FRAMEWORK DE MÁQUINA DE TURING:
Uma biblioteca Java para a construção e uso de máquinas de Turing

Pedro Ryan Coelho Iplinski[1]; *Rodrigo Curvêllo*[2]

[1] Estudante de Graduação em Bacharelado em Ciência da Computação, IFC - *Campus* Rio do Sul. E-mail: pedroryancoelhoiplinski@gmail.com
[2] Orientador, Professor EBTT, IFC - *Campus* Rio do Sul. E-mail: rodrigo.curvello@ifc.edu.br

RESUMO

A linguagem é um dos conceitos mais fundamentais da computação e da informática. Uma linguagem formal é um conjunto de palavras (também chamadas de cadeias de caracteres) baseadas em um conjunto de símbolos. Essas linguagens são classificadas desde a classe de linguagens mais simples até à classe das linguagens recursivamente enumeráveis, sendo que esta última corresponde à maioria das linguagens. É possível verificar se uma cadeia pertence ou não a uma linguagem com o uso de autômatos, ou seja, algoritmos que reconhecem a linguagem. O algoritmo que reconhece as linguagens recursivamente enumeráveis é a máquina de Turing (MT). Considerando a utilidade de uma MT, este trabalho apresenta o desenvolvimento de um framework para a linguagem de programação Java que permite a construção e uso de máquinas de Turing para, assim, verificar se uma cadeia pertence ou não a uma linguagem recursivamente enumerável. Após ser realizado um estudo sobre a estrutura e a lógica de funcionamento da MT, essa biblioteca foi implementada com a linguagem de programação Java, seguindo o paradigma de orientação a objetos e aplicando o padrão de projeto *Fluent Interface*. No fim, o framework desenvolvido oferece funções que podem ser utilizadas na linguagem Java para a estruturação de máquinas de Turing e seu uso para reconhecer as cadeias de uma linguagem.

Palavras-chave: Autômato; Linguagem; Algoritmo; Computação; Informática.

INTRODUÇÃO

A linguagem é um dos conceitos mais fundamentais da computação e da informática. Uma linguagem formal é um conjunto de palavras (também chamadas de cadeias de caracteres) baseadas em um conjunto de símbolos, ou seja, um alfabeto. Segundo a hierarquia elaborada por Noam Chomsky, as linguagens formais são classificadas desde a classe de linguagens mais simples até à classe das linguagens recursivamente enumeráveis, sendo que esta última contém todas as outras classes e, assim, corresponde à maioria das linguagens.

É possível verificar se uma cadeia pertence ou não ao conjunto de cadeias de uma linguagem com o uso de autômatos, os quais são algoritmos/mecanismos que reconhecem a linguagem. O algoritmo que reconhece as linguagens recursivamente enumeráveis é a máquina de Turing (MT), proposta por Alan Turing em 1936 (Menezes, 2011).

De acordo com Hopcroft *et al.* (2002), uma máquina de Turing é, essencialmente, um autômato finito que tem uma única fita de comprimento infinito, a qual ele pode ler ou gravar dados. Para verificar se uma cadeia pertence ou não a uma linguagem recursivamente enumerável, essa cadeia é adicionada à fita e, então, a MT realiza a leitura de cada símbolo da cadeia se movimentando pela fita através de transições definidas.

ISSN 2318-2342

A definição formal de uma MT é:

M = (Q, Σ, Γ, δ, q_0, B, F)

em que:

- Q é o conjunto finito de estados;
- Σ é o conjunto finito de símbolos de entrada (ou seja, o alfabeto da cadeia);
- Γ é o conjunto finito de símbolos da fita, em que Σ ⊆ Γ;
- δ é a função de transição, a qual depende de dois argumentos:
  - δ(q, *x*), em que q é o estado atual da MT e *x* é o símbolo lido na fita;
  - E tem, como saída, δ(p, Y, D), em que p é o estado que a MT assume, Y é o símbolo que é gravado na fita e D é a direção em que a cabeça de controle da fita se movimenta (podendo ser para a esquerda ou para a direita).
- q_0 é o estado inicial da MT;
- B é o símbolo branco que se encontra nas extremidades da fita;
- F é o conjunto finito de estados de aceitação, ou seja, estados finais.

