**Alexandre Maciel**

# Aplicação de autômatos finitos nebulosos no reconhecimento aproximado de cadeias

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

São Paulo
2006

**Alexandre Maciel**

# Aplicação de autômatos finitos nebulosos no reconhecimento aproximado de cadeias

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia.

Área de concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Marco Túlio Carvalho de Andrade

São Paulo
2006

**Ficha Catalográfica**

Maciel, Alexandre

Aplicação de autômatos finitos nebulosos no reconhecimento aproximado de cadeias / Alexandre Maciel - São Paulo, 2006.

63 p.

Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Fuzzy. 2. Reconhecimento de Texto. 3. Autômatos Finitos. I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais. II.t.

Dedico esse trabalho a todos aqueles que ressentiram-se com minha ausência e todos aqueles que suportaram minha presença nesse tempo difícil.

# Agradecimentos

Ao meu orientador, Profº Dr. Marco Túlio Carvalho de Andrade, pela oportunidade oferecida.

Aos meus pais e à minha família, os quais nunca me faltaram com o apoio, nem nas horas mais difíceis.

À Dalva, foram dias nublados, não foram?

Ao meu amigo e colega Humberto, como eu poderia agradecer tudo nesse espaço?

Por fim, a todos os colegas, professores e funcionários da USP sem os quais nada disso teria sido possível.

# Resumo

MACIEL, A. **Aplicação de autômatos finitos nebulosos no reconhecimento aproximado de cadeias**. 2006. 62 p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2006.

O reconhecimento aproximado de cadeias de texto é um problema recorrente em diversas aplicações onde o computador é utilizado como meio de processamento de uma massa de dados sujeita a imprecisões, erros e distorções. Existem inúmeras metodologias, técnicas e métricas criadas e empregadas na resolução deste tipo de problema, mas a maioria delas é inflexível em pelo menos um dos seguintes pontos: arquitetura, métrica utilizada para aferir o erro encontrado ou especificidade na aplicação.

Esse trabalho propõe e analisa a utilização dos Autômatos Finitos Nebulosos para a resolução desse tipo de problema. A teoria nebulosa oferece uma base teórica sólida para o tratamento de informações inexatas ou sujeita a erros, enquanto o modelo matemático dos autômatos finitos é uma ferramenta consolidada para o problema de reconhecimento de cadeias de texto. Um modelo híbrido não só oferece uma solução flexível para a resolução do problema proposto, como serve de base para a resolução de inúmeros outros problemas que dependem do tratamento de informações imprecisas.

Palavras-chave: Fuzzy. Reconhecimento de Texto. Autômato Finito.

# Abstract

MACIEL, A. **Aplicação de autômatos finitos nebulosos no reconhecimento aproximado de cadeias**. 2006. 62 p. Dissertation (Mastering) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2006.

The approximate string matching problem is recurring in many applications where computer is used to process imprecise, fuzzy or spurious data. An uncountable number of methods, techniques and metrics to solve this class of problem are available, but many of them are inflexible at least in one of following: architecture, metric or application specifics.

This work proposes and analyzes the use of Fuzzy Finite State Automata to solve this class of problems. The fuzzy theory grants a solid base to handle imprecise or fuzzy information; the finite state automata is a classic tool in string matching problems. A hybrid model offers a flexible solution for this class of problem and can be a base for other problems related with imprecise data processing.

Keyworks: Fuzzy. String Matching. Finite Automata.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas

**AF** Autômato Finito

**AFs** Autômatos Finitos

**AFD** Autômato Finito Determinístico

**AFN** Autômato Finito Nebuloso

**AFNs** Autômatos Finitos Nebulosos

**AFND** Autômato Finito Não-Determinístico

**AFNDs** Autômatos Finitos Não-Determinísticos

# 1 Introdução

Com a evolução da computação na última década, a quantidade e a capacidade dos computadores cresceu vertiginosamente, atingindo patamares inimagináveis nas décadas de 1970 e 1980. Processadores, unidades de armazenamento e interfaces de comunicação tiveram seus preços drasticamente reduzidos. Atualmente, qualquer indivíduo pode ter acesso a um computador pessoal com capacidade equivalente a um supercomputador da década passada, e uma empresa de médio porte ou instituto de pesquisa pode ter um supercomputador construído a partir de componentes disponíveis comercialmente por algumas centenas de milhares de dólares.

Essa popularização e barateamento fez com que dispositivos computacionais passassem a ser empregados em automóveis, eletrodomésticos, brinquedos e produtos de consumo em massa. Uma quantidade de dados inimaginável há apenas alguns anos existe hoje em unidades de armazenamento diversificadas, porém potencialmente interligáveis e acessíveis. Novos múltiplos para as unidades de dados se fazem necessários: o armazenamento de gigabytes ou terabytes de dados tornou-se comum, enquanto o processamento de petabytes ou exabytes de dados se tornará uma ocorrência trivial em um futuro muito próximo.

Entretanto, a evolução da computação não se trata simplesmente em equipamentos mais baratos ou com maior capacidade.

O emprego do computador em aplicações diferentes daquelas para as quais ele vem sendo desenvolvido e empregado desde a Segunda Guerra Mundial criou novos e instigantes problemas para os cientistas da computação. Algumas aplicações não funcionam bem ou simplesmente não funcionam com as técnicas tradicionalmente empregadas pela ciência da computação. Aplicações como o reconhecimento de padrões, computação em linguagem natural e processamento de dados inexatos são alguns exemplos dessas aplicações.

Esses problemas exigem outras abordagens para serem solucionados. A computação em linguagem natural, por exemplo, exige soluções não-tradicionais como

a *lógica nebulosa* (ZADEH, 1996). Entretanto, é evidente que a ciência da computação ainda está longe de ter respostas satisfatórias para a totalidade dos problemas que lhe são apresentados e mais longe ainda de ter soluções ótimas para todos eles.

## 1.1 Objetivos

Um desses novos problemas, particularmente recorrente na computação, é o reconhecimento aproximado de cadeias de texto. É bem verdade que esse problema não é atual: as primeiras referências rivalizam em antecedência com o próprio computador digital (NAVARRO, 2001).

Por outro lado, também é verdade que esse problema nunca recebeu tanta atenção. O crescimento vertiginoso da quantidade de dados disponível em unidades de armazenamento (seja num banco de dados corporativo, seja na Internet como um todo) tornou impraticável a retificação manual desses dados, garantindo que estes estejam rigorosamente livres de erros. Dessa forma, contando apenas com as abordagens computacionais estritas - incapazes de tratar dados sujeitos a erros - esses dados tem uma utilidade bastante limitada e a informação armazenada neles pode ser perdida de maneira definitiva.

Dessa forma, o objetivo desse trabalho é propor e analisar o uso do Autômato Finito Nebuloso como ferramenta para a resolução de problemas de reconhecimento aproximado de cadeias, discutindo suas vantagens e desvantagens.

## 1.2 Justificativas e Motivação

Embora alguns trabalhos esporádicos de reconhecimento aproximado de cadeias utilizem Autômato Finito Nebuloso, a absoluta maioria deles empregam outras ferramentas para resolver esse tipo de problema.

Estas outras ferramentas, por outro lado, quase sempre possuem algumas ou todas seguintes desvantagens em relação ao emprego do Autômato Finito Nebuloso no reconhecimento aproximado de cadeias: complexidade computacional, arquitetura pouco flexível, falta de expressividade do modelo ou incapacidade de representar o conhecimento do projetista sobre peculiaridades do problema tratado (em outras palavras, a dificuldade ou mesmo a impossibilidade do projetista criar métricas específicas para o problema).

Por um lado, a lógica nebulosa é uma ferramenta consagrada e poderosa no

tratamento de problemas de manipulação e processamento de dados quantitativos e/ou sujeitos a erros inerentes à aplicação (ZADEH, 1973). Por outro, o modelo dos autômatos finitos clássicos é o modelo mínimo suficiente para a aplicação do reconhecimento estrito de cadeias. Dessa forma, parece claro que um modelo capaz de combinar a lógica nebulosa com os autômatos finitos é um bom ponto de partida para um modelo eficiente capaz de realizar o reconhecimento aproximado de cadeias.

A partir dessa premissa o trabalho é desenvolvido, estudando as vantagens e desvantagens desta solução. A expectativa é que este trabalho sirva como ponto de partida para trabalhos futuros: algumas sugestões a respeito de possíveis extensões são feitas no final do trabalho.

## 1.3 Estrutura do Texto

O capítulo 2 introduz alguns conceitos fundamentais para a compreensão do trabalho. Uma pequena introdução sobre autômatos finitos e sobre lógica nebulosa é apresentada para que o leitor se situe e se sinta confortável nos capítulos posteriores.

O capítulo 3 define e discorre a respeito do problema do reconhecimento aproximado de cadeias, terminando com alguns exemplos de áreas e aplicações onde ele é empregado com freqüência.

O capítulo 4 define o modelo de Autômato Finito Nebuloso utilizado nesse trabalho e mostra como empregá-lo no reconhecimento aproximado de cadeias. Alguns exemplos ajudam a esclarecer o funcionamento do modelo, que é detalhado nesse capítulo.

O capítulo 5 exibe os resultados obtidos com o modelo proposto; nominalmente: eficiência computacional, expressividade e flexibilidade.

Por fim, o capítulo 6 faz algumas considerações finais sobre os resultados; assim como faz algumas avaliações sobre as conclusões práticas e teóricas do trabalho. Algumas sugestões de trabalhos futuros também são citadas nesse capítulo.

# 2 Conceitos e Metodologia

## 2.1 Teoria Nebulosa

Enquanto a abordagem clássica da computação pressupõe o processamento de dados rigorosamente livres de erros e bem definidos em modelos matemáticos categóricos; os problemas reais com os quais o raciocínio humano tem que lidar exigem a capacidade de manipular dados com uma quantidade de erros relativamente alta em modelos aproximados.

Ainda que a forma clássica trate com extrema eficiência os problemas onde se pode contar com dados e modelos precisos e bem definidos, ela exclui uma grande variedade de problemas onde esses pré-requisitos não são verdadeiros. Dessa forma, torna-se necessária uma maneira diferente de tratar esses problemas, uma abordagem capaz de imitar algumas características do raciocínio humano tais como a tolerância a falhas, o tratamento de dados contendo verdades parciais e a modelagem aproximada do problema.

A *Teoria Nebulosa* é uma das abordagens que podem ser utilizadas no tratamento desse tipo de problema, permitindo que os computadores resolvam com grande eficiência problemas anteriormente impossíveis de serem resolvidos computacionalmente, ou pelo menos, incapazes de serem solucionados computacionalmente em tempo hábil.

### 2.1.1 Conjuntos Nebulosos

A teoria clássica dos conjuntos, sustentáculo da computação convencional, é baseada na suposição da existência de um universo de discurso que abrange todos os elementos relevantes ao contexto. Dentro desse universo de discurso, existem conjuntos que agrupam esses elementos de acordo com critérios definidos de maneira arbitrária, utilizando relações de pertinência e não-pertinência, representadas pelos símbolos $\in$ e $\notin$ respectivamente.

Entretanto, é freqüente a ocorrência de casos onde a teoria clássica dos conjuntos não se mostra adequada para descrever o universo de discurso, notoriamente a descrição de conjuntos qualitativos. Como exemplo, seja o conjunto A = {números reais muito maiores que 10}. Claramente, os elementos '100' e '150' fazem parte do conjunto, enquanto os elementos '8' e '12' não fazem. Porém, como classificar os elementos '20' e '25'?

A *teoria dos conjuntos nebulosos* trata justamente dos casos onde a teoria clássica dos conjuntos não se mostra adequada. Mais do que um conceito completamente novo, os conjuntos nebulosos nada mais são do que uma generalização na qual os conjuntos clássicos são um caso particular: ao invés de definir a relação de um elemento com um conjunto em termos de "pertence" ou "não-pertence", um número real no intervalo entre 0 e 1 é utilizado para qualificar a "afinidade" do elemento com determinado conjunto.

Usando o exemplo anterior, é possível dizer que '20' "pertence" ao conjunto A com um grau 0,5 e que '25' pertence a A com grau 0,6. O valor que denota a relação de um elemento com um conjunto em particular é denominado *grau de pertinência* do elemento em relação ao conjunto; os graus de pertinência 0 e 1 significam na teoria dos conjuntos nebulosos o que "não-pertence" e "pertence", respectivamente, significam na teoria clássica dos conjuntos. Por esse motivo, é possível dizer que os conjuntos clássicos são um caso particular dos conjuntos nebulosos, onde todos os graus de pertinência dos elementos desse conjunto são iguais a 0 ou 1.

Os conjuntos nebulosos são representados da seguinte maneira: $\tilde{F} = \{a_1/p_1, a_2/p_2, ..., a_n/p_n\}$, onde $\tilde{F}$ é o nome do conjunto, $a_1, a_2, ..., a_n$ são os elementos do conjunto e $p_1, p_2, ..., p_n$ são os graus de pertinência associados aos respectivos elementos. A cardinalidade de qualquer conjunto nebuloso é sempre igual à cardinalidade do conjunto universo que o contém (todos os elementos do conjunto universo pertencem ao conjunto nebuloso contido neste, ainda que parte deles possuam um grau de pertinência igual a zero). Para facilitar a notação, por convenção, os elementos cujo grau de pertinência é igual a zero são suprimidos da notação, ficando subentendidos.

Dessa forma, o conjunto do exemplo anterior pode ser representado da seguinte maneira: $\tilde{A} = \{20/0,5; 25/0,6; 100/1; 150/1\}$.

É possível definir a relação entre os elementos de um conjunto nebuloso e seus respectivos graus de pertinência através de uma função que relaciona todos os elementos do conjunto nebuloso a um valor no intervalo entre 0 e 1. Essa

função é denominada *função de pertinência* e é representada da seguinte maneira: $\mu_{\tilde{F}}(a) \rightarrow [0,1]$. A função de pertinência característica pode ser utilizada para representar conjuntos nebulosos, isso é especialmente útil no caso de conjuntos nebulosos com um número grande ou infinito de elementos.

De volta ao exemplo anterior, o conjunto nebuloso $\tilde{A}$ (não necessariamente com os mesmos graus de pertinência das representações anteriores) poderia ser representado pela seguinte função de pertinência característica:

$$\mu_{\tilde{A}}(a) = \begin{cases} 0, \text{ se } a < 10; \\ 1, \text{ se } a > 60; \\ (a-10)/50, \text{ caso contrário} \end{cases}$$

Um estudo mais aprofundado sobre a teoria nebulosa pode ser feito consultando-se Zadeh (1965, 1973, 1996); além de Lee (1990a, 1990b) e Nguyen e Walker (2000).

### 2.1.2 Operações com Conjuntos Nebulosos

Sejam os conjuntos nebulosos $\tilde{A}$ e $\tilde{B}$. O conjunto $\tilde{A}$ representa o conjunto dos números muito maiores do que 10, enquanto o conjunto $\tilde{B}$ representa o conjunto dos números mais ou menos iguais a 30. Uma representação parcial destes conjuntos poderia ser: $\tilde{A} = \{20/0,5; 25/0,6; 30/0,7; 35/0,8; 40/0,9; 45/1\}$ e $\tilde{B} = \{20/0,3; 25/0,7; 30/1; 35/0,7; 40/0,3; 45/0,1\}$.

Considere-se agora a intersecção entre esses dois conjuntos, de maneira que $\tilde{C} = \tilde{A} \cap \tilde{B}$. Em outras palavras, o conjunto $\tilde{C}$ é o conjunto dos números muito maiores do que 10 *e* mais ou menos iguais a 30 (a operação entre conjuntos "intersecção" é equivalente ao "e" da lógica booleana).

