Type Theories as Foundational Languages

Zhaohui Luo [罗朝晖]
Royal Holloway, Univ of London
Zhaohui.Luo@hotmail.co.uk
https://www.cs.rhul.ac.uk/home/zhaohui/

This talk – three parts

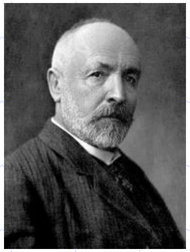
- I. Introduction to (modern) type theories [现代类型论]
  - History (simple/dependent), basics, logic, (im)predicativity, …
- II. Type theories as foundational languages
  - Representational adequacy, meta-theory, equalities, …
- III. Type theory for foundations of mathematics
  - Univalent foundations and homotopy type theory [单价基础与同伦类型论]

Part I. Modern Type Theories

[现代类型论]

Origin of type theory

- Foundations of mathematics and paradoxes
  - Naïve set theory (Cantor, …)
  - Paradox in naïve set theory (Russell 1903)
  - Crisis in foundations of mathematics
- Set theory by Zermelo
  - Axiomatic set theory (1908; later ZFC etc.)
  - Widely accepted foundations in math community
- Type theory by Russell
  - Ramified type theory (*Principia Math.* 1910-13, 1925)
  - Vicious circle principle ("impredicativity" like $\forall X.X$)
  - Ramified hierarchy – problematic "axiom of reducibility"



Simple type theory

- Ramsey (1926)
  - Logical v.s. semantic paradoxes
  - Russell's paradox v.s. (e.g.) Liar's paradox
  - Impredicativity is circular, but not vicious
  - So, Russell's ramified TT can be "<u>simplified</u>" to simple TT.

- Church's simple type theory (1940)
  - Formal system based on λ-calculus
  - Types as in ramified TT (e, t, $e \to t$, …)
  - Higher-order logic (formulas like $\forall X.X$)
  - Wide applications (Montague semantics, proof assistants, …)

Note: "Simple" could have another meaning: only "simple" types …

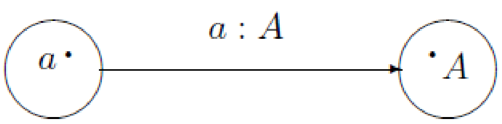
Modern Type Theories

- Foundations of constructive math (Bishop 67)
  - Feferman, Friedman, Martin-Löf, Myhill

- Martin-Löf has introduced/employed
  - Basic concepts:
    judgements, contexts, definitional equality
  - Type constructors:
    dependent types, inductive types, type universes
  - Curry-Howard principle of propositions-as-types
- From now on, by type theories, we mean
  Modern Type Theories (or MTTs),
  rather than the simple type theory.

Judgements – basic notion in type theory

- Membership judgement
  - a : A – a is an object of type A.



- What is A? A can be:
  - data type: eg, Nat, A→B, Πx:A.B [see next slide for Π-type]
  - propositional type: eg, ∀x:A.P
  - type universe: a type of some other types
- Comparison with set theory:
  - Judgement "a : A" is <u>not</u> a logical formula
  - Different from "s ∈ S", which is a formula (say in FOL)
  - Logic is a part of type theory (propositional types) [see slide]

Π-types: example of dependent types [*]

- Πx:A.B(x) – dependent function type
  - Informally, representing collection { f ∈ A→$\bigcup_{a \in A}$B(a) | ∀a∈A. f(a)∈B(a) }
  - f : Πx:Human.Parent(x) ➔ f(h) is father/mother of h (not others'!)
- Formally rules for Π-types
  - Formation rule
  - Introduction rule
  - Elimination rule
  - Computation rule

$$\frac{\Gamma \vdash A\ type \quad \Gamma,\ x{:}A \vdash B\ type}{\Gamma \vdash \Pi x{:}A.B\ type}$$

$$\frac{\Gamma,\ x{:}A \vdash b : B}{\Gamma \vdash \lambda x{:}A.b : \Pi x{:}A.B}$$

$$\frac{\Gamma \vdash f : \Pi x{:}A.B \quad \Gamma \vdash a : A}{\Gamma \vdash f(a) : [a/x]B}$$

$$\frac{\Gamma,\ x{:}A \vdash b : B \quad \Gamma \vdash a : A}{\Gamma \vdash (\lambda x{:}A.b)(a) = [a/x]b : [a/x]B}$$

