UNIVERSIDADE ESTADUAL PAULISTA

FACULDADE DE CIÊNCIAS E TECNOLOGIA

PROJETO MÁQUINA DE TURING

RELATÓRIO – PARTE 2

TEORIA DA COMPUTAÇÃO

PROF. DR. CELSO OLIVETE JÚNIOR

BRUNO SANTOS DE LIMA

LEANDRO UNGARI CAYRES

PRESIDENTE PRUDENTE

JULHO - 2017

BRUNO SANTOS DE LIMA

LEANDRO UNGARI CAYRES

PROJETO MÁQUINA DE TURING

RELATÓRIO

PARTE II

Relatório do projeto prático parte 2, da disciplina de Teoria da Computação, lecionada pelo docente Dr. Celso Olivete Júnior, no curso Bacharelado em Ciência da Computação – Departamento de Matemática e Computação da Faculdade de Ciências e Tecnologia (FCT Unesp – Presidente Prudente).

PRESIDENTE PRUDENTE

JULHO – 2017

**SUMÁRIO**

[1 INTRODUÇÃO 5](#_Toc488054514)

[2 MÁQUINA DE TURING 6](#_Toc488054515)

[2.1. Como utilizar o simulador da Máquina de Turing 7](#_Toc488054516)

[2.2 Implementação do simulador de Autômatos Finitos 12](#_Toc488054517)

**LISTA DE FIGURAS**

[Figura 1 - Representação da Máquina de Turing com uma única fita. 6](#_Toc488054500)

[Figura 2 - Representação da Máquina de Turing com três fitas 7](#_Toc488054501)

[Figura 3 - Selecionando a Máquina de Turing. 7](#_Toc488054502)

[Figura 4 - Interface de escolha do número de fitas 8](#_Toc488054503)

[Figura 5 - Ilustração de uma Máquina de Turing com três fitas feita utilizando a aplicação. 9](#_Toc488054504)

[Figura 6 - Inserir entradas para a Máquina de Turing. 10](#_Toc488054505)

[Figura 7 - Execução para entrada única. 10](#_Toc488054506)

[Figura 8 - Execução de entradas múltiplas. 11](#_Toc488054507)

[Figura 9 - Execução passo a passo. 12](#_Toc488054508)

# **1 INTRODUÇÃO**

Neste trabalho da disciplina de Teoria da Computação o objetivo da segunda parte do trabalho consiste no aprimoramento da ferramenta desenvolvida para a construção de diferentes categorias de autômatos.

Os aprimoramentos viabilizam a concepção e simulação de Máquinas de Turing com múltiplas fitas de entrada, além disso a realização de transições sem deslocamento de posição na fita, e por fim, a utilização de fitas infinitas para ambos os lados.

A ferramenta utiliza a base do projeto da disciplina de Linguagens Formais e Autômatos, a qual foi implementada utilizando a linguagem de programação Java, com a utilização do JavaFX para construção da interface gráfica, necessitando assim que o usuário utilize a versão 8 do Java. Como ambiente integrado de desenvolvimento foi utilizado o NetBeans, contando também com um sistema de controle de versão Git para versionamento do projeto.

Este relatório explica brevemente a Máquina de Turing com seus aprimoramentos finais e seu funcionamento relatados na seção 2, informando como utilizar a ferramenta que simula esta máquina e como foi o processo de implementação da função que processa o autômato, ressaltando o processamento das multifitas.

# **2 MÁQUINA DE TURING**

A Máquina de Turing é um dispositivo pertencente ao conjunto de máquinas universais, proposto por Alan Turing em 1936, sendo universalmente conhecida e aceita como formalização de algoritmos, possuindo o mesmo poder computacional que um computador de propósito geral.

A máquina tem um mecanismo no qual utiliza-se de dados iniciais presentes na fita, através de deslocamentos para a esquerda e direita realizando operações de leitura e escrita, uma por vez, podendo realizar quaisquer tipos de computação. É permitido ler um símbolo, escrever outro símbolo em seu lugar e movimentar-se na fita. Adicionalmente, esta máquina pode ser representada através de um autômato finito como ilustrado na Figura 1.

