Coq

LASER 2011 Summerschool
Elba Island, Italy

Christine Paulin-Mohring
`http://www.lri.fr/~paulin/LASER`

Université Paris Sud & INRIA Saclay - Île-de-France

September 2011

Lecture 1 : Basics of COQ system

- Covers section 1 and 2 of course notes
- Challenges section 5.1

Outline

- Introduction
- Basics of Coq system
  - First steps in Coq
  - Logical rules and tactics
- Using simple inductive definitions
- Functional programming with Coq
- Automating proofs

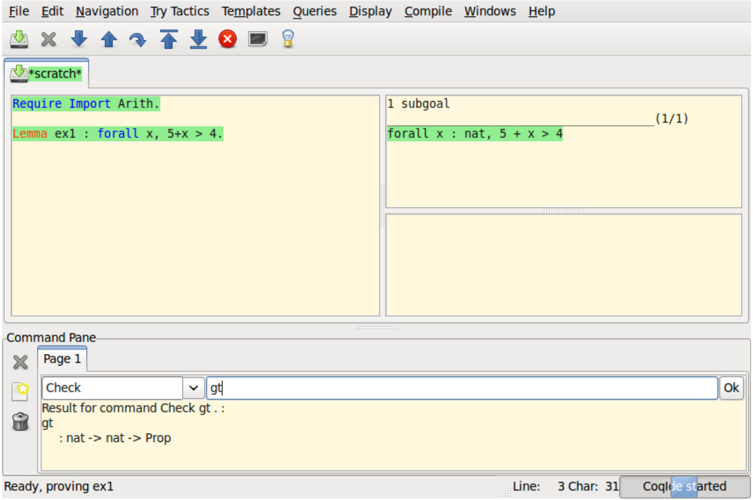
Introduction

- Coq is an interactive system
  it builds libraries of definitions and facts.
- It includes a basic functional programming language.
- It contains standard logical constructions (arithmetic).
- It provides additional tools : queries, libraries, notations . . .

Coq interface

Using `coqide`

alternatives: `emacs+proof-general` or `coqtop`



Coq environment

- A Coq object in the environment has a name.
- A Coq object has a type.

Basic objects in the environment

```
Coq? Check nat.
nat
     : Set

Coq? Check S.
S
     : nat -> nat

Coq? Check plus.
plus
     : nat -> nat -> nat

Coq? Check false.
false
     : bool

Coq? Check pair.
pair
     : forall A B : Type, A -> B -> A * B
```

Basic propositions and proofs

```
Coq? Check False.
False
     : Prop

Coq? Check not.
not
     : Prop -> Prop

Coq? Check and.
and
     : Prop -> Prop -> Prop

Coq? Check eq_refl.
eq_refl
     : forall (A : Type) (x : A), x = x
```

Looking for definitions

```
Coq? Print not.
not =
fun A : Prop => A -> False
     : Prop -> Prop
Argument scope is [type_scope]

Coq? Print nat.
Inductive nat : Set :=  O : nat | S : nat -> nat
For S: Argument scope is [nat_scope]
```

Composed objects

```
Coq? Check (2,true).
(2, true)
     : nat * bool

Coq? Check (1+1).
1 + 1
     : nat

Coq? Check (1+1=3).
1 + 1 = 3
     : Prop
```

Using libraries

Environment is organized in a modular way.

```
Coq? Print Libraries.
Loaded and imported library files:
  Coq.Init.Notations
  Coq.Init.Logic
  Coq.Init.Datatypes
  Coq.Init.Specif
  Coq.Init.Peano
  Coq.Init.Wf
  Coq.Init.Tactics
  Coq.Init.Prelude
  Coq.Init.Logic_Type
Loaded and not imported library files: none

Coq? Locate le.
Inductive Coq.Init.Peano.le
```

Loading libraries

COQ libraries can be compiled and loaded.

```
Coq? SearchAbout le.
le_n: forall n : nat, n <= n
le_S: forall n m : nat, n <= m -> n <= S m
le_ind:
  forall (n : nat) (P : nat -> Prop),
  P n ->
  (forall m : nat, n <= m -> P m -> P (S m)) ->
  forall n0 : nat, n <= n0 -> P n0

Coq? Require Import Arith.

Coq? SearchAbout [le plus minus].
le_plus_minus:
  forall n m : nat, n <= m -> m = n + (m - n)
le_plus_minus_r:
  forall n m : nat, n <= m -> n + (m - n) = m
```

Basic help (summary)

```
Check term.
SearchAbout [idents].
Print ident.
Print LoadPath.
```

Outline

- Introduction
- Basics of Coq system
  - First steps in Coq
  - Logical rules and tactics
- Using simple inductive definitions
- Functional programming with Coq
- Automating proofs

How to prove a new statement ?

