Trabalho Prático I - Teoria de Linguagens

Implementação do Trabalho Prático I da disciplana Teoria de Linguagens ministrada pela professora Dra. Carolina Ribeiro Xavier. Disciplina do Programa de Pós-Graduação em Ciência da Computação (PPGCC) da Universidade Federal de São João del-Rei (UFSJ).

Aluno: Luan Luiz Gonçalves

Data: Março de 2020

Descrição do Trabalho

Este trabalho prático (TP) deve ser realizado individualmente, e consiste na implementação de um programa e a redação de um relatório.

Proposta de TP

O TP proposto consiste em implementar um autômato finito determinístico para a sua linguagem. Conforme mencionado, os detalhes de implementação devem ser descritos no relatório que deve acompanhar o programa.

O relatório deve constar ainda de detalhes sobre cada formalismo usado para as linguagens regulares, a saber: AFD, AFN, AFN-e, ER e gramáticas lineares.

Sua linguagem

Você poderá conhecer sua linguagem de acordo a expressão regular que segue, para chegar ao AFD a ser implementado você seguirá a seguinte sequência:

```
ER -> AFN-e -> AFN -> AFD
```

Dado que o sua matrícula é `d1d2d3d4d5d6d7d8d9` e as três primeiras letras de seu nome são `111213`, sua ER é $x_1(d_2l_1 + d_9l_2)^+x_2$. Sendo o `x` da forma:

1. `x1` é o número de letras de seu primeiro nome;
2. `x2` é o número de letras de seu segundo nome.

Sobre o AFD

O AFD é um formalismo reconhecedor de linguagens regulares, neste TP iremos implementar uma linguagem diferente para cada aluno.

Objetivos da implementação

A implementação deve ser capaz de receber uma entrada de duas formas alternativas: individualmente (via terminal ou em interface à sua escolha) ou um conjunto de entradas, via arquivo texto (uma palavra por linha). E responder, para cada entrada dada, se a palavra foi aceita ou se foi rejeitada, em caso de rejeição, colocar também o motivo (indefinição ou fim da leitura em estado não-final).

Para leitura individual é interessante explicitar os passos do autômato na leitura da cada símbolo.

> Você deve entregar junto ao seu código e relatório um arquivo com 10 palavras aceitas pela sua linguagem, 10 paralavras rejeitadas (5 por indefinição e 5 que após toda palavra lida, assume um estado não final).

Relatório das características das linguagens do tipo 3, da Implementação e da Utilização

O relatório deve constar de uma breve exposição do assunto visto até agora na disciplina de teoria de linguagens, fale sobre os formalismos e as ferramentas vistas até agora de uma forma clara e resumida, explicando claramente como você chegou no AFD implementado.

O relatório também deve descrever sucintamente a estrutura do programa e descrição dos algoritmos utilizados. Além disso, também deve ser fornecido um manual de utilização do código. Tal relatório deve ser entregue em formato eletrônico juntamente com o programa.

O relatório deve incluir uma descrição das dificuldades encontradas para implementar o TP.

> Somente arquivos no formato PDF serão considerados.

Prazos

A entrega do TP pode ser efetuada até o dia 29/03/2021 às 23:59.

Submissão

Os TPs devem ser enviados pelo Portal didático/UFSJ, alternativamente, em caso de instabilidade do sistema, utilizar o endereço carolinaxavier@ufsj.edu.br.

Itens necessários:

1. Código fonte com documentação;
2. Arquivo com as palavras;
3. Relatório.

Critérios de avalilação

Este trabalho vale 30% da nota da disciplina.

- 50% implementação correta do código do autômato e corretude nos padrões de entrada e saída;
- 10% para os aspectos de boas práticas de programação:

    - Modularização;
    - Legibilidade (nomes de variáveis significativos, código bem formatado, uso de comentários);
    - Consistência (formatação uniforme);
    - Boa documentação descritiva da implementação e do uso do programa;
- 30% Texto resumindo o tema visto em aula;
- 10% áudio ou vídeo explicativo da linguagem de até 5 minutos;
- (+5% das Notas acima somadas) Extras para uma interface amigável.

