UNIVERSIDADE FEDERAL DE VIÇOSA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Erian Alírio De Oliveira Alves - 3862
Gabriel Miranda - 3857
Guilherme Sérgio De Oliveira - 3854
Leonardo F. Rodrigues - 5159
Maria Theresa Arruda E Henriques - 3486
Mariana de Souza - 3898

TRABALHO PRÁTICO:
AUTÔMATO FINITO DETERMINÍSTICO

Trabalho prático da disciplina Fundamentos da Teoria da Computação - CCF 131, do curso de Ciência da Computação da Universidade Federal de Viçosa - Campus Florestal.

Professor: Daniel Mendes Barbosa

FLORESTAL
2021

SUMÁRIO

1. Introdução

Este trabalho tem como objetivo implementar um programa que seja capaz de, a partir de determinada entrada, configurar um autômato finito determinístico (AFD), e logo em seguida utilizá-lo para aceitar ou não uma palavra de acordo com a linguagem configurada. Para além dessa implementação base, também foi desenvolvido uma funcionalidade em que, dado um alfabeto arbitrário, o AFD pode utilizá-lo juntamente com seus parâmetros já anteriormente definidos (como a linguagem a que ele pertence), com o objetivo de reconhecer ou rejeitar uma palavra.

2. Desenvolvimento

Para entender um autômato finito determinístico, primeiro precisamos entender o que cada palavra em particular significa. Assim, um autômato constitui um sistema matemático formal, em que inúmeras propriedades podem ser inferidas a partir de verdades previamente conhecidas ou admitidas por hipótese, sendo em seguida expressas como demonstrações de um raciocínio lógico (RAMOS, 2008).

Já um autômato finito trata-se de um modelo computacional que é capaz de reconhecer linguagens formais que podem ser expressas em sistemas de estados finitos. Deste modo, uma máquina de estados finitos consiste de um conjunto finito de instruções que especifica quais as operações disponíveis e necessárias para realizar uma determinada tarefa. Por fim, um autômato finito determinístico é aquele em que há apenas uma possibilidade de transição, sendo assim determinística, para que o autômato, a partir de uma determinada configuração, se desloque para outro estado. Sendo assim, um AFD não pode ter um mesmo símbolo ocorrendo mais de uma vez para dois estados.

2.1 Autômato Finito Determinístico escolhido

O AFD escolhido pode ser descrito pela quíntupla: (E = {a, b, c, d, e, f, g, h}, Σ = {0, 1}, δ, i = {a}, F = {h}). Em português, este AFD é definido como o conjunto de palavras do alfabeto {0,1} que possuem tamanho maior que 6. A expressão regular $(0 + 1)^6(0 + 1)^+$ também denota o presente AFD. Nela, $(0 + 1)^6$ descreve as palavras com tamanho maior que 6 e, quando concatenada com $(0 + 1)^+$ a máquina define que somente palavras de tamanho maior que 6 podem ser reconhecidas, haja visto que o Fecho de Kleene positivo (“+”) faz com que a expressão retorne em uma ou mais ocorrências de 0’s e 1’s.

A Figura 1 abaixo apresenta o diagrama de estados - feito na ferramenta JFLAP - do AFD escolhido:

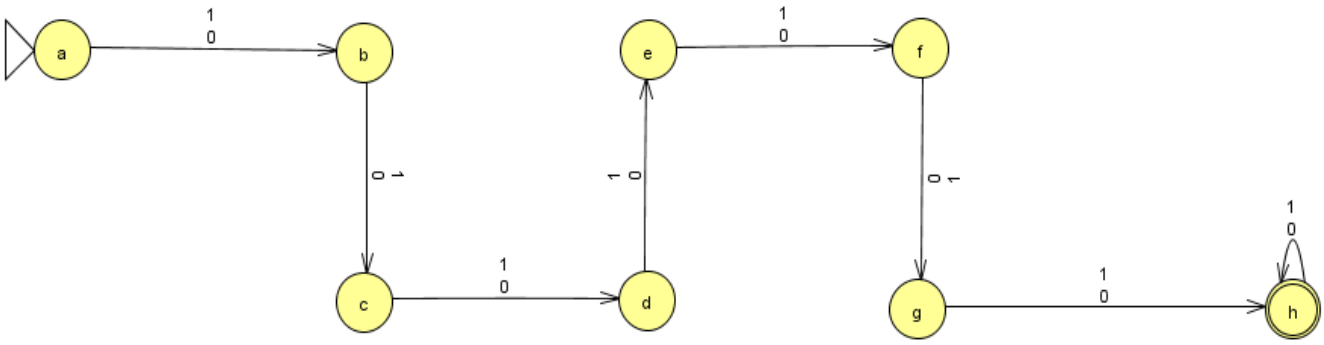

Figura 1. Diagrama de estados do AFD escolhido

Conforme pode ser visto na Figura 1, o diagrama não possui estado de erro, visto que a máquina possui todas as transições possíveis para cada estado. Além disso, o fato de uma palavra não ser reconhecida não caracterizaria um erro, pois uma sequência de símbolos não reconhecidos só ocorreria se em um determinado caso de teste houvesse uma determinada palavra com tamanho menor ou igual a 6.

3. Implementação

O trabalho foi implementado em linguagem Python, que é uma linguagem de código aberto, de fácil entendimento e bastante difundida na Teoria da Computação em codificações de autômatos. Os experimentos foram realizados na plataforma Google Colab, que fornece um ambiente de desenvolvimento colaborativo de tal forma que os usuários podem fazer modificações na implementação em tempo real, onde quer que estejam. A execução foi feita em uma GPU da própria ferramenta.

3.1 Entradas

Para as entradas temos que o autômato é composto por um alfabeto de {0, 1}, onde a primeira linha contém os caracteres dos nomes de cada estado separados por espaço de “a” até “h”. A segunda linha descreve o estado inicial “a” e a terceira linha apresenta o estado final “h”. É possível observar abaixo na Figura 2 as entradas descritas nesta seção e que foram utilizadas na implementação do presente trabalho:

```
1  Q: a b c d e f g h
2  I: a
3  F: h
4  a -> b | 0 1
5  b -> c | 0 1
6  c -> d | 0 1
7  d -> e | 0 1
8  e -> f | 0 1
9  f -> g | 0 1
10 g -> h | 0 1
11 h -> h | 0 1
12 ---
13 0000000
14 1111111
15 100100100
16 000000
17 111111
18 100100
```

Figura 2. Entradas utilizadas na implementação

Na Figura 2, “Q” representa o conjunto de estados, “I” denota o estado inicial, “F” identifica o estado final. Posteriormente, a partir da quarta linha ocorrem as transições dos estados do AFD, todas ocorrendo para 0 ou 1 em cada transição. Após as transições, a partir da linha 13, estão os casos de teste. Os três primeiros casos devem ser aceitos, pois descrevem palavras de tamanho maior que 6. Os três últimos devem ser rejeitados, visto que possuem tamanho igual a 6.

3.2 Codificação

Inicialmente, foi definida uma função denominada “AFD”, que inicializa o estado atual como sendo o estado inicial. Em seguida, essa função será percorrida de acordo com o tamanho da palavra de entrada. Além disso, a palavra é percorrida ao passo que é verificado se para o símbolo lido no momento há alguma transição partindo do estado atual. Por fim, a função retorna o estado no qual parou a computação da palavra de entrada.

Quanto à função “VerificarReconhecimento”, a mesma é responsável por chamar a função anterior, “AFD”, e utiliza seu retorno para verificar se o resultado da função, que se trata do estado atual, é um estado final.

As funções de leitura de arquivo, “read_file” e “read_file_ap”, possuem a mesma lógica de execução, ambas tratam linha por linha do arquivo de entrada, onde para cada linha o tratamento será de acordo com a significância do tipo de entrada. Por exemplo, ao receber um arquivo com a especificação de um AFD, sua primeira linha após o tipo do autômato representará os estados do autômato, necessitando de uma utilização isolada de cada estado da linha para a criação do AFD. Assim, o procedimento necessário para essa linha é basicamente a transformação do conjunto de estados em uma lista em linguagem Python.

