Teste de Dados e Computação 2020

Universidade do Minho, 5 de Junho de 2020

I. **Autómatos Finitos**

Considere as linguagens sobre o alfabeto $\Sigma = \{0, 1\}$ definidas informalmente como se segue:

- L_{pares} é a linguagem de todos os números pares em representação binária, ou seja, a linguagens de todas as palavras que terminam em 0: 0, 10, 100, 110, 1000, 1010, ...
- $L_{n1impar}$ é a linguagem das palavras que têm um número de ocorrências ímpar do símbolo 1
- $L_{paresn1impar}$ é a linguagem das palavras que terminam em 0 e têm um número de ocorrências ímpar do símbolo 1

1.
   a. Apresente um autómato determinístico para reconhecer L_{pares}.
   b. Apresente um autómato determinístico para reconhecer $L_{n1impar}$.
   c. Apresente um autómato determinístico para reconhecer $L_{paresn1impar}$.

2. Generalizando a sua resposta à questão anterior, admitindo que são dados dois DFAs D_1 e D_2 para o reconhecimento respectivamente das linguagens L_1 e L_2, diga como se pode obter de forma sistemática um autómato D_{12} para o reconhecimento da linguagem $L_1 \cap L_2$.

3. Recorde o resultado provado pelo *pumping lemma* sobre a linguagem $\{a^i b^i \mid i \geq 0\}$. Considere agora a linguagem L das palavras sobre $\{a, b\}$ que contêm tantos a como b, e note que $\{a^i b^i \mid i \geq 0\} = L \cap \{a\}^* \{b\}^*$. Diga, justificando, se L é ou não uma linguagem regular.

4.
   a. Escreva uma expressão regular r para descrever as palavras da linguagem $L_{paresn1impar}$.

b. Aplique o processo sistemático estudado nas aulas, para obter a partir de r um autómato **não-determinístico** para o reconhecimento desta linguagem. Compare o DFA obtido em 1c. com este NFA.

II. **Autómatos de Pilha**

Considere a seguinte gramática livre de contexto para expressões aritméticas construídas com os operadores + e *, parêntesis (,), e variáveis id $\in \{x, y, z, \ldots\}$ (já apresentada nos exercícios resolvidos)

- S → S+X
- S → X
- X → X*Y
- X → Y
- Y → (S)
- Y → id (uma regra para cada variável id)

a. Desenhe a árvore de sintaxe da expressão $x * (y + z)$ segundo esta gramática
b. Defina um autómato de pilha de um único estado para o reconhecimento da linguagem definida por esta gramática, recorrendo ao método *top-down* apresentado nas aulas
c. Simule uma execução do autómato que aceite a palavra $x * (y + z)$, mostrando os passos de construção da árvore de sintaxe.

III. **Máquinas de Turing**

Considerando a representação unária dos números naturais, conceba uma máquina de Turing para implementar a operação "-1", que calcula o antecessor de um número. Considere que a fita contém inicialmente o número cujo antecessor se pretende calcular, representado por uma sequência de "1". A cabeça da máquina está inicialmente posicionada sobre o primeiro símbolo do primeiro número, e deverá estar assim no final.

Por exemplo se a fita contiver inicialmente (B é o símbolo "blank")

... | 1 | 1 | 1 | 1 | B |...

no final deverá conter

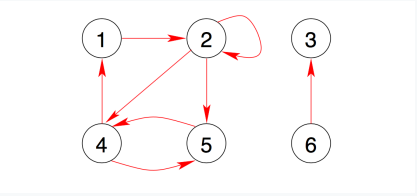
... | 1 | 1 | 1 | B | ...

Note que, caso o input seja o número 0, a execução da máquina não deverá parar,

executando em ciclo infinito, sinalizando assim que o antecessor de 0 não é um número natural. Apresente a sua MT de forma gráfica.

IV. **Programação com Grafos**

Tomando por base a representação de grafos orientados estudada, e a função básica de travessia em profundidade:



Grafo orientado g

```
g = {}
g[1] = [2]
g[2] = [2,4,5]
g[3] = []
g[4] = [1,5]
g[5] = [4]
g[6] = [3]
```

Escreva uma função `reachable` que, quando chamada com um grafo e dois vértices, indica se existe ou não um caminho do primeiro para o segundo vértice:

```
print (reachable(g, 1, 5))
True
print (reachable(g, 1, 3))
False
print (reachable(g, 6, 3))
True
print (reachable(g, 3, 6))
False
```

FIM