Projeto 2: λ-cálculo e Criptografia

Edeyson Gomes Lais Salvador

24 de outubro de 2022

1 Introdução

Embora o paradigma de programação imperativo seja o mais popular entre programadores, ele é apenas um entre vários meios de codificar os nossos algoritmos. Além dele, há também a **Programação em Lógica** (ex.: Prolog), **Dataflow** (ex.: o motor interno de uma planilha), **Function-Level** (os programas não manipulam dados, mas sim outros programas), **Programação Funcional** (ex.: Lisp), entre outros. E, é claro, o mais conceitual/teórico, a **máquina de Turing** - à qual todos os paradigmas anteriores são computacionalmente equivalentes.

Cada um desses paradigmas apresenta vantagens e desvantagens. A principal razão, pela qual se usa um paradigma ou outro, é o grau de facilidade de codificação de um programa para realizar uma determinada tarefa, como também a concisão e legibilidade de suas representações. Por exemplo, a princípio, todos poderíamos estar programando com máquinas de Turing, mas além da dificuldade de se ler/escrever/compreender tais programas, também seriam por demais extensos em relação ao que fazem de útil. Assim, paradigmas diferentes oferecem estilos de programação mais adequados a certos domínios de aplicação (ainda que suas linguagens representativas sejam **Turing-completas**).

O **λ-cálculo** (cálculo Lambda) é a base da programação funcional e, à medida que o mundo adota mais e mais a programação funcional, talvez seja útil conhecer suas raízes. É um cálculo importante na Teoria de Linguagem de Programação e também na Teoria da Computação: foi proposto por *Alonzo Church*, é da mesma época da Máquina de Turing, e o próprio *Turing* demonstrou a equivalência entre a sua máquina universal e o λ Cálculo.

Pode ser considerada a menor linguagem de programação universal: consiste apenas em definição e aplicação de funções. Qualquer função computacional/computável pode ser avaliada no contexto de λ-cálculo e a avaliação de funções consiste em uma única regra de transformação: um tipo específico de **redução** baseada em substituição dos parâmetros da função por termos λ (variáveis ou funções).

O λ-cálculo puro não contém nada além de funções. A função mais simples é a identidade $\lambda x.x$ - que aplicada a qualquer termo lambda M resulta no próprio M. Como exemplificado em:

$$(\lambda x.x)\ M \rightarrow\ M \tag{1}$$

Na equação 1 temos a aplicação da função identidade ao argumento M através de uma β-redução, a computação do λ-cálculo.

Por sua vez, para representar números, valores lógicos é necessário codificá-los como funções. Felizmente, *Alonzo Church* já criou uma codificação, onde o valor de um número é equivalente ao número de vezes que uma função é aplicada a um argumento[1]. O mesmo acontece para os booleanos e operações aritméticas, ou qualquer outra "coisa" que quisermos representar no λ-cálculo.

2 Problema

A troca de mensagens codificadas, guardando segredo sobre seu conteúdo, é uma necessidade que se tornou evidente com a diplomacia e, principalmente, com as guerras. A importância de uma comunicação segura, salvaguardando segredos, motivou o desenvolvimento de técnicas para codificar mensagens. À medida que a informação se torna um bem cada vez mais valioso, tais técnicas tornam-se cada vez mais relevantes para a sociedade.

Um estudante de pós-graduação da UFBA, ao se deparar com a necessidade de interoperar múltiplos sistemas de saúde, resolveu implementar uma

[1] Pesquise sobre codificação de Church

camada de ciframento sobre os dados dos pacientes, a fim de assegurar seu requerido sigilo.

Como prova de conceito, o estudante resolveu fazer testes usando um método de ciframento por **transposição** e dois métodos distintos de ciframento por **substituição** [2].

Para a implementação da prova de conceito, o estudante resolveu solicitar ajuda a seus colegas da UFBA para que formalizem os três métodos de ciframento usando λ-cálculo e os implementem em Haskell [3], apresentando testes que forneçam evidências da corretude das formalizações. Em cada teste, os dados devem ser cifrados e decifrados corretamente, e todo o processo deve ser documentado.

3 Produto

Você deverá postar no Moodle UFBA até às 23 : 59 do dia 05/*dezembro*/2022, no espaço apropriado para tal, um relatório no modelo de artigos da SBC com uma discussão sobre: 1. Os métodos de ciframento escolhidos; 2. Como os métodos escolhidos se diferenciam; 3. A formalização dos métodos em λ-cálculo puro; 4. A implementação dos métodos em Haskell; 5. Testes dos métodos.

No relatório também deverá constar as aplicações de funções λ-cálculo com as possíveis reduções.

Para fins de documentação e avaliação dos termos em λ-cálculo puro, é obrigatória a utilização de uma ferramenta como o *Lambda Calculator*[4].

Adicionalmente, toda a implementação e/ou formalização deve ser provida em arquivo texto (ASCII editável) como .txt, .java, .hs, etc.

4 Recursos para aprendizagem

DIVERIO, Tiaraju A.; MENEZES, Paulo F. Blauth. **Teoria da Computação – Máquinas Universais e Computabilidade**. Porto Alegre: Sagra-Luzzatto,1999. 205p.

MARTINS, Raul Cesar Baptista; MOURA, Arnaldo Vieira; **Desenvolvimento sistemático de programas corretos: a abordagem denotacional**. ESCOLA DE COMPUTAÇÃO 6. Campinas, SP: Ed. da UNICAMP, 1988.

SILVA, Flavio S. C.; MELO, Ana Cristina V. **Modelos Clássicos de Computação**. Cengage, 2010.

GAREY, Michael R.; JOHNSON, David S. **Computers and intractability**. New York: wh freeman, 2002.

BORGES, Fábio; BROWN, Lawrie. **Criptografia e Segurança em Rede Capítulo 2**. Disponível em: https://www.lncc.br/~borges/ist/SIN/cap02.pdf (Acessado: 24 de Outubro de 2022).

Criptografia e Segurança em Rede Técnicas Clássicas de Encriptação - University of São Paulo. Disponível em: http://wiki.stoa.usp.br/images/c/cf/Stallings-cap2e3.pdf (Acessado: 24 de Outubro de 2022).

MALAQUIAS, J.R. **Programação Funcional em Haskell**. Disponível em: http://www.decom.ufop.br/romildo/2014-2/bcc222/ (Acessado: 24 de Outubro de 2022).

BEZERRA, Débora J.; MALAGUTTI, Pedro L.; RODRIGUES, Vânia C. S. **Aprendendo Criptologia de Forma Divertida**. Disponível em: http://www.mat.ufpb.br/bienalsbm/arquivos/Oficinas/PedroMalagutti-TemasInterdisciplinares/Aprendendo_Criptologia_de_Forma_Divertida_Final.pdf (Acessado: 24 de Outubro de 2022).

Referências

Este problema é baseado em *What is Lambda Calculus and should you care*?[5]

[2] Para fins de segurança, a substituição simples não é uma opção de projeto

[3] https://www.haskell.org/

[4] Disponível para download em `http://www.cburch.com/proj/lambda`

[5] Disponível em `https://zeroturnaround.com/rebellabs/what-is-lambda-calculus-and-why-should-you-care`