UNIP
UNIVERSIDADE PAULISTA
Interativa

Linguagens Formais e Autômatos

Autora: Profa. Miryam de Moraes

Professora conteudista: Miryam de Moraes

Miryam de Moraes possui graduação, mestrado e doutorado em Engenharia Elétrica pela Universidade de São Paulo. Seu doutorado diz respeito ao Tratamento de Dependências de Contexto empregando Tecnologia Adaptativa. Interessa-se por aplicações da Tecnologia Adaptativa à Engenharia da Computação, particularmente na análise e no processamento de linguagens naturais, processos de aprendizagem e inferências automáticas. Leciona Linguagens Formais e Aspectos Teóricos da Computação na Universidade Paulista desde 1998.

Dados Internacionais de Catalogação na Publicação (CIP)

M827L Moraes, Miryam de

Linguagens formais e autômatos / Miryam de Moraes. - São Paulo: Editora Sol, 2013.

88 p. il.

1. Linguagens regulares. 2.Linguagens livres de contexto. 3. Linguagens recursivas. I. Título.

CDU 004.43

© Todos os direitos reservados. Nenhuma parte desta obra pode ser reproduzida ou transmitida por qualquer forma e/ou quaisquer meios (eletrônico, incluindo fotocópia e gravação) ou arquivada em qualquer sistema ou banco de dados sem permissão escrita da Universidade Paulista.

Prof. Dr. João Carlos Di Genio
Reitor

Prof. Fábio Romeu de Carvalho
Vice-Reitor de Planejamento, Administração e Finanças

Profa. Melânia Dalla Torre
Vice-Reitora de Unidades Universitárias

Prof. Dr. Yugo Okida
Vice-Reitor de Pós-Graduação e Pesquisa

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Graduação

Unip Interativa – EaD

Profa. Elisabete Brihy

Prof. Marcelo Souza

Profa. Melissa Larrabure

Material Didático – EaD

Comissão editorial:
Dra. Angélica L. Carlini (UNIP)
Dr. Cid Santos Gesteira (UFBA)
Dra. Divane Alves da Silva (UNIP)
Dr. Ivan Dias da Motta (CESUMAR)
Dra. Kátia Mosorov Alonso (UFMT)
Dra. Valéria de Carvalho (UNIP)

Apoio:
Profa. Cláudia Regina Baptista – EaD
Profa. Betisa Malaman – Comissão de Qualificação e Avaliação de Cursos

Projeto gráfico:
Prof. Alexandre Ponzetto

Revisão:
Carla Moro
Andréia Andrade

Sumário

Linguagens Formais e Autômatos

APRESENTAÇÃO

A disciplina *Linguagens Formais e Autômatos* tem por objetivo conduzir o aluno ao conhecimento das classes das linguagens compreendidas pela Hierarquia de Chomsky e das características estruturais de tais linguagens, bem como das gramáticas que as geram.

O estudo das Linguagens Regulares e das Linguagens Livres de Contexto fornece subsídios para o estudo da compilação de linguagens de programação de alto nível.

Serão apresentados os dispositivos reconhecedores das Linguagens Regulares e das Linguagens Livres de Contexto, a saber: Autômatos Finitos e Autômatos de Pilha, respectivamente.

A disciplina desenvolve-se em duas unidades.

A unidade I diz respeito ao estudo das Linguagens Regulares, e tem por objetivos específicos:

- discutir o conceito de autômatos finitos e mostrar que são reconhecedores de linguagens regulares;
- identificar uma linguagem regular representada através de expressões regulares e projetar autômatos finitos determinísticos e não determinísticos que realizem o reconhecimento das mesmas.
- identificar qual linguagem regular é reconhecida por um determinado autômato finito.

A unidade II, que apresenta a Linguagem Livre de Contexto, tem por objetivos específicos:

- aduzir o conceito de gramáticas livres de contexto, dependentes de contexto e irrestritas;
- mostrar que um autômato de pilha é um dispositivo reconhecedor de uma linguagem gerada por uma gramática livre de contexto;
- explicar, pelo menos, um algoritmo de análise sintática (*top-down* ou *botton-up*);
- fazer uma breve explanação das classes das Linguagens Sensíveis a Contexto e Recursivamente Enumeráveis e compará-las com as Linguagens Regulares e Livres de Contexto.

INTRODUÇÃO

O estudante desta disciplina já deve ter se debruçado sobre as tarefas de desenvolver um programa em uma linguagem de programação e ter obtido o resultado do processamento do mesmo em um computador.

As linguagens de programação como C#, Java e outras são notações para descrever computações por pessoas e máquinas. Os sistemas de *software* que fazem a tradução das linguagens de programação para os computadores são denominados compiladores (AHO *et al.*, 2008).

O compilador, antes de executar a função de tradução propriamente dita, realiza três tarefas, a saber: Análise Léxica, Análise Sintática e Análise Semântica.

Na Análise Léxica, o compilador, entre outros procedimentos, lê um fluxo de caracteres que compõem o programa fonte (normalmente descrito em uma linguagem de programação de alto nível) e os agrupa em sequências significativas denominadas lexemas. Esses lexemas dizem respeito a variáveis, constantes, palavras reservadas, operadores aritméticos, lógicos etc.

Na Análise Sintática, o compilador verifica se estruturas sintáticas, como por exemplo, uma **classe** ou os comandos ***if*** e ***while*** etc., estão corretas.

A Análise Semântica realiza tarefas como a verificação de concordância entre a declaração de tipos das variáveis e uso das mesmas, verificação da concordância entre o número de parâmetros na declaração de um método de uma classe e o número de argumentos no uso do método por um objeto etc.

Assim, as Análises Léxica, Sintática e Semântica estão associadas ao processamento das componentes Regular, Livre de Contexto e Dependente de Contexto, respectivamente, da Linguagem de Programação.

Noam Chomsky, estudioso das Linguagens Naturais, propôs uma classificação de linguagens. Tal classificação é conhecida como a Hierarquia de Chomsky e distingue quatro classes, a saber:

- Linguagens do tipo 0 ou recursivamente enumeráveis;
- Linguagens do tipo 1 ou sensíveis a contexto;
- Linguagens do tipo 2 ou livres de contexto;
- Linguagens do tipo 3 ou regulares.

Linguagens Formais e Autômatos é um tópico que há muitos anos faz parte dos currículos de Computação e consiste basicamente no estudo da Hierarquia de Chomsky. O estudo das componentes regular e livre de contexto é de particular interesse para o projeto e a implementação das Linguagens de Programação, ou seja, para o projeto de Compiladores.

Assim, inicialmente apresentar-se-ão algumas definições sobre conjuntos e relações, as quais são fundamentais para a apresentação que se sucede dos conceitos de Linguagens Formais e Gramáticas. Prossegue-se, então, com a apresentação das Linguagens Regulares, a classe mais simples, prevista na Hierarquia de Chomsky.

Em seguida, serão apresentadas as Linguagens Livres de Contexto e também uma breve explanação das Linguagens Sensíveis a Contextos e as Recursivamente Enumeráveis.

Unidade I

1 A CLASSE DAS LINGUAGENS REGULARES — CONCEITOS FUNDAMENTAIS

Neste item são apresentados alguns conceitos sobre conjuntos e relações sobre eles. Naturalmente, o objetivo não é esgotar o assunto, mas sim fornecer subsídios para o conteúdo que se apresenta posteriormente nesta disciplina.

1.1 Conjuntos

Um conjunto é uma coleção de objetos, os quais são denominados como elementos, átomos ou símbolos.

Exemplo: A = {casa, parque, árvore}. A é uma coleção de três palavras da língua portuguesa.

Exemplo: B = {0, 1}. B é uma coleção de dois dígitos.

Exemplo: C = {!, @, #, %}. C é uma coleção de quatro caracteres.

Para representar que ! pertence a C, escreve-se ! $\in$ C. Para representar que ? não pertence a C, escreve-se ? $\notin$ C.

Não existe uma relação de ordem entre os elementos de um conjunto, e é irrelevante o número de ocorrências de um mesmo átomo em um conjunto. Assim sendo, tem-se que:

A = {a, b, c} = {c, a, b} = {c, a, a, b, b, b}

Os conjuntos podem ser finitos ou infinitos.

Se os conjuntos forem finitos, eles podem ser representados pela enumeração de seus elementos, separados entre si por vírgulas e delimitados por chaves.

Um conjunto sem elementos é denominado conjunto vazio e é denotado como { } ou $\varnothing$.

Os conjuntos podem ser infinitos. São exemplos de conjuntos infinitos o conjunto **N** dos números naturais, o conjunto **Z** dos números inteiros etc.

Os conjuntos também podem ser representados através da especificação de suas propriedades, tal como nos exemplos que se seguem:

Exemplo: $X = \{x \mid x \in N \text{ e } 0 < x < 10\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Exemplo: $A = \{x \mid x \in N \text{ e } x > 2\}$

Exemplo: O conjunto dos números pares pode ser denotado como:

$$\{x \mid x = 2n \text{ e } n \in N\}$$

A é um subconjunto de B, se cada elemento de A é também um elemento de B. Representa-se por: $A \subseteq B$ (Lê-se: "A está contido em B"). Note que qualquer conjunto é subconjunto de si mesmo. Diz-se que A é um **subconjunto próprio** de B se A é **subconjunto** de B, mas não coincide com o conjunto B. Neste caso, representa-se $A \subset B$ (Lê-se: "A está contido propriamente em B"). O conjunto vazio é subconjunto de qualquer conjunto.

Exemplo: Sejam $A = \{1, 2, 3, 4, 10, 20, 30, 40\}$ e $B = \{1, 10, 2, 20\}$, pode-se afirmar que: $B \subseteq A$, $B \subset A$, $A \subseteq A$, $B \subseteq B$.

Dois **conjuntos são iguais** se, e somente se, $A \subseteq B$ e $B \subseteq A$.

Dois conjuntos A e B quaisquer podem ser combinados e formar um terceiro através de várias operações sobre conjuntos. As principais operações são:

a) **União**: Sejam A e B dois conjuntos, então $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$.

Exemplo: Sejam $A = \{0, 1, 2, 6, 7\}$ e $B = \{0, 3, 5, 7, 8, 9\}$. Tem-se que:

$$A \cup B = \{0, 1, 2, 3, 5, 6, 7, 8, 9\}$$

b) **Intersecção:** Sejam A e B dois conjuntos, então $A \cap B = \{x \mid x \in A \text{ e } x \in B\}$.

Exemplo: Sejam $A = \{0, 1, 2, 6, 7\}$ e $B = \{0, 3, 5, 7, 8, 9\}$. Tem-se que:

$$A \cap B = \{0, 7\}$$

c) **Diferença:** Sejam A e B dois conjuntos, então $A - B = \{x \mid x \in A \text{ e } x \notin B\}$.

Exemplo: Sejam $A = \{0, 1, 2, 6, 7\}$ e $B = \{0, 3, 5, 7, 8, 9\}$. Tem-se que:

$$A - B = \{1, 2, 6\}$$

$$B - A = \{3, 5, 8, 9\}$$

d) **Complementação:** A operação de complementação é definida em relação a um conjunto fixo **U** denominado **conjunto universo**. Seja A um conjunto, então a complementação de A é $A' = \{x \mid x \in \mathbf{U} \text{ e } x \notin A\}$.