Relationship between logic and set/type theory

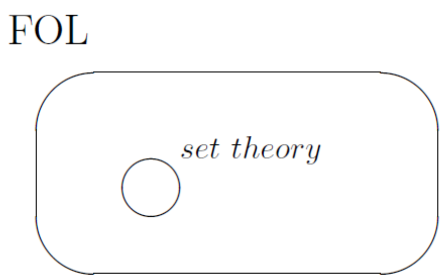


Figure 1: Set theory – a theory in first-order logic

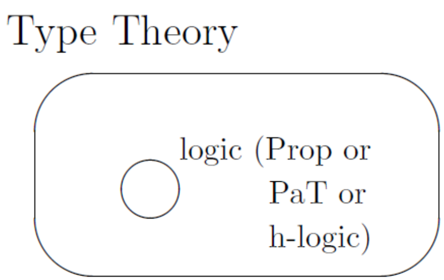


Figure 2: Logic is a part of type theory

MLTT [Martin-Löf 1975] – predicative type theory

- Propositions as types (Curry-Howard) in type theory:

0	1	$A+B$	$A\times B$	$A\rightarrow B$	$\sum_{x:A} B(x)$	$\prod_{x:A} B(x)$
$\bot$	$\top$	$A\vee B$	$A\wedge B$	$A\Rightarrow B$	$\exists_{x:A} B(x)$	$\forall_{x:A} B(x)$

  - PaT logic in MLTT: e.g., $A\wedge B = A\times B$, $A\vee B = A+B$, …
  - Note: $\vee/\exists$ thus defined are non-standard – "double role" problem: (1) "image(f)" in math [Escardo 17] (2) size in sem [Luo 18]
- MLTT is predicative.
  - Strictly hierarchical constructions.
  - There is no impredicative type formation such as "$\forall X.X$".

Predicativity v.s. Impredicativity

- A type of all types
  - Martin-Löf 1971: a (too strong) impredicative type theory with a type V of all types ➔ Girard's paradox ➔ inconsistency
  - Analysis: the origin of V came from two ideas:

    (1) All propositions form an (impredicative) type.

    (2) props = types (not just PaT, but all types are props as well!)

    You get V by substituting "types" for "propositions" in (1).
- Predicative v.s. impredicative types theories:
  - Insisting on (2) and disregarding (1) ➔ predicative MLTT
  - Following (1) and disagreeing with (2) ➔ impredicative UTT (UTT – next page)

UTT [Luo 89,94] – an impredicative type theory

- UTT – Unifying theory of Dependent Types (MLTT + CC)

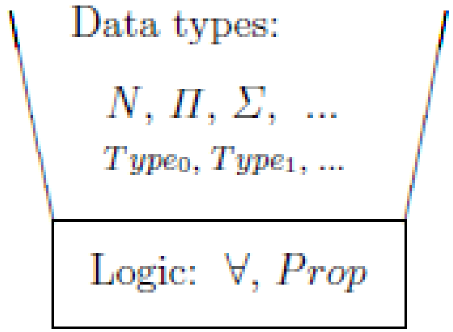


Fig. 1. The type structure in UTT.

- UTT has nice meta-theoretic properties
  - Goguen's PhD thesis on "Typed Operational Semantics" (1994)
  - Strong normalisation, which implies, e.g., consistency etc.

