

**UNIVERSIDADE ESTADUAL DO PARANÁ**

**-**

***CAMPUS***

**APUCARANA**

**João Vitor Pastori Leme Batista**

**RELATÓRIO TÉCNICO**

**-**

**LFA**

APUCARANA

–

PR

2024

**João Vitor Pastori Leme Batista**

**RELATÓRIO TÉCNICO – LFA**

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** ............................................................................................... 03

**CAPÍTULO 1: OBJETIVOS**  ........................................................................... 04

**CAPÍTULO 2: MOTIVAÇÃO E RECURSOS UTILIZADOS**  ........................... 05

**2.1 Motivação** ................................................................................................. 05

**2.1 Estrutura de Dados** ................................................................................. 06

**2.2 Linguagem de programação e demais informações** ............................ 07

**CAPÍTULO 3: RESULTADOS** ........................................................................ 08

[**CONCLUSÃO** 13](#_Toc9618)

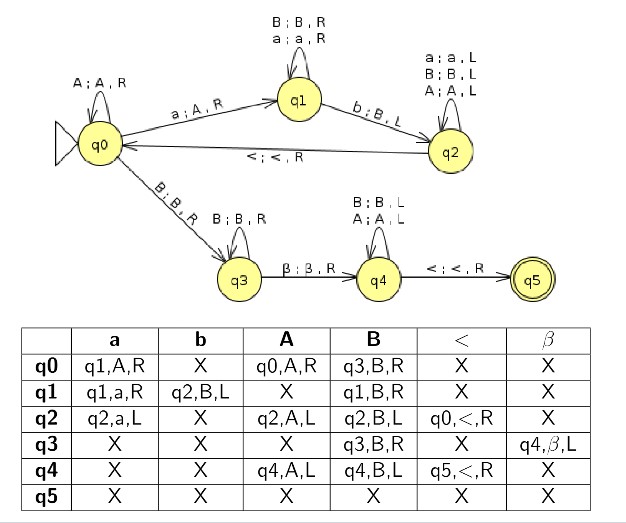
**INTRODUÇÃO**

Diferente de outras disciplinas do curso de Ciência da Computação, Linguagens Formais e Autômatos se destacam por seu caráter prático, permitindo que os alunos realizem atividades concretas, como a construção de autômatos finitos, autômatos a pilha e máquinas de Turing. Esta última é o foco principal da nossa análise e desenvolvimento nesta atividade. As aulas demonstram que há conexões reais entre a máquina de Turing e a arquitetura dos computadores modernos, com o processador funcionando como o cabeçote da fita, a fita representando a memória e os padrões de bits correspondendo ao alfabeto da fita. Se um problema não pode ser resolvido por uma máquina de Turing, ele não pode ser solucionado por nenhum sistema algorítmico.

Isso cria uma oportunidade interessante para a aplicação prática das máquinas de Turing, especialmente em relação ao desenvolvimento de algoritmos. O uso da programação em contextos relacionados à disciplina de LFA se revela eficaz para conectar com outras áreas do conhecimento e promover uma compreensão mais profunda e prática. Um exemplo é uma atividade interdisciplinar realizada em uma instituição federal de ensino superior, onde o desenvolvimento de um programa que simulava o funcionamento da máquina de Turing obteve um resultado positivo. Apesar das dificuldades encontradas por cerca de 40% dos alunos do primeiro semestre para aprender uma linguagem de programação e implementar o código, aproximadamente 88% não consideraram o projeto desestimulante.

De modo geral, aplicar a disciplina de Máquinas de Turing de maneira prática pode ser altamente benéfico para o aprendizado interdisciplinar e contribuir para uma melhor assimilação do conteúdo, devido ao envolvimento mais profundo na criação e funcionamento do código.

