RELATÓRIO

T1: Fractais em máquina de Mealy

Professora: Karina Girardi Roggia
Alunos: Heitor Henrique Klein e Leonardo Floriani Monteiro

20/10/2025
Universidade do Estado de Santa Catarina - UDESC
Linguagens Formais e Autômatos

UDESC UNIVERSIDADE DO ESTADO DE SANTA CATARINA

SUMÁRIO

UDESC

1. Instruções de Compilação e Execução

Esta seção detalha os passos necessários para configurar o ambiente e executar o simulador da Máquina de Mealy.

1.1 Pré-requisitos

O único pré-requisito para executar o simulador é ter o interpretador **Python 3** instalado no sistema. A aplicação foi desenvolvida utilizando Python 3 e não depende de bibliotecas externas.

- Caso não o tenha instalado, o Python pode ser baixado gratuitamente em seu site oficial: https://www.python.org.

Por se tratar de uma aplicação em Python, uma linguagem interpretada, não há um passo de compilação explícito. O código-fonte é executado diretamente pelo interpretador.

1.2. Configuração do Ambiente

Para a correta execução, os seguintes arquivos devem estar localizados no **mesmo diretório**:

1. O script do simulador (o arquivo .py).
2. O arquivo de texto (.txt) com a descrição da Máquina de Mealy.
3. O arquivo de texto (.txt) com a palavra de entrada a ser processada.

1.3. Execução do Programa

O programa é executado via linha de comando e interage com o usuário para obter os nomes dos arquivos necessários. Siga os passos abaixo:

1. Abra um terminal (no Linux/macOS) ou um Prompt de Comando/PowerShell (no Windows).

2. Navegue até o diretório que contém os arquivos do projeto.

```
# Exemplo de comando para navegar até a pasta
cd /caminho/para/a/pasta/do/projeto
```

3. Execute o script utilizando o interpretador python3.

```
python3 simulador.py
```

*(**Nota**: Dependendo da configuração do sistema, o comando pode ser apenas python em vez de python3)*.

4. O programa solicitará interativamente os nomes dos arquivos. Insira o nome de cada arquivo conforme solicitado e pressione Enter.

1.4. Exemplo de Uso

Abaixo, um exemplo completo de uma sessão de execução no terminal, assumindo que os arquivos se chamam simulador.py, m1.txt e w.txt.

```
  # Passo 3: Executar o script
$ python3 simulador.py

  # Passo 4: Interagir com o programa
=== Simulador de Máquina de Mealy com saída PPM ===
Digite o nome do arquivo da máquina (ex: m1.txt): m1.txt
Digite o nome do arquivo de entrada (ex: w16.txt): w.txt
Digite o nome do arquivo de saída PPM (ex: saida.ppm): fractal_saida.ppm

  # Mensagem de sucesso
✅ Imagem PPM gerada com sucesso em 'fractal_saida.ppm'.
```

Após a execução, um novo arquivo chamado `fractal_saida.ppm` será criado no mesmo diretório, contendo o fractal gerado.

1.5. Execução em Ambientes Online

O simulador também pode ser executado em plataformas online como o **Google Colab** ou **Online GDB Compiler**. Para isso, é necessário primeiro fazer o upload do script `.py` e dos dois arquivos de entrada `.txt` para o ambiente da plataforma antes de executar o código.

