Tiago Nunes de Cerqueira

Trabalho Prático 1 - Fundamentos Teóricos da Computação

Brasil

2024

Tiago Nunes de Cerqueira

Trabalho Prático 1 - Fundamentos Teóricos da Computação

Trabalho com o objetivo de simular um autômato finito determinístico. Documento feito utilizando do código fonte da equipe abnTEX2.

Pontifícia Universidade Católica de Minas Gerais – PUC-MG

Faculdade de Ciência da Computação

Programa de Graduação

Professor: Zenilton Kleber Gonçalves do Patrocínio Júnior

Brasil

2024

Sumário

Lista de ilustrações

1 Introdução

Neste trabalho foi desenvolvidos um código em **Python** com o objetivo de simular um Autômato Finito Determinístico (AFD). Um AFD M é formado da seguinte forma $M = (E, \Sigma, \delta, i, F)$, sendo:

- $E \equiv$ Conjunto finito de estados vazios;
- $\Sigma \equiv$ Alfabeto de entrada;
- $\delta \equiv$ Função de transição (função total):
  - $\delta : E \times \Sigma \mapsto E$
- $i \equiv$ Estado inicial
- $F \equiv$ Conjunto de estados finais $F \subseteq E$

Para a simulação do AFD capaz de avaliar se uma sentença pertence a uma expressão regular é preciso:

1. Converter a Expressão Regular para um AFN
2. Converter o AFN para um AFD

2 Implementação

A implementação do algoritmo foi feita da seguinte forma:

1. Criação de um arquivo .jff contendo a Expressão Regular, esta foi feita utilizando o aplicativo do jflap
2. Algoritmo em python
   a) Converter arquivo .jff para Regex
   b) Converter arquivo Regex para um AFN
   c) Converter AFN para um AFD
   d) Converter arquivo AFD para xmj
   e) Gerar um arquivo com sentenças para teste
   f) Simular o AFD
3. Testar arquivo xmj no jflap para verificar se a simulação ocorreu corretamente

2.1 main

Na main estão as chamadas das classes que realizam os processos de converter arquivos e construir o afd, nela também está a função "frases_generator"que é responsável por criar o conjunto de testes do autômato. A função de geração de testes gera 10 sentenças contendo de 10 a 50 caracteres e todos pertencentes a linguagem.

2.2 Conversor_jff_regex

Esta classe simplesmente converte o arquivo jff. para um regex.

2.3 Conversor_Regex_NFA

1. A expressão regular é processada para adicionar concatenações implícitas e convertida para notação postfix.
2. Uma árvore de expressão é construída a partir da notação postfix.
3. Um E-NFA é gerado a partir da árvore de expressão.
4. As transições do E-NFA são arranjadas

5. O E-NFA é salvo em um arquivo JSON.

2.4 Conversor_NFA_DFA

1. Carregamento do NFA: O código lê um arquivo JSON contendo a definição do NFA.
2. Inicialização do DFA: Define a estrutura do DFA e inicializa variáveis e mapeamentos de estados.
3. Conversão de NFA para DFA:
   - Calcula o fechamento-ε do estado inicial do NFA.
   - Usa uma fila (*deque*) para processar estados não marcados do DFA.
   - Para cada estado do DFA e símbolo de entrada, calcula o conjunto de estados de destino, incluindo fechamentos-ε.
   - Mapeia novos estados não vistos e adiciona transições ao DFA.
4. Definição de Estados Finais: Identifica e mapeia os estados finais do DFA com base nos estados finais do NFA.
5. Remapeamento de Transições: Converte a função de transição do DFA para usar nomes de estados remapeados.
6. Exportação do DFA: Salva o DFA resultante em um arquivo JSON.

2.5 Converter_json_xml

- O arquivo JSON do DFA é aberto e carregado.
- Os estados do DFA são mapeados para identificadores únicos.
- O XML é construído com a estrutura de estados e transições.
- Coordenadas para visualização são geradas aleatoriamente.
- Estados iniciais e finais são marcados no XML.
- O XML gerado é salvo em um arquivo xmj.

