Trabalho 02 - Simulador da Máquina de Turing

Matheus Arataque Uema - 10276949

SCC5832 - Teoria da Computação

Resumo

Este relatório apresenta os conceitos estudados sobre Máquinas de Tring. Foi desenvolvido um simulador de Máquina de Turing em *Python*, o qual foi avaliado a partir de três modelos diferentes. Observou-se que o simulador acertou em todos testes realizados. Também foi analisada a qualidade do código implementado.

1 Introdução

Máquinas de Turing são modelos de computação universal, conseguindo resolver qualquer problema computacional algoritmicamente solucionável. Para isso, possuem uma fita de memória - teoricamente infinita - para leitura e escrita de símbolos. Isso é feito através de uma cabeça de leitura que pode se movimentar entre as posições da fita, e manipular os símbolos nela presente. O movimento da cabeça de leitura é descrito por funções de transições complexas que definem seu comportamento com base no seu estado atual e no símbolo lido da fita. A partir dessas informações, ela decide a direção de movimento (direita, esquerda, ou permanecer) e o símbolo a ser escrito. Por fim, dentre os estados da MT, pelo menos um deve ser de parada para avaliar se determinado algoritmo está validado ou não.

Se comparada ao tema do trabalho anterior sobre Autômatos Finitos, as MTs possuem capacidade computacional e memória superiores, o que as permitem aceitar linguagens recursivamente enumeráveis, não apenas linguagens regulares como os AFs. As linguagens recursivamente enumeráveis são aquelas de tipo 0 na hierarquia de Chomsky.

Em termos de desempenho, a máquina de Turing depende do problema a ser resolvido, podendo levar um tempo exponencial ou mesmo não chegar em uma resposta finita, como seria o caso de problemas indecidíveis. Por conta disso, em relação a modelos modernos elas são consideradas lentas, mesmo que teoricamente elas possam computar qualquer coisa que os computadores atuais conseguem. Por fim, ela serve como base para simulações de algoritmos, o que as torna ideais para simulação de linguagens do tipo 0.

2 Metodologia

2.1 Dados

Para avaliar o código, foram preparados modelos de Máquinas de Turing para três linguagens, cada um com uma quantidade de cadeias testes os quais foram previamente testados pelo autor no *JFlap*.Para cada modelo, um documento de texto foi feito com todas as configurações da máquina de turing, assim como da linguagem, e com o conjunto de cadeias de teste utilizado. Isso foi feito seguindo o padrão da descrição de trabalho presente no Apêndice A.

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

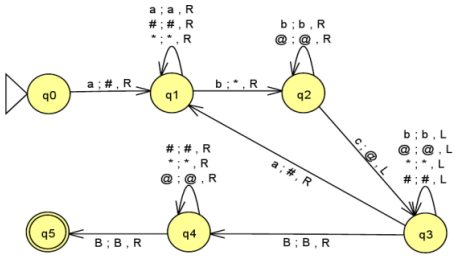


Figura 1: Máquina de Turing 01 que processa a linguagem $a^n b^n c^n$, sendo $n > 0$

Cadeia	**Resultado esperado**
abbcca	Rejeita
aabbcc	Aceita
bac	Rejeita
aaabbbcccc	Rejeita
-	Rejeita
abcabc	Rejeita
abc	Aceita
abcc	Rejeita
c	Rejeita
aaabbbbccc	Rejeita

Tabela 1: Cadeias para o MT 01

2.2 Implementação

A implementação foi feita seguindo as especificações no Apêndice A em linguagem *Python*. A máquina de Turing é implementado usando três classes principais: `State`, `Transition` e `Turing_Machine`. Estas classes modelam os estados

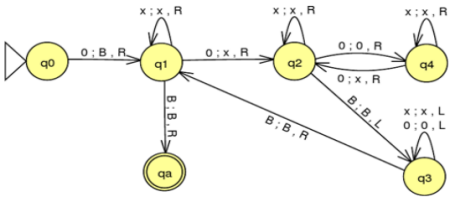


Figura 2: Máquina de Turing 02 que processa a linguagem $L = \{0^k,\ onde\ k = 2^n \mid n \geq 0\}$

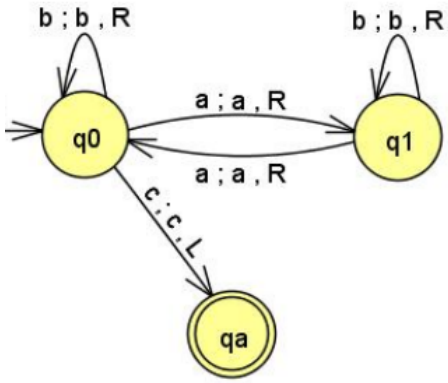


Figura 3: MT 03 que processa a Linguagem $L = \{w \mid w(a + b)^*\ com\ número\ par\ de\ a's\}$.

