Coq cheatsheet

General remarks

- $\neg P$ ("not P") is equivalent to $P \to$ `False`.

Simple commands

- `Proof`: Begin proving a theorem.
- `reflexivity`. Prove a goal of the form $a = a$.
- `contradiction`. Prove any goal given two logically contradictory hypotheses.
  - `contradiction (H2 H1)`. Use this if `H1` : P and `H2` : $\neg P$ are the two contradictory hypotheses. Note that the order matters.
  - `contradiction H`. Use this if `H` is one of the two contradictory hypotheses. (It doesn't matter which one; Coq will search for the other one.)
  - `contradiction`. If you don't specify either hypothesis, Coq will search for both.
- `contradiction H`. Prove any goal if `H` is `False`.
- `exact H`. Prove a goal that matches `H` exactly.
  - `exact I`. Prove a goal that equals `True` (the keyword `I` in Coq is the proof of `True`).
- `unfold` *name*. Use a definition in the goal.
  - To find the name of a definition, use `Locate`, e.g. `Locate "-"`. (with the quotes and period)
  - `unfold` *name* `in H`. Use a definition in `H`.
- `admit`. Assume a goal without finishing its proof.
- `Admitted`. Assume a theorem without finishing its proof.
- `Qed`. Finish the proof of a theorem. All goals must be achieved before you can use `Qed`.

Manipulating goals

- `intros`. If your goal is $\forall\ n,\ P \to Q$, introduce a variable n and a hypothesis P, and change your goal to Q.
  - Usually you can start your proofs with `intros`.
- `intro H`. Introduce one hypothesis at a time.
- `split`. Your goal is $P \wedge Q$, and you want to first prove P, then prove Q.
- `left`. Your goal is $P \vee Q$, and you want to prove P.
- `right`. Your goal is $P \vee Q$, and you want to prove Q.
- `exists x`. Your goal is $\exists\ n, P$, and you plan to prove that the choice $n = x$ satisfies P.
- `absurd P`. Use this if you think you can complete the proof by proving a contradiction $P \wedge \neg P$. You will be asked to prove $\neg P$, and then asked to prove P.
  - If you think you can prove `False` directly, use `exfalso` instead of `absurd False`.
- `assert P`. Replace the current goal with P. After proving P, it appears as a new hypothesis, and you return to proving your original goal.

Manipulating equalities

- `rewrite H`. Given `H` : $a = b$, if your goal contains a, replace a with b in your goal.
  - `rewrite H at 2`. Use this variant if your goal has more than one occurence of a, and you want to replace only the second occurrence.
- `rewrite <-H`. Given `H` : $a = b$, if your goal contains b, replace b with a in your goal.
- `rewrite H1 in H2`. Given `H1` : $a = b$ and another hypothesis `H2` that contains a, replace a with b in `H2`.
  - `rewrite <-H1 in H2`. Replace b with a in `H2`.
- `replace a with b`. If your goal contains a, replace a with b in your goal.
  - If you don't already have a hypothesis stating $a = b$ or one stating $b = a$, you will have to come back later and prove $a = b$.
- `symmetry`. If your goal is $a = b$, change it to $b = a$.
- `symmetry in H`. Change $a = b$ to $b = a$ in a hypothesis.

Coq does have a built-in theorem for the transitive property of equality (namely, `eq_trans`), but it's usually easier just to use `rewrite` and `replace`.

Manipulating hypotheses

- `apply H`. Given `H` : $P \to Q$, if your goal is Q, change your goal to P.
  - `apply (H x y)`. You have `H` : $\forall\ m\ n, P \to Q$, and you want to use `H` with $m = x$ and $n = y$.
- `apply H1 in H2`. Given `H1` : $P \to Q$ and `H2` : P, change `H2` to Q.
  - These can be combined: `apply (H1 x H2) in H3`.
- `destruct H`. (Always replaces `H`.)
  - Given `H` : $P \wedge Q$, replace `H` with two new hypotheses P and Q.
  - Given `H` : $P \vee Q$, replace `H` first with P (and prove your goal), and then with Q (and prove your goal again).
  - Given `H` : $\exists\ n, P$, create a new variable x and replace `H` with a new hypothesis stating that P holds for x.
- `case H`. (Never replaces `H`.)
  - Given `H` : $P \vee Q$ and a goal R, replace the goal with $P \to R$ (which you must prove), followed by $Q \to R$ (which you also must prove).
- If `H` is `False`, then `destruct H` or `case H` immediately proves the current goal. (`contradiction H` also works.)

Creating new hypotheses

The `pose proof` command allows you to create new hypotheses out of existing hypotheses and/or previously proved theorems.

- `pose proof Theorem`. Create a new hypothesis using the previously proved theorem named `Theorem`.
  - Example: `pose proof (eq_refl a)` creates a new hypothesis $a = a$.
- `pose proof (H x)`. Given `H` : $\forall\ m, P$, create a new hypothesis stating that P holds for x.
- `pose proof (H1 H2)`. Given `H1` : $P \to Q$ and `H2` : P, create a new hypothesis Q.

These can be combined. For example, suppose we have

$$\begin{array}{rcl} x & : & \mathrm{Z} \\ \mathrm{H1} & : & \forall\ n, n = 1 \to n + 1 = 2 \\ \mathrm{H2} & : & x = 1 \\ \mathrm{H3} & : & \forall\ n, n + 1 = 2 \to n + 2 = 3 \end{array}$$

Then `pose proof (H3 x (H1 x H2))` yields H4 : $x + 2 = 3$. (Alternatively, `apply (H1 x) in H2` followed by `apply (H3 x) in H2` yields a similar but not identical configuration.)

Law of the excluded middle

If you need to do case analysis on an arbitrary statement P, the command `pose proof (classic P)` produces a new hypothesis `H` which states $P \vee \neg P$. You can then do `case H`, `destruct H`, etc. as needed.

Example ("Not Not P implies P" a.k.a. `NNPP`):

```
Require Export Classical_Prop Utf8.
Notation "x → y" := (x → y) (at level 99).

Theorem NNPP :  ∀ P : Prop, ¬¬ P → P.
Proof.
  intros.
  pose proof (classic P) as H0.
  destruct H0.
  - exact H0.
  - contradiction.
Qed.
```

(The package `Classical_Prop` must be loaded in order to use the law of the excluded middle. This particular theorem cannot be proved without the law of the excluded middle.)

Braces and Bullets

The proof of `NNPP` above makes use of *bullets*, which are an optional (but very useful) feature to help you organize your proofs. Bullets (once used) are not merely decorative; if you use them, Coq keeps track of indentation levels and forces you to finish one block before beginning another.

For more information about bullets and braces (a related feature), see `http://prl.ccs.neu.edu/blog/2017/02/21/bullets-are-good-for-your-coq-proofs/`.