ICMC-USP

Trabalho 1

SCC-0205

2º. Semestre de 2020

Vitor Augusto de Oliveira - 9360815

Descrição

O trabalho consistia na implementação de um simulador de autômatos finitos que, através de um input padronizado fornecido pelo usuário, elaborasse um autômato finito que seguisse as especificações fornecidas e fosse capaz de validar um conjunto de cadeias.

Foram empregados conhecimentos teóricos desenvolvidos na disciplina durante o semestre e de modelagem e algoritmos em grafos.

Modelagem

Conforme citado na seção anterior, o problema foi modelado como um grafo. Um autômato finito (AFD ou AFN) possui como atributos um conjunto de vértices e arestas que representam, respectivamente, os estados e transições.

Desta forma, a validação de uma cadeia C se dá percorrendo, a partir do(s) estado(s) inicial(is), todos os possíveis caminhos onde o valor da transição entre o estado atual S_i e um estado S_{i+1} seja equivalente ao caractere c_i de C. Este processo se repete até atingir a configuração onde não há caracteres restantes ou se não há ramificações possíveis a partir do estado atual. Quando a cadeia se esgota, é averiguado se o estado atual pertence ao conjunto de estados de aceitação F. Em caso positivo, aceita-se a cadeia, e rejeita-se caso contrário.

Implementação

O problema foi implementado na linguagem C++ com o objetivo de se usufruir das vantagens que a mesma fornece, principalmente da Standard Template Library (STL).

Um autômato finito é representado pela classe ***AutomatoFinito*** que contém um vetor de ***State*** (estado), o número de estados, vetores de inteiros que armazenavam os estados de aceitação e iniciais, e uma string que representa os caracteres terminais.

A classe ***State*** tem como atributos: um vetor da classe ***Transition*** e uma flag que armazena a informação se o estado é terminal ou não. A classe ***Transition*** contém dois inteiros que representam os endpoints de uma aresta (mapeados para corresponderem ao id dos vértices) e o caractere que os une.

Solução

A solução resume-se ao algoritmo de busca DFS (Depth First Search) que é invocada a partir de todos os estados iniciais (se AFN) ou do estado 0 (se AFD).

Para isso, a partir do estado atual, sempre que houver uma ramificação onde o caractere atual for equivalente ao caractere de aresta no conjunto de transições, é invocada recursivamente a função DFS, incrementando-se um indexador que percorre a string da cadeia e atualizando o estado atual. A função também é chamada quando λ é o caractere de uma das arestas, porém não incrementa-se o indexador pois arcos λ não consomem a cadeia. No caso onde não há match, simplesmente retorna-se a execução anterior.

O critério de parada da função consiste no caso onde o indexador tem valor maior ou igual ao tamanho da string da cadeia. Em seguida, verifica-se se o atributo ***isFinal*** do estado atual, que indica se o estado pertence ao conjunto de aceitação, é verdadeiro. Por fim, retorna-se este valor.

Eficiência

Por ser representado por um vetor de estados, a solução tem complexidade de espaço $O(|Q|)$, onde Q é o conjunto de estados.

A validação da cadeia tem complexidade de tempo $O(|Q| + |\delta|)$ onde δ é o conjunto de todas as transições no autômato finito.