



UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

Theo Okagawa Rodrigues

Relatório Técnico - LFA

APUCARANA – PR
2024

Theo Okagawa Rodrigues

**Relatório Técnico - Linguagens Formais Autômatos e
Computabilidade**

Trabalho apresentado à disciplina de
Linguagens Formais Autômatos e
Computabilidade do curso de Bacharelado em
Ciência da Computação.

Professor: Guilherme Henrique de Souza Nakahata

APUCARANA – PR
2024

SUMÁRIO

INTRODUÇÃO	
CAPÍTULO 1: OBJETIVO	
CAPÍTULO 2: RECURSOS E MOTIVAÇÕES.....	
2.1 Motivação	
2.2 Recursos utilizados	
2.2.1 Biblioteca string.h	
2.3 Linguagem de Programação	
2.4 Estrutura de Dados.....	
2.4.1 Matriz de Transição	
2.4.2 Array de Estados Finais.....	
2.4.3 Array para o Alfabeto.....	
2.4.4 Variáveis de Controle	
CAPÍTULO 3: RESULTADOS	
CAPÍTULO 4: CONCLUSÃO	

INTRODUÇÃO

O estudo de Autômatos Finitos Determinísticos (AFD) é uma parte fundamental da teoria da computação, com grande importância no curso de Ciência da Computação e tem aplicações importantes na análise e design de linguagens formais, compiladores e sistemas de processamento de strings. Este trabalho apresenta o desenvolvimento de uma ferramenta computacional em linguagem C, destinada a validar se palavras inseridas pelos usuários são aceitas por um AFD definido conforme regras também fornecidas pelos usuários. Além de proporcionar uma compreensão prática dos conceitos teóricos, este projeto visa servir como um recurso educacional valioso para estudantes e profissionais da área, auxiliando no entendimento e aplicação dos autômatos finitos em diversos contextos computacionais.

CAPÍTULO 1

OBJETIVO

O principal objetivo deste trabalho é o desenvolvimento e implementação de um código fonte feito em linguagem C que verifique se uma sequência de caracteres únicos (chamadas de “palavras”). De modo geral, o código tem o objetivo de receber uma descrição formal de um AFD, ou seja, as características de um conjunto de símbolos e suas respectivas transições. Mediante tais informações, o código deve ser capaz de reconhecer se a palavra digitada pertence ao conjunto de palavras do AFD.

CAPÍTULO 2

MOTIVAÇÃO E RECURSOS UTILIZADOS

2.1 Motivação:

Como citado no capítulo anterior, o objetivo é simular um Autômato Finito Determinístico como uma ferramenta educacional que facilite o entendimento de conceitos de um Autômato Finito Determinístico. A implementação prática de um AFD permite uma ampla compreensão destes conceitos teóricos apresentados na disciplina, ao transformar abstrações matemáticas em soluções computacionais através de um código. Além disso, o desenvolvimento oferece uma oportunidade de exercitar habilidades de programação, estrutura de dados, lógica computacional e conceitos de um AFD.

2.2 Recursos utilizados:

Os recursos utilizados para a realização deste projeto incluem a linguagem de programação C, devido sua eficiência e controle detalhado sobre a memória do computador, sendo crucial tal controle para evitar erros e conflitos como falhas de segmentação.

2.2.1 Biblioteca string.h

Para este projeto, a biblioteca `string.h` foi essencial para o pleno funcionamento do código, sendo utilizado principalmente para comparação de strings utilizando o comando `strcmp` como destacado na figura 1.



Figura 1

A linha de código exemplifica o uso da função `strcmp` para comparar um *input* do usuário para se for "0", finaliza a execução.

2.3 Linguagem de Programação:

Como citado no tópico de recursos utilizados, a escolha da linguagem de programação C foi um elemento vital para a implementação deste projeto devido várias razões fundamentadas na eficiência, gerenciamento direto e detalhado da memória, facilitando a compreensão e manutenção do código.

2.4 Estrutura de dados:

Para a implementação do AFD, foram utilizadas diversas estruturas fundamentais, cada uma desempenhando um papel crucial e específico na modelagem e verificação do autômato.

2.4.1 Matriz de Transição:

A tabela de transição é representada por uma matriz bidimensional de inteiros. Esta armazena transições do autômato digitados pelo usuário, onde as linhas correspondem aos estados e colunas aos símbolos do alfabeto. Cada

entrada na matriz indica o estado de destino para o símbolo. A utilização de uma matriz permite um acesso rápido, fácil e eficiente às transições necessárias durante a verificação da palavra.

