Overview on Formalization - Type Theory part 1

George McNinch

2024-04-08

Parallel to the course discussions of Lean usage, I want to give at least a bit of a description of the underlying *dependent type theory*.

I'm going to follow the first chapter of Egbert Rijke's new text **Homotopy Type Theory** [arXiv: Homotopy Type Theory] – (though for now I ignore the potential homotopy in the type theory.)

The second section of Theorem Proving in Lean vi(Avigad et al. n.d.) is also good - and `Lean` specific! - but still comes across (to me) as more of a guide to using Lean than an introduction to type theory (this is not a criticism, of course!)

All of this really does bump up against an important pedagogical question: how much type theory does one really need to know in order to *use* `Lean` to formalize mathematics?

I'm sure I don't know the answer! In the limited time in this third-of-a-course, I'm hoping to say a little about "using Lean" as well as a little about the type theory than enables it.

Martin-Löf's Type Theory

I'm mostly viewing these notes as a quick exposition of the first chapter of Rijke's text (Rijke n.d.); probably you should just read the text! But maybe these notes will encourage you to do so… At any rate, quotes below are taken from (Rijke n.d.).

To the newcomer, one of the main questions one probably faces in this subject is: "what are the differences between type theory and the usual description of math via set theory?" The introduction to Rijke's chapter tells us that in type theory:

- every element comes equipped with its type
- set theory is axiomatized in the formal system of first-order logic, while type theory is its own formal system.

  "the expression $a : A$ is not considered a *proposition* - i.e. something which one can assert about an arbitrary element and an arbitrary type – but is considered to be a judgment … that is part of the construction of the element $a : A$."

- there is a greater emphasis in type theory on equality of elements than in set theory.

  Unlike in set theory, where equality is a decidable proposition of first-order logic, the type $x = y$ of identifications of two elements $x, y : A$ is itself a type.

By way of a general description, we have:

> Dependent type theory is a system of inference rules that can be combined to make *derivations* … the goal is often to construct an element of a certain type.

Judgments and contexts in type theory

In type theory, an *inference rule* is represented symbolically as follows:

$$\frac{\mathscr{H}_1 \ \mathscr{H}_2 \ \cdots \ \mathscr{H}_n}{C}.$$

Above the horizontal line is a finite list of *judgments* $\mathscr{H}_i$ for the *premises*, and below the horizontal line is a single judgment C for the conclusion.

When we later introduce *function types*, we'll see the following inference rule appear:

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash f : A \to B}{\Gamma \vdash f(a) : B}.$$

This means: "in a context Γ (more on the notion of context below!) given an element [1] a of type A and a function $f : A \to B$, we obtain an element $f(a)$ of type B."

Here the *judgments* include $\Gamma \vdash a : A$ and $\Gamma \vdash f : A \to B$.

In fact, there are just four sorts of judgments in Martin-Löf's dependent type theory:

- judgment that A is a well-formed type in context Γ:

$$\Gamma \vdash A \text{ type}$$

- judgment that A and B are *judgmentally equal* (or *definitionally equal*) in context Γ:

$$\Gamma \vdash A \doteq B \text{ type}$$

- judgment that a is an element of type A in context Γ:

$$\Gamma \vdash a : A$$

- judgment that a and b are *judgmentally equal* elements of type A

$$\Gamma \vdash a \doteq b : A.$$

Notice that any judgment has the form $\Gamma \vdash \mathscr{J}$ where Γ is the *context* and $\mathscr{J}$ is the *judgment thesis*. It remains to explain the *context* Γ.

The role of the context is to declare what *hypothetical elements* (variables) are assumed.

Definition A *context* is a finite list of variable declarations

$$x_1 : A_1, x : 2 : A_2(x_1), \cdots, x_n : A_n(x_1, \cdots, x_{n-1})$$

such that for $1 \leq k \leq n$ we can derive the judgment

$$(\clubsuit) \quad x_1 : A_1, x_2 : A_2(x_1), \cdots, x_{k-1} : A_{k-1}(x_1, \cdots, x_{k-2}) \vdash A_k(x_1, \cdots, x_{k-1}) \text{ type}$$

using inference rules of type theory.