## CAPÍTULO 1 OBJETIVOS

O principal objetivo do código desenvolvido é simular o funcionamento prático de uma máquina de Turing, mas por meio de um programa. Em termos gerais, o código fonte visa processar uma descrição formal que define um conjunto de estados representando uma máquina de Turing. Com base nessa descrição, o programa pode avaliar se uma palavra pertence ou não à linguagem especificada, de acordo com as transições definidas. Além disso, a máquina de Turing simulada deve ser capaz de modificar a fita, permitindo que as transições sejam verificáveis. Um exemplo de uma máquina de Turing que o algoritmo deve ser capaz de reconhecer, e verificar a aceitação de palavras, está ilustrado na Figura1.

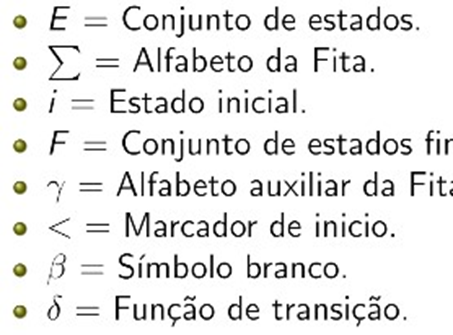
*Figura 1*

## CAPÍTULO 2 MOTIVAÇÃO E RECURSOS UTILIZADOS

Baseando-se no exposto anteriormente, devemos explicitar os motivos para a realização do trabalho, ou seja, o objetivo final e os recursos utilizados para que isso seja cumprido.

**2.1 Motivação**

Como mencionado no capítulo sobre os objetivos do projeto, a motivação principal é desenvolver um código funcional que permita demonstrar o funcionamento completo de uma máquina de Turing. O objetivo é receber informações necessárias para determinar se palavras específicas pertencem à linguagem definida e, em casos onde a máquina de Turing seja uma transdutora, modificar a fita conforme necessário. Para isso, é essencial obter dados como a descrição formal e a tabela de transições, e então criar uma lógica de programação eficaz para alcançar os resultados esperados. A Figura 2 ilustra como a descrição formal deve ser recebida e exibida no formato requerido pelo modelo.



*Figura 2*

Portanto, é crucial, considerando os objetivos e motivações, detalhar os dados relevantes relacionados à Estrutura de Dados, Linguagem de Programação e outras questões pertinentes à implementação do código.

**2.2 Estrutura de Dados**

A classe `Transicao` define as transições da Máquina de Turing. Seus atributos são:

- int eo: Estado de origem da transição.

- char sl: Símbolo lido da fita.

- int ed: Estado de destino.

- char se: Símbolo a ser escrito na fita.

- char d: Direção de movimento da cabeça de leitura/escrita ('D' para direita, 'E' para esquerda).

O construtor padrão inicializa o estado de origem `eo` com `-1`, indicando que a transição é inválida até ser configurada.

Classe Maq

A classe Maq contém os dados e métodos para a operação da Máquina de Turing. Aqui estão os principais componentes:

1. Constantes

- static final int MAX\_ESTADOS = 100`\*\*: Define o número máximo de estados.

- static final int MAX\_SIMBOLOS = 100`\*\*: Define o número máximo de símbolos no alfabeto.

- static final int TAMANHO\_MAXIMO\_FITA = 1000`\*\*: Define o tamanho máximo da fita.

2. Arrays Estáticos

- static Tran[][] transi: Matriz que armazena a tabela de transições. Cada posição `[i][j]` representa uma transição para o estado `i` com o símbolo `j` do alfabeto.

- static int[] ea: Array que contém os estados de aceitação.

- static char[] alfa: Array que armazena os símbolos do alfabeto.

- static char[] fita: Array que representa a fita de entrada da Máquina de Turing.

3. Métodos

- static void iniciar(int ne, int ns): Inicializa a tabela de transições com transições inválidas. Recebe como parâmetros o número de estados (`ne`) e o número de símbolos do alfabeto (`ns`).

- static int indice(char s, int ns): Retorna o índice de um símbolo no alfabeto. Se o símbolo não estiver no alfabeto, retorna `-1`.

- static void tabela(Scanner sc, int ns): Lê as transições definidas pelo usuário e as armazena na matriz `transi`. Solicita ao usuário o estado de origem, símbolo lido, estado de destino, símbolo a ser escrito e direção de movimento.

