CONVERSOR DE AUTÔMATO FINITO COM MOVIMENTOS VAZIOS (AFE) PARA AUTÔMATO FINITO NÃO DETERMINÍSTICO (AFN)

Disciplina: Linguagens Formais e Autômatos

Orientador: Dr. Thales L. A. Valente

Universidade Federal do Maranhão

São Luís - Maranhão. | 14 e 16 de jan. 2025

INTEGRANTES - GRUPO 3

FERNANDA SOUSA DE ASSUNÇÃO VALE
Matrícula: 2022024316

JHONES DE SOUSA SOARES
Matrícula: 2020002730

ÍNDICE

1. INTRODUÇÃO

O que é um autômato finito?

Dispositivo de tamanho finito com certas partes especificadas como entradas e saídas, de modo que o que acontece nas saídas é determinado, ou pelo menos sua função de distribuição de probabilidade é determinada, pelo que aconteceu nas entradas.

Fita

Unidade de controle

Programa ou função de transição

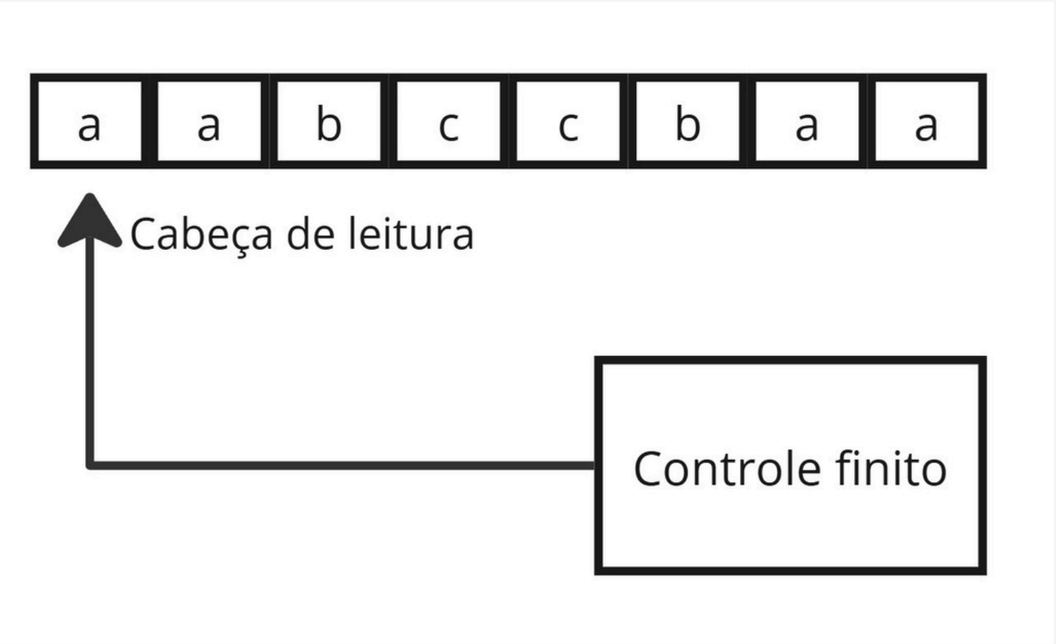


Figura 1.1 - Exemplificação da composição de um automato finito

1. INTRODUÇÃO

Objetivos do trabalho

Implementar com sucesso a transformação de um Autômato Finito de movimentos vazios(AFε) para um Autômato Finito não determinístico(AFD).

