# Relatório sobre Máquina de Turing

**Braz G. S. Souza[1], Davi B. Franco[1], Gabriel S. Amaral[1]**

[1]Faculdade de Computação – Universidade Federal do Pará (UFPA)

Caixa postal 479. PABX +55 91 3201-7000. Belém - Pará - Brasil

{braz.souza,davi.franco,gabriel.amaral}@icen.ufpa.br

***Resumo.*** *Este relatório se baseia na resolução de um conjunto de problemas que requerem a criação de máquinas de Turing. Para a resolução foi escolhida a linguagem de programação Python, se utilizando de figuras, para explicar e obter um programa que alcance o objetivo desejado da máquina. Tomando, por fim, testes experimentais que comprovam a qualidade e funcionalidade das máquinas produzidas durante o relatório.*

***Abstract.*** *This report is based on solving a set of problems that require the creation of Turing machines. For the resolution, the Python programming language was chosen, using figures, to explain and obtain a program that achieves the desired objective of the machine. Taking, finally, experimental tests that prove the quality and functionality of the machines produced during the report.*

## 1. Introdução

Em princípio, para a compreensão dos programas desenvolvidos a seguir, é necessário entender o básico do funcionamento de Máquinas de Turing, que são dispositivos não determinísticos de reconhecimento de cadeias. Em que analisa uma fita, podendo ser limitada ou ilimitada a direita, e sempre limitada a esquerda pelo símbolo "<". Uma representação de uma máquina de turing com fita ilimitada a direita pode ser observada na Figura 1.

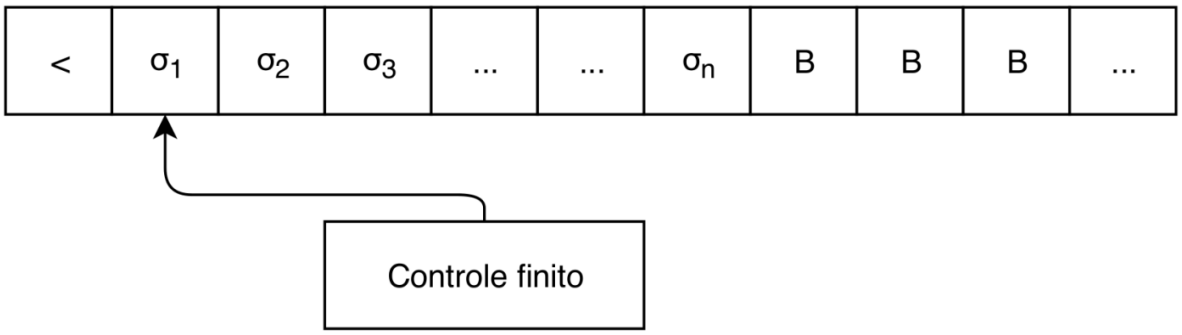


**Figura 1.** Máquina de turing

Também sendo definido por uma óctupla apresentada na equação 1. Onde Q é o conjunto finito de estados, Σ é o alfabeto de entrada, Γ é um conjunto finito de símbolos que podem ser lidos e escritos na fita, δ é a função de transição, q0 é o estado inicial, < é a primeira posição da fita de trabalho, não podendo ser gravado, B preenche todas as posições à direita da cadeia, podendo ser gravado em qualquer posição, representando o vazio, F é o conjunto de estados finais.

$$M = (Q, \Sigma, \Gamma, \delta, q0, <, B, F)$$

**Equação 1.** Definição algébrica da Máquina de Turing.

A Máquina de Turing também pode representar linguagens por meio de figuras, como se observa a Figura 2, que representa a linguagem L(M) = abc(a | b | c)*.

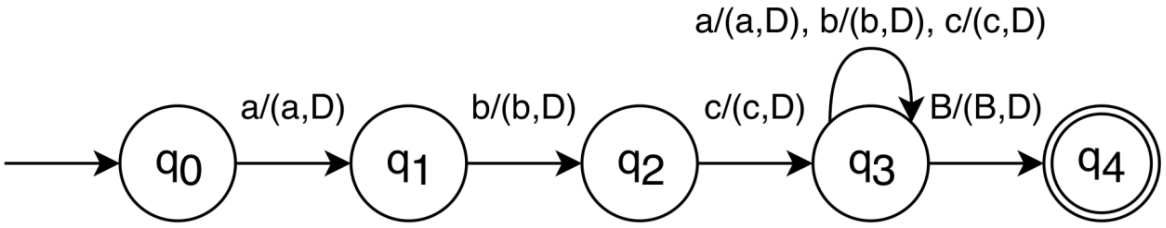


**Figura 2.** Máquina de Turing da expressão regular abc(a | b | c)*

Tendo em vista a Máquina de Turing e suas representações, todas as questões do trabalho pediram a formulação de Máquinas de Turing. Na questão 1 foi pedida a formação de uma função que utilize x e retorne x + 1, sendo x o reverso de um número binário. Na questão 2 é solicitado a implementação de uma Máquina que recebe $1^i$ e retorna 1 caso i seja menor que 3, e $1^{i-3}$ caso contrário. Na questão 3 é requisitada a produção de uma função utilizando uma Máquina de Turing que calcule max(m - n, 0), retornando 0 caso n seja igual ou maior que m, e $1^{m-n}$ em caso contrário. E por fim, na quarta questão, a atividade pede a formulação de outras 4 Máquinas de Turing, cada uma cumprindo seu propósito em decidir as linguagens pedidas por cada seção da questão.

## 2. Materiais e métodos

A linguagem de programação Python foi utilizada para o desenvolvimento dos códigos que agem como as Máquinas pedidas, em específico a sua versão 3.11.2, sendo utilizada em diversas áreas da computação. Sendo escolhida por sua facilidade e praticidade para a solução de problemas reais e ser de fácil compreensão, além de ser uma linguagem comum a todos os integrantes do grupo.

Somado a isso, foi utilizado o Visual Studio Code como ambiente de desenvolvimento integrado para produzir e testar o código durante a composição do código. Ademais, o subsistema do Windows para Linux (WSL), versão Ubuntu 22.04.2 LTS foi essencial para a execução dos programas, assim como o GitHub, plataforma de hospedagem de código, que foi utilizado para a hospedagem do código, relatório e imagens utilizadas neste. Para o planejamento e desenvolvimento do texto foi utilizado o google documentos devido sua facilidade e praticidade para compartilhar textos produzidos e a possibilidade de manutenção em colaboração em tempo real.

O código produzido por este relatório será separado em 5 arquivos, sendo um para cada questão e um por último de testagens, que funciona para comprovar a funcionalidade das máquinas produzidas pelos integrantes do grupo.

## 3. Primeira questão

A questão pede a formulação de uma Máquina de Turing que compute a função f(x) = x + 1, x sendo o reverso da representação de um número binário. Para conseguir desenvolver um código para a resolução da questão é necessário que seja feita a representação gráfica do autômato, para assim poder compreender os estados que devem ser programados na função. Tendo isso em vista, a representação da Máquina de Turing pedida pode ser observada na Figura 3, representando a máquina pedida pela questão, recebendo uma um número binário reverso, e retornando esse número somado a 1.

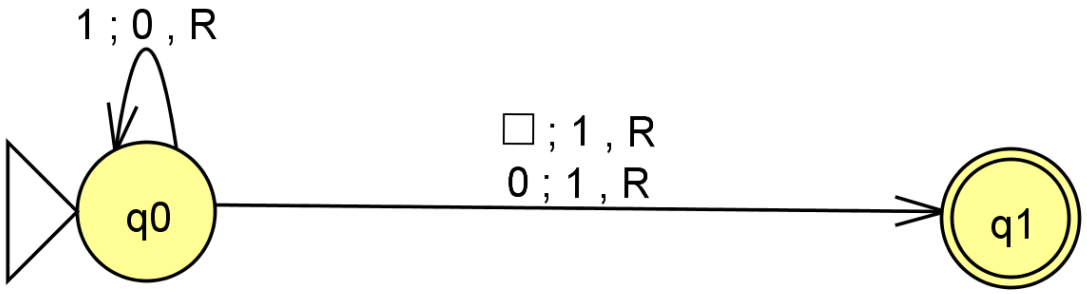


**Figura 3.** Máquina de Turing para Somar 1 a um número binário reverso

Então, foi desenvolvido a nomeação da função, definindo o nome e os parâmetros da função, sendo "x" o valor fornecido para a máquina, e o "estado_atual" o estado atual da máquina, sendo inicialmente q0, e por fim foi definido um valor "i", com valor padrão igual a 0, agindo como um cursor para a fita, a atribuição da função pode ser observado na Figura 4.

```
def maquina_questao_1(x, estado_atual='q0', i=0):
```

