FREE eBook

LEARNING

coq

Free unaffiliated eBook created from
Stack Overflow contributors.

#coq

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: coq

It is an unofficial and free coq ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official coq.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with coq

Remarks

This section provides an overview of what coq is, and why a developer might want to use it.

It should also mention any large subjects within coq, and link out to the related topics. Since the Documentation for coq is new, you may need to create initial versions of those related topics.

Examples

Testing and installating Coq

Testing without installing

For new users who wish to start testing Coq without installing it on their machine, there is an online IDE called JsCoq (presentation here). The package sub-window allows testing various well-known additional packages.

Installation

The download page contains installers for Windows and MacOS.

Users of Linux are generally advised to compile from source using opam, in order to get the latest version. Basic instructions on how to do so are given here.

A simple proof

```
Theorem my_first_theorem : 1 + 1 = 2.
Proof.
  reflexivity.
Qed.
```

Try it in your browser.

Install Coq on MacOS

You can install the whole bundle by downloading the dmg package from here.

The bundle contains a CoqIDE that can be used for writing your proofs or you can use `coqtop` command to run the interpreter on your terminal

Installation of Coq on MacOS is easy using homebrew as well

```
brew install coq
```

or if you use MacPorts

```
sudo port install coq
```

There is no good vi support for Coq. You can use Proof General within emacs which has a good usability.

To install Proof General remove old versions of Proof General clone the new version from GitHub

```
git clone https://github.com/ProofGeneral/PG ~/.emacs.d/lisp/PG
cd ~/.emacs.d/lisp/PG
make
```

Then add the following to your .emacs:

```
;; Open .v files with Proof General's Coq mode
(load "~/.emacs.d/lisp/PG/generic/proof-site")
```

Make sure that the emacs you are running the actual Emacs. If you face version mismatch problems you might have to run makefile again specifying Emacs path explicitly

```
make clean; make EMACS=/Applications/Emacs.app/Contents/MacOS/Emacs
```

Installation with Nix

Warning: this is not the standard way of installing Coq.

For users of Linux (and MacOS) who wish to gain access to up-to-date versions of Coq or to be able to use several versions of Coq on the same machine, without the hassle of using opam, and without having to compile from source, this is an alternative solution.

Nix is a package manager for Unix-type OS such as Linux and MacOS. It comes with its own collection of packages which is generally kept much more up-to-date than Debian's or Ubuntu's. It does not conflict with your distribution's package manager because it does not install anything in `/usr/bin` and such.

First, you need to install Nix:

```
$ curl https://nixos.org/nix/install | sh
```

To ensure that the necessary environment variables are set, either log in again, or type:

```
. $HOME/.nix-profile/etc/profile.d/nix.sh
```

Then the following command will install the latest version of Coq:

```
$ nix-env -iA nixpkgs.coq_8_6
```

You can also run CoqIDE without adding anything to your PATH:

```
$ nix-shell -p coq_8_6 --run coqide
```

Similarly (supposing you already have Emacs and Proof-General installed):

```
$ nix-shell -p coq_8_6 --run emacs
```

This is very useful to run different versions when you need them. For instance, to run Coq 8.5 use the following command:

```
$ nix-shell -p coq_8_5 --run coqide
```

Example of a proof by induction

Here is a simple proof by induction.

```
Require Import Coq.Setoids.Setoid.
Require Import Coq.Arith.Lt.

(* A number is less than or equal to itself *)
Theorem aLTEa : forall a,
    a <= a.
    auto with arith. (* This follows by simple arithmetic *)
    Qed.

Theorem simplALTE : forall a b,
    S a <= S b <-> a <= b. (* If a <= b, then a + 1 <= b + 1 *)
Proof.
Admitted.

Theorem ltAlwaysLt: forall a b,
    a <= a + b.
Proof.
  intros. (* Introduce relevant variables *)
  induction a, b. (* Induction on every variable *)
  simpl. apply aLTEa. (* 0 <= 0 + S b *)
  rewrite -> plus_O_n. auto with arith. (* 0 <= S b *)
  rewrite <- plus_n_O. apply aLTEa. (* S a <= S a + 0 *)
  rewrite <- simplALTE in IHa. (* IHa: a <= a + S b. Goal: S a <= S a + S b. *)
  apply IHa. (* We rewrote the induction hypothesis to be in the same form as the goal, so it
applies immediately now *)
  Qed.
```

