UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

ENGENHARIA ELETRÔNICA

THAIS YURIKO DE GROOT MIDORIKAWA

**SIMULADOR DE MÁQUINA DE ESTADOS EM C**

Atividade Prática Supervisionada

Curitiba

2017

THAIS YURIKO DE GROOT MIDORIKAWA

**SIMULADOR DE MÁQUINA DE ESTADOS EM C**

Atividade prática supervisionada apresentada como requisito parcial para aprovação na disciplina de Estrutura de Dados 1, do curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Bruno Sens Chang.

Curitiba

2017

**SUMÁRIO**

[1 INTRODUÇÃO 4](#_Toc484526412)

[1.1 MOTIVAÇÃO 4](#_Toc484526413)

[2 DESCRIÇÃO DO PROBLEMA 4](#_Toc484526414)

[2.1 RELAÇÃO COM A DISCIPLINA 6](#_Toc484526415)

[2.2 FLUXOGRAMA 6](#_Toc484526416)

**REFERÊNCIAS 9**

1. INTRODUÇÃO

Máquinas de estados finitos consistem em, basicamente, representações dos estados de um problema e como estes estão relacionados entre si a partir de ações. Assim, tais máquinas podem ser utilizadas para se modelar problemas lógicos complexos de forma a dividir as tarefas a serem resolvidas em problemas menores de uma maneira sistemática. Estas máquinas variam quanto aos tipos de saída gerada, podendo ser aceitadoras ou transdutoras, além de poderem ser distinguidas quanto à quantidade de transições para cada entrada, sendo classificadas como determinísticas ou não-determinísticas.

As máquinas de estados finitos determinísticas, ou autômatos finitos determinísticos (AFD), consistem em máquinas que possuam apenas um estado correspondente a uma determinada entrada, sendo os autômatos finitos não-determinísticas (AFN) o oposto. Apesar de possuir uma complexidade de inferior aos AFNs, os AFDs podem ser aplicados em inúmeros problemas de diferentes áreas, cobrindo a maior parte das necessidades cotidianas que possam vir a necessitar o uso destas máquinas.

* 1. MOTIVAÇÃO

Visto isso, este estudo tem como motivação principal a sua utilização posterior em projetos pessoais envolvendo eletrônica e microcontroladores, além de experiências com o conceito de máquinas de estados em disciplinas cursadas anteriormente. Assim, objetiva a implementação de um simulador não-gráfico de máquinas de estados finitos em linguagem C.

1. DESCRIÇÃO DO PROBLEMA

O conceito de máquinas de estados finitos é bastante antigo e popular, havendo hoje diversas implementações de simuladores destas máquinas em diferentes linguagens de programação e com formas diferentes de construção destas máquinas, como pode ser visto em [1-4].

Em [1], a implementação visual da máquina é obtida a partir de uma expressão regular inserida pelo usuário, de forma que os estados podem ser percorridos dado uma entrada pelo usuário. Já em [2], o usuário manipula graficamente os elementos da página a fim de se montar um diagrama de blocos correspondente à máquina desejada, podendo inicia-la ou percorrê-la passo-a-passo. Por sua vez, o simulador de [3] consiste em um arquivo executável, no qual o usuário, assim como em [2], deve utilizar-se de elementos gráficos para criar a máquina desejada, porém sua representação visual é a partir de grafos. Já [4] consiste no código fonte de um simulador de máquinas de estados finitos na linguagem Python. Por fim, [5] e [6] apresentam simuladores de AFD em C, nos quais o usuário fornece ao programa todos os dados necessários para a criação da AFD e, no fim, este pode inserir uma sequência de símbolos e obter o estado resultante desta sequência.