**Figura 4.** Atribuição da função da Máquina de Turing da primeira questão.

Após a definição da função, é possível gerar a Máquina de Turing tendo em vista que ela funciona basicamente com a utilização de vários if, com outros ifs dentro de si. Sendo uma verificação para cada estado da máquina, e os ifs dentro dos ifs de estado de máquina sendo ifs que verificam cada transição, lendo se o valor é o desejado, e se for, transforma o valor da fita para o novo valor, fazendo a transição do cursor para a esquerda ou direita dependendo do caso. Na Figura 5 se observa a definição de uma transição do estado q0 no caso da leitura de '1' na fita.

```
if estado_atual == 'q0':
    if x[i] == 1:
        x[i] = 0
        i += 1
        novo_estado = 'q0'
        return maquina_questao_1(x, novo_estado, i)
```

**Figura 5.** Trecho de código

Com isso, basta repetir os ifs para cada uma das transições da Máquina. Também é importante destacar que para o caso do cursor chegar no fim da lista, foi feita uma verificação para adicionar um novo elemento na lista e continuar como uma transição normal. Por fim de todas as verificações, foi retornado o valor da fita, obtendo a função finalizada pedida pela questão, como observado na Figura 6.

```
def maquina_questao_1(x, estado_atual='q0', i=0):
    if estado_atual == 'q0':
        if len(x) == i:
            x.append(1)
            i += 1
            novo_estado = 'q1'
            return maquina_questao_1(x, novo_estado, i)
        if x[i] == 1:
            x[i] = 0
            i += 1
            novo_estado = 'q0'
            return maquina_questao_1(x, novo_estado, i)
        if x[i] == 0:
            x[i] = 1
            i += 1
            novo_estado = 'q1'
            return maquina_questao_1(x, novo_estado, i)
    return x
```

**Figura 6.** Máquina de Turing da primeira questão.

Também é importante destacar que a função somente aceita uma lista de inteiros, inteiros que somente podem ser um dos valores, 0 ou 1, como ocorre, por exemplo, no caso da lista [0, 0, 1, 1], que retorna [1, 0, 1, 1].

## 4. Segunda questão

A segunda questão pede que seja formulada uma Máquina de Turing que trabalha com uma fita $1^i$, em que i é um número natural, retornando uma fita com somente 1 no caso de i ser menor que 3, e retornar uma fita com $1^i$ elementos no caso de i maior ou igual a 3. Para desenvolver o código pedido foi criado uma representação gráfica da máquina para a compreensão das verificações a serem geradas para o programa desenvolvido em Python.

A representação gráfica da máquina de turing pedida pode ser observada na Figura 7.

