

ACH2043 - Introdução à Teoria da Computação

Exercício de Programação: Emulação de Autômato Finito Não Determinístico

Objetivo:

Desenvolver um programa emulador de autômatos finitos não-determinísticos (AFNs).

Seu programa deve receber um único arquivo-texto contendo as especificações de m AFNs

M_1, M_2, \dots, M_m e, para cada autômato M_i , um conjunto de cadeias de teste $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n$.

O programa deverá executar os autômatos M_1, M_2, \dots, M_m sobre suas respectivas cadeias de teste e devolver um arquivo-texto de valores 0/1, indicando quais cadeias são aceitas (1) ou rejeitadas (0) pelo seu autômato correspondente.

Em outras palavras, seu programa deverá decidir se cada cadeia de teste \mathbf{w}_i pertence ou não à linguagem reconhecida pelo respectivo autômato.

O programa deverá ser desenvolvido em C, Java, R ou Python, e deverá ser executável via linha de comando do DOS ou Linux.

Chamada do programa e especificações dos arquivos

A chamada do programa será:

Implementação em C: afn.exe

Implementação em Java: java afn

Implementação em Python: python afn.py

Implementação em R: Rscript afn.r

O programa receberá **um arquivo de entrada** denominado `entrada.txt` e retornará **um arquivo de saída** denominado `saida.txt`. Esses dois nomes deverão ser *hard-coded*, e deverá ser assumido que ambos serão lidos e escritos no mesmo diretório do script ou módulo executável do programa.

Arquivo `entrada.txt`:

Este arquivo conterá as especificações dos AFNs e suas respectivas cadeias de testes.

O arquivo será organizado em blocos de linhas, da seguinte maneira:

m (número de autômatos de teste)

Bloco de linhas com a especificação do autômato 1

Bloco de linhas com as cadeias de teste do autômato 1

Bloco de linhas com a especificação do autômato 2

Bloco de linhas com as cadeias de teste do autômato 2

...

Bloco de linhas com a especificação do autômato m

Bloco de linhas com as cadeias de teste do autômato m

Cada bloco de linhas com a especificação de um autômato será organizado da seguinte forma:

- A primeira linha será um cabeçalho contendo seguintes campos separados por espaços:

$q \ s \ t \ q_0 \ a$

onde:

- q é um inteiro positivo indicando a quantidade de estados do autômato
Obs: Os estados serão representados por números inteiros de 0 a $(q - 1)$
- s é um inteiro positivo indicando a quantidade de símbolos do alfabeto estendido Σ_ϵ (*incluindo a cadeia vazia*)
Obs: o algarismo 0 é reservado para representar a cadeia vazia; todos os símbolos do alfabeto deverão ser representados por números 1 a $(s - 1)$
P.ex se o alfabeto de um AFN é $\Sigma = \{1, 2\}$, então teremos $s = 3$ e o alfabeto estendido Σ_ϵ será $\{0, 1, 2\}$ (0 representando ϵ).
- t é um inteiro positivo indicando a quantidade de transições do AFN
- $q_0 \in \{0, 1, \dots, q - 1\}$ é o índice do estado inicial
- a é um inteiro ≥ 0 indicando o número de estados de aceitação do AFN.
Obs: Note que, embora não usual, a poderia ser igual a 0, indicando que o AFN não teria nenhum estado de aceitação e, portanto, rejeitaria quaisquer cadeias de entrada. Todavia, nos exemplos que trataremos na correção, não consideraremos este caso.

s é o número de símbolos do alfabeto

- A segunda linha conterá os índices dos estados de aceitação (todos entre 0 e $(q-1)$), separados por espaços.
- As demais linhas do bloco (linhas 3 a $t + 2$) conterão as especificações das transições, na forma:

<índice estado corrente> <índice símbolo> <índice novo estado>.

Exemplos:

0 1 2 \rightarrow transição do estado q_0 para o estado q_2 com o símbolo 1

0 0 2 \rightarrow transição do estado q_0 para o estado q_2 com a cadeia vazia (ϵ)

