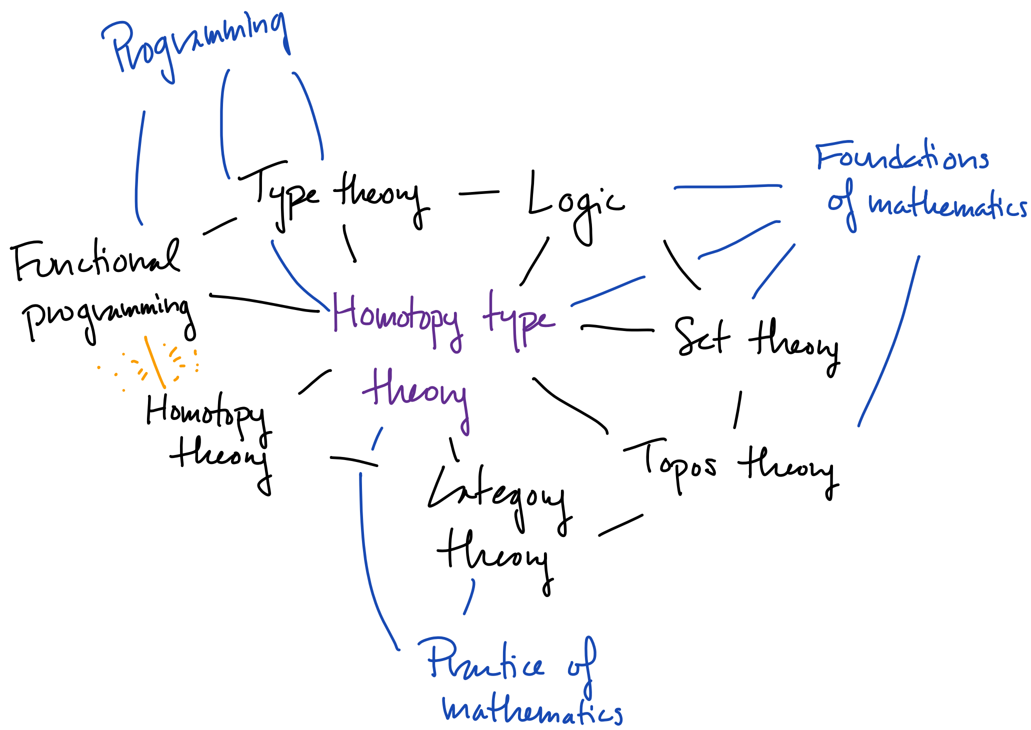
Map of subjects surrounding homotopy type theory:



Martin-Löf type theory

(Similar to CIC, but no universe Prop of propositions and with Σ-types)

- 3 basic judgments
  - Γ ctx
  - $\Gamma \vdash T$ TYPE
  - $\Gamma \vdash t : T$
- Corresponding equality judgments
  - $\Gamma \doteq \Gamma'$ ctx
  - $\Gamma \vdash T \doteq T'$ TYPE
  - $\Gamma \vdash t \doteq t' : T$
- Type formers
  - $\emptyset, \mathbb{1}, \mathbb{B}, \mathbb{N}$, W-types
  - $=$ types
  - Σ types, Π types

- universe hierarchy U_i

Univalent foundations / type theory

MLTT + univalence axiom (focus on h-levels)

Homotopy type theory

UF + higher inductive types

Inductive types

Inductive types are freely generated by their canonical terms.

