UNIVERSIDADE FEDERAL DE VIÇOSA
Campus Rio Paranaíba
Sistemas de Informação
Introdução a Teoria da Computação - SIN 131

RELATÓRIO FINAL SOBRE O PROJETO DA DISCIPLINA

6319 - Gabriel Savio de Lima Mota
6954 - Victor Alves de Oliveira

09/2024

UNIVERSIDADE FEDERAL DE VIÇOSA
Campus Rio Paranaíba
Sistemas de Informação
Introdução a Teoria da Computação - SIN 131

Sumário

UNIVERSIDADE FEDERAL DE VIÇOSA
Campus Rio Paranaíba
Sistemas de Informação
Introdução a Teoria da Computação - SIN 131

1. Sobre o Projeto

Este projeto implementa um sistema para manipular Autômatos Finitos Determinísticos (AFD) e Autômatos Finitos Não-determinísticos (AFN). Ele permite a conversão, simulação e minimização de autômatos, demonstrando a equivalência entre AFN e AFD, assim como a manipulação de uma Máquina de Turing (MT), possibilitando a verificação de uma palavra em uma determinada linguagem.

2. Tecnologias Utilizadas

- Visual Studio Code = IDE e Editor de Código
- Git = Versionamento de Código
- Github = Repositório do Projeto
- Python 3.12 = Linguagem de Programação
- Streamlit = Biblioteca para visualização de dados na web
- Graphviz = Biblioteca gráfica que permite a manipulação e processamento de grafos.

3. Como Executar o Projeto

Consulte a execução do projeto atualizada no arquivo Readme.md no repositório: https://github.com/VictorAlves08/Teoria_da_Computacao-AFD_AFN

4. Funcionalidades

- Criar AFD
- Criar AFN
- Converter AFN para AFD
- Minimizar AFD
- Verificar Equivalência
- Testar Palavras
- Reconhecimento de palavras pela Máquina de Turing

5. Arquitetura do Projeto

Consulte a arquitetura do projeto atualizada no arquivo Readme.md no repositório: https://github.com/VictorAlves08/Teoria_da_Computacao-AFD_AFN

6. Explicação das Funcionalidades

Antes de entendermos cada funcionalidade em seus detalhes, é importante compreender algumas funções utilitárias do projeto que antecedem a chamada para as funções que serão explicadas nos tópicos subsequentes.

No inicio da execução do projeto, o arquivo ***automata_app.py*** *(localizado dentro da pasta frontend)* é executado, contendo nele algumas funções importantes como:

- ***show_examples(self)*** **=** Responsavel por facilitar o acesso do usuários aos exemplos de automatos, AFN e/ou AFD.
- ***input_fields(self, label="")*** **=** Coleta informações sobre estados, alfabeto, estado inicial, estados finais e transições de um autômato a partir de entradas de texto e as retorna como listas e strings.
- ***route(self)*** **=** Executa a função correspondente à escolha do usuário, como criar autômatos, converter, minimizar, testar palavras, ou verificar equivalência entre autômatos.
- ***render_and_display_automaton(self, automaton)*** **=** Renderiza e exibe visualmente o autômato na interface usando Graphviz, permitindo ao usuário visualizar sua estrutura.

6.1 Criar AFD

A função de criar um ***AFD*** começa com a chamada ***create_automaton***. Nessa função, o usuário é solicitado a fornecer detalhes sobre o autômato, como estados, alfabeto, estado inicial, estados finais e transições. Após coletar essas informações, a função verifica se o botão "Criar" foi clicado e, se sim, cria uma instância de ***AFD*** usando os dados fornecidos. Em seguida, o autômato é renderizado e exibido na interface.

O ***AFD*** é uma classe que representa um autômato finito determinístico. Ela é inicializada com os estados, alfabeto, estado inicial, estados finais e transições, e começa com o estado inicial definido. A função ***accepts*** verifica se uma palavra é aceita pelo ***AFD***, seguindo as transições e verificando se o estado final é alcançado. A função ***run*** executa o autômato em uma entrada, atualizando o estado atual conforme as transições e retornando se o estado final foi alcançado. As funções ***get_next_states, get_transitions, get_states, get_alphabet, get_initial_state, get_final_states, get_current_state, set_current_state e set_current_transition*** fornecem acesso e manipulação dos diferentes aspectos do autômato, como estados, alfabeto, transições e estado atual.

6.2 Criar AFN

A função de criar um **AFN** começa com a chamada ***create_automaton***. Nessa função, o usuário é solicitado a fornecer detalhes sobre o autômato, como estados, alfabeto, estado inicial, estados finais e transições. Após coletar essas informações, a função verifica se o botão "Criar" foi clicado e, se sim, cria uma instância de ***AFN*** usando os dados fornecidos. Em seguida, o autômato é renderizado e exibido na interface.

O ***AFN*** é uma classe que representa um autômato finito determinístico. Ela é inicializada com os estados, alfabeto, estado inicial, estados finais e transições, e começa com o estado inicial definido. A função ***accepts*** verifica se uma palavra é aceita pelo ***AFN***, seguindo as transições e verificando se o estado final é alcançado. A função ***run*** executa o autômato em uma entrada, atualizando o estado atual conforme as transições e retornando se o estado final foi alcançado. As funções ***get_next_states, get_transitions, get_states, get_alphabet, get_initial_state, get_final_states, get_current_state, set_current_state e set_current_transition*** fornecem acesso e manipulação dos diferentes aspectos do autômato, como estados, alfabeto, transições e estado atual.

