Trabalho 01 - Simulador de Autômato Finito

Matheus Arataque Uema - 10276949

SCC5832 - Teoria da Computação

Resumo

Este relatório apresenta os conceitos estudados de autômatos finitos. Foi desenvolvido um simulador de autômato finito em *Python*, o qual foi avaliado a partir de cinco autômatos diferentes, três determinísticos e dois não-determinísticos.

1 Introdução

Autômatos Finitos são modelos matemáticos de sistemas computacionais que processam cadeias de símbolos e decidem se essas cadeias pertencem a uma determinada linguagem formal. Eles são constituídos por um alfabeto de símbolos terminais, um conjunto finito de estados - dentre os quais deve haver no mínimo um estado inicial e um de aceitação -, e um conjunto de funções de transições que guia a cadeia entre os estados. Um estado inicial determina onde a cadeia deve começar, e um estado de aceitação determina se determinada cadeia será aceita ou rejeitada.

Os Autômatos Finitos podem ser classificados em determinísticos (AFD) e não-determinísticos (AFN). Os AFDs devem possuir apenas uma transição para cada par de estado e símbolo terminal, tornando-o previsível. Por outro lado, os AFNs podem ter mais de uma transição para dado estado e símbolo, além de permitirem transições λ, onde não há consumo de símbolo, o que aumenta o número de possibilidades de caminhos de uma determinada cadeia. Apesar disso, vale ressaltar que qualquer linguagem reconhecida por um AFN também pode ser consumida por um AFD.

As linguagens reconhecidas pelos Autômatos Finitos são denominadas Linguagens Regulares. Essas linguagens podem ser regidas por Gramáticas Regulares, e são do tipo mais simples dentre aqueles descritos pela Hierarquia de Chomsky (tipo 3). Ao contrário das demais linguagens, essa não exige memória o que permite que ela seja processada por um AF. Por essa razão, linguagens mais complexas - Livres de Contexto (Tipo 2), Sensíveis ao Contexto (Tipo 1) e Irrestritas (Tipo 0) - não podem ser consumidas por um autômato finito. Para elas seria necessário um modelo com memória e maior poder computacional.

Este trabalho implementa a partir de um arquivo de entrada um **Autômato Finito** que pode determinar se uma determinada cadeia é ou não é aceita por

ele. O objetivo é avaliar o poder de simulação do programa a partir do seu desempenho para um conjunto de cadeias.

2 Metodologia

2.1 Dados

Para avaliar o código, foram preparados cinco autômatos finitos com 10 cadeias de teste para cada um. Dentre esses cinco autômatos, três são determinísticos e dois são não-determinísticos. Para cada autômato foi preparado um documento de texto com as entradas do autômato e as cadeias de testes seguindo o padrão da descrição de trabalho presente no Apêndice A.

A seguir estão presentes as Figuras 1, 2, 3, 4 e 5, montadas via *JFlap*, e Tabelas 1, 2, 3, 4 e 5 para cada autômato e as 10 cadeias usadas para teste assim como o resultado esperado para cada cadeia:

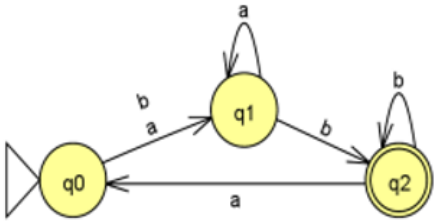


Figura 1: Autômato Finito 01

Cadeia	Resultado esperado
abbbba	Rejeita
aabbbb	Aceita
bbabbabbabbb	Aceita
bbbbbbbbbbb	Aceita
-	Rejeita
abababababab	Rejeita
bbbbaabbbb	Aceita
abba	Rejeita
a	Rejeita
aaa	Rejeita

