Towards Quotient Inductive Types in Observational Type Theory

Thiago Felicissimo and Nicolas Tabareau

INRIA, France

Quotient Inductive Types (QITs) are a family of inductive types supporting not only the declaration of generators but also equations. A well-known example of QIT is the following type of finite multisets, whose elements are basically lists considered up to permutation.

$$\begin{aligned}&\text{Inductive } \mathsf{MSet}\ (A : \mathrm{Type}) : \mathrm{Type} :=\\ &\quad |\ [] : \mathsf{MSet}\ A\\ &\quad |\ \_ :: \_\ (x : A)(m : \mathsf{MSet}\ A) : \mathsf{MSet}\ A\\ &\quad |\ \mathsf{MSet}_=\ (x\ y : A)(m : \mathsf{MSet}\ A) : (x :: y :: m) = (y :: x :: m)\end{aligned}$$

QITs are a special case of Higher Inductive Types (HITs) in which the defined type is implicitly declared to be an hSet. Such types has been gaining much interest in the last years [Uni13]. Notably, Altenkirch and Kaposi have proposed the use of Quotient Inductive-Inductive Types (QIITs), an extension of QITs with induction-induction, to formalize the metatheory of type theory in an intrinsic fashion [AK16].

QI(I)Ts have been investigated in works by Kovács *et al.* [KKA19, KK20] and Fiore *et al.* [FPS22], in both cases by proposing universal types which can be used for encoding other QI(I)Ts. However, they only focus on the case of extensional type theory (ETT), which is hard to implement in proof assistants due to its undecidable conversion. On the other hand, it is also known that the intensional type theories of proof assistants such as Rocq, Lean and Agda do not behave well with QITs. Indeed, the equality axioms that are added when declaring a QIT break canonicity, given that J does not reduce when eliminating them. Thankfully, we can instead shift the focus to Observational Type Theory (OTT) [AMS07, PT22], in which the eliminator for equality does not inspect the equality proof but instead computes using its endpoints. Because of this, (consistent) equality axioms can be safely added to the theory without breaking canonicity.

This work We report a work in progress metatheoretic study of OTT with QITs. In order to achieve this, many directions seem possible. Pujet and Tabareau have recently justified extensions of OTT with both quotient types Q [PT22] and inductive types [PLT25][1], therefore it can seem tempting to justify QITs by encoding them as regular inductive types quotiented by Q. Unfortunately, this encoding does not yield an eliminator with the proper definitional equalities, and for infinitary QITs this construction does not seem even possible [LS20]. Another approach would be to directly extend OTT with an inductive scheme for QITs and try to prove its metatheory, yet inductive schemes are hard to study in a formal manner due to the proliferation of indexes and vector notations. Therefore, we instead take a similar approach as the previous work on QITs, and extend OTT with a universal type which can be used to encode (non-indexed) QITs, by proposing an adaptation of Fiore *et al.*'s QW types.

[1] A version of OTT with Q and inductive types was actually already implemented in the Epigram 2 proof assistant, though, to the best of our knowledge, their metatheory had not yet been studied.

A universal QIT We now present a finitary version of our universal QIT, an infinitary version can also be found in the formalization.[2] Similarly to QW types, our definition goes in two steps: we start by defining a (unquotiented) type $\overline{\mathsf{Tm}}$, required to formulate the equations used for the quotient later. To do this, let signatures Sig be the record type with a field $\mathsf{C} : \mathrm{Type}$ to represent the type of constructors, and a field $\mathsf{arity} : \mathsf{C} \to \mathsf{Nat}$ which maps each constructor to its number of recursive arguments. We then define $\overline{\mathsf{Tm}}$ in the following way. Here, the parameter $\Gamma : \mathrm{Type}$ should be seen as a context which adds a new inhabitant $\mathsf{var}\ x$ for each $x : \Gamma$.

