CSCI 8980 Higher-Dimensional Type Theory
Lecture Notes

Favonia

January {21,28}, 2020

1 Principles of Type Theory

Type theory serves as a foundation of mathematics alternative to set theory. Although it is difficult to pin down a precise definition of type theory, we have pretty good ideas about what principles a type theory should follow. Here are two important principles:

1. Type theory is about **abstractions.**

   Type theory comes with built-in abstractions. For example, in many type theories with natural numbers, the number zero and the successor operator are primitive notions, not defined in terms of other more "basic" notions; in contrast, all numbers and operators are encoded as sets in set theory. Type theory is arguably closer to how mathematicians actually work in practice: for example, a reasonable proof in number theory should not depend on how natural numbers are encoded. Type theory formally enforces such abstractions.

   One may draw an analogy from Euclid's famous book Elements, which starts with abstract notions of points and lines without using the Cartesian coordinates. The Homotopy Type Theory, which we will introduce in this semester, can be seen as abstractions of topological spaces up to continuous deformations without using point topology.

2. Type theory is about **constructions.**

   Another important feature of most type theories is to treat *being true* as *having a proof*. We pay close attention to how proofs are constructed. There are two significant consequences of this principle:

- Type theory works well for computer science. The types are specifications and proofs are programs. It is not surprising that almost every programming language employs some type system to classify its programs.
- The global law of excluded middle (LEM) (which means, for any formula A, either A or $\neg A$) no longer makes sense, because it is unreasonable to assume a procedure that can give either a proof or a refutation for any type A. The A can be the Riemann Hypothesis. That said, for specific A, say, whether two natural numbers are equal, we can either prove A or refute A because the equality can be determined algorithmically.

Note that, even though the global LEM is usually not valid in a type theory, we can show the law is *irrefutable* ($\neg\neg(A + \neg A)$), which means while we cannot prove a global LEM, we cannot refute it either.[1]

2 How to Define a Type Theory?

2.1 Judgments

Judgments represent the knowledge in a type theory. They encode mathematical statements such that "$\lambda(x{:}A).x$ is of type $A \to A$." For now, we will focus on the following four basic judgment forms:

Basic Judgments	Meanings
$A\ \mathtt{type}$	A is a type
$A \equiv B$	A and B are equal types
$M : A$	M is an element of type A
$M \equiv N : A$	M and N are the same element of type A

We can then define *hypothetical* judgements, which are basic judgments parametrized by variables. The collection of available variables form the *context*. Formally, a context is a list of hypotheses, i.e., variables annotated with their types,[2]

$$x_1{:}A_1, x_1{:}A_2, \cdots, x_n{:}A_n$$

[1]This is a difficult theorem that you should try to prove in Agda.

[2]We will introduce new forms of hypotheses in the later part of this course.

which says variable x_i is of type A_i. A hypothetical judgment takes the following form:

$$x_1{:}A_1, x_1{:}A_2, \cdots, x_n{:}A_n \vdash \mathcal{J}$$

which says $\mathcal{J}$ holds under the hypotheses $x_1{:}A_1, x_1{:}A_2, \cdots, x_n{:}A_n$. As a special case, $\cdot$ is the empty context and

$$\cdot \vdash \mathcal{J}$$

emphasizes that $\mathcal{J}$ holds with access to no variables.

2.2 Inference Rules

A type theory is specified by a collection of *(inference) rules*, which represent all the valid reasoning steps in the theory. To start with, a type theory might have this rule:

$$\frac{A \equiv B}{B \equiv A}$$

which states the symmetry of type equality. It might also have this one:

$$\frac{A\ \texttt{type}}{A \equiv A}$$

which states every type is equal to itself. The general form of an inference rule consists of a finite number of premises and one conclusion, separated by a horizontal line as follows:

$$\frac{\mathcal{J}_1 \quad \mathcal{J}_2 \quad \cdots \quad \mathcal{J}_n}{\mathcal{J}}\ \text{(rule name)}$$

