UNIVERSIDADE FEDERAL DE LAVRAS

GCC108 – TEORIA DA COMPUTAÇÃO

Prof. Mayron Moreira

Diogo Oliveira Carvalho 202120533

Simulador de Máquina de Turing Universal
Linguagem C++

Lavras

2024

SUMÁRIO

1. INTRODUÇÃO

O objetivo deste trabalho é simular uma Máquina de Turing Universal (MTU) através de uma linguagem de programação.

A MTU deve receber como entrada uma Máquina de Turing Específica (MTS) e simular sua execução, executando sua função de transição, atualizando a fita, a posição da cabeça de leitura/escrita e o estado atual até que a MTS pare (alcance um estado final) ou entre em um loop infinito.

A entrada da MTS consiste em: MTS = (Q, ∑, Γ, δ, q0, qaceita, qrejeita, _), onde Q é o conjunto de estados, ∑ o alfabeto de entrada, Γ o alfabeto da fita, δ a função de transição, q0 o estado inicial pertencente à Q, qaceita e qrejeita os estados finais de aceitação e rejeição, respectivamente, pertencentes à Q e _ o símbolo branco pertencente à Γ.

2. PRINCIPAIS ESTRUTURAS UTILIZADAS:

2.1) enum EstadoFinal {
NAO_FINAL,
ACEITACAO,
REJEICAO
};
Indica se um estado é não final ou final de aceitação ou rejeição;

2.2) struct estado {
string nome;
bool estadoInicial;
EstadoFinal estadoFinal;
};
Representa um estado de Q, com um nome, a informação sobre ele ser ou não um estado inicial e uma informação do tipo enum EstadoFinal (não final, aceitação ou rejeição);

2.3) struct transicao {
estado estadoOrigem;
string simboloLido;
estado estadoDestino;
string novoSimbolo;
char direcao;
};
Representa uma transição da função de transição (δ), com o estado atual da transição, símbolo lido pela cabeça de leitura/escrita, estado

de destino da transição, símbolo a ser escrito na fita e a direção para a qual a cabeça de leitura/escrita será movida na fita (E ou D);

2.4) pair<estado *, int> estados: o primeiro elemento do par é um vetor do tipo da struct estado contendo utilizado para armazenar o conjunto de estados (Q), enquanto o segundo elemento é utilizado para armazenar o seu tamanho (|Q|);

2.5) string *alfabeto: vetor de string utilizado para armazenar o alfabeto de entrada (∑);

2.6) string *alfabetoFita: vetor de string utilizado para armazenar o alfabeto da fita (Γ);

2.7) pair<vector<transicao> *, int> fTransicao: o primeiro elemento do par é um vetor do tipo da struct transicao utilizado para armazenar a função de transição (δ), enquanto que o segundo elemento armazena o seu tamanho (|δ|);

2.8) pair<vector<string> *, int> fita: o primeiro elemento do par é um vetor do tipo string utilizado para armazenar a fita de entrada e o seu conteúdo durante toda a execução da MTS, enquanto que o segundo elemento armazena o tamanho da fita (considerando a palavra escrita e o primeiro símbolo branco após ela);

2.9) string branco: variável utilizada para armazenar o símbolo branco (_);

3. PRINCIPAIS FUNÇÕES IMPLEMENTADAS

3.1) lerMT

a) **Função:** void lerMT(pair<estado *, int> *estados, string **alfabeto, string **alfabetoFita, pair<vector<transicao> *, int> *funcaoTransicao, string &branco)

b) **Parâmetros:**

estados: par que contém o vetor que armazena o conjunto de estados (Q) da MT e o seu tamanho;

alfabeto: vetor que armazena o alfabeto de entrada (∑) da MT;

alfabetoFita: vetor que armazena o alfabeto da fita (Γ) da MT;

funcaoTransicao: par que contém o vetor que armazena a função de transição (δ) da MT e o seu tamanho;

branco: variável que armazena o símbolo branco (_) do alfabeto da fita;

c) **Descrição:** Lê a descrição de uma MTS (Q, ∑, Γ, δ, q0, qaceita, qrejeita, _) e a armazena nas estruturas recebidas por parâmetro.

d) **Pontos importantes** de uma MT verificados na função lerMT():
   i) ter, ao menos, dois estados (aceitacao e rejeicao);
   ii) ter um estado inicial;
   iii) ter, ao menos, um símbolo no alfabeto de entrada (∑);
   iv) Γ deve possuir, ao menos, todos os símbolos de ∑ e o símbolo branco;
   v) função de transição (δ) deve ser válida, isto é, estados e símbolos da transição devem pertencer a Q e Γ, respectivamente;

3.2) lerFita

a) **Função:** void lerFita(pair<vector<string> *, int> *fita, string branco)

b) **Parâmetros:**
fita: par com o vetor que contém a fita da MT e o tamanho da fita;
branco: variável que armazena o símbolo branco (_) do alfabeto da fita;

c) **Descrição:** Lê a fita de entrada, adiciona um símbolo branco ao final, para indicar o fim da fita, e a armazena.