$\forall$-propositions: impredicative types [*]

- Universal quantification in Prop

$$\frac{\Gamma \vdash A\ type \quad \Gamma,\ x{:}A \vdash P : Prop}{\Gamma \vdash \forall x{:}A.B : Prop}$$

- Other logical operators can be defined by
  - As in second/higher-order logics (c.f. Prawitz's work)
  - For example, $P \wedge Q = \forall X : Prop.\ (P \Rightarrow Q \Rightarrow X) \Rightarrow X.$

    $\exists x : A.\ P(x) = \forall X : Prop.\ (\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X.$
- Formation of $\forall$ is impredicative (different from Π-types)
  - Prop, the collection of propositions, is a type itself.
  - Impredicative universe with "circular" props like $\forall$X:Prop.X

Proof technology based on type theories

- Proof assistants – interactive proof development
  - MTT-based: Agda, Coq, Lean, Lego, NuPRL, Plastic, …
  - HOL-based: HOL, Isabelle, …
- Applications of proof assistants

  - Formalisation of mathematics
    - 4-colour theorem (Coq), Kepler conjecture (Isabelle)
    - Univalent foundations of mathematics (Agda, …)
  - Computer Science:
    - program verification and advanced programming
  - Computational Linguistics
    - NL reasoning based on MTT-semantics (Coq)

Part II. Type Theories as Foundational Languages

Foundational adequacy – four aspects

(1) Basic representational adequacy

(2) Meta-theoretical justifications

(3) Various adequacy requirements in applications

(4) Two notions of equality

1. Natural numbers: example of basic adequacy

- Peano axioms: logical theory for natural numbers.
  [N is a predicate and n∈N stands for N(n)]

$$(P1)\ \ 0 \in N$$
$$(P2)\ \ \forall x.\ x \in N \Rightarrow succ(x) \in N$$
$$(P3)\ \ \forall x, y.\ x, y \in N \wedge succ(x) = succ(y) \Rightarrow x = y$$
$$(P4)\ \ \forall x.\ x \in N \Rightarrow 0 \neq succ(x)$$
$$(P5)\ \ \forall P.\ P(0) \wedge [\forall x.\ x \in N \wedge P(x) \Rightarrow P(succ(x))] \Rightarrow \forall z.\ z \in N \Rightarrow P(z)$$

- Martin-Löf's idea
  - Inductive types as "computational theories"
  - Example – Nat, the type of natural numbers

Rules for Nat

- Formation and introduction rules (canonical nats)

$$\frac{}{Nat\ type} \qquad \frac{}{0 : Nat} \qquad \frac{n : Nat}{succ(n) : Nat}$$

- Elimination rule (induction)

$$\frac{\begin{array}{l}\Gamma, z : Nat \vdash C(z)\ type \quad \Gamma \vdash n : Nat \\ \Gamma \vdash c : C(0) \quad \Gamma, x : Nat, y : C(x) \vdash f(x, y) : C(succ(x))\end{array}}{\Gamma \vdash \mathcal{E}_{Nat}(c, f, n) : C(n)}$$

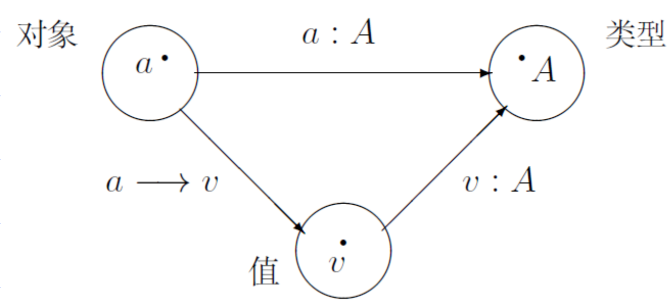
- Computation rules (primitive recursion)

$$\begin{array}{rcl}\mathcal{E}_{Nat}(c, f, 0) & = & c \\ \mathcal{E}_{Nat}(c, f, succ(n)) & = & f(n, \mathcal{E}_{Nat}(c, f, n))\end{array}$$

- Notes: All Peano axioms are either rules or theorems.

2. Meta-theoretic studies

- Intuitive understanding based on computation:

- 

Example: A = Nat, a = 3+4, v = 7.

- How to guarantee that computation a→v terminates ?