- Remoção das operações movimentos vazios;
- Visualização dos automatos formados

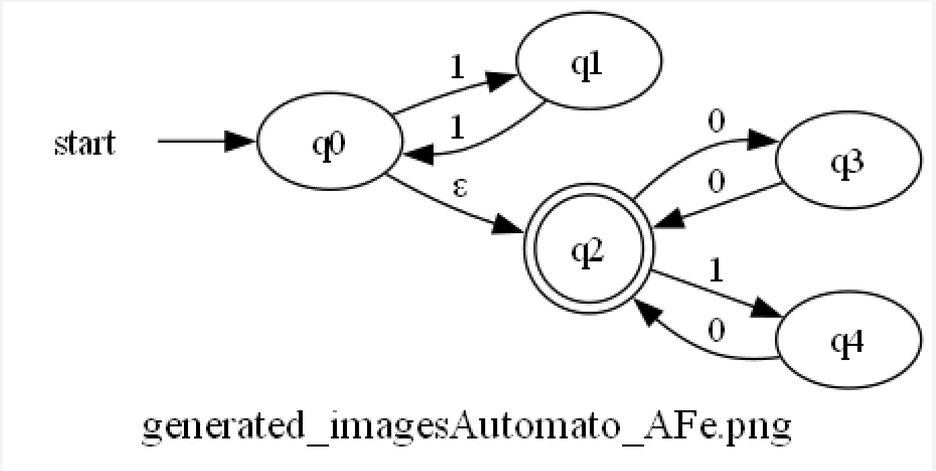


Figura 1.2 - Automato Finito com movimentos vazios

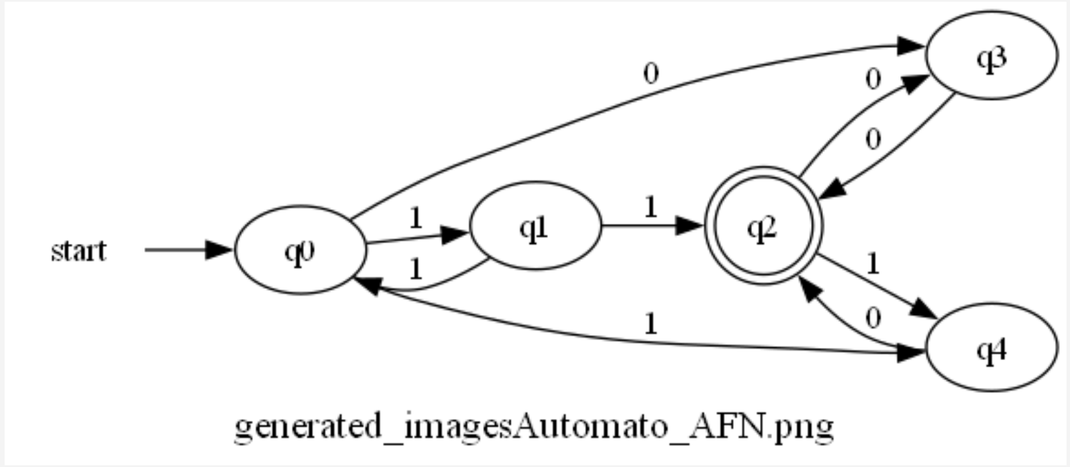


Figura 1.3 - Automato Finito não Determinístico equivalente

2. FUNDAMENTAÇÃO TEÓRICA

Autômato Finito não determinístico(AFN)

Um Autômato Finito Não-Determinístico (AFN) M é uma 5-upla:

$$M = (\Sigma, Q, \delta, q_o, F)$$

Σ: alfabeto de símbolos de entrada;

Q: conjunto de estados possíveis do autômato, o qual é finito;

δ: função programa ou função de transição:

- $\delta: Q \times \Sigma \rightarrow 2^Q$

q_o: estado inicial, tal que q_o é elemento de Q;

F: conjunto de estados finais, tal que F está contido em Q.

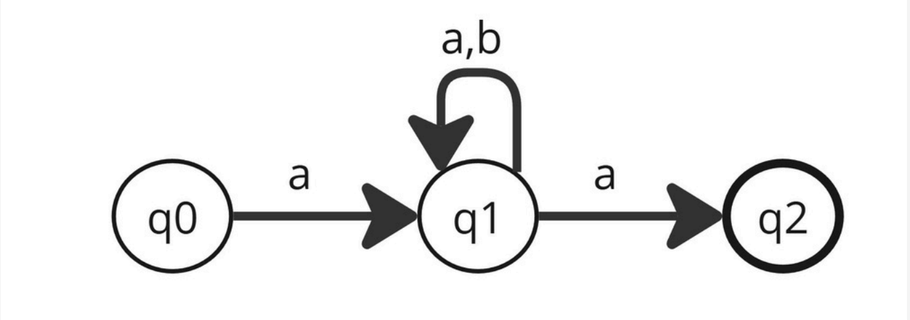


Figura 2.1 - Automato Finito não determinístico

2. FUNDAMENTAÇÃO TEÓRICA

Autômato Finito com Movimentos Vazios(AFNε)

Um Autômato Finito de movimentos vazios(AFNε) M é uma 5-upla:

$$M = (\Sigma, Q, \delta, q_o, F)$$

Σ: alfabeto de símbolos de entrada;

Q: conjunto de estados possíveis do autômato, o qual é finito;

δ: função programa ou função de transição:

- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$

q_o: estado inicial, tal que q_o é elemento de Q;

F: conjunto de estados finais, tal que F está contido em Q.

