Type Theory with Paige North 7/5

Jason Schuchardt

July 5, 2019

Correction:

Last time we wrote the rule

$$\frac{T \text{ TYPE}}{x_i^T : T},$$

but this should really be

$$\frac{T \text{ TYPE}}{x_i^T : T \vdash x_i^T : T},$$

1 And Types

Definition 1.1. The rules for $\wedge$-types are the following

1. $\wedge$-type formation:
$$\frac{S \text{ TYPE}, \quad T \text{ TYPE}}{S \wedge T \text{ TYPE}},$$
2. term construction
$$\frac{\Gamma \vdash s : S, \quad \Gamma \vdash t : T}{\Gamma \vdash (s,t) : S \wedge T},$$
3. and term deconstruction
$$\frac{\Gamma \vdash c : S \wedge T}{\Gamma \vdash \pi_1(c) : S, \quad \Gamma \vdash \pi_2(c) : T}$$
4. compatibility
$$\frac{\Gamma \vdash s : S, \quad \Gamma \vdash t : T}{\Gamma \vdash \pi_1(s,t) = s : S, \quad \Gamma \vdash \pi_2(s,t) = t : T}$$

Example 1.1. We get functions $\pi_1 : S \wedge T \implies S$ and $\pi_2 : S \wedge T \implies T$.

These are derived in the following way: We start with

$$c : S \wedge T \vdash c : S \wedge T,$$

by deconstruction this gives

$$c : S \wedge T \vdash \pi_1(c) : S,$$

and then by lambda abstraction, we get

$$\vdash \lambda c.\pi_1(c) : (S \wedge T) \implies S.$$

Whenever we get a function from

$$\frac{\Gamma, x : S \vdash y : T}{\Gamma \vdash \lambda x.y : S \implies T},$$

Object		Programatic View		Logical View
Types	$\longleftrightarrow$	Program Spaces	$\longleftrightarrow$	Propositions
Terms	$\longleftrightarrow$	Programs	$\longleftrightarrow$	Proofs
Terms in context	$\longleftrightarrow$	Programs that take input	$\longleftrightarrow$	Proofs that take hypotheses

Table 1: Interpretations of type theory

we call this *Currying*, and say we *curry* the x.
Example 1.2. We can derive a term of

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

We start with the following things we know:

$$c : (S \implies T) \wedge S \vdash \pi_1(c) : S \implies T,$$

and

$$c : (S \implies T) \wedge S \vdash \pi_2(c) : S.$$

Then function application gives

$$c : (S \implies T) \wedge S \vdash \pi_1(c)\ \pi_2(c) : T,$$

so finally, by lambda abstraction, we have

$$\vdash \lambda c.(\pi_1(c)\ \pi_2(c)) : (S \implies T) \wedge S \implies T$$

2 Philosophy

There is a logical interpretation of type theory.

For example, we might have the type 12 is composition, a proof might be $12 = 3 \cdot 4$. Another proposition/type would be T has a term, a proof might require hypotheses/context, and we could use the context to produce a term of T.
Example 2.1. $f : S \implies T$ is a function that turns a proof of S into a proof of T.

For example, if $S =$ "12 is composite.", and $T =$ "24 is composite.".
Example 2.2. A term of the type $c : S \wedge T$ represents a proof of S and T, since from c we can obtain proofs $\pi_1(c) : S$ and $\pi_2(c) : T$.

As a silly example, we could say $S =$ "12 is composite", and $T =$ "4 is composite".

In the logical interpretation, the term we just constructed of type $(S \implies T) \wedge S \implies T$, tells us that if we know $P \implies Q$, and P, then Q is true. This rule is called *modus ponens* in propositional logic.

This logic/program correspondence is called the Curry-Howard correspondence, and dates from 58-69. It is also called the Proofs-as-programs paradigm.

Usually in mathematics, proofs are often thought of as metamathematical. In type theory, proofs are actual mathematical objects. Thus we say type theory is *proof relevant*. It is often important that there is more than one proof.

Note that in type theory, proofs are always constructive. This is because we are always constructing a term in a type. This is constructive mathematics, and we don't use the law of excluded middle.

3 Disjunction: ∨-types

Definition 3.1. Rules for ∨-types.

1. type construction:
$$\frac{S \text{ TYPE}, \quad T \text{ TYPE}}{S \vee T \text{ TYPE}}$$
2. term construction:
$$\frac{\Gamma \vdash s : S}{\Gamma \vdash i_1(s) : S \vee T}$$
$$\frac{\Gamma \vdash t : T}{\Gamma \vdash i_2(t) : S \vee T}$$
3. term deconstruction:
$$\frac{\Gamma, x : S \vdash r : R, \quad \Gamma, y : T \vdash r' : R}{\Gamma, z : S \vee T \vdash j_{r,r'}(z) : R}$$
4. compatibility
$$\Gamma, x : S \vdash j_{r,r'}(i_1(x)) = r : R$$
$$\Gamma, y : S \vdash j_{r,r'}(i_2(y)) = r' : R$$

To construct a term of $S \vee T$, we can either supply an $s : S$, or a $t : T$. In the logical interpretation, we can say that to prove $S \vee T$, it suffices to prove S or T.