Meta-theory

- Meta-theory of type theories
  - Computation is central.
    - Strong normalisation: All computations terminate.
    - This usually implies canonicity and logical consistency.
  - Sophisticated, tedious and rather hard to do
    - Many many theorems/lemmas/concepts/… [examples in next 2 slides]
  - ECC/UTT's meta-theoretic studies [Luo 1990, Goguen 1994]
- Caveat:
  - Meta-theory depends on consistency of meta-language (set theory) – believed to be true, but …
  - Desire/wish: can we argue for "correctness" directly? (meaning theory …, not in this talk)

Meta-theoretic theorems: examples [*]

- Church-Rosser theorem (CR) [CR定理]
  - If a=b : A, then there exists c : A s.t. a → c and b → c.
- Subject Reduction (SR) [主题归约]
  - If a : A and a → b, then b : A.
- Strong Normalisation (SN) [强正规化]
  - Every computation from a well-typed term terminates.
- Logical consistency (in UTT) [逻辑相容性]
  - ∀X:Prop.X (false) is not provable (in the empty context).
- Decidability (of type-checking) [（类型检测的）可判定性]
  - It is decidable whether a judgement is correct (derivable).
- Equality reflection [等式反射性质] (proof on next page – omitted)
  - In the empty context, a = b : A is correct if and only if a $=_A$ b is provable, where = is definitional and $=_A$ is propositional (Leibniz/Id).

Example proof: equality reflection (in empty ctxt) [*]

- Theorem. $|\!-\ a=b : A$ iff $|\!-\ p : a =_A b$ for some p.
- Proof. Necessity is straightforward. Sufficiency:
  - Let $p : a=_A b$ where be in normal form (by SN & SR).
  - By SN/SR, we may assume that p/a/b/A are in normal form.
  - Note that $a=_A b = \forall P{:}A\to Prop.P(a)\to P(b)$, so
    $$p \equiv \lambda P{:}A\to Prop.\ \lambda x{:}P(a).\ M$$
  - Then, (by analysis) $M \equiv x : P(a)$ and hence $|\!-\ P(a) = P(b)$.
  - So, by CR, $|\!-\ a = b : A$.

3. Features and adequacy in applications

❖ Type theory as a foundational language of …

	Univalent Math	**NL semantics**
Extensionality/univalence	√	X
Proof Irrelevance	X	√
Higher Inductive Types	√	??
Subtyping	??	√

Example features

- Extensionality
  - Propositional extensionality (eg, Church 1940): $P \Leftrightarrow Q \rightarrow Id(P,Q)$
  - Functional extensionality: $\forall x.Id(f(x),g(x)) \rightarrow Id(f,g)$
  - Univalence (Voevodsky 2009): $Id(A,B) \cong (A \cong B)$
- [*] Proof irrelevance
  - $p, q : P \rightarrow p = q$, for any proposition P.
  - Only possible if there's distinction between props and other types
- [*] Subtyping
  - $A \leq B$: every object of A can be regarded as an object of B.
  - Subsumptive (inadequate) ➔ coercive subtyping (Luo et al 2012)
- … …

4. Two notions of equality

- Definitional equality (a = b : A)
  - 3+4 = 7 : Nat
- Propositional equality ($=_A$ -- "Leibniz"/Id)
  - $\forall$x,y:Nat. ($x+y =_{Nat} y+x$)
- Why definitional equality?
  - Capturing computation (eg, for nats)
  - Dependent typing: ($x \leq 3+4$) = ($x \leq 7$) have the same objects.
- Equality reflection [等式反射性质] – usually:
  - Theorem. $\Diamond \vdash a = b : A \;\Leftrightarrow\; \Diamond \vdash p : (a =_A b)$ for some p.
  - See (Martin-Löf 73) for Id and (Martin-Löf 71/Luo 90) for Leibniz.

Informally, definitional & propositional equalities "coincide".

“Sameness” in type theories

	Prog verification	NL semantics	Univalent Math
Equality	$=$ and $=_A$	$=$ and $=_A$	$\cong$ (and equivalent Id)

- “Sameness”:
  - Usually given by definitional/propositional equality $=/=_A$.
    - E.g., MTT-semantics for natural language; programming/verification.
  - Different in some theories.
    - [Part III] In univalent foundations, type isomorphism $\cong$ is made equivalent to Id that becomes different from definitional equality $=$.