6.3 Converter AFN para AFD

A função que converte um ***AFN*** em um ***AFD*** começa com a chamada ***convert_afn_to_afd***. Nessa função, o usuário é solicitado a inserir os detalhes do ***AFN***, como estados, alfabeto, estado inicial, estados finais e transições. Após receber esses dados, a função verifica se o botão "Converter" foi clicado e, em caso afirmativo, cria uma instância do ***AFN*** usando as informações fornecidas e, em seguida, chama um método para realizar a conversão desse ***AFN*** em um ***AFD***. O ***AFD*** resultante é então renderizado e exibido.

O processo de conversão de ***AFN*** para ***AFD*** é detalhado na função ***afn_to_afd***. Primeiramente, os estados, transições e mapeamentos do novo ***AFD*** são inicializados. O estado inicial do ***AFN*** é convertido em uma tupla ordenada, e uma fila de estados a serem processados é criada, começando com esse estado inicial. A conversão funciona processando cada estado na fila: para cada símbolo do alfabeto, a função determina quais estados podem ser alcançados a partir dos estados atuais no ***AFN***. Esses estados de destino são combinados em novos estados no ***AFD***, que são então adicionados às transições do ***AFD***. Esse processo continua até que todos os estados possíveis tenham sido processados. Durante a conversão, a função ***parse_transitions*** é usada para transformar a entrada de transições do

usuário em um formato que pode ser manipulado, separando as transições do ***AFN*** e organizando-as em um dicionário de estados e símbolos.

Finalmente, o ***AFD*** resultante é retornado, representando uma versão determinística do ***AFN*** original, com novos estados, transições e definições de estado final baseadas na combinação dos estados do ***AFN***.

6.4 Minimizar AFD

A função ***minimize_afd*** começa pedindo ao usuário detalhes do ***AFD***, como estados, alfabeto, estado inicial, estados finais e transições. Se o botão "Minimizar" for clicado, a função cria um ***AFD*** a partir desses dados e chama um método para minimizar o autômato. O ***AFD*** minimizado é então renderizado e exibido.

A minimização ocorre na função ***minimize_afd***, que inicialmente divide os estados em dois grupos: estados finais e não finais. A partir daí, o algoritmo refina essas partições, verificando quais estados podem ser diferenciados com base nas transições para símbolos do alfabeto. Esse processo continua até que não haja mais divisões possíveis, resultando em novos estados que agrupam os estados equivalentes.

Após o refinamento, um novo ***AFD*** é construído com menos estados e transições, mas que ainda reconhece a mesma linguagem. Este ***AFD*** minimizado é então retornado e exibido ao usuário.

6.5 Verificar Equivalência

A função ***check_equivalence*** verifica se dois autômatos são equivalentes, ou seja, se aceitam as mesmas palavras. Inicialmente, o usuário insere os detalhes de dois autômatos, como estados, alfabeto, estado inicial, estados finais e transições, para ambos. Em seguida, o usuário insere as palavras de teste que serão usadas para verificar a equivalência.

Se o botão "Verificar" for clicado, a função converte as entradas de transições em um formato manipulável e cria instâncias de ***AFNs*** para ambos os autômatos. A função então chama um método para comparar os dois autômatos usando as palavras de teste fornecidas. Dependendo do resultado, a interface exibe se os autômatos são equivalentes ou não, ou seja, se ambos aceitam as mesmas palavras de teste.

6.6 Testar Palavras

A função ***test_words*** permite ao usuário verificar se determinadas palavras são aceitas ou rejeitadas por um autômato, seja ele um ***AFD*** ou um ***AFN***. Inicialmente, o usuário insere os detalhes do autômato, como estados, alfabeto, estado inicial, estados finais e transições, além das palavras que deseja testar.

Quando o botão "Testar" é clicado, a função converte as transições em um formato manipulável e verifica se o autômato é ***AFD*** ou ***AFN***. Dependendo da estrutura das transições, a função cria a instância apropriada do autômato.

Em seguida, o autômato é renderizado e exibido na interface, e cada palavra de teste fornecida pelo usuário é avaliada para verificar se é aceita ou rejeitada. A função então exibe os resultados na tela, indicando para cada palavra se ela é aceita ou não pelo autômato.

6.7 Reconhecimento de palavras pela Máquina de Turing

- **Verificação de Palíndromos**

  A função ***palindrome_turing_machine*** permite ao usuário verificar se uma dada palavra é um palíndromo, ou seja, se pode ser lida da mesma forma da direita para a esquerda e vice-versa. Para isso, ela simula o funcionamento de uma ***Máquina de Turing***.

  Inicialmente, o usuário fornece os detalhes da Máquina de Turing, como os estados possíveis, o alfabeto utilizado, o estado inicial, os estados finais e as regras de transição. Além disso, o usuário insere a palavra que deseja verificar.

  Ao clicar em "Testar", a função converte as regras de transição para um formato interno mais fácil de processar. Em seguida, cria uma instância da Máquina de Turing com a configuração fornecida pelo usuário. A palavra a ser testada é então escrita na fita da máquina.

