Nome **: Angemydelson Saint-Bert**
Matrícula **: 2121101002**
Prof **: Braulio Adriano de Mello**
Disciplina **: Linguagens Formais e Autômatos**

Artigo sobre a construção de uma aplicação para gerar Autômato Finito Determinístico.

Objetivo:

O objetivo deste trabalho é criar e manipular autômatos finitos determinísticos (AFD) a partir de uma descrição textual da gramática regular. Ele realiza algumas etapas principais:

Carga do autômato finito a partir do arquivo de entrada, que contém informações sobre os tokens da linguagem e as regras da gramática regular.
Eliminação de transições vazias (épsilon-transições).
Determinização do autômato, convertendo-o de um autômato finito não determinístico (AFN) para um autômato finito determinístico (AFD).
Remoção de estados inalcançáveis e estados mortos do autômato determinizado.
Impressão do autômato resultante após todas as etapas de manipulação.

Descrição:

O código é escrito em Python. O arquivo de entrada contém as regras da gramática regular e é utilizado para construir o autômato finito. A classe principal do código é a classe Autômato, que representa o autômato e possui os seguintes atributos:

- Estados: um dicionário que armazena os estados do autômato e suas transições;
- Alfabeto: um conjunto que contém os símbolos do alfabeto da linguagem;
- Finais: um conjunto que armazena os estados finais do autômato;
- Texto: uma string que contém o texto de entrada do arquivo;
- NovosEstados: um dicionário utilizado para identificar a origem das novas produções criadas na determinização;

- TransicoesVisitadas: uma lista que indica quais transições já foram visitadas na busca em profundidade da remoção de estados inúteis;
- AutomatoMinimizado: um dicionário que armazena o autômato após a etapa de minimização.

O código também possui as classes EpsilonTransicao, Determinizacao, Inalcancaveis e Mortos, que herdam da classe Automato. Cada uma dessas classes é responsável por realizar uma etapa específica da manipulação do autômato.

Determinização:

A determinização é a etapa que transforma um autômato finito não determinístico (AFN) em um autômato finito determinístico (AFD). O AFN pode ter várias transições para um mesmo símbolo em um estado, enquanto o AFD deve ser determinístico, ou seja, cada símbolo de entrada deve ter uma única transição possível em cada estado. Para realizar a determinização, o código percorre os estados do autómato e trata os casos de indeterminismo, criando novos estados e transições, conforme necessário.

Eliminação de épsilon-transições:

As épsilon-transições são transições vazias, que permitem que o autômato passe de um estado para outro sem consumir nenhum símbolo de entrada. A eliminação de épsilon-transições é feita pela classe EpsilonTransicao, que busca e trata as transições vazias no autômato, criando novas produções e estados quando necessário.

Remoção de estados inalcançáveis:

Os estados inalcançáveis são aqueles que não podem ser alcançados a partir do estado inicial do autômato. A classe Inalcancaveis é responsável por remover esses estados, fazendo uma busca em profundidade a partir do estado inicial e marcando os estados que são alcançáveis. Em seguida, os estados inalcançáveis são removidos do autômato.

Remoção de estados mortos:

Os estados mortos são aqueles que não levam a nenhum estado final do autômato. A classe Mortos é responsável por remover esses estados, fazendo uma busca em profundidade a partir dos estados finais e marcando os estados

que levam a um estado final. Em seguida, os estados mortos são removidos do autômato.

Estado de erro:

O autômato pode ter um estado de erro que representa uma transição não definida para um determinado símbolo de entrada. Essa transição leva o autômato a um estado de erro em caso de entrada inválida. O estado de erro é representado no código pelo valor -1.

Impressão dos autômatos

Entrada

```
se
entao
senao
<S> ::= a<A> | e<A> | i<A> | o<A> | u<A>
<A> ::= a<A> | e<A> | i<A> | o<A> | u<A> | #
```

Exemplo de saída incompleto AFND

```
angemydelson@angemydelson-IPMH310G:~/Github/Semestre2023.1/
-==========- O autômato lido: -==========-
 0 = s [1, 8], e [3], < [13, 53],
 1 = e [2],
*2 =
 3 = n [4],
 4 = t [5],
 5 = a [6],
 6 = o [7],
*7 =
 8 = e [9],
 9 = n [10],
 10 = a [11],
 11 = o [12],
*12 =
```

Explicação

1- A gramática consiste em duas regras de produção: uma para o símbolo inicial <S> e outra para o símbolo não terminal <A>. Cada regra de produção possui várias opções, separadas pelo símbolo |.

2- Saída: A saída representa o AFND construído a partir da gramática. Cada estado é representado por um número, e as transições de entrada são especificadas ao lado de cada estado. Algumas transições possuem múltiplos destinos, indicados entre colchetes e separados por vírgulas.
Por exemplo, vejamos o primeiro estado:
Isso significa que, a partir do estado 0, podemos fazer transições para os estados 1 e 8 se a entrada for 's', para o estado 3 se a entrada for 'e' e para os estados 13 e 53 se a entrada for '<'.
Os estados marcados com asterisco (*) representam os estados finais do autômato. No caso, os estados finais são 2, 7 e 12.

Referências:

- **Apostila do professor.**
- **Anotações do professor.**