**1º Projeto de CTC 34 – Automata e linguagens formais**

Alunos: Gustavo Ceci Guimarães e Luiz Felipeh Aguiar de Lima Alves

**1. Implementação**

O projeto foi concebido na linguagem *Python*, no arquivo *turing.py*, e sua entrada foi escolhida como um arquivo de texto. Sua padronização de *input* foi feita em conjunto com outro grupo, e suas instruções de uso se encontram detalhadamente explicadas no arquivo de texto que se encontrar no GitHub do projeto. Basicamente, as informações passadas neste arquivo são: alfabeto, nome dos estados, estados finais, transições e cadeia a ser lida. Foi feita a assunção de que, como o projeto foi desenvolvido para autômatos determinísticos, a entrada sempre conterá todas as transições necessárias e todas as transições são válidas.

A lógica de armazenamento de transições foi implementada utilizando o conceito de matriz e seguindo a seguinte lógica: as linhas se referem ao estado atual, as colunas se referem ao atual símbolo a ser lido, o primeiro elemento de cada elemento possui duas informações: módulos dos valores se referem ao próximo estado e os sinais dos valores se referem a direção da leitura (positivo: direita; negativo: esquerda). O segundo elemento de cada elemento possui a informação de qual caracter deve ser escrito no lugar. Além disso, na matriz, a linha 0 não poderia ser usada, pois a informação “sinal” seria perdida. A seguir, um exemplo:

Alfabeto: [a, b] Estados: [q0, q1, q2] Estado inicial: q0 Estado final: q1

Transições: q0 --a--> q1, a, R q1 --a--> q1, a, L q2 --a--> q2, a, L

q0 --b--> q2, b, R q1 --b--> q0, a, R q2 --b--> q0, a, L

Matriz:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 0 | 1 |
|  |  | a | b |
| 0 |  |  |  |
| 1 | q0 | [+2,a] | [+3,b] |
| 2 | q1 | [-2,a] | [+1,a] |
| 3 | q2 | [-3,a] | [-1,a] |

Os principais métodos do projeto são:

* **GetStates**: Após a leitura do arquivo, esse método monta a matriz da maneira indicada acima.
* **IsValid**: Executa a simulação.
* **DebugLog**: Método que organiza a impressão de informação na tela, garantindo um debug mais efeitvo

A execução do programa se dá de duas formas:

* python turing.py –b \_\_\_\_\_\_\_\_.txt
  + Modo normal de execução.
* python turing.py –d \_\_\_\_\_\_\_\_.txt
  + Modo debug de execução.

Depois da implementação da máquina de Turing, foi desenvolvido um grafo com a finalidade de criar a sequencia de Fibbonacci utilizando nossa maquina, com a seguinte sintaxe:

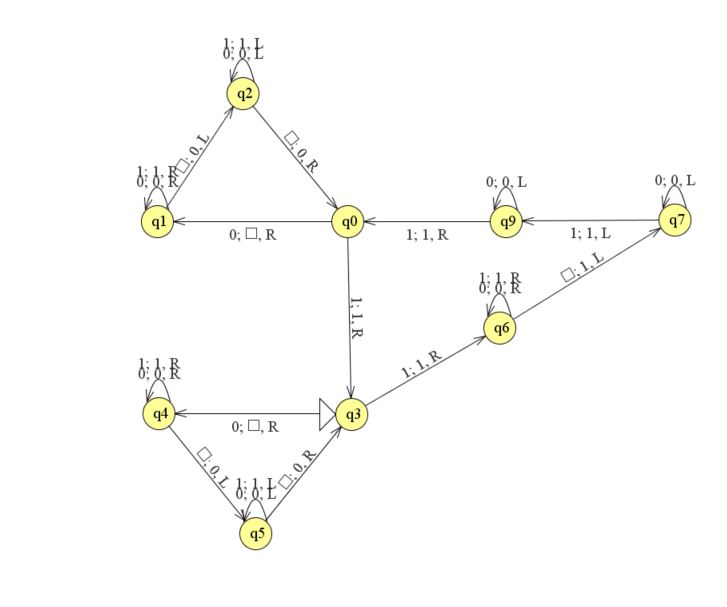
10101001000100000100000000100000000000001...

Dado como input 1010 (os valores iniciais 1 e 1 da sequencia) geramos até o próximo 1 após atingido o tamanho máximo desejado. O algoritmo consiste, basicamente, em 3 loops:   
Loop 1: Adicionar o 1 ao final da geração dos 0’s que compõe o número de Fibonacci.

Loop 2: Copiar os 0’s do primeiro número que vai compor o número de Fibonacci em questão

Loop 3: Copiar os 0’s do segundo número que vai compor o número de Fibonacci em questão

Abaixo, segue a representação gráfica do grafo acima descrito:



**2. Resultados**

A sequência de Fibonacci foi gerada com sucesso! Segue, abaixo, um exemplo:



**3. Conclusão**

Baseado no resultado obtido, o objetivo do projeto foi obtido. Foi feito um programa em Python relativamente simples, bem comentado e de fácil entendimento, seguindo também as diretrizes de Engenharia de Software aprendidas no curso. A entrada foi relativamente trabalhosa de ler, mas foi escolhida por ser mais intuitiva pro usuário. Todos os resultados foram comparados com sucesso a seus teóricos. Por esses motivos, considera-se que o objetivo foi cumprido.

**4. Código**

O código também se encontra no repositório <https://github.com/LuizFelipeh/turingmachine> . A seguir, segue-se o código para leitura:

