Intro to
Homotopy Type Theory

Martin-Löf Type Theory

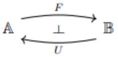
The Language of Homotopy Type Theory

What is MLTT?

Martin-Löf Type Theory is a **formal language** and **deductive system** which has the form of an abstract **typed programming language** and can be used to reason about both the **topology of higher-dimensional spaces** and **higher-order intuitionistic logic**.

0 Speaking the Language

iff



Proof. The proof i[...] legs of a naturality square for a natural [...]ism of presheaves):

$$\phi : \mathrm{Hom}_{\mathbb{B}}(F(-),-) \xrightarrow{\sim} \mathrm{Hom}_{\mathbb{A}}(-,U(-))$$

$$\begin{array}{ccc} C(c,c) & \overset{\eta_c}{\to} & X(c) \\ {\scriptstyle C(f,c)}\downarrow & & \downarrow{\scriptstyle X(f)} \\ C(b,c) & \underset{\eta_b}{\to} & X(b) \end{array} \qquad \begin{array}{ccc} \mathrm{Id}_c & \mapsto & \eta_c(\mathrm{Id}_c) \overset{\text{def}}{=} \xi \\ \downarrow & & \downarrow{\scriptstyle X(f)} \\ f & \mapsto & \eta_b(f) \end{array}$$

surjective

acyclic

monotone

homeomorphism

TFAE

What this diagram shows is that the entire transformation $\eta : C(-,c) \to X$ is completely determined from the single value $\xi \coloneqq \eta_c(\mathrm{Id}_c) \in X(c)$, because for each object b of C, the component $\eta_b : C(b,c) \to X(b)$ must take an element $f \in C(b,c)$ (i.e., a morphism $f : b \to c$) to $X(f)(\xi)$, according to the commutativity of this diagram.

The crucial point is that the naturality condition on any natural transformation $\eta : C(-,c) \Rightarrow X$ is sufficient to ensure that η is already entirely fixed by the value $\eta_c(\mathrm{Id}_c) \in X(c)$ of its component $\eta_c : C(c,c) \to X(c)$ on the identity morphism Id_c. And every such value extends to a natural transformation η.

bisimilar

More in detail, the bijection is established by the map

$$[C^{\mathrm{op}}, \mathrm{Set}](C(-,c), X) \xrightarrow{|_c} \mathrm{Set}(C(c,c), X(c)) \xrightarrow{\mathrm{ev}_{\mathrm{Id}_c}} X(c)$$

abelian

where the first step is taking the component of a natural transformation at $c \in C$ and the second step is evaluation at $\mathrm{Id}_c \in C(c,c)$.

integrable

The inverse of this map takes $f \in X(c)$ to the natural transformation η^f with components

