Lab 13: Proving properties about programs

Functional Programming (ITI0212)

This week, we explored how to prove properties about programs using two styles: term mode (constructing terms directly) and tactic mode (guiding the proof interactively with steps).

In term mode, we build proof terms just like writing functions — this style is compact but can become hard to manage for complex proofs. Tactic mode, on the other hand, lets us build proofs incrementally by applying small steps such as `intro`, `apply`, and `rw`. It corresponds to building the same term under the hood, but in a way that's easier to follow and debug. We also practiced inductive proofs over natural numbers and lists, learned how to use helper lemmas, and saw how tail-recursive and standard implementations can be proven extensionally equal.

An overview of the tactics we have seen in the lecture:

- `exact` - use this to provide a term that matches the goal
- `apply` - use this to apply a theorem to the goal
- `intro` - use this to introduce a variable into the context
- `induction` - use this to perform induction on a value (think of this as syntactic sugar for `match`-expressions)
- `rw` - use this to rewrite the goal using an equation

Task 1
Use tactic mode to prove the following properties:

```
or_comm : p ∧ q ⇔ q ∧ p
or_assoc : (p ∧ q) ∧ r ⇔ p ∧ (q ∧ r)
```

Task 2
Use tactic mode to prove the following standard facts about addition on natural numbers:

```
zero_add : ∀ (a : Nat), 0 + a = a
add_assoc : ∀ (a b c : Nat), a + (b + c) = (a + b) + c
add_comm : ∀ (a b : Nat), a + b = b + a
```

Task 3
If you're feeling brave (this is an optional task), adapt the proof of `sum_equiv_final` to the `reverse` and `reverse_tailrec` functions. You can find the definitions of these in the lecture script.

The property we are trying to prove is then:
`theorem reverse_equiv_final : reverse = reverse_tailrec`.