Type Theory with Paige North 7/3

Jason Schuchardt

July 3, 2019

1 Contexts

Contexts are important for math.
Example 1.1. For any natural number n, $2n$ is even.

The phrase "$2n$ is even" is the math part. The phrase "For any natural number n" is the context. It declares our variable basically.

In a type theoretic way, we might say: For $n : \mathbb{N}$, $2n$ is even.
Example 1.2. Another sentence might be "Suppose $f : \mathbb{R} \to \mathbb{R}$ is a continuous function such that $f(0) < 0$ and $f(1) > 0$, then there is some $c \in \mathbb{R}$, $0 < c < 1$ such that $f(c) = 0$." The context here declares the variable f.
Example 1.3. For $f : T$, there is some $\cdots$. We can encapsulate all of the hypotheses on f into the type T. We can have T be the type whose terms t are continuous functions $\mathbb{R} \to \mathbb{R}$ such that $t(0) < 0$ and $t(1) > 0$.
Example 1.4. For any variable $x : T$, there is a term $x : T$. In symbols, we might write:

$$x : T \vdash x : T,$$

where the left $x : T$ is the context, and $x : T$ is the product.
Example 1.5. We can also write something similar for function application. For any variable $f : S \implies T$, and any $x : S$, we get $fx : T$. In symbols, this is

$$f : S \implies T, x : S \vdash fx : T.$$

Example 1.6. In the natural numbers, we have the statement

$$\vdash 0 : \mathbb{N}.$$

This requires no context, since there are no hypotheses necessary.

2 Simply Typed Lambda Calculus Again

We'll redefine the STLC with some notation that became standard a bit over twenty years ago, that will help us fit it into a more modern framework.

There are three kinds of *judgments*:

1. (Is a type judgement) ____ TYPE (ex. $T_1 \implies T_2$ TYPE)
2. (Is of type judgement) ____ $\vdash$ ____ : ____ (ex. $f : S \implies T, x : S \vdash fx : T$)
3. (Equality judgement) ____ $\vdash$ ____ $=$ ____ : ____ (ex. $x : T \vdash (\lambda y.y)x = x : T$)

These are the things that we can say about our type theory. A *rule* consists of

$$\frac{\text{(some judgements)}}{\text{(a judgement)}}.$$

For example, we might have the rule

$$\frac{\vdash f : S \implies T, \vdash S : S}{\vdash f(S) : T}.$$

Definition 2.1. The *simply typed lambda calculus* is given by the following rules:

1. We have types:

$$\frac{}{\vdash T_1 \text{ TYPE}}, \ldots, \frac{}{\vdash T_n \text{ TYPE}}$$

2. And we have terms, we think of as variables:

$$\frac{T \text{ TYPE}}{\vdash x_1^T : T}, \ldots, \frac{T \text{ TYPE}}{\vdash x_m^T : T}$$

3. We have a secret judgement for contexts, satisfying the rules

$$\frac{}{\varnothing \text{ ctxt}}, \quad \frac{\Gamma \text{ ctxt}, \quad T \text{ TYPE}}{\Gamma, x_i^T : T \text{ ctxt}},$$

$$\frac{}{\Gamma, x : T, \Delta \vdash x : T}, \quad \frac{\Gamma \vdash t : T, \quad x : T \vdash s : S}{\Gamma \vdash s[t/x] : S}$$

4. We have rules for equality that are fairly obvious

$$\frac{t : T}{t = t : T}, \quad \frac{t = s : T}{s = t : T}, \quad \frac{t = s : T, \quad s = u : T}{t = u : T}$$

   We also have an equality rule for substitution.

5. Rules for $\implies$-types

$$\frac{S \text{ TYPE}, \quad T \text{ TYPE}}{S \implies T \text{ TYPE}}, \quad \frac{S \text{ TYPE}, \quad T \text{ TYPE}, \quad x_i^S : S \vdash t : T}{\lambda x_i^S . t : S \implies T}$$

   omitting the type judgements, we have

$$\frac{f : S \implies T, \quad s : S}{fs : T}, \quad \frac{x_i^S : S \vdash t : T, \quad s : S}{(\lambda x_i^S . t)s = t[s/x_i^S] : T}$$

$$\frac{f : S \implies T, \quad x_i^S : S}{\lambda x_i^S . (f x_i^S) = f : S \implies T}$$

   when x_i^S doesn't appear in f.

A whole bunch of turnstiles were omitted, which is because we want all the contexts here to implicitly be generic, they might all be some context Γ.

For example, we might change the function application rule to be

$$\frac{\Gamma \vdash f : S \implies T, \quad \Gamma \vdash s : S}{\Gamma \vdash fs : T}.$$

2.1 Free and bound variables

In computer science, there is a big distinction between free and bound variables.

Consider the statement

$$x : \mathbb{N} \vdash x^2 : \mathbb{N}.$$

Here x is *free*. We can then use our λ-abstraction rule, to form the function

$$\vdash \lambda x.x^2 : \mathbb{N} \implies \mathbb{N}.$$

In this sentence, x is *bound*. We say λ is a *binder* and that it *binds* x.

Another example of a binder, in $\mathbb{R}$ is

$$\int_0^1 x^2 \, dx.$$

The integral sign, $\int_a^b dx$ binds the x in the formula x^2.

Recall the rule

$$\frac{\Gamma, x_1^S : S \vdash t : T}{\Gamma \vdash \lambda x_1^S.t : S \implies T}.$$

Example 2.1. We can derive a term of

$$S \implies (S \implies T) \implies T.$$

Intuitively, we can define $\epsilon(s)(f) : T$. This is the evaluation map, $\epsilon(s)(f) = f(s)$.

How do we get this? We apply our rules. We want to derive

$$s : S,\ f : S \implies T \vdash fs : T$$

so that we can apply λ-abstraction.

We have to start with our function application rule

$$\frac{s : S,\ f : S \implies T}{fs : T}.$$

We know

$$s : S,\ f : S \implies T \vdash s : S$$

and

$$s : S,\ f : S \implies T \vdash f : S \implies T,$$

so by our function application rule, we conclude

$$s : S,\ f : S \implies T \vdash fs : T.$$

Then by the lambda abstraction rule, we have

$$s : S \vdash \lambda f.fs : (S \implies T) \implies T,$$

and abstracting a second time, we have

$$\lambda s.\lambda f.fs : S \implies (S \implies T) \implies T,$$

as desired.

Question: Is the idea of a context, roughly the following?

In ordinary logic, we have formulas which have free variables in them. However, in type theory, all of our terms have types, so our free variables also need to have types. A context is a way of recording the free variables in your formulas and tracking their types.

Answer: Yes, that's one way of thinking about contexts.

Question: Are these rules reversible? Answer, no.

Question: Would we formulate ring theory in as judgements? Answer: No, instead the axioms of ring theory will be terms that live in certains types.

Question: What about associativity? Is that a judgment? Answer: No, rather $a + (b + c) = (a + b) + c$ is a type, and a proof of associativity is a term in this type.