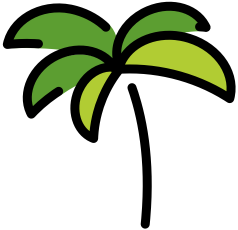
Formalizing and Proving Theorems in Coq — Lecture 3

Tobias Kappé

Today's lecture

You will learn how to. . .

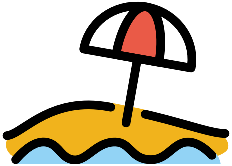
- Use Coq's *tacticals* to ease proof writing.
- Use some of Coq's built in automation tactics.
- Use *evars* to discover parameters on-the-fly.



Taking it easy

The `easy` tactic closes easy proofs where, for instance

- the goal is already given as a hypothesis; or
- your assumptions contain a contradiction ("ex falso").



Instead of `foo. easy.` you can also write `now foo.`

Example: revisit `add_zero_right`.

Semicolons and you

The ; (semicolon) operator combines tactics.

- `foo; bar.` runs `bar` on *all* goals resulting from `foo`.
- Goals resolved by `bar` will disappear.
- Useful if all your subproofs start with, e.g., `simpl`.

Example: revisit `add_succ`.

If at first you don't succeed

The `try` tactical suppresses errors.

- `try foo.` runs `foo`, and does nothing in case of an error.
- Most useful when combined with the semicolon operator.

Example: re-revisit `add_succ`.

Cleaning up `add_commute`

Group exercise:

- Form groups of two (maybe three).
- Compare your proofs for `add_commute`.
- Simplify them using the tactics you saw.

} 10–15 minutes

Stop `repeating` yourself

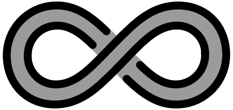
The `repeat` tactical runs a tactic until it fails.

- Very useful to converge on some desired goal.
- Example: re-associating brackets to one side.
- Can be combined with, e.g., `now`.
- What if the tactic fails immediately?
  Nothing happens!

Tactics as programs

Think of proofs scripts as little programs.

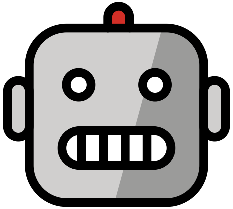
- If you are not careful, they can get stuck!
- In bad cases, this grinds your machine to a halt.
- Example: `repeat` that keeps on going.
- Workaround: the `Timeout` vernacular.



Automation

Some goals can be proved automatically.

- Mainly statements in simple logical fragments.
- Typically reached at the end of a goal.



Go with your `intuition`

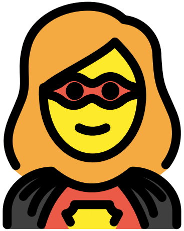
The `intuition` tactic clears intuitionistic tautologies.

- Implements decision procedure for intuitionistic propositional logic.
- Example: almost all goals from the first lecture.
- What about properties that need classical reasoning?

A `valiant` ally

The `lia` tactic clears **l**inear **i**nteger **a**rithmetic goals.

- Must use Coq's built-in `nat` (not our own!)
- Must also import `Coq.micromega.Lia` first.
- Can handle almost all goals from previous lecture.
- Very useful to clear boring goals about `nat`.



Searching for an answer

The Search vernacular helps you find lemmas.

- Saves you from having to write your own.
- Takes queries that contain wildcards.
- Example: reworking fib_multiply.
- How to find a commutativity lemma?

Unification variables

Writing arguments for `apply` can be tedious.

- Use `evars`: placeholders for arguments.
- Refinement as other tactics are applied.
- Example: `less_than_equal_mono_add`.

Next lecture

- More about adding custom notation.
- Rewriting using setoids.

Homework:

- Revisit your proofs in `lecture-3a.v`.

Emojis designed by OpenMoji (CC BY-SA 4.0)