A Flexible Multimodal Proof Assistant

Philipp Stassen, Daniel Gratzer, and Lars Birkedal

Aarhus University
stassen@cs.au.dk, gratzer@cs.au.dk, birkedal@cs.au.dk

A fundamental benefit to working type theoretically is the possibility of working within a proof assistant, which can check and even aid in the construction of complex theorems. Implementing a proof assistant, however, is a highly nontrivial task. In addition to a solid theoretical foundation for the particular type theory, numerous practical implementation issues must be addressed.

Recently, there has been a growing interest in type theories which include *modalities*, unary type constructors which need not commute with substitution. While there has been rapid progress on modal type theories, it is unknown whether standard implementation techniques extend to them. Mainstream proof assistants have begun to experiment with modalities [Vez18], but these implementations are costly and must be specialized to a particular modal situation. Realistically, a practitioner may use a collection of modalities for only one proof and it is impractical to invest in a specialized proof assistant each time. Similar churn has afflicted modal type theories generally and it has pushed type theorists to define *general* systems which can be instantiated with various modalities [Gra+20; LSR17]. We believe that a similar approach alleviate the constant need to implement new modal proof assistants.

Here we focus on MTT [Gra+21], a general modal type theory which can internalize arbitrary collections of (dependent) right adjoints [Bir+20]. These modalities are specified by mode theories [LS16], 2-categories whose objects correspond to modes, morphisms to modalities, and 2-cells to natural transformations between modalities.

MTT has been used to model a variety of existing modal type theories including calculi for guarded recursion, internalized parametricity, and axiomatic cohesion. Better still, MTT has a robustly developed metatheory [Gra+21] which applies *irrespective* of the chosen modalities. An implementation of MTT could therefore conceivably be designed to allow the user to freely change the mode theory without reimplementing the entire proof assistant each time. In fact, recently Gratzer [Gra22] has proven a generic normalization result for MTT. While this result provides the foundation of a flexible implementation, it remains to convert this proof into a practical type checking algorithm.

From theory to practice Converting the theoretical guarantee of normalization into an executable program is not a small step. A first obstacle is the syntax of MTT itself: prior work has exclusively considered an algebraic presentation of the syntax as a generalized algebraic theory. While mathematically elegant, a proof assistant requires a more streamlined and ergonomic syntax. Once a more convenient syntax has been designed, one must adapt the normalization proof to a normalization algorithm. Normalization is proven by a sophisticated *gluing* argument, and while the proof is reminiscent of normalization-by-evaluation [Abe13] it remains to extract such an algorithm.

The majority of subtleties in the implementation flow not from the modal types *per se*, but from the extensions to contexts and substitutions needed to support them. In particular, each 2-cell in the mode theory induces a new form of substitution and these *key substitutions* accumulate at variables. As a result, each variable in MTT is annotated with a 2-cell describing its extraction from the context. This additional piece of data disrupts a crucial property of modern

NbE: variables can no longer be represented in a way invariant under weakening. Without this invariant, one cannot adapt the substitution-free NbE familiar from MLTT [Abe13].

We therefore only consider the restricted class of *preordered* mode theories—mode theories with at most one 2-cell between any pair of modalities. In this case, the modal substitution apparatus for MTT is dramatically simplified, and the problematic substitutions evaporate. Without this obstruction, we have successfully implemented a normalization algorithm which elegantly handles multiple modalities. Crucially, the normalization algorithm does not depend on the particulars of the mode theory and can be applied without change to any preordered collection of modalities.

Normalization-by-evaluation The normalization proof for MTT follows the structure of a normalization-by-evaluation proof. Rather than fixing a rewriting system, a term is *evaluated* into a carefully chosen semantics equipped with a quotation function reifying an element of the semantic domain to a normal form. The entire normalization function is then a round-trip from syntax to semantics and then back to normal forms. While the proof of normalization uses a traditional denotational model for a semantic domain, this is impractical for implementation.

