Mechanizing logical relations

Josselin Poiret

Nantes Université, École Centrale Nantes, CNRS, INRIA, LS2N, UMR 6004, F-44000 Nantes, France

Meta-theoretical studies of type theories often rely on realizability techniques, including the so-called "logical relations", to establish interesting properties of the systems. The prototypical examples of such results are normalization for simply typed lambda calculus with Tait's computability technique [Tai67], then System F with Girard's reducibility candidates [Gir72], the same year as Martin-Löf Type Theory (MLTT) with [Mar98].

However, type theorists and practitioners did not stop with MLTT, and newer, more convenient systems were created and gradually refined for specific applications. Today, a plethora of systems are used to prove and program, with examples as diverse as Homotopy Type Theory (HoTT) and Cubical Type Theory for synthetic homotopy theory, Multi-Modal Type Theory for the internal logic of specific toposes, or 2-Level Type Theory for interaction between HoTT and a type theory with uniqueness of identity proofs (UIP). With a proliferation of new, more exotic type systems, the demand for meta-theoretical verification has grown significantly.

Proving properties of such systems has also gotten harder, proportionally with their expressive power and with the expectation that such proofs must now be formalized. Some formalization efforts have focused on the engineering part of such proofs, while also providing a solid base for experimentation with new features. Projects such as MetaCoq [Soz+20], Agda Core or Lean4Lean [Car24] focus on the idiosyncrasies of specific proof assistants, while LogRel-MLTT [AÖV18] and its direct descendant LogRel-Coq [Adj+24] study a more idealized version of MLTT.

In the case of the latter, we have been trying to simplify its implementation, as some design choices for the propagated invariants can sometimes feel very arbitrary and redundant. We are also trying to precisely understand what comparisons can be made with high level abstractions like proofs by normalisation by evaluation [Fio22] and Synthetic Tait Computability (STC) [Ste21] and whether some of their tools can be used in our developments. I will present some work-in-progress improvements, proof techniques, as well as future directions for the development, highlighting our expectations for the project.

What's in a logical relation formalization? Both of LogRel-MLTT and LogRel-Coq follow a similar pattern of defining realizers in an inductive-recursive manner (with a trick when encoding in Rocq). What kind of information these realizers contain is not fixed though! The definition of realizers is parametrized by another notion of "semantic typing", which has to follow quite a lot of axioms for the fundamental lemma to go through. The choice of semantic typing influences what kind of result one gets via the logical relation argument, but also constrains what kind of type theory we can interpret.

In the current development, to prove the decidability of typing and conversion, this semantic typing is instantiated by the graph of a bidirectional typing algorithm. However, that system does not satisfy the axioms mentioned above at first glance. To prove that it does, one needs to go through the whole fundamental lemma with a different choice of semantic typing, which seems very inefficient (and not very principled). We would like to only instantiate the construction once with a system that gives us the expected corollaries directly.

Canonical derivations Standard presentations of dependent type theories include the general application inference rule and its congruence counterpart

$$\frac{\Gamma \vdash f : \Pi\, A\, B \qquad \Gamma \vdash a : A}{\Gamma \vdash f : B[\uparrow a]}\ \textsc{App} \qquad \frac{\Gamma \vdash f \equiv g : \Pi\, A\, B \qquad \Gamma \vdash a \equiv b : A}{\Gamma \vdash f\ a \equiv g\ b : B[\uparrow a]}\ \textsc{AppCong}$$

These inference rules, more specifically AppCong, cause derivations of conversion to not be unique, as for example $\Gamma \vdash (\lambda\ x : A, x)\ y \equiv (\lambda\ x : A, x)\ y : A$ can be derived because of either AppCong directly, or by a transitive combination of β-reductions. For the same reason, derivations do not contain normal form information, as applications can be typed without reducing them.

A different presentation of dependent type theory can also be given, with a work-in-progress formalization, whose derivations all faithfully reflect β-normal η-long reduction information. This can be achieved by avoiding the general App and AppCong rules, and instead always relying on reduction, even for typing. This technique is reminiscent of cut-free presentations of first-order logic, and is behind some bidirectional type systems, including the one present in LogRel-Coq, although only for the conversion judgements [Len21].

Once the system is established, a key property to show first is that derivations are indeed unique, as well as inferred types. With this uniqueness result, basic properties like transitivity of conversion are shown with straightforward strong inductions. Then, the crux of the realizability technique should be able to show that general elimination rules are admissible for our new system.