Considerando a utilidade de uma MT, este trabalho apresenta o desenvolvimento de um framework, ou seja, uma biblioteca de funções, para a linguagem de programação Java que permite a construção e uso de máquinas de Turing para verificar se uma cadeia pertence ou não a uma linguagem recursivamente enumerável.

PROCEDIMENTOS METODOLÓGICOS

Para implementar o framework de máquina de Turing, primeiramente foi necessário estudar a estrutura e a lógica de funcionamento da MT. Com isso feito, a biblioteca foi desenvolvida com a linguagem de programação Java, seguindo o paradigma de orientação a objetos e aplicando o padrão de projeto *Fluent Interface* em sua construção.

RESULTADOS E DISCUSSÃO

Foram desenvolvidas as classes que representam a máquina de Turing, os estados e as transições. Seus métodos, ou seja, suas funções, permitem criar os estados da MT; definir o símbolo branco, o estado inicial e quais estados são finais; e criar as transições a partir do estado que é seu argumento. Com as transições estabelecidas, o framework reconhece automaticamente quais são os símbolos de entrada e os símbolos da fita e gera a definição formal da função de transição.

Para verificar se uma cadeia pertence à linguagem reconhecida pela MT, foram implementados métodos que montam a fita (adicionando a cadeia) e aplicam as transições para fazer a leitura. Além disso, foi desenvolvido um método que gera a descrição instantânea (em inglês, *Instantaneous Description* – ID) da verificação de uma cadeia, a qual representa a configuração/estado da fita em cada momento da análise.

Na figura 01 a seguir, é apresentado o diagrama de classes que demonstra a estrutura do framework.