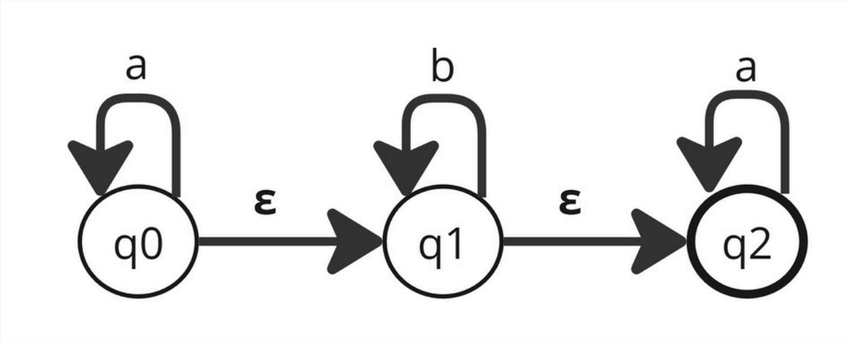


Figura 2.2 - Automato Finito com movimentos vazios

3. DESENVOLVIMENTO

PARTE 1

Codigo “AFe_to_AFN.py”.

PARTE 2

Código “main.py” + arquivos JSON

PARTE 1
Código
AFe_to_AFN.py

1. Sobre as Bibliotecas

```python
#Biblioteca para gerar visualizações gráficas de autômatos.
import graphviz
#Estrutura de dicionário que inicializa automaticamente
#valores padrão, útil para evitar KeyError.
from collections import defaultdict
```

Figura 3.1 - Bibliotecas do arquivo AFe_to_AFN.py

2. Sobre as Funções

PARTE 1
Código
AFe_to_AFN.py

```python
def visualize_automaton(transitions, start_state, final_states, title):
    #ver os autômatos:
    #Cria um objeto Digraph do graphviz para gerar o grafo em formato PNG
    dot = graphviz.Digraph(format="png", engine="dot")
    #Define a direção do grafo da esquerda para a direita
    dot.attr(rankdir="LR")

    #Adiciona os estados ao grafo.
    for state in transitions:
        #Se um estado estiver na lista de estados finais,
        #ele recebe a forma de "doublecircle" (indica estado final).
        if state in final_states:
            dot.node(state, shape="doublecircle")
        else:
            dot.node(state) #Estados comuns são representados normalmente.
    #add qo
    #Cria um nó fictício "start" sem forma (shape="none") para indicar visualmente o estado inicial.
    dot.node("start", shape="none")
    dot.edge("start", start_state) #Conecta "start" ao start_state com uma aresta.
```

Figura 3.2 - Função para visualizar autômato do arquivo

PARTE 1
Código
AFe_to_AFN.py

2. Sobre as Funções

```
29      #adiciona transições
30      #Percorre as transições do autômato.
31      for state, moves in transitions.items():
32          #Usa defaultdict(list) para armazenar transições organizadas por estado origem → destino.
33          edges = defaultdict(list)
34           #Agrupa os símbolos que levam de um estado a outro.
35          for symbol, next_states in moves.items():
36              for next_state in next_states:
37                  edges[(state, next_state)].append(symbol)
38          #Para cada transição src -> dest, cria uma aresta com um rótulo contendo os
39          #símbolos separados por vírgula.
40          for (src, dest), symbols in edges.items():
41              dot.edge(src, dest, label=",".join(symbols))
42      #Define o título do grafo.
43      dot.attr(label=title, fontsize="16")
44      #Renderiza e exibe o grafo, nomeando o arquivo com base no título.
45      dot.render(f"{title.replace(' ', '_')}", cleanup=True, view=True)
```

Figura 3.3 - Função para visualizar autômato do arquivo

2. Sobre as Funções

PARTE 1
Código
AFe_to_AFN.py

```python
#Calcula o fecho-ε (ε-closure) de um estado, ou seja,
# todos os estados acessíveis apenas por transições vazias (ε).
def e_closure(state, transitions):
    #Inicializa uma pilha (stack) e um conjunto (closure) com o estado inicial
    stack = [state]
    closure = set(stack)
    #Enquanto houver estados na pilha, remove o topo (pop()).
    while stack:
        current_state = stack.pop()
        #Para cada estado acessível via transição ε, adiciona ao
        # conjunto closure se ainda não estiver lá.
        #O novo estado é colocado na pilha para continuar a busca recursiva.
        for next_state in transitions[current_state].get("ε", []):
            if next_state not in closure:
                closure.add(next_state)
                stack.append(next_state)
    #Retorna o fecho-ε do estado inicial.
    return closure
```

