UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA-CCET

ENGENHARIA DA COMPUTAÇÃO

DISCENTES: THALES GUSTAVO MENDES NUNES, ANTONIO LISTER AZEVEDO SOUSA

LINGUAGENS FORMAIS E AUTÔMATOS

ALGORITMO DE CONVERSÃO DE AUTÔMATOS FINITOS NÃO-DETERMINÍSTICOS PARA AUTÔMATOS FINITOS DETERMINÍSTICOS

Conversão de autômatos

Um autômato finito é uma máquina abstrata usada para reconhecer linguagens formais. Existem dois tipos principais de autômatos finitos:

- Autômato Finito Determinístico (AFD): Cada estado possui transições únicas para cada símbolo de entrada. Então pensando um pouco nisto como um jogo, podemos linkar ele como se fosse um mapa que indica o caminho que devemos percorrer até o final, e que o avanço só ocorre caso tenhamos as ações corretas realizadas em cada fase. Havendo um pré requisito para cada uma das sequências de fases.

- Autômato Finito Não Determinístico (AFN): Cada estado pode ter múltiplas transições para o mesmo símbolo de entrada ou transições sem consumir símbolos (epsilon-transições). Usando da mesma analogia de jogo, aqui teríamos um caso de jogos de tabuleiros onde você define sua próxima jogada, fazendo com que não esteja determinada a trajetória que deve percorrer para a vitória.

Pensando nisso, devemos ter em mente que na estrutura básica de um autômato, possuímos estados e transições, e que as Transições definem o comportamento do

autômato para diferentes entradas. Em um AFN, pode haver múltiplas transições para o mesmo símbolo, enquanto em um AFD, cada símbolo leva a um único estado.

Portanto, para fazer a conversão de um AFD para o nosso AFN, devemos possuir um certo conjunto de estados de aceitação para que nossa entrada seja válida. Para isso devemos realizar a criação de subconjuntos que nos auxiliam com o fechamento de epsilon, dentro do nosso código.

A importância dessa conversão, ou seja, a construção de subconjuntos como estes podem reduzir significativamente o tamanho de um autômato. Ao eliminar estados e transições redundantes, a construção de subconjuntos pode criar um autômato menor e mais gerenciável. Essa redução no tamanho pode levar a tempos de processamento mais rápidos e melhor utilização da memória.

Resultados

Exemplo 1:

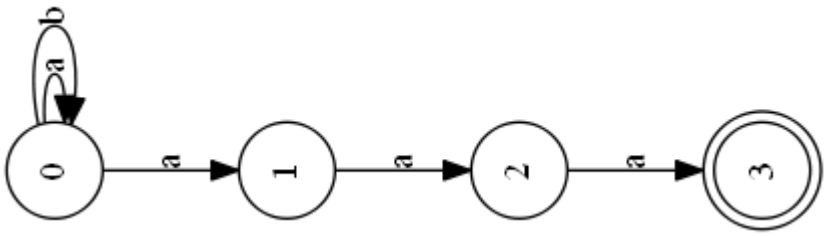
Autômato finito não-determinístico que aceita cadeias que contém aaa com sufixo.

Σ: {a, b}

Q: {q0, q1, q2, q3}

q0: {q0}

F: {q3}



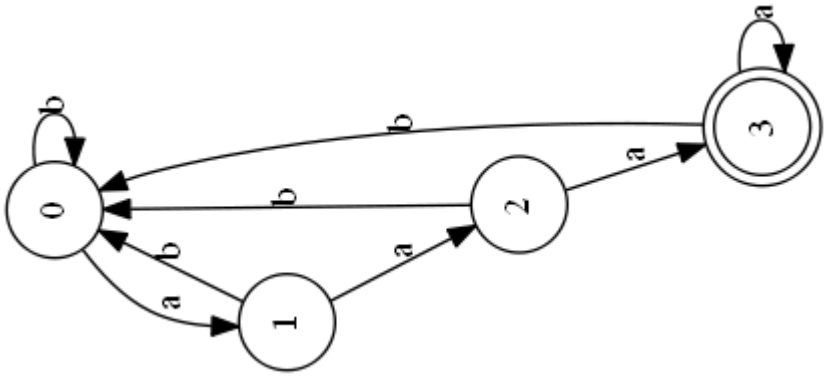
Autômato finito determinístico resultante da conversão:

Σ: {a, b}

Q: {q0, q1, q2, q3}

q0: {q0}

F: {q3}



Exemplo 2:

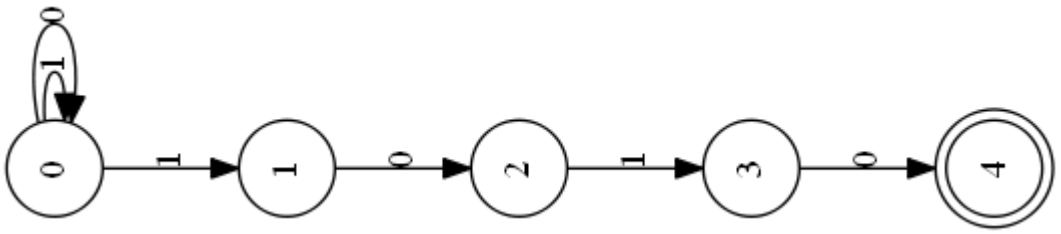
Autômato finito não-determinístico que aceita cadeias que contém a sequência 1010.

Σ: {0, 1}

Q: {q0, q1, q2, q3, q4}

q0: {q0}

F: {q4}



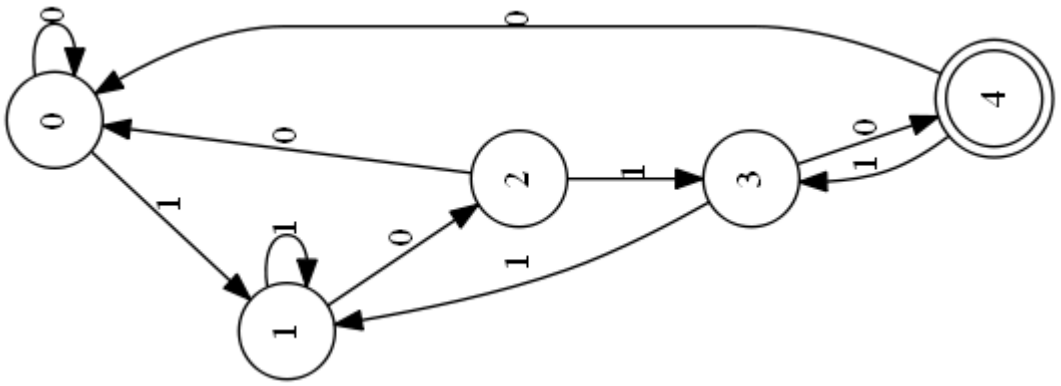
Autômato finito determinístico resultante da conversão:

Σ: {a, b}

Q: {q0, q1, q2, q3, q4}

q0: {q0}

F: {q4}



Exemplo 3:

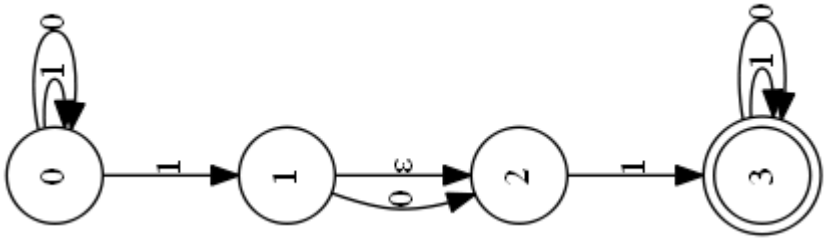
Autômato finito não-determinístico que aceita cadeias que contém a sequência 11 ou 101.