WLOG

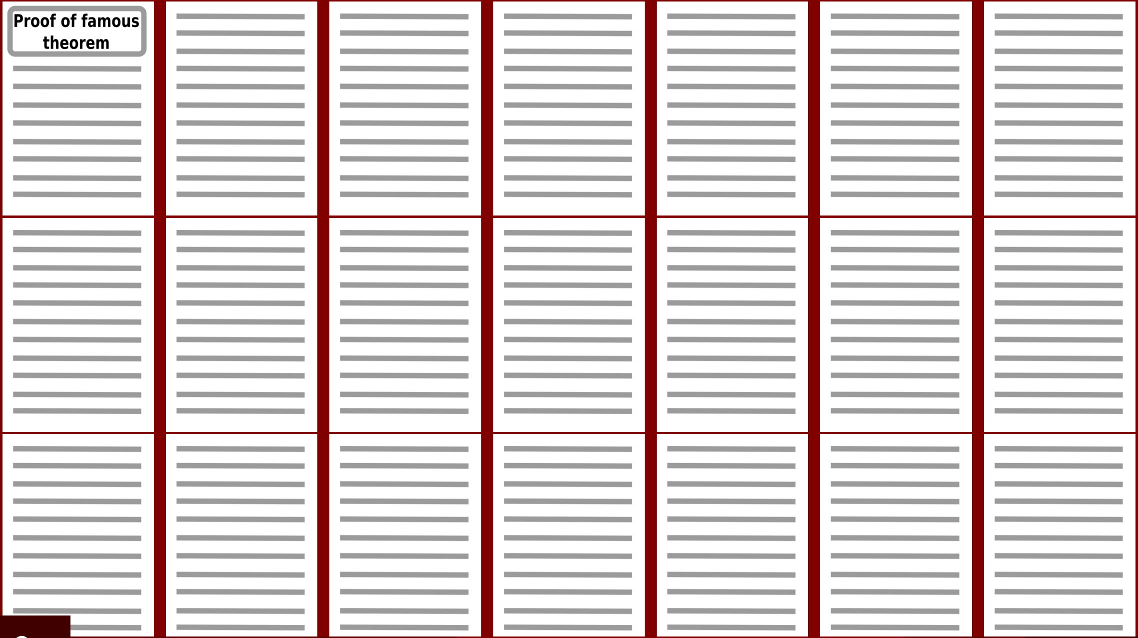
$$\iint_{\Sigma} (\nabla \times \mathbf{A}) \cdot d\mathbf{a} = \oint_{\partial\Sigma} \mathbf{A} \cdot d\mathbf{l}.$$

contravariant

$$\eta^f_d : = X(-)(f) : C(d,c) \to X(d).$$

∎

https://ncatlab.org/nlab/show/Yoneda+lemma


Proof of famous theorem

Titans of Mathematics Clash Over Epic Proof of ABC Conjecture

Two mathematicians have found what they say is a hole at the heart of a proof that has convulsed the mathematics community for nearly six years.

Despite multiple conferences dedicated to explicating Mochizuki's proof, number theorists have struggled to come to grips with its underlying ideas. His series of papers, which total more than 500 pages, are written in an impenetrable style, and refer back to a further 500 pages or so of previous work by Mochizuki, creating what one mathematician, Brian Conrad of Stanford University, has called "a sense of infinite regress."

But the meeting led to an oddly unsatisfying conclusion: Mochizuki couldn't convince Scholze and Stix that his argument was sound, but they couldn't convince him that it was unsound. Mochizuki has now posted Scholze's and Stix's report on his website, along with several reports of his own in rebuttal. (Mochizuki and Hoshi did not respond to requests for comments for this article.)

Language & Deduction

❄♒♏ ♐◆■♎♋❍♏■⧫♋● ♑❒□◆□ □♐ ♋
⧫⍓□♏ ♓⬧ ⧫♒♏ ⌘♏❒□✍⧫❒◆■♍♋⧫♓□■
□♐ ❄✍ ❖♓♏⬥♏♎ ♋⬧ ♋ ♑❒□◆□

⬧♈🗁 ♓⬧ ♎♏♐♓■♏♎ ♌⍓
♌♋⬧♏ 💻 ⬧♈🗁
●□□□ 💻 ♌♋⬧♏ 🖬 ♌♋⬧♏

Therefore...

❄♒♏ ♐◆■♎♋❍♏■⧫♋● ♑❒□◆□ □♐ ⬧♈🗁 ♓⬧
⧫♒♏ ♓■⧫♏♑♏❒⬧

There are certain general conditions under which the structure of a language is regarded as *exactly specified*. Thus, to specify the structure of a language, we must characterize unambiguously the class of those words and expressions which are to be considered *meaningful*. In particular, we must indicate all words which we decide to use without defining them, and which are called "*undefined* (or *primitive*) terms"; and we must give the so-called *rules of definition* for introducing new or *defined terms*. Furthermore, we must set up criteria for distinguishing within the class of expressions those which we call "*sentences*." Finally, we must formulate the conditions under which a sentence of the language can be *asserted*. In particular, we must indicate all *axioms* (or *primitive sentences*), i.e., those sentences which we decide to assert without proof; and we must give the so-called *rules of inference* (or *rules of proof*) by means of which we can deduce new asserted sentences from other sentences which have been previously asserted. Axioms, as well as sentences deduced from them by means of rules of inference, are referred to as "*theorems*" or "*provable sentences*."

