AdapTT: A Type Theory with Functorial Types

Arthur Adjedj[1,2], Thibaut Benjamin[2], Meven Lennon-Bertrand[2], and Kenji Maillard[3]

[1] ENS Paris-Saclay, Saclay, France
[2] University of Cambridge, Cambridge, United Kingdom
[3] Gallinette Project Team, Inria, Nantes, France

Abstract

How can we make systematic and precise the idea that type formers in dependent type theory are *functorial*? To examine this question, we propose a type theory `AdapTT` with adapters, following ideas of McBride and Nordvall Forsberg [MN21] and Coraglia and Emmenegger [CE23].

Coercions, coercions everywhere Diverse type theories feature operators to turn a term of one type into one of another: coercive subtyping [LA08; LLM24], transport for observational equality [AMS07; PT22], casts for gradual typing [Len+22], or adapters [MN21]. These operators share a similar decomposition of coercions along the structure of the types they operate on. For instance, coercing a function f from a function type $A_0 \to B_0$ to $A_1 \to B_1$ amounts to coercing its argument from A_1 to A_0, and its return value from B_0 to B_1.

Semantically, this shared core can be understood as a form of functoriality of type constructors —see for instance Castellan et al. [CCD17, Lemma 4.8], which is essentially the same as the behaviour of coercions at function types outlined above. However, the statement of this lemma is somewhat contorted, as standard CwFs are not up to the task of expressing this functoriality aspect. Indeed, if we want to talk about functoriality, we need *categories* between which functorial type constructors map. Yet, types in CwFs only form a *set*. As already observed by Coraglia and Emmenegger [CE23], comprehension categories [Jac93] lift this discreteness constraint, providing suitable categorical models of subtyping.

Some pieces are however still missing for a full-fledged understanding of coercions in type theory. First, rather than CwF, Coraglia and Emmenegger [CE23] generalise natural models. Second, the key operation of coercion is not directly explicitated. Finally, only specific type constructors (Π and Σ) are considered, those for which the functorial action is admissible thanks to η-rules. Our goal is to fill in these missing pieces, and provide a good setting to talk about functoriality of type formers and coercions as a general concept.

Adapters As we already hinted at, our first step is to modify the standard CwF structure to $\mathrm{CwF}_{<:}$, which incorporate extra data: types in a $\mathrm{CwF}_{<:}$ form a category rather than a mere set —see full definition of $\mathrm{CwF}_{<:}$ in appendix A. We call arrows between types *adapters*, after McBride and Nordvall Forsberg [MN21]. Each type has an identity adapter, and adapters compose. More importantly, adapters act on terms: if $\Gamma \vdash t : A$ and a is an adapter from A to B, we can form a term $\Gamma \vdash t\,\langle a \rangle : B$, the coercion of t along a. Subtyping can be construed as the special case where types are posets, making the witness adapter irrelevant —only the endpoints matter.

Coercions should satisfy a number of properties in their interactions with identity, composition, substitution... Coraglia and Emmenegger's gCwF structure [CE23] succinctly captures this: although they do not derive it, we have shown that the coercion operation can be constructed in any gCwF. In short, when considered fiberwise over a fixed context, terms form a

discrete opfibration over types. This means their setting is a good semantic fit for adapters, and since they show its equivalence with general comprehension categories, this gives us assurance that $\mathrm{CwF}_{<:}$, which reformulate gCwF in the language of CwF, are a reasonable notion.

Parameter contexts We now have an idea of what the codomain of a functorial type constructor should be. But what should its domain be? We propose to use *parameter contexts*, which contain two kinds of variable. The usual, term-level ones, and *type variables*. These come with two pieces of information: a variance, which we will use to differentiate between co- and contravariant functors[1]; and a context of bound variable that a type can depend on. As an example, the following represents the parameters of the dependent function type constructor Π, with one contravariant type, and a covariant one depending on the first

$$\Gamma_\Pi \coloneqq (X : \mathrm{Ty}_-) \triangleright (Y : (x : X).\, \mathrm{Ty}_+)$$