- **Goal** *prop*.
- **Theorem** *id*:*prop*.
- **Lemma** *id*:*prop*.

```
Coq? Goal forall A B C, (A->B->C)->(A->B)->(A->C).
1 subgoal

  ============================
   forall A B C : Type,
   (A -> B -> C) -> (A -> B) -> A -> C
```

- tactics transform a goal into a set of subgoals
- solving subgoals is sufficent to solve the original goal
- proof succeeds when no subgoals left

Natural deduction : logical rules for $\to$ and $\forall$

Axiom	$\dfrac{\Gamma \vdash h : A}{\Gamma \vdash ? : A}$	`exact` h
$A \to B$	$\dfrac{\Gamma, h : A \vdash ? : B}{\Gamma \vdash ? : A \to B}$	`intro` h
	$\dfrac{\Gamma \vdash h : A \to B \quad \Gamma \vdash ? : A}{\Gamma \vdash ? : B}$	`apply` h
$\forall x : A, B$	$\dfrac{\Gamma, x : A \vdash ? : B}{\Gamma \vdash ? : \forall x : A, B}$	`intro` x
	$\dfrac{\Gamma \vdash h : \forall x : A, B \quad \Gamma \vdash t : A}{\Gamma \vdash ? : B[x \leftarrow t]}$	`apply` h `with` $(x := t)$

- `intros` multiple introductions
- `apply` $h : \forall x_1 \ldots x_n, A_1 \to \cdots A_p \to C$
  tries infering values of x_i, generates subgoals for A_i.

Example

```
Coq? Goal forall A B C, (A->B->C)->(A->B)->(A->C).

Coq? intros A B C H1 H2 H3.

Coq? apply H1.

Coq? exact H3.

Coq? apply H2.

Coq? exact H3.

Coq? Qed.
```

Logical rules for `False` and $\neg$

`False`	$\dfrac{\Gamma \vdash h : \texttt{False}}{\Gamma \vdash {\color{blue}?} : C}$	`destruct` h
$\neg A$	$\dfrac{\Gamma, h : A \vdash \texttt{False}}{\Gamma \vdash {\color{blue}?} : \neg A}$	`intro` h
$(\equiv A \to \texttt{False})$	$\dfrac{\Gamma \vdash h : \neg A \qquad \Gamma \vdash A}{\Gamma \vdash {\color{blue}?} : C}$	`destruct` h

Exercise

```
Coq? Goal forall A, ~~~ A -> ~A.
```

Logical rules for $\wedge$, *exists*

$A \wedge B$	$$\frac{\Gamma \vdash ? : A \quad \Gamma \vdash ? : B}{\Gamma \vdash ? : A \wedge B}$$	`split`
	$$\frac{\Gamma \vdash h : A \wedge B \quad \Gamma, h_A : A, h_B : B \vdash ? : C}{\Gamma \vdash ? : C}$$	`destruct` h `as` (h_A, h_B)
$\exists x : A, B$	$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash ? : B[x \leftarrow t]}{\Gamma \vdash ? : \exists x : A, B}$$	`exists` t
	$$\frac{\Gamma \vdash h : \exists x : A, B \quad \Gamma, x : A, h_B : B \vdash ? : C}{\Gamma \vdash ? : C}$$	`destruct` h `as` (x, h_B)

Logical rules for $\vee$

$A \vee B$	$$\frac{\Gamma \vdash ? : A}{\Gamma \vdash ? : A \vee B}$$	`left`
	$$\frac{\Gamma \vdash ? : B}{\Gamma \vdash ? : A \vee B}$$	`right`
	$$\frac{\Gamma \vdash h : A \vee B \quad \Gamma, h_A : A \vdash ? : C \quad \Gamma, h_B : B \vdash ? : C}{\Gamma \vdash ? : C}$$	`destruct` h `as` $[h_A \vert h_B]$

Exercise

```
Coq? Goal forall A B, A \/ B -> ~ (~ A /\ ~ B).
```

Other useful tactics

Forward reasoning

$$\frac{\Gamma \vdash ? : B \qquad \Gamma, h : B \vdash ? : A}{\Gamma \vdash ? : A}$$	`assert` $h : B$

