Concatenação de 2 Máquinas de Turing

Alisson da Silva Vieira, RA: 2046229
Higor Luiz Farinha Celante, RA.: 1602870
Lucas Henrique Malaquias da Silva Donadi, RA: 1711598
Maria Fernanda Pinguelli Sodré, RA: 2046407

22 de setembro de 2019

1 Introdução

1.1 Sobre a Máquina de Turing

Uma máquina de Turing é uma máquina hipotética pensada pelo matemático Alan Turing em 1936. Ela é conhecida como máquina universal muitos anos antes de existirem os modernos computadores digitais. Em um sentido preciso, é um modelo abstrato de um computador, que se restringe apenas aos aspectos lógicos do seu funcionamento (memória, estados e transições), e não a sua implementação física. Apesar de sua simplicidade, a máquina pode simular QUALQUER algoritmo de computador, por mais complicado que seja. (COPELAND, 2000) (MULLINS, 2012)

1.2 Estados

A cabeça contém um sub-dispositivo que é chamado de indicador, que é uma segunda forma de memória de trabalho. O indicador pode ser definido para qualquer uma das várias posições. No jargão da máquina de Turing, a posição do indicador a qualquer momento é chamada de estado da máquina naquele momento. Para dar um exemplo simples da função do indicador, ele pode ser usado para controlar se o último símbolo encontrado foi '0' ou '1'. Se '0', o indicador está na sua primeira posição e, se '1', na sua segunda posição. (SISPER, 1997)

2 Desenvolvimento

2.1 Concatenação

A concatenação é um termo usado em computação para designar a operação de unir algum conteúdo. Usando o exemplo de duas strings, e considerando as strings "casa"e "mento"a concatenação da primeira com a segunda

gera a string "casamento". No caso do nosso projeto, iremos concatenar duas máquinas de turing, como será visto no exemplo a baixo.

2.1.1 Exemplo

Pegamos 2 máquinas de turing, cujo o alfabeto é: a, b. A primeira máquina L1, só aceita um conjunto de palavras, cujo a primeira delas seja um 'a', caso contrário, independente do que vier não será aceita.

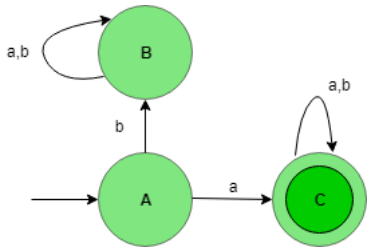

Figura 1 – Representação da máquina L1. (RAY, 2000)

O estado 'A' verifica se a entrada começa com 'a', caso negativo, ele vai para o estado B, onde nesse ele apenas chega no final do palavra. Caso positivo, ele vai para o estado C, este que é o estado de aceitação.

Já a segunda máquina, só aceita um conjunto de palavras que termina com 'b'. Caso o conjunto de entrada não termine com 'a', a palavra não será aceita.

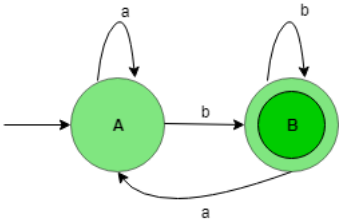

Figura 2 – Representação da máquina 2. (RAY, 2000)

O estado 'A', verifica se a palavra em questão é um 'a', ou um 'b', caso seja um 'a', ela roda até achar um 'b'. Caso não ache, a máquina rejeita a linguagem. Por consequência, caso a máquina ache um 'b', ela vai para o estado 'B'. Neste estado, caso a máquina ache um 'a', ela volta ao estado 'A', caso contrário, ela verifica caso a palavra termine com 'b'.

Logo a concatenação dessas duas máquinas resultaria em uma máquina L3 que apenas aceita palavras que comecem com a letra 'a', como é o estado de aceitação na máquina L1, e termine com a letra 'b', que novamente, é o estado de aceitação da L2.

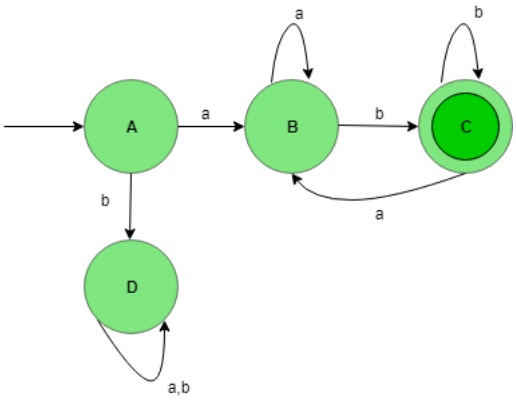

Figura 3 – Representação da máquina L3 resultante da concatenação de L1 e L2 (RAY, 2000)

Esta máquina L3 tem 4 estados: 'A', 'B', 'C' e 'D'. No começo, no estado 'A', a maquina verifica se a palavra de entrada começa com a letra 'a'. Caso negativo, a máquina pula para o estado 'D', onde independete do que vier, a máquina não aceitará a palavra. Caso positivo, a máquina irá ao estado 'B', onde ela verifica se a letra é 'b' ou 'a'. Se for 'a', ela permanece no mesmo estado, se não, ela vai para o estado 'C', onde neste a máquina verifica novamente se a letra é 'a' ou 'b'. Caso for 'a', ela volta ao estado 'B', caso for 'B', ela repete o estado 'C' até o final da palavra.

