The Univalence Axiom in Dependent Type Theory

Marc Bezem[1]

[1]Lecture based on: Thierry COQUAND, *Théorie des Types Dépendants et Axiome d'Univalence*, Séminaire Bourbaki, 66ème année, 2013-2014, n^o 1085

Fall 2015

Higher-order Logic (HOL)

- First-order logic: predicate logic (e.g., group theory, ZFC)
- Higher-order logic (Church):
  - Types: I (individuals), *bool* (propositions), and with A, B also $A \to B$ (these are called *simple* types)
  - Terms are classified by their types: e.g., $f : I \to I$; $c, f(c) : I$; $P : bool$; $\wedge, \to : bool \to (bool \to bool)$; $P \vee \neg P : bool$; $Q : I \to bool$; $\forall_I, \exists_I : (I \to bool) \to bool$, $(\forall_I\, Q) : bool$
  - We also have, e.g., $\forall_{I \to bool}, \exists_{I \to I}$, 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

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

  - 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:

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

- Equivalent propositions are equal:

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

- 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 and Σ-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: $a : A$, $A : U_0$, $U_0 : U_1, \ldots$, the U_i are called universes
- Fundamental notion: family of types $B(x),\ x : A$; for every $a : A$ we have $B(a) : \mathcal{U}$ ('a has property B')
- Context: $x_1 : A_1,\ x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, ..., x_{n-1})$
- Example: $x : N,\ p : P(x),\ y : N,\ q : Q(x, y)$
- If $B(x),\ x : A$ type family, then $\Pi x{:}A.\, B(x)$ is the type of dependent functions (later: *sections*): $f(x) = b$ in context $x : A$, i.e., b depending on x, $f(a) = (a/x)b : B(a)$ if $a : A$
- Actually, $A \to B$ is $\Pi x{:}A.\, B(x)$ with $B(x) = B$
- Dually, we have $\Sigma x{:}A.\, B(x)$, the type of dependent pairs (a, b) with $a : A$ and $b : B(a)$.

Representation of Logic in DTT

- Curry-Howard-de Bruijn: formulas as types, (constructive) proofs as programs
- Define $f(x, y) = x$ for $x : A,\ y : B$, then $f : A \to (B \to A)$
- Curry: f is a proof of the tautology $A \to (B \to A)$ (!!!)
- Similarly, $g(x, y, z) = x(y(z))$ (composition) is a proof of

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

- Modus ponens: if $f : A \to B$, $a : A$, then $f(a) : B$
- $\forall x : A.\ B(x)$ as $\Pi x{:}A.\ B(x)$
- $\exists x : A.\ B(x)$ as $\Sigma x{:}A.\ B(x)$
- $A \wedge B$ as $A \times B = \Sigma x{:}A.\ B(x)$ with constant $B(x) = B$
- $A \vee B$ as disjoint sum $A + B$ (below)
- $\perp$ as the empty type N_0 (below)

Inductive Types

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

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

- For constant $C(z) = C$ this is Gentzen's $\vee$-elimination
- Also inductively: $0 : N$ and if $n : N$, then $S(n) : N$
- Destruction: $f : \Pi n{:}N.\, C(n)$ can be defined by, given $z : C(0)$ and $s : \Pi n{:}N.\, (C(n) \to C(S(n)))$:

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

- 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 Types (less familiar)

- N_0 (the empty type, or empty sum, representing *false*, $\neg A = (A \to N_0)$), 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, probably used for the first time by VV to prove $\forall x, y : N_0.\ Eq_{N_0}(x, y)$
- $Eq_A(x, y)$ (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
- Beautiful structure arises: an ∞-groupoid (miracle!)

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 $Eq_{B(a)}(transp_B(b, 1_a), b)$
- These are all provable by induction
- 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 $transp_{\_=x}\ refl_x$ (!)
- 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 $idtoEquiv : Eq_{\mathcal{U}}(A, B) \to (A \simeq B)$,

$$ua : isEquiv(idtoEquiv)$$

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: formalizing homotopy theory
- Practical: transport of structure and results between equivalent types, without the need for [Bourbaki 4] 'transportability criteria' .
  wiki/Equivalent_definitions_of_mathematical_structures
- 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? (should be $\mathbb{Z}$)
- ...