Implementação de DFAs e ENFAs configuráveis em Java

Antônio Galvão S. Freitas, Mikeias Gabriel M. Azevedo, Leandro Carlos de Souza

Departamento de Ciências Exatas e Naturais – Universidade Federal Rural do Semi-Árido (UFERSA) 59625-900 – Mossoró – RN – Brasil

{goantonio80,contato.mikeiasgabriel}@gmail.com,
leandro.souza@ufersa.edu.br

Abstract. *This article aimed to develop, according to computational theory, an application that helps in teaching and learning the discipline of formal languages and automata, of the computer science course. The application is used to recognize deterministic finite automata and non-deterministic finite automata, configurable for any regular language. It receives as input a text file with the specified models and generates the automata that represent these languages as well as the acceptance test for certain chains provided by the user.*

Resumo. *Este artigo teve como objetivo desenvolver, de acordo com a teoria computacional, uma aplicação que serve de auxílio no ensino e aprendizagem da disciplina de linguagens formais e autômatos, do curso de ciência da computação. Sendo usada para reconhecer autômatos finitos determinísticos e autômatos finitos não-determinísticos configuráveis para qualquer linguagem regular, a aplicação recebe como entrada um arquivo de texto com os modelos especificados e gera os autômatos que representam essas linguagens, bem como o teste de aceitação de determinadas cadeias fornecidas pelo usuário.*

1. Introdução

Desde os primórdios da computação, quando lógicos matemáticos começaram a explorar os seus significados, uma questão sempre esteve sob enfoque em uma perspectiva teórica: “Quais são as capacidades e limitações fundamentais dos computadores?”. No entanto, os avanços tecnológicos e o aumento da capacidade computacional movimentaram essa problemática do mundo abstrato para a nossa realidade concreta. Com isso, três nichos teóricos surgiram para compreender as fronteiras computacionais.

As três áreas principais que surgiram foram a da teoria da complexidade, que lida com o nível de dificuldade para solucionar problemas computacionais (fáceis ou difíceis), a teoria da computabilidade, que lida com a possibilidade de resolver problemas computacionalmente, e a teoria dos autômatos, que se debruça sobre modelos que determinam definições e propriedades amplamente aplicáveis nas áreas da ciência da computação.

A teoria dos autômatos permite análises teóricas a partir de abstrações do computador e que possibilitam a concepção das limitações computacionais daquele modelo mediante provas matemáticas, ou seja, esses modelos auxiliam nos estudos de

computabilidade e complexidade, já que esses últimos necessitam de um modelo computacional bem definido. Além das implicações teóricas, há também benefícios práticos na modelagem de problemas.

Os autômatos estão intimamente ligados à teoria das linguagens formais, sendo que também podem ser definidos como uma representação formal de uma linguagem formal, ou seja, um formalismo reconhecedor. As principais classes de autômatos são descritas na tabela a seguir com a respectiva classe de linguagem que eles reconhecem:

Tabela 1: Correspondência entre a classe de autômatos e as classes de linguagens reconhecidas

Classes dos autômatos	Classes das linguagens reconhecidas
Autômatos finitos determinísticos (DFA)	Linguagens regulares
Autômatos finitos não-determinísticos (NFA)	Linguagens regulares
Autômatos finitos não-determinísticos com transições vazias (ENFA)	Linguagens regulares
Autômatos com pilha (PDA)	Linguagens livres de contexto
Máquinas de Turing	Linguagens recursivamente enumeráveis

Fonte: Elaborado com base no livro “Introdução da Teoria da Computação” do autor Michael Sipser, 2019.