2.4.2 Array de Estados Finais:

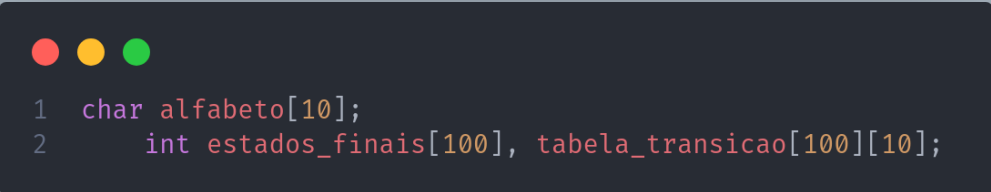
Os estados finais do autômato são armazenados em um array de inteiros. Este array permite uma verificação rápida para determinar se o estado atual, após o processamento da palavra, é um estado de aceitação. A escolha do array é justificada pela necessidade de percorrer e comparar rapidamente os estados finais durante a fase de validação.

2.4.3 Array para o Alfabeto:

Os símbolos do alfabeto são armazenados em um array de caracteres. Este array é usado para mapear os caracteres da palavra de entrada para os índices apropriados na matriz de transição. A estrutura simples do array de caracteres também facilita a verificação rápida do pertencimento de cada símbolo ao alfabeto definido pelo usuário.

2.4.4 Variáveis de Controle:

Variáveis inteiras são usadas para armazenar o estado inicial, a quantidade de estados, a quantidade de estados finais e a quantidade de símbolos do alfabeto. Estas variáveis são fundamentais para dimensionar corretamente as outras estruturas de dados e para controlar o fluxo do algoritmo durante a execução.



```
1 char alfabeto[10];  
2 int estados_finais[100], tabela_transicao[100][10];
```

Figura 2

Acima está um exemplo das variáveis citadas no capítulo de estrutura de dados. A combinação de todas essas estruturas permite o pleno funcionamento do código de uma maneira eficiente e eficaz, assegurando que as palavras sejam verificadas conforme as regras do AFD impostas pelo usuário. A escolha de estruturas simples, mas poderosas, garante um simples entendimento e manutenção do código.

CAPÍTULO 3

RESULTADOS

Mediante os objetivos e a estrutura apresentada, o resultado esperado é o pleno funcionamento de um código que exemplifique e simule como um AFD reconhece ou não palavras de uma determinada linguagem dado sua descrição formal e tabela de transições. Por fim, após a implementação completa e revisada, resultados foram atingidos e testes realizados evidenciam a correta identificação e aceitação/rejeição de palavras de acordo com regras definidas pelo usuário. Além disso, a análise do desempenho computacional revelou um tempo satisfatório de processamento, destacando a viabilidade e eficiência do AFD em diversas aplicações práticas. Estes resultados fornecem uma sólida base de conhecimento de um AFD como sistema de reconhecimento e processamento de linguagens formais. Abaixo, na figura 3 podemos ver o pleno funcionamento do código:

```
LFA_: saida — Konsole
New Tab Split View Copy Paste Find...
> ./saida
Digite o estado inicial: 1
Quantidade de estados: 4
Digite a quantidade de estados finais: 1
Digite os estados finais: 4
Digite a quantidade de símbolos do alfabeto: 4
Símbolos do alfabeto: a
b
c
d
Preencha a tabela de transição (estado x símbolo):
(Q1, a): -1
(Q1, b): -1
(Q1, c): 2
(Q1, d): -1
(Q2, a): 2
(Q2, b): 3
(Q2, c): -1
(Q2, d): 2
(Q3, a): 4
(Q3, b): -1
(Q3, c): -1
(Q3, d): -1
(Q4, a): -1
(Q4, b): 1
(Q4, c): 2
(Q4, d): -1

Tabela de Transição:
Estado | a | b | c | d
q1      | -1 | -1 | 2 | -1
q2      | 2 | 3 | -1 | 2
q3      | 4 | -1 | -1 | -1
q4      | -1 | 1 | 2 | -1

Digite a palavra a ser verificada ('0' pra sair): cadba
A palavra "cadba" é aceita pelo AFD.

Digite a palavra a ser verificada ('0' pra sair): cadadadadadadaba
A palavra "cadadadadadadaba" é aceita pelo AFD.

Digite a palavra a ser verificada ('0' pra sair): cadadadabac
A palavra "cadadadabac" não é aceita pelo AFD.

Digite a palavra a ser verificada ('0' pra sair):
```

Figura 3

CAPÍTULO 4

CONCLUSÃO

Em conclusão, a implementação e simulação do Autômato Finito Determinístico (AFD) representou um passo significativo no entendimento e aplicação de teorias fundamentais da computação. Os resultados obtidos validam a eficácia do modelo na análise de linguagens regulares. A capacidade do AFD em processar cadeias de entrada de forma eficiente reforça sua relevância como uma ferramenta robusta e versátil para resolver uma variedade de problemas computacionais.

Podemos evidenciar que a aplicação prática é vantajosa e apropriada para o entendimento da disciplina de Linguagens Formais e Autômatos, obtendo-se conhecimentos externos à matéria e, também, auxiliando na fixação do conteúdo discutido em sala de aula.