Hoffman & Streicher

groupoid model of type theory.

What's your Type?

Was sind und was sollen Hott ?

Basic notion in Type theory

— Each object is assigned a type.

1st order vs 2nd order logic

Set is completely characterized by it's elements, but type is not by it's terms

HoTT tries to provide new foundation of maths (ohne ZFC)

$a:A$ "a is of type A"

types can themselves inhabit types called "Universes"

& cumulatively ∃ chain of types $\mathcal{U}_0 : \mathcal{U}_1 : \mathcal{U}_2 : \cdots$

$A:\mathcal{U}_i \Rightarrow A:\mathcal{U}_{i+1}$

Notion of equality

Equality itself is a type, if $a,b:A$

then we have type $a =_A b$

if $a =_A b$ is inhabited, then $a = b$ → apostériori → propositional equality

we can also have $a \equiv b : A$ → definitional / judgemental equality

↳ apriori

in fact $a \equiv b : A$ implies $a =_A b$ is inhabited.

Rules of Judgemental equality

- Reflexivity : Given type A, and $a:A$, then $a \equiv a : A$
- Symmetry : Given $a \equiv b : A$, then $b \equiv a : A$
- Transitivity : Given $a \equiv b : A$, $b \equiv c : A$, then $a \equiv c : A$

- Function type $A \to B$

let $f : A \to B$ (f is not a map from $A \to B$, f is an inhabitant of type $A \to B$)

↓

$a:A$, $f(a):B$

λ-calculus

we can define $f(x) \equiv \phi : B$ where x is an arbitrary term of type A & $\phi : B$ for $x:A$. In order to make fn out of ϕ, we use λ-abstraction

$$(\lambda(x:A).\phi) : A \to B$$

Eg, $A, B = \mathbb{R}$ $f : A \to B$, $f(x) = x^2+2x+1$

but x^2+2x+2 is not a function

but $\lambda(x:\mathbb{R}).(x^2+2x+2) : \mathbb{R} \to \mathbb{R}$

① Introduction

— Sets vs Types

- Uniquely characterized by its elements.
- Sets aren't spatial
- Russels paradox
- Sets, Axioms and predicate logic ⇒ Mathematics
- Types vs Sets
  - Everything in set theory is a set
    given, A, B, we can ask $A \in B$?
  - In Type theory Types & elements have separate existance
    every element is assigned a designated type.
  - In Set theory, a set A is uniquely determined by $y \in A$ for elements / sets.
  - In type theory, new types are formed using formation rules
- $t : T$, t is a term of type T
- Martin-Löf Dependent Type theory
  ↓
  Homotopy interpretation
- Univalence axiom

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

violates ZFC.
we can have many singleton sets
but only one singleton type.

1.1 The Holy Trinity of HoTT

3 ways to intepret HoTT

Eins Programming Interpretation

Zwei Homotopic Interpretation

Types as spaces & terms as point

Drei Logical Interpretation

Types are proposition & terms are proofs.

Jacob Neumann Was Soll HoTTP

HoTT
- Programming Interpretation
- Homotopic Interpretation
- Logic Interpretation

The Unit type $\mathbb{1}$: Unit is the contractible type

- $\mathbb{N}$, natural numbers
  $0 : \mathbb{N}$

Typed vs Untyped programming language.

unit type $() : \text{Unit}$, single term

Homotopic

Problems with sets, sets aren't spatial!!!

HoTT are natively spatial

Types are spaces

terms are point

Unit → 1 unit space ... contractible space

Logic

- Constructive vs Nonconstructive
- HoTT built around constructive.

"Curry - Howard Correspondence"

- Types are propositions

inhabited ⇒ true.