Read Getting started with coq online: https://riptutorial.com/coq/topic/3762/getting-started-with-coq

Chapter 2: Searching for an existing fact with Search and variants

Syntax

- `Search qualid.` (* for Coq 8.4 and newer versions *)
- `SearchAbout qualid.` (* deprecated synonym. *)

Parameters

Parameter	Description
`qualid`	The identifier or pattern to search for. It can involve notations

Remarks

Before Coq 8.4, `Search` had the meaning of the current `SearchHead`: only search for facts where the pattern matches in the conclusion of the statement.

Examples

Facts about a particular identifier

To see all the facts involving the `le` relation from the prelude:

```
Coq < Search le.
le_n: forall n : nat, n <= n
le_S: forall n m : nat, n <= m -> n <= S m
...
max_l: forall n m : nat, m <= n -> Nat.max n m = n
max_r: forall n m : nat, n <= m -> Nat.max n m = m
...
```

To search about all the facts involving the notation `<`:

```
Coq < Search "<".
exists_lt: forall (Q : nat -> Prop) (k l : nat), exists_between Q k l -> k < l
in_int_intro: forall p q r : nat, p <= r -> r < q -> in_int p q r
in_int_lt: forall p q r : nat, in_int p q r -> p < q
...
```

Searching for a pattern

Search for all facts involving a pattern in an hypothesis or conclusion:

```
Coq < Search (_ + O).
plus_n_O: forall n : nat, n = n + 0
```

The _ character serves as a wildcard, it can be used multiple times:

```
Coq < Search (S _ <= _).
le_S_n: forall n m : nat, S n <= S m -> n <= m
le_n_S: forall n m : nat, n <= m -> S n <= S m
```

You can also search for non-linear patterns:

```
Coq < Search (?x <= ?x).
le_n: forall n : nat, n <= n
```

Searching for a pattern in the conclusion of a lemma

Search for a lemma when you know what its conclusion ought to be:

```
Coq < SearchPattern (S _ <= _).
le_n_S: forall n m : nat, n <= m -> S n <= S m
```

You can also search on a partial conclusion (the conclusion and one or several last hypotheses).

```
Coq < Require Import Arith.
Coq < SearchPattern (?x <= ?y -> ?y <= _ -> ?x <= _).
Nat.le_trans: forall n m p : nat, n <= m -> m <= p -> n <= p
```

Warning: if you mix up the order of the hypotheses, you won't find anything:

```
Coq < SearchPattern (?y <= _ -> ?x <= ?y -> ?x <= _).
```

Read Searching for an existing fact with Search and variants online:
https://riptutorial.com/coq/topic/4007/searching-for-an-existing-fact-with-search-and-variants

Chapter 3: Using Tactics

Introduction

This section includes information on how to use various Coq tactics and techniques (case analysis, proof by induction, auto, etc.) to prove theorems.

Examples

Trivial example of a case analysis

In Coq, `destruct` more or less corresponds to a case analysis. It is similar to induction except that there's no induction hypothesis. Here is a (admittedly rather trivial) example of this tactic:

```
Require Import Coq.Arith.Lt.

Theorem atLeastZero : forall a,
0 <= a.
Proof.
  intros.
  destruct a. (* Case analysis *)
  - reflexivity. (* 0 >= 0 *)
  - apply le_0_n. (* S a is always greater than zero *)
  Qed.
```

Read Using Tactics online: https://riptutorial.com/coq/topic/8335/using-tactics

Credits

S. No	Chapters	Contributors
1	Getting started with coq	Anton Trunov, Apoorv Ingle, bukzor, Community, EJoshuaS, Zimm i48
2	Searching for an existing fact with Search and variants	pintoch, Tej Chajed, Zimm i48
3	Using Tactics	EJoshuaS, Zimm i48