Representing type theories in two-level type theory

Nicolai Kraus and Tom de Jong

University of Nottingham, Nottingham, UK
{nicolai.kraus,tom.dejong}@nottingham.ac.uk

Summary. We describe how two-level type theory can be used to represent a type theory by turning structural extensions into axiomatic extensions, using Riehl and Shulman's simplicial type theory as an example. This talk explains the motivation behind work in progress.

2LTT and extensions of type theories. *Two-level type theory (*2LTT*)* [18, 2, 3, 1] is a framework that allows internalising some of the meta-theory of a given type theory. Simplified, it can be described as one type theory sitting inside another type theory (*inner/fibrant* and *outer/strict/exo-level*). A useful example arises if we start with a version of extensional MLTT (as the strict layer) and assume that some types carry a flag marking them as fibrant, done in a way that ensures that the fibrant layer is exactly homotopy type theory (HoTT) [17]. This situation is modelled by some presheaf models such as the simplicial sets model, where only some maps (Kan fibrations) correspond to (fibrant) types [9]. With the correct specification, the fibrant layer is exactly as expressive as HoTT in the sense of a conservativity property [2, 10].

In the current work, we explore an application of 2LTT in the context of extensions of type theories. We consider two types of extensions:

1. *Axiomatic extensions*, where a type theory T_2 can be represented as a type theory T_1 extended with axioms. Examples are the extension MLTT $\hookrightarrow$ MLTT + funext, or MLTT $\hookrightarrow$ HoTT, if HoTT is the theory developed in the book [17] (where univalence is added as an axiom), without the judgemental computational rules for higher inductive types.
2. *Structural extensions*, where T_2 has all (or most) of the structure of T_1 but also contains additional structural rules. Examples are the extensions of MLTT to (certain versions of) Cubical Type Theory [4], or of HoTT to Riehl and Shulman's type theory of $(\infty, 1)$-categories [15].

Generally speaking, axiomatic extensions are the nicer ones. If we are already familiar with T_1, we merely need to get intuition for internal postulates in order to understand what we might be able to prove in T_2, and if we have a proof assistant for T_1, we can use it as a proof assistant for T_2 by simply adding a postulate. The same is not true for structural extensions; for example, cubical Agda depends on modifications to Agda's source code by the development team, and for Riehl and Shulman's type theory, Kudasov developed the new proof assistant Rzk [11, 12].

The extension HoTT $\hookrightarrow$ 2LTT is a structural extension but it is, in some sense, harmless; since one is ultimately only interested in the fibrant fragment, the mentioned conservativity guarantees that the same internal theorems hold as in HoTT. The point is that:

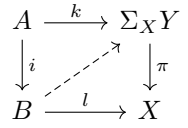
A given structural extension of HoTT *might be an axiomatic extension of* 2LTT*.*

While this seems not necessarily useful from a computational point of view, the axiomatic extension may be easier to get intuition for, and to reason about, than the structural extension; and the setup may enable us to view structurally different type theories in the same setting, a situation somewhat reminiscent of a logical framework [8]. Moreover, the additional language that 2LTT provides may enable us to study properties in the type theory that would ordinarily be meta-theoretic. As a more practical benefit, using Agda's `--two-level` flag [5], we might be able to directly use Agda to implement the new type theory, bypassing the need for the development of a custom-made proof assistant.

Simplicial type theory via two-level type theory. In the remainder of this talk abstract, we sketch how Riehl and Shulman's type theory for $(\infty, 1)$-categories [15] (*simplicial type theory*, STT) can be represented within 2LTT. The theory STT can be described as HoTT with two additional ingredients. The first are two additional context layers that make it possible to talk about *shape inclusions* and *extension types*; the second is the assumption of a directed interval on the context level which ensures that the usual simplicial shapes, such as horns or boundary inclusions, can be constructed as shape inclusions.

Extension types are a concept that multiple type theories make use of, cf. [19] for a summary. The idea is to allow a version of dependent functions $f : \Pi_X Y$, where the type fixes the value of f along some (possibly meta-theoretic) map $i : A \to X$. For example in the cubical case [4], one considers extensions of functions defined on partial boxes.

In the setting of 2LTT, the concept of extension types is very natural. A function $i : A \to B$ on the strict layer is called a *cofibration* (corresponding to what some type theories call shape inclusions) if the *Leibniz exponential* [16] (a.k.a. *pullback exponential*) of i preserves fibrations and trivial fibrations [2, §3.4]. This condition means that, given a fibrant family Y over a not-necessarily fibrant X in a strictly commuting square as shown on the right, the type of strict fillers (i.e. the type of functions $B \to \Sigma_X Y$) is fibrant, and fibrewise contractible if Y is a contractible family. The extension type described above occurs when l is the identity on X.



The second assumption of STT is an interval on the context level, equipped with a bounded linear order, and the new structural rules make it possible to create standard simplicial maps from this interval. This can be replicated directly in 2LTT, with the interval on the strict layer satisfying a cofibrancy assumption. An alternative way is to postulate a functor shape from the category $\mathcal{S}$ to the universe of strict types, where $\mathcal{S}$ is the subcategory of simplicial sets spanned by subfunctors of representables, with the assumption that monos are mapped to cofibrations and finite (co)limits are preserved. The original interval with order can be used to define such a functor and is used in STT to construct the components that the functor provides, which leads us to believe that postulating the functor directly may be the easier approach.