Of course, in a context we may use other variable names than x_i, so long as no variable is declared more than once.

Examples

- the empty context has length 0 and declares no variables. Examples of well-formed types in an empty context include the natural numbers $\mathbb{N}$.
- a context of length 1 has the form $x_1 : A_1$ where A_1 is a well-formed type in the empty context.

  The type `Vec n` of *vectors of integers of length* `n:` is a type in context $n : \mathbb{N}$. e.g. we have `[8,22,-17] : Vec 3`
- For any type A in the empty context, and any term $a : A$, the equality type $x = a$ is a type in context $x : A$.

  (for a term $x_0 : A$, the type $x_0 = a$ has the term `refl:`$x_0 = a$ if x_0 and a are judgmentally equal; i.e. we have the judgment

$$\emptyset \vdash x_0 \doteq a : A$$

)
- a context of length 2 has the form $x_1 : A_1, x_2 : A_2(x_1)$ and we will explain what is meant by the notation $A_2(x_1)$ in the next section on *type families*.

[1] Rijke reads "$a : A$" as "a is an element of type A". Some other authors read this as "a is a term of type A."

Type families

Definition Consider a type A in context Γ. A *family of types over A in context* Γ is a type $B(x)$ in context $\Gamma, x : A$.

In other words, we have the judgment

$$\Gamma, x : A \vdash B(x) \text{ type}.$$

Terminology: we say that B is a family of types over A in context Γ, or that $B(x)$ is a type *indexed* by $x : A$.

Examples:

- `Vec n` is a type indexed by $n : \mathbb{N}$.
- For $a : A$, `x = a` is a typed indexed by $x : A$. More precisely, we have the inference rule

$$\frac{\Gamma \vdash a : A}{\Gamma, x : A \vdash a = x \text{ type}}.$$

Definition Consider a family of type B over A in context Γ. A *section* of B over A in context Γ is an element of type $B(x)$ in context $\Gamma, x : A$; i.e. the judgment

$$\Gamma, x : A \vdash b(x) : B(x).$$

For example, consider the type `Vec n` indexed by $n : \mathbb{N}$. An example of a section is the term $b(n)$ where

$$b(0) = []$$

and for $n > 0$,

$$b(n) = [0, 0, \cdots, 0] = 0 :: b(n-1)$$

Inference Rules

Here are the inference rules "underlying" dependent type theory.

- Rules for formation of contexts, types and elements

  The following rules amount to "presuppositions"; for example, If we have the judgment that $B(x)$ is a well-formed type in context $\Gamma, x : A$, then A is already a well-formed type in context Γ.

$$\frac{\Gamma, x : A \vdash B(x) \text{ type}}{\Gamma \vdash A \text{ type}} \quad \frac{\Gamma \vdash A \doteq B \text{ type}}{\Gamma \vdash A \text{ type}} \quad \frac{\Gamma \vdash A \doteq B \text{ type}}{\Gamma \vdash B \text{ type}}$$

$$\frac{\Gamma \vdash a : A \text{ type}}{\Gamma \vdash A \text{ type}} \quad \frac{\Gamma \vdash a \doteq b : A}{\Gamma \vdash a : A} \quad \frac{\Gamma \vdash a \doteq b : A}{\Gamma \vdash b : A \text{ type}}$$

- Judgmental equality is an equivalence relation

  We have to require this for types:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \doteq A \text{ type}} \quad \frac{\Gamma \vdash A \doteq B \text{ type}}{\Gamma \vdash B \doteq A \text{ type}} \quad \frac{\Gamma \vdash A \doteq B \text{ type} \quad \Gamma \vdash B \doteq C \text{ type}}{\Gamma \vdash A \doteq C \text{ type}}$$

  and for terms:

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \doteq a : A} \quad \frac{\Gamma \vdash a \doteq b : A}{\Gamma \vdash b \doteq a : A} \quad \frac{\Gamma \vdash a \doteq b : A \quad \Gamma \vdash b \doteq c : A}{\Gamma \vdash a \doteq c : A}$$

- variable conversion rules

  One example of such a rule is:

$$(\diamondsuit) \quad \frac{\Gamma \vdash A \doteq A' \text{ type} \quad \Gamma, x : A, \Delta \vdash B(x) \text{ type}}{\Gamma, x : A', \Delta \vdash B(x) \text{ type}}$$

thus if x is a variable of type A and if $A \doteq A'$ (in the given context), then we can replace $x : A$ by $x : A'$ when describing families of types.

