UNIVERSIDADE ESTADUAL DO PARANÁ - *CAMPUS* APUCARANA

Paloma de Castro Leite

RELATÓRIO TÉCNICO - LFA

APUCARANA – PR
2024

Paloma de Castro Leite

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

O estudo de Linguagens Formais e Autômatos é fundamental para a Ciência da Computação, pois fornece as bases para a compreensão e especificação de linguagens, bem como o reconhecimento de padrões. Esta disciplina investiga classificações, estruturas, propriedades e relações entre diferentes tipos de autómatos e linguagens, além de embasar outros aspectos mais teóricos, como decidibilidade, computabilidade e complexidade computacional.

Neste trabalho, iremos analisar especificamente uma Máquina de Turing (MT), que é capaz de reconhecer uma linguagem recursivamente enumerável (como as de programas de computador) e possui dois tipos: reconhecedora (responde sim ou não para uma palavra, se pertence ou não a linguagem) e transdutora (gera uma palavra na própria fita que é a saída da MT). Uma MT é composta pela fita, cabeçote da fita e função de transição, que possibilita fazer leitura e escrita, exibir a posição atual da fita e partir de um símbolo indo para um estado, além de se mover para direita e esquerda respectivamente. As funcionalidades do simulador incluem a leitura da cadeia de entrada, a utilização de uma tabela de transição que define o comportamento da máquina, o processamento da cadeia de entrada de acordo com as regras definidas na tabela de transição, a atualização da fita passo a passo e a movimentação da cabeça de leitura/escrita conforme especificado. Ao final do processamento, o simulador indica se a cadeia foi aceita ou rejeitada, de acordo com os estados finais definidos.

CAPÍTULO 1
OBJETIVOS

A linguagem de programação Java foi escolhida para o desenvolvimento do código, ela é abordada principalmente durante o segundo ano do curso, e utilizá-la para o desenvolvimento do programa permite conhecer mais sobre a linguagem e desenvolver um aprendizado ao longo do trabalho, além de pôr em prática os conhecimentos que adquirimos durante as aulas de LFA, Estrutura de dados e Programação orientada a objeto. Esta linguagem fornece a possibilidade de utilizar orientação a objetos, uma prática de programação que torna possível elaborar um software a partir da geração de objetos que se comunicam entre si, e o uso de classes para representar objetos do mundo real, onde declararmos atributos e métodos, que representam, respectivamente, as características e comportamentos desse objeto.

Dessa maneira, a linguagem de programação Java se tornou ideal para este caso específico, em que o código fonte foi feito para simular uma Máquina de Turing que possa aceitar diversas linguagens, afinal o uso de classes permite mudanças, compartilhamento de dados, melhor organização e estruturação do código, manipulação de diversas partes do código, entre outras possibilidades, tornando assim o código mais propício a futuras melhorias e implementações sem tê-lo que reescrever por inteiro novamente, da mesma maneira que facilita sua visualização e compreensão para quem analisá-lo.

CAPÍTULO 2

MOTIVAÇÃO E RECURSOS UTILIZADOS

Baseando-se no que foi descrito anteriormente, devemos explicitar os motivos para a realização do trabalho e seu objetivo final, além dos recursos utilizados para que o trabalho seja executado de maneira eficiente e concisa, de acordo com os requisitos especificados.

2.1 Motivação

Conforme mencionado no capítulo que trata acerca dos objetivos do trabalho, a motivação seria a execução de um código fonte funcional em que possamos demonstrar e exemplificar o funcionamento de uma Máquina de Turing (MT), em que recebemos as informações necessárias pelas entradas do usuário e assim determinamos se a cadeia é ou não aceita, demonstrando como um MT funciona de maneira prática e visual, a fim de facilitar a aprendizagem e possibilitar ver o passo a passo de seu funcionamento para assim pôr em prática.

Para esse fim, precisamos receber os dados que possibilitem o funcionamento dessa máquina, tais como a descrição formal e a tabela de transições. Na figura a seguir, por exemplo, podemos visualizar como a descrição formal deverá ser recebida:

- E = Conjunto de estados.
- $\sum$ = Alfabeto da Fita.
- i = Estado inicial.
- F = Conjunto de estados finais.
- γ = Alfabeto auxiliar da Fita.
- $<$ = Marcador de inicio.
- β = Símbolo branco.
- δ = Função de transição.