Part III

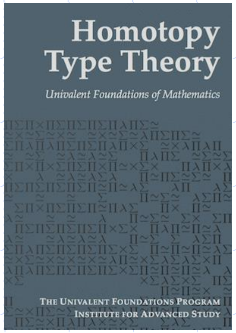
Univalent foundations of mathematics

[数学的单价基础]

Univalent Foundations – alternative to set theory

- Vladimir Voevodsky (1966–2017)
  - Russian mathematician; Fields medalist (2002);
  - Worked on UF since 2005 (homotopy lambda calculus), and developed UF library in Coq from 2010.
    - V. Voevodsky. An experimental lib of formalized math based on UF. MSCS, 2015.
- Voevodsky's key motivations and ideas
  - Proof-checking – we need foundations that make it possible.
    - Errors in his own papers, only discovered/confirmed 15/20 yrs later …
  - Groupoid [群胚] conception for higher dimensional math.
    - Groupoids, rather than categories, are "sets in the next dimension".
  - H-levels (homotopy levels of n-types) [Voevodsky 2010]
    - Propositions, sets, groupoids, … (e.g., sets as types of h-level 2)
    - Voevodsky: UF "is the first adequate formalization of set theory" (2014)

Homotopy type theory (HoTT 2013)



- Development of HoTT
  - Formalisation of univalent foundations
  - Special year on univalent foundations of math.
    - 2012-13 at Inst of Advanced Study, Princeton, USA.
- HoTT = MLTT + UA + HITs
  - UA – univalence axiom [单价公理]
    - Univalence may be understood as generalised extensionality.
  - HITs – higher inductive types [高等归纳类型]
    - Extensional concepts such as quotients as types (omitted in this talk)

Univalence (Voevodsky 2006~2009)

- Univalence axiom (≅/Id for equivalence/identity of types):

  $$(\mathrm{UA}) \qquad \mathrm{Id}(A,B) \cong (A \cong B)$$

  - Informally: isomorphic/equivalent types are equal.
  - Mathematical structuralism (invariance under equivalence)
  - Theorem (Voevodsky). MLTT with (UA) is consistent.
- UA is "unusual"
  - E.g., $A\mathrm{x}B \cong B\mathrm{x}A$ – they are isomorphic (have the same cardinality).
  - Justification: equivalent types have same "internal properties".
- UA implies extensionality (functional & propositional)
  - Note: Mathematics is extensional! (Other fields may not be.)
- Comparison with set theory (again):
  - Extensionality: type theory is intensional & set theory extensional.
  - Univalence: such "structuralism" is absent in set theory.

Cubical type theory [*]

- UA as an <u>axiom</u> (as in HoTT) – problematic!
  - Some "natural numbers" don't compute to canonical ones …
  - Correctness/adequacy of the foundational language is in doubt …!
- Cubical type theory [立方类型论]
  - Started in 2012-13 at Princeton, by Coquand (TYPES15, LICS18), when Voevodsky had the conjecture: canonicity holds.
- Univalence is a <u>theorem</u> in the cubical type theory.
  - Canonicity for nats holds – a big step forward!
  - Normalisation and decidability? (to be proved)

Q: Is the cubical type theory <u>the</u> correct solution?

Analysis & comments from one angle [*]

- First, a comment on current philosophical analysis
  - Some (most?) are superficial (not as deep, at least), except:
    - Centrone et. al (eds.) Reflections on the Foundations of Mathematics: Univalent Foundations, Set Theory and General Thoughts. Springer 2019.
- Maddy's analysis on "foundational roles" (2019, in book above)
  - Compared with set theory, category theory & univalent foundations only "add" a new foundational role/goal, resp.
  - Category theory (CT) by "essential guidance"
  - Univalent foundations (UF) by "proof checking"
- Concerning univalent foundations, this may have overlooked:
  - Mathematical structuralism
  - Foundations as "practical tool" for working mathematician

Research monograph on MTTs in Chinese

罗朝晖：现代类型论的发展与应用。
清华大学出版社，2024年。

Z. Luo. Modern Type Theories: Their Development and Applications. Tsinghua Univ Press, 2024. (In Chinese)

网址：http://www.tup.tsinghua.edu.cn/booksCenter/book_09109701.html