Analysis 2 Paper Draft

Yuxuan Sun

May 13, 2022

Contents

1 Introduction

Mathematical proofs could get more difficult to read and verify as they grow to be more complicated. What else could we rely on to verify the proofs are correct besides trust on each other's professionalism? Building on homotopy type theory, automated theorem proving has the ability to verify our proofs. As there are many moderm theorem provers, such as Agda, Lean, Coq, the intention of the paper is to give a brief introduction on how to construct basic analysis definitions and to prove theorems using *Lean*, with sufficient explanations for mathematicians on how the language works.

When preparing for this paper, I mainly rely on [2] and [4], and sometimes from [1]. The choice of using *Lean* is based on [3]. Since the paper in general comes from understanding the mixture of the sources and the online library of *Lean*, I won't be able to cite them separately in below.

We will start with some very mathematically simple definitions and proofs, then move to a sequence-limit proof in Analysis I. Due to page limit, I will briefly introduce what the proofs related to functional analysis look like at the end.

2 Basic Definition and Proofs

Specifically, let's look at the definition of upper bounds written in Lean.

```
def upperBounds (A : set ℝ) := { x : ℝ  |  ∀ a ∈ A,  a ≤ x }
```

Here we defined a function named `upperBounds`, which takes one input: `A` whose **type** is a set of real numbers. The output of the function is a set (indicated

by { }, and the type of elements `x` in the set is real numbers. `x` also need to satisfy the property that for all `a` ∈ `A`, `a` ≤ `x`.

we could also define the maximum element in a set as the following.

```
def isMaximum (a : ℝ) (A : set ℝ) := a ∈ A ∧ a ∈ upperBounds A
```

This function **isMaximum** takes two inputs: `a` whose type is real number; `A` whose type is a set of real numbers.

Its output is a boolean: `a` is in `A` **and** `a` is in upperBounds `A`. Notice that here we called the function we defined above so upperBounds A is the output of the function defined above.

An observation is that since the logical connection here is **and**, if **isMaximum a A** returns true, we are guaranteed two truth statements: a is in A and a is in upperBounds A.

Now we could prove something!

Let's prove that if both x and y are maximum of A, then $x = y$

First a longer and more detailed proof is the following.

```
1  lemma uniqueMax (A : set ℝ) (x y : ℝ)
2  (hx : is_maximum x A) (hy : is_maximum y A)
3  : x = y
4  :=
5  begin
6          cases hx with x_in x_up,
7          cases hy with y_in y_up,
8          specialize x_up y,
9          specialize x_up y_in,
10         specialize y_up x x_in,
11         linarith,
12 end
```

The lemma takes four assumptions:

1. `A` whose type is a set of real numbers

2. `x,y` whose types are real numbers

3. `hx` whose type is a boolean derived from `isMaximum` above. Recall that then it contains two statements:

- **a.** x is in A
- **b.** x is in upperBounds A (x is bigger than anything in A)

4. `hy` is similar as `hx`

Assumptions are usually stated before colon **:**

After the colon it's the result we want to prove, namely: $x = y$

:= indicates that our proof begins.

Now let's look at the details of the proof.

At line 6, we have `cases ... with ...`, operations like this are called tactics in *Lean*. It splits the hypothesis `hx` into two pieces, named `x_in` and `x_up`, corresponding to **3a** and **3b** respectively, because `hx` is a conjunction, it will be easier for us to use.

```
1  lemma uniqueMax (A : set ℝ) (x y : ℝ)
2  (hx : is_maximum x A) (hy : is_maximum y A)
3  : x = y
4  :=
5  begin
6          have : x ≤ y, from hy.2 x hx.1,
7          have : y ≤ x, from hx.2 y hy.1,
8          linarith,
9  end
```

To show $x = y$, we have two things to show: $x \leq y$ and $y \leq x$. Let's focus on $x \leq y$ for now.

In order to **have** $x \leq y$, we need to derive **from** `hy.2 x hx.1`. It might be easier to understand to view it as `(hy.2 x) hx.1` where `hy.2 x` yields the statement $x \in A \to x \leq y$ because `hy.2` corresponds to the second component of `isMaximum y A` which is y is bigger than anything in A. Then, `hx.1` tells us that x is indeed in A so the antecedent is satisfied, and we have $x \leq y$ eventually.

Similar for $y \leq x$.

linarith is a basic linear arithmetic package lean has that allow us to turn $x \leq y$ and $y \leq x$ into $x = y$. Thus our proof is done.

3 Sequence-Limit Definition and Proof

Let's prove another thing in analysis, namely, given two sequences u_n, v_n, if $u_n \to l$ and $v_n \to l'$ then $(u_n + v_n) \to (l + l')$

```
lemma sequenceAdd (hu : seq_limit u l1) (hv : seq_limit v l2) :
seq_limit ( u + v) (l1 + l2)
:=
begin
!!! fill in later
end
```

4 functional analysis

choose one of them

Theorem 4.1: $L^1(\mathbb{R})$ closed under addition

Given $f_1, f_2 \in L^1(\mathbb{R})$, then $f_1 + f_2 \in L^1(\mathbb{R})$, namely

$$\int_{-\infty}^{\infty} f_1(x) + f_2(x)dx = \int_{-\infty}^{\infty} f_1(x)dx + \int_{-\infty}^{\infty} f_2(x)dx$$

```
lemma lintegral_nnnorm_add
  {f : a → b} {g : a → y}
  (hf : ae_strongly_measurable f u)
  (hg : ae_strongly_measurable g u)
  : ∫⁻ a, nnnorm (f a) + nnnorm (g a) du = ∫⁻ a, nnnorm (f a) du + ∫⁻ a, nnnorm (g a) du
```

Theorem 4.2: Monotone Convergence Theorem in L^0

Suppose we have a sequence $f_n \in L^0(\mathbb{R})$ with $f_1(x) \leq f_2(x) \leq f_3(x) \leq \ldots$ for all $x \in \mathbb{R}$ and suppose there exists a constant M s.t. $\int_{-\infty}^{\infty} f_n(x)dx \leq M$ for all n.
Then there eixsts $f \in L^0(\mathbb{R})$ s.t. $f_n \to f$ except possibly on a set of measure zero and

$$\lim_{n\to\infty} \int_{-\infty}^{\infty} f_n(x)dx = \int_{-\infty}^{\infty} f(x)dx$$

```
theorem integral_supr {f : ℕ → a → ℝ≥0∞}
(hf : ∀ n, ae_measurable (f n) u)
  (h_mono : ∀ᵐ x du, monotone (λ n, f n x)) :
  (∫⁻ a, ⨆n, f n a du) = (⨆ n, ∫⁻ a, f n a du) :=
```

References

[1] Reynald Affeldt et al. "Formalizing functional analysis structures in dependent type theory". en. In: (), p. 18.

[2] Jeremy Avigad, Leonardo de Moura, and Soonho Kong. "Theorem Proving in Lean". In: (Jan. 2016). DOI: 10.1184/R1/6492902.v1.

[3] Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond. "Formalization of real analysis: a survey of proof assistants and libraries". en. In: *Mathematical Structures in Computer Science* 26.7 (Oct. 2016), pp. 1196–1233. DOI: 10.1017/S0960129514000437.

[4] The Univalent Foundations Program. "Homotopy Type Theory: Univalent Foundations of Mathematics". en. In: *arXiv:1308.0729 [cs, math]* (Aug. 2013). arXiv: 1308.0729.