# The Univalence Axiom in Dependent Type Theory

Marc Bezem, lectures based on [1] and [2]

[1]The Univalent Foundations Program, *Homotopy Type Theory*, `https://homotopytypetheory.org/book/`

[2]Thierry Coquand, *Théorie des Types Dépendants et Axiome d'Univalence*, Séminaire Bourbaki, 66ème année, 2013-2014, n$^o$ 1085

Spring 2018

## Overview of Logics

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

- First-order logic: predicate logic (e.g., set theory ZFC)
- $I$ is the type of individuals, $B$ of propositions
- 1-sorted FOL is usually presented untyped
- In 1-sorted FOL types are left implicit, apart from *arity*
- 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 k-sorted FOL types are explicitly given in the *signature*

# Higher-order Logic (Church 1940)

- Types: $I$ (individuals), *bool* (propositions), and if $A, B$ are types, then also $A \to B$ (these types are called *simple* types)
- Terms are classified by their types, e.g.,
  - $c : I$
  - $f : I \to I$
  - $f(c) : I$
  - $P : bool$
  - $Q : I \to bool$ ('propositional function')
  - $\to : bool \to (bool \to bool)$
  - $\neg : bool \to bool$
  - $P \to \neg Q(f(c)) : bool$
  - $\forall_I : (I \to bool) \to bool$ (universal quantifier over $I$)
  - $(\forall_I\, Q) : bool$, also denoted $\forall x{:}I.\ Q(x)$
- We also have, e.g., $\exists_I$, $\forall_{I\to bool}$, $\exists_{I \to I}$ for quantification over $I$, over unary predicates, and over unary functions, respectively
- In fact, we have $\forall_A, \exists_A$ for any type $A$: HOL

# Higher-order Logic (Cntd)

- Inference system defines the 'theorems' of type *bool*
- Natural semantics in set theory: $bool = \{0, 1\}$, $I$ a set
- Example: we can express equality $Eq_A(t, u) : bool$ as

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

- Exercise: prove that $Eq_A$ is an equivalence relation for any $A$
- Refinement: prove symmetry of $Eq_A$ without using the law of the excluded middle
- Moral of the exercise: higher-order quantification is powerful

# Extensionality Axioms in HOL, anticipating UA

- Pointwise equal functions are equal:

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

- Equivalent propositions are equal:

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

- Neither of these axioms is provable in HOL (but they are true in the set-theoretic semantics)
- Univalence Axiom (UA): 'equivalent things are equal' (the meaning of 'equivalent' depends on the 'thing')
- UA is not true set-theoretically, since sets can be 'equivalent' without being equal. This is vague, but can be made precise by taking sets to be 'equivalent' when they are in bijective correspondence (same cardinality). Another example will come later.

# Dependent Type Theory, Π-types

- Limitation of HOL: not natural 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: $a : A$, $A : U_0$, $U_0 : U_1, \ldots$, the $U_i$ are called universes ($\mathcal{U}$)
- Fundamental in DTT: family of types $B(x)$, $x : A$; that is, for every $a : A$ we have $B(a) : \mathcal{U}$ (so, $B$ has type $A \to U_0$)
- Context: $x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, ..., x_{n-1})$
- Example: $n : N$, $x : R(n)$, $y : R(n)$ in $n$-dim LinAlg
- If $B(x)$, $x : A$ 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, $A \to B$ 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 can be defined in DTT as ($=_G$, equality on $G$, will be explained later):

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

## 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 $f : A \to (B \to A)$
- Curry, 1958: $f$ is a proof of the tautology $A \to (B \to A)$
- Modus ponens: if $f : A \to B$, $a : A$, then $f(a) : B$
- Similarly, $g(x, y, z) = x(y(z))$ (composition) is a proof of

$$(B \to C) \to ((A \to B) \to (A \to C))$$

- 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 $\Pi x{:}A.\,B(x)$
- Implication $A \to B$ by, indeed, $A \to B$ ( $= \Pi x{:}A.\,B!$)
- Existential quantification $\exists x{:}A.\ B(x)$ by $\Sigma x{:}A.\,B(x)$
- $A \wedge B$ by $A \times B = \Sigma x{:}A.\,B(x)$ with constant $B(x) = B$
- $A \vee B$ is represented by disjoint sum $A + B$ (next slide)
- $\perp$ is represented by the empty type $N_0$ (next slide)
- Negation $\neg A$ is represented by $A \to N_0$
- NB: $\Sigma$ and $+$ are stronger than in ordinary logic (explain ...)