- Alfred Tarski, The Semantic Conception of Truth (1944)

Terms

```
>    b=0
>    if (b=4 or b=5):
>      do_thing1()
>    else:
>      do_thing2()
```

Two Judgments of MLTT

$$\underset{\text{Term}}{x} : \underset{\text{Type}}{T}$$

$$\underset{\text{Judgmental Equality}}{x \doteq x'} : T$$

❄♒♏ ♐◆■♎♋❍♏■⧫♋● ♑❒□◆◻ □♐ ♋
⧫⍓◻♏ ❄ ♓⬧ ⧫♒♏ ⌘♏❒□📪⧫❒◆■♍♋⧫♓□■
□♐ ❄📫 ❖♓♏⬥♏♎ ♋⬧ ♋ ♑❒□◆◻

💧♈🗁 ♓⬧ ♎♏♐♓■♏♎ ♌⍓
♌♋⬧♏ 🖳 💧♈🗁
●□□◻ 🖳 ♌♋⬧♏ 🖫 ♌♋⬧♏

Therefore...

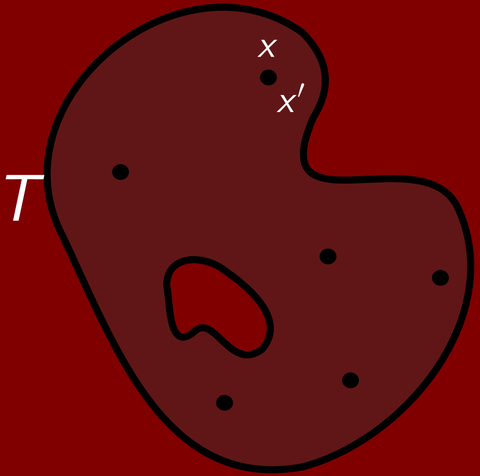
❄♒♏ ♐◆■♎♋❍♏■⧫♋● ♑❒□◆◻ □♐ 💧♈🗁 ♓⬧
⧫♒♏ ♓■⧫♏♑♏❒⬧

Interpretation 1: Spaces

$$x : T$$

$$x \doteq x' : T$$

Types – Spaces
Terms – Points



Interpretation 2: Logic (Curry-Howard)

$$w : P$$

witness Proposition

$$w \doteq w' : P$$

Equality of witnesses

Types – Propositions

Terms – Witnesses

- Inhabited propositions are "true"
- Uninhabited propositions are "false"

We informally say "assume that P" to mean "assume there is a term of type P"

This logic is **proof-relevant**: there may be distinct witnesses of the same proposition

Interpretation

- Type Theory: MLTT describes *terms* and *types*
- Homotopy: MLTT describes *points* and *spaces*
- Logic: MLTT describes *witnesses* and *propositions*

By discussing these in a common language, we can

- identify similarities
- "transpose" concepts

1 Judgments, Contexts, and Types

Previously on Intro to HoTT...

Four Judgments of MLTT

$$T \text{ type}$$

$$x : T$$

$$T \doteq T' \text{ type}$$

$$x \doteq x' : T$$

What MLTT is made of

- Types (built up recursively, along with the terms-in-context)
- Terms-in-context (built up recursively, along with the types)
- Contexts
- Inference Rules
- Derivations

Contexts give MLTT "memory"

Suppose we have types $T_1, \ldots, T_n$. A **context** consists of a finite (possibly empty), ordered list of typing judgments

$$x_1 : T_1, x_2 : T_2(x_1), \ldots, x_n : T_n(x_1, \ldots, x_{n-1})$$

- Type Theory: Declaring some typed variables
- Logic: Assuming the truth of some propositions (with witnesses)
- Homotopy: Declaring names for points of given spaces

Judgments-in-Context

Let Γ be a context.