1.6. Ambiente de Desenvolvimento

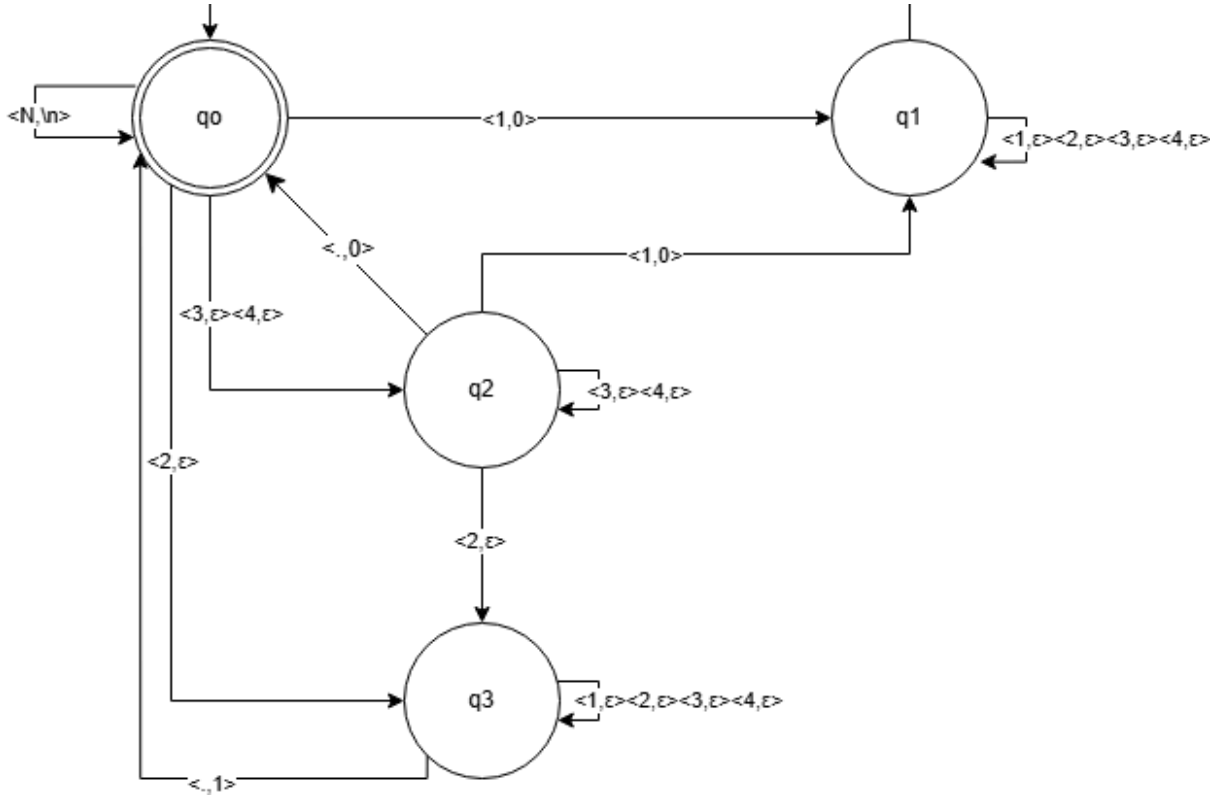
A aplicação foi desenvolvida e testada utilizando as seguintes ferramentas:

* Editor de Código: Microsoft Visual Studio Code
* Linguagem: Python 3.10
* Sistema Operacional: A portabilidade foi verificada, garantindo a execução em ambientes Windows e Linux (como o **Ubuntu 24.04 LTS**, especificado nos requisitos do trabalho).

2. Modelagem das expressões regulares

2.1 Expressão (3+4)*2(1+2+3+4)*

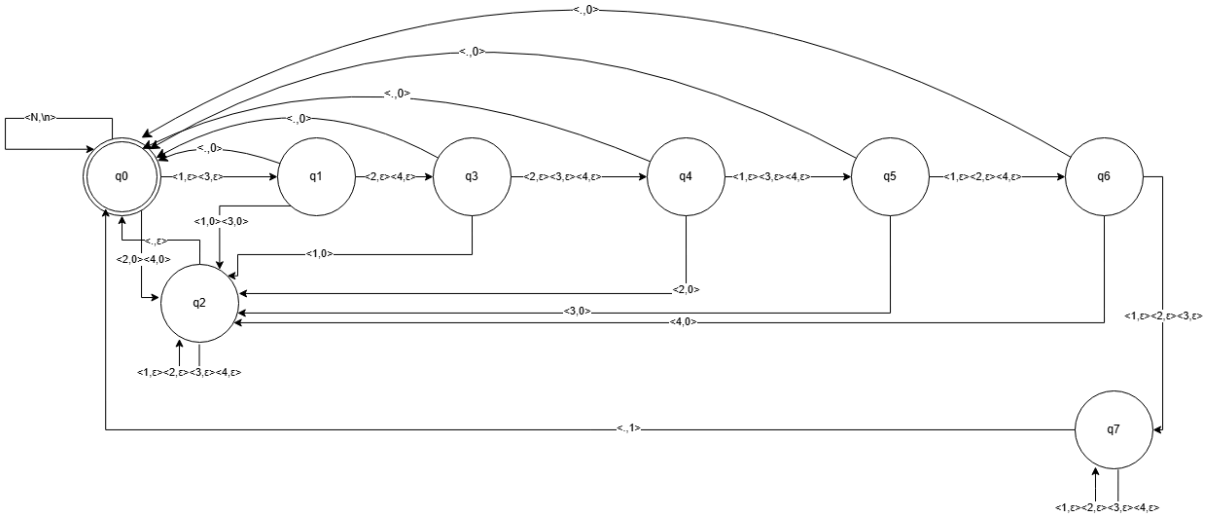
Máquina 1:




```
q0 q1 q2 q3
q0
q0
1 2 3 4 . N
0 1 \n
q0 N q0 \n
q0 1 q1 0
q0 2 q3 e
q0 3 q2 e
q0 4 q2 e
q1 1 q1 e
q1 2 q1 e
q1 3 q1 e
q1 4 q1 e
q1 . q0 e
q2 1 q1 0
q2 2 q3 e
q2 3 q2 e
q2 4 q2 e
q2 . q0 0
q3 1 q3 e
q3 2 q3 e
q3 3 q3 e
q3 4 q3 e
q3 . q0 1
```