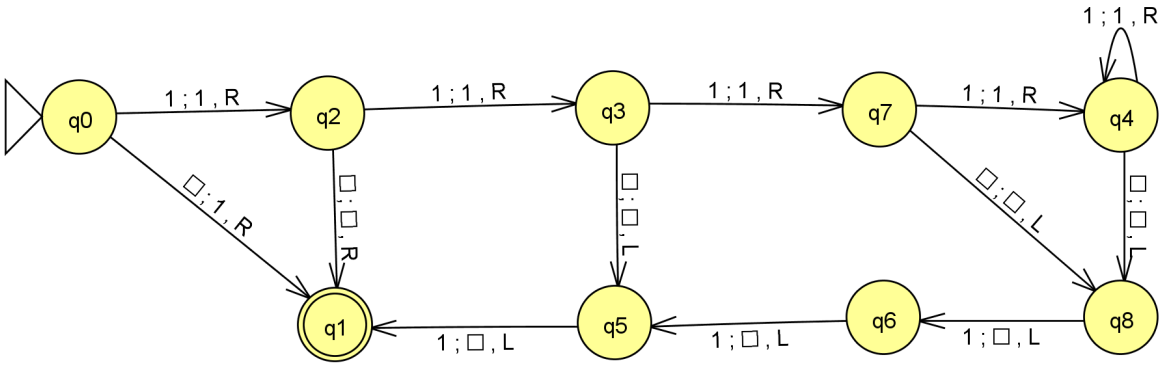


**Figura 7.** Representação gráfica da máquina de turing da segunda questão

Pode-se observar, com a representação gráfica da Figura 7, todas as transições da máquina, assim basta fazer verificações para cada um dos estados, verificando o elemento atual do cursor, considerado como a variável "i" no programa, e caso o elemento atual seja igual a transição a função deve fazer a recursão da função, alterando o valor da fita até o fim do código. Para verificar o caractere vazio foi calculado se o tamanho da lista é igual a posição do cursor "i". E no caso de ser necessário remover um item da lista foi utilizado a função pop do Python. A recursão tem fim no estado q1, em que retorna a lista chamada "fita".

Uma seção do código que pode ser observado para entender o processo de criação está na Figura 8, apresentando uma verificação para o estado q0, agindo de forma parecida para os outros estados, alterando o estado final, a ordem do cursor e a possibilidade de ser removido e adicionado um elemento na fita.

```
if estado_atual == 'q0':
    if len(fita) == i:
        fita.append(1)
        return maquina_questao_2(fita, 'q1', i + 1)
    if fita[i] == 1:
        return maquina_questao_2(fita, 'q2', i + 1)
```

**Figura 8.** Verificação para o estado q0 da Máquina de Turing da segunda questão.

Dessa forma, o programa que executa como uma Máquina de Turing já está completamente funcional, podendo ser observado sua funcionalidade na seção de testes experimentais deste relatório. Devido ao tamanho do código, não haverá uma figura apresentando a sua forma completa, mas é possível observar ele em sua forma completa no Github, estando presente no arquivo com nome "questão2.py", e seus testes pertencendo ao arquivo "testes_experimentais.py".

## 5. Terceira Questão

Nessa questão, foi requisitada uma Máquina de Turing que calculasse uma função max(m - n, 0), sendo representado por $1^m01^n$ na fita de entrada, retornando $1^{m-n}$ no caso de "m" ser maior que "n", e vazia no caso contrário. Para desenvolver o programa que faça a função desejada, é necessário ter uma representação gráfica para melhor compreensão das transições e do que foi pedido.

De tal forma, se obtém a Figura 9, sendo a representação gráfica da Máquina de Turing da terceira questão.