$$\Gamma \vdash \mathcal{J}$$

$$\Gamma \vdash T \text{ type}$$

$$\Gamma \vdash x : T$$

$$\Gamma \vdash T \doteq T' \text{ type}$$

Inference Rules

An **inference rule** is of the form

$$\frac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \cdots \quad \mathcal{H}_k}{\mathcal{C}}$$

For instance,

$$\frac{\Gamma \vdash T \text{ type}}{\Gamma \vdash T \doteq T \text{ type}} \qquad \frac{\Gamma \vdash T \text{ type} \quad \Gamma \vdash \mathcal{J}}{\Gamma, x : T \vdash \mathcal{J}}$$

Example: Booleans

```
$  true
>      true :  bool
$  false
>      false :  bool
$  (if true
$   then 5
$   else 4)
>      5
$  (if false then 5 else 4)
>      4
```

Example: Booleans

The type of booleans will be denoted $\mathbf{2}$ and contain exactly two terms, $0_{\mathbf{2}}$ and $1_{\mathbf{2}}$. We'll formally express this using inference rules.

- **Formation**:

$$\frac{}{\Gamma \vdash \mathbf{2}\ \mathsf{type}}$$

- **Introduction**:

$$\frac{}{\Gamma \vdash 0_{\mathbf{2}} : \mathbf{2}} \qquad \frac{}{\Gamma \vdash 1_{\mathbf{2}} : \mathbf{2}}$$

Boolean Elimination & Computation (non-dependent)

- **Elimination**

$$\frac{\Gamma \vdash T \text{ type} \quad \Gamma \vdash p_0 : T \quad \Gamma \vdash p_1 : T}{\Gamma, x : \mathbf{2} \vdash \mathrm{ind}_{\mathbf{2}}(p_0, p_1, x) : T}$$

- **Computation**:

$$\frac{\Gamma \vdash T \text{ type} \quad \Gamma \vdash p_0 : T \quad \Gamma \vdash p_1 : T}{\Gamma \vdash \mathrm{ind}_{\mathbf{2}}(p_0, p_1, 0_{\mathbf{2}}) \doteq p_0 : T}$$

$$\frac{\Gamma \vdash T \text{ type} \quad \Gamma \vdash p_0 : T \quad \Gamma \vdash p_1 : T}{\Gamma \vdash \mathrm{ind}_{\mathbf{2}}(p_0, p_1, 1_{\mathbf{2}}) \doteq p_1 : T}$$

Example: Binary Products

Example: Binary Products

- Formation:

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \times B \text{ type}}$$

- Introduction:

$$\frac{\Gamma \vdash x : A \quad \Gamma \vdash y : B}{\Gamma \vdash (x, y) : A \times B}$$

(also need "Congruence Rule" to state that if $x \doteq x'$ and $y \doteq y'$, then $(x, y) \doteq (x', y')$)

- Elimination and Computation: Next time!

Check Your Understanding

- List the terms of type $\mathbf{2} \times \mathbf{2}$
- Given terms $b_1 : \mathbf{2}$ and $b_2 : \mathbf{2}$, use $\mathsf{ind}_{\mathbf{2}}$ to come up with
  - a term $b_3 : \mathbf{2}$ which is judgmentally equal to $1_{\mathbf{2}}$ if $b_1 \doteq 0_{\mathbf{2}}$, and $0_{\mathbf{2}}$ if $b_1 \doteq 1_{\mathbf{2}}$
  - a term $b_4 : \mathbf{2}$ which is judgmentally equal to $1_{\mathbf{2}}$ if both b_1 and b_2 are judgmentally equal to $1_{\mathbf{2}}$, and $0_{\mathbf{2}}$ otherwise
  - a term $b_5 : \mathbf{2}$ which is judgmentally equal to $1_{\mathbf{2}}$ if either $b_1 \doteq 1_{\mathbf{2}}$ or $b_2 \doteq 1_{\mathbf{2}}$, and $0_{\mathbf{2}}$ otherwise
- Verify that, up to a trivial relabelling, $(A \times B) \times C$ has the same terms as $A \times (B \times C)$
- Give the analogous introduction of a type $\mathbf{3}$ with exactly three terms.
- How many terms are there of type $\mathbf{2} \times \mathbf{2} \times \mathbf{3}$?