2.2 Expressão (1+3)(2+4)(2+3+4)(1+3+4)(1+2+4)(1+2+3)(1+2+3+4)*

Máquina 2:

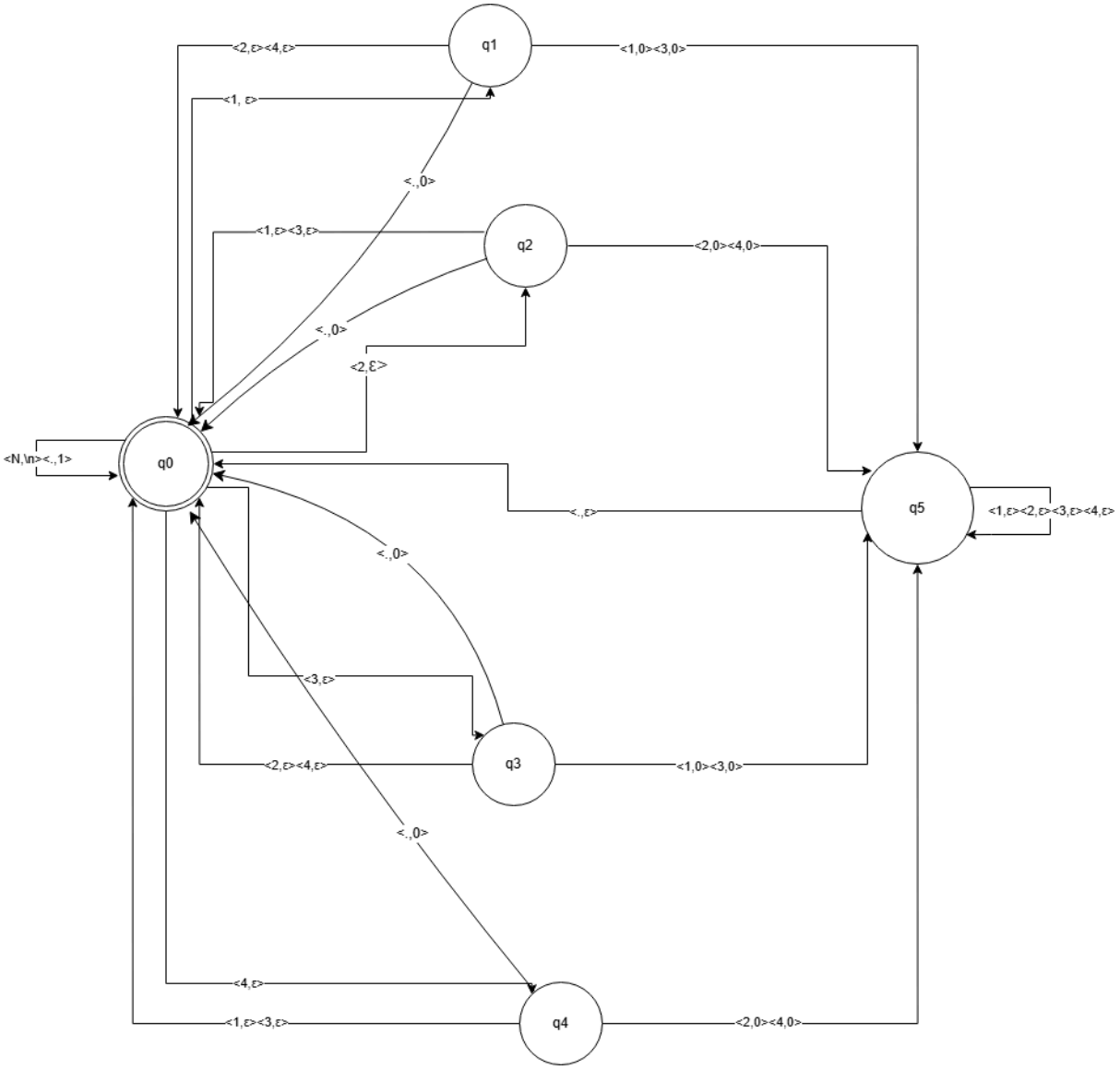



```
q0 q1 q2 q3 q4 q5 q6 q7
q0
q0
1 2 3 4 . N
0 1 \n
q0 N q0 \n
q0 1 q1 e
q0 2 q2 0
q0 3 q1 e
q0 4 q2 0
q1 1 q2 0
q1 2 q3 e
q1 3 q2 0
q1 4 q3 e
q1 . q0 0
q2 1 q2 e
q2 2 q2 e
q2 3 q2 e
q2 4 q2 e
q2 . q0 e
q3 1 q2 0
q3 2 q4 e
q3 3 q4 e
q3 4 q4 e
q3 . q0 0
q4 1 q5 e
q4 2 q2 0
q4 3 q5 e
q4 4 q5 e
q4 . q0 0
q5 1 q6 e
q5 2 q6 e
q5 3 q2 0
q5 4 q6 e
q5 . q0 0
q6 1 q7 e
q6 2 q7 e
q6 3 q7 e
q6 4 q2 0
q6 . q0 0
q7 1 q7 e
q7 2 q7 e
q7 3 q7 e
q7 4 q7 e
q7 . q0 1
```