It means that, if the premises $\mathcal{J}_1, \mathcal{J}_2, \cdots, \mathcal{J}_n$ are valid, then the conclusion $\mathcal{J}$ is valid, too. If we manage to prove a judgment $\mathcal{J}$ using the inference rules in a type theory, the history of the reasoning may be represented as a tree, which is called *derivation*. For example, a possible derivation is as follows:

$$\frac{\overline{\top\ \texttt{type}}}{\top \equiv \top}$$

We say a judgment $\mathcal{J}$ is *derivable* if there is a complete derivation tree whose root (the bottommost judgment) is the judgment $\mathcal{J}$.

3 Structural Rules

These rules are called *structural rules* because they do not mention any connectives or type constructors. For example, we can force the type equality to be an equivalence relation:

$$\frac{A \equiv B}{B \equiv A} \qquad \frac{A \ \texttt{type}}{A \equiv A} \qquad \frac{A \equiv B \quad B \equiv C}{A \equiv C}$$

Note that the rule

$$\frac{A \equiv B}{B \equiv A}$$

actually means

$$\frac{\Gamma \vdash A \equiv B}{\Gamma \vdash B \equiv A}$$

It is very common to omit ambient contexts Γ that do not interact with other parts of a rule.

It is encouraged to read *Homotopy Type Theory: Univalent Foundations for Mathematics*, Appendix A.2 to see what structural rules a complete type theory might have. One of the most important rules is the *variable rule*, which enables us to use the hypotheses in the context:

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x : A} \text{ (variable)}$$

In addition to the variable rule, there are several rules that are common in a type theory:

$$\frac{\Gamma \vdash \mathcal{J}}{\Gamma, x{:}A \vdash \mathcal{J}} \text{ (weakening)} \qquad \frac{\Gamma, x{:}A, y{:}A, \Delta \vdash \mathcal{J}}{\Gamma, z{:}A, \Delta \vdash \mathcal{J}[z, z/x, y]} \text{ (contraction)}$$

$$\frac{\Gamma, x{:}A, y{:}B, \Delta \vdash \mathcal{J} \qquad (B \text{ does not depend on x})}{\Gamma, y{:}B, x{:}A, \Delta \vdash \mathcal{J}} \text{ (exchange)}$$

$$\frac{\Gamma, x{:}A, \Delta \vdash \mathcal{J} \qquad \Gamma \vdash M : A}{\Gamma, \Delta[M/x] \vdash \mathcal{J}[M/x]} \text{ (substitution)}$$

These rules are either part of the definition of a type theory or are *admissible*, meaning that the theory is carefully arranged so that the rules are are *provable* in the following sense.

Definition 1 (admissible rules)**.** *An inference rule*

$$\frac{\mathcal{J}_1 \quad \mathcal{J}_2 \quad \cdots \quad \mathcal{J}_n}{\mathcal{J}}$$

in a theory $\mathcal{T}$ if whenever there are derivations of the judgments $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_n$, there is a derivation of the judgment $\mathcal{J}$.

In other words, an admissible rule is redundant, not adding new power in proving judgments.

4 How to Define a Type?

Since type theory is about abstractions, there will be many types capturing different abstractions. To define types, one should answer the following five questions:

1. Formation: What are the types?
2. Introduction: What are the canonical ways to construct elements in the types?
3. Elimination: How to use an element of those types?
4. Computation: What will happen when you are using elements constructed using the canonical ways?
5. Uniqueness: Is any element equal to some element constructed in one of the canonical ways? (This is optional.)

5 Basic Types

For each type, we are going to answer the above five questions using judgments.