Dito isso, pode-se construir a percepção de um autômato formalmente através da sua definição matemática. Para a classe dos autômatos finitos, dizemos que eles são 5-tuplas (Q, Σ, δ, q0, F), onde Q é um conjunto finito de estados, Σ é um conjunto de símbolos chamado de alfabeto, δ é a função de transição a partir de um estado e um símbolo de entrada que resulta em um único estado (para os autômatos finitos determinísticos) ou um subconjunto de estados (para os autômatos finitos não-determinísticos), q0 é um estado inicial pertencente a Q e F é um subconjunto de Q chamado de estados de aceitação. Para os autômatos finitos não-determinísticos com transições vazias, δ toma, não apenas um estado e um símbolo, mas também pode haver transições espontâneas (a partir da cadeia vazia) que resultam em um subconjunto de estados.

Também é importante enunciar a relação de equivalência entre os autômatos finitos determinísticos e os autômatos finitos não-determinísticos, uma vez que eles reconhecem a mesma classe de linguagem. A prova para essa afirmação é feita por construção e se fundamenta no seguinte teorema: Todo autômato finito não-determiní´stico tem um autômato finito determiní´stico equivalente.

Com base no que foi apresentado e as contribuições no aprendizado na teoria dos autômatos, o presente trabalho tem como objetivo a implementação de autômatos finitos determinísticos (DFA) e não-determinísticos com transições vazias, aproveitando-se da equivalência com os autômatos não-determinísticos (NFA), em linguagem de programação que podem ser configurados de forma genérica para reconhecer linguagens

regulares, podendo ser utilizados para fins didáticos e teóricos também para verificação de propriedades da teoria dos autômatos. Para o projeto em questão, é importante considerar que as diferentes classes de autômatos implementadas reconhecem a mesma classe de linguagens formais, as regulares.

2. Metodologia

Para o desenvolvimento do trabalho foi utilizado como fundamentação teórica o livro "Introdução à Teoria da Computação" do autor Michael Sipser também "Introdução à Teoria dos Autômatos, Linguagens e Computação" dos autores John E. Hopcroft, Jeffrey D. Ullman e Rajeev Motwani.

Após o levantamento bibliográfico, a implementação foi realizada em Java devido ao alto grau de abstração fornecido pela linguagem e algumas classes que otimizam o processo de desenvolvimento. A implementação foi dividida em etapas. Primeiramente, foi implementada uma forma dos autômatos receberem as configurações de um arquivo de texto em um formato predefinido, esse arquivo deve conter o estado inicial, os estados aceitação e as transições. Após a obtenção desses dados, os autômatos podem ser executados, verificando cadeias passadas. É importante considerar que o alfabeto também é definido nesse arquivo de configuração. Para fazer a leitura desses dados de forma consistente, foi desenvolvida uma classe "StringUtils" no pacote "utils" que contém métodos para manipular Strings e vetores de inteiros (que foi a forma como foram armazenados os estados).

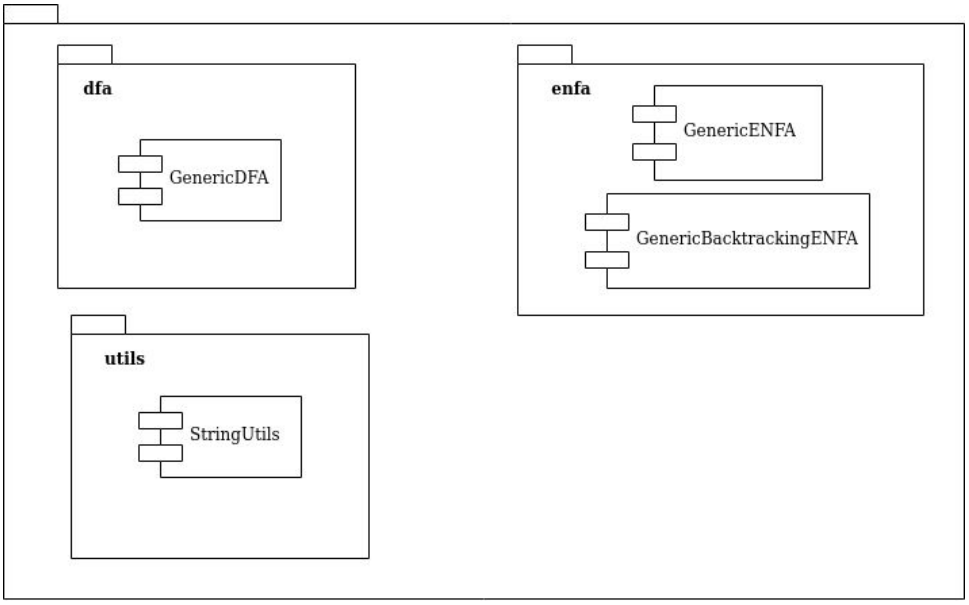
A segunda parte envolve as especificidades dos autômatos, dos quais os mais complexos foram os autômatos ENFAs, pois envolvem o não-determinismo e a possibilidade de transições espontâneas. Dessa forma, foi preciso implementar métodos para calcular o E-Fecho, que por consequência exigia um método para união de estados.

