Naive Type Theory

Thorsten Altenkirch
Functional Programming Laboratory
School of Computer Science

You guys are both my witnesses... He insinuated that ZFC set theory is superior to Type Theory!

Set theory

$\mathbb{N}$ set of natural numbers

$3 \in \mathbb{N}$

$3 + 4 = 7$

$\forall x.x \in \mathbb{N} \rightarrow x + x = 2 \times x$

Type theory

$\mathbb{N}$ type of natural numbers

$3 : \mathbb{N}$

$3 + 4 \equiv 7$

$\Pi x : \mathbb{N}.x + x = 2 \times x$

What's the difference ?

- In Set Theory, elements are first
- They are collected into sets
- In Type Theory Types come first
- Elements are associated with their types

$a \in A$ is a proposition

$a : A$ is a judgement

$\forall x.x \in A \to x \in B$

not expressible in Type Theory

$A \subseteq B$

$A \to B$ ✓

$A \cap B$

$A \times B$ ✓

$A \cup B$

$A + B$ ✓

$A \simeq B$ isomorphism / bijection

$\{a, b\} \simeq \{c, d\}$

$\{a, b\} \cup \{a\} \not\simeq \{c, d\} \cup \{a\}$

$\{a, b\} + \{a\} \simeq \{c, d\} + \{a\}$

Intensional vs extensional

- Operations like $\cup$ and $\cap$ are intensional.
- Their behaviour depends on the intensional properties of objects.
- In contrast operations like $\times$, $+$ and $\rightarrow$ are extensional.
- They behaviour is independent on the choice of interpretation

- In Intensional Type Theory all operations are extensional.
- In HoTT we go one step further:
  - Isomorphic types are equal (univalence principle)

What is a function?

$$f : A \to B$$

- Functions are a primitive concepts in Type Theory

- We use the language of λ- calculus

- Functions are effective
  (a function that doesn't function shouldn't be called a function)

Dependent functions

- Fix $a : A$ now from $n : \mathbb{N}$ construct

$$\underbrace{(a, a, \ldots, a)}_{n \text{ times}} : A^n$$

$$\mathrm{tup} : \Pi n : \mathbb{N}.A^n$$

$$\mathrm{tup}\, n \equiv \underbrace{(a, a, \ldots, a)}_{n \text{ times}}$$

- The normal function type is a special case of Π-types

$$A \to B \equiv \Pi - : A.B$$

Dependent pairs

- A list is a tuple of arbitrary size.

$$\mathrm{List}\ A \equiv \Sigma n : \mathbb{N}.A^n$$

- The normal product type is a special case of Σ - types

$$A \times B \equiv \Sigma\_ : A.B$$

Disjoint union

- We can define $+$ using Σ-types

$$A + B \equiv \Sigma b : \mathrm{Bool}. \begin{cases} A & \text{if } b \equiv \mathrm{true} \\ B & \text{if } b \equiv \mathrm{false} \end{cases}$$

- injections are definable

$$\mathrm{inl} : A \to A + B$$

$$\mathrm{inl}\, a \equiv (\mathrm{true}, a)$$

An alternative definition of products

- We can use the same idea to define products

$$A \times B \equiv \Pi b : \mathrm{Bool}. \begin{cases} A & \text{if } b \equiv \mathrm{true} \\ B & \text{if } b \equiv \mathrm{false} \end{cases}$$

- This reflects the fact that product can be defined introduction-based or elimination-based.

Concepts in Type Theory

Π-types $\Pi x : A.B(x)$	$A \to B$ $A \times B$
Σ-types $\Sigma x : A.B(x)$	$A \times B$ $A + B$
inductive types	$\mathbb{N}$ Bool

Propositions as types

- Set Theory uses classical predicate logic.
- A proposition is interpreted as saying what is true.
- In Type Theory we avoid the idea of truth and define what is evidence for a proposition.
- This is achieved by assigning the type of evidence to a proposition.
- We accept a proposition if there is evidence for it.

Proposition as types

implication	$A \Rightarrow B$	$A \to B$
conjunction	$A \wedge B$	$A \times B$
disjunction	$A \vee B$	$A + B$
universal quantification	$\forall x : A.B(x)$	$\Pi x : A.B(x)$
existential quantification	$\exists x : A.B(x)$	$\Sigma x : A.B(x)$

Examples

- $A \wedge (B \vee C) \Rightarrow A \wedge B \vee A \wedge C$
$(\exists x : A.B\, x \vee C\, x) \Rightarrow (\exists x : A.B\, x) \vee (\exists x : A.C\, x)$

are tautologies.

