Two-Level Type Theory

Nicolai Kraus

12 March 2025,
9th Southern and Midlands Logic Seminar,
Birmingham

Field: MLTT-style type theories

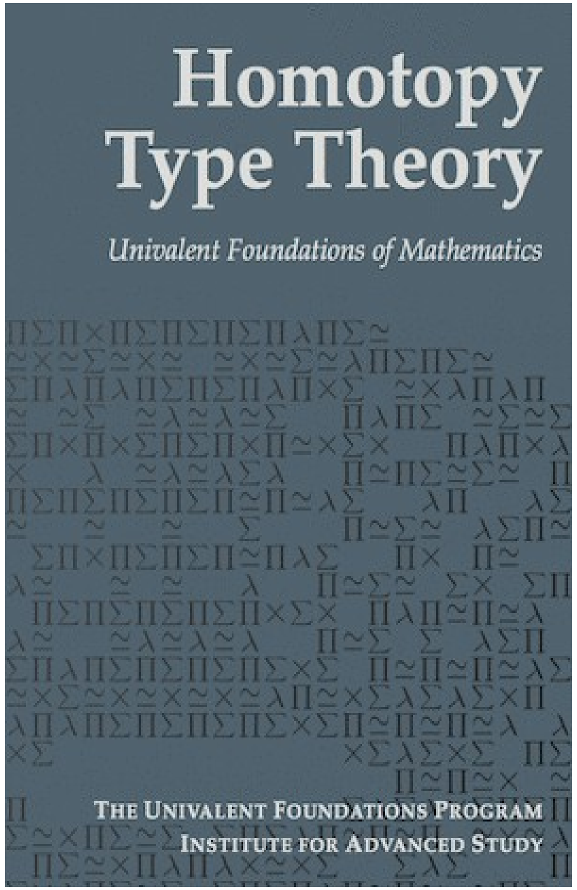
Idris

Agda

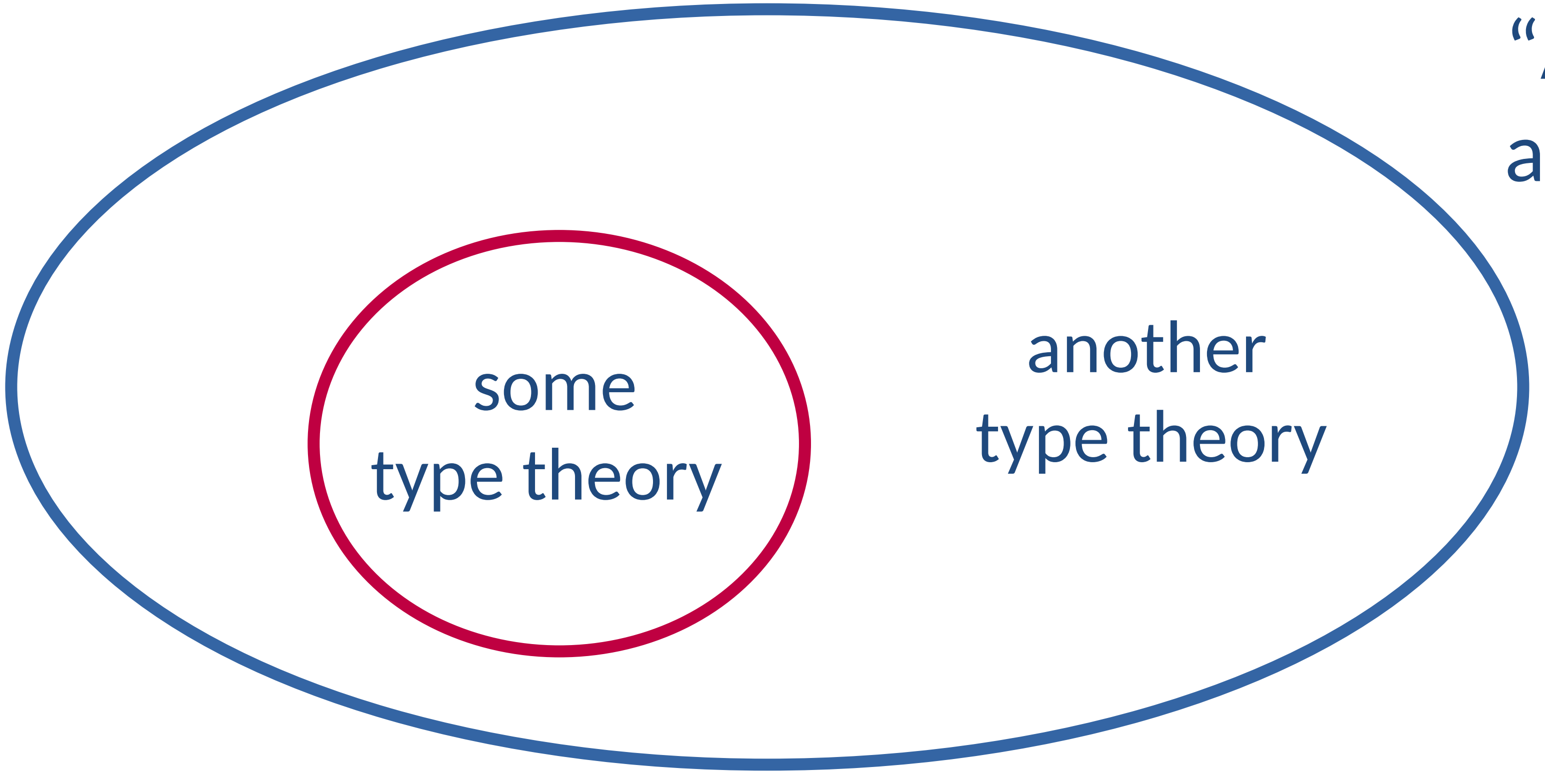
Coq/Rocq

LEAN THEOREM PROVER

Arend

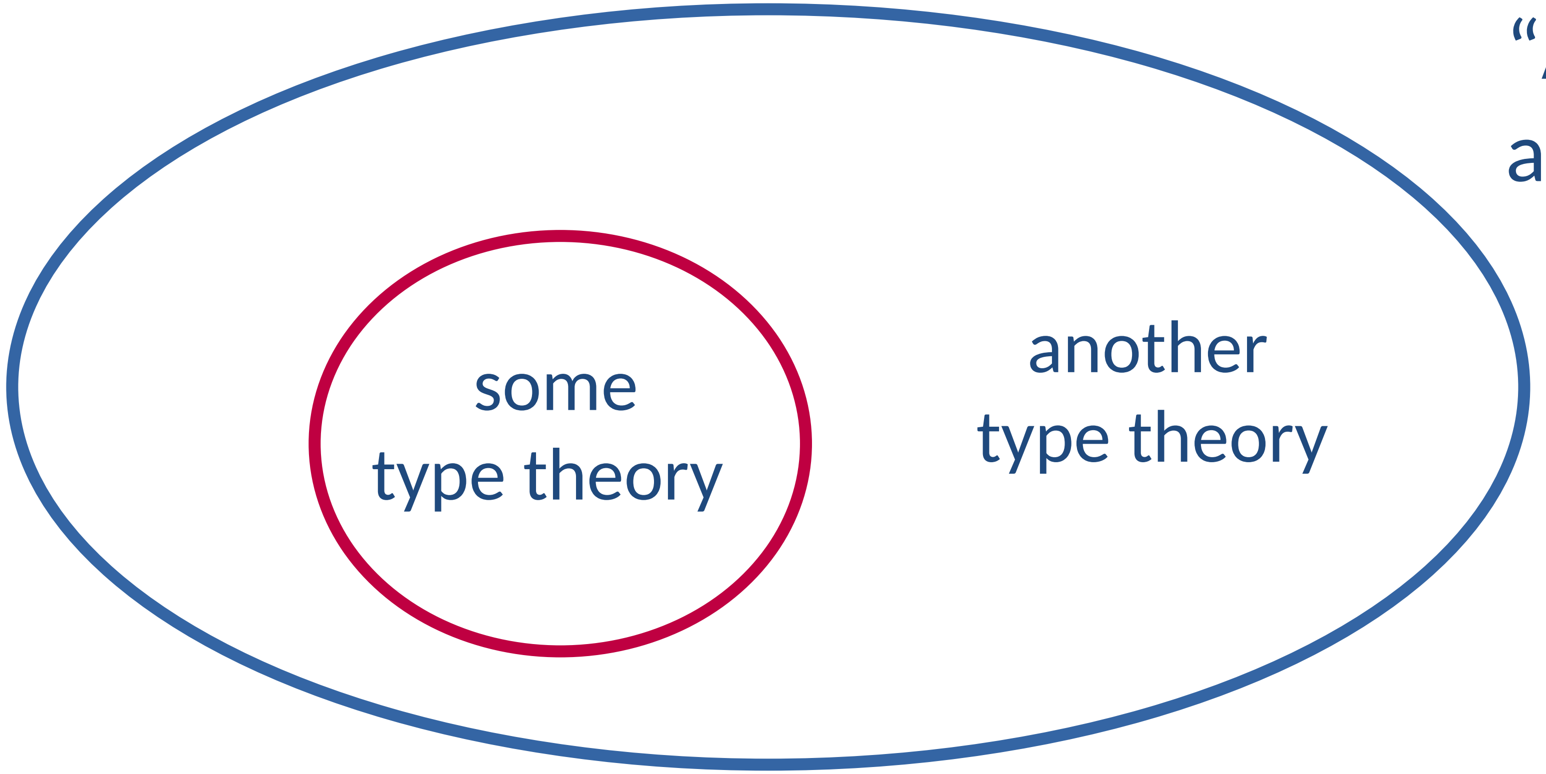


Two-Level Type Theory (2LTT)



"A type theory sitting in another type theory"

Two-Level Type Theory (2LTT)



"A type theory sitting in another type theory"

… ok, but why?

Early instance of 2LTT:

Voevodsky's HTS (Homotopy Type System), 2013

what: HoTT, with the ability to reason about judgmental equalities

why: We want an internal theory of higher categories, via semisimplicial types

Early instance of 2LTT:
Voevodsky's HTS (Homotopy Type System), 2013

what: HoTT, with the ability to reason about judgmental equalities

