Formalization of Mathematics, with Lean4

Rindra Razafy, Hagamena

September 7, 2023

Contents

1 Introduction

The main focus of this is the digitalization of mathematics, we won't be going too deep into the programming topics. In this course we will be using the formal proof assistant Lean4.

1.1 Recommended Textbooks

- Theorem proving in Lean4
- Mathematics in Lean
- Hitchiker's guide to logical verification

Note that the Hitchiker's guide is written in Lean 3 but the content is still useful.

2 Basics in Dependent Type Theory

Lean uses *Dependent Type Theory* (DTT), a flavour of type theory. In DTT everything is a term and every term has a type. We can think of types as analoguous to sets in traditional set theory.

If in traditional mathematics we say that G is a group, then in DTT we say that G is a type with a group structure.

Example 2.0.1. $\mathbb{N}$ is the type of natural numbers, but it is also a term of type **Type** (or Type 0).

What about the type of Type 0? It's Type 1 of course! We can think of Type 0 as the "Collection of all sets", which is a proper class.

Other examples of types

1. $\mathbb{N}$ is the type of the natural numbers
2. *String* contains terms which are (UTF-8 encoded) strings
3. *Char* is the type of characters
4. List $\mathbb{N}$ is the type of lists of natural numbers (e.g. [1,2,3])
5. ...

2.1 Functions

In set theory $f: A \to B$ is defined as a subset of $A \times B$ with some conditions.

In type theory functions are primitive objects:

```
ℕ → ℕ: Type
```

How do construct functions in Lean?

```
fun n : ℕ ⇒ n^2
```

The type of this function is $\mathbb{N} \to \mathbb{N}$. In some sense if we say in maths $f: A \to B$, we actually mean f is a term of type $A \to B$.

2.2 Matrices

In programming Matrices can be thought as a list of list. While we can also define them like this in lean it is not convinient in theorem proving.

Since everything is formally verfied we need to prove that the index we are using to extract is less than the length of the list.

```
#eval [1,2,3][5] -- Lean tries to prove that 5 ≤ 3 and crashes.
#eval [1,2,3][5]? -- this will return an Option ℕ
```

Vectors In Lean we think of vectors $v \in \mathbb{R}^n$ as a function $v: \{1, \ldots, n\} \to \mathbb{R}$. The syntax is:

```
![2,3,4] -- Type Fin 3 → ℕ
```

2×2 matrix: The type of a 2×2 matrix is:

```
Fin 2→ Fin 2→ ℕ
```

Notation: Fin n is the type of $\{0, \ldots, n\}$.

2.3 Propositions:

In DTT we model propositions using types. The "Curry-Howard correspondence" tells us that is P is a proposition, then terms of type P are proofs of P. As usual we write this as

```
t:P
```

A proposition is of type Prop.

Definition 2.1. Proof Irrelevance in type theory, tells us that two proofs of some proposition are equal. This is what is used in Lean. Proof Irrelevance is also incompatible with Homotopy type theory.

Example 2.1.1. $0 < 1$ is a proposition with proof:

```
    zero_lt_one

```

2.4 What does dependent mean in DTT?

A dependent type is a type that depends on another type. If we want to use Type Theory to do mathematics we need dependence.

Example 2.1.2. If we want to show that $\forall n$, $n = 0 \pmod 2$ or $n = 1 \pmod 1$ we need to make the proposition depend on the natural number n.

Dependent types: Fin is a dependent type, it s a function that takes $n \colon \mathbb{N} \to Type$. The type of Fin is $\mathbb{N} \to Type$, it's a function from a type to the Type of all types.