Menkar: Towards a Multimode Presheaf Proof Assistant

Andreas Nuyts[1] and Dominique Devriese[2]

[1] imec-DistriNet, KU Leuven, Belgium
[2] Vrije Universiteit Brussel, Belgium

Recently, a number of extensions to dependent type theory have been proposed or refined that are not easily added to pre-existing proof assistants as they severely impact the nature and manipulation of type-theoretic judgements and therefore require principled implementation choices from the start. For this reason, we have started the implementation in Haskell of a novel proof assistant Menkar[1] with the aim of supporting features that are relevant to studying modal and multimode type theory as well as type-systems based on the internalization of properties of a presheaf model. We take a pragmatic approach, looking for a compromise between soundness, user-friendliness, ease of implementation, and flexibility with respect to foreseen and unforeseen modifications of the type system. Unlike projects such as NuPRL and Cedille, we do not intend to demonstrate an alternative view on type theory. Instead, we keep Menkar very Agda-like and merely aim for an implementation of some recently proposed features. Below, we discuss how we (plan to) handle each of the desired features. We conclude by listing possible applications.

Modal type theory In modal type theory, all functions and all variables are annotated with a *modality* describing the behaviour of the dependency. Applications include: modal logic (eponymously) [PD01], variance of functors [Abe06, Abe08, LH11], intensionality vs. extensionality [Pfe01], irrelevance [Pfe01, Miq01, BB08, MS08, Ree03, AS12, AVW17, ND18], shape-irrelevance [AVW17, ND18], parametricity [NVD17], axiomatic cohesion [LS16] and globality [LOPS18]. Pfenning and Abel [Pfe01, Abe06, Abe08] have gradually developed a treatment in terms of an ordered monoid where left multiplication $\mu \circ \sqcup$ has a left adjoint (a Galois connection) $\mu \setminus \sqcup$ which we call *left division*. When type-checking a term $\Gamma \vdash t : T$, μ-modal subterms are type-checked in context $\mu \setminus \Gamma$, which is obtained by applying $\mu \setminus \sqcup$ to the modalities of all variables in Γ. Agda supports modalities for irrelevance and shape-irrelevance based on the ordered monoid approach and Vezzosi has made use of this to extend Agda with support for a global (a.k.a. crisp/flat) modality in `agda-flat` and with support for parametric modalities in `agda-parametric`.[2]

Multimode type theory Sometimes, the set of available modalities μ for functions $(\mu \shortmid x : A) \to B$ depends on the types A and B. For example, in previous work [ND18] we developed a type system which we will refer to as **RelDTT**, in which functions from $\mathbb{N}$ to Bool are either ad hoc or irrelevant, whereas functions from the universe to Bool can also be parametric and functions from $\mathbb{N}$ to the universe can also be shape-irrelevant. In System F, there is always at most one modality applicable, but it is not always the same: functions between types are always ad hoc, while functions from kinds to types are always parametric. Recently, Licata and Shulman have explained these phenomena by moving from an ordered monoid to a 2-category, whose objects are called **modes** and whose morphisms serve as modalities. If there happens to be only a single mode, then we are essentially back in the ordered monoid setting. In case there are or may be multiple modes, we speak of multimode type theory, which is thus a generalization of modal type theory. Here, one assigns a mode to every type, and the modality of a function must match the domain and codomain modes. For System F, we could have 2 modes: `data` for types classifying data and `type` for kinds classifying types. The modes of RelDTT are called $-1, 0, 1, 2, \dots$ but could be read as `proof`, `data`, `type`, `kind`, etc. In Menkar, we aim to support arbitrary multimode type systems. Every declaration and every variable in Menkar is annotated (implicitly or explicitly) with a mode matching its type, and a modality that lifts it to the mode of the enclosing module or context. When type-checking a declaration, all other declarations that are in scope because

[1] http://github.com/anuyts/menkar

[2] See the `flat` and `parametric` branches at https://github.com/agda/agda.

they are part of enclosing modules, are simply added to the context. This ensures that they too are subject to the correct left divisions as we move into modal subterms.

Internal mode and modality polymorphism RelDTT [ND18] has infinite sets of modes and modalities. For this reason, Menkar support for internal mode and modality polymorphism is more than desirable. For most type systems, this will be stretching the semantics, but in general we intend to parametrize Π- and Σ-types as well as the universe with mode and/or modality arguments, on which they depend crisply [LOPS18]. The most striking consequence of modality polymorphism is that it becomes impossible to compute $\mu \backslash \Gamma$ by dividing every individual modality in Γ, because μ may depend on all variables in Γ. Hence, in our implementation, left division is a constructor of the context type. When type-checking a variable, it is checked that the variable's modality is less than the composite of all modalities it has been divided by.

