

**exemplo 2.9** – Linguagem formal

- a** O conjunto vazio  $\emptyset$  e o conjunto formado pela palavra vazia  $\{\epsilon\}$  são linguagens sobre qualquer alfabeto. Obviamente, vale que:

$$\emptyset \neq \{\epsilon\}$$

- b** Os conjuntos  $\Sigma^*$  e  $\Sigma^+$  são linguagens sobre um alfabeto  $\Sigma$  qualquer. Obviamente, vale que:

$$\Sigma^* \neq \Sigma^+$$

- c** Suponha o alfabeto  $\Sigma = \{a, b\}$ . Então, o conjunto de *páindromos* (palavras que têm a mesma leitura da esquerda para a direita e vice-versa) sobre  $\Sigma$  é um exemplo de linguagem infinita. Assim, são palavras desta linguagem:

$$\epsilon, a, bb, aaa, aba, bab, bbb, aaaa, \dots$$

- exemplo 2.10** – Conjunto de todas as linguagens sobre um alfabeto
- O conjunto de todas as linguagens sobre um alfabeto  $\Sigma$  é o conjunto das partes de  $\Sigma^*$ , ou seja:

$$2^{\Sigma^*}$$

- exemplo 2.11** – Linguagem formal: linguagem de programação
- Uma linguagem de programação como Pascal é formalmente definida pelo conjunto de todos os programas (palavras) da linguagem.

## 2.4 → gramática

Já foi dito que uma linguagem de programação é formalmente definida pelo conjunto de todos os programas (palavras) da linguagem. Como, em geral, o conjunto de todos os programas de uma linguagem de propósitos gerais como Pascal é infinito, não é uma definição adequada para ser implementada em computador. Uma maneira de especificar de forma finita linguagens (eventualmente) infinitas é usando o formalismo *gramática*.

Uma gramática é, basicamente, um conjunto *finito* de regras as quais, quando aplicadas sucessivamente, geram palavras. O conjunto de todas as palavras geradas por uma gramática define a linguagem. As gramáticas usadas para as linguagens naturais como português são as mesmas que as usadas para linguagens artificiais como Pascal. Eventualmente, gramáticas também são usadas para definir *semântica* de linguagens. Entretanto, para tratar semântica, em geral, são usados outros formalismos.

**definição 2.9** – Gramática

Uma *gramática de Chomsky*, *gramática irrestrita* ou *simplesmente gramática* é uma quádrupla ordenada:

$$G = (V, T, P, S)$$

na qual:

- a**  $V$ , um conjunto finito de símbolos variáveis ou não terminais;
- b**  $T$ , um conjunto finito de símbolos terminais disjunto de  $V$ ;
- c**  $P: (V \cup T)^+ \rightarrow (V \cup T)^*$  é uma relação finita (ou seja,  $P$  é um conjunto finito de pares), denominada de *relação de produção* ou simplesmente *produções*. Cada par da relação é denominado de *regra de produção* ou simplesmente de *produção*;
- d**  $S$ , um elemento distinguido de  $V$  denominado *símbolo inicial* ou *variável inicial*.

Uma regra de produção  $(\alpha, \beta)$  é representada como segue:

$$\alpha \rightarrow \beta$$

Por simplicidade, um grupo de regras de produção da forma:

$$\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_n$$

(mesma componente no lado esquerdo) é usualmente abreviada como:

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

As regras de produção definem as condições de geração das palavras da linguagem. A aplicação de uma regra de produção é denominada *derivação* de uma palavra e é formalmente definida como um par de uma regra. A aplicação sucessiva de regras de produção (fecho transitivo da relação de derivação) permite derivar as palavras da linguagem representada pela gramática.

**definição 2.10** – Relação de derivação

Seja  $G = (V, T, P, S)$  uma gramática. Uma *derivação* é um par da *relação de derivação* denotada por  $\Rightarrow$  com domínio em  $(V \cup T)^+$  e codomínio em  $(V \cup T)^*$ . Um par  $\langle \alpha, \beta \rangle$  da relação de derivação é representado de forma infixada como segue:

$$\alpha \Rightarrow \beta$$

A relação de derivação  $\Rightarrow$  é indutivamente definida como segue:

- a** Para toda produção da forma  $S \rightarrow \beta$  ( $S$  é o símbolo inicial de  $G$ ), o seguinte par pertence à relação de derivação:

$$S \Rightarrow \beta$$

- b** Para todo par  $\eta \Rightarrow \rho$   $\alpha$  da relação de derivação, se  $\alpha \Rightarrow \beta$  é regra de  $P$ , então o seguinte par também pertence à relação de derivação:

$$\eta \Rightarrow \rho \beta \sigma$$

Portanto, uma derivação é a substituição de uma subpalavra de acordo com uma regra de produção.

Sucessivos passos de derivação são definidos como segue:

- $\Rightarrow^*$  fecho transitivo e reflexivo da relação  $\Rightarrow$ , ou seja, zero ou mais passos de derivações sucessivos;

$\Rightarrow^+$  fecho transitivo da relação  $\Rightarrow$ , ou seja, um ou mais passos de derivações sucessivos;  
 $\Rightarrow^i$  exatos i passos de derivações sucessivos, sendo i um número natural.

Gramáticas são consideradas formalismos de geração, pois permitem derivar ("gerar") todas as palavras da linguagem que representam.

### definição 2.11 – Linguagem gerada

Seja  $G = (V, T, P, S)$  uma gramática. A *linguagem gerada* pela gramática  $G$ , denotada por  $L(G)$  ou  $GERA(G)$ , é composta por todas as palavras de símbolos terminais deriváveis a partir do símbolo inicial  $S$ , ou seja:

$$L(G) = \{ w \in T^* \mid S \Rightarrow^+ w \}$$

**exemplo 2.12 – Gramática, derivação, linguagem gerada: números naturais**

Suponha que se deseja definir uma gramática capaz de gerar qualquer número natural válido em uma linguagem de programação. Assim, a gramática  $G = (V, T, P, N)$  na qual:

$$V = \{ N, D \}$$

$$T = \{ 0, 1, 2, \dots, 9 \}$$

$$P = \{ N \rightarrow D, N \rightarrow DN, D \rightarrow 0 \mid 1 \mid \dots \mid 9 \}$$

gera, sintaticamente, o conjunto dos números naturais. Note-se que se distinguem os zeros à esquerda. Por exemplo, distingue-se 123 de 0123 (sugere-se como exercício o desenvolvimento de uma gramática a qual não distingue zeros à esquerda). Como ilustração, uma derivação do número 243 é como segue (na coluna à direita, é apresentada a regra usada em cada passo de derivação):

$$\begin{array}{ll} N \Rightarrow & N \rightarrow DN \\ DN \Rightarrow & DN \rightarrow D \\ 2N \Rightarrow & 2N \rightarrow 2 \\ 2DN \Rightarrow & 2DN \rightarrow 4 \\ 24N \Rightarrow & 24N \rightarrow D \\ 24D \Rightarrow & 24D \rightarrow 3 \\ 243 \Rightarrow & \end{array}$$

