An Agda Formalisation of Modalities and Erasure in a Dependently Typed Language

Oskar Eriksson and Andreas Abel

Department of Computer Science and Engineering, Chalmers and Gothenburg University,
`{oskeri,abela}@chalmers.se`

Modal types, like their counterparts in logic, allow modifiers to be attached to types. Given different sets of modifiers and different rules for the treatment of such modifiers by the type system, this allows types to be given a range of different interpretations. These interpretations can range from being quantitive in nature [6, 4], expressing, for instance, linearity or erasure [10, 12], to a variety of other interpretations such as data privacy [2]. Often, modal type systems are designed with specific interpretations in mind and the modifiers and rules are chosen based on this interpretation. An example of this is McBride's modality for erasure and linearity [9]. Others treat modal types in a more general sense and use a more general set of rules to allow different interpretations to be used in the same system.

There are different approaches to generalizing modalities [11, 8]. We have made a formalisation[1] in Agda of one such system for a dependently typed language [7], based on the ideas presented by Abel [1, 2] and Bernardy [2]. A modality is a ring-like structure whose elements are used to annotate terms in order to achieve the desired interpretation. By varying the structure, one can achieve a wide range of different interpretations but the common algebraic properties of the structures are used to define a type system that can check that annotations have been put down correctly.

The Agda formalisation (ca. 26000 lines of code) builds on a formalisation of decidability of type conversion by Abel et al. [3] (ca. 15000 lines of code) with the following novelties:

1. We adapt the syntax and typing judgements with modality annotations.
2. We introduce a new typing judgement for checking the validity of annotations.
3. As a case study, we instantiate it to the erasure modality with extraction to an untyped language.

Most closely related to our work is the Agda formalization by Wood [13] of a simply typed version of our calculus. In comparison to Atkey [5], our calculus also features a weak and a strong Σ-type, but we omit it in the following for lack of space.

The typing judgement for modality annotations relates modality contexts (assigning a modality element to each free variable) with terms. It is defined as follows, making use of the ring-like structure of the modality elements with its operations lifted to act pointwise on contexts γ, δ, a module for this ring.

$$\frac{}{\mathbf{0} \triangleright \mathsf{U}} \qquad \frac{}{\mathbf{0} \triangleright \mathbb{N}} \qquad \frac{\gamma \triangleright F \quad \delta, q \triangleright G}{\gamma + \delta \triangleright \Pi_p^q\, F\, G} \qquad \frac{}{\mathbf{e}_i \triangleright x_i} \qquad \frac{\gamma, p \triangleright t}{\gamma \triangleright \lambda^p\, t}$$

$$\frac{\gamma \triangleright t \quad \delta \triangleright u}{\gamma + p\delta \triangleright t\,{}^p u} \qquad \frac{}{\mathbf{0} \triangleright \mathsf{zero}} \qquad \frac{\gamma \triangleright t}{\gamma \triangleright \mathsf{suc}\, t} \qquad \frac{\gamma \triangleright t}{\delta \triangleright t}\,\delta \leq \gamma$$

[1] Available at https://fhlkfy.github.io/modalities_and_erasure/Logrel-MLTT.html

The erasure case study considers a modality with two elements, 0 and ω which are used to annotate computationally irrelevant and relevant terms respectively. For instance, $\lambda^0\, t$ represents a function whose argument is not used during computation whereas $\lambda^\omega\, t$ represents a function whose argument is. The point of using these erasure annotations is, of course, to achieve a kind of optimisation in which terms that have been marked as erasable can be removed. For this, we use an erasure function $\_^\bullet$ which translates terms into the untyped lambda calculus while removing erasable terms. Most notably $(t\,{}^0 u)^\bullet = t\, ↯$ where the erasable function argument u is replaced with $↯$, representing an undefined value.

If sound, applying this erasure function should not affect the result of fully evaluating a closed term. The proof of soundness makes use of two logical relations. The first, *reducibility* of terms is designed such that a term belongs to the relation iff it reduces to canonical form. For types, it is defined inductively as below. The Π-type case makes use of the reducibility relation for terms of some type F. This relation is defined recursively over $\mathscr{F}$, a proof that F is a reducible type.

$$\langle \mathsf{U} \rangle \frac{}{\Vdash_\ell \mathsf{U}_{\ell'}}\ \ell' < \ell \qquad \langle \mathbb{N} \rangle \frac{A \longrightarrow^* \mathbb{N}}{\Vdash_\ell A}$$

$$\langle \Pi \rangle \frac{A \longrightarrow^* \Pi^q_p\, F\, G \qquad \mathscr{F} : \Vdash_\ell F \qquad \forall a.\ (\Vdash_\ell a : F/\mathscr{F}) \Rightarrow (\Vdash_\ell G[a])}{\Vdash_\ell A}$$

The fundamental lemma for this relation is that $\vdash A$ and $\vdash t : A$ imply $\mathscr{A} : \Vdash_\ell A$ and $\Vdash_\ell t : A/\mathscr{A}$. This relation, in a more general form not restricted to closed terms, was part already in the formalisation by Abel et al. [3].

