



UNIVERSIDADE FEDERAL DO MARANHÃO

Máquina de Turing e Autômato de Pilha

DISCIPLINA: LINGUAGENS FORMAIS E AUTÔMATOS

Discentes: Delryson da Silva Saraiva
Katarina Ires de Castro Pinheiro

Docente: Dr. Thales Levi Azevedo Valente

Índice

03 Introdução

05 Fundamentação Teórica

12 Desenvolvimento

21 Análise de Resultados

22 Conclusão

23 Programa em Funcionamento

Introdução

OBJETIVOS DO TRABALHO

- Objetivo Geral:
 - Implementar e simular o funcionamento de uma Máquina de Turing e comparar com um Autômato com Pilha, demonstrando sua execução
- Objetivos Específicos:
 - Validar a capacidade desses autômatos de processar cadeias de entrada, de acordo com regras de transição predefinidas;
 - Analisar a evolução dos estados e alterações na fita da Máquina de Turing e na pilha do Autômato com Pilha durante a computação;
 - Gerar representações visuais da execução da Máquina de Turing, auxiliando na compreensão do processamento.

Introdução

O QUE É UMA MÁQUINA DE TURING E UM AUTÔMATO COM PILHA?

- O trabalho apresenta a implementação e simulação de dois modelos computacionais fundamentais:
 - Máquina de Turing: Modelo teórico capaz de simular qualquer algoritmo, manipulando símbolos em uma fita infinita.
 - Autômato com Pilha: Modelo que reconhece linguagens livres de contexto, utilizando uma pilha como memória auxiliar.

Característica	Autômato com Pilha (PDA)	Máquina de Turing (TM)
Memória	Pilha (LIFO)	Fita (acesso aleatório)
Poder Computacional	Reconhece linguagens livres de contexto	Reconhece linguagens recursivamente enumeráveis
Direção de leitura	Somente avança	Pode mover para esquerda e direita
Aplicações	Análise sintática, validação de cadeias平衡adas	Simulação de algoritmos, estudo da computabilidade

Figura 1 - Comparação Máquina de Turing e Autômato com Pilha

Fundamentação Teórica

MÁQUINAS DE TURING

- Pode-se incluir a definição formal de uma Máquina de Turing, que é dada por uma quintupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, onde:

$$M = (Q, \Sigma, \Gamma, \Delta, Q_0, F)$$

- Q : Conjunto finito de estados.
- Σ : Conjunto finito de símbolos de entrada.
- Γ : Conjunto finito de símbolos da fita ($\Sigma \subseteq \Gamma$).
- δ : Função de transição, que define o comportamento da máquina, dada pelo formato:

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$, onde:

- L: significa mover o cabeçote para a esquerda.
- R: significa mover o cabeçote para a direita.
- N: significa não mover o cabeçote.
- q_0 : Estado inicial.
- F : Conjunto de estados de aceitação.

Fundamentação Teórica

ELEMENTOS DE UMA MÁQUINA DE TURING

- As Máquinas de Turing são compostas por elementos-chave que interagem para realizar cálculos.

ALFABETO

ESTADO INICIAL

ESTADO FINAL

- Um conjunto finito de símbolos que podem ser escritos na fita.
- O estado em que a máquina inicia a computação.
- O estado que indica que a computação foi concluída com sucesso.

Fundamentação Teórica

FUNCIONAMENTO DE UMA MÁQUINA DE TURING

- A máquina lê o símbolo atual na fita, consulta sua tabela de transições e realiza uma ação.



LEITURA

1

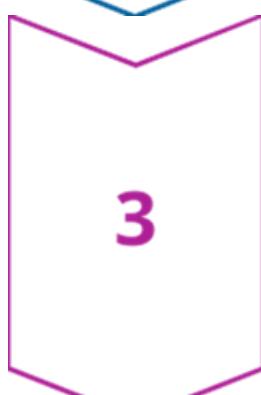
A máquina lê o símbolo atual na fita.



