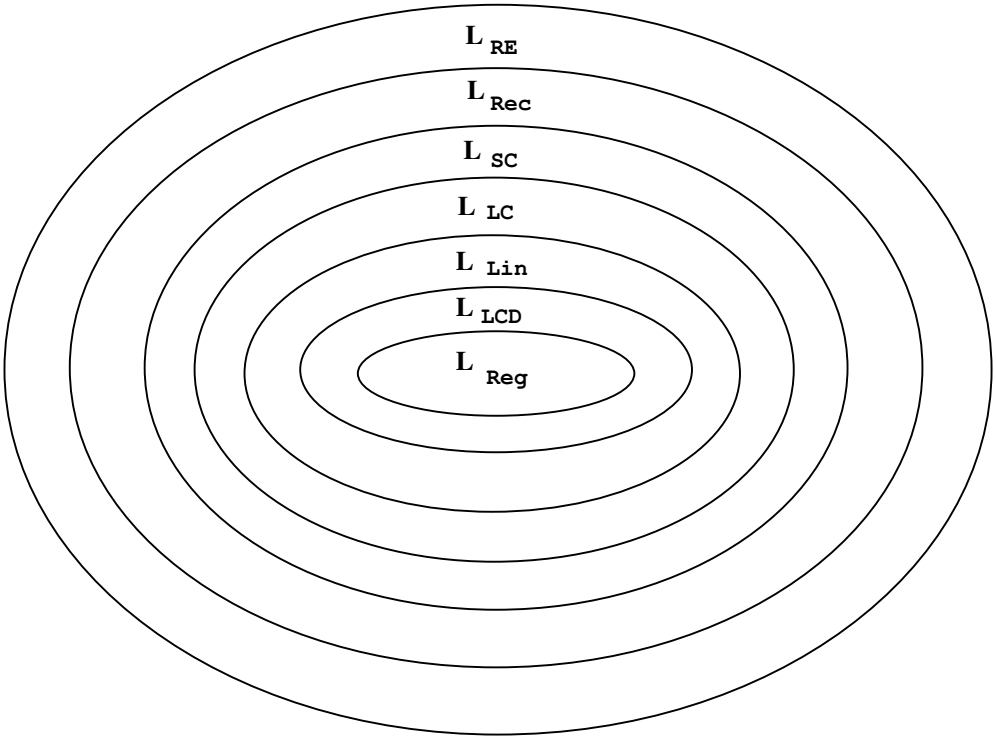

Figura 9.3: Hierarquia de Chomsky.

9.6 Complexidade Computacional

Em nosso estudo dos processos algoritmos que consideramos até aqui levamos em conta, somente, a possibilidade ou impossibilidade de computações específicas. Não encontramos diferenças essenciais entre os vários modelos de máquinas de Turing. Poderemos, sempre, usar simulações para mudar de um modelo para outro de modo que a decibilidade e a computabilidade não sejam afetados pelos detalhes da máquina de Turing que usamos. Isso, entretanto, ignora alguns fatos que se tornam relevantes quando são considerados aplicações, tais como exigências de tempo e espaço de computações em diferentes modelos. Na prática, estes elementos são muito importantes. Assim, nossos modelos teriam meios de incorporá-los quando levamos em consideração. Isto nos leva a um novo tópico, chamado **teoria da complexidade**. No estudo de complexidade, nosso interesse principal é com eficiência de uma computação como medida para requisitos de recursos.

A teoria da complexidade computacional é um tópico muito extenso, muitos dos quais estão inteiramente fora do nosso curso. Entretanto, existem alguns resultados que podem ser estabelecidos de modo simples, trazendo alguns esclarecimentos sobre a natureza de linguagens e computações. Daremos, abaixo, um esboço de alguns resultados de complexidade. Não faremos a maior parte das provas difíceis, recomendando, [HU79, Pap94, Smi94], ao leitor interessado. Nossa intenção, aqui, é apresentar a relevância do assunto e mostrar como ele se relaciona com o que sabemos sobre linguagens e autômatos.

9.6.1 Medida e Complexidade

Na prática da computação, a questão fundamental sobre uma função f, não é se f é computável, mas se f é computável em termos úteis, isto é, se existe um programa que compute f no tempo e espaço