Tabela 1: Cadeias para o Autômato Finito 01

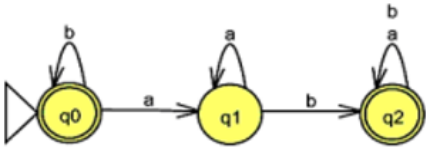


Figura 2: Autômato Finito 02

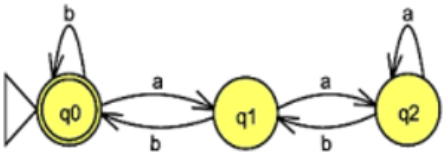


Figura 3: Autômato Finito 03

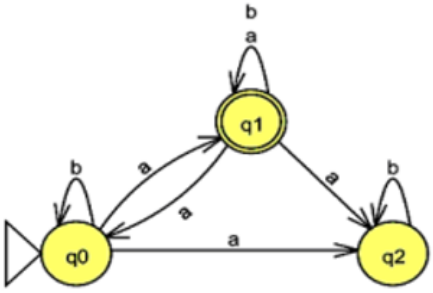


Figura 4: Autômato Finito 04

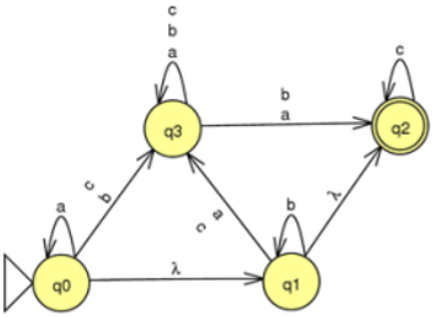


Figura 5: Autômato Finito 05

Cadeia	Resultado esperado
b	Aceita
baaaaaaababababab	Aceita
baabb	Aceita
baaba	Aceita
baba	Aceita
babb	Aceita
bbbb	Aceita
bbababa	Aceita
bbbaaaa	Rejeita
babbb	Aceita

Tabela 2: Cadeias para o Autômato Finito 02

Cadeia	Resultado esperado
-	Aceita
baaaaab	Rejeita
aab	Rejeita
aaabbbb	Aceita
ababbb	Aceita
bababababa	Rejeita
baaaaaaabbbbab	Aceita
babaaabbaabbb	Aceita
aabb	Aceita
aaaaabbbb	Aceita

Tabela 3: Cadeias para o Autômato Finito 03

Cadeia	Resultado esperado
bbbbbba	Aceita
baa	Aceita
baaab	Aceita
aaabbb	Aceita
aabba	Aceita
bbbbb	Rejeita
aabbb	Aceita
bb	Rejeita
bbb	Rejeita
bababa	Aceita

Tabela 4: Cadeias para o Autômato Finito 04

Cadeia	Resultado esperado
a–	Aceita
a	Aceita
a-bbbb-	Aceita
abbbb	Aceita
abcabc	Aceita
bbb	Aceita
bbc	Aceita
bba	Aceita
bc	Aceita
bcc	Aceita

Tabela 5: Cadeias para o Autômato Finito 05

2.2 Implementação

A implementação foi feita seguindo as especificações no Apêndice A em linguagem *Python*. O autômato é implementado usando três classes principais: `State`, `Transition` e `Finite_Automaton`. Estas classes modelam os estados do autômato, as transições entre estados e o próprio autômato como um todo.

Primeiro, foi feita uma leitura do arquivo de entrada para que o autômato possa ser definido e, então, as cadeias são registradas são testadas no autômato. O teste é feito de maneira recursiva onde é testado todas funções de transição de cada estado para determinado terminal, levando também em consideração a possibilidade de estados λ. Como condições de aceitação, é necessário o autômato esteja em um estado de aceitação quando o autômato estiver no final da cadeia, e basta que a cadeia seja aceita em apenas uma das possibilidades dentre as transições (caso seja um AFN). Por fim, o código escreve em um arquivo de saída os resultados obtidos.

Feita a implementação, testou-se os AF criados. Os autômatos e as cadeias foram determinados de forma a extrapolar possibilidades de erro no código. Os determinísticos, AF 1 e 2, foram feitos para que ele resolva os casos mais simples. Os autômatos não-determinísticos AF 3 e 4 foram criados para garantir a execução correta do simulador para casos de múltiplas possibilidades de transição. E por fim, o AF 5 visa que a simulação correta em casos de transição λ.

Ao fim da implementação e testes, com a biblioteca *pyinstaller* criou-se um arquivo executável do programa de forma a garantir a execução de todos requisitos considerados ao projeto.

A implementação completa do código, assim como os arquivos de entrada utilizados, e de saída obtidos, estão presentes em um repositório do código no *GitHub*. No link também é possível encontrar um manual para execução do arquivo `.exe`[1]

[1]Repositório do Código Disponível em: https://github.com/MatheusUema/finite-

3 Discussão

Ao aplicar o arquivo executável nos códigos, obteve-se os resultados presentes nas Tabelas 6, 7, 8, 9 e 10. Vê-se que para todos os casos, o simulador gerou uma saída condizente com os resultados esperados, o que pode indicar um desempenho aceitável para o código criado. Somado a isso, como foram utilizados diversos tipos de autômatos, acredita-se que o simulador possa representar bem qualquer autômato finito que seja testado. Entretanto, como os mesmos autômatos utilizados para testar o código foram aqueles utilizados como avaliação, é possível que apareçam erros não considerados quando novos autômatos forem testados, visto que alguns erros não foram testados no desenvolvimento do código como o uso de terminais nas cadeias não presentes no autômato, e autômatos com múltiplos estados iniciais e/ou de aceitação.