CONSULTA

2

A máquina consulta a tabela de transições para determinar a ação a ser tomada.



AÇÃO

3

A máquina escreve um novo símbolo na fita, move a cabeça para a esquerda ou direita e muda para um novo estado.

Fundamentação Teórica

MÁQUINAS DE TURING

Definição

- Uma Máquina de Turing é um modelo matemático de computação que consiste em uma fita infinita, uma cabeça de leitura/escrita e um conjunto de estados.

FITA INFINTA

A fita contém símbolos que representam os dados de entrada e saída da máquina.

CABEÇA DE LEITURA/ESCRITA

A cabeça lê e escreve símbolos na fita, movendo-se para a esquerda ou direita

ESTADOS

O estado atual da máquina determina a ação a ser tomada.

Fundamentação Teórica

AUTÔMATO COM PILHA

- Para o Autômato com Pilha, você pode usar a definição formal que envolve os estados, o alfabeto de entrada, a pilha e as transições. Ela é dada por uma quintupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, onde:

$$A = (Q, \Sigma, \Gamma, \Delta, Q_0, F)$$

- Q : Conjunto de estados;
- Σ : Alfabeto de entrada;
- Γ : Alfabeto da pilha;
- δ : Função de transição, que pode ser representada como:

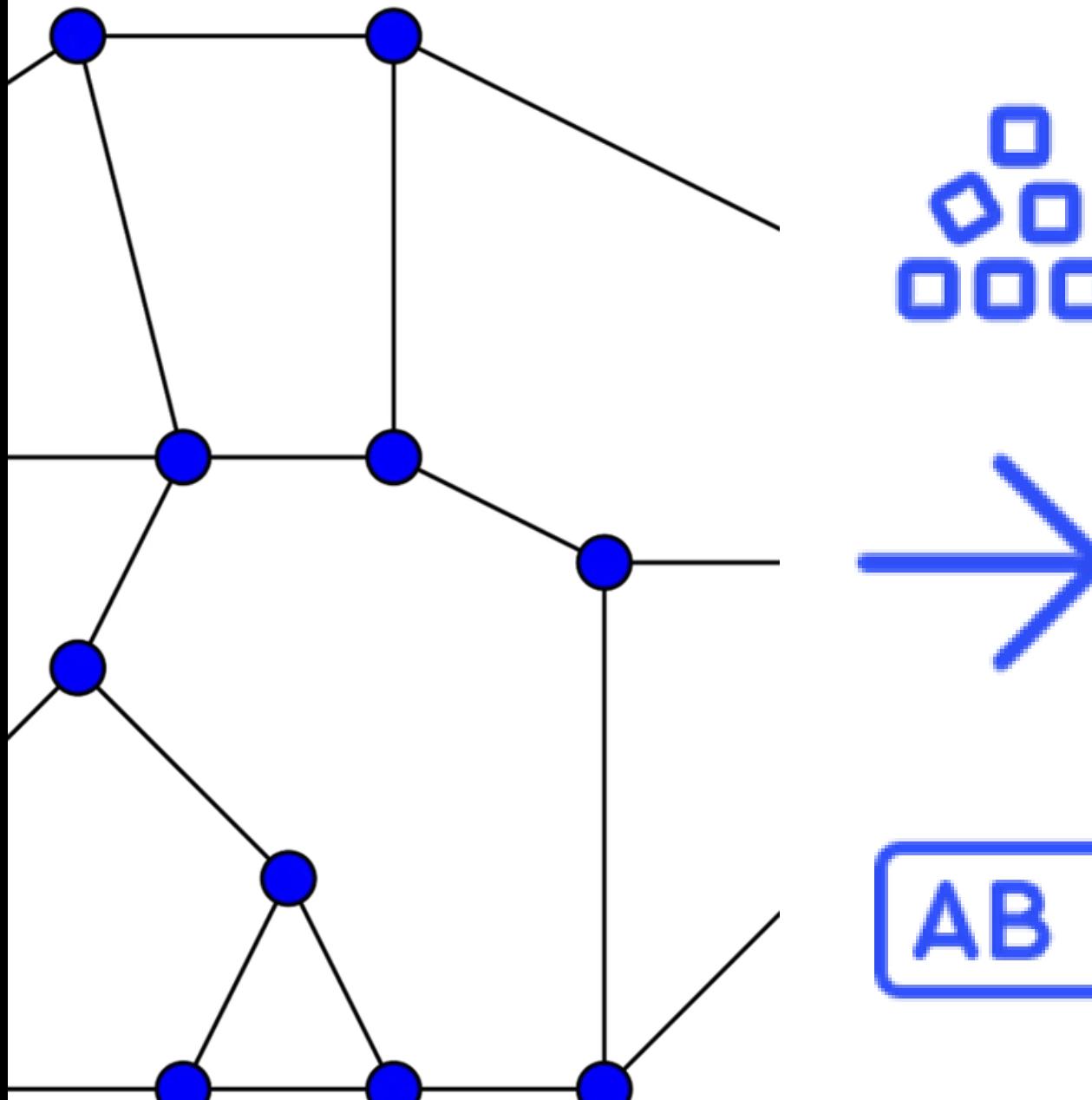
$\delta: Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$, ou seja, o autômato faz transições com base no estado atual, o símbolo de entrada e o topo da pilha, podendo empurrar ou retirar símbolos da pilha;