The second relation relates closed terms in the source language with closed terms in the target language, $t \circledR_\ell v : A/\mathscr{A}$ and is also defined by recursion on $\mathscr{A}$, a proof that A is reducible:

- If $\mathscr{A} = \langle \mathsf{U} \rangle$ then $t \circledR_\ell v : A/\mathscr{A}$ holds iff $\vdash t : \mathsf{U}$.
- If $\mathscr{A} = \langle \mathbb{N} \rangle$ then $t \circledR_\ell v : A/\mathscr{A}$ holds iff either
  - $t \longrightarrow^* \mathsf{zero} : \mathbb{N}$ and $v \longrightarrow^* \mathsf{zero}$, or
  - $t \longrightarrow^* \mathsf{suc}\, t' : \mathbb{N}$ and $v \longrightarrow^* \mathsf{suc}\, v'$ and $t' \circledR_\ell v' : A/\mathscr{A}$.
- If $\mathscr{A} = \langle \Pi \rangle$, then $A \longrightarrow^* \Pi^q_p\, F\, G$ holds and there are derivations $\mathscr{F} : \Vdash_\ell F$ and $\mathscr{G} : \forall a.\ \Vdash_\ell a : F/\mathscr{F} \Rightarrow \Vdash_\ell G[a]$. We then define $t \circledR_\ell v : A/\mathscr{A}$ to hold iff either
  - $p = 0$ and $t\,{}^0 a \circledR_\ell v\, ↯ : G[a]/\mathscr{G}(a)$ for all $\Vdash_\ell a : F/\mathscr{F}$, or
  - $p = \omega$ and $t\,{}^\omega a \circledR_\ell v\, w : G[a]/\mathscr{G}(a)$ for all $\Vdash_\ell a : F/\mathscr{F}$ and $a \circledR_\ell w : F/\mathscr{F}$.

Especially noteworthy is the case for Π-types where, non-erased function terms, t and v are related if they are related when applied to related arguments. For erasable functions, however, the argument on the target language side is replaced with $↯$, mirroring the definition of the extraction function.

The fundamental lemma for this relation is that $\vdash t : A$ and $\triangleright\, t$ imply $t \circledR_\ell t^\bullet : A/\mathscr{A}$. Using this property, the extraction function can be shown to be sound. In particular, all terms of type $\mathbb{N}$ represent the same natural number before and after extraction.

References

[1] Andreas Abel. Resourceful dependent types. In *Presentation at 24th International Conference on Types for Proofs and Programs (TYPES 2018), Braga, Portugal*, 2018. abstract.

[2] Andreas Abel and Jean-Philippe Bernardy. A unified view of modalities in type systems. *Proc. ACM Program. Lang.*, 4(ICFP):90:1–90:28, 2020.

[3] Andreas Abel, Joakim Öhman, and Andrea Vezzosi. Decidability of conversion for type theory in type theory. *Proc. ACM Program. Lang.*, 2(POPL), December 2017.

[4] Robert Atkey. The syntax and semantics of quantitative type theory. In Anuj Dawar and Erich Grädel, editors, *33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 9-12, 2018*, pages 56–65. ACM Press, 2018.

[5] Robert Atkey. Syntax and semantics of quantitative type theory. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 56–65. ACM, 2018.

[6] Aloïs Brunel, Marco Gaboardi, Damiano Mazza, and Steve Zdancewic. A core quantitative coeffect calculus. In Zhong Shao, editor, *Programming Languages and Systems. ESOP 2014*, volume 8410 of *Lecture Notes in Computer Science*, pages 351–370. Springer, 2014.

[7] Oskar Eriksson. An Agda formalization of modalities and erasures in a dependently typed language. Master's thesis, Chalmers University of Technology, 2021.

[8] Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. Multimodal dependent type theory. *Log. Methods Comput. Sci.*, 17(3), 2021.

[9] Conor McBride. I got plenty o'nuttin'. In *A List of Successes That Can Change the World*, pages 207–233. Springer, 2016.

[10] Nathan Mishra-Linger and Tim Sheard. Erasure and polymorphism in pure type systems. In *International Conference on Foundations of Software Science and Computational Structures*, pages 350–364. Springer, 2008.

[11] Klaas Pruiksma and Frank Pfenning. A message-passing interpretation of adjoint logic. *J. Log. Algebraic Methods Program.*, 120:100637, 2021.

[12] Matúš Tejiščák. A dependently typed calculus with pattern matching and erasure inference. *Proc. ACM Program. Lang.*, 4(ICFP), August 2020.

[13] James Wood and Robert Atkey. A linear algebra approach to linear metatheory. 353:195–212, 2020.