Cadeia	Resultado esperado
0	Aceita
00	Aceita
000	Rejeita
0000	Aceita

Tabela 2: Cadeias para MT 02

Cadeia	Resultado esperado
abaa	Rejeita
aabaa	Aceita
abbbbba	Aceita
abba	Aceita
a	Rejeita
aba	Aceita

Tabela 3: Cadeias para a MT 03

da máquina de Turing, as transições entre estados e o próprio autômato como um todo. Foi adaptado o desenvolvimento feito no trabalho anterior, alterando a leitura do arquivo, o modelo de transições e a simulação das cadeias.

Primeiro, foi feita uma leitura do arquivo de entrada para que a máquina seja configurada e, então, as cadeias são registradas e testadas. O teste é feito de maneira recursiva onde é testado a função de transição de cada estado para determinado terminal, até que se chegue a um estado de aceitação. Como condições de aceitação, é necessário. Por fim, o código escreve em um arquivo de saída os resultados obtidos.

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]

Analisando a implementação, obteve-se as seguintes observações para complexidade de tempo:

- Inicialização de Estados e Transições: essa etapa leva tempo $O(n)$, sendo n a quantidade de linhas do arquivo de entrada.

- Processamento de Cadeias: aqui itera-se pelas cadeias de forma recursiva passando pelas transições em cada estado. O pior caso encontra-se entre

[1]Repositório do Código Disponível em: https://github.com/MatheusUema/finite-automaton/tree/main. Acesso em: 13 de nov. de 2024.

uma $O(n)$ ou $O(2^n)$, a depender da complexidade da linguagem analisada.

Em termos de espaço, acredita-se que as etapas podem ser consideradas todas $O(n)$.

3 Discussão

Ao aplicar o arquivo executável nos códigos, obteve-se os resultados presentes nas Tabelas 4, 5 e 6. Vê-se que para todos os casos, o simulador gerou uma saída igual a dos resultados esperados, indicando um desempenho aceitável para o código criado. Como foram utilizados modelos diferentes de máquinas de Turing, afirma-se que o simulador tenha capacidade de simular qualquer máquina de Turing que seja desejado.

Cadeia	**Resultado esperado**	**Resultado do simulador**
abbcca	Rejeita	Rejeita
aabbcc	Aceita	Aceita
bac	Rejeita	Rejeita
aaabbbcccc	Rejeita	Rejeita
-	Rejeita	Rejeita
abcabc	Rejeita	Rejeita
abc	Aceita	Aceita
abcc	Rejeita	Rejeita
c	Rejeita	Rejeita
aaabbbbccc	Rejeita	Rejeita

Tabela 4: Resultados para MT 01

Cadeia	**Resultado esperado**	**Resultado do simulador**
0	Aceita	Aceita
00	Aceita	Aceita
000	Rejeita	Rejeita
0000	Aceita	Aceita

Tabela 5: Resultados para MT 02

4 Conclusão

O simulador criado atende todos os requisitos da especificação e, dado os testes realizados, aparenta possuir um bom desempenho na representação das máquinas de Turing. Somado a isso, a implementação do código possibilitou

Cadeia	**Resultado esperado**	**Resultado do simulador**
abaa	Rejeita	Rejeita
aabaa	Aceita	Aceita
abbbbba	Aceita	Aceita
abba	Aceita	Aceita
a	Rejeita	Rejeita
aba	Aceita	Aceita

Tabela 6: Resultados para MT 03

o aprofundamento dos conceitos de Máquinas de Turing, assim como sobre conceitos de análise de performance de tempo e de espaço.