Σ: {0, 1}

Q: {q0, q1, q2, q3, q4}

q0: {q0}

F: {q4}



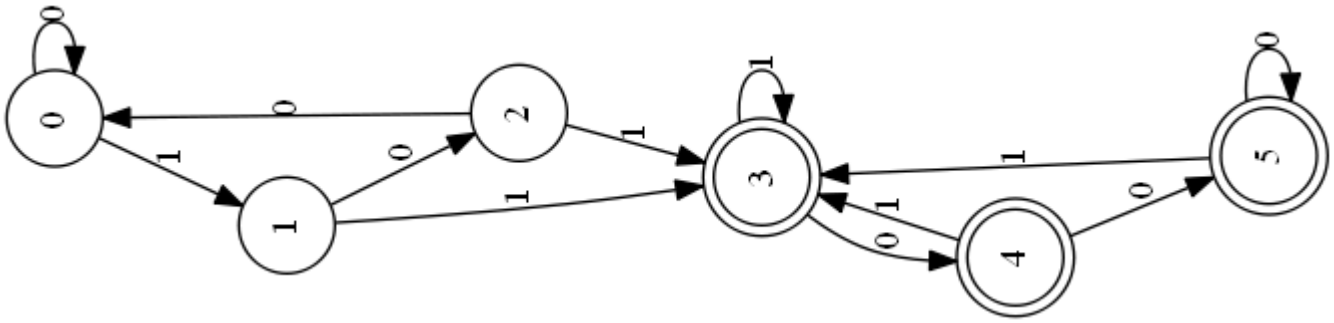
Autômato finito determinístico resultante da conversão:

Σ: {0, 1}

Q: {q0, q1, q2, q3, q4, q5}

q0: {q0}

F: {q3, q4, q5}



Manual do usuário

Visão Geral

A classe Automaton é usada para criar e manipular autômatos finitos não determinísticos (AFN) e convertê-los em autômatos finitos determinísticos (AFD). Este manual irá guiá-lo pelo processo de inicialização do autômato, adição de transições, conversão para AFD, e impressão do autômato.

Instalação da Biblioteca graphviz

Para usar a visualização de grafos, você precisa instalar a biblioteca graphviz e garantir que o executável do Graphviz esteja no PATH do sistema.

```
pip install graphviz
```

Inicialização

Para iniciar um autômato, você deve especificar o número de estados e os estados de aceitação.

```
aut = Automaton(num_states=5, accepting_states={1, 3})
```

- num_states: Número total de estados no autômato.
- accepting_states: Conjunto de estados que são estados de aceitação.

Adicionando Transições

Você pode adicionar transições entre estados usando o método add_transition.

```
aut.add_transition(start_state=0, characters='a', end_state=1)
aut.add_transition(start_state=1, characters=('b', 'c'),
end_state=2)
```

- start_state: Estado inicial da transição.
- characters: Caracter(es) que causam a transição. Pode ser um único caractere ou uma tupla de caracteres.
- end_state: Estado de destino da transição.

Fechamento Epsilon

O método epsilon_closure calcula o fechamento epsilon para um conjunto de estados, que inclui todos os estados que podem ser alcançados a partir dos estados dados através de transições epsilon.

```
closure = aut.epsilon_closure({0, 1})
```

- states: Conjunto de estados para os quais calcular o fechamento epsilon.
- Retorna: Conjunto de estados que são alcançáveis a partir dos estados fornecidos, incluindo eles próprios

Conversão para AFD

O método convert_to_dfa converte o autômato não determinístico (AFN) em um autômato determinístico (AFD).

```
dfa = aut.convert_to_dfa()
```

- Sem parâmetros.
- Retorna: Um novo objeto Automaton que representa o AFD resultante da conversão.

Plotando o Grafo do Autômato

O método convert_to_dfa plota o autômato como um grafo e salva a imagem resultante.

```
aut.plot_automaton('my_automaton')
```

- filename: Nome do arquivo para salvar a imagem do grafo (sem extensão).
- Retorna: Um novo objeto Automaton que representa o AFD resultante da conversão.

Notas Adicionais

- As transições epsilon são representadas pelo caractere vazio ''.
- As transições são armazenadas em uma lista de dicionários, onde cada dicionário mapeia caracteres para conjuntos de estados de destino.
- O método epsilon_closure é um método auxiliar usado internamente para calcular o fechamento epsilon.
- A conversão para AFD é feita utilizando um algoritmo padrão de construção de subconjuntos, garantindo que o resultado seja um autômato determinístico equivalente ao AFN original.