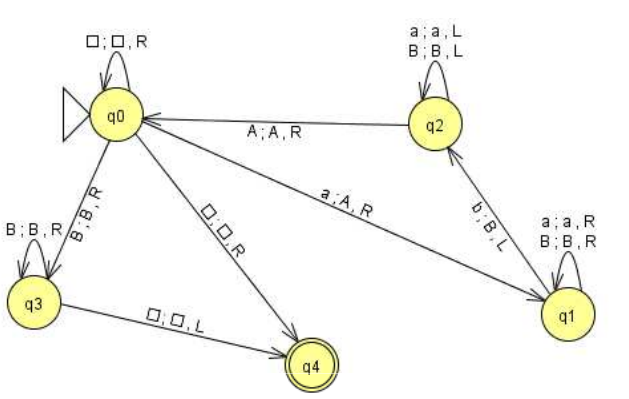


Figura 1 - Representação da Máquina de Turing com uma única fita.

Na Figura 2 é ilustrado uma Máquina de Turing que utiliza múltiplas fitas, no caso três fitas. Assim o usuário insere as três fitas de entrada e elas são processadas pela máquina de modo simultâneo, com isso cada transição tem o número de fitas que o usuário deseja trabalhar, cada regra presente nessas transições são separadas pelo caractere “|”.

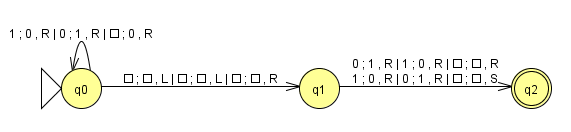


Figura 2 - Representação da Máquina de Turing com três fitas.

## **2.1. Como utilizar o simulador da Máquina de Turing**

A ferramenta utiliza para a realização do desenho gráfico uma biblioteca gráfica nativa da plataforma JavaFX, que por sua vez é nativa da linguagem Java. Esta biblioteca fornece classes para desenho de arcos, círculos, reta, polígonos, entre outras figuras geométricas. Em seguida, foram desenvolvidas as classes responsáveis pela movimentação dos estados. Por fim, foram implementados os métodos que controlavam a máquina e também para o processamento desta.

Na ferramenta desenvolvida, temos um simulador da Máquina de Turing, o qual permite a inserção e remoção de estados assim como de suas respectivas transições, inserção de rótulos para cada estado, possibilitando ainda a movimentação dos estados no painel de visualização.

Para inserir e trabalhar com uma Máquina de Turing acesse o menu Máquinas e clique na opção Máquina Turing.

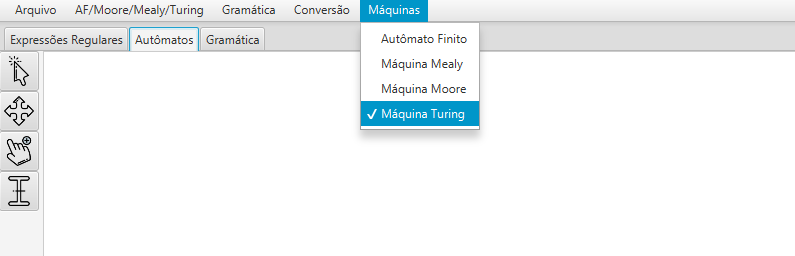


Figura 3 - Selecionando a Máquina de Turing.

Após selecionar a opção Máquina de Turing aparecera uma caixa pedindo para o usuário inserir o número de fitas no qual ele deseja construir a máquina. Tal caixa permite o usuário escolher o número de fitas de 1 a 5.

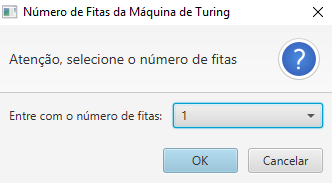


Figura 4 - Interface de escolha do número de fitas.

Após escolher o número de fitas, a tela de desenho é apresentada ao usuário, que pode começar a construir sua Máquina de Turing multifitas.

