Homotopy Type Theory

- (Formal) foundation of mathematics (↔ ZF set theory, ↔ LEAN mathlib foundation)
- sheds some new light on mathematical concepts & way of thinking

I. Type Theory

What are types? $\boxed{\text{Sets like mathematicians want them to be}}$

I.1. Primitive notions of type theory

types X

objects $x : X$ of type X

judgmental / definitional equalities of objects of the same type

$x \equiv y : X$

ZF set theory: sets, $\in$, first-order predicate logic with $=$

- all objects MUST have a type
- types are objects in a universe $\mathcal{U}$ ⇝ hierarchy of universes

I.2. Rules to construct types, objects of a certain type and judgmental equalities

= deductive system with decidable & terminating checks for object-type relations & judgmental equalities

(similar to syntax checks in programming languages)

⇝ $\boxed{\text{safety of proof checkers}}$

- A, B types ⇝ function type $A \to B$

  $f : A \to B$ function ⇝ $f \equiv \lambda(x:A),\ f\,x$ ← fun $(x:A) \mapsto f\,x$ $\boxed{\lambda\text{-calculus}}$

  $a : A$ ⇝ $(\lambda(x:A),\ f\,x)\ a \equiv f\,a$

Example: identity function $id_A :\equiv \lambda(x:A),\ x$

- Dependent function type: $A : \mathcal{U}$, type family $B : A \to \mathcal{U}$ ⇝ $\Pi(a:A),\ B\,a : \mathcal{U}$

  dependent pairs = Σ-type: — " — ⇝ $\Sigma(a:A),\ B\,a : \mathcal{U}$

  special case: function & pairs: $A : \mathcal{U}$, $C : \mathcal{U}$ ⇝ $f : \Pi(a:A), C \equiv A \to C$ & $(a,c) : \Sigma(a:A),\ C \equiv A \times C$ — Currying of functions

  Example 1: magma = type with binary operation

  ↳ family of types $\lambda(X:\mathcal{U}),\ X \to (X \to X) : \mathcal{U} \to \mathcal{U}$

  type of a magma: $\Sigma(X:\mathcal{U}),\ X \to X \to X$

Example 2: surjective functions

$\forall \mathrel{\hat{=}} \Pi$, $\exists \mathrel{\hat{=}} \Sigma$, $=$ not yet defined

$f : A \to B$ ⇝ $\Pi(b:B),\ \Sigma(a:A),\ f\,a = b \;\eqqcolon\; \text{is\_surjective } f$ (property of being surjective)

(surjective functions have type $\Sigma(f : A \to B),\ \text{is\_surjective } f$)

I.3. Inductive Types

Example 1: Construct type $\mathbb{N}$ of natural numbers inductively

- 0 is a natural number ⇝ $0 : \mathbb{N}$
- successor $S(n)$ of a natural number n is a natural number ⇝ $S : \mathbb{N} \to \mathbb{N}$

} constructors

↳ recursion that is terminating

Construction of function $add : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ by induction:

↳ recursion that is terminating

$\cancel{(C \Rightarrow D) \Rightarrow C}$

not allowed as constructor

Construction of function $\mathrm{add} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ by induction:

- $n + 0 \equiv n$
- $n + S(m) \equiv S(n+m)$

(0) $S(0) + S(0) \equiv S(S(0)+0) \equiv \underset{\text{normal form}}{S(S(0))} : \mathbb{N}$

equality is inductively defined

(1) $\forall n, m : \mathbb{N},\ S(n) + m = S(n+m)$:

Construction of an object, using inductive eliminator:

Assume $n, m : \mathbb{N}$. Do induction on m:

tactics

$\lambda\, n\, m : \mathbb{N}$

- $S(n) + 0 \equiv S(n) \equiv S(n+0)$
- $S(n) + S(m') \equiv S(S(n) + m') \underset{IH}{=} S(S(n+m')) \equiv S(n + S(m'))$

(2) $\forall n : \mathbb{N} : 0 + n = n$

Assume $n : \mathbb{N}$. Do induction on n:

- $0 + 0 \equiv 0$
- $0 + S(n') \equiv S(0 + n') \underset{IH}{=} S(n')$

(3) $\forall n, m : \mathbb{N},\ n + m = m$

Assume $n, m : \mathbb{N}$. Do inducti[on on] m:

- $n + 0 \equiv n \overset{(2)}{=} 0 + n$
- $n + S(m') \equiv S(n+m') \overset{IH}{=} S(m'+n) \overset{(1)}{=} S(m') + n$.

Curry-Howard isomorphism: Proofs are objects, type is the statement to prove

program

Example 2: Σ-types $\Sigma(a:A), Ba$ as inductive type

constructor: $\mathrm{dpair} : \Pi(a:A), Ba \to \Sigma a : A, Ba$ $\mathrm{dpair}\ a\ b \equiv: \langle a, b \rangle$

eliminator: $\mathrm{ind}_{\Sigma(a:A), Ba} : \Pi(C : \Sigma(a:A), Ba \to U), (\Pi(a:A)(b:Ba), C\langle a,b\rangle) \to \Pi(p : \Sigma(a:A), Ba), Cp$

$$\mathrm{ind}_{\Sigma(a:A), Ba}(C, h, \langle a, b\rangle) \equiv \underbrace{h(a,b)}_{C\langle a,b\rangle}$$

$\mathrm{pr}_1 : \Sigma(a:A), Ba \to A$

$\equiv$

$\mathrm{ind}_{\Sigma(a:A), Ba}(\lambda(p : \Sigma(a:A), Ba), A), \lambda(a:A)(b:Ba), a)$