O desenvolvimento dos ENFAs foi dividido em duas classes com formas de execução distintas: a classe "GenericENFA" do pacote "enfa" realiza a verificação da árvore de derivação de forma extensiva, agrupando todos os estados possíveis na forma de um conjunto de estados. Já a classe "GenericBacktrackingENFA", também do pacote "enfa", realiza a verificação utilizando busca em profundidade, verificando todas os caminhos de derivação possíveis até alcançar a aceitação. A classe "GenericDFA" do pacote "dfa" contém o autômato determinístico que realiza as verificações de forma direta na tabela de transições.

3. Resultados e discussões

Ao final da implementação, o projeto contendo as classes ficou dividido em três pacotes que subdividem os autômatos e as classes de apoio. O pacote "utils" contém a classe "StringUtils" para manipulação de String e vetores de inteiros, o pacote "dfa" contém a classe "GenericDFA" que implementa o autômato finito determinístico e o pacote "enfa" que comporta as classes "GenericBacktrackingENFA" que implementa um autômato finito não-determinístico com transições vazias por recursão, realizando a busca em profundidade, e a classe "GenericENFA" que implementa um autômato de mesma classe que o anterior, mas realizando a busca por extensão. Os pacotes "dfa" e "enfa" possuem um arquivo de texto que possui um exemplo de configuração para as máquinas.

Figura 1. Organização dos componentes do projeto



Fonte: Autoria própria (2019)

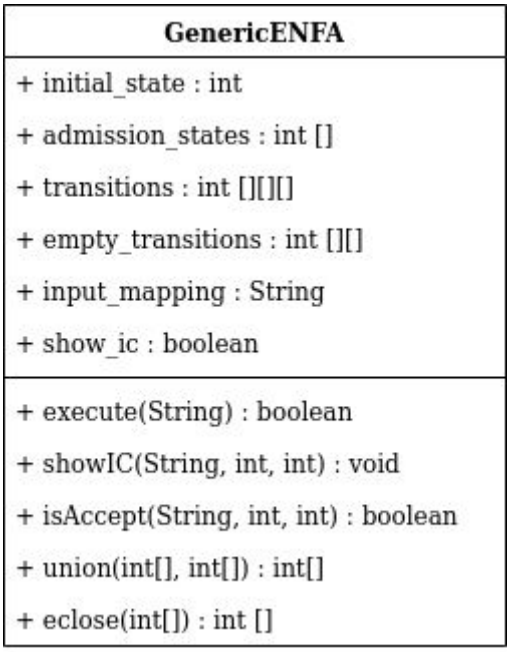
Os autômatos implementados possuem uma interface semelhante, segundo a sua classe. Como já citado, os ENFAs necessitam de métodos para lidar com o não-determinismo e com as transições vazias.

