Two-Level Type Theory (2LTT) - What is it, what can it do, and does Agda need it?

Nicolai Kraus

AIM XXXIII, 20 Oct 2020

(Thanks + apologies to Jesper!)

What is the problem?

There are "schematic" definitions that cannot be internalised.

E.g.: In Agda, we **can** do 1-categories, 2-cat's, 3-cat's, $\ldots$,
27-cat's, $\ldots$, 2020-cat's, $\ldots$

[Remark: $(\_,1)$-cat's (Capriotti-K'18), general ones wip]

However, we **cannot** do n-categories.

What is the problem?

There are "schematic" definitions that cannot be internalised.

E.g.: In Agda, we **can** do 1-categories, 2-cat's, 3-cat's, $\ldots$,
27-cat's, $\ldots$, 2020-cat's, $\ldots$

[Remark: $(\_, 1)$-cat's (Capriotti-K'18), general ones wip]

However, we **cannot** do n-categories.

How can this happen?

For numerals, expressions normalise and type-check. Example:

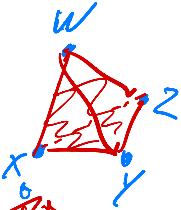
suc-lemma : $(n : \mathbb{N}) \to 1 + n \equiv n + 1$

suc-lemma$(0) \equiv$ refl

suc-lemma$(17) \equiv$ refl

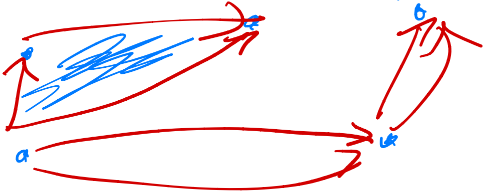
$(n : \mathbb{N}) \to$ suc-lemma$(n) \equiv$ refl

When was this noticed?



It doesn't happen with K (and funext?).

The HoTT community knows the problem of defining *semisimplicial types* since 2012:



$$A_0 : \mathcal{U}$$

$$A_1 : A_0 \to A_0 \to \mathcal{U}$$

$$A_2 : \{x\, y\, z : A_0\} \to (A_1\, x\, y) \to (A_1\, y\, z) \to (A_1\, x\, z) \to \mathcal{U}$$

$$\begin{aligned} A_3 : \{x\, y\, z\, w : A_0\} &\to \{f : A_1\, x\, y\} \to \{g : A_1\, y\, z\} \to \{h : A_1\, z\, w\} \\ &\to \{i : A_1\, x\, z\} \to \{j : A_1\, y\, w\} \to \{k : A_1\, x\, w\} \\ &\to (a : A_2\, f\, g\, i) \to (b : A_2\, f\, j\, k) \to (c : A_2\, i\, h\, k) \\ &\to (d : A_2\, g\, h\, j) \to \mathcal{U} \end{aligned}$$

$$F : \mathbb{N} \to \mathrm{Set}_1$$

Unsolvable (??) task: Define the type of tuples $(A_0, \ldots, A_n)$.

What else is affected?

(My stuff: coherently constant functions, ∞-CwF's.)

MLTT (without K) is based on ∞-groupoids/categories. If we want to formalise a math concept (beyond the set-level), there are two cases:

(1) It can be expressed using only finitely many levels of the ∞-categorical structure.
$\Rightarrow$ Lucky! Often elegant (cf. *synthetic homotopy theory*).
Example: (∞-) Groups as pointed connected types.

(2) No such “shortcut”.
$\Rightarrow$ Can't do it! Example: (∞-) Monoids (?)

[Why even care about (2)? Constructions of (2) can have implications for (1). Plus: Not having (2) is unnatural in terms of models.]

Other descriptions

- *Very dependent function types*
  by Jason J. Hickey, "Formal Objects in Type Theory Using Very Dependent Types", 1996.
  $\{f \mid x : A \to B\}$
  Type of codomain at $a : A$ depends on $f(y)$ for $y < a$.
  Does this make sense for MLTT?
- "(potentially) infinite record types"

Idea of 2LTT

Since we can do [placeholder] for every external natural number, we add a "type" that behaves like the external natural numbers (original Voevodsky 2013, *HTS*: reuse $\mathbb{N}$).

New type: $\mathbb{N}^{\mathsf{s}}$ (*strict* natural numbers)
To make this work, we also need[1]: $\equiv^s$ (*strict* equality)

If a type does not contain $\mathbb{N}^{\mathsf{s}}$ or $\equiv^s$, it corresponds to a "normal" type in "normal" MLTT/HoTT;
more useable: close by strict iso $\rightsquigarrow$ *fibrant* types

Elimination principles of fibrant types only work with fibrant families, to avoid

$$x \equiv y \leftrightarrow x \equiv^s y$$

[1] During the AIM, $\equiv$ is the internal identity type.

Conservativity

Does adding $\mathbb{N}^s$, $\equiv^s$ change the theory (from the point of view of fibrant types)?

$$\text{HoTT/MLTT} \hookrightarrow \text{2LTT} \cong \text{MLTT} + \mathbb{N}^s + \equiv^s$$

A weak version of conservativity can be found in *Two-Level Type Theory and Applications*, Annenkov-Capriotti-K-Sattler 2019.

Wish list

2LTT consists essentially of two parallel theories
– but maybe this is a useful simplification:

- types can carry a flag which indicates *fibrancy*
- all "normal" types of Agda are fibrant
- if $A \simeq^s B$ and A fibrant, then B fibrant;
  can be proved but should be inferred when possible
- when declaring an inductive type, one can choose whether it is fibrant (but only if all indices are fibrant!)
- elimination/pattern matching of fibrant types only allowed for fibrant families
- Conservativity: Fibrant types (in fibrant contexts) can be translated to "normal" types with "normal" inhabitants (not sure about theory and/or practice of this!)