MINIMIZAÇÃO DE AUTÔMATOS FINITOS DETERMINÍSTICOS (DFA)

JUSTINO FELIPE LOPES NUNES

MINIMIZAÇÃO DE AUTÔMATOS FINITOS DETERMINÍSTICOS (DFA)

Minimização de autômatos é o processo de transformar um autômato finito determinístico (DFA) em um equivalente que possui um número mínimo de estados. A minimização de DFAs é importante porque pode resultar em autômatos mais eficientes, tanto em termos de tempo de processamento quanto de utilização de memória.

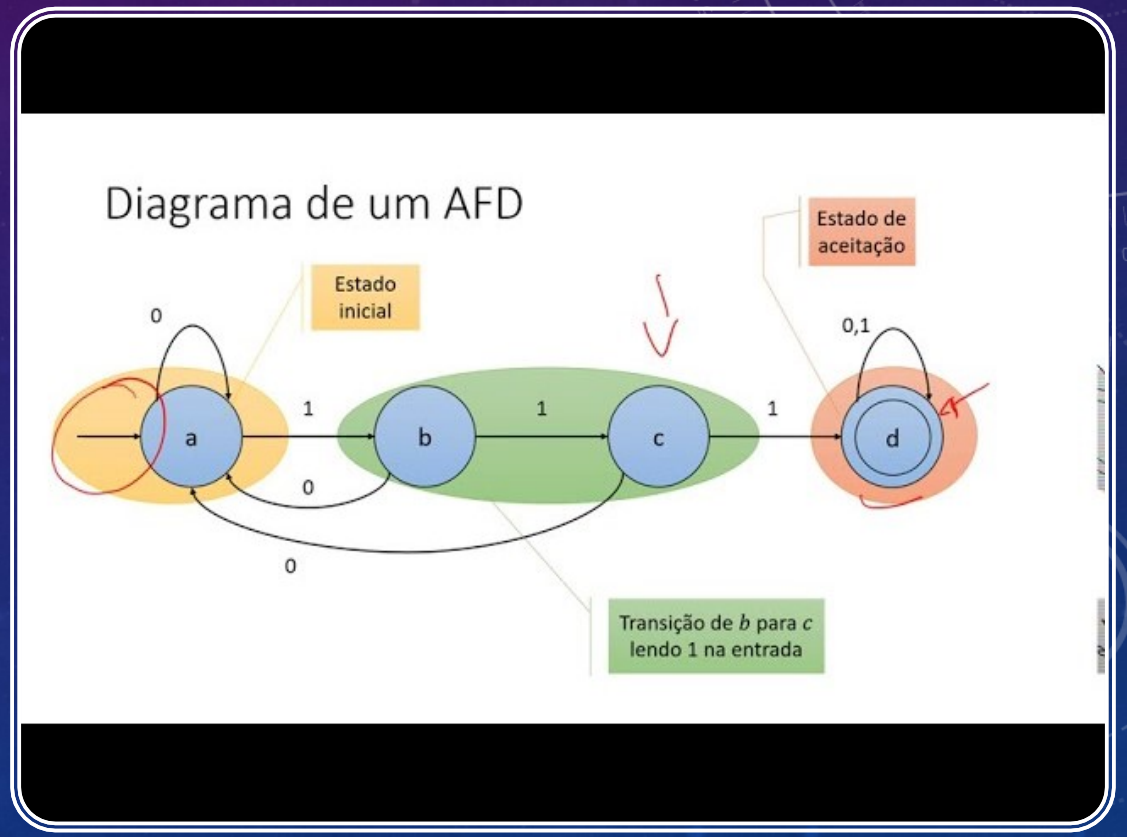
REQUISITOS

- Para poder realizar a minimização do autômato, devem ser verificados os seguintes pré-requisitos :
  Ele deve ser determinístico.
  Não pode ter estados inacessíveis
  A função programa deve ser total
- Esses pré-requisitos, caso necessário, podem ser satisfeitos das seguintes formas:
  Transformar o AFN ou o AFϵ em AFD
  Eliminar estados inacessíveis.
  Para transformar uma função parcial em total, basta introduzir um novo estado não-final d e incluir as transições não-previstas, tendo d como estado destino.

AUTÔMATO FINITO DETERMINÍSTICO (DFA)

Um DFA é uma máquina abstrata usada para reconhecer linguagens formais. Um DFA é definido por:

- Um conjunto finito de estados.
- Um alfabeto finito de símbolos.
- Uma função de transição que define a mudança de estados com base em um símbolo de entrada.
- Um estado inicial.
- Um conjunto de estados de aceitação.