- q_0 : Estado inicial;
- F : Conjunto de estados de aceitação.

Fundamentação Teórica

AUTÔMATO COM PILHA

- Os Autômatos de Pilha são modelos de computação abstratos que se assemelham às Máquinas de Turing, mas com uma pilha em vez de uma fita.



PILHA

- Uma estrutura de dados que armazena os símbolos em uma ordem específica, com acesso apenas ao elemento superior.

ESTADO ATUAL

- Representa a configuração atual do autômato.

ENTRADA

- Uma sequência de símbolos que o autômato recebe.

Fundamentação Teórica

- As Máquinas de Turing e os Autômatos de Pilha têm aplicações importantes em diversas áreas da computação.

LINGUAGENS FORMAIS

As Máquinas de Turing podem ser usadas para reconhecer e gerar linguagens formais, como linguagens regulares e livres de contexto



COMPILEDORES

Os Autômatos de Pilha são usados em compiladores para analisar a estrutura do código e gerar código de máquina.



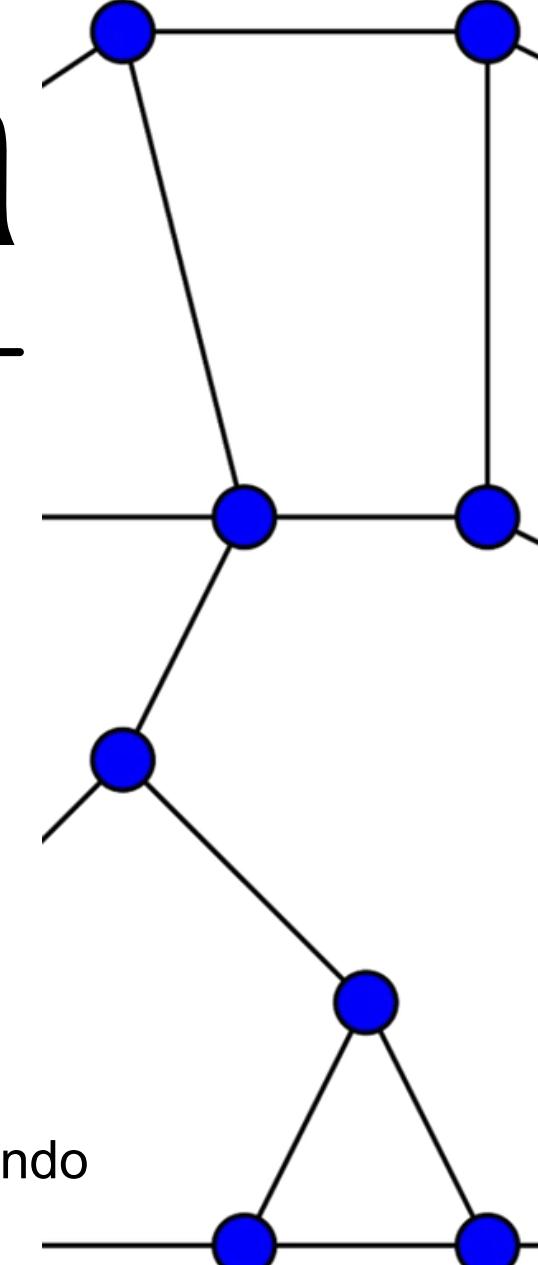
TEORIA DA COMPUTABILIDADE

As Máquinas de Turing fornecem um modelo teórico para a computação, definindo os limites da capacidade computacional



ANÁLISE SINTÁTICA

Os Autômatos de Pilha são usados na análise sintática de linguagens de programação, verificando se o código fonte está gramaticalmente correto.



VERIFICAÇÃO DE MODELOS

As Máquinas de Turing são usadas para verificar a corretude de sistemas complexos, como sistemas de software.

DESENVOLVIMENTO

MÁQUINA DE TURING

ESTRUTURA DA CLASSE TURINGMACHINE

```
class TuringMachine:  
    def __init__(self, tape, initial_state, accepting_states, transitions):  
        self.tape = list(tape) + [' '] * 100 # Adiciona espaço extra na fita  
        self.head_position = 0 # Inicializa a posição da cabeça de leitura  
        self.current_state = initial_state # Define o estado inicial  
        self.accepting_states = accepting_states # Conjunto de estados de aceitação  
