Tomaz Gomes Mascarenhas

Formalização da Complexidade Temporal de Algoritmos de Ordenação em Lean

Belo Horizonte, Minas Gerais
2020

Tomaz Gomes Mascarenhas

Formalização da Complexidade Temporal de Algoritmos de Ordenação em Lean

Proposta de Pesquisa Científica

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Haniel Barbosa

Belo Horizonte, Minas Gerais
2020

Sumário

1 INTRODUÇÃO 3
1.1 Contextualização 3
1.2 Objetivos 3
2 REFERENCIAL TEÓRICO 5
3 METODOLOGIA 6
4 RESULTADOS ESPERADOS 7
5 CRONOGRAMA 8
REFERÊNCIAS 9

1 Introdução

1.1 Contextualização

Uma das propriedades mais desejáveis para um software é a corretude, isto é, a garantia de que o programa respeita uma determinada especificação para qualquer entrada que receba. Existem áreas nas quais uma falha de corretude tem um custo altíssimo financeiro e até humano, como é o caso de softwares que controlam carros autônomos, recursos de uma aeronave ou equipamentos médicos.

Para garantir que um software não irá falhar, é necessário definir corretamente qual especificação ele deve seguir. Se a especificação for incorreta, apresentar evidências que o sistema sempre a respeita não irá impedí-lo de falhar. Diversas técnicas podem ser adotadas para abordar o problema de obter evidências de corretude, uma vez que a especificação foi feita. A mais comum é por meio de testes, que são simples de se realizar, e já se conhecem técnicas que os fazem ser muito eficientes em encontrar erros. Contudo, como observou (Dijkstra et al. 1972) "Os testes podem mostrar apenas a presença de erros, e não sua ausência". Ou seja, por mais que esse método seja preciso, não é possível ter garantia total de corretude usando ele, e, em casos como aqueles citados no final do parágrafo anterior, essa certeza é uma necessidade.

Uma outra abordagem para se tentar garantir que o programa atende à especificação é o uso de assistentes de demonstração. Estas são ferramentas que dão suporte a linguagens de programação e fazem uso de relações profundas entre lógica e computação para que elas sejam capazes de funcionar como provadores interativos de teoremas, além de serem capazes de verificar a corretude de demonstrações fornecidas. Embora seja muito mais trabalhoso e exigir um conhecimento mais específico do que a realização de testes, ao usar esse método é possível ter uma garantia matemática que o programa não irá falhar, supondo que o assistente em questão também esteja correto.

1.2 Objetivos

O objetivo deste trabalho é usar Lean(1), um assistente de demonstração, para formalizar algum conceito que ainda não esteja em sua biblioteca e seja relevante para a comunidade, de modo a criar uma contribuição e adquirir familiaridade com tal ferramenta. De acordo com membros ativos da comunidade [1] que utiliza Lean, a complexidade temporal de algoritmos é um conceito relevante que ainda não foi formalizado em nenhum módulo da biblioteca. Em particular, a complexidade temporal de algoritmos básicos de ordenação,

[1] <https://leanprover.zulipchat.com/#narrow/stream/113488-general/topic/BSc.20Final.20Project>

que poderia ser formalizada sem usar técnicas muito avançadas. Assim, o escopo do trabalho será a demonstração de um limite, com base no tamanho da entrada, para o número de operações feitas pelos algoritmos Insertion Sort e Merge Sort, implementados na biblioteca do Lean[2].

[2] <https://github.com/leanprover-community/mathlib/blob/master/src/data/list/sort.lean>

2 Referencial Teórico

Formalizar uma propriedade, como a complexidade temporal, significa definir precisamente o que ela representa em termos matemáticos. Uma vez que a formalização foi feita, é possível tentar construir um argumento lógico para mostrar que um certo objeto satisfaz aquela propriedade (no caso desse trabalho, que um algoritmo possui uma certa complexidade). Embora seja um tópico importante no projeto de algoritmos, a complexidade temporal ainda não foi amplamente explorada pelos assistentes de demonstração, possuindo poucos exemplos para serem seguidos. Um deles é a formalização desse conceito em Agda, um outro assistente, feita por Twan Van Laarhoven [1].

Lean é um assistente de demonstrações desenvolvido na Microsoft Research, liderado por Leonardo de Moura. Além de sua biblioteca padrão, o assistente também possui uma vasta biblioteca mantida pela comunidade, chamada *mathlib*[2]. Esse repositório apresenta a formalização de diversas áreas da matemática, além de algoritmos e estruturas de dados. A execução do trabalho será feita com base nos algoritmos de ordenação implementados na *mathlib*.

[1] <https://www.twanvl.nl/blog/agda/sorting>
[2] <https://github.com/leanprover-community/mathlib>

3 Metodologia

Para realizar o trabalho, serão seguidos os seguintes passos:

- Estudo do método utilizado por Twan Van Laarhoven (mencionado na sessão anterior)

  Ler e entender, detalhadamente, como a demonstração foi feita em Agda e como o conceito foi formalizado.

- Estudo do funcionamento do assistente

  Resolver exercícios e demonstrar teoremas mais simples para adquirir familiaridade com a ferramenta.

- Definição do conceito de complexidade temporal em Lean

  Implementar a formalização do conceito no assistente, baseado no que foi aprendido no tutorial de Laarhoven.

- Definição de teoremas e lemas

  Definir todos os teoremas que farão parte do trabalho, assim como os lemas que serão necessários.

- Demonstração dos teoremas

  Implementar as demonstrações. É possível que também seja realizada a demonstração convencional dos teoremas.

4 Resultados Esperados

Ao fim do trabalho, espera-se obter demonstrações em Lean de limites assintóticos para os algoritmos de ordenação da *mathlib*, sendo, em uma entrada de tamanho n, $\mathcal{O}(n^2)$ para o Insertion Sort e $\mathcal{O}(n * \log(n))$ para o Merge Sort. Além disso, também é desejável que tais demonstrações sejam adicionadas no repositório oficial da biblioteca. Possivelmente, esse trabalho também pode aperfeiçoar técnicas que existem para formalizar a complexidade de algoritmos de forma geral.

5 Cronograma

Semana	**Tarefa**
16/12	Definição do Tema
06/01	Estudo do Tutorial
13/01	Resolução de Exercícios
20/01	Resolução de Exercícios
27/01	Formalização do Conceito
03/02	Produção do Pitch Parcial
10/02	Definição dos Teoremas
17/02	Demonstração dos Teoremas
24/02	Escrita do Relatório
03/03	Escrita do Relatório
10/03	Escrita do Relatório
17/03	Produção do Pitch Final

Referências

1 MOURA, L. de et al. The lean theorem prover. *25th International Conference on Automated Deduction (CADE-25)*, Berlin, Germany, 2015.