

Engenharia De Computação

Teoria da Computação

**Trabalho Final: Autômato Finito Não-Determinístico para
Autômato Finito Determinístico**

Eduardo Yuji Yoshida Yamada

Apucarana-PR, 2022

Eduardo Yuji Yoshida Yamada

**Trabalho Final: Autômato Finito Não-Determinístico para Autômato Finito
Determinístico**

Relatório Técnico do Trabalho Disciplinar apresentado como requisito parcial à obtenção de nota na disciplina de Teoria da Computação do Curso Superior de Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Lucio Agostinho Rocha

Apucarana-PR, 2023

RESUMO

O presente trabalho tem como objetivo apresentar a implementação de um Autômato Finito Determinístico que foi gerado a partir de um Autômato Finito Não-Determinístico, utilizando o teorema da equivalência, na qual dado uma linguagem com alfabeto de 0 e 1, o autômato chegará ao estado final e aceitará a sentença ou não. O autômato foi implementado na linguagem C.

SUMÁRIO

1. INTRODUÇÃO.....	5
1.1 AUTÔMATO FINITO NÃO-DETERMINÍSTICO.....	5
1.2 AUTÔMATO FINITO DETERMINÍSTICO.....	6
2. DESENVOLVIMENTO.....	7
3. RESULTADOS.....	12
4. CONCLUSÃO.....	14
5. REFERÊNCIAS.....	15

1. INTRODUÇÃO

Um Autômato Finito é um sistema de estados pré definidos e finitos, possuindo estados e transições. Ele pode ser dividido em determinísticos, não-determinísticos e com movimentos vazios.

“Um autômato finito sempre para ao processar qualquer entrada pois, como qualquer palavra é finita, e como um novo símbolo da entrada é lido a cada aplicação da função programa, não existe a possibilidade de ciclo (loop) infinito”. (MENEZES, 2011, p. 71)

Foi explorado, neste trabalho, um Autômato Finito Não-Determinístico (AFN). Essa AFN foi transformada em um Autômato Finito Determinístico (AFD).

1.1 AUTÔMATO FINITO NÃO-DETERMINÍSTICO

Um AFN é um Autômato cujo, a partir do estado atual, ao receber uma entrada, pode transitar para um conjunto de estados alternativos. A transição depende do estado atual e do símbolo de entrada.

“Portanto, a cada transição não determinista, novos caminhos alternativos são possíveis, definindo uma árvore de opções”.(MENEZES, 2011, p. 78)

O AFN pode ser descrito por uma quintupla $M = (\Sigma, Q, \delta, q_0, F)$.

Sendo nessa quintupla:

- Σ : alfabeto (finito) de entrada;
- Q : conjunto (finito) de estados;
- δ : conjunto de transições (função parcial, função de transição ou programa)
 $\delta: Q \times \Sigma \rightarrow 2^Q$;
- q_0 : estado inicial ($q_0 \in Q$);
- F : conjunto de estados finais ($F \subseteq Q$).

A função de transição é definida como: $\delta(p, a) = \{q_1, q_2, \dots, q_n\}$

Uma sentença, no AFN, é válida se a transição chegar no estado final. Caso a transição termine em um estado não final ou não possua transição para um estado com o símbolo da sentença, a sentença não será reconhecida. Não existe sentença inválida, ela apenas não é reconhecida pela linguagem.

Em relação a parada de processamento. A entrada é aceita se, após processar o último símbolo da fita, existir pelo menos um estado final que pertence ao conjunto de estados

alternativos alcançados. A entrada é rejeitada se, após processar o último símbolo da fita, todos os estados alternativos alcançados são não-finais ou não há transições para o símbolo da sentença.

Um AFN pode ser transformado em um AFD.

1.2 AUTÔMATO FINITO DETERMINÍSTICO

Um AFD é um sistema de estados pré definidos e finitos.

O AFD pode ser descrito por uma quintupla $M = (\Sigma, Q, \delta, q_0, F)$.

