Modern Type Theories and Their Applications

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

This talk – two parts

I. Modern Type Theories: brief introduction

- Basics of MTTs
- Meta-theory and meaning theory
- Application in proof assistants based on MTTs

II. Two applications of MTTs:

- Univalent foundations & homotopy type theory
- Formal semantics in MTTs (MTT-semantics)

(*) Note: two forms of type systems

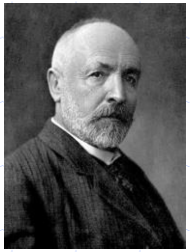
- Type systems (for programming languages)
  - Type systems (Milner, …) in ML, Haskell, …
  - General recursion, polymorphism, modules, …
  - No consistent logic under propositions-as-types principle
- Type theories (in proof assistants)
  - Type theories (Martin-Löf, …) in Agda, Coq, Lego, …
  - Dependent/inductive/logical/… types
  - Consistent logic under propositions-as-types principle

This talk is about the 2nd, with occasional comparisons.

Part I. Modern Type Theories

Origin of type theory

- Foundations of mathematics and paradoxes
  - Naïve set theory (Cantor, …)
  - Paradox in naïve set theory (Russell 1903) [next slide]
  - 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”



Russell's paradox in naïve set theory

- Naïve concept of set with unrestricted comprehension:

  $\{ x \mid P(x) \}$ for any predicate P in FOL
- Russell's paradoxical set would exist if we accepted this:

  $R = \{ x \mid x \notin x \}$
- Then, by definition, we would have an absurd equivalence:

  $R \in R \Leftrightarrow R \notin R \qquad (*)$
- BTW, R exists ➔ (*) ➔ logical inconsistency.

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

- Martin-Löf has introduced/employed
  - Judgements, contexts, definitional equality
  - Dependent/inductive types, type universes
  - Curry-Howard principle of propositions-as-types
- Examples of MTTs [& implementing proof assistants]:
  - Predicative TTs:
    - MLTT – Martin-Löf's type theory [1975]; Agda
  - Impredicative TTs:
    - CC [Coquand & Huet 1988] and pCIC; Coq/Lean
    - UTT [Luo 1990, 1994]; Lego/Plastic

UTT – an example MTT

- UTT – Unifying theory of Dependent Types (MLTT + CC) [Luo 1994, Oxford Univ Press]

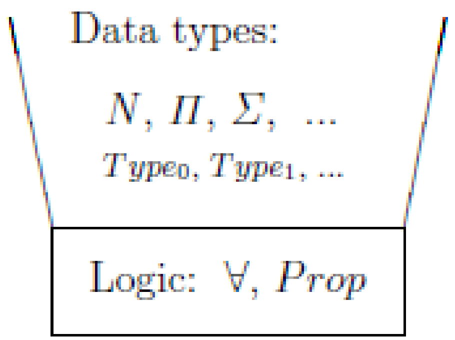
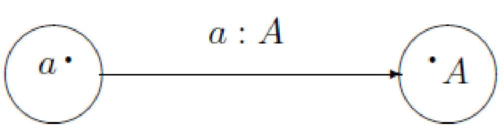


Fig. 1. The type structure in UTT.

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

Judgements – basic notion in type theory

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



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

Π-types and $\forall$-props: examples of dependent types

- $\Pi x{:}A.B(x)$ – dependent function type
  - Type for collection

    $\{ f \in A \to \bigcup_{a \in A} B(a) \mid \forall a \in A.\ f(a) \in B(a) \}$
  - $f : \Pi x{:}Human.Parent(x)$

    ➔ f(h) is father/mother of h (not others!)

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

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

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

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

- Universal quantification

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

  - Prop, the collection of propositions, is a type itself
    [impredicative universe with "circular" props like $\forall X{:}Prop.X$]
  - Propositions-as-types: propositions are (some) types
    [So, logic(s) is only a part of type theory – see next slide.]

