02 - Substitution

Lean: First Steps

Tariq Rashid

August 25, 2024

Task

- Let's continue with another simple task. Imagine we have the following formula.

$$y = x + 4$$

- Given $x = 3$, our task is to prove

$$y = 7$$

- Here, x, y, 3 and 4 are all real numbers.

- Easy task - an effort to think about the steps involved.

$$y = x + 4 \qquad \text{given fact} \qquad (1)$$

$$x = 3 \qquad \text{given fact} \qquad (2)$$

$$\begin{aligned} y &= x + 4 && \text{using fact (1)} \\ &= (3) + 4 && \text{substitution using fact (2)} \\ &= 7 && \text{using arithmetic} \qquad \square \end{aligned}$$

- We start by listing the two given facts, $y = x + 4$ and $x = 3$.
- We want to prove something about y. What is y? The first fact tells us $y = x + 4$.
- At this point, we have $y = x + 4$, which is fine, but we do want to resolve that x into a number.
- The second fact tells us $x = 3$. We can use it to substitute 3 for the x in $x + 4$. This gives us $y = (3) + 4$.
- Finally, we can use arithmetic to evaluate $(3) + 4$ as 7. That gives us what we want, $y = 7$.

Maths

- It is this kind of structured step-by-step thinking that we'll need to write proofs in Lean.
- It may seem disproportionate for simple tasks, but it is better to develop that thinking with simple tasks.

Code

```
-- 02 - Simple Proof by Calculation

import Mathlib.Tactic

example {x y : ℝ} (h1 : y = x + 4) (h2 : x = 3) : y = 7 := by
  calc
    y = x + 4 := by rw [h1]
    _ = 3 + 4 := by rw [h2]
    _ = 7 := by norm_num
```

Code

- Lines beginning with `--` are comments.
- `import Mathlib.Tactic` loads into Lean information about fundamental results and common methods used in proofs, **tactics**.
- The next line is the beginning of the proof:
  - Theorem **name**, but `example` creates anonymous theorem.
  - **Variable types** `{x y : ℝ}`.
  - Named **hypotheses** `(h1 : y = x + 4) (h2 : x = 3)`.
  - After `:` is the **objective** or **goal** statement we want to prove, `y = 7`.
  - Finally, `:= by` signals the subsequent code will seek to prove the objective.

Code

name types hypotheses objective

```
example {x y : ℝ} (h1 : y = x + 4) (h2 : x = 3) : y = 7 := by
```

Code

- `calc` tells Lean we intend to do a proof by direct calculation.
- After that is the core of the proof.
  - `y = x + 4` is **justified** using `by rw [h1]`, a tactic for **rewriting** `y` using hypothesis `h1`.
  - Previous expression `x + 4`, denoted by the shorthand `_`, is equal to `3 + 4`. This is **justified** by hypothesis `h2`, which allows us to **rewrite** `x` as `3`.
  - Finally, `y = 7`. Simplification of `3 + 4` to `7` is justified with the `norm_num` tactic, which can do numerical arithmetic.

InfoView

- Lean provides feedback through its **Infoview**, which will appear in a separate pane.
  - Usually to the right of your main code.
- **Updated as we edit our code**, provides **warnings** and **errors**.

InfoView

- With the previous (correct) Lean code, Infoview tells us:

```
All Messages (0)
No messages.
```

- No messages means no warnings or errors. Lean thinks our proof is correct.

InfoView - Deliberate Error

- Change the hypothesis `h2`, used to rewrite `x` as 3, to the incorrect hypothesis `h1`.

- Infoview updates with error message.

```
01_simple.lean:8:24
tactic 'rewrite' failed, did not find instance of the
    pattern in the target expression
  y
```

- At line 8 of the code, Lean found the rewrite tactic had failed.

- Changing the hypothesis back to `h2` sees all error messages go away.

Easy Exercise

- Write a Lean program to prove $y = 0$ given $y = x^2 - 9$ and $x = -3$, where $x, y \in \mathbb{R}$.