CSCI 8980 Higher-Dimensional Type Theory Lecture Notes

Zhuyang Wang, Bowen Wang

February {4}, 2020

1 Dependent Types

Dependent types can be seen as families of types indexed by other types. For example, if we want to represent the type of a fixed length vector, then it should be a family of types indexed by the natural number, which is its length. Suppose we already have the natural number type $\mathbb{N}$, the formation rule of the vector type $\texttt{Vec}$ then should be $x{:}\mathbb{N} \vdash \texttt{Vec}(x)\ \texttt{type}$. Thus $\texttt{Vec}(1)$ is the type of length-1 vector, and $\texttt{Vec}(0)$ is another type representing the type of empty vector. One can also think of dependent types as "functions", mapping inhabitants of some types to new types.

All the structural rules introduced in the last lecture are actually the dependent version. For example, the exchange rule has the requirement that "B does not depend on x". That said, other basic types are simply the non-dependent version, like the $\times$-type and the $\to$-type. In the dependent version of pair types, the type of the second element depends on the type of the first element, which makes the rule much more complicated. The following sections will show how to define the dependent version of function types, product types and disjoint sum types, by providing the answers to the five (or four) questions. Unit types and empty types are almost the same as the original ones.

2 Dependent Function Types

Dependent function types, also known as pi-types. There are several ways to represent a dependent function type: $\prod_{x:A} B$, $\prod_{x:A} B(x)$, or $\prod_{x:A} B_x$.

1. Formation
$$\frac{A\ \texttt{type} \qquad x{:}A \vdash B\ \texttt{type}}{\prod\limits_{x:A} B\ \texttt{type}}$$

2. Introduction
$$\frac{x{:}A \vdash M : B}{\lambda x.M : \prod\limits_{x:A} B}$$

3. Elimination
$$\frac{M : \prod\limits_{x:A} B \qquad N : A}{M\ N : B[N/x]}$$

4. Computation
$$\frac{x{:}A \vdash M : B \qquad N : A}{(\lambda x.M)\ N \equiv M[N/x] : B[N/x]}$$

5. Uniqueness
$$\frac{M : \prod\limits_{x:A} B}{M \equiv \lambda x.M\ x : \prod\limits_{x:A} B}$$

3 Dependent Pair Types

1. Formation:
$$\frac{A\ \texttt{type} \qquad x{:}A \vdash B\ \texttt{type}}{\sum\limits_{x:A} B\ \texttt{type}}$$

2. Introduction
$$\frac{M : A \qquad x{:}A \vdash B\ \texttt{type} \qquad N : B[M/x]}{\langle M, N\rangle : \sum\limits_{x:A} B}$$

3. Elimination
$$\frac{M : \sum\limits_{x:A} B}{\pi_1(M) : A} \qquad\qquad \frac{M : \sum\limits_{x:A} B}{\pi_2(M) : B[\pi_1(M)/x]}$$

The second elimination rule depends on the first one.

4. Computation

$$\frac{M : A \qquad x{:}A \vdash B\ \mathtt{type} \qquad N : B[M/x]}{\pi_1(\langle M, N\rangle) \equiv M : A}$$

$$\frac{M : A \qquad x{:}A \vdash B\ \mathtt{type} \qquad N : B[M/x]}{\pi_2(\langle M, N\rangle) \equiv N : B[M/x]}$$

$\pi_2(\langle M, N\rangle)$ is of type $B[M/x]$ because $\pi_1(\langle M, N\rangle) \equiv M : A$.

5. Uniqueness

$$\frac{M : \sum\limits_{x:A} B}{\langle \pi_1(M), \pi_2(M)\rangle \equiv M : \sum\limits_{x:A} B}$$

Notice that the elimination rules tell us $\pi_1(M) : A$ and $\pi_2(M) : B[\pi_1(M)/x]$. Then by the introduction rule we know $\langle \pi_1(M), \pi_2(M)\rangle : \sum\limits_{x:A} B$.

4 Disjoint Sum Types

1. Formation:

$$\frac{A\ \mathtt{type} \qquad B\ \mathtt{type}}{A + B\ \mathtt{type}}$$

2. Introduction:

$$\frac{M : A}{\mathtt{inl}(M) : A + B} \qquad\qquad \frac{M : B}{\mathtt{inr}(M) : A + B}$$

3. Elimination:

$$\frac{z{:}A + B \vdash C\ \mathtt{type} \qquad O : A + B \qquad x{:}A \vdash M : C[\mathtt{inl}(x)/z] \qquad y{:}B \vdash N : C[\mathtt{inr}(y)/z]}{\mathtt{case}[z.C](x.M; y.N; O) : C[O/z]}$$

4. Computation:

$$\frac{z{:}A + B \vdash C\ \mathtt{type} \qquad O : A \qquad x{:}A \vdash M : C[\mathtt{inl}(x)/z] \qquad y{:}B \vdash N : C[\mathtt{inr}(y)/z]}{\mathtt{case}[z.C](x.M; y.N; \mathtt{inl}(O)) \equiv M[O/x] : C[O/z]}$$

$$\frac{z{:}A + B \vdash C\ \mathtt{type} \qquad O : B \qquad x{:}A \vdash M : C[\mathtt{inl}(x)/z] \qquad y{:}B \vdash N : C[\mathtt{inr}(y)/z]}{\mathtt{case}[z.C](x.M; y.N; \mathtt{inr}(O)) \equiv N[O/y] : C[O/z]}$$

5 Polarity

Classification	Category Theory	Sequent Calculus	Type Theory
Negative	mapping in universal property	right rules invertible	$\prod \sum \times \to \top$
Positive	mapping out universal property	left rules invertible	$+ \bot$

Difference between negative and positive in type theory:

1. Negative: The introduction rule is constructed from the elimination rule.
2. Positive: The elimination rule is constructed from the introduction rule.

However, we know there are two possible ways to define the pair type. The first one is using the two projections we present here. The other way is using the `split` operator mentioned in the previous lecture. If we define the pair type using the second way, its polarity then becomes positive.

6 Martin-Löf ethos/mythos

Every "cool" type is an internalization of some judgmental structure.

1. $\prod$-type $\lambda x.M : \prod\limits_{x:A} B$ corresponds to the entailment $x{:}A \vdash M : B$.
2. $\sum$-type $\sum\limits_{x:\Gamma} A$ corresponds to the context extension Γ, A.
3. The idenditity type corresponds to the judgmental equality rules.
4. The universe type corresponds to the A `type` judgments.