

The Univalence Axiom in Dependent Type Theory

Marc Bezem¹

¹Lectures partly based on: Thierry Coquand, *Théorie des Types Dépendants et Axiome d'Univalence*, Séminaire Bourbaki, 66ème année, 2013-2014, n° 1085

December 2016

Overview of Logics (more on this later)

Logic	Types	$\forall\exists$ -domains
1-sorted FOL	$I^n \rightarrow B$	I
k -sorted FOL	$[I_1 \cdots I_k]^n \rightarrow B$	I_1, \dots, I_k
HOL (later)	$T ::= B \mid I \mid (T \rightarrow T)$	any T
DTT (later)	\mathcal{U} (universes), Π -types, inductive types	any $A : \mathcal{U}$

- ▶ First-order logic: predicate logic (e.g., group theory, ZFC)
- ▶ I is the type of individuals, B of propositions
- ▶ 1-sorted FOL is usually presented untyped
- ▶ E.g., in $\exists x. \forall y. \neg E(y, x)$ quantification is over I , and $E(y, x)$, $\neg E(y, x)$, $\forall y. \neg E(y, x)$, $\exists x. \forall y. \neg E(y, x)$ are all of type B
- ▶ In 1-sorted FOL types are left implicit, apart from *arity*
- ▶ In k -sorted FOL types are explicitly given in the *signature*

Higher-order Logic (Church 1940)

- ▶ Types: I (individuals), $bool$ (propositions), and with A, B also $A \rightarrow B$ (these are called *simple types*)
- ▶ Terms are classified by their types: e.g., $c : I$; $f : I \rightarrow I$;
 $P : bool$; $\rightarrow : bool \rightarrow (bool \rightarrow bool)$; $Q : I \rightarrow bool$;
 $\neg Q(f(c)) : bool$; $\forall_I, \exists_I : (I \rightarrow bool) \rightarrow bool$, $(\forall_I Q) : bool$
- ▶ We also have, e.g., $\forall_{I \rightarrow bool}, \exists_{I \rightarrow I}$, quantification over unary predicates and functions, in fact, \forall_A, \exists_A for any type A : **HOL**
- ▶ Notation: $\forall x : A. Q(x)$ for $\forall_A Q$, $\exists x : A. Q(x)$ for $\exists_A Q$
- ▶ Example: we can express $Eq_A(t, u) : bool$ as

$$(\forall P : A \rightarrow bool. P(t) \rightarrow P(u)) : bool$$

- ▶ Inference system defines the ‘theorems’ of type $bool$
- ▶ Natural semantics in set theory: $bool = \{0, 1\}$, I a set

Extensionality Axioms in HOL

- ▶ Pointwise equal functions are equal:

$$(\forall x : A. Eq_B(f(x), g(x))) \rightarrow Eq_{A \rightarrow B}(f, g)$$

- ▶ Equivalent propositions are equal:

$$((P \rightarrow Q) \wedge (Q \rightarrow P)) \rightarrow Eq_{bool}(P, Q)$$

- ▶ Univalence Axiom (UA): 'equivalent things are equal', where the meaning of 'equivalent' depends on the 'thing'

Exercise: prove that Eq_A is an equivalence relation for all A

Dependent Type Theory, Π -types

- ▶ Limitation of HOL: not possible to define, e.g., algebraic structure on an arbitrary type; DTT can express this.
- ▶ Every mathematical object has a type, even types have a type: $a : A, A : U_0, U_0 : U_1, \dots$, the U_i are called universes (\mathcal{U})
- ▶ Fundamental in DTT: family of types $B(x), x : A$; for every $a : A$ we have $B(a) : \mathcal{U}$
- ▶ Context: $x_1 : A_1, x_2 : A_2(x_1), \dots, x_n : A_n(x_1, \dots, x_{n-1})$
- ▶ Example: $n : N, x : R(n), y : R(n)$ in n -dim lin alg
- ▶ If $B(x), x : A$ type family, then $\Pi_{x:A}. B(x)$ is the type of *dependent* functions $f(x) = b$ in context $x : A$, that is, b and its type may depend on x , $f(a) = (a/x)b : B(a)$ if $a : A$
- ▶ Example: $0 : \Pi n:N. R(n)$, $0(n)$ is n -dimensional zero vector
- ▶ Actually, $A \rightarrow B$ is $\Pi_{x:A}. B(x)$ with $B(x) = B$