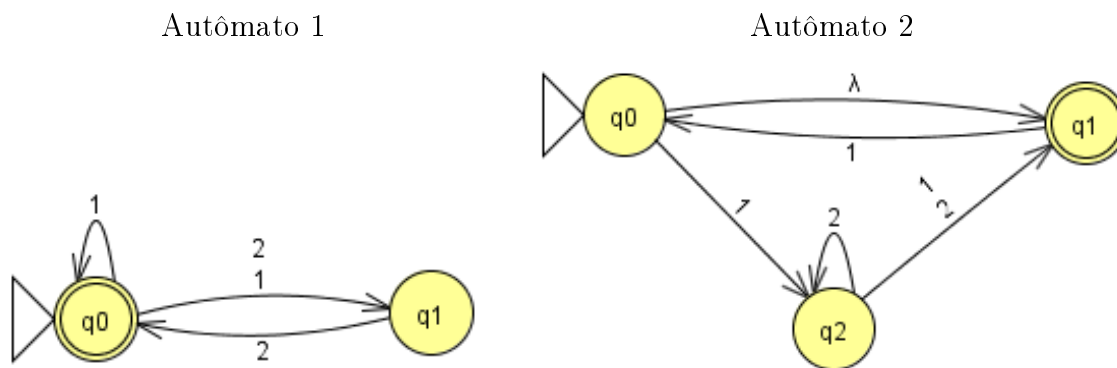
A especificação das cadeias será formada por um bloco de linha no seguinte formato:

- A primeira linha conterá o número de cadeias de teste (inteiro positivo).
- Cada uma das demais linhas do bloco conterá uma cadeia, com seus símbolos separados por espaços. A cadeia vazia será representada por 0.

Obs: o símbolo 0 só deverá aparecer se for a cadeia vazia. Ele não aparecerá em cadeias não vazias

P.ex. o arquivo de entrada não deverá conter cadeias como “0 2 1” ou “1 2 0 1”

O exemplo abaixo ilustra um arquivo de testes referente aos dois automatos da figura abaixo:



```

2          numero de automatos de teste
2 3 4 0 1  AFN 1: 2 estados, 3 simbolos, 4 transicoes, est.inicial=0, 1 est. aceit
0          F = {0}
0 1 0      especificações das transicoes
0 1 1
0 2 1
1 2 0

6          numero de cadeias de teste do automato 1
1          1a cadeia de teste
1 1        2a cadeia
1 1 1
1 2 2 1 1 1 2 2 1
2 2 2 1
2 1 2 2

3 3 6 0 1  AFN 2: 3 estados, 3 simbolos, 6 transicoes, est.inicial=0, 1 est. aceit
1          F = {1}
0 0 1      exemplo de transicao com cadeia vazia
1 1 0
0 1 2
2 2 2

```

```

2 1 1
2 2 1
8          numero de cadeias de teste do automato 1
0          cadeia de teste 1 (cadeia vazia)
1          cadeia de teste 2
1 2 2 2 2 2 1
1 1 1 2 1 1 1
1 2 1 1
1 2 2 1 2 2
2
1 1 2 2 1 2

```

<ArqSaida>:

O arquivo de saída conterá seqüências de 0s e 1s separados por espaço, indicando a aceitação (1) ou rejeição (0) das cadeias pelos respectivos autômatos. Cada linha conterá os resultados dos testes das cadeias de um autômato.

Para arquivo de entrada ilustrado acima, o arquivo de saída resultante seria composto pelas seguintes linhas

```

1 1 1 1 0 0      Aceitacoes/rejeicoes das cadeias de teste do automato 1
1 1 1 1 1 0 0 0  Aceitacoes/rejeicoes das cadeias de teste do automato 2

```

Entrega do trabalho:

Condições da entrega:

- O trabalho poderá ser feito em duplas, devidamente identificados na primeira linha do código-fonte.
- A entrega deverá ser realizada através do ambiente e-disciplinas, até **20/10/2020**. O projeto deverá ser entregue em um diretório compactado (formato ZIP), nomeado da seguinte forma:
d<numerouspaluno1>_<numerouspaluno2> (note que o prefixo “d” é importante para posterior processamento do nome do arquivo)

Esse diretório deverá conter:

- O programa-fonte (em C, Java, Python ou R) com nome `afn.c`, `afn.java`, `afn.py` ou `afn.r`
- Se houver outros módulos-fonte adicionais, deverão estar no mesmo diretório do programa-fonte principal
- Para entregas em C ou Java, a respectiva versão compilada, de nome `afn.exe` ou `afn.class`.

- Para entregas em R ou em Python, um arquivo-texto intitulado `packages.txt` com os nomes de todos os pacotes externos que deverão ser instalados via comando `"pip install"`(um nome de pacote por linha).
- O código-fonte deverá ser compilável no gcc ou no Eclipse-Java/Python.
- Não é necessário que os dois alunos da dupla enviem o mesmo EP, basta um envio por dupla.
- Dúvidas a respeito das especificações ou a respeito da implementação do trabalho serão sanadas até o dia 16/10/2020. Dúvidas encaminhadas após este prazo serão ignoradas.
- Além da correção do programa, será considerada a qualidade da documentação do código fonte.
- Se houver evidência de plágio entre trabalhos de grupos distintos de qualquer uma das turmas, os mesmos serão desconsiderados.