Caso $\tilde{A}$ e $\tilde{B}$ fossem conjuntos clássicos, a tabela 2.1 representaria a operação de intersecção, onde 0 significa "falso" e 1 significa "verdadeiro".

| A/B | 1 | 0 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

**Tabela 2.1:** Representação da operação de "intersecção" de conjuntos clássicos

A operação de intersecção entre conjuntos clássicos cria a função $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$. Porém, para que essa operação seja capaz de representar a intersecção entre conjuntos nebulosos, é necessário definir uma função tal que: $\otimes : [0,1] \times [0,1] \rightarrow [0,1]$. Além disso, uma vez que os conjuntos nebulosos são uma generalização dos conjuntos clássicos, essa função tem que apresentar os mesmos resultados de uma operação de intersecção clássica.

O primeiro operador utilizado para fazer a intersecção de conjuntos nebulosos foi $x \otimes y = min(x, y)$, no artigo original de Zadeh (1965). As seguintes propriedades são relevantes e podem ser observadas no operador:

- *Elemento Neutro*, 1 age como elemento neutro, tal que $x \otimes 1 = x$;
- *Comutativa*, ou seja, $x \otimes y = y \otimes x$;
- *Associativa*, ou seja, $x \otimes (y \otimes z) = (x \otimes y) \otimes z$;
- *Monotonicidade*, ou seja, se $v \leq w$ e $x \leq y$, então $v \otimes x \leq w \otimes y$.

Todos os operadores que satisfazem essas propriedades são chamados de *normas triangulares* ou simplesmente de *t-normas*; e podem ser utilizados para fazer a intersecção de conjuntos nebulosos (e como operadores lógicas nebulosos "e"). Na realidade, os operadores de norma triangular precedem a lógica nebulosa: eles surgiram no contexto dos espaços métricos estatísticos (probabilísticos) e foram introduzidos no artigo de Menge (1942).

Embora existam incontáveis operadores que preencham os requisitos para serem considerados normas triangulares - cada um adequado para uma determinada situação - Lee (1990b) destaca os seguintes como os mais relevantes:

- *Intersecção*: $x \wedge y = min(x, y)$;
- *Produto Algébrico*: $x \cdot y = xy$;
- *Produto Restrito*: $x \triangle y = max(0, x + y - 1)$;
- *Produto Drástico*: $x \sqcap y = \begin{cases} x, \text{ se } y = 1; \\ y, \text{ se } x = 1; \\ 0, \text{ se } x, y < 1. \end{cases}$

Uma outra propriedade opcional dos operadores de norma, importante e decisiva na escolha do operador em algumas aplicações, é a *idempotência*. A propriedade da idempotência nada mais faz do que garantir que

$x \otimes x \equiv x \otimes x \otimes x \equiv x \otimes x \otimes x \otimes x \equiv ...$; definindo de outra maneira, o operador de norma deve ser capaz de fazer com que $x \otimes x = x$ seja uma afirmação verdadeira. O único operador de norma existente que apresenta a propriedade da idempotência é a *intersecção* (NGUYEN; WALKER, 2000, p. 85).

Da mesma forma que a intersecção, é necessário definir a união entre dois conjuntos nebulosos. Usando os mesmos conjuntos do exemplo anterior, $\tilde{C} = \tilde{A} \cup \tilde{B}$, de maneira que $\tilde{C}$ é o conjunto dos números muito maiores do que 10 *ou* mais ou menos iguais a 30 (a operação entre conjuntos "união" é equivalente ao "ou" da lógica booleana).

Supondo mais uma vez que $\tilde{A}$ e $\tilde{B}$ são conjuntos clássicos, a tabela 2.2 representaria a operação de união entre os conjuntos, onde 0 significa "falso" e 1 significa "verdadeiro".

| A/B | 1 | 0 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

**Tabela 2.2:** Representação da operação de "união" de conjuntos clássicos

Da mesma forma que a intersecção, a união entre conjuntos clássicos cria a função $\{0,1\} \times \{0,1\} \rightarrow \{0,1\}$. Dessa forma a união entre conjuntos nebulosos também é representada por uma função tal que: $\oplus : [0,1] \times [0,1] \rightarrow [0,1]$. Também de maneira semelhante à intersecção, a união entre conjuntos nebulosos deve ser uma generalização da qual a união entre conjuntos clássicos é um caso específico.

No artigo original de Zadeh (1965), o operador escolhido para fazer a união foi $x \oplus y = max(x, y)$. As seguintes propriedades relevantes desse operador podem ser observadas:

- *Elemento Neutro*, 0 age como elemento neutro, tal que $x \oplus 0 = x$;
- *Comutativa*, ou seja, $x \oplus y = y \oplus x$;
- *Associativa*, ou seja, $x \oplus (y \oplus z) = (x \oplus y) \oplus z$;
- *Monotonicidade*, ou seja, se $v \leq w$ e $x \leq y$, então $v \oplus x \leq w \oplus y$.

Qualquer operador que satisfaça essas propriedades pode ser utilizado para fazer a união de conjuntos nebulosos e é chamado de *conorma triangular* ou simplesmente *t-conorma*. Assim como os operadores de norma triangular, os operadores de conorma triangular precedem a lógica nebulosa.

Assim como os operadores de norma triangular, há um operador de conorma triangular mais apropriado para cada situação, abaixo estão listados os mais relevantes segundo Lee (1990b):

- *União*: $x \vee y = max(x, y)$;
- *Soma Algébrica*: $x \tilde{+} y = x + y - xy$;
- *Soma Restrita*: $x \bigtriangledown y = min(1, x + y)$;
- *Soma Drástica*: $x \sqcup y = \begin{cases} x, \text{ se } y = 0; \\ y, \text{ se } x = 0; \\ 1, \text{ se } x, y > 0. \end{cases}$
- *Soma Disjunta*: $x \odot y = max\{min(x, 1 - y), min(1 - x, y)\}$.

Da mesma forma que acontece com os operadores de norma, algumas aplicações necessitam que o operador de conorma triangular apresente uma outra propriedade, a *idempotência*. Isso significa que, para qualquer valor de $x \in [0, 1]$, $x \oplus x \equiv x \oplus x \oplus x \equiv x \oplus x \oplus x \oplus x \equiv ....$ Em outras palavras, o operador deve garantir que $x \oplus x = x$. O único operador de conorma triangular que apresenta a propriedade da idempotência é a *união* (NGUYEN; WALKER, 2000, p. 96).

Um estudo mais profundo sobre normas e conormas triangulares pode ser feito consultando-se Nguyen e Walker (2000), Albert (1978) e Voxman e Goetshchel (1983).

## 2.2 Autômatos Finitos

Entre os modelos matemáticos computacionais existentes, os Autômatos Finitos são os mais simples e menos poderosos disponíveis. Ainda que possuam capacidade de processar informações, recebendo dados de entrada e exibindo uma saída; a falta de uma memória auxiliar (de maneira que toda a memória que o Autômato Finito (AF) dispõe - representada pela sua configuração - é definida junto com o próprio autômato, não podendo ser alterada em tempo de execução) impõe sérias restrições ao tipo de processamento possível ao modelo.

Levando-se em conta as limitações dos Autômatos Finitos, porque utilizá-los neste trabalho, ao invés de modelos mais poderosos?

Em primeiro lugar, porque os Autômatos Finitos são exatamente adequados à função de análise léxica e reconhecimento de cadeias de texto (LEWIS; PAPADIMITRIOU, 2000).

Em segundo lugar, porque o uso de modelos mais complexos (como os Autômatos de Pilha e as Máquinas de Turing) não traria nenhum benefício à resolução do problema de reconhecimento aproximado de cadeias como proposto neste trabalho, ao mesmo tempo que aumentaria o custo computacional de execução do programa que implementaria tais modelos. Isso acontece porque a solução proposta faz a análise da cadeia símbolo a símbolo, sem utilizar memória auxiliar.

Em terceiro lugar, mesmo que se deseje fazer uma análise mais complexa do que simplesmente a análise sintática ou ainda, recorrer à memória auxiliar para fazer o reconhecimento de cadeias levando em conta um conjunto de símbolos, existem modelos matemáticos equivalentes aos modelos de autômato mais complexos, próximos o suficiente dos Autômatos Finitos para que a transposição da estrutura nebulosa apresentada nesse trabalho seja feita sem alterações drásticas. No caso dos Autômatos de Pilha, pode-se recorrer ao Autômato de Pilha Estruturada (NETO, 1994); esse modelo define um conjunto de Autômatos Finitos capazes de chamar um ao outro recursivamente através de operações de pilha, sendo que a pilha não é utilizada em nenhuma outra tarefa. No caso das Máquinas de Turing, existem os Autômatos Adaptativos (NETO; BRAVO, 2002); esse modelo consiste de um Autômato Finito capaz de alterar sua própria estrutura, tal capacidade o torna capaz de processar linguagens dependentes de contexto.

Justificado o motivo pelo qual o autômato finito foi escolhido como modelo computacional utilizado nesse trabalho, é necessário ver com mais detalhes a classe de linguagens representadas por ele e a sua definição formal.

### 2.2.1 Linguagens e Linguagens Regulares

A matemática possui uma série de aparatos teóricos adequados a cada um dos seus campos de estudo. Como parte indivisível da matemática, a teoria da computação também tem um conjunto próprio de conceitos para tratar a manipulação de símbolos.

A primeiro deles é o *alfabeto*. O alfabeto nada mais é do que um conjunto finito qualquer de símbolos, como por exemplo o alfabeto das vogais $\{a, e, i, o, u\}$ ou o alfabeto decimal $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

O segundo conceito importante é a *cadeia*. Uma cadeia nada mais é do que uma seqüência justaposta de símbolos pertencentes a um alfabeto. Por exemplo, *uia* é uma cadeia válida sobre o alfabeto $\{a, e, i, o, u\}$; assim como 1620 é uma cadeia válida sobre o alfabeto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Uma cadeia vazia pode pertencer a qualquer linguagem e é representada pelo símbolo $\varepsilon$. O conjunto de todas as cadeias sobre um alfabeto $\Sigma$, incluindo a cadeia vazia, é representado pela notação $\Sigma^*$.

Por fim, uma linguagem nada mais é do que qualquer conjunto de cadeias sobre um alfabeto $\Sigma$; em outras palavras, um subconjunto de $\Sigma^*$. Dessa forma, $\emptyset$, $\Sigma$ e $\Sigma^*$ são linguagens válidas; assim como $L_1 = \{ai, ui, uia\}$ é uma linguagem válida sobre o alfabeto $\{a, e, i, o, u\}$ e $L_2 = \{w \in \Sigma^* : \text{w é um número par}\}$ é uma linguagem válida sobre o alfabeto $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Como qualquer outro conjunto, toda linguagem tem o seu *complemento* (ou seja, o complemento de uma linguagem $L$ é igual a $\Sigma^* - L$). Além disso, as operações de *união*, *intersecção* e *diferença* funcionam com as linguagens de maneira idêntica a qualquer conjunto.

Além dessas operações, existem operações que só fazem sentido com linguagens, não com conjuntos ordinários. A primeira delas é a *concatenação*, que nada mais é do que a justaposição de todas as cadeias de uma linguagem com todas as cadeias de outra linguagem. Seja $L_1$ e $L_2$ duas linguagens sobre $\Sigma$, a concatenação das duas linguagens tal que $L = L_1 \circ L_2$ ou simplesmente $L = L_1 L_2$ é igual a:

$$L = \{w \circ v \text{ para qualquer } w \in L_1 \text{ e } v \in L_2\}$$

Outra operação importante é a *estrela de Kleene*, também conhecida como *fechamento de Kleene*. Essa operação resulta no conjunto de todas as cadeias formadas pela concatenação sucessiva de zero ou mais cadeias de uma linguagem. Qualquer cadeia concatenada zero vezes é igual a $\varepsilon$, enquanto qualquer cadeia concatenada uma vez é igual a ela mesma. Dessa forma:

$$L^* = \{w \in \Sigma^* : w = w_1 \circ w_2 \circ ... \circ w_k, \text{ para qualquer } k \geq 0 \text{ e qualquer } w_1, w_2, ..., w_n \in L\}$$

Por fim, existem diversas classes de linguagem, sendo que a classe de linguagem relevante à esse trabalho é a classe das *linguagens regulares*. A classe das linguagens regulares é equivalente ao conjunto de cadeias aceitáveis por um autômato finito, em outras palavras, toda linguagem regular tem um autômato finito capaz de representá-la e todo autômato finito representa uma linguagem regular. A prova desses teoremas são um pouco extensas e fogem do escopo desse trabalho, mas podem ser encontradas em Lewis e Papadimitriou (2000).

Para uma linguagem ser regular (e, conseqüentemente possuir um autômato finito equivalente), é necessário que ela obedeça as seguintes propriedades:

1. Se $L = \{\}$ ou $L = \{a : \text{para qualquer } a \in \Sigma\}$, então $L$ é regular;
2. Se $L_1$ e $L_2$ são regulares, então $L = L_1 \cup L_2$ também é regular;
3. Se $L_1$ e $L_2$ são regulares, então $L = L_1 \circ L_2$ também é regular;
4. Se $L_1$ é regular, então $L = L_1^*$ também é regular.

Resumindo, uma linguagem é regular quando é formada pela linguagem vazia, pelos símbolos do alfabeto, pela concatenação ou união de duas linguagens regulares, ou pelo fechamento de Kleene de uma linguagem regular.

### 2.2.2 Definição dos Autômatos Finitos

Uma vez definidos conceitos básicos como alfabeto, cadeia e linguagem e limitadas as linguagens que podem ser reconhecidas por um autômato finito, se faz necessário o estudo do funcionamento do autômato finito propriamente dito.

Um autômato finito é composto por uma *fita de entrada*, um *cabeçote de leitura* e um *controle finito*, que é um conjunto finito de estados internos distintos, conforme mostrado na figura 2.1.

Inicialmente, o cabeçote de leitura encontra-se na posição mais à esquerda da fita de entrada e o controle finito encontra-se em um estado especial chamado *estado inicial*. Periodicamente o autômato lê o símbolo atual na fita de entrada, move o cabeçote da fita de entrada para a próxima posição à direita e muda o estado do controle finito. O novo estado selecionado pelo controle finito depende apenas do símbolo de entrada lido anteriormente e do estado que o controle finito se encontrava anteriormente. Essa descrição corresponde exatamente ao Autômato Finito Determinístico (AFD). Esse processo se repete sucessivamente até que o autômato leia o último símbolo na fita de entrada e faça a transição

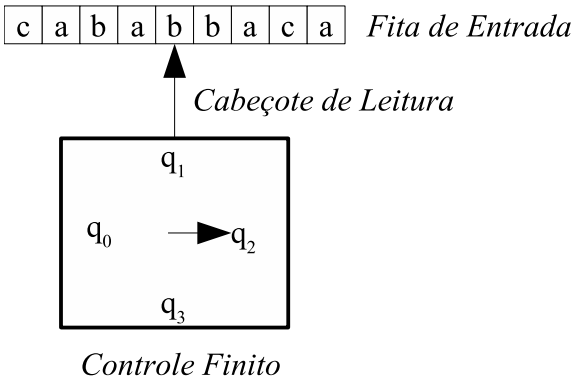


**Figura 2.1:** Autômato Finito (LEWIS; PAPADIMITRIOU, 2000).

correspondente. Caso o controle finito tenha selecionado um estado interno que pertence ao subconjunto dos *estados finais*, a cadeia de entrada é considerada aceita. Caso contrário, ela é rejeitada. O conjunto de cadeias que um determinado autômato finito aceita é a *linguagem aceita pelo autômato*, de maneira que é possível fazer a representação finita de algumas linguagens através de autômatos finitos, inclusive *linguagens infinitas*.

