Universidade Federal de Viçosa – Campus UFV-Florestal
Ciência da Computação – Fundamentos da Teoria da Computação
Professor: Daniel Mendes Barbosa

Trabalho Prático 3

Alunos:

Germano Barcelos dos Santos (3873)
Guilherme Corrêa Melos (3882)
Fábio Trindade Ramos (3869)
Samuel Aparecido Delfino Rodrigues (3476)
Taís Batista dos Santos (3036)
Vinícius Júlio Martins Barbosa (3495)

Florestal - Outubro de 2021

Introdução	**3**
Desenvolvimento	**3**
Conclusão	**4**

Introdução

O trabalho consiste em, a partir da entrada, configurar um autômato e logo em seguida utilizá-lo para aceitar ou rejeitar palavras de acordo com a linguagem configurada. Também foi pedido que se implementasse ao menos uma funcionalidade extra dentre as listadas, o grupo optou por desenvolver o alfabeto de entrada e o autômato finito não determinístico.

Desenvolvimento

O grupo escolheu a linguagem de programação python para desenvolver o trabalho, isso porque é uma linguagem que todos integrantes do grupo possuem conhecimento.

Para modelar os autômatos, utilizamos um grafo orientado com arestas representadas por uma tupla (destino, símbolo aceito). O grafo possui uma lista de estados iniciais, dando a possibilidade de representar autômatos finitos não determinísticos e uma lista de estados finais. Como é possível visualizar abaixo.

```python
class Node:
    def __init__(self, name, initial=False, final=False) -> None:
        self.name = name
        self.initial = initial
        self.final = final

class Graph:
    def __init__(self) -> None:
        self.edges = {}
        self.nodes = {}
        self.init_states = []
        self.final_states = []

    def add_nodes(self, node: Node):
        self.edges[node.name] = []
        self.nodes[node.name] = node

        if node.initial:
            self.init_states.append(node.name)

        if node.final:
            self.final_states.append(node.name)

    def get_node(self, source_name: str) -> Node:
        if source_name in self.nodes.keys():
            return self.nodes[source_name]
        return None
```

Figura 1 - Representação por meio de grafo

Para percorrer o grafo usou-se o DFS, algoritmo que faz busca em profundidade, começando pela raiz. Para cada transição possível do nó, uma

chamada recursiva é feita, incrementa-se o valor da posição em uma unidade da lista de símbolos se o símbolo é diferente de '\' ou existe um lambda ('').

```python
def dfs(self, edges: list, curr_state: Node, list_symbols: str, pos: int) -> bool:
    ans = False

    if pos >= len(list_symbols):
        if curr_state.final:
            return True
        else:
            return False

    neighs, transitions = zip(*edges)

    for i in range(len(transitions)):
        if transitions[i] == list_symbols[pos] or transitions[i] == '\\':
            next_state_name = neighs[i]
            n = self.get_node(next_state_name)
            edges = self.edges[n.name]

            if transitions[i] == '\\':
                ans |= self.dfs(edges, n, list_symbols, pos)
            else:
                ans |= self.dfs(edges, n, list_symbols, pos+ 1)

    if list_symbols[pos] == '':
        ans |= self.dfs(edges, curr_state, list_symbols, pos+1)

    return ans
```

Figura 2 - DFS

Para ler o arquivo de entrada, foi preciso identificar o padrão de leitura para as transições para cada estado. É possível que uma mesma transição seja feita por mais de um símbolo do alfabeto. Nesse caso, cada símbolo é separado por um espaço, assim todo e qualquer símbolo pode ser lido de 2 em 2 caracteres.

Conclusão

Foi possível, com este trabalho, abstrair a modelagem de um autômato para um grafo. Assim podemos utilizar estruturas de dados já estudadas durante o curso e aplicá-las em um projeto prático.