Σ -types and algebraic structure

- ▶ If $B(x)$, $x : A$ type family, then $\Sigma_{x:A}. B(x)$ is the type of dependent pairs (a, b) with $a : A$ and $b : B(a)$
- ▶ Actually, $A \times B$ is $\Sigma_{x:A}. B(x)$ with $B(x) = B$
- ▶ A type of semigroups ($=_G$ explained later):

$$\Sigma G:\mathcal{U}. \Sigma m:G \rightarrow G \rightarrow G. \Pi x, y, z:G. m(x, m(y, z)) =_G m(m(x, y), z)$$

Representation of Logic in DTT

- ▶ Curry-Howard-de Bruijn: formulas as types, (constructive) proofs as programs (see Sørensen&Urzyczyn, Elsevier, 2006)
- ▶ Example: $f(x, y) = x$ for $x : A$, $y : B$, then $f : A \rightarrow (B \rightarrow A)$
- ▶ Curry, 1958: f is a proof of the tautology $A \rightarrow (B \rightarrow A)$
- ▶ Modus ponens: if $f : A \rightarrow B$, $a : A$, then $f(a) : B$
- ▶ Similarly, $g(x, y, z) = x(y(z))$ (composition) is a proof of

$$(B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

- ▶ Breakthrough in FOM: proofs as first-class citizens (!!!)
Constructive proofs can be executed as functional programs.
- ▶ Profound influence on computer science, constructive mathematics, computational linguistics

Representation of Logic in DTT (ctnd)

- ▶ A family of types $B(x)$, $x : A$ represents a unary predicate
- ▶ Truth (or rather: provability) is represented by inhabitation
- ▶ Universal quantification $\forall x : A. B(x)$ by $\prod_{x:A} B(x)$
- ▶ Implication $A \rightarrow B$ by, indeed, $A \rightarrow B$ ($= \prod_{x:A} B!$)
- ▶ Existential quantification $\exists x : A. B(x)$ by $\sum_{x:A} B(x)$
- ▶ $A \wedge B$ by $A \times B = \sum_{x:A} B(x)$ with constant $B(x) = B$
- ▶ $A \vee B$ by disjoint sum $A + B$ (next slide)
- ▶ \perp by the empty type N_0 (next slide)

Inductive Types

- ▶ $A + B$ is inductively defined by two *constructors*
 $inl : A \rightarrow (A + B)$, $inr : B \rightarrow (A + B)$
- ▶ What to do with objects $inl(a)$, $inr(b)$? Definition by cases!
- ▶ Destruction: $h : \prod z:A + B. C(z)$ can be defined given
 $f : \prod x:A. C(inl(x))$ and $g : \prod y:B. C(inr(y))$:

$$h(inl(x)) = f(x) \quad h(inr(y)) = g(y)$$

- ▶ Moral: $inl(a)$, $inr(b)$ are the only objects of type $A + B$
- ▶ For constant $C(z) = C$ this is Gentzen's \vee -elimination:
 $f : A \rightarrow C$, $g : B \rightarrow C$ define $h : A + B \rightarrow C$
- ▶ In words: if we can infer C from A , and from B , then we can prove C from $A + B$ (as there is no '3rd case')
- ▶ Extra: $p : A + B$ can be used in $C(p)$

Inductive Types (ctnd)

- ▶ Also inductively: $0 : N$ and if $n : N$, then $S(n) : N$
- ▶ What to do with numerals $S^k(0)$? Recursion and induction!
- ▶ Destruction: $f : \prod n:N. C(n)$ can be defined given $z : C(0)$ and $s : \prod n:N. (C(n) \rightarrow C(S(n)))$:

$$f(0) = z \quad f(S(n)) = s(n, f(n))$$

- ▶ Moral: numerals $S^k(0)$ are the only objects in N
- ▶ For constant $C(n) = C$ this is primitive recursion
- ▶ For non-constant $C(n)$: inductive proof of $\forall n : N. C(n)$
- ▶ Moral: primitive recursion is the non-dependent version of induction; Both replace the constructors by suitable terms.