Parametric Tarski-universes Our type systems for parametricity [NVD17, ND18] share the remarkable property that the type T of a term $\Gamma \vdash t : T$ is checked in context $\mathbf{par} \setminus \Gamma$, i.e. divided by parametricity. Fortunately, this does not require any heavyweight language support. Instead, the language provides a (typically non-fibrant) universe `UniHS` (typically modelled by the Hofmann-Streicher universe [HS97] available in all presheaf models) such that $\Gamma \vdash t : T$ requires $\Gamma \vdash T :$ `UniHS`. The fibrant universe is then a different type `Uni` equipped with a parametric function `El` $: (\mathbf{par} \mid$ `Uni`$) \to$ `UniHS`.

Transpension and affineness In parallel work [ND19] we propose a novel *transpension* type as key to internalizing presheaf semantics. This type requires unusual context manipulation, including the disappearance of variables and universal quantification of other variables. We expect that these operations can be captured using a variable-indexed modality system with modalities expressing 'fresh for i', 'for all i' and 'transpend over i'. We expect these same modalities can be used to capture semantically related phenomena related to the substructural affine-like interval variables used by Bernardy, Coquand and Moulin [BCM15, Mou16].

Proving fibrancy internally As mentioned before, Menkar provides a type `UniHS` which is in general non-fibrant. Indeed, our intention is to model Menkar in the default CwF on an arbitrary presheaf category [Hof97] and not in a CwF that restricts to fibrant types. Instead, we want to prove fibrancy internally, avoiding unreadable technical reports such as [Nuy18]. This should often be possible using the transpension type, which is more general than the $\surd$-operator used by Licata et al. [LOPS18], to internalize CCHM fibrancy [CCHM16]. The fibrancy proofs might be made available in a practical way using instance arguments [DP11].

Relatedness-checking RelDTT [ND18] relies on the notion of *judgemental relatedness* [Vez17]. We have implemented the core of a relatedness-checker for Menkar, though we are not sure how to use it. In RelDTT, all types are fibrant (i.e. discrete) meaning that equality coincides with 0-relatedness. Thus, we need not distinguish between judgemental equality and judgemental 0-relatedness and can seamlessly move from conversion-checking to relatedness-checking, ultimately ignoring irrelevant subterms. However, in Menkar, types are non-fibrant unless *proven* otherwise; hence, a definitional mechanism cannot rely on fibrancy.

Interestingly, using instance arguments it may also be possible to implement a relatedness-checker *within* Menkar, which produces propositional evidence. To do so, we would provide an instance for every term constructor of the language. In order to avoid that the instance for e.g. application fires always, we should be able to restrict the instance to neutral function terms.

Applications Our own motivation to start the work on Menkar is to obtain an implementation of RelDTT and of the transpension type, as well as to research a directed version of RelDTT. However, we believe that Menkar's features are also valuable for studying cubical HoTT, as well as guarded type theory including clock-irrelevance [BGC+16] and time warps [Gua18].

Acknowledgements Thanks to Andrea Vezzosi for discussions about type-checking a type-system with extents, and relatedness-checking, and to Andreas Abel for related discussions. Andreas Nuyts holds a Ph.D. Fellowship from the Research Foundation - Flanders (FWO).

References

[Abe06] Andreas Abel. *A Polymorphic Lambda-Calculus with Sized Higher-Order Types.* PhD thesis, Ludwig-Maximilians-Universität München, 2006.

[Abe08] Andreas Abel. Polarised subtyping for sized types. *Mathematical Structures in Computer Science*, 18(5):797–822, 2008. URL: https://doi.org/10.1017/S0960129508006853, doi:10.1017/S0960129508006853.

[AS12] Andreas Abel and Gabriel Scherer. On irrelevance and algorithmic equality in predicative type theory. *Logical Methods in Computer Science*, 8(1):1–36, 2012. TYPES'10 special issue. doi:http://dx.doi.org/10.2168/LMCS-8(1:29)2012.

[AVW17] Andreas Abel, Andrea Vezzosi, and Theo Winterhalter. Normalization by evaluation for sized dependent types. *Proc. ACM Program. Lang.*, 1(ICFP):33:1–33:30, August 2017. URL: http://doi.acm.org/10.1145/3110277, doi:10.1145/3110277.