why: We want an internal theory of higher categories, via semisimplicial types

...ok, better explanation:

something doesn't type-check, but we want it to type-check!

```
refl : 2 + 1 = 1 + 2

refl : 5 + 1 = 1 + 5

refl : 843 + 1 = 1 + 843
```

Early instance of 2LTT:

Voevodsky's HTS (Homotopy Type System), 2013

what: HoTT, with the ability to reason about judgmental equalities

why: We want an internal theory of higher categories, via semisimplicial types

...ok, better explanation:
something doesn't type-check, but we want it to type-check!

```
refl : 2 + 1 = 1 + 2
refl : 5 + 1 = 1 + 5
refl : 843 + 1 = 1 + 843
```

~~refl~~ `: {n : Nat} → n + 1 = 1 + n`

Early instance of 2LTT:

Voevodsky's HTS (Homotopy Type System), 2013

what: HoTT, with the ability to reason about judgmental equalities

why: We want an internal theory of higher categories, via semisimplicial types

...ok, better explanation:

something doesn't type-check, but we want it to type-check!

```
refl : 2 + 1 = 1 + 2

refl : 5 + 1 = 1 + 5

refl : 843 + 1 = 1 + 843
```

~~refl~~ `: {n : Nat} → n + 1 = 1 + n`

What if:

`Construction` $\xrightarrow{\text{implies}}$ `judgemental equality`

`judgemental equality` $\xrightarrow{\text{makes possible}}$ `Construction`

Early instance of 2LTT:

Voevodsky's HTS (Homotopy Type System), 2013

Motivation: "Semisimplicial types"

Problem: construct a type of Reedy fibrant contravariant functors $\Delta_+ \to \mathrm{Type}$

```
A₀ : Type
A₁ : A₀ → A₀ → Type
A₂ : (x y z : A₀) → A₁ x y → A₁ x z → A₁ y z → Type
A₃ : ...
```

Goal: Write down a function $S : \mathbb{N} \to \mathrm{Type}_1$ such that $S(n) \simeq$ type of the tuple $(A_0, A_1, A_2, \ldots, A_n)$.

We can only write down an expression S(x) such that S(n) is correct for *external* n.