A definição matemática de um AFD é feita da seguinte maneira:

**Definição 1** *Um Autômato Finito Determinístico é uma quíntupla* $M = (Q, \Sigma, \delta, s, F)$*, onde:*

$Q$ *é o conjunto finito de estados,*
$\Sigma$ *é o conjunto de símbolos (alfabeto) da máquina,*
$\delta$ *é a função de transição de* $Q \times \Sigma$ *para* $Q$*,*
$s \in Q$ *é o estado inicial,*
$F \subseteq Q$ *é o conjunto de estados finais.*

Dessa forma, através da função de transição, pode-se determinar qual o próximo estado a partir do estado atual e do símbolo lido. Uma vez formalizada a estrutura do autômato, é necessário formalizar também a noção da computação realizada por ele. A computação de um autômato finito é feita através de sucessivas transições, de maneira que é possível representar esse processo simplesmente utilizando a notação da configuração da máquina a cada transição. Cada configuração é dada por uma tupla formada pelo estado atual e pela porção não-lida da cadeia de entrada, ou seja, $(q, w) \in Q \times \Sigma^*$.

Sejam $(q, w)$ e $(q', w')$ duas configurações distintas do AFD M. A relação binária $(q, w) \vdash_M (q', w')$ é verdadeira se, e somente se, $w = w'a$, $a \in \Sigma$ e $\delta(q, a) = q'$. Cada transição através da relação binária $\vdash_M$ pode ser visualizada como um passo de computação do autômato finito.

O fechamento transitivo e reflexivo de $\vdash_M$ é representado por $\vdash_M^*$, de maneira que $(q, w) \vdash_M^* (q', w')$ significa que $(q, w)$ produz $(q', w')$ após um número de passos sucessivos maior ou igual a zero. Um autômato aceita uma cadeia de símbolos $w$ quando $(q, w) \vdash_M^* (q', \varepsilon)$; onde $q' \in F$ e $\varepsilon$ significa cadeia vazia (como visto na subseção 2.2.1).

Abaixo um exemplo de um AFD:

**Exemplo 1** *Seja $M_1$ um AFD, tal que $M_1$ aceita qualquer cadeia de texto em $\{a, b\}^*$, desde que haja um número par de a's e b's. Esse autômato é dado por $M_1 = (Q, \Sigma, \delta, s, F)$, onde:*

$$
\begin{aligned}
&Q = \{q_0, q_1, q_2, q_3\} \\
&\Sigma = \{a, b\} \\
&s = q_0 \\
&F = \{q_0\} \\
&\delta = \{((q_0, a), q_1), ((q_0, b), q_2), ((q_1, a), q_0), ((q_1, b), q_3), ((q_2, a), q_3), ((q_2, b), q_0), \\
&\qquad ((q_3, a), q_2), ((q_3, b), q_1)\}
\end{aligned}
$$

O funcionamento do autômato do exemplo é o seguinte. O estado inicial (e final) $q_0$ representa a ocorrência par de a's e b's. O estado $q_1$ representa uma quantidade ímpar de a's e par de b's, enquanto o estado $q_2$ representa uma quantidade par de a's e ímpar de b's. Por fim, o estado $q_3$ representa sempre uma quantidade ímpar de a's e b's. Conforme os símbolos são lidos na entrada, a função de transição determina o estado atual a partir do estado anterior (que representa a sub-cadeia lida até o passo anterior) e do símbolo atual.

Supondo que $M_1$ tenha como cadeia de entrada $w = abba$, a sua configuração inicial é $(q_0, abba)$. Dessa forma:

$$(q_0, abba) \vdash_{M_1} (q_1, bba) \vdash_{M_1} (q_3, ba) \vdash_{M_1} (q_1, a) \vdash_{M_1} (q_0, \varepsilon)$$

Embora a representação exaustiva da função de transição permita a compreensão do funcionamento do autômato, em muitos casos, é necessária uma representação diferente. Os AFD também podem ser representados utilizando um tipo

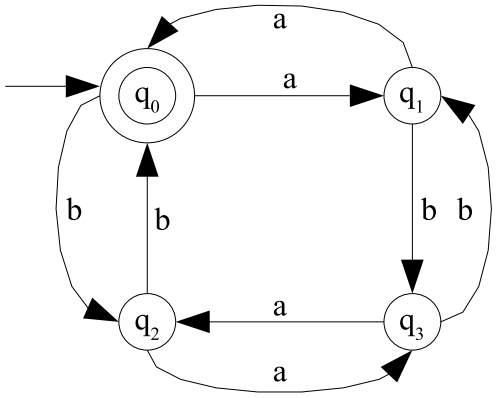


**Figura 2.2:** Representação do Autômato $M_1$ através do diagrama de estados.

especial de grafo dirigido chamado *diagrama de estados*. A figura 2.2 mostra a representação do autômato $M_1$ através de um diagrama de estados.

### 2.2.3 Não-Determinismo e Transições Vazias

O modelo dos autômatos finitos pode ser aprimorado adicionando-se as capacidades de *não-determinismo* e de executar *transições vazias*. O não-determinismo faz com que não seja mais possível determinar o próximo estado apenas com o estado atual e símbolo lido, uma vez que a *relação* (não mais função, como nos AFD) de transição retorna não apenas um estado, como no caso dos AFD, mas sim um conjunto de estados (possivelmente, vazio).

Uma outra possibilidade é permitir que o autômato finito faça transições sem ler nenhum símbolo na fita de entrada nem mover o cabeçote de leitura para a próxima posição. Uma vez que, $w = abc = \varepsilon abc = \varepsilon a\varepsilon bc = \varepsilon a\varepsilon b\varepsilon c = \varepsilon a\varepsilon b\varepsilon c\varepsilon$, e assim por diante, as *transições vazias* podem ocorrer em qualquer ponto da cadeia de entrada; inclusive, um Autômato Finito com transições vazias é capaz de fazer infinitas transições vazias apenas com uma regra de transição.

Definindo matematicamente o não-determinismo e a capacidade de executar transições vazias, eis que:

**Definição 2** *Um Autômato Finito Não-Determinístico é uma quíntupla* $M = (Q, \Sigma, \Delta, s, F)$*, onde:*

$Q$ *é o conjunto finito de estados,*
$\Sigma$ *é o conjunto de símbolos (alfabeto) da máquina,*
$\Delta$ *é a relação de transição, tal que* $\Delta \subseteq Q \times \{\Sigma \cup \varepsilon\} \times Q$
$s \in Q$ *é o estado inicial,*
$F \subseteq Q$ *é o conjunto de estados finais.*

É interessante notar que a única diferença entre um AFD e um Autômato Finito Não-Determinístico (AFND) é a função/relação de transição. Na realidade, o AFND pode ser encarado apenas como uma ferramenta de notação resumida para um AFD. A implementação computacional direta de um AFND é bastante complexa: na ocorrência de um não-determinismo, é necessário testar cada possibilidade individualmente até encontrar uma que aceite ou rejeite a cadeia dada. Em determinados problemas, isto pode significar que a cadeia só pode ser aceita ou rejeitada em tempo exponencialmente proporcional ao comprimento da cadeia.

No entanto, é interessante notar que a capacidade de não-determinismo e de realizar transições vazias de um AFND não o torna capaz de aceitar linguagens não poderiam ser aceitas por um AFD. Isso significa que, para cada AFND existe um AFD equivalente. A demonstração e o procedimento para a conversão de um AFND para um AFD podem ser encontrados em Lewis e Papadimitriou (2000).

Os exemplos abaixo ilustram um AFND e seu equivalente determinístico:

**Exemplo 2** *Seja* $M_2$ *um AFND, tal que* $M_2$ *aceita qualquer* $w \in \Sigma^*$*, desde que 'ab' ou 'cb' sejam sub-cadeias de* $w$*. Esse autômato é dado por* $M_2 = (Q, \Sigma, \Delta, s, F)$*, onde:*

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
$\Sigma = \{a, b, c\}$
$\Delta$*, conforme o diagrama de estados da figura 2.3*
$s = q_0$
$F = \{q_7\}$

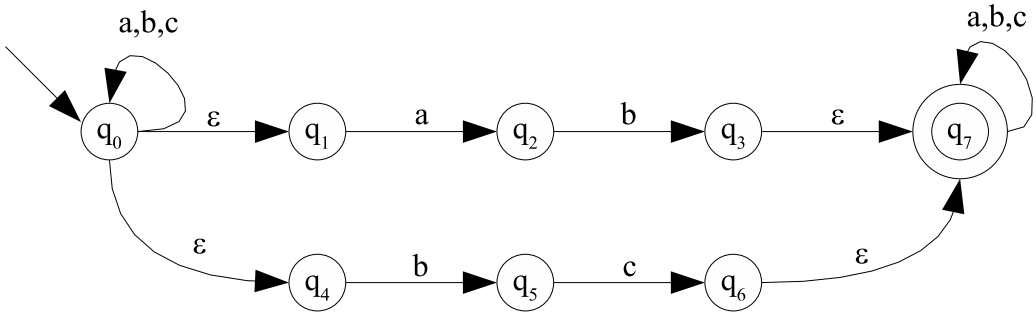


**Figura 2.3:** Autômato Finito Não-Determinístico $M_2$.

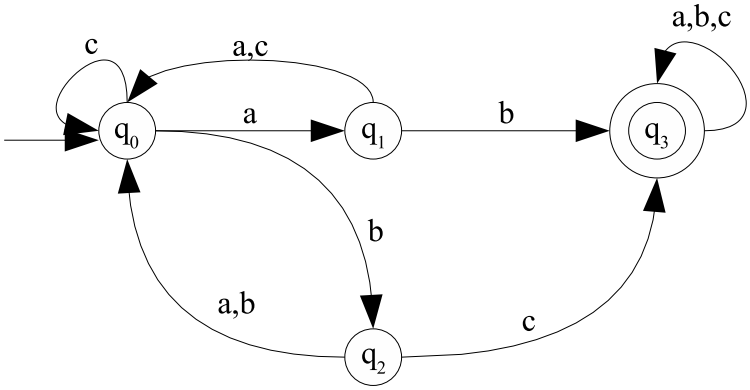


**Figura 2.4:** Autômato Finito Determinístico $M_3$ equivalente ao Autômato Finito Não-Determinístico $M_2$.

**Exemplo 3** *Seja $M_3$ um AFD tal que $M_3$ aceite a mesma linguagem aceita por $M_2$:*

$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b, c\}$
$\delta$, *conforme o diagrama de estados da figura 2.4*
$s = q_0$
$F = \{q_3\}$

# 3 Reconhecimento Aproximado de Cadeias

## 3.1 Definição do Reconhecimento Aproximado de Cadeias

De maneira simplificada, é possível definir o reconhecimento aproximado de cadeias como a busca de uma cadeia (chamada de *padrão*) em uma outra cadeia (chamada de *texto*), tolerando e quantificando diferenças entre o padrão procurado e o texto encontrado, e retornando a posição onde o padrão foi reconhecido, assim como um valor associado à diferença (ou similaridade) entre o padrão procurado e o texto encontrado. Essa tolerância é definida de maneira arbitrária, assim como a métrica utilizada para aferir a diferença ou similaridade entre as seqüências.

Entretanto, para dar seguimento a este trabalho, é necessário definir o problema de maneira mais formal. Sendo assim, eis que:

**Definição 3** *Considere-se que:*

*Seja* $\Sigma$ *um alfabeto finito com* $n = |\Sigma|$ *símbolos distintos;*
*Seja* $\omega$ *o padrão procurado, tal que* $\omega = \{x_1 x_2 ... x_m\}$, $\omega \in \Sigma^*$ *e* $m = |\omega|$;
*Seja* $\alpha$ *o texto onde a busca é realizada, tal que* $\alpha = \{y_1 y_2 ... y_o\}$, $\alpha \in \Sigma^*$ *e* $o = |\alpha|$;
*Seja* $k \in \Re$ *o erro máximo tolerado;*
*Seja* $d : \Sigma^* \times \Sigma^* \rightarrow \Re$ *a função de distância;*
*Seja* $d' : \Sigma^* \times \Sigma^* \rightarrow \Re$ *a função de similaridade.*

*Dessa forma, o problema do reconhecimento aproximado de cadeias é definido da seguinte maneira: dado* $\omega$, $\alpha$, $k$ *e* $d(\cdot)$, *quais os valores de* $j$, *tal que exista um* $i$, *de maneira que* $d(\omega, (y_i ... y_j)) \leq k$ *(distância) ou* $d(\omega, (y_i ... y_j)) \geq k$ *(similaridade).*

Define-se a função $d(\beta, \omega)$ (ou a função $d'(\beta, \omega)$) como o menor custo possível associado às *operações* necessárias para transformar $\beta$ em $\omega$. As operações, por sua vez, nada mais são do que um conjunto finito de regras no formato $\kappa(a, b) = c$, onde $a, b \in \Sigma$, $a \neq b$ e $c \in \Re^+$. O custo total da função $d(\cdot)$ é contabilizado a partir dos custos individuais das operações (geralmente através da soma, mas não necessariamente).

Embora teoricamente seja possível definir uma quantidade infinita de operações arbitrárias, as quatro operações seguintes podem ser consideradas básicas pela sua generalidade (uma vez que não são específicas de nenhum domínio):

- *Inserção*: Insere um símbolo ausente na cadeia $y_i...y_j$, de maneira que a operação $\kappa(\varepsilon, a)$ a transforma em $y_i...a...y_j$, onde $a \in \Sigma$.
- *Exclusão*: Excluí um símbolo presente na cadeia $y_i...y_{k-1}y_ky_{k+1}...y_j$, de maneira que a operação $\kappa(y_k, \varepsilon)$ a transforma em $y_i...y_{k-1}y_{k+1}...y_j$.
- *Substituição*: Substitui um símbolo presente na cadeia $y_i...y_{k-1}y_ky_{k+1}...y_j$ por outro símbolo, de maneira que a operação $\kappa(y_k, a)$ a transforma em $y_i...y_{k-1}ay_{k+1}...y_j$, onde $a \in \Sigma$ e $a \neq y_k$.
- *Transposição*: Troca a posição de dois símbolos consecutivos presentes na cadeia $y_i...y_{k-1}y_ky_{k+1}...y_j$, de maneira que a operação $\kappa(y_ky_{k+1}, y_{k+1}y_k)$ a transforma em $y_i...y_{k-1}y_{k+1}y_k...y_j$, onde $y_k \neq y_{k+1}$.

Nem todos os mecanismos de reconhecimento aproximado e métricas de distância/similaridade trabalham com todas as operações acima descritas. De fato, pode-se simular uma operação de substituição através de uma inserção seguida de uma exclusão; assim como pode-se simular uma operação de transposição através de duas operações de substituição consecutivas. Obviamente, uma vez que são necessárias duas operações (ou quatro, quando simulando uma transposição apenas com inclusões e exclusões) para simular uma única operação, o custo associado é diferente e, dependendo da métrica utilizada, os resultados também podem ser diferentes. Também é importante ressaltar que, embora os erros de transposição sejam muito freqüentes, existem poucos mecanismos de reconhecimento aproximado de cadeias capazes de lidar com transposições, já que o tratamento das transposições exige que o mecanismo de reconhecimento armazene o símbolo anteriormente lido (NAVARRO, 2001).

Uma vez que o problema e as operações foram definidos, é importante conhecer quais são as funções/métricas de distância/similaridade mais utilizadas. Exceto quando dito o contrário, o valor da função de distância/similaridade é obtido através da soma dos custos individuais das operações utilizadas pela função:

- *Distância de Levenshtein* ou *Distância de Edição*: permite as operações de inserção, exclusão e substituição; atribuindo um custo $\kappa(\cdot) = 1$, independente da operação ou dos símbolos envolvidos. A distância de edição, na realidade, nada mais é do que o número mínimo de inserções, exclusões e substituições para tornar duas cadeias iguais. A distância é simétrica (ou seja, $d_{Levenshtein}(\alpha, \omega) = d_{Levenshtein}(\omega, \alpha)$, e é igual a $0 \leq \ d_{Levenshtein}(\alpha, \omega) \leq max(|\alpha|,|\omega|)$.