Sendo nessa quintupla:

- Σ : alfabeto (finito) de entrada;
- Q : conjunto (finito) de estados;
- δ : conjunto de transições (função parcial, função de transição ou programa)
 $\delta: Q \times \Sigma \rightarrow Q$;
- q_0 : estado inicial ($q_0 \in Q$);
- F : conjunto de estados finais ($F \subseteq Q$).

A função de transição é definida como: $\delta(p, a) = q$

Uma sentença, no AFD, é válida se a transição chegar no estado final. Caso a transição termine em um estado não final ou não possua transição para um estado com o símbolo da sentença, a sentença não será reconhecida. Não existe sentença inválida, ela apenas não é reconhecida pela linguagem.

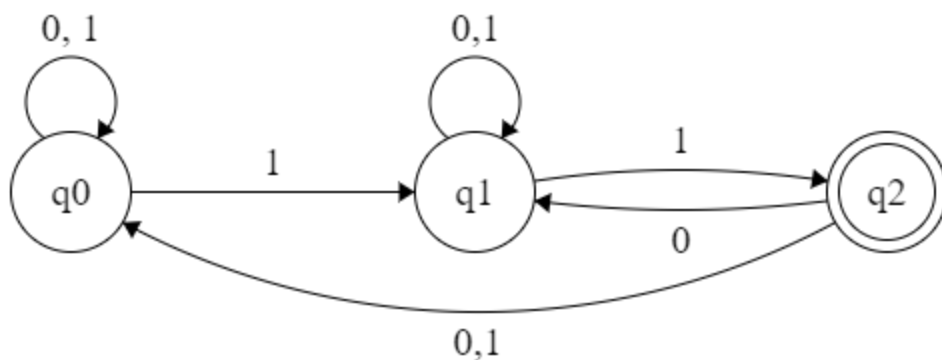
Em relação a parada de processamento. A palavra é aceita se a função de transição alcançar um estado final. A palavra é não-reconhecida se: a função de transição terminar a leitura em um estado não final ou a função transição não possui transição para um estado com o símbolo da palavra. Não há palavra inválida, apenas não é reconhecida pela linguagem.

2. DESENVOLVIMENTO

Para o trabalho final, foi escolhido um Autômato Finito Não-Determinístico representado pela quintupla: $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, q_2)$.

O AFN pode ser representado pela figura 1 a seguir:

Figura 1: AFN



Fonte: Autoria Própria (2023)

Nesse autômato q_0 é onde ocorre a entrada, a primeira iteração, podendo ir para q_1 ou permanecer em q_0 . Essa dualidade na transição para saber em qual estado ele se encontra é o que faz dele um autômato não determinístico.

Matriz de transições:

Tabela 1: Matriz de Transições do AFN

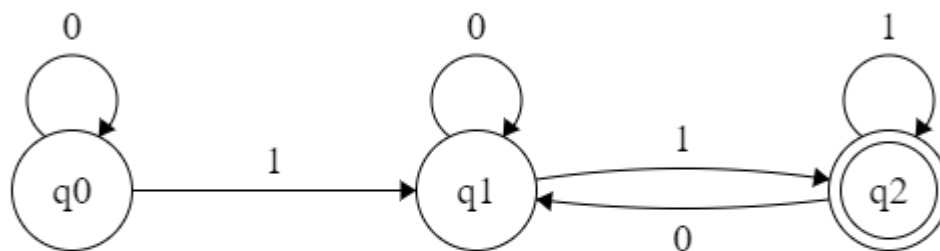
Estados/Entradas	1	0
q_0	$\langle q_0, q_1 \rangle$	$\langle q_0 \rangle$
q_1	$\langle q_1, q_2 \rangle$	$\langle q_1 \rangle$
q_2	$\langle q_0 \rangle$	$\langle q_0, q_1 \rangle$

Fonte: Autoria Própria (2023)

Fazendo uma análise é possível verificar que o autômato acima é um não-determinístico, pois a partir de uma entrada é possível haver mais de uma transição.

