# Logic

24 April 2024 11:05

## IV. Logic, revisited

### IV.1. Proposition as types

**Theorem:** $LEM : \prod_{A:\mathcal{U}} A + \neg A$ contradicts univalence.

Step 1: LEM $\Rightarrow$ Double Negation $DN : \prod_{A:\mathcal{U}} \neg\neg A \to A$

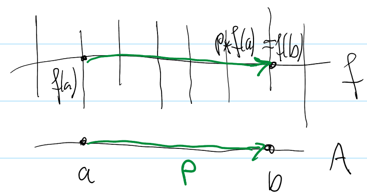
Proof: Given $A$, $LEM(A) : A + \neg A$ leads to cases $a : A$ and $na : \neg A$.
Given $nna : \neg\neg A$, the first case yields the required $a : A$.
second case $\rightsquigarrow$ $nna(na) : \mathrm{False} \underset{\text{Ex Falso}}{\Longrightarrow} a : A$

Step 2: Function Extensionality $\Rightarrow$ $p : \prod_{u,v : \neg\neg A} u = v$

Proof: Given $x : \neg A$, $u(x), v(x) : \mathrm{False} \underset{\text{Ex Falso}}{\Longrightarrow} \prod_{x:\neg A} u(x) = v(x) \overset{\text{Function Extensionality}}{\Longrightarrow} u = v$

Idea: If the type $A$ "changes continuously" to a type $B$ (along an equality $p : A = B$),
then a function $\neg\neg A \to A$ should "change continuously" to a function $\neg\neg B \to B$.
Because of Step 2, this is impossible for equalities coming from fixed-point free autoequivalences.

Lemma: Continuity of dependent functions $f : \prod_{a:A} P(a)$.
For $p : a =_A b$, there is $apd_f(p) : p_* f(a) = f(b)$.




Proof: By induction on $p$.

Proof of Theorem: Step 1 $\Rightarrow$ Enough to show that Double Negation is false.
Consider fixed-point free autoequivalence $\mathrm{non\_id}_2 : 2 \to 2$ (see Thm. in II.3)
Univalence $\Rightarrow$ corresponding equality $\mathrm{non\text{-}id}p_2 : 2 = 2$

$$DN(2)(u) \overset{\text{Lemma}}{=} (\mathrm{non\_id}p_{2*} DN(2))(u) = \mathrm{non\_id}p_{2*}\left(DN(2)\left((\mathrm{non\_id}p_2^{-1})_*(u)\right)\right)$$

Lemma: $A :\equiv \mathcal{U}$, $P(a) :\equiv \neg\neg A \to A$, $f :\equiv DN$
$a \equiv b :\equiv 2$, $p :\equiv \mathrm{non\_id}p_2$

induction on $p : A =_{\mathcal{U}} B$ :
$$\begin{array}{ccc} P(A) & \xrightarrow{f} & Q(A) \\ \uparrow p_*^{-1} & & \downarrow p_* \\ u : P(B) & \xrightarrow[p_* f]{} & Q(B) \end{array}$$

$= \mathrm{non\_id}p_{2*}(DN(2)(u)) = \mathrm{non\_id}_2(DN_2(u))$ ...

$a \equiv b :\equiv \mathbb{2}$, $p :\equiv \text{non-id}p_{\mathbb{2}}$ $\quad u : T(b) \xrightarrow[p*f]{} U(b)$

$$\underset{\text{Step 2}}{=} \text{non-id}p_{\mathbb{2}} * (DN(\mathbb{2})(u)) \underset{\text{univalence}}{=} \text{non-id}_{\mathbb{2}}(DN_{\mathbb{2}}(u)) \quad : \text{contradiction!}$$

<ins>Remarks</ins>: (1) It is not possible to exhibit a type $A$ for which $A + \neg A$ is false.

(2) The contradiction arises from fixed-point free autoequivalences

$\Rightarrow$ LEM can hold when restricted to types containing at most one element (up to propositional equality)

## IV.2. <ins>Proposition, sets, ... : n-types</ins>

<ins>Definition</ins>: $\boxed{\text{Propositions are types } P \text{ such that for all } x, y : P \text{ we have } p : x = y.}$ condition on identity types

We can assume LEM for propositions $\equiv$ $\text{LEM}_{-1}$ as an axiom.

More generally, we can do logic just on propositions: $\boxed{\text{"Proof Irrelevance"}}$

<ins>Definition</ins>: $\boxed{\text{Sets are types } S \text{ such that for all } x, y : S,\ x =_S y \text{ is a proposition.}}$ ... of identity types

Several equalities require "inner structure" of objects

$\rightsquigarrow$ cumulative hierarchy

upwards

<ins>Lemma 1</ins>: Propositions are sets.

<ins>Proof</ins>: Assume $p(x,y) : x =_A y$ for all $x, y : A$. Fix $x : A$ and define $q(y) :\equiv p(x,y)$

For all $y, z : A$, $r : y = z$ we have $\text{apd}\, q\, r : r_*(q(y)) = q(z)$

$\quad\quad \overset{\text{induction on } r}{=}\ q(y) \bullet r$

$\Rightarrow$ For all $r, s : y = z$ we have $r = q(y)^{-1} \bullet q(z) = s$.

<ins>Lemma 2</ins>: A type $A$ is a proposition if and only if for all $x, y : A$, the type $x =_A y$ is contractible. downwards

<ins>Proof</ins>: "$\Leftarrow$" $x =_A y$ is contractible $\Rightarrow$ $x =_A y$ is inhabited

"$\Rightarrow$" $A$ is a proposition $\underset{\text{Lem. 1}}{\Rightarrow}$ $A$ is a set $\Rightarrow$ $x =_A y$ is a proposition, contractible to any of its proofs.

<ins>Definition</ins>: $\boxed{n\text{-types are types such that for all } x, y : A,\ x =_A y \text{ is an } (n-1)\text{-type.}}$

base case $\equiv$ contractible types $\equiv$ $(-2)$-types

$\cap$

propositions $\equiv$ $(-1)$-types

$\cap$

sets $\equiv$ $0$-types (historically / pragmatically)

$\cap$

...

## IV.3. Equivalences of sets

Proposition: Bijections between sets and equalities of sets are equivalent.

Proof: Equalities of set elements are equal, by definition. $\Rightarrow$ Bijections between sets are always equivalences

$\Rightarrow$ equivalent to equalities, by univalence.

Corollary: Sets of the same cardinality are equal.

Remark: Most mathematicians are used to "sets are equal if they have the same elements".

= Axiom of Extensionality in ZFC

HoTT point of view:
- It is impossible to distinguish elements in sets without further structure
- elements in two sets = objects in two types cannot be the same, they can only be paired up by a bijection.
- But bijections correspond to equalities of the sets.

Proposition: The type of sets is not a set.

Proof: $id_2$ and $non\text{-}id_2$ are two different autoequivalences of the set $2$

encode-decode method, see II.3

$\Rightarrow$ $\exists$ two different equalities $2 = 2$

univalence

## IV.4. Ways to do logic in type theory

- proposition as types: too much structure $\Rightarrow$ e.g. LEM fails
- propositions as mere types $\equiv$ all objects are equal, as in HoTT
- Assume Axiom K, as in Agda: $\prod_{x,y:A} x = y$ is a proposition.

  not consistent with univalence (but can be removed in Agda!)

- work in "logic-enriched" type theory, as in Coq, Lean 3/4:

  introduce a separate "sort" for propositions like equalities, behaving somewhat like a type, but with restricted inductive constructions; one of them, called "Singleton Elimination", is not consistent with univalence.