

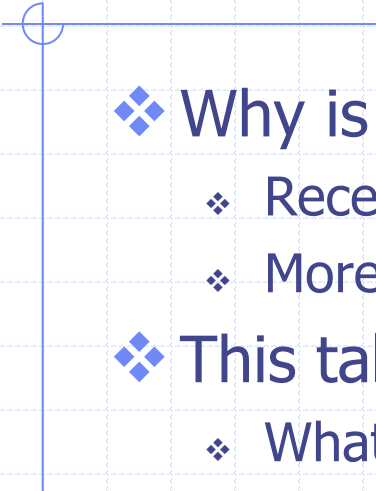
Type Theory: New Foundations and AI-Assisted Proofs

Zhaohui Luo [罗朝晖]

Royal Holloway, Univ of London

Zhaohui.Luo@hotmail.co.uk

<https://www.cs.rhul.ac.uk/home/zhaohui/>



❖ Why is type theory important?

- ❖ Recent: AI in proof development – crucial in verification.
- ❖ More generally, it offers new foundations (c.f., set theory).

❖ This talk – three parts

- ❖ What is (modern) type theory?
- ❖ Type-theoretic foundations: mathematics and semantics.
- ❖ Proof assistants and how to trust AI-generated proofs.

What is type theory?

❖ Dependent (modern) type theory

- ❖ Types: dependent/inductive/propositional/...
- ❖ Mechanisms: context/judgement/defn eq/...



❖ Relationship between logic and set/type theory

Logic (e.g., FOL)

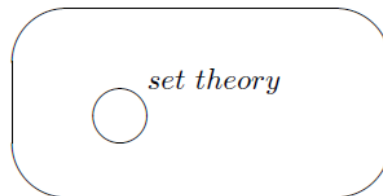


Figure 1: Set theory – a theory in FOL

Type Theory

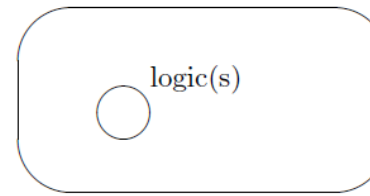


Figure 2: Logic is a part of type theory

An example type theory

❖ Unifying Theory of dependent Types (UTT) [L 1994]

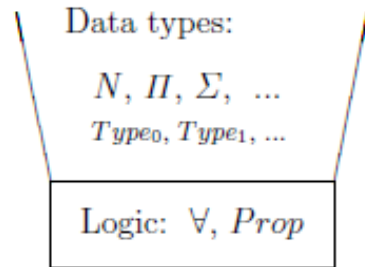


Fig. 1. The type structure in UTT.

For example

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

❖ Key features

- ❖ Computation + inference (natural deduction, rules)
- ❖ "Manageable set theory" (cf, decidable type-checking)
- ❖ Meaning theory and metatheory (eg, use theory + proof assistants)

Foundations and Applications

- ❖ Type-theoretic foundations for
 - ❖ Mathematics
 - ❖ Natural language semantics
 - ❖ Program verification
 - ❖ Automated proof development on computers
- ❖ Research monographs on type theory
 - ❖ 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. 2026.
(Updated English edition in Sept 26, in press)

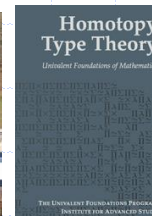


Most views in this talk can be found in the above English edition.

Type theory as new foundations of mathematics

❖ Univalent foundations of mathematics

- ❖ Vladimir Voevodsky (Fields medallist 1966-2017)
- ❖ Special year 2012-13 at IAS on univalent math.



❖ Voevodsky's key motivations:

- ❖ Automated proof-checking (c.f., errors in VV's own papers ...)
- ❖ Groupoids (not categories!) as "sets in a higher dimension"
- ❖ Propositions/sets/groupoids as types (set theory is only a part.)

❖ Univalence

- ❖ Equivalent ("isomorphic") types are equal.
- ❖ Mathematical structuralism (absent in set theory)
- ❖ OK in type theory: unusual but can be made internally valid.

❖ Successful and promising (for both classical/constructive maths)

Type-theoretic foundation of NL semantics

- ❖ Montague's sem in set theory (simple TT as intermediate)
- ❖ MTT-semantics in modern type theories
 - ❖ Ranta 1994, L 2012, Chatzikyriakidis & L 2020, L 2024/26.
 - ❖ MTTs "comparable" to set theory, but proof-theoretic.
 - ❖ Many advantages of MTT-sem:
 - ❖ eg, selection restriction by $a:A$ (not $s \in S$)
- ❖ Proof-theoretic meaning theory – "extra plus" for MTT-sem
 - ❖ Use theory of meaning: Dummett, Martin-Löf, ...
 - ❖ MTT-sem is both model-/proof-theoretic [L 14/26, 罗&石 26]

