Engineering logical relations for MLTT in Coq

Arthur Adjedj[1][2], Meven Lennon-Bertrand[3], Kenji Maillard[1], and Loïc Pujet[1]

[1] Gallinette Project-Team, Inria, Nantes, France
[2] ENS Paris-Saclay, Gif-sur-Yvette, France
[3] University of Cambridge, UK

Abstract

We report on a mechanization in the Coq proof assistant of the decidability of conversion and type-checking for Martin-Löf Type Theory (MLTT), extending a previous Agda formalization. Our development proves the decidability not only of conversion, but also of type-checking, using bidirectional derivations that are canonical for typing. Moreover, we wish to narrow the gap between the object theory we formalize (currently MLTT with Π, Σ, $\mathbb{N}$ and one universe) and the metatheory used to prove the normalization result, *e.g.,* MLTT, to a mere difference of universe levels. We thus avoid induction-recursion or impredicativity, which are central in previous work. Working in Coq, we also investigate how its features, including universe polymorphism and the metaprogramming facilities provided by tactics, impact the development of the formalization compared to the development style in Agda. The development is freely accessible on GitHub [2].

MLTT and its metatheory Establishing meta-theoretic properties such as the existence of canonical forms or decidability of the derivability of judgements of dependent type systems is a notoriously complex endeavour. For instance, the MetaCoq project [11, 12] aims at entirely formalizing the Predicative Calculus of Cumulative Inductive Constructions (PCUIC) underlying Coq, and showing the correctness of an implementation of a typechecker. However, to do so it assumes an axiom stating that the theory is normalizing.

Indeed, for these meta-theoretical properties, type dependency precludes most proof strategies, which ultimately rely on a stratification between types and terms. To go beyond these limitations, Abel et al. [1] formalize an inductive-recursive [5] definition of a logical relation for a representative fragment of MLTT in Agda, to show normalization and decidability of conversion for this theory. This technique was further extended to more complex theories [6, 10].

The use of the induction-recursion scheme however introduces a new gap between the object theory being formalized (which only supports a handful of simple inductive types), and the meta-theory used to formalize the result. While exploring normalization proofs for complex inductive schema is a very valuable endeavour, we wish to go the other way around, and narrow this gap by using only regular indexed inductive types. This is both a requirement and a benefit of working in Coq, which only handles this class of inductive types. Thus, we reformulate the logical relation using *small induction-recursion*, which can in turn be encoded using simple indexed inductive types [7]. This strategy requires definitions that are replicated across several universe levels, for which the universe polymorphic features of Coq come in handy.

A bidirectional presentation of MLTT In [1], only decidability of conversion is shown. While this is definitely the most intricate part of showing decidability of type-checking, going from the former to the latter is non-trivial. Indeed, type-checking for MLTT as defined in [1] is not, in general, decidable, for lack of annotations [4, 15].

In our development, we show decidability of typing, by extending algorithmic conversion-checking to a full account of algorithmic typing, described in a bidirectional fashion [8, 9].

Following the strategy implemented for instance by Coq, we use annotated (Church-style) abstractions, so that inference is complete, *i.e.,* every well-typed term infers a type.

More generally, ideas from bidirectional typing help greatly in guiding the definition and handling of the algorithmic parts of the system. For instance, for most proofs on the algorithmic system we rely on a custom induction principle which threads the invariants maintained by a bidirectional algorithm, giving us extra hypotheses for each induction step. This lets us handle once and for all these invariants that are required for most proofs, rather than bundling them in the predicate proven by induction, which would mean showing their preservation again during each proof by induction.

Three logical relations in one We show that a bidirectional presentation of our type theory is equivalent to its standard declarative presentation given in the Martin-Löf Logical Framework. To do so, we parametrize the definition of the logical relation by a generic typing interface. The interface is instantiated 3 times: once with the declarative typing, once with a mixed system with declarative typing and algorithmic conversion, and finally with a fully algorithmic system using bidirectional typing. These different instances are used to gradually show properties of the system: the declarative instance lets us show enough good properties of the declarative system to be able to show that the mixed system fits the generic typing interface; the mixed instance proves that declarative and algorithmic conversion coincide, which is used to show that bidirectional typing fits in the generic interface; finally the fully algorithmic instance gets us the desired equivalence, showing that the type-checking algorithm is sound and complete.

Engineering aspects We rely on Autosubst [14, 3] to deal with all the aspect of the raw syntax, defining untyped renamings and substitutions, generating boilerplate lemmas on these, as well as providing tactical support to discharge equational obligations.