A substitution into that context $\Delta \vdash \sigma : \Gamma_\Pi$ accordingly consists of two types, the second depending on the first. Since we have arrows between types, we also get arrows between these substitutions: if $\Delta \vdash \tau : \Gamma_\Pi$ is another substitution into Γ_Π, a *transformation* $\Delta \vdash \mu : \sigma \Rightarrow \tau$ consists of an adapter between $\tau(X)$ and $\sigma(X)$ (since X is contravariant), and one between $\sigma(Y)$ and $\tau(Y)$, suitably lying over the first. This is exactly the data we need to build an adapter between $\Pi\, x : \sigma(X).\sigma(Y)$ and $\Pi\, x : \tau(X).\tau(Y)$ by pre- and post-composition, as above.

Generalising this example, a type constructor of "arity" Γ amounts to a type $\Gamma \vdash T$, which generates types in all other contexts and the relevant adapters, by the respective actions of substitutions and transformations. In particular, such a type gives rise to a functor from the category of substitutions $\mathrm{Sub}(\Delta, \Gamma)$ to that of types in Δ: given two substitutions $\Delta \vdash \sigma, \tau : \Gamma$ and a transformation $\Delta \vdash \mu : \sigma \Rightarrow \tau$, we get an adapter $T[\![\mu]\!]$ between $T[\sigma]$ and $T[\tau]$.

We succinctly describe this with a 2-category of parameter contexts, substitutions and transformations, the latter built of adapters. The above intuition can then be quickly summarised by saying that Ty is a 2-functor. In the most general view, a type constructor can be understood as a natural transformation from a (Cat-valued) presheaf on the category of contexts to this 2-functor Ty. Our notion of parameter contexts is rich enough to make many such presheaves representable. By the (2-categorical) Yoneda lemma —which gives an isomorphism between the categories $\mathrm{Cat}^{\mathrm{Ctx}^{\mathrm{op}}}[\mathrm{y}\,\Gamma, \mathrm{Ty}]$ and $\mathrm{Ty}(\Gamma)$— parameter contexts let us represent these type constructors and their functorial action more directly, as we did with Γ_Π.

We are exploring the formalisation of all this in Agda as a quotient-inductive-inductive-recursive type, representing the syntax of a putative type theory `AdapTT`. In particular, we can check that type formers such as Π, Σ, or the identity type, adapt to this functorial setting. And they do, following the familiar pattern of structural coercions/casts/transport.

The next direction we hope to explore, which was part of our original motivation, is to provide a setting to generally derive this functorial structure for datatypes. That is, to design a theory of signatures which incorporates —and checks— variance information, and generically derives the adapter equivalent of a map operation [LLM24].

Towards 2-CwF The 2-categorical aspects of the category of contexts suggests a connection to the 2-dimensional type theory 2DTT of Licata and Harper [LH11]. Indeed, internalizing type variables as quantifications over a universe of sets yields a theory very close to 2DTT. However, 2DTT does not have an explicit judgement for the categorical structure on types corresponding to adapters, merely relying on the corresponding notion between terms. A unifying notion of 2-dimensional type theory should combine our $\mathrm{CwF}_{<:}$ with type variables and 2DTT.

[1] For simplification we do not consider equivariance, which should however be relatively easy to add.

A Full definition of CwF with adapters

Definition 1 (Families indexed by categories)**.** $\mathbf{Fam}_{<:}$ is the category whose objects are pairs of a category X and a functor $X \to \mathbf{Set}$. An arrow $(X, F) \to (Y, G)$ is a pair of a functor $H : X \to Y$ and a natural transformation $F \dot{\to} G \circ H$. Abstractly (and ignoring size issues), $\mathbf{Fam}_{<:}$ is the lax slice category $\mathbf{Cat} /\!\!/ \mathbf{Set}$.

Definition 2 (Categories with families and adapters ($\mathrm{CwF}_{<:}$))**.** A *category with families and adapters* ($\mathrm{CwF}_{<:}$) is given by the following data.