In 2LTT, the pairs (cofibrations, trivial fibrations) and (trivial cofibrations, fibrations) are *orthogonal factorisation systems*, and STT is concerned with an orthogonal factorisation system that lies between these, generated by the cofibration $\Lambda^2_1 \hookrightarrow \Delta^2$ as a member of the left class. If $X \to 1$ lies in the right class, then X is called a *Segal type*; if it additionally satisfies a univalence/completeness condition, it carries the structure required for it to be considered an $(\infty, 1)$-category, following Rezk's model of complete Segal spaces [14].

One appeal of representing STT within 2LTT is that the two layers of the latter can be used to cleanly separate the "combinatorial/set level part" from the "homotopical part" of STT. The first is concerned with general results about simplicial sets, e.g. Joyal's lemma (cf. [13, Lem. 2.3.2.1]), and works on the strict/outer level; the second is the part of the theory which depends on working with spaces rather than sets, here given by the inner/fibrant theory.

As demonstrated by recent work [6, 7], STT can also be represented as an axiomatic extension of HoTT. This requires the stronger assumption that the interval is fibrant, so that everything above can be carried out in HoTT. An explanation for why this works well is that, assuming the corners of the square above are fibrant, the type of strict fillers (i.e. the "meta theoretic" extension type) is equivalent to the type of homotopy fillers (the "internal" extension type). The assumption that the interval is fibrant is easily justified but not completely free: a model that gets lost is the original simplicial sets model [9], where the interval is cofibrant but not fibrant. However, while it can be viewed as a model of STT with the 2LTT approach, the model is "too small" to have undirected equalities, and all types turn out to be ∞-groupoids.

References

[1] Thorsten Altenkirch, Paolo Capriotti, and Nicolai Kraus. Extending homotopy type theory with strict equality. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic (CSL 2016)*, volume 62 of *Leipniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2018.

[2] Danil Annenkov, Paolo Capriotti, Nicolai Kraus, and Christian Sattler. Two-level type theory and applications. *Mathematical Structures in Computer Science*, 33(8):688–743, 2023.

[3] Paolo Capriotti. *Models of type theory with strict equality*. PhD thesis, University of Nottingham, 2017. https://eprints.nottingham.ac.uk/id/eprint/39382.

[4] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In Tarmo Uustalu, editor, *21st International Conference on Types for Proofs and Programs (TYPES 2015)*, volume 69 of *Leipniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2018.

[5] Agda documentation. Two-level type theory. https://agda.readthedocs.io/en/v2.6.3/language/two-level.html.

[6] Daniel Gratzer, Jonathan Weinberger, and Ulrik Buchholtz. Directed univalence in simplicial homotopy type theory. arXiv: 2407.09146, 2024.

[7] Daniel Gratzer, Jonathan Weinberger, and Ulrik Buchholtz. The Yoneda embedding in simplicial homotopy type theory. arXiv: 2501.13229, to appear in the proceedings of *Logic in Computer Science 2025*, 2025.

[8] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the ACM*, 40(1):143–184, 1993.

[9] Krzysztof Kapulkin and Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after Voevodsky). *Journal of the European Mathematical Society*, 23(6):2071–2126, 2021.

[10] András Kovács. Staged compilation with two-level type theory. In Philip Wadler, editor, *ICFP*, volume 6 of *Proceedings of the ACM on Programming Languages*, pages 540–569. Association for Computing Machinery, 2022.

[11] Nikolai Kudasov. Rzk proof assistant. https://github.com/rzk-lang/rzk.

[12] Nikolai Kudasov, Emily Riehl, and Jonathan Weinberger. Formalizing the ∞-categorical Yoneda lemma. In Amin Timany, Dmitriy Traytel, Brigitte Pientka, and Sandrine Blazy, editors, *CPP '24*, pages 274–290. Association for Computing Machinery, 2024.

[13] Jacob Lurie. *Higher Topos Theory*, volume 170 of *Annals of Mathematics Studies*. Princeton University Press, 2009.

[14] Charles Rezk. A model for the homotopy theory of homotopy theory. *Transactions of the American Mathematical Society*, 353(3):973–1007, 2001.

[15] Emily Riehl and Michael Shulman. A type theory for synthetic ∞-categories. *Higher Structures*, 1(1):147–224, 2017.

[16] Emily Riehl and Dominic Verity. The theory and practice of Reedy categories. *Theory and Applications of Categories*, 29(9):256–301, 2013.

[17] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[18] Vladimir Voevodsky. A simple type theory with two identity types. Unpublished note, available at https://www.math.ias.edu/vladimir/sites/math.ias.edu.vladimir/files/HTS.pdf, 2013.

[19] Tesla Zhang. Three non-cubical applications of extension types. arXiv: 2311.05658, 2023.