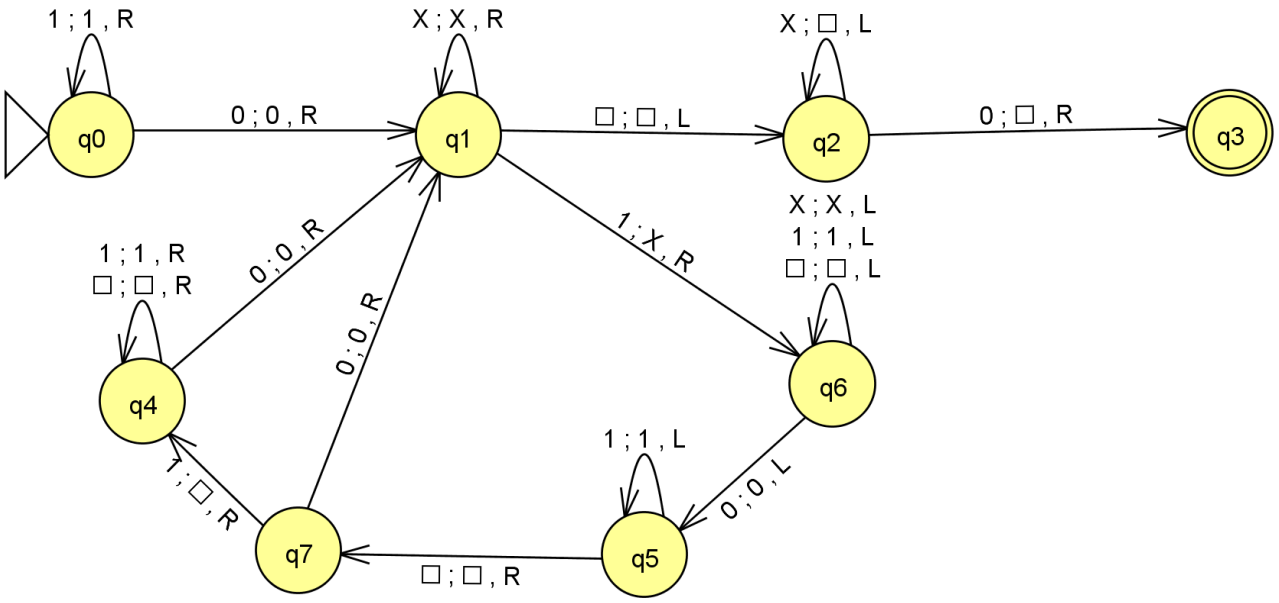


**Figura 9.** Representação gráfica da Máquina de Turing da terceira questão

De forma igual as questões anteriores, será produzido o código de uma forma parecida, entretanto irá ocorrer uma mudança no final, devido o pop no python remover o item e puxar todos os itens da lista para onde ele estava, assim não será utilizado para remover o 1 no início da fita, mas será utilizado o None no lugar do antigo número, que será removido no momento de retorno da função, utilizando a função desenvolvida e apresentada na Figura 10.

```
def remover_none_em_lista(lista):
    lista = list(filter(lambda x: x != None, lista))
    return lista
```

**Figura 10.** Função para remover todos os None presente em uma lista

Também será verificado no estado q5 se o "i" é menor que 0, e caso for deve transitar para o estado q7 voltando "i" para o valor de 0, impossibilitando o caso da lista acabar lendo o valor do último elemento sem perceber. Por causa da linguagem utilizada considerar o index

-1 como o último index de uma lista. Após fazer o código e suas verificações, assim como suas alterações para que não ocorram erros inesperados, obtemos a função que age como Máquina de Turing como pedida pela questão.

## 6. Quarta questão

A quarta questão pediu para a formulação de 4 diferentes Máquinas de Turing para 4 diferentes linguagens, com suas devidas características. Dessa forma, serão utilizados os conhecimentos obtidos para o desenvolvimento das funções anteriores para a resolução das seguintes funções.

**a) $L_1 = \{w\#w^R \mid w \{0, 1\}^*\}$**

Com a linguagem fornecida, é possível gerar uma representação gráfica para a Máquina, obtendo a Figura 11.

