Faculdade de Ciências e Tecnologia
Universidade Estadual Paulista
"Júlio de Mesquita Filho"

Darlan Murilo Nakamura de Araújo

Relatório referente a implementação da Máquina de Turing - parte 2

Prof. Dr. Celso Olivete Júnior
Disciplina: Teoria da Computação

Presidente Prudente
Junho - 2018

Índice

Lista de Figuras

1 Como Usar a Aplicação

1.1 Menu Principal

Através do **Menu Principal**, é possível:

- **Abrir um Arquivo:** é necessário selecionar um arquivo de seu computador que seja no formato ".jff", o formato aceito pelo JFLAP versão 7.
- **Inicializar uma Máquina de Turing Multifita:** é possível criar uma nova máquina de Turing, selecionando a quantidade de fita (variando de 1 a 5).

Ambas as opções irá abrir uma nova interface, na qual será responsável por interagir diretamente com a máquina de Turing aberta. A Figura 1 demonstra como é o menu principal da aplicação. No exemplo, ao selecionar a quantidade de fitas e clicar em "OK", será aberto a interface principal da aplicação, responsável por interagir e testar a máquina criada.

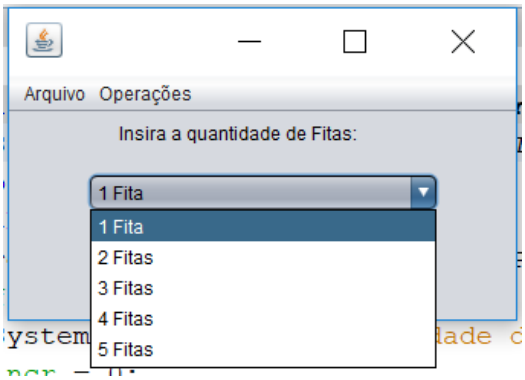

Figura 1: Menu Principal

1.2 Interface Principal

Através da interface principal é possível realizar as seguinte operações (da esquerda para a direita):

- **Mover estados**
- **Adicionar novo estado**
- **Adicionar transição**

- **Apagar estado/transição**
- **Marcar estado como inicial**
- **Marcar estado como final**
- **Alterar estado/transição**

