(PROJETO PRÁTICO)
CONSTRUÇÃO DE APLICAÇÃO PARA GERAÇÃO DE AUTÔMATOS FINITOS DETERMINÍSTICOS.

Curso de Ciência da Computação
Andrew Gabriel Gomes, Igor Radtke.
andrew.gabrielgomes@gmail.com | igor.radtke@hotmail.com

Abstract. *In this report, the presentation and implementation of the practical project "Building applications for the generation of deterministic finite automata" for the subject of Formal Languages and Automata is presented in a succinct and well-crafted way. The general objective of this project is the validation of knowledge from the study carried out during the academic semester.*

Resumo. *Neste relatório é apresentado de forma sucinta e bem elaborada, a apresentação e implementação do projeto prático "Construção de aplicações para geração de autômatos finitos determinísticos" para a matéria de Linguagens Formais e Autômatos. O seu desenvolvimento partiu da linguagem de programação python 3. O objetivo geral deste projeto é a validação do conhecimento a partir do estudo realizado durante o semestre letivo.*

1. Introdução

A proposta deste relatório, através de uma implementação via código, é explicar o processo de um gerador de autômato finito determinístico (AFD), livre de épsilon transição, porém sem a utilização de classes de equivalência. foi implementado um programa gerador e determinador de autômatos finitos, onde, a partir da leitura de um arquivo fonte, que contempla gramáticas regulares, palavras reservadas, símbolos e operadores, será gerado o respectivo autômato finito.

Será apresentado também, o referencial teórico, para o entendimento completo da execução e resultados da aplicação, bem como todo o processo de desenvolvimento da mesma.

2. Referencial Teórico

Para melhor compreender esse projeto, temos como conhecimento, de que um autômato finito é como uma máquina de estados que atua como um reconhecedor de padrões de uma determinada linguagem, sendo bastante utilizado em processamento de texto e na etapa de análise léxica de compiladores, para verificar se uma sentença pertence ou não a essa linguagem.

Um Autômato Finito possui um número finito e predefinido de estados, constitui de um modelo computacional sequencial, e o próximo passo somente é executado após o fim do passo anterior. Segundo Menezes, um Autômato Finito, além disso, é uma quíntupla composta por um alfabeto, um conjunto dos possíveis estados, função de transição, um estado inicial e um conjunto de estados finais que pertencem ao conjunto de estados do autômato.

Autômatos finitos são categorizados como determinísticos ou não-determinísticos. Os AFD têm a característica de que, dado um estado atual "q" e um caractere de entrada "x", tem-se no máximo um estado possível para transição, ou seja, para cada cadeia de caracteres há um único caminho a ser seguido no autômato. Nos AFND pode-se ter funções de transição do tipo δ(q,x) = {a,b}, onde a e b são dois estados possíveis de se chegar através do estado q, por meio da entrada do caractere x.

Uma máquina de estados finita não-determinística ou um autômato finito não-determinístico (AFND) é uma máquina de estados finita onde para cada par de estado e símbolo de entrada pode haver vários próximos estados possíveis. Isso o distingue do autômato finito determinístico (AFD), onde o próximo estado possível é univocamente determinado.

Muitas vezes, nem todos os estados de um autômato são úteis ou alcançáveis, então outro procedimento adotado para a otimização do mesmo é a eliminação dos estados inúteis e inalcançáveis.

3. Implementação

A seguir será demonstrado de forma a entender como foram realizados todos os passos executados na implementação do programa. Identificação de tokens, carregamento da gramática, construção do autômato finito não determinístico (AFND), geração do autômato finito determinístico (AFD) e os respectivos resultados gerados.

Como primeira parte da aplicação do projeto, foi realizado o carregamento do conjunto de tokens e de uma gramática regular, como um arquivo de entrada separadamente. A partir dos dados contidos neste arquivo, o autômato deverá reconhecer as gramáticas, palavras e símbolos que compõem a linguagem.

O programa inicia a leitura de cada linha do arquivo de entrada e faz a separação dos dados a partir dos espaços, reconhecendo os tokens e gramáticas. Com esses dados foi se dando continuidade na aplicação.

A partir da identificação das gramáticas geradas durante o carregamento dos dados de entrada, será construído o autômato finito. Durante o processo de construção do autômato, pode ocorrer de em cada par de estado e símbolo haver uma transição para mais de um estado, assim, gerando uma indeterminação. Esse é o conceito que define a classe de autômatos conhecida como autômatos finitos não-determinísticos (AFND).

O algoritmo para a construção do autômato finito é construído a partir de vetores. Onde é percorrido e analisado cada etapa para que assim seja construída conforme suas regras.

Abaixo seguem as chamadas das funções responsáveis pelo tratamento das informações de entrada.

```
allData = getData(openFile(getParam(1)))  # Pega os dados do arquivo de entrada e coloca em um vetor
tokens = getTokens(allData)               # Pega os vetor gerado e coloca os tokens em um vetor
grs = getGR(allData)                      # Pega os vetor gerado e coloca as regras em um vetor
tokensFromGRS = getGRStokens(grs)         # Pega os tokens presentes nas regras e coloca em um vetor
allTokens = removeBlankSpace(tokens) + removeBlankSpace(tokensFromGRS) # Faz uma junção dos tokens (das regras e os fornecidos na entrada)
allTokens = sorted(set(allTokens))        # Remove as duplicatas e ordena
parsedGRs = parseGR(grs)                  # Pega o vetor gerado e coloca as regras parseadas em um vetor
allTokens = removeEpsilon(allTokens)      # Remove o epsilon das gramáticas
```

Chamadas de funções para a construção do autômato finito não determinístico:

```
afnd = createTable(allTokens, parsedGRs)    # cria a tabela (dataframe) para o AFND
afnd = afnd[allTokens].astype(str)          # altera o tipo de dados para string
afnd = fillWithGRs(afnd, parsedGRs)         # preenche a tabela com as regras
afnd = removeNan(afnd)                      # remove os 'nan'
print(afnd)                                 # printa a tabela
```

Uma forma de contornar a implementação de alto grau de complexidade de um AFND, é a utilização do algoritmo de determinização nesse autômato, para transformá- lo em um autômato finito determinístico (AFD).

Chamadas de funções para a construção do autômato finito determinístico:

```
afd= determiniza(afnd, allTokens)          # determiniza o afnd
afd = afd[allTokens].astype(str)           # altera o tipo de dados para string
afd = removeNan(afd)                       # remove os 'nan'
afd = ordenaTable(afd)                     # ordena a tabela e normaliza
```

```
while temRegraNova(afd):                   # função que verifica se há novas regras geradas a partir das regras já determinizadas
    afd = determiniza(afd, allTokens)      # determiniza o afnd
    afd = ordenaTable(afd)                 # ordena a tabela e normaliza
    print(afd)                             # printa a tabela a cada etapa
    print("\nAFD\n")
```

Ao final da aplicação são gerados dois arquivos .csv, um contendo o AFND e o outro o AFD.

4. Conclusão

Compreendemos que para a realização do projeto, foi preciso ter domínio sobre os princípios e conceitos de linguagens, gramáticas, símbolos, sentenças e autômatos finitos. A partir dele tiramos como conclusão um entendimento aprofundado sobre o conteúdo repassado em aula. O programa implementado é capaz de construir um autômato finito a partir de um conjunto de regras de gramática, que vem do arquivo de entrada, atendendo a proposta de trabalho repassada.

Referência

MENEZES, P. Linguagens formais e autômatos. Série livros didáticos. Bookman, 2008

MENEZES, P. B. Linguagens Formais e Autômatos. Ed. 6. Editora Bookman, 2011.