Exemplo: Seja A = {0, 1, 2, 3} e U = N (conjunto dos números naturais). Tem-se que:

$$A' = \{x \in \mathbf{N} \mid x > 3\}$$

e) **Conjunto Potência:** Seja A um conjunto. Tem-se que o conjunto potência é definido como:

$$2^A = \{ S \mid S \subseteq A\}$$

Exemplo: Seja A = {1, 2, 3}, tem-se que:

$$2^A = \{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

Observação

O conjunto potência de A é o conjunto de todos os subconjuntos de A.

f) **Produto Cartesiano:** Sejam A e B dois conjuntos. O produto cartesiano A × B define-se como:

$$A \times B = \{(a, b) \mid a \in A \text{ e } b \in B\}$$

O elemento do produto cartesiano (a, b) é denominado par ordenado, ou seja, a ordem em que os componentes a e b se apresentam é relevante.

Exemplo: Sejam A = {1, 2} e B = {x, y, z}. Tem-se que:

$$A \times B = \{(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)\}$$

$$B \times A = \{(x, 1), (x, 2), (y, 1), (y, 2), (z, 1), (z, 2)\}$$

$$A \times A = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$$

$$B \times B = \{(x, x), (x, y), (x, z), (y, x), (y, y), (y, z), (z, x), (z, y), (z, z)\}$$

1.2 Relações

Sejam dois conjuntos A e B: uma **relação** R sobre dois conjuntos (binária) é um subconjunto de um produto cartesiano A × B.

Exemplo: Sejam A = {1, 2, 3} e B = {2, 4, 6}. O conjunto {(1,2), (2,4), (3, 6)} é formado por elementos de A × B, logo é uma relação binária de A para B.

Exemplo: Sejam Q = $\{q_1, q_2, q_3\}$ e Σ = {x, y}. O conjunto $\{(q_1, x), (q_2, x), (q_3, x), (q_3, y)\}$ é formado por elementos de Q × Σ, logo é uma relação binária de Q para Σ.

Exemplo: Sejam os conjuntos R = $\{(q_1, x), (q_1, y), (q_2, x), (q_3, x), (q_3, y)\}$ e Q = $\{q_1, q_2, q_3\}$. O conjunto $\{((q_1, x), q1), ((q_1, y), q2), ((q_2, x), q1), ((q_3, x), q2), ((q_3, y), q1)\}$ é formado por elementos de R × Q, logo é uma relação binária de R para Q.

2 ALFABETOS, CADEIAS, LINGUAGENS E GRAMÁTICAS

2.1 Definições de Alfabeto, Cadeias, Linguagens

Alfabeto é um conjunto finito de símbolos. Cada símbolo é, portanto, uma unidade atômica empregada na construção de cadeias e se trata de um conceito primitivo, sendo a sua representação visual irrelevante.

São exemplos de alfabetos:

$\sum_1 = \{a, b, o, d\}$

$\sum_2 = \{0, 1\}$

$\sum_3 = \{\$, @, \#\}$

$\sum_4 = \{\ \} = \varnothing$

Uma **cadeia de caracteres** é uma sequência **finita** de símbolos de um alfabeto justapostos.

Considere o alfabeto $\sum_1 = \{a, b, o, d\}$.

São exemplos de cadeias definidas sobre o alfabeto $\sum_1$ as seguintes sequências de caracteres (ou símbolos) justapostos: abba, oba, dado, boda, abad, boa, boba, bddb, aaaa, a.

Uma **cadeia vazia** é uma cadeia sem símbolos, e é representada pelo símbolo ε.

O **comprimento de uma cadeia** ω é o número de símbolos justapostos que se apresentam na mesma. Representa-se como |ω|. Exemplos:

Para ω = aaaa, tem-se que |ω| = 4.

Para ω = a, tem-se que |ω| = 1.

Para ω = ε, tem-se que |ω| = 0.

Duas cadeias definidas sobre o mesmo alfabeto podem ser combinadas e formar uma terceira, a partir da operação de **concatenação**. A concatenação das cadeias *v* e ω é representada por *v*ω e formada pela justaposição de *v* e ω.

Exemplo:

Seja o alfabeto $\Sigma = \{a, s, m, o, r\}$.

Seja ω_1 = amor e ω_2 = as. Tem-se que $\omega_1\omega_2$ = amoras.

A operação de concatenação é associativa, ou seja, (uv)w = u(vw) quaisquer que sejam as cadeias u, v e w.

Uma cadeia v é uma **subcadeia** de uma cadeia ω se, e somente se, houver cadeias x e y, tal que ω = xvy.

Um **prefixo** (respectivamente, **sufixo**) de uma cadeia é qualquer sequência de símbolos inicial (respectivamente, final) da cadeia.

Considere, por exemplo, ω = amoras. São prefixos da cadeia ω: a, am, amo, amor, amora e a própria cadeia amoras. São sufixos de ω: s, as, ras, oras, moras e a própria cadeia amoras.

Para cada cadeia ω e cada número natural i, define-se ω^i, conforme se segue:

$$\omega^0 = \varepsilon \text{ (cadeia vazia)}$$

$$\omega^{i+1} = \omega^i \omega$$

Exemplo: Sejam o alfabeto $\Sigma = \{a, b\}$ e a cadeia ω = ba. Tem-se que:

$\omega^3 = \omega\omega\omega$ = bababa

$\omega^1 = \omega$ = ba

$\omega^0 = \varepsilon$

O **reverso de uma cadeia** ω, denotada por ω^R, é definido como:

- se $\omega = \varepsilon$, então $\omega R = \omega = \varepsilon$. Note que, neste caso, o comprimento da cadeia ω é 0.
- se ω é uma cadeia de comprimento n+1 > 0, então ω = ua para algum $a \in \Sigma$ e w e $w^R = au^R$.

Exemplos:

Sejam o alfabeto $\Sigma = \{a, b\}$ e ω = ba. Tem-se que ω^R = ab.

Sejam o alfabeto $\Sigma = \{a, i, m, r\}$ e ω = rima. Tem-se que ω^R = amir

Linguagem formal é um conjunto finito ou infinito, de cadeias de comprimento finito, sobre um alfabeto finito e não vazio.

Seja o alfabeto $\Sigma = \{0, 1\}$. A seguir, são apresentados exemplos de linguagens cujas cadeias são definidas sobre o alfabeto Σ.

a) $L_1 = \varnothing$

b) $L_2 = \{\varepsilon\}$

c) $L_3 = \{1, 01, 10, 11, 111\}$

Além das operações definidas para conjuntos, definem-se a seguir as operações de concatenação e fechamentos.

Sejam L_1, L_2 duas linguagens, a **concatenação** $L_1.L_2$, ou simplesmente L_1L_2 é definida formalmente como:

$$L_1L_2 = \{w = w_1w_2 \mid w_1 \in L_1 \text{ e } w_2 \in L_2\}$$

O **fechamento recursivo** e **transitivo** de um alfabeto Σ é definido como o conjunto infinito que contém todas as possíveis cadeias que podem ser construídas sobre o alfabeto dado, incluindo a cadeia vazia, e é denotado por Σ^*.

O **fechamento transitivo** denotado por Σ^+ representa o conjunto de todas as cadeias sobre o alfabeto Σ, excetuando-se a cadeia vazia, ou seja:

$$\Sigma^+ = \Sigma^* - \{\varepsilon\}$$

Exemplo: Seja o alfabeto unário $\Sigma = \{ 1 \}$, tem-se que:

$$\Sigma^* = \{\varepsilon, 1, 11, 111, 1111, 11111, ...\}$$

$$\Sigma^+ = \{1, 11, 111, 1111, 11111, ...\}$$

A **complementação** de uma linguagem L definida sobre um alfabeto é:

$$L' = \Sigma^* - L$$

2.2 Gramática: dispositivo gerador de uma linguagem

Gramáticas são dispositivos geradores das cadeias que pertencem a uma linguagem.

Formalmente, uma gramática é uma quádrupla $G = (V, \Sigma, P, S)$, onde:

V = conjunto finito de símbolos não terminais.

Σ = conjunto finito de símbolos terminais da gramática. Esse conjunto também é denominado **alfabeto**.

$S \in V$ = é denominado símbolo inicial ou **raiz** da gramática.

P = conjunto de produções ou regras de substituição da gramática.

Uma **regra de produção** é representada da seguinte forma:

$\alpha \rightarrow \beta$, onde $\alpha \in (V \cup \Sigma)^+$ e $\beta \in (V \cup \Sigma)^*$.

Naturalmente, $\alpha \in (V \cup \Sigma)^+$ informa que, no lado esquerdo da produção, deve existir um ou mais símbolos, terminais ou não terminais. Analogamente, a expressão $\beta \in (V \cup T)^*$ indica que, no lado direito da produção, podem figurar quaisquer cadeias de símbolos terminais, não terminais e até mesmo isoladamente, a cadeia vazia.

Exemplo: Seja $G_1 = (V, \Sigma, P, S)$, onde:

$$V = \{S, X, Y\}$$

$$\Sigma = \{a, 0, 1\}$$

$$P = \{S \rightarrow aX$$

$$X \rightarrow a\,X \mid 0X \mid 1\,X \mid \varepsilon\}$$

S é o símbolo inicial da gramática.

Observação

O símbolo "|" é traduzido como "ou". Assim sendo, escrever

$X \rightarrow a\,X \mid 0X \mid 1\,X \mid \varepsilon$

equivale a representar o seguinte conjunto de regras:

$X \rightarrow a\,X$; $X \rightarrow 0\,X$; $X \rightarrow 1\,X$ e $X \rightarrow \varepsilon$

Exemplo:

$G_2 = (V, \Sigma, P, S)$, onde:

$V = \{$Expressão, Termo, Fator$\}$

$\Sigma = \{x, y, +, *\}$

Expressão é o símbolo inicial da gramática.

O conjunto das produções é representado por:

P = {Expressão → Expressão + Termo

Expressão→Termo

Termo → Termo * Fator

Termo → Fator

Fator → x

Fator → y}

2.3 Derivação de cadeias e árvores de derivação

A aplicação de uma regra de produção $\alpha \rightarrow \beta$ é denominada derivação. A derivação é representada pelo símbolo $\Rightarrow$.

Exemplo: Considere a linguagem:

$$L = \{w \mid w \text{ começa com o símbolo } 0\}$$

A gramática geradora da linguagem L é $G = (V, \Sigma, P, S)$, onde: $V = \{S, A\}$, $\Sigma = \{1, 0\}$, S é o símbolo inicial e $P = \{S \rightarrow 0A; A \rightarrow 0A \mid 1A \mid \varepsilon\}$.

Naturalmente, a palavra w = 010 pertence a L, o que pode ser verificado através da seguinte sequência de aplicações das produções:

$$S \Rightarrow 0A \Rightarrow 01A \Rightarrow 010A \Rightarrow 010\,\varepsilon \Rightarrow 010$$

Exemplo: Considere a gramática G_2 apresentada anteriormente. A aplicação da regra Expressão → Termo é representada por:

Expressão ⇒ Termo.

Diz-se: "Expressão deriva Termo".

