Trabalho Prático: Construção de uma API REST Utilizando a Biblioteca Automata

Esta API REST foi desenvolvida para manipular e analisar autômatos, conectando os conceitos teóricos da Teoria da Computação com uma aplicação prática em Ciência da Computação e está disponível em https://github.com/Vicius1/automata-api. A API permite a criação, consulta, teste e visualização gráfica de diversos tipos de autômatos, incluindo:

- Autômatos Finitos Determinísticos (AFD)

- Autômatos Finitos Não Determinísticos (NFA)

- Autômatos com Pilha Determinísticos (DPDA)

- Autômatos com Pilha Não Determinísticos (NPDA)

- Máquinas de Turing (TM)

- Máquinas de Turing Não Determinísticas (NTM)

- Máquinas de Turing Multitape Não Determinísticas (MNTM)

O projeto utiliza a biblioteca Automata para a manipulação dos autômatos e o framework FastAPI para expor a API REST. Um frontend básico, desenvolvido com HTML, CSS, JavaScript e Bootstrap, também foi implementado para facilitar a interação com a API.

Como Configurar e Executar o Projeto

Pré-requisitos:

Python 3.8 ou superior.

Pip.

(Opcional) Ambiente virtual (por exemplo, venv).

Passos para configuração:

Clone o repositório:

git clone https://github.com/Vicius1/automata-api.git

Entre no diretório: cd automata-api

Crie e ative um ambiente virtual (recomendado):

No Linux/macOS:

python -m venv venv

source venv/bin/activate

No Windows:

python -m venv venv

venv\\Scripts\\activate

Instale as dependências:

pip install fastapi uvicorn pydot

Instale a biblioteca Automata:

pip install git+https://github.com/caleb531/automata.git

Certifique-se de que o Graphviz esteja instalado e configurado no sistema.

Inicie o servidor:

Execute: uvicorn main:app --reload

A API ficará disponível em: http://127.0.0.1:8000

Acesse a documentação interativa (Swagger UI) em: http://127.0.0.1:8000/docs

Exemplos de Uso da API

A API possui endpoints para criação, consulta, teste e visualização gráfica de cada tipo de autômato. A seguir, são apresentados alguns exemplos de configuração e entradas de teste para cada categoria:

AFD (Autômato Finito Determinístico):

```
{
  "states": ["q0", "q1"],
  "input_symbols": ["0", "1"],
  "transitions": {
    "q0": {
      "0": "q1",
```

```
    "1": "q0"
   },
   "q1": {
    "0": "q1",
    "1": "q0"
   }
  },
  "initial_state": "q0",
  "final_states": ["q1"]
}
```

Entrada teste: “1010”

Saída esperada: “Entrada aceita”

NFA (Autômato Finito Não Determinístico)

```
{
  "states": ["q0", "q1", "q2"],
  "input_symbols": ["a", "b"],
  "transitions": {
   "q0": {
    "a": ["q0", "q1"],
    "b": ["q0"]
   },
   "q1": {
    "b": ["q2"]
   },
   "q2": {}
  },
```

```
 "initial_state": "q0",
 "final_states": ["q2"]
}
```

Entrada teste: “abab”

Saída esperada: “Entrada aceita”

DPDA (Autômato com Pilha Determinístico)

```
{
 "states": ["q0", "q1", "q2", "q3"],
 "input_symbols": ["a", "b"],
 "stack_symbols": ["0", "1"],
 "transitions": {
  "q0": {
   "a": {"0": ["q1", "1", "0"]}
  },
  "q1": {
   "a": {"1": ["q1", "1", "1"]},
   "b": {"1": ["q2", ""]}
  },
  "q2": {
   "b": {"1": ["q2", ""]},
   "": {"0": ["q3", ""]}
  }
 },
 "initial_state": "q0",
 "initial_stack_symbol": "0",
```

```
 "final_states": ["q3"],
 "acceptance_mode": "final_state"
}
```

Entrada teste: “aabb”

Saída esperada: “Entrada aceita”

TM (Máquina de Turing)

```
{
 "states": ["q0", "q1", "q2", "q1a", "q2a", "q5", "q_accept", "q_reject"],
 "input_symbols": ["0", "1"],
 "tape_symbols": ["0", "1", "X", "_"],
 "transitions": {
  "q0": {
   "0": ["q1", "X", "R"],
   "1": ["q2", "X", "R"],
   "X": ["q0", "X", "R"],
   "": ["q_accept", "", "S"]
  },
  "q1": {
   "0": ["q1", "0", "R"],
   "1": ["q1", "1", "R"],
   "X": ["q1", "X", "R"],
   "": ["q1a", "", "L"]
  },
  "q2": {
   "0": ["q2", "0", "R"],
```

```
    "1": ["q2", "1", "R"],
    "X": ["q2", "X", "R"],
    "": ["q2a", "", "L"]
   },
   "q1a": {
    "X": ["q1a", "X", "L"],
    "0": ["q5", "X", "L"],
    "1": ["q_reject", "1", "S"],
    "": ["q_reject", "", "S"]
   },
   "q2a": {
    "X": ["q2a", "X", "L"],
    "1": ["q5", "X", "L"],
    "0": ["q_reject", "0", "S"],
    "": ["q_reject", "", "S"]
   },
   "q5": {
    "0": ["q5", "0", "L"],
    "1": ["q5", "1", "L"],
    "X": ["q5", "X", "L"],
    "": ["q0", "", "R"]
   }
  },
  "initial_state": "q0",
  "blank_symbol": "_",
  "final_states": ["q_accept"]
}
```