$t : T$ → t is a proof of T

$2^n = x^2$

- Proof Relevance

- Unit $\mathbb{1}$ true proposition

Moreover $\exists!$ proof

② Martin-Löf Dependent Type theory

- 4 types of Judgements
  - $\Gamma \vdash A$ type
  - $\Gamma \vdash a : A$
  - $\Gamma \vdash A \doteq B$ type
  - $\Gamma \vdash a \doteq b : A$

Γ, the context
finite list of variable declaration
$x_1 : A_1,\ x_2 : A_2(x_1),$
$x_n : A_n(x_1 \dots x_{n-1})$ satisfying
given $1 \le k \le n$
$x_1 : A_1, \dots x_{k-1} : A_{k-1}(x_1, \dots x_{k-2})$
$\vdash A_k(x_1, \dots x_{k-1})$ type

- Dependent type theory

- $\Gamma, x : A \vdash B(x)$ type
  a family of type over A / type indexed by $x : A$ in Γ
- section of family B
  $\Gamma, x : A \vdash b(x) : B(x)$ b is the section

• Inference rules

- structural rules
  - ↳ 6 sets
    1. Formation of contexts, types etc
    2. Judgemental Equality is Eqv.
    3. Variable conversion
    4. Substitution S
    5. Weakening W
    6. Generic element 8

① $\dfrac{\Gamma \vdash a : A}{\Gamma \vdash A \text{ type}}$ $\qquad$ $\dfrac{\Gamma, x : A \vdash B(x) \text{ type}}{\Gamma \vdash A \text{ type}}$

② $\dfrac{\Gamma \vdash A \doteq B \text{ type}}{\Gamma \vdash B \doteq A \text{ type}}$

③ $\dfrac{\Gamma \vdash A \doteq A' \text{ type} \qquad \Gamma, x : A, \Delta \vdash J}{\Gamma, x : A', \Delta \vdash J}$

④ $\dfrac{\Gamma \vdash a : A \qquad \Gamma, x : A, \Delta \vdash J}{\Gamma, \Delta[a/x] \vdash J[a/x]}$

⑤ $\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma, \Delta \vdash J}{\Gamma, x : A, \Delta \vdash J}$ eg $\dfrac{\Gamma \vdash A \text{ type} \qquad \Gamma \vdash B \text{ type}}{\Gamma, x : A \vdash B \text{ type}}$

⑥ $\dfrac{\Gamma \vdash A \text{ type}}{\Gamma, x : A \vdash x : A}$

Change of Variable derivation

$$\frac{\Gamma \vdash x:A,\ \Delta \vdash \mathcal{J}}{\Gamma, x':A,\ \Delta[x'/x] \vdash \mathcal{J}[x'/x]} \longrightarrow \text{Change of variable.}$$

⇓ Derivation

$$\frac{\dfrac{\Gamma \vdash A\ \text{type}}{\Gamma, x':A \vdash x':A}\ s \qquad \dfrac{\Gamma \vdash A\ \text{type} \quad \Gamma, x:A, \Delta \vdash \mathcal{J}}{\Gamma, x':A, x:A, \Delta \vdash \mathcal{J}}}{\Gamma, x':A, \Delta[x'/x] \vdash \mathcal{J}[x'/x]}$$

③ Type Constructors

- Formation, Introduction, Elimination, Computation
  - Formation: Construct a type
  - Introduction: Construct a term
  - Elimination: Use the term
  - Computation: Relate intro & Elim

• Product type $A \times B$ (And)

- $\dfrac{\Gamma \vdash A\ \text{type} \quad \Gamma \vdash B\ \text{type}}{\Gamma \vdash A \times B\ \text{type}}$ ×-Form
- $\dfrac{\Gamma \vdash a:A \quad \Gamma \vdash b:B}{\Gamma \vdash (a,b):A \times B}$ ×intro
- $\dfrac{\Gamma \vdash p:A \times B}{\Gamma \vdash \mathrm{pr}_1(p):A \quad \Gamma \vdash \mathrm{pr}_2(p):B}$ ×elim
- $\dfrac{\Gamma \vdash a:A \quad \Gamma \vdash b:B}{\Gamma \vdash \mathrm{pr}_1(a,b) = a:A}$ Cmp b
- $\dfrac{\Gamma \vdash p:A \times B}{\Gamma \vdash (\mathrm{pr}_1(p), \mathrm{pr}_2(p)) = p : A \times B}$

• Function type $A \to B$ (Implication)