$$
\begin{aligned}
&\mathrm{Inductive}\ \overline{\mathsf{Tm}}\ (\Sigma : \mathsf{Sig})(\Gamma : \mathrm{Type}) : \mathrm{Type} :=\\
&\quad |\ \mathsf{var}\ (x : \Gamma) : \overline{\mathsf{Tm}}\ \Sigma\ \Gamma\\
&\quad |\ \mathsf{sym}\ (c : \Sigma.\mathsf{C})\ (\mathbf{t} : \mathsf{Vec}\ (\overline{\mathsf{Tm}}\ \Sigma\ \Gamma)\ (\Sigma.\mathsf{arity}\ c)) : \overline{\mathsf{Tm}}\ \Sigma\ \Gamma
\end{aligned}
$$

Using the above type, we define equational theories $\mathsf{EqTh}\ \Sigma$ to be the record type with a field $\mathsf{E} : \mathrm{Type}$ for the type of equations, a field $\mathsf{Ctx} : \mathsf{E} \to \mathrm{Type}$ mapping each equation to a type representing its context, and fields $\mathsf{lhs}, \mathsf{rhs} : (e : \mathsf{E}) \to \overline{\mathsf{Tm}}\ \Sigma\ (\mathsf{Ctx}\ e)$ mapping each equation to its left- and right-hand sides. We then define our universal QIT in the following manner. In the type of eq, we write $\_\langle\_\rangle : \overline{\mathsf{Tm}}\ \Sigma\ \Gamma \to (\Gamma \to \mathsf{Tm}\ \Sigma\ \mathcal{E}) \to \mathsf{Tm}\ \Sigma\ \mathcal{E}$ for the "substitution" function defined by $(\mathsf{sym}\ c\ [t_1 \dots t_k])\langle\gamma\rangle := \mathsf{sym}\ c\ [t_1\langle\gamma\rangle \dots t_k\langle\gamma\rangle]$ and $(\mathsf{var}\ x)\langle\gamma\rangle := \gamma\ x$.

$$
\begin{aligned}
&\mathrm{Inductive}\ \mathsf{Tm}\ (\Sigma : \mathsf{Sig})\ (\mathcal{E} : \mathsf{EqTh}\ \Sigma) : \mathrm{Type} :=\\
&\quad |\ \mathsf{sym}\ (c : \Sigma.\mathsf{C})\ (\mathbf{t} : \mathsf{Vec}\ (\mathsf{Tm}\ \Sigma\ \mathcal{E})\ (\Sigma.\mathsf{arity}\ c)) : \mathsf{Tm}\ \Sigma\ \mathcal{E}\\
&\quad |\ \mathsf{eq}\ (e : \mathcal{E}.\mathsf{E})\ (\gamma : \mathcal{E}.\mathsf{Ctx}\ e \to \mathsf{Tm}\ \Sigma\ \mathcal{E}) : (\mathcal{E}.\mathsf{lhs}\ e)\langle\gamma\rangle = (\mathcal{E}.\mathsf{rhs}\ e)\langle\gamma\rangle
\end{aligned}
$$

The above type can be used to define other (non-indexed) QITs. For instance, in the formalization we use it to define the type of finite multisets as well as the untyped SK calculus. We also provide the example of countably-branching trees, using the infinitary version of Tm.

Compared with QW types, our type is designed to employ a first-order representation of recursive arguments (an approach notably promoted by Dagand and McBride [Dag13]), by using vectors instead of functions. This aspect seems essential if we want encodings of QITs to satisfy canonicity: for instance, the encoding of the boolean type using QW types would have infinitely many normal inhabitants in the empty context.[3] Moreover, unlike with W and QW types, encodings using our type do not require functional extensionality or fancy tricks like [Hug21]: they work directly in intensional type theory, as witnessed by our Agda formalization.

The plan We plan to provide a metatheoretic justification of OTT with QITs using the above universal type, in three steps. (1) First, we will give an inductive scheme for non-indexed QITs and prove that all types can be encoded using our universal QIT. Importantly, we aim at obtaining eliminators that compute definitionally, and types that satisfy canonicity (eg., the encoding of the boolean type should only have two normal forms in the empty context). (2) Then, the second step will be to prove normalization of OTT extended with our universal QIT. While a pen-and-paper proof seems within reach, we expect that a formalization would require some important effort. Indeed, to the best of our knowledge, even the simpler W types lack a formalized normalization proof. (3) Finally, because in OTT consistency is not a consequence of normalization, we will adapt the set-theoretic model of Pujet and Tabareau to handle our universal QIT. Putting (2) and (3) together, we will then conclude canonicity of the type theory, from which we can also deduce canonicity of the encoded QITs.