Figura 1 - Descrição formal de uma MT

Portanto, faz-se necessário, com as informações pertinentes acerca dos objetivos e motivações, detalhar os dados mais relevantes à Estrutura de Dados, Linguagem de Programação e demais questões acerca da implementação do código fonte em questão para melhor análise de suas funcionalidades, a fim de obter uma conclusão geral.

2.2 Estrutura de Dados

Este código em Java simula o funcionamento de uma Máquina de Turing, um dispositivo teórico conhecido como *máquina universal*, que foi criada pelo matemático Alan Turing em 1936. Inicialmente, o código foi separado nas classes principais *ControlaMT* e *UsarMT,* e em uma interface chamada *MT* para facilitar a compreensão e gerenciamento de cada parte do código, além de possibilitar novas implementações e alterações em suas funcionalidades sem afetar outras partes do sistema. A seguir serão explicadas o funcionamento de cada classe e da interface para melhor entendimento:

a) ControlaMT: implementando a interface MT, inicialmente são incluídos o pacote padrão *java.util.** que contém uma coleção de classes e interfaces utilitárias e *java.util.stream.Stream*, além das variáveis de instância que armazenam os dados para configurar e executar a MT e um método construtor que inicializa o objeto “scanner” para ler as entradas do usuário. Essa classe é a implementação do autômato em que estamos trabalhando e que permite a entrada de estados, alfabetos, transições, e a execução de cadeias de entrada para determinar se são aceitas ou rejeitadas pela máquina de Turing. Ela contém duas classes internas que serão explicadas a seguir:

   - *ParEstadoSimbolo:* é uma classe auxiliar que representa um par de estado e símbolo, utilizado como chave para identificar as transições especificadas na máquina. Os métodos *equals*(utilizado para comparar e verificar se os objetos são da mesma classe e se têm os mesmo valores para os atributos ‘estado’ e ‘símbolo’) e hashCode(gera um código hash baseado nos atributos ‘estado’ e ‘simbolo’, garantindo que quando tiverem os mesmos valores, também tenham o mesmo código hash) são

sobrescritos para permitir a correta comparação e armazenamento em coleções baseadas em hash.

- *Transicao:* armazena informações sobre uma transição na MT, incluindo o novo estado, símbolo a ser escrito na fita, e a direção do movimento do cabeçote (E para esquerda, D para direita).

Ademais, também foi incluído alguns métodos que serão explicados a seguir:

- *entradaUsuario():* lê as definições usadas na MT a partir das entradas do usuário (número de estados, estado inicial, estados finais, alfabeto principal, alfabeto auxiliar, marcador de início e símbolo branco), que serão solicitadas e armazenadas em suas respectivas variáveis. O mapa de transições é inicializado e para cada par de estado e símbolo será dada uma transição (estado futuro, alfabeto futuro e direção futura). Esse método também válida algumas entradas do usuário a partir de dois métodos denominados *lerEstadoInicial* e *lerNumeroPositivo* que serão explicados futuramente e são usados para garantir o desempenho adequado do simulador.
- *testarCadeias()*: permite o teste de diversas fitas por meio de um loop que solicita a entrada desses caracteres, verificando se os mesmos pertencem a descrição formal inserida anteriormente e realizando as transições de estados, alfabeto futuro e direção futura, assim verificando se essa fita é aceita ou rejeitada. O loop irá ler o símbolo atual na posição do cabeçote e buscar a transição correspondente, em seguida irá

atualizar de acordo com a transição e então verificar se o estado atual é de aceitação. Isso se repete até que o usuário digite ‘1’ para fechar o programa, ‘2’ para testar uma nova MT ou ‘3’ para testar uma nova fita.

