Relatório do 2º Projeto Bimestral de CTC-34

Gabriel Santana, Lucas França, Mateus Menezes, Miguel Macedo

"Escovador 7: Contador de "erros" ortográficos em cadeia
Projetar e implementar uma MT para contar a quantidade de símbolos distintos numa mesma posição em duas cadeias u e v de comprimentos iguais.
Entrada: w e u cadeias sobre {a,b,c} com comprimentos iguais separadas por um "1".
Saída: número (representação unária) de símbolos distintos numa mesma posição das cadeias w e u.
Exemplos: w = baabc1bacba, saída = 00; w = a1b, saída = 0
Testes a serem descritos: 4 exemplos de operação da MT."

Para este problema consideramos:

- "a", "b" e "c": Caracteres de leitura, fazem partes dos argumentos a serem comparados.
- "1": Utilizado para separar os argumentos da função de comparação.
- "0" e "X": Caracteres de controle. São parte essencial da lógica da Máquina. Utilizamos os 0's para marcar caracteres que diferiram na comparação e, os X's para marcar os que já foram processados.

O estado inicial q0 ler um caractere; substitui por o mesmo por B; avança para a direita; e decide que fluxo tomar com no que foi lido.

Os estados q1, q2 e q3 apenas avançam a cabeça de leitura para a direita até encontrar o caractere de separação "1".

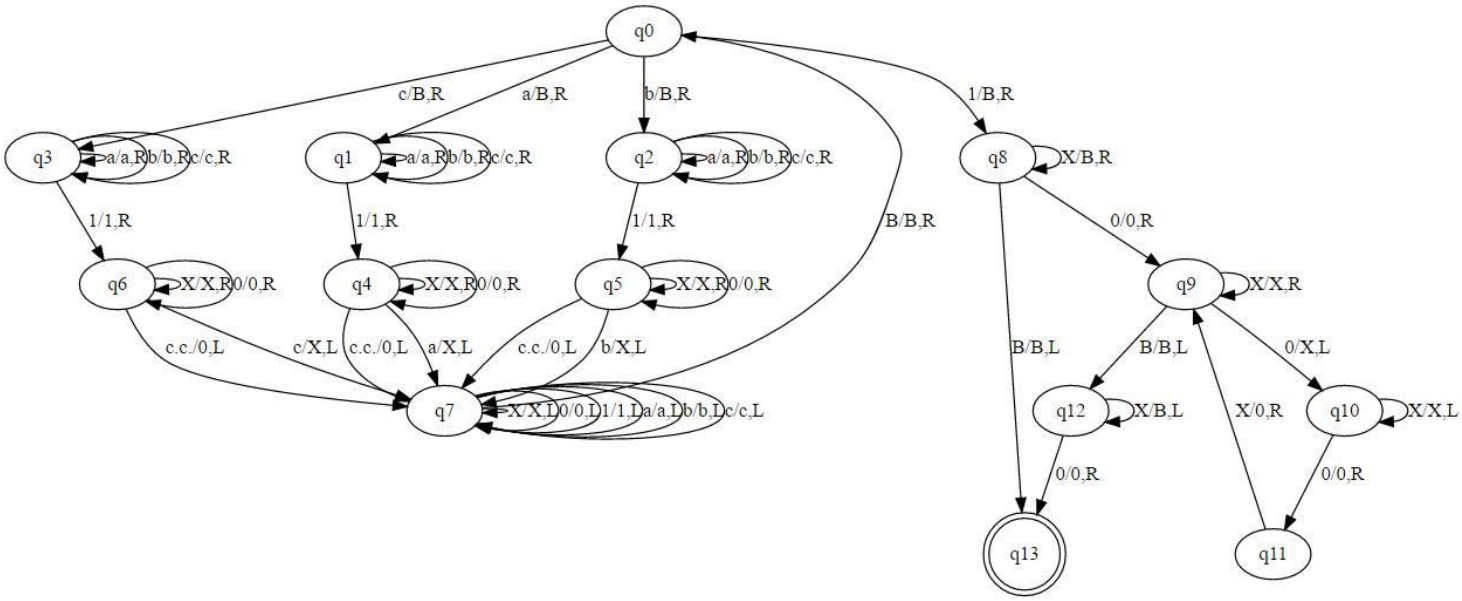
Nos estados q4, q5, e q6 a cabeça de leitura segue à direita passando por todos os caracteres de marcação "0"'s e "X"'s, até encontrar o primeiro caractere de escrita. Se este for igual ao que foi lido na primeira cadeia, substituímos por "X"; caso contrário substituímos por "0". Movemos a cabeça para a esquerda seguimos para o estado q7.

Em q7, movemos a cabeça de leitura para a esquerda até encontrarmos um símbolo "B". Então, movemos a cabeça para a direita para que aponte para o próximo símbolo da primeira cadeia a ser lido e voltamos ao estado q0.

Repetimos este processo até que a primeira cadeia tenha sido completamente processada, ou seja, quando encontramos o "1" no estado q0.

Em q8, lemos os caracteres da cadeia 2, que já foram processados e trocamos todos os "X"'s por "B"'s. Em seguida, ao encontrar o primeiro "0", a máquina concatena todos os outros "0"'s com este, e no processo vai substituindo todos os "X"'s e "0"'s fora de posição por "B"'s. O processo termina quando encontramos um caractere "B" ao percorrer a string para a direita. Neste momento teremos uma sequência de bits, contendo somente zeros, em que o número de "0"'s concatenados representa o número de caracteres que diferiram de uma cadeia para a a outra.