Figura 2. Interface da classe GenericDFA

GenericDFA
+ initial_state : int + admission_states : int [] + transitions : int [][] + input_mapping : String + show_ic : boolean
+ execute(String) : boolean + showIC(String, int, int) : void + isAccept(String, int, int) : boolean

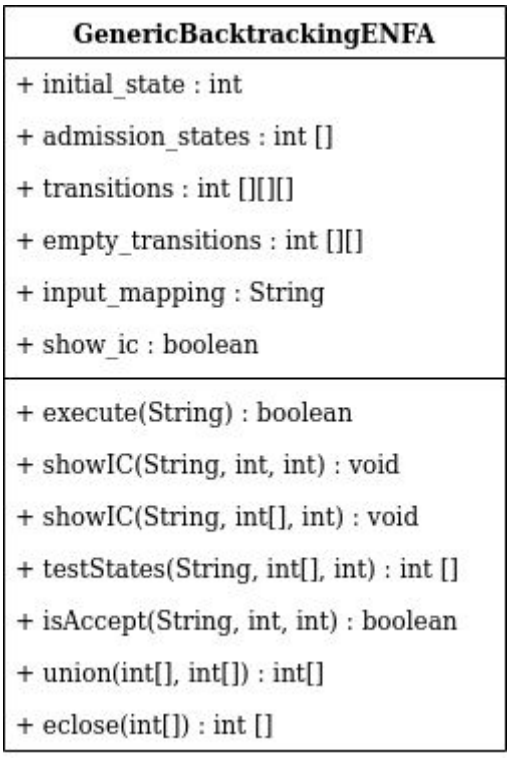
Fonte: Autoria própria (2019)

Figura 3. Interface da classe GenericENFA



Fonte: Autoria própria (2019)

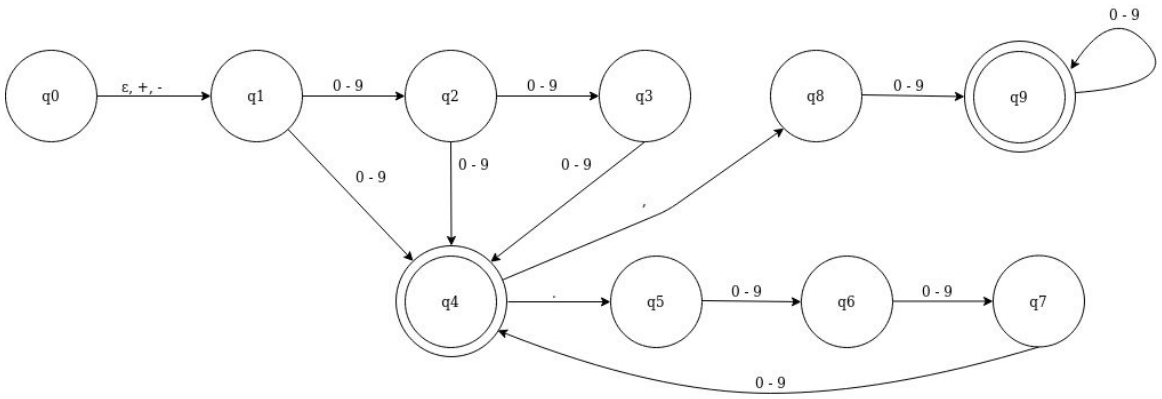
Figura 4. Interface da classe GenericBacktrackingENFA



Fonte: Autoria própria (2019)

O resultado após uma execução pode ser verificado via console dentro do sistema. Por exemplo, para o autômato ENFA denotado pelo diagrama a seguir e que reconhece a linguagem de números inteiros e decimais considerando o ponto "." a cada três casas decimais para a cadeia "134.500":

Figura 5. Diagrama de estados para o autômato que reconhece a linguagem regular dos números inteiros e decimais com "." (ponto) a cada três casas decimais.