# Inductive Types

- $A + B$ is inductively defined by two *constructors* $inl : A \to (A + B)$, $inr : B \to (A + B)$
- How to destruct objects $inl(a), inr(b)$? Definition by cases!
- Destruction: $h : \Pi z{:}A + B.\, C(z)$ can be defined given $f : \Pi x{:}A.\, C(inl(x))$ and $g : \Pi y{:}B.\, C(inr(y))$:

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

- Moral: $inl(a), inr(b)$ are the only objects of type $A + B$
- For constant $C(z) = C$ this is Gentzen's $\vee$-elimination: $f : A \to C,\ g : B \to C$ define $h : A + B \to C$
- In words: if we can prove $C$ from $A$, and from $B$, then we can prove $C$ from $A + B$
- Extra in DTT: $p : A + B$ can be used in $C(p)$

# Inductive Types (ctnd)

- Also inductively: $0 : N$ and if $n : N$, then $S(n) : N$
- How to destruct numerals $S^k(0)$? Recursion and induction!
- Destruction: $f : \Pi n{:}N.\, C(n)$ can be defined given $z : C(0)$ and $s : \Pi n{:}N.\, (C(n) \to C(S(n)))$:

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

- 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

## Inductive Absurdity

- $N_0$, the empty type or empty sum, representing *false* or absurdity, is inductively defined by no constructors
- Destruction: $h : \Pi z{:}N_0.\, C(z)$ 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 $N_0 \to C$
- For non-constant $C(z)$: refinement of Ex Falso, used elegantly by VV to prove $\Pi x, y{:}N_0.\ Eq_{N_0}(x, y)$ (for $Eq_{N_0}$: next slide)
- Negation: $\neg A = (A \to N_0)$
- $N_1$ (aka Unit) 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 $\infty$-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

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

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

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

$$bpi_C : C(a, 1_a) \to \Pi p{:}Eq_A(a, x).\, C(x, p)$$

  plus natural equalities like $transp_B(b, 1_a) = b$
- $bpi_C$ 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. $. \cdot . : x = y \to y = z \to x = z$
  2. $.^{-1} : x = y \to y = x$
  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 $transp_{x=\_}$, $^{-1}$ is $transp_{\_=x}\, 1_x$ (!)
- Also: $x, y : A,\ p, q : Eq_A(x, y),\ pq : Eq_{Eq_A(x,y)}(p, q)$ ...

# The Homotopy Interpretation [A+W+V]

- Type $A$: topological space
- Object $a : A$: point in topological space
- Object $f : A \to B$: continuous function
- Universe $\mathcal{U}$: space of spaces
- Type family $B : A \to \mathcal{U}$: a specific fibration $E \to A$, 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 $-1$: $prop(P) = \Pi x, y{:}P.\, Eq_P(x, y)$
- Example: $N_0$ is a proposition, $prop(N_0)$ also (!)
- Level 0: $set(A) = \Pi x, y{:}A.\, prop(Eq_A(x, y))$
- Example: $N$ is a set, $set(N)$ is a proposition
- Proved above: $N$ is not a proposition (Peano's 4-th axiom)
- Level 1: $groupoid(A) = \Pi x, y{:}A.\, set(Eq_A(x, y))$
- Examples: $N_0$, $N$ (silly, the hierarchy is cumulative)
- Without UA it is consistent to assume $\Pi A{:}\mathcal{U}.\, set(A)$
- 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: $N_1$, $\Sigma x{:}B.\, Eq_B(x, b)$ for all $b : B$
- *Fiber* of $f : A \to B$ over $b : B$ is the type

$$Fib_f(b) = \Sigma x{:}A.\, Eq_B(f(x), b)$$

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

$$ua : isEquiv(idtoEquiv)$$

# More on UA

- Consequence of UA: $Eq_{\mathcal{U}}(A, B) \simeq (A \simeq B)$ inhabited
- Weak UA, $wua : (A \simeq B) \to Eq_{\mathcal{U}}(A, B)$
- Informal: homotopy equivalent types in $U$ can be identified
- Example: $\mathbb{N}$ and $\mathbb{Z}$ can be identified, don't forget transport
- Is this good or bad, what means 'can be identified' here?
- Why not so in ZF? E.g., $0 = \{\}$, $Sn = n \cup \{n\}$, $S'n = \{n\}$. 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
- Practical: formalizing homotopy theory synthetically
- Higher inductive types, example: the circle $\mathbb{S}^1$
  - a point constructor base : $\mathbb{S}^1$
  - a path constructor loop : base $=_{\mathbb{S}^1}$ base
  - induction + computation
- What is base $=_{\mathbb{S}^1}$ base? (provably equivalent to $\mathbb{Z}$)
- ...

# Consumer Test of Logics

| Logic | Types | $\forall\exists$-domains | Rem. |
|---|---|---|---|
| 1-sorted FOL | $I^n \to B$ (opt. $I^n \to I$) | $I$ | 1 |
| $k$-sorted FOL | $[I_1 \mid \cdots \mid I_k]^n \to B$ $(\ldots)$ | $I_1, \ldots, I_k$ | 1 |
| HOL | $T ::= B \mid I \mid (T \to T)$ | any $T$ | 1,3 |
| DTT | $\Pi$-types, $\mathcal{U}$ (universes), 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 $\epsilon$.
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'