- *Distância de Levenshtein Generalizada*: assim como a distância de Levenshtein simples, permite apenas as operações de inserção, exclusão e substituição. No entanto, o custo das operações pode ser definido de maneira menos restrita, podendo ser qualquer número real não-negativo. Além disso, diferentes operações com diferentes símbolos podem ter custos específicos atribuídos, permitindo que se represente o conhecimento da aplicação sem comprometer a generalidade do modelo. Por fim, ao contrário da distância de Levenshtein simples, qualquer função pode ser utilizada para agregar o custo individual das operações $\kappa(\cdot) \rightarrow \Re^+$ e retornar o valor da distância para a função $d_{GLevenshtein}(\cdot)$. À despeito do nome e de acordo com a escolha dessa função, a distância de Levenshtein generalizada pode ser também utilizada para medir o grau de similaridade entre duas cadeias.

- *Distância de Hamming*: só permite a operação de substituição, à qual é atribuído um custo fixo igual a 1. Essa métrica é bastante utilizada na área de processamento de sinais. Obviamente, $d_{Hamming}(\omega, \alpha) = \infty$, apenas se $|\omega| \neq |\alpha|$, caso contrário, a distância é finita e simétrica, e é igual a $0 \leq \ d_{Hamming}(\alpha, \omega) \leq |\alpha|$.

- *Distância Episódica*: permite apenas operações de inserção, todas com custo 1. É utilizada em problemas onde é necessário encontrar a sucessão correta de uma série de eventos ocorridos num período de tempo relativamente curto. Essa métrica de distância não é simétrica e existem casos onde não é possível transformar $\alpha$ em $\omega$ (na realidade, só é possível transformar $\alpha$ em $\omega$ quando $\alpha$ é uma sub-cadeia de $\omega$). Dessa forma, $d_{Episódica}(\alpha, \omega) = (|\omega| - |\alpha|), \infty$.

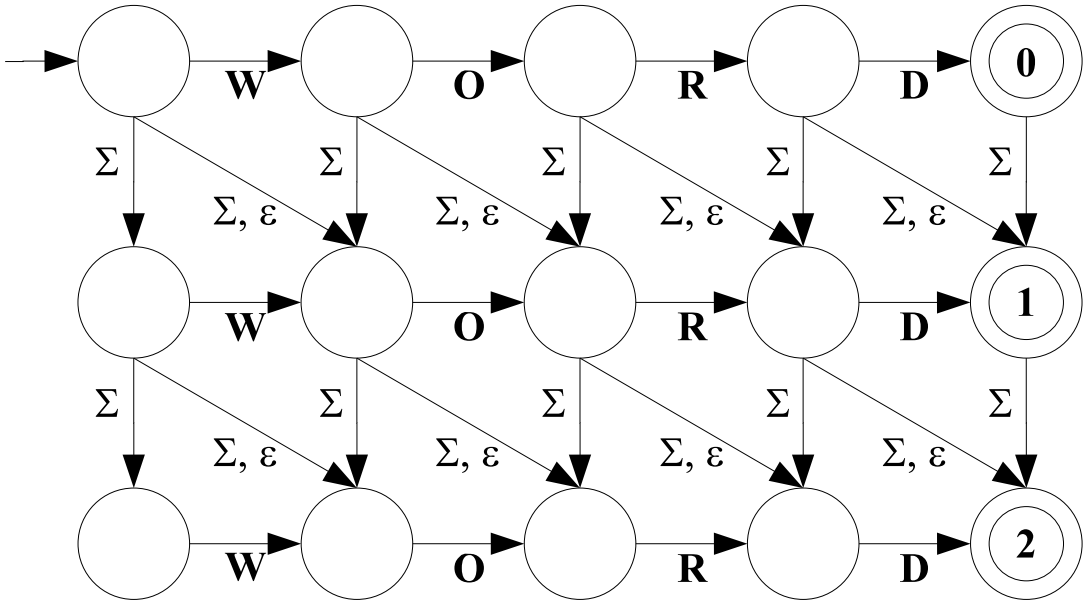


**Figura 3.1:** Autômato Finito Não-Determinístico capaz de reconhecer a cadeia "WORD", utilizando distância de Levenshtein simples e permitindo um erro igual a $k \leq 2$. Os números indicados dentro dos estados finais dizem respeito à quantidade de erros encontrada.

- *Distância da Subseqüência Comum Mais Longa*: permite apenas as operações de inserção e exclusão, atribuindo a todas custo 1. Como o nome indica, ela mede o comprimento da maior subseqüência coincidente entre as cadeias do padrão procurado e do texto, respeitando a posição dos símbolos. A distância indica o número de símbolos não-coincidentes. A distância é simétrica e é igual a $0 \leq d_{DSCML}(\alpha, \omega) \leq (|\omega| + |\alpha|)$.

Uma das mais antigas abordagens para a implementação do reconhecimento aproximado de cadeias é através do uso de *Autômatos Finitos Não-Determinísticos*. Flexíveis, permitem o reconhecimento de cadeias de comprimento arbitrário, com uma taxa de erro também arbitrária e utilizando qualquer uma das métricas apresentadas (exceto a distância de Levenshtein generalizada). Expressivos, permitem que variáveis como operações permitidas, custos atribuídos, métrica utilizada e taxa de erro tolerado sejam rapidamente identificados (vide o exemplo da figura 3.1). Por outro lado, a quantidade excessiva de estados e a complexidade computacional necessária para o processamento dos não-determinismos (ou o número ainda maior de estados necessários para convertê-lo em uma versão determinística) dificultam o seu uso na prática.

Não é o objetivo desse trabalho, no entanto, se aprofundar na análise do uso de Autômato Finito Determinístico e de Autômato Finito Não-Determinístico

no reconhecimento aproximado de cadeias. Entretanto, é possível um estudo mais detalhado no assunto consultando-se Ukkonen (1985), Wu e Manber (1992), Baeza-Yates (1991, 1996) e Baeza-Yates e Navarro (1999).

## 3.2 Aplicações do Reconhecimento Aproximado de Cadeias

Como foi antecipado na introdução, o reconhecimento aproximado de cadeias é utilizado quando se necessita fazer uma busca por um padrão em uma cadeia sujeita a diversos erros indesejáveis. Existem diversas áreas onde o reconhecimento aproximado de cadeias é utilizado, porém este trabalho detalhará apenas três delas: biologia computacional, processamento de sinais e reconhecimento de textos.

### 3.2.1 Biologia Computacional

O Projeto Genoma e as demais pesquisas recentes no campo da engenharia genética trouxeram à tona problemas interessantes e novos, inclusive problemas para a ciência da computação. O DNA é responsável pela codificação das características de um organismo vivo: estudar e decifrar o seu significado poderá, no futuro, garantir o acesso à tecnologia médica e biológica inimaginável atualmente.

A despeito de sua importância, o DNA nada mais é do que um longuíssimo encadeamento de quatro bases nitrogenadas distintas, a saber: adenina, citosina, guanina e timina. Além disso, em situações anômalas e/ou incomuns é possível encontrar uma quinta base nitrogenada, a uracila, resultado da conversão da Citosina.

Apesar de ser uma aplicação de reconhecimento de cadeias, as aplicações onde o reconhecimento estrito de cadeias podem ser utilizados são extremamente restritos: diversos fatores tais como erros inerentes ao processo de coleta de dados, mutações genéticas menos significativas, alterações evolucionárias e outros fatores fazem com que a busca exata por seqüências de bases nitrogenadas raramente retornem algum resultado útil.

O reconhecimento aproximado de cadeias, por outro lado, oferece resultados muito mais satisfatórios para os pesquisadores. Em anos de Projeto Genoma e outras pesquisas, os geneticistas adquiriram um extenso conhecimento sobre as mutações observadas com freqüência nas seqüências de DNA: se o mecanismo

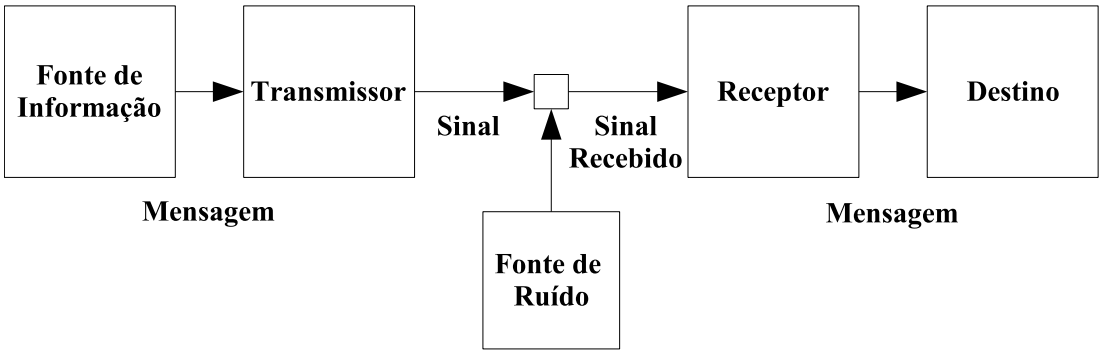


**Figura 3.2:** Diagrama de um sistema de comunicação arbitrário (SHANNON, 1948).

utilizado no reconhecimento aproximado de cadeias for capaz de levar em conta esse conhecimento e atribuir um peso menor a essas mutações ao quantificar a diferença entre a seqüência genética desejada e a observada, melhor ainda. Além disso, a quantificação das diferenças entre duas seqüências de genes é bastante útil para a reconstrução de árvores evolucionárias (árvores filogenéticas).

Um bom ponto de partida para estudar a aplicação do reconhecimento aproximado de cadeias aplicado à biologia computacional pode ser obtido consultando-se Needleman e Wunsch (1970), Sellers (1974) e também Altschul et al. (1990).

### 3.2.2 Processamento de Sinais

Uma outra área onde o reconhecimento aproximado é bastante utilizado é no processamento de sinais, sobretudo na correção de erros e no reconhecimento de voz.

Uma vez que toda informação transmitida através de um meio físico está inexoravelmente sujeita a erros (vide figura 3.2), é possível recorrer à Teoria da Informação (SHANNON, 1948) para obter as probabilidades de ocorrência de erros e usar algum mecanismo para recuperar a informação contida na mensagem original (na realidade, processar o sinal recebido e obter uma mensagem recebida mais próxima da mensagem transmitida, uma vez que a própria Teoria da Informação determina a impossibilidade de transmitir uma mensagens sem erros).

Entretanto, falar em "reconhecimento aproximado de cadeias" em correção de erros não é o mais apropriado: geralmente não se deseja encontrar um padrão, apenas eliminar os erros de uma mensagem recebida (seja qual for o critério utilizado pelo mecanismo para definir 'erro'). Exatamente por isso, as contribuições do reconhecimento de erros quanto à busca aproximada são bastante incipientes,

no entanto, a área é responsável pela mais importante métrica de similaridade entre cadeias, a *distância de Levenshtein* (explicada com detalhes na seção 3.1)

O reconhecimento de voz, por outro lado, é uma aplicação bastante característica do reconhecimento aproximado de cadeias. Consiste, basicamente, em determinar uma cadeia de símbolos (ou seja, um texto), a partir de um sinal de áudio. É praticamente impossível obter o reconhecimento estrito perfeito: existem diversos deslocamentos temporais no sinal, diferenças de timbre e pronúncia mesmo por dois sinais de áudio gerados pela mesma pessoa e sons ambientes que interferem no sinal, por exemplo.

Dessa forma, mesmo distinguir uma palavra de um repertório extremamente limitado (algo como 20 ou 30 palavras, por exemplo), é uma tarefa que não pode ser feita sem a utilização do reconhecimento aproximado.

O aumento do uso de interfaces homem-máquina não-escritas pelos sistemas computacionais mais modernos abre grandes possibilidades para o uso de mecanismos de reconhecimento aproximado de cadeias. Bancos de dados multimídia, por exemplo, armazenam imagens, sons e vídeos que, eventualmente, precisaram passar por buscas aproximadas, não-estritas (existem poucas aplicações onde a busca estrita de imagens, sons ou vídeos é realmente útil, dada a dificuldade em se obter amostras estritas deste tipo de mídia sem recorrer ao material original); formas de interação com computadores através de comandos de voz e gestos (VOLLET, 2001): todas essas aplicações demandam mecanismos de reconhecimento aproximado.

Além disso, a disseminação do uso de sistemas de comunicação sem fio (*wireless*), demanda por mecanismos de correção de erros cada vez mais sofisticados: o ar por si só é um péssimo meio de transmissão de sinais digitais, a saturação cada vez maior das bandas de transmissão disponíveis torna o problema ainda pior. Dessa forma, há um enorme potencial do uso de mecanismos de reconhecimento aproximado de cadeias e/ou correção de erros na área de processamento de sinais para os próximos anos.

Para obter maiores informações sobre o uso do reconhecimento aproximado de cadeias no processamento de sinais, os artigos de Levenshtein (1965), Vintsyuk (1968) e Dixon e Martin (1979) são ótimas referências.

### 3.2.3 Recuperação de Texto

A recuperação de texto é uma das mais antigas e disseminadas aplicações onde o reconhecimento aproximado de cadeias é utilizado. Existem diversas áreas onde esse problema aparece, mas a Recuperação de Informações é uma das mais críticas.

Embora pequenas quantidades de texto possam ser corrigidas manualmente e servir como base para a recuperação de informações utilizando ferramentas de reconhecimento estrito de cadeias; a recuperação de informações só se torna atraente com uma quantidade substancialmente maior de texto, como a World Wide Web, com uma quantidade de texto da ordem de dezenas de gigabytes ou maior.

Utilizando um mecanismo de reconhecimento estrito de cadeias, é possível que um termo inserido com algum erro na base de dados jamais possa ser recuperado. Por exemplo, textos inseridos através de OCR (Optical Character Recognition - reconhecimento óptico de caracteres) possuem uma taxa de erro entre 7% e 16%. Erros de digitação atingem entre 1% e 3,2% dos textos, enquanto erros ortográficos respondem por taxas de erro entre 1,5% e 2,5%. A despeito da freqüência substancial de erros que ocorrem ao inserir o texto na base do sistema de recuperação de informações - seja manualmente ou não - 80% desses erros podem ser corrigidos através de uma única operação básica de reconhecimento aproximado de cadeias (inserção, exclusão, substituição e transposição - maiores detalhes na seção 3.1. Todos esses valores numéricos foram retiradas do trabalho de Navarro (2001).

Levando-se em conta a crescente necessidade de fazer a recuperação de informações de bases multi-linguais (como a World Wide Web ou um banco de dados corporativo de uma empresa multinacional) e geradas em épocas diferentes; a utilização de aplicações que implementam algum algoritmo de reconhecimento aproximado de cadeias se faz cada vez mais necessário pois, na realidade, poucas são as aplicações de recuperação de informações que não utilizam. Dessa forma, é possível encontrar informações a partir de uma palavra ou nome estrangeiro grafado erroneamente (por exemplo, 'Greenhalg' ao invés de 'Greenhalgh') ou encontrar uma informação escrita em forma arcaica do idioma utilizando um padrão na forma moderna (como procurar por 'farmácia' e encontrar 'pharmácia').

Alguns trabalhos importantes que tratam da utilização do reconhecimento aproximado de cadeias para a recuperação de texto e que podem ser consultados


**Figura 3.3:** Exemplo de busca utilizando a grafia errônea de 'Greenhalgh'. É interessante notar que o próprio sistema sugere a grafia correta.