Proof-engineering techniques in Rocq As is usual in meta-theory, proofs are conceptually clear but require a lot of tedious work. I will also mention a couple of techniques that we employ to stay within Rocq's restrictions while keeping our proofs manageable.

One drastic change is to only use partial equivalence relations (PERs) for the judgements of our type theory, and create a completely binary version of the system. As an example, the usual judgement $\Gamma \vdash a : A$ would correspond to $\Gamma \equiv \Gamma \vdash a \equiv a : A \equiv A$, and $\Gamma \vdash a \equiv b : A$ to $\Gamma \equiv \Gamma \vdash a \equiv b : A \equiv A$. Though this might seem like a needless increase in complexity, this lets us treat all rules uniformly while not repeating ourselves when proving properties about both the typing rules and the congruence rules.

More pragmatically, mutual inductive types in Rocq are quite impractical to use, especially since one needs to manually generate the combined schemes, as well as provide numerous induction motives on use. Instead, I use a single indexed inductive type, leading to a net gain in conciseness when writing down the type signatures of properties. Using the unary parametricity translation of that inductive lets me define a strong induction principle, along the lines of [Tas19]. With these generalizations, proving properties of our canonical system is relatively speedy.

References

[Adj+24] Arthur Adjedj et al. "Martin-Löf à la Coq". In: *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2024, London, UK, January 15-16, 2024*. Ed. by Amin Timany et al. ACM, 2024, pp. 230–245. DOI: 10.1145/3636501.3636951. URL: https://doi.org/10.1145/3636501.3636951.

[AÖV18] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. "Decidability of conversion for type theory in type theory". In: *Proc. ACM Program. Lang.* 2.POPL (2018), 23:1–23:29. DOI: 10.1145/3158111. URL: https://doi.org/10.1145/3158111.

[Car24] Mario Carneiro. "Lean4Lean: Towards a formalized metatheory for the Lean theorem prover". In: *CoRR* abs/2403.14064 (2024). DOI: 10.48550/ARXIV.2403.14064. arXiv: 2403.14064. URL: https://doi.org/10.48550/arXiv.2403.14064.

[Fio22] Marcelo Fiore. "Semantic analysis of normalisation by evaluation for typed lambda calculus". In: *Math. Struct. Comput. Sci.* 32.8 (2022), pp. 1028–1065. DOI: 10.1017/S0960129522000263. URL: https://doi.org/10.1017/S0960129522000263.

[Gir72] Jean-Yves Girard. "Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur". PhD thesis. Université Paris VII, June 1972.

[Len21] Meven Lennon-Bertrand. "Complete Bidirectional Typing for the Calculus of Inductive Constructions". In: *12th International Conference on Interactive Theorem Proving, ITP 2021, June 29 to July 1, 2021, Rome, Italy (Virtual Conference)*. Ed. by Liron Cohen and Cezary Kaliszyk. Vol. 193. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 24:1–24:19. DOI: 10.4230/LIPICS.ITP.2021.24. URL: https://doi.org/10.4230/LIPIcs.ITP.2021.24.

[Mar98] Per Martin-Löf. "An Intuitionistic Theory of Types". In: *Twenty Five Years of Constructive Type Theory*. Ed. by Giovanni Sambin and Jan M. Smith. Also known as MLTT72, published version of an unpublished 1972 note. Clarendon Press, 1998, pp. 127–172.

[Soz+20] Matthieu Sozeau et al. "The MetaCoq Project". In: *J. Autom. Reason.* 64.5 (2020), pp. 947–999. DOI: 10.1007/S10817-019-09540-0. URL: https://doi.org/10.1007/s10817-019-09540-0.

[Ste21] Jonathan Sterling. "First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory". Version 1.1, revised May 2022. PhD thesis. Carnegie Mellon University, 2021. DOI: 10.5281/zenodo.6990769.

[Tai67] William W. Tait. "Intensional Interpretations of Functionals of Finite Type I". In: *J. Symb. Log.* 32.2 (1967), pp. 198–212. DOI: 10.2307/2271658. URL: https://doi.org/10.2307/2271658.

[Tas19] Enrico Tassi. "Deriving Proved Equality Tests in Coq-Elpi: Stronger Induction Principles for Containers in Coq". In: *10th International Conference on Interactive Theorem Proving, ITP 2019, September 9-12, 2019, Portland, OR, USA*. Ed. by John Harrison, John O'Leary, and Andrew Tolmach. Vol. 141. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 29:1–29:18. DOI: 10.4230/LIPICS.ITP.2019.29. URL: https://doi.org/10.4230/LIPIcs.ITP.2019.29.