  A simulação da Máquina de Turing começa no estado inicial e segue as regras de transição, lendo o símbolo na fita, movendo a cabeça de leitura e escrevendo um novo símbolo na fita. Esse processo se repete até que a máquina alcance um estado final (indicando que a palavra é um palíndromo) ou até que não haja mais regras aplicáveis (indicando que a palavra não é um palíndromo). Finalmente, o resultado da simulação é exibido ao usuário, informando se a palavra de entrada é ou não um palíndromo.

- **Cópia de Cadeia de Caracteres**

  A função ***copy_string_turing_machine*** simula o processo de duplicar uma sequência de caracteres usando uma Máquina de Turing. Dado uma fita onde seja possivel escrever uma palavra e a máquina precisa criar uma cópia idêntica dessa palavra em outro lugar da fita.

  Para isso, o usuário define os detalhes da máquina: quais são os estados possíveis, quais símbolos podem aparecer na fita, como a máquina deve se comportar em cada situação (as transições) e qual é a palavra que se deseja copiar.Quando a simulação começa, a máquina lê o primeiro caractere da palavra, move-se para uma região livre da fita e escreve uma cópia desse caractere. Em seguida, ela volta para o início da palavra original e repete o processo para o próximo caractere, até que toda a palavra tenha sido copiada.

7. Exemplos de Execução

O tópico de exemplos de execução será contemplado da seguinte forma: Antes de iniciarmos os *bullet points*, informarei quais serão o AFD, AFN, palavras de testes e na Máquina de Turing utilizados nos exemplos, cada tópico terá o print de dois exemplos de execução, realizado no sistema.

Segue abaixo os AFDs, AFNs e as Palavras de Teste que serão utilizados:

- **Exemplo 1 para AFD**
  - Estados: q0, q1, q2, q3, q4, q5
  - Alfabeto: a, b
  - Estado inicial: q0
  - Estados finais: q3, q4
  - Transições: q0,a=q1
    q0,b=q5
    q1,a=q3
    q1,b=q2
    q5,a=q5
    q5,b=q5
    q3,a=q3
    q3,b=q2
    q2,b=q5
    q2,a=q4
    q4,a=q3
    q4,b=q2

- **Exemplo 2 para AFD**
  - Estados: q0,q1,q2
  - Alfabeto: a, b
  - Estado inicial: q0
  - Estados finais: q2
  - Transições: q0,a=q1
    q0,b=q0
    q1,a=q2
    q1,b=q1
    q2,b=q2

- **Exemplo 1 AFN**
  - Estados: q0, q1, q2, q3, q4, q5
  - Alfabeto: a, b
  - Estado inicial: q0
  - Estados finais: q3, q4
  - Transições: q0,a=q1
    q0,b=q5
    q1,a=q3
    q1,b=q2
    q5,a=q5
    q5,b=q5
    q3,a=q3
    q3,b=q2
    q2,b=q5
    q2,a=q4
    q4,a=q3
    q4,b=q2

- **Exemplo 2 AFN**
  - Estados: q0,q1,q2,q3,q4
  - Alfabeto: a, b
  - Estado inicial: q0
  - Estados finais: q2
  - Transições: q0,a=q1
    q0,b=q0
    q0,ε=q2
    q1,a=q3
    q1,b=q1
    q2,a=q3
    q2,ε=q4
    q3,b=q4

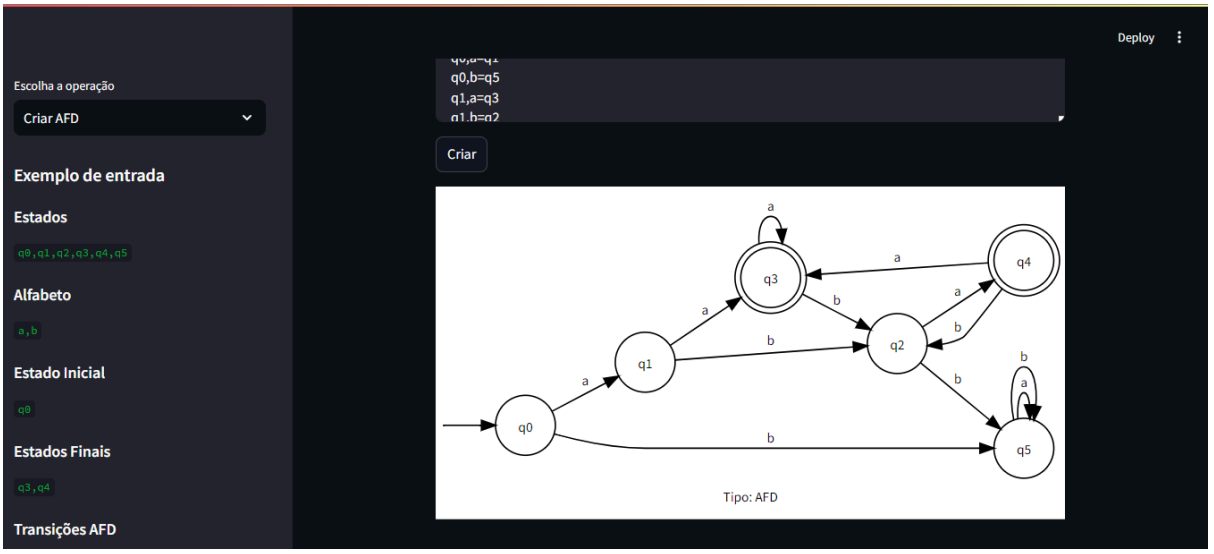
- **Palavras de Teste (AFD/AFN)**
  - bab, ab, ba, aaa, aba, aaba