3.3) simularMT

a) **Função:** int simularMT(pair<estado *, int> *estados, pair<vector<transicao> *, int> *fTransicao, pair<vector<string> *, int> *fita, string branco)

b) **Parâmetros:**
estados: par que contém o vetor que armazena o conjunto de estados (Q) da MT e o seu tamanho;
funcaoTransicao: par que contém o vetor que armazena a função de transição (δ) da MT e o seu tamanho;
branco: variável que armazena o símbolo branco (_) do alfabeto da fita;

c) **Descrição:** Simula a execução da MTS lida anteriormente. Ao final de sua execução, retorna 1 caso a palavra de entrada pertença à linguagem da MTS, 0 caso não pertença ou -1 caso haja um possível loop (muitas execuções sem um estado final encontrado). Caso haja

um possível loop, o usuário deve escolher entre continuar a execução, digitando a letra S, ou finalizá-la, digitando a letra N.

d) **Pontos importantes** da simulação da MTS assegurados pela função simularMT:
    i) execução iniciada pelo estado inicial e cabeça de leitura/escrita posicionada na posição 0 da fita;
    ii) atualização correta dos estados e da cabeça de leitura/escrita, não permitindo que seja acessada uma posição inválida;
    iii) apenas transições válidas (pertencentes à δ) são executadas;
    iv) execução finaliza imediatamente ao encontrar um estado final de aceitação ou de rejeição;

4. EXECUÇÃO DO PROGRAMA

Quando o programa é iniciado, a primeira função executada é a função lerMT(), para que todos os dados da descrição da MTS sejam lidos e armazenados nas devidas estruturas de forma correta.

Em seguida, a função lerFita() é chamada para que seja lida a fita de entrada que será utilizada durante a simulação da MTS lida.

Após as leituras da MTS e da fita de entrada, a função simularMT() é executada para que a simulação da MTS seja iniciada. Ela executa a função de transição, realizando as devidas alterações de estados e de posição da cabeça de leitura/escrita, até que seja encontrado um estado final ou que aconteça um possível loop na MT.

Em seguida, de acordo com o retorno da função simularMT(), é exibida uma mensagem ao usuário, informando se a palavra foi aceita, se a palavra foi rejeitada, ou se a MT possivelmente entrou em loop, não havendo resposta. Para os casos de parada em estado final, o conteúdo final da fita é exibido na tela.

Por fim, após a execução da MT com a fita de entrada lida, o programa pergunta ao usuário se ele deseja continuar e informar outra fita de entrada para ser executada pela MT. Caso a resposta seja sim (denotada pela letra ‘S’), o programa volta ao passo de leitura da fita de entrada.

O programa é finalizado quando o usuário informa que não deseja mais executar a MT com outras fitas de entrada.

5. CASOS DE TESTE

5.1) Máquina de Turing aceitadora sobre a linguagem A = {w#w | w ∈ {0,1}*}
Aceita palavras sobre o alfabeto {0,1}* com duas subpalavras iguais separadas por um símbolo #.

Descrição da MT (entrada do código):

```
10                                  -> Número de estados (|Q|)
q1 q2 q3 q4 q5 q6 q7 q8 qA qR       -> Conjunto de estados (Q)
q1                                  -> Estado inicial q0
qA qR                               -> Estados de aceitação e rejeição
3                                   -> Tamanho do alfabeto de entrada
0 1 #                               -> Alfabeto de entrada (∑)
5                                   -> Tamanho do alfabeto da fita
0 1 # x B                           -> Alfabeto da fita (Γ)
B                                   -> Símbolo branco
q1 1 q3 x D                         -> Função de transição (δ)
q1 # q8 # D
q1 0 q2 x D
q2 0 q2 0 D
q2 1 q2 1 D
q2 # q4 # D
q3 0 q3 0 D
q3 1 q3 1 D
q3 # q5 # D
q4 x q4 x D
q4 0 q6 x E
q5 x q5 x D
q5 1 q6 x E
q6 x q6 x E
q6 # q7 # E
q7 0 q7 0 E
q7 1 q7 1 E
q7 x q1 x D
q8 x q8 x D
q8 B qA B D
-1                                  -> Fim da descrição da MT
```

Fita de entrada 1:

```
9
0 1 0 1 # 0 1 0 1
```

Saída:

A palavra pertence à linguagem da MT (duas palavras iguais separadas por #)

Fita de entrada 2:

9

0 0 0 0 # 1 1 1 1

Saída:

A palavra não pertence à linguagem da MT (duas palavras diferentes antes e depois do #)

Fita de entrada 3:

7

a b c # a b c

Saída:

A palavra não pertence à linguagem da MT (duas palavras iguais, porém os símbolos não pertencem ao alfabeto de entrada {0,1}*)

5.2) Máquina de Turing que recebe um número binário e soma 1 a ele. Retorna o resultado da soma na fita de entrada (desconsiderando os símbolos branco).