ISSN 2318-2342

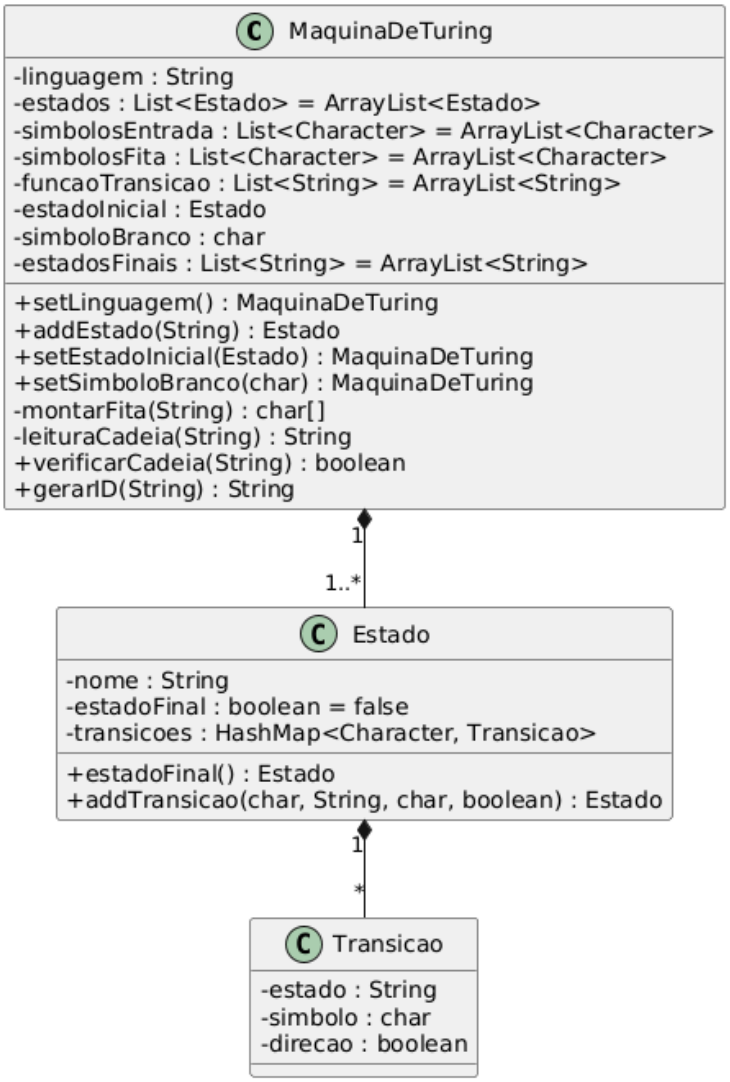

Figura 01 – Diagrama de classes do framework

A figura 02 demonstra o código da construção de uma MT, com o uso dos métodos desenvolvidos, que reconhece a linguagem das cadeias com a mesma quantidade de “0” e “1” em qualquer ordem. Nas figuras 03 e 04, é apresentado o código da verificação de uma cadeia juntamente de sua ID e seu resultado/saída.

ISSN 2318-2342

```
MaquinaDeTuring mt = new MaquinaDeTuring('B');

Estado q0 = mt.addEstado("q0").addTransicao('1', "q2", 'Y', true)
    .addTransicao('0', "q4", 'X', true).addTransicao('B', "q6", 'B', true)
    .addTransicao('X', "q0", 'X', true).addTransicao('Y', "q0", 'Y', true);
mt.setEstadoInicial(q0);

mt.addEstado("q2").addTransicao('0', "q3", 'X', false)
    .addTransicao('1', "q2", '1', true).addTransicao('X', "q2", 'X', true)
    .addTransicao('Y', "q2", 'Y', true);

mt.addEstado("q3").addTransicao('Y', "q0", 'Y', true)
    .addTransicao('0', "q3", '0', false).addTransicao('1', "q3", '1', false)
    .addTransicao('X', "q3", 'X', false);

mt.addEstado("q4").addTransicao('1', "q5", 'Y', false)
    .addTransicao('0', "q4", '0', true).addTransicao('X', "q4", 'X', true)
    .addTransicao('Y', "q4", 'Y', true);

mt.addEstado("q5").addTransicao('X', "q0", 'X', true)
    .addTransicao('0', "q5", '0', false).addTransicao('1', "q5", '1', false)
    .addTransicao('Y', "q5", 'Y', false);

mt.addEstado("q6").estadoFinal();
```

Figura 02 – Código da construção da estrutura de uma MT

```
String cadeia = "10";

System.out.println(mt.verificarCadeia(cadeia));
System.out.println(mt.gerarID(cadeia));
```

Figura 03 – Código da verificação de uma cadeia

```
true
(Bq010B) -> (BYq20B) -> (Bq3YXB) -> (BYq0XB) -> (BYXq0B) -> (BYXBq6) -> Aceita
```

Figura 04 – Resultado da verificação de uma cadeia

CONSIDERAÇÕES FINAIS

No fim, o framework desenvolvido cumpre o objetivo do trabalho, fornecendo uma biblioteca de funções que pode ser utilizada na linguagem Java para a estruturação de máquinas de Turing e seu uso para reconhecer as cadeias de uma linguagem. Em versões futuras, pode ser implementado um método que gera uma visualização gráfica da ID da verificação de uma cadeia. Além disso, o framework pode ser ampliado com um método que recebe somente a expressão de uma linguagem recursivamente enumerável e, analisando tal expressão, o método automaticamente constrói a máquina de Turing por completo, criando seus estados e transições, definindo seu estado inicial e estados finais, etc.

ISSN 2318-2342

REFERÊNCIAS

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introdução à teoria dos autômatos, linguagens e computação**. Rio de Janeiro: Elsevier, 2002.

MENEZES, Paulo Blauth. **Linguagens formais e autômatos**. 6. ed. Porto Alegre: Bookman, 2011.

ISSN 2318-2342