Σ-types

Given $b:B \vdash E(b)$ type (in log $E\ (b:B) : \mathcal{U}$), want to form a type whose terms are dependent pairs $\langle b, e\rangle$ where $b:B$, $e:E(b)$

Dependent pair types Σ

Σ-form: $$\frac{\Gamma, x:P \vdash Q(x)}{\Gamma \vdash \sum_{x:P} Q(x)}$$

Σ-intro: $$\frac{\Gamma \vdash p:P \qquad \Gamma \vdash q : Q(p)}{\Gamma \vdash \mathrm{pair}(p,q) : \sum_{x:P} Q(p)}$$

Exercise. Construct a function $\pi_1 : \sum_{x:P} Q(x) \to P$

Types as logic, sets, programs (Curry-Howard, Brouwer-Heyting-Kolmogorov)

	Logic	Sets	Program
Γ ctx	hypotheses	indexing set	names in scope
$\Gamma \vdash T$ type	predicate T on Γ	family T of sets on Γ $\begin{matrix} T \\ \downarrow \\ \Gamma \end{matrix}$	program specification using values from Γ
$\Gamma \vdash t : T$	proof of T	section, i.e., $T(\gamma)$ for all $\gamma \in \Gamma$	program
$\mathbb{N}$	—	$\mathbb{N}$	program w/ no input that outputs a nn
$S+T$ $(\sum_{i:\text{bool}} T_i)$	$\vee$	$\cup$	$\vee$
$S \times T$ $(\sum_{s:S} T)$	$\wedge$	$\times$	$\wedge$
$S \to T$ $(\prod_{s:S} T)$	$\Rightarrow$	$\to$	turn one kind of program into another
$\sum_{b:B} E(b)$	$\exists$	$\bigsqcup$	Σ
$\prod_{b:B} E(b)$	$\forall$	$\prod$ (set of sections)	Π

The strongest inductive type : Id

Why do we need the identity type?

(If we're not interested in homotopy.)

A1: There are many equalities that hold only propositionally.

Ex. $\text{add}(x,0) \doteq x$
$\text{add}(x, sy) \doteq s\,\text{add}(x,y)$
One cannot prove $\text{add}(0,x) \doteq x$.

To prove this, we need to induct on n (i.e. use $\mathbb{N}$-elimination), but this only allows us to construct a term of a type.

We <u>will</u> be able to prove $\mathrm{add}(0,x) = x$.

<u>A2</u>: We already have a notion of equality:

<u>judgmental equality</u> $\doteq$

(The identity type is called <u>propositional equality</u> $=$.)

<u>Logical interpretation</u>: propositions are types / proofs are terms.

To <u>prove</u> an equality (and be consistent with the logical interpretation) we want to produce a <u>term</u> of a <u>type</u> of equalities.

<u>Type constructors often internalize structure</u>

- bool
- $\mathbb{N}$
- $\emptyset$
- $\mathbb{1}$

} can also be seen as internalizing external versions.

- The universe type internalizes the judgment of the form

  A type

- We'll see how the identity type internalizes judgmental equality...

Identity type $=$

$=$-form
$$\frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : A}{\Gamma \vdash a =_A b \text{ type}}$$

$=$-intro
$$\frac{\Gamma \vdash a : A}{\Gamma \vdash r_a : a =_A a}$$

Type constructors often internalize structure

Note that the rules governing equality say that

if $a \doteq b : A$, then $r_a : a =_A b$.

→ Reflexivity (r_-) turns 'judgmental equalities into propositional equalities.

Exercise • $\mathrm{add}(0, n) = n$ for all $n : \mathbb{N}$.

(Note: one of $\mathrm{add}(0,n) \doteq n$ and $\mathrm{add}(n,0) \doteq n$ will hold definitionally depending on how you defined add. Show the other one holds 'judgmentally'.)

(Ex.) take the inverse of an equality (if $q : b =_A c$, then $q^{-1} : c =_A b$)

(Ex.) take composition of equalities (if $p : a =_A b$ and $q : b =_A c$, then $p \cdot q : a =_A c$)

(Ex) functions $A \to B$ respect equality (i.e. map $a =_A a'$ to $fa =_B fa'$)

(Ex) For any dependent type $x : B \vdash E(x)$ type, any terms $b, b' : B$, and any equality $p : b =_B b'$, there is a function $tr_p : E(b) \to E(b')$.