A interface gráfica da tela de desenho é dividida em duas partes: painel de desenho da máquina e o painel lateral. No painel lateral, estão dispostos quatro botões para a realização de eventos sobre a máquina, a seguir estão descritas as suas funcionalidades:

* **Botão Ponteiro:** criação de arestas entre os vértices, além do clique com o botão direito sobre vértices e transições, nas quais é possível remover transições ou indicar que um vértice é inicial e/ou final.
* **Botão Mover:** responsável pela movimentação de deslocamento de todos os vértices pertencentes ao autômato.
* **Botão Inserção de Vértice:** responsável pela inserção de novos estados no autômato.
* **Botão Texto:** responsável pela criação de rótulos de texto no painel de desenho do autômato.

Abaixo estarão descritos o passo a passo das atividades:

* **Inserção de estados:** para inserir estados deve-se clicar no botão Inserção de Vértice e em seguida, em qualquer lugar do painel de desenho.
* **Remoção de estados:** deve-se clicar no botão Ponteiro e em seguida clicar com o botão direito e depois em Remover.
* **Marcar como inicial:** deve-se clicar no botão Ponteiro e em seguida clicar com o botão direito e depois em Inicial.
* **Marcar como final:** deve-se clicar no botão Ponteiro e em seguida clicar com o botão direito e depois em Final.
* **Inserir legenda:** deve-se clicar no botão Ponteiro e em seguida clicar com o botão direito e depois em Inserir Legenda.
* **Inserir transição:** deve-se clicar no botão Ponteiro e em seguida no vértice inicial e arrastar até o vértice final.
* **Remover transição:** deve-se clicar no botão Ponteiro e em seguida com o botão direito clicar sobre a transição, o qual listará todas as transições disponíveis e depois basta clicar sobre a opção da transição a ser removida.
* **Adicionar rótulo geral:** deve clicar no botão Texto e depois em alguma posição do painel de desenho e depois inserir o texto no campo disponível e clicar Enter.
* **Movimentação geral:** para movimentar qualquer elemento basta clicar no botão e arrastar o elemento desejado para a posição que quiser.

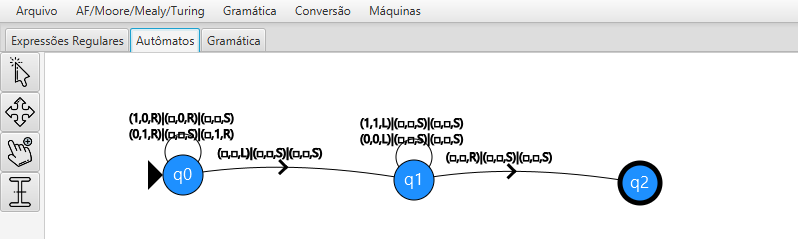


Figura 5 - Ilustração de uma Máquina de Turing com três fitas feita utilizando a aplicação.

Após inserir a Máquina de turing seguindo as instruções acima pode-se acessar o menu **AF/Moore/Mealy/Turing** para poder inserir entradas e testar a máquina que foi desenhada, seja com Entrada única, múltipla ou execução passo a passo, além de contar com uma opção para apagar a máquina que foi desenhada.

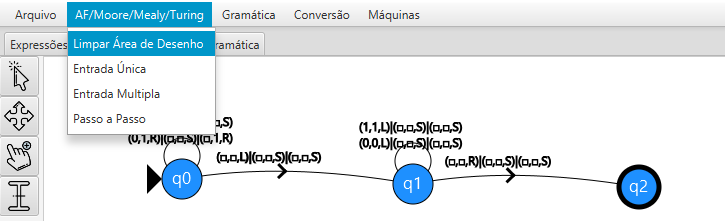


Figura 6 - Inserir entradas para a Máquina de Turing.

Para entrada única insira o conteúdo das fitas e clique em executar, a resposta informará se a entrada foi aceita e caso seja aceita informará os estados na qual a máquina.