Advanced topics in MTT-semantics: examples

- ❖ Copredication and dot-types [L 09, Xue & L 12, Chatzikyriakidis & L 18]
- ❖ Linguistic coercions via coercive subtyping [Asher & L 12]
- ❖ Signatures for linguistic contexts [L 14, Lungu & L 16]
- ❖ MTT event semantics (dependent event types) [L & Soloviev 17]
- ❖ Propositional forms of judgemental inter. [Xue, L & Chatzikyriakidis 18, 23)]
- ❖ MTT-semantics in $MLTT_h$ [L 18,26]
- ❖ Subtype universes [Maclea & L 20 , 罗24, Bradley & L 26]
- ❖ Dependent categorial grammar [罗24]
- ❖ CNs as setoids [L 12, Chatzikyriakidis & L 18] (plus CNs as HITs [L 24])
- ❖ Type-theoretic analysis of event semantics [L & Shi 25]

Adequacy in applications of type theory

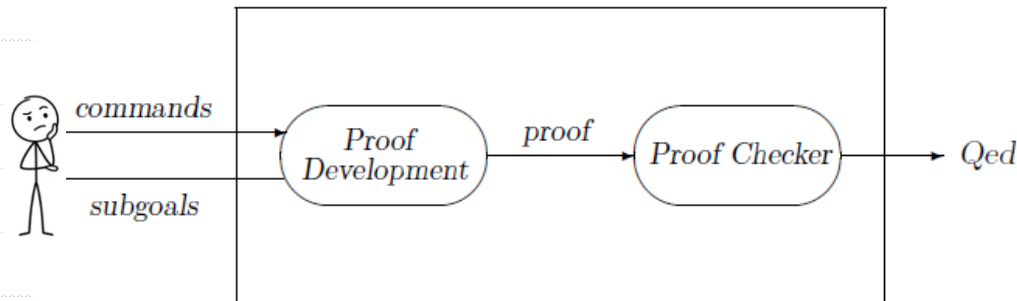
	Univalent Math	NL semantics	Program Verification
<i>Univalence</i>	✓	X	X
<i>Subtyping</i>	OK	✓	✓
<i>Proof Irrelevance</i>	X	✓	OK
<i>Higher Ind Types</i>	✓	??	??

✓ – necessary X – unsuitable OK – OK (but unnecessary) ?? – unknown

Proof assistants based on type theory



- ❖ AUTOMATH (1st checker in late 60s) by de Bruijn
- ❖ Proof assistants from 80-90s onwards:
 - ❖ Agda, Coq/Rocq, HOL, Isabelle, **Lean**, Lego, NuPRL, ...



- ❖ Small example:

```
Theorem inf_many_primes :  
  not (exists n:nat, forall x:nat, prime x -> x < n)
```
- ❖ Tedious and taking long time: eg, 12 years for Kepler's conjecture!
- ❖ Fermat's Last Theorem: Buzzard's funded project from 2014 ...

Trust a machine checked proof?

- ❖ Can we? Yes, if you trust the proof checker.
 - ❖ In type theory, proof checking (type checking) is decidable.
- ❖ The proof checker can be trusted (Pollack 1997):
 - ❖ Small checking algorithm
 - ❖ Independent checking
- ❖ Caveats:
 - ❖ Knowledge on type theory
 - ❖ Or trust experts ...
 - ❖ And trust the environment!

Syntax of terms in BNF:

$M ::= x \mid \Pi x:M.M \mid \lambda x:M.M \mid M(M) \mid \text{Nat} \mid 0 \mid \text{succ}(M) \mid \mathcal{E}_{\text{Nat}}(M, M, M) \mid \dots$

Algorithm $\mathcal{C}(\Gamma; _)$ (in each case, $\mathcal{C}(\Gamma; _)\uparrow$ otherwise):

Context legitimacy:

- $\mathcal{C}(\langle \rangle; \bullet)\downarrow$.
- $\mathcal{C}(\Gamma, x:A; \bullet)\downarrow$ if $\mathcal{C}(\Gamma; \bullet)\downarrow$ and $\mathcal{C}(\Gamma; A)\downarrow$.