Figura 3.4 - Função para calcular o fecho-ε para um estado específico

2. Sobre as Funções

PARTE 1
Código
AFe_to_AFN.py

```python
#Converte um Autômato Finito com ε-transições (AFE) em um
# Autômato Finito Não Determinístico (AFN).
def convert_afe_to_afn(afe_transitions, start_state, final_states):

    #Inicializa a estrutura de transições do AFN, onde cada
    # estado tem um dicionário de símbolos mapeando para conjuntos de estados alcançáveis.
    afn_transitions = defaultdict(lambda: defaultdict(set))
    #calcula o fecho-ε para cada estado no autômato e armazena no dicionário closures.
    closures = {state: e_closure(state, afe_transitions) for state in afe_transitions}
    #Coleta todos os símbolos usados no autômato, excluindo ε.
    all_symbols = set(
        symbol
        for state_transitions in afe_transitions.values()
        for symbol in state_transitions.keys()
        if symbol != "ε"
    )
```

Figura 3.5 - Função para converter o AFe em AFN

2. Sobre as Funções

PARTE 1
Código
AFe_to_AFN.py

```
83      #monta o AFN
84      #Para cada estado e seu fecho-ε, inicia a análise das transições para cada símbolo.
85      for state, closure in closures.items():
86          for symbol in all_symbols:
87              reachable_states = set()
88              #Percorre os estados no fecho-ε e verifica para onde podem ir com determinado símbolo.
89              for intermediate_state in closure:
90                  reachable_states.update(afe_transitions[intermediate_state].get(symbol, []))
91              #Para cada estado acessível diretamente, adiciona ao novo autômato o
92              # conjunto de estados acessíveis via fecho-ε.
93              for reachable_state in reachable_states:
94                  afn_transitions[state][symbol].update(closures[reachable_state])
95
96          #Se qualquer estado no fecho-ε for um estado final, o estado original
97          # também deve ser tratado como final.
98          #atualiza os estados finais
99          if closure.intersection(final_states):
100             final_states.update(closure)
101     #Retorna as transições do AFN e a lista de estados finais atualizada.
102     return {state: dict(moves) for state, moves in afn_transitions.items()}, final_states
103
```

Figura 3.6 - Função para converter o AFe em AFN

PARTE 2
Código “main.py” + arquivos JSON

```json
{
    "transitions": {
        "q0": {"ε": ["q1", "q2"]},
        "q1": {"a": ["q1", "q3"]},
        "q2": {"b": ["q2"]},
        "q3": {"ε": ["q4"]},
        "q4": {"c": ["q4"]}
    },
    "start_state": "q0",
    "final_states": ["q4"]
}
```

Figura 3.7 - Estrutura do arquivo JSON

PARTE 2
Código “main.py” + arquivos JSON

```
2   #Utilizado para visualizar o autômato.
3   import graphviz
4   #Para carregar o autômato a partir de um arquivo .json.
5   import json
6   #Um tipo de dicionário que inicializa automaticamente valores padrão.
7   from collections import defaultdict
8   #biblioteca para interface gráfica (tkinter é a
9   # biblioteca principal, messagebox e filedialog são módulos auxiliares).
10  import tkinter as tk
11  from tkinter import messagebox, filedialog
12  #Biblioteca para expressões regulares (não está sendo
13  # usada no código, pode ser um resquício de código antigo).
14  import re
15  #Funções importadas de um módulo (src.AFe_to_AFN),
16  # que convertem autômatos e os visualizam.
17  from src.AFe_to_AFN import convert_afe_to_afn, visualize_automaton
```

Figura 3.8 - Importação das bibliotecas e funções

PARTE 2
Código “main.py” + arquivos JSON

```python
#Carrega os dados do autômato a partir de um arquivo JSON.
def load_automaton_from_json(filepath):
    #Abre o arquivo JSON (filepath) em modo leitura
    # ('r'), garantindo compatibilidade com caracteres especiais (utf-8).
    with open(filepath, 'r', encoding='utf-8') as file:
        #Usa json.load(file) para carregar o conteúdo do arquivo em um dicionário Python.
        data = json.load(file)
    #Retorna as transições do autômato, o estado inicial e
    # os estados finais (convertidos em um set).
    return data['transitions'], data['start_state'], set(data['final_states'])
```

Figura 3.9 - Função para carregar o arquivo JSON

PARTE 2
Código “main.py” + arquivos JSON

```python
#Abre uma janela para seleção de arquivos JSON e executa a conversão do AFe para AFN.
def select_file():
    #Abre uma caixa de diálogo para o usuário selecionar um arquivo .json
    filepath = filedialog.askopenfilename(filetypes=[("JSON files", "*.json")])
    #Verifica se um arquivo foi selecionado.
    if filepath:
        #Carrega o autômato do arquivo JSON.
        afe_transitions, start_state, final_states = load_automaton_from_json(filepath)

        #Exibe no console as transições do AFe, o estado inicial e os estados finais.
        print("Transições do AFe:")
        print(afe_transitions)
        print("\nEstado Inicial:", start_state)
        print("\nEstados Finais:", final_states)

        #Gera e exibe um gráfico do AFe.
        visualize_automaton(afe_transitions, start_state, final_states, f"Automato - AFe")

        #Converte o AFe para AFN chamando convert_afe_to_afn().
        afn_transitions, afn_final_states = convert_afe_to_afn(afe_transitions, start_state, final_states)

        #Exibe no console as transições do AFN e seus estados finais.
        print("\nTransições do AFN:")
        print(afn_transitions)
        print("\nEstados Finais do AFN:", afn_final_states)

        #Gera e exibe um gráfico do AFN.
        visualize_automaton(afn_transitions, start_state, afn_final_states, f"Automato - AFN")
```