Existe mais alguma derivação do número 243? Portanto, pode-se indicar que:

$$\begin{array}{l} S \Rightarrow^* 243 \\ S \Rightarrow^+ 243 \\ S \Rightarrow 243 \end{array}$$

Observe que, no exemplo acima, a seguinte interpretação indutiva pode ser dada à gramática em questão:

■ *Base de indução*: todo dígito é um número natural;

■ *Passo de indução*: se  $n$  é um número natural, então a concatenação de  $n$  com qualquer dígito também é um número natural.

O exemplo que segue pode parecer relativamente complexo para um segundo exemplo de gramática. O objetivo é apenas apresentar algumas das potencialidades que serão exploradas ao longo de todo o livro. De fato, nos capítulos subsequentes, são mostrados, em níveis crescentes de dificuldade, muitos outros exemplos de gramáticas. Neste ponto, alguma dificuldade de entendimento, ou de resolução dos exercícios propostos, é normal.

**exemplo 2.13 – Gramática, derivação, linguagem gerada: palavra duplicada**

A seguinte gramática:

$$G = \{ S, X, Y, A, B, F \}, \{ a, b \}, P, S \}$$

na qual:

$$\begin{aligned} P = \{ & S \rightarrow XY \\ & X \rightarrow XaA \mid XbB \mid F \\ & Aa \rightarrow aA, Ab \rightarrow bA, AY \rightarrow Ya, \\ & Ba \rightarrow aB, Bb \rightarrow bB, BY \rightarrow Yb, \\ & Fa \rightarrow aF, Fb \rightarrow bF, FY \rightarrow \epsilon \} \end{aligned}$$

gera a linguagem cujas palavras são tão tais que a primeira metade é igual à segunda metade:

$$\{ ww \mid w \text{ é palavra de } \{ a, b \}^* \}$$

Como ilustração, uma derivação da palavra **baba** é como segue (na coluna à direita, é apresentada a regra usada em cada passo de derivação):

$$\begin{array}{ll} S \Rightarrow & S \rightarrow XY \\ & XY \Rightarrow \\ & \quad XaAY \Rightarrow \\ & \quad XaYa \Rightarrow \\ & \quad XbBaYa \Rightarrow \\ & \quad XbaBYa \Rightarrow \\ & \quad XbaYba \Rightarrow \\ & \quad FbaYba \Rightarrow \\ & \quad bFaYba \Rightarrow \\ & \quad baFYba \Rightarrow \\ & \quad baba \end{array}$$

Existe mais alguma derivação da palavra **baba**? A gramática apresentada gera o primeiro  $w$  após  $X$  e o segundo  $w$  após  $Y$ , como segue:

- a cada símbolo terminal gerado após  $X$ , é gerada uma variável correspondente;
- esta variável "caminha" na palavra até passar por  $Y$ , quando deriva o correspondente terminal;
- para encerrar,  $X$  deriva a variável  $F$ , a qual "caminha" até encontrar  $Y$ , quando deriva a palavra vazia. Lembre-se:  $\epsilon$  é o elemento neutro da concatenação e, portanto,  $baba = baba$ .

□

**definição 2.12** – Gramáticas equivalentes  
Duas gramáticas  $G_1$  e  $G_2$  são ditas *gramáticas equivalentes* se e somente se:

$$GERA(G_1) = GERA(G_2)$$

No texto que segue, frequentemente são usadas as seguintes convenções:

- $A, B, C, \dots, S$ , T para símbolos variáveis;
- $a, b, c, \dots, s, t$  para símbolos terminais;
- $u, v, w, x, y, z$  para palavras de símbolos terminais;
- $\alpha, \beta, \dots$  para palavras de símbolos variáveis ou terminais.

## 2.5

### → exercícios

**exercício 2.1** Marque os conjuntos que são alfabetos:

- a** Conjunto dos números inteiros [ ]
- b** Conjunto dos números primos [ ]
- c** Conjunto das letras do alfabeto brasileiro [ ]
- d** Conjunto dos algarismos árabicos [ ]
- e** Conjunto dos algarismos romanos [ ]
- f** Conjunto  $\{ a, b, c, d \}$  [ ]
- g** Conjunto das partes de  $\{ a, b, c \}$  [ ]
- h** Conjunto das vogais [ ]
- i** Conjunto das letras gregas [ ]

**exercício 2.2** Apresente os possíveis prefixos e sufixos de cada uma das seguintes palavras:

- a** teoria
- b** universidade
- c** aaa
- d** abccba
- e** abcabc

**exercício 2.3** Exemplifique, comprovando ou negando as seguintes propriedades algébricas da operação de concatenação de palavras:

- a** Total (suponha a operação definida sobre uma determinada linguagem);
- b** Comutativa;
- c** Elemento neutro;
- d** Associativa;

**e** Elemento absorvente, genericamente definida como segue: para uma dada operação binária  $\oplus$  sobre um conjunto  $A$ , afirma-se que a operação possui elemento absorvente se existe  $a \in A$  tal que, para qualquer  $x \in A$  vale que:

$$a \oplus x = x \oplus a = a$$

**f** Elemento inverso, genericamente definido como segue: para uma dada operação binária  $\oplus$  sobre um conjunto  $A$ , afirma-se que a operação  $\oplus$  possui elemento *inverso* se possui elemento neutro  $e$  e, para qualquer  $a \in A$ , existe  $\underline{a} \in A$  tal que:

$$a \oplus \underline{a} = \underline{a} \oplus a = e$$

**exercício 2.4** Sejam as linguagens  $L_1$ ,  $L_2$  e  $L_3$ . Verifique se é verdadeira a igualdade:

$$L_1 (L_2 \cap L_3) = L_1 L_2 \cap L_1 L_3$$

**exercício 2.5** Relativamente ao exemplo 2.12 – Gramática, derivação, linguagem gerada: números naturais:

- a** Existe mais alguma derivação do número 243? Caso exista, quantas?
- b** Modifique a gramática de tal forma a não distinguir zeros à esquerda.