Relationship between logic and set/type theory

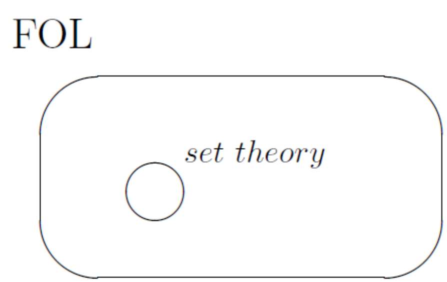


Figure 1: Set theory – a theory in first-order logic

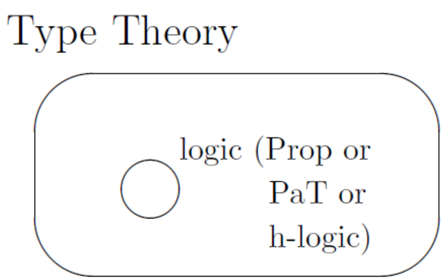


Figure 2: Logic is a part of type theory

Inductive types: an example

- 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

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

- Elimination rule

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

- Notes:
  - Introduction rules specify canonical objects.
  - Elimination rule is Nat-induction + primitive recursion.
  - All Peano axioms are either rules or provable.

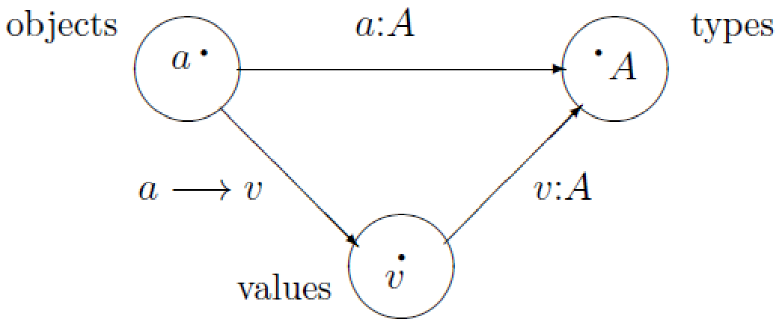
(*) Two notions of typing (c.f. [Luo et al. 2012])

- Type assignment
  - Types and objects exist independently.
  - Types are assigned to objects.
  - Therefore, it may be natural to have type polymorphism.
- Types with canonical objects
  - Types and objects <u>do not</u> exist independently.
  - Types consist of canonical objects and the existence of these objects depends on that of the type (by introduction rules). [Canonical nats 0 and succ(n) don't exist if Nat does not.]
  - Therefore, it is not unnatural to have type uniqueness.

This also leads to two different views of subtyping (see later).

Meaning explanation

- Understanding based on computation:



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

- How to guarantee that computation a→v terminates !?
  - Meta-theoretic study (eg, strong normalisation of UTT)
  - Meaning-theoretic argument (harmony of intro/elim rules)

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?

Meta-theoretic theorems: examples

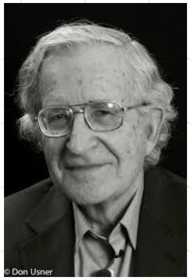
- Church-Rosser theorem (CR)
  - If a=b : A, then there exists c : A s.t. a $\to$ c and b $\to$ c.
- Subject Reduction (SR)
  - If a : A and a $\to$ b, then b : A.
- Strong Normalisation (SN)
  - Every computation from a well-typed term terminates.
- Logical consistency (in UTT)
  - $\forall$X:Prop.X (false) is not provable (in the empty context).
- Decidability (of type-checking)
  - It is decidable whether a judgement is correct (derivable).

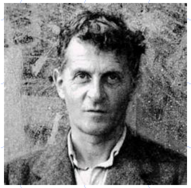
Example proof: logical consistency

- Proof (of consistency)
  - Assume that $M : \forall X{:}Prop.X$.
  - By SN & SR, we may assume that M is in normal form.
  - So, $M \equiv \lambda X{:}Prop.M'$ s.t. $X{:}Prop \vdash M' : X$ for some $M' \equiv M_1 \ldots M_n X$ (or other forms of "base term").
  - But we can then show that this would imply either Prop = X or Prop = Qx:A.B which, by CR, is impossible.
  - Therefore, M does not exist ($\forall X{:}Prop.X$ is not provable).

