

Implementação de uma Máquina de Turing Universal

Rafael Rios

¹Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre

Resumo. *Este trabalho consiste na implementação de um programa feito em linguagem C que permite processar Máquinas de Turing na notação de 5-tuplas. O programa também permite a compilação e decompilação de máquinas para código binário. A execução pode ser tanto imediata quanto passo a passo, para que se possa depurar com mais facilidade o processamento de uma Máquina de Turing.*

1. Máquina de Turing

Uma Máquina de Turing (MT) é um modelo matemático que consiste de uma fita infinita dividida em células onde é possível realizar operações de leituras e escrita. A posição da fita é definida pelo cabeçote, que permite realizar movimentos unitários para a esquerda, direita ou permanecer na mesma posição. Uma MT pode ser definida como uma 7-tupla $(Q, X, \Sigma, d, q_0, B, F)$, onde:

- Q é o conjunto finito de estados
- X é o alfabeto da fita, tal que $\Sigma \subseteq X$
- Σ é o conjunto finito de símbolos da entrada
- d é a função de transição; $d : Q \times X \rightarrow Q \times X \times \{L, R, S\}$
- q_0 é o estado inicial
- B é o símbolo que representa uma célula vazia
- F é o conjunto de estados de aceitação.

Para esse trabalho, também é relevante a representação através de 5-tuplas que representam as transições entre os estados da MT. Neste modelo, as 5-tuplas são definidas como (Estado_origem, Estado_destino, Entrada, Saida, Movimento)

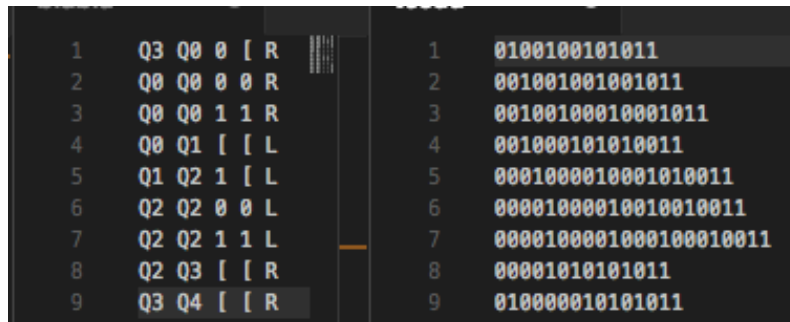
2. Implementação

A implementação através do programa em C foi relativamente simples, podendo ser dividida logicamente em três partes: a leitura e interpretação das transições que definem a MT, a leitura e interpretação das entradas que serão processadas pela MT e o processamento propriamente dito. A codificação e decodificação em binário, no entanto, aumentaram a dificuldade da tarefa. O algoritmo utilizado para a codificação foi:

1. Atribuir um número inteiro a cada estado da MT
2. Atribuir um número inteiro a cada símbolo do alfabeto da fita, sendo o número '0' reservado para representar a célula vazia
3. Atribuir um número inteiro para cada movimento do cabeçote, por exemplo: '1' para a direita, '2' para a esquerda e '3' para permanecer na mesma posição
4. Substituir os valores dentro de uma 5-tupla pela representação unária (símbolo 0) dos números atribuídos anteriormente e separá-los com o símbolo '1'. Desse modo, (Q_0, Q_1, a, b, R) é representado como 01001010010

5. Por fim, concatenar todas as transições (ou 5-tuplas) com '11'

O algoritmo da decodificação é basicamente o processo reverso da codificação, sendo atribuídos números inteiros para cada estado decompilado, e letras minúsculas para cada símbolo decompilado. No entanto, é possível que haja discrepâncias entre os valores dados aos estados símbolos da descrição original e da conversão de binário para 5-tupla, por exemplo, os estados {Q0,Q1,Q2} e as entradas {a,j,h}, depois de transformados em binários e convertidos novamente, tornarão-se {1,2,3} e {a,b,c}



1	Q3	Q0	0	[R	1	0100100101011
2	Q0	Q0	0	0	R	2	001001001001011
3	Q0	Q0	1	1	R	3	00100100010001011
4	Q0	Q1	[[L	4	001000101010011
5	Q1	Q2	1	[L	5	0001000010001010011
6	Q2	Q2	0	0	L	6	00001000010010010011
7	Q2	Q2	1	1	L	7	000010000100010001011
8	Q2	Q3	[[R	8	00001010101011
9	Q3	Q4	[[R	9	010000010101011

Figura 1. À direita, codificação binária da máquina descrita à esquerda

3. Entrada

A entrada do programa deve seguir os mesmos padrões na entrada tanto por arquivo quanto pela linha de comando. Para começar a descrição da máquina, deve-se inserir a tag "BEGIN_MACHINE". Nas próximas linhas, são descritas as 5-tuplas com cada item separado por um espaço (' ') ou simplesmente coloca-se o código binário da máquina para que seja decodificado. Por fim, encerra-se a descrição da máquina com "END_MACHINE". A descrição das entradas é semelhante ao processo anterior. Começa-se com "BEGIN_INPUT", inserem-se as entradas nas linhas seguintes e encerra-se com "END_INPUT". A próxima etapa é definir se a execução deve ocorrer sem parar ou passo a passo. Para isso, insere-se "RUN" para processar imediatamente ou "STEP" para o processo passo a passo. Ao final da execução de todas as entradas, existe a possibilidade de codificar a máquina em binário.

4. Utilização do Programa

O desenvolvimento do trabalho foi feito em ambiente Linux, o comando utilizado para compilação foi o seguinte:

```
gcc *.c -o t1 -g -Wall -Wextra
```

Onde:

- **-g** é a flag que permite depuração através do GNU Project Debugger (GDB)
- **-Wall** habilita diversos warnings desabilitados por padrão, de modo que se possam corrigir bugs os quais teriam passado despercebidos
- **-Wextra** habilita outros warnings que **-Wall** não habilita

Após a compilação, para executar com arquivo de entrada utilizou-se o comando:

```
./t1 <nome\_do\_arquivo>
```

A execução através da linha de comando é ainda mais simples:

```
./t1
```

```
1 BEGIN_MACHINE
2 Q0 Q0 b b R
3 Q0 Q1 a a R
4 Q0 Q2 [ [ L
5 Q1 Q0 a b R
6 Q1 Q0 b b R
7 Q1 Q2 [ [ L
8 Q2 Q2 b b L
9 Q2 Q2 a a L
10 Q2 Q3 [ [ R
11 END_MACHINE
12 BEGIN_INPUT
13 aabaab
14 baaaaa
15 babab
16 END_INPUT
17 RUN
```

Figura 2. Exemplo de arquivo de entrada

4.1. Samples

Junto com o programa, encontram-se 4 arquivos para serem utilizados como entrada, sendo 3 deles MTs no formato de 5-tupla e o outro está na codificação binária. A compilação/decompilação mais recente de uma MT é guardada no arquivo mt.b.