- *adicionarBranco():* amplia a fita para a esquerda ou direita, quando o cabeçote se move além dos limites determinados, adicionando um novo símbolo branco. Esse método cria uma nova fita com espaço extra e copia os dados da fita original para adicionar o símbolo branco no início ou final da fita.
- *lerNumeroPositivo():* garante que a entrada do usuário seja um número positivo, assim evitará entradas inválidas que podem acarretar erros na simulação da MT.
- *lerEstadoInicial():* garante que o estado inicial esteja dentro do intervalo de estados válidos, definido por *“0 a numEstados -1”*, caso seja inválido solicita uma nova entrada.

b) MT: interface que estabelece os métodos *entradaUsuario* e *testarCadeias*, necessários para configurar e testar a MT. Definindo assim o que a classe *ControlaMT* deve fornecer ao implementar essa interface.

c) UsarMT: classe usada para instanciar a implementação da Máquina de Turing, utilizando os métodos inseridos na interface *MT* e fornecendo uma maneira de executar essas operações. Ela serve como intermediário na interação entre o usuário e a simulação da MT.

A execução do programa finaliza ao usuário digitar “1”, garantindo que o programa feche apenas após termos obtido e visualizado todos os testes de fita desejados pelo usuário. Algumas outras maneiras de organizar e gerir o código fonte foram utilizadas, como o código ANSI para cores que foi empregado apenas para

fins estéticos e de melhor visualização, porém apenas com o objetivo de facilitar e possibilitar o desempenho geral do mesmo.

2.3 Linguagem de Programação e demais informações

A linguagem de programação Java foi escolhida para o desenvolvimento do código, ela é abordada principalmente durante o segundo ano do curso, e utilizá-la para o desenvolvimento do programa permite conhecer mais sobre a linguagem e desenvolver um aprendizado ao longo do trabalho. Além de pôr em prática os conhecimentos que adquirimos durante as aulas de LFA, Estrutura de dados e Programação orientada a objeto Esta linguagem fornece a possibilidade de utilizar orientação a objetos, uma prática de programação que torna possível elaborar um software a partir da geração de objetos que se comunicam entre si, e o uso de classes para representar objetos do mundo real, onde declararmos atributos e métodos, que representam, respectivamente, as características e comportamentos desse objeto.

Dessa maneira, a linguagem de programação Java se tornou ideal para este caso específico, em que o código fonte foi feito para simular uma Máquina de Turing, afinal o uso de classes permite mudanças, compartilhamento de dados, melhor organização e estruturação do código, etc, possibilitando um código mais propício a futuras melhorias e implementações sem tê-lo que reescrever por inteiro novamente.

CAPÍTULO 3
RESULTADOS

Diante dos expostos apresentados ao longo do trabalho, a finalidade esperada seria o pleno funcionamento de um código que exemplifica como um simulador de Máquina de Turing funciona de maneira prática, demonstrando se uma determinada fita é aceita ou rejeitada, levando em consideração sua tabela de transições e a descrição formal. Dessa maneira, com a implementação total e sua revisão, o objetivo principal do código fonte foi atingido, resultando em uma aplicação funcional no qual recebemos o número total de estados, estado inicial, estado(s) final(is), alfabeto, alfabeto auxiliar, símbolo branco e marcador de início, e em seguida realizamos testes para ver se uma cadeia de caracteres faz parte da linguagem.

Abaixo, nas figuras 2, 4, 5, 6, 7 e 8 podemos ver o funcionamento do simulador de MT a partir do exemplo dado pelo professor nas especificações do trabalho. Resultados:

```
 --- Linguagens Formais, Autômatos e Computabilidade ---

Trabalho de LFA - 2º Bimestre - C.C UNESPAR

Paloma de Castro Leite - 2ª Ano - 22.07.24
 --- SIMULADOR DE MÁQUINA DE TURING ---

Digite o número de estados:
5

Digite o estado inicial (entre 0 e 4):
0

Digite os estados finais (Separe por espaços):
4

Digite o alfabeto (Separe por espaços):
a b

Digite o alfabeto auxiliar (Sem marcador de ínicio e branco):
A B

Digite o marcador de ínicio:
>

Digite um símbolo branco:
<
```

Figura 2

```
---TRANSIÇÕES --

Preencha as transições a seguir:

OBS: Caso não haja transição, insira X para anular o campo

Transição para q0 lendo 'a': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo 'b': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo 'A': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo 'B': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo '>': 0
Alfabeto futuro da transição: >
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo '<': 0
Alfabeto futuro da transição: <
Direção futura da transição (E para esquerda/D para direita): D

Transição para q1 lendo 'a': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D
```