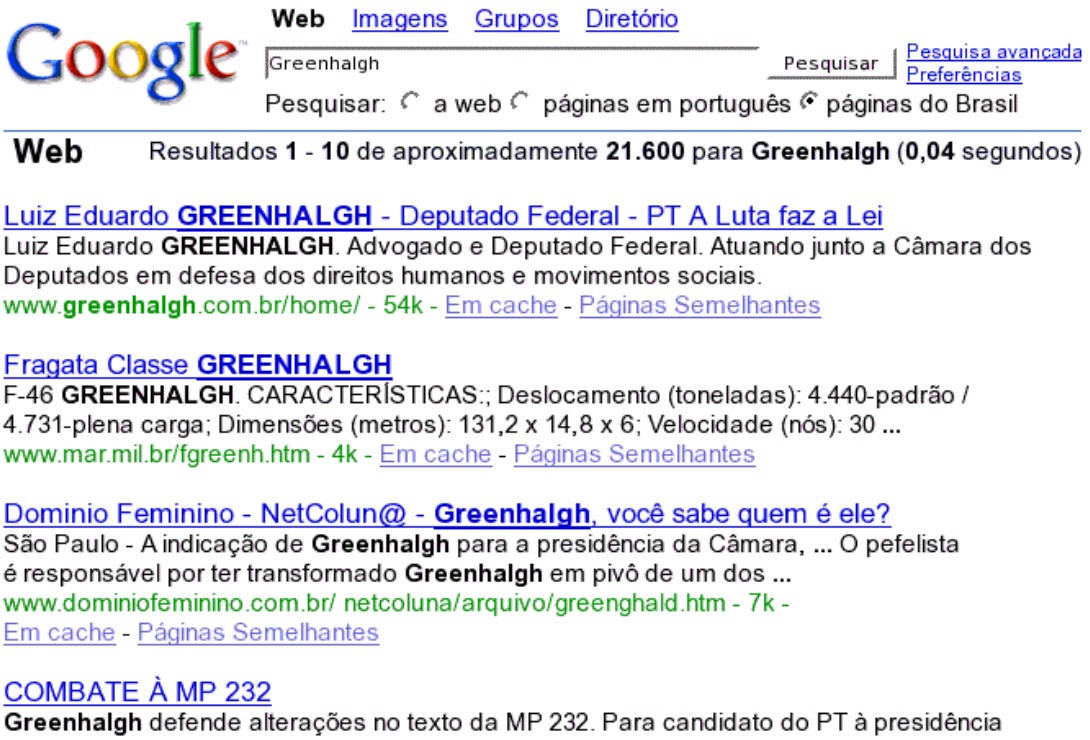


**Figura 3.4:** Exemplo de busca utilizando a grafia correta de 'Greenhalgh'. É interessante notar que 9,6% da informação recuperada através dos dois padrões apresentados está grafada de maneira errônea.

incluem Wagner e Fisher (1974), Lowrance e Wagner (1975), Nesbit (1986) e Owolabi e McGregor (1988).

# 4 Autômatos Finitos Nebulosos no Reconhecimento Aproximado de Cadeias

## 4.1 Autômatos Finitos Nebulosos

Reiterando as afirmações feitas na seção 3.1, os Autômatos Finitos Não-Determinísticos possuem diversas características desejáveis para resolver o problema de reconhecimento aproximado de cadeias: são flexíveis, intuitivos e os Autômatos Finitos por si mesmos são excelentes no reconhecimento de cadeias.

Entretanto, a ferramenta apresenta uma grande desvantagem durante a implementação: o custo computacional necessário para permitir ao modelo tolerar uma quantidade relativamente pequena de erros em uma cadeia de comprimento relativamente curto é alto, escalando rapidamente conforme se torna necessário tolerar taxas de erro maiores e/ou reconhecer cadeias mais longas.

Dessa forma, a maneira estrita pela qual os Autômatos Finitos Não-Determinísticos fazem o reconhecimento de cadeias em determinados problemas pode se tornar inviável devido ao custo computacional envolvido no processamento do autômato.

Uma forma de contornar esse custo computacional é através da utilização dos *Autômatos Finitos Nebulosos*. Um Autômato Finito Nebuloso é um modelo derivado do autômato finito convencional, no qual alguns conjuntos clássicos são convertidos em conjuntos nebulosos, dando ao AFN capacidade de fazer um processamento menos estrito, representando através dos valores de pertinência (e não através de estados adicionais, como no modelo implementado via AFND) os erros encontrados no processamento da cadeia.

Segue abaixo então a definição formal do AFN:

**Definição 4** *Um Autômato Finito Nebuloso é uma quíntupla* $\tilde{M} = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$*, onde:*

$Q$ *é o conjunto (clássico) finito de estados,*
$\Sigma$ *é o conjunto de símbolos (alfabeto) da máquina,*
$\mu$ *é a relação de transição nebulosa* $Q \times \{\Sigma \cup \varepsilon\} \times Q \rightarrow [0, 1]$,
$\tilde{S} \subseteq Q \times [0, 1]$ *é o conjunto nebuloso de estados iniciais,*
$\tilde{F} \subseteq Q \times [0, 1]$ *é o conjunto nebuloso de estados finais.*

Na realidade, existem várias maneiras diferentes de fazer a definição da relação de transição nebulosa do Autômato Finito Nebuloso: cada uma determina capacidades diferentes para o autômato como não-determinismo, transições vazias, etc. O modelo de AFN apresentado por esse trabalho é um caso genérico de diversos outros modelos semelhantes propostos anteriormente, como os modelos presentes no trabalho de Mateescu et al. (1995) e de Lee (2000).

As transições são processadas da maneira descrita na definição (5):

**Definição 5** *Seja* $F(Q)$ *o conjunto de todos os conjuntos nebulosos possíveis sobre* $Q$*. Dessa forma, a relação* $\mu^* : F(Q) \times \Sigma^* \rightarrow F(Q)$ *é definida como se segue:*

$$\mu^*(\tilde{P}, \omega) = \begin{cases} \mu^{\varepsilon}(\tilde{P}),\ \tilde{P} \in F(Q),\ se\ \omega = \varepsilon; \\ \mu^{\varepsilon}(\hat{\mu}(\mu^*(\tilde{P}, \omega'), a)),\ \omega = \omega' a,\ \omega' \in \Sigma^*,\ a \in \Sigma, \tilde{P} \in F(Q), \\ \qquad\qquad caso\ contrário. \end{cases}$$

A relação $\hat{\mu}$ é responsável por realizar as transições que consomem símbolos, enquanto a relação $\mu^{\varepsilon}$ é responsável por realizar transições vazias. Dessa forma, o processamento de uma cadeia por um autômato finito nebuloso é realizado através da execução alternada das duas funções; de maneira que numa primeira etapa, as transições vazias são "propagadas", e numa segunda etapa, as transições não-vazias são executadas. Esse processo é repetido sucessivamente até que o último símbolo seja consumido; quando isso acontece, uma vez mais as transições vazias são verificadas e o processamento do autômato termina.

Para facilitar a compreensão das relações $\hat{\mu}$ e $\mu^{\varepsilon}$; é útil conhecer qual o comportamento esperado do Autômato Finito Nebuloso. O Autômato Finito Nebuloso processa uma cadeia de símbolos, obtendo os valores de pertinência de

cada transição realizada (tanto as que consomem símbolos quanto as transições vazias) e aplica o operador de norma triangular entre eles.

Uma vez que o Autômato Finito Nebuloso é não-determinístico, é possível a existência de mais de uma transição para um determinado estado. Ao contrário do Autômato Finito Não-Determinístico convencional, o Autômato Finito Nebuloso necessita avaliar todas elas (uma vez que é possível - e esperado - que cada transição apresente um grau de pertinência distinto). Para contabilizar o valor de pertinência de toda seqüência de estados quando existem múltiplas transições concorrentes, o autômato utiliza o operador de conorma triangular para calcular qual o valor de pertinência depois das transições.

Segue, então, a definição da relação $\hat{\mu}$:

**Definição 6** *A relação* $\hat{\mu} : F(Q) \times \Sigma \rightarrow F(Q)$ *é definida como:*

$$\hat{\mu}(\tilde{P}, x) = \{(p, \mu) \mid \mu = \oplus_{p \in Q}(\mu(q, x, p) \otimes \mu_{\tilde{P}}(q)), p \in Q\},\ \textit{onde}\ \tilde{P} \subseteq F(Q),\ x \in \Sigma$$

Os símbolos $\otimes$ e $\oplus$ representam, respectivamente, operadores de norma triangular e operadores de conorma triangular, conforme foi visto na subseção 2.1.2.

A definição da relação $\mu^{\varepsilon}$ é um pouco mais complicada, e por isso foi dividida em etapas intermediárias para facilitar sua compreensão.

**Definição 7** *Seja* $F(Q)$ *todos os possíveis conjuntos nebulosos sobre* $Q$*, e seja a relação* $\mu^{\varepsilon} : F(Q) \rightarrow F(Q)$*. Para calcular* $\mu^{\varepsilon}$*, é necessário fazer algumas definições adicionais.*

*Seja* $\langle q_{k_0} q_{k_1} ... q_{k_l} \rangle$ *uma seqüência de estados de* $Q$ *finita e maior do que zero, de maneira que* $\forall i, j, 0 \leq i, j \leq l, i \neq j$ *então* $q_{k_i} \neq q_{k_j}$*. Em outras palavras, significa que a seqüência de estados* $\langle q_{k_0} q_{k_1} ... q_{k_l} \rangle$ *é acíclica, ou seja, a seqüência não percorre duas vezes o mesmo estado.*

*Seja* $\tilde{E}(q)$ *um conjunto nebuloso de estados tal que* $\tilde{E}(q) \in F(Q)$*. Esse conjunto determina representa os estados e seus respectivos valores de pertinência que podem ser atingidos a partir do estado 'q' usando apenas transições vazias, e é definido como:*

$$\tilde{E}(q) = \{(p, \mu) \mid \mu = \oplus_{\langle q_{k_0} ... q_{k_l} \rangle, q_{k_0} = q, q_{k_l} = p}(\otimes_{0 < i \leq l} \mu(q_{k_{i-1}}, \varepsilon, q_{k_i})), p \in Q\}$$

*Dessa forma, a relação $\mu^{\varepsilon}$ fica:*

$$\mu^{\varepsilon}(\tilde{P}) = \{(p,\mu) \mid \mu = \mu_{\tilde{P}}(p) \oplus (\oplus_{q \in Q}(\mu_{\tilde{E}(q)}(p) \otimes \mu_{\tilde{P}(q)})), p \in Q\},\ \textit{onde}\ \tilde{P} \subseteq F(Q)$$

Os símbolos $\otimes$ e $\oplus$ representam, respectivamente, operadores de norma triangular e operadores de conorma triangular, assim como na relação $\hat{\mu}$, exceto por um pequeno detalhe: o operador de conorma triangular necessariamente deve ser a *união nebulosa (máximo).* Isso se deve ao fato de que $\mid \varepsilon \mid = 0$ implica que $\varepsilon \equiv \varepsilon\varepsilon \equiv \varepsilon\varepsilon\varepsilon \equiv ....$ Dessa forma, a conorma triangular deve ser escolhida de maneira que $\mu^{\varepsilon}(\tilde{P}) = \mu^{\varepsilon}(\mu^{\varepsilon}(\tilde{P})) = ...$, garantindo a propriedade da idempotência para os autômatos nebulosos. A única conorma triangular idempotente, como foi visto na subseção 2.1.2, é a união nebulosa.

Uma vez que o funcionamento das transições nebulosas foi compreendido, é possível descrever como um Autômato Finito Nebuloso aceita ou rejeita uma cadeia, determinando uma linguagem característica.

Estritamente falando, não é correto utilizar os termos "aceitação", "rejeição" ou "linguagem" no contexto de um Autômato Finito Nebuloso. De fato, qualquer cadeia sobre $\Sigma^*$ pode ser "reconhecida" por um AFN, embora seja possível que algumas delas sejam "reconhecidas" com grau de pertinência nulo. Da mesma forma, não existe uma "linguagem" determinada pelo AFN: toda cadeia criada sobre o alfabeto dado faria parte desta linguagem.

Entretanto, para este trabalho, convencionou-se utilizar o termo *linguagem nebulosa*, ou simplesmente *linguagem*, para definir o conjunto nebuloso de cadeias característico de um Autômato Finito Nebuloso e o termo *reconhecimento* para definir o processamento feito pelo AFN para obter o grau de pertinência de uma determinada cadeia. Utilizando os termos dados dessa maneira, fica mais fácil fazer a translação dos conceitos do Autômato Finito Não-Determinístico para o Autômato Finito Nebuloso.

Uma vez que tais considerações iniciais foram feitas, segue abaixo a definição do reconhecimento de cadeias e de linguagens nebulosas características de um AFN:

**Definição 8** *Seja $\omega$ uma cadeia qualquer, de maneira que $\omega \in \Sigma^*$ e $\tilde{M}$ o autômato finito nebuloso que determina o valor de pertinência desta cadeia. Portanto, a linguagem nebulosa $\tilde{L}$ característica de $\tilde{M}$ é definida por:*

$$\tilde{L}(\tilde{M}) = \{(\omega, \mu_{\tilde{M}}(\omega)) \,|\, \omega \in \Sigma^*\}$$

*Por fim, a função $\mu_{\tilde{M}}(\omega)$, responsável por calcular os valores de pertinência para cada cadeia reconhecida pelo Autômato Finito Nebuloso $\tilde{M}$, é dada por:*

$$\mu_{\tilde{M}}(\omega) = \oplus_{q \in Q}(\mu_{\mu^*(\tilde{S},\omega)}(q) \otimes \mu_{\tilde{F}}(q))$$

O exemplo 4 facilita a compreensão do funcionamento do Autômato Finito Nebuloso, demonstrando o funcionamento passo-a-passo do autômato para a aplicação de reconhecimento aproximado de cadeias.

## 4.2 AFN Aplicados no Reconhecimento Aproximado de Cadeias

Uma vez que as definições e o funcionamento do Autômato Finito Nebuloso foi determinado, é possível definir a arquitetura utilizada no reconhecimento aproximado de cadeias.

Em primeiro lugar, deve-se determinar os operadores de norma/conorma triangular que serão utilizados. Esse trabalho se baseia no uso do operador de norma *produto algébrico (multiplicação)* e no operador de conorma *união nebulosa (máximo)*. Utilizando esses operadores, o AFN implementa uma função de similaridade do tipo distância de Levenshtein generalizada, de acordo com as métricas vistas na seção 3.1. Entretanto, é importante salientar que outras combinações de operadores podem ser utilizadas, possivelmente apresentando melhores resultados para problemas específicos.