Figura 3.10 - Função para geração de display dos resultados

PARTE 2
Código “main.py” + arquivos JSON

```python
#Cria a interface gráfica para o conversor de AFe para AFN.
def main():
    #Cria a janela principal (Tk()) e define o título.
    root = tk.Tk()
    root.title("Conversor de AFe para AFN")

    #Cria um frame dentro da janela principal,
    # com espaçamento interno e externo de 10 pixels.
    frame = tk.Frame(root, padx=10, pady=10)
    frame.pack(padx=10, pady=10)

    #Adiciona um rótulo (Label) com a instrução para selecionar um arquivo JSON.
    label = tk.Label(frame, text="Selecionar arquivo JSON com o AFe")
    label.pack(pady=5)

    #Adiciona um botão que, quando clicado, chama select_file().
    button = tk.Button(frame, text="Selecionar arquivo", command=select_file)
    button.pack(pady=5)

    #Mantém a janela aberta aguardando interação do usuário.
    root.mainloop()
```

Figura 3.11 - Função para geração e display da interface

PARTE 2
Código “main.py” + arquivos JSON

```
82  #Verifica se o script está sendo
83  # executado diretamente (__name__ == "__main__").
84  #Se for, chama main() para iniciar a interface gráfica.
85  if __name__ == "__main__":
86      main()
```