UDESC

2.3 Expressão (12+14+32+34+21+23+41+43)*

Máquina 3:



```
q0 q1 q2 q3 q4 q5
q0
q0
1 2 3 4 . N
0 1 \n
q0 N q0 \n
q0 1 q1 e
q0 2 q2 e
q0 3 q3 e
q0 4 q4 e
q0 . q0 1
q1 1 q5 0
q1 2 q0 e
q1 3 q5 0
q1 4 q0 e
q1 . q0 0
q2 1 q0 e
q2 2 q5 0
q2 3 q0 e
q2 4 q5 0
q2 . q0 0
q3 1 q5 0
q3 2 q0 e
q3 3 q5 0
q3 4 q0 e
q3 . q0 0
q4 1 q0 e
q4 2 q5 0
q4 3 q0 e
q4 4 q5 0
q4 . q0 0
q5 1 q5 e
q5 2 q5 e
q5 3 q5 e
q5 4 q5 e
q5 . q0 e
```

3. Resultados de saídas

3.1 Saída para M1, utilizando w8.txt (disponibilizado no moodle).

```
P1
8 8
1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0
1 1 0 0 1 1 0 0
1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0
```

Nesse caso, as células possuem tamanho 3 e o elemento central deve ser 2.

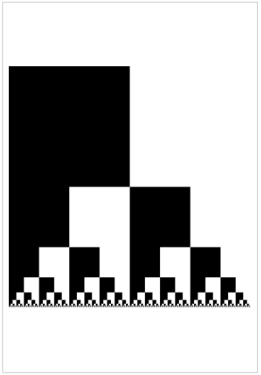
 Imagem gerada.