[2] https://github.com/thiagofelicissimo/universal-QITs

[3] In the case of ITT, this can be fixed using Jasper Hugunin's trick [Hug21], however this is not sufficient in the case of OTT because equality does not satisfy canonicity in this setting.

Based on the above metatheoretic justification, we plan in the future to add QITs to the Rocq implementation of OTT [PLT25, Section 7]. We would of course also like to consider more complex classes of types, such as indexed QITs and QIITs. A promising direction would be to consider the universal type of Kovács *et al.* [KKA19]. However, because of the size of its definition, we are worried that a normalization proof would be harder to achieve. A different approach would be to justify this type using more basic type formers [Kov23, Section 4.6], however we are not sure this can be made to yield an eliminator with the proper definitional equalities.

Finally, an alternative approach to the one we propose here would be to instead consider Cubical Type Theory [CCHM18], which also behaves well with QITs, and whose implementation in Cubical Agda supports HITs in general [VMA19]. Although canonicity and normalization are known to hold in the presence of some specific HITs [Ste22, Hub19], to the best of our knowledge a full metatheoretic treatment of QITs in Cubical Type Theory is also lacking. A possible way to address this could be to use van der Weide and Geuvers's construction of finitary QITs from quotient types and proposition truncation in HoTT [vG19], yet the elimination principles they obtain only satisfy the expected equalities propositionally. Alternatively, one could try to adapt the strategy we propose here, by first constructing QITs from Tm and then studying the metatheory of Cubical Type Theory with Tm. Yet here we chose to focus only on the case of OTT, because we want a type theory for set-level mathematics.

References

[AK16] Thorsten Altenkirch and Ambrus Kaposi. Type theory in type theory using quotient inductive types. *SIGPLAN Not.*, 51(1):18–29, jan 2016.

[AMS07] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In *Proceedings of the 2007 workshop on Programming languages meets program verification*, pages 57–68, 2007.

[CCHM18] 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 *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:34, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Dag13] Pierre-Évariste Dagand. *A cosmology of datatypes : reusability and dependent types*. PhD thesis, University of Strathclyde, Glasgow, UK, 2013.

[FPS22] Marcelo P Fiore, Andrew M Pitts, and SC Steenkamp. Quotients, inductive types, and quotient inductive types. *Logical Methods in Computer Science*, 18, 2022.

[Hub19] Simon Huber. Canonicity for cubical type theory. *J. Autom. Reason.*, 63(2):173–210, August 2019.

[Hug21] Jasper Hugunin. Why Not W? In *26th International Conference on Types for Proofs and Programs (TYPES 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[KK20] András Kovács and Ambrus Kaposi. Large and infinitary quotient inductive-inductive types. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20, page 648–661, New York, NY, USA, 2020. Association for Computing Machinery.

[KKA19] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing quotient inductive-inductive types. *Proc. ACM Program. Lang.*, 3(POPL), jan 2019.

[Kov23] András Kovács. Type-theoretic signatures for algebraic theories and inductive types, 2023.

[LS20] Peter LeFanu Lumsdaine and Michael Shulman. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 169(1):159–208, 2020.

[PLT25] Loïc Pujet, Yann Leray, and Nicolas Tabareau. Observational Equality Meets CIC. *ACM Trans. Program. Lang. Syst.*, February 2025. Just Accepted.

[PT22] Loïc Pujet and Nicolas Tabareau. Observational equality: now for good. *Proceedings of the ACM on Programming Languages*, 6(POPL):1–27, 2022.

[Ste22] Jonathan Sterling. *First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory*. PhD thesis, 2022.

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[vG19] Niels van der Weide and Herman Geuvers. The construction of set-truncated higher inductive types. *Electronic Notes in Theoretical Computer Science*, 347:261–280, 2019. Proceedings of the Thirty-Fifth Conference on the Mathematical Foundations of Programming Semantics.

[VMA19] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. Cubical agda: a dependently typed programming language with univalence and higher inductive types. *Proc. ACM Program. Lang.*, 3(ICFP), July 2019.