FormalProofs-coq: Mechanized Verification of Basic Logical Theorems Using Coq

Mufitha Majeed

July 7, 2025

Abstract

This project, **FormalProofs-coq**, presents a formal exploration of fundamental theorems in propositional and predicate logic using the **Coq proof assistant**. It demonstrates both the manual construction of interactive proofs and the implementation of a simple logic evaluator that verifies tautologies automatically. This dual approach highlights the role of mechanized reasoning in formal methods and the value of theorem provers in ensuring correctness in logic-based systems.

1. Introduction

The ability to reason formally about logical statements is central to fields such as formal verification, artificial intelligence safety, and software correctness. By encoding logical theorems and their proofs in a mechanized framework like **Coq**, one ensures that each inference step is verifiable and free from human error.

The goal of this project is twofold:

1. To formalize and prove fundamental results in propositional and predicate logic using Coq's interactive theorem proving environment.
2. To develop a minimal decision procedure that evaluates logical formulas represented as inductive data structures, verifying their validity automatically within Coq.

2. Project Overview

2.1 Manual Formal Proofs in Coq

In the first part of the project, a series of well-known logical theorems were formalized and verified interactively in Coq. These include:

- **Law of Excluded Middle** (classical logic)
- **Modus Ponens** (constructive)
- **Double Negation Introduction** (constructive)
- **De Morgan's Laws** (both classical and constructive versions)
- Basic theorems in **predicate logic**, including relationships between universal and existential quantifiers.

These proofs were constructed using Coq tactics such as `intros`, `apply`, `destruct`, and `exfalso`, and demonstrate a clear understanding of the distinction between constructive and classical reasoning within a formal system.

2.2 Logic Evaluator and Tautology Verification

To extend the project beyond manual proof writing, a small decision procedure was implemented:

- An **inductive data type** representing logical formulas with connectives such as conjunction, disjunction, implication, and negation.
- A recursive **evaluation function** that computes the boolean value of a formula given a specific environment (truth assignment to variables).
- A verified example showing that a formula equivalent to **Modus Ponens** evaluates to true in all possible environments, formally proving it as a tautology.

This simple evaluator demonstrates how Coq can not only be used for interactive theorem proving but also for automating logical reasoning with machine-checked guarantees.

3. Significance and Applications

The project reflects foundational skills in:

- **Mechanized logic and interactive proof development**
- Understanding and distinguishing between **constructive and classical logic**
- Building and verifying **small decision procedures**

These skills are crucial in a wide range of domains, including:

- Formal verification of software and hardware systems
- AI safety and trustworthy computing
- Cryptography and secure system design

By formalizing even basic logical principles, the project establishes a strong base for future work in formal methods, type theory, and verified programming.

4. Future Work

To build upon this work, the following directions are proposed:

- Extending the evaluator to handle more complex logical systems (e.g., first-order logic).
- Formal verification of small functional programs and algorithms.
- Applying Coq's program extraction capabilities to generate executable, formally verified code.

5. Tools and Technologies Used

- **Coq Proof Assistant** (interactive theorem prover)
- textbfVisual Studio Code with Coq extension

6. Repository Link

The complete code, proofs, evaluator, and this document are available on GitHub:
`https://github.com/mufithamajeed/FormalProofs-coq`

References

- Bertot, Yves, and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions.* Springer Science & Business Media, 2013.
- Coq Development Team. *The Coq Proof Assistant Reference Manual.* Available online: `https://coq.inria.fr/documentation`
- Pierce, Benjamin C. *Software Foundations.* Electronic textbook. Available online: `https://softwarefoundations.cis.upenn.edu/`