2.2 Código

Para a implementação do algoritmo que realiza a concatenação foi utilizado a linguagem de programação Python e a biblioteca *sys*. Nesta seção está descrito cada função criada e suas respectivas entradas e saídas.

2.2.1 Função: Replace

A função **replace(maquina1, maquina2, 1)** é um *looping* recursivo que compara os nomes de estado das duas máquinas recebidas por parâmetro, caso houver nomes iguais:

1. tenta a concatenação dos dois nomes iguais para gerar o novo nome para o estado na máquina 2, caso já houver algum estado com esse nome em alguma das duas máquinas, cai no caso 2.

2. Chama um método secundário com o nome de **buscaNNE()** que busca um número inteiro que não esteja sendo usado como nome.

Por fim ela retorna a máquina 2 com a linha de nomes do estados renomeada.

```
51  def renomeia(maq2, auxList2):
52      nomes1 = []
53      nomes1 = auxList2[3]
54      cont = 7
55      aux = (auxList2[7:])
56      indini = nomes1.index(auxList2[4][0]) # indice do estado inicial
57      maq2[4] = [maq2[3][indini] ] # lista com a string dentro # string referente estado inicial
58      indifi = nomes1.index(auxList2[5][0]) # indice do estado inicial
59      maq2[5] = [maq2[3][indifi] ] # lista com a string dentro # string referente estado final
60      for tr in aux:
61          indsub = nomes1.index(tr[0])
62          maq2[cont][0] = maq2[3][indsub]
63          indsub2 = nomes1.index(tr[1])
64          maq2[cont][1] = maq2[3][indsub2]
65          cont += 1
66      return maq2
```

Figura 4 – Código da função renomeia

2.2.2 Função: Renomeia

A função renomeia(maquina2, maquina2old) recebe a máquina 2 nova, já com a linha de nomes dos estados renomeados, que a função replace retornou, e a máquina 2 na sua versão original que não sofreu alterações, e servirá de referência para os antigos estados. Aqui o código da função:

Ela compara os nomes dos estados de ambas as máquinas e a partir dos estados e seus respectivos índices substitui os nomes antigos dos estados que estão na máquina 2 antiga pelos novos nomes que estão na máquina 2 nova. Por fim retorna a máquina 2 totalmente renomeada.

2.2.3 Função: Monta

Essa função tem como objetivo montar a 3 máquina que será a resultante da concatenação de L1 e L2. O código abaixo mostra como foi implementada a função.

Inicialmente é declarado 'maq3', que é a nossa L3. Dentro do primeiro for, fazemos a passagem de valor de L1 e L2, por exemplo, maq3[4] é a quarta linha da maquina, e nesta aparece qual o estado inicial da máquina 3, que no caso é o mesmo estado inicial da máquina 1. Da mesma maneira, maq3[5], a quinta linha de L3, aparece o estado final da mesma, que no caso, é o mesmo estado final da máquina 2. Já no segundo for, copiamos todas as transições da L1 para L3. Como inicialmente L3 está sem nenhuma transição, podemos apenas copiar L1 para L3, porem precisamos fazer uma nova transição entre a transição final de L1 e a transição inicial de L2 para a concatenação. Resolvemos esse problema na linha 87, onde chamamos uma função auxiliar chamada: 'cria-nova-tr'.

```
76  def monta(maq1, maq2):
77      maq3 = []
78
79      for i in range(4):
80          maq3.append(maq1[i] + maq2[i])
81      maq3.append(maq1[4])
82      maq3.append(maq2[5])
83      maq3.append(maq1[6])
84      for trans in (maq1[7:]):
85          maq3.append(trans)
86      lista_trs = []
87      lista_trs = cria_nova_tr(maq1,maq2)
88      for trs in lista_trs:
89
90          print("> Transicao de conexao adicionada")
91          print(" - ",trs)
92          maq3.append(trs)
93
94      for tr2 in (maq2[7:]):
95          maq3.append(tr2)
96
97      return maq3
```

Figura 5 – Código da função Monta

```
99  def cria_nova_tr(maq1,maq2):
100     tr_novas = []
101     for final in maq1[5]:
102         for inicial in maq2[4]:
103                 # para cada final, uma transicao para cada inicial da maq2 [final_maq1, inicial_maq2, branco, branco, Stay]
104                 tr_novas.append([final,inicial,maq1[2][0],maq1[2][0],'S'])
105     return tr_novas
106
```

Figura 6 – Código para montar uma nova transição

Logo, essa função pega a transição final de L1, pelo 'for final in maq1[5]', onde maq1[5] é a transição final de L1. E 'inicial in maq2[4]', onde maq2[4] é onde está a transição inicial de L2. Assim, a linha 104 cria uma nova transição e retorna ela. Assim, após adicionar a nova transição, a função adiciona as transições restantes de L2 em L3, através do for da linha 94.

Referências

COPELAND, J. *What is a turing machine?* 2000. Site: http://www.alanturing.net/turingarchive.html. Acesso em: 13.09.2019. Citado na página 1.

MULLINS, R. *What is a turing machine?* 2012. Site: https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html. Acesso em: 4.9.2019. Citado na página 1.

RAY, K. *TOC | Concatenation process in DFA*. 2000. Site: https://www.geeksforgeeks.org/toc-concatenation-process-in-dfa/. Acesso em: 16.09.2019. Citado 2 vezes nas páginas 2 e 3.

SISPER, M. *Introdução a Teoria da Computação*. [S.l.: s.n.], 1997. Citado na página 1.