



UNIVERSIDADE ESTADUAL DO PARANÁ - CAMPUS APUCARANA

NOME DO(S) ALUNO(S)
Gabriel Ricetto Da Rocha

AUTÔMATOS FINITOS DETERMINÍSTICOS



APUCARANA – PR
2024

Gabriel Ricetto Da Rocha

AUTÔMATOS FINITOS DETERMINÍSTICOS

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

Professor: Guilherme Nakahata

APUCARANA – PR 2024

SUMÁRIO

INTRODUÇÃO	3
CAPÍTULO 1: OBJETIVO	4
CAPÍTULO 2: MOTIVAÇÕES E RECURSOS UTILIZADOS	6
2.1 Estrutura de dados	7
2.2 Linguagens da programação	7
2.3 Bibliotecas	7
CAPÍTULO 3: RESULTADOS	8
3.1 Testes na prática	8
CONCLUSÃO	10
REFERÊNCIAS	11

INTRODUÇÃO

Os Autômatos Finitos Determinísticos (AFDs) são uma ferramenta essencial na Ciência da Computação e Engenharia, ajudando-nos a entender e modelar sistemas que operam de maneira sequencial e discreta. Eles são especialmente úteis para tarefas como reconhecimento de padrões e análise léxica, fundamentais em muitas áreas, incluindo a criação de compiladores.

Neste relatório, exploraremos os AFDs de maneira detalhada e acessível. Começaremos com os conceitos básicos, explicando o que são, como funcionam e por que são importantes. Discutiremos os componentes principais de um AFD, como estados, transições e a função de transição, e veremos como tudo isso se encaixa para formar um modelo eficaz.

Nosso objetivo com este relatório é fornecer uma introdução compreensível e útil aos Autômatos Finitos Determinísticos, auxiliando a compreenderem melhor essa ferramenta essencial e suas aplicações práticas no mundo da computação.

CAPÍTULO 1

OBJETIVO

O objetivo do estudo dos Autômatos Finitos Determinísticos (AFDs) é proporcionar uma compreensão sólida e aplicável dessa ferramenta essencial na teoria da computação e linguagens formais. Os AFDs são utilizados para modelar e analisar sistemas computacionais que operam com entradas discretas e sequenciais. Eles são fundamentais na construção de analisadores léxicos, que são componentes essenciais na criação de compiladores, e na validação de cadeias de caracteres conforme determinadas regras de linguagem.

Especificamente, o estudo dos AFDs busca:

Modelagem de Sistemas: Facilitar a representação de sistemas e processos que podem ser descritos por um conjunto finito de estados e transições. Isso inclui aplicações em linguagens formais, reconhecimento de padrões e análise sintática.

Validação e Reconhecimento: Permitir a verificação de sequências de entrada contra padrões predefinidos, garantindo que apenas as sequências válidas sejam aceitas. Este aspecto é crucial para a análise léxica em compiladores e outras aplicações de processamento de texto.

Simplificação de Problemas: Auxiliar na simplificação e resolução de problemas complexos através da decomposição em estados e transições, facilitando a análise e a implementação de soluções algorítmicas.

Educação e Formação: Servir como uma ferramenta pedagógica para ensinar conceitos fundamentais da teoria da computação, auxiliando os estudantes a compreenderem a estrutura e o funcionamento dos sistemas computacionais básicos.

Desenvolvimento de Software: Contribuir para o desenvolvimento de software mais robusto e eficiente, fornecendo uma base teórica para a implementação de algoritmos que dependem da manipulação de estados e transições.

O estudo e a aplicação dos AFDs são essenciais para o avanço da ciência da computação, permitindo o desenvolvimento de tecnologias mais avançadas e a solução de problemas complexos de maneira eficiente e precisa.

CAPÍTULO 2

MOTIVAÇÕES E RECURSOS UTILIZADOS

Teoria da Computação: AFDs são fundamentais para a teoria da computação, proporcionando uma base para entender conceitos mais complexos como gramáticas, linguagens formais e máquinas de Turing.

Desenvolvimento de Compiladores: A análise léxica, uma fase essencial no desenvolvimento de compiladores, utiliza AFDs para reconhecer padrões em cadeias de caracteres, facilitando a tradução de código-fonte para código-máquina.

Reconhecimento de Padrões: AFDs são amplamente utilizados em aplicações de reconhecimento de padrões e processamento de linguagens naturais, onde a validação de sequências de entrada é necessária.

Simplicidade e Eficiência: AFDs oferecem uma maneira simples e eficiente de modelar sistemas com um número finito de estados, permitindo a implementação de algoritmos que são rápidos e utilizam poucos recursos computacionais.

Formação Acadêmica: O estudo de AFDs é essencial na formação de estudantes de ciência da computação, pois ajuda a desenvolver habilidades analíticas e a compreensão de conceitos fundamentais da área.