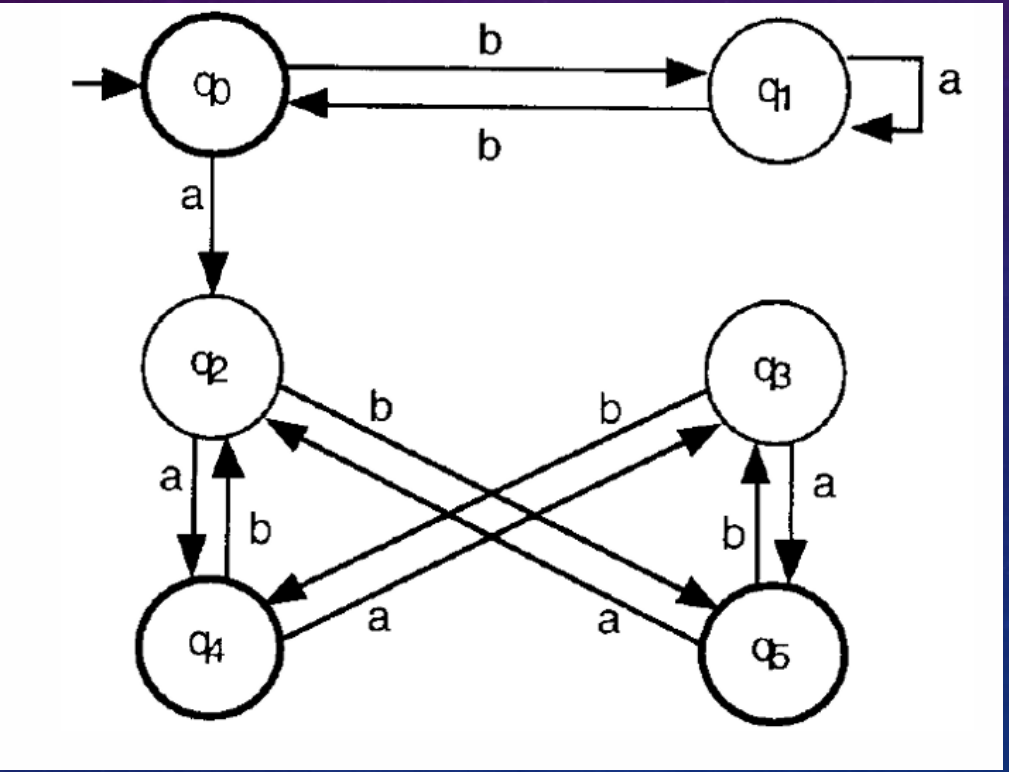


PROCESSO DE MINIMIZAÇÃO

A minimização de um DFA envolve a eliminação de estados redundantes e a fusão de estados equivalentes. Uma breve descrição passo a passo do processo de minimização:

1. **Remover estados inacessíveis**:
   - Identifique e remova todos os estados que não são acessíveis a partir do estado inicial.
2. **Inicializar a partição**:
   - Separe os estados em dois grupos: estados de aceitação e estados não aceitação.
3. **Refinar a partição**:
   - Continuamente divida os grupos de estados em subconjuntos menores até que não seja mais possível realizar subdivisões. Dois estados são considerados equivalentes se, para cada símbolo do alfabeto, as transições desses estados levam a estados que são equivalentes na partição atual.
4. **Construir o DFA minimizado**:
   - Cada grupo de estados equivalentes se torna um único estado no DFA minimizado. Defina as transições do novo DFA de acordo com as transições dos estados originais.

REPRESENTAÇÃO GRÁFICA



EXEMPLOS

Original DFA 1:

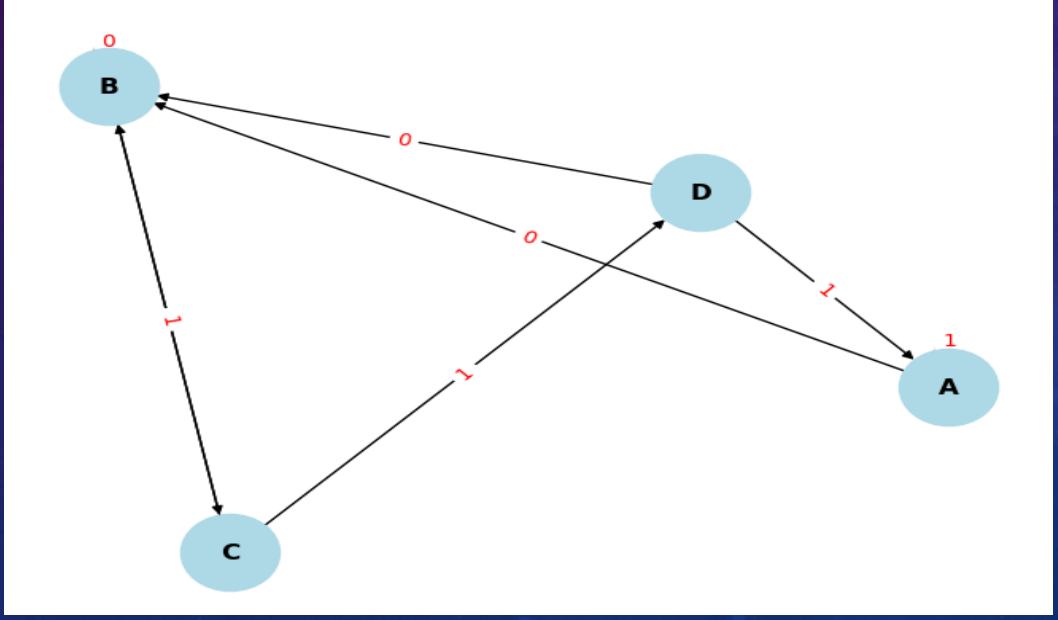
States: {'C', 'D', 'A', 'B'}

Alphabet: {'1', '0'}

Start state: A

Accept states: {'C'}

Transitions: {('A', '0'): 'B', ('A', '1'): 'A', ('B', '0'): 'B', ('B', '1'): 'C', ('C', '0'): 'B', ('C', '1'): 'D', ('D', '0'): 'B', ('D', '1'): 'A'}



