Type Theory with Paige North 7/10

Jason Schuchardt

July 11, 2019

1 Sum Types

If we have a dependent type,

$$x : S \vdash T(x) \text{ TYPE},$$

we want to take a union, which we write as a sum:

$$\sum_{x:S} T(x).$$

1. $\sum$-formation
$$\frac{\Gamma, x : S \vdash T(x) \text{ TYPE}}{\Gamma \vdash \sum_{x:S} T(x) \text{ TYPE}}$$

2. $\sum$-introduction
$$\frac{\Gamma \vdash s : S \qquad \Gamma \vdash t : T(s)}{\Gamma \vdash (s,t) : \sum_{x:S} T(x)}$$

3. $\sum$-elimination
$$\frac{\Gamma, z : \sum_{x:S} T(x) \vdash C(z) \qquad \Gamma, s : S, t : T(s) \vdash c(s,t) : C(s,t)}{\Gamma, z : \sum_{x:S} T(x) \vdash j_c(z) : C(z)}$$

4. $\sum$-computation
$$\frac{\Gamma, z : \sum_{x:S} T(x) \vdash C(z) \qquad \Gamma, s : S, t : T(s) \vdash c(s,t) : C(s,t)}{\Gamma, s : S, t : T(s) \vdash j_c(s,t) = c(s,t) : C(s,t)}$$

Example 1.1. There are two functions

$$z : \sum_{x:S} T(s) \vdash \pi_1(z) : S,$$

and

$$z : \sum_{x:S} T(s) \vdash \pi_2(z) : T(\pi_1(z)).$$

We need to use the elimination rule. We start with

$$z : \sum_{x:S} T(x) \vdash S \text{ TYPE},$$

and

$$s : S, t : T(s) \vdash s : S,$$

so by the elimination rule, we can produce

$$z : \sum_{x:S} T(x) \vdash \pi_1(z) : S.$$

For π_2, we have

$$z : \sum_{x:S} T(x) \vdash T(\pi_1 z) \text{ TYPE},$$

and

$$s : S, t : T(s) \vdash t : T(\pi_1(s,t)) = T(s).$$

Thus by the elimination rule, we have

$$z : \sum_{x:S} T(x) \vdash \pi_2(z) : T(\pi_1 z).$$

Types	Sets	Logic
$\sum\limits_{x:S} T(x)$	$\coprod\limits_{x\in S} T(x)$	To prove $\sum_{x:S} T(x)$, it suffices to prove $T(s)$ for a specific $s : S$. In other words, sum types are like existential quantifiers. "There exists $x : S$ such that $T(x)$ holds." In usual math, proving something exists, doesn't mean you can actually find it. For example, if we have a continuous function $f : \mathbb{R} \to \mathbb{R}$ such that $\lim\limits_{x\to\infty} f(x) > 0$ and $\lim\limits_{x\to-\infty} f(x) < 0$. Then $T(x)$ is "$f(x) = 0$," and $S = \mathbb{R}$. Then $\exists_{x:\mathbb{R}} T(x)$.

We automatically get $\wedge$-types. Consider two types $\vdash S$ TYPE and $\vdash T$ TYPE, then we can derive

$$x : S \vdash T \text{ TYPE},$$

so we can conclude from $\sum$-formation,

$$\vdash \sum_{x:S} T \text{ TYPE}.$$

Then

$$\frac{\vdash s : S \qquad \vdash t : T}{\vdash (s,t) : \sum_{x:S} T},$$

is our $\wedge$-introduction rule. For wedge-elimination, we already have

$$\frac{\vdash z : \sum_{x:S} T}{\vdash \pi_1 z : S} \qquad \frac{\vdash z : \sum_{x:S} T}{\vdash \pi_2(z) : T}.$$

Finally, $\wedge$-computation follows from the $\sum$-computation rule.

The $\sum$-type of a dependent type, $x : S \vdash T$, where T depends on S only *trivially* is an $\wedge$-type.