Figura 3

```
Transição para q1 lendo 'b': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D

Transição para q1 lendo 'A': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q1 lendo 'B': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D

Transição para q1 lendo '>': X

Transição para q1 lendo '<': 3
Alfabeto futuro da transição: <
Direção futura da transição (E para esquerda/D para direita): E

Transição para q2 lendo 'a': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo 'b': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo 'A': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo 'B': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D
```

Figura 4

```
Transição para q2 lendo '>': X

Transição para q2 lendo '<': 4
Alfabeto futuro da transição: <
Direção futura da transição (E para esquerda/D para direita): E

Transição para q3 lendo 'a': 3
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): E

Transição para q3 lendo 'b': 3
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): E

Transição para q3 lendo 'A': 3
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): E

Transição para q3 lendo 'B': 3
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): E
```

Figura 5

```
Transição para q3 lendo '>': X

Transição para q3 lendo '<': 4
Alfabeto futuro da transição: <
Direção futura da transição (E para esquerda/D para direita): E

Transição para q4 lendo 'a': 4
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): E

Transição para q4 lendo 'b': 4
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): E

Transição para q4 lendo 'A': 4
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): E

Transição para q4 lendo 'B': 4
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): E

Transição para q4 lendo '>': X

Transição para q4 lendo '<': 4
Alfabeto futuro da transição: <
Direção futura da transição (E para esquerda/D para direita): E
```

Figura 6

```
--- TESTAR FITA ---

Informe uma fita a ser testada (Ou '1' para encerrar, '2' para novo MT, '3' para testar outra
fita):
aabbb

Estado atual: q0
Símbolo atual: >
Fita: >aabbb
Próxima transição: (q0,>) -> (q0,>,D)
```