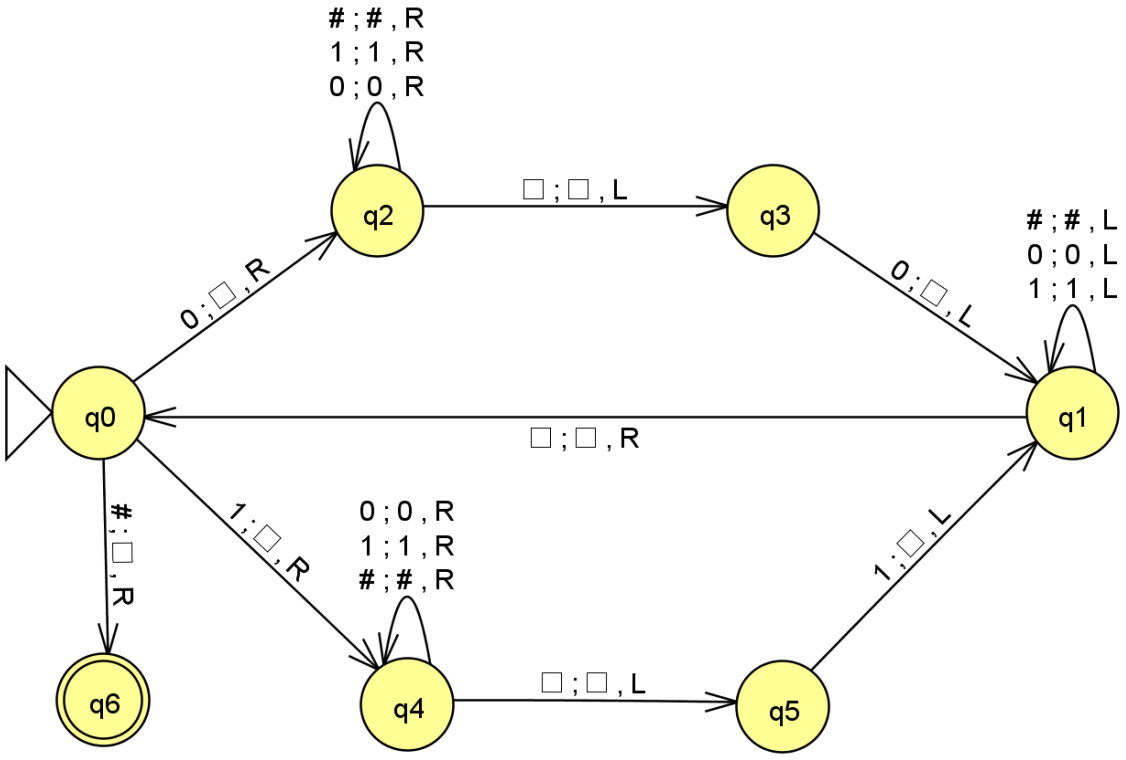


**Figura 11.** Representação gráfica de Máquina de Turing para $L_1 = \{w\#w^R \mid w \{0, 1\}^*\}$

Com a representação gráfica já é possível desenvolver o código, utilizando-se de ifs para verificar o estado e o item atual da fita, para depois alterar para um novo valor até o fim da máquina, o código completo pode ser observado no arquivo "questao4.py", com a função de nome "maquina_questao4_a" que não foi apresentada em uma figura devido seu grande tamanho.

**b) $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0$ e $i = j$ ou $j = k\}$**

Primeiramente, para o desenvolvimento da função para a linguagem é necessário ter uma representação gráfica para a melhor compreensão do que foi pedido, representação gráfica que pode ser observada pela Figura 12.

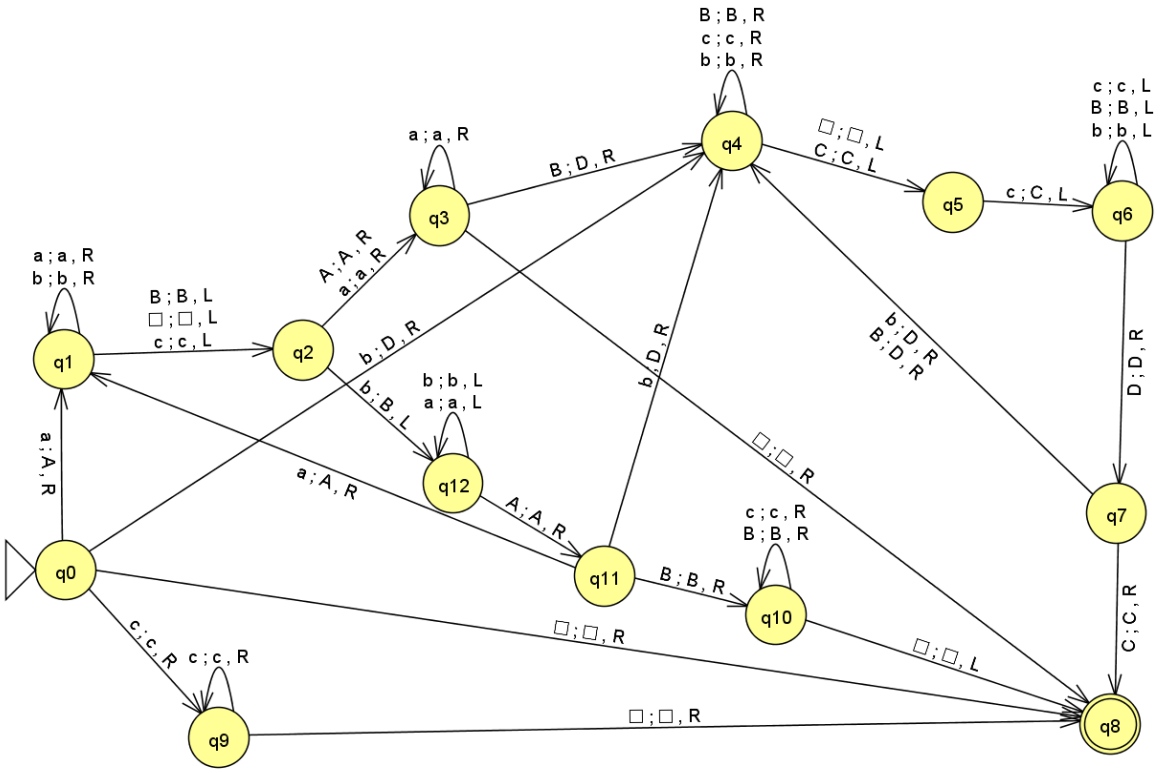


**Figura 12.**

Representação gráfica de Máquina de Turing para $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i = j \text{ ou } j = k\}$

Utilizando-se das mesmas ideias propostas anteriormente, foi possível gerar o código que funcionasse como máquina de turing para tal linguagem. O código desta seção também esta presente no arquivo “questao4.py” e não foi apresentado devido sua densidade de texto.

