## Working Principles Of Proof Assistant



And Formalization Of Some Proofs In Agda

Ashwot Acharya, Bishesh Bohora, Supervisor: Mr K.B Manandhar Supreme Chaudhary

Kathmandu University

### Proof Assistants

#### What are proof assistant

Proof Assistants
What are proof
assistant
Why digital
verification is
needed?

Logical foundation

Architecture of proof assistant

Comparativ Study

Formalization Of Some proofs

proof assistant, are software more specifically a type of programming language thats allows us to formalize mathematical proofs in computer for digital verification.



# Proof Assistants What are proof assistant Why digital verification is needed?

Logical foundation

Architecture of proof assistant

Comparative Study

Formalization Of Some proofs

### Need of digital verification



- faster computation for complex problems
- many exceptional cases can be explored which would take mathematicians long time

ex: The Kepler Conjecture's proof , which was so complex that verifying it manually would take 20 person-years, but proof assistants made this verification feasible and fast.

Logical foundation

#### Proof Assistants

#### Logical foundation

Naturl deduction Ins  $\lambda$ -Calculus

Architecture of proof assistant

Comparative Study

Formalization Of Some proofs

- $\diamond$  Based on logic (natural deduction, intuitionistic logic),  $\lambda$ -calculus, and type theory.
- ⋄ Curry–Howard Correspondence:

Propositions  $\leftrightarrow$  Types Proofs  $\leftrightarrow$  Programs

 Dependently Typed Languages: Types can depend on values, enabling encoding of properties and proofs.

#### **Natural Deduction**

Proof Assistants
Logical

foundation
Naturl deduction

Ins  $\lambda$ -Calculus

Architecture of proof assistant

Comparative Study

- ♦ **Natural Deduction** is a rule-based system for deriving conclusions from assumptions in logic.
- Instead of using exhaustive truth tables, proofs are built step-by-step using inference rules.
- $\diamond$  Example: Proving from  $A \land (A \rightarrow \bot)$  that  $\bot$  (contradiction) can be derived.
- Basis for how proof assistants check the logical structure of proofs.



#### Intuitionistic Logic

**Proof Assistants** 

Logical foundation Naturl deduction Ins

Architecture of proof assistant

Comparative Study

- Intuitionistic Logic formalizes constructive mathematics, where a statement is only true if a proof can be constructed.
- Omits some classical logic principles, such as the Law of Excluded Middle.
- Stronger requirement: to prove existence, a method or algorithm must be given.
- Proof assistants leverage this constructive approach for digital verification.



#### $\lambda$ -Calculus and Type Theory

**Proof Assistants** 

Logical foundation Naturl deduction Ins

 $\lambda$ -Calculus

Architecture of proof assistant

Study

Formalization Of Some proofs

- $\diamond$   $\lambda$ -Calculus: A foundational system for defining and applying functions using abstraction and application.
- ♦ Type Theory: Assigns types to every term; ensures correctness of operations.
- **⋄ Curry–Howard Correspondence**:

Propositions  $\leftrightarrow$  Types Proofs  $\leftrightarrow$  Programs

 Dependent types allow types to depend on values, expressing complex logical properties.



## Architecture of proof assistant

#### **Architecture of a Proof Assistant**

100 PO 10

Logical foundation

Proof Assistants

Architecture of proof assistant

Study

- ♦ **Kernel**: Minimal, trustworthy codebase enforcing logical rules and validating proofs.
- ♦ **Tactic Engine**: Helps build and automate proofs step by step.
- Formal Proof Language: Rigorously expresses definitions, statements, and proofs.
- Libraries: Collections of verified mathematical foundations for reuse.
- User Interface: IDEs and plugins for interactive, efficient proof development.

Comparative Study

Logical foundation

Architecture of proof assistant

Study Study

Agda Rocq (Coo Lean



- Dependently typed functional programming language and proof assistant.
- No separate tactic layer—proofs are constructed explicitly and directly.
- Uses bidirectional type checking, minimal kernel, and integrates normalization.
- ♦ Emphasizes explicitness and formal clarity over automation.
- Best suited for foundational work and educational purposes.

Comparation Study

Rocq (Coq) Lean

Formalization Of Some proofs

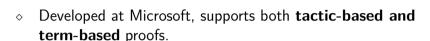
### Rocq (Coq)



- → Tactic-based interactive theorem prover famous for large formalizations (e.g., Four Color Theorem).
- Kernel is based on Calculus of Inductive Constructions (CIC), written in Coq and extracted to OCaml.
- Proofs often built with backward reasoning using tactics—automation is heavily supported.
- Strong type-checking with bidirectional algorithms and conversion checking.
- Widely used for complex, real-world formal verification tasks.

### Comparative Study

Agda Rocq (Coo Lean



- $\diamond$  Kernel based on the Calculus of Inductive Constructions, implemented in C++/C.
- Allows seamless switching between tactic and term styles—user-friendly and flexible.
- Smart elaborator with coercion, backtracking, and overloading.
- Popular in research and education, especially for combinatorial and mathematical formalizations.



Thank you!