Figura 7

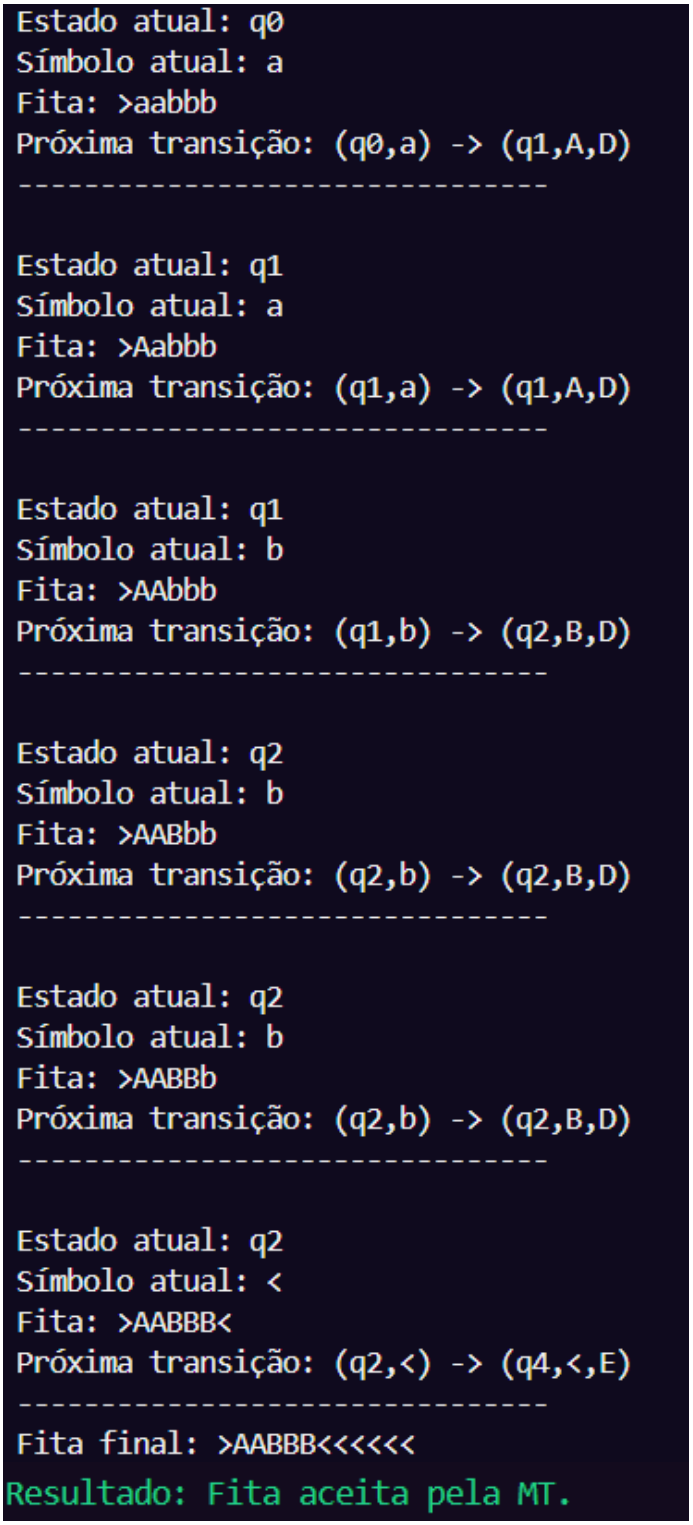

Estado atual: q0
Símbolo atual: a
Fita: >aabbb
Próxima transição: (q0,a) -> (q1,A,D)
Estado atual: q1
Símbolo atual: a
Fita: >Aabbb
Próxima transição: (q1,a) -> (q1,A,D)
Estado atual: q1
Símbolo atual: b
Fita: >AAbbb
Próxima transição: (q1,b) -> (q2,B,D)
Estado atual: q2
Símbolo atual: b
Fita: >AABbb
Próxima transição: (q2,b) -> (q2,B,D)
Estado atual: q2
Símbolo atual: b
Fita: >AABBb
Próxima transição: (q2,b) -> (q2,B,D)
Estado atual: q2
Símbolo atual: <
Fita: >AABBB<
Próxima transição: (q2,<) -> (q4,<,E)
Fita final: >AABBB<<<<<<
Resultado: Fita aceita pela MT.

Figura 8

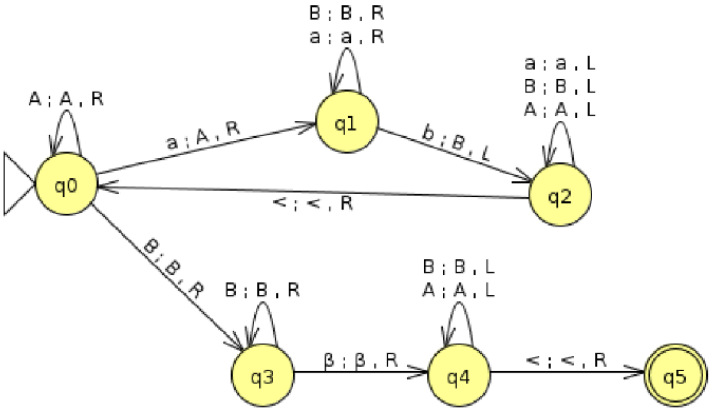
```
Informe uma fita a ser testada (Ou '1' para encerrar, '2' para novo MT, '3' para testar outra
fita):
abba
Fita final: >ABBA<<<<<<

Fita rejeitada.

Informe uma fita a ser testada (Ou '1' para encerrar, '2' para novo MT, '3' para testar outra
fita):
1
```

Figura 9

A seguir outro teste feito a partir de uma máquina de Turing (Figura 10) vista em sala de aula, em que os resultados podem ser vistos nas figuras 11, 12, 13 e 14:



	a	b	A	B	<	β
q0	q1,A,R	X	q0,A,R	q3,B,R	X	X
q1	q1,a,R	q2,B,L	X	q1,B,R	X	X
q2	q2,a,L	X	q2,A,L	q2,B,L	q0,<,R	X
q3	X	X	X	q3,B,R	X	q4,β,R
q4	X	X	q4,A,L	q4,B,L	q5,<,R	X
q5	X	X	X	X	X	X

Figura 10 - Exemplo L = {a^n b^n | n > 0}

Resultados:

```
 --- Linguagens Formais, Autômatos e Computabilidade ---

Trabalho de LFA - 2º Bimestre - C.C UNESPAR

Paloma de Castro Leite - 2ª Ano - 22.07.24
 --- SIMULADOR DE MÁQUINA DE TURING ---

Digite o número de estados:
3

Digite o estado inicial (entre 0 e 2):
0

Digite os estados finais (Separe por espaços):
2

Digite o alfabeto (Separe por espaços):
a b

Digite o alfabeto auxiliar (Sem marcador de ínicio e branco):
A B

Digite o marcador de início:
>

Digite um símbolo branco:
<
```