**c) $L_3 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } k = i + j\}$**

Em primeira análise, é fundamental uma representação em forma gráfica para a produção do código, tal representação que foi desenvolvida e está presente na Figura 13.

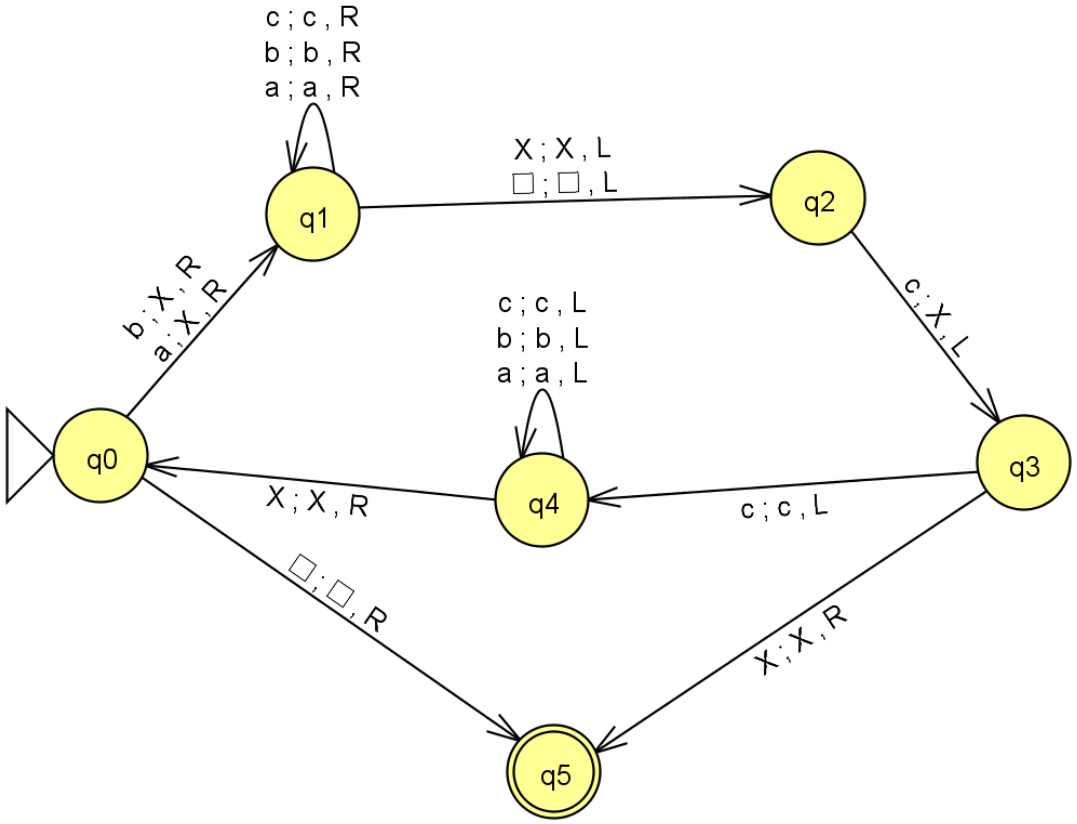


**Figura 13.**

Representação gráfica de Máquina de Turing para $L_3 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } k = i + j\}$

Dessa forma, foi utilizado da representação gráfica para a produção do código, utilizando de suas transições para a formação de verificações, e considerando None como o espaço vazio. O código apesar de simples não foi colocado devido consumir grande espaço, sendo possível observar no Github, no arquivo em que retém as resoluções da quarta questão.

**d) $L_4 = \{a^n b^n c^n \mid n \geq 0\}$**

Com isso, para desenvolver o código foi gerada a Máquina de Turing da Figura 14, para facilitar a compreensão e desenvolvimento da função que segue devidamente a linguagem pedida.