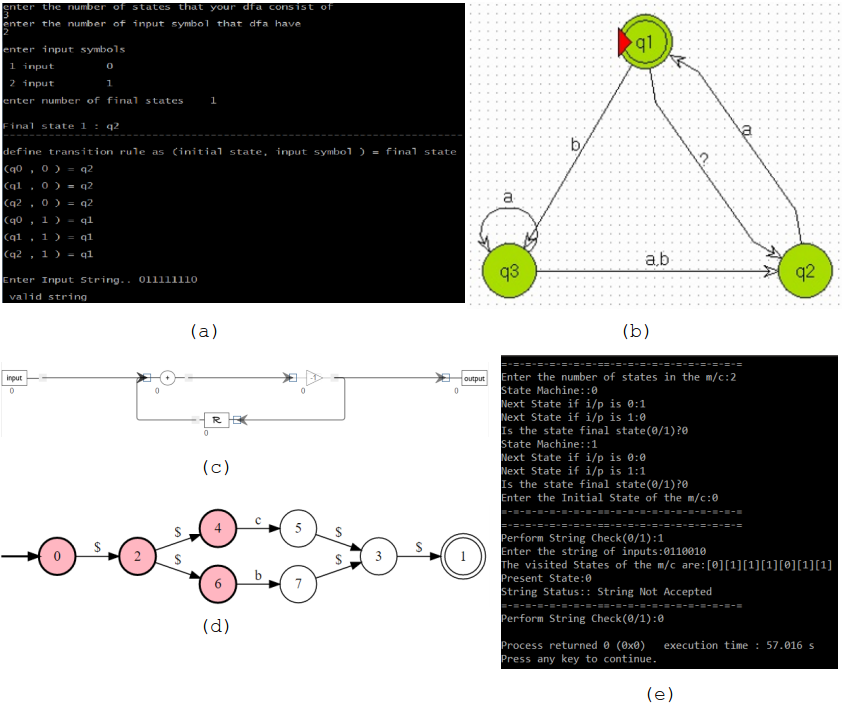


Figura 1. (a) Tela do programa em [6]; (b) Grafo do programa em [3]; (c) Diagrama de blocos de [2]; (d) Grafos a partir de expressão regular em [1]; (e) Tela do programa em [5].

Assim, o simulador a ser implementado neste projeto visa cobrir um determinado número de estados a fim de, além da elaboração de um simulador de AFD, pôr em prática os conhecimentos aprendidos ao longo do curso. O programa segue a fórmula:

sendo o conjunto de estados da máquina, o conjunto de símbolos de entrada, a tabela de transição de estados, o estado inicial e o conjunto de estados finais.

O usuário deverá inicialmente definir a quantidade de estados de sua máquina e os estados resultantes para entradas de 1 bit, bem como o estado inicial e os estados finais da máquina. Assim, a partir de uma sequência de símbolos informada pelo usuário, o programa deverá informar qual é o estado final tendo cada elemento do vetor como símbolo de entrada para os estados. Por fim, a máquina informará se a sequência corresponde a uma entrada válida ou não, de forma que, para ser válida, a sequência deve retornar como estado final um dos estados inicialmente definidos como finais.

* 1. RELAÇÃO COM A DISCIPLINA

Os conceitos aprendidos em sala que foram explorados no decorrer deste trabalho compreendem listas encadeadas, recursões e grafos. À princípio, a conexão entre os estados foi feita utilizando o conceito de listas encadeadas, mas devido a complexidade do problema, este método de solução foi substituído pelo conceito de grafos. A solução teve como base a solução apresentada em [4] e [5]. A recursão foi utilizada para os casos em que o usuário deseja inserir uma nova sequência de bits a fim de testá-la na máquina. Por fim, o conceito de grafos foi utilizado para a conexão entre os estados de acordo com a entrada do usuário.