Theories of meaning

- Meaning is reference ("referential theory")
  - Word meanings are (abstract/concrete) objects.
  - c.f., platonism: Frege, …
- Meaning is concept ("internalist theory")
  - Word meanings are ideas in the mind.
  - c.f., Aristotle, Chomsky, …
- Meaning is use ("use theory")
  - Word meanings are understood by its uses.
  - c.f., Wittgenstein, …





Proof-theoretic semantics – use theory for logics


- Proof-theoretic semantics
  - Use theory for logical systems
  - Dummett, Prawitz, …
- Ideas
  - Pre-mathematical justification of logical rules (informally from "first principles", not meta-theoretically)
  - For logic: two aspects of use – verification and consequence
  - Harmony: intro/elim rules should be harmonious.
- Proof-theoretic semantics for type theories
  - Martin-Löf's meaning explanations (1984)
  - Type theory potentially has PTS, while set theory does not.
  - Current investigations: hypothetical judgements, impredicativity, …

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

  - Computer Science:
    - program verification and advanced programming
  - Computational Linguistics
    - NL reasoning based on MTT-semantics (Coq)

Part II. Two applications of MTTs

- Univalent foundations & homotopy type theory
- Formal semantics in MTTs (MTT-semantics)

Part II(1)

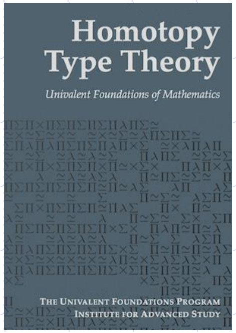
Univalent foundations of mathematics

Univalent Foundations – alternative to set theory

- Vladimir Voevodsky (1966–2017)
  - Russian mathematician; Fields medalist (2002);
    Professor at Inst of Advanced Study, Princeton, USA
  - Worked on UF since 2005 (homotopy lambda calculus), 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 conception for higher dimensional math.
    - Groupoids, rather than categories, are "sets in the next dimension".
  - H-levels (homotopy levels of n-types) [Voevodsky 2009]
    - Propositions, sets, groupoids, …

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
  - HITs – higher inductive types



Univalence

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

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

  - Mathematical structuralism (invariance under equivalence)
    - UA is "unusual" ($A \times B \cong B \times A$ – they have same <u>expressible</u> properties.)
  - UA implies extensionality, both functional and propositional.
    - Note: Mathematics is extensional!
    - HoTT v.s. Extensional TT [Martin-Löf 1984] (ETT is problematic)
- UA as an <u>axiom</u> (in HoTT)?
  - "Axioms" are problematic in type theory!
  - With axioms, canonicity fails to hold.
    - Some "natural numbers" don't compute to canonical ones …
    - Correctness/adequacy of the foundational language is in doubt …!

Cubical type theory (Coquand et al, TYPES15, LICS18, …)

- Cubical type theory
  - Research started in 2012-13 at Princeton, by Coquand et al, 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)
- Experimental implementation in Agda-Cubical

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

Higher inductive types

- Basic idea of HITs:
  - Ordinary induction is only about "points" (eg, 0 & succ(n)).
  - Higher induction extends it to "equalities/paths".
- Quotient types "A/R" – typical example (with ad hoc notation =)

$$|\_| \;:\; A \to A/R$$

$$\forall x, y : A.\;\; R(x, y) \to |x| = |y|$$

  - Quotient types were problematic ("setoid hell") – so real progress!
  - Current implementation (eg, Agda-cubical) still a bit cumbersome.
- Notes: Several research topics, including:
  - General schemata for HITs (still unknown)
  - Independent understanding of HITs

Direct v.s. indirect formalisations (side remark)

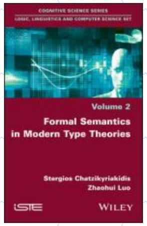
- Type theory is more effective (much more) when built-in entities are used <u>directly</u>.
- Application examples:
  - Formalisation of mathematics
    - HoTT-based proof development (e.g., HITs for quotients) [HoTT 2013]
    - In contrast with, e.g., setoids and related proofs (cumbersome …)
  - Program verification
    - Built-in functions as FP programs (and their verification)
    - In contrast with, e.g., "deep embedding + semantics" (cumbersome …)
  - Linguistic semantics
    - CNs-as-types in MTT-semantics (see below)
    - In contrast with, e.g., CNs-as-predicates in Montague semantics.