A Descrição do Trabalho

ICMC-USP

Trabalho em Grupo 2

SCC-5832 - Teoria da Computação

2^o. Semestre de 2024
Professor: João Luís G.Rosa - e-mail: joaoluis@icmc.usp.br
Estagiário PAE: Marcela Tiemi Shinzato - e-mail: marcelats@usp.br
22/11/2024

1 Objetivo

Desenvolver o entendimento de Linguagens Formais e seu potencial de representação através da implementação de simuladores de máquinas de Turing.

2 Descrição

O trabalho deve ser preferencialmente realizado em grupos de no máximo cinco alunos. Cada grupo deve projetar e desenvolver um *Simulador Universal de Máquinas de Turing*, empregando qualquer linguagem de programação. Um documento com instruções de execução deve ser anexado ao programa fonte, além do executável:

- *Simulador Universal de Máquinas de Turing*: O programa deve aceitar a especificação de uma máquina de Turing determinística 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 pela máquina.

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 (`https://ae4.tidia-ae.usp.br/portal/site/`), na página da disciplina SCC5832_Ver5_7_2024_CT, até às 23h59 do dia **22 de novembro 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,

 Continua na próxima página...

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 **manualT2.txt**. Não compacte os arquivos. Portanto, quatro arquivos separados 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). Portanto, basta entrar com o número de estados. Assuma $1 \leq n \leq 10$;
- 2^a. Linha: o conjunto de símbolos terminais - alfabeto de fita (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: a quantidade de símbolos de Σ' (alfabeto estendido de fita) não presentes em Σ, seguido pelos símbolos, separados por espaço simples;
- 4^a. Linha: o estado de aceitação: entrar com o estado de aceitação (q_a). 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: $q\ x\ q'\ y\ D$, onde $q, q' \in Q$, $x, y \in \Sigma'$ e $D \in \{R, L, S\}$. Assuma os limites da fita com o símbolo B, que aqui representa o branco ($\square$). 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.

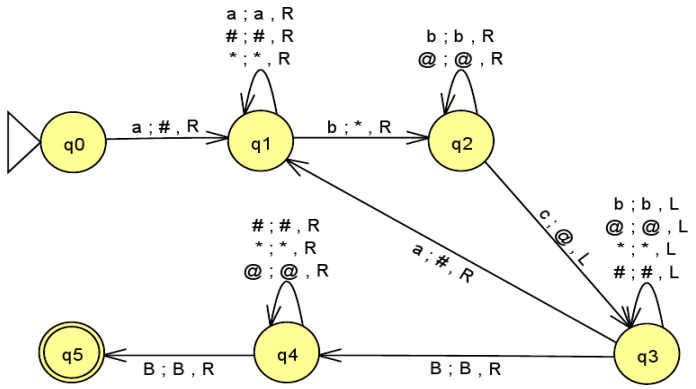
 Continua na próxima página...

Arquivo Texto de Saída:

- a partir da 1ª. 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".

6 Exemplo

- Exemplo: Máquina de Turing determinística que processa a linguagem $a^n b^n c^n$, com $n > 0$.



Arquivo Texto de Entrada[1]:

1. 6
2. 3 a b c
3. 4 # * @ B
4. 5
5. 18
6. 0 a 1 # R
7. 1 # 1 # R
8. 1 a 1 a R
9. 1 * 1 * R
10. 1 b 2 * R
11. 2 b 2 b R

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

 Continua na próxima página...

12. 2 @ 2 @ R
13. 2 c 3 @ L
14. 3 # 3 # L
15. 3 * 3 * L
16. 3 @ 3 @ L
17. 3 b 3 b L
18. 3 a 1 # R
19. 3 B 4 B R
20. 4 # 4 # R
21. 4 * 4 * R
22. 4 @ 4 @ R
23. 4 B 5 B R
24. 10
25. abbcca
26. aabbcc
27. bac
28. aaabbbcccc
29. -
30. abcabc
31. abc
32. abcc
33. c
34. aaabbbbccc

Arquivo Texto de Saída:

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

 Continua na próxima página...

7 Notas

1. Apenas máquinas determinísticas serão testadas.

2. Apenas linguagens recursivas serão testadas, ou seja, linguagens para as quais a máquina de Turing sempre para independentemente da aceitação.

3. O processamento da cadeia começa no estado q_0 e no símbolo mais à esquerda de w. Exemplo: para a fita $abba$, a descrição instantânea inicial é q_0abba.