A função “Execucao_AFD” é responsável por chamar a função “read_file” que utilizará as informações retornadas por ela para averiguar o reconhecimento das palavras informadas. Para cada palavra de entrada presente no arquivo de especificação do AFD, será chamada a função “VerificarReconhecimento” que fará uso dos outros dados retornados pela “read_file” para conferir se o autômato reconhece ou não tal palavra.

```
def Execucao_AFD(filename):
    machine, i, F, casos_teste = read_file(filename)
    for teste in casos_teste:
      print(list(map(int, teste)))
      estadoInicial = i[0]
      estadoFinal = F
      VerificarReconhecimento(list(map(int, teste)), machine, estadoInicial, estadoFinal)
```

Figura 3. Função “Execucao_AFD”

3.3 Funcionalidades extras

Na primeira funcionalidade extra foi definida uma função denominada “AP”, muito semelhante com a função “AFD”. Ela inicializa um estado atual como sendo o estado inicial. Em seguida, essa função será percorrida de acordo com o tamanho da palavra de entrada e seu estado atual. Além disso, a palavra é percorrida e ao passo que a cada símbolo lido, é verificada uma possibilidade de transição. As possibilidades de transição incluem todas as configurações para um determinado símbolo, acrescido das configurações para transições com o símbolo vazio. Por fim, a função retorna o estado no qual parou a computação da palavra de entrada, o tamanho da pilha e a quantidade de símbolos lidos.

Todos esses valores retornados são utilizados em uma função chamada “VerificarReconhecimento_ap”, que indica se uma palavra é aceita ou não utilizando o método de reconhecimento por pilha vazia e estado final.

Outra funcionalidade extra escolhida foi a de leitura de um alfabeto arbitrário, para isso foram incluídas linhas de tratamento nas funções pré-existentes, com tratamentos que deem suporte para que o AFD consiga reconhecer qualquer palavra de um determinado alfabeto.

Por fim, como um outro tipo de autômato foi implementado, a funcionalidade de permitir múltiplos autômatos também foi implementada. Dessa forma, é possível informar na primeira linha do arquivo o tipo autômato que será utilizado, sendo "@AFD" para autômatos finitos determinísticos e "@AP" para autômatos de pilha.

4. Resultados

A Figura 4 abaixo ilustra os resultados obtidos para o arquivo de entrada "linguagemTeste.txt". No teste, foi utilizada a funcionalidade de alfabeto arbitrário. É possível observar que os três primeiros casos foram reconhecidos e os três últimos casos foram rejeitados, haja visto que são os casos de testes descritos na Figura 3. Neste caso, foi utilizada a funcionalidade de alfabeto arbitrário.

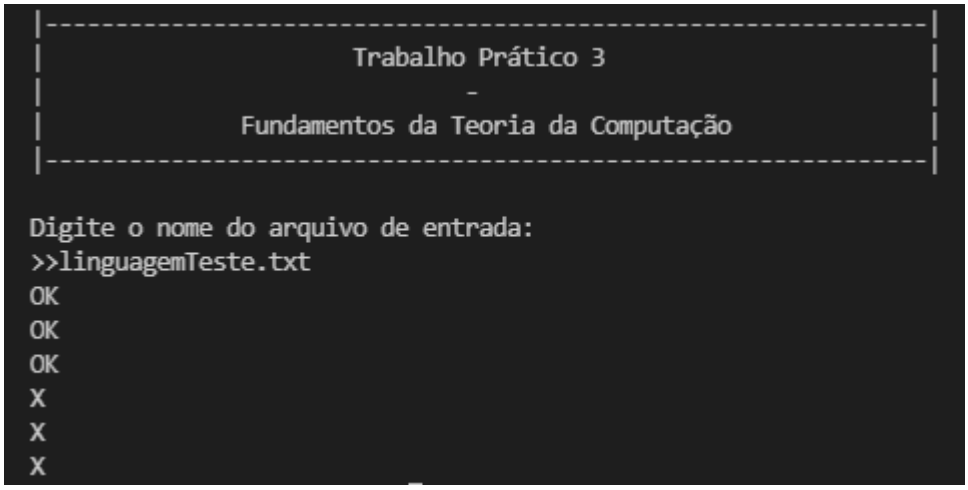

Figura 4. Resultado com a entrada "linguagemTeste.txt" que utiliza a funcionalidade de alfabeto arbitrário