Early instance of 2LTT:

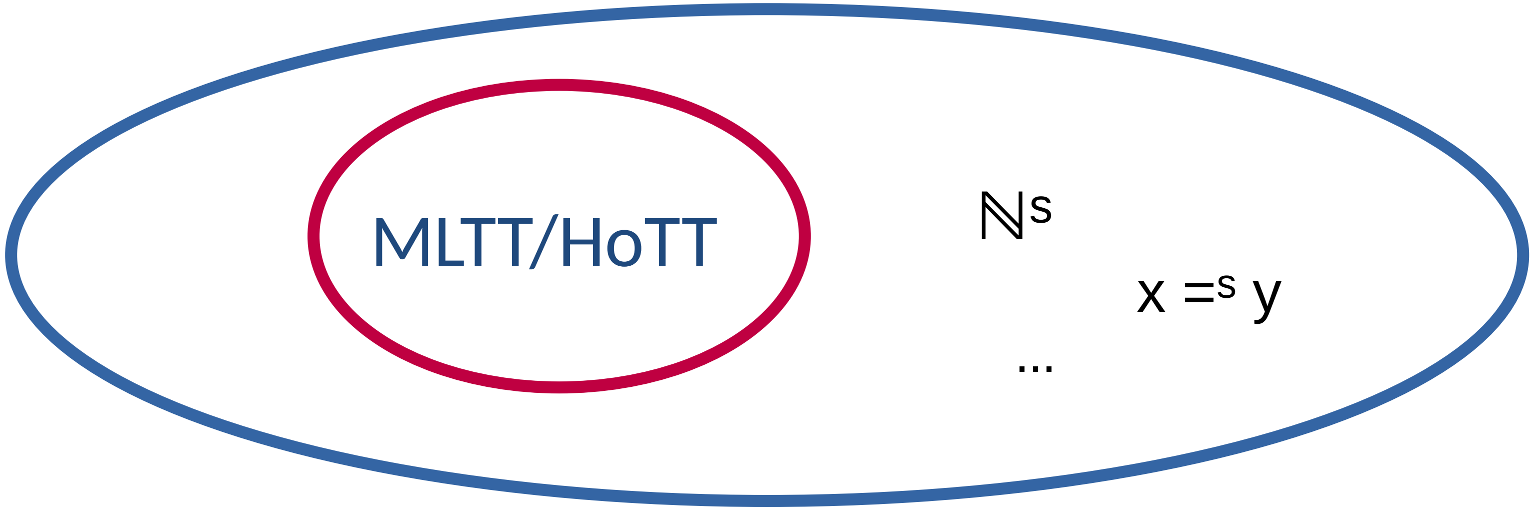
Voevodsky's HTS (Homotopy Type System), 2013

Early instance of 2LTT:

Voevodsky's HTS (Homotopy Type System), 2013

HTS: HoTT extended with:

- "external/strict natural numbers" type
- "external/strict equality"
- ... and the infrastructure to make this work

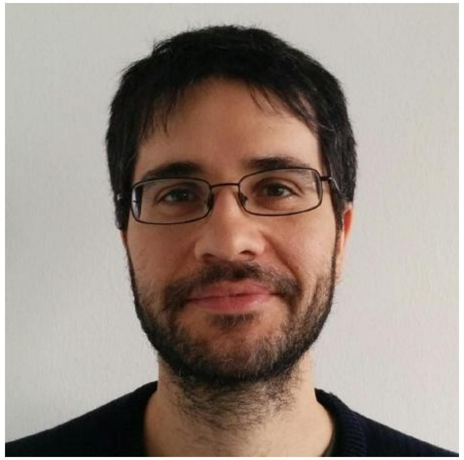


Axiom of HTS:

$\mathbb{N}^s \equiv \mathbb{N}$

(justified by sSet model)

=> Problem solved.

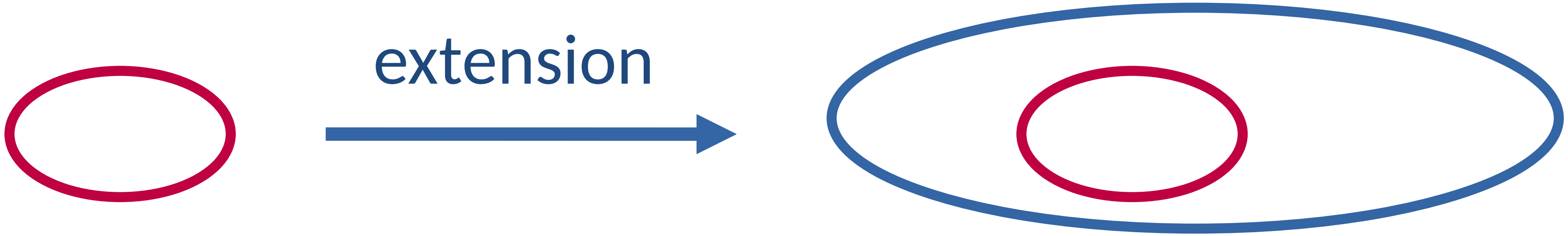


Capriotti's insight:

Without such axioms, we get conservativity.