Figura 3.12 - Função principal

4. ANÁLISE DE RESULTADOS

RESULTADOS ESPERADOS VS OBTIDOS

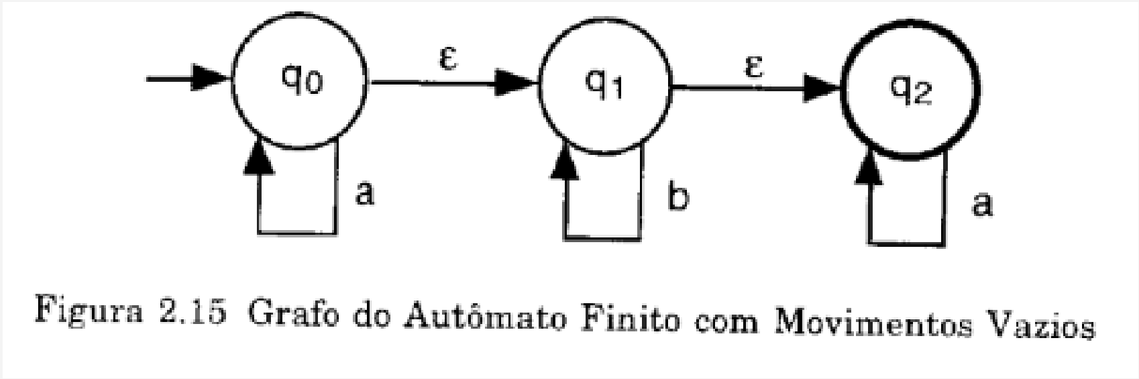


Figura 2.15 Grafo do Autômato Finito com Movimentos Vazios

Figura 4.1 - AFe de exemplo para teste do projeto

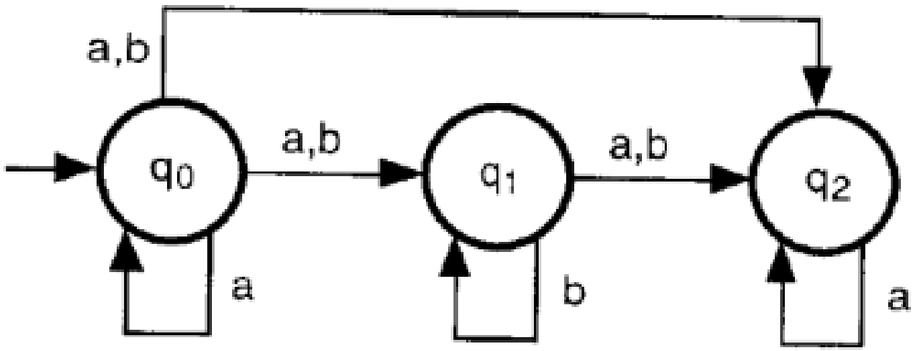


Figura: Grafo do AFN de M_9

Figura 4.2 - AFN equivalente ao AFe anterior

RESULTADOS ESPERADOS VS OBTIDOS

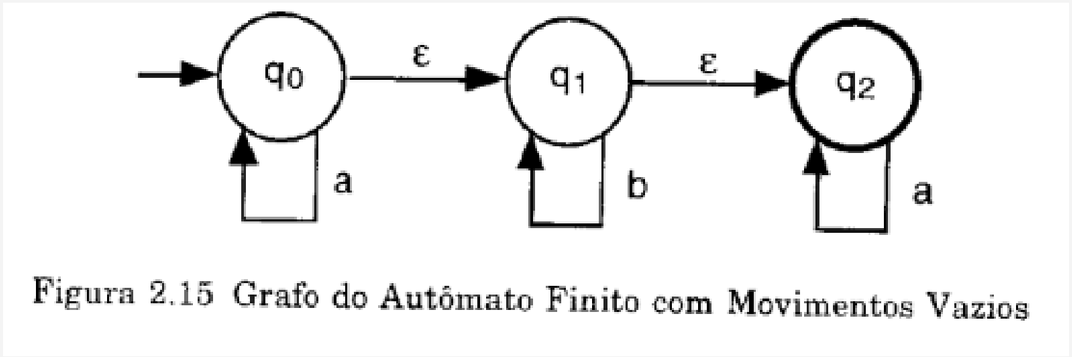


Figura 2.15 Grafo do Autômato Finito com Movimentos Vazios

Figura 4.3 - AFe esperado

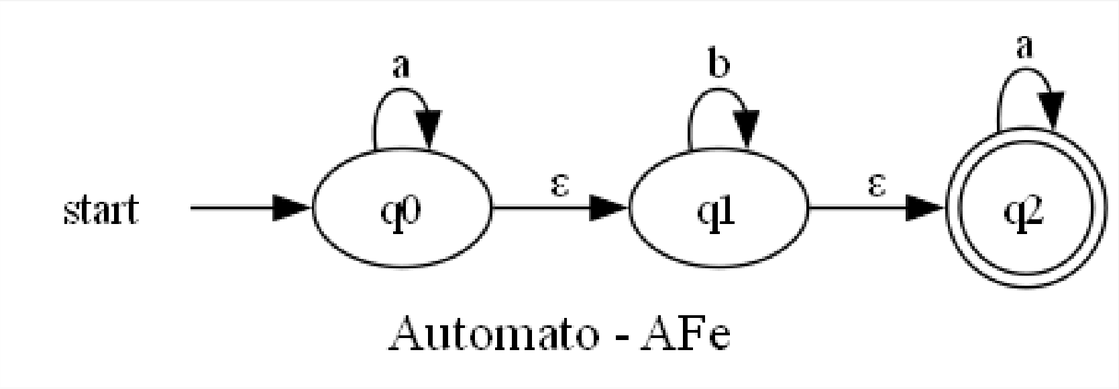


Figura 4.4 - AFe obtido

RESULTADOS ESPERADOS VS OBTIDOS

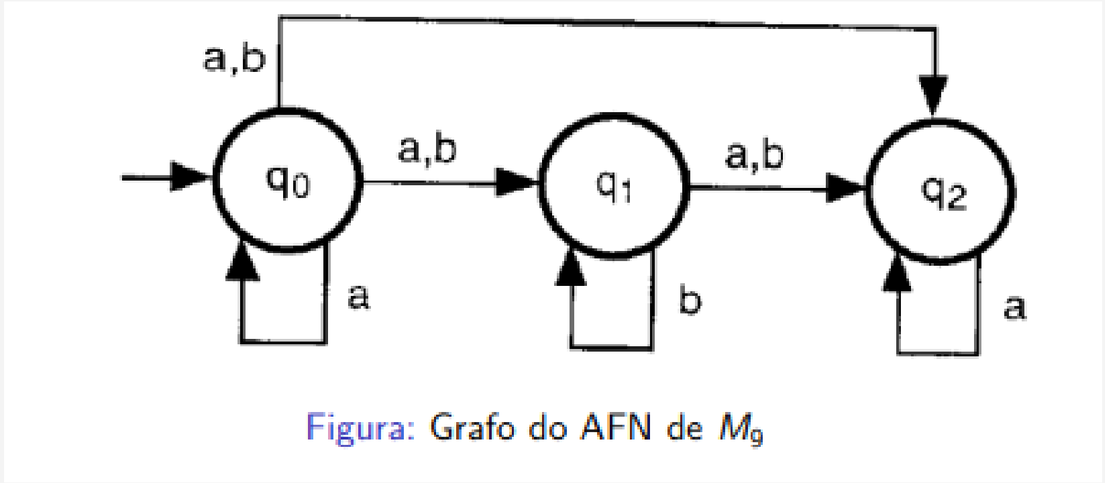


Figura 4.5 - AFN esperado

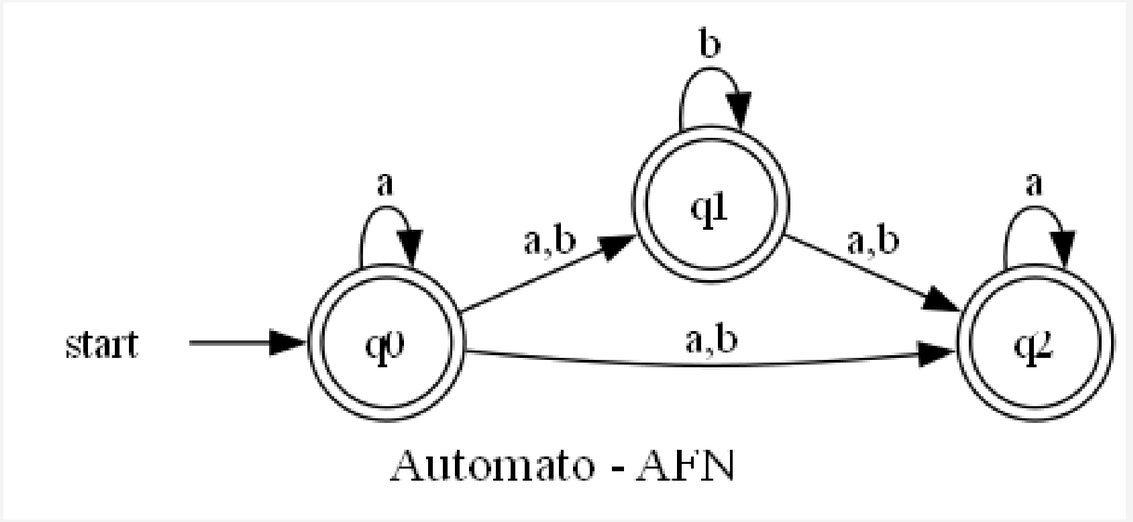