**exercício 2.6** Considere o exemplo 2.13 – Gramática, derivação, linguagem gerada: palavra duplicada. Existe mais alguma derivação da palavra **baba**?

**exercício 2.7** Desenvolva uma gramática que gere a linguagem correspondente aos identificadores da linguagem Pascal (palavras formadas por uma ou mais letras ou dígitos, as quais sempre iniciam por uma letra). Analogamente para os identificadores em Pascal com tamanho máximo de seis caracteres.

**exercício 2.8** Desenvolva uma gramática que gere expressões aritméticas com parênteses平衡eados, dois operadores (representados por \* e +) e um operando (representado por x). Por exemplo, as seguintes palavras são expressões aritméticas válidas:

$$\begin{array}{ll} x & x*(x+x) \\ & (((((x)))))) \\ & \{ a^n b^n c^n \mid n \geq 0 \} \end{array}$$

**exercício 2.9** Desenvolva uma gramática que gere a seguinte linguagem:

### 3.7 → gramática regular

Usando o conceito de gramáticas como apresentado no capítulo 2 – Linguagens e gramáticas, é possível definir tanto linguagens regulares como linguagens não regulares. Entretanto, é possível estabelecer restrições nas regras de produção, de tal forma a definir exatamente a classe das linguagens regulares.

Existe mais de uma forma de restringir as regras de produção de forma a definir uma *gramática regular*. A seguir, são apresentadas quatro dessas formas, denominadas *gramáticas lineares*.

#### definição 3.23 – Gramáticas lineares

Seja  $G = (V, T, P, S)$  uma gramática. Sejam  $A$  e  $B$  elementos de  $V$  e  $w$  uma palavra de  $T^*$ . Então  $G$  é uma *gramática linear* se todas as suas produções encontram-se em *uma* e em *só* mente *uma* das seguintes formas:

**a** Gramática *linear à direita* (abreviada por **GLD**). Todas as regras de produção são da forma:

$$A \rightarrow wB \quad \text{ou} \quad A \rightarrow w$$

**b** Gramática *linear à esquerda* (abreviada por **GLE**). Todas as regras de produção são da forma:

$$S \rightarrow aA$$

$$A \rightarrow bA \mid \epsilon$$

**c** Linear à esquerda.  $G = (\{S\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow Sba \mid a$$

**d** Linear unitária à direita.  $G = (\{S, A, B\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow aA$$

$$A \rightarrow bB \mid \epsilon$$

**e** Linear unitária à esquerda.  $G = (\{S, A\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow Aa \mid a$$

$$A \rightarrow Sb$$

**f** Linear unitária à esquerda (abreviada por **GLUE**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

$$|w| \leq 1$$

$$|w| \leq 1$$

Note-se que as gramáticas lineares possuem forte restrição no formato de suas produções:

- O lado esquerdo possui exatamente uma variável;
- O lado direito de uma produção é constituído por, no máximo, uma variável. Adicionalmente, esta variável, se existir, sempre antecede (linear à esquerda) ou sucede (linear à direita) qualquer subpalavra (eventualmente vazia) de terminais.

Sugere-se como exercício verificar se é possível definir uma gramática que satisfaça simultaneamente as quatro formas lineares.

#### teorema 3.24 – Equivalência das gramáticas lineares

Seja  $L$  uma linguagem. Então:

L é gerada por uma **GLD** se, e somente se,  
L é gerada por uma **GLE** se, e somente se,

L é gerada por uma **GLUD** se, e somente se,  
L é gerada por uma **GLUE**.

Ou seja, as diversas formas das gramáticas lineares são formalismos equivalentes. A demonstração do teorema é sugerida como exercício.

#### definição 3.25 – Gramática regular

Uma gramática  $G$  é dita uma *gramática regular*, eventualmente abreviada por **GR**, se  $G$  é uma gramática linear.

#### definição 3.26 – Linguagem gerada

Seja  $G = (V, T, P, S)$  uma gramática. A *linguagem gerada* pela gramática  $G$ , denotada por:

$$L(G) \quad \text{ou} \quad \text{GERA}(G)$$

$$L(G) = \{w \in T^* \mid S \Rightarrow^+ w\}$$

#### exemplo 3.14 – Gramática regular: $a(ba)^*$

A linguagem  $a(ba)^*$  é gerada pelas seguintes gramáticas regulares:

**a** Linear à direita.  $G = (\{S, A\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow aA$$

$$A \rightarrow baA \mid \epsilon$$

**b** Linear à esquerda.  $G = (\{S\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow Sba \mid a$$

$$C$$

**c** Linear unitária à direita.  $G = (\{S, A, B\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow aA$$

$$A \rightarrow bB \mid \epsilon$$

$$B \rightarrow aA$$

**d** Linear unitária à esquerda.  $G = (\{S, A\}, \{a, b\}, P, S)$ , e  $P$  possui as seguintes regras de produção:

$$S \rightarrow Aa \mid a$$

$$A \rightarrow Sb$$

**e** Linear unitária à direita (abreviada por **GLUE**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

**f** Gramática *linear unitária à esquerda* (abreviada por **GLUE**). Todas as regras de produção são como na gramática linear à esquerda e, adicionalmente:

**g** Gramática *linear unitária à direita* (abreviada por **GLD**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

**h** Gramática *linear unitária à esquerda* (abreviada por **GLE**). Todas as regras de produção são como na gramática linear à esquerda e, adicionalmente:

**i** Gramática *linear unitária à direita* (abreviada por **GLUE**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

**j** Gramática *linear unitária à esquerda* (abreviada por **GLD**). Todas as regras de produção são como na gramática linear à esquerda e, adicionalmente:

**k** Gramática *linear unitária à direita* (abreviada por **GLE**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

**l** Gramática *linear unitária à esquerda* (abreviada por **GLUE**). Todas as regras de produção são como na gramática linear à esquerda e, adicionalmente:

**m** Gramática *linear unitária à direita* (abreviada por **GLD**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

**n** Gramática *linear unitária à esquerda* (abreviada por **GLE**). Todas as regras de produção são como na gramática linear à esquerda e, adicionalmente:

**o** Gramática *linear unitária à direita* (abreviada por **GLUE**). Todas as regras de produção são como na gramática linear à direita e, adicionalmente:

**observação 3.27** – Gramática linear à esquerda e linear à direita  
Suponha  $|w| \geq 1$ . Se uma gramática tiver simultaneamente produções do tipo  $A \rightarrow wB$  (linear à direita) e do tipo  $A \rightarrow Bw$  (linear à esquerda), então a correspondente linguagem gerada poderá não ser regular, ou seja, esta não é uma gramática regular. Por exemplo, já foi discutido que uma linguagem que possua duplo balanceamento não é regular. Em particular, a seguinte linguagem não é regular (adiante, é estudado como provar que tal linguagem não é regular):

$$\{ a^n b^n \mid n \in \mathbb{N} \}$$

Entretanto, é possível desenvolver uma gramática, com produções lineares à direita e à esquerda, que gera esta linguagem, o que se sugere como exercício.

Os dois teoremas a seguir mostram que a classe das gramáticas regulares denota exatamente a classe das linguagens regulares.

**teorema 3.28** – Gramática regular  $\rightarrow$  linguagem regular

Se  $L$  é uma linguagem gerada por uma gramática regular, então  $L$  é uma linguagem regular.

► **Prova: (por indução)**

Para mostrar que uma linguagem é regular, é suficiente construir um autômato finito que a reconheça. Suponha  $G = (V, T, P, S)$  uma gramática linear unitária à direita. Então o AFN $\epsilon$ :

$$M = (\Sigma, Q, \delta, q_0, F)$$

tal que (suponha  $q_f \notin V$ ):

$$\Sigma = T$$

$$Q = V \cup \{ q_f \}$$

$$F = \{ q_f \}$$

$$q_0 = S$$

$\delta$  é construída como ilustrado na figura 3.32 (suponha  $A$  e  $B$  variáveis e a terminal) simula as derivações de  $G$ , ou seja:

$$ACEITA(M) = GERA(G)$$

tipo da produção	transição gerada
$A \rightarrow \epsilon$	$\delta(A, \epsilon) = \{ q_f \}$
$A \rightarrow a$	$\delta(A, a) = \{ q_f \}$
$A \rightarrow B_1 \mid \dots \mid B_n$	$\delta(A, \epsilon) = \{ B_1, \dots, B_n \}$
$A \rightarrow aB_1 \mid \dots \mid aB_n$	$\delta(A, a) = \{ B_1, \dots, B_n \}$

$A \rightarrow \epsilon$	$\delta(A, \epsilon) = \{ q_f \}$
$A \rightarrow a$	$\delta(A, a) = \{ q_f \}$
$A \rightarrow B_1 \mid \dots \mid B_n$	$\delta(A, \epsilon) = \{ B_1, \dots, B_n \}$
$A \rightarrow aB_1 \mid \dots \mid aB_n$	$\delta(A, a) = \{ B_1, \dots, B_n \}$

onde  $P$  é tal que:

$$\begin{aligned} S &\rightarrow aA \\ A &\rightarrow bB \quad | \quad \epsilon \\ B &\rightarrow aA \end{aligned}$$

O AFN $\epsilon$   $M$  que reconhece a linguagem gerada pela gramática  $G$  é:

$$M = (\{ a, b \}, \{ S, A, B, q_f \}, \delta, S, \{ q_f \})$$

o qual é ilustrado na figura 3.34, onde  $\delta$  é dada pela tabela na figura 3.33.

A demonstração que, de fato,  $ACEITA(M) = GERA(G)$  é por indução no número de derivações, como segue (suponha  $\alpha$  elemento de  $(T \cup V)^*$  e  $w$  elemento de  $T^*$ ):



**figura 3.32** Transições construídas a partir das produções.

► **a** Base de indução. Suponha que  $S \Rightarrow^1 \alpha$ . Então, se:

- a.1)  $\alpha = \epsilon$ , existe uma regra  $S \rightarrow \epsilon$ . Por construção de  $M$ ,  $\delta(S, \epsilon) = \{ q_f \}$
- a.2)  $\alpha = a$ , existe uma regra  $S \rightarrow a$ . Por construção de  $M$ ,  $\delta(S, a) = \{ q_f \}$
- a.3)  $\alpha = A$ , existe uma regra  $S \rightarrow A$ . Por construção de  $M$ ,  $\delta(S, \epsilon) = \{ A \}$
- a.4)  $\alpha = aA$ , existe uma regra  $S \rightarrow aA$ . Por construção de  $M$ ,  $\delta(S, a) = \{ A \}$

► **b** Hipótese de indução. Suponha que  $S \Rightarrow^n \alpha$ ,  $n > 1$  tal que, se:

- b.1)  $\alpha = w$ , então  $\delta^*(S, w) = \{ q_f \}$
- b.2)  $\alpha = wA$ , então  $\delta^*(S, w) = \{ A \}$

► **c** Passo de indução. Suponha que  $S \Rightarrow^{n+1} \alpha$ . Obrigatoriamente, ocorre somente a hipótese b.2) acima e, portanto:

$$S \Rightarrow^n wA \Rightarrow^1 \alpha$$

Então, se:

- c.1)  $\alpha = w\epsilon = w$ , existe uma regra  $A \rightarrow \epsilon$ . Logo:  
 $\delta^*(S, w\epsilon) = \delta(\delta^*(S, w), \epsilon) = \delta(A, \epsilon) = \{ q_f \}$
- c.2)  $\alpha = wb$ , existe uma regra  $A \rightarrow b$ . Logo:  
 $\delta^*(S, wb) = \delta(\delta^*(S, w), b) = \delta(A, b) = \{ q_f \}$
- c.3)  $\alpha = wbB$ , existe uma regra  $A \rightarrow B$ . Logo:  
 $\delta^*(S, wb) = \delta(\delta^*(S, w), b) = \delta(A, \epsilon) = \{ B \}$
- c.4)  $\alpha = wbB$ , existe uma regra  $A \rightarrow bB$ . Logo:  
 $\delta^*(S, wb) = \delta(\delta^*(S, w), b) = \delta(A, b) = \{ B \}$

► **exemplo 3.16** – Construção de um AFN $\epsilon$  a partir de uma gramática regular  
Considere a seguinte gramática linear unitária à direita, introduzida no exemplo 3.14 – Gramática regular:

$$G = (\{ S, A, B \}, \{ a, b \}, P, S)$$



produção	transição
$S \rightarrow aA$	$\delta(S, a) = \{A\}$
$A \rightarrow bB$	$\delta(A, b) = \{B\}$
$A \rightarrow \epsilon$	$\delta(A, \epsilon) = \{q_f\}$
$A \rightarrow aA$	$\delta(B, a) = \{A\}$

figura 3.33 Transições construídas a partir das produções.