Uma vez feitas tais ressalvas, segue abaixo a definição da arquitetura capaz de fazer o reconhecimento aproximado de cadeias utilizando Autômatos Finitos Nebulosos:

**Definição 9** *Seja $\omega = \{x_1 x_2 ... x_m\}$ uma cadeia de símbolos sobre $\Sigma$, de maneira que $m =\mid \omega \mid$; e seja $\tilde{M}_\omega = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$ o Autômato Finito Nebuloso capaz de fazer o reconhecimento aproximado dessa cadeia. Portanto:*

$Q = \{q_0, q_1, ..., q_m\}$ *é o conjunto clássico de estados do autômato,*
$\Sigma$ *é o conjunto finito de símbolos (alfabeto) do autômato*
$\mu$ *é a relação de transição* $Q \times \{\Sigma \cup \varepsilon\} \times Q \rightarrow [0, 1]$, *descrita separadamente.*
$\tilde{S} = \{q_0/1\}$ *é o conjunto nebuloso de estados iniciais,*
$\tilde{F} = \{q_m/1\}$ *é o conjunto nebuloso de estados finais.*

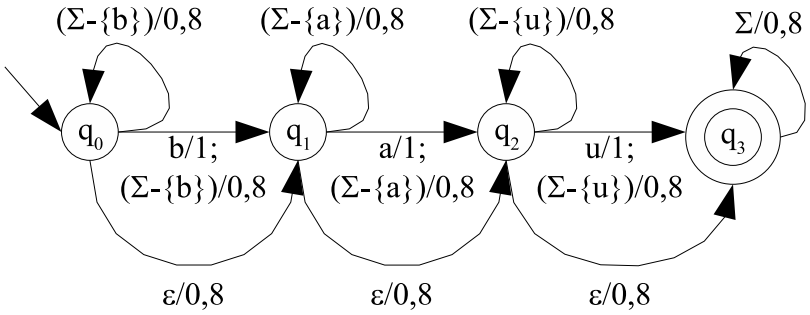


**Figura 4.1:** Autômato Finito Nebuloso $\tilde{M}_4$.

*A função de transição é responsável pela representação das operações de inclusão, exclusão e substituição no autômato nebuloso. Seja* $0 < i \leq |\ \omega\ |$, $j \in (\Sigma - x_i)$ *e* $k \in \Sigma$. *Dessa maneira, a função de transição* $\mu$ *é dada por:*

- $\forall i$: $\mu(q_{i-1}, x_i, q_i) = 1$ *(aceitação);*
- $\forall i, \forall j$: $\mu(q_{i-1}, j, q_i) = \kappa(j, x_i)$ *(substituição);*
- $\forall i$: $\mu(q_{i-1}, \varepsilon, q_i) = \kappa(\varepsilon, x_i)$ *(inserção);*
- $\forall i, \forall j$: $\mu(q_{i-1}, j, q_{i-1}) = \kappa(j, \varepsilon)$ *(exclusão I);*
- $\forall k$: $\mu(q_m, k, q_m) = \kappa(k, \varepsilon)$ *(exclusão II).*

Uma vez que a arquitetura do reconhecimento aproximado de cadeias utilizando o Autômato Finito Nebuloso foi definida, é útil recorrer a um exemplo para ilustrar o funcionamento, permitindo uma melhor compreensão.

**Exemplo 4** *Seja a seqüência de símbolos* $\omega =$ *'bau'. Segue abaixo o Autômato Finito Nebuloso capaz de reconhecê-la, bem como o seu funcionamento utilizando a seqüência de símbolos* $\alpha =$ *'aa' como cadeia de entrada.*

*Seja* $\tilde{M}_4$ *um AFN, tal que* $\tilde{M}_4 = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$. *Assim:*
$Q = \{q_0, q_1, q_2, q_3\}$,
$\Sigma = \{a, b, ..., z\}$,
$\mu$ *é definido como na figura 4.1,*
$\tilde{S} = \{q_0/1\}$,
$\tilde{F} = \{q_3/1\}$.

*Uma vez que o conjunto nebuloso de estados finais possui apenas o estado $q_3$ com grau de pertinência diferente de zero, pode-se considerar que:*

$$\mu_{\tilde{M}_1}(aa) = (\mu_{\mu^*(\tilde{S},aa)}(q_3) \cdot 1) = \mu_{\mu^*(\tilde{S},aa)}(q_3)$$

*Por outro lado, expandindo $\mu^*$:*

$$\mu^*(\tilde{S}, aa) = \mu^{\varepsilon}(\hat{\mu}((\mu^*(\tilde{S}, a)), a)) = \mu^{\varepsilon}(\hat{\mu}((\mu^{\varepsilon}(\hat{\mu}((\mu^*(\tilde{S}, \varepsilon)), a))), a)) = \mu^{\varepsilon}(\hat{\mu}((\mu^{\varepsilon}(\hat{\mu}(\mu^{\varepsilon}(\tilde{S})), a))), a))$$

*A Tabela 4.1 ajuda a compreender cada etapa do processamento do Autômato Finito Nebuloso.*

| | $\mu(q_0)$ | $\mu(q_1)$ | $\mu(q_2)$ | $\mu(q_3)$ |
|---|---|---|---|---|
| $\tilde{Q}_0 \equiv \tilde{S}$ | 1 | 0 | 0 | 0 |
| $\tilde{Q}_1 \equiv (\mu^{\varepsilon}(\tilde{S}))$ | 1 | 0,8 | 0,64 | 0,512 |
| $\tilde{Q}_2 \equiv \hat{\mu}(\mu^{\varepsilon}(\tilde{S}))$ | 0,8 | 0,8 | 0,8 | 0,512 |
| $\tilde{Q}_3 \equiv \mu^{\varepsilon}(\hat{\mu}(\mu^{\varepsilon}(\tilde{S})))$ | 0,8 | 0,8 | 0,8 | 0,64 |
| $\tilde{Q}_4 \equiv \hat{\mu}(\mu^{\varepsilon}(\hat{\mu}(\mu^{\varepsilon}(\tilde{S}))))$ | 0,64 | 0,64 | 0,8 | 0,64 |
| $\tilde{Q}_5 \equiv \mu^{\varepsilon}(\hat{\mu}(\mu^{\varepsilon}(\hat{\mu}(\mu^{\varepsilon}(\tilde{S})))))$ | 0,64 | 0,64 | 0,8 | **0,64** |

**Tabela 4.1:** Processamento da cadeia $\alpha =$'$aa$' pelo AFN $\tilde{M}_4$ passo-a-passo

*Dessa forma: $\mu_{\tilde{M}_4}(aa) = \kappa(a, b) \cdot 1 \cdot \kappa(\varepsilon, u) = 0,64$. Isso indica que a quantidade mínima de operações necessária para transformar $\omega$ em $\alpha$ é uma substituição e uma inclusão.*

# 5 Resultados

## 5.1 Custo Computacional

Uma das principais limitações na implementação de modelos é o seu custo computacional. Teoricamente, computadores tem uma capacidade de processamento infinita e uma memória também infinita. Entretanto, computadores reais executam algoritmos derivados dos modelos propostos; computadores com processamento e memória finitos.

Sendo assim, a análise do custo computacional nos permite conhecer a viabilidade prática de um modelo e em quais situações sua utilização é vantajosa ou não. Existem pelo menos dois limitantes importantes no cálculo do custo computacional: tempo de processamento e espaço de variáveis.

### 5.1.1 Tempo de Processamento

Para calcular o tempo de processamento do algoritmo que implementa um modelo existe uma metodologia consagrada na ciência da computação. Está fora do escopo deste trabalho mostrar como é feito o cálculo, mas basicamente trata-se de descobrir o pior caso possível e determinar um polinômio $T(n)$ proporcional ao tempo que o algoritmo leva para processar uma quantidade de dados igual a $n$. Feito isso, basta isolar o termo de maior grau do polinômio de seu coeficiente e dos demais termos, e a partir dai obter a taxa de crescimento da função representada por aquele algoritmo.

Por exemplo, supondo que um determinado algoritmo tenha tempo de processamento proporcional a $T(n) = 2n^2 + 4n + 3$. A taxa de crescimento da função representada por esse algoritmo é proporcional a $O(n^2)$. Maiores informações sobre o cálculo da complexidade computacional podem ser obtidas em Lewis e Papadimitriou (2000).

Para calcular a taxa de crescimento do modelo de Autômato Finito Não-Determinístico empregado no reconhecimento aproximado de cadeias como des-

crito na seção 3.1, basta utilizar as transições para elaborar o polinômio $T(n)$ característico. Observando o modelo, fica evidente que a quantidade máxima de transições acontece quando uma cadeia necessita de $k$ operações de exclusão e $n - k$ operações de aceitação para ser reconhecida pelo autômato. Dessa forma, chega-se a conclusão que $T_{AFND}(n) = n$ e que a taxa de crescimento da função característica é O(n), onde $n = |\alpha|$ e $k$ é a taxa de erro máximo tolerado.

Uma vez calculada a complexidade computacional da solução empregando AFND, é possível calcular também para a solução com Autômato Finito Nebuloso. Para determinar o pior caso é importante lembrar que é possível que uma operação de substituição tenha um custo bem menor do que uma operação de inserção seguida de uma operação de deleção (embora seja improvável que isto aconteça). De maneira que $\mu(a, b) < \mu(a, \varepsilon) \cdot \mu(\varepsilon, b)$, onde $a, b \in \Sigma$ e $a, b \neq \varepsilon$. Nesses casos, o autômato vai optar por realizar duas operações sucessivas, uma de inserção e outra de exclusão, ao invés de realizar uma operação de substituição.

Levando isto em consideração, existem três situações diferentes que precisam ser analisadas para o pior caso possível: $|\alpha| = |\omega|$, $|\alpha| < |\omega|$ e $|\alpha| > |\omega|$. Segue abaixo a analise minuciosamente de cada uma delas.

Na primeira delas, é evidente que o pior caso ocorre quando as duas cadeias são completamente diferentes e ambas são igualadas através de operações de inserção e deleção, sem recorrer a operações de substituição. Nesse caso, o número de transições é igual a $2n$, nesta situação e apenas nesta, equivale a dizer que é igual a $n + |\omega|$, onde $n = |\alpha|$.

Na segunda, o que ocorre é que o pior caso acontece quando as cadeias são completamente diferentes, assim como no caso anterior. Entretanto, só haverão operações de inserção e deleção para os primeiros $|\omega|$ símbolos, os demais símbolos de $\alpha$ serão tratados através de operações simples de deleção. Dessa forma, a quantidade de transições é igual a $2|\omega| + n - |\omega| = n + |\omega|$, onde $n = |\alpha|$.

Por fim, na terceira situação, o pior caso também é quando as cadeias são completamente diferentes. Apesar disso, só haverão operações de inserção e deleção para coincidir as cadeias dos primeiros $|\alpha|$ símbolos. Os demais símbolos esperados pela cadeia $\omega$ serão tratados através de operações simples de inserção. De maneira semelhante ao caso anterior, a quantidade de transições seria igual a $2n + |\omega| - n = n + |\omega|$, onde $n = |\alpha|$.

Dessa forma, é possível determinar que a complexidade computacional dos Autômatos Finitos Nebulosos também é O(n). E concluir que as duas abordagens

possuem uma complexidade computacional semelhante, já que possuem a mesma taxa de crescimento de suas funções características.

### 5.1.2 Espaço de Variáveis

Um outro aspecto importante para a determinação do custo computacional está no espaço de variáveis, ou seja, quanto de memória é utilizada pelo algoritmo que implementa o modelo.

No caso de autômatos finitos, nebulosos ou não, é razoável supor que o espaço de variáveis é diretamente proporcional à quantidade de estados. Sendo assim, é necessário determinar quais variáveis influenciam na quantidade de estados e como ela ocorre.

O modelo apresentado de Autômato Finito Não-Determinístico empregado no reconhecimento aproximado de cadeias possui um número de estados igual a $|Q| = (m + 1) \cdot (k + 1)$, onde $m = |\omega|$ e $k$ é o valor de erro máximo tolerado.

Entretanto, é incomum fazer o processamento direto de um Autômato Finito Não-Determinístico, geralmente ele é transformado em um Autômato Finito Determinístico antes. Para o solucionar o problema proposto, isso criaria um autômato com $|Q| = (k + 1)^m$ estados (NAVARRO, 2001).

Já a solução proposta neste trabalho, empregando Autômatos Finitos Nebulosos necessita de apenas $|Q| = (m + 1)$ estados; ainda que seja necessária memória adicional para a tabela de pertinência das transições nebulosas, no contexto do reconhecimento aproximado de cadeias esta tabela é proporcional ao alfabeto e não à cadeia de entrada.

Dessa forma, foi verificado que o custo computacional dos Autômatos Finitos Nebulosos quanto ao espaço de variáveis no reconhecimento aproximado de cadeias apresenta vantagens em relação à sua contraparte implementada por Autômatos Finitos Não-Determinísticos.

## 5.2 Capacidade Intuitiva e de Expressão

Essa seção tem como objetivo estudar a capacidade de expressão do modelo, bem como o quão intuitiva é a sua utilização. O método escolhido para ilustrar a capacidade de expressão do modelo é demonstrar um problema real modelado através de um AFN.

Como discutido na subseção 3.2.1, embora o DNA seja responsável por codificar as características que definem um ser vivo, ele nada mais é do que um duplo encadeamento de 4 bases nitrogenadas: adenina, citosina, guanina e timina; em algumas situações incomuns a citosina é convertida em uma quinta base, a uracila. Também foi visto que são poucas as aplicações do reconhecimento estrito de cadeias nesse domínio.

Pois bem, usando a definição (9), é simples modelar passo-a-passo um problema de reconhecimento de uma seqüência de DNA:

**Exemplo 5** *A análise filogenética permite determinar a proximidade genética entre dois indivíduos, comparando trechos selecionados do DNA e determinando se os indivíduos pertencem à mesma espécie, espécies próximas ou espécies completamente diferentes. Essa comparação é feita em nível de base nitrogenada.*

*Entretanto, apenas determinar linearmente a diferença entre as duas seqüências não é a melhor solução. Isso porque as bases nos dois encadeamentos que formam uma seqüência de DNA não são livremente distribuídos: sempre que em um dos encadeamentos houver uma adenina, no outro haverá uma timina; e sempre que houver uma citosina, no outro haverá uma guanina. A existência da uracila é irrelevante no contexto do estudo filogenético: o próprio processo de obtenção das seqüências de DNA elimina essa base.*

*Dessa forma, é razoável assumir que encontrar uma cadeia onde uma timina está na posição esperada de uma adenina ou uma citosina ocupa o lugar onde deveria estar uma guanina é um resultado bastante relevante. Levando essas informações em consideração, segue abaixo um AFN capaz de reconhecer a seqüência "AACT":*

*Seja* $\tilde{M}_{AACT}$ *um AFN, tal que* $\tilde{M}_{AACT} = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$. *Assim:*
$Q = \{q_0, q_1, q_2, q_3, q_4\}$,
$\Sigma = \{A, C, G, T\}$,
$\mu$ *é definido como na figura 5.1 e na tabela 5.1*
$\tilde{S} = \{q_0/1\}$,
$\tilde{F} = \{q_4/1\}$.

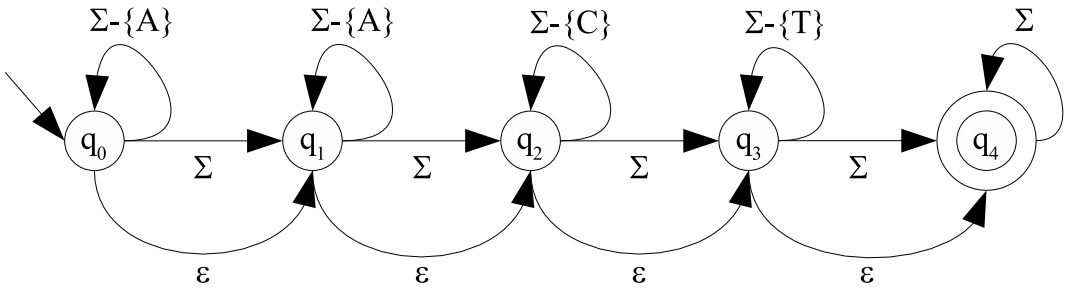


**Figura 5.1:** Autômato Finito Nebuloso $\tilde{M}_{AACT}$, capaz de fazer o reconhecimento aproximado da cadeia "AACT".

| | | | | |
|---|---|---|---|---|
| $\mu(A,A) = 1,0$ | $\mu(A,C) = 0,6$ | $\mu(A,G) = 0,6$ | $\mu(A,T) = 0,9$ | $\mu(A,\varepsilon) = 0,5$ |
| $\mu(C,A) = 0,6$ | $\mu(C,C) = 1,0$ | $\mu(C,G) = 0,9$ | $\mu(C,T) = 0,6$ | $\mu(C,\varepsilon) = 0,5$ |
| $\mu(G,A) = 0,6$ | $\mu(G,C) = 0,9$ | $\mu(G,G) = 1,0$ | $\mu(G,T) = 0,6$ | $\mu(G,\varepsilon) = 0,5$ |
| $\mu(T,A) = 0,9$ | $\mu(T,C) = 0,6$ | $\mu(T,G) = 0,6$ | $\mu(T,T) = 1,0$ | $\mu(T,\varepsilon) = 0,5$ |
| $\mu(\varepsilon,A) = 0,5$ | $\mu(\varepsilon,C) = 0,5$ | $\mu(\varepsilon,G) = 0,5$ | $\mu(\varepsilon,T) = 0,5$ | $\mu(\varepsilon,\varepsilon) = 0,0$ |

**Tabela 5.1:** Valores de pertinência para as transições do AFN $\tilde{M}_{AACT}$.

Em geral, os valores de pertinência presentes na tabela são escolhidos arbitrariamente de acordo com o conhecimento do especialista. As substituições de bases dentro dos pares adenina-timina e citosina-guanina são menos relevantes ao contexto da diferenciação dos indivíduos e, por isso, devem apresentar um grau de pertinência alto. Já as substituições fora dos pares (adenina por citosina ou guanina, por exemplo) devem ter um grau de pertinência baixo por serem mais significativas quanto a análise filogenética. Já as inclusões e exclusões são relativamente incomuns e deslocam toda a seqüência de genes; por outro lado, é improvável que dois indivíduos de espécies distantes possuam uma seqüência de bases nitrogenadas diferentes deslocada apenas por algumas bases a mais ou a menos.

