Automating Proof - Tactics in LEAN

Zachary Banken WXML Spring 2023

Introduction: Mathematicians prove things in Lean by writing a procedure of *tactics*, commands that manipulate the proof state to unify hypotheses with goals. These tactics are themselves written in the Lean language, and can greatly reduce the effort of proving a theorem.

Functional Programming:

In Lean, a proposition is a function from hypotheses to goals, and a proof is a closed term of this type. For example, a proof of p -> p is `(λ hp : p, hp)`.

Monads

Monads capture the idea of effects or state in functional programming. A monad is a wrapper for a type that adds additional information (e.g. the proof state).

Wrapping into a monad

```
pure : α -> m α
```

Binding monads

```
bind :
m α -> (α -> m β) -> m β
```

Metaprogramming in LEAN:

Metaprogramming is writing Lean programs that manipulate Lean terms (e.g. simplifying).

The `tactic` monad

The `tactic _` type is a map from the proof state to a new proof state and a term of type `_`.

`do` Blocks

Composing monads can be abstracted to an imperative-like procedure with the `do` syntax. For example,

`do tgt ← target,` is syntactic sugar for

```
bind target (λ tgt, …)
```

Pattern matching

Pattern matching is used in tactics to access terms in expressions, and to dictate control flow. The code below prints the value of `a`:

```
do `(%%a + %%b = %%c) ← target,
trace a
```

Example Code:

This is Lean code for a tactic that closes the goal if it is a hypothesis.

```
meta def assump : tactic unit
:=
do ctx ← local_context,
  ctx.mfirst (λ e, exact e)


example (P Q : Prop)
(hp : P) (hq : Q) : P :=
by assump
```

Check out more on the computers!

References:
[1] Avigad et al. Theorem Proving in Lean, 2023.
[2] Rob Lewis. Metaprogramming in Lean Tutorials, 2020.
[3] Baanen et al. The Hitchhiker's Guide to Logical Verification, 2021.