* 1. FLUXOGRAMA

A solução do problema, cujo fluxograma é ilustrado na Figura 3, além do fluxo de um AFD, conta com verificações das entradas do usuário para que se minimize as chances de quebra do programa. Desta forma, inicia-se com entradas do usuário sobre a quantidade de estados e o estado inicial, sendo seguidas por um loop que visa a conexão entre os estados de acordo com uma entrada informada pelo programa. Ainda, o programa informa os parâmetros do AFD e aguarda que o usuário insira uma sequência de ações, a qual é verificada pelo programa se é uma sequência de símbolos permitidos (se é uma entrada binária ou não). Assim, o programa prossegue para o próximo estado para cada termo até o fim da sequência, informando se o estado final corresponde a um dos estados informados como finais pelo usuário, o que determina se a sequência é uma entrada aceita pela máquina ou não. Por fim, o usuário deve informar se deseja testar uma nova sequência para a mesma máquina ou não, encerrando o programa e removendo da memória todos os estados. A implementação de todas estas etapas é ilustrada na Figura 2.

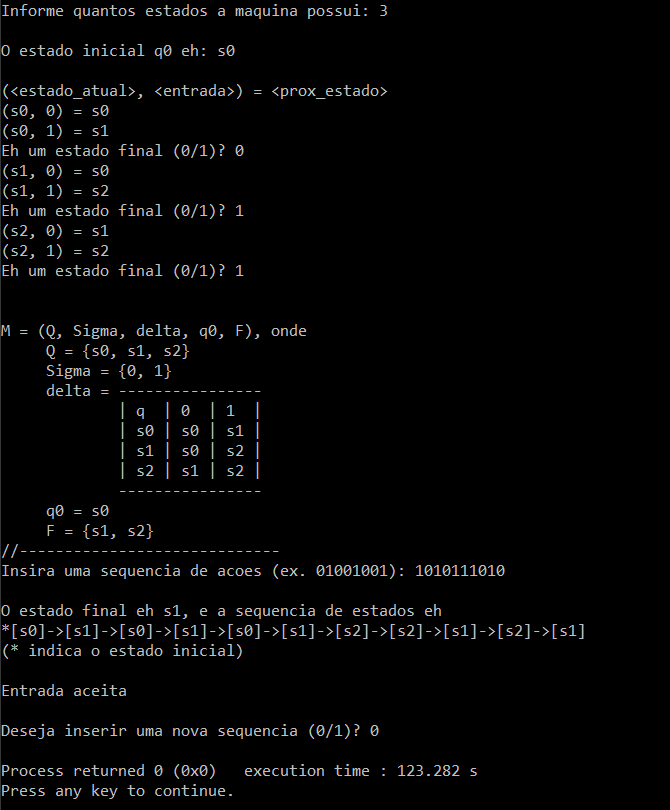


Figura 2. Tela do programa contendo todas as etapas.