Porém, pode-se determinar os valores de pertinência através de uma fórmula que faz o cálculo a partir do valor de erro máximo tolerado e a quantidade máxima aceitável de erros desse determinado tipo. Segue a definição:

**Definição 10** *Seja $k \in [0,1]$ o valor de erro máximo tolerado, seja $n$ um número natural correspondente à quantidade máxima de erros correspondentes à pertinência e $\mu \in [0,1]$ o valor de pertinência desejado. Eis que:*

$$\sqrt[n]{k} \leq \mu \leq \sqrt[n+1]{k}$$

Usando o exemplo anterior para verificar como ficaria a tabela 5.1:

**Exemplo 6** *Utilizando as informações do exemplo 5 e o cálculo da definição (10), recalcule os valores da tabela de pertinência, permitindo apenas uma inclusão ou exclusão, duas substituições entre bases não-relacionadas ou quatro substituições entre bases relacionadas (use $k \geq 0,5$).*

*Eis que:*

- *Inclusões ou exclusões:* $\sqrt[1]{0,5} \leq \mu \leq \sqrt[2]{0,5} \Rightarrow 0,5 \leq \mu \leq 0,71$
- *Substituições de bases não-relacionadas:* $\sqrt[2]{0,5} \leq \mu \leq \sqrt[3]{0,5} \Rightarrow$ $0,71 \leq \mu \leq 0,79$
- *Substituições de bases relacionadas:* $\sqrt[4]{0,5} \leq \mu \leq \sqrt[5]{0,5} \Rightarrow 0,85 \leq \mu \leq 0,87$

*A partir desse cálculo, os valores de pertinências possíveis para o exemplo 5 usando as limitações impostas nesse exemplo poderiam ser como os da tabela 5.2:*

| | | | | |
|---|---|---|---|---|
| $\mu(A,A) = 1,0$ | $\mu(A,C) = 0,75$ | $\mu(A,G) = 0,75$ | $\mu(A,T) = 0,85$ | $\mu(A,\varepsilon) = 0,55$ |
| $\mu(C,A) = 0,75$ | $\mu(C,C) = 1,0$ | $\mu(C,G) = 0,85$ | $\mu(C,T) = 0,75$ | $\mu(C,\varepsilon) = 0,55$ |
| $\mu(G,A) = 0,75$ | $\mu(G,C) = 0,85$ | $\mu(G,G) = 1,0$ | $\mu(G,T) = 0,75$ | $\mu(G,\varepsilon) = 0,55$ |
| $\mu(T,A) = 0,85$ | $\mu(T,C) = 0,75$ | $\mu(T,G) = 0,75$ | $\mu(T,T) = 1,0$ | $\mu(T,\varepsilon) = 0,55$ |
| $\mu(\varepsilon,A) = 0,55$ | $\mu(\varepsilon,C) = 0,55$ | $\mu(\varepsilon,G) = 0,55$ | $\mu(\varepsilon,T) = 0,55$ | $\mu(\varepsilon,\varepsilon) = 0,0$ |

**Tabela 5.2:** Valores de pertinência para as transições do AFN $\tilde{M}_{AACT}$ segundo o cálculo da definição (10).

Infelizmente, o cálculo apresentado não é capaz de contabilizar vários graus de pertinência diferentes para obter valores de pertinência e/ou valor de erro máximo tolerado adequados para diversas operações distintas (por exemplo, permitir apenas uma exclusão e uma substituição de base não-relacionada). A sugestão é que o cálculo seja utilizado como base e os valores sejam obtidos experimentalmente. Essa é uma limitação atual do modelo proposto neste trabalho.

## 5.3 Flexibilidade do Modelo

Por último, será apresentado um estudo sobre a capacidade do modelo em trabalhar com diferentes métricas e diferentes taxas de erros. Conforme visto na seção 3.1, o uso dos Autômatos Finitos Não-Determinísticos para fazer o reconhecimento aproximado de cadeias permite utilizar qualquer uma das métricas apresentadas com uma taxa arbitrária de erro. Porém, como esses parâmetros são codificados na arquitetura, é impossível alterá-los sem modificar toda a estrutura do autômato.

Por outro lado, uma das vantagens do emprego dos Autômatos Finitos Nebulosos é a possibilidade de ajustar a métrica utilizada e a taxa de erro empregado sem a necessidade de refazer a arquitetura do autômato. No caso da métrica empregada, essa mudança é obtida ajustando os valores de pertinência de acordo com as necessidades. Já o valor de erro máximo tolerado é um parâmetro externo à estrutura e aos valores de pertinência (embora deva ser ajustado após os valores de pertinência).

Devido ao seu enorme número e variedade, está fora do escopo deste trabalho oferecer orientações gerais para empregar todos os tipos de métricas e ajustar os valores de erro máximo tolerado em todas as situações. Entretanto, será demonstrado como empregar Autômatos Finitos Nebulosos no reconhecimento aproximado de cadeias utilizando as demais métricas vistas na seção 3.1: distância de Hamming, distância episódica e distância da sub-cadeia comum mais longa. Através destes exemplos, é possível utilizar as orientações para trabalhar com qualquer outra métrica e/ou ajustar o erro em qualquer situação.

### 5.3.1 Distância de Hamming

A distância de Hamming é uma métrica bastante empregada em aplicações de processamento de sinais. Ela permite apenas operações de substituição (portanto, se as duas cadeias têm diferentes comprimentos, a distância entre elas é infinita), todas as operações tem um valor associado igual a um e a distância é obtida através da soma dos valores individuais das operações.

É bastante simples utilizar a distância de Hamming no reconhecimento aproximado de cadeias com Autômato Finito Nebuloso. Em primeiro lugar, deve-se escolher um valor arbitrário para o grau de pertinência associado às operações de substituição, tal que $\mu(a, b) = p$, onde $a \neq b$, $a, b \neq \varepsilon$ e $0 < p < 1$.

Em seguida, é preciso transformar o valor máximo de erro tolerado da distância de Hamming no grau de pertinência mínimo do Autômato Finito Nebuloso. Isso é feito de maneira simples, calculando-se $K_{AFN} = p^{K_{Hamming}}$.

Realizadas essas operações, basta montar o Autômato Finito Nebuloso, como visto na definição (9) e preencher os valores de pertinência de acordo com as seguinte diretrizes adicionais:

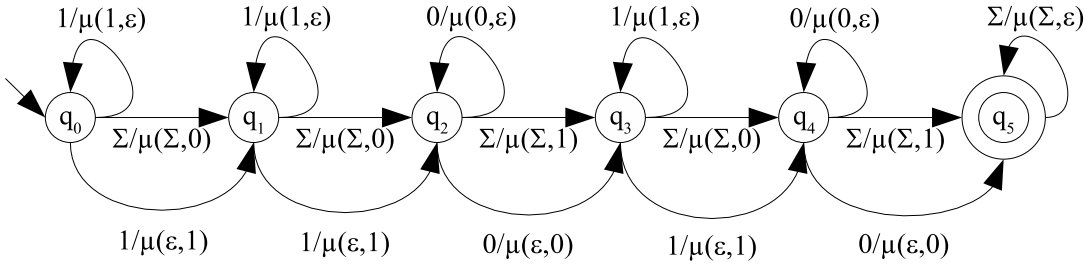


**Figura 5.2:** Autômato Finito Nebuloso $\tilde{M}_{00101}$, que faz o reconhecimento aproximado da cadeia "00101" usando da distância de Hamming.

- $\forall i \in \Sigma$: $\mu(i,i) = 1$ (aceitação);
- $\forall i \in \Sigma$: $\mu(i,\varepsilon) = 0$ (exclusão)
- $\forall i \in \Sigma$: $\mu(\varepsilon,i) = 0$ (inserção);
- $\forall i \in \Sigma$, $\forall j \in \Sigma$, $i \neq j$ : $\mu(i,j) = p$ (substituição).

Por fim, caso seja necessário ou desejável transformar o valor de pertinência encontrado em um valor de erro compatível com a definição da distância de Hamming, basta fazer o seguinte cálculo: $d_{Hamming}(\alpha,\omega) = \log_p(\mu_{\tilde{M}_\omega}(\alpha))$.

Recorrerendo a um exemplo para ilustrar o procedimento apresentado:

**Exemplo 7** *Seja $\omega$ = "00101" o padrão procurado e seja $\alpha$ = "01100" o texto encontrado. Pode-se elaborar um Autômato Finito Nebuloso capaz de fazer o reconhecimento aproximado de cadeias utilizando a distância de Hamming como métrica, com taxa máxima de erro tolerado igual a 3. Simule o funcionamento do autômato e transforme o valor de pertinência obtido em um valor de erro de acordo com a definição da distância de Hamming.*

*Em primeiro lugar, é necessário definir arbitrariamente o valor de $p = 0,5$. Logo em seguida, deve-se transformar o valor de erro para ser utilizado pelo AFN: $K_{AFN} = 0,5^3 \Rightarrow K_{AFN} = 0,125$.*

*Feito isso, é possível definir o autômato $\tilde{M}_{00101}$:*

*Seja $\tilde{M}_{00101}$ um AFN, tal que $\tilde{M}_{00101} = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$. Assim:*
$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$,
$\Sigma = \{0, 1\}$,
*$\mu$ é definido como na figura 5.2 e na tabela 5.3*
$\tilde{S} = \{q_0/1\}$,
$\tilde{F} = \{q_5/1\}$.

| | | |
|---|---|---|
| $\mu(0,0) = 1,0$ | $\mu(0,1) = 0,5$ | $\mu(0,\varepsilon) = 0,0$ |
| $\mu(1,0) = 0,5$ | $\mu(1,1) = 1,0$ | $\mu(1,\varepsilon) = 0,0$ |
| $\mu(\varepsilon,0) = 0,0$ | $\mu(\varepsilon,1) = 0,0$ | $\mu(\varepsilon,\varepsilon) = 0,0$ |

**Tabela 5.3:** Valores de pertinência para as transições do AFN $\tilde{M}_{00101}$.

*Uma vez definido o autômato, o seu funcionamento pode ser simulado e o valor de pertinência associado à cadeia encontrada pode ser obtido:*

$$\mu_{\tilde{M}_{00101}}(01100) = 1 \cdot 0,5 \cdot 1 \cdot 1 \cdot 0,5 = 0,25.$$

*Finalmente, deve-se transformar o valor de pertinência obtido em um valor de erro:*

$$\log_{0,5}\left(\mu_{\tilde{M}_{00101}}(01100)\right) = \log_{0,5}(0,25) = 2.$$

### 5.3.2 Distância Episódica

A distância episódica permite apenas operações de inserção, todas essas operações tem o valor associado de 1 e a distância é obtida através da soma dos valores individuais das operações. Uma vez que permite apenas inserções, o texto encontrado deve ser uma sub-cadeia do padrão procurado, caso contrário, a distância é infinita.

O procedimento que permite utilizar a distância episódica no reconhecimento aproximado de cadeias com AFN é tão simples quanto utilizar a distância de Hamming. Da mesma forma, o primeiro passo é escolher um valor arbitrário para o grau de pertinência associado às operações de inserção, tal que $\mu(\varepsilon, a) = p$, onde $a \neq \varepsilon$ e $0 < p < 1$.

Em seguida, o valor máximo de erro tolerado da distância episódica deve ser convertido para o grau de pertinência mínimo do Autômato Finito Nebuloso. O procedimento é idêntico ao utilizado com a distância de Hamming, portanto $K_{AFN} = p^{K_{Episodica}}$.

Deve-se agora construir o Autômato Finito Nebuloso seguindo as diretrizes da definição (9). Além disso, as seguintes diretrizes devem ser seguidas:

- $\forall i \in \Sigma$: $\mu(i,i) = 1$ (aceitação);
- $\forall i \in \Sigma$: $\mu(i,\varepsilon) = 0$ (exclusão)
- $\forall i \in \Sigma$: $\mu(\varepsilon,i) = p$ (inserção);
- $\forall i \in \Sigma$, $\forall j \in \Sigma$, $i \neq j$ : $\mu(i,j) = 0$ (substituição).

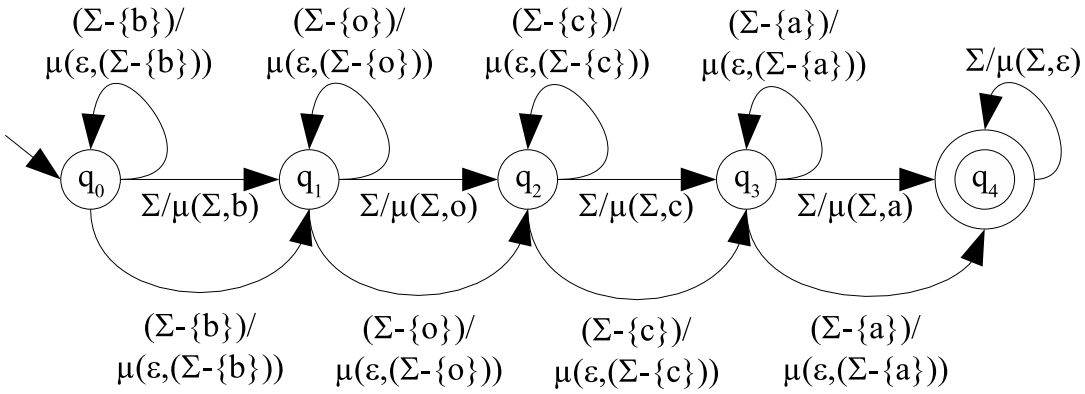


**Figura 5.3:** Autômato Finito Nebuloso $\tilde{M}_{boca}$, que realiza o reconhecimento aproximado da cadeia "boca" usando da distância episódica.

Finalmente, para transformar o valor de pertinência obtido no valor de erro compatível com a distância episódica, basta calcular: $d_{Episodica}(\alpha, \omega) = \log_p(\mu_{\tilde{M}_\omega}(\alpha))$.

O próximo exemplo vai ajudar a entender melhor o procedimento, realizando-o passo-a-passo:

**Exemplo 8** *Sejam $\omega$ = "boca" o padrão procurado e $\alpha$ = "boa" o texto encontrado. Construa um AFN capaz de fazer o reconhecimento aproximado de cadeias através da distância episódica, com taxa máxima de erro tolerado igual à 2. Verifique o funcionamento do autômato para a cadeia dada e calcule o erro encontrado (de acordo com a definição da distância episódica) a partir do valor de pertinência obtido*

*Antes de mais nada, o valor de $p = 0,8$ é definido arbitrariamente e o valor de erro máximo tolerado: $K_{AFN} = 0,8^2 \Rightarrow K_{AFN} = 0,64$ é transformado.*

*Feito isso, é possível definir o autômato $\tilde{M}_{boca}$:*

*Seja $\tilde{M}_{boca}$ um AFN, tal que $\tilde{M}_{boca} = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$. Assim:*
$Q = \{q_0, q_1, q_2, q_3, q_4\}$,
$\Sigma = \{a, b, ..., z\}$,
*$\mu$ é definido como na figura 5.3 e na tabela 5.4*
$\tilde{S} = \{q_0/1\}$,
$\tilde{F} = \{q_4/1\}$.