- static boolean ehFinal(int e, int nea): Verifica se um estado é um estado de aceitação. Compara o estado atual com o array `ea` que contém os estados de aceitação.

- static void exec(Scanner sc, int tf, int ns, int nea): Executa a Máquina de Turing com base na fita de entrada e nas transições definidas. Inicializa a fita com a palavra de entrada, movimenta a cabeça de leitura/escrita e atualiza o estado conforme as transições. Verifica se a palavra é aceita ou rejeitada.

- public static void main(String[] args): Método principal que coordena a entrada do usuário, inicializa a Máquina de Turing e executa a simulação. Solicita informações sobre o número de estados, símbolos do alfabeto, tamanho da fita, estados de aceitação, e configura a tabela de transições.

Funcionamento da Máquina de Turing

1. Inicialização: A tabela de transições é inicializada com transições inválidas utilizando o método `iniciar`. Isso garante que todas as transições sejam configuradas antes de serem usadas.

2. Definição de Alfabeto e Estados de Aceitação: O usuário define o alfabeto e os estados de aceitação. Estes são armazenados nos arrays `alfa` e `ea`.

3. Configuração das Transições: O método `tabela` permite ao usuário definir as transições da máquina, que são armazenadas na matriz `transi`. Cada transição especifica a mudança de estado e a modificação da fita com base no símbolo lido.

4. Execução da Máquina: O método `exec` simula a operação da Máquina de Turing. A fita é preenchida com a palavra de entrada e a cabeça de leitura/escrita começa no meio da fita. O método executa as transições definidas até que a palavra seja aceita (se o estado final é um estado de aceitação) ou rejeitada (se uma transição inválida ocorre).

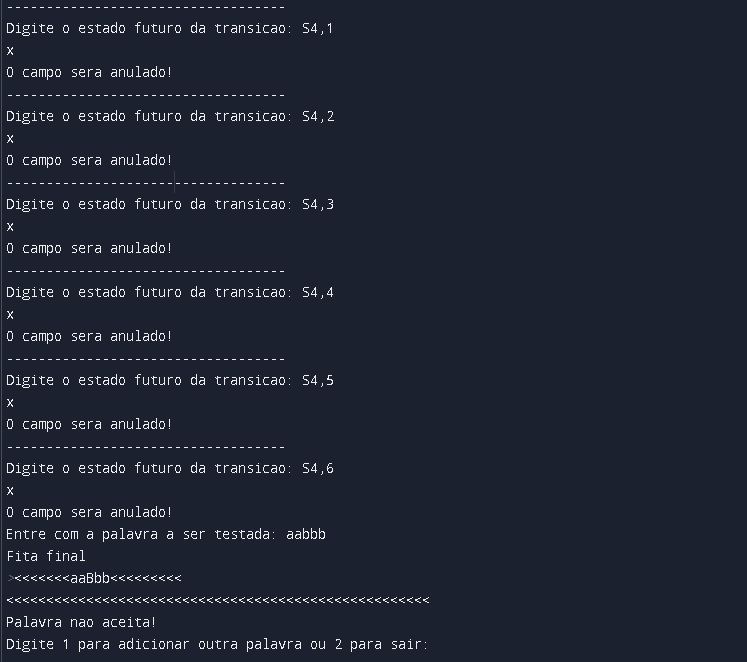
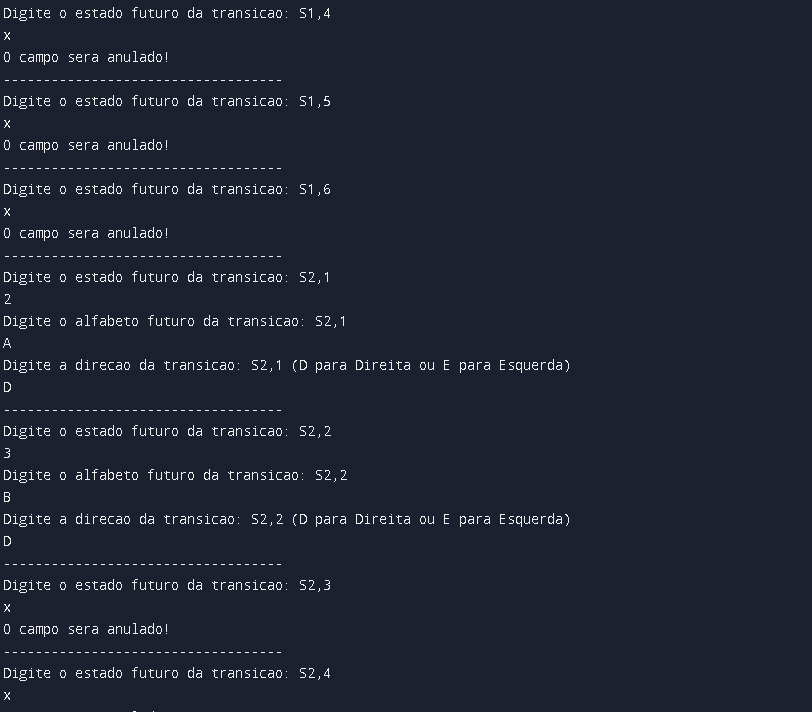
5. Interação com o Usuário: O método `main` gerencia a entrada do usuário, coleta as informações necessárias e executa a Máquina de Turing conforme as definições fornecidas.