        self.transitions = transitions # Dicionário com as regras de transição
```

- A fita é inicializada como uma lista de caracteres e é extendida com espaços para evitar problemas de borda;
- A cabeça de leitura começa na posição 0;
- A máquina começa em um estado inicial (`initial_state`);
- `accepting_states` define quando a máquina para e aceita a entrada;
- `transitions` contém as regras de transição do autômato.

DESENVOLVIMENTO

MÁQUINA DE TURING

MÉTODO STEP()

```
def step(self):
    if self.current_state in self.accepting_states:
        return False # Se estiver em um estado de aceitação, para a execução

    current_symbol = self.tape[self.head_position] # Lê o símbolo atual
    action = self.transitions.get((self.current_state, current_symbol)) # Obtém a regra

    if action is None:
        return False # Se não houver transição, a máquina para

    new_state, new_symbol, direction = action # Desempacota a transição
    self.tape[self.head_position] = new_symbol # Substitui o símbolo na fita
    self.current_state = new_state # Atualiza o estado

    if direction == 'R':
        self.head_position += 1 # Move para a direita
    elif direction == 'L':
        self.head_position -= 1 # Move para a esquerda

    return True # Indica que o passo foi concluído
```

- Se a máquina já estiver em um estado de aceitação, para;
- Obtém o símbolo atual e verifica se há uma transição válida;
- Se houver, altera o símbolo na fita, muda o estado e move a cabeça de leitura;
- Se não houver uma transição definida, a máquina para.

DESENVOLVIMENTO

MÁQUINA DE TURING

MÉTODO RUN()

```
def run(self):
    while self.step(): # Executa enquanto houver transições válidas
        pass
    return ''.join(self.tape), self.current_state # Retorna a fita final e o estado final
```

- Chama step() repetidamente até que não haja mais transições válidas.
- Retorna a fita resultante e o estado final.