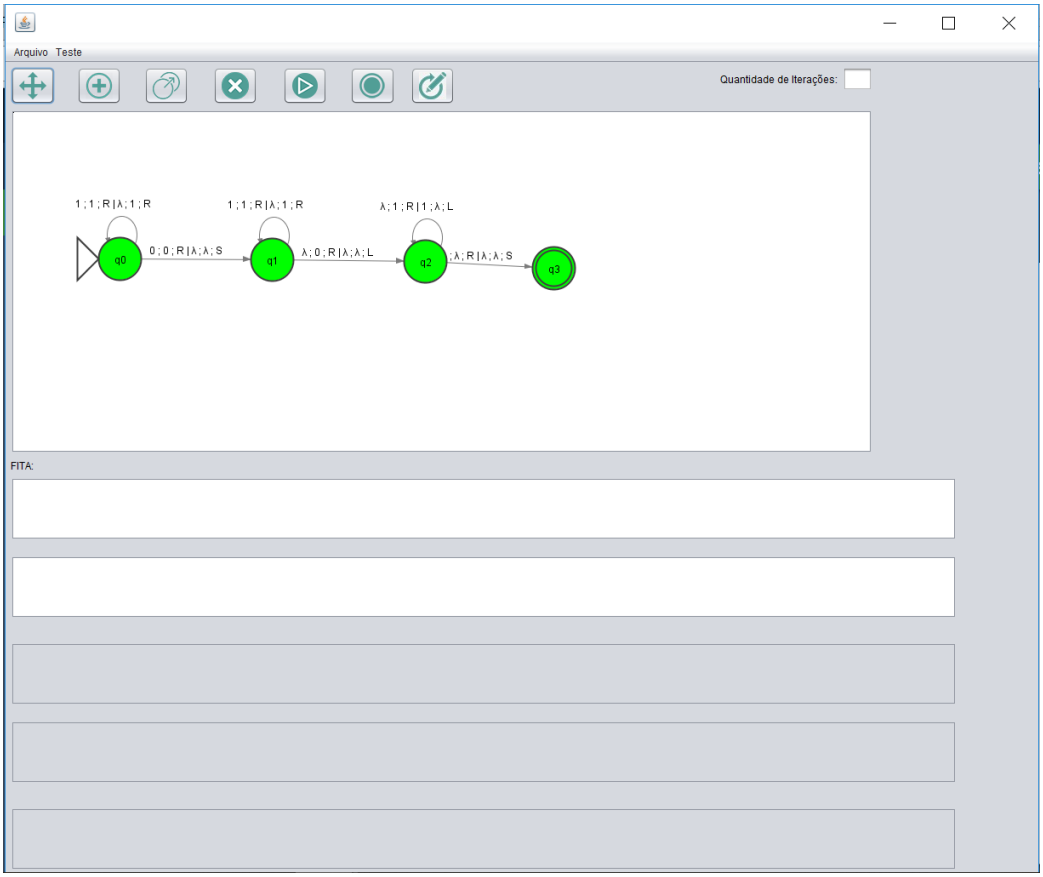

Figura 2: Interface Principal - com uma máquina de turing com duas fitas

Para cada operação, é necessário clicar sobre o botão correspondente e logo em seguida clicar no painel em branco, para que seja realizada a operação selecionada. A Figura 2 traz uma máquina de Turing carregada de um arquivo no formato ".jff".

Estas operações são as operações utilizadas para criar uma máquina de Turing. É possível, através do menu superior, abrir uma nova máquina de Turing, selecionando um arquivo ".jff", salvar a máquina construída (clicando em "Salvar Como"ou em "Salvar"caso o arquivo já tenha sido aberto previamente).

1.3 Testando a Aplicação

No submenu "Testes", é possível realizar três tipos de teste da máquina de Turing construída:

- **Teste Rápido**
- **Teste Passo a Passo**
- **Múltiplas Entradas**

A máquina exemplificada na Figura 2 é uma máquina capaz de realizar a soma unária entre dois números, separados pelo caractere '0', podendo receber de entrada:

- Entrada: 111011
- Saída: 111011011111

Desta maneira, será apresentado abaixo, as simulações realizadas sobre esta máquina:

1.3.1 Teste Rápido

A saída para o teste rápido com a entrada '111011' é apresentada na Figura 3. Pode-se observar que a interface traz a quantidade de iterações que ocorreram para a determinada entrada, traz como a fita 1 e 2 respectivamente (de cima para baixo), após a computação da entrada. A saída na fita 1 é a saída esperada, indicando que a máquina foi desenvolvida de maneira correta. O cursor azul, indica aonde a cabeça da fita correspondente parou.

1.4 Teste Não-Determinístico

A Figura 4 traz um exemplo do resultado da execução da entrada 'aaab' em uma máquina de Turing não-determinística. Neste caso, só é possível atingir o estado final se for não-determinística.

1.4.1 Teste Passo a Passo

O teste passo a passo pode ser realizado de forma semelhante ao teste rápido, informando a entrada (para o caso de Multifita, a entrada informada pelo usuário será carregada apenas na Fita 1 e as demais Fitas serão carregadas sem nenhum dado, apenas com vazio).