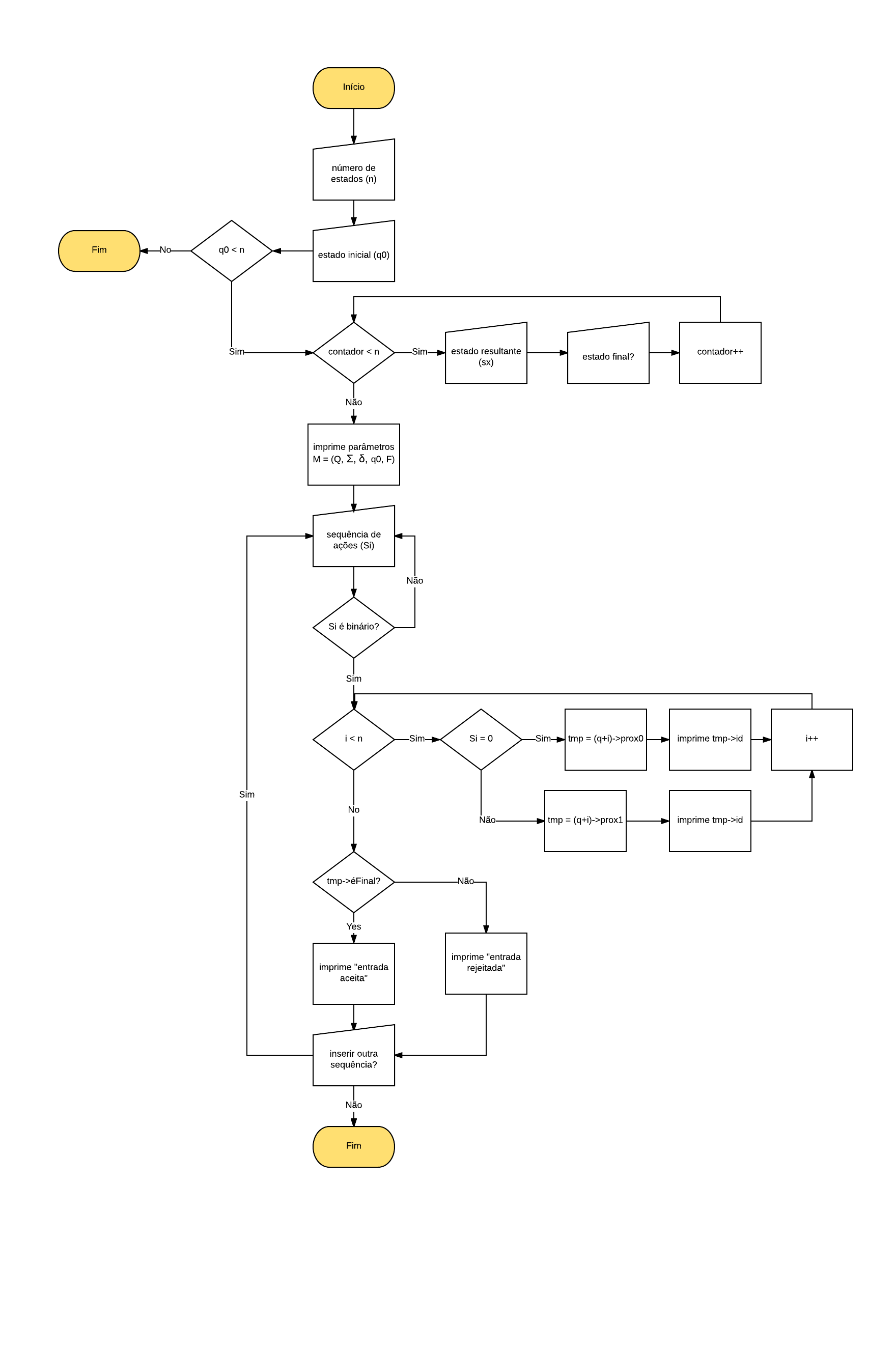


Figura 3. Fluxograma da solução do problema

**REFERÊNCIAS**

[1] ZUZAK, I.; JANKOVIC, V. **FSM Simulator**. s/d. Disponível em: <<http://ivanzuzak.info/noam/webapps/fsm_simulator/>>. Acesso em: 09 mai. 2017.

[2] LU, R.; KEZA, F. **6.01 State Machine Simulator**. s/d. Disponível em: <<http://web.mit.edu/6.mitx/www/6.01-statemachine/mvc.html>>. Acesso em: 09 mai. 2017.

[3] RIBEIRO, N. G. Jr. **Simulador de Autômatos**. s/d. Disponível em: <<http://www.simuladordeautomatos.com/>>. Acesso em: 09 mai. 2017

[4] BHATTACHARYA, A. **FSM Simulator**. 2014. Disponível em: <<https://gist.github.com/hoodwink73/11165659>>. Acesso em: 09 mai. 2017

[5] SUBARNO, P. **Simulating a Deterministic Finite Automaton (DFA) in C**. 2015. Disponível em: <<https://wowmoron.wordpress.com/2015/02/14/simulating-a-dfa-in-c/>>. Acesso em: 30 mai. 2017.

[6] UPADHYAY, A. **C Program for DFA (Deterministic Finite Automata)**. 2013. Disponível em: <<http://www.cprogramto.com/c-program-for-dfa/>>. Acesso em: 30 mai. 2017