- **Máquina de Turing**
  - **Exemplo 1 - Verificação de Palíndromos**
    - Estados: q0,q1,q2,q3,q4,qf
    - Alfabeto: a, b
    - Símbolo Branco: _
    - Estado inicial: q0
    - Estados finais: qf
    - Palavra de Entrada: abba
    - Transições: q0,a=(q1,_,R)
      q0,b=(q2,_,R)
      q0,_=(qf,_,R)
      q1,a=(q1,a,R)
      q1,b=(q1,b,R)
      q1,_=(q3,_,L)
      q2,a=(q2,a,R)
      q2,b=(q2,b,R)
      q2,_=(q4,_,L)
      q3,a=(q0,_,R)
      q3,_=(qf,_,R)
      q4,b=(q0,_,R)
      q4,_=(qf,_,R)

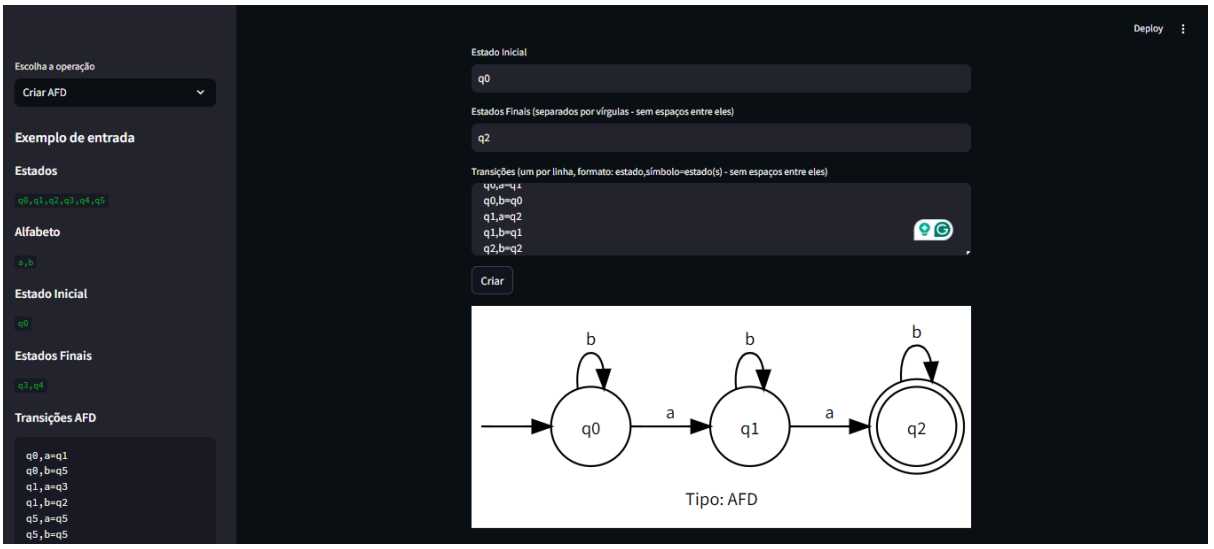
- **Exemplo 2 - Cópia de Cadeia de Caracteres**
  - Estados: q0,q1,q2,qf
  - Alfabeto: a, b
  - Símbolo Branco: _
  - Estado inicial: q0
  - Estados finais: qf
  - Palavra de Entrada: ab
  - Transições: q0,a=(q1,_,R)
    q1,_=(q2,a,L)
    q2,=(q0,_,R)
    q0,_=(qf,_,R)
    q1,b=(q1,b,R)
    q2,b=(q0,b,L)