- classically
- type-theoretically

$$A \wedge (B \vee C) \Rightarrow A \wedge B \vee A \wedge C$$

classically

A	B	C	$A \wedge (B \vee C)$	$A \wedge B \vee A \wedge C$	$\cdots \Rightarrow \ldots$
false	false	false	false	false	true
false	false	true	false	false	true
false	true	false	false	false	true
false	true	true	false	false	true
true	false	false	false	false	true
true	false	true	true	true	true
true	true	false	true	true	true
true	true	true	true	true	true

$$A \wedge (B \vee C) \Rightarrow A \wedge B \vee A \wedge C$$

type-theoretically

$$f : A \times (B + C) \to A \times B + A \times C$$

$$f\,(a, (\text{true}, b)) \equiv (\text{true}, (a, b))$$

$$f\,(a, (\text{false}, c)) \equiv (\text{false}, (a, c))$$

$$(\exists x : A.B\,x \vee C\,x) \Rightarrow (\exists x : A.B\,x) \vee (\exists x : A.C\,x)$$

classically

- we need infinite truth tables (1st order structures)
- We can argue on the meta level reflecting the tautology.

$$(\exists x : A.B\,x \vee C\,x) \Rightarrow (\exists x : A.B\,x) \vee (\exists x : A.C\,x)$$

type-theoretically

$$g : (\Sigma x : A.B\,x + C\,x) \to (\Sigma x : A.B\,x) + (\Sigma x : A.C\,x)$$

$$g\,(a, (\mathrm{true}, b)) \equiv (\mathrm{true}, (a, b))$$
$$g\,(a, (\mathrm{false}, c)) \equiv (\mathrm{false}, (a, c))$$

Classical principles

- Define $\neg A \equiv A \to \emptyset$
- There is no evidence for:
  - excluded middle $A \vee \neg A$
  - indirect proof $\neg\neg A \to A$
- However there is evidence for

$$\neg\neg(A \vee \neg A)$$

The axiom of choice?

$$R \subseteq A \times B$$

$$(\forall x : A.\exists y : B.R\,(x, y))$$
$$\rightarrow (\exists f : A \rightarrow B.\forall x : A.R(x, f(x)))$$

AC in Type Theory

$$R : A \times B \to \mathbf{Type}$$

$$\mathrm{ac} : (\Pi x : A.\Sigma y : B.R\,(x, y))$$
$$\to (\Sigma f : A \to B.\forall x : A.R(x, f(x)))$$

$$\mathrm{ac}\; f \equiv (\lambda x.\pi_1\,(f\; x), \lambda x.\pi_2\,(f\; x))$$

$$\pi_1 : (\Sigma x : A.B\; x) \to A$$
$$\pi_1(a, b) \equiv a$$
$$\pi_2 : \Pi y : (\Sigma x : A.B\; x).B\,(\pi_1\; y)$$
$$\pi_2\,(a, b) \equiv b$$

What happened?

- AC is usually considered classical.
- But now we were able to prove it in Type Theory?
- Types unlike propositions can carry information.
- What really is a proposition in Type Theory?

Propositions in Type Theory

- We call a Type a proposition if it has at most one element.

  $\mathrm{Prop} \equiv \Sigma A : \mathbf{Type}.\Pi x, y : A.x = y$

- Most logical operators are closed under Prop, e.g. $A : \mathbf{Type}$, $B : A \to \mathbf{Prop}$

  $\Pi x : A.B\,x : \mathbf{Prop}$

- But $+$ and Σ are not closed under Prop.

Propositional truncation

- We introduce an operation

  $||A|| : \mathbf{Prop}$,given $A : \mathbf{Type}$

  $\eta : A \to ||A||$

- This is a sort of a black box.

Propositions as types (2)

implication	$A \Rightarrow B$	$A \to B$
conjunction	$A \wedge B$	$A \times B$
disjunction	$A \vee B$	$\|\|A + B\|\|$
universal quantification	$\forall x : A.B(x)$	$\Pi x : A.B(x)$
existential quantification	$\exists x : A.B(x)$	$\|\|\Sigma x : A.B\ x\|\|$

AC in Type Theory (2)

$$R : A \times B \to \mathbf{Prop}$$

$$(\Pi x : A.||\Sigma y : B.R\,(x, y)||)$$
$$\to ||\Sigma f : A \to B.\forall x : A.R(x, f(x))||$$

- This type is not inhabited.
- Indeed it implies excluded middle (Diaconescu's construction)

Equality types

- Given $a, b : A$ we introduce the type $a = b$ of reasons that a is equal to b.
- We always have $\mathrm{refl} : a = a$
- In Intensional Type Theory this was viewed as an inductive definition.

Is equality propositional?

- Is $a = b : \mathbf{Prop}$?
- This was settled to the negative by Hofmann and Streicher using the Groupoid model.
- In HoTT univalence identifies equality of sets with isomophism.
- Hence there are 2 elements of $\mathrm{Bool} = \mathrm{Bool}$

The Hierarchy of Types

h-level	truncation level		
0	-2	**Contr**	$\Sigma x : A.\Pi y : A.x = y$
1	-1	**Prop**	$\Pi x, y : A.\mathrm{isContr}\,(x = y)$
2	0	**Set**	$\Pi x, y : A.\mathrm{isProp}\,(x = y)$
3	1	**Gpd**	$\Pi x, y : A.\mathrm{isSet}\,(x = y)$
...	...	...	...

