Propositional Logic in Lean

Heshan Du
University of Nottingham Ningbo China

September 2024

Aims and Learning Objectives

- To be able to understand the meanings of a proposition, logical connectives, a tautology, and a contradiction;
- To be able to apply the tactics *assume*, *exact*, *apply* in Lean.
- To be able to construct proofs for tautologies in propositional logic using Lean.

Reading

- Thorsten Altenkirch, *Introduction to Formal Reasoning*, 2023.
  - Chapter 2. Propositional Logic
- Kenneth H. Rosen, *Discrete Mathematics and its Applications*, 6th edition, 2007.
  - Chapter 1. The Foundations: Logic and Proofs

Content

- Proposition, propositional variables, logical connectives
- Precedence of logical operators
- Proving tautologies using Lean

What is a proposition?

- A *proposition* is a declarative sentence (i.e., a sentence that declares a fact) that is either true or false, but not both.
- A *proposition* is a definitive statement which we may be able to prove.
- In Lean, we write $P : Prop$ to express that P is a proposition.

Propositional Variables

- *Propositional variables* are variables that represent propositions.
- In Lean, we declare propositional variables as follows:

```
variables P Q R : Prop
```

- This means that P, Q, R are propositional variables which may be substituted by any concrete propositions.

Logical Connectives

We introduce a number of logical connectives and logical constants to construct propositions:

- Implication ($\rightarrow$), read $P \rightarrow Q$ as **if** P **then** Q.
- Conjunction ($\wedge$), read $P \wedge Q$ as P **and** Q.
- Disjunction ($\vee$), read $P \vee Q$ as P **or** Q.
- **false**, read **false** as *Pigs can fly*.
- **true**, read **true** as *It sometimes rains in England*.
- Negation ($\neg$), read $\neg P$ as **not** P. We define $\neg P$ as $P \rightarrow$ *false*.
- Equivalence ($\leftrightarrow$), read $P \leftrightarrow Q$ as P **is equivalent to** Q. We define $P \leftrightarrow Q$ as $(P \rightarrow Q) \wedge (Q \rightarrow P)$.

Logical Connectives in Lean

The Lean commands of logical operators are displayed below.

Operator	Command
$\neg$	`\neg`
$\wedge$	`\and`
$\vee$	`\or`
$\rightarrow$	`\->`
$\leftrightarrow$	`\iff`

Precedence of Logical Operators 1

The precedence of logical operators is displayed below.

Operator	Precedence
$\neg$	1
$\wedge$	2
$\vee$	3
$\rightarrow$	4
$\leftrightarrow$	5

Precedence of Logical Operators 2

- Implication and equivalence bind weaker than conjunction and disjunction. E.g., $P \vee Q \to R$ is read as $(P \vee Q) \to R$.
- Implication binds stronger than equivalence. E.g., $P \to Q \leftrightarrow R$ is read as $(P \to Q) \leftrightarrow R$.
- Conjunction binds stronger than disjunction. E.g., $P \wedge Q \vee R$ is read as $(P \wedge Q) \vee R$.
- Negation binds stronger than all the other connectives. E.g., $\neg P \wedge Q$ is read as $(\neg P) \wedge Q$.
- *Implication is right associative.* E.g., $P \to Q \to R$ is read as $P \to (Q \to R)$.

If in doubt, then one may use parentheses to specify the order in which logical operators are applied.

Tautology, Contradiction and Contingency

- A proposition that is always true, no mater what the truth values of the propositions that occur in it, is called a *tautology*.
- A proposition that is always false is called a *contradiction*.
- A proposition that is neither a tautology nor a contradiction is called a *contingency*.

Tautologies and contradictions are often important in mathematical reasoning.

A Proof in Lean

- In Lean, we write $p : P$ for p proves the proposition P.
- A proof is a sequence of *tactics* affecting the current proof state which is the sequence of assumptions we have made and the current goal.
- A proof begins with *begin* and ends with *end*, and every tactic is terminated with , .

Our First Proof

- A very simple tautology $P \to P$
- E.g., if the sun shines, then the sun shines.
- How to prove it in Lean?

```
variables P : Prop

theorem I: P → P :=
begin

end
```

Initial Proof State

The initial proof state of theorem I is:

```
P : Prop
⊢ P → P
```

This means we assume that P is a proposition and want to prove $P \to P$. The $\vdash$ symbol (pronounced *turnstile*) separates the assumptions and the goal.

Tactics: Assume

- **assume h** means that we are going to prove an implication by assuming the premise and using this assumption to prove the conclusion.
- After *assume h*, the proof state is

  ```
  P : Prop,
  h : P
  ⊢ P
  ```

- This means our goal is now P but we have an additional assumption $h : P$.

Tactics: Exact

- **exact h** means there is an assumption that *exactly* matches the current goal.
- In the first proof, if you move the cursor after *exact h*, you see *no goals*. We are done.

```
variables P : Prop

theorem I: P → P :=
begin
  assume h,
  exact h,
end
```

Using Assumptions

- Another tautology: $(P \to Q) \to (Q \to R) \to P \to R$.
- A translation: *If if the sun shines then we go to the zoo then if if we go to the zoo then we are happy then if the sun shines then we are happy.*
- Maybe this already shows why it is better to use formulas to write propositions.
- How to prove $(P \to Q) \to (Q \to R) \to P \to R$ in Lean?

Tactics: Assume and Apply

- **assume h** means that we are going to prove an implication $P \to Q$ by assuming P (and we call this assumption h) and then proving Q with this assumption.
- **apply h** : if we have assumed an implication $h : P \to Q$ and our current goal matches Q we can use this assumption to *reduce* the problem to showing P.

The Proving Process

- After the three *assume* we are in the following state:

```
P Q R : Prop,
p2q : P → Q,
q2r : Q → R,
p : P
⊢ R
```

- Now we use an implication. Clearly it is *q2r* which can be of any help because it promises to show *R* given *Q*.

```
P Q R : Prop,
p2q : P → Q,
q2r : Q → R,
p : P
⊢ Q
```

- The next step is to use *apply p2q* to reduce the goal to *P*, which can be shown using *exact P*.

The Proof in Lean

Here is the proof in Lean.

```
variables P Q R : Prop

theorem C : (P → Q) → (Q → R) → P → R :=
begin
  assume p2q,
  assume q2r,
  assume p,
  apply q2r,
  apply p2q,
  exact p,
end
```

Exercise

Prove $(P \to Q \to R) \to (Q \to P \to R)$ in Lean.

What is a proof?

- It looks like a proof in Lean is a sequence of tactics.
- The tactics are actually more like editor commands which generate the real proof which is a **program**.
- We can see the programs generated from proofs by using the **#print** operation in Lean:

  ```
  #print I
  #print C
  ```

Proof terms

- The proof term associated to *I* is

```
theorem I : ∀ (P : Prop), P → P :=
λ (P : Prop) (h : P), h
```

- The proof term for *C* is

```
theorem C : ∀ (P Q R : Prop), (P → Q) → (Q → R) → P → R :=
λ (P Q R : Prop) (p2q : P → Q) (q2r : Q → R) (p : P), q2r (p2q p)
```

- Proofs are **functional programs**.

Functional Programs and Lean

- Lean exploits the *propositions as types translation* (aka *the Curry-Howard-Equivalence*) and associates to every proposition the type of evidence for this proposition.
- This means that to see that a proposition holds, all we need to do is to find a program in the type associated to it.
- The functional language on which Lean relies on is called *dependent type theory* or more specifically *The Calculus of Inductive Constructions*.