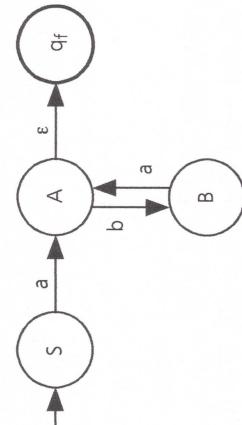


figura 3.34 Autômato finito construído a partir de uma gramática regular.

**teorema 3.29** – Linguagem regular  $\rightarrow$  Gramática regular  
Se  $L$  é linguagem regular, então existe  $G$ , gramática regular que gera  $L$ .

→ **Prova: (por indução)**

Se  $L$  é uma linguagem regular, então existe um AFD  $M = (\Sigma, Q, \delta, q_0, F)$  tal que  $ACEITA(M) = L$ . A ideia central da demonstração é construir uma gramática linear à direita  $G$  a partir de  $M$  tal que:

$$GERA(G) = ACEITA(M)$$

ou seja, cuja derivação simula a função programa estendida. Seja a gramática regular:

$$G = (V, T, P, S_0)$$

tal que (suponha  $S \notin Q$ ):

$$\begin{aligned} V &= Q \cup \{S\} \\ T &= \Sigma \\ P &\quad \text{é construído como ilustrado na figura 3.35 (suponha } q_i, q_k \text{ elementos de } Q, q_f \\ &\quad \text{elemento de } F \text{ e } a \text{ elemento de } \Sigma) \end{aligned}$$

ou seja, cuja derivação simula a função programa estendida. Seja a gramática regular:

produção	transição
$S \rightarrow q_0$	$S \rightarrow q_0$
$q_f \rightarrow \epsilon$	$q_f \rightarrow \epsilon$

figura 3.35 Produções construídas a partir de transições.

A demonstração de que, de fato,  $GERA(G) = ACEITA(M)$  é por indução no tamanho da palavra, como segue (suponha  $w$  elemento de  $\Sigma^*$ ):

- a Base de indução. Seja  $w$  tal que  $|w| = 0$ . Por definição, vale que  $S \rightarrow q_0$  é produção. Se  $\epsilon \in ACEITA(M)$ , então  $q_0$  é estado final e, por definição, vale que  $q_0 \rightarrow \epsilon$  é produção. Logo:

$$S \Rightarrow q_0 \Rightarrow \epsilon$$

b Hipótese de indução. Seja  $w$  tal que  $|w| = n (n \geq 1)$  e  $\delta^*(q_0, w) = q$ . Assim, se:

- b.1)  $q$  não é estado final, então suponha que  $S \Rightarrow^n wq$
- b.2)  $q$  é estado final, então suponha que  $S \Rightarrow^n wq \Rightarrow w$  (este caso não é importante para o passo de indução);

c Passo de indução. Seja  $w$  tal que  $|wa| = n + 1$  e  $\delta^*(q_0, wa) = p$ . Então:

$$\delta(\delta^*(q_0, w), a) = \delta(q, a) = p$$

Portanto, obrigatoriamente, ocorre somente a hipótese b.1) acima e, se:

- c.1)  $p$  não é estado final, então  $S \Rightarrow^n wq \Rightarrow^1 wap$
- c.2)  $p$  é estado final, então  $S \Rightarrow^n wq \Rightarrow^1 wap \Rightarrow^1 wa$

**exemplo 3.17** – Construção de uma gramática regular a partir de um AFD

Considere o autômato finito determinístico:

$$M = (\{a, b, c\}, \{q_0, q_1, q_2\}, \delta, q_0, \{q_0, q_1, q_2\})$$

ilustrado na figura 3.37. A correspondente gramática regular construída é:

$$G = (\{q_0, q_1, q_2, S\}, \{a, b, c\}, P, S)$$

onde  $P$  é dado pela tabela na figura 3.36.

transição	produção
-	$S \rightarrow q_0$
-	$q_0 \rightarrow \epsilon$
-	$q_1 \rightarrow \epsilon$
-	$q_2 \rightarrow \epsilon$
$\delta(q_0, a) = q_0$	$q_0 \rightarrow aq_0$
$\delta(q_0, a) = q_1$	$q_0 \rightarrow bq_1$
$\delta(q_1, b) = q_1$	$q_1 \rightarrow bq_1$
$\delta(q_1, c) = q_2$	$q_1 \rightarrow cq_2$
$\delta(q_2, c) = q_2$	$q_2 \rightarrow cq_2$

figura 3.36 Produções construídas a partir de transições.

produção	transição
$S \rightarrow q_0$	-
$q_f \rightarrow \epsilon$	-
$q_i \rightarrow aq_k$	$\delta(q_i, a) = q_k$

figura 3.35 Produções construídas a partir de transições.

produção	transição
$S \rightarrow aA$	$\delta(S, a) = \{A\}$
$A \rightarrow bB$	$\delta(A, b) = \{B\}$
$A \rightarrow \epsilon$	$\delta(A, \epsilon) = \{q_f\}$
$A \rightarrow aA$	$\delta(B, a) = \{A\}$

figura 3.33 Transições construídas a partir das produções.

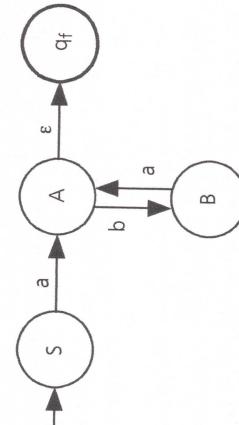


figura 3.34 Autômato finito construído a partir de uma gramática regular.

**teorema 3.29 – Linguagem regular → Gramática regular**  
Se  $L$  é linguagem regular, então existe  $G$ , gramática regular que gera  $L$ .

→ **Prova: (por indução)**

Se  $L$  é uma linguagem regular, então existe um AFD  $M = (\Sigma, Q, \delta, q_0, F)$  tal que  $ACEITA(M) = L$ . A ideia central da demonstração é construir uma gramática linear à direita  $G$  a partir de  $M$  tal que:

$$GERA(G) = ACEITA(M)$$

ou seja, cuja derivação simula a função programa estendida. Seja a gramática regular:

$$G = (V, T, P, S)$$

tal que (suponha  $S \notin Q$ ):

$$\begin{aligned} V &= Q \cup \{S\} \\ T &= \Sigma \\ P & \end{aligned}$$

é construído como ilustrado na figura 3.35 (suponha  $q_i, q_k$  elementos de  $Q$ ;  $q_i$  elemento de  $F$  e  $a$  elemento de  $\Sigma$ )

transição	produção
$S \rightarrow q_0$	$S \rightarrow q_0$
$q_f \rightarrow \epsilon$	$q_0 \rightarrow bq_1$
$q_i \rightarrow aq_k$	$q_1 \rightarrow cq_2$

figura 3.35 Produções construídas a partir de transições.

À demonstração de que, de fato,  $GERA(G) = ACEITA(M)$  é por indução no tamanho da palavra, como segue (suponha  $w$  elemento de  $\Sigma^*$ ):

- a) **Base de indução.** Seja  $w$  tal que  $|w| = 0$ . Por definição, vale que  $S \rightarrow q_0$  é produção. Se  $\epsilon \in ACEITA(M)$ , então  $q_0$  é estado final e, por definição, vale que  $q_0 \rightarrow \epsilon$  é produção. Logo:

$$S \Rightarrow q_0 \Rightarrow \epsilon$$

b) **Hipótese de indução.** Seja  $w$  tal que  $|w| = n$  ( $n \geq 1$ ) e  $\delta^*(q_0, w) = q$ . Assim, se:

- b.1)  $q$  não é estado final, então suponha que  $S \Rightarrow^n wq$
- b.2)  $q$  é estado final, então suponha que  $S \Rightarrow^n wq \Rightarrow w$  (este caso não é importante para o passo de indução);

c) **Passo de indução.** Seja  $w$  tal que  $|wa| = n + 1$  e  $\delta^*(q_0, wa) = p$ . Então:

$$\delta(\delta^*(q_0, w), a) = \delta(q, a) = p$$

Portanto, obrigatoriamente, ocorre somente a hipótese b.1) acima e, se:

- c.1)  $p$  não é estado final, então  $S \Rightarrow^n wq \Rightarrow^1 wap$
- c.2)  $p$  é estado final, então  $S \Rightarrow^n wq \Rightarrow^1 wap \Rightarrow^1 wa$

**exemplo 3.17 – Construção de uma gramática regular a partir de um AFD**

Considere o autômato finito determinístico:

$$M = (\{a, b, c\}, \{q_0, q_1, q_2\}, \delta, q_0, \{q_0, q_1, q_2\})$$

ilustrado na figura 3.37. A correspondente gramática regular construída é:

$$G = (\{q_0, q_1, q_2, S\}, \{a, b, c\}, P, S)$$

onde  $P$  é dado pela tabela na figura 3.36.

transição	produção
$S \rightarrow q_0$	$S \rightarrow q_0$
$q_0 \rightarrow \epsilon$	$q_0 \rightarrow bq_1$
$q_1 \rightarrow \epsilon$	$q_1 \rightarrow cq_2$
$q_2 \rightarrow \epsilon$	$q_1 \rightarrow cq_2$
$q_0 \rightarrow aq_0$	$q_0 \rightarrow cq_2$
$q_0 \rightarrow aq_1$	$q_0 \rightarrow cq_2$
$q_1 \rightarrow aq_1$	$q_1 \rightarrow cq_2$
$q_1 \rightarrow aq_2$	$q_1 \rightarrow cq_2$
$q_2 \rightarrow aq_2$	$q_1 \rightarrow cq_2$

figura 3.36 Produções construídas a partir de transições.

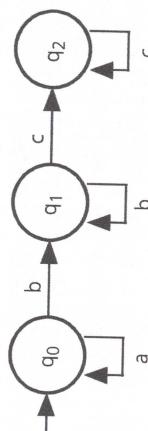


figura 3.37 Diagrama do autômato finito determinístico.

## → exercícios

**exercício 3.1** Sobre as linguagens regulares:

- a Qual a importância do seu estudo?
- b Exemplifique suas aplicações (para os diversos formalismos);
- c Você imagina algum tipo de linguagem cujo algoritmo de reconhecimento seja mais eficiente que o das regulares? E menos eficiente? Explique a sua resposta.

**exercício 3.2** Desenvolva autômatos finitos determinísticos que reconheçam as seguintes linguagens sobre  $\Sigma = \{a, b\}$ :

- a  $\{w \mid o \text{ sufixo de } w \text{ é } aa\}$
- b  $\{w \mid w \text{ possui } aaa \text{ como subpalavra}\}$
- c  $\{w \mid w \text{ possui número ímpar de } a \text{ e número ímpar de } b\}$
- d  $\{w \mid w \text{ possui número par de } a \text{ e ímpar de } b \text{ ou } w \text{ possui número par de } b \text{ e ímpar de } a\}$

- e  $\{w \mid o \text{ quinto símbolo da direita para a esquerda de } w \text{ é } a\}$

Dica: o autômato resultante possui um número relativamente grande de estados.

**exercício 3.3** Qual a condição para que a palavra vazia pertença à linguagem gerada por um autômato finito determinístico?

**exercício 3.4** Desenvolva autômatos finitos não determinísticos que reconheçam as seguintes linguagens sobre o alfabeto  $\Sigma = \{a, b\}$ :

- a Qualquer ocorrência de **a** é imediatamente sucedida por **b**;
- b Qualquer ocorrência de **a** é imediatamente antecedida e imediatamente sucedida por **b**.

**exercício 3.5** Quais das seguintes palavras são aceitas pelos autômatos finitos não determinísticos sobre o alfabeto  $\Sigma = \{a, b\}$ :

- a Ilustrado na figura 3.38:  
    *ε?*
- b *aa?*
- c *bb?*
- d *aba?*
- e *bbaabababa?*

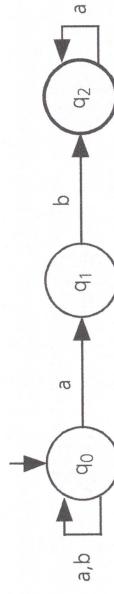


figura 3.38 Diagrama (AFN).

**b** Ilustrado na figura 3.39:

*ε?*

*bb?*

*bbbbaaaa?*  
*bbbbbbbababababababa?*

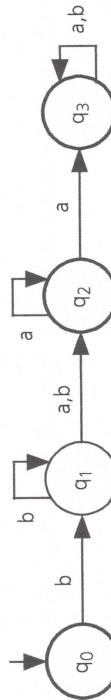


figura 3.39 Diagrama (AFN).

**exercício 3.6** Desenvolva autômatos finitos não determinísticos, com ou sem movimentos vazios, que reconheçam as seguintes linguagens sobre o alfabeto  $\Sigma = \{a, b\}$ :

- a  $\{w_1 w_2 w_1 \mid w_2 \text{ é qualquer } e \mid |w_1| = 3\}$
- b  $\{w \mid o \text{ décimo símbolo da direita para a esquerda de } w \text{ é } a\}$
- c  $\{w \mid w \text{ possui igual número de símbolos } a \text{ e } b \text{ (qualquer prefixo de } w \text{ possui, no máximo, dois } a \text{ ou mais que } b \text{ ou qualquer prefixo de } w \text{ possui, no máximo, dois } b \text{ a mais que } a\}$

**exercício 3.7** Desenvolva, sobre o alfabeto  $\Sigma = \{a, b, c\}$ :

- a Autômato não determinístico que reconheça a seguinte linguagem:  
 $\{w \mid a \text{ ou } bb \text{ ou } ccc \text{ é sufixo de } w\}$
- b Autômato finito não determinístico com movimentos vazios que reconheça a seguinte linguagem:

**exercício 3.8** Considere o autômato sobre o alfabeto  $\Sigma = \{\epsilon, \eta, \mu\}$  ilustrado na figura 3.40. Justifique ou refute as seguintes afirmações:

- a O autômato possui movimentos vazios;
  - b Aceita a linguagem  $(\eta + \mu)^*$
- Dica: observe bem a definição do alfabeto.



figura 3.40 Diagrama: autômato finito.

**exercício 3.9** Por que se pode afirmar que um autômato finito com movimentos vazios sempre para (ao processar qualquer entrada)? Em particular, analise para as seguintes transições que caracterizam um ciclo (suponha que  $q$  e  $p$  são estados do autômato), considerando a definição de função programa estendida:

$$\begin{aligned}\delta(q, \varepsilon) &= \{p\} \\ \delta(p, \varepsilon) &= \{q\}\end{aligned}$$

**exercício 3.10** Complete a prova referente ao teorema 3.19 – Equivalência entre AFN e AFN.

**exercício 3.11** Seja  $L$  uma linguagem regular. Prove que  $L^R$  também é regular sendo que:

$$L^R = \{w \mid o \text{ reverso de } w \text{ está em } L\}$$

Dica: se  $L$  é regular, então existe autômato finito que a reconhece. Como fazer o autômato “reverso”?

**exercício 3.12** Adição binária é regular. A operação de adição de dois números binários pode ser expressa em termos de sequências de triplas. Por exemplo, considere a seguinte adição binária:

$$\begin{array}{r} 0 \ 1 \ 0 \ 1 \\ 1 \ 0 \ 0 \ 1 \\ \hline 1 \ 1 \ 1 \ 0 \end{array}$$

a qual pode ser expressa pela seguinte sequência de triplas sendo que cada tripla representa uma coluna:

$$(0, 1, 1), (1, 0, 1), (0, 0, 1), (1, 1, 0)$$

e que pode ser melhor visualizada usando a seguinte notação para sequência de triplas:

$$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Seja  $B = \{0, 1\}$  e  $\Sigma = B \times B \times B$ . Construa um autômato finito sobre o alfabeto  $\Sigma$  que reconhece todas as palavras que representam a operação de adição binária.

**exercício 3.13** Descreva em palavras as linguagens geradas pelas seguintes expressões regulares:

- a**  $(a + b + c)^*(a + b + c)$
- b**  $(aa + b)^*(a + bb)$
- c**  $(b + ab)^*(\varepsilon + a)$
- d**  $c^*(a + b)(a + b + c)^*$
- e**  $((b + c)^* + a(b + c)^*)^*$
- f**  $(aa + bb + (aa + bb)(ab + ba)(aa + bb))^*$

**exercício 3.14** Desenvolva expressões regulares que gerem as seguintes linguagens sobre  $\Sigma = \{0, 1, \dots, 9, a, b, \dots, z, +, -\}$ :

- a** Identificadores em uma linguagem tipo Pascal, sendo que o primeiro caractere sempre é uma letra e os demais podem ser zero ou mais letras ou dígitos;
- b** Números inteiros em uma linguagem tipo Pascal, compostos por qualquer sequência não vazia de dígitos, precedidos ou não por um sinal.

**exercício 3.15** Desenvolva expressões regulares que gerem as seguintes linguagens sobre  $\Sigma = \{a, b\}$ :

- a**  $\{w \mid w \text{ não possui } aba \text{ como subpalavra}\}$
- b**  $\{w \mid \text{qualquer par de } a \text{ antecede qualquer par de } b\}$
- c**  $\{w \mid w \text{ tem no máximo um par de } a \text{ como subpalavra e no máximo um par de } b \text{ como subpalavra}\}$

**exercício 3.16** Aponte, justificando, quais das seguintes expressões regulares são equivalentes:

$$\begin{aligned}(a + b)^*a^* \\ (a + b)^* \\ ((a + b)a)^*\end{aligned}$$

**exercício 3.17** Aplique o algoritmo de tradução de formalismo de expressão regular para autômato finito:

- a**  $a(aa + bb)^* + \varepsilon$
- b**  $(\varnothing^* + \varepsilon^*)^*$
- c**  $(ab + ba)^*(aa + bb)^*$
- d**  $ab(ab^* + baa^*)^*ba$

**exercício 3.18** Desenvolva gramáticas regulares que gerem as seguintes linguagens sobre  $\Sigma = \{0, 1, \dots, 9, a, b, \dots, z, +, -\}$ :

- a** Identificadores em uma linguagem tipo Pascal, sendo que o primeiro caractere sempre é uma letra e os demais podem ser zero ou mais letras ou dígitos;
- b** Números inteiros em uma linguagem tipo Pascal, compostos por qualquer sequência não vazia de dígitos, precedidos ou não por um sinal.

**exercício 3.19** Desenvolva gramáticas regulares que gerem as seguintes linguagens sobre  $\Sigma = \{a, b\}$ :

- a**  $\{w \mid w \text{ não possui } aba \text{ como subpalavra}\}$
- b**  $\{w \mid \text{qualquer par de } a \text{ antecede qualquer par de } b\}$
- c**  $\{w \mid w \text{ tem no máximo um par de } a \text{ como subpalavra e no máximo um par de } b \text{ como subpalavra}\}$

**exercício 3.20** É possível definir uma gramática regular que satisfaça simultaneamente as quatro formas lineares? Justifique a sua resposta.

**exercício 3.21** Demonstre a equivalência dos quatro tipos de gramáticas lineares.

**exercício 3.22** Relativamente às gramáticas regulares, justifique ou refute as seguintes afirmações:

**a** Considere uma gramática com as seguintes produções (suponha que **A** é o símbolo inicial):

$$\begin{array}{l} A \rightarrow aB \\ B \rightarrow aC \mid bC \\ C \rightarrow aD \mid bD \\ D \rightarrow aS \mid bS \mid \epsilon \\ S \rightarrow D \end{array}$$

Então:

- a.1) A gramática é regular;  
a.2) A linguagem gerada é composta por todas as palavras cujo terceiro símbolo da direita para a esquerda é **a**;

**b** Considere uma gramática com as seguintes produções (suponha que **S** é o símbolo inicial):

$$\begin{array}{l} S \rightarrow aS \mid Xb \mid \epsilon \\ X \rightarrow Sb \end{array}$$

Então:

- b.1) A gramática é não regular;  
b.2) A linguagem gerada é regular.

**exercício 3.23** Aplique os algoritmos de tradução de formalismos apresentados e, a partir da seguinte expressão regular:

$$(b + \epsilon)(a + bb)^*$$

realize as diversas etapas até gerar a gramática regular correspondente, ou seja:

$$ER \rightarrow AFN\epsilon \rightarrow AFN \rightarrow AFD \rightarrow GR$$

**exercício 3.24** Foi afirmado que a seguinte linguagem é não regular:

$$\{ a^n b^n \mid n \in \mathbb{N} \}$$

Desenvolva uma gramática com produções lineares à direita e à esquerda que gera esta linguagem.

**exercício 3.25** Foi declarado que uma importante aplicação das linguagens regulares é a especificação e o desenvolvimento de analisadores léxicos. Para reforçar tal ideia, apresente a seguinte linguagem baseada em unidades léxicas da linguagem de programação Pascal (ou alguma outra de seu domínio), usando um formalismo regular de sua escolha:

$$\{ w \mid w \text{ é número inteiro ou } w \text{ é número real ou } w \text{ é identificador } \}$$

**exercício 3.26** Foi afirmado que linguagens de propósitos gerais como Pascal, C, Java, etc., são não regulares, pois possuem estruturas de duplo balanceamento como parênteses em expressões aritméticas. Entretanto, é possível definir linguagens com duplo平衡amento

nos quais são regulares: basta limitar o número máximo de encadeamentos (e portanto, limitar o número máximo de alternativas, ou seja, de estados possíveis, permitindo uma representação finita). De fato, muitas linguagens de programação, até a década de 1970, possuíam limites desse tipo, com o objetivo de otimizar a compilação. Para exemplificar tal afirmação, desenvolva um formalismo regular (de qualquer tipo) sobre o alfabeto  $\Sigma = \{ x, (, ) \}$  o qual reconhece/gera qualquer palavra com parênteses平衡ados, desde que limitados a três níveis de encadeamento. Por exemplo:

$\epsilon$  e  $x(x(xx))x$  são palavras da linguagem;  
 $((((x)))$  não é palavra da linguagem.

**exercício 3.27** Considere o alfabeto  $\Sigma = \{ a, b \}$ . A linguagem  $L_n$ , para algum  $n$  natural fixo, é o conjunto de todas as palavras sobre  $\Sigma$  cujo comprimento máximo é  $n$ . Então, para qualquer  $n$  fixo, pode-se afirmar que sempre existe um formalismo regular que aceita/gera a linguagem  $L_n$ .

**exercício 3.28** Desenvolva um programa em computador que simule o processamento de qualquer autômato finito determinístico, como segue:

- entrada: função de transição, estado inicial, conjunto de estados finais e as palavras a serem processadas;
- saída: condição de parada ACEITA/REJEITA e identificação do estado de parada.

Dica: a parte central do algoritmo de simulação é realmente pequena, e pode ser expressa em poucas unidades de linhas de código e consiste, basicamente, em um algoritmo que controla a mudança de estado do autômato lido da entrada. Tal resultado comprova a simplicidade que é desenvolver um simulador genérico de autômato finito.

**exercício 3.29** Desenvolva um programa em computador que implemente os seguintes algoritmos:

- a** Tradução de AFN para AFD;
- b** Tradução de AFN $\epsilon$  para AFN;
- c** Tradução de ER para AFN $\epsilon$ ;
- d** Tradução de GR para AFN $\epsilon$  equivalente.

Dica: basta implementar o algoritmo apresentado em cada prova do correspondente teorema.

Observe que, usando os tradutores em combinação com o simulador do exercício acima, pode-se desenvolver um simulador de linguagens regulares tendo como entrada qualquer dos formalismos estudados.

**exercício 3.30** Desenvolva um algoritmo que gere, em ordem lexicográfica, todas as palavras representadas por:

- a** Uma expressão regular qualquer;
- b** Uma gramática regular qualquer.

Dica: considere os algoritmos desenvolvidos nos dois exercícios acima.