More than an analogy:

$\text{yoneda} : \; C \to [C^{op}, \text{Set}]$



Any type theory extends to a two-level type theory.

Details: Annenkov-Capriotti-Kraus-Sattler, *Two-level type theory and applications*.

Definition of general 2LTT

An instance of two-level type theory consists of:

* a category **Con** of *contexts*;
* **Ty**^{i} and **Tm**^{i} such that (Con, Ty^i, Tm^i) forms a cwf (the "inner/fibrant level")
* **Ty**^{s} and **Tm**^{s} such that (Con, Ty^s, Tm^s) forms a cwf (the "outer/strict/exo level")
* a conversion morphism **c** from the inner to the outer theory,
  s.t.: - **c** is the identity on contexts
  - **c** preserves context extension
    (but not necessarily type formers!)

Useful special case of 2LTT

A type theory that has lots of type formers:

Π , Σ , 1 , 0 , $+$, $=$, $\mathbb{N}$, higher inductive types , univalent universes

Π , Σ , 1 , 0 , $+$, $=$, $\mathbb{N}$, inductive types , universes , equality reflection
(or UIP & funext)

Useful special case of 2LTT

A type theory that has lots of type formers:

Π , Σ , 1 , 0 , $+$, $=$, $\mathbb{N}$, higher inductive types , univalent universes

Π , Σ , 1 , 0 , $+$, $=$, $\mathbb{N}$, inductive types , universes , equality reflection

(or UIP & funext)

Useful special case of 2LTT

Fibrant types: Π , Σ , 1 , 0 , + , = , $\mathbb{N}$, HITs , univalent universes;
Strict types: 0^s, $+^s$, $=^s$, $\mathbb{N}^s$, strict universes.

Rules: = only works for fibrant types, $=^s$ works for everything.
Induction principles of fibrant types can only eliminate into fibrant types.

Example: $x =^s y \;\rightarrow\; x = y$ but not vice versa.
$\mathbb{N}^s \rightarrow \mathbb{N}$ but not vice versa.
$A +^s B \;\rightarrow\; A + B$ but not vice versa.

Useful special case of 2LTT

Fibrant types: Π , Σ , 1 , 0 , + , = , $\mathbb{N}$, HITs , univalent universes;

Strict types: 0^s, $+^s$, $=^s$, $\mathbb{N}^s$, strict universes.

Voevodsky's HTS is the special case with the assumptions $\mathbb{N}^s \equiv \mathbb{N}$, $0^s \equiv 0$, $+^s \equiv +$.

Example model

Simplicial sets (sSet):
 * Every simplicial set is a context.
 * inner/fibrant level: Kan fibrations (cf Kapulkin-Lumsdaine).
 * outer/strict level: usual presheaf model.

Applications

* Language to formulate new axioms
    e.g. HTS.
* Formalise meta-theoretic statements
    e.g. Shulman’s Reedy fibrant inverse diagrams,
    e.g. “HoTT can define semisimplicial types up to any externally fixed n”.
* “Template programming”
    e.g. for any strict number n, we can develop a theory of univalent n-categories;
        plug in 1, 2, 3, … to get developments in HoTT.
* Staged Compilation with Two-Level Type Theory (ICFP paper by András Kovács).
* (conjectural:) factoring a structural extension $T_1 \to T_2$ as $T_1 \to$ 2LTT $\to T_2$, where
    the second step is an axiomatic extension;
    use Agda's --two-level flag to work in T_2.

Applications

* Language to formulate new axioms
  e.g. HTS.

* Formalise meta-theoretic statements
  e.g. Shulman’s Reedy fibrant inverse diagrams,
  e.g. “HoTT can define semisimplicial types up to any externally fixed n”.

* “Template programming”
  e.g. for any strict number n, we can develop a theory of univalent n-categories;
  plug in 1, 2, 3, … to get developments in HoTT.

* Staged Compilation with Two-Level Type Theory (ICFP paper by András Kovács).

* (conjectural:) factoring a structural extension $T_1 \to T_2$ as $T_1 \to 2LTT \to T_2$, where
  the second step is an axiomatic extension;
  use Agda's --two-level flag to work in T_2.

Thanks for your attention!