$\mathrm{pr}_2 : \Pi(p : \Sigma(a:A), Ba), B(\mathrm{pr}_1(p)) :\equiv$ Assume $p : \Sigma(a:A), Ba$. Do induction on p:

$\mathrm{pr}_2(\langle a, b\rangle) \equiv b : Ba$

$\leadsto$ $\boxed{\text{Calculus of Inductive Constructions}}$

rich enough deductive systems for all of mathematics

I.4. Logic

True : inductive type with constructor $tt : \text{True}$

False : inductive type without constructor

Induction on $ff : \text{False}$ $\rightsquigarrow$ yields everything

$\boxed{\text{Ex Falso quodlibet}}$

Not $\equiv \neg$: $\Pi(A : \mathcal{U}),\ A \to \text{False}$

And $\equiv \times$: Inductive type with constructor $A \to B \to A \times B$

Or $\equiv +$: Inductive type with two constructors $inl : A \to A + B$

$inr : B \to A + B$

For all : $\underset{\overset{\updownarrow}{\forall}}{}\Pi(a : A),\ P\,a$ where $P : A \to \mathcal{U}$

There exists: $\Sigma(a : A),\ P\,a$ ($\Sigma \leftrightarrow \exists$)

(1) $A \times B \to B \times A$: Assume $ass : A \times B$. Do induction on ass

$(a, b) \mapsto (b, a) : B \times A$

(2) $A \to \neg\neg A \equiv (A \to \text{False}) \to \text{False}$: Assume $a : A$. Assume $na : A \to \text{False}$

Do induction on $na\ a$.

(3) $A + \neg A$: Cannot be proven = no object of this type can be constructed

Excluded Middle can be stated as an axiom

$\rightsquigarrow$ Logic is constructive

(4) $A \to \Pi(a : A), P\,a \to \Sigma(a : A), P\,a$: Assume $a : A$, $f : \Pi(a : A), P\,a$

Take $(a, f\,a) : \Sigma(a : A), P\,a$

II. Equality & Univalence

II.1. Equality

family of inductive types $Id_A : A \to A \to \mathcal{U}$ with indices $a, b : A$ $\rightsquigarrow$ $Id_A(a,b) \equiv a =_A b$

one constructor : $idpath_A\, a : a =_A a$

$\boxed{\text{Martin Löf equality}}$ = propositional equality

- judgmental equality $\underset{c}{a} \equiv \underset{c}{b} : A$ $\rightsquigarrow$ object $idpath\ c$ (or a, or b)

  proofs of statements in $\mathbb{N}$: $1 + 1 = 2$

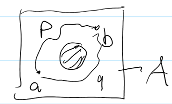
- induction only works for the whole type family : To obtain $c : C$ from $p : a = b$

  a and b must be arbitrary independent objects in A

- induction only works for the whole type family: To obtain $c : C$ from $p : a = b$
  a and b must be arbitrary independent objects in A

- transport over equality through family of types:
  $B : A \to \mathcal{U}$, $a_1, a_2 : A$, $p : a_1 = a_2$, $b_1 : B a_1 \longmapsto \underset{p_*(b_1)}{\overset{}{b_2}} : B a_2$: Do induction on p
  $b_2 \stackrel{!!}{=} p_*(b_1)$ $\quad$ $p : a_1 = a_1$. Take $b_1 : B a_1$

- homotopy interpretation:
  equality $p : a =_A b$ $\longleftrightarrow$ path from a to b in A

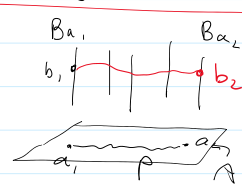


(path) induction on p $\longleftrightarrow$ base point free paths are homotopic to constant paths (in connected spaces)

equalities of equalities $\longleftrightarrow$ homotopy between paths with fixed endpoints
$p, q : a =_A b \rightsquigarrow p =_{a =_A b} q$
$p \neq q$ $\longleftrightarrow$ paths are not homotopic

> Existence of non-equal equalities is the big difference to "standard" dependent type theory, as used e.g. for LEAN's mathlib

transport over $p : a_1 =_A a_2$ in $B : A \to \mathcal{U}$ $\longleftrightarrow$



- inductive construction of equalities in inductive types

<u>Example:</u> $P : A \to \mathcal{U}$ type family , $w, w' : \Sigma(a : A), Pa$

(1) $f : \prod_{w,w'} (w = w') \longrightarrow \Sigma(p : pr_1(w) = pr_1(w')), \; p_*(pr_2(w)) = pr_2(w')$:
Do induction on $p \rightsquigarrow p \equiv idpath\ w$
Take $(idpath\ pr_1(w),\ idpath\ pr_2(w))$
↳ Note $(idpath_A\ pr_1(w))_*\ pr_2(w) \equiv pr_2(w)$

(2) $g : \prod_{w,w'} \Sigma(p : pr_1(w) = pr_1(w')), \; p_*(pr_2(w)) = pr_2(w') \longrightarrow w = w'$:

Do induction on $w, w' \rightsquigarrow w \equiv (a, b)$, $w' \equiv (a', b')$ (& calculate)
Do induction on $p \rightsquigarrow p \equiv idpath_A\ a$, $q \equiv (idpath\ a)_*\ b = b' \equiv b = b'$
Do induction on $q \rightsquigarrow q \equiv idpath_{Pa}\ b$
Take $idpath_{\Sigma(a:A), Pa}\ (a, b)$

(3) $f(g(v)) = v$: Do induction on w, w', v (& calculate using the inductive definition of f and g)
Do path induction on the two components of v
Take $idpath$

(4) $g(f(p)) = p$: Do induction on p $\rightsquigarrow$ $p : w = w$

Do induction on w (& calculate)

Take idpath

What about equalities of functions $f, g : A \to B$? Equalities of types A, B ?