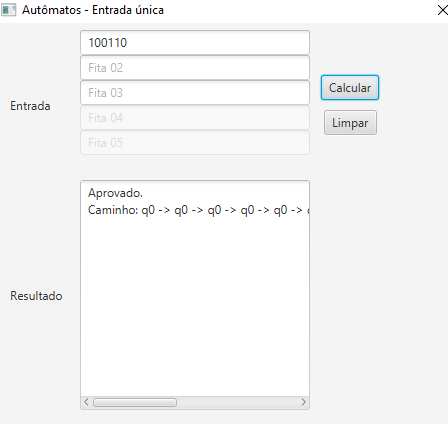


Figura 7 - Execução para entrada única.

Para a execução de múltiplas entradas, basta adicionar as entradas desejadas com os respectivos valores de cada fita e depois clicar em executar, a aplicação informará quais dessa entrada foram Aprovadas e quais foram Rejeitadas.

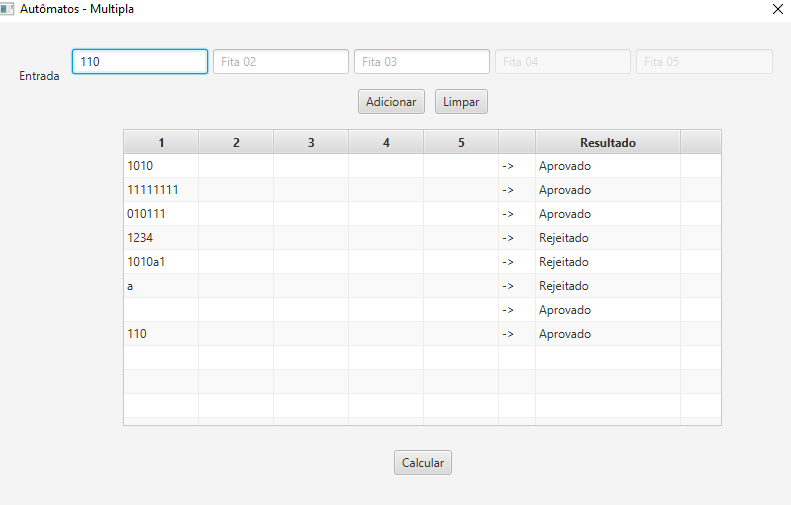
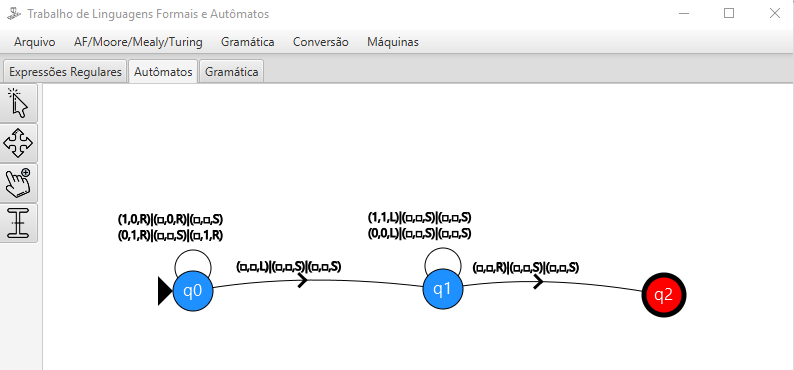


Figura 8 - Execução de entradas múltiplas.

Por fim temos a execução passo a passo, onde basta inserir a fita de entrada e executar passo por passo, o desenho da máquina pinará de vermelho o estado na qual ela está naquele momento, além disso uma janela informará o estado da final naquele passo, caso a entrada seja aceita está janela exibira uma tonalidade verde, caso contrário exibirá uma tonalidade vermelha.