Minimizeado DFA 1:

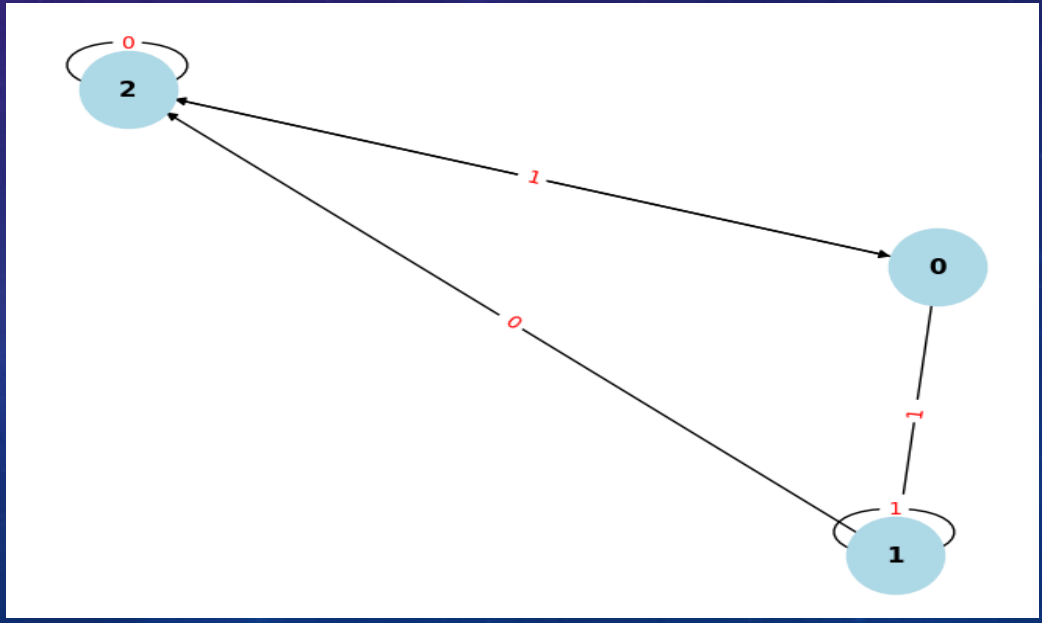
States: {0, 1, 2}

Alphabet: {'1', '0'}

Start state: 1

Accept states: {0}

Transitions: {(0, '1'): 1, (0, '0'): 2, (1, '1'): 1, (1, '0'): 2, (2, '1'): 0, (2, '0'): 2}



PASSO A PASSO

- **Remover Estados Inacessíveis:**

  Verificamos se todos os estados podem ser alcançados a partir do estado inicial A.

  Neste caso, todos os estados {'A', 'B', 'C', 'D'} são acessíveis.

- **Inicializar a Partição:**

  Criamos duas partições iniciais: uma para estados de aceitação e outra para estados não aceitação.

  {C} (estado de aceitação)

  {A, B, D} (estados não aceitação)

- **Refinar a Partição:**

  Começamos a verificar transições e dividir grupos até não ser mais possível.

  Verificamos as transições de cada estado em relação aos símbolos do alfabeto:

  Para o estado 'A', a transição para '0' leva a 'B' e para '1' leva a 'A'.

  Para o estado 'B', a transição para '0' leva a 'B' e para '1' leva a 'C'.

  Para o estado 'D', a transição para '0' e '1' leva a 'D'.

  Vemos que 'B' e 'D' devem ser separados em diferentes grupos devido às suas transições distintas. Então, refinamos a partição:

  {C} (estado de aceitação)

  {A} (estado não aceitação)

  {B} (estado não aceitação com transição diferente)

  {D} (estado não aceitação com transições diferentes)

- **Construir o DFA Minimizado:**

  Cada grupo de estados equivalentes se torna um único estado no DFA minimizado.

  Temos três grupos finais: {A}, {B}, {C, D}.

  Nomeamos esses novos estados como 1, 2 e 0, respectivamente.

  Definimos as transições do novo DFA com base nas transições dos estados originais:

  (1, '0') -> 2

  (1, '1') -> 1

  (2, '0') -> 2

  (2, '1') -> 0

  (0, '0') -> 2

  (0, '1') -> 1

CONCLUSÃO

A minimização de DFAs é um processo crucial para otimizar a eficiência dos autômatos. Ao remover estados redundantes e combinar estados equivalentes, podemos criar um DFA mais compacto e eficiente sem alterar a linguagem que ele reconhece. Esse processo pode ser aplicado a qualquer DFA para melhorar seu desempenho e reduzir o consumo de recursos computacionais.