Tarefa 3 — Funções com Domínios Infinitos, Autômatos e Expressões Regulares & Loops e Infinitude & Modelos Equivalentes de Computação

Teoria da Computação — PPComp

Prof. Jefferson O. Andrade

Campus Serra — Ifes

2021/2

1 Introdução

Para esta tarefa você deve resolver os problemas da Seção 3. A sua resposta deve ser preparada em um documento em LaTeX. A classe do documento deve ser `scrartcl`, com fonte tamanho 11. Cada exercício deve ser respondido em uma seção própria. Caso o exercício possua itens/partes, cada item/parte deve ser respondido em uma subseção.

A qualidade da formatação do seu relatório será um dos fatores considerados na correção. O LaTeX permite a criação de documentos extremamente elegantes, use o potencial da ferramenta.

2 Orientações

Colaboração: você pode colaborar com outros alunos que estão atualmente matriculados nesta disciplina em *brainstorming* e pensando em abordagens para soluções, mas você deve escrever as soluções por conta própria e não pode compartilhá-las com outros alunos.

Violações graves: compartilhar perguntas ou soluções com qualquer pessoa fora desta disciplina, incluindo postagem em sites externos, constitui uma violação do código de ética. Em particular, você não pode obter ajuda de alunos ou materiais de anos anteriores desta disciplina ou equivalente. Casos de **plágio** serão relatados à coordenação de curso e encaminhados ao conselho de ética.

Formato de submissão: Esta tarefa inclui questões teóricas, mas também questões de programação que serão testadas pelo professor. Deste modo devem ser entregues dois artefatos: um relatório de solução, e os códigos fontes. Os dois artefatos devem ser entregues como **um único arquivo ZIP** (não é RAR, não é LHA, é ZIP). O arquivo ZIP deve obrigatoriamente ter a seguinte estrutura:

```
task3
├── <<nome-do-relatório>>.pdf
└── src
    ├── probl1b
    │   ├── probl1b.clj
    │   └── README.md
    └── ...
```

1. **Relatório de solução:** Contendo a solução de **todas** as questões, inclusive as que pedem implementação. As questões que pedem implementação de código como solução devem incluir um breve comentário sobre a ideia gera da solução e apontar para o diretório que contém o código fonte.

   O PDF submetido deve ser digitado no mesmo formato que este. Inclua o texto dos problemas e escreva "Solução X" antes da sua solução. Poderão ser deduzidos pontos se você enviar em um formato diferente.

2. **Códigos fontes:** Os códigos fontes de todos os problemas devem estar organizados dentro de um diretório chamado `src`. Nesse diretório haverá um subdiretório para cada problema, e.g, `probl1b`. Nos subdiretórios haverá o(s) arquivo(s) com código fonte. Para cada problema deve ser entregue o código fonte do programa que resolve o problema e um arquivo `README.md`, em formato Markdown, contendo:

   - Nome do autor.
   - Uma **breve** explicação do código que foi implementado.
   - Uma descrição de como executar as funções.

   Um exemplo do conteúdo do arquivo `README.md` é dados abaixo.

````
# Tarefa 3 --- Teoria da Computação --- 2021/2

**Funções com Domínios Infinitos, Autômatos e Expressões Regulares & Loops
 ↪  e Infinitude & Modelos Equivalentes de Computação**

**Autor:** Jefferson O. Andrade

## Problema 0

Este problema pede a construção de uma função que receba como entrada um
número $n$ representando o tamanho de uma cadeia de bits e retorne uma
sequência com todas as cadeias em binário com o tanho indicado.

**Solução**

A solução adotada foi implementar uma recursão. O caso base é o
comprimento zero. A única cadeia de tamanho zero é a cadeia vazia. Nos
demais casos, gera-se todas as cadeias de tamanho $n-1$ e em seguida
gera-se duas semi-duplicadas, uma com 0 acrescentando à frente de cada
cadeia, e outra com 1 acrescentado à frente de cada cadeia. O resultado
é a concatenação das duas semi-duplicatas.

**Execução**

Para executar a função no prompt do Clojure execute, por exemplo:

```
(all-bin-strings 5)
```
````

3 Problemas

Problema 1

Considere a definição "tradicional" de *autômato finito determinístico* (AFD):

$$M = \langle Q, \Sigma, \delta, q_0, F \rangle$$

Onde,

- Q — é o conjunto de estados de M;
- Σ — é o alfabeto da cadeia de entrada;
- $\delta : Q \times \Sigma \to Q$ — é a função de transição de M;
- $q_0 \in Q$ — é o estado inicial de M;
- $F \subseteq Q$ é o conjunto de estados de aceitação de M.