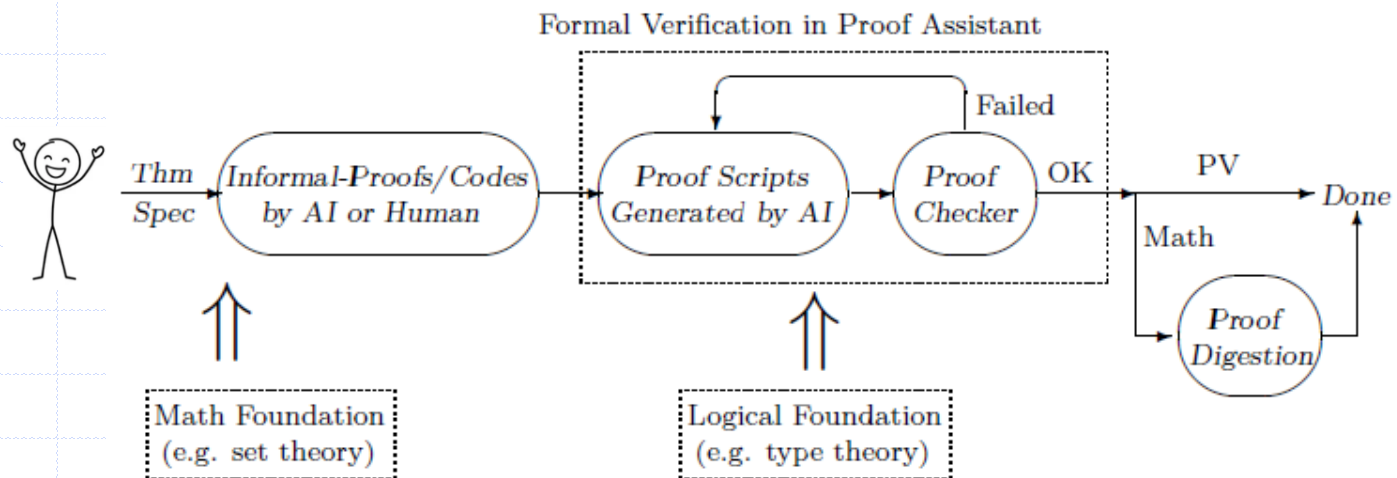
Term legitimacy, where $\mathcal{C}(\Gamma; \bullet)\downarrow$:

- $M \equiv x$, then $\mathcal{C}(\Gamma; M) = A_i$ if $x \equiv x_i$ for some $x_i:A_i \in \Gamma$.
- $M \equiv \Pi x:A.B$, then $\mathcal{C}(\Gamma; M)\downarrow$ if $\mathcal{C}(\Gamma; A)\downarrow$ and $\mathcal{C}(\Gamma, x:A; B)\downarrow$.
- $M \equiv \lambda x:A.b$, then $\mathcal{C}(\Gamma; M) = \Pi x:A.B$ if $\mathcal{C}(\Gamma; A)\downarrow$ and $\mathcal{C}(\Gamma, x:A; b) = B$.
- $M \equiv f(a)$, then $\mathcal{C}(\Gamma; M) = [a/x]B$ if $\text{nf}(\mathcal{C}(\Gamma; f)) \equiv \Pi x:A.B$, $\text{nf}(\mathcal{C}(\Gamma; a)) \equiv A$.
- ... (for other type constructors such as *Nat*)

Figure 5.5: Proof-checking algorithm $\mathcal{C}$: pseudo code.

Proof development with Artificial Intelligence

❖ Current process of proof development with AI:*



❖ Eg, Fermat's Last Theorem (cf, Buzzard's ongoing project):

- ❖ AI did it in 11 days (!), with 13m lines of codes, proving 29,500 lemmas.
- ❖ Note: this is autoformalisation, with given formal theorem/environment!
- ❖ Very different from other claims with AI doing autoformalisation (next slide).

*Thanks to Xue Tao for the figure.

Trust an AI-generated & machine checked proof? *

❖ Can we? Not really, because:

- ❖ Autoformalisation of theorems is not guaranteed correct/faithful !
- ❖ Significant gaps: informal $\rightarrow ? \rightarrow$ formal / set-theory $\rightarrow ? \rightarrow$ type theory

❖ Two simple examples to illustrate this:

- ❖ My 1991 paper: $s=s$ in Lego (wrong/unfaithful – should be $s=s'$!)
- ❖ Escardo [2017] (L 26, Appendix D.2, defn of “image” by Σ):
 - ❖ Incorrect formalisation in HoTT using PaT in Martin-Lof’s type theory.
 - ❖ Though wrong, OK in most cases: misleading AI training $\rightarrow$ incorrectness.

❖ Recent [BCH26] supports the above view:

- ❖ “Navier-Stokes Lost in Translation” (arXiv:2610.08144v1, 6 Oct 26)

❖ Conclusion: From informal to formal, human checking is nec.

- ❖ Note: different for FLT, whose formal statement was already human-checked.

* cf, My book (English edition) in press.

Disclaimer and clarification

❖ I'm not saying:

“AI (informal/NL) proofs are incorrect.”
(They need be further checked.)

❖ I'm only saying:

The claims by AI companies that the existence of formal proofs verifies the informal ones are not sound, because the formal statements are not checked to be faithful with the informal theorems. So, such claims of verification are problematic.