Cadeia	**Resultado esperado**	**Resultado do simulador**
abbbba	Rejeita	Rejeita
aabbbb	Aceita	Aceita
bbabbabbabbb	Aceita	Aceita
bbbbbbbbbbb	Aceita	Aceita
-	Rejeita	Rejeita
abababababab	Rejeita	Rejeita
bbbbaabbbb	Aceita	Aceita
abba	Rejeita	Rejeita
a	Rejeita	Rejeita
aaa	Rejeita	Rejeita

Tabela 6: Resultados para o Autômato Finito 01

Cadeia	**Resultado esperado**	**Resultado do simulador**
b	Aceita	Aceita
baaaaaaababababab	Aceita	Aceita
baabb	Aceita	Aceita
baaba	Aceita	Aceita
baba	Aceita	Aceita
babb	Aceita	Aceita
bbbb	Aceita	Aceita
bbababa	Aceita	Aceita
bbbaaaa	Rejeita	Rejeita
babbb	Aceita	Aceita

Tabela 7: Resultados para o Autômato Finito 02

automaton/tree/main. Acesso em: 20 de out. de 2024.

Cadeia	Resultado esperado	Resultado do simulador
-	Aceita	Aceita
baaaaab	Rejeita	Rejeita
aab	Rejeita	Rejeita
aaabbbb	Aceita	Aceita
ababbb	Aceita	Aceita
bababababa	Rejeita	Rejeita
baaaaaaabbbbab	Aceita	Aceita
babaaabbaabbb	Aceita	Aceita
aabb	Aceita	Aceita
aaaaabbbb	Aceita	Aceita

Tabela 8: Resultados para o Autômato Finito 03

Cadeia	Resultado esperado	Resultado do simulador
bbbbbba	Aceita	Aceita
baa	Aceita	Aceita
baaab	Aceita	Aceita
aaabbb	Aceita	Aceita
aabba	Aceita	Aceita
bbbbb	Rejeita	Rejeita
aabbb	Aceita	Aceita
bb	Rejeita	Rejeita
bbb	Rejeita	Rejeita
bababa	Aceita	Aceita

Tabela 9: Resultados para o Autômato Finito 04

Cadeia	Resultado esperado	Resultado do simulador
a–	Aceita	Aceita
a	Aceita	Aceita
a-bbbb-	Aceita	Aceita
abbbb	Aceita	Aceita
abcabc	Aceita	Aceita
bbb	Aceita	Aceita
bbc	Aceita	Aceita
bba	Aceita	Aceita
bc	Aceita	Aceita
bcc	Aceita	Aceita

Tabela 10: Resultados para o Autômato Finito 05

4 Conclusão

O simulador de autômatos criado atende todos os requisitos da especificação e, dado os testes realizados, aparenta possuir um bom desempenho na representação dos autômatos finitos. Somado a isso, a implementação do código possibilitou o aprofundamento dos conceitos de Autômatos Finitos, assim como a visualização na prática do seu funcionamento. Por fim, acredita-se que mais testes com novos autômatos finitos poderiam ser realizados para uma garantia da generalização do código.

A Descrição do Trabalho

ICMC-USP

Trabalho em Grupo 1

SCC-5832 e SCC-0205

2^o. Semestre de 2024
V1 - 18/9/2024

1 Objetivo

Desenvolver o entendimento de Linguagens Formais e seu potencial de representação através da implementação de simuladores de autômatos finitos.

2 Descrição

O trabalho deve ser realizado em grupos de no máximo cinco alunos. Cada grupo deve projetar e desenvolver a aplicação abaixo, empregando qualquer linguagem de programação.

- *Simulador Universal de Autômatos Finitos*: O programa deve aceitar a especificação de um AFD ou AFN e a partir daí para uma dada lista de cadeias, dizer quais as que pertencem (saída: **aceita**) e quais as que não pertencem (saída: **rejeita**) à linguagem reconhecida pelo autômato.

3 Produto

O programa a ser implementado neste projeto deve seguir rigorosamente os formatos de entrada e saída (ver seção "Arquivos Texto de Entrada e de Saída" abaixo), e enviado ao Escaninho de um membro do grupo na plataforma Tidia-Ae 4.0 (`ae4.tidia-ae.usp.br`), na página da disciplina SCC-5832, ou na página da disciplina SCC-0205 - turma 1, até às 23h59 do dia **20 de outubro de 2024**. **O prazo final é improrrogável**. Além do programa fonte e um executável em Windows, um relatório com a descrição do trabalho deverá ser entregue (ver seção "Critérios" abaixo).

