Analysis 2 Paper Draft

Yuxuan Sun

April 14, 2022

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.

2 Examples from Analysis I

Let's first look at some examples from analysis I. The definitions of real numbers and basic logical operators(and, or) are already defined in Lean. explain more how they are defined? maybe not necessary

2.1 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 $x = y$

```
lemma uniqueMax (A : set ℝ) (x y : ℝ) (hx : is_maximum x A) (hy : is_maximum y A)
: x = y
:=
begin
         have : x ≤ y, from hy.2 x hx.1,
        have : y ≤ x, from hx.2 y hy.1,
        linarith,
end
```

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

After := , our proof begins.

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.

2.2 sequence-limit 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
```

3 functional analysis

choose one of them

Theorem 3.1: $L^1(\mathbb{R})$ 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 3.2: 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)) :
  (int⁻ a, ∪n, f n a du) = (∪ n, ∫⁻ a, f n a du) :=
```

References

R. Affeldt, C. Cohen, M. Kerjean, A. Mahboubi, D. Rouhling, and K. Sakaguchi. Formalizing functional analysis structures in dependent type theory. page 18.

J. Avigad, L. de Moura, and S. Kong. Theorem Proving in Lean. 1 2016. doi: 10.1184/R1/6492902.v1. URL https://kilthub.cmu.edu/articles/journal_contribution/Theorem_Proving_in_Lean/6492902.

M. Bezem, U. Buchholtz, P. Cagne, B. I. Dundas, and D. R. Grayson. Werner Heisenberg, Der Teil und das Ganze: Gespräche im Umkreis der Atomphysik, 1969, English translation, Physics and Beyond, 1971. page 209.

S. Boldo, C. Lelay, and G. Melquiond. Formalization of real analysis: a survey of proof assistants and libraries. *Mathematical Structures in Computer Science*, 26(7):1196–1233, Oct. 2016. ISSN 0960-1295, 1469-8072. doi: 10.1017/S0960129514000437. URL https://www.cambridge.org/core/product/identifier/S0960129514000437/type/journal_article.

D. R. Grayson. An introduction to univalent foundations for mathematicians. *Bulletin of the American Mathematical Society*, 55(4):427–450, Mar. 2018. ISSN 0273-0979, 1088-9485. doi: 10.1090/bull/1616. URL http://arxiv.org/abs/1711.01477. arXiv: 1711.01477.

M. Lundfall. Formalizing Real Numbers in Agda. page 62.

J. Palmer. Set Theory, Type Theory and the future of Proof Verification Software. page 33.

T. U. F. Program. Homotopy Type Theory: Univalent Foundations of Mathematics. *arXiv:1308.0729 [cs, math]*, Aug. 2013. URL http://arxiv.org/abs/1308.0729. arXiv: 1308.0729.

E. Rijke. Introduction to Homotopy Type Theory. page 478, 2021.