Aplicando o algoritmo do Teorema da Equivalência é possível transformar o AFN em um AFD, dessa forma obtemos o seguinte autômato:

Figura 2: AFD



Fonte: Autoria Própria (2023)

Para esse AFD temos a seguinte matriz de transições:

Tabela 2: Matriz de Transições do AFD

Estados/Entradas	1	0
q0	<q1>	<q0>
q1	<q2>	<q1>
q2	<q2>	<q1>

Fonte: Autoria Própria (2023)

A partir desse AFD foi gerado um código em linguagem ‘C’ que simula esse autômato.

O código utilizado está descrito abaixo:

```
/*
```

```
Automato Finito Deterministico
```

```
Autor: Eduardo Yuji Yoshida Yamada
```


Recebe valores 0 e 1

```
*/
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h> //strlen
```

```
int verificaAlfabeto(char input[200], int max, char E[2]){
```

```
    for (int i = 0; i < max; i++){
```

```
        if (input[i] != E[0] && input[i] != E[1]){
```

```
            printf("\nCaractere %c NAO foi aceito.\n", input[i]);
```

```
            return 0; //se não for nenhum das possibilidades do alfabeto, fecha
```

imediatamente pois não é preciso

```
        }
```

```
    }
```

```
    return 1;
```

```
}
```

```
int AFD(char input[200], int max, int init){
```

```
    int inputPointer = 0; //primeiro valor de input
```

```
    int statePointer = init; //primeiro estado = init
```

```
    int sizeInput = strlen(input);
```

```
    int cont = 0;
```

```
    printf("\nQuantidade de iteracoes a ser realizado: %d\n\n", sizeInput);
```

```
    // Q[3] = {'0','01','012'};
```

```
    while(inputPointer < max){
```

```

printf("Iteracao: %d\nEstado atual: q%d ---- Caractere atual: %c\n\n", cont,
statePointer, input[inputPointer]);
cont = cont + 1;
switch(statePointer){ //verifica qual nó está apontando
    case 0:{ //{q0}
        if (input[inputPointer] == '0') //Transição laço
            break;
        if (input[inputPointer] == '1') //Transição para q1
            statePointer = 1;
        break;
    }
    case 1:{ //{q0, q1}
        if (input[inputPointer] == '0') //Transição laço
            break;
        if (input[inputPointer] == '1') //Transição para q2
            statePointer = 2;
        break;
    }
    case 2:{ //{q0, q1, q2}
        if (input[inputPointer] == '0') //Transição para q1
            statePointer = 1;
        if (input[inputPointer] == '1') //Transição laço
            break;
        break;
    }
}
inputPointer++; //avança 1 casa na leitura do input
setbuf(stdin, NULL);
getchar();

}

printf("Estado final: q%d\n", statePointer);

```

```

    if (statePointer == 2) //verifica se o nó é final
        return 1; //caiu em {q0, q1, q2}
    else
        return 0; //caiu em {q0} ou {q0, q1}
}

```

```

int main() {
    char input[200]; //string para ser recebido como input

    printf("-----AUTOMATO FINITO
    DETERMINISTICO-----\nAlfabeto: {0,1}\nEstado Inicial: q0\nEstado Final:
    q2\n\n");

```

```

    printf("Digite a palavra: ");
    setbuf(stdin, NULL);
    fgets(input, 200, stdin);
    input[strcspn(input, "\n")] = '\0';

```

```

    int max; //máximo de iterações, que é a quantidade de símbolos no input
    max = strlen(input);

```

```

/*
* 5-upla:  $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, q_2)$ 
*  $\delta = Q \times \Sigma \rightarrow Q$ 
*/

//int Q[3] = {0, 1, 2}; //estados, apenas para visualização
char E[2] = {'0', '1'}; //alfabeto
int init = 0; //estado inicial
//int F[1] = {2}; //estado final, apenas para visualização

```

```

    if (verificaAlfabeto(input, max, E) == 0){ //verifica se input está dentro do alfabeto

```

```
        printf("\nERRO: INPUT CONTEM CARACTERE FORA DO
ALFABETO\n\n");
        return(2);
    }

    if (AFD(input, max, init) == 1){ //executa o AFD
        printf("\nSENTENCA VALIDA\n\n");
    }
    else{
        printf("\nSENTENCA INVALIDA\n\n");
    }

    return (EXIT_SUCCESS);
}
```

3. RESULTADOS

Utilizando algumas entradas diferentes para testar o programa é possível verificar a iteração que está ocorrendo, o estado atual e o caractere que está sendo lido.