Algumas dicas

Durante o desenvolvimento, é importante não se perder nos detalhes. Portanto, é recomendado que os alunos comecem o desenvolvimento implementando as funcionalidades básicas. Só depois de garantir que as funcionalidades básicas estão funcionando conforme o planejado, os alunos devem considerar a implementação de melhorias e funcionalidades adicionais, como intereface gráfica, por exemplo. Também recomenda-se que trechos mais complicados do código sejam acompanhados de comentários que esclareçam o seu funcionamento/objetivo/parâmetros de entrada e resultados.

Instruções

O código foi implementado na linguagem de programação Python 3 e testado apenas no sistema operacional Linux -- distribuição Ubuntu 20.04.2 LTS.

Executando

Acesse via termial a pasta raiz:

```
cd linguagens-regulares
```

A entrada pode ser passada por argumento, com a opção de informar uma única entrada:

```
python3 linguagens-regulares.py -p <palavra>
```

ou conjunto de entradas definidas em um arquivo de texto:

```
python3 linguagens-regulares.py -a <nome do arquivo>
```

> Os argumentos são opcionais. Caso não seja utilizado, o usuário deverá inserir uma única palavra em tempo de execução.

Ajuda

Também é possível exigir o menu de ajuda utilizando o comanda:

```
linguagens-regulares.py -help
```

Menu de ajuda:

```
Comando: linguagens-regulares.py [-h] [-p PALAVRA] [-a ARQUIVO]

Argumentos opcionais:

  -h, --help            Exibe essa mensagem de ajuda e sai.
  -p PALAVRA, --palavra PALAVRA
                        Entrar com a palavra.
  -a ARQUIVO, --arquivo ARQUIVO
                        Ler arquivo com conjunto de entradas.
```

Exemplo de arquivo de entrada

Na pasta raiz do programa encontra um arquivo de entrada chamado `entrada.txt`. Conteúdo do arquivo:

```
42l4
45u4
42l5u4
45u2l5u4
42l2l2l2l5u5u2l5u5u4
45u2l5u5u5u4
42l2l2l2l2l4
45u5u5u5u5u4
42l2l2l2l2l2l2l2l2l2l2l5u5u5u5u5u5u5u5u4
45u5u5u5u5u5u5u5u5u5u5u5u5u5u5u5u5u2l4
42l42
4u54
425u4
45l2l5u4
42l2l2l2lu5u2l5u5u4
45u2l5u
42l2l2
45u5u
42l2l2l2l2l2l2l2l2l2l2l5u5u5u5
45u5u5u5u5u5u5u5u5u5u5u5u5u5u5u5u2l
```

Como podemos observar, o arquivo de entrada contém as palavras separadas por linhas (caractere separador: `\n`).

Exemplo de execução:

Processando as palavras do arquivo de entrada (`entrada.txt`):

```
cd linguagens-regulares
python3 linguagens-regulares.py -a entrada.txt
```