Instead `mitten` follows the literature on normalization-by-evaluation and uses a *defunctionalized* and *syntactic* semantic domain [Abe13]. This approach has previously been adapted to work with a modal type theory [GSB19], but adapting it to MTT introduces several novel challenges. In particular, variables must be annotated with 2-cells from the mode theory, and managing these annotations ripples out through the entire algorithm. We show that for preordered mode theories said annotations omitted and reconstructed during type checking.

Mode theories Normalization for MTT does not immediately imply the decidability of type equality. Terms (and therefore types) mention both 1- and 2-cells from the mode theory, and deciding their equality is a necessary precondition for deciding type equality. Moreover, deciding the equality of 1- and 2-cells, even in a finitely presented 2-category, is well-known to be undecidable.[1] Special attention is therefore necessary for each mode theory to ensure that the normalization algorithm for MTT is sufficient to yield a type-checker. While this rules out a truly generic proof assistant for MTT which works regardless of the choice of mode theory, `mitten` shows that the next best result is obtainable. We implement `mitten` parametrically in a module fully describing the mode theory and show that the type-checker can be designed to rely only on the existence of such a decision procedure.

Contributions We contribute a defunctionalized NbE algorithm which reduces the type checking problem for MTT to deciding the word problem for the mode theory.

Furthermore, we specify a bidirectional syntax for MTT together with a type checking algorithm. Type checking is decidable if and only if the word problem of the mode theory is.

We have put these results into practice with `mitten`, an prototype implementation of MTT based on this algorithm. `mitten` supports the replacement of the underlying mode theory with minimal alterations, allowing a user to construct specialized proof assistants for modal type theories by merely supplying a single module specifying the mode theory together with equality functions for 0-, 1-, and 2-cells.

We presented a version of this ongoing work at the WITS workshop in January 2022.

[1]The word problem is well-known to be undecidable for finitely presented groups which can be realized as finitely-presented categories and therefore locally discrete finitely-presentable 2-categories.

References

[Abe13] Andreas Abel. "Normalization by Evaluation: Dependent Types and Impredicativity". Habilitation. Ludwig-Maximilians-Universität München, 2013 (cit. on pp. 1, 2).

[Bir+20] Lars Birkedal, Ranald Clouston, Bassel Mannaa, Rasmus Ejlers Møgelberg, Andrew M. Pitts, and Bas Spitters. "Modal dependent type theory and dependent right adjoints". In: *Mathematical Structures in Computer Science* 30.2 (2020), pp. 118–138. DOI: 10.1017/S0960129519000197. eprint: 1804.05236 (cit. on p. 1).

[Gra22] Daniel Gratzer. "Normalization for multimodal type theory". In: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2022. DOI: 10.1145/3531130.3532398 (cit. on p. 1).

[Gra+21] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. "Multimodal Dependent Type Theory". In: *Logical Methods in Computer Science* Volume 17, Issue 3 (July 2021). DOI: 10.46298/lmcs-17(3:11)2021. URL: https://lmcs.episciences.org/7713 (cit. on p. 1).

[Gra+20] Daniel Gratzer, G.A. Kavvos, Andreas Nuyts, and Lars Birkedal. "Multimodal Dependent Type Theory". In: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*. LICS '20. ACM, 2020. DOI: 10.1145/3373718.3394736 (cit. on p. 1).

[GSB19] Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. "Implementing a Modal Dependent Type Theory". In: *Proc. ACM Program. Lang.* 3 (ICFP 2019). DOI: 10.1145/3341711 (cit. on p. 2).

[LS16] Daniel R. Licata and Michael Shulman. "Adjoint Logic with a 2-Category of Modes". In: *Logical Foundations of Computer Science*. Ed. by Sergei Artemov and Anil Nerode. Springer International Publishing, 2016, pp. 219–235. DOI: 10.1007/978-3-319-27683-0_16 (cit. on p. 1).

[LSR17] Daniel R. Licata, Michael Shulman, and Mitchell Riley. "A Fibrational Framework for Substructural and Modal Logics". In: *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*. Ed. by Dale Miller. Vol. 84. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 25:1–25:22. DOI: 10.4230/LIPIcs.FSCD.2017.25 (cit. on p. 1).

[Vez18] Andrea Vezzosi. *agda-flat*. 2018. URL: https://github.com/agda/agda/tree/flat (cit. on p. 1).