A seguir, a título de ilustração, são apresentadas aplicações sucessivas das regras da gramática G_2. Para ainda mais claramente se explanar, as produções são apresentadas novamente, enumeradas.

Expressão → Expressão + Termo	(regra 1)
Expressão→Termo	(regra 2)
Termo → Termo * Fator	(regra 3)
Termo → Fator	(regra 4)
Fator → x	(regra 5)
Fator → y	(regra 6)

Considere a seguinte sequência de derivações, a partir do símbolo inicial Expressão.

Expressão ⇒ Expressão + Termo	(aplicação da regra 1)
⇒Termo + Termo	(aplicação da regra 2)
⇒ Termo * Fator + Termo	(aplicação da regra 3)
⇒ Fator * Fator + Termo	(aplicação da regra 4)
⇒ x* Fator + Termo	(aplicação da regra 5)
⇒ x* y + Termo	(aplicação da regra 6)
⇒ x* y + Fator	(aplicação da regra 4)
⇒ x* y + x	(aplicação da regra 5)

Diz-se que a cadeia x*y+x é gerada pela gramática G_1.

Um segundo exemplo de aplicações sucessivas de produções é apresentado a seguir:

Expressão ⇒	Expressão + Termo	(aplicação da regra 1)
⇒	Expressão + Termo + Termo	(aplicação da regra 1)
⇒	Termo + Termo + Termo	(aplicação da regra 2)
⇒	Fator + Termo + Termo	(aplicação da regra 4)
⇒	x + Termo + Termo	(aplicação da regra 5)
⇒	x + Fator + Termo	(aplicação da regra 4)
⇒	x + x + Termo	(aplicação da regra 5)
⇒	x + x + Fator	(aplicação da regra 4)
⇒	x + x + x	(aplicação da regra 5)

Diz-se que a cadeia x+x+x é gerada pela gramática G_1.

Seja G = (V, Σ, P, S) uma gramática. Uma derivação, denotada por "⇒", é uma relação com domínio em $(V \cup \Sigma)+$ e contradomínio em $(V \cup \Sigma)^*$.

Lembrete

Sejam dois conjuntos A e B: uma **relação** R sobre dois conjuntos (binária) é um subconjunto de um produto cartesiano A × B.

A relação ⇒ pode ser definida como:

- para toda a produção da forma $S \rightarrow \beta$, tem-se que: $S \Rightarrow \beta$.
- Sejam $\alpha, \beta, \gamma, \delta$ cadeias de símbolos (terminais ou não terminais da gramática) e $\delta \rightarrow \gamma$ uma regra da gramática, então: $\alpha\ \delta\ \beta \Rightarrow \alpha\ \gamma\ \beta$.

Derivações em que ocorre a aplicação de pelo menos uma produção são denotadas por $\Rightarrow^+$.

Exemplo: Expressão $\Rightarrow^+ x^*\ y + x$

Exemplo: Expressão $\Rightarrow^+ x + x + x$

Uma sequência de zero ou mais derivações é denotada por $\Rightarrow^*$.

Ao conjunto de todas as sentenças ω geradas por uma gramática G dá-se o nome de **linguagem definida pela gramática G** ou, simplesmente, L(G). Tem-se que:

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

Diz-se que duas gramáticas G_1 e G_2 são equivalentes se, e somente se, $L(G_1) = L(G_2)$.

Exemplo: Seja $G_1 = \{V_1, \Sigma, P_1, S_1\}$, onde:

$V_1 = \{S_1\}$

$\Sigma = \{x, y\}$

$P_1 = \{S_1 \rightarrow S_1 y | x\}$ e S_1 é o símbolo inicial da gramática.

Considere-se também a gramática $G_2 = \{V_2, \Sigma, P_2, S_2\}$, onde:

$V_2 = \{S_2, F\}$

$\Sigma = \{x, y\}$

$P_2 = \{S_2 \rightarrow xF; F \rightarrow yF \mid \varepsilon\}$

Tem-se que G_1 é equivalente a G_2, pois ambas geram a linguagem:

$L = \{\omega \mid \omega = xy^*\}$

De fato, tem-se que:

$S_1 \Rightarrow x$

$S_1 \Rightarrow S_1 y \Rightarrow xy$ e, portanto, $S_1 \Rightarrow^2 xy$

$S_1 \Rightarrow S_1 y \Rightarrow S_1yy \Rightarrow xyy$ e, portanto, $S_1 \Rightarrow^3 xy^2$

$S_1 \Rightarrow S_1 y \Rightarrow S_1yy \Rightarrow S_1yyy \Rightarrow xyyy$ e, portanto, $S_1 \Rightarrow^4 xy^3$ $S_1 \Rightarrow^{n+1} xy^n$, $n \geq 0$

Lembrete

A **concatenação** $L_1.L_2$, ou simplesmente L_1L_2, é definida formalmente como: $L_1L_2 = \{w = w_1w_2 \mid w_1 \in L_1 \text{ e } w_2 \in L_2\}$

Por outro lado, tem-se que:

$S_2 \Rightarrow xF \Rightarrow x\varepsilon = x$ e, portanto, $S_2 \Rightarrow^2 x$

$S_2 \Rightarrow xF \Rightarrow xyF \Rightarrow xy\varepsilon = xy$ e, portanto, $S_2 \Rightarrow^3 xy$

$S_2 \Rightarrow xF \Rightarrow xyF \Rightarrow xyyF \Rightarrow xyy\varepsilon$ e, portanto, $S_2 \Rightarrow^4 xyy$

Assim, $S_2 \Rightarrow^{n+2} xy^n$, $n \geq 0$.

3 LINGUAGENS REGULARES — PARTE I

3.1 A Hierarquia de Chomsky

Noam Chomsky, estudioso das linguagens naturais e professor desde 1955 no *Massachusetts Institute Technology*, desenvolveu a Gramática Transformacional ou Gramática Gerativa, pela qual a linguagem é um sistema de conhecimentos interiorizados na mente humana. Por outro lado, Raposo (1992) afirma que a linguagem é para Chomsky um **sistema formal** interpretado e contempla a gramática (das linguagens naturais) como um sistema computacional.

Chomsky propôs uma classificação de linguagens que, segundo Ramos (2009), exibe o mérito de agrupar as linguagens em classes, de tal forma que elas possam ser hierarquizadas segundo a sua complexidade relativa.

Os estudos linguísticos foram objeto de interesse em diversas áreas do conhecimento e, em particular, na Ciência da Computação.

A Hierarquia de Chomsky define quatro classes distintas de linguagens.

- Linguagens do tipo 0 ou recursivamente enumeráveis.
- Linguagens do tipo 1 ou sensíveis a contexto.
- Linguagens do tipo 2 ou livres de contexto.
- Linguagens do tipo 3 ou regulares.

A hierarquia das linguagens definida por Chomsky apresenta uma relação de inclusão que pode ser representada graficamente, de acordo com a figura 1.

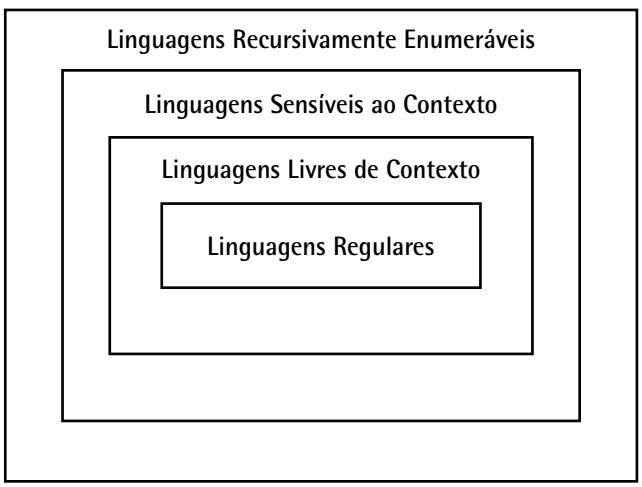

Figura 1 – Hierarquia de Chomsky: relação de inclusão entre as classes das linguagens

A classe da linguagem do tipo 3 está incluída propriamente na classe do tipo 2. Assim, toda Linguagem Regular também é Livre de Contexto.

A classe da linguagem do tipo 2 está incluída propriamente na classe do tipo 1. Toda linguagem Livre de Contexto também é Sensível a Contexto.

A classe da linguagem do tipo 1 está incluída propriamente na classe do tipo 0. Toda linguagem Sensível a Contexto também é recursivamente enumerável.

Esta Unidade apresenta as Linguagens Regulares, que, na Hierarquia de Chomsky, são as de menor complexidade.

As Linguagens Regulares são geradas pelas Gramáticas Regulares.

3.2 Gramáticas Regulares: dispositivos geradores das Linguagens Regulares

Seja G = (V, Σ, P, S) uma gramática e sejam A e B símbolos não terminais e ω uma cadeia de Σ*:

a) G é uma gramática linear à direita, se todas as produções são da forma:

$$A \rightarrow \omega B \text{ ou } A \rightarrow \omega.$$

b) G é uma gramática linear à esquerda, se todas as produções são da forma:

$$A \rightarrow B\omega \text{ ou } A \rightarrow \omega.$$

Uma Gramática Regular é qualquer Gramática Linear.

Exemplo: A gramática G_1 é linear à direita.

$G_1 = (V, \Sigma, P, S)$, onde:

$V = \{S, F\}$

$\Sigma = \{a, b\}$

$P = \{ S \rightarrow abF$

$F \rightarrow abF \mid ab\}$

Exemplo: A gramática G_2 é linear à esquerda.

$G_2 = (V, \Sigma, P, S)$, onde:

$V = \{S\}$

$\Sigma = \{a, b\}$

$P = \{S \rightarrow Sab \mid ab\}$

S é o símbolo inicial.

Os seguintes teoremas mostram que a Classe das Gramáticas Regulares denota exatamente a Classe das Linguagens Regulares.

Teorema 1: Se L é uma linguagem gerada por uma Gramática Regular, então L é uma Linguagem Regular.

Teorema 2: Se L é uma Linguagem Regular, então existe G, Gramática Regular que gera L.

Exemplo: Considere-se A Gramática G_3 Linear à direita.

$G_3 = (V, \Sigma, P, S)$, onde:

$V = \{S, X, Y, F\}$

$\Sigma = \{a, b\}$

S é o símbolo inicial.

$P = \{S \rightarrow aX$

$X \rightarrow bF$

$F \rightarrow aY \mid \varepsilon$

$Y \rightarrow bF\}$

É fácil constatar, essa gramática gera a seguinte linguagem:

$$L(G_3) = \{ab, abab, ababab, abababab, ...\}$$

De fato, tem-se que:

$S \Rightarrow aX \Rightarrow abF \Rightarrow ab\varepsilon = ab$

$S \Rightarrow aX \Rightarrow abF \Rightarrow abaY \Rightarrow ababF \Rightarrow abab\varepsilon = abab$

$S \Rightarrow aX \Rightarrow abF \Rightarrow abaY \Rightarrow ababaY \Rightarrow abababF \Rightarrow ababab\varepsilon = ababab$

Uma vez que G_3 é regular, a linguagem $L(G_3)$ é regular.

Exemplo: A Gramática G_4 é linear à esquerda.