To construct a term of R from a term of $S \vee T$, we can supply functions $f : S \implies R$, and $g : T \implies R$. In the logical interpretation, this says that if we want to prove R using $S \vee T$, we can prove $S \implies R$ and $S \implies T$.

Example 3.1. From these rules, we get some functions:

1. $i_1 = \lambda s.i_1(s) : S \implies S \vee T$,
2. $i_2 = \lambda t.i_2(t) : T \implies S \vee T$,

Now we want to prove
$$R \wedge (S \vee T) \implies (R \wedge S) \vee (R \wedge T).$$

We begin with
$$\frac{x : R, y : S \vdash i_1(x, y) : (R \wedge S) \vee (R \wedge T), \quad x : R, z : T \vdash i_2(x, z) : (R \wedge S) \vee (R \wedge T)}{x : R, w : S \vee T \vdash j_{i_1,i_2}(x, w) : (R \wedge S) \vee (R \wedge T)}$$

Then we can conclude
$$v : R \wedge (S \vee T) \vdash j_{i_1,i_2}(\pi_1 v, \pi_2 v) : (R \wedge S) \vee (R \wedge T).$$

Finally, we lambda abstract to get
$$\vdash \lambda v.j_{i_1,i_2}(\pi_1 v, \pi_2 v) : R \wedge (S \vee T) \implies (R \wedge S) \vee (R \wedge T).$$

Question: In what sense is the string unambiguous? Can we work out things from the type? Answer: We're abbreviating slightly, but we can work out the types of the important things from the type of the whole expression.

Note from note taker, we're eliding a rather important piece from this proof, where we construct a function $i_r : S \implies R \wedge S$.

4 Bottom and Top

Definition 4.1. We have the rules for true or top: $\top$, and false or bottom: $\bot$

1.
$$\frac{}{\top \text{ TYPE}}, \quad \frac{}{* : \top}$$

2.
$$\frac{}{\bot \text{ TYPE}}, \quad \frac{\Gamma \vdash f : \bot, \quad S \text{ TYPE}}{\Gamma \vdash j_f : S}$$

Exercise 1. Prove $S \implies \top$, $\bot \implies S$.

Definition 4.2. Define $\neg S$ as $S \implies \bot$. We cannot construct a term of

$$S \vee \neg S$$

for every S.

This corresponds to unprovable statements in logic. This fact is related to Gödels incompleteness theorem.

A term of $S \vee \neg S$ is called the *law of excluded middle*.

We can't prove the law of excluded middle in type theory, and we can't prove its negation, so the law of excluded middle is *independent* of the theory.

We say a statment is *independent* of axioms/a proof system if when you add that statement, you cannot prove false, and when you add the negation of that statement, you still cannot prove false. The existence of independent statements means that the theory is incomplete.

We *model* theory in different categories. Since a model is more concrete, we often have statements that are true in the model that are not true in the theory.

When we construct models, we are looking for more complete systems.

We can construct a model in the category of sets. Types are sets, terms are elements, implies is the set of functions, $\wedge = \times$, $\vee = \sqcup$, $\top = \{*\}$, $\bot = \varnothing$. Then in **Set**,

$$S \vee \neg S = S \sqcup (S \to \varnothing).$$

Can we construct a term of this type? (I.e., an element of this set.) If $S = \varnothing$, then there is an element of $S \to \varnothing$, so $S \sqcup (S \to \varnothing)$ is nonempty, and if S is nonempty, then $S \sqcup (S \to \varnothing)$ is nonempty. Thus in either case, the law of excluded middle holds for **Set**.

The law of excluded middle fails for presheaves on the arrow category. I.e., the category

$$[2, \mathbf{Set}].$$

The objects are triples (A, B, f), with A, B, sets and $f : A \to B$ a function. The arrows in this category are pairs $(g, h) : (A, B, e) \to (C, D, f)$, of functions $g : A \to C$, $h : B \to D$ such that

$$\begin{array}{ccc} A & \xrightarrow{e} & B \\ \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle h} \\ C & \xrightarrow{f} & D \end{array}$$

commutes.

Then $\bot = (\varnothing, \varnothing, \mathrm{id})$, $\top = (*, *, \mathrm{id})$. Then we can show that for some objects S in this category, $S \vee \neg S$ is 'empty' generally. Terms of an object S here are elements of $\mathrm{Hom}(\top, S)$.