Introduction to Rocq — Arthur Azevedo de Amorim

Lecture 2 - *June 22, 2026*

This lecture walks through `sorting.v` and discusses using libraries and proving properties about data structures and algorithms.

1 Notations & Searching

Rocq libraries usually come with a lot of predefined notations. If not familiar with them, one can choose to turn notations off by the command `Unset Printing Notations`.

The `Search` command can be used inside Rocq to lookup definitions and theorems that are relevant to the provided name or notation.

2 Tactics

This lecture introduces several new tactics, and some new usages of known tactics.

`lia`. Linear integer arithmetics, often useful for dealing with math-related goals.

`auto`, `eauto`. They discharge simple goals, and also conduct proof search using registered databases of theorems. `eauto` is more powerful (and more demanding) by allowing existential holes in searching.

`destruct` can work on an expression, but the expression itself will be erased after destruction. An `eqn:Hxxx` clause can be added to keep `Hxx` recording the equality between the original expression and the destructed values.

The `rewrite` used in this lecture comes from SSReflect. There are some different notations:

- `-H` for rewriting `H` in the opposite direction.
- `!` prefix for repeatedly applying the rewrite.

Compiled By:
Bas Laarakker, Songlin Jia, Ulises Torrella

- `[...]` annotation for constraining rewrite locations.

`assert` is used to define a local, temporary hypothesis. The assertion has to be proven first, as an inserted subgoal. The defined hypothesis can then be used in the subgoal where it is originally asserted. Think of `assert` as if defining a lemma, but can refer to local states and maybe not reusable as a lemma should be.

3 Working with Data Types

Rocq has `true/false` as bool and `True/False` as props. Usually, `true/false` are to be computed by algorithms, and `True/False` should require manual proving. There are lemmas reflecting between boolean predicates and propositional predicates, *e.g.,* `Nat.leb_le`. Boolean predicates in Rocq usually have their names ending in `b` and notations ending in `?`.

stdpp has defined the `elem_of` operation for many data types, including lists. For goals/hypotheses in the form of $\forall x \in l, ...$, the convention is to convert the primitive `forall` into the library defined `Forall` for easier interaction.

4 Extraction

Rocq allows to extract definitions into OCaml, so that the verified soundness can apply to production code, modulo some increase in the trust base. To do so, users have to provide the denotation for used inductive data types.