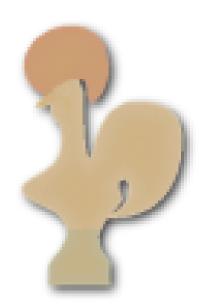
# Coq

An Introduction



#### A Brief General

- Coq is a formal proof management system. It provides a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs.
- Software Foundations (https://softwarefoundations.cis.upenn.edu/)
- Mathematical Components (https://math-comp.github.io/mcb/)
- Standard Library (https://coq.inria.fr/distrib/current/stdlib/)

#### A Bit More

- Coq is a tactic-based proof assistant, which is different from others like Agda, Idris (term-based).
- In the aspect of theory, Coq works within CIC(Calculus of Inducive Constructions), while others like Agda(MLTT), Isabelle(HOL), Mizar(Set Theory).
- Coq is a purely functional programming language, supporting pattern match, higher order function and type polymorphism.
- Coq is a dependent type language similar to Idris and Agda.

#### How to Start

- Don't ask for how much money can you make by learning Coq, which is the most important thing you should know about.
- If you don't learn anything like haskell or ocaml, Software Foundations will be the best choice for you.
- If you are good at math and just curious about interactive theorem proving, Mathematical Components can help you learn Coq-proving quickly.
- Practice more and try to do better!

## **Basic Operations**

Inductive

Definition

Fixpoint

Notation

Theorem/ Lemma/ Example

### Inductive

create type by yourself

### **Definition**

define functions for our types

# **Fixpoint**

deal with the recursive functions

#### Notation

simplify the operations by using some symbols

### Theorem

the theorem we want to prove

# **Begin Proving**

- Proof tells us that the proof begins
- intros helps us import parameters
- When there is no more subgoals, using Qed to tell Coq that the proof comes to an end

### **Basic Tactics**

- simpl
- rewrite
- destruct
- induction
- reflexivity

# simpl

A tactic that can simplify our proof by definitions.

# reflexivity

A tactic checks whether both sides are equal.

### destruct

A tactic used for classification discussion.

### induction

A tactic used when applying mathematical induction.

### rewrite

A tactic that helps us rewrite the proof with what we already know.

### Talk is cheap, show me the code!

Let's open Emacs, try to do it at Proof General!

# Thanks for Listening

