

Type Theory with Paige North 7/9

Jason Schuchardt

July 9, 2019

1 Set interpretation of type theory

We were talking about the set interpretation of type theory last time. See a summary in Table 1.

Type	Set
Term	Element
Dependent term, $x : T \vdash s(x) : S$	Function $T \rightarrow S$
$S \Rightarrow T$	Set of functions, $\text{Hom}(S, T)$
$S \wedge T$	Cartesian product, $S \times T$
$S \vee T$	Disjoint union, $S \sqcup T$
\perp	\emptyset
\top	$\{*\}$
$\neg A := A \Rightarrow \perp$	$\text{Hom}(A, \emptyset)$
$A \vee \neg A$	$A \sqcup (A \rightarrow \emptyset) = \begin{cases} \{*\} & \text{if } A = \emptyset \\ A & \text{otherwise} \end{cases}$

Table 1: The correspondence between type theoretical notions and their interpretations in set theory.

2 Natural Numbers

What should they be?

In any type definition, we introduced terms. For example, we had

$$\frac{}{\vdash * : T} \quad \frac{\vdash a : A}{\vdash i_1(a) : A \vee B} \quad \frac{x : S \vdash t : T}{\vdash \lambda x. t : S \Rightarrow T}$$

These elements are called canonical terms. What are the canonical terms of \mathbb{N} ?

Definition 2.1. The natural numbers type will be defined by the following rules:

1. The natural numbers is a type:

$$\overline{\mathbb{N} \text{ TYPE}}$$

2. Term construction

$$\frac{}{0 : \mathbb{N}} \quad \frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash sn : \mathbb{N}}.$$

3. We also need a way to access the terms in the type, which we will do by specifying how we can build functions out of the type. This is like with the \vee type, where we had the rule

2.1 Examples of using the function construction rule for the natural numbers 2 NATURAL NUMBERS

$$\frac{\Gamma, a : A \vdash y : Z \quad \Gamma, b : B \vdash z : Z}{\Gamma, c : A \vee B \vdash j_{y,z}(c) : Z}$$

For the natural numbers, we have the rule (which we might call induction)

$$\frac{T \text{ TYPE} \quad \Gamma \vdash z : T \quad \Gamma, t : T \vdash \sigma t : T}{\Gamma, n : \mathbb{N} \vdash j_{z,\sigma}(n) : T}$$

4. We need this to satisfy the following equality rules:

$$\Gamma \vdash j_{z,\sigma}(0) = z : T,$$

and

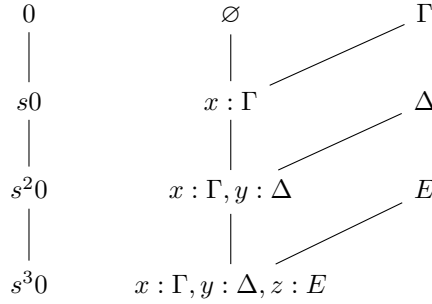
$$\Gamma, n : \mathbb{N} \vdash j_{z,\sigma}(sn) = \sigma(j_{z,\sigma}(n)) : T.$$

We say \mathbb{N} is *inductively* or *recursively* defined. \mathbb{N} is defined to be the type whose terms are $0 : \mathbb{N}$, $sn : \mathbb{N}$ for every n .

Contexts were also recursively defined, as a list, with the rules:

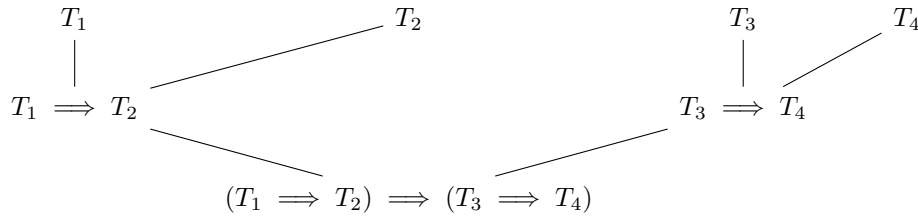
$$\frac{}{\emptyset \text{ ctxt}} \quad \frac{\Gamma \text{ ctxt} \quad T \text{ TYPE}}{\Gamma, x : T}$$

The natural numbers form a sort of tree, as do contexts.



On the left, we have the construction of $3 : \mathbb{N}$, and on the right, we have the construction of the context $x : \Gamma, y : \Delta, z : E$.

The types in the simply typed lambda calculus are also recursively defined. We started with T_1, \dots, T_n . For example, the tree for $(T_1 \Rightarrow T_2) \Rightarrow (T_3 \Rightarrow T_4)$ is



2.1 Examples of using the function construction rule for the natural numbers

We'll construct a function

$$f : T \Rightarrow T, n : \mathbb{N} \vdash f^n : T \Rightarrow T,$$

where $f^n = f \circ f \circ f \circ \dots \circ f$ n times.