2.6 Dfa_Simulator

Uma variável representa o estado atual do afd e começa como o estado inicial, ao ler o carácter da entrada é feita uma busca por uma transição deste estado para algum, caso aja a variável que marca o estado atual é atualizada e o próximo caractere é analisado. O programa encerra quando o caractere inserido não faz parte do alfabeto ou quando todos já foram lidos, neste segundo caso é analisado se o estado atual é final

3 Resultados

Os resultados foram muito satisfatórios, conseguindo gerar afd's para as expressões regulares, além disso a simulação ocorre perfeitamente.

3.1 Teste no Algoritmo

3.1.1 Expressão Regular: $(a + b + c)^*c$

- **Sentenças:**
  - cccbabaacbbbabbb
  - caabbaccaca
  - bbccabaabcacabba
  - bbccabcaabbaccbcabacaaba
  - ccccbcbcccaabcacccaaaabbaaaaacaaccbaacbccc
  - cbacbccbcaccaccaa
  - ccaccbcbabbbbcbaabaabccbccbcacbcccccaaccbbc
  - cbbacacbbbaccbbaaacaccaccccb
  - aaaababacaccaccb
  - cbcbcbabbacacbaaccacbaabcccbbbbbcc

Figura 1 – Simulação Terminal 1

```
Qual ER usar:

1: t1.jff
2: t2.jff
3: t3.jff
4: t4.jff

> 4
A sentença 'cccbabaacbbbabbb' é aceita pelo AFD: False
A sentença 'caabbaccaca' é aceita pelo AFD: False
A sentença 'bbccabaabcacabba' é aceita pelo AFD: False
A sentença 'bbccabcaabbaccbcabacaaba' é aceita pelo AFD: False
A sentença 'ccccbcbcccaabcacccaaaabbaaaaacaaccbaacbccc' é aceita pelo AFD: True
A sentença 'cbacbccbcaccaccaa' é aceita pelo AFD: False
A sentença 'ccaccbcbabbbbcbaabaabccbccbcacbcccccaaccbbc' é aceita pelo AFD: True
A sentença 'cbbacacbbbaccbbaaacaccaccccb' é aceita pelo AFD: False
A sentença 'aaaababacaccaccb' é aceita pelo AFD: False
A sentença 'cbcbcbabbacacbaaccacbaabcccbbbbbcc' é aceita pelo AFD: True
PS F:\6º Periodo\FTC\TP2-FTC>
```

3.1.2 Expressão Regular: $(a^* + ab^*)$

- **Sentenças:**
  - ababaaaaabaaaababbbaba
  - babbaaabbbabbbaabbbbaa
  - aabbabbaabbbbabaaabaaabaaaabbababababbabaabbb
  - ababaaabbabaaabbbbbbbaaaaabaabbaabaaa
  - bbabaaabababaaaabababbaaabbbabbaaabababbabaabbababbbb
  - aaaababbba
  - bababbbbaabbb
  - babbbbaabaabaaaaabababaabababbabbaaba
  - babaaababaabbaabaaaabbabbbbbbaabbbbbbbbaaa
  - bbaabababbaaabbaaaabbaabba

