



Universidade Federal de Viçosa

Campus Florestal

Disciplina: Fundamentos da Teoria da  
Computação

Professor: Daniel Mendes Barbosa

## **Trabalho Prático 3**

Eduardo Vinicius – 3498

Isabella Ramos – 3474

Pablo Ferreira – 3480

Samuel Sena – 3494

Florestal

2020

## SUMÁRIO

1.	Introdução.....	3
2.	Decisões do projeto.....	3
3.	Implementação.....	3
4.	Execução.....	5
5.	Considerações Finais.....	6

## 1. Introdução

O trabalho apresentado a seguir traz um algoritmo implementado na linguagem de programação Python 3 e é capaz de realizar a leitura de arquivos (devidamente formatados) com a descrição de Autômatos Finitos e suas respectivas palavras de entradas. Em seguida, o algoritmo executa a computação, imprimindo para cada entrada as saídas: “OK” para uma palavra reconhecida e “X” para uma palavra não reconhecida.

## 2. Decisões do Projeto

A linguagem de programação Python foi escolhida devido a sua facilidade em manipulação de listas e comparação de objetos e Strings.

Escolhemos por realizar a implementação das funcionalidades extras 1, 2 e 3, citada na descrição do trabalho como **Alfabeto de entrada**, **Não determinismo** e **Autômato de Pilha** respectivamente. No momento da execução do algoritmo, é perguntado o nome do arquivo de entrada. Em seguida, um menu é exibido e o usuário deve escolher se deseja executar o programa com o alfabeto padrão {0,1} ou com o alfabeto arbitrário, em seguida é perguntado se o autônomo se trata de um AFD, AFN, APD ou APN. Estas escolhas devem ser feitas tendo em mente a formatação e descrição do autônomo no arquivo de entrada, uma vez que o arquivo de entrada com alfabeto arbitrário apresenta uma linha com a declaração do alfabeto.

## 3. Implementação

A implementação do algoritmo consistiu inicialmente na utilização de estruturas para a correta leitura dos arquivos de entrada. Tendo diferença na forma como a leitura é tratada de acordo com o alfabeto de entrada escolhido inicialmente. Em seguida, a classe Automato foi definida e os métodos a seguir foram implementados:

- **Construtor**
  - Realiza a inicialização do objeto instanciado, recebendo como parâmetro e ajustando estados, estados iniciais e estados finais.
- **AdicionaTransicao**
  - Adiciona transições em devidas listas internas do objeto.

- **AdicionaEntrada**

- Adiciona as palavras de entrada na devida lista interna do objeto.

- **RealizaComputação**

- Realiza a computação de todas as palavras de entradas presentes na lista do objeto, além de imprimir as saídas para cada palavra de entrada.

- **MultiplaComputação**

- Utilizada em classes não determinística, calcula as possibilidades e reconhece uma linguagem se a mesma chegar em um estado final, e pilha vazia(em AP's)

As figuras abaixo ilustram partes do código citadas nas descrições anteriores:

```
class Automato:
    def __init__(self,Q,I,F): #Construtor inicialmente seta Estados, Iniciais e Finais, alem de remover letras Q,I e F de vetores
        self.Estados=Q.split()
        del(self.Estados[0])
        self.EstadoI=I.split()
        del(self.EstadoI[0])
        self.EstadosF=F.split()
        del(self.EstadosF[0])
        self.Origem=[]
        self.Destino = []
        self.SimbolosEntrada = []
        self.Entradas = []

    def AdicionaTransicao(self,Linha):
        Linha = Linha.split() #Posicao 1 e 3 devem ser desconsideradas ( -> e | )
        del(Linha[1])
        del(Linha[2])
        self.Origem.append(Linha[0]) #Pega primeiro termo e depois o remove
        del(Linha[0])
        self.Destino.append(Linha[0]) #Com remoção de estado de origem, segundo termo vira primeiro e após pega-lo, o remove tb
        del(Linha[0])
        self.SimbolosEntrada.append(Linha[1]) #Restante dos termos correspondem a valores aceitos na entrada

    def AdicionaEntrada(self,Entrada):
        self.Entradas.append(Entrada)
```

```
def RealizaComputacao(self):
    print(" ") #quebra de linha
    for i in self.Entradas: #Cada entrada é iterada
        # print("\n\t->Entrada:",i)
        self.EstadoAtual = self.EstadoI[0] #Estado inicial setado como atual
        #print("EstadoAtual= ", self.EstadoAtual)
        Tam=len(i)
        index=0 #index para indexar entrada termo a termo
        for z in range(len(i)-1): #cada iteracao corresponde a uma tentativa de executar um simbolo da entrada
            verificador=0 #verificador para transicao que leva para estado de erro
            Posicoes=[]
            for j in range(len(self.Origem)):
                if(self.Origem[j] == self.EstadoAtual):
                    Posicoes.append(j) #Posicoes onde EstadoAtual possui transicoes
            # print("Posições do estado: ",self.EstadoAtual," :",Posicoes)
            for k in Posicoes:
                if i[index] in self.SimbolosEntrada[k]: #Caso o simbolo esteja nos simbolos aceitos pela transicao, ela é realizada
                    # print("Simbolo Lido:",i[index], " index: ", index, " tam: ", Tam)
                    # print("EstadoDestino= ", self.Destino)
                    self.EstadoAtual = self.Destino[k] #Atualizando estado atual
                    #print("\nSimbolo Lido->", i[index])
                    #print("EstadoAtual= ", self.EstadoAtual)
                    verificador = 1
                    if(index<Tam-1):
                        index+=1 #Caso transicao aconteca, o index é incrementado
                    break
            if(verificador==0): #Caso nenhuma transicao tenha acontecido, simbolo de entrada levou para o estado de erro
                self.EstadoAtual = "Estado_Erro"
            # print("Estado Atual antes de verificar:",self.EstadoAtual)
            if(self.EstadoAtual in self.EstadosF): #Caso o estado atual final esteja em um estado final, palavra reconhecida!
                print("OK -> Entrada: ",i)
            else:
                print("X -> Entrada: ",i)
```