The plan is to define $c(f, n)$ by the rules $c(f, 0) = \text{id}_T$ and $c(f, sn) = f \circ c(f, n)$.

The value at zero is

$$f : T \Longrightarrow T \vdash \lambda x. x : T \Longrightarrow T,$$

and the inductive step is

$$f : T \Longrightarrow T, g : T \Longrightarrow T \vdash f \circ g : T \Longrightarrow T.$$

These yield

$$f : T \Longrightarrow T, n : \mathbb{N} \vdash j_{\lambda x. x, f \circ -}(n) : T \Longrightarrow T.$$

Then we know that $j(f, 0) = \lambda x. x$, and $j(f, sn) = f \circ j(f, n)$.

Question. It seems like we could do the composition in the other order, and its not clear that they are equal.

Answer: Yes. We need a stronger type theory to prove that. We can prove it for a specific number, e.g., we can prove $f \circ (f \circ f) = (f \circ f) \circ f$.

Example 2.1. Define the function $\lambda x. s^2 x : \mathbb{N} \rightarrow \mathbb{N}$ using induction rather than lambda abstraction. (This is the function $x \mapsto x + 2$)

For 0, we have

$$\vdash s(s(0)),$$

and for the inductive step, we have

$$n : \mathbb{N} \vdash sn : \mathbb{N},$$

so applying the inductive function formation rule, we have

$$n : \mathbb{N} \vdash j_{s^2 0, s}(n) : \mathbb{N}.$$

Checking, we have $\vdash j(0) = s^2 0 : \mathbb{N}$, and $n : \mathbb{N} \vdash j(sn) = s(jn) : \mathbb{N}$.

Question: It seems like we could prove that this function is actually equal to $\lambda x. s^2 x$ with some sort of induction. Would that be a function from \mathbb{N} to some equality type?

Answer: Yes, but we don't have an equality type yet. Hopefully we'll talk about the dependently typed lambda calculus tomorrow, and then we can introduce that type.

Example 2.2. Want to define $n : \mathbb{N}, m : \mathbb{N} \vdash \text{add}(n, m) : \mathbb{N}$. For zero we have

$$n : \mathbb{N} \vdash n : \mathbb{N},$$

and for the inductive step, we have

$$n : \mathbb{N}, x : \mathbb{N} \vdash sx : \mathbb{N}.$$

This yields

$$n : \mathbb{N}, m : \mathbb{N} \vdash j_{n, \lambda x. sx}(m) : \mathbb{N}.$$

Checking this, we have $n : \mathbb{N} \vdash j_n(0) = n$, and $n : \mathbb{N}, m : \mathbb{N} \vdash j_n(sm) = sj_n(m) : \mathbb{N}$.

Notice the asymmetry here. We inducted on m , and we could have inducted on n . Metatheoretically, we can see that these two ways of defining addition are the same, but hopefully next time, we can prove that they are the same inside the type theory.

Example 2.3. Now we can define multiplication! $n : \mathbb{N}, m : \mathbb{N} \vdash \text{mult}(n, m) : \mathbb{N}$.

Once again, we start with 0:

$$n : \mathbb{N} \vdash 0 : \mathbb{N},$$

and the inductive step:

$$n : \mathbb{N}, x : \mathbb{N} \vdash \text{add}(x, n).$$

Then by induction, we get the function

$$n : \mathbb{N}, m : \mathbb{N} \vdash \text{mult}(n, m) : \mathbb{N},$$

and we know that $\text{mult}(n, 0) = 0$, and $\text{mult}(n, sm) = \text{add}(\text{mult}(n, m), n)$.

3 The list type

Lists are defined by the rules:

1.

$$\frac{T \text{ TYPE}}{\text{LIST}(T) \text{ TYPE}}$$

2. Canonical elements:

$$\frac{}{\text{nil} : \text{LIST}(T)}, \quad \frac{\Gamma \vdash \ell : \text{LIST}(T), \quad t : T}{\Gamma \vdash \text{con}(\ell, t) : \text{LIST}(T)}$$

3. Induction:

$$\frac{\Gamma \vdash s : S, \quad \Gamma, x : \text{LIST}(T), y : T \vdash c(x, y) : S}{\Gamma, \ell : \text{LIST}(T) \vdash j_{s,c}(\ell) : S}$$

4. Where induction satisfies the coherence rules:

$$j_{s,c}(\text{nil}) = s, \text{ and } j_{s,c}(\text{con}(x, y)) = c(x, y)$$