5.1 Pair Types

1. Formation:

$$\frac{A \; \texttt{type} \qquad B \; \texttt{type}}{A \times B \; \texttt{type}}$$

2. Introduction:
$$\frac{M : A \qquad N : B}{\langle M, N \rangle : A \times B}$$

3. Elimination:
$$\frac{M : A \times B}{\pi_1(M) : A} \qquad \frac{M : A \times B}{\pi_2(M) : B}$$

4. Computation:
$$\frac{M : A \qquad M : B}{\pi_1(\langle M, N \rangle) \equiv M : A} \qquad \frac{M : A \qquad M : B}{\pi_2(\langle M, N \rangle) \equiv N : B}$$

5. Uniqueness:
$$\frac{M : A \times B}{M \equiv \langle \pi_1(M), \pi_2(M) \rangle : A \times B}$$

5.2 Disjoint Sum Types

1. Formation:
$$\frac{A\ \texttt{type} \qquad B\ \texttt{type}}{A + B\ \texttt{type}}$$

2. Introduction:
$$\frac{M : A}{\texttt{inl}(M) : A + B} \qquad \frac{M : B}{\texttt{inr}(M) : A + B}$$

3. Elimination:
$$\frac{x{:}A \vdash M : C \qquad y{:}B \vdash N : C \qquad O : A + B}{\texttt{case}(x.M; y.N; O) : C}$$

4. Computation:
$$\frac{x{:}A \vdash M : C \qquad y{:}B \vdash N : C \qquad O : A}{\texttt{case}(x.M; y.N; \texttt{inl}(O)) \equiv M[O/x] : C}$$
$$\frac{x{:}A \vdash M : C \qquad y{:}B \vdash N : C \qquad O : B}{\texttt{case}(x.M; y.N; \texttt{inr}(O)) \equiv N[O/y] : C}$$

5.3 The Unit Type

1. Formation:
$$\frac{}{\top\ \texttt{type}}$$
2. Introduction:
$$\frac{}{\diamond : \top}$$
3. Elimination: (no rules)
4. Computation: (no rules)
5. Uniqueness:
$$\frac{M : \top}{M \equiv \diamond : \top}$$

5.4 The Bottom Type

1. Formation:
$$\frac{}{\bot\ \texttt{type}}$$
2. Introduction: (no rules)
3. Elimination:
$$\frac{M : \bot}{\texttt{abort}(M) : C}$$
4. Computation: (no rules)

5.5 Function Types

1. Formation:
$$\frac{A\ \texttt{type} \qquad B\ \texttt{type}}{A \longrightarrow B\ \texttt{type}}$$
2. Introduction:
$$\frac{x{:}A \vdash M : B}{\lambda x.M : A \longrightarrow B}$$
3. Elimination:
$$\frac{M : A \longrightarrow B \qquad N : A}{M\ N : B}$$

4. Computation:

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

5. Uniqueness:

$$\frac{M : A \to B}{M \equiv \lambda x.M\ x : A \to B}$$

6 Principle of Harmony

One important principle is the harmony between the answers to **Question (2)** (introductions) and **Question (3)** (eliminations). They need to satisfy the law of conservation of knowledge, like the law of conservation of energy in physics. The data you put in using the answers in **Question (2)** must equal to what you can learn via the answers to **Question (3).** For example, there are two projections from a pair, and thus it takes two components to make a pair. There are two (canonical) ways to make an element of a disjoint sum type, and thus there are two branches in `case`. There are no projection from an element of the unit type, and thus it takes nothing to construct an element of that type.

The principle of harmony is important because it *often* (not as a mathematical theorem yet) implies that everything eventually reduces to some canonical form if it does not have free variables. For example, one can show that any pair (without free variables) will eventually reduce to a term of the form $\langle M, N \rangle$, and thus a projection out of the pair will eventually be resolved successfully, using the answers to **Question (4)** (computations). Suppose, as a counter example, we introduced a new form of pair, $\bigcirc$:

$$\frac{}{\bigcirc : A \times B}$$

This new rule will break the harmony because the projections $\pi_1(\bigcirc)$ and $\pi_2(\bigcirc)$ would be stuck. Thus, the harmony is a necessary condition of the global property that everything reduces. In many cases, the harmony of all types is a sufficient condition as well..

References

[1] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations for Mathematics*. Institute for Advanced Study, 2013. URL: `http://homotopytypetheory.org/book`.