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

Supervisor: K.B manandar

Ashwot Acharya, Bishesh Bohora, Supreme chaudary

June 25, 2025

1 Introduction

In 1998 Thomas C. Hales announced that he had proved the kepler conjecture (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 Dependent Type Theory 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

Proofs or sometimes called the mathematical Proofs are at the foundation of all of mathematics and no mathematical reason are accepted unless they are proven by mathematical reasoning. The construction of proofs begins with initiation of terms, axioms and may bed dependent on previously proven theorems. Each statement in mathematical proofs must be correct which means to say that each step of the

proof must be logically followed from its preceeding steps. (Grami Ali, 2023)
Proof assistant are comnputer system designed to do mathematics but not in the computational sense but more towards the aspect of reasoning, proving and defining. In proof assistant a user can setuo a mathematical theory, defines its properities and be able to perform logical task on them. Most proof assistant now a days are used my professionals to formalize mathematical theories and prove them which is difficult as it requires adding lot of details for it to be valid in the assisting software. This step of writing proofs in a formal language is called the formalization of proofs. (H Geuvers, 2009)
The earliest work on computer assisted proof dates back to the 1950's and 1960s [Davis,1957; Gilmore, 1960; Davis and Putnam, 1960; Wang, 1960; Prawitz et al., 1960] who were devoted to automated theorem proving. Interest in more interactive arrangement began much later on when full automated system became a very difficult problem than supporting human guided proofs.
AutoMath is considered possibly the earliest of human guided theorem provers which gave rise to various other systems of theorem prover and is the first program to use curry howard isomorphism for encoding of proofs (Harrison, Urban and Wiedijk ,2014)
Curry-Howard isomorphism states that any dertivation written in Intuitionstic Logic can be writing in their correosponding typable λterms. which means that proofs can be represented in programing terms, In short an isomorphism can be made between programs and logical proofs (Sørensen, M.H. and Urzyczyin, 2006). Purely functional programming languages are based on λ-calculus (H.P Barendregt, 1990), this enables computability of forementioned derivations or proofs.

5 Methodology

In this project we will be investigating the Coq and the agda proof assistant and assessing the Logic that each proof assistant use and how 0roof 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.435436 doi:https://doi.org/10.1038/26609.

2. Szpiro, G. (2003). Does the proof stack up? Nature, 424(6944), pp.123, doi:https://doi.org/10.1038/424012a

3. Grami, A. (2022). Proof Methods. Elsevier eBooks, [online] pp.5565 doi:https://doi.org/10.1016/b978-0-12-820656-0.00004-6.

4. Geuvers, H. (2009). Proof assistants: History, ideas and future. Sadhana, 34(1), pp.325 doi:https://doi.org/10.1007/s12046-009-0001-5.

5. Harrison, J, Urban, J. and Wiedijk, F. (2014). History of Interactive Theorem Proving. North-Holland, pp.135214. doi:https://doi.org/10.1016/B978-0-444-51624-4.50004-6

6. Sørensen, M.H. and Urzyczyin, P. (2006). The Curry-Howard isomorphism. Studies in Logic and the Foundations of Mathematics, pp.77101. doi:https://doi.org/10.1016/s0049-237x(06)80005-4.

7. BARENDREGT, H.P. (1990). Functional Programming and Lambda Calculus. Formal Models and Semantics, pp.321363 doi:https://doi.org/10.1016/b978-0-444-88074-1.50012-3.