Example: Arrow Types

Claim For any sets A, B,

$$A \cap B \subseteq A$$

i.e. $x \in A \cap B$ implies $x \in A$.

Proof. Assume $x \in A \cap B$. Then we have $x \in A$ by definition of set intersection. So $x \in A$, as desired. □

Proof Relevant Version

Claim For any sets A, B,

$$A \cap B \subseteq A$$

i.e. there is a witness of $(x \in A \cap B) \to (x \in A)$.

Proof. Given $h : (x \in A \cap B)$, we have $h_1 : (x \in A)$ and $h_2 : (x \in B)$ by definition of set intersection. So output $h_1 : (x \in A)$. □

Moral

In proof-relevant mathematics, a proof of $P \to Q$ is a transformation converting witnesses of P into witnesses of Q.

Modus Ponens

$$\frac{h : P(x \in A \cap B) \quad f : P \to Q(x \in A \cap B) \to (x \in A}{f(h) : Q(x \in A)}$$

Lambda Expressions

$$\lambda x.e$$

$$(\lambda h.h_1) : (x \in A \cap B) \to (x \in A)$$

Check Your Understanding

Write terms of the following types

- $P \to P$
- $P \to (Q \to P)$
- $(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$
- $(Q \to R) \to ((P \to Q) \to (P \to R))$

Example Solutions

- $P \to P$

$$\lambda h.h$$

- $P \to (Q \to P)$

$$\lambda p.\lambda q.p$$

Example: Arrow Types

- Formation:

$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash B \text{ type}}{\Gamma \vdash A \to B \text{ type}}$$

- Introduction:

$$\frac{\Gamma, x : A \vdash e(x) : B}{\Gamma \vdash (\lambda x.e(x)) : A \to B}\ \lambda$$

(also need "Congruence Rule" to state that if $e(x) \doteq e'(x)$ for arbitrary x, then $(\lambda x.e(x)) \doteq (\lambda x.e'(x))$)

- Elimination:

$$\frac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash f(x) : B}\ \text{ev}$$

- Computation:

$$\frac{\Gamma, x : A \vdash e(x) : B}{\Gamma, x : A \vdash (\lambda y.e(y))(x) \doteq e(x) : B}\ \beta$$

$$\frac{\Gamma \vdash f : A \to B}{\Gamma \vdash (\lambda x.f(x)) \doteq f : A \to B}\ \eta$$

Summary

	$\mathbf{2}$	$\times$	$\to$
Type Theory	Booleans	?	Functions
Homotopy	Discrete 2-point space	Product spaces	?
Logic	?	Conjunction	Implication

2 Deduction in MLTT

Idea

$$\dfrac{\dfrac{\begin{matrix}\mathcal{H}_1\quad\mathcal{H}_2\\ \vdots\\ \mathcal{J}_{2,1}\end{matrix}\quad \mathcal{H}_3}{\mathcal{J}_{1,1}} \quad \dfrac{\begin{matrix}\mathcal{H}_4\\ \vdots\\ \mathcal{J}_{2,2}\end{matrix}}{\mathcal{J}_{1,2}} \quad \dfrac{\mathcal{H}_5\quad\mathcal{H}_6\quad\mathcal{H}_7}{\mathcal{J}_{1,3}} \quad \dfrac{}{\mathcal{J}_{1,4}} \quad \dfrac{\dfrac{\dfrac{\mathcal{H}_8}{\mathcal{J}_{3,1}}\quad\mathcal{H}_9}{\mathcal{J}_{2,3}}\quad\dfrac{\mathcal{H}_{10}\quad\mathcal{H}_{11}}{\mathcal{J}_{2,4}}}{\mathcal{J}_{1,5}}}{\mathcal{C}}$$

is a deduction of

$$\dfrac{\mathcal{H}_1\quad\mathcal{H}_2\quad\ldots\mathcal{H}_{11}}{\mathcal{C}}$$

Idea

A derived rule

$$\frac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \ldots \quad \mathcal{H}_k}{\mathcal{C}}$$

can

- Be used to derive more rules
- Serve as a formally-proven theorem about how our type theory works

We'll need some simple rules to make our deduction system work.