- $\dfrac{\Gamma \vdash A\ \text{type} \quad \Gamma \vdash B\ \text{type}}{\Gamma \vdash A \to B\ \text{type}}$
- $\dfrac{\Gamma \vdash f:A \to B \quad \Gamma \vdash a:A}{\Gamma a:A \vdash f(a):B}$
- $\dfrac{\Gamma, x:A \vdash b(x):B}{\Gamma \vdash \lambda x.b(x):A \to B}$
- $\dfrac{\Gamma \vdash f:A \to B}{\Gamma, x:A \vdash \lambda x.f(x) = f : A \to B}$
- $\dfrac{\Gamma, x:A \vdash b(x):B}{\Gamma, x:A \vdash (\lambda y.b(y))(x) = b(x) : B(x)}$

Forming new types from known ones

- Families of type indexed by another type.
ie if A is a type, we can have $B(x)$ for each $x:A$
thought of as $B : A \to \mathcal{U}$

New types are built using following rules

① Formation rule : Tells how to form a new type.
② Introduction rule : Tells how to construct terms of that type.
③ Elimination rule : Tells how to use/apply terms of that type.
④ Computational rule : Tells how eliminator acts on constructor.

optional
⑤ Uniqueness principle : Express uniqueness of maps into/out of type.

Dependent Function type Π

Π-formation : Given $A:\mathcal{U}$ & $B:A\to\mathcal{U}$ we form dependent fn type

$$\prod_{x:A} B(x)$$

Π-Introduction : if $x:A$ then $b:B(x)$, then

$$\lambda x.b : \prod_{x:A} B(x)$$

Π-Elimination : Given $f : \prod_{x:A} B(x)$ and $a:A$, then $f(a):B(a)$

Π-Computation : if $x:A$ then $b:B(x)$, and $a:A$

$\lambda x.b(a) : B(a)$ in fact $\lambda x.b(a) \equiv b[a/x] : B(a)$

$b[a/x]$ denotes expression obtained by substituting a for x in b

Π-Uniqueness : Given $f : \prod_{x:A} B(x)$,

$$f \equiv \lambda x.f(x) : \prod_{x:A} B(x)$$

More generally, we have

Dependent function type $\prod_{(x:A)} B(x)$ &

Dependent pair type $\sum_{(x:A)} B(x)$

For any types A, B $\quad (A \times A \to B) \to B$ Modus ponens:

(PF) By λ-intro, enough to

$$\frac{\Gamma \vdash d : (A \times (A \to B))}{\dfrac{\Gamma \vdash pr_1(d) : A \quad \Gamma \vdash pr_2(d) : A \to B}{\dfrac{\Gamma \vdash pr_1(d) : A \quad \Gamma, x:A \vdash pr_2(d)(x) : B}{\Gamma, p:A \vdash pr_2(d)\,p : B}}}$$

④ Natural Number type $\mathbb{N}$

$$\frac{}{\vdash \mathbb{N}\ \text{type}} \quad ; \quad \frac{}{\vdash 0_{\mathbb{N}} : \mathbb{N}} \quad ; \quad \frac{}{\vdash succ_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}$$

- elimination: Induction principle

Predicate P over $\mathbb{N}$ is $\rightsquigarrow$ Type Family over $\mathbb{N}$

$\Gamma, n:\mathbb{N} \vdash P(n)$ type

Then type theoretic induction principle is

$$\frac{\Gamma, n:\mathbb{N} \vdash P(n)\ \text{type} \quad \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \quad \Gamma \vdash p_s : \prod_{(n:\mathbb{N})} P(n) \to P(succ_{\mathbb{N}}(n))}{\Gamma, ind_{\mathbb{N}}(p_0, p_s) : \prod_{n:\mathbb{N}} P(n)}$$

or

$$\frac{\Gamma, n:\mathbb{N} \vdash P(n)\ \text{type}}{\Gamma \vdash ind_{\mathbb{N}} : P(0) \to \left[\left(\prod_{n:\mathbb{N}} (P(n) \to P(succ_{\mathbb{N}} n))\right) \to \prod_{(n:\mathbb{N})} P(n)\right]}$$