DESENVOLVIMENTO

MÁQUINA DE TURING

EXEMPLO DE USO

```
tape = "0110"

initial_state = "q0"

accepting_states = {"q_accept"}

transitions = {

    ("q0", "0"): ("q1", "1", "R"),
    ("q1", "1"): ("q_accept", "1", "R"),
}

}
```

- A fita inicial é "0110";
- O estado inicial é "q0", e o estado de aceitação é "q_accept";
- As regras de transição determinam como a máquina processa a entrada.

DESENVOLVIMENTO

AUTÔMATO COM PILHA

ESTRUTURA DA CLASSE

```
class StackAutomaton:

    def __init__(self, initial_state, accepting_states, transitions):
        self.stack = [] # Inicializa a pilha vazia
        self.current_state = initial_state # Define o estado inicial
        self.accepting_states = accepting_states # Define os estados de aceitação
        self.transitions = transitions # Dicionário de transições
```

- O autômato começa com uma pilha vazia;
- O estado inicial e os estados de aceitação são definidos;
- “transitions” contém as regras de funcionamento.

DESENVOLVIMENTO

AUTÔMATO COM PILHA

MÉTODO STEP()

```
def step(self, input_symbol):
    top_symbol = self.stack[-1] if self.stack else None # Verifica o topo da pilha
    action = self.transitions.get((self.current_state, input_symbol, top_symbol))

    if action is None:
        return False # Se não houver transição válida, para

    new_state, stack_action, stack_symbol = action
    self.current_state = new_state # Atualiza o estado

    if stack_action == "PUSH":
        self.stack.append(stack_symbol) # Empilha um símbolo
    elif stack_action == "POP":
        self.stack.pop() # Desempilha

    return True
```

- O autômato lê um símbolo de entrada e verifica o topo da pilha;
- Busca uma transição correspondente e a executa;
- Se a ação for "PUSH", empilha um símbolo; se for "POP", remove o topo da pilha.

DESENVOLVIMENTO

AUTÔMATO COM PILHA

MÉTODO RUN()

```
def run(self, input_string):
    for symbol in input_string:
        if not self.step(symbol): # Para se não houver transição válida
            return False
    return self.current_state in self.accepting_states and not self.stack
```

- Processa cada símbolo da entrada e aplica as transições;
- A entrada é aceita se o estado final for um estado de aceitação e a pilha estiver vazia.

DESENVOLVIMENTO

AUTÔMATO COM PILHA

EXEMPLO DE USO

```
initial_state = "q0"
accepting_states = {"q_accept"}
transitions = {
    ("q0", "a", None): ("q1", "PUSH", "A"),
    ("q1", "b", "A"): ("q_accept", "POP", None),
}
```

- O autômato empilha "A" quando lê "a";
- Desempilha "A" quando lê "b";
- Se a pilha estiver vazia ao final e o estado for "q_accept", a entrada é aceita.

DESENVOLVIMENTO

VISUALIZAÇÃO DA MÁQUINA DE TURING

```
from PIL import Image, ImageDraw, ImageFont  
from IPython.display import display
```

- Usa a biblioteca PIL para desenhar a fita e a posição da cabeça de leitura;
- Armazena imagens de cada passo e gera um GIF animado.

ANÁLISE DE RESULTADOS

XaaYbb

State: q2

Conclusão

CONSIDERAÇÕES FINAIS

- A Máquina de Turing demonstrou sua capacidade de simular qualquer algoritmo computável;
- O Autômato a Pilha permitiu a análise de linguagens livres de contexto, destacando sua utilidade na modelagem de processos como análise sintática em linguagens de programação;
- A implementação prática dessas estruturas em Python possibilitou uma melhor compreensão de seus mecanismos operacionais, facilitando a visualização e validação de suas execuções;
- Concluímos que esses modelos são fundamentais para o estudo de linguagens formais e autômatos, servindo como base para diversas aplicações em computação teórica e prática.

PROGRAMA EM FUNCIONAMENTO

OBRIGADO