[BB08] Bruno Barras and Bruno Bernardo. *The Implicit Calculus of Constructions as a Programming Language with Dependent Types*, pages 365–379. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-78499-9_26.

[BCM15] Jean-Philippe Bernardy, Thierry Coquand, and Guilhem Moulin. A presheaf model of parametric type theory. *Electron. Notes in Theor. Comput. Sci.*, 319:67 – 82, 2015. doi:http://dx.doi.org/10.1016/j.entcs.2015.12.006.

[BGC+16] Aleš Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal. Guarded dependent type theory with coinductive types. In *FOSSACS '16*, 2016. doi:10.1007/978-3-662-49630-5_2.

[CCHM16] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. *CoRR*, abs/1611.02108, 2016. URL: http://arxiv.org/abs/1611.02108.

[DP11] Dominique Devriese and Frank Piessens. On the bright side of type classes: Instance arguments in Agda. In *16th International Conference on Functional Programming*, pages 143–155. ACM, 2011.

[Gua18] Adrien Guatto. A generalized modality for recursion. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 482–491, 2018. URL: https://doi.org/10.1145/3209108.3209148, doi:10.1145/3209108.3209148.

[Hof97] Martin Hofmann. *Syntax and semantics of dependent types*, chapter 4, pages 13–54. Springer London, London, 1997. doi:10.1007/978-1-4471-0963-1_2.

[HS97] Martin Hofmann and Thomas Streicher. Lifting grothendieck universes. Unpublished note, 1997.

[LH11] Daniel R. Licata and Robert Harper. 2-dimensional directed type theory. *Electr. Notes Theor. Comput. Sci.*, 276:263–289, 2011. URL: https://doi.org/10.1016/j.entcs.2011.09.026, doi:10.1016/j.entcs.2011.09.026.

[LOPS18] Daniel R. Licata, Ian Orton, Andrew M. Pitts, and Bas Spitters. Internal universes in models of homotopy type theory. In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, pages 22:1–22:17, 2018. URL: https://doi.org/10.4230/LIPIcs.FSCD.2018.22, doi:10.4230/LIPIcs.FSCD.2018.22.

[LS16] Daniel R. Licata and Michael Shulman. *Adjoint Logic with a 2-Category of Modes*, pages 219–235. Springer International Publishing, 2016. doi:10.1007/978-3-319-27683-0_16.

[Miq01] Alexandre Miquel. The implicit calculus of constructions. In *TLCA*, pages 344–359, 2001. URL: https://doi.org/10.1007/3-540-45413-6_27, doi:10.1007/3-540-45413-6_27.

[Mou16] Guilhem Moulin. *Internalizing Parametricity.* PhD thesis, Chalmers University of Technology, Sweden, 2016.

[MS08] Nathan Mishra-Linger and Tim Sheard. *Erasure and Polymorphism in Pure Type Systems*, pages 350–364. 2008. URL: https://doi.org/10.1007/978-3-540-78499-9_25, doi:10.1007/978-3-540-78499-9_25.

[ND18] Andreas Nuyts and Dominique Devriese. Degrees of relatedness: A unified framework for parametricity, irrelevance, ad hoc polymorphism, intersections, unions and algebra in dependent type theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 779–788, 2018. URL: https://doi.org/10.1145/3209108.3209119, doi:10.1145/3209108.3209119.

[ND19] Andreas Nuyts and Dominique Devriese. Dependable atomicity in type theory. In *TYPES*, 2019.

[Nuy18] Andreas Nuyts. Presheaf models of relational modalities in dependent type theory. *CoRR*, abs/1805.08684, 2018. arXiv:1805.08684.

[NVD17] Andreas Nuyts, Andrea Vezzosi, and Dominique Devriese. Parametric quantifiers for dependent type theory. *PACMPL*, 1(ICFP):32:1–32:29, 2017. URL: http://doi.acm.org/10.1145/3110276, doi:10.1145/3110276.

[PD01] Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001. doi:10.1017/S0960129501003322.

[Pfe01] Frank Pfenning. Intensionality, extensionality, and proof irrelevance in modal type theory. In *LICS '01*, pages 221–230, 2001. URL: https://doi.org/10.1109/LICS.2001.932499, doi:10.1109/LICS.2001.932499.

[Ree03] Jason Reed. Extending higher-order unification to support proof irrelevance. In *TPHOLs 2003*, pages 238–252. 2003. URL: https://doi.org/10.1007/10930755_16, doi:10.1007/10930755_16.

[Vez17] Andrea Vezzosi. Judgemental relatedness. Private communication, 2017.