Inductive Absurdity

- ▶ N_0 , the empty type or empty sum, representing *false* or absurdity, is inductively defined by **no** constructors
- ▶ Destruction: $h : \prod z:N_0. C(z)$ can be defined by **zero** cases, presuming nothing, h is 'for free' (induction principle for N_0)
- ▶ For constant $C(z) = C$ this is the Ex Falso rule $N_0 \rightarrow C$
- ▶ For non-constant $C(z)$: refinement of Ex Falso, probably used for the first time by VV to prove $\forall x, y : N_0. Eq_{N_0}(x, y)$, with Eq_A as on next slide
- ▶ Negation: $\neg A = (A \rightarrow N_0)$
- ▶ N_1 is the inductive type with one constructor, N_2 (aka Bool) with two constructors, and so on

Inductive Equality

- ▶ $Eq_A(x, y)$ (equality, Martin-Löf), in context $A : \mathcal{U}$, $x, y : A$, inductively defined by $1_a : Eq_A(a, a)$ for all $a : A$ (diagonal!)
- ▶ Since $Eq_A(x, y)$ is itself a type in \mathcal{U} , we can iterate:
 $Eq_{Eq_A(x, y)}(p, q)$ is equality of equality proofs of x and y
- ▶ Homotopy interpretation: $Eq_A(x, y)$ as path space, $Eq_{Eq_A(x, y)}(p, q)$ as higher path space, and so on
- ▶ Beautiful structure arises: an ∞ -groupoid
- ▶ Footnote (opinion): a miracle, unintended by Martin-Löf
- ▶ Discussion: a discovery comparable to the countable model of ZF, or to non-Euclidean geometries (without changing the theory)

Laws of Equality

- ▶ $(1_a : Eq_A(a, a) \text{ for all } a : A) + \text{induction} + \text{computation}$
- ▶ We actually want *transport*, for all type families B :

$$transp_B : B(a) \rightarrow (Eq_A(a, x) \rightarrow B(x))$$

and *based path induction*, for all type families C :

$$bpi_C : C(a, 1_a) \rightarrow \prod p:Eq_A(a, x). C(x, p)$$

plus natural equalities like $Eq_{B(a)}(transp_B(b, 1_a), b)$

- ▶ bpi_C is provable by induction, $transp_B$ special case of bpi_C
- ▶ Also provable: Peano's 4-th axiom $\neg Eq_N(0, S(0))$
- ▶ Proof: define recursively $B(0) = N$, $B(S(n)) = N_0$ and assume $p : Eq_N(0, S(0))$. We have $0 : B(0)$ and hence $transp_B(0, p) : N_0$.

Groupoid

- ▶ THM [H+S]: every type A is a groupoid with objects of type A and morphisms $p : Eq_A(a, a')$ for $a : A, a' : A$
- ▶ In more relaxed notation (only here with $=$ for Eq):
 1. $\cdot : x = y \rightarrow y = z \rightarrow x = z$
 2. $\cdot^{-1} : x = y \rightarrow y = x$
 3. $p = 1_x \cdot p = p \cdot 1_y$
 4. $p \cdot p^{-1} = 1_x, p^{-1} \cdot p = 1_y$
 5. $(p^{-1})^{-1} = p$
 6. $p \cdot (q \cdot r) = (p \cdot q) \cdot r$
- ▶ Proofs by induction: \cdot is $transp_{x=_}$, \cdot^{-1} is $transp_{_=x} refl_x$ (!)
- ▶ Also: $x, y : A, p, q : Eq_A(x, y), pq : Eq_{Eq_A(x, y)}(p, q) \dots$

The Homotopy Interpretation [A+W+V]

- ▶ Type A : topological space
- ▶ Object $a : A$: point in topological space
- ▶ Object $f : A \rightarrow B$: continuous function
- ▶ Universe \mathcal{U} : space of spaces
- ▶ Type family $B : A \rightarrow \mathcal{U}$: a specific fibration $E \rightarrow A$, where the fiber of $a : A$ is $B(a)$, and
- ▶ E is the interpretation of $\Sigma A B$: the total space
- ▶ $\Pi A B$: the space of sections of the fibration interpreting B
- ▶ $Eq_A(a, a')$: the space of paths from a to a' in A
- ▶ Correct interpretation of Eq_A (in particular, transport) is ensured by taking Kan fibrations (yielding homotopy equivalent fibers of connected points)

Some Homotopy Levels [V]

- ▶ Level -1 : $prop(P) = \prod_{x, y: P}. Eq_P(x, y)$
- ▶ Example: N_0 is a proposition, $prop(N_0)$ also (!)
- ▶ Level 0 : $set(A) = \prod_{x, y: A}. prop(Eq_A(x, y))$
- ▶ Example: N is a set, $set(N)$ is a proposition
- ▶ Proved above: N is not a proposition (Peano's 4-th axiom)
- ▶ Level 1 : $groupoid(A) = \prod_{x, y: A}. set(Eq_A(x, y))$
- ▶ Examples: N_0 , N (silly, the hierarchy is cumulative)
- ▶ Without UA it is consistent to assume $\prod A: \mathcal{U}. set(A)$
- ▶ With UA, \mathcal{U} is not a set (U_0 not a set, U_1 not a groupoid, ...)