Figura 2 – Simulação Terminal 2

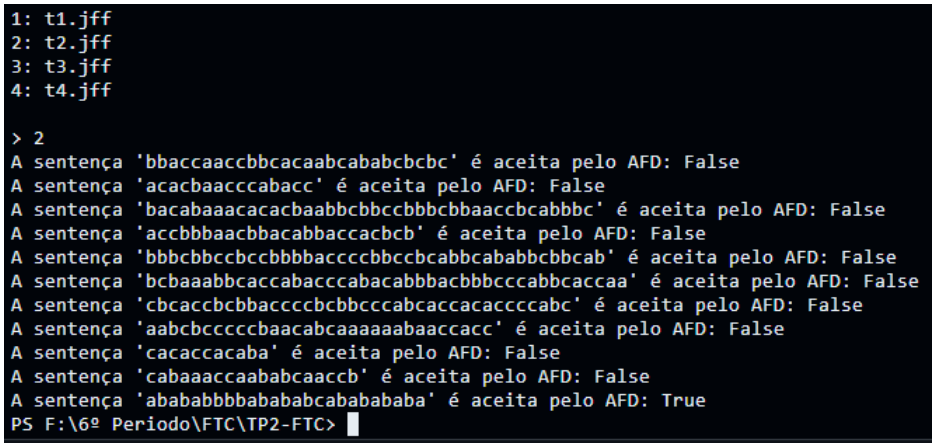
```
1: t1.jff
2: t2.jff
3: t3.jff
4: t4.jff

> 1
A sentença 'ababaaaaabaaaababbbaba' é aceita pelo AFD: False
A sentença 'babbaaabbbabbbaabbbbaa' é aceita pelo AFD: False
A sentença 'aabbabbaabbbbabaaabaaabaaaabbababababbabaabbb' é aceita pelo AFD: False
A sentença 'ababaaabbabaaabbbbbbbaaaaabaabbaabaaa' é aceita pelo AFD: False
A sentença 'bbabaaabababaaaabababbaaabbbabbaaabababbabaabbababbbb' é aceita pelo AFD: False
A sentença 'aaaababbba' é aceita pelo AFD: False
A sentença 'bababbbbaabbb' é aceita pelo AFD: False
A sentença 'babbbbaabaabaaaaabababaabababbabbaaba' é aceita pelo AFD: False
A sentença 'babaaababaabbaabaaaabbabbbbbbaabbbbbbbbaaa' é aceita pelo AFD: False
A sentença 'bbaabababbaaabbaaaabbaabba' é aceita pelo AFD: False
PS F:\6º Periodo\FTC\TP2-FTC>
```

3.1.3 Expressão Regular: $(a + b)^*c(a + b)^*$

- **Sentenças:**
  - bbaccaaccbbcacaabcababcbcbc
  - acacbaacccabacc
  - bacabaaacacacbaabbcbbccbbbcbbaaccbcabbbc
  - accbbbaacbbacabbaccacbcb
  - bbbcbbccbccbbbbaccccbbccbcabbcababbcbbcab

- bcbaaabbcaccabacccabacabbbacbbbcccabbcaccaa
- cbcaccbcbbaccccbcbbcccabcaccacaccccabc
- aabcbcccccbaacabcaaaaaabaaccacc
- cacaccacaba
- abababbbbabababcabababababa

Figura 3 – Simulação Terminal 3



3.1.4 Expressão Regular: $(a + b)^*$

- **Sentenças:**
  - bbaaaaaaaabaaabaabbbbbbbabaabab
  - babbabbbabbbbbbbbbabaaaaababbaa
  - abbbaaaaabbbababbaabbbaabababbbaaabbaabb
  - babbabaaabaaaaaabbaabbbababbabbaba
  - aababaaaabaababbbabbbba
  - bbbbabbabbaaaabbabbbaabababbbaba
  - abaabaabbbababbbaaabbabbaabb
  - bbbaaabababbbbba
  - bbbaababbbaaabaababbaabbbaaaabb
  - baaababaababbbaabaabab

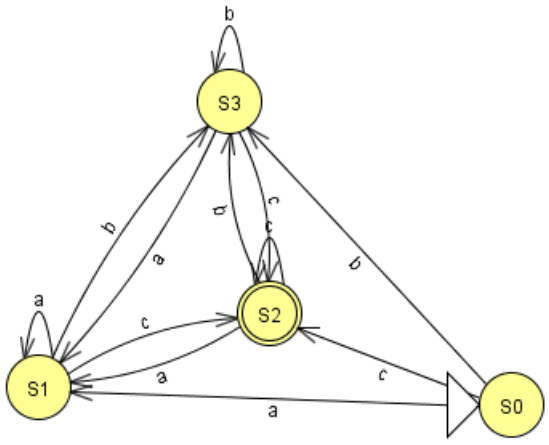
Figura 4 – Simulação Terminal 4

```
1: t1.jff
2: t2.jff
3: t3.jff
4: t4.jff

> 3
A sentença 'bbaaaaaaaabaaabaabbbbbbbabaababb' é aceita pelo AFD: True
A sentença 'babbabbbabbbbbbbbbbabaaaaaabbbaa' é aceita pelo AFD: True
A sentença 'abbbaaaaabbbababbaabbbaabababbbaaabbaabb' é aceita pelo AFD: True
A sentença 'babbabaaabaaaaaabbaabbbbababbabbaba' é aceita pelo AFD: True
A sentença 'aababaaaabaabababbbabbba' é aceita pelo AFD: True
A sentença 'bbbbabbabbaaaabbabbbaabababbbaba' é aceita pelo AFD: True
A sentença 'abaabaabbbabababbbaaabbabbaabb' é aceita pelo AFD: True
A sentença 'bbbaaabababbbbba' é aceita pelo AFD: True
A sentença 'bbbaabababbbaaabaabababbbaaaabb' é aceita pelo AFD: True
A sentença 'baaababaabababbbaabaabab' é aceita pelo AFD: True
A sentença 'aaababb' é aceita pelo AFD: True
PS F:\6º Periodo\FTC\TP2-FTC>
```