Note that writing $B(x)$ just emphasizes the dependence on $x : A$; of course, B depends on the whole context $\Gamma, x : A, \Delta$.

We must give rules like $(\diamondsuit)$ not only for judgments of well-formed types, but of judgments for type-equality, for elements, and for element-equality. We write $\mathscr{J}$ for a generic *judgment thesis*; the general form of the variable conversion rule is then

$$(\diamondsuit') \quad \frac{\Gamma \vdash A \doteq A' \quad \Gamma, x : A, \Delta \vdash \mathscr{J}}{\Gamma, x : A', \Delta \vdash \mathscr{J}}$$

- substitution

  Consider a section $f(x) : B(x)$ indexed by $x : A$ in context Γ and suppose give $a : A$.

  We can simultaneously substitute a for all occurrences of x in $f(x)$ and $B(x)$ to obtain a new element $f[a/x] : B[a/x]$.

  Somewhat more precisely, consider the following. Given the judgment

  $$(*) \quad \Gamma, x : A, y_1 : B_a, \cdots, y_n : B_n \vdash C \text{ type}$$

  and a term $a : A$, we can substitute to obtain the judgment

  $$\Gamma, y_1 : B_1[a/x], \cdots, y_n : B_n[a/x] \vdash C[a/x] \text{ type}$$

  Note that in this latter judgment, the *variables* $y_1, \cdots, y_n$ are assigned new types after substitution.

  Similarly, given C as in $(*)$, from a term $c : C$, we obtain a new term $c[a/x] : C[a/x]$.

  For a generic judgment $\mathscr{J}$, we have the substitution rule

  $$\frac{\Gamma \vdash a : A \quad \Gamma, x : A, \Delta \vdash \mathscr{J}}{\Gamma, \Delta[a/x] \vdash \mathscr{J}[a/x]} S$$

  Definition When B is a family of types over A in context Γ and when $a : A$, we refer to the type $B[a/x]$ as the *fiber* of B at a. It is often written $B(a)$.

  Similarly, given a section b of B, we write $b(a)$ for $b[a/x]$.

- weakening

  Given a type A in context Γ, any judgment made in a longer context Γ, Δ can be made in the context $\Gamma, x : A, \Delta$ for a new variable x. The weakening rule says that well-formed-ness and judgmental equality of types and elements are preserved by this operation: for a generic judgement $\mathscr{J}$ we have

  $$\frac{\Gamma \vdash A \text{ type} \quad \Gamma, \Delta \vdash \mathscr{J}}{\Gamma, x : A, \Delta \vdash \mathscr{J}} W,$$

  The process of expanding the context by $x : A$ is called *weakening* (by A).

  Example If we have two types A, B in context Γ, then we can weaken B by A to obtain

  $$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash B \text{ type}}{\Gamma, x : A \vdash B \text{ type}} W$$

  The type B in context $\Gamma, x : A$ is called the *constant* (or *trivial*) family B.

- the generic element

  The last general inference rule concerns the so-called *generic element* of a type family. Given a type A in context Γ, we can weaken A by itself to obtain that A is a type in context $\Gamma, x : A$. The rule then stipulates that the variable x may be viewed as an element of $x : A$ in this context; i.e.

  $$\frac{\Gamma : A \text{ type}}{\Gamma, x : A \vdash x : A} \delta$$

Bibliography

Bibliography

Avigad, Jeremy, Leondardo de Moura, Soonho Kong, and Sebastian Ullrich. n.d. "Theorem Proving in Lean 4." https://leanprover.github.io/theorem_proving_in_lean4/. Accessed January 11, 2024.

Rijke, Egbert. n.d. "Introduction to Homotopy Type Theory." https://arxiv.org/abs/2212.11082. Accessed April 10, 2024.