Sendo q0 o estado inicial e q2 o estado final reconhecedor, foram feitos alguns testes:

Primeiro para a palavra '11':

Figura 3: Entrada 11

```
Digite a palavra: 11

Quantidade de iteracoes a ser realizado: 2

Iteracao: 0
Estado atual: q0 ---- Caractere atual: 1

Iteracao: 1
Estado atual: q1 ---- Caractere atual: 1

Estado final: q2

SENTENCA VALIDA
```

Fonte: Autoria Própria (2023)

É possível visualizar que ocorre a transição de $q0 \rightarrow q1$ e em seguida de $q1 \rightarrow q2$. A saída aparece como válida, pois chegou em q2 (estado final)

Utilizando a entrada '00', temos:

Figura 4: Entrada 00

```
Digite a palavra: 00

Quantidade de iteracoes a ser realizado: 2

Iteracao: 0
Estado atual: q0 ---- Caractere atual: 0

Iteracao: 1
Estado atual: q0 ---- Caractere atual: 0

Estado final: q0

SENTENCA INVALIDA
```

Fonte: Autoria Própria (2023)

Nesse caso a sentença é inválida, pois o estado continuou em q0, não chegando em q2.

Testando uma palavra maior:

Figura 5: Entrada 10101

```
Digite a palavra: 10101

Quantidade de iteracoes a ser realizado: 5

Iteracao: 0
Estado atual: q0 ---- Caractere atual: 1

Iteracao: 1
Estado atual: q1 ---- Caractere atual: 0

Iteracao: 2
Estado atual: q1 ---- Caractere atual: 1

Iteracao: 3
Estado atual: q2 ---- Caractere atual: 0

Iteracao: 4
Estado atual: q1 ---- Caractere atual: 1

Estado final: q2

SENTENCA VALIDA
```

Fonte: Autoria Própria (2023)

A sequência de transição foi $q_0 \rightarrow q_1$, $q_1 \rightarrow q_1$, $q_1 \rightarrow q_2$, $q_2 \rightarrow q_1$, $q_1 \rightarrow q_2$. Na iteração 3 o estado sai de q_2 para q_1 , mas como a palavra começa com '1', possui pelo menos dois símbolos '1' e termina em '1', ela finaliza em q_2 .

Testando, agora, uma entrada com símbolo não reconhecido pelo autômato:

Figura 6: Entrada 1012

```
-----AUTOMATO FINITO DETERMINISTICO-----
Alfabeto: {0,1}
Estado Inicial: q0
Estado Final: q2

Digite a palavra: 1012

Caractere 2 NAO foi aceito.

ERRO: INPUT CONTEM CARACTERE FORA DO ALFABETO
```

Fonte: Autoria Própria (2023)

4. CONCLUSÃO

O software funcionou perfeitamente e foi possível verificar diferentes entradas, mostrando até mesmo com entradas inválidas. Até mesmo transformar o AFN em um AFD foi importante, exigindo o conhecimento básico dos Autômatos. Concluo então, que o projeto foi importante para gerar novos conhecimentos, foi necessário realizar várias pesquisas para a implementação do código, unindo com o conhecimento adquirido em sala ao longo do curso. E dessa forma o intuito dele foi cumprido com sucesso.

5. REFERÊNCIAS

MENEZES, Paulo Blauth. Linguagens formais e autômatos. 6. ed. Porto Alegre, RS: Bookman, 2011. 256 p. (Livros didáticos (Universidade Federal do Rio Grande do Sul. Instituto de Informática) ; 3).

TOSCANI, Laira V.; VELOSO, Paulo A. S. Complexidade de Algoritmos. Editora Bookman. 3a edição. 2012.

VIEIRA, Newton José Introdução aos Fundamentos da Computação. São Paulo: Cengage Learning, 2006.