A Figura 5 abaixo ilustra a entrada utilizada na funcionalidade autômato de pilha. Os casos de teste para essa funcionalidade foram definidos como: "ab", "aab", "bba", "aba" " " e "aabb".

```
@AP
Q: pri seg
S: ab\
G: Y
I: pri
F: pri seg
pri -> pri | a,\/Y
pri -> seg | b,Y/\
seg -> seg | b,Y/\
---
ab
aab
bba
aba

aabb
```

Figura 5. Entradas para a funcionalidade autômato de pilha

A Figura 6 abaixo ilustra os resultados obtidos para o arquivo de entrada “ap1.txt”.No teste, foram utilizadas as entradas da Figura 4. É possível observar que o primeiro caso (“ab”) foi reconhecido, os três casos posteriores (“aab”, “bba”, “aba”) foram rejeitados e os dois últimos (“ “ e “aabb”) também foram reconhecidos.

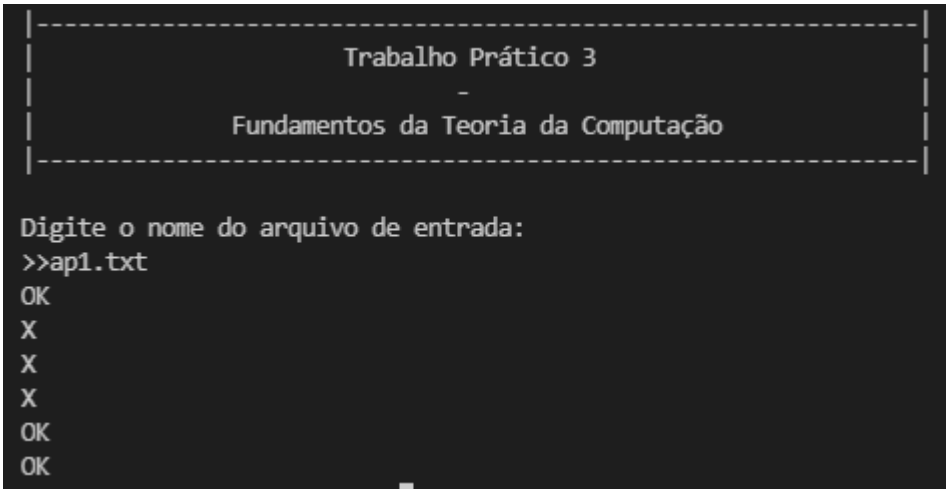

Figura 6. Resultado da entrada para a funcionalidade autômato de pilha

5. Conclusão

Neste trabalho prático, foi explorado um tema de grande importância para o estudo da teoria da computação, foi desenvolvido um programa capaz de, a partir de uma determinada entrada configurar um

autômato finito determinístico (AFD), e logo em seguida utilizá-lo para aceitar ou rejeitar determinadas palavras.

Alguns desafios foram encontrados durante o desenvolvimento desta tarefa, como como seria feita a implementação do trabalho prático, e como seria representada as máquinas, mas nada complexivo, foi um impasse, mas tivemos um entendimento rápido de como fazer esta tarefa.

Portanto, esta abordagem mais prática da implementação de um AFD, melhorou ainda mais o conhecimento adquirido em aula dos autômatos finitos determinísticos, pois fez com que conceitos apresentados durante a disciplina fossem utilizados para a implementação de tal autômato.

6. Referências

RAMOS, Marcus Vinícius Midena. *Linguagens Formais e Autômatos*. Apostila de “Linguagens Formais - Teoria, Modelagem e Implementação”. Disponível em: <http://www2.fct.unesp.br/docentes/dmec/olivete/lfa/arquivos/Apostila.pdf>. Acesso em 19 de Outubro de 2021.

Vieira, Newton José; **Introdução aos Fundamentos da Computação:** linguagens e máquinas. São Paulo: Pioneira Thomson Learning, 2006. 334p.