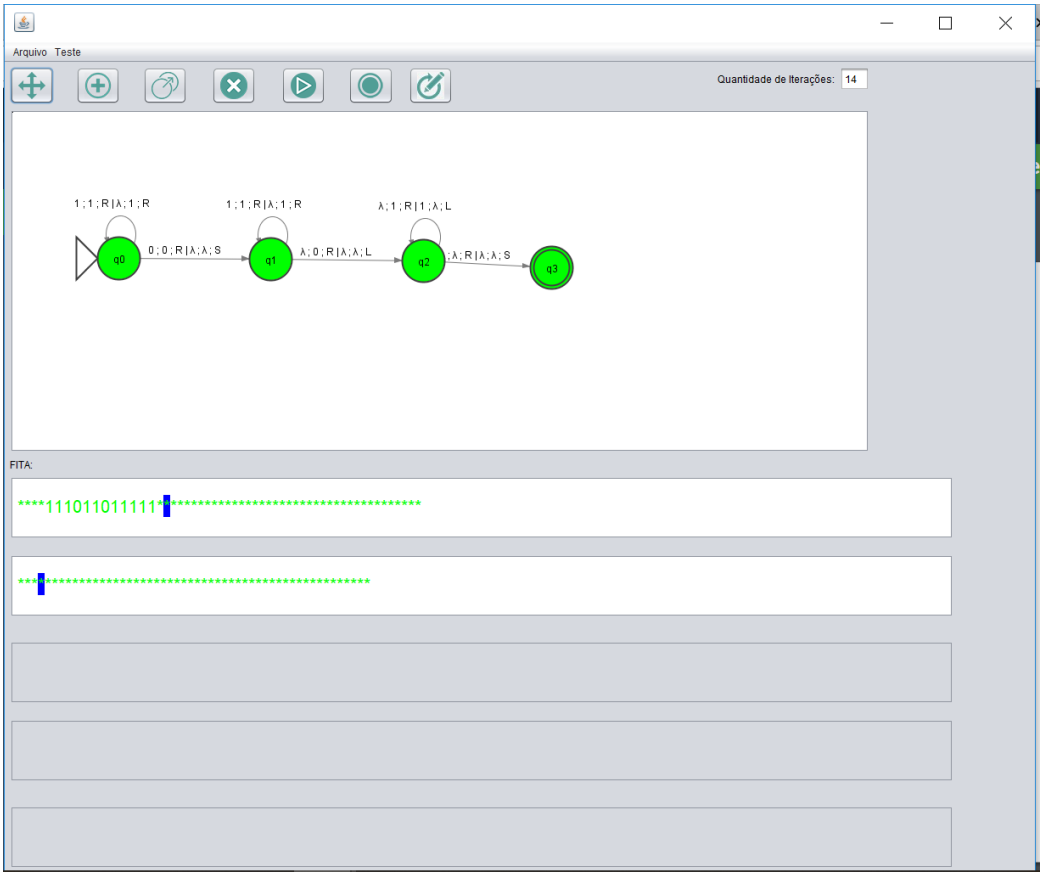

Figura 3: Máquina de Turing capaz de realizar a soma unária - Resultado do Teste rápido para a entrada '111011'

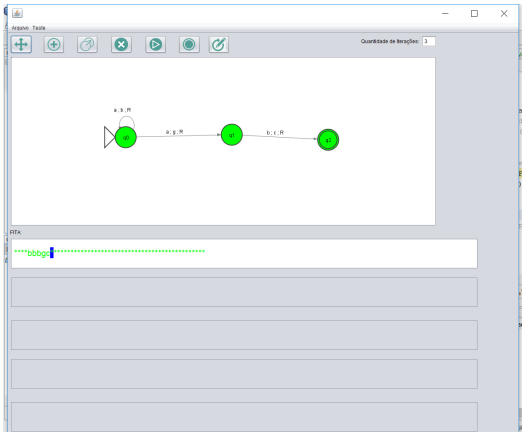

Figura 4: Máquina de Turing não-determinística

Ao informar a entrada, o usuário é orientado a pressionar o botão de próximo para avançar para as próximas iterações. O estado que está sendo executado fica em cinza, e é possível observar o cursor nas fitas se movendo.

A Figura 5 é um exemplo da execução do teste passo a passo. O teste só é apresentado para caso a entrada seja aceita pela máquina, ou seja, quando se atinge o estado final.

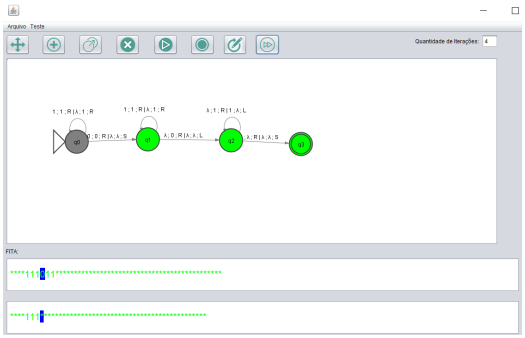

Figura 5: Máquina de Turing capaz de realizar a soma unária - execução do teste passo a passo para a entrada '111011'

1.4.2 Teste Múltiplas Entradas

O teste múltiplas entradas é interessante para o usuário para testar de uma única vez, diversas entradas e verificar se a máquina realmente está bem construída. Ao clicar sobre a opção "Múltiplas Entradas"uma nova interface é aberta, com uma tabela, onde o usuário deve inserir, na coluna "ENTRADA", as entradas que deseja testar, e ao terminar de preencher, clicar em "Testar". Para adicionar novas linhas, é indicado pressionar o botão "TAB"do teclado.

A Figura 6 traz o resultado da execução do teste múltiplas entradas para a máquina exemplificada anteriormente. Desta maneira, pode-se perceber que para as entradas apresentadas, a máquina produziu a saída esperada.

2 Implementação

A implementação da aplicação foi realizada na Linguagem *JAVA*, utilizando a *IDE NetBeans* versão 8.2. Para a Inteface Gráfica, foi utilizado os componentes *Swing* e para a comunicação de entradas e mensagens, os componentes *JOptionPane*, ambos nativos do *JDK* 8.

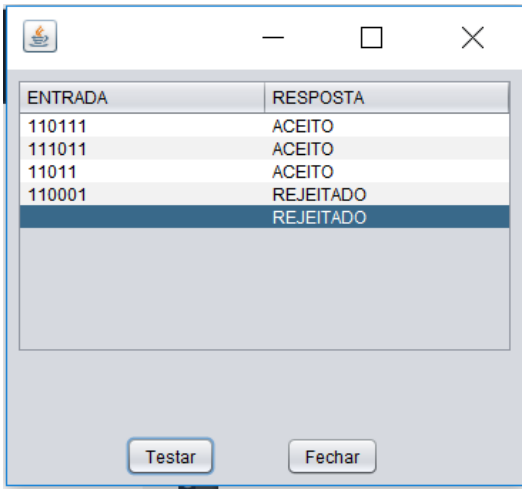

Figura 6: Máquina de Turing capaz de realizar a soma unária - resultado do teste múltiplas entradas

A seguir será apresentado os tópicos referentes às técnicas utilizadas (quais as estruturas computacionais e de maneira teórica que fora utilizado no trabalho), qualidade da solução, na qual será discutido os problemas inerentes à aplicação desenvolvida e por fim, como está estruturado o código-fonte, assim como as classes e particularidades do código.