3.2 Recursos Utilizados

Para a implementação e estudo de AFDs, diversos recursos são utilizados, incluindo:

Estrutura de Dados

Matrizes de Transição: Uma matriz de transição é utilizada para armazenar as transições de estado, onde as linhas representam estados e as colunas representam símbolos do alfabeto.

Vetores: Vetores são utilizados para armazenar os estados finais e os símbolos do alfabeto.

Variáveis Inteiras e Ponteiros: São utilizados para gerenciar os estados atuais, estados finais e índices de símbolos.

Linguagem de Programação

C: A linguagem C é escolhida por sua eficiência e controle de baixo nível sobre os recursos de memória e por ser a linguagem com maior familiaridade.

Bibliotecas

stdio.h: utilizada para funções de entrada e saída, como printf e scanf, essenciais para a interação com o usuário e a exibição de resultados.

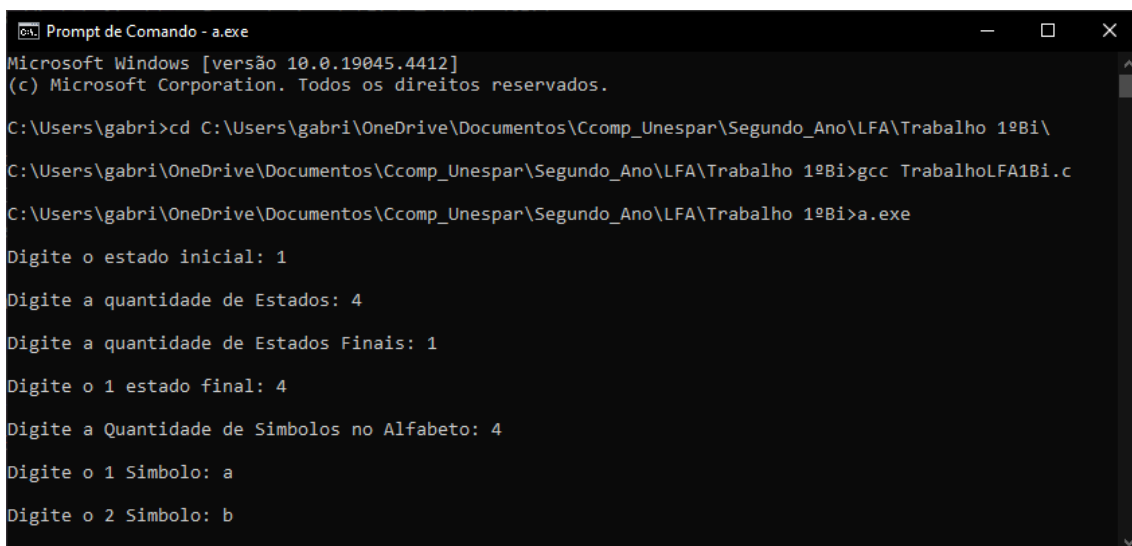
stdlib.h: Utilizada para a alocação dinâmica de memória, como malloc e free, que permitem criar estruturas de dados flexíveis para armazenar estados e transições.

string.h: Utilizada para manipulação de cadeias de caracteres, incluindo funções como strlen e strcmp, necessárias para o processamento das palavras de entrada no autômato.