To support working with multiple different notions of conversion and typing, sometimes simultaneously, we devised a generic notation system based on type classes in the Math Classes style [13], with a system of tags that lets us disambiguate between the different notions when needed, but can be ignored safely when working on a single notion or parametrically.

To ease the development, we use tactics to provide for judgement-independent notions, e.g. we use a single `irrelevance` tactic to discharge goals requiring a lemma stating that some form of the logical relation is irrelevant in some of its parameters. An important part of the work achieved by the definition of the logical relation consist in its generalization of typing contexts through Kripke-style quantifications over renamings and substitutions, and we use instantiation tactics to automatically apply lemmas to the relevant hypotheses.

Future work We could add more universes to obtain a hierarchy of arbitrary finite length, and see no theoretical obstacle in doing so. We plan to extend the formalization to a scheme of indexed inductive types used in Coq, hence narrowing the gap between the object theory and the metatheory used to prove its normalization: this would lead to a formalization of MLTT with n universes into MLTT with $n + k$ universes for a (small but strictly positive) constant k.

On the bidirectional side, we do not cover the common pattern, used for instance in the kernel of Agda, of having some terms that only check (typically, unannotated abstraction). It would be interesting to obtain a decidability result that covers these as well.

Finally, there is a large space left to improve automation, taking inspiration from the rich Coq ecosystem. Indeed, the main difficulty for a proof by logical relations is in the setup of the relation, but most proof obligations are rather repetitive and unsurprising. While our tactics already relieve us from quite a bit of this tedious work, they are far from making it all disappear.

References

[1] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. Decidability of conversion for type theory in type theory. *Proc. ACM Program. Lang.*, 2(POPL):23:1–23:29, 2018.

[2] Arthur Adjedj, Meven Lennon-Bertrand, Kenji Maillard, and Loïc Pujet. Logical Relation for MLTT in Coq. `https://github.com/CoqHott/logrel-coq`, 2023.

[3] Adrian Dapprich. Generating Infrastructural Code for Terms with Binders using MetaCoq and OCaml. Bachelor thesis, Saarland University, 2021.

[4] Gilles Dowek. The undecidability of typability in the Lambda-Pi-calculus. In Marc Bezem and Jan Friso Groote, editors, *Typed Lambda Calculi and Applications*, pages 139–145. Springer Berlin Heidelberg, 1993.

[5] Peter Dybjer and Anton Setzer. Induction-recursion and initial algebras. *Annals of Pure and Applied Logic*, 124(1-3):1–47, 2003.

[6] Gaëtan Gilbert, Jesper Cockx, Matthieu Sozeau, and Nicolas Tabareau. Definitional Proof-Irrelevance without K. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–28, 2019.

[7] Peter Hancock, Conor McBride, Neil Ghani, Lorenzo Malatesta, and Thorsten Altenkirch. Small Induction Recursion. In Masahito Hasegawa, editor, *Typed Lambda Calculi and Applications*, pages 156–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] Meven Lennon-Bertrand. Complete Bidirectional Typing for the Calculus of Inductive Constructions. In Liron Cohen and Cezary Kaliszyk, editors, *12th International Conference on Interactive Theorem Proving, ITP 2021, June 29 to July 1, 2021, Rome, Italy (Virtual Conference)*, volume 193 of *LIPIcs*, pages 24:1–24:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[9] Meven Lennon-Bertrand. *Bidirectional Typing for the Calculus of Inductive Constructions*. PhD thesis, Nantes Université, 2022.

[10] Loïc Pujet and Nicolas Tabareau. Impredicative Observational Equality. *Proc. ACM Program. Lang.*, 7(POPL), 2023.

[11] Matthieu Sozeau, Abhishek Anand, Simon Boulier, Cyril Cohen, Yannick Forster, Fabian Kunze, Gregory Malecha, Nicolas Tabareau, and Théo Winterhalter. The MetaCoq Project. *J. Autom. Reason.*, 64(5):947–999, 2020.

[12] Matthieu Sozeau, Simon Boulier, Yannick Forster, Nicolas Tabareau, and Théo Winterhalter. Coq Coq correct! verification of type checking and erasure for Coq, in Coq. *Proc. ACM Program. Lang.*, 4(POPL):8:1–8:28, 2020.

[13] Bas Spitters and Eelis Van Der Weegen. Type classes for mathematics in type theory. *Mathematical Structures in Computer Science*, 21(4):795–825, 2011.

[14] Kathrin Stark, Steven Schäfer, and Jonas Kaiser. Autosubst 2: reasoning with multi-sorted de Bruijn terms and vector substitutions. In Assia Mahboubi and Magnus O. Myreen, editors, *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019*, pages 166–180. ACM, 2019.

[15] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, 2006.