7.1 Criar AFD

Entrada Executada: Exemplo 1 AFD

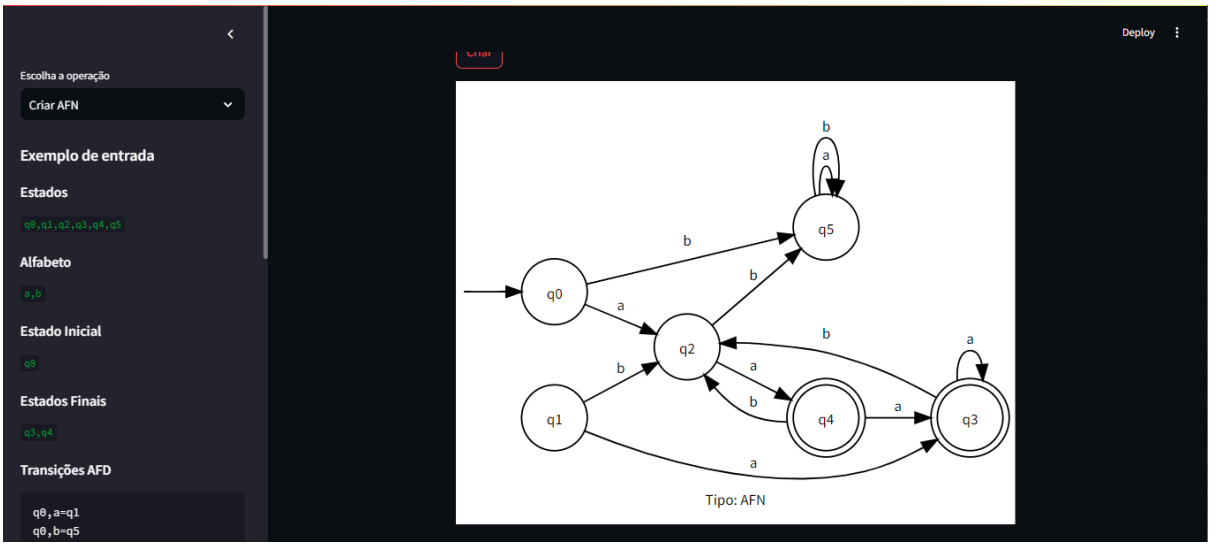


Entrada Executada: Exemplo 2 AFD

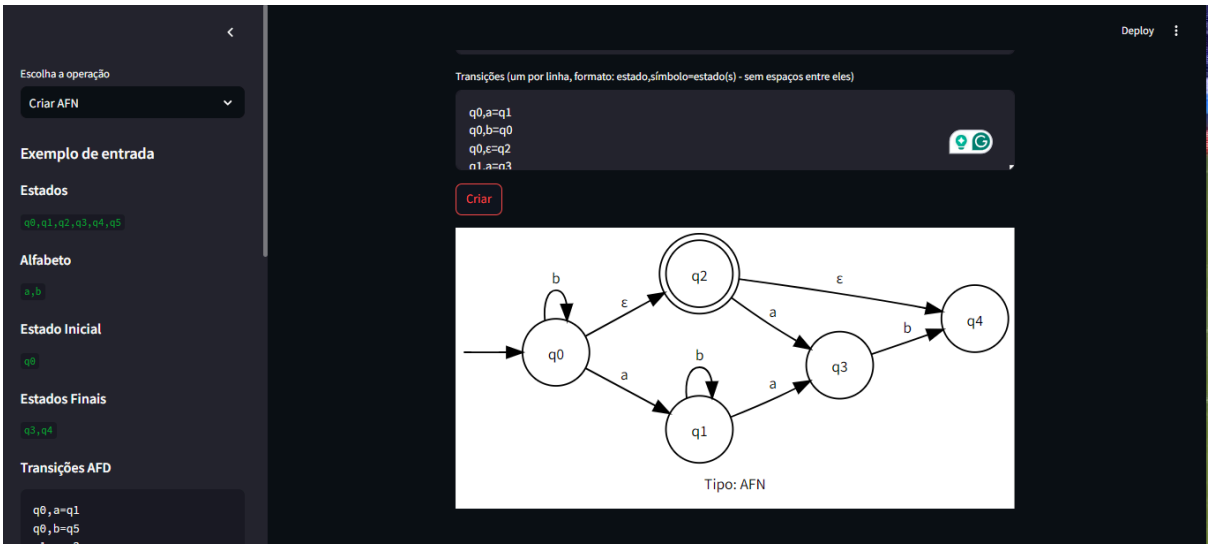


7.2 Criar AFN

Entrada Executada: Exemplo 1 AFN

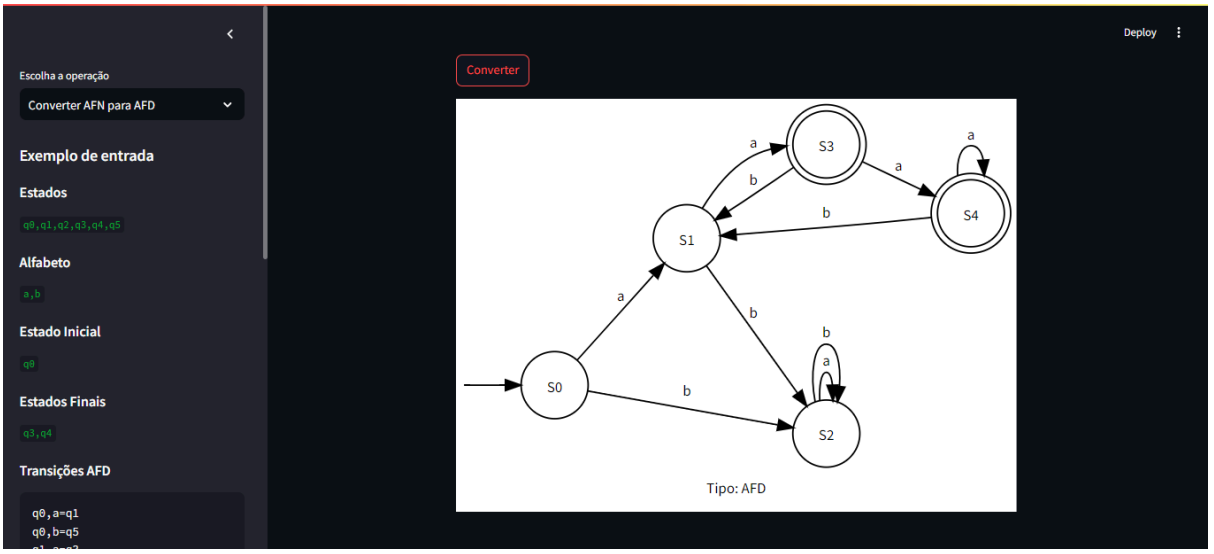


