AFE
RELATÓRIO

Luiz Eduardo Pereira – 0021619

Marcos Martins Vieira – 0026756

Instituto Federal de Minas Gerais, Formiga, MG.

INTRODUÇÃO

O problema envolve a representação de AFs sem pilha implementado na linguagem Java, utilizando o Eclipse neon.3 versão 4.6.3 do Windows. Foi utilizado o programa JFLAP 7.0, sendo que os autômatos são salvos em extensão “.jff”.

Este programa realiza conversão de autômato, e para AFD’s realiza minimização, verificação de equivalência e operações.

IMPLEMENTAÇÃO

O código foi dividido nas seguintes classes:

- Estado
- Transicao
- AFE, sendo que esta classe possui uma lista de estados, de transições, de estados iniciais, estados finais, alfabeto e todas as funcionalidades requeridas na especificação do trabalho.

Foi utilizado a biblioteca JDOM para fazer a leitura/escrita dos arquivos .jff.

A estrutura utilizada para a representação dos dados foi o “List”. Mas para alguns casos em que havia a necessidade de ordenação e não repetição, foi utilizado o “TreeSet”.

A seguir é apresentado as funções implementadas e suas funcionalidades:

- add/delete State/Transition: Funções disponibilizadas para fazer a alteração do AFE.
- deletaEstadoInutil: Recebe um automato e remove os estados que não se conectam com o conjunto no qual o estado inicial faz parte. Utilizado em minimização.
- existe Estado/NomeEstado/Transicao: Verifica a existência do estado ou transição no automato, retorna um boolean.
- verifica Inicial/Final: Verifica se o estado passado faz parte dos estados iniciais ou estados finais.
- accept: Recebe uma palavra e retorna se o automato aceita ou não.
- move: Variação do accept, recebendo o estado de origem e a palavra (Usado internamente.

- MoveSimbolo: Recebe um estado de origem e um símbolo, retorna o estado de destino para qual este símbolo foi.
- save/load: Salva e carrega o automato escolhido. Foi utilizado a biblioteca JDOM para essas funções.
- maiorID: Usado na minimização. Retorna o id do estado com maior id.
- idEquivalenteOriginal: Usado na minimização. Recebe o nome do estado no automato minimizado e retorna o Id no automato normal.
- idEquivalenteMinimizado: Usado na minimização. Recebe o Id do automato normal e retorna o Id no automato minimizado.
- concatenaNome: Usado na conversão de AFN para AFD. Recebe um "TreeSet" de Id de estados, e o transforma em um único nome, para a criação de um novo estado.
- desconcatenaNome: Usado na conversão de AFN para AFD. Recebe um nome e o transforma em um "TreeSet" com os Id dos estados que compõem aquele nome.
- verificaFinalConjunto: Usado na conversão de AFN para AFD. Verifica se algum estado do conjunto é estado final, se for, todo aquele conjunto também será final.
- copy: Faz a cópia do automato alvo.

Para a minimização, o automato de entrada é transformado em automato com estado de erro explícito automaticamente.

As outras funções foram implementadas de acordo com a especificação. Os algoritmos implementados foram realizados da seguinte maneira. Foi gerado um automato pequeno e simples e anotado o passo a passo para fazer as operações/conversões, e seguindo esses passos o algoritmo foi implementado. Depois, foi feito um automato maior, para corrigir as falhas e particularidades encontradas.

VALIDAÇÃO

Para a validação do programa foram gerados autômatos no JFLAP e em seguida executados. Alguns casos foram comparados com o resultado do JFLAP, outros foram comparados com autômatos resolvidos a mão.

CONCLUSÃO

Este é um trabalho em que no início foi muito difícil de se fazer, mas a medida que o desenvolvimento evoluiu, foi ficando mais fácil de se desenvolver. Os algoritmos de conversão foram feitos usando os mesmos métodos utilizados em sala de aula, é até mesmo possível ver a separação de cada etapa no código.

REFERENCIA

Implementando Autômatos Finitos em Java – Paulo Kinjo
http://www.academia.edu/2543590/Implementando_Automatos_Finitos_em_Java