Combining tactics with tacticals

t_1 `;` t_2	t_1 then t_2 on generated subgoals
t_1 `||` t_2	t_1 or else t_2
`try` t	does nothing when t fails
`repeat` t	repeats t until it fails

More tactics

Shortcuts, avoiding explicit naming

`assumption`	`exact` on one of the goal hypothesis
`contradiction`	one of the hypothesis is `False`
`exfalso`	assert `False` and use it to solve the current goal

Automation

`tauto`	propositional tautologies
`trivial`	try very simple lemmas to solve the goal
`auto`	search in a base of lemmas to solve the goal
`intuition`	remove the propositional structure before auto
`omega`	solve goals in linear arithmetic

About classical logic

- Both $A \vee B$ and $\exists x : A, B$ have a strong computational meaning.
- From $\vdash \exists x : A, B$, one can compute t such that $\vdash B[x \leftarrow t]$
- $A \vee B$, is equivalent to:
  $\exists b : \texttt{bool}, b = \texttt{true} \rightarrow A \wedge b = \texttt{false} \rightarrow A$

Classical reasoning

- Add an axiom:

```
Coq? Require Import Classical.

Coq? Check not_all_not_ex.
not_all_not_ex
     : forall (U : Type) (P : U -> Prop),
       ~ (forall n : U, ~ P n) ->
       exists n : U, P n

Coq? Print Assumptions not_all_not_ex.
Axioms:
classic : forall P : Prop, P \/ ~ P
```

- Use classical forms of propositions:

```
Coq? Definition class (A : Prop) : Prop := ~ ~ A -> A.

Coq? Definition orc (A B:Prop) : Prop := ~ (~ A /\ ~ B).
```

Basic objects

- sorts: $\mathsf{Prop} : \mathsf{Type}_1$, $\mathsf{Type}_i : \mathsf{Type}_{i+1}$
- variables
- one product: $\forall x : A, B$ (includes $A \to B$)
- function abstraction: $\textbf{fun}\ x : A \Rightarrow t$
- function application: $t\ u$

Computation:

- $(\textbf{fun}\ x : A \Rightarrow t)\ u \equiv t[x \leftarrow u]$

Basic rules

Environment ($\Gamma\vdash$) :

$$\frac{}{[]\vdash} \qquad \frac{\Gamma\vdash A : s}{\Gamma, x : A\ \vdash}$$

Constant and variable:

$$\frac{\Gamma\ \vdash}{\Gamma\vdash s_1 : s_2} \qquad \frac{\Gamma\ \vdash \quad (x, A)\in\Gamma}{\Gamma\vdash x : A}$$

Product:

$$\frac{\Gamma\vdash A : s_1 \quad \Gamma, x : A\vdash B : s_2}{\Gamma\vdash \forall x : A, B : s_2}$$

Function:

$$\frac{\Gamma, x : A\vdash t : B}{\Gamma\vdash \textbf{fun}\ x : A\Rightarrow t : \forall x : A, B} \qquad \frac{\Gamma\vdash t : \forall x : A, B \quad \Gamma\vdash u : A}{\Gamma\vdash t\,u : B[x\leftarrow u]}$$

Computation:

$$\frac{\Gamma\vdash t : A \quad \Gamma\vdash B : s \quad A\equiv B}{\Gamma\vdash t : B}$$

Coq theory

Local definitions

let $x := e$ **in** t

Inductive definitions

See next lecture

Modules

independant level

Examples of terms and types in Coq

- $A \to B$ stands for $\forall x : A, B$ when x not free in B

Examples:

- $\forall A : \mathrm{Prop}, A$ of type Prop (corresponding to $\bot$).
- Programming propositions:
  fun $A\ (P : A \to Prop)\ (x : A) \Rightarrow P\ x \to \texttt{False}$
  of type $\forall A, (A \to \mathrm{Prop}) \to (A \to \mathrm{Prop})$.
- Writing proof-terms:
  fun $A\ B\ C\ (h_A : A \to B)(h_B : B \to C)(x : A) \Rightarrow h_B\ (h_A\ x)$
  of type $\forall A\ B\ C\ (A \to B) \to (B \to C) \to A \to C$

Summary

- Coq provides multiple libraries
- lemmas can be combined using simple rules
- intuitionistic versus classical logic
- functional terms to represent proofs
- simple exercises : file `propositional.v`