Figura 4.6 - AFN obtido

RESULTADOS ESPERADOS VS OBTIDOS

	a	b	c
q0	{q0, q1, q2}	{q1, q2}	{q2}
q1	{}	{q1, q2}	{q2}
q2	{}	{}	{q2}

Tabela 1 - Tabela obtida pelo AFe da Figura 4.7

RESULTADOS ESPERADOS VS OBTIDOS

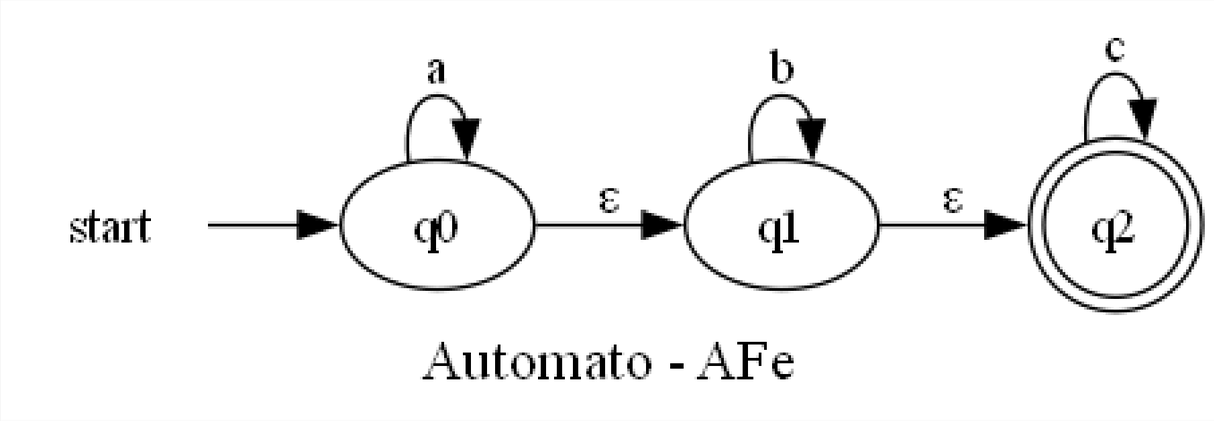


Figura 4.7 - AFe correspondente à tabela 1

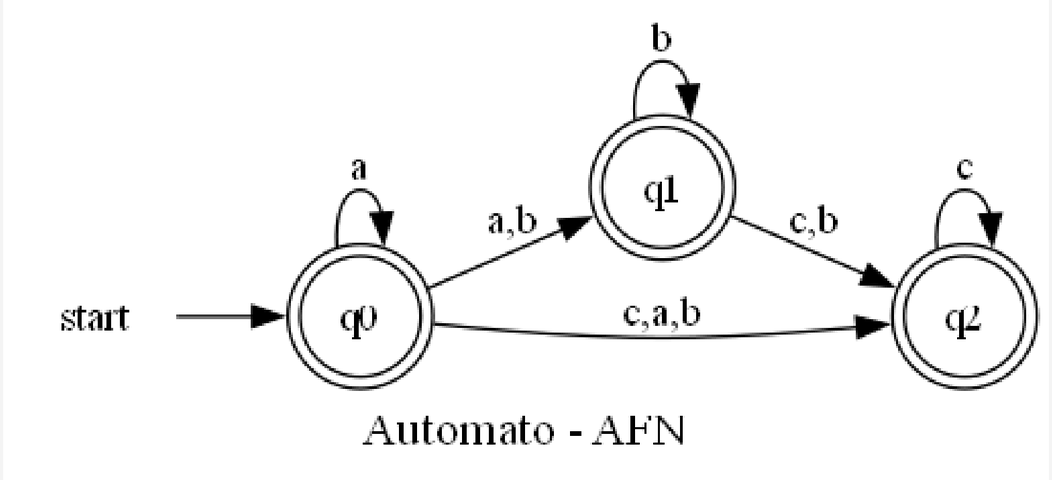


Figura 4.8 - AFN obtido pela conversão do AFe da Figura 4.7

CONCLUSÃO

- **Desenvolvimento de um Conversor Funcional**
- **Contribuição significativa**
- **Potencial para expansão**

PROGRAMA EM FUNCIONAMENTO

REFERÊNCIAS

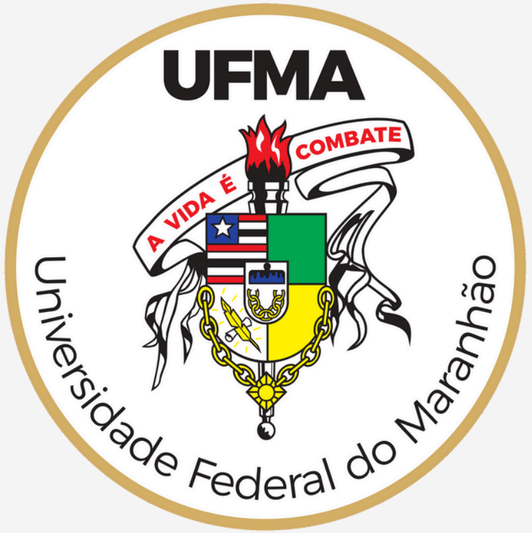
RAMOS, MARCUS V. M. LINGUAGENS FORMAIS: TEORIA, MODELAGEM E IMPLEMENTAÇÃO. 1ª ED. PORTO ALEGRE: BOOKMAN, 2009.