*Verificando o funcionamento do autômato processando a cadeia dada:*

$$\mu_{\tilde{M}_{boca}}(boa) = 1 \cdot 1 \cdot 0,8 \cdot 1 = 0,8.$$

| | | | | |
|---|---|---|---|---|
| $\mu(a,a) = 1,0$ | $\mu(a,b) = 0,0$ | $\cdots$ | $\mu(a,z) = 0,0$ | $\mu(a,\varepsilon) = 0,0$ |
| $\mu(b,a) = 0,0$ | $\mu(b,b) = 1,0$ | $\cdots$ | $\mu(b,z) = 0,0$ | $\mu(b,\varepsilon) = 0,0$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\mu(z,a) = 0,0$ | $\mu(z,b) = 0,0$ | $\cdots$ | $\mu(z,z) = 1,0$ | $\mu(z,\varepsilon) = 0,0$ |
| $\mu(\varepsilon,a) = 0,8$ | $\mu(\varepsilon,b) = 0,8$ | $\cdots$ | $\mu(\varepsilon,z) = 0,8$ | $\mu(\varepsilon,\varepsilon) = 0,0$ |

**Tabela 5.4:** Valores de pertinência para as transições do AFN $\tilde{M}_{boca}$.

*Convertendo o valor de pertinência obtido em um valor de erro, eis que:*

$$\log_{0,8}\left(\mu_{\tilde{M}_{boca}}(boa)\right) = \log_{0,8}(0,8) = 1.$$

### 5.3.3 Distância da Subseqüência Comum Mais Longa

A distância da subseqüência comum mais longa é capaz de medir a maior seqüência comum de símbolos presente em duas cadeias distintas. Ela utiliza operações de inserção e exclusão, atribuindo a todas as operações valor 1 e a distância é calculada através da soma dos custos individuais das operações.

O procedimento que permite utilizar Autômatos Finitos Nebulosos (AFNs) no reconhecimento aproximado de cadeias utilizando essa métrica é bastante parecido com o procedimento utilizado com a distância de Hamming e a distância episódica.

Deve-se começar calculando o valor de pertinência associado às operações de inserção e exclusão $\mu(\varepsilon,a) = \mu(a,\varepsilon) = p$, onde $a \neq \varepsilon$ e $0 < p < 1$ e transformando o valor máximo de erro tolerado da distância da subseqüência comum mais longa em um grau de pertinência, de maneira que $K_{AFN} = p^{K_{DSCML}}$.

O próximo passo é montar o Autômato Finito Nebuloso de acordo com a definição (9) e com as seguintes diretrizes:

- $\forall i \in \Sigma$: $\mu(i,i) = 1$ (aceitação);
- $\forall i \in \Sigma$: $\mu(i,\varepsilon) = p$ (exclusão)
- $\forall i \in \Sigma$: $\mu(\varepsilon,i) = p$ (inserção);
- $\forall i \in \Sigma$, $\forall j \in \Sigma$, $i \neq j$ : $\mu(i,j) = 0$ (substituição).

Por último, caso seja necessário, valor de pertinência obtido pode ser transformado no valor de erro compatível com a distância da subseqüência comum mais longa, bastando calcular: $d_{DSCML}(\alpha,\omega) = \log_p\left(\mu_{\tilde{M}_\omega}(\alpha)\right)$.

Segue abaixo um exemplo que ilustra a aplicação do procedimento acima descrito:

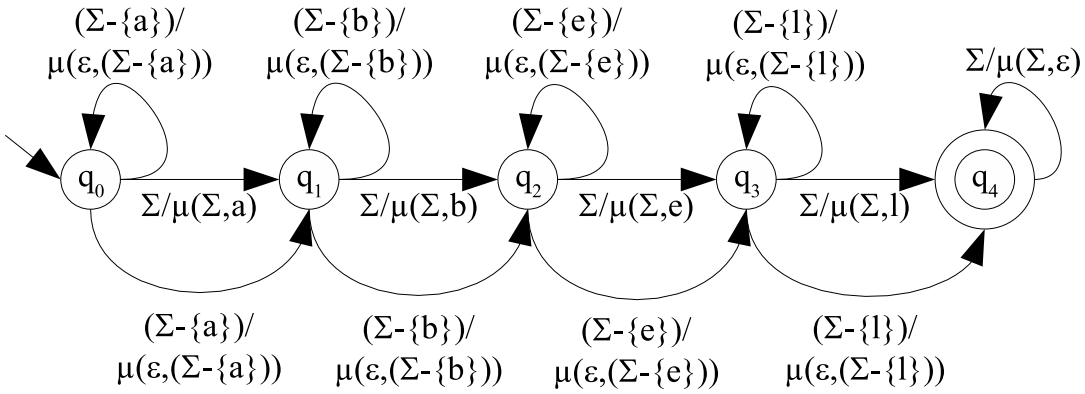


**Figura 5.4:** Autômato Finito Nebuloso $\tilde{M}_{abel}$, que faz o reconhecimento aproximado da cadeia "abel" utilizando a distância da subseqüência comum mais longa.

**Exemplo 9** *Considerando $\omega$ = "abel" como sendo o padrão procurado e $\alpha$ = "belo" o texto encontrado, segue abaixo a descrição de um AFN que faça o reconhecimento aproximado de cadeias utilizando a distância da subseqüência comum mais longa, considerando uma taxa máxima de erro tolerado igual a 1. Segue abaixo também o teste do funcionamento do autômato de acordo com as cadeias apresentadas e o calculo do erro encontrado (de acordo com a definição da distância da subseqüência comum mais longa).*

*Seguindo o procedimento apresentado, em primeiro lugar o valor de pertinência das operações de inserção e exclusão deve ser definido. Portanto: $p = 0,6$. Conseqüentemente, o erro máximo tolerado se torna $K_{AFN} = 0,6^1 \Rightarrow K_{AFN} = 0,6$.*

*Uma vez feitas tais definições e cálculos, é possível elaborar a estrutura do AFN:*

*Seja $\tilde{M}_{abel}$ um AFN, tal que $\tilde{M}_{abel} = (Q, \Sigma, \mu, \tilde{S}, \tilde{F})$. Assim:*
$Q = \{q_0, q_1, q_2, q_3, q_4\}$,
$\Sigma = \{a, b, ..., z\}$,
*$\mu$ é definido como na figura 5.4 e na tabela 5.5*
$\tilde{S} = \{q_0/1\}$,
$\tilde{F} = \{q_4/1\}$.

*Usando a cadeia $\alpha$ = "belo":*

$$\mu_{\tilde{M}_{abel}}(belo) = 0,6 \cdot 1 \cdot 1 \cdot 0,6 = 0,36.$$

| | | | | |
|---|---|---|---|---|
| $\mu(a,a) = 1,0$ | $\mu(a,b) = 0,0$ | $\cdots$ | $\mu(a,z) = 0,0$ | $\mu(a,\varepsilon) = 0,6$ |
| $\mu(b,a) = 0,0$ | $\mu(b,b) = 1,0$ | $\cdots$ | $\mu(b,z) = 0,0$ | $\mu(b,\varepsilon) = 0,6$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $\mu(z,a) = 0,0$ | $\mu(z,b) = 0,0$ | $\cdots$ | $\mu(z,z) = 1,0$ | $\mu(z,\varepsilon) = 0,6$ |
| $\mu(\varepsilon,a) = 0,6$ | $\mu(\varepsilon,b) = 0,6$ | $\cdots$ | $\mu(\varepsilon,z) = 0,6$ | $\mu(\varepsilon,\varepsilon) = 0,0$ |

**Tabela 5.5:** Valores de pertinência para as transições do AFN $\tilde{M}_{abel}$.

*Para obter o valor de erro a partir do valor de pertinência obtido, basta calcular:*

$$\log_{0,6}\left(\mu_{\tilde{M}_{abel}}(belo)\right) = \log_{0,6}\left(0,36\right) = 2.$$

*Verifica-se que nesse exemplo ao contrário dos outros, o erro obtido é maior do que o valor máximo de erro tolerado, logo a cadeia $\alpha =$ "belo" não deve ser reconhecida.*

# 6 Considerações Finais

O aumento exponencial de dados e informações disponíveis na atualidade demandam de novas abordagens e soluções práticas para o rastreamento e tratamento das mesmas. Nesse contexto, o objetivo deste trabalho em contribuir com o repositório de soluções para este tipo de problema foi cumprido: o emprego dos Autômatos Finitos Nebulosos no reconhecimento aproximado de cadeia foi estudado em profundidade, revelando seus pontos fortes e fracos, bem como os pontos que ainda merecem uma melhor atenção.

Este trabalho, no entanto, não se deteve apenas em estudar o uso dos Autômatos Finitos Nebulosos no reconhecimento aproximado de cadeias; algumas contribuições e melhorias foram apresentadas.

Em primeiro lugar, o próprio modelo apresentado, embora fortemente baseado no trabalho de Garitagoitia et al. (2003), possui diferenças sutis e pode ser considerado a generalização de diversos outros modelos encontrados na literatura, pode-se citar (além do modelo presente no trabalho citado anteriormente), os modelos apresentados em Mateescu et al. (1995), Lee (2000), entre outros. Em segundo lugar, o cálculo de pertinência apresentado na seção 5.2 (definição (10)) é uma contribuição totalmente inédita. Por fim, o método de transposição das métricas utilizadas com autômatos finitos clássicos, presente na seção 5.3 também é uma contribuição inteiramente nova.

Desta forma, o trabalho conclui que os Autômatos Finitos Nebulosos são uma alternativa bastante atraente na resolução de problemas envolvendo o reconhecimento aproximado de cadeias. Apresentam um custo computacional igual ou menor do que suas contrapartes clássicas, ao mesmo tempo que possuem melhor capacidade de representação sem perder a facilidade de uso e também são mais flexíveis em relação a definição de métricas.

Tais vantagens, no entanto, não querem dizer que não existam pontos em aberto. Alguns pontos foram abordados neste trabalho, como contribuições inéditas, enquanto outros ainda estão por ser estudados. Como exemplos, pode-se

citar um método mais elaborado para fazer o cálculo dos valores de pertinência do que o apresentado neste trabalho e a otimização do processamento das transições, melhorando o tempo de processamento médio do autômato.

Portanto, este trabalho não pretende ser uma referência completa e definitiva sobre o tema, longe disso, pretende ser apenas um ponto de partida para diversos outros, servindo como base tanto para aqueles que pretendam empregar o modelo apresentado na resolução de problemas como para aqueles que pretendem aprimorar tal modelo.

# Referências

ALBERT, P. The algebra of fuzzy logic. *Fuzzy Sets and Systems*, v. 1, p. 203–230, 1978.

ALTSCHUL, S.; GISH, W.; MILLER, W.; MYERS, G.; LIPMAN, D. Basic local alignment search tool. *J. Mol. Biol.*, n. 215, p. 403–410, 1990.

BAEZA-YATES, R. Some new results on approximate string matching. In: *Workshop on Data Structures*. Dagstuhl, Germany: [s.n.], 1991.

BAEZA-YATES, R. A unified view of string matching algorithms. In: *Proceedings of Theory and Practice of Informatics (SOFSEM '96)*. [S.l.: s.n.], 1996.

BAEZA-YATES, R.; NAVARRO, G. Faster approximate string matching. *Algorithmica*, v. 23, n. 2, p. 127–158, 1999.

DIXON, R.; MARTIN, T. *Automatic Speech and Speaker Recognition*. New York: IEEE Press, 1979.

GARITAGOITIA, J. R.; MENDíVIL, J. R. G. de; ECHANOBE, J.; ASTRAIN, J. J.; FARIñA, F. Deformed fuzzy automata for correcting imperfect strings of fuzzy symbols. *IEEE Transactions on Fuzzy Systems*, v. 11, n. 3, p. 299–310, 2003.

LEE, C. C. Fuzzy logic in control systems: Fuzzy logic controller - part i. *IEEE Transactions on Systems, Man and Cybernetics*, v. 20, n. 2, p. 404–418, March/April 1990.

LEE, C. C. Fuzzy logic in control systems: Fuzzy logic controller - part ii. *IEEE Transactions on Systems, Man and Cybernetics*, v. 20, n. 2, p. 419–435, March/April 1990.

LEE, H. S. Minimizing fuzzy finite automata. *9th IEEE International Conference in Fuzzy*, p. 65–70, 2000.

LEVENSHTEIN, V. Binary codes capable of correcting spurious insertions and deletions of ones. *Prob. Inf. Transmission*, n. 1, p. 8–17, 1965.

LEWIS, H. R.; PAPADIMITRIOU, C. H. *Elementos de Teoria da Computação*. 2. ed. Porto Alegre: Bookman, 2000.

LOWRANCE, R.; WAGNER, R. An extension of the string-to-string correction problem. *J. ACM*, n. 22, p. 177–183, 1975.

MATEESCU, A.; SALOMAA, A.; SALOMAA, K.; YU, S. Lexical analysis with a simple finite-fuzzy-automaton model. *Journal of Computader Science*, v. 1, n. 5, p. 292–311, 1995.

MENGE, K. Statistical metrics. *Procedures of National Academy of Science USA*, v. 28, p. 535–537, 1942.

NAVARRO, G. A guided tour to approximate string matching. *ACM Computing Surveys*, v. 33, n. 1, p. 31–88, March 2001.

NEEDLEMAN, S.; WUNSCH, C. A general method applicable to the search of similarities in the amino acid sequences of two proteins. *J. Mol. Biol.*, n. 48, p. 444–453, 1970.

NESBIT, J. The accuracy of approximate string matching algorithms. *J. Comput.-Based. Instr.*, v. 13, n. 3, p. 80–83, 1986.

NETO, J. J. Adaptive automata for context-sensitive languages. *SIGPLAN NOTICES*, 1994.

NETO, J. J.; BRAVO, C. Adaptative automata - a reduced complexity proposal. In: CHAMPARNAUD, J.; MAUREL, D. (Ed.). [S.l.: s.n.], 2002. v. 2608, p. 158–168.

NGUYEN, H. T.; WALKER, E. A. *A First Course in Fuzzy Logic*. 2. ed. New York: Chapman & Hall/CRC, 2000.

OWOLABI, O.; MCGREGOR, R. Fast approximate string matching algorithms. *Software Practice Exper.*, v. 18, n. 4, p. 387–393, 1988.

SELLERS, P. On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, n. 26, p. 787–793, 1974.

SHANNON, C. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, p. 379–423, 623–656, July-October 1948.

UKKONEN, E. Finding approximate patterns in strings. *J. Algor.*, n. 6, p. 132–137, 1985.

VINTSYUK, T. Speech discrimination by dynamic programming. *Cybernetics*, n. 4, p. 52–58, 1968.

VOLLET, R. *Reconhecimento de Movimentos com Redes Neurais Artificiais*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, São Paulo, 2001.

VOXMAN, W.; GOETSHCHEL, R. A note on the characterization of the max and min operators. *Information Sciences*, v. 30, p. 5–10, 1983.

WAGNER, R.; FISHER, M. The string to string correction problem. *J. ACM*, n. 21, p. 168–178, 1974.

WU, S.; MANBER. Fast text searching allowing errors. *Commun. ACM*, v. 35, n. 10, p. 83–91, 1992.

ZADEH, L. A. Fuzzy sets. *Information and Control*, v. 8, p. 338–353, 1965.

ZADEH, L. A. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, v. 3, p. 28–44, 1973.

ZADEH, L. A. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, v. 4, p. 103–111, May 1996.