**2.3 Linguagem de Programação e demais informações**

A escolha da linguagem para implementar o código foi Java, que é o foco principal do segundo ano do curso de formação para desenvolver o ciclo de instruções. Java, criada pela Sun Microsystems, é uma linguagem de alto nível orientada a objetos e amplamente utilizada em diversos setores da programação. No código, essa orientação a objetos é empregada, e a única biblioteca utilizada foi o Scanner, para permitir a entrada de dados pelo usuário através do console.

## CAPÍTULO 3 RESULTADOS

Mediante os objetivos apresentados, o resultado esperado seria o pleno funcionamento de um código que exemplifique como uma máquina de Turing reconhece ou não palavras de uma determinada linguagem dado a sua descrição formal e tabela de transições. Dessarte, com a implementação completa e revisada, os resultados foram atingidos, resultando em uma aplicação interativa e funcional, recebendo a quantidade de letras no alfabeto, as letras, uma letra para representar os estados, a quantidade de estados, a quantidade de estados finais, o estado inicial, os estados finais, marcador de início, símbolo de branco e as transições contendo estado futuro, alfabeto futuro e direção (quando não anulado). Abaixo, podemos ver o funcionamento do exemplo de máquina de Turing



# CONCLUSÃO

Certamente! Aqui está uma versão revisada do texto, sem citar Mazur:

---

Com base no exposto anteriormente, podemos evidenciar que uma abordagem prática, mesmo que para uma disciplina mais aplicada, pode ser vantajosa e apropriada para adquirir conhecimentos além da área específica e também para uma melhor fixação do conteúdo estudado. A atividade interdisciplinar mencionada anteriormente demonstra que, mesmo disciplinas que parecem distantes à primeira vista, podem ter conexões significativas. Um exemplo disso é a interseção entre Linguagens Formais e Automatos (LFA) e Programação Orientada a Objetos (POO), como evidenciado pela implementação em Java usando conceitos de objetos.

De maneira geral, métodos de ensino diferenciados, por mais que possam parecer simples, têm o potencial de criar conexões mais profundas entre os alunos e o conhecimento. Abordagens ativas de ensino diminuem a abstração e aproximam os alunos de uma resolução mais prática dos problemas. Isso ocorre porque os alunos buscam soluções por conta própria e têm a oportunidade de colaborar e ajudar seus colegas, promovendo um aprendizado mais interativo e eficaz. Em resumo, o código apresentado não só cumpriu sua função prática, mas também serviu como um exemplo de um método de aprendizado mais ativo e engajador.