O grafo de transição de estados resultante desse processo está abaixo (ele será enviado junto com esse relatório e estará também no link que contém o código):



Usando a linguagem Java, o grafo foi implementado. A simulação para a cadeia sugerida w = baabc1bacba é a seguinte:

```
Cadeia final de transições [estado, estado seguinte, transição]:
0->2 (b/B,R)
2->2 (a/a,R)
2->2 (a/a,R)
2->2 (b/b,R)
2->2 (c/c,R)
2->5 (1/1,R)
5->7 (b/X,L)
7->7 (1/1,L)
7->7 (c/c,L)
7->7 (b/b,L)
7->7 (a/a,L)
7->7 (a/a,L)
7->0 (B/B,R)
0->1 (a/B,R)
1->1 (a/a,R)
1->1 (b/b,R)
1->1 (c/c,R)
1->4 (1/1,R)
4->4 (X/X,R)
4->7 (a/X,L)
7->7 (X/X,L)
7->7 (1/1,L)
7->7 (c/c,L)
7->7 (b/b,L)
7->7 (a/a,L)
7->0 (B/B,R)
0->1 (a/B,R)
1->1 (b/b,R)
1->1 (c/c,R)
1->4 (1/1,R)
4->4 (X/X,R)
4->4 (X/X,R)
```

```
4->7 (e/0,L)
7->7 (X/X,L)
7->7 (X/X,L)
7->7 (1/1,L)
7->7 (c/c,L)
7->7 (b/b,L)
7->0 (B/B,R)
0->2 (b/B,R)
2->2 (c/c,R)
2->5 (1/1,R)
5->5 (X/X,R)
5->5 (X/X,R)
5->5 (0/0,R)
5->7 (b/X,L)
7->7 (0/0,L)
7->7 (X/X,L)
7->7 (X/X,L)
7->7 (1/1,L)
7->7 (c/c,L)
7->0 (B/B,R)
0->3 (c/B,R)
3->6 (1/1,R)
6->6 (X/X,R)
6->6 (X/X,R)
6->6 (0/0,R)
6->6 (X/X,R)
6->7 (e/0,L)
7->7 (X/X,L)
7->7 (0/0,L)
7->7 (X/X,L)
7->7 (X/X,L)
7->7 (1/1,L)
7->0 (B/B,R)
0->8 (1/B,R)
8->8 (X/B,R)
8->8 (X/B,R)
8->9 (0/0,R)
9->9 (X/X,R)
9->10 (0/X,L)
10->10 (X/X,L)
10->11 (0/0,R)
11->9 (X/0,R)
9->9 (X/X,R)
9->12 (B/B,L)
12->12 (X/B,L)
12->13 (0/0,R)
Cadeia aceita
BBBBBBBB00BBBBBBBBBBBBBBBBB
```

O programa feito percorre o grafo em profundidade. Se um estado de onde não há mais possíveis transições é alcançado, o programa retorna para o estado anterior até que, por exaustão, tente alcançar o estado de aceitação. Se não alcança após percorrer todos os caminhos possíveis, a cadeia é dada como inválida. Enquanto simula a escovação, o programa

imprime os passos da simulação, mostrando toda a varredura do grafo até o ponto em que a cadeia é aceita ou não. Como exemplos, temos:

- Cadeia 1: a1b (saída esperada: 0, pois as duas cadeias diferem de um termo)

```
Cadeia final de transições:
0->1 (a/B,R)
1->4 (1/1,R)
4->7 (e/0,L)
7->7 (1/1,L)
7->0 (B/B,R)
0->8 (1/B,R)
8->9 (0/0,R)
9->12 (B/B,L)
12->13 (0/0,R)
Cadeia aceita
BB0BBBBBBBBBBBBBBBBBBBBBB
```

- Cadeia 2: aa (espera-se não aceitação da cadeia, por não ser do formato v1u)

```
SimulaÃ§Ã£o:
state 0
aaBBBBBBBBBBBBBBBBBBBBBBBB
^
state 1
BaBBBBBBBBBBBBBBBBBBBBBBB
 ^
state 1
BaBBBBBBBBBBBBBBBBBBBBBBB
  ^
Cadeia final de transiÃ§Ãµes:
Cadeia rejeitada
aaBBBBBBBBBBBBBBBBBBBBBBBB
```

Aqui, foi colocada a impressão da simulação, para poder mostrar que o programa tenta percorrer, mas a cadeia é inválida

- Cadeia 3: aa1a (saída esperada: 0, por ter um ‘a’ a mais em um lado)

```
Cadeia final de transições:
0->1 (a/B,R)
1->1 (a/a,R)
1->4 (1/1,R)
4->7 (a/X,L)
7->7 (1/1,L)
7->7 (a/a,L)
7->0 (B/B,R)
0->1 (a/B,R)
1->4 (1/1,R)
4->4 (X/X,R)
4->7 (e/0,L)
7->7 (X/X,L)
7->7 (1/1,L)
7->0 (B/B,R)
0->8 (1/B,R)
8->8 (X/B,R)
```

```
8->9 (0/0,R)
9->12 (B/B,L)
12->13 (0/0,R)
Cadeia aceita
BBBB0BBBBBBBBBBBBBBBBBBBB
```

- Cadeia 4: a1a (saída esperada: ε, porque as cadeias são iguais)

```
Cadeia final de transições:
0->1 (a/B,R)
1->4 (1/1,R)
4->7 (a/X,L)
7->7 (1/1,L)
7->0 (B/B,R)
0->8 (1/B,R)
8->8 (X/B,R)
8->13 (B/B,L)
Cadeia aceita
BBBBBBBBBBBBBBBBBBBBBBBBB
```

Referências:

- Link do repositório com o código e com o grafo: https://github.com/MexicanoT18/Lab2_CTC34
- Slides de aula da disciplina de CTC-34