Judgmental Equality is an equivalence relation

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \doteq A \text{ type}} \quad \frac{\Gamma \vdash A \doteq B \text{ type}}{\Gamma \vdash B \doteq A \text{ type}} \quad \frac{\Gamma \vdash A \doteq B \text{ type} \quad \Gamma \vdash B \doteq C \text{ type}}{\Gamma \vdash A \doteq C \text{ type}}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \doteq a : A} \quad \frac{\Gamma \vdash a \doteq b : A}{\Gamma \vdash b \doteq a : A} \quad \frac{\Gamma \vdash a \doteq b : A \quad \Gamma \vdash b \doteq c : A}{\Gamma \vdash a \doteq c : A}$$

Variable Rule and Weakening

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A}\ \delta$$

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma, \Delta \vdash \mathcal{J}}{\Gamma, x : A, \Delta \vdash \mathcal{J}}\ W$$

Allows us to define the **constant type family** B over A:

$$\frac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, x : A \vdash B \text{ type}}\ W$$

Moving variables around

- Variable Conversion Rule

$$\frac{\Gamma \vdash A \doteq A' \quad \Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, x : A', \Delta \vdash \mathcal{J}}$$

Substitution

Moving variables around

- Substitution Rule

$$\frac{\Gamma \vdash a : A \quad \Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, \Delta[a/x] \vdash \mathcal{J}[a/x]} \; S$$

- Substitution Congruence Rules

$$\frac{\Gamma \vdash a \doteq a' : A \quad \Gamma, x : A, \Delta \vdash B \text{ type}}{\Gamma, \Delta[a/x] \vdash B[a/x] \doteq B[a'/x] \text{ type}}$$

$$\frac{\Gamma \vdash a \doteq a' : A \quad \Gamma, x : A, \Delta \vdash b : B}{\Gamma, \Delta[a/x] \vdash b[a/x] \doteq b[a'/x] : B[a/x]}$$

Derived Structural Rules

- Substituting with a fresh variable

$$\frac{\Gamma, x : A, \Delta \vdash \mathcal{J}}{\Gamma, x' : A, \Delta[x'/x] \vdash \mathcal{J}[x'/x]}\ x'/x$$

- Interchange rule

$$\frac{\Gamma \vdash B \text{ type} \quad \Gamma, x : A, y : B, \Delta \vdash \mathcal{J}}{\Gamma, y : B, x : A, \Delta \vdash \mathcal{J}}$$

Derivation

$\mathbf{2} \rightarrow \mathbf{2}$

$\mathbf{2} \to \mathbf{2}$

$\mathbf{2} \to \mathbf{2}$

$$\dfrac{\dfrac{\overline{\Gamma \vdash 0_{\mathbf{2}} : \mathbf{2}}}{\Gamma, x : \mathbf{2} \vdash 0_{\mathbf{2}} : \mathbf{2}}\, W}{\Gamma \vdash (\lambda x.0_{\mathbf{2}}) : \mathbf{2} \to \mathbf{2}}\, \lambda$$

$$\dfrac{\dfrac{\overline{\Gamma \vdash \mathbf{2}\ \mathsf{type}}}{\Gamma, x : \mathbf{2} \vdash x : \mathbf{2}}\, \delta}{\Gamma \vdash (\lambda x.x) : \mathbf{2} \to \mathbf{2}}\, \lambda$$

$$\dfrac{\dfrac{\overline{\Gamma \vdash 1_{\mathbf{2}} : \mathbf{2}}}{\Gamma, x : \mathbf{2} \vdash 1_{\mathbf{2}} : \mathbf{2}}\, W}{\Gamma \vdash (\lambda x.1_{\mathbf{2}}) : \mathbf{2} \to \mathbf{2}}\, \lambda$$