Entrada teste: ”0110”

Saída esperada: “Entrada aceita”

NPDA (Autômato com Pilha Não Determinístico)

```
{
  "states": ["q0", "q1", "q2"],
  "input_symbols": ["a", "b"],
  "stack_symbols": ["A", "B", "#"],
  "transitions": {
    "q0": {
      "": {
        "#": [ ["q2", "#"] ]
      },
      "a": {
        "#": [ ["q0", "A", "#"] ],
        "A": [ ["q0", "A", "A"], ["q1", ""] ],
        "B": [ ["q0", "A", "B"] ]
      },
      "b": {
        "#": [ ["q0", "B", "#"] ],
        "A": [ ["q0", "B", "A"] ],
        "B": [ ["q0", "B", "B"], ["q1", ""] ]
      }
    },
    "q1": {
      "": {
```

```
      "#": [ ["q2", "#"] ]
    },
    "a": {
      "A": [ ["q1", ""] ]
    },
    "b": {
      "B": [ ["q1", ""] ]
    }
   }
  },
  "initial_state": "q0",
  "initial_stack_symbol": "#",
  "final_states": ["q2"],
  "acceptance_mode": "final_state"
}
```

Entrada teste: “abba”

Saída esperada: “Entrada aceita”

NTM (Máquina de Turing Não Determinística)

```
{
  "states": ["q0", "q1", "q2", "q3", "q4"],
  "input_symbols": ["0", "1"],
  "tape_symbols": ["0", "1", "x", "."],
  "transitions": {
    "q0": {
      "0": [ ["q1", "x", "R"] ],
```

```
    "x": [ ["q3", "x", "R"] ]
   },
   "q1": {
    "0": [ ["q1", "0", "R"] ],
    "1": [ ["q2", "x", "L"] ],
    "x": [ ["q1", "x", "R"] ]
   },
   "q2": {
    "0": [ ["q2", "0", "L"] ],
    "x": [ ["q0", "x", "R"] ],
    "1": [ ["q2", "1", "L"] ]
   },
   "q3": {
    "x": [ ["q3", "x", "R"] ],
    ":": [ ["q4", ":", "R"] ]
   }
 },
 "initial_state": "q0",
 "blank_symbol": ":",
 "final_states": ["q4"]
}
```


Entrada teste: “0011”

Saída esperada: “Entrada aceita”

MNTM (Máquina de Turing Multitape Não Determinística)


```
{
```

```
 "states": ["q0", "q1"],
 "input_symbols": ["0", "1"],
 "tape_symbols": ["0", "1", "#"],
 "n_tapes": 2,
 "transitions": {
  "q0": {
   "1,#": [
    ["q0", [ ["1", "R"], ["1", "R"] ]]
   ],
   "0,#": [
    ["q0", [ ["0", "R"], ["#", "N"] ]]
   ],
   "#,#": [
    ["q1", [ ["#", "N"], ["#", "N"] ]]
   ]
  }
 },
 "initial_state": "q0",
 "blank_symbol": "#",
 "final_states": ["q1"]
}
```

Entrada teste: “1010”

Saída esperada: “Entrada aceita”

Limitações e Pressupostos

Os autômatos são armazenados em memória. Assim, reiniciar o servidor fará com que os autômatos criados sejam perdidos.

A API espera que o usuário insira configurações JSON compatíveis com os modelos definidos. A validação básica é feita com Pydantic, mas pressupõe-se que os dados estejam no formato correto para a biblioteca Automata.

Frontend

O projeto inclui um frontend simples desenvolvido com HTML, CSS, JavaScript e Bootstrap. Com ele, o usuário pode:

- Selecionar o tipo de autômato (AFD, NFA, DPDA, NPDA, TM, NTM, MNTM).

- Inserir a configuração do autômato em formato JSON para criação.

- Consultar os detalhes do autômato por ID.

- Testar uma entrada, recebendo a resposta se a cadeia é aceita ou rejeitada.

- Visualizar graficamente o autômato por meio de uma imagem PNG gerada pela API.

- Para usar o frontend, basta abrir o arquivo HTML no navegador (por exemplo, index.html) ou utilizar o endereço “127.0.0.1:8000/static/index.html” .