Resultado:

```
Palavra "42l4" ==> ACEITA!
Palavra "45u4" ==> ACEITA!
Palavra "42l5u4" ==> ACEITA!
Palavra "45u2l5u4" ==> ACEITA!
Palavra "42l2l2l2l5u5u2l5u5u4" ==> ACEITA!
Palavra "45u2l5u5u5u4" ==> ACEITA!
Palavra "42l2l2l2l2l4" ==> ACEITA!
Palavra "45u5u5u5u5u4" ==> ACEITA!
Palavra "42l2l2l2l2l2l2l2l2l2l2l5u5u5u5u5u5u5u5u4" ==> ACEITA!
Palavra "45u5u5u5u5u5u5u5u5u5u5u5u5u5u5u5u5u2l4" ==> ACEITA!
Palavra "42l42" ==> REJEITADA por indefinição!
Palavra "4u54" ==> REJEITADA por indefinição!
Palavra "425u4" ==> REJEITADA por indefinição!
Palavra "45l2l5u4" ==> REJEITADA por indefinição!
Palavra "42l2l2l2lu5u2l5u5u4" ==> REJEITADA por indefinição!
Palavra "45u2l5u" ==> REJEITADA (fim da leitura em estado não-final!)
Palavra "42l2l2" ==> REJEITADA (fim da leitura em estado não-final!)
Palavra "45u5u" ==> REJEITADA (fim da leitura em estado não-final!)
Palavra "42l2l2l2l2l2l2l2l2l2l2l5u5u5u5" ==> REJEITADA (fim da leitura em estado não-final!)
Palavra "45u5u5u5u5u5u5u5u5u5u5u5u5u5u5u5u2l" ==> REJEITADA (fim da leitura em estado não-final!)
```

Detalhes de implementação

Foi implementado a classe AFD como abstração do AFD.

Classe AFD

A classe AFD representa um Autômato Finito Deterministico.

Atributos

Lista de atributos da classe AFD

- __afd:

```
# Abstração de um AFD
__afd = {}
```

  Este atributo utiliza um dicionário python, sendo que:

  - O indice é o nome do estado;
  - O valor é um dicionário, onde:
    - O índice é o símbolo lido;
    - O valor é o nome do próximo estado (destino).

- __estadoInicial:

```
# Guarda o nome do estado inicial
__estadoInicial = None
```

  Este atributo recebe uma string com o nome do estado inicial.

Métodos

Lista de métodos da classe AFD

- add_estado:

```
add_estado(nome, inicial=False, final=False)
  Define um novo estado

  Parâmetros
  ----------
  nome : str, opcional
      Nome do estado

  inicial : boolean, opcional
      Define se o estado é inicial ou não

  final : boolean, opcional
      Define se o estado é finão ou não
```

  O estado é a dicionado no atributo `__afd`, com o dicionário de transições vazio.

- add_transicao:

```
add_transicao(origem, simbolo, destino)
  Define uma nova transição

  Parâmetros
  ----------
  origem : str
      Estado de origem

  simbolo : str
      Símbolo lido

  destino : str
      Estado de destino
```

  Adiciona a transição no dicionario de transições, de um estado já existente.

- executar:

```
executar(palavra)
  Processa a palavra

  Parâmetro
  ----------
  palavra : str
      Palavra a ser processada

  Retorno
  -------
  mensagem : str
      Retorna uma mensagem de aceitação ou rejeição da palavra
```

Para cada caractere da palavra lido: é consultado, no dicionário de transições, se a trasição foi definida. Se sim, é obtido o nome do novo estado (o estado atual passa a ser o estado novo).

Exemplo de utilização da classe AFD

Código:

```
 # Instância do AFD
afd = AFD()

# Define os estados
afd.add_estado('q0', True)  # Estado inicial
afd.add_estado('q1')
afd.add_estado('q2')
afd.add_estado('q3')
afd.add_estado('q4')
afd.add_estado('q5')
afd.add_estado('q6', False, True)  # Estado final

# Define as transições
afd.add_transicao('q0', '4', 'q1')
afd.add_transicao('q1', '2', 'q2')
afd.add_transicao('q1', '5', 'q3')
afd.add_transicao('q2', 'l', 'q4')
afd.add_transicao('q3', 'u', 'q5')
afd.add_transicao('q4', '2', 'q2')
afd.add_transicao('q4', '5', 'q3')
afd.add_transicao('q4', '4', 'q6')
afd.add_transicao('q5', '5', 'q3')
afd.add_transicao('q5', '2', 'q2')
afd.add_transicao('q5', '4', 'q6')

# Processa a palavra
afd.executar('42l5u4')
```