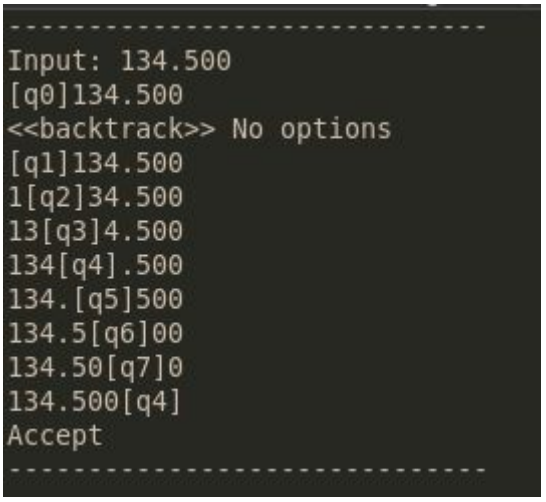
Fonte: Autoria própria (2019)

Figura 6. Modelo de arquivo de configuração para a máquina de estados anterior

```
Initial state:0
Admission states:{4,9}
Number of states:10
START
    +       -       0-9     ,       .       {}
0   {1}     {1}     {}      {}      {}      {1}
1   {}      {}      {2,4}   {}      {}      {}
2   {}      {}      {3,4}   {}      {}      {}
3   {}      {}      {4}     {}      {}      {}
4   {}      {}      {}      {8}     {5}     {}
5   {}      {}      {6}     {}      {}      {}
6   {}      {}      {7}     {}      {}      {}
7   {}      {}      {4}     {}      {}      {}
8   {}      {}      {9}     {}      {}      {}
9   {}      {}      {9}     {}      {}      {}
END
```

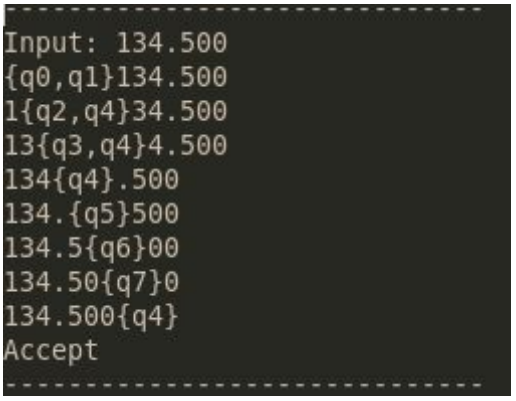
Fonte: Autoria própria (2019)

Figura 7. Saída para a cadeia “134.500” no ENFA na busca por profundidade



Fonte: Autoria própria (2019)

Figura 8. Saída para a cadeia “134.500” no ENFA na busca por extensão



Fonte: Autoria própria (2019)

Considerando a complexidade no desenvolvimento, pode-se dizer que os autômatos finitos determinísticos possuem uma implementação menos problemática, devido a sua natureza bem definida. Já os autômatos finitos não-determinísticos com transições vazias apresentam um maior grau complexidade que os anteriormente citados. É importante considerar a possibilidade de conversão entre estes autômatos, ou seja, de ENFA para DFA em alguns casos em que a natureza determinística seja conveniente graças a equivalência entre a capacidade de reconhecimento desses formalismos.

4. Considerações finais

O projeto de implementação dos autômatos de forma genérica foi desenvolvido com base na fundamentação teórica da teoria dos autômatos e linguagens formais, possuindo como principal contribuição a construção de uma percepção prática dos conceitos antes

citados, podendo ser utilizada como ferramenta pedagógica de ensino de linguagens formais e autômatos.

Para trabalhos futuros, seria interessante o desenvolvimento de uma interface gráfica para tornar ainda mais simples a utilização dessa implementação. Além disso, também é pertinente a implementação de autômatos para reconhecer outras classes de linguagens formais, como os autômatos com pilha (PDA), por exemplo, que reconhecem a classe das linguagens livres de contexto.

Referências

HOPCROFT, John E.; MOTWANI, Rajeev; ULLMAN, Jeffrey D. Introdução à Teoria dos Autômatos, Linguagens e Computação. 2ª edição. Editora Campus, 2002.

SIPSER, Michael. Introdução à Teoria dos Computação. Versão parcial. CENGAGE, 2005.

Generic Automatas. Disponível em https://github.com/mikeiasgabriel/generic_automatas. Último acesso em: 10/03/2020.