$G_4 = (V, \Sigma, P, S)$, onde:

$V = \{S, X, Y\}$

$T = \{a, b\}$

$P = \{S \rightarrow Xb$

$X \rightarrow Ya \mid a$

$Y \rightarrow Xb \mid \varepsilon\}$

S é o símbolo inicial.

A gramática G_4 é equivalente a G_3 e também gera a linguagem regular:

$L(G_4) = \{ab, abab, ababab, abababab,...\}$

De fato, considere-se:

$S \Rightarrow Xb \Rightarrow Yab \Rightarrow \varepsilon ab = ab$

$S \Rightarrow Xb \Rightarrow Yab \Rightarrow Xb\,ab \Rightarrow Yabab \Rightarrow \varepsilon abab = abab$

$S \Rightarrow Xb \Rightarrow Yab \Rightarrow Xbab \Rightarrow Yabab \Rightarrow Xbabab \Rightarrow Yababab \Rightarrow \varepsilon ababab = ababab$

3.3 Expressões Regulares

As expressões regulares, notação desenvolvida por Kleene na década de 1950, constituem-se em uma alternativa para representar as linguagens regulares.

Uma expressão regular sobre um alfabeto $\sum$ é indutivamente definida como se segue:

- o conjunto vazio $\varnothing$, que é uma linguagem vazia, é uma expressão regular;
- a cadeia vazia ε é uma expressão regular e, portanto, a linguagem $L = \{\varepsilon\}$ é regular;
- qualquer símbolo $x \in \sum$ é uma expressão regular e, portanto, a linguagem $L = \{x\}$ é regular;
- se r e s são expressões regulares e, consequentemente, as linguagens R e S são regulares, então:
  - (r | s) é uma expressão regular e a linguagem $R \cup S$ é regular.
  - (rs) é uma expressão regular e a linguagem $RS = \{xy \mid x \in R \text{ e } y \in S\}$ é regular.
  - (r*) é uma expressão regular.

Exemplo: A expressão regular a* representa a linguagem:

$$L_1 = \{\varepsilon, a, aa, aaa,...\}$$

Exemplo: A expressão regular (b|a*) representa a linguagem:

$$L_2 = \{\varepsilon, b, a, aa, aaa,...\}$$

Exemplo: A expressão regular (ba*) representa a linguagem:

$$L_3 = \{b, ba, baa, baaa, baaaa,...\}$$

Exemplo: A expressão regular (a | ba*) representa a linguagem:

$$L_4 = \{a, aa, aaa, aaaa, b, ba, baa, baaa, baaaa,...\}$$

Exemplo: A expressão regular a(b|c)* representa a linguagem:

$$L_5 = \{a, ab, ac, abb, acc, abc, acb, abbc,...\}$$

Exemplo: A expressão regular (a | ba)* representa a linguagem:

$$L_6 = \{\varepsilon, ba, a, baa, aba, baba, aa,...\}$$

Exemplo: A expressão regular $(ab)^+$ representa a linguagem:

$$L_7 = \{ab, abab, ababab,...\}$$

3.4 Autômatos Finitos — dispositivos reconhecedores de Linguagens Regulares

Os autômatos finitos são formalismos de aceitação das sentenças das Linguagens Regulares, ou seja, o autômato finito aceita toda e qualquer cadeia pertencente à linguagem para o qual foi projetado e rejeita todas as cadeias não pertencentes à mesma.

Os autômatos finitos apresentam os seguintes componentes:

- Fita de entrada

Trata-se de um dispositivo de armazenamento, uma memória, que contém a cadeia a ser analisada pelo reconhecedor. A fita é finita, dividida em células, e cada célula armazena um símbolo da cadeia de entrada. O comprimento da **fita de entrada**, portanto, é igual ao comprimento da cadeia de entrada. A leitura dos símbolos gravados na fita de entrada é efetuada mediante o uso de um **cursor**, o qual sempre aponta o próximo símbolo da cadeia a ser processado. Não há operações de escrita sobre a fita. O cursor movimenta-se exclusivamente da esquerda para a direita.

- Unidade de controle finito ou máquina de estados

Trata-se de um controlador central do reconhecedor. A unidade de controle dispõe da especificação de movimentações possíveis do cursor, descritas por um conjunto finito de **estados** e **transições**. O conceito de estado diz respeito ao registro de informações capturadas no passado e relevantes para o posterior processamento da cadeia de entrada. Inicialmente, o cursor aponta para o símbolo mais à esquerda da cadeia. Nessa configuração, em que a cadeia completa ainda será analisada, o controlador encontra-se no **estado inicial**, que deve ser único. Uma transição em um autômato finito é definida pela tripla (estado corrente, símbolo corrente, próximo estado). O símbolo pode ser cadeia vazia ε ou qualquer elemento do alfabeto da cadeia de entrada sobre o qual a linguagem é definida. Considere a figura seguinte:

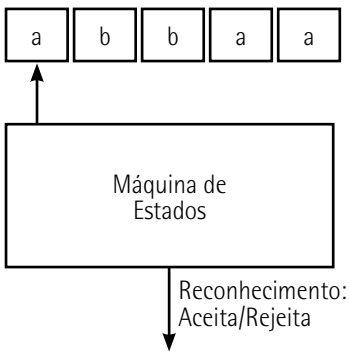

Figura 2 - Estrutura de um autômato finito

Os autômatos finitos podem ser determinísticos ou não determinísticos.

A seguir, apresenta-se a definição formal para autômatos finitos determinísticos.

Um Autômato Finito Determinístico (AFD) ou simplesmente um Autômato Finito (AF) M é uma quíntupla:

$$M = (Q, \Sigma, g, q0, F)$$

- Q é um conjunto finito de estados;
- Σ é um alfabeto (finito e não vazio) de entrada;
- g é uma função de transição, g: $Q \times \Sigma \rightarrow Q$;
- q0 é o estado inicial, $q0 \in Q$;
- F é um conjunto de estados finais, $F \subseteq Q$.

Os autômatos podem ser representados, algebricamente, como anteriormente descrito ou visualmente por um diagrama de transição de estados, um grafo orientado, rotulado nos vértices com os nomes dos estados e nos arcos, com os símbolos dos alfabetos de entrada do autômato finito.

Exemplo: Seja M um autômato finito determinístico (Q, Σ, g, q0, F), onde:

$$Q = \{q0, q1, q2\}$$

$$\Sigma = \{a, b, c\}$$

q0 é o estado inicial

$$F = \{q2\}$$

e a função g pode ser apresentada como na tabela a seguir:

Tabela 1

g	a	b	c
q0	q1	-	-
q1	-	q2	-
q2	-	-	q2

Pode-se também denotar a função g, como:

$$g= \{((q0, a), q1), ((q1, b), q2), ((q2, c), q2)\}$$

Ainda, a função g pode ser especificada como:

$$g(q0, a) = g1; g(q1, b) = q2; g(q2, c) = q2.$$

M pode ser representado como na figura seguinte.

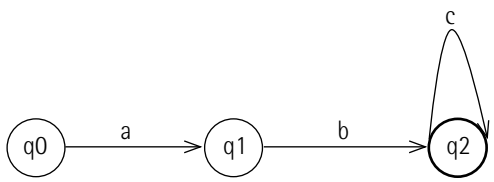

Figura 3 – Autômato Finito Reconhecedor da Linguagem $L_1 = \{\omega \mid \omega = abc^*\}$

Através da observação da figura 3, é possível identificar qual é a Linguagem L_1 definida pelo autômato M. Tem-se que:

$$L_1 = \{\omega \mid \omega = abc^*\}$$

Considere a cadeia ω_1 = abcc $\in L_1$

- O processamento da cadeia começa no estado inicial q0, e o cursor aponta para o símbolo mais à esquerda da cadeia, ou seja, o símbolo **a**.

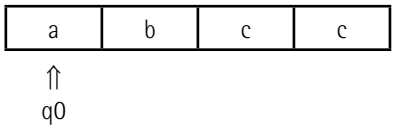

Figura 4

- Após a leitura do símbolo **a**, o cursor deve ser movido para a próxima célula à direita. A função de transição indica que, para o par (q0, a), o estado que se sucede é o estado q1.

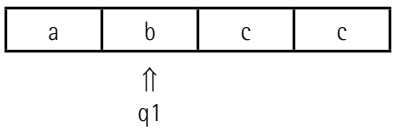

Figura 5

- Após a leitura do símbolo **b**, o cursor deve ser movido para a próxima célula à direita. A função de transição indica que, para o par (q1, a), o estado que se sucede é o estado q2.

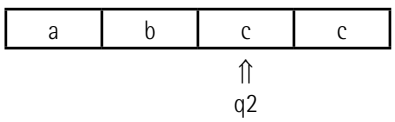

Figura 6

- Após a leitura do símbolo **c**, novamente o cursor deve ser movido para a próxima célula à direita. A função de transição indica que, para o par (q2, c), o estado se mantém em q2.

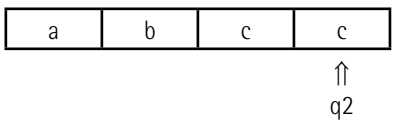

Figura 7

- Na cadeia de entrada, novamente ocorre o símbolo **c**. Assim, após a leitura do mesmo, o cursor é movido à direita. O estado do reconhecimento se mantém em q2.

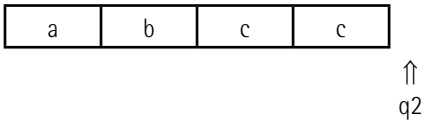

Figura 8

- O autômato atingiu a configuração final, visto que o estado q2 é um estado final e o cursor aponta para a direita do último símbolo da cadeia de entrada. Nessa configuração, diz-se que o autômato finito aceita a cadeia de entrada, ou seja, a cadeia de entrada pertence à linguagem reconhecida pelo autômato.

Cumpre salientar que a cadeia abcc foi aceita pelo autômato porque seu reconhecimento foi finalizado e o autômato alcançou o estado final q2.

Considere-se a cadeia ω = abbc $\notin L_1$. O processamento dessa cadeia pode ser descrito como se segue:

- O processamento da cadeia começa no estado inicial q0 e o cursor aponta para o símbolo mais à esquerda da cadeia, ou seja, o símbolo **a**.

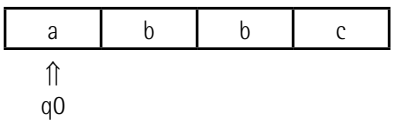

Figura 9

- Após a leitura do símbolo **a**, o cursor deve ser movido para a próxima célula à direita. A função de transição indica que, para o par (q0, a), o estado que se sucede é o estado q1.

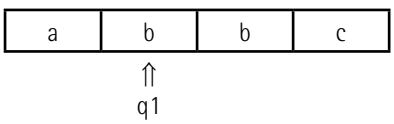

Figura 10

- Após a leitura do símbolo **b**, o cursor deve ser movido para a próxima célula à direita. A função de transição indica que, para o par (q1, b), o estado que se sucede é o estado q2.

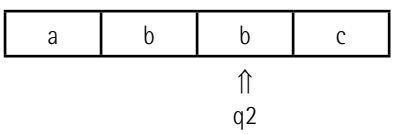

Figura 11

- Observe que o reconhecimento não pode prosseguir após a leitura do símbolo **b** em sua segunda ocorrência, visto que a função de transição não contempla uma transição para outro estado para o par (q2, b).