- A category Ctx whose objects are called *contexts* and arrows *substitutions*.
- A functor $T : \mathrm{Ctx}^{\mathrm{op}} \to \mathbf{Fam}_{<:}$, that is, given a context $\Gamma : \mathrm{Ctx}$, we have
  - a category $\mathrm{Ty}(\Gamma)$ of *types* in Γ, we write $\mathrm{Ad}(\Gamma, A, B)$ (*adapters*) for the collection of arrows between two types A and B;
  - for any $A : \mathrm{Ty}(\Gamma)$, a set of *terms* $\mathrm{Tm}(\Gamma, A)$;
  - an action of substitution $\cdot\,[\cdot]$ on types, adapters and terms,
  - and an action $\cdot\,\langle\cdot\rangle$ of adapters on terms: if $t : \mathrm{Tm}(\Gamma, A)$ and $a : \mathrm{Ad}(\Gamma, A, B)$, then $t\,\langle a\rangle : \mathrm{Tm}(\Gamma, B)$;
  - suitable equalities for the compatibility of the actions of substitution and adapters with identities, compositions and each other.
- For any context Γ and type $A : \mathrm{Ty}(\Gamma)$, a *context extension* $\Gamma \triangleright A$, equipped with
  - a substitution $\mathrm{wk} : \Gamma \triangleright A \to \Gamma$ (*weakening*)
  - and a term $\mathrm{vz} : \mathrm{Tm}(\Gamma \triangleright A, A\,[\mathrm{wk}])$ (*variable zero*)
  - in a universal way: for any context Δ equipped with $\sigma : \Delta \to \Gamma$ and $t : \mathrm{Tm}(\Delta, A\,[\sigma])$, there exists a substitution $\sigma \triangleright t : \Delta \to \Gamma \triangleright A$ (*substitution extension*) such that $\mathrm{wk} \circ (\sigma \triangleright t) = \sigma$ and $\mathrm{vz}\,[\sigma \triangleright t] = t$, which is unique, *i.e.* for any $\sigma : \Delta \to (\Gamma \triangleright A)$ we have $\sigma = (\mathrm{wk} \circ \sigma) \triangleright \mathrm{vz}\,[\sigma]$.

Theorem 3. *Every $\mathrm{CwF}_{<:}$ in the sense of definition 2 is a gCwF in the sense of Coraglia and Emmenegger [CE23] that is moreover split, and vice-versa any split gCwF is a $\mathrm{CwF}_{<:}$.*

This notion of category of families with adapters can alternatively be presentated via the following Second-Order Generalized Algebraic Theory [Uem21; KX24]:

$$\mathrm{ty} \;:\; \mathbf{Sort} \qquad \mathrm{ad} \;:\; \mathrm{ty} \to \mathrm{ty} \to \mathbf{Sort}$$
$$\mathrm{tm} \;:\; \mathrm{ty} \to \mathbf{RepSort} \qquad \cdot\,\langle\cdot\rangle \;:\; \{A\,B : \mathrm{ty}\} \to \mathrm{ad}\,A\,B \to \mathrm{tm}\,A \to \mathrm{tm}\,B$$
$$\mathrm{id} \;:\; \{A : \mathrm{ty}\} \to \mathrm{ad}\,A\,A \qquad \circ \;:\; \{A\,B\,C : \mathrm{ty}\} \to \mathrm{ad}\,B\,C \to \mathrm{ad}\,A\,B \to \mathrm{ad}\,A\,C$$
$$\mathrm{idl} \;:\; \{A\,B : \mathrm{ty}\}(a : \mathrm{ad}\,A\,B) \to \mathrm{id} \circ a \equiv a \qquad \mathrm{idr} \;:\; \{A\,B : \mathrm{ty}\}(a : \mathrm{ad}\,A\,B) \to a \circ \mathrm{id} \equiv a$$
$$\mathrm{assoc} \;:\; \{A\,B\,C\,D : \mathrm{ty}\}(a : \mathrm{ad}\,A\,B)(b : \mathrm{ad}\,B\,C)(c : \mathrm{ad}\,C\,D) \to c \circ (b \circ a) \equiv (c \circ b) \circ a$$
$$\mathrm{adapt/id} \;:\; \{A : \mathrm{ty}\}(t : \mathrm{tm}\,A) \to t\langle \mathrm{id}\rangle \equiv t$$
$$\mathrm{adapt}/\circ \;:\; \{A\,B\,C : \mathrm{ty}\}(a : \mathrm{ad}\,A\,B)(b : \mathrm{ad}\,B\,C)(t : \mathrm{tm}\,A) \to t\langle b \circ a\rangle \equiv t\langle a\rangle\langle b\rangle$$

This presentation exhibits the adapters as the hom-sets of a category structure on types (internally to presheaves over the category of contexts) and their action on terms as an (internal) presheaf structure over the category of types and adapters.