4 Critérios

Os critérios de correção dos trabalhos são:

1. (80%) **Implementação**: O programa funciona corretamente para todos os casos de teste;

2. (20%) **Documentação**: Relatório simples que explica as técnicas utilizadas para implementar a máquina escolhida. Discutir a qualidade da solução implementada, a estruturação do código e a eficiência da solução em termos de espaço e tempo. A documentação deverá ser submetida ao Tidia-Ae, juntamente com o código fonte e um arquivo executável em Windows. **IMPORTANTE**: Incluir explicações claras sobre como executar o programa em um arquivo texto de nome **manualT1.txt**. Portanto,

 Continua na próxima página...

quatro arquivos devem ser anexados: o relatório (arquivo txt ou PDF), o arquivo fonte (arquivo txt), executável e manual.

Atenção: **O plágio (cópia) de programas não será tolerado. Quaisquer programas similares terão nota zero independente de qual for o original e qual for a cópia.**

5 Arquivos Texto de Entrada e de Saída

Arquivo Texto de Entrada:

- 1^a. Linha: número de estados: para o conjunto de estados Q, assume-se os nomes dos estados de q_0 a q_{n-1}, onde n é o número de estados (Obs.: q_0 é o estado inicial, quando houver um único estado inicial (AFD)). Assuma $1 \leq n \leq 10$;
- 2^a. Linha: o conjunto de símbolos terminais (Σ): entrar com a quantidade de símbolos terminais seguida dos elementos separados por espaço simples. Assume-se tamanho máximo igual a 10;
- 3^a. Linha: o número de estados iniciais (se for AFD, é igual a 1: q_0; se for AFN, usa-se q_0, q_1, etc. para os estados iniciais[1]). Assume-se tamanho máximo igual a 10;
- 4^a. Linha: o conjunto de estados de aceitação (F): entrar com a quantidade de estados de aceitação seguida dos elementos separados por espaços. Lembre-se de entrar apenas com os números de 0 a 9;
- 5^a. Linha: o número de transições (δ) da máquina (máximo de 50).
- a partir da 6^a Linha: as transições: entra-se com um δ em cada linha, com os elementos separados por espaço: $q\ x\ q'$, onde $q, q' \in Q$, $x \in \Sigma \cup \{\lambda\}$. Represente a cadeia vazia (λ) como "-".
- Linha depois das transições: entrar com o número de cadeias de entrada (máximo de 10).
- Próximas Linhas: cadeias de entrada: entrar com uma em cada linha. Comprimento máximo de cada cadeia = 20 símbolos.

Arquivo Texto de Saída:

[1] Alguns autores admitem a possibilidade de haver mais de um estado inicial quando o autômato é não-determinístico.

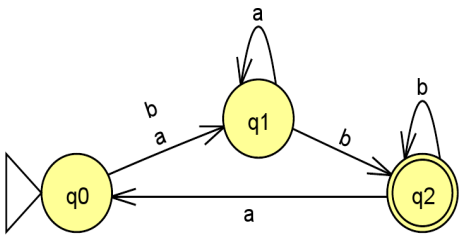
 Continua na próxima página...

- a partir da 1^a. Linha: a informação sobre a aceitação ou não da respectiva cadeia de entrada, **na ordem** do arquivo de entrada. Se a cadeia de entrada pertencer à linguagem reconhecida pelo autômato, a cadeia de saída será "aceita". Caso a cadeia de entrada não pertença à linguagem reconhecida pelo autômato, a cadeia de saída será "rejeita".

 Continua na próxima página...

6 Exemplo

- Autômato finito determinístico (AFD) que processa a linguagem regular $(a+b)a^*bb^*(a(a+b)a^*bb^*)^*$.



Arquivo Texto de Entrada[2]:

1. 3
2. 2 a b
3. 1
4. 1 2
5. 6
6. 0 a 1
7. 0 b 1
8. 1 a 1
9. 1 b 2
10. 2 a 0
11. 2 b 2
12. 10
13. abbbba
14. aabbbb
15. bbabbabbabbb
16. bbbbbbbbbbb
17. -
18. abababababab
19. bbbbaabbbb
20. abba
21. a
22. aaa

[2]Os números das linhas **não** devem aparecer no arquivo-texto. Estão colocados aqui apenas para facilitar o entendimento.

 Continua na próxima página...

Arquivo Texto de Saída:

1. rejeita
2. aceita
3. aceita
4. aceita
5. rejeita
6. rejeita
7. aceita
8. rejeita
9. rejeita
10. rejeita