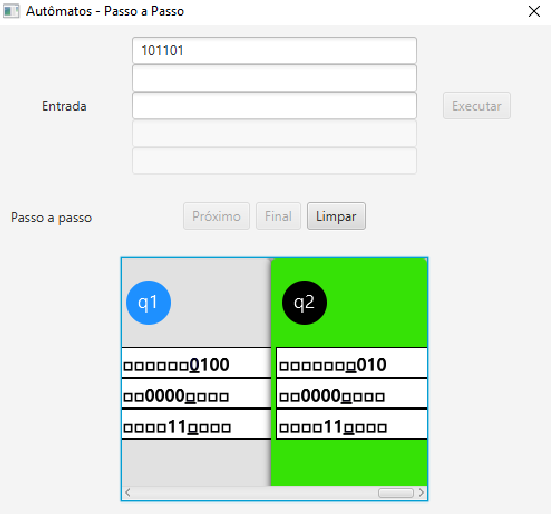


Figura 9 - Execução passo a passo.

## **2.2 Implementação do simulador de Autômatos Finitos**

A implementação do simulador da máquina de turing com multifitas, utiliza como base um algoritmo bastante famoso denominado Busca em profundidade, a ideia do algoritmo de foi preservada e adaptada para que o processamento apresente resultados de acordo com a máquina de turing, a ideia de utilizar o algoritmo de busca em profundidade veio ao observar a estrutura do autômato que representa a máquina de turing, pois o mesmo tem uma estrutura semelhante a um grafo direcionado.

Tem-se duas funções denominadas *processamentoAutomatoTuring* e *verificaRegraTuring*. A primeira função é responsável por realizar a primeira chamada na segunda função, *verificaRegraTuring*, que é uma função de recursão, essa primeira chamada passa o estado inicial da máquina para começar seu processamento afim de encontrar os possíveis caminhos que a máquina de turing pode passar de acordo com sua fita de entrada.

A função *verificaRegraTuring* é recursiva e tem como o objetivo percorrer os estados da máquina de turing de forma com que a palavra de entrada, fita, seja validada, encontrando um desses caminhos, ou seja chegando ao estado final, após percorrer as fitas de acordo com os movimentos de esquerda e direita, L e R respectivamente ou parada representada por S. A função sai das pilhas recursivas já com o armazenamento desse caminho e com a resposta se a palavra de entrada é válida em um *boolean*.

Vale destacar em a cada chamada recursiva parte da palavra de entrada, caractere a caractere é comparado com o do autômato que representa a máquina, assim se as comparações não baterem não é gerado caminho de solução e a palavra de entrada é marcada como invalida. A codificação está localizada dentro da pasta *src* na classe *GerenciadorAutomatos.* Existe uma classe ResultTuring que apenas guarda os resultados do processamento, ou seja, se a entrada foi válida, situação das fitas e estados percorridos. Observe as codificações a seguir:

**public** **static** ResultTuring processamentoAutomatoTuring(String[] palavraDeEntrada) {

//PosPalavra é a variavel que guarda a posição em do cursor em cada fita.

**int** posPalavra[] = **new** **int**[numFitasTuring];

ResultTuring r = **new** ResultTuring();

//Convertendo as palavraDeEntradas (fitas) em arrays de caracteres

ArrayList<ArrayList<Character>> fitas = **new** ArrayList();

**for** (**int** i = 0; i < numFitasTuring; i++) {

ArrayList<Character> letras = **new** ArrayList<>();

**for** (**int** j = 0; j < palavraDeEntrada[i].length(); j++) {

letras.add(palavraDeEntrada[i].charAt(j));

}

fitas.add(letras);

}

r.addCaminho("q" + automato.getLista().get(automato.getLista().indexOf(automato.getInicial())).getValor());

r.addEstado(automato.getLista().get(automato.getLista().indexOf(automato.getInicial())).getValor());

//Salvando situação atual da fita em cada passo.

r.addFita(transformarArrayEmString(fitas));

//Primeira chamada partindo do estado inicial

**return** verificaRegraTuring(r, fitas, posPalavra, **new** **int**[numFitasTuring], automato.getLista().indexOf(automato.getInicial()));

}

**public** **static** ResultTuring verificaRegraTuring(ResultTuring r, ArrayList<ArrayList<Character>> fitas, **int**[] posPalavra, **int**[] adicao, **int** estadoAtual) {