Sejam dois AFDs[1] $M_1 = \langle Q_1, \Sigma, \delta_1, q_0, F_1 \rangle$ e $M_2 = \langle Q_2, \Sigma, \delta_2, p_0, F_2 \rangle$. O *produto assíncrono* de M_1 e M_2, denotado por $M_1 \odot M_2$, é definido como $M_1 \odot M_2 = \langle Q_\pi, \Sigma, \delta_\pi, s_0, F_\pi \rangle$, onde:

- $Q_\pi = Q_1 \times Q_2$
- $\delta_\pi((q_i, p_i), \sigma) = (\delta_1(q_i, \sigma), \delta_2(p_i, \sigma))$
- $s_0 = (q_0, p_0)$
- $F_\pi = \{(q_f, p_f) \mid q_f \in F_1 \text{ ou } p_f \in F_2\}$

Parte I — (10 pts)

Prove ou refute a afirmação: o autômato $M_1 \odot M_2$ reconhece a linguagem $L(M_1) \cup L(M_2)$.

Parte II — (10 pts)

Escreva uma função em Clojure que receba como entra a representação de dois AFDs e retorne o AFD correspondente ao *produto assíncrono* dos dois. O formato dos AFDs deve seguir o que é mostrado no exemplo abaixo.

```
{:alphabet #{0 1}
 :states #{"q0" "q1" "q2" "q3"}
 :initial "q0"
 :accepting #{"q3"}
 :transitions {["q0" 0] "q1"
               ["q0" 1] "q0"
               ["q1" 0] "q2"
               ["q1" 1] "q0"
               ["q2" 0] "q2"
               ["q2" 1] "q3"
               ["q3" 0] "q3"
               ["q3" 1] "q3"}}
```

Os símbolos do alfabeto podem ser caracteres ou inteiros. Os estados podem ser inteiros ou strings.

Parte III — (10 pts)

Sejam $M_1 = \langle Q_1, \Sigma, \delta_1, q_0, F_1 \rangle$ e $M_2 = \langle Q_2, \Sigma, \delta_2, p_0, F_2 \rangle$ dois AFDs completos. Defina a operação *produto síncrono* de M_1 e M_2, denotado por $M_1 \otimes M_2$, de tal modo que $M_1 \otimes M_2$ reconheça a linguagem $L(M_1) \cap L(M_2)$. Ou então, demonstre que não é possível definir esta operação para autômatos finitos determinísticos.

[1] Considere que os autômatos M_1 e M_2 são *completos*, ou seja, que cada estado possui transições para todos os caracteres do alfabeto.

Problema 2 (*15 pts*)

Considere o alfabeto Σ_3 definido abaixo.[2]

$$\Sigma_3 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$$

Uma cadeia de símbolos de Σ_3 pode ser vista como 3 linhas de 0s e 1s. Considere cada linha como um número binário. Seja $\mathcal{B}$ a linguagem definida abaixo:

$$\mathcal{B} = \{w \in \Sigma_3^\star \mid \text{a última linha de } w \text{ é a soma das duas primeiras}\}$$

Por exemplo, $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} \in \mathcal{B}$ mas $\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix} \notin \mathcal{B}$.

Mostre que a linguagem $\mathcal{B}$ é regular.

Problema 3 (*20 pts*)

Para cada uma das funções especificadas abaixo, defina a Máquina de Turing que computa a função. A Máquina de Turing deve ser descrita utilizando a representação definida pelo simulador de Máquina de Turing de Anthony Morphett.

(a) INC — Função que recebe um número binário na fita e produz o seu incremento. O resultado é separado da entrada por um caractere '`#`'. Por exemplo, para a entrada `10011`, após a execução da Máquina de Turing o estado da fita seria `10100#10011`.

(b) ADD — Função que recebe dois números binários na fita e produz a sua soma. Os números de entrada e o resultado são separados uns dos outros pelo caractere '`#`'. Por exemplo, para a entrada `1011#110`, o estado da fita após a execução da Máquina de Turing seria `10001#1011#110`.

Problema 4 (*15 pts*)

Sejam $\mathbb{0} = \lambda x.\lambda y.y$ e $\mathbb{1} = \lambda x.\lambda y.x$. Defina

$$\mathsf{ALT} = \lambda a, b, c.(a(b\,\mathbb{1}(c\,\mathbb{1}\,\mathbb{0}))(b\,c\,\mathbb{0}))$$

Prove que ALT é uma expressão λ que computa a função "*pelo menos dois*". Ou seja, que para quaisquer $a, b, c \in \{\mathbb{0}, \mathbb{1}\}$ (conforme definidos acima) $\mathsf{ALT}\,a\,b\,c = \mathbb{1}$ se e somente se ao menos duas das entradas forem iguais a $\mathbb{1}$.

Problema 5 (*20 pts*)

Usando a Codificação de Church para Booleanos, números naturais, pares e lista em cálculo λ, implemente as funções pedidas abaixo.[3]

(a) MIN — recebe uma lista de números e retorna o menor deles.

(b) MAX — recebe uma lista de números e retorna o maior deles.

(c) APPEND — recebe duas listas e retorna a concatenação destas duas listas.

(d) REVERSE — recebe uma lista e retorna a lista invertida.

[2] Note que os símbolos deste alfabeto são as matrizes.

[3] Pode ser que você ache útil testar as suas funções em um interpretador de cálculo λ. Uma sugestão é usar o interpretador Lambster pois ele tem uma sintaxe bastante próxima da sintaxe do cálculo λ original.