References

[AMS07] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. "Observational Equality, Now!" In: *Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification*. PLPV '07. Freiburg, Germany: Association for Computing Machinery, 2007, pp. 57–68. ISBN: 9781595936776. DOI: 10.1145/1292597.1292608.

[CCD17] Simon Castellan, Pierre Clairambault, and Peter Dybjer. "Undecidability of equality in the free locally cartesian closed category (extended version)". In: *Logical Methods in Computer Science* 13 (2017).

[CE23] Greta Coraglia and Jacopo Emmenegger. *Categorical models of subtyping*. 2023. DOI: 10.48550/arxiv.2312.14600. arXiv: 2312.14600 [cs.LO].

[Jac93] Bart Jacobs. "Comprehension Categories and the Semantics of Type Dependency". In: *Theoretical Computer Science* 107.2 (1993), pp. 169–207. DOI: 10.1016/0304-3975(93)90169-T.

[KX24] Ambrus Kaposi and Szumi Xie. "Second-Order Generalised Algebraic Theories: Signatures and First-Order Semantics". In: *9th International Conference on Formal Structures for Computation and Deduction, FSCD 2024, July 10-13, 2024, Tallinn, Estonia*. Ed. by Jakob Rehof. Vol. 299. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 10:1–10:24. ISBN: 978-3-95977-323-2. DOI: 10.4230/LIPICS.FSCD.2024.10. URL: https://doi.org/10.4230/LIPIcs.FSCD.2024.10.

[LA08] Zhaohui Luo and Robin Adams. "Structural subtyping for inductive types with functorial equality rules". In: *Mathematical Structures in Computer Science* 18.5 (2008), pp. 931–972. ISSN: 0960-1295. DOI: 10.1017/S0960129508006956. URL: https://www.cambridge.org/core/article/structural-subtyping-for-inductive-types-with-functorial-equality-rules/03A1AD06A0D2F0E6037C2D54DBA68CCC.

[Len+22] Meven Lennon-Bertrand, Kenji Maillard, Nicolas Tabareau, and Éric Tanter. "Gradualizing the Calculus of Inductive Constructions". In: *ACM Transactions on Programming Languages and Systems* 44.2 (Apr. 2022). ISSN: 0164-0925. DOI: 10.1145/3495528.

[LH11] Daniel R. Licata and Robert Harper. "2-Dimensional Directed Type Theory". In: *Twenty-seventh Conference on the Mathematical Foundations of Programming Semantics, MFPS 2011*. Ed. by Michael W. Mislove and Joël Ouaknine. Vol. 276. Electronic Notes in Theoretical Computer Science. Elsevier, 2011, pp. 263–289. DOI: 10.1016/J.ENTCS.2011.09.026.

[LLM24] Théo Laurent, Meven Lennon-Bertrand, and Kenji Maillard. "Definitional Functoriality for Dependent (Sub)Types". In: *33rd European Symposium on Programming, ESOP 2024*. Ed. by Stephanie Weirich. Vol. 14576. Lecture Notes in Computer Science. Springer, 2024, pp. 302–331. DOI: 10.1007/978-3-031-57262-3_13.

[MN21] Conor McBride and Frederik Nordvall Forsberg. "Functorial Adapters". In: *27th International Conference on Types for Proofs and Programs*. 2021.

[PT22] Loïc Pujet and Nicolas Tabareau. "Observational Equality: Now for Good". In: *Proc. ACM Program. Lang.* 6.POPL (2022). DOI: 10.1145/3498693.

[Uem21] Taichi Uemura. "Abstract and Concrete Type Theories". PhD thesis. University of Amsterdam, 2021. URL: https://eprints.illc.uva.nl/id/document/12150.