Working Principles Of Proof Assistants And Formalization Of Some Proofs in Agda

Supervisor : K.B manandar

Ashwot Acharya, Bishesh Bohora, Supreme chaudary

May 31, 2025

1 Introduction

In 1998 Thomas C. Hales announced that he had proved the kepler conjecture in 1998 (Solane,1998) However the peer review took over 4 years especially due to the proof being incredibely difficult to check , with over a dozen of mathematician to refree the proof and although they were 99% certain it was not untill the proof became formalized that they were able to completely validate the correctness of the proof, which according to Thomas Hales would have taken 20 person-year of manual work.(Szpiro, G, 2003) Proof assistants helps with formalization of mathematical proofs in computer which enables for their verification digitally. Some exhaustive proofs may contain large number of cases which is not feasible to write and verify manually and requires computer assitance.
Such Proof assistants(systems) work by treating proofs as computational objects and checking them using strict logical rules. To enable this, proofs must be written in a formal language that removes ambiguity and allows machines to interpret them accurately. Underlying this process is a theoretical framework—often type theory—that defines how propositions and proofs relate to each other.
This method bridges programming and logic: writing a proof resembles writing a program, and verifying it is like type-checking. By interpreting logic computationally, proof assistants ensure that every proof is exact, complete, and verifiable by a machine. This project invloves investigating this process. Among all available proof assistants Agda was chosen for this project due to its constructive nature and minimal yet expressive language.

2 Problem statement

Proof assistants are reliable tools for verification of formal mathematical theorems and computer programs. This project aims to analyze the theoretical foundations of such, and investigates the connection between mathematical proofs and computer programs through the Curry Howard Correspondence. On practice, this project will demonstrate the use of Agda, a proof assistant by formalizing some well known proofs and dissect the verification process in Agda, through lens of Martin-Löf Type Theory (MLTT) on which it is based upon.

3 Research objectives

1. To explore the correspondence between Proofs and Programs, Type Theory and Intuitionstic Logic.
2. To investigate a current exsisting proof assistant (Agda) and identify the mathematics behind it.
3. To formalize some selected proofs in Agda and demonstrate verification process.

4 Literature Review

Proof assistant rely extremely heavily on the type of logic it is coded on, for example

5 Methodology

Investigating the Coq and the agda proof assistant and assessing the Logic that each proof assistant use and how the proof assistant use such logic to help check if the proof is valid. We will also be inverstigating how to formalize some proofs in one of the proof assistant, and seeing how the logic holds in formalization of proofs.

6 Expected Outcomes

Learning indepth about type theory, curry-howard correospondance and formalization of some proofs.

7 Significiance

Writing proofs in a formal language introduces more rigor and reduces ambiguity. A formally written proof in proof assitants enables them to be treated as executable code. This allows it to be checked or verified mechanically. This way we can offload trust from a human reader to a computer which applies logic rules without oversight, reducing errors in writing and verifying proofs for mathematical theorems. On the other side, with the same theoretical base, treating programs written in some programming language and verifying them or checking their correctness like proofs introduces a stronger reliablity on them. This makes codes written for sensitive applications secure and assures that they work as intended.

8 Work Plan

Week	Work plan
1	Understanding (Dependent) Type Theory and Proof Theory
2	Understanding the implementation of type theory models in digital proof assistant
3	Understanding Purely functional Programming paradigm and λ-calculus
4	Working Principles of Agda and its core implementation
5	Formalization of some proofs in Agda

9 Refrences

1. Sloane, N.J.A. (1998). Kepler's conjecture confirmed. Nature, 395(6701), pp.435-436 doi:https://doi.org/10.1038/26609.

2. Szpiro, G. (2003). Does the proof stack up? Nature, 424(6944), pp.12–13, doi:https://doi.org/10.1038/424012a.