Linguagens Formais e Autômatos
Prof. Andrei Rimsa Álvares

Trabalho Prático

1. Objetivo

O objetivo desse trabalho é permitir que os alunos apliquem os conceitos assimilados na disciplina em um trabalho prático de implementação. A ideia é desenvolver um dos algoritmos vistos na disciplina em um programa de computador.

2. Descrição

Implementar um programa de computador que receba uma especificação de uma Máquina de Turing (MT) não-determinística e uma palavra de entrada e verifique se essa palavra pertence ou não a linguagem descrita por essa máquina.

3. Instruções

Considere uma Máquina de Turing não-determinística $M = (E, \Sigma, \Gamma, x, y, \delta, i, F)$, tal que:

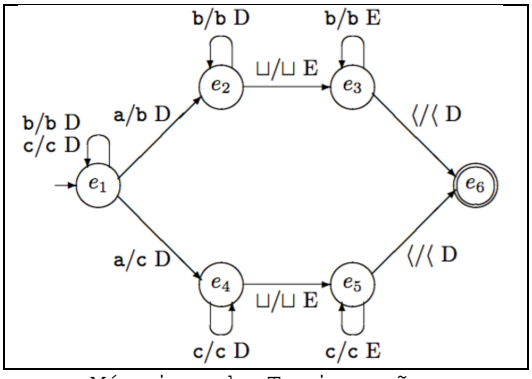
- E é um conjunto finito de estados;
- $\Sigma \subseteq \Gamma$ é o alfabeto de entrada;
- Γ é o alfabeto da fita;
- x é um símbolo marcador de início da fita ($x \in \Gamma - \Sigma$);
- y é um símbolo de células vazias da fita ($y \in \Gamma - \Sigma$, $y \neq x$);
- $\delta: E \times \Gamma \rightarrow P(E \times \Gamma \times \{<,>\})$ é a função de transição;
- i é o estado inicial;
- F é o conjunto de estados finais.

Essa máquina pode ser expressa em formato JSON conforme a seguir, onde símbolos do alfabeto da fita são formados por exatamente um caractere, enquanto nomes de estados podem ser formados por um ou mais caracteres.

```
{ "mt": [
  [e - ∀e ∈ E],
  [s - ∀s ∈ Σ],
  [x, y, s, t - ∀s ∈Σ ; ∀t ∈ Γ],
  x,
  y,
  [
    [e1, a, e2, b, d - e1,e2 ∈ E ; a,b ∈ Γ ; d ∈ {<,>}]
  ]
  i - i ∈ E,
  [f - ∀f ∈ F]
]}
```

Considere o exemplo a seguir de uma Máquina de Turing não-determinística que reconhece a linguagem $L = b^*ab^* + c^*ac^*$. O diagrama dessa máquina pode ser vista a seguir à esquerda e seu respectivo formato em JSON à direita. Nessa conversão foram escolhidos os símbolos `[` (abre colchetes)

e _ (*underline*) para expressar o início de fita e branco respectivamente. **Porém, quaisquer outros dois símbolos poderiam ter sido utilizados, dadas as restrições da definição acima.**



Máquina de Turing não-determinística.

```
{ "mt": [
  ["e1", "e2", "e3", "e4", "e5", "e6"],
  ["a", "b", "c"],
  ["[", "_", "a", "b", "c"],
  "[",
  "_",
  [
    ["e1", "b", "e1", "b", ">"],
    ["e1", "c", "e1", "c", ">"],
    ["e1", "a", "e2", "b", ">"],
    ["e1", "a", "e4", "c", ">"],
    ["e2", "b", "e2", "b", ">"],
    ["e2", "_", "e3", "_", "<"],
    ["e3", "b", "e3", "b", "<"],
    ["e3", "[", "e6", "[", ">"],
    ["e4", "c", "e4", "c", ">"],
    ["e4", "_", "e5", "_", "<"],
    ["e5", "c", "e5", "c", "<"],
    ["e5", "[", "e6", "[", ">"]
  ],
  "e1",
  ["e6"]
]}
```

mt.json

Dada uma especificação de uma máquina de Turing em formato JSON e uma palavra de entrada, deve-se desenvolver um programa de computador capaz de simular a execução dessa máquina para essa determinada palavra de entrada. Ela deve ser capaz de responder *Sim* se a palavra pertencer a linguagem descrita por essa máquina ou *Não* caso contrário. As máquinas de testes nunca entrarão em loop infinito. **Essa máquina deve operar em uma fita virtualmente infinita à direita**, onde o tamanho dela não deve ter um limite pré-estabelecido, mas sim estar limitada a quantidade de memória do sistema onde ela será executada.

Essa implementação pode ser feita em qualquer linguagem de programação, desde que exista compilador/interpretador gratuito disponível para ela. **Esse programa deve funcionar exclusivamente em linha de comando, onde suas entradas são recebidas via argumentos em linha de comando e seu resultado exibido na saída padrão do terminal conforme exemplificado a seguir.** Note que nenhuma outra saída não especificada deve ser exibida, como mensagens de depuração por exemplo. **Para cada palavra de teste, sua solução não pode demorar mais de 3 segundos para dar providenciar uma resposta, independente da linguagem utilizada.**

```
$ ./mt
Usar: ./mt [MT] [Palavra]
$ ./mt mt.json "bbabbbb"
Sim
$ ./mt mt.json "ccac"
Sim
$ ./mt mt.json "a"
Sim
$ ./mt mt.json ""
Não
```

Linguagens Formais e Autômatos
Prof. Andrei Rimsa Álvares

4. Avaliação

O trabalho deve ser feito em grupo de até dois alunos, sendo esse limite superior estrito. O trabalho será avaliado em 10 pontos, onde essa nota será multiplicada por um fator entre 0.0 e 1.0 para compor a nota de cada aluno individualmente. Esse fator poderá estar condicionado a apresentações presenciais a critério do professor. A avaliação é feita exclusivamente executando casos de testes criados pelo professor. Portanto, códigos que não compilam ou não funcionam serão avaliados com nota zero.

Trabalhos copiados, parcialmente ou integralmente, serão avaliados com nota **ZERO** do valor da prática, sem direito a contestação. Você é responsável pela segurança de seu código, não podendo alegar que outro grupo o utilizou sem o seu consentimento.

Programas fora do padrão que recebam entradas ou produzam saídas de forma diferente da especificada não serão considerados e, portanto, serão avaliados com nota ZERO.

5. Submissão

O trabalho deverá ser submetido até as 23:59 do dia 24/06/2024 (segunda-feira) exclusivamente via sistema acadêmico em pasta específica. **Não serão aceitos, em hipótese alguma, trabalhos enviados por e-mail ou por quaisquer outras fontes**. A submissão deverá incluir todo o código-fonte do programa desenvolvido e arquivos de apoio necessários em um arquivo compactado (zip ou rar). **Não serão consideradas submissões com links para hospedagens externas**. Deve-se incluir um arquivo README na raiz do projeto com o(s) nome(s) do(s) desenvolvedor(es) e com instruções claras de como compilar e executar o programa implementado. Para trabalhos feitos em dupla, a submissão deverá ser feita por apenas um de seus integrantes.