3.1.2 Saída para M1, utilizando w16.txt (disponibilizado no moodle).

```
P1
16 16
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

 Imagem gerada

3.1.3 Saída para M1, utilizando w512.txt (disponibilizado no moodle).
(Omitido o texto do ppm, está disponível na pasta do trabalho e no repositório github)



Aqui temos células de tamanho 9.

3.2 Saída para M2, utilizando w8.txt (disponibilizado no moodle).

P1
8 8
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

Vale ressaltar que todas as células tinham tamanho 3, por isso nenhuma poderia ser aceita pela expressão/máquina. Então obtemos uma matriz binária zerada.

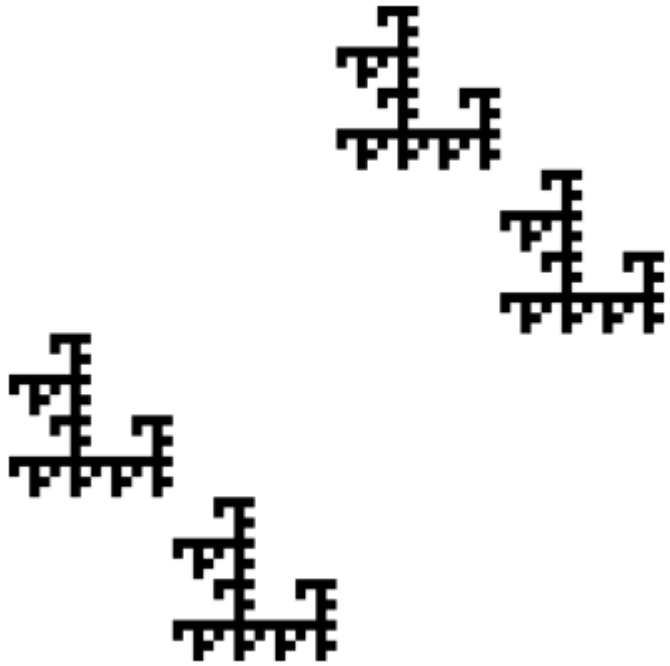
3.2.2 Saída para M2, utilizando w16.txt (disponibilizado no moodle).

P1
16 16
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Novamente, vale ressaltar que a matriz binária é zerada pois nenhuma célula de w possui o tamanho adequado. Como o arquivo w é “w16” , temos células de tamanho log(16) na base 2 = 4.

3.2.3 Saída para M2, utilizando w512.txt (disponibilizado no moodle).
(Omitido o texto do ppm, está disponível na pasta do trabalho e no repositório github)



3.3.1 Saída para M3, utilizando w8.txt (disponibilizado no moodle).

```
P1
8 8
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

Todas as células do “w8” são de tamanho 3. Como a expressão só aceita palavras de tamanho par, o resultado matriz “zerada” era previsto. O fractal não é visível pois será completamente branco.

3.3.2 Saída para M3, utilizando w16.txt (disponibilizado no moodle).

```
P1
16 16
0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0
0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
1 0 0 1 0 0 0 0 0 0 0 1 0 0 1
1 0 0 1 0 0 0 0 0 0 0 1 0 0 1
0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
1 0 0 1 0 0 0 0 0 0 0 1 0 0 1
1 0 0 1 0 0 0 0 0 0 0 1 0 0 1
0 1 1 0 0 0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0
0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0
```

Como todas as células nesse caso tem tamanho 4 (par), algumas células são aceitas e é gerado um fractal com círculos.

Fractal gerado:

3.3.3 Saída para M3, utilizando w512.txt (disponibilizado no moodle).
(Omitido o texto do ppm, está disponível na pasta do trabalho e no repositório github)

A matriz de saída da máquina é composta somente de zeros, pois o tamanho das palavras em w512.txt é ímpar (tamanho 9).

4. Algoritmo leitor de Máquinas:

O programa é dividido em três funções principais e uma função main:

1 - cria_mealy_maquina(file_path):

Lê um arquivo de texto (linha por linha) e estrurtua a Máquina de Mealy no formato do dicinário do Python. As linhas finais referentes as transições são tratadas no formato: estado_atual, símbolo_entrada, próximo_estado e símbolo_saída. Os símbolos de saída “\n” (literal no arquivo) e “e” (que representa a string vazia) são convertidos para o caractere de nova linha e uma string vazia, respectivamente, para que possam ser usados corretamente na simulação.

2 - simula(maquina, palavra_entrada):

Executa a máquina com uma palavra de entrada e gera a saída. Começa no “estado_atual” definido como o “initial_estado”. A “palavra_saida” é inicializada como vazia. Cada símbolo da entrada (palavra_entrada) é verificado por meio de uma comparação. A verificação é se o símbolo pertence ou não ao alfabeto de entrada. Após isso ocorrer, a tabela de transições é consultada para o par (estado_atual, symbol). Então, a transição retorna o próximo estado e o símbolo de saída. O “simbolo_saida” é anexado à “palavra_saida” e o “estado_atual” é atualizado para o “próximo_estado”. Finalmente, retorna a “palavra_saida” gerada pela máquina.

3 - gera_ppm(saida_matrix_str, arq_saida_path):

Converte a saída para o formato PPM. Conforme orientação da professora, usamos P1. A “string_saida” é então tratada como uma matriz onde cada linha na string representa uma linha de pixels da nossa imagem. Escreve-se então o identificador de formato (P1) e as

dimensões. A matriz é percorrida e cada caractere na linha da string de saída é escrito no arquivo (separados por espaços).

Main:

A Main ficou encarregada de gerenciar o processo solicitando informações do usuário e chamando os métodos listados. Importante ressaltar que no final é exibida uma mensagem de êxito ou fracasso.

Extra - link repositório: https://github.com/heitork1/LFA/tree/master