Figura 11

```
---TRANSIÇÕES --

Preencha as transições a seguir:

OBS: Caso não haja transição, insira X para anular o campo

Transição para q0 lendo 'a': 1
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo 'b': X

Transição para q0 lendo 'A': X

Transição para q0 lendo 'B': X

Transição para q0 lendo '>': 0
Alfabeto futuro da transição: >
Direção futura da transição (E para esquerda/D para direita): D

Transição para q0 lendo '<': X

Transição para q1 lendo 'a': X

Transição para q1 lendo 'b': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D

Transição para q1 lendo 'A': X

Transição para q1 lendo 'B': X

Transição para q1 lendo '>': X

Transição para q1 lendo '<': X
```

Figura 12

```
Transição para q2 lendo 'a': 2
Alfabeto futuro da transição: a
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo 'b': 2
Alfabeto futuro da transição: b
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo 'A': 2
Alfabeto futuro da transição: A
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo 'B': 2
Alfabeto futuro da transição: B
Direção futura da transição (E para esquerda/D para direita): D
Alfabeto futuro da transição: >
Direção futura da transição (E para esquerda/D para direita): D

Transição para q2 lendo '<': 2
Alfabeto futuro da transição: <
Direção futura da transição (E para esquerda/D para direita): D

 --- TESTAR FITA ---

Informe uma fita a ser testada (Ou '1' para encerrar, '2' para novo MT, '3' para testar outra fita):
ab

Estado atual: q0
Símbolo atual: >
Fita: >ab
Próxima transição: (q0,>) -> (q0,>,D)
```

Figura 13

```
Estado atual: q0
Símbolo atual: a
Fita: >ab
Próxima transição: (q0,a) -> (q1,A,D)
--------------------------------

Estado atual: q1
Símbolo atual: b
Fita: >Ab
Próxima transição: (q1,b) -> (q2,B,D)
--------------------------------
Fita final: >AB<<<<<<

Resultado: Fita aceita pela MT.
```

Figura 14

CONCLUSÃO

Com base no que foi descrito ao longo do desenvolvimento do simulador de Máquina de Turing, podemos observar que essa aplicação prática se torna essencial para demonstrar de maneira didática e interativa como as Máquinas de Turing funcionam. Ao permitir que os usuários definam estados, alfabetos, estados finais e transições, e ao possibilitar a inserção e teste de cadeias de entrada, o simulador facilita a visualização imediata dos resultados.

Os resultados obtidos durante a execução desse programa podem ser utilizados como uma ferramenta de estudo, auxiliando na compreensão dos conceitos teóricos apresentados em sala de aula. A capacidade de realizar testes com diferentes tipos de Máquinas de Turing e observar seus comportamentos em tempo real, evidencia como uma Máquina de Turing pode ser utilizada para projetar computadores, algoritmos e programas, demonstrando sua capacidade de reconhecer qualquer gramática regular e resolver diversos problemas computacionais presentes no dia-a-dia.

Em suma, a aplicação prática do simulador de Máquina de Turing não apenas reforça o aprendizado teórico, mas também destaca a importância e a versatilidade dessa máquina no campo da computação. Ela exemplifica de forma clara e prática como conceitos abstratos podem ser aplicados para solucionar problemas reais.

.

REFERÊNCIAS

Linguagens Formais, Autômatos e Computabilidade. Disponível em: <https://github.com/GuilhermeNakahata/UNESPAR-2024/blob/main/Linguagens%20Formais%20Automatos%20e%20Computabilidade/2°%20Bimestre/Aulas/Aulas_LFA_11-06-2024.pdf>

FURGERI, SÉRGIO. Java 8-Ensino Didático: Desenvolvimento e Implementações de Aplicações. Saraiva Educação SA, 2018.

PROF. DR. DAVID BUZATTO. Live Coding: Building a Turing Machine Simulator from scratch using Java and NetBeans. Disponível em: <https://www.youtube.com/watch?v=WA2eeyGtDwc>.

TuringMachine.java. Disponível em: <https://introcs.cs.princeton.edu/java/52turing/TuringMachine.java.html>.