- Assim sendo, diz-se que o autômato M rejeita a cadeia de entrada abbc.

Finalmente, considere-se a cadeia de entrada ω=a.

- De forma análoga aos casos anteriores, o processamento da cadeia começa no estado inicial q0, e o cursor aponta para o símbolo mais à esquerda da cadeia, ou seja, o símbolo **a**.

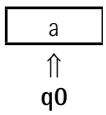

Figura 12

- Após a leitura do símbolo **a**, o cursor é movido à direita, com o autômato no estado q1.

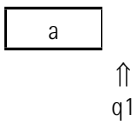

Figura 13

- Obviamente, o processamento da cadeia é finalizado, visto que é atingido o final da mesma. Note-se, no entanto, que o autômato não atinge um estado final. Nessa configuração, ele também rejeita a cadeia de entrada ω =a.

Observe que o autômato sempre para ao processar qualquer que seja a cadeia de entrada, pois apenas uma das seguintes condições ocorre:

a) a cadeia é processada até o último símbolo e o autômato assume um estado final. Trata-se da configuração de aceitação;

b) após o processamento do último símbolo da fita, o autômato finito assume um estado não final. O autômato para e a cadeia de entrada é rejeitada;

c) a função de transição é indefinida para um símbolo da cadeia de entrada. O autômato para e a cadeia de entrada é rejeitada.

Retomando a definição formal de um autômato finito, observa-se que a função de transição é tal que:

$$g: Q \times \Sigma \rightarrow Q$$

Em outras palavras, o domínio da função é o produto cartesiano $Q \times \Sigma$, ou seja, o conjunto de todos os pares (q, x), tal que $q \in Q$ e $x \in \Sigma$.

Essa definição de transição permite, portanto, aplicar a função de transição apenas a um símbolo do alfabeto.

Considere, para o exemplo comentado, a função g, onde:

$$g = \{((q0, a), q1), ((q1,b), q2), ((q2, c), q2)\}$$

Tem-se que: g(q0,a) = q1, g(q1,b) = q2, g(q2, c) = q2.

Pode-se estender a definição da função g, de forma que esta possa ser aplicada a uma cadeia. Para isso, o domínio da função g deve contemplar o conjunto de todas as cadeias definidas a partir do alfabeto, ou seja, Σ^*.

Seja M = (Q, Σ, g, q0, F) um autômato finito determinístico, a função de transição estendida é denotada por:

$$\underline{g}: Q \times \Sigma^* \rightarrow Q$$

A função $\underline{g}$ é definida indutivamente como se segue:

$\underline{g}(q, \varepsilon) = q$

$\underline{g}(q, a\omega) = \underline{g}(g(q, a), \omega)$

Exemplo: Considere o autômato finito M = ({q0, q1, q2}, {a, b, c}, g, q0, {q2}), com a função de transição dada por:

$$g= \{((q0, a), q1), ((q1, b), q2), ((q2, c), q2)\}$$

Tem-se que a função de transição estendida aplicada à cadeia ω_1 = abcc, a partir do estado inicial q0, é:

$$\underline{g}(q0, abcc) = \underline{g}(g(q0, a), bcc) = \underline{g}(q1, bcc) = \underline{g}(g(q1,b), cc)= \underline{g}(q2,cc) = \underline{g}(g(q2,c),c) = \underline{g}(q2,c) = \underline{g}(g(q2,c), \varepsilon) = \underline{g}(q2,\varepsilon) = q2 \in F$$

Uma vez que a extensão da definição da função g foi introduzida, pode-se enunciar formalmente a definição de uma linguagem reconhecida por um autômato finito determinístico.

A linguagem aceita por um autômato finito M = (Q, Σ, g, q0, F), denotada por L(M), é o conjunto de todas as cadeias pertencentes a Σ^* aceitas por M, ou seja:

$$L(M) = \{\omega \in \Sigma^* | \underline{g}(q0,\omega) \in F\}$$

Uma linguagem aceita por um autômato finito determinístico é uma Linguagem Regular ou do tipo 3.

Exemplo: Sejam Σ = {x, y} e L uma linguagem definida sobre o alfabeto Σ, tal que:

$$L= \{\omega \mid \omega \text{ apresenta a subcadeia xyxy}\}$$

A representação algébrica para o autômato finito é:

$Q = \{q0, q1, q2, q3, q4\}$

$\Sigma = \{x, y\}$

$g = \{((q0, x), q1), \{((q0, y), q0), ((q0, x), q1), ((q1, x), q1), ((q1, y), q2),$

$((q2, x), q3), ((q2, y), q0), ((q3, x), q1), ((q3, y), q4), ((q4, x), q4), ((q4, y), q4)\}$

q0 é o estado inicial.

$F = \{q4\}$

A representação gráfica do autômato é como se ilustra na figura seguinte:

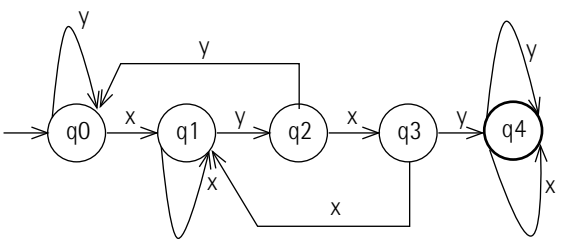

Figura 14 – Autômato Finito para reconhecimento de L= $\{\omega \mid \omega$ apresenta a subcadeia xyxy$\}$

Exercício de aplicação

Identifique qual é a linguagem reconhecida pelo autômato finito ilustrado a seguir. Apresente também a representação algébrica do autômato.

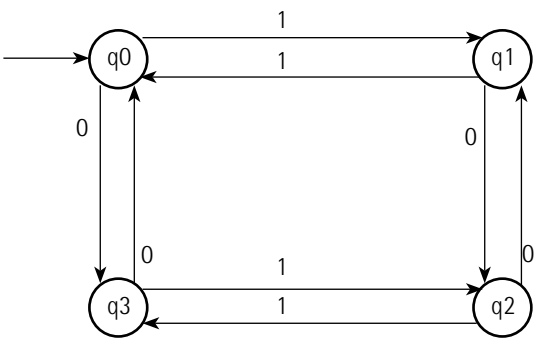

Figura 15 – Autômato Finito

Cumpre observar que a **configuração** de um autômato finito é definida pelo seu estado corrente e pela parte da cadeia de entrada ainda não analisada. Assim, a **configuração inicial** de um autômato

finito é aquela em que o estado corrente é o estado inicial q_0, e o cursor de leitura encontra-se posicionado sobre o símbolo mais à esquerda da cadeia de entrada. Uma **configuração final** é aquela em que o cursor aponta para a posição imediatamente além do último símbolo da cadeia, e o estado corrente q_f pertence ao conjunto de estados finais, ou seja, $q_f \in F$.

A operação de movimentação de um autômato finito, de uma dada configuração para a configuração seguinte, por meio da aplicação da função de transição g, é denotada por $\vdash$.

Exemplo: Considere o autômato M representado em sua forma algébrica como:

$$M = (Q, \Sigma, g, q0, F)$$

Onde $Q = \{q0, q1, q2, q3, q4, qf\}$.

$\Sigma = \{a, b\}$

q0 é o estado inicial e $F = \{qf\}$

Tem-se que:

g(q0, a) = q1	g(q0,b) = q3	g(q1,a) = q1
g(q1, b) = q2	g(q2,a) = q3	g(q2, b) = q4
g(q3, a) = qf	g(q3, b) = q4	g(q4, b) = q3

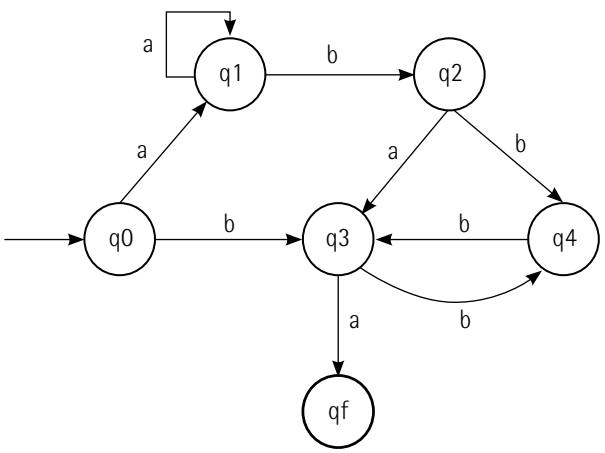

Figura 16 – Exemplo de autômato finito

Seja ω = aabbbbba

A configuração inicial do autômato é (q0, aabbbbba).

A configuração final do autômato é (qf,ε).

Tem-se que a movimentação do autômato, ao longo do processamento do reconhecimento da cadeia aabbbbba, é representada por:

$(q0, aabbbbba) \vdash (q1, abbbbba) \vdash (q1, bbbbba) \vdash (q2, bbbba) \vdash (q4,bbba) \vdash (q3,bba) \vdash (q4,ba) \vdash (q3,a) \vdash (qf, \varepsilon)$

3.5 Autômatos Finitos não determinísticos

Os autômatos finitos não determinísticos permitem que o par (estado corrente, símbolo corrente) apresente diversos "próximos estados".

Transições em vazio podem figurar nos autômatos finitos não determinísticos.

Um autômato finito não determinístico (AFN) sem transições em vazio,

$M = (Q, \Sigma, g, q0, F)$, onde:

- Q é um conjunto finito de estados;
- Σ é um alfabeto (finito e não vazio) de entrada;
- g é uma **relação** de transição, $g: Q \times \Sigma \rightarrow 2^Q$;
- $q0$ é o estado inicial, $q0 \in Q$;
- F é um conjunto de estados finais, $F \subseteq Q$.

Observa-se que a **relação de transição** pode associar a cada par (estado corrente, símbolo corrente) um conjunto de estados do autômato.

Exemplo: Considere a linguagem $L = \{\omega \mid \omega = (xy \mid xyx)^*\}$ definida sobre o alfabeto $\Sigma = \{x, y\}$.

O autômato finito não determinístico que representa a linguagem L é representado graficamente conforme a figura a seguir.

A sua representação algébrica é:

$$M = (Q, \Sigma, g, q0, F)$$

onde:

$Q = \{q0, q1, q2\}$ $\Sigma = \{x, y\}$ qo é o estado inicial $F = \{q2\}$

A relação de transição g é dada por:

$g(q0, x) = \{q1\}$ $g(q1,y) = \{q2,q0\}$ $g(q2,x)= q0$

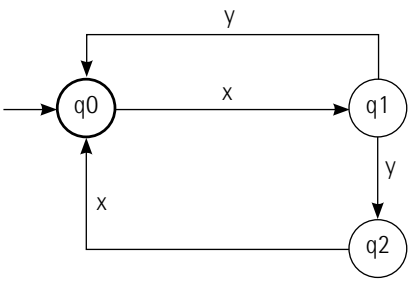

Figura 17 - Autômato Finito Determinístico reconhecedor de L = { ω | ω = (xy | xyx)*}

Por meio desse exemplo, o leitor poderá perceber que muitas vezes o projeto de um autômato finito não determinístico é muito mais intuitivo que o projeto de um autômato finito determinístico.