Entrada Executada: Exemplo 2 AFN

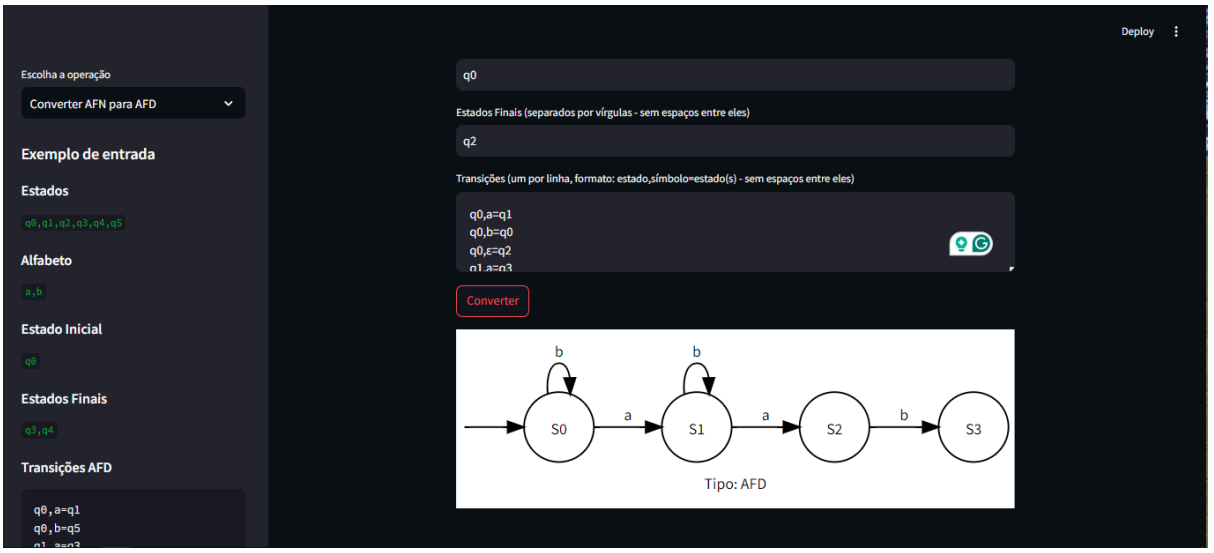


7.3 Converter AFN para AFD

Entrada Executada: Exemplo 1 AFN

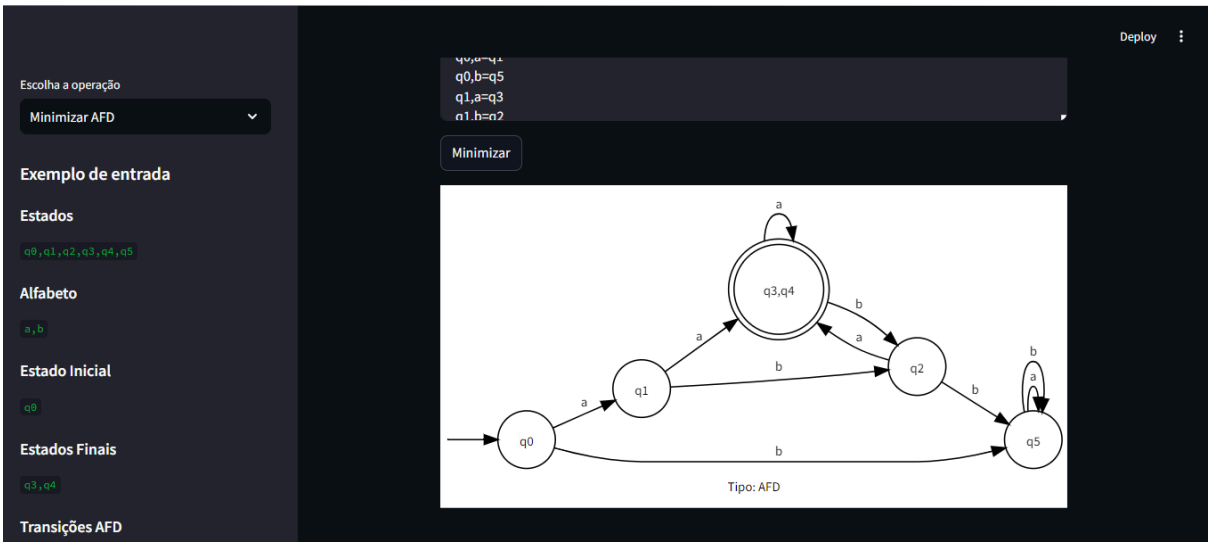


Entrada Executada: Exemplo 2 AFN

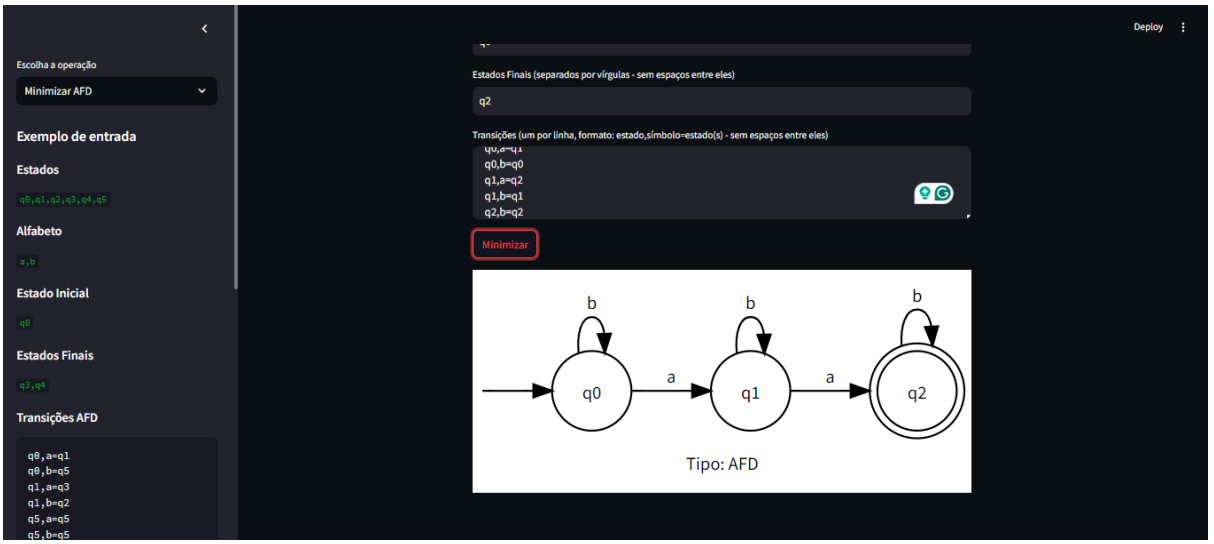


