Generalized Rewriting in Lean

Jovan Gerbscheid

Lean workshop Reykjavik 02-10-2025

Motivation

In my work I try to improve the user experience of theorem proving in Lean

- Unlike FRO, I try to make mathematicians happy
- Tactics
- Elaboration/delaboration
- Performance improvements

The `grw` tactic

`grw` works the same as `rw`, but using any relation, like

- `≤`/`<`
- `≡ [ZMOD n]`, congruence modulo `n`
- `⊆`, subset
- `→`, implication
- `∣`, divisibility
- `=ᶠ[ae μ]`, almost everywhere equality w.r.t. a measure `μ`

Demo

live-lean demo

The `gcongr` tactic

- Implemented in 2023 by Heather Macbeth
- Generalization of the congruence tactic, `congr`
- Recursively applies generalized congurence lemmas to goals of the form `F a₁ ... aₙ < F b₁ ... bₙ`
- `· < ·` can be any relation, including `· → ·`

How does `grw` work?

- `grw` decides where to rewrite in the same way as `rw`
- Given a proof of `a < b` and goal `p[a]`
  `grw` proves `p[b] → p[a]` using `gcongr`.
- Given a proof of `a < b` and hypothesis `p[a]`
  `grw` proves `p[a] → p[b]` using `gcongr`.
- `grw` then applies this to turn `p[a]` into `p[b]`.

Extensibility

To extend `grw`/`gcongr` to support a definition:

- Monotonicity lemmas/generalized congruence lemmas need to be tagged with `@[gcongr]`.
- Transitive relations need an `IsTrans` instance.

Possible improvement for `grw`

- A `grw?` suggestion tactic analogous to `rw??`.
- Let `grw` rewrite terms that contain bound variables
  I will implement this once `rw` can do this.
- Let `grw` rewrite inside of `-2 * _`, not just `2 * _`.
- Let `grw` determine which positions are valid,
  instead of rewriting everywhere and hoping for the best.
- Let `grw` change the strictness: when rewriting with `a < b` in `a < c`, replace it with `b ≤ c`.