CAPÍTULO 3

RESULTADOS

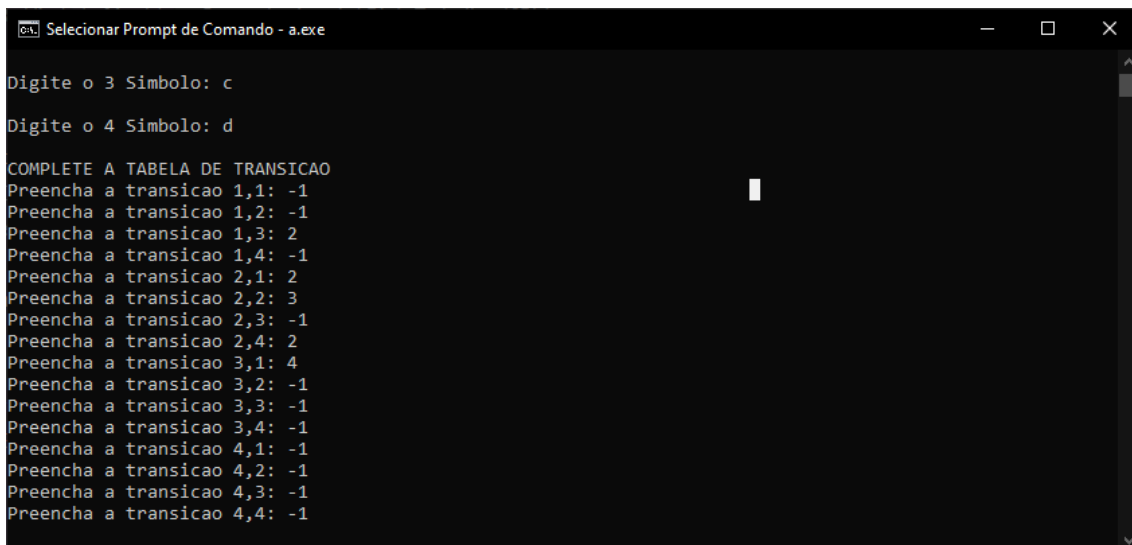
Mediante os objetivos apresentados, o resultado esperado seria o pleno funcionamento de um código que exemplifique como um AFD reconhece ou não palavras de determinada linguagem dado a sua descrição formal e tabela de transições. Portanto, após a implementação completa e revisada, alcançamos os resultados desejados, Como mostra as imagens 1,2,3.



```
Prompt de Comando - a.exe
Microsoft Windows [versão 10.0.19045.4412]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\gabri>cd C:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar\Segundo_Ano\LFA\Trabalho 1ºBi\
C:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar\Segundo_Ano\LFA\Trabalho 1ºBi>gcc TrabalhoLFA1Bi.c
C:\Users\gabri\OneDrive\Documentos\Ccomp_Unespar\Segundo_Ano\LFA\Trabalho 1ºBi>a.exe

Digite o estado inicial: 1
Digite a quantidade de Estados: 4
Digite a quantidade de Estados Finais: 1
Digite o 1 estado final: 4
Digite a Quantidade de Simbolos no Alfabeto: 4
Digite o 1 Simbolo: a
Digite o 2 Simbolo: b
```



```
Selecionar Prompt de Comando - a.exe

Digite o 3 Simbolo: c
Digite o 4 Simbolo: d

COMPLETE A TABELA DE TRANSICAO
Preencha a transicao 1,1: -1
Preencha a transicao 1,2: -1
Preencha a transicao 1,3: 2
Preencha a transicao 1,4: -1
Preencha a transicao 2,1: 2
Preencha a transicao 2,2: 3
Preencha a transicao 2,3: -1
Preencha a transicao 2,4: 2
Preencha a transicao 3,1: 4
Preencha a transicao 3,2: -1
Preencha a transicao 3,3: -1
Preencha a transicao 3,4: -1
Preencha a transicao 4,1: -1
Preencha a transicao 4,2: -1
Preencha a transicao 4,3: -1
Preencha a transicao 4,4: -1
```

```
Selecionar Prompt de Comando - a.exe

Preencha a transicao 2,1: 2
Preencha a transicao 2,2: 3
Preencha a transicao 2,3: -1
Preencha a transicao 2,4: 2
Preencha a transicao 3,1: 4
Preencha a transicao 3,2: -1
Preencha a transicao 3,3: -1
Preencha a transicao 3,4: -1
Preencha a transicao 4,1: -1
Preencha a transicao 4,2: -1
Preencha a transicao 4,3: -1
Preencha a transicao 4,4: -1

==== TABELA DE TRANSICAO ====

Estado | a | b | c | d
Q0     | -1 | -1 | 2 | -1
Q1     | 2 | 3 | -1 | 2
Q2     | 4 | -1 | -1 | -1
Q3     | -1 | -1 | -1 | -1

Digite a palavra a ser testada: cadadaba
A palavra cadadaba foi aceita
```

CONCLUSÃO

Com o desenvolvimento e aprimoramento do código do Autômato Finito Determinístico (AFD), chegamos a uma solução eficaz e confiável para validar cadeias de caracteres conforme uma linguagem formal específica. O código, agora completo e revisado, permite receber informações detalhadas sobre o alfabeto, os estados, e as transições do autômato, determinando de forma precisa se uma palavra é aceita ou não.

A implementação foi pensada para ser intuitiva e funcional, facilitando o entendimento dos conceitos fundamentais de teoria da computação e linguagens formais. Este projeto não apenas atendeu aos objetivos iniciais, mas também se mostrou uma ferramenta valiosa tanto para o aprendizado quanto para aplicações práticas, oferecendo uma base sólida para futuras extensões e aprimoramentos. Ao final, temos um sistema interativo e eficiente, pronto para ser utilizado em diversas situações que requerem a verificação de linguagens regulares.

REFERÊNCIAS

[1] Desenvolvimento de um aplicativo para auxiliar o estudo de autômatos finitos determinísticos, De Daniel Figueiredo, Acessado em 15/05/2024
<https://repositorio.pucgoias.edu.br/jspui/handle/123456789/4348>

[2] Wikipedia. Acessado em 16/05/2024
https://pt.wikipedia.org/wiki/Aut%C3%B4mato_finito_determin%C3%ADstico

[3] Autômatos Finitos Determinísticos AFD. Acessado em 16/05/2024
<https://www.kits.ai/pt/glossary/deterministic-finite-automata-dfa>