7.4 Minimizar AFD

Entrada Executada: Exemplo 1 AFD

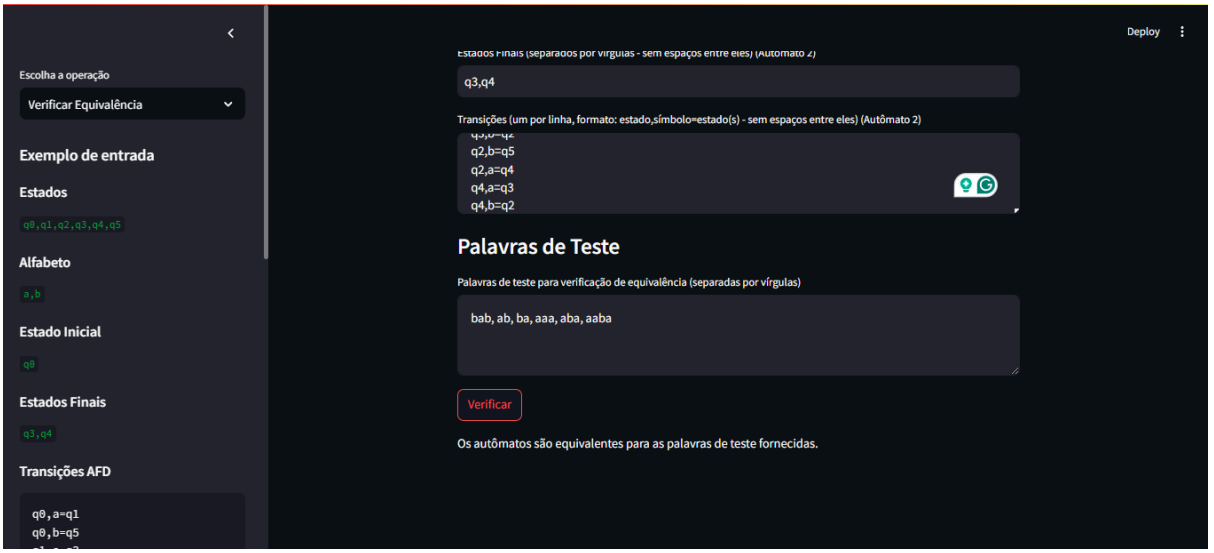


Entrada Executada: Exemplo 2 AFD



7.5 Verificar Equivalência

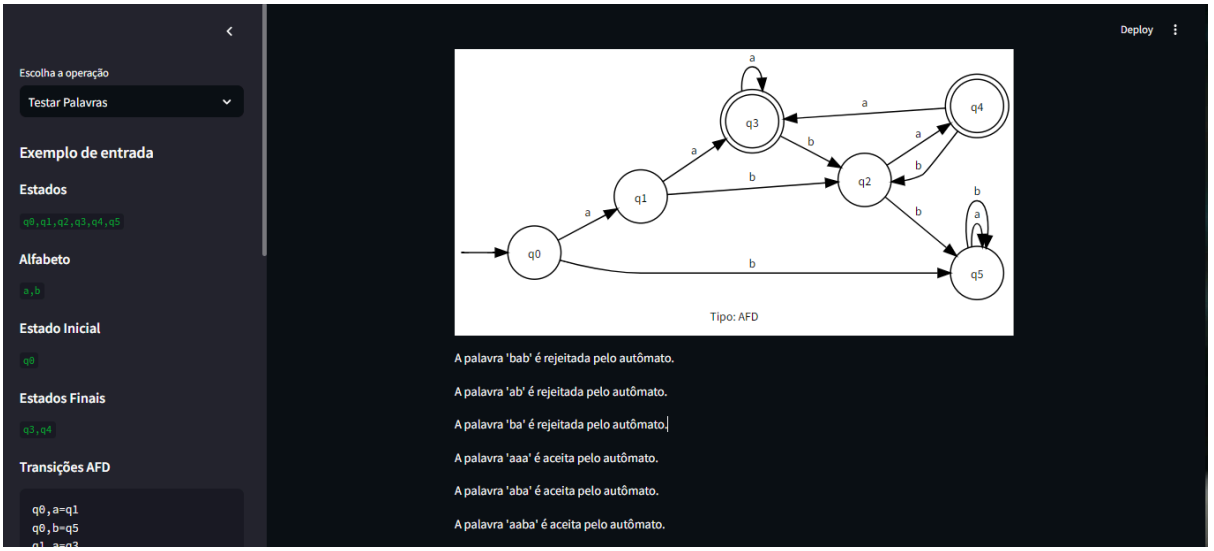
Entrada Executada: Exemplo 1 AFD com Exemplo 1 AFN