**int** contador;

**boolean** verificaComparacao;

ResultTuring clone = **null**;

//atualizando as posições das fitas

**for** (**int** i = 0; i < FXMLPrincipalController.numeroFitasTuring; i++) {

posPalavra[i] += adicao[i];

}

//Se a maquina chegou ao seu estado final

**if** (automato.getLista().get(estadoAtual).isIsFinal()) {

r.setVerificacao(**true**);

**return** r;

}

/\*\*

\* Para cada transição do nó

\*/

**for** (**int** i = 0; i < automato.getLista().get(estadoAtual).getLista().size(); i++) {

contador = 0;

**while** (contador < numFitasTuring) {

**if** (posPalavra[contador] >= fitas.get(contador).size()) {

//Se andou tudo para direita e chegou uma posição a frente do fim da fita

fitas.get(contador).add('\u25A1');

} **else** **if** (posPalavra[contador] <= -1) {

//Se andou tudo para a esquerda e chegou uma posição antes do inicio da fita.

fitas.get(contador).add(0, '\u25A1');

posPalavra[contador]++; //De -1 vira 0 (inicial da lista)

}

contador++;

}

//Comparando se todos os simbolosLidos são iguais aos simbolos de cada uma das fitas.

contador = 0;

verificaComparacao = **true**;

**while** (contador < numFitasTuring) {

**if**(!(String.valueOf(fitas.get(contador).get(posPalavra[contador])).equals(automato.getLista().get(estadoAtual).getLista().get(i).getSimboloLido(contador)))) {

verificaComparacao = **false**;

**break**;

}

contador++;

}

//Se a comparação anterior for verdadeira significa que pode realizar o movimento, escrita de simbolo e mudança de estado

**if** (verificaComparacao) {

contador = 0;

clone = r.clone();

String[] m = **new** String[FXMLPrincipalController.numeroFitasTuring];

**int**[] l = **new** **int**[FXMLPrincipalController.numeroFitasTuring];

**while** (contador < numFitasTuring) {

//int deslocamento = 0;

//Está dentro dos limites da fita

fitas.get(contador).set(posPalavra[contador], automato.getLista().get(estadoAtual).getLista().get(i).getSimboloEscrito(contador).charAt(0));

**if**(automato.getLista().get(estadoAtual).getLista().get(i).getDirecao(contador).equals("L")) {

//posPalavra[contador]--;

l[contador] = -1;

//clone.addMovimento("L");

m[contador] = "L";

}

**elseif**(automato.getLista().get(estadoAtual).getLista().get(i).getDirecao(contador).equals("R")) {

//posPalavra[contador]++;

l[contador] = 1;

//deslocamento = 1;

//clone.addMovimento("R");

m[contador] = "R";

}

**elseif**(automato.getLista().get(estadoAtual).getLista().get(i).getDirecao(contador).equals("S")) {

//deslocamento = 0;

l[contador] = 0;

// clone.addMovimento("S");

m[contador] = "S";

}

contador++;

}

clone.addMovimento(m);

clone.addCaminho(" -> q" + automato.getLista().get(estadoAtual).getLista().get(i).getAlvo().getValor());

clone.addEstado(automato.getLista().get(estadoAtual).getLista().get(i).getAlvo().getValor());

clone.addFita(transformarArrayEmString(fitas));

ResultTuring resposta = verificaRegraTuring(clone, fitas, posPalavra, l, automato.getLista().indexOf(automato.getLista().get(estadoAtual).getLista().get(i).getAlvo()));

**if** (resposta.isVerificacao()) {

clone = resposta;

**return** (clone);

}

}//Fim do if - resultado da verificacao

}//Fim do For externo - para cada nó da transição

r = (clone != **null** ? clone : r);

r.setVerificacao(**false**);

**return** r;

}

**public** **class** ResultTuring {

**private** **boolean** verificacao;

**private** ArrayList<String[]> movimentos;

**private** ArrayList<String[]> fitas;

**private** ArrayList<Integer> estados;

**private** String caminho;

.....

}