## 4. Execução

A execução do algoritmo ocorre de maneira simples. Para executar, basta executar o arquivo “*main.py*” com o terminal devidamente navegado até a pasta do arquivo, com o seguinte comando :

```
python3 main.py
```

Em seguida, uma mensagem requisitando o nome do arquivo de entrada localizado na pasta “Arq\_Entrada” que o usuário deseja abrir, será impressa na tela. Após digitar o nome do respectivo arquivo, o usuário deverá escolher em um menu se o arquivo de entrada é um AF com alfabeto padrão ( $\{0,1\}$ ) ou com alfabeto arbitrário, em seguida deverá escolher também se o autônomo lido se trata de um AFD, AFN, APD ou APN. Por fim, cada palavra de entrada é impressa e sua saída em seguida. Sendo que, como dito anteriormente, a saída “OK” significa palavra reconhecida e “X” significa palavra não reconhecida. Ressaltamos que o arquivo “Linguagens.txt” contido na mesma pasta que os arquivos de entrada não é um arquivo de entrada, ele apenas armazena as linguagens reconhecidas por cada um dos autômatos contidos nos demais arquivos de entrada. As imagens abaixo ilustram testes de execução:

AFD

```
Entre com o nome do arquivo a ser aberto:A4.txt
Menu:
    1 - Alfabeto {0, 1}
    2 - Alfabeto Arbitrario
Entre com uma opcao:1
    1 - AFD
    2 - AFN
    3 - APD
    4 - APN
Entre com uma opcao:1
OK -> Entrada: 1010
OK -> Entrada: 0110
OK -> Entrada: 1010101010
X -> Entrada: 0001
X -> Entrada: 0010
```

AFD com Alfabeto Arbitrário

```
Entre com o nome do arquivo a ser aberto:Entrada2.txt
Menu:
    1 - Alfabeto {0, 1}
    2 - Alfabeto Arbitrario
Entre com uma opcao:2
    1 - AFD
    2 - AFN
    3 - APD
    4 - APN
Entre com uma opcao:1
OK -> Entrada: !!@
X -> Entrada: @!!!
OK -> Entrada:
OK -> Entrada: @!!!!@
X -> Entrada: @@@
X -> Entrada: @
```

```

Entre com o nome do arquivo a ser aberto:Entrada4.txt
Menu:
    1 - Alfabeto {0, 1}
    2 - Alfabeto Arbitrario
Entre com uma opcao:1
    1 - AFD
    2 - AFN
    3 - APD
    4 - APN
Entre com uma opcao:2
X -> Entrada: 0000
X -> Entrada: 11111
X -> Entrada:
OK -> Entrada: 1000001
OK -> Entrada: 101010101
OK -> Entrada: 1111101

```

```

Entre com o nome do arquivo a ser aberto:Entrada5.txt
Menu:
    1 - Alfabeto {0, 1}
    2 - Alfabeto Arbitrario
Entre com uma opcao:1
    1 - AFD
    2 - AFN
    3 - APD
    4 - APN
Entre com uma opcao:3
OK -> Entrada: 110
X -> Entrada: 000
X -> Entrada: 011
OK -> Entrada: 101
X -> Entrada: 1101
X -> Entrada: 1001
X -> Entrada: 1111
X -> Entrada: 0101
X -> Entrada: 11
X -> Entrada: 10101010

```

## 5. Considerações Finais

A realização do trabalho foi uma tarefa sem grandes empecilhos. A linguagem de programação escolhida nos proporcionou um auxílio muito grande na forma como o fluxo de código foi desenvolvido. Ficamos muito satisfeitos com o resultado final obtido e apenas não implementamos mais funcionalidades extras devido ao tempo limite ter sido alcançado.