2 Fibers

In set theory, we had

$$\{\text{families indexed by } B\} \longleftrightarrow \{\text{functions with codomain } B\}.$$

In type theory, we want a correspondence

$$\{\text{types dependent on } B\} \longleftrightarrow \{\text{Functions with codomain } B,\ x : A \vdash a(x) : B\}.$$

We don't have nearly enough to prove this yet. Suppose we have a dependent type, $b : B \vdash E(b)$. Then we have

$$z : \sum_{b:B} E(b) \vdash \pi_1(z) : B,$$

which is the thing we want on the right hand side.

Can we go the other way? In set theory, what happens when we have a function with codomain B and want to produce a family indexed by B? We have

$$E \xrightarrow{f} B,$$

and we get

$$f^{-1}(b) = \{e \in E | fe = b\}.$$

We can't do this in type theory. In set theory, we have logic and the set theory as two different layers. The set theory is built on top of the logic. On the other hand, in type theory, everything is a type, and there is only one layer. Thus we need to reformulate this in type theory.

Define a set

$$[fe = b] = \begin{cases} \{e\} & \text{if } fe = b \\ \varnothing & \text{if } fe \neq b. \end{cases}$$

Then we could define $f^{-1}(b) = \bigsqcup_{e \in E} [fe = b]$. This contains an e if and only if $fe = b$.

Now this is something that we can emulate in type theory. In the type theory, if we have $e : E \vdash f(e) : B$, then we define

$$f^{-1}(b) := \sum_{e:E} \mathrm{Id}_B(fe, b).$$

The canonical terms in this type are (e, p), where p is a witness to the equality of $f(e)$ and b.

Now, we can write

$$b : B \vdash \sum_{e:E} \mathrm{Id}_B(fe, b).$$

This should be inverse to the previous process, but we don't yet have enough to prove that.

3 Unions

The $\sum$-type also generalizes the $\vee$-type that we had before. The $\sum$-types are unions of types $T(x)$ indexed by S. To get a twofold union, we need S to have two elements.

The type with two terms was defined on the first homework, $\mathbb{B}$, with the following rules.

1. $\mathbb{B}$-formation

$$\frac{}{\mathbb{B}\ \text{TYPE}}$$

2. $\mathbb{B}$-introduction

$$\frac{}{0 : \mathbb{B} \qquad 1 : \mathbb{B}}$$

3. $\mathbb{B}$-elimination

$$\frac{\Gamma, b : \mathbb{B} \vdash C(b)\ \text{TYPE} \qquad \Gamma \vdash c_0 : C(0) \qquad \Gamma \vdash c_1 : C(1)}{\Gamma, b : \mathbb{B} \vdash j_{c_0,c_1}(b) : C(b)}$$

4. $\mathbb{B}$-computation

$$\frac{\Gamma, b : \mathbb{B} \vdash C(b) \text{ TYPE} \qquad \Gamma \vdash c_0 : C(0) \qquad \Gamma \vdash c_1 : C(1)}{\Gamma \vdash j_{c_0,c_1}(0) = c_0 : C(0) \qquad \Gamma \vdash j_{c_0,c_1}(1) = c_1 : C(1)}$$

In dependent type theory, we assume that every type is a term of a universe, which is a type. Actually, there is a hierarchy of universes $U_1, U_2, \ldots$. With each universe a term of the next, and all the terms of U_i a term of U_{i+1}. This is the first time we've said that a term is of more than one type, but this is something that can be addressed in several ways.

We'll ignore this for now, and just talk about a universe U.

Now we have

$$b : \mathbb{B} \vdash U \text{ TYPE},$$

and

$$\vdash S : U \qquad \vdash T : U,$$

so by $\mathbb{B}$-elimination, we have

$$b : \mathbb{B} \vdash j_{S,T}(b) : U.$$

Then by $\sum$-formation, we have

$$\vdash \sum_{b:\mathbb{B}} j_{S,T}(b)$$

Then given $\vdash s : S$, we have

$$\vdash (0, s) : \sum_{b:\mathbb{B}} j_{S,T}(b).$$

Similarly, given $\vdash t : T$, we can form

$$\vdash (1, t) : \sum_{b:\mathbb{B}} j_{S,T}(b).$$

These give us our $\vee$-introduction rules.