Graduação em Ciência da Computação. Disciplina: Linguagens Formais e Autômatos. Ano 2017. Prof. Walace De Almeida Rodrigues.

Algoritmos de Conversão de Autômatos

Ana Paula da Silva Cunha - Matrícula: 0011252[1]
Suena Batista Galoneti - Matrícula: 0011251[1]
[1]Instituto Federal de Minas Gerais, Formiga, MG

Resumo: Este documento apresenta o relatório do desenvolvimento do primeiro trabalho da disciplina de Linguagens Formais e Autômatos (LFA) que tem como objetivo implementar algoritmos para manipulação de autômatos finitos determinísticos, a partir de um documento XML .

Palavras-Chaves: LFA; Algoritmos.

1 Introdução

O conceito de autômato finito, o define como um conjunto finito de estados. Um AFD (Autômato Finito Determinístico) é uma Quíntupla (E, Σ, δ, i, F), tal que:

- E: um conjunto finito não vazio de estados;
- Σ: um alfabeto;
- δ: E $\times$ σ $\rightarrow$ E é a função de transição, uma função total;
- i: é o estado inicial que pertence à E;
- F: é o conjunto de estados finais.

Descrevendo seu funcionamento ou caminhamento vemos que um autômato parte de um estado inicial e caminha para um próximo estado de acordo com a função de transição e a palavra que está sendo reconhecida, e esta deve conter apenas elementos do alfabeto do autômato para ser aceita.

2 Desenvolvimento

Foi sugerido que os algoritmos para manipulação de AFDs, fossem feitos na linguagem de programação Java ou Python, deste modo foi optado pela linguagem Java.

Para a montagem dos autômatos e saída gráfica foi utilizado o JFlap. Este programa tem como extensão um arquivo .jff, que ao abri-lo em formato texto, tem formato XML, sendo assim uma maneira fácil de manipula-lo.

Inicialmente foram criadas três classes nomeadas como:

- TuplaEstados;
- FuncaoTransicao;
- AFD.

Estas funções serão descritas nas subseções a seguir.

2.1 Classe: *TuplaEstados*

Classe feita para manipular estados 'visitados' e à 'visitar' em uma multiplicação de dois autômatos determinísticos. Ao criar um objeto do tipo *TuplaEstados*, seu conteúdo seria estado do primeiro automato e do segundo (afd1E, afd2E). Seus métodos são básicos para manipulação de um objeto, *Geters*, *Seters* e *Override* do método *toString()*.

2.2 Classe: *FuncaoTransicao*

A classe *FuncaoTransicao*, foi feita para manipular cada função de transição do autômato. O objeto do tipo *FuncaoTransicao* é constituído de: o estado de partida do autômato, o estado de chegada do mesmo autômato e uma letra do alfabeto deste (de, para, caracter).

2.3 Classe: *AFD*

Dentro da classe *AFD* foi implementado o método para leitura do arquivo XML, montando um objeto do tipo *AFD*, tendo os seguintes parâmetros para sua construção: *ArrayList* do tipo *Integer* para o conjunto de estados e conjunto de estados finais do autômato, *ArrayList* do tipo *Character* para o alfabeto do AFD, *ArrayList* do tipo *FuncaoTransicao* para todas as funções de transições do autômato, e o estado inicial do tipo *int*.

Outro método presente nesta classe é para a obtenção do complemento, aonde verifica quais estados são finais e quais não são, e transforma os estados não finais nos novos estados finais e os finais anteriores tornando-os não finais.

Para a implementação da união, interseção e diferença de autômatos, fez necessário a implementação do método *multiplicacao*. Este método consistiu em unir os alfabetos dos dois autômatos em um novo alfabeto e fazer um novo caminhamento verificando os nós "visitados"e os nós à "visitar"(um *ArrayList* do tipo *TuplaEstados* foi feito para cada um), utilizando a função transição obtendo novos estados para o autômato final (o autômato final terá seus estados finais definidos de acordo com a operação escolhida).

A partir da lista de "visitados"obtida na multiplicação, os novos estados serão renomeados e os finais serão marcados de acordo com a operação:

- União: os estados marcados como finais serão as combinações dos estados finais originais, em outras palavras, os estados novos, obtidos numa lista de "visitados", começados com o estado final do primeiro autômato e terminados com o estado final do segundo autômato;
- Interseção: serão finais apenas os novos estados que forem compostos exclusivamente por estados finais dos autômatos originais;
- Diferença: nada mais é do que a interseção do primeiro automato com o complemento do segundo, ou seja, ela irá aceitar como estados finais novos apenas os estados começados pelo estados finais do primeiro autômato e terminado pelos estados não finais do segundo autômato.

Foi implementado um método para cada item anteriormente descrito.

Na implementação ainda encontram-se outros pequenos métodos com sub tarefas utilizadas para a implementação das operações do trabalho, além das requeridas na especificação do trabalho, como:

- existeNaLista: verifica se existe um objeto em uma lista do tipo *TuplaEstados*
- getIndex: pega o índice do objeto de uma lista do tipo *TuplaEstados*

3 Conclusão

Algumas das funções que foram requeridas na especificação, não foram implementadas, tais como, equivalencia de estados, equivalencia de automatos, minimização.

Referências

[Ferreira]FERREIRA, B. *Introdução a linguagem Java.*

[Vieira e Barbosa]VIEIRA, N. J.; BARBOSA, I. G. *Máquinas de Estados Finitos.*