Trabalho 1 — Teoria da Computação

Computabilidade — Cálculo λ

Prof. Jefferson O. Andrade

Ifes — Campus Serra — PPComp

2022/2

1 Introdução

Para este trabalho 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:

```
trab1
├── <<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[1], 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 na Figura 1.

3 Problemas

Problema 1 (*10 pt*)

Escreva um programa de computador P, não vazio, cuja saída seja $\langle P\rangle\langle P\rangle$, ou seja, o próprio código fonte de P impresso duas vezes em seguida. O programa P pode ser implementado na sua linguagem de programação favorita, desde que ela seja livre e disponível para Ubuntu Linux nos repositórios oficiais.

Problema 2 (*20 pt*)

(a) Demonstre que o conjunto das linguagens (problemas computacionais) decidíveis é um subconjunto próprio do conjunto das linguagens reconhecíveis.

[1] Você não precisa escrever o arquivo `README.md` "as cegas" se não quiser. Existem vários bons editores de Markdown, inclusive alguns bons editores online livres, como o StackEdit e o Dillinger, por exemplo.

```
# 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)
```
```

Figura 1: Exemplo de arquivo `README.md` que deve ser entregue com os códigos fonte.

(b) Seja $HALT = \{\langle M \rangle \mid M(w)$ termina para toda string $w\}$. Ou seja, $HALT$ é a linguagem formada pelas string que representam (codificações de) Máquinas de Turing que sempre param para qualquer entrada. Mostre que $\overline{HALT}$ não é reconhecível.

Problema 3 (*15 pt*)

O *Problema de Correspondência de Post* (PCP) [1, Pag. 134, 135] é não decidível. Entretanto, existe uma variação do PCP conhecida como *Problema de Correspondência de Post bobo* (*silly Post Correspondence Problem* – SPCP), se faz a restrição de que as strings da parte de cima e da parte de baixo do dominó tem que ter o mesmo tamanho. Mostre que o SPCP é decidível.

Problema 4 (*15 pt*)

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 (*30 pt*)

Usando a Codificação de Church para Booleanos, números naturais, pares e lista em cálculo λ, implemente as funções pedidas abaixo.[2]

(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.

Problema 6 (*10 pt*)

Utilizando uma linguagem de programação que tenha a construção *lambda*, i.e., definição de funções anônimas, implemente as funções dadas no Problema 5. Sua implementação deve ser o mais próxima do Cálculo λ que a linguagem em questão permitir.

Referências

[1] John MacCormick. *What Can Be Computed? A Practical Guide to the Theory of Computation*. Princeton, New Jersey: Princeton University Press, 2018. 383 pp. ISBN: 978-0-691-17066-4.

[2] 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.