Going Through Logical Foundations

Zhang Shaoqian
A0177310M

Abstract

The goal of the project is to finish the exercises of the textbook, *Logical Foundations*, the first volume of the series *Software Foundations*. Main focuses of the book include constructive logic, inductive proofs, and the proof assistant, Coq. I have finished the entire book except for a few optional /advanced exercises, and progressed to Volume 2 in the series, *Programming Language Foundations*, by three chapters.

1 Metadata

1.1 Reasons why do this as a project

The content of *Logical Foundations* (including my own solutions to exercises) are fully checked by Coq, meaning that I can easily test the correctness my answers by simply running the proof scripts and see if it passes the rigorous examinations. Thus, it is easier to keep track of the work done and get immediate feedback compared to traditional books, making *Logical Foundations* especially suitable for self-study. Rather than to research a dark, new area, a guided path fits my current situation better.

1.2 Schedule and organization

The book is structured like a dependency graph: the are core chapters and optional chapters, and a suggested route to tackle the most important ones. I followed it; I completed the route and then went through the optionals.

The first chapter (`Basics.v`) was finished on 8 October. The last chapter (`Rel.v`) was finished on 24 November. After that, I continued to work on Volume 2 because it was fun.

1.3 Deliverables

The proof scripts (i.e. the content of the book, plus my solutions to the exercises) are uploaded to a GitHub repository. Since the authors forbid public distributions of the solutions to the exercises, I try not to reference the name of the book in the repository.

2 Notes on core chapters

The order follows the suggested route. Here, when I state ”all exercises done”, I do not consider written assignments that require manual grading (they cannot be formally checked, so I did not pay much attention to them).

2.1 `Basics.v`

This chapter is the indroduction to all introductions. It familiarizes the reader with the basic syntax of Coq and essential concepts of functional programming. Logic-wise, it touches inductively defined types, and proof using basic tactics like simplification, rewriting and case analysis.

All exercises done.

2.2 `Induction.v`

This is a mind-blowing chapter that surveys the consolidation of mathematical thoughts into trusted programs. It covers proof by induction and relationship between formal and informal proofs. This is one of the most important chapters in the book: it shows how reasoning and programming can be combined to establish knowledge (even though, for now it is only trivial ones like `n + 0 = n`).

All exercises done, including one gigantic, 5-star proof from an earlier version of the book, which got deleted this year.

2.3 `Lists.v`

This chapter introduces data structures that can be reasoned. Like all books on functional programming, it starts with defining pairs, then lists. The more interesting part is where it extends induction from natural numbers to more inductively defined data structures, which enables the reasoning about lists. It also covers more data structures including options and partial maps.

All exercises done.

2.4 `Poly.v`

This chapter subtly shifts the focus to types by introducing polymorphism and higher-order functions. The content is similar to a typical Haskell book. Most reasoning is still done with induction, so logic-wise, nothing much new.

All exercises done.

2.5 `Tactics.v`

This chapter introduces various tactics that can be used in Coq to complete proofs; most of them are Coq specific. Logic-wise, a discussion on how general induction hypotheses should be is helpful.

All exercises done.

2.6 `Logic.v`

This chapter focuses on propositions: what they are and how to manipulate them. Topics covered include:

- logical connectives (conjunction, disjunction and nagation)
- truth and falsehoods
- logical equivalence
- existential qualification

and how to program and reason about these concepts. Finally, it discusses differences between standard mathematics' logic foundation, Set Theory, and Coq's logical core, the Calculus of Inductive Constructions.

- functional extensionality
- relationship between propositions and booleans
- excludded middle.

This chapter is closely related to our module.

All exercises done, except for the last 5-star, optional problem, which was too difficult for me.

2.7 `IndProp.v`

This chapter focuses on inductively defined propositions. It shows a powerful way to reason: inductively define propositions with evidence and, naturally, use induction on them. The importance of this chapter lies in subsequent chapters.

This chapter has some hard problems. Most exercises are done. Two optional exercises related to regex parsers are currently too difficult.

2.8 `Maps.v`

This is a short chapter that defines the data structure, map, which will be used to store states of another programming language in the following chapter.

All exercises done.

2.9 `Imp.v`

This is an exciting chapter where a imperative toy language, Imp, is designed, implemented and reasoned. Coq provides powerful tools to express the syntax and semantics of Imp, to run Imp programs, and to reason about their behaviors, all using `IndProps`.

Most exercises done; 4 optional problems left.

2.10 `Auto.v`

This chapter presents more ways to streamline proof scripts.

There are no exercises for this chapter.

3 Notes on optional chapters

3.1 `ImpParser.v` & `Extraction.v`

These chapters takes a closer look at the underlying lexing and parsing mechanism for Imp programs and shows how to compile Coq code to efficient ML code.

There are no exercises for these chapters.

3.2 `ImpCEvalFun.v`

In `Imp.v`, evaluations of Imp programs are defined using relations. In this chapter we attempt to represent evaluation using Coq functions. It provides deeper understanding on differences and similarities between functions and inductively defined propositions.

All exercises done.

3.3 `ProofObjects.v`

This chapter reveals a deeper connection between logic and computation: how propositions correspond to types, and how proof corresponds to data values. Topics covered include:

- using proof objects as values
- relationship between quantifiers, implications and functions
- logical connectives as inductive types
- inductive definition of equality

Though short, this chapter is particularly interesting.

All exercises done.

3.4 `IndPrinciples.v`

This chapter takes a closer look at induction: how the way of reasoning is realized using induction principles, which are in fact theorems. The chapter also discusses the relationship between formal and informal inductive proofs.

All exercises done.

3.5 `Rel.v`

This chapter is a case study on binary relations, the main focus being proving properties of them. It will be useful in *Programming Language Foundations.*

All exercises done.

4 Notes on Volume 2: *Programming Language Foundations*

4.1 `Equiv.v`

From this chapter on, the training in logic and Coq gained in *Logical Foundations* is put to use to study programming language theory. This chapter focuses on establishing equivalence between Imp programs.

All exercises done, except for one 5-star optional problem.

4.2 `Hoare.v` & `Hoare2.v`

This chapter focuses on formalizing Hoare Logic, a systematic compositional approach used to annotate programs and reason about their behaviors.

Most exercises done; 4 optional, 4/5-star problems left.

5 Conclusion

Nothing much to say; I find it extremely pleasurable and satisfying working through the exercises, and it has indeed prepared me for deeper content on logic and PLT.