Part II(2). MTT-semantics

Type-Theoretical Semantics

- Montague semantics (Montague 1930–1971)
  - MG: formal natural language semantics in set theory
  - Dominating in linguistic semantics since 1970s
  - Set-theoretic, using simple type theory as intermediate

- MTT-semantics: formal semantics in modern type theories
  - Ranta (1994): formal semantics in Martin-Löf's type theory
  - Recent study on MTT-sem ➔ full-scale alternative to MG
    - Z. Luo. Formal Sem. in MTTs with Coercive Subtyping. L&P 35(6). 2012.
    - S. Chatzikyriakidis and Z. Luo. Formal Semantics in MTTs. Wiley, 2020. (monograph on MTT-semantics)



  - Research context on rich typing in NL (many researchers …)
    - S. Chatzikyriakidis and Z. Luo (eds.) Modern Perspectives in Type Theoretical Semantics. Springer, 2017.

MTT-semantics: basic categories

Category	Semantic Type
S	Prop (the type of all propositions)
CNs (book, human, …)	types (each common noun is interpreted as a type)
IV	$A \rightarrow Prop$ (A is the "meaningful domain" of a verb)
Adj	$A \rightarrow Prop$ (A is the "meaningful domain" of an adjective)
Adv	$\Pi A{:}CN.(A \rightarrow Prop) \rightarrow (A \rightarrow Prop)$ (polymorphic on CNs)

Simple example: [John talks] = talk(j) : Prop
where j : Human and talk : Human → Prop.

(*) In MTT-semantics, common nouns (CNs) are types rather than predicates as in Montague semantics.

Modelling Adjectival Modification: Case Study

Classical classification	Example	Characterisation	MTT-semantics
intersective	handsome man	Adj(N) ➔ N & Adj	$\Sigma x{:}Man.handsome(x)$
subsective	large mouse	Adj(N) ➔ N (Adj depends on N)	large : $\Pi A{:}CN.\ A \rightarrow Prop$ large(mouse) : $Mouse \rightarrow Prop$
privative	fake gun	Adj(N) ➔ ¬N	$G = G_R + G_F$ with $G_R \leq_{inl} G$, $G_F \leq_{inr} G$
non-committal	alleged criminal	Adj(N) ➔ nothing	$H_{h,Adj} : Prop \rightarrow Prop$

- [Chatzikyriakidis & Luo 13, 17 & 20; Luo, Shi & Xue 22]

Note on Subtyping in MTT-semantics

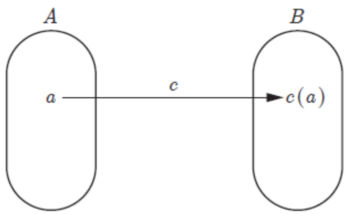
- Simple example
  - A human talks. Paul is a handsome man. Does Paul talk?
  - Semantically, can we type talk(p)?
    - talk : Human→Prop and p : [handsome man]
  - Yes, because p : [handsome man] ≤ Man ≤ Human
- Subtyping is crucial for MTT-semantics
  - Coercive subtyping [Luo 1999, Luo, Soloviev & Xue 2012] is adequate for MTTs and we use it in MTT-semantics.
  - Note: Traditional subsumptive subtyping is inadequate for MTTs
    - Canonicity fails with subsumptive subtyping.