• Computation rule

① $$\frac{\Gamma, n:\mathbb{N} \vdash P(n)\ \text{type} \qquad \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \qquad \Gamma \vdash p_s : \prod_{n:\mathbb{N}} P(n) \to P(\text{succ}_{\mathbb{N}}(n))}{\Gamma \vdash \text{ind}_{\mathbb{N}}(p_0, p_s, 0_{\mathbb{N}}) = p_0 : P(0_{\mathbb{N}})}$$

② $$\frac{\Gamma, n:\mathbb{N} \vdash P(n)\ \text{type} \qquad \Gamma \vdash p_0 : P(0_{\mathbb{N}}) \qquad \Gamma \vdash p_s : \prod_{n:\mathbb{N}} P(n) \to P(\text{succ}_{\mathbb{N}}(n))}{\Gamma, n:\mathbb{N} \vdash \text{ind}_{\mathbb{N}}(p_0, p_s, \text{succ}_{\mathbb{N}}(n)) \doteq p_s(n, \text{ind}_{\mathbb{N}}(p_0, p_s, n)) : P(\text{succ}_{\mathbb{N}}(n))}$$

⑤ The Identity Type

$$\frac{\Gamma \vdash a : A}{\Gamma, x:A \vdash a =_A x\ \text{type}} \quad ; \quad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{refl}_a : a =_A a}$$

(E1) $$\frac{\Gamma \vdash a : A \qquad \Gamma, x:A, p : a =_A x \vdash P(x,p)\ \text{type}}{\Gamma \vdash \text{ind-eq}_a : P(a, \text{refl}_a) \to \prod_{(x:A)} \prod_{(p:a=_A x)} P(x,p)}$$

(Com) $$\frac{\Gamma \vdash a : A \qquad \Gamma, x:A, p : a =_A x \vdash P(x,p)\ \text{type}}{\Gamma, u : P(a, \text{refl}_a) \vdash \text{ind-eq}_a(u, a, \text{refl}_a) \doteq u : P(a, \text{refl}_a)}$$

Principle of substitution

To prove that every x, y with $x = y$ have Property P, it suffices to prove $x = x$ has property P

Inverse

- Symmetry, For all x, y $x = y$ then $y = x$. (here $P(x,y) := y = x$)
  enough to prove $x = x$ (true by reflexivity)
- Transitivity, for all x, y, z if $x = y$ and $y = z$ then $x = z$

$Q(x,y) := \forall z\ (y = z \to x = z)$

Composition

⑥ Homotopy Type theory

Iterated identity type

$x, y : A \qquad x =_A y$ type

if $p, q : x =_A y$, what about $p =_{x=_A y} q$.

The Curry-Howard-Voevodsky correspondence

type theory	set theory	logic	homotopy theory
A	set	proposition	space
$x : A$	element	proof	point
$\emptyset, 1$	$\emptyset, \{\emptyset\}$	$\bot, \top$	$\emptyset, *$
$A \times B$	set of pairs	A and B	product space
$A + B$	disjoint union	A or B	coproduct
$A \to B$	set of functions	A implies B	function space
$x : A \vdash B(x)$	family of sets	predicate	fibration
$x : A \vdash b : B(x)$	fam. of elements	conditional proof	section
$\prod_{x:A} B(x)$	product	$\forall x.B(x)$	space of sections
$\sum_{x:A} B(x)$	disjoint sum	$\exists x.B(x)$	total space
$p : x =_A y$	$x = y$	proof of equality	path from x to y
$\sum_{x,y:A} x =_A y$	diagonal	equality relation	path space for A

Theorem (Lumsdaine, Gaerner - van den Berg)

The terms belonging to the iterated identity types of any type A form an ∞-groupoid.

- **Contractible type**

  A type A is contractible if it comes with a term of type

$$\sum_{a:A} \prod_{x:A} a =_A x$$

- Equivalence of type

  A, B are equivalent if the following type is inhabited

$$A \simeq B :\equiv \sum_{f:A\to B} \left(\sum_{g:B\to A} \prod_{a:A} g(f(a)) =_A a \right) \times \left(\sum_{h:B\to A} \prod_{b:B} f(h(b)) =_B b \right)$$

Univalence Axiom

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

Identity is equivalent to equivalence.

- For set based structures (gps, monoids, rngs) isomorphic structures are identical