3.2 Teste no jflap

3.2.1 Expressão Regular: $(a + b + c)^*c$

Figura 5 – Simulação jflap



3.2.2 Expressão Regular: $(a^* + ab^*)$

3.2.3 Expressão Regular: $(a + b)^*c(a + b)^*$

3.2.4 Expressão Regular: $(a + b)^*$

Figura 6 – Simulação jflap 2

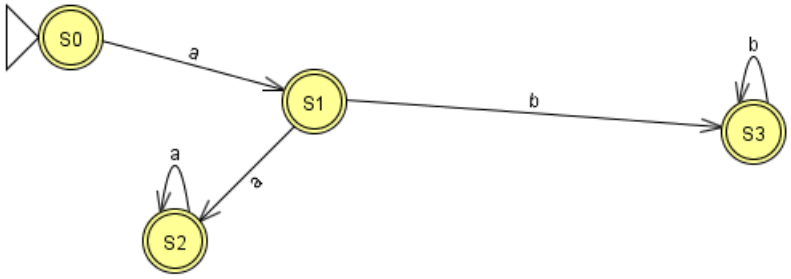


Figura 7 – Simulação jflap 3

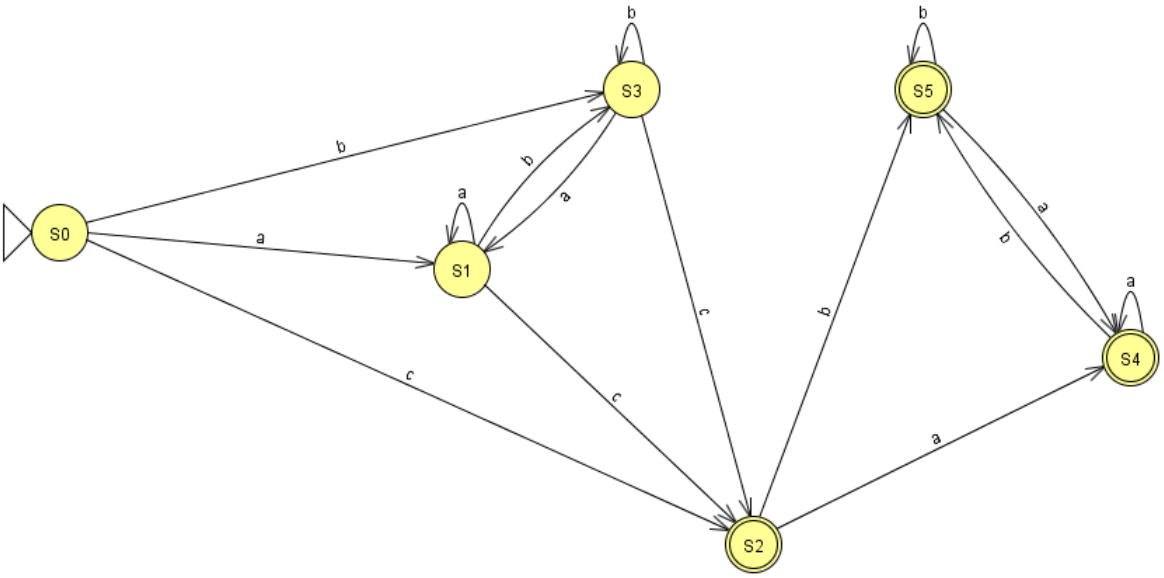


Figura 8 – Simulação jflap 4