Descrição da MT (entrada do código):

```
9                          -> Número de estados (|Q|)
q0 q1 q2 q3 q4 q5 q6 qA qR -> Conjunto de estados (Q)
q0                         -> Estado inicial q0
qA qR                      -> Estados de aceitação e rejeição
2                          -> Tamanho do alfabeto de entrada
0 1                        -> Alfabeto de entrada (∑)
3                          -> Tamanho do alfabeto da fita
0 1 B                      -> Alfabeto da fita (Γ)
B                          -> Símbolo branco
q0 0 q1 B D                -> Função de transição (δ)
q0 1 q2 B D
q1 0 q1 0 D
q1 1 q2 0 D
q1 B q3 0 D
q2 1 q2 1 D
q2 0 q1 1 D
q2 B q3 1 D
q3 B q4 B E
q4 1 q4 0 E
q4 0 qA 1 E
q4 B q5 B D
q5 0 q6 1 D
q6 0 q6 0 D
q6 B qA 0 E
-1                         -> Fim da descrição da MT
```

Fita de entrada 1:

2
1 1

Saída:

1 0 0

Explicação: 11 em binário = 3 em decimal; 3 + 1 = 4 = 100 em binário

Fita de entrada 2:

4
1 0 1 0

Saída:

1 0 1 1

Explicação: 1010 em binário = 8 em decimal; 8 + 1 = 9 = 1011 em binário

Fita de entrada 3:

8
1 0 0 0 0 0 1 0

Saída:

1 0 0 0 0 0 1 1

Explicação: 10000010 em binário = 130 + 1 = 131 = 10000011 em binário

5.3) Máquina de Turing aceitadora sobre o alfabeto {0}* tal que o número de 0's é uma potência de 2

Descrição da MT (entrada do código):

7	-> Número de estados (|Q|)
q1 q2 q3 q4 q5 qA qR	-> Conjunto de estados (Q)
q1	-> Estado inicial q0
qA qR	-> Estados de aceitação e rejeição
1	-> Tamanho do alfabeto de entrada
0	-> Alfabeto de entrada (∑)
3	-> Tamanho do alfabeto da fita
0 x B	-> Alfabeto da fita (Γ)
B	-> Símbolo branco
q1 0 q2 B D	-> Função de transição (δ)
q2 x q2 x D	
q2 B qA B D	
q2 0 q3 x D	
q3 x q3 x D	
q3 0 q4 0 D	
q3 B q5 B E	
q4 x q4 x D	
q4 0 q3 x D	
q5 0 q5 0 E	

q5 x q5 x E
q5 B q2 B D
-1 -> Fim da descrição da MT

Fita de entrada 1:

1
0

Saída:

A palavra de entrada pertence a linguagem da MT (quantidade de 0's = 1 e 1 é potência de 2)

Fita de entrada 2:

2
0 0

Saída:

A palavra de entrada pertence a linguagem da MT (quantidade de 0's = 2 e 2 é potência de 2)

Fita de entrada 3:

4
0 0 0 0

Saída:

A palavra de entrada pertence a linguagem da MT (quantidade de 0's = 4 e 4 é potência de 2)

Fita de entrada 4:

11
0 0 0 0 0 0 0 0 0 0 0

Saída:

A palavra de entrada não pertence a linguagem da MT (quantidade de 0's = 11 e 11 não é potência de 2)

6. CONCLUSÃO

Este trabalho abordou a implementação e simulação de uma Máquina de Turing Universal (MTU) utilizando uma linguagem de programação. A Máquina de Turing Universal foi projetada para receber como entrada uma Máquina de Turing Específica (MTS) e simular sua execução completa. O objetivo principal foi garantir que a MTU possa interpretar e executar corretamente a função de transição da MTS, atualizar o estado, a fita e a posição da cabeça de leitura/escrita até que a MTS alcance um estado final ou entre em um possível loop infinito.

7. REFERÊNCIAS

SIPSER, Michael. Introdução à teoria da computação. São Paulo Cengage Learning 2007