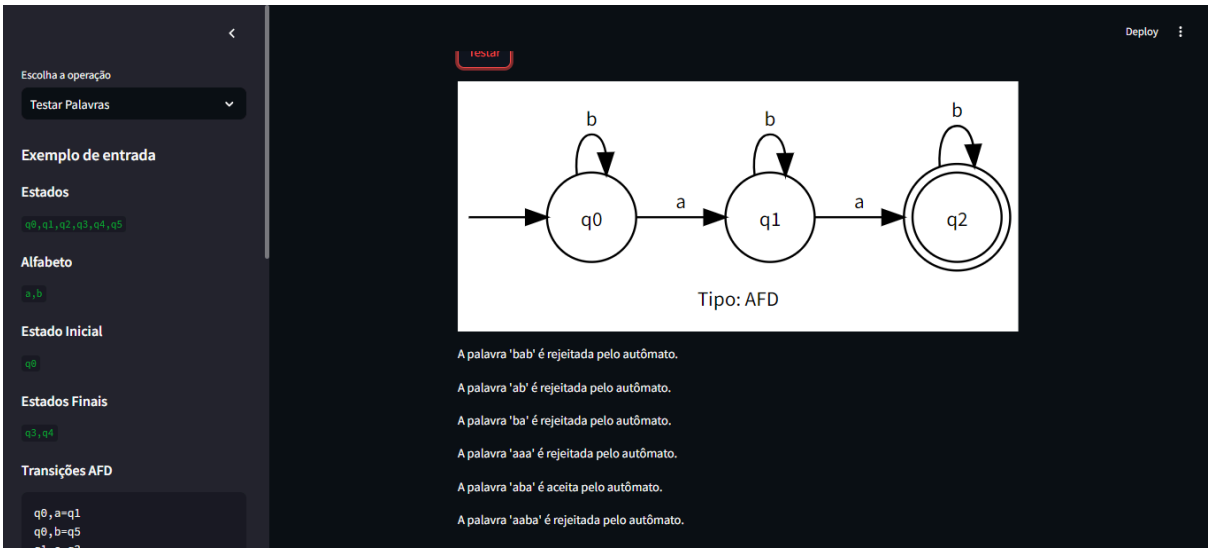
Entrada Executada: Exemplo 2 AFD com Exemplo 2 AFN

7.6 Testar Palavras

Entrada Executada: Exemplo 1 AFD

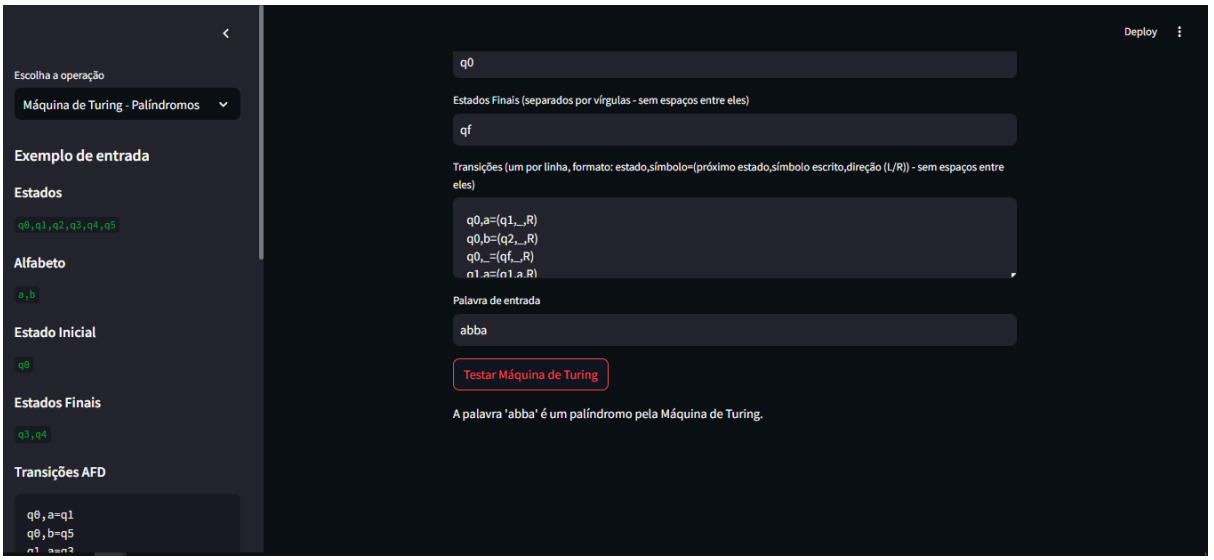


Entrada Executada: Exemplo 2 AFD



7.7 Reconhecimento de palavras pela Máquina de Turing

Entrada Executada: Exemplo 1 - Verificação de Palíndromos



Entrada Executada: Exemplo 2 - Cópia de Cadeia de Caracteres