PAULO BLAUTH MENEZES. LINGUAGENS FORMAIS E AUTÔMATOS: VOLUME 3 DA SÉRIE LIVROS DIDÁTICOS INFORMÁTICA UFRGS. [S.L.] BOOKMAN EDITORA, 2009.

VALENTE, THALES. FORMAL LANGUAGES AND AUTOMATA. 2023. DISPONÍVEL EM: <HTTPS://GITHUB.COM/THALESVALENTE/TEACHING/TREE/MAIN/FORMAL-LANGUAGES-AND-AUTOMATA/1-LESSONS>. ACESSO EM: 10 JAN. 2025.

RUI, JOSÉ. LFA10 - AFNE TO AFN. [VÍDEO]. PUBLICADO EM: 8 FEV. 2021. DISPONÍVEL EM: <HTTPS://WWW.YOUTUBE.COM/WATCH?V=RUTXRCZI9CG>. ACESSO EM: 10 JAN. 2025.

MCNAUGHTON, R. (1961). THE THEORY OF AUTOMATA, A SURVEY. ADVANCES IN COMPUTERS VOLUME 2, 379–421.



OBRIGADO

CONVERSOR DE AUTÔMATO FINITO COM MOVIMENTOS VAZIOS (AFE) PARA AUTÔMATO FINITO NÃO DETERMINÍSTICO (AFN)

Orientador: Dr. Thales L. A. Valente

FERNANDA SOUSA DE ASSUNÇÃO VALE

JHONES DE SOUSA SOARES