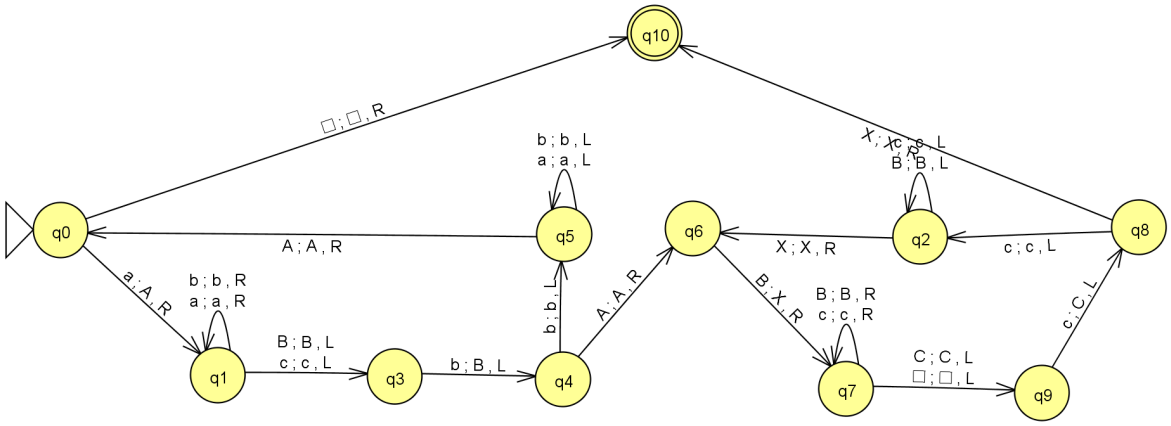


**Figura 14.** Representação gráfica de Máquina de Turing para $L_4 = \{a^n b^n c^n \mid n \geq 0\}$

Por fim, foi gerado o código com a utilização de verificações e recursos, dependendo do estado atual e do item atual da fita. Alcançando o código que pode ser observado no código fonte.

## 7. Testes experimentais

Para a verificar se as funções desenvolvidas atingiram seu objetivo, foram feitos testes experimentais utilizando o "assert" no python, que ao retornar um valor False gera uma mensagem de erro. Observe como exemplo a Figura 14, em que ocorre o teste do autômato utilizando do assert e apresenta uma mensagem no final caso ocorra o sucesso.

```
def testes_questao_1():
    assert maquina_questao_1([0,0,1,1]) == [1,0,1,1]
    assert maquina_questao_1([1,1,1,1,0,1]) == [0,0,0,0,1,1]
    assert maquina_questao_1([1,1,1,1,1,1,1]) == [0,0,0,0,0,0,0,1]
    assert maquina_questao_1([0]) == [1]
    assert maquina_questao_1([1]) == [0,1]
    print("Testes da primeira questao passaram com sucesso!")
```

**Figura 14.** Coleção de testes para a Máquina de Turing da primeira questão

Com isso, basta executar a função que a função da primeira questão será testada e apresentará a mensagem de sucesso no caso de sucesso. Foi desenvolvido uma coleção de testes para cada uma das funções, e por fim todas foram executadas, e passadas, resultando no sucesso das funções desenvolvidas neste relatório, como é possível observar na Figura 15.

```
PS C:\Trabalho Máquinas de Turing> python testes_experimentais.py
Testes da primeira questao passaram com sucesso!
Testes da segunda questao passaram com sucesso!
Testes da terceira questao passaram com sucesso!
Testes da quarta questao passaram com sucesso!
```

**Figura 15.** Retorno do terminal apresentando sucesso

## 7. Comentários finais

Enfim, se percebe com tal trabalho foi possível aprender como é possível criar e utilizar Máquinas de Turing tendo como base linguagens atuais, além da solidificação do conhecimento sobre o assunto estudado em sala de aula. Somado a isso também é importante destacar a extrema importância da produção do relatório para a compreensão de forma geral da matéria de Linguagens Formais, entendendo a produção de expressões regulares, autômatos finitos, transdutores e máquinas de turing. Com a geração de figuras, programas e testes para alcançar um bom relatório e o resultado esperado das máquinas pedidas.

## Apêndice A

Código fonte gerado com o desenvolvimento deste relatório:

https://github.com/Braz-Souza/Trabalho-Maquina-Turing