The structure of equality types

Contr	trivial
Prop	trivial
Set	Equivalence relation
Gpd	Groupoid
...	...

Equality in HoTT

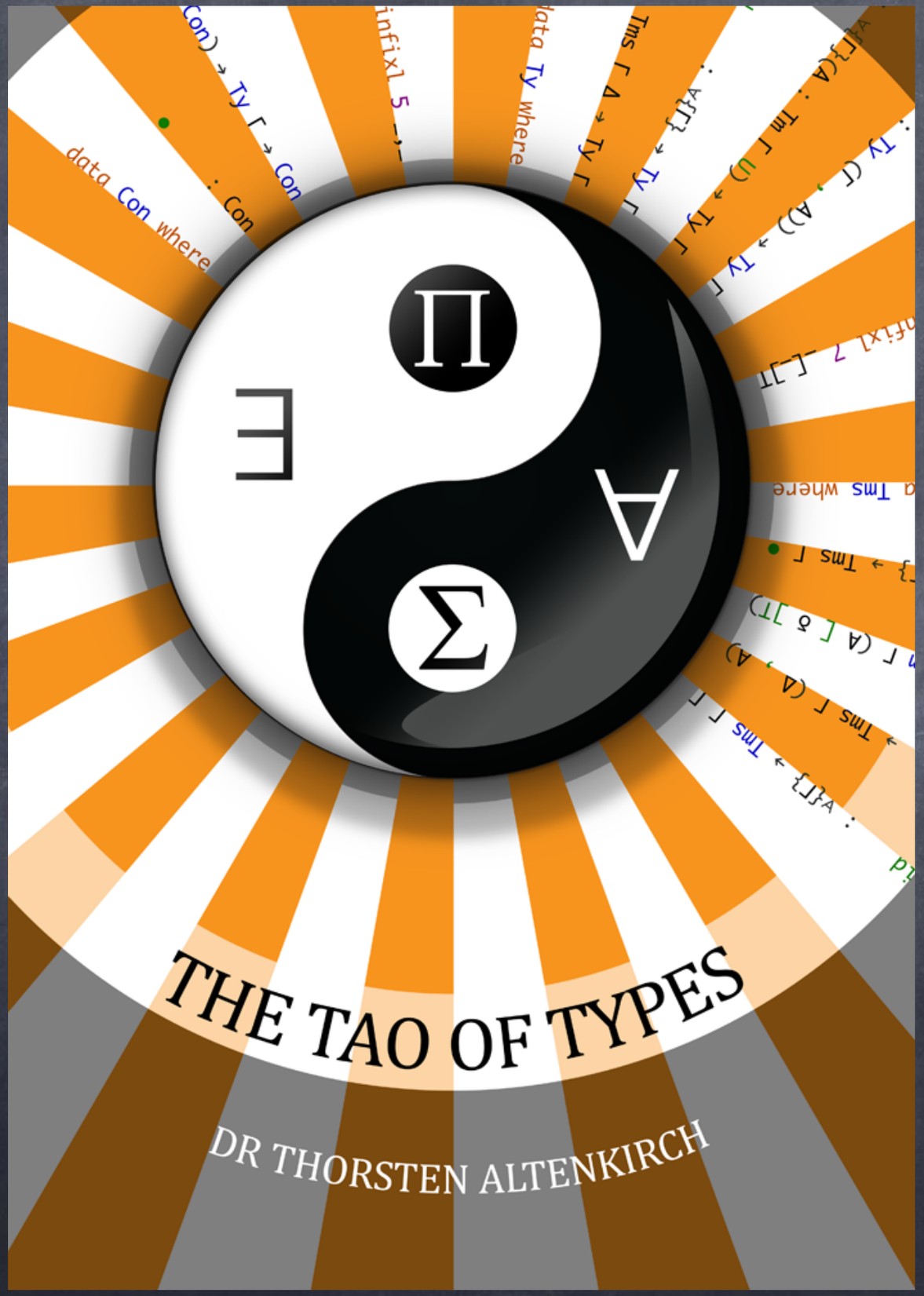
- Types in HoTT are viewed as weak ω-Groupoids
- Equality types expose this structure.
- This allows us to have a very extensional equality.
- It also enables the introduction of Higher Inductive Types (HITs)

Concepts in Type Theory

Π -types $\Pi x : A.B(x)$	$A \to B \quad A \Rightarrow B$ $\forall x : A.B\,x$
Σ -types $\Sigma x : A.B(x)$	$A \times B \quad A \wedge B$ $A + B \quad A \vee B$ $\exists x : A.B\,x$
inductive types	$\mathbb{N} \quad \mathrm{Bool} \quad \|\|A\|\|$
Equality types	$a = b$
Universes	$\mathbf{Type}_i : \mathbf{Type}_{i+1}$

Conclusions

- Modern Type Theory is constructive in two ways
  - Elements of types are constructed (not collected as for sets) giving rise to univalence
  - Truth is replaced by Evidence giving rise to propositions as types


Π
Σ
∃
∀
THE TAO OF TYPES
DR THORSTEN ALTENKIRCH