2.1 Técnicas Utilizadas

Neste tópico, será abordado especificamente do algoritmo utilizado para computar a entrada e gerar a saída (não será abordado questões de interface e outras questões).

A base para o algoritmo para computar a saída e portanto, verificar se uma entrada é válida para uma determinada máquina, é basicamente a **busca em largura**. Como é necessário a execução não-determinística, é adotada a execução da **busca em largura recursiva**, para atender a característica não-determística.

Segue abaixo, o pseudocódigo para a verificação da máquina:

```
boolean buscaLargura(estado, fita):

    se o estado e final:
        retorna VERDADEIRO;

    para cada aresta do estado:
        se para cada posicao da cabeca de cada fita e
```

```
        correspondente ao valor de leitura da aresta:
            escreve na fita o valor de aux.write;
            anda na fita para a posicao aux.move;

            //vamos disparar uma busca em largura recursiva
            //para o estado de destino desta aresta:
            resposta = buscaLargura(aux.destino, fita);
            se resposta for verdadeira:
                retorna VERDADEIRO;

    //se nenhuma das buscas retornar o valor verdadeiro.
    retorna FALSO
```

OBS: Uma outra maneira para se implementar a solução da máquina de Turing, porém sem o reconhecimento de máquinas não-determínisticas é utilizar a estrutura base da busca em largura, ou seja, o uso de uma fila, para verificar os estados.

3 Qualidade da Solução

Referente a qualidade da solução, a execução da máquina de turing (tanto determinística como não-determística), e as multifitas, está funcionando perfeitamente. Há alguns problemas referentes a questão da interface, como para alterar a transição, é um pouco complicado detectar a transição selecionada quando existem 2 ou mais transições perto, pois a verificação pelo raio não funciona.

Acredita-se que o objetivo do trabalho tenha sido atingido, que é aprender o funcionamento da Máquina de Turing e implementá-la de maneira que o usuário consiga criar uma máquina de Turing e realizar testes para determinadas entradas, verificando se a máquina é capaz de atingir e produzir o resultado esperado.

4 Estruturação do Código

O código foi desenvolvido utilizando duas estruturas principais:

- Uma estrutura visual
- Uma estrutura lógica

A estrutura visual, é a estrutura utilizada pela interface para imprimir os elementos na tela. As classes pertencentes a esta estrutura são: *Vertex* (representa os estados), *Edge* (representa as arestas). A classe *Graph* funciona como um controlador e possui tanto a estrutura visual como a lógica. Quando o usuário solicita realizar um teste, é utilizado a conversão da estrutura visual para a estrutura lógica, através do método "*transformaNaNovaEstrutura*".

A estrutura lógica é representada pelas classes: *Aresta* (que representa uma transição de um estado para outro), *Estado* (representando um estado). O método chamado "buscaLarguraRecursiva"é responsável por realizar a verificação da máquina, para a entrada dada. Desta maneira, este método se utiliza exclusivamente da estrutura lógica.

Ambas as estruturas possuem classes, auxiliares, como TransicaoPorFita, Transicao, etc.

A seguir será apresentado todas as classes auxiliares e suas funções:

- **FilaIndices:** a execução passo a passo, é armazenado um *snapshot* (uma foto) de cada iteração da fita, armazenando o conteúdo da fita e onde a cabeça da fita está posicionado. Portanto a estrutura FilaIndices, é uma estrutura do tipo Fila, na qual armazena estas "fotos"da fila.

- **Elemento:** o elemento é a classe que representa a "foto"da fita, como explicitado anteriormente.

- **Transicao:** classe utilizada na estrutura visual. É utilizada pois contém o posicionamento da transição (para verificar caso o usuário selecione uma transição), possui o agrupamento das transições por fita (valor de leitura, valor de escrita, e movimentação da fita) para cada fita, e o *label*, que será impresso esta determinada transição.

- **TransicaoPorFita:** classe intermediária, utilizada tanto na estrutura visual como na estrutura lógica. Esta classe representa uma transição para uma determinada fita. Por exemplo, uma transição para uma máquina de duas fitas, é composta por duas transições por fita (valor de leitura da primeira fita, valor de escrita para a primeira fita, movimentação da primeira fita, valor de leitura para a segunda fita, valor de escrita para a segunda fita, movimentação da segunda fita).

- **Controlador:** classe responsável por tratar da abertura e criação dos arquivos ".jflap".

- **Validator:** responsável por tratar as entradas nas transições da fita.