$$\dfrac{\dfrac{\overline{\Gamma \vdash \mathbf{2}\ \mathsf{type}} \quad \overline{\Gamma \vdash 1_{\mathbf{2}} : \mathbf{2}} \quad \overline{\Gamma \vdash 0_{\mathbf{2}} : \mathbf{2}}}{\Gamma, x : \mathbf{2} \vdash \mathsf{ind}_{\mathbf{2}}(1_{\mathbf{2}}, 0_{\mathbf{2}}, x) : \mathbf{2}}}{\Gamma \vdash (\lambda x.\mathsf{ind}_{\mathbf{2}}(1_{\mathbf{2}}, 0_{\mathbf{2}}, x)) : \mathbf{2} \to \mathbf{2}}\, \lambda$$

Appending Definitions To Derivations

$$\dfrac{\dfrac{\begin{matrix}\mathcal{H}_1 & \mathcal{H}_2 & & \mathcal{H}_k \\ \vdots & \vdots & \ldots & \vdots\end{matrix}}{\Gamma \vdash a : A}}{\Gamma \vdash c :\equiv a : A}$$

$$\dfrac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \cdots \quad \mathcal{H}_k}{\Gamma \vdash c : A} \qquad \dfrac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \cdots \quad \mathcal{H}_k}{\Gamma \vdash c \doteq a : A}$$

Example: The Identity Function

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A}\,\delta}{\Gamma \vdash (\lambda x.x) : A \to A}\,\lambda}{\Gamma \vdash \mathrm{id}_A :\equiv (\lambda x.x) : A \to A}$$

Example: Composition

$$\text{comp} := (\lambda g.\lambda f.\lambda x.g(f(x))) \ : \ (B \to C) \to (A \to B) \to (A \to C)$$

(See book for formal derivation)

$$g \circ f := ((\text{comp } g) \ f) \ : \ A \to C$$

Example: The Left Unit Law

Check Your Understanding Derive:

$$\frac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash \mathrm{id}_B(f(x)) \doteq f(x) : B}\,(a)$$

Then...

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash f : A \to B}{\Gamma, x : A \vdash \mathrm{id}_B(f(x)) \doteq f(x)}\,(a)}{\Gamma \vdash \lambda x.\mathrm{id}_B(f(x)) \doteq \lambda x.f(x)} \qquad \dfrac{\Gamma \vdash f : A \to B}{\Gamma \vdash \lambda x.f(x) \doteq f}\,\eta}{\Gamma \vdash \mathrm{id}_B \circ f \doteq f}$$

3 How we'll use MLTT

Blending with interpretations

Moving forward, we'll be more casual about interpretations, switching between them as suits our purposes

Informal Type Theory

The formal framework of contexts, type judgments, etc. can often be too clunky and get in the way. So we'll work in an **informal** style, e.g.

- The context is usually implicit
- "Let x be of type T" (and similar) means "$x : T$ is in our context"
- "Assume T" means "Assume T is inhabited"
- "Let X be a gadget" means "Let X be a term (of the appropriate type) such that is_gadget(X) is inhabited"
- We'll have informal ways of reading (and using) the formal inference rules we use to define our types

Formalization

A key benefit of HoTT is its amenability to **formalization**: even though we usually work informally, our informal methods closely mirror our formal rules so it's easily to "translate" into formal derivations.

Interactice proof assistants (like Agda or Coq) allow us to write our formal proofs in a computer-readable format, so the computer can check our proofs and verify their correctness!

Next Time...

- More discussion of type families
- Dependent Types & their interpretations
- "Official" rules for **2**, $\times$, etc.
- More types

Thanks for watching!

Designed, written, and performed by
Jacob Neumann

Based on the textbook
Introduction to Homotopy Type Theory
by
Egbert Rijke

Next video

Music:
"*Wholesome*" and "*Fluidscape*"
Kevin MacLeod (incompetech.com)

Full lecture

Next video

Full lecture