The Univalence Axiom [V]

- ▶ Level -2 : $Contr(A) = A \times prop(A)$, A is *contractible*
- ▶ Examples: N_1 , $\sum x:B. Eq_B(x, b)$ for all $b : B$
- ▶ *Fiber* of $f : A \rightarrow B$ over $b : B$ is the type

$$Fib_f(b) = \sum x:A. Eq_B(f(x), b)$$

- ▶ *Equivalence (function)*: $isEquiv(f) = \prod b:B. contr(Fib_f(b))$
- ▶ *Equivalence (types)*: $(A \simeq B) = \sum f:A \rightarrow B. isEquiv(f)$
- ▶ Examples:
 - ▶ Logical equivalence of propositions
 - ▶ Bijections of sets
 - ▶ The identity function $A \rightarrow A$ is an equivalence, $A \simeq A$
- ▶ UA: for the canonical $idtoEquiv : Eq_{\mathcal{U}}(A, B) \rightarrow (A \simeq B)$,

$$ua : isEquiv(idtoEquiv)$$

More on UA

- ▶ Weak UA, $wua : (A \simeq B) \rightarrow Eq_{\mathcal{U}}(A, B)$
- ▶ Consequence: $Eq_{\mathcal{U}}(A, B) \simeq (A \simeq B)$ inhabited
- ▶ Informal: homotopy equivalent types in \mathcal{U} can be identified
- ▶ Example: \mathbb{N} and $\mathbb{Z}^{\geq 0}$ can be identified, etc.
- ▶ What means 'can be identified' here? Leibniz equal!
- ▶ Why not so in ZF? E.g., $0 = \{\}$, $S_n = n \cup \{n\}$, $S'n = \{n\}$.
Two encodings of \mathbb{N} , they disagree on $0 \in 2$.
- ▶ Crucial: the language of type theory strikes a balance
 - ▶ it is expressive (not too much encoding)
 - ▶ is not too expressive (cannot express things it shouldn't)

Consequences and Applications of UA/HoTT

- ▶ Function extensionality
- ▶ Description operator (define functions by their graph)
- ▶ The universe is not a set ($Eq_{\mathcal{U}}(N, N)$ refutes UIP)
- ▶ Practical: transport of structure and results between equivalent types, without the need for ‘transportability criteria’ [Bourbaki 4].

[wiki/Equivalent_definitions_of_mathematical_structures](https://en.wikipedia.org/wiki/Equivalent_definitions_of_mathematical_structures)

- ▶ Practical: formalizing homotopy theory **synthetically**
- ▶ Higher inductive types, example: the circle \mathbb{S}^1
 - ▶ a point constructor $\text{base} : \mathbb{S}^1$
 - ▶ a path constructor $\text{loop} : \text{base} =_{\mathbb{S}^1} \text{base}$
 - ▶ induction + computation
- ▶ What is $\text{base} =_{\mathbb{S}^1} \text{base}$? (provably equivalent to \mathbb{Z})
- ▶ ...

Consumer Test of Logics

Logic	Types	$\forall\exists$ -domains	Rem.
1-sorted FOL	$I^n \rightarrow B$	I	1
k -sorted FOL	$[I_1 \cdots I_k]^n \rightarrow B$	I_1, \dots, I_k	1
HOL	$T ::= B \mid I \mid (T \rightarrow T)$	any T	1,3
DTT	\mathcal{U} (universes), Π -types, inductive types	any $A : \mathcal{U}$	2,4,5

1. Proofs are not first-class citizens.
2. Proofs are first-class citizens (part of object language).
3. Strength depends on comprehension axioms and similar devices, e.g., $\exists P. \forall x. (Px \leftrightarrow \phi)$ or Hilbert's ϵ .
4. Strength depends on inductive types and im/predicativity, e.g., type $\Pi A : \mathcal{U}_0. A$ landing in universe $\mathcal{U}_0/\mathcal{U}_1$.
5. In DTT we often reason logically with 'inhabited types'