(*) Two notions of subtyping

- Corresponding to type assignment & canonical objects
- Subsumptive subtyping (A ≤ B)
  - "Simple" & widely accepted (esp for PLs)
  - Unfortunately, not suitable for MTTs (eg, canonicity fails)

$$\frac{a : A \quad A \leqslant B}{a : B}$$

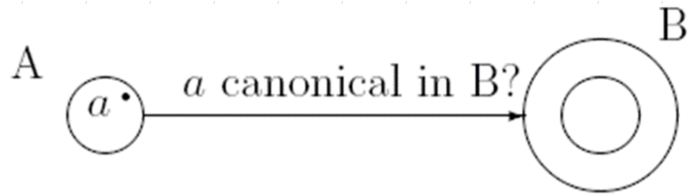
- Coercive subtyping ($A \leq_c B$) [Luo 97, Luo, Soloviev & Xue 12]
  - Subtyping as abbreviations (implicit coercions)
  - General & flexible (eg, projective subtyping)
  - Conservativity over original MTT (properties preserved)



$$\frac{\Gamma, y : B \vdash f(y) : C(y) \quad \Gamma \vdash a : A \quad \Gamma \vdash A \leqslant_c B}{\Gamma \vdash f(a) = f(c(a)) : C(c(a))}$$

(*) Canonicity fails in subsumptive subtyping

- Q: If A≤B and a is canonical in A, is it canonical in B?



- No, if A≤B is subsumptive.
- Examples:

$$\frac{A \leq B}{List(A) \leq List(B)}$$

  - nil(A) : List(B), but nil(A) ≠ nil(B) or cons(B,b,l).

$$\Sigma x : Nat.P(x) \leq Nat$$

  - (0,p) : Nat, but (0,p) ≠ $succ^n(0)$.

Advanced features in MTT-semantics: examples

- Copredication and dot-types [Luo 09, XL 12, CL 18]
- Linguistic coercions via coercive subtyping [Asher & Luo 12]
- Signatures for linguistic contexts [Luo 14, Lungu & Luo 16]
- MTT event sem. (dependent event types) [Luo & Soloviev 17]
- Propositional forms of judgemental inter. [Xue et al 18, 23)]
- **MTT-semantics in $\mathbf{MLTT_h}$ [Luo (LACompLing 2018)]** **(*)**
- CNs as setoids [Luo 12, CL 18] (and CNs as HITs – in progress)
- Dependent categorial grammar [Luo 24]

(*) MTT-semantics in a predicative type theory? – next two slides.

MTT-semantics in Martin-Löf's TT – a problem

- Martin-Löf's type theory in formal semantics
  - Munnick, Sundholm, Ranta & many others
  - All use PaT logic – propositions as types.
  - But Martin-Löf goes one step further: types = propositions!
  - This is where the problem arises [Luo (LACL 2012)].
- Example: a handsome man is (m,p) : Σx:Man.handsome(x)
  - Two handsome men are the same iff they are the same man (and how to prove they are handsome should be irrelevant!)
  - Proof irrelevance (any two proofs of the same proposition are the same.)
  - But, in MLTT with PaT logic, this would mean every type collapses! Absurd.
- So, MLTT with PaT logic is <u>inadequate</u> for MTT-semantics.
  - Developing MTT-semantics in UTT is OK where proof irrelevance is possible.

$MLTT_h$: Extension of MLTT with H-logic

- H-logic ("H" for h-levels due to Voevodsky)
  - A proposition is a type with at most one object.
  - Logical operators (examples):
    - $P \supset Q = P \rightarrow Q$ and $\forall x{:}A.P = \prod x{:}A.P$
    - $P \vee Q = |P+Q|$ and $\exists x{:}A.P = |\Sigma x{:}A.P|$

    where $|A|$ is <u>propositional truncation</u> (a form of HITs).
- $MLTT_h$ = MLTT + h-logic (subsystem of HoTT) [Luo 2019]
  - Proof irrelevance is "built-in" in h-logic (by definition).
  - Note: $MLTT_h$ is a <u>proper extension</u> of MLTT.
  - Claim: $MLTT_h$ is adequate for MTT-semantics.

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