A representação gráfica do autômato finito determinístico que reconhece a linguagem L = {ω | ω = (xy | xyx)*} é apresentada na figura seguinte.

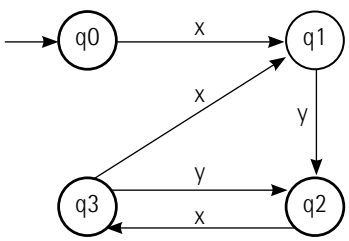

Figura 18 - Autômato Finito Determinístico reconhecedor de L = {ω | ω = (xy | xyx)*}

Autômatos finitos não determinísticos que apresentam **transições em vazio** são aqueles que admitem transições de um estado para outro com ε, além das transições normais, que utilizam os símbolos do alfabeto de entrada. Transições em vazio podem ser executadas sem que seja necessário consultar o símbolo corrente da fita de entrada e, consequentemente, não causam o deslocamento do cursor de leitura.

Um autômato finito não determinístico com transições em vazio apresenta a **relação** de transição g, definida como:

$$g: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q.$$

A figura a seguir apresenta o autômato finito com transições em vazio para a mesma linguagem L anteriormente especificada.

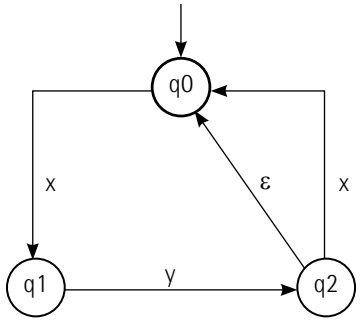

Figura 19 - Exemplo de autômato finito com transição em vazio

3.6 Obtenção de Autômatos Finitos a partir da Gramática Regular

Seja G = (V, Σ, P, S) uma Gramática Regular linear à direita. Pode-se obter o autômato finito determinístico, como se segue:

- Para cada símbolo não terminal da gramática, há um correspondente estado para o autômato, tal que:

Q = V e se A ∈ V e A → ε, então A ∈ F

- O símbolo inicial S da gramática G corresponde ao estado inicial do autômato M.

- O alfabeto de entrada Σ de M é o mesmo alfabeto Σ de G.

- A função de transição g do autômato é construída a partir do mapeamento das produções da gramática em transições, conforme a tabela a seguir:

Tabela 2

Produção	Transição
S → aA	g(S, a) = A, S estado inicial
A → bB	g(A, b) = B
A → ε	A é estado final

Exemplo: Considere a seguinte gramática regular:

G = (V, Σ, P, S), onde:

V = {A, B, C, D}

Σ = {a, b}

A é o símbolo inicial da gramática

$P = \{A \rightarrow bA \mid a\ B$

$B \rightarrow a\ C \mid bA$

$C \rightarrow a\ D \mid b\ A$

$D \rightarrow bA \mid aD \mid \varepsilon\}$

Para obter o autômato correspondente, tem-se que:

- A é o estado inicial
- $Q = \{A, B, C, D\}$
- $\Sigma = \{a, b\}$
- $F = \{D\}$

A função g pode ser obtida a partir das produções da gramática G.

Considere-se a seguinte tabela.

Tabela 3

Produção	Transição
$A \rightarrow bA$	g(A, b) = A
$A \rightarrow aB$	g(A, a) = B
$B \rightarrow aC$	g(B, a) = C
$B \rightarrow bA$	g(B, b) = A
$C \rightarrow aD$	g(C, a) = D
$C \rightarrow bA$	g(C, b) = A
$D \rightarrow bA$	g(D, b) = A
$D \rightarrow aD$	g(D, a) = D
$D \rightarrow \varepsilon$	D é estado final

$g = \{(g(A,b),A), (g(A,a),B), (g(B,a),C), (g(B,b),A), (g(C,a),D), (g(C,b),A), (g(D,b),A), (g(D,a),D)\}$

O autômato é apresentado na figura a seguir:

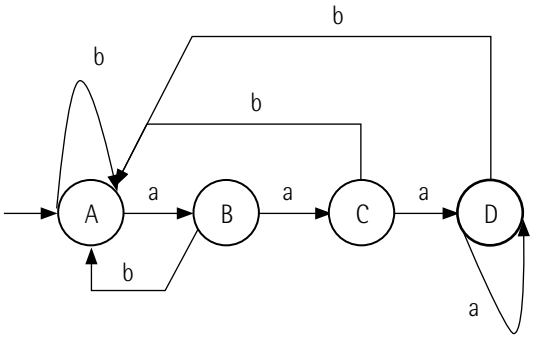

Figura 20 – Exemplo de autômato finito

3.7 Obtenção da Gramática Regular a partir de autômatos finitos

Seja o autômato finito determinístico M = (Q, Σ, g, q0, F). Pode-se obter uma gramática regular à direita G = (V, Σ, P, Q_0), tal que L(M) = L(G).

- A cada estado $q_i \in Q$ cria-se um símbolo não terminal $Q_i \in V$ correspondente. Em particular, ao estado inicial q_0 corresponderá o símbolo inicial Q_0 da gramática.
- O alfabeto de entrada Σ de M é o mesmo alfabeto Σ de G.
- O conjunto das produções P é obtido segundo a tabela a a seguir.

Tabela 4

Transição	Produção
-	$Q_f \rightarrow \varepsilon$
$g(q_i, a) = q_k$	$Q_i \rightarrow aQ_k$

Exemplo: Considere-se o autômato representado na figura:

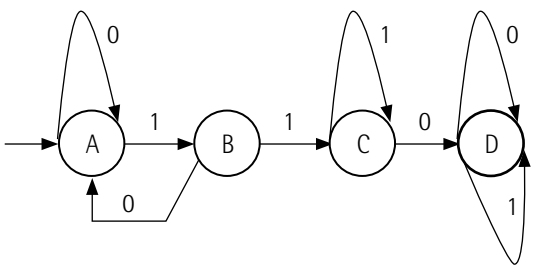

Figura 21 – Autômato finito reconhecedor da linguagem L(M) = {ω | ω= (0|1)*11⁺0 (0|1)*}

Esse autômato reconhece a linguagem L(M) = {ω | ω= (0|1)*11⁺0 (0|1)*}.

É possível obter a gramática que gera a linguagem reconhecida pelo autômato. Tem-se que:

G = (V, Σ, P, A), onde:

V = {A, B, C, D} e Σ= {0,1}

O conjunto das produções P é obtido segundo a tabela que segue.

A tabela de correspondência entre as transições e regras de produção é:

Tabela 5

Transição	Produção
g(A, 0) = A	A → 0 A
g(A,1) = B	A → 1 B
g(B, 0) = A	B → 0 A
g(B, 1) = C	B → 1C
g(C, 0) = D	C → 0 D
g(C, 1) = C	C → 1C
g(D, 0) = D	D → 0D
g(D, 1) = D	D → 1D
	D → ε

P = {A → 0A | 1 B ; B → 1C | 0A; C → 1C | 0D; D → 0D | 1D | ε }

Note a correspondência entre os estados do autômato e os símbolos não terminais da gramática. Em particular, A é o símbolo inicial da gramática.

3.8 Equivalência entre autômatos finitos não determinísticos e determinísticos

Dois autômatos finitos M1 e M2 (determinísticos ou não determinísticos) são equivalentes se, e somente se, L(M1) e L(M2).

A classe dos autômatos finitos determinísticos é equivalente à classe dos autômatos finitos não determinísticos.

Para melhor enunciar as etapas do procedimento que obtém um autômato finito determinístico a partir de um autômato finito não determinístico, a definição de Fechamento - ε é apresentada a seguir.

Fechamento-ε de um estado q_i $E(q_i)$ é o conjunto de estados que podem ser atingidos a partir do estado q_i pela aplicação exclusiva de transições em vazio.

O cálculo de $E(q_i)$ é efetuado segundo o seguinte algoritmo:

Inicialmente, $E(q_i) = \{q_i\}$.

Enquanto houver uma transição tal que $g(p,\varepsilon) = r$, com $p \in E(q_i)$ e $r \notin E(q_i)$, faça: $E(q_i) \leftarrow E(q_i) \cup \{r\}$.

Exemplo: Considere o autômato apresentado na figura a seguir:

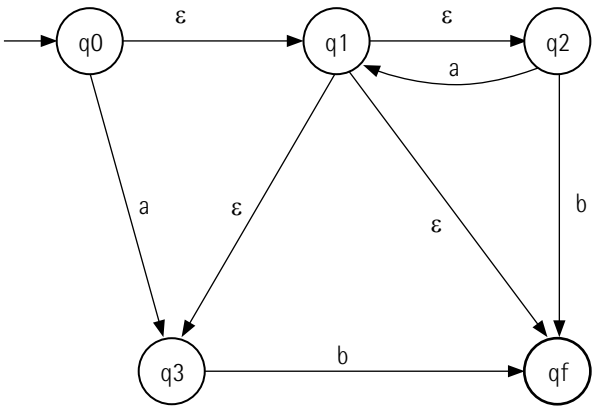

Figura 22 – Autômato finito

Tem-se que $E(q0) = \{q0,q1,q2,q3, qf\}$

$E(q1) = \{q1,q2, q3, qf\}$

$E(q2) = \{q2\}$

$E(q3) = \{q3\}$

$E(qf) = \{qf\}$

Sejam $M_1 = (Q_1, \Sigma, g_1, q_{01}, F_1)$ um autômato finito não determinístico qualquer e $M_2 = (Q_2, \Sigma, g_2, q_02, F_2)$ um autômato finito determinístico construído a partir de M_1. M_2 é equivalente a M_1, e é tal que:

- $Q_2 = 2^{Q1}$. Cada estado do autômato finito determinístico corresponde a um subconjunto do conjunto de estados do autômato finito não determinístico. Observe que somente alguns serão passíveis de serem acessados pelo estado inicial q_{02} e, portanto, os demais estados são irrelevantes para a operação de M_2.

- O conjunto $F_2 \subseteq Q_2$ é o conjunto de estados finais. Cada estado final em F_2 corresponde a um subconjunto de Q_1, que contém um estado final.

- O estado incial $q_{02} = E(q_{01})$.

- Para cada subconjunto $Q \subseteq Q_1$ e cada símbolo $a \in \Sigma$, define-se:

$$g_2(Q, a) = \cup \{E(p) \mid p \in Q_1 \text{ e } g_1(q, a) = p \text{ para algum } q \in Q\}$$

Exemplo: Seja a linguagem $L = \{\omega \mid \omega = a\ (\ b|a)^*\ bc\}$ definida sobre o alfabeto $\Sigma = \{a,b,c\}$. O autômato finito não determinístico M_1 reconhecedor dessa linguagem é apresentado na figura seguinte:

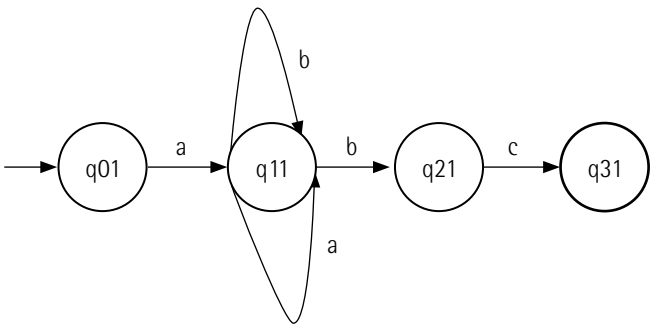

Figura 23 – Autômato finito não determinístico $L = \{\omega \mid \omega = a\ (b|a)^*\ bc\}$

A representação algébrica de M_1 é dada por:

$Q = \{q01, q11, q21, q31\}$

$F = \{q31\}$

$\Sigma = \{a, b, c\}$

O estado inicial é q01 e a função estendida g_1 é dada por:

$g_1((q01,a)) = \{q11\}$

$g_1((q11,a)) = \{q11\}$

$g_1((q11,b)) = \{q11, q21\}$

$g_1((q21,c)) = \{q31\}$

Tem-se que:

$q02 = E(q01) = q01$

$g_2(\{q01\},a) = \{q11\}$

$g_2(\{q11\},b) = \{q11, q21\}$

$g_2(\{q11,q21\},a) = \{q11\}$

$g_2(\{q11,q21\},b) = \{q11,q21\}$

$g_2(\{q11,q21\},c) = \{q31\}$

Os estados do autômato M_2 podem ser nomeados como:

q_{02}, $q_{12} = \{q11\}$, $q_{22} = \{q11, q21\}$ e $q_{32} = \{q_{31}\}$.

A representação gráfica de M_2 é apresentada a seguir.

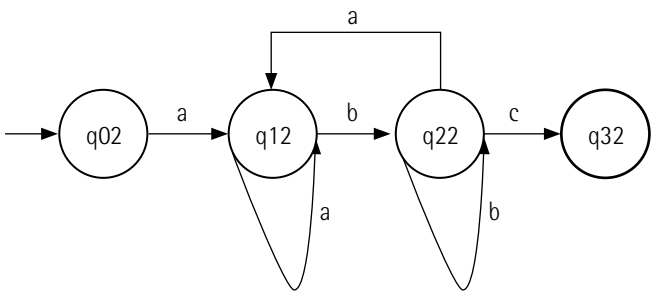

Figura 24 – Autômato finito determinístico L = {ω | ω = a (b|a)* bc}

Lembrete

Para uma linguagem L Regular, denotada por uma expressão regular, existe uma Gramática Regular G geradora, bem como um Autômato Finito Mínimo determinístico L(M) e L(M) = L(G).

4 LINGUAGENS REGULARES — PARTE 2

4.1 O lema do bombeamento para Linguagens Regulares

Muitas questões se apresentam para as Linguagens Regulares, tais como:

- Como verificar se uma Linguagem Regular é infinita, finita ou até mesmo vazia?
- É possível analisar duas Linguagens Regulares e concluir se são iguais ou diferentes?

O enunciado seguinte, conhecido como lema do bombeamento para as Linguagens Regulares, é útil no estudo dessas propriedades.

Se L é uma Linguagem Regular, então existe uma constante n tal que, para qualquer palavra ω de L, onde $|\omega| \geq n$, ω pode ser definida como ω = uvz, onde $|uv| \leq n$ e $|v| \geq 1$ e, para todo $i \geq 0$, uv^iz é a palavra de L.

Saiba mais

Para um estudo mais aprofundado, recomenda-se as seguintes leituras:

LEWIS, H. R.; PAPADIMITRIOU, C. H. *Elements of the Theory of Computation*. 2. ed. Upper Saddle River: Prentice Hall, 1998.

RAMOS, M. V. M.; NETO, J. J.; VEGA, I. S. *Linguagens Formais*. Porto Alegre: Bookman, 2009.

MENEZES, P. B. *Linguagens formais e autômatos*. Porto Alegre: Bookman, 2008.

4.2 Minimização de estados

Os autômatos finitos determinísticos são dispositivos que reconhecem linguagens regulares. Os algoritmos associados aos mesmos apresentam desempenho em consumo de espaço de memória dependente do número de estados. Nesse sentido, é importante que o autômato apresente o menor número de estados possíveis. Existe teorema que garante a minimização de estados de um autômato finito M qualquer. Sejam considerados os teoremas enunciados a seguir.

Teorema: Seja M = (Q, Σ, g, q0, F) um autômato finito. Então, existe um autômato finito M' = (Q', Σ, g', q0, F') que possui o menor número possível de estados, tal que L(M) = L(M').

Teorema: Unicidade do Autômato Mínimo: o autômato finito mínimo determinístico de uma linguagem é único, a menos de ismorfismo.

Para se aplicar o algoritmo de minimização, o autômato finito:

a) deve ser determinístico;

b) não pode apresentar estados inacessíveis;

c) sua função de transição g deve ser total, ou seja, a partir de qualquer estado são previstas transições para todos os símbolos do alfabeto.

Caso o autômato não satisfaça a essas condições, faz-se necessário gerar um autômato equivalente. Para tanto, deve-se:

a) gerar um autômato finito determinístico a partir de um autômato finito não determinístico;

b) eliminar os estados inacessíveis e suas correspondentes transições;

c) transformar a função de transição g em total, inserindo um estado auxiliar q_{aux} e incluindo as transições não previstas para esse estado.

Eliminação dos estados inacessíveis

Estados inacessíveis são aqueles para os quais não existe no autômato qualquer caminho formado por transições válidas que permita atingi-los a partir do estado inicial do autômato.

O conjunto de estados acessíveis pode ser definido como o fechamento de $\{q0\}$, segundo a relação:

$$\{(p, q) \mid g(p, a) = q \text{ para } a \in \Sigma\}$$

Em outras palavras, o conjunto de estados **acessíveis** pode ser obtido a partir deste algoritmo:

- Inicializar o conjunto EA, como EA = $\{q0\}$
- Para **todo estado** p, p $\in$ EA, e para **todo** símbolo a $\in \Sigma$, tal que g(p,a) = q $\notin$ EA, inserir q em EA.

Exemplo: Considere o autômato M = (Q, Σ, g, q0, F), onde:

Q = $\{q1, q2, q3, q4, q5, q6, q7, q8\}$ (conjunto de estados);

F = $\{q1, q3, q7\}$ (conjunto de estados finais);

Σ = $\{x, y\}$ (alfabeto de entrada);

q1 é o estado inicial.

A função programa g é dada por:

g = $\{$((q1,x), q2), $\{$((q1,y), q4), $\{$((q2,x), q5), $\{$((q2,y), q3), $\{$((q3,x), q2), ((q3,y), q6), ((q4,x), q1), ((q4,y), q5), ((q5,x), q5), ((q5,y), q5), ((q6,x), q3), ((q6,y), q5, ((q7, x), q6), ((q7,y),q8), ((q8,x), q7), (q8, y), q3)$\}$.

Pede-se obter o conjunto **EA**, ou seja, o conjunto de estados acessíveis do autômato.

O algoritmo apresentado inicializa o conjunto EA com o estado inicial. Tem-se, portanto, que:

$$EA = \{q1\}$$

A próxima etapa, concernente ao levantamento dos estados acessíveis, consiste em identificar para **todo estado** p, p $\in$ EA, e para **todo** símbolo a $\in \Sigma$, o estado q, tal que q = g(p,a).

Assim, para prosseguir o exemplo, representa-se a função g, de forma tabular para que mais facilmente sejam identificados os pares de estados possíveis (p,q).

Tabela 6

	x	y
q1	q2	q4
q2	q5	q3
q3	q2	q6
q4	q1	q5
q5	q5	q5
q6	q3	q5
q7	q6	q8
q8	q7	q3

O único estado que pertence a EA é q1. Como o alfabeto Σ é binário, há que se considerar apenas os pares (q1,x) e (q1, y). Tem-se que g(q1,x) = q2 e g(q1, y) = q4. Os estados q2 e q4 passam a pertencer ao conjunto EA. Tem-se que:

$$EA = \{q1, q2, q4\}$$

As próximas iterações dizem respeito à consideração dos pares: (q2,x), (q2, y), (q4,x), (q4, y). Tem-se que:

$$g(q2, x) = q5,\ g(q2, y) = q3,\ g(q4,x) = \mathbf{q1},\ g(q4, y) = \mathbf{q5}$$

Dessa forma, os estados q5, q3 devem ser acrescentados aos elementos de EA. Tem-se que:

$$EA = \{q1, q2, q4, q3, q5\}$$

A iteração seguinte consiste em considerar os pares (q3,x), (q3, y) e (q5,x) e (q5, y), recém-inseridos em EA. Tem-se que:

$$g(q3, x) = \mathbf{q2},\ g(q3, y) = q6,\ g(q5,x) = \mathbf{q5},\ g(q5, y) = \mathbf{q5}.$$

Observe que o único estado a inserir é q6, visto que q2 e q5 pertencem a EA. Tem-se que:

$$EA = \{q1, q2, q4, q3, q5, q6\}$$

Analogamente, sejam considerados os pares (q6,x) e (q6, y).

Tem-se que: g(q6, x) = **q3** e g(q6, y) = **q5**

Constata-se que não há mais estados a inserir em EA. Portanto, o conjunto de estados acessíveis coincide com o EA obtido na última iteração do algoritmo. Naturalmente, o conjunto dos estados inacessíveis é:

$$EA' = Q - EA = \{q7, q8\}.$$

Como se pode constatar em autômato finito, podem ocorrer estados inacessíveis, que devem ser eliminados para minimizar os estados.

A eliminação dos estados não alcançáveis resulta no seguinte autômato:

M = (Q, Σ, g, q0, F), onde:

Q = {q1, q2, q3, q4, q5} (conjunto de estados).

F = {q1, q3} (conjunto de estados finais);

Σ = {x, y} (alfabeto de entrada);

q1 é o estado inicial;

A função programa g é dada por:

g = {((q1, x), q2), {((q1,y), q4), {((q2,x), q5), {((q2,y), q3), {((q3,x), q2), ((q3,y), q6), ((q4,x), q1), ((q4,y), q5), ((q5,x), q5), ((q5,y), q5), ((q6,x), q3), ((q6,y), q5)}.

Minimização de estados

a) Para se prosseguir com o algoritmo de minimização, constrói-se uma tabela relacionando os estados distintos, em que cada par de estados ocorre somente uma vez. Considere a tabela triangular a seguir:

q2					
q3					
q4					
q5					
q6					
	q1	q2	q3	q4	q5

Figura 25

b) Em seguida, marcam-se estados trivialmente não equivalentes. Um par trivialmente não equivalente é aquele em que um dos elementos é estado final e o outro não. Assim sendo, são

trivialmente não equivalentes: (q1, q2), (q1, q4), (q1, q5), (q1, q6), (q2, q3), (q3, q4), (q3, q5) e, finalmente, (q3, q6).

q2	X				
q3		x			
q4	X		x		
q5	X		x		
q6	X		x		
	q1	q2	q3	q4	q5

Figura 26

c) Finalmente, marcam-se os estados não equivalentes. Para tanto, são analisados os demais pares não contemplados no item anterior.

c.1: análise do par (q1, q3).

g(q1, x) = q2

g(q1, y) = q4

g(q3, x) = q2

g(q3, y) = q6.

Nada se pode afirmar a respeito do par (q1, q3), pois não se sabe se q4 e q6 são equivalentes ou não. Assim, a análise de (q1, q3) fica pendente até que se possa constatar a equivalência ou não do par (q4, q6).

c.2: análise do par (q2, q4).

g(q2, x) = q5

g(q4, x) = q1

g(q2, y) = q3

g(q4, y) = q5

Os pares q1 e q5, bem como os pares q3 e q5, são trivialmente não equivalentes, portanto o par (q2, q4) deve ser marcado como não equivalente. De fato, se o autômato estiver no estado q2 e ocorrer na cadeia de entrada o símbolo x, o processamento alcança o estado q5. Se, por outro lado, o autômato estiver no estado q4 e ocorrer o símbolo x, o autômato transita para o estado q1. Como q1 e q5 são trivialmente não equivalentes, diz-se que q2 e q4 também são não equivalentes e devem ser marcados.

q2	X				
q3					
q4	X	x	x		
q5	X		x		
q6	X		x		
	q1	q2	q3	q4	q5

Figura 27

c.3: análise do par (q2, q5).

g(q2, x) = q5

g(q2, y) = q3

g(q5, x) = q5

g(q5, y) = q5

O par (q3, q5) está marcado, portanto o par (q2, q5) deve ser marcado. De fato, g(q2, y) = q3 e g(q5, y) = q5. Como q3 e q5 são trivialmente não equivalentes, então q2 e q5 são não equivalentes.

c.4: análise do par (q2, q6).

g(q2,x) = q5

g(q2, y) = q3

g(q6, x) = q3.

g(q6,y) = q5

Como q3 e q5 são trivialmente não equivalentes, tem-se que q2 e q6 são não equivalentes.

c.5: análise do par (q4,q5).

g(q4, x) = q1

g(q4, y) = q5

g(q5, x) = q5

g(q5, y) = q5

O par (q1, q5) está marcado, pois é trivialmente não equivalente, portanto o par (q4, q5) deve ser marcado como não equivalente.

c.6: análise do par (q4, q6).

g(q4, x) = q1

g(q4, y) = q5

g(q6, x) = q3

g(q6, y) = q5

Nada se pode afirmar sobre o par (q4, q6), pois o par (q1, q3) não está marcado.

c.7: análise do par (q5, q6).

G(q5, x) = q5

G(q5, y) = q5

G(q6, x) = q3

G(q6, y) = q5

O par (q3, q5) está marcado, pois são trivialmente não equivalentes, portanto o par (q5, q6) deve ser marcado como estados não equivalentes.

Q2	x				
Q3		x			
Q4	x	x	x		
Q5	x	x	x	X	
Q6	x	x	x		x
	Q1	Q2	Q3	Q4	Q5

Figura 28

a) Ao final da análise dos diferentes pares ficaram não marcados os pares (q4, q6) e (q1, q3). Diz-se que são equivalentes. Assim sendo, a próxima etapa consiste na unificação dos estados equivalentes.

Tabela 7

g	x	y
Q13	Q2	Q46
Q2	Q5	Q13
Q46	Q13	Q5
Q5	Q5	Q5

b) Ao final, eliminam-se os estados inúteis. Observa-se que, a partir do estado q5, não é possível alcançar o estado final. Assim sendo, q5 é inútil. Tem-se, portanto, que os estados do autômato mínimo são q13, q2 e q46, com a função de transição dada por:

Tabela 8

g	x	y
Q13	Q2	Q46
Q2	-	Q13
Q46	Q13	-

Como os estados q1 e q3 eram estados finais, no autômato mínimo, o estado q13, resultante da unificação dos estados q1 e q3, também é estado final.

4.3 Problemas decidíveis concernentes às linguagens regulares

Um problema de decisão é saber se existe algum algoritmo para decidir se proposições individuais, dentro de uma grande classe de proposições, são verdadeiras (GERSTING, 1999).

Diz-se que um problema é decidível com solução positiva se tem solução mediante a existência de um algoritmo. Diz-se que ele é decidível com solução negativa, quando se demonstra que não existe algoritmo para o problema.

Os aspectos algorítmicos relacionados às linguagens regulares e suas representações podem ser resumidas no seguinte teorema:

Teorema:

a) Há um algoritmo exponencial para o seguinte problema: construir um autômato finito determinístico equivalente a um dado autômato finito não determinístico.

b) Há um algoritmo polinomial para o seguinte problema: dada uma expressão regular, construir um autômato finito não determinístico equivalente.

c) Há um algoritmo exponencial para o seguinte problema: dado autômato finito não determinístico, criar uma expressão regular equivalente.

d) Há um algoritmo polinomial para o seguinte problema: dado um autômato finito determinístico, construir um autômato finito determinístico equivalente, com o menor número possível de estados.

e) Há um algoritmo polinomial para o seguinte problema: dados dois autômatos determinísticos quaisquer decidir se ambos são equivalentes entre si.

f) Há um algoritmo exponencial para o seguinte problema: dados dois autômatos finitos não determinísticos decidir se ambos são equivalentes entre si. O mesmo se pode afirmar para a existência de um algoritmo que decide sobre a equivalência de duas expressões regulares.

4.4 Máquinas de Mealy e Moore

A definição de autômato finito pode ser estendida, de forma que, além de a informação fornecida por esses dispositivos ser aquela de aceitar ou não uma cadeia de entrada, o dispositivo gera cadeias de saída. De fato, as máquinas de estados finitos que geram palavras podem ser duas, a saber: a máquina de Mealy e a máquina de Moore.

A **máquina de Mealy** é um autômato finito modificado de forma a gerar uma palavra de saída para cada transição do autômato.

A máquina de Mealy pode ser representada por uma sêxtupla:

$$M = (Q, \Sigma, g, q0, F, \Delta), \text{ onde:}$$

Q: conjunto finito de estados do autômato;

Σ: alfabeto dos símbolos de entrada.

g: função programa ou função de transição:

$g: Q \times \Sigma \rightarrow Q \times \Delta^*$

$q0$ é o estado inicial do autômato e $q0 \in Q$.

F: conjunto de estados finais, tal que $F \subseteq Q$;

Δ: alfabeto de símbolos de saída.

A máquina de Moore gera uma cadeia de saída para cada estado da máquina.

Uma máquina de Moore é um Autômato Finito Determinístico com saídas associadas aos estados. É representada por uma sétupla.

$M = (Q, \Sigma, g, q0, F, \Delta, h)$, onde:

Q: conjunto finito de estados do autômato.

Σ: alfabeto dos símbolos de entrada.

g: função programa ou função de transição:

$$g: Q \times \Sigma \rightarrow Q$$

q0 é o estado inicial do autômato e $q0 \in Q$.

F: conjunto de estados finais tal que $F \subseteq Q$.

Δ: alfabeto de símbolos de saída.

h: função de **saída:**

$$h: Q \rightarrow \Delta^*$$

Exemplo: Seja a Máquina de Moore $M = (Q, \Sigma, g, q0, F, \Delta, h)$, onde:

$Q = \{q0, q1, q2, q3, q4\}$

$\Sigma = \Delta = \{a,b\}$

$F = \{q4\}$.

A função de transição g e a função de saída h são representadas conforme a tabela a seguir:

Tabela 9

Estado atual	Próximo estado		Saída
	Entrada atual		
	a	b	
q0	Q0	Q3	a
q1	Q0	Q4	b
q2	Q2	Q1	b
q3	Q0	Q4	a
q4	Q1	Q2	b

Considere que a cadeia de entrada abbabbaba seja submetida à entrada. Tem-se que, ao longo do processamento, a sequência de símbolos de saída obtida é: aaabbbbbba.

Os compiladores empregam as Máquinas de Moore na Análise Léxica. De fato, o analisador léxico lê uma sequência de caracteres e classifica-os em *tokens*. Os *tokens* identificam os diversos tipos de unidades léxicas presentes em uma Linguagem de Programação, por exemplo: identificadores, operadores, constantes, delimitadores etc. Assim, sempre que um lexema é identificado e, portanto, atinge um estado final do autômato, o *token* correspondente é gerado como saída, representando assim a sua classificação. Os estados não finais não geram saídas.

Resumo

Vimos que o alfabeto é um conjunto finito de símbolos e a linguagem formal é um conjunto de cadeias definidas sobre um alfabeto.

A gramática é um dispositivo gerador da linguagem. Define-se formalmente a gramática como uma quádrupla G = (V, Σ, P, S), onde V é um conjunto de símbolos não terminais. T é um conjunto de símbolos terminais e coincide com o alfabeto da linguagem e S é o símbolo inicial da gramática. P é o conjunto de regras ou produções.

A aplicação de uma regra de produção $\alpha \rightarrow \beta$ é denominada derivação.

Noam Chomsky classificou a linguagem em quatro tipos, a saber:

Linguagens Regulares, Linguagens Livres de Contexto, Linguagens Recursivas e Linguagens Recursivamente Enumeráveis.

As Linguagens Regulares são geradas pelas Gramáticas Regulares. Estas podem ser de dois tipos: Linear à Direita e Linear à Esquerda.

Uma gramática é linear à direita, se todas as produções são da forma:

$A \rightarrow \omega B$ ou $A \rightarrow \omega.$, onde $A, B \in V$ e $\omega \in \Sigma^*$

Uma gramática é linear à esquerda, se todas as produções são da forma:

$A \rightarrow B\omega$ ou $A \rightarrow \omega$. $A, B \in V$ e $\omega \in \Sigma^*$

São enunciados dois teoremas que relacionam Linguagens Regulares e Gramáticas Regulares: se L é uma linguagem gerada por uma Gramática Regular, então L é uma Linguagem Regular. Se L é uma Linguagem Regular, então existe G, Gramática Regular que gera L.

Autômatos Finitos são dispositivos reconhecedores das Linguagens Regulares. Podem ser determinísticos ou não determinísticos, podendo estes últimos apresentar-se com transições em vazio.

A partir de uma Gramática Regular, é possível obter um autômato finito. Também é possível obter o autômato finito que reconhece uma linguagem, a partir de sua gramática.

É provado que a classe dos autômatos finitos determinísticos é equivalente à dos autômatos finitos não determinísticos. Em outras palavras, se por um lado o projeto de autômatos finitos não determinísticos é mais intuitivo, por outro, o não determinismo não aumenta o poder de reconhecimento de linguagens regulares.

Existe um algoritmo para construir o autômato finito determinístico mínimo, ou seja, o autômato com menor número de estados. O autômato finito determinístico de uma linguagem é único, a menos de isomorfismo.

Há que se salientar que o lema conhecido como lema do bombeamento permite estudar as propriedades de uma linguagem regular.

As linguagens regulares apresentam, além do formalismo gerador e do formalismo reconhecedor, o formalismo denotacional, conhecido como expressões regulares.

A definição de autômato finito pode ser estendida, de forma que, além da informação fornecida por esses dispositivos ser aquela de aceitar ou não uma cadeia de entrada, o dispositivo gera cadeias de saída.

A máquina de Mealy é um autômato